Planning engine driven by task allocation and symbolic planners for multi-UAV assembly missions optimization

Jorge Munoz-Morera, Ivan Maza,
Fernando Caballero and Anibal Ollero

Abstract— **The work presented in this paper is part of the autonomous planning architecture of a team of aerial robots equipped with on-board robotic arms. The mission of the team is the construction of structures in places where the access is difficult by conventional means, which is the scenario considered in the framework of the ARCAS European Project. This paper presents a planning engine for this context. From the 3D CAD model of the structure an assembly planner computes the required assembly operations, which are the inputs for the system. These operations are assigned to the available aerial robots by a task allocation planner, which computes an assignment and optimizes it by communicating with a symbolic planner that estimates the cost of the sequence of actions needed in the mission execution for the given assignment. This paper centers on plan optimization and includes benchmarking and simulation results that show the feasibility of the approach and the properties of the solutions.**

Index Terms— **Assembly Planning; Symbolic Planning; Task Planning; Task Allocation; Multiple UAS**

I. Introduction and Related Work

The work described in this paper is part of the ARCAS European Project [1] funded by the European Commission. One of the goals of this project is to build a structure by using a team of aerial robots equipped with on-board manipulators. The practical interest of this system can be found in situations where it is required to build a structure in places with difficult access by conventional means.

Although Unmanned Aerial Systems (UAS) have been designed and developed for performing military missions, currently the trend is to extend their applicability to civil mission such as fire-fighting, critical infrastructure protection or remote surveillance, among many others. The multiple variety of platforms, control systems and ground-based equipments [2], [3], and the heterogeneity of the communication devices have made difficult the interoperability among different systems.

Several works [4], [5] have addressed cooperation in teams of aerial robots for multi-purpose missions. However, in those papers the dexterous manipulation with aerial robots was not present. The use of aerial robots allows to perform assembly operations in any point of the 3D space, which can represent a relevant advantage compared to ground robots in areas of difficult access. Assembly planning is the process of creating a detailed assembly plan to craft a whole product from separate parts by taking into account the final product geometry, available resources to manufacture that product, fixture design, feeder and tool descriptions, etc. Efficient assembly plans can reduce time and costs significantly. The assembly planning problem has been shown to be an NP-complete problem [6] and covers three main assembly subproblems: sequence planning, line balancing, and path planning. Reference [7] presents a survey on assembly sequencing from a combinatorial and geometrical perspective.

Most existing classical planners can be classified into two categories [8]: planners which use domain-dependent knowledge and planners which use domain-independent knowledge. The former can exploit their specific knowledge to guide the planning process and solve larger problems faster than other planners, with the disadvantage of needing a person who gives the knowledge on how to solve the problems. Such knowledge

This work has been partially supported by the ARCAS Project (FP7-ICT-287617) funded by the EU FP7 and the RANCOM Project (P11-TIC-7066)

Robotics, Vision and Control Group, University of Seville, Avd. de los Descubrimientos s/n, 41092, Sevilla, Spain. Email: `jorgemunoz@us.es`, `imaza@us.es`, `fcaballero@us.es`, `aollero@us.es`

can be designed using temporal logic (TLPlan [9] and TALplanner [10]) or task decomposition (SHOP2 [11], SIPE-2 [12], O-PLAN [13]). On the other hand, a planner that uses domain-independent knowledge (SGPlan [14], FastDownward [15], LPG [16]) does not need specific knowledge so the domain formalization is simpler, with the disadvantage that the performance of the planner may be lower than that of domain-dependent planners. The integration of both types of planners, which let use their advantages and avoid their disadvantages, is a matter of study, and some works in that direction can be read in [17] and [18].

Below the task planning is situated the motion and manipulation planning, which should take into account the geometry, kinematics and dynamics of the problem. For motion planning there are consolidated techniques which produce very good results, such these based on Rapidly-exploring Random Trees [19]. Combined task and motion planning have been studied in different works [20], [21], [22], but this work only addresses up to task planning. A system on which teams of quadrotors assemble a 2.5-D structure from simple parts can be found in [23]. In that work the robots were equipped with grippers and the structure was supposed to be assembled sequentially, so no manipulation planning was done after picking the parts and the assembly operations were not parallelizable.

This paper presents a planning engine to solve a general structure assembly problem with a team of UAS. From the 3D CAD model of the structure the assembly planner presented in Sect. II computes the required assembly operations, which are the inputs for the system. These operations are assigned to the available aerial robots by the task allocation planner described in Sect. III. This planner optimizes the computed assignment by communicating with the symbolic planner explained in Sect. IV, which estimates the cost of the sequence of actions needed in the mission execution for the given assignment and gives feedback to the task allocation planner in the search of a better assignment. This paper centers on plan optimization and includes benchmarking and simulation results in Sect. V that show the feasibility of the approach and the properties of the solutions. Section VI closes the paper with the conclusions and future work.

II. Assembly Planner

From the 3D CAD model of the structure, a framework for assembly planning has been developed taking into account the physical stability of the partial substructures in the presence of the external force of the gravity. The assembly planner is composed by a classical planner, a simulation engine based on the Bullet Physics library [24] and a visualization component that is provided as an optional tool.

The purpose of the assembly planner is to obtain an assembly plan with all the assembly operations needed to build the structure from an initial state where no part is assembled to a final state where all parts are assembled, given by the 3D CAD model itself. The assembly plan is computed by using the *assembly-by-disassembly* technique, consisting on finding a plan to disassemble the whole structure and reversing the order of its operations to get the final assembly plan.

The output of the assembly planner is an XML file containing the assembly operations. Each operation represents the assembly of one specific part of the structure and contains the preconditions that must be met prior to its execution, namely the assembly operations that must be done before the assembly of that specific part. In the output plan file the assembly operations are partially ordered, meaning that the assembly could be correctly done by doing the operations in appereance order, but operations that appear later in the file may be executed in parallel or even before some of the previous.

III. Tasks Assignment in a multi-UAS context

After the generation of a plan composed of assembly tasks, the next step is the generation of an initial assignment of these tasks to the available UAS. This section contains the problem definition and the details of the planner that solves it.

Let us consider a mission $\mathcal{M}$ consisting on the assembly of a structure composed by several parts initially located around the environment. The parts have to be assembled on specific locations by a team of n UAS starting the mission from their home locations. Then the mission is composed by a set of assembly tasks $\mathcal{T}$. Each of the parts has a weight and a dependency list consisting on the tasks that must be done prior to its assembly. Let us define $\mathcal{L}$ as the set of stock parts locations, $\mathcal{L}'$ as the set of locations where the parts have to be assembled and $\mathcal{H}$ as the home locations of the UAS. The objective is to assemble all the parts on their locations minimizing the travel flight times of the vehicles and exploiting the potential parallelism that can be achieved using multiple UAS.

The implicit combinatorial problem can be expressed by the edges of a graph $G(V, E)$ considering the following notation:

- $\mathcal{T} = \{t_1, t_2, ..., t_m\}$ is a set of m assembly tasks.
- n is the number of UAS.
- $\mathcal{L} = \{l_1, l_2, ..., l_m\}$ is the set of stock parts locations and $\mathcal{L}' = \{l'_1, l'_2, ..., l'_m\}$ is the set of final assembly locations.
- $\mathcal{H} = \{h_1, h_2, ..., h_n\}$ is the set of UAS home locations.
- $V = \{\mathcal{L} \cup \mathcal{L}' \cup \mathcal{H}\} = \{v_1, v_2, ..., v_{2m+n}\}$ is the set of vertices of the G graph.
- $E = \{(v_i, v_j) | v_i, v_j \in V; i < j\}$ is the edge set.
- $R_k = \{r_1, r_2, ..., r_s\} \subseteq V$ is the route for the k-th UAS, composed by a subset of s vertices from V.
- Cost c_{r_i,r_j} is a non-negative travel time between vertex r_i and r_j, where $c_{r_i,r_j} = c_{r_j,r_i}$.
- $w = \{w_1, w_2, ..., w_m\}$ is a vector containing the weights of the parts.
- $p = \{p_1, p_2, ..., p_n\}$ is a vector with the maximum payloads of the UAS.

The problem consists in determining a set $\mathcal{R}$ of UAS routes with minimal cost, each starting at the home locations of the vehicles, such that every vertex in $\mathcal{L}$ is visited at least by one vehicle and followed by its subsequent vertex in $\mathcal{L}'$, without exceeding the payload of each vehicle. The possibility of visiting the same location with multiple UAS is given by the fact that some parts should be carried cooperatively by more than one UAS if they are too heavy. The problem described is similar to the well-known Vehicle Routing Problem (VRP) [26]. For the k-th UAS, the cost of a route is given by

$$C(R_k) = \sum_{i=1}^{s-1} c_{r_i,r_{i+1}} \,, \tag{1}$$

where $r_1 \in \mathcal{H}$, $r_i \in V$ and $r_j \in \mathcal{L} \implies r_{j+1} \in \mathcal{L}'$. Considering that up to two UAS can cooperatively transport a single heavy part, this route R_k is feasible if $(p_k \geq w_j) \vee (\exists R_z | r_j \in R_z \wedge (p_k + p_z) \geq w_j)$, i.e. the weight of each part does not exceed the maximum payload of the UAS transporting it or there is another available UAS so that both can transport it cooperatively.

On the other hand, a score has been defined for each solution in order to balance the workload of the different UAS. This score consists on a numerical value that is increased as a penalty whenever the task allocation for each UAS diverges from the allocation where all the UAS have the same number of tasks allocated.

The goal of the planner is to minimize the total travel time $\sum_{i=1}^{n} C(R_i)$ of the feasible routes executing all the assembly tasks of the mission, and minimizing also the score mentioned above.

The planner chosen to solve the assignment problem presented in this section was OptaPlanner [27], an open source, multi-platform planning engine written in Java and released under Apache Software License. OptaPlanner is aimed to solve planning problems with resource usage optimization. It is capable of generating near-optimal plans by applying optimization heuristics and meta-heuristics combined with score calculation. One of his main advantages is that the solver's algorithm is highly configurable. In OptaPlanner it is possible to use different heuristics and meta-heuristics algorithms applied in sequence, so that the user can select the most suitable for the problem in question. The optimization is done in base of a score calculation that is computed after all the planning entities have been assigned. This score determines the suitability of the last computed solution: if after searching for a new solution, the new score is higher than the score calculated in the previous solution, then the last solution is discarded and the process continues trying to generate a solution with a lower score.

It should be mentioned that one task may be required to be executed by more than a single UAS. In order to do so, certain tasks are automatically divided and the workload is shared. This is the case of heavy parts that could not be carried by a single UAS. The associated assembly task is divided in several assembly tasks with less weight that can be executed by several cooperative UAS.

IV. Assembly Tasks Sequencing

Up to this point, the assembly plan contains the sequence of assembly tasks that are needed to consistently build the structure. Some of the tasks in the computed plan could be executed in parallel to reduce the whole mission assembly time. This information is implicitly encoded in the plan as task preconditions. Each assembly task has a list associated containing all the preconditions that must be met in order to be able to perform the operation. At any given instant, all the assembly tasks that have their preconditions met, could be executed in parallel.

In addition to the minimization of the assembly time, a low level plan must be computed for each UAS. Feasible paths must be computed for the navigation of the robots, both for the part picking and the part assembling. Also, those assembly tasks that must be executed cooperatively between robots require the synchronization among the involved UAS.

In this paper, the JSHOP2 symbolic planner [11] has been applied to order the assembly tasks as well as to compute a low level plan for each UAS. JSHOP2 is a domain-independent planning system based on Hierarchical Task Network (HTN) that decomposes the tasks into subtasks, then the subtasks into smaller subtasks and so on, until obtaining low-level tasks that can not be further decomposed and thus can be directly performed. These low-level tasks are called primitive tasks.

The inputs to JSHOP2 are a planning domain and a planning problem. The planning domains are composed of methods, operators and axioms, while the planning problems are composed of logical atoms and task lists. The methods, operators and axioms that compose a planning domain all involve logical expressions that combine logical atoms. The logical atoms conform the initial state of the problem, and the task lists represent the high-level actions that must be performed (the goals) by decomposing them into lower-level actions called primitives. The decomposition is done by the use of methods, which decompose the high level tasks into smaller ones until obtaining the primitives, which are represented by the operators.

The format of the problem and domain files in JSHOP2 is very similar to a Planning Domain Definition Language (PDDL) description. But JSHOP2 does not support durative and concurrent actions that are present in PDDL Level 3. However the JSHOP2 format is expressive enough to represent this type of actions because in the preconditions of the methods and operators of the domain some calculations can be done. There are different techniques to accomplish that but the one used in this work is the explained in [11], which is called Multi-Timeline Preprocessing (MTP). By using this technique it is possible to mark each operator with a start and duration stamp. This requires to insert in the JSHOP2 problem definition the read/write times of the logical atoms whose values could be modified after the execution of a method. After a plan is found by JSHOP2, all the operators that appear in the plan are correctly sorted in time due to its temporal marks.

A. Domain Description

In this subsection the designed JSHOP2 domain is presented. This domain aims to solve the problem of selecting and sequencing the assembly tasks from a previously computed assembly plan on which the tasks are partially ordered, to minimize the assembly time. The problem requires the search of an optimal scheduling for the assembly tasks selecting them from the set of tasks given in the assembly plan, taking into account the preconditions of the tasks that are selected.

The problem is centered in the task selection and its optimal sorting to minimize the assembly time in an environment where different factors affect the final time of the structure assembly. For this reason, an estimation of the duration of all actions needed to carry out the assembly has been done. These actions include the take-off for the UAS, the displacements between locations, the part picking, the part assembly and the synchronization between different UAS to do cooperative tasks, among others.

To model the problem, the following considerations were taken into account:

- Each UAS has a battery that limits its operation time.
- Each part has a dependency list, which consists on the list of parts that have to be assembled before the part could be assembled.
- Each part is assigned to one or two UAS, depending on its weight. Heavier parts are assigned to two UAS while lighter parts are assigned to only one. The assignment of parts to UAS is done in previous phases of the planning architecture, so this information is previously known.
- A UAS may synchronize itself before picking a part, waiting for the part to be correctly located and ready to be picked.
- Two UAS that carry one part cooperatively must synchronize between them to ensure that they are at the part location in the same time instant.

For our problem one high-level task was defined to try to find an optimal scheduling for all the previously computed assembly tasks, which receives as argument the type of problem to solve, *assembly* in the present study. In the domain designed this argument can take other values for other problems, to solve them in a different form. Listing 1 shows a simplified method that matches that high-level task. The method was defined to be recursive, trying to assemble one part on each call and containing three pairs preconditions-subtasks.

The first pair corresponds to the case of having unassembled parts without dependencies. Parts that do not depends on others must be assembled first to avoid dependent parts to wait the less for them to be assembled and thus minimize the assembly time. From all the parts that make these preconditions true, the planner chooses one. In the subtasks, the part is tried to be assembled using the assigned UAS and calling a lower-level method. After that, a recursive call for the high-level method is done to try to assemble another part.

The second pair corresponds to the case of having only unassembled parts with dependencies. From all the unassembled parts with dependencies, only these that have all the parts from its precondition list assembled can be chosen to be assembled. From the set of parts that make these preconditions true, the planner chooses one. In the subtasks, the part is tried to be assembled using the assigned UAS and calling a lower-level method. After that, a recursive call for the high-level method is done to try to assemble another part.

Finally, the third pair corresponds to the case of having all the parts assembled. In that case, the only subtask is a primitive task, so an operator is directly called. That operator has its delete list and add list void, it is only used to exit the recursive method. The implementation of the method is thus omitted.

Listing 1: JSHOP2 Simplified high-level task method for the assembly domain. The case of parts that need to be carried by two UAS is not shown.

```
(:method
 (mission ?missionType)
 ;precondition(
  (call = ?missionType assemble)
  (remaining_tasks ?remaining)
  (call > ?remaining 0)
  (object ?partName) (not(assembled ?partName))
  (depends ?partName ?dependency_list)
  (call = ?dependency_list ())
  (quadrotor ?uav) (assigned ?partName ?uav)
  (location ?loc)
  (assembly_location ?partName ?loc))
 ;subtasks(
  (synchro_mission assemble ?uav ?loc ?partName
      )(mission ?missionType))
 ;precondition(
  (call = ?missionType assemble)
  (remaining_tasks ?remaining)
  (call > ?remaining 0)
  (object ?partName) (not(assembled ?partName))
  (depends ?partName ?dependency_list)
  (forall (?z)
   ((object ?z)(call Member ?z ?dependency_list
       ))
   (assembled ?z))
  (quadrotor ?uav) (assigned ?partName ?uav)
  (location ?loc)
  (assembly_location ?partName ?loc))
 ;subtasks(
  (synchro_mission assemble ?uav ?loc ?partName
      ) (mission ?missionType))
 ;precondition(
  (call = ?missionType assemble)
  (remaining_tasks 0))
 ;subtasks(
  (!assembly_plan_finished) ))
```

Listing 1 shows a simplified high-level method for the task goal that compose the assembly problem. For the sake of simplicity in the explanation, the method shown take into account only the case of parts that do not need to be transported cooperatively among different UAS. However the case of parts carried by two UAS cooperatively is included in the domain designed. The low-level method for that case can be seen in Listing 2. The method shares almost all the preconditions with the upper-level method, and is composed of several subtasks. It moves the two assigned UAS to the part location and introduces a synchro_wait among the UAS to synchronize them to be in the part location at the same time. Also a synchro_wait for each of them is introduced to wait until the part is ready to be picked. The rest of the subtasks such as the pick, the move to the assembly location with the transported part and the assemble are cooperative.

Listing 2: JSHOP2 Two-UAS version of the low-level method for the assembly domain. The method tries to assemble one part given its two assigned UAS.

```
(:method
 (synchro_mission ?missionType ?uav1 ?uav2 ?
     targetLocation ?partName)
 ;precondition(
  (quadrotor ?uav1) (quadrotor ?uav2)
  (object ?partName)
  (assigned ?partName ?uav1)
  (assigned ?partName ?uav2)
  (location ?loc)
  (at ?partName ?loc) (location ?targetLocation
      )
  (not (transporting ?uav1))
  (not (transporting ?uav2))
  (not (transported ?partName))
  (not (assembled ?partName))
  (depends ?partName ?dependency_list)
  (forall (?z)
   ((object ?z)(call Member ?z ?dependency_list
       ))
   (assembled ?z)))
 ;subtasks(
  (move ?uav1 ?loc) (move ?uav2 ?loc)
  (synchro_wait_v ?uav1 ?uav2 )
  (synchro_wait ?uav1 ?partName)
  (synchro_wait ?uav2 ?partName)
  (pick ?uav1 ?uav2 ?partName)
  (move ?uav1 ?uav2 ?targetLocation)
  (execute_mission ?missionType ?uav1 ?uav2 ?
      targetLocation ?partName)))
```

The designed domain also takes into account some precedence rules to minimize the amount of time for building the whole structure. That occurs in the case of having multiple parts with dependencies that can be actually chosen to be assembled. For that set, the parts that are known to be dependencies for other parts that can not be assembled yet are chosen first. This selection is done to try to 'unlock' as many parts as possible from the set of parts that can not still be assembled.

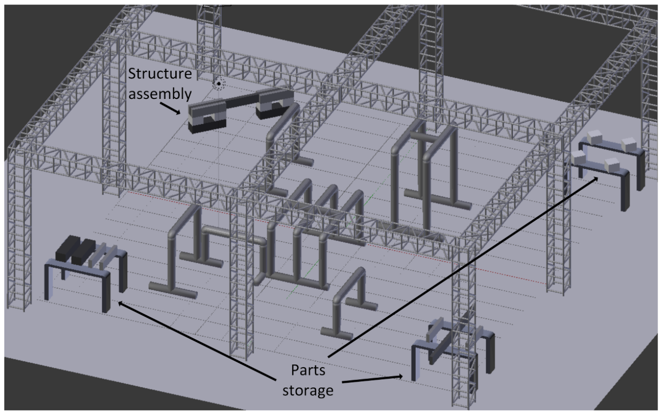

Fig. 1: Environment modelled for the mission. The parts are initially stored in three different stock areas and finally assembled in the top left area of this figure.

V. Structure Assembly Example

In this section a representative mission will be used to illustrate the operation of the planning approach in a multi-UAS context. In the mission presented, a fleet composed of four unmanned aerial vehicles is available, where all of the vehicles are simulated. The environment contains twenty-six locations of interest (see Fig. 1) where eleven of them are the initial locations for the parts, another eleven are the assembly locations and four are home locations for the UAS. The structure to be assembled can be viewed in Fig. 2. The mission is composed by eleven assembly tasks corresponding to the eleven parts that conform the structure.

The entry point to the system is the 3D CAD environment model that contains the initial state (stock parts and home locations of the UAS) and the 3D CAD model of the structure to be built. The assembly planner reads the 3D CAD model of the structure and generates a valid assembly plan. Each of the assembly tasks contain two nodes: one with the *effect* of the operation and one with the *preconditions* for the operation. The effect of the operation is the part that will be assembled after the task execution, whereas the preconditions for the operation are the parts that must be assembled prior to the execution of the task. Part of the assembly plan generated by the assembly planner

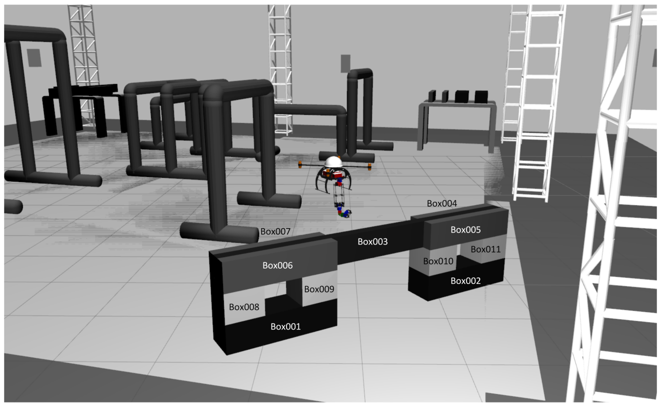

Fig. 2: Structure to assemble for the mission. The structure is composed of eleven parts, enumerated from Box001 to Box011. All the parts are supposed to have a flat handle from which the robotic arms of the UAS can pick them.

is shown in Listing 3.

Listing 3: First tasks of the assembly plan generated by the assembly planner. The tasks are partially ordered meaning that a single UAS could do the assembly correctly by executing the tasks in that order. However tasks that appear later in the file may be executed before some of the previous as is the case of the part Box001 which constitutes a base part and thus do not depend on any other.

```
<plans>
 <assemblyPlan>
  <assemblyOperation action="base">
   <effect>
    <at part="Box002"/>
   </effect>
  </assemblyOperation>
  <assemblyOperation action="connection">
   <precondition>
    <at part="Box002"/>
   </precondition>
   <effect>
    <at part="Box011"/>
   </effect>
  </assemblyOperation>
  <assemblyOperation action="connection">
   <precondition>
    <at part="Box002"/>
   </precondition>
   <effect>
    <at part="Box010"/>
   </effect>
  </assemblyOperation>
  <assemblyOperation action="base">
   <effect>
    <at part="Box001"/>
   </effect>
  </assemblyOperation>
  ...
 </assemblyPlan>
</plans>
```

As it has been previously commented in Section IV-A, in order to exploit the capabilities of a team with multiple UAS, at the allocation planner level, tasks can be decomposed to be shared among different UAS. That is the case of the assembly tasks associated to Box001, Box002 and Box003 whose weight of 900 grams is higher than the payload of a single UAS. The task for each of these parts is automatically decomposed into two different tasks of 450 grams.

The task allocation planner solves the assignment problem and in the solution it can be seen that the parts are assigned to their closer vehicles, and the three parts with higher weight are shared between two UAS. The solver was configured to apply two phases: a Construction Heuristic and a Metaheuristic. The Construction Heuristic chosen was the so called First Fit Decreasing, which assigns the more difficult planning entities first (those tasks with a higher part weight in our case), so it sorts the planning entities on decreasing difficulty. The Metaheuristic chosen was the Late Acceptance, a variant of the Hill Climbing local search. Late Acceptance does, for the assignment initially computed by the Construction Heuristic, some moves between the planning entities, one per iteration, and accepts any move that leads to a score that is better than the best score of a number of moves ago. This allows to do one move that initially leads to a worse score than the previous to improve the score calculated some moves ago.

After the initial assignment problem has been solved, the task allocation planner generates a planning problem in a format suitable for the symbolic planner based on JSHOP2. This JSHOP2 planning problem contains the high level task that represent the whole structure assembly and that should be decomposed into primitives, as well as the initial states of all the entities involved. Listing 4 shows the planning problem generated for our mission containing, among others, the assignment from parts to UAS generated by the task allocation planner and the dependencies computed for each of the parts by the assembly planner.

Listing 4: JSHOP2 planning problem.

```
; FACTS
(
 ; UAV defs
 (quadrotor uav1)(quadrotor uav2)
```

```
  (quadrotor uav3)(quadrotor uav4)
  ; location defs
  (location 1)
  ...
  (location 26)
  ; object defs
  (object Box001)
  ...
  (object Box011)
  ; object state defs
  (at Box001 1)
  ...
  (at Box011 21)
  ; ObjectDependencies
  (depends Box001 ())
  (depends Box002 ())
  (depends Box003 (Box008 Box009 Box010 Box011)
      )
  (depends Box004 (Box011 Box010))
  (depends Box005 (Box011 Box010))
  (depends Box006 (Box008 Box009))
  (depends Box007 (Box008 Box009))
  (depends Box008 (Box001))
  (depends Box009 (Box001))
  (depends Box010 (Box002))
  (depends Box011 (Box002))
  ; assembly locations
  (assembly_location Box001 2)
  ...
  (assembly_location Box011 22)
  ; part assignments
  (assigned Box003 uav1)
  (assigned Box004 uav1)
  (assigned Box001 uav1)
  (assigned Box008 uav1)
  (assigned Box002 uav2)
  (assigned Box007 uav2)
  (assigned Box011 uav2)
  (assigned Box006 uav3)
  (assigned Box009 uav3)
  (assigned Box003 uav3)
  (assigned Box002 uav3)
  (assigned Box010 uav4)
  (assigned Box005 uav4)
  (assigned Box001 uav4)
  ; UAS state defs
  (battery uav1 1200)
  (at uav1 23)(landed uav1)
  (battery uav2 1200)
  (at uav2 24)(landed uav2)
  (battery uav3 1200)
  (at uav3 25)(landed uav3)
  (battery uav4 1200)
  (at uav4 26)(landed uav4)
  ...
  ; remaining tasks
  (remaining_tasks 11)
)
; GOALS
((mission assemble))
```

The plan computed by the symbolic planner contains all the primitives for each of the UAS involved in the mission execution and it has been represented with a Gantt chart in Fig. 3. The symbolic planner produced a schedule of all the assembly tasks computed by the assembly planner. In the cases where multiple choices could be done, the planner decided to assemble first those parts that were dependencies for other parts that could not be assembled yet. That produced for the assignment computed by the task allocation planner the optimal scheduling of the assembly tasks.

VI. Conclusions and Future Work

The main contribution of this paper is the implementation of an assembly planner capable of generating assembly plans doing stability checks during the planning process, taking into account external forces such as the gravity. The system developed performs task assignment and ordering given the number of UAS in order to maximize the parallelism and cooperation in the mission.

The current system computes the optimal scheduling of the assembly tasks computed by the assembly planner given the assignment produced by the task allocation planner. Future work will explore an integrated optimal scheduling combining both task allocation and temporal task sequencing. In addition, as the forces generated by the UAS can have an impact in some cases in the stability of the structure, they will be added in the simulation stage of the assembly planner.

Finally the approach has been tested in missions involving multiple simulated UAS. In future work the goal is to execute the mission with real UAS in order to find more realistic and complex contexts to test the architecture developed.

References

[1] K. Kondak, K. Krieger, A. Albu-Schaeffer, M. Schwarzbach, M. Laiacker, I. Maza, A. Rodriguez-Castano, and A. Ollero, “Closed-loop behavior of an autonomous helicopter equipped with a robotic arm for aerial manipulation tasks,” *International Journal of Advanced Robotic Systems*, vol. 10, no. 145, pp. 1–9, February 2013. [Online]. Available: http://dx.doi.org/10.5772/53754

[2] D. Perez, I. Maza, F. Caballero, D. Scarlatti, E. Casado, and A. Ollero, “A ground control station for a multi-UAV surveillance system,” *Journal of Intelligent and Robotic Systems*, vol. 69, no. 1-4, pp. 119–130, January 2013. [Online]. Available: http://dx.doi.org/10.1007/s10846-012-9759-5

[3] I. Maza, F. Caballero, R. Molina, N. P. na, and A. Ollero, “Multimodal interface technologies for UAV ground control stations. a comparative analysis,” *Journal of Intelligent and Robotic Systems*, vol. 57, no. 1–4, pp. 371–391, 2010. [Online]. Available: http://dx.doi.org/10.1007/s10846-009-9351-9

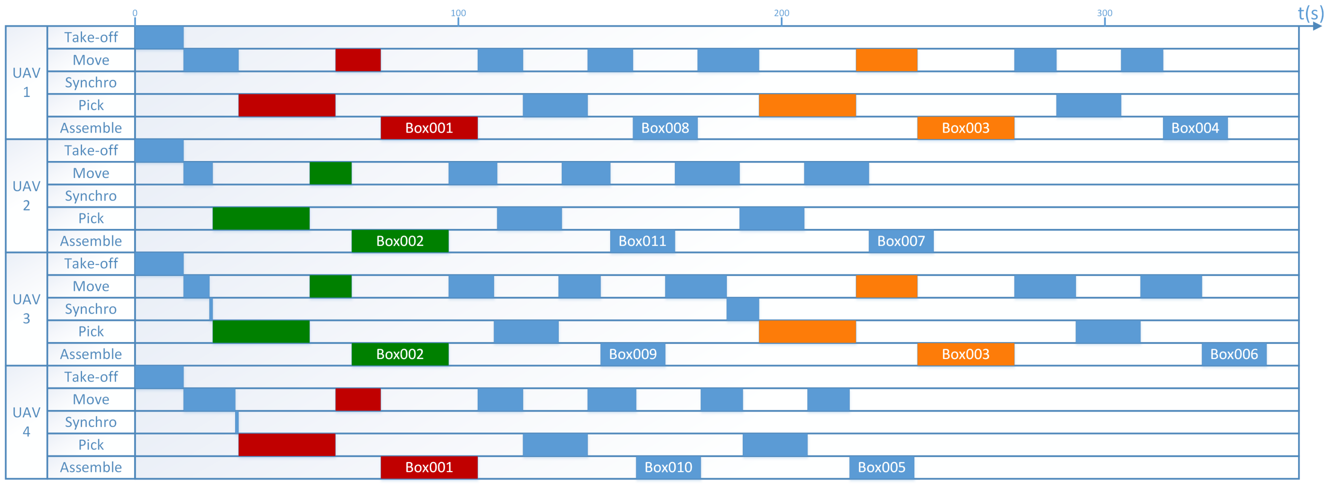

Fig. 3: Gantt chart of the final plan computed by the symbolic planner. Each rectangle represents a primitive task. Primitives with a string on top represent the assembly task of the part with the given name. Primitives of blue color are independent and are executed by the vehicles individually. Primitives of red, green and orange color are cooperatives and thus are executed by the UAS on which they appear simultaneously. Cooperative primitives appear on parts that must be managed by two UAS simultaneously due to their weight. Red primitives are executed cooperatively and simultaneosly by UAV1 and UAV4 in order to assemble the part Box001, green primitives by UAV2 and UAV3 to assemble Box002 and orange primitives by UAV1 and UAV3 to assemble Box003.

[4] I. Maza, J. Munoz-Morera, F. Caballero, E. Casado, V. Perez-Villar, and A. Ollero, "Architecture and tools for the generation of flight intent from mission intent for a fleet of unmanned aerial systems," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*. IEEE, May 2014, pp. 9–19. [Online]. Available: http://dx.doi.org/10.1109/ICUAS.2014.6842234

[5] I. Maza, F. Caballero, J. Capitan, J. M. de Dios, and A. Ollero, "A distributed architecture for a robotic platform with aerial sensor transportation and self-deployment capabilities," *Journal of Field Robotics*, vol. 28, no. 3, pp. 303–328, 2011. [Online]. Available: http://dx.doi.org/10.1002/rob.20383

[6] L. Kavraki, J.-C. Latombe, and R. H. Wilson, "On the complexity of assembly partitioning," *Information Processing Letters*, vol. 48, no. 5, pp. 229 – 235, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/002001909390085N

[7] P. Jimenez, "Survey on assembly sequencing: a combinatorial and geometrical perspective," *Journal of Intelligent Manufacturing*, pp. 1–16, 2011. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-79961144486&partnerID=40&md5=33af318d197c04cd5079192aab49a916

[8] F. Ingrand and M. Ghallab, "Robotics and Artificial Intelligence: A perspective on deliberation functions," *AI Communications*, vol. 27, no. 1, pp. 63–80, 2014.

[9] F. Bacchus, F. Kabanza, and U. D. Sherbrooke, "Using temporal logics to express search control knowledge for planning," *Artificial Intelligence*, vol. 116, p. 2000, 2000.

[10] J. Kvarnström and P. Doherty, "TALplanner: A temporal logic based forward chaining planner," *Annals of mathematics and Artificial Intelligence*, vol. 30, p. 2001, 2001.

[11] D. Nau, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman, "Shop2: An HTN planning system," *Journal of Artificial Intelligence Research*, vol. 20, pp. 379–404, 2003.

[12] D. E. Wilkins, *Practical Planning: Extending the Classical AI Planning Paradigm*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[13] K. Currie, A. Tate, and S. Bridge, "O-Plan: the open planning architecture," 1990.

[14] Y. Chen, B. W. Wah, and C. wei Hsu, "Temporal planning using subgoal partitioning and resolution in SGPlan," *J. of Artificial Intelligence Research*, vol. 26, p. 369, 2006.

[15] M. Helmert, "The fast downward planning system," *Journal of Artifical Intelligence Research*, pp. 191–246, 2006.

[16] A. Gerevini, A. Saetti, and I. Serina, "Planning through stochastic local search and temporal action graphs," *Journal of Artifical Intelligence Research*, vol. 20, pp. 239–290, 2003.

[17] A. Gerevini, U. Kuter, D. Nau, A. Saetti, and N. Waisbrot, "Combining domain-independent planning and HTN planing: The Duet Planner," 2008.

[18] V. Shivashankar, R. Alford, U. Kuter, and D. Nau, "The GoDel Planning System: A more perfect union of domain-independent and hierarchical planning."

[19] S. M. LaValle, "Planning algorithms," Cambridge, U.K., 2006, available at http://planning.cs.uiuc.edu/.

[20] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning," *The International Journal of Robotics Research*, vol. 28, no. 1, pp. 104–126, 2009. [Online]. Available: http://ijr.sagepub.com/content/28/1/104.abstract

[21] J. Wolfe, B. Marthi, and S. J. Russell, "Combined task and motion planning for mobile manipulation," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-27, March 2010. [Online].

Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-27.html

[22] F. Lagriffoul, D. Dimitrov, J. Bidot, A. Saffiotti, and L. Karlsson, "Efficiently combining task and motion planning using geometric constraints," *The International Journal of Robotics Research*, vol. 33, no. 14, pp. 1726–1747, 2014. [Online]. Available: http://ijr.sagepub.com/content/33/14/1726.abstract

[23] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction of cubic structures with quadrotor teams," in *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.

[24] Erwin Coumans, et al., *Bullet Physics Library*, 2015 (accessed February 17, 2015). [Online]. Available: http://www.bulletphysics.org

[25] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959. [Online]. Available: http://dx.doi.org/10.1287/mnsc.6.1.80

[26] Red Hat open source community, *OptaPlanner*, 2014 (accessed December 12, 2014). [Online]. Available: http://www.optaplanner.org/