Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
May-June 2018: 1–13

DOI: 10.1177/1729881418782088
journals.sagepub.com/home/arx
SAGE

Combining a hierarchical task network planner with a constraint satisfaction solver for assembly operations involving routing problems in a multi-robot context

Jorge Muñoz-Morera[1], Francisco Alarcon[1], Ivan Maza[2] and Anibal Ollero[2]

Abstract

This work addresses the combination of a symbolic hierarchical task network planner and a constraint satisfaction solver for the vehicle routing problem in a multi-robot context for structure assembly operations. Each planner has its own problem domain and search space, and the article describes how both planners interact in a loop sharing information in order to improve the cost of the solutions. The vehicle routing problem solver gives an initial assignment of parts to robots, making the distribution based on the distance among parts and robots, trying also to maximize the parallelism of the future assembly operations evaluating during the process the dependencies among the parts assigned to each robot. Then, the hierarchical task network planner computes a scheduling for the given assignment and estimates the cost in terms of time spent on the structure assembly. This cost value is then given back to the vehicle routing problem solver as feedback to compute a better assignment, closing the loop and repeating again the whole process. This interaction scheme has been tested with different constraint satisfaction solvers for the vehicle routing problem. The article presents simulation results in a scenario with a team of aerial robots assembling a structure, comparing the results obtained with different configurations of the vehicle routing problem solver and showing the suitability of using this approach.

Keywords

Planning and scheduling, symbolic planning, vehicle routing problem, multi-robot systems

Date received: 5 February 2018; accepted: 13 May 2018

Topic: AI in Robotics; Human Robot/Machine Interaction
Topic Editor: Nak-Young Chong
Associate Editor: M Bernadine Dias

Introduction and related work

The main goal of the European project that inspired our work was constructing one structure defined in a computer-aided design (CAD) model using multiple drones equipped with robotic arms. This kind of system is of great interest in situations where the assembly of a structure is required, but the characteristics of the terrain or the environment make the assembly operation difficult. This type of situations may arise in civilian missions such as a mountain rescue

[1] Center for Advanced Aerospace Technologies (CATEC), Seville, Spain
[2] Robotics, Vision and Control Group, University of Seville, Spain

Corresponding author:
Jorge Muñoz-Morera, Center for Advanced Aerospace Technologies (CATEC), Calle Wilbur y Orville Wright, 19, 41300 La Rinconada, Sevilla, Spain.
Email: jjmunoz@catec.aero

or fire, but also in military missions like building a bridge. Different scheduling and planning problems are involved in this context: assembly planning, multi-robot task allocation, symbolic planning, and motion planning. There is a huge amount of related work in any of these topics independently, but the problem of combining these different planning levels has been less addressed in the literature until the last 10 years.

Assembly planning can be defined as the process of constructing a specific structure given a set of parts, by computing a plan that is composed of different assembly operations and the order on which they must be executed to build the structure. During the plan generation, different variables are taken into account, such as the geometry of the parts, the geometry of the final structure, the resources to handle the parts and build the structure, the tools available, and so on. It has been proven in Kavraki et al.[1] that all assembly problems have a nondeterministic polynomial time (NP)-complete nature. A complete survey on assembly sequencing was presented by Jimenez,[2] taking into account the geometry of the problem and its combinatorial nature. The most complete and recent taxonomy on the topic can be found in the study by Ghandi and Masehian.[3] In the context of multiple aerial robots, a team that cooperatively constructs a structure is presented by Lindsey et al.[4] In this study, the different parts had a simple geometry and the tools used by the aerial robots were grippers, so picking and placing the parts did not need any kind of manipulation planning. In addition, the parts were placed sequentially, so the benefits of using a team of robots for parallelization were not fully exploited and the assembly tasks were done sequentially. On the other hand, an automated system that uses a team of robots equipped with different tools for the assembly of furnishing is presented by Knepper et al.[5] In that work, a symbolic planner determines the order of operations over the parts for the assembly operation. However, the allocation of tasks to robots is done at the symbolic level by using preconditions and postconditions in an object-oriented symbolic planning specification language, so the task allocation does not use any optimization heuristic.

It can be seen that the assembly planning and sequencing topics have been addressed since many decades ago, but it still remains as an interesting research field and in fact, nowadays the need to have robots with precise assembly capabilities is increasing. One of the trends is to enhance the precision of robots by using new data models and sensors with better precision in their measurements. In the study by Udai and Saha,[6] the authors present a system for the automatic generation of "*depth maps*" for peg-in-hole assembly operations. Depth maps are two-dimensional (2-D) arrays that contain the perpendicular distances of a peg with respect to its mating hole and are commonly used in assembly operations. Given a CAD model the system automatically generates a depth map for the assembly operation. Another way of improving the precision of assembly operations is by measuring the sound produced by mating parts, as presented by Li and Gu.[7] In this study, an acoustic contacting detection is presented to substitute the traditional use of strain gauge load cells. By putting a receiver into a part, when two parts come into contact, part of the sound wave energy is transmitted from the part to the receiver, making it possible to detect the contacting event. Gu et al.[8] perform object localization using a monocular camera. The authors use an eye-in-hand manipulator and a mobile platform for the task. Initially, a speeded up robust features algorithm is applied for feature detection and initial localization. Then, a new probability-based natural right angle detection algorithm is applied, and finally, a 2-D template matching algorithm is used to fine-tune the object localization. Another interesting trend is the use of augmented reality (AR) to improve the accuracy of the assembly process in teleoperation. In the study by Brizzi et al.,[9] the effects of using an AR system are evaluated with the intention of overcoming the differences in perception between telepresence and real presence. The system used an RGB-D camera (Microsoft Kinect 360), a head-mounted display for the operator, and a Baxter robot on the other side. With this setup, the authors demonstrated that by using their AR system, the accuracy and efficiency of the robot in the assembly tasks were improved. Regarding high precision measurements, a case study of the error chain is done by Zhao et al.[10] In that work, a robotized assembly system is studied, and an assembly accuracy analysis model for misalignment errors is proposed. This model provides an assembly accuracy estimation and has been tested in different assembly experiments, giving a reliable worst-case accuracy estimation.

An interesting application of assembly planning can arise from the use of multiple robots. Self-assembly is a process in which a disordered system of preexisting components forms an organized structure or pattern as a consequence of specific local interactions among the components themselves, without external direction. Some studies can be found in this matter. JianJu and YunJian.[11] have addressed the self-assembly process for swarm robots. In that study, the authors propose an enhanced self-assembling morphology-distributed control algorithm for swarm robots, enabling dynamic local navigation according to the distance of seed robot and docking robot. The work also presents time measurements for line-shaped, arrow-shaped, T-shaped and star-shaped morphologies. The case of using a heterogeneous group of robots for self-assembling is studied by Dutta.[12] In that work, each agent can only become the neighbor of a specific set of agents in the target pattern. A constrained bipartite graph-matching algorithm is used to allocate the agents to spots in the target pattern. The allocation is done in a way that adjacent agents are allocated only if they are compatible. The presented algorithm is also compared with other optimal matching algorithms, showing lower run times.

Another application of assembly planning that also requires cooperation among the different robots is

collaborative assembly. Marino et al.[13] address the problem of moving objects with a group of autonomous robots. Instead of using different planning strategies described in the literature for pick and place, object passing, object regrasping, and so on, the authors propose a planning scheme that aims to unify the different solutions. The implemented planner can exploit support surfaces if required in order to reach the goal, for example, putting an object in a region of a table that lets another robot to pick it. The planner relies on the geometric information stored in a database about support surfaces, possible approximations, and feasible grasps, among others. Filipescu et al.[14] present a model for an assembly/dissasembly line that uses two-wheeled robots working in parallel. One of the robots has a robotic manipulator used for part manipulation, while the other robot is used for transporting the parts. During the assembly, if a part does not pass the quality test, the whole assembly is canceled and the disassembly starts to recover the different parts. The work is focused on task planning, modeling and simulation of the assembly line, and use Synchronized Hybrid Petri Nets to control the assembly/disassembly. Regarding safety, some human–robot cells trigger a safety stop when humans leave the safety zone. Collaborative human–robot assembly requires further research to avoid completely stopping robot operations when humans are near the working area of the robots. In Unhelkar et al.[15], a human-aware robotic system is presented. The system is capable of predicting human motion and plan in time to execute safe motions during automotive assembly tasks, without needing to trigger a safety stop. The main interest of this system lies in its ability to adapt the behavior of the robot to the behavior of the human. The robot can operate in a "*Planning with Prediction*" mode, without knowing the task of the human. The robot uses the detected human position and a set of predictions to adapt its motion to the motion of the human. By this way, the robot can pause its task or move to another zone to let the human move freely, without needing to trigger a safety stop.

In our research, it was required to deal with the combination of a symbolic state and a geometric configuration, where a trajectory can modify the symbolic state. Hence, when an action is applied, both the symbolic and geometric states can change. In the literature, there are different approaches for the composition of the symbolic and geometric states: in Şucan and Kavraki, Lozano-Pérez and Kaelbling, Lagriffoul et al., Srivastava et al., and Kaelbling and Lozano-Pérez,[16–20] the symbolic level calls the geometric level, the geometric level calls the symbolic level in Garrett et al., Plaku and Hager, and Choi and Amir.,[21–23] and the compound state is used directly in Cambon et al., Hauser, and Hauser and Latombe.[24–26] The approach adopted in our project belongs to the former group, where the symbolic planner calls the geometric level for motion planning purposes. However, this article is focused on the combination of a symbolic hierarchical task network (HTN) planner and a constraint satisfaction solver for the vehicle routing problem (VRP). The scheme proposed for this connection in a multi-robot context for assembly operations with aerial robots is the main novelty of this work. Although there is a huge amount of work in robotics related to the use of symbolic planners and the multi-robot task allocation problem, to the best of our knowledge, it is the first article that details the interactions between a symbolic HTN planner and a constraint satisfaction solver for the VRP in such a context.

This article is structured as follows. The problem statement is presented in the first section. The assignment of assembly tasks to aerial robots done by the solver is described in the section "VRP solver." After computing the assignment, the optimization stage starts the symbolic HTN planner described in the section "Symbolic HTN planner." This planner gives feedback to the previous solver by scheduling the assembly operations and computing the time cost of executing the different actions. That cost estimation allows the first solver to search for a better assignment. The section "Connecting the VRP solver and the HTN planner" explains in detail how both levels are interleaved. After that section, simulation results using different configurations for the VRP solver are presented, in a scenario of structure assembly with multiple aerial robots. Finally, the article is closed presenting the conclusions and future work.

Problem statement

Given a set of parts that compose a structure, with the parts distributed along a scenario, and given a set of aerial robots also distributed along the scenario, our goal is to assemble the whole structure using the aerial robots, minimizing the total assembly time and maximizing the potential parallelism of using a team of robots in a collaborative way.

The parts have a simple geometry (rectangular parallelepiped), with a handle on top which makes it possible to pick the part, and with a cavity beneath which allows stacking the parts. All the robots are equipped with robotic manipulators that let them pick and place the parts. In addition, the assembly plan is known in advance, so for each part, it is known which other parts must be assembled first. The locations and orientations where the parts must be picked and assembled are already known.

The problem of assembling the structure can be seen as two problems highly coupled one with each other: the problem of assigning the parts to the robots and the problem of scheduling the different task of the robots in time. They are highly coupled because changing the assignment will lead to a different scheduling, and changing the scheduling, that is, changing the time on which the different tasks are planned to be executed, will probably invalidate the assignment.

The assignment problem consists in assigning the different parts to the available robots by following some criterion. One valid criterion could be trying to minimize the

routes of the used aerial robots, because it is reasonable to think that by minimizing the routes of the robots, the total assembly time will be minimized, as the robots will travel the shortest paths possible. However, this is not always true. If we think in the assembly plan, we know that the parts must be assembled following some order, and one part may need other parts to be assembled first. If the criterion of minimizing the path is the only one used, this could result in one "unbalanced" assignment, where some robots were assigned lots of parts because they were near them, and the other robots will have to wait until these unbalanced aerial robots assemble their parts. So, it seems clear that the criterion of minimizing the routes of the robots must be accompanied by another criterion. The other criterion could be to do the assignment in a "balanced" way, by inspecting the dependencies of each part and assigning them in a way that the amount of time that the robots have to wait for the other is also minimized. In any case, this problem may be seen as a variant of the VRP described by Dantzig and Ramser.[27]

On the other hand, once an assignment has been computed, the aerial robots must have a detailed plan to execute the assembly of the parts. The tasks that compose their plans must be correctly scheduled. These tasks include actions such as taking off, traveling to the points of interest, synchronizing with other aerial robots, picking the parts, placing the parts, and so on. The start and end times of each action must be planned and computed.

A formal description of the problem described here was presented in one of the previous works by Muñoz-Morera et al.[28] In the following sections, we describe a new approach that connects a VRP solver and a symbolic HTN planner in a bidirectional way to solve the whole problem.

Symbolic HTN planner

Our planning system needs to know how the structure is built, that is, the parts that compose it, their geometry, and how they are related to each other. Thus, the three-dimensional (3-D) CAD model of the structure is given as the input for the system. From this model, an external planner extracts all the required information to compute an assembly plan using the *assembly-by-disassembly* technique described by Jimenez[2] and Ghandi and Masehian.[3] This technique starts with the assembled structure, and on each iteration disassembles one of the parts that are assembled in the structure. After all the parts of the structure have been disassembled, the order in which the parts were taken from the structure is reversed and the assembly plan is composed of that reversed order. We call this the assembly planner. Although it is out of the scope of this article, a detailed description can be found in Muñoz-Morera et al.[28]

The definition of the computed assembly plan is as follows. Each of the parts that compose the structure appears in the assembly plan as one *assembly task*. The assembly task represents the assembly of one specific part into the structure and contains additional information needed to assemble the part in the form of *preconditions*. The preconditions consist of the parts that must be assembled in the structure before the insertion of that part. We call this set of preconditions the *dependencies* of the assembly task. As the dependencies consist of a set of parts that have to be already assembled, they are also assembly tasks. So, the requirement to execute one assembly task at a given moment is that its dependencies are met, that is, the whole set of assembly tasks that appear as preconditions have been already executed.

This way of defining the assembly plan makes it independent from the number of available aerial robots for the assembly. In a given time, one part can be chosen to be assembled if and only if all its dependencies are met. If there are enough robots, then all the parts that have their dependencies met could be assembled simultaneously and cooperatively, decreasing the assembly time. Of course, this situation would require the correct synchronization of the involved aerial robots.

HTN planning

To deal with all the aspects of the assembly operations and to produce plans for the aerial robots, we have chosen the java simple hierarchical ordered planner2 (JSHOP2) planner described in Nau et al.[29] JSHOP2 is a symbolic planner that uses HTN to solve problems. This type of symbolic planner has been chosen due to his successful application in the robotics area in past years.

The idea behind HTN is to try to solve higher level problems by decomposing them into lower level problems, which in turn can be decomposed into simpler problems, and so on, until there is one available action that can be directly applied to solve any of the simplest problems. Once there is an action that can be applied to every simplest problem, then the high-level problem is solved and the sequence of actions needed to solve it can be constructed. In general, the higher level problem is called the higher level task, which can be decomposed into more simple subtasks until finding actions that can be applied directly to execute the simplest tasks. The way the high-level task can be decomposed into subtasks results in a hierarchy of tasks that can be represented as a tree. This is called a *Task Network*. To help the reader understand this concept, we present the following formalization, which is largely based on Ghallab et al.[30]

Definition 1. Task network: A *task network* is a pair $w = (U, C)$, where U is the set of task nodes and C is a set of precedence constraints.

Definition 2. HTN method: An HTN method is a four-tuple m = <name(m),task(m),(subtasks(m),constr (m))>, where name (m) contains the name and variables of the method, task(m) is the decomposable abstract task that this method

can be applied to, and (subtasks(m), constr(m)) is the task network resulting from decomposing of the task.

Definition 3. Operator: An operator is a four-tuple o = <name(o),task(o),precond(o),effects(o)>, where name (o) contains the name and variables of the operator, task(o) is a nondecomposable abstract task achieved by this operator, precond(o) is a set of predicates that must hold true for the operator to be applicable and effects(o) represents the effects of the action.

Definition 4. HTN planning problem and domain: An HTN planning domain is a pair $\mathcal{D} = < O, M >$ and an HTN planning problem is a three-tuple $\mathcal{P} = < s_0, w, \mathcal{D} >$, where s_0 is the initial state, w is the initial task network, O is a set of operators, and M is a set of HTN methods.

The previous definitions show the hierarchical nature of task networks. A task network is composed of a set of tasks with precedence constraints, that is, the tasks on the set can be partially or totally ordered. Additionally, every task on the network can be decomposed into another task network if it is a decomposable task, or into a single operator if it is a nondecomposable task. Operators are the leaves of the tree representation of the task network decomposition and represent the actions that are applicable and compose the final plan.

To know how to decompose tasks, JSHOP2 needs the definition of a planning domain using a language very similar to a Planning Domain Definition Language. In the domain, high-level task methods must be defined to represent the tasks to decompose. In JSHOP2, the task methods also contain a number of preconditions in the form of logical expressions. In a given state, if the preconditions are met, then the effect of applying the task method over a task is its decomposition on smaller subtasks (methods, operators, or both). A task is said to be *feasible* if the preconditions of its task method and the preconditions of all its lower level subtasks methods and operators are all true, in which case the task is decomposed into several actions.

The planning process works as follows. The tasks are decomposed by using a depth-first algorithm. At each iteration, the task m with the lowest precedence from w is selected. If it is a nondecomposable task, then an applicable operator from O is selected and applied (if possible). If it is a decomposable task, then an applicable method from M is selected and applied (if possible), decomposing the task into the task network $(subtasks(m), constr(m))$ and inserting these new tasks into the queue of tasks. The process is repeated until the initial task network w has been completely decomposed into actions.

Symbolic domain

In JSHOP2, the current state is composed of a set of logical predicates that define the entities and their states that take part in the planning process. Taking into account the problem definition previously described, the following elements should be present in the symbolic domain:

- The assembly parts and their preconditions lists.
- The initial and final poses of the parts in the structure.
- The aerial robots and their home locations.
- The number of parts that remain unassembled.
- An assignment of assembly tasks (parts) to robots.

For our domain, we have designed one main task method which is the task method at the highest level of the hierarchy in the task network. This method is called to solve the problem of assembling one specific structure, given its parts and their dependencies. The method has been defined to be recursive, so on each iteration, it will be decomposed into two new subtasks: the task of assembling one part from the set of unassembled parts and the task of calling himself. As the engine of JSHOP2 plans for the tasks in the order in which they appear, this will guarantee that on each recursive call to himself, the size of the problem will be decreased in one unit because the previous task assembled one part, so if there is a solution for the problem, JSHOP2 will find it and stop.

Once a part is selected, the robots that are assigned to execute that assembly task are checked, and the assembly task is tried to be decomposed into an ordered set of smaller subtasks that compose the low-level plans for the involved robots.

In our defined scenario, with multiple aerial robots used for the simulations, the subtasks on which an assembly task is decomposed can include operations such as take-off, move to specific locations, pick and place parts, or synchronize the aerial robots when the assembly task is cooperative. During the assembly task decomposition, JSHOP2 computes a cost value for each subtask. This cost is an estimation that represents the time needed to execute the subtask, that is, its duration. In addition to the duration, the planner computes the start time of the given subtasks. With an estimation of the start time and duration of each of the subtasks, it is possible to have the plans for each of the aerial robots scheduled in time. A decomposition example of a cooperative assembly task is shown in Figure 1.

VRP solver

The problem presented in this article is a complex problem composed of several subproblems: part assignment, route computation for the aerial vehicles, scheduling of the assembly tasks, parallelization, and so on. Each of these subproblems have been well-studied during the last decades. In this section, we focus our attention on the route computation problem.

It can be easily seen that the routing problem described in the previous sections is a variant of the VRP from Dantzig and Ramser.[27] In the study by Lenstra and Kan,[31] it was

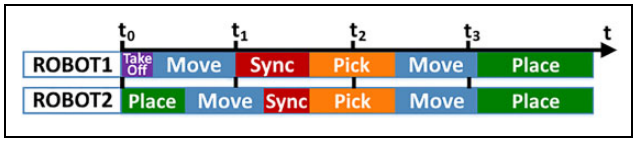

Figure 1. Decomposition of a cooperative assembly task computed by the symbolic planner. An assembly task is the process of assembling a part on a specific location. In this example, the actions needed to assemble the heaviest part of a structure are shown in a time line. As a single robot cannot lift the part due to its weight, the symbolic planner assigned the part to two different robots. Both robots have to pick and place the part cooperatively. The time lines for the two robots are shown. The boxes in the time line represent the subtasks on which the assembly task has been decomposed by the symbolic planner. The first robot, which was initially idle before time t_0, takes off and moves to the part location. After that, it launches a synchronization subtask at time t_1 to wait for the second robot, which is initially finishing another assembly task. When the second robot arrives at the part location, it synchronizes with the first robot. Once they are synchronized, they can execute actions cooperatively, so the subtasks at times t_2, and t_3 onwards are executed by both. The two aerial robots pick, move, and place the part on its assembly location. The start time and duration estimations for the subtasks are computed by the symbolic planner.

demonstrated that the VRP is an NP-hard problem, so the complexity of our routing problem is at least the same as the original VRP, that is, the routing problem presented here is an NP-hard problem. As many symbolic planners use a classical graph, search algorithm to discover states such as depth-first (as is the case of JSHOP2) or breadth-first, among many others, the possibility of using JSHOP2 or any other symbolic planner to try to solve the routing problem was not an option. The combinatorial nature of the VRP leads to huge search trees, and classical graph search algorithms will lead to far-from-optimized solutions with high search times.

To solve this kind of problem, it seems more appropriate the use of meta-heuristics (MH). The idea behind MH is to find reasonably good solutions in reasonably good times. They do not guarantee the finding of the optimal solution. Indeed, almost sure they will not find it, but they can find a good one with less computational effort than other approximations. For this reason, to solve the routing problem we choose OptaPlanner [version v6.2.0].

OptaPlanner[32] is a Java planning engine with a highly configurable planner algorithm that lets the user select different MH algorithms, which can be applied in different phases, to solve some predetermined real-life problems such as the VRP, or some classical problems such as the N-queens problem (described in the study by Bruen et al.[33]). Despite having numerous predetermined problems that can be used and solved as examples, the user can define its own problems, using the Java language. The key point of OptaPlanner is that it has implemented different MH algorithms that can be applied to solve any kind of problem, producing good solutions in fast times. In addition, it has an optimization mechanism based on a score calculation. All solution found is given a score that acts as a measure of how good the solution is. Instead of stopping, after finding a solution, it continues the search trying to find a solution whose score is better than the last solution found. By this way, as the time goes by, the solution is improved. The planning engine stops after the time configured by the user expires. Of course, this has a drawback: The quality of the solution depends on the time we give to OptaPlanner. Usually, the longer the time, the better the solution, but this is not always true, as OptaPlanner may be stuck on a local minimum.

Constraint satisfaction

The OptaPlanner planning engine relies on a score calculation mechanism that lets the solver optimize the solution found as the time goes by. The score of a solution is computed based on different constraints that are imposed by the user when implementing the domain. In OptaPlanner, there are three different types of constraints:

- Hard-constraints: This type of constraints represent rules that should not be broken in any case. They are the most restrictive constraints.
- Medium-constraints: This type of constraints represent rules that should be broken the less possible.
- Soft-constraints: This type of constraints represent rules with the lower priority, and thus can be broken. Still, the broken soft-constraints must be minimized as much as possible.

Based on the broken constraints, OptaPlanner computes a score composed of three negative values that match the number of broken constraints of each type. Any solution with a negative number for the hard-constraints will be immediately discarded as unfeasible.

Solver phases

When searching a solution, the solver can go through different phases defined by the user. Each of these phases is called a Solver Phase in the OptaPlanner terminology, and basically consists on applying one algorithm to make an initial assignment, to optimize a solution, or to try to solve the problem directly.

The first phase is called the construction heuristic (CH) phase, and it consists of selecting and applying one algorithm among the different algorithms available to try to get an initial assignment for the problem. This initial assignment would serve as input for the second phase and can be considered as an initial solution.

The second phase is called the MH phase, and it consists on taking the initial assignment from the CH phase and selecting and applying one algorithm among the different local search algorithms available to try to optimize the assignment and get a better solution. This is an iterative

phase and stops when the time configured by the user expires. After finished, the solution with the best score is given as the best solution.

The third phase is called the exhaustive search and is the only phase that can be executed on its own. It consists of applying the brute force or the branch and bound algorithms to try to solve the problem. Regarding scalability, this is the worst option as the applied algorithms explore the whole search tree, but for problems of small size, it will guarantee that the optimal solution is found.

VRP domain

An assignment problem can be defined in the simplest way as assigning a set of entities to a set of resources available which will manage these entities. In OptaPlanner, the set of entities are called *planning entities* and the set of resources are called *planning variables*. The planning entities are modeled as Java classes and the planning variables as Java variables or lists which must be assigned a value (the planning entities). For our problem, the domain was implemented as follows: the assembly tasks are the planning entities. Each one represents a part that must be assembled by one or more robots, depending on the part weight and the payload capabilities of the robots. In addition to the weight, each part has a dependency list that contains all the parts that have to be assembled in the structure before that part. Each of the robots has a list on which the assigned assembly tasks will be stored. The same assembly task can appear in the list of multiple robots if the weight of the part requires it to be assembled by more than one, but the sum of the payload weights of the given robots must be equal or greater than the part weight.

Connecting the VRP solver and the HTN planner

The VRP solver previously presented has been designed to compute the location's assignment to the robots. In its domain, only the robots and assembly tasks along with their dependencies are considered as entities, but the temporal aspects of the problem are not present. The values of the hard and medium constraints are computed within this domain. The hard-constraint value indicates if the weight of the assigned parts does not exceed the sum of the payloads of the assigned robots. Once an assembly task is allocated, the medium-constraint value indicates how many of its dependencies (parts that should be already assembled) are also allocated to the same robot.

On the other hand, the symbolic domain is designed to compute the assembly tasks decomposition and scheduling of the problem. In this case, the temporal domain is considered and the soft-constraint value is computed as the total assembly time for the whole structure within this symbolic domain.

```
Data: Assembly tasks list A, robots list R, locations list
      L, time limit T
Result: plan P for all the robots
begin
    P ← ∅, P_new ← ∅ ;
    H ← −∞, M ← −∞, S ← −∞;
    H_max ← −∞, M_max ← −∞, S_max ← −∞;
    Loop
        routes ← VRP_Planner.computeRoutes(A, R,
        L) ;
        (H, M) ← VRP_Planner.computeScore(routes)
        ;
        if H ≠ 0 then
            continue ;
        (P_new, S) ← HTN_Planner(routes) ;
        if P_new ≠ ∅ then
            if VRP_Planner.comparePlans(P, P_new)
            then
                H_max ← H, M_max ← M, S_max ← S;
                P ← P_new;
        if timeReached(T) then
            exit loop ;
    return P;
```

Algorithm 1. Pseudo-code showing the connection between the involved planners. The inputs for the system are a precomputed assembly plan composed of assembly tasks, the list of available robots, the list of locations, and the time limit specified for the computations. Initially, the values for the current hard- (*H*), medium- (*M*), and soft- (*S*) constraints are set to the minimum possible negative value. The best values computed during the planning process for these variables are also kept and set to the same minimum possible value. On each iteration, the VRP solver calls the *computeRoutes* function, which computes an assignment of assembly tasks to robots and defines the routes for each one. After that, it calls the *computeScore* function to compute the related hard- and medium-constraints values. If the hard-constraints are zero, it calls the symbolic planner through the *HTN_Planner* function, which in turn computes the decomposition of the assembly tasks (the low-level plan for each aerial robot) and the related soft-constraints value. If the decomposition was possible, then the VRP solver compares the new values for the hard-, medium-, and soft-constraints with the best values that have been found by calling the *comparePlans* function, and updates the best values if the new ones are better, also storing the decomposition computed by the symbolic planner, which is then the best plan found. The process is repeated until the time is exhausted.

The whole score calculation needs both planners to be connected and to communicate in a bidirectional way. The pseudo-code for the whole planning process can be seen in Algorithm 1. First, the VRP planner must solve the assignment problem and compute the related hard- and medium-constraints values. After that, and only if the hard-constraints for the given assignment are zero, it sends the computed assignment to the symbolic planner, which solves the decomposition and scheduling problem and computes the soft-constraint value. Then this value is sent back

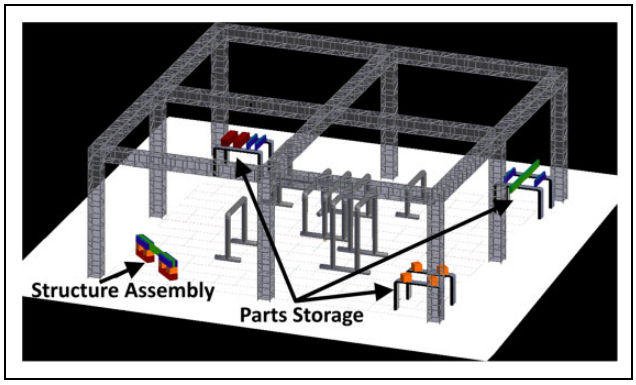

Figure 2. CAD model of the indoor test bed used for the experiments of our European project. The parts are placed over tables in three corners of the scene and are assembled on a designated location.

to the VRP planner, closing the score calculation loop. With the total score of the whole solution, the VRP planner can compare different solutions and optimize the search to try to find new assignments that lead to better scores and improved solutions. Hence, the optimization is done cooperatively between both planners, preserving each of them its own domain and solving a different part of the whole problem.

Simulation results

Different simulations have been carried out in the environment shown in Figure 2, which is the 3-D model of the indoor test bed used for the experiments. Within the test bed, localization of the robots is given by the Vicon motion tracking system (Vicon motion systems Inc., Oxford, United Kingdom) with millimeter precision. In the study done by Merriaux et al.,[34] it has been proven that the Vicon system can achieve errors below 2 mm at common speeds, and below 1 mm for static objects. For the simulations, the Robot Operating System (ROS), the global coordinate system is used. The tests have been done on a machine with an Intel i7 CPU at 2 GHz and 8 GB RAM. The goal of the simulations is to compare different solvers.

A team of four aerial robots equipped with manipulators has to assemble a given structure. Figure 3 shows one of the prototypes developed in the context of the project that is modeled and used in the simulations of this section.

Three structures with a different number of parts (see Figure 4) have been considered and, for each of the structures, 10 data sets have been generated changing randomly the initial locations of the parts.

The solver has been configured to use one CH phase followed by one MH phase. The purpose of the former is to obtain an initial solution for the assignment problem, which will be later optimized by the second solver phase. Three CH algorithms have been applied to our problem: First Fit, First Fit Decreasing, and Cheapest Insertion. A detailed description of each one can be found in Red Hat open source community.[35]

Figure 3. Aerial robot prototype equipped with a robotic arm in the indoor test bed located in the CATEC facilities in Seville (Spain). The model of this prototype has been used in the simulations of the missions.

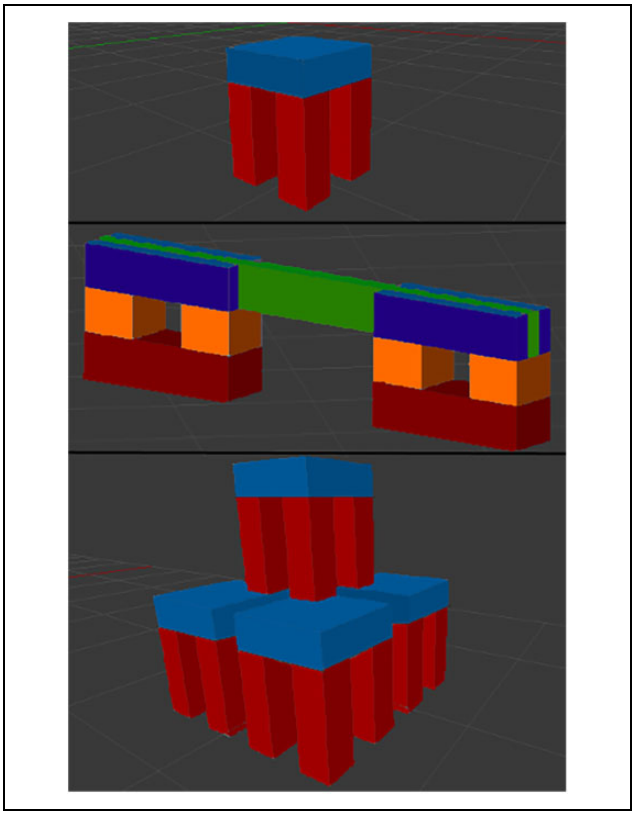

Figure 4. Structures used for the benchmark with sizes of five, eleven, and twenty-five parts (structures 1, 2, and 3 from top to bottom for later reference).

The results shown in Table 1 have been computed with a time limit of 10 min if the search does not finish before. It can be seen that the First Fit algorithm obtained slightly better values, even reducing to zero the medium-constraints. In addition, its computation times are lower than the others.

The second phase is the MH phase, which tries to optimize the initial locations assignment computed by the

Table 1. CH solver phase results for 30 simulations.[a]

CH algorithm	Hard (SD)	Medium (SD)	Soft (SD)	*t* (s)
First fit	−0.33 (0.48)	0 (0)	−625.80 (418.34)	3.12
First fit decreasing	−0.33 (0.48)	−0.33 (0.48)	−630.15 (422.36)	9.56
Cheapest insertion	−0.33 (0.48)	−0.33 (0.48)	−630.15 (422.36)	21.33

CH: construction heuristic.
[a]For each algorithm, the mean and standard deviations for the hard-, medium- and soft-constraint values are presented, as well as the mean computation time. The broken constraints are represented as negative values.

Table 2. MH solver phase results of the soft-constraints generated after 30 simulations with 3 different structures.[a]

CH algorithm	Struct. 1 (SD)	Struct. 2 (SD)	Struct. 3 (SD)	Total (SD)
Hill Climbing	−205.80 (14.19)	−415.80 (37.91)	−1020.0 (54.33)	−547.27 (353.15)
Tabu Search	−205.80 (14.19)	−403.10 (29.64)	−1020.9 (55.95)	−543.27 (354.99)
Simulated Annealing	−203.70 (15.29)	−413.40 (31.34)	−1031.9 (49.81)	−549.67 (359.18)
Late Acceptance	−203.80 (15.33)	−410.80 (31.27)	−991.30 (73.75)	−535.30 (342.06)
Step Counting Hill Climbing	−203.80 (15.33)	−410.80 (31.27)	−991.30 (73.75)	−535.30 (342.06)

CH: construction heuristic; MH: meta-heuristics.
[a]The solver was configured with a time limit of 10 min if the search did not finish before. However, all the algorithms reached the time limit without exhausting the search.

previous CH phase. Five local search algorithms, whose description can also be found on Red Hat open source community,[35] have been compared: Hill Climbing, Tabu Search, Simulated Annealing, Late Acceptance, and Step Counting Hill Climbing. As this phase requires the use of a previous CH phase, the First Fit algorithm was configured as CH. The results are shown in Table 2. All the MH algorithms reduced to zero, the values of the hard- and medium-constraints, so only the mean and standard deviations of the soft-constraint values are shown. The results show that the Late Acceptance and Step Counting Hill Climbing algorithms tie, obtaining better values than the others.

To study the effects over the solutions of changing the number of aerial vehicles used, we have decided to focus our attention on the Late Acceptance algorithm, although the Step Counting Hill Climbing would have also been a good choice. We have used only the third structure, since it is the most complex one with the higher number of parts (25). Five parts need to be transported between two aerial robots due to its high weight, so these five parts are divided into two assembly tasks, resulting in 30 different assembly tasks. Then, the difficulty in solving the problem is greater than using the other structures. To test the scalability of the system when increasing the number of available aerial vehicles, five data sets for the given structure have been created with a number of available aerial vehicles of 10, 20, 30, 40, and 50, respectively. The results of the tests are presented in Figure 5. Figure 5(a) shows the score (assembly time) obtained for each of the data sets, whereas Figure 5(b) shows the number of aerial robots used in the solutions.

As it is shown in Figure 5(a), increasing the number of available aerial robots leads to better plans, as the assembly time tends to decrease. However, when using 30 or more aerial robots the assembly time decreases slower than previous data sets. In fact, the 50 data set gets worse assembly times than the 30 and 40 aerial robots data sets.

The resulting number of aerial robots used for each data set is displayed in Figure 5(b). As the number of available aerial robots is increased, the number of aerial robots used in the solutions also tends to increase. For the data sets that have a number of available aerial robots lower than or equal to the number of assembly tasks (30), the solver uses a number of vehicles that is near the maximum number of vehicles available, as it can be seen in the 10, 20, and 30 aerial robots data sets. For a higher number of available aerial robots, the number of used aerial robots stabilizes near 30, which is the number of assembly tasks for the structure. This fact tells us that the solver will always try to use the maximum number of available aerial robots, even using one aerial robot per assembly task if there are enough aerial robots available. This may seem logical because it is an (extreme) way of maximizing parallelism: in fact, many people will think on this as the optimal solution. However, two associated issues should also be taken into account:

- Having many aerial robots working in our test bed with a size of tens of meters is unrealistic due to the associated air traffic density. As the combined planner will always try to use the maximum number of available resources, the usage of these resources should be limited, for instance, introducing hard-constraints that saturate the maximum number of used vehicles.

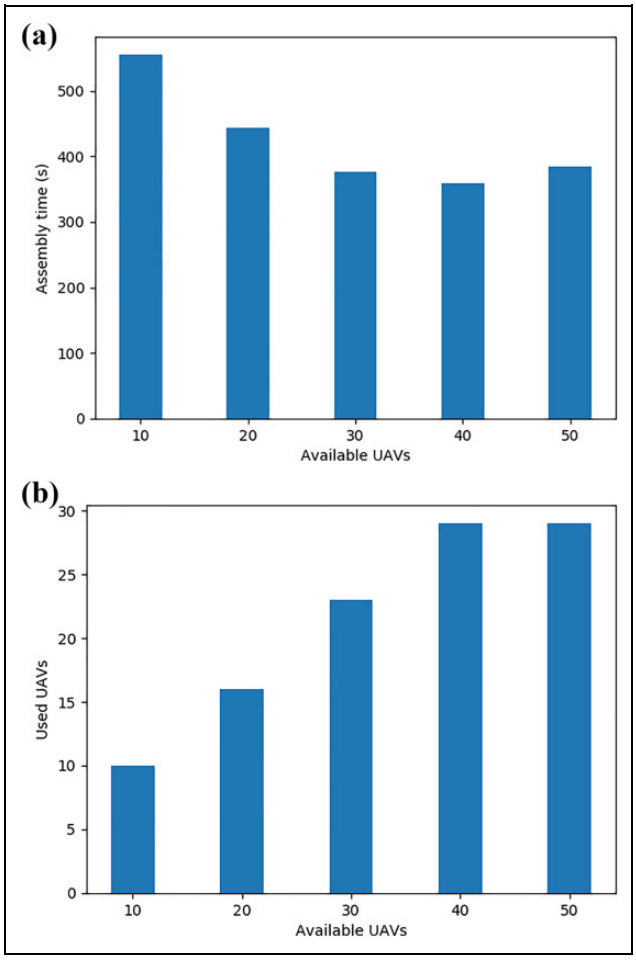

Figure 5. Results of the scalability tests done in a range of available aerial robots between 10 and 50. The third structure from Figure 4 has been used in the tests since it is the most complex one with a higher number of parts. First Fit and Late Acceptance algorithms have been configured in the solver. Five data sets have been created with a number of available aerial vehicles of 10, 20, 30, 40, and 50 respectively. Increasing the number of available aerial robots leads to better plans since the assembly times tend to decrease. As the number of available aerial robots grows, the number of aerial robots used in the solutions tends to increase. (a) Assembly times for tests in a range of available aerial robots between 10 and 50. (b) Number of aerial robots used for tests in a range of available aerial robots between 10 and 50.

- Increasing the number of available aerial robots also increases the problem size indirectly. The VRP planner has a greater number of options to choose when assigning assembly tasks, which can lead to obtaining better plans but can also have the opposite effect since the search tree size is increased, and many more options would be available to be check. We can see this in the results for the 50 available aerial robots data set, whose assembly time is slightly worse than the times for the 30 and 40 available aerial robots data sets. Thus, if the number of available aerial robots is increased, then the solver's computing time should also be increased.

Although it is out of the scope of this article, our planning framework includes the possibility to simulate the execution of the low-level plans computed for each vehicle. An execution layer has been implemented as a C++ graphical user interface application to read the low-level plans. The interface, implemented using the Qt framework [version v5.1], checks for the correct execution and synchronization of the tasks and generates Gantt charts to display the different time lines of the aerial vehicles. The application communicates with a middleware developed using the ROS framework that connects with the Gazebo simulator (Open source robotics foundation Inc.). Figure 6 shows a screenshot of a mission execution on the Gazebo simulator by one aerial robot. A video of the execution can be downloaded from https://grvc.us.es/symballoc#simulationPaper.

It should be mentioned that the motion planning, multi-robot collision avoidance, and the control levels have also been implemented in ROS. In particular, the approach followed at the control level is described by Ruggiero et al.,[36] whereas for multi-robot collision avoidance the techniques implemented are presented by Alejo et al.[37] Regarding motion planning, a comparative study was presented by Ragel et al.[38] that lead to the use of the rapidly-exploring random tree-Connect algorithm in our simulations.

The whole ROS stack developed for the integrated planning framework has been used in the real aerial robots equipped with manipulators. However, the implementation details of the other planners and their interconnection are out of the scope of this article. As a reference, there are some videos available also in https://grvc.us.es/symballoc that show these additional planning capabilities and the execution of plans with several aerial robots both in simulation and in the test bed located in CATEC.

Conclusions and future work

The combination of planners presented in this article performs task assignment and scheduling to improve cooperation and maximize parallelism in domains that mix symbolic reasoning with the VRP. The main contribution is the connection of a VRP solver with a symbolic HTN planner in the field of structure assembly. The approach has been tested successfully in missions involving multiple simulated aerial vehicles.

Our approach has been able to generate aerial vehicles to parts assignment as well as the low-level plans for each of the vehicles in all the tested data sets. The bidirectional communication between the planners has allowed the optimization of the solutions found by the VRP planner, and thus, the feedback of the symbolic layer has been a key aspect to drive the search towards better solutions.

Different MH algorithms have been tested and compared. Although these algorithms have not been able to guarantee optimal solutions, they have computed feasible

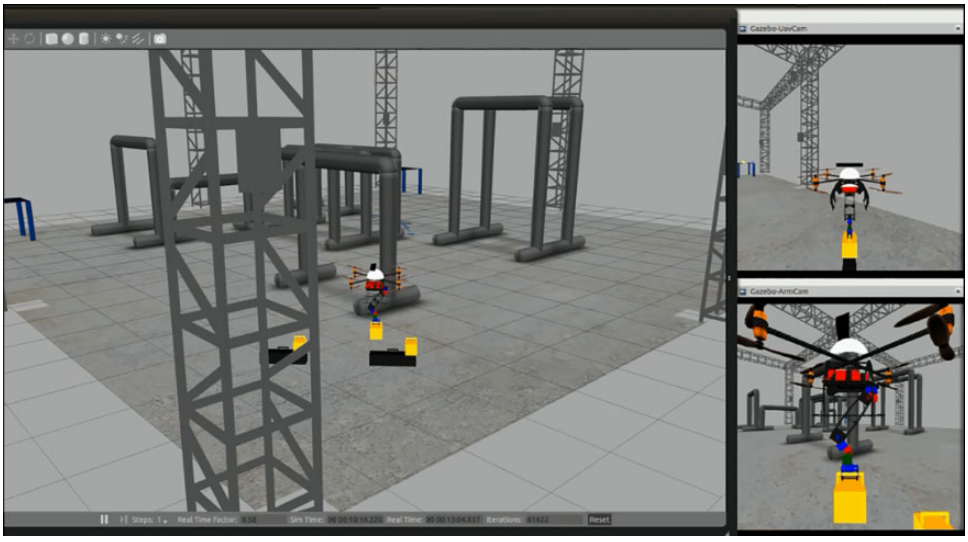

Figure 6. Screenshot of the simulation of an assembly action executed by one aerial vehicle during one of the missions. All the parts have a handle to hold and move them, and can be stacked.

solutions close to the optimal in short times, proving their effectiveness.

One of the main drawbacks of the approach is the impossibility of ensuring the completeness of the system. JSHOP2 is proven to be sound and complete, but that is not the case of OptaPlanner. The OptaPlanner engine guarantees to find a solution, but this solution may not be feasible if it breaks hard-constraints. In addition, the planning time is set beforehand by the user, so when the timer expires, OptaPlanner may not have found a feasible solution. When these situations arise, the planning time has to be increased and the constraints for the problem domain have to be relaxed. This tunning is problem-specific, and it is done by experience and knowledge on the domain, or by trial and error. Although in all the simulations, we have been able to find and generate a feasible solution without increasing the planning time or tunning the constraints, our system cannot ensure that it will always find a feasible solution if one exists.

There are several advantages in using an aerial robot with manipulation capabilities, and practical applications of the research presented in this article can be found in different industry fields. The AEROARMS (https://aeroarms-project.eu) European project aims to develop the first aerial robotic platform equipped with multiple arms and advanced manipulation capabilities, with the intention to be used in inspection and maintenance tasks in industrial plants. This project is based on the results obtained from the aerial robotics cooperative assembly system (ARCAS) (http://www.arcas-project.eu) European project that inspired the work presented in this article, and one of its main objectives is the development of systems which are able to grab and dock with one or more arms and perform dexterous accurate manipulation with another arm. Another practical application can be found in the AEROMAIN (https://grvc.us.es/national-projects/) Spanish project, which is also based on the results of the ARCAS project. AEROMAIN proposes the development of an aerial robotic system with advanced manipulation capabilities to be applied in inspection and maintenance of energy systems. The system aims to be used particularly in the maintenance of wind turbines, where the risk for human operators is very high. Contact inspection and blade repairing of surface damage or impacted areas are considered among other tasks.

In future work, the goal is to enhance the planning domain based on the realistic conditions with the prototypes developed in our project. In addition, modifying the architecture to ensure the completeness of the system is one of our current goals. To achieve this, a new sound and complete symbolic HTN planner with geometric reasoning capabilities is under development, with the intention of replacing the OptaPlanner planning engine.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The first author has been supported by the ARCAS (FP7-ICT-287617) European project and the last two authors received funding from the AEROARMS (H2020-ICT-644271) and MULTIDRONE (H2020-ICT-731667) European projects.

ORCID iD

Ivan Maza http://orcid.org/0000-0003-3502-8372

References

1. Kavraki L, Latombe JC, and Wilson RH. On the complexity of assembly partitioning. *Inf Processing Lett* 1993; 48(5): 229–235. DOI: 10.1016/0020-0190(93)90085-N.
2. Jimenez P. Survey on assembly sequencing: a combinatorial and geometrical perspective. *J Intelligent Manuf* 2011; 1–16. DOI: 10.1007/s10845–011–0578–5.
3. Ghandi S and Masehian E. Review and taxonomies of assembly and disassembly path planning problems and approaches. *Computer Aided Design* 2015; 67–68: 58–86. DOI: 10.1016/j.cad.2015.05.001.
4. Lindsey Q, Mellinger D and Kumar V. Construction of cubic structures with quadrotor teams. In: *Proceedings of robotics: science and systems*, Los Angeles, CA, USA.
5. Knepper RA, Layton T, Romanishin J, et al. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In: *Proceedings—IEEE international conference on robotics and automation*, Karlsruhe, Germany, 6–10 May 2013, pp. 855–862, IEEE Xplore Digital Library.
6. Udai AD and Saha SK. A framework for CAD-based offline depth-map preparation for automated assembly tasks. In: *2016 international conference on robotics and automation for humanitarian applications (RAHA)*, Kollam, India, 18-20 December. 2016, pp. 1–6, IEEE. DOI: 10.1109/ RAHA. 2016.7931906.
7. Li S and Gu H. Acoustic contacting detection in robotic accurate assembly. In: *2017 IEEE international conference on robotics and biomimetics (ROBIO)*, Macau, China, 5–8 December. 2017, pp. 1829–1832, IEEE. DOI: 10.1109/ ROBIO.2017.8324684.
8. Gu J, Wang H, Chen W, et al. Monocular visual object-localization using natural corners for assembly tasks. In: *2016 IEEE international conference on robotics and biomimetics (ROBIO)*, Qingdao, China, 3–7 December. 2016, pp. 1383–1388, IEEE. DOI: 10.1109/ROBIO. 2016.7866520.
9. Brizzi F, Peppoloni L, Graziano A, et al. Effects of augmented reality on the performance of teleoperated industrial assembly tasks in a robotic embodiment. *IEEE Trans Hum Mach Syst* 2018; 48(2): 197–206. DOI: 10.1109/THMS.2017.2782490.
10. Zhao F, Gu H, Li C, et al. Accuracy analysis for robotized assembly system. In: *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Macau, China, 5–8 Dec. 2017, pp. 1850–1855, IEEE. DOI: 10.1109/ROBIO.2017.8324688.
11. JianJu J and YunJian G. A enhanced self-assembly morphology distributed control algorithm of swarm robots. In: *2017 2nd International Conference on Robotics and Automation Engineering (ICRAE)*, Shanghai, China, 29–31 December. 2017, pp. 57–62, IEEE. DOI: 10.1109/ICRAE.2017.8291353.
12. Dutta A.Self-assembly in heterogeneous multi-agent system using constrained matching algorithm. In: *2016 IEEE/WIC/ACM International conference on web intelligence (WI)*, Omaha, NE, USA, 13-16 October. 2016, pp. 351–358, IEEE. DOI: 10.1109/WI.2016.0056.
13. Marino H, Ferrati M, Settimi A, et al. On the problem of moving objects with autonomous robots: A unifying high-level planning approach. *IEEE Robot Autom Lett* 2016; 1(1): 469–476. DOI: 10.1109/LRA.2016.2519149.
14. Filipescu A, Filipescu A, Voda A, et al. Hybrid modeling, balancing and control of a mechatronics line served by two mobile robots. In: *2016 20th International conference on system theory, control and computing (ICSTCC)*, Sinaia, Romania, 13–15 October. 2016, pp. 234–239, IEEE. DOI: 10.1109/ICSTCC.2016.7790671.
15. Unhelkar VV, Lasota PA, Tyroller Q, et al. Human-aware robotic assistant for collaborative assembly: Integrating human motion prediction with planning in time. *IEEE Robot Autom Lett* 2018; PP(99): 1–1. DOI: 10.1109/LRA.2018.2812906.
16. Şucan IA and Kavraki LE. Mobile manipulation: Encoding motion planning options using task motion multigraphs. In: *Proceedings of the IEEE International Conference on Robotics and Automation*. pp. 5492–5498. DOI: 10.1109/ICRA.2011.5980212.
17. Lozano-Pérez T and Kaelbling LP. A constraint-based method for solving sequential manipulation planning problems. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.
18. Lagriffoul F, Dimitrov D, Bidot J, et al. Efficiently combining task and motion planning using geometric constraints. *Int J Rob Res* 2014; 33(14): 1726–1747. DOI: 10.1177/0278364914545811.
19. Srivastava S, Fang E, Riano L, et al. Combined task and motion planning through an extensible planner-independent interface layer. In: *Proceedings of the IEEE international conference on robotics and automation*. pp. 639–646. DOI: 10.1109/ICRA.2014.6906922.
20. Kaelbling LP and Lozano-Pérez T. Integrated task and motion planning in belief space. *The Int J Rob Res* 2013; 32(9–10): 1194–1227. DOI: 10.1177/0278364913484072.
21. Garrett CR, Lozano-Pérez T and Kaelbling LP. Backward-forward search for manipulation planning. In: *Proceedings of the IEEE/RSJ international conference on intelligent robots and systems*. Hamburg, Germany, 28 September–2 October 2015, pp. 6366–6373. IEEE Xplore Digital Library.
22. Plaku E and Hager GD. Sampling-based motion and symbolic action planning with geometric and differential constraints. In: *Proceedings of the IEEE international conference on robotics and automation*, Anchorage, AK, USA, 3–7 May 2010, pp. 5002–5008, IEEE. DOI: 10.1109/ROBOT.2010.5509563.
23. Choi J and Amir E. Combining planning and motion planning. In: *Proceedings of the IEEE international conference on robotics and automation*. pp. 238–244. DOI: 10.1109/ROBOT. 2009.5152872.
24. Cambon S, Alami R and Gravot F. A hybrid approach to intricate motion, manipulation and task planning. *Int J Rob Res* 2009; 28(1): 104–126. DOI: 10.1177/0278364908097884.
25. Hauser K. Task planning with continuous actions and nondeterministic motion planning queries. In: *Proceedings of*

AAAI workshop on bridging the gap between task and motion planning, Atlanta, GA, USA, 11 July 2010.
26. Hauser K and Latombe JC. Integrating task and PRM motion planning: Dealing with many infeasible motion planning queries. In: *Proceedings of AAAI workshop on bridging the gap between task and motion planning*, thessaloniki, Greece, September 2009.
27. Dantzig GB and Ramser JH. The truck dispatching problem. *Manag Sci* 1959; 6(1): 80–91. DOI: 10.1287/mnsc.6.1.80.
28. Muñoz-Morera J, Maza I, Fernandez-Aguera CJ, et al. Assembly planning for the construction of structures with multiple UAS equipped with robotic arms. In: *Unmanned aircraft systems (ICUAS), 2015 international conference on*, Denver, CO, USA, 9–12 June 2015, pp. 1049–1058, IEEE Xplore Digital Library.
29. Nau D, Ilghami O, Kuter U, et al. SHOP2: an HTN planning system. *J Art Intelligence Res* 2003; 20: 379–404.
30. Ghallab M, Nau D and Traverso P (eds.) *Automated Planning*. The Morgan Kaufmann Series in Artificial Intelligence, Burlington: Morgan Kaufmann, 2004. ISBN 978-1-55860856-6. DOI: 10.1016/B978-155860856-6/50000-4.
31. Lenstra JK and Kan AHGR. Complexity of vehicle routing and scheduling problems. *Networks* 1981; 11(2): 221–227.
32. Red Hat open source community. *OptaPlanner*, http://www.optaplanner.org/ (2014, accessed December 12, 2014
33. Bruen A and Dixon R. The n-queens problem. *Discrete Mathematics* 1975; 12(4): 393–395.
34. Merriaux P, Dupuis Y, Boutteau R, et al. A study of Vicon system positioning performance. *Sensors* 2017; 17(7). DOI: 10.3390/s17071591.
35. Red Hat open source community. *OptaPlanner User Guide, version 6.2.0*, http://docs.jboss.org/optaplanner/release/6.2.0.Final/optaplanner-docs/pdf/optaplanner-docs.pdf (accessed 1 May 2015).
36. Ruggiero F, Trujillo MA, Cano R, et al. A multilayer control for multirotor UAVs equipped with a servo robot arm. In: *2015 IEEE international conference on robotics and automation (ICRA)*, Seattle, WA, USA, 26–30 May 2015, pp. 4014–4020, IEEE. DOI: 10.1109/ICRA.2015.7139760.
37. Alejo D, Cobano JA, Heredia G, et al. A reactive method for collision avoidance in industrial environments. *J Intelligent Rob Syst* 2016; 84(1): 745–758. DOI: 10.1007/s10846-016-0359-7.
38. Ragel R, Maza I, Caballero F, et al. Comparison of motion planning techniques for a multi-rotor UAS equipped with a multi-joint manipulator arm. In: *2015 Workshop on research, education and development of unmanned aerial systems (RED-UAS)*, Cancun, Mexico, 23–25 November. 2015, pp. 133–141, IEEE. DOI: 10.1109/RED-UAS.2015.7441000.