



Decomposition and Cut Generation Strategies for Solving Multi-Robot Deployment Problems

Adriana Pacheco^(✉), Cédric Pralet, and Stéphanie Roussel

ONERA/DTIS, Université de Toulouse, 31055 Toulouse, France
{adriana.pacheco, cedric.pralet, stephanie.roussel}@onera.fr

Abstract. In this paper, we consider a multi-robot deployment problem involving a set of robots which must realize observation tasks at different locations and navigate through a shared network of waypoints. To solve this problem, we develop a two-level approach which alternates between (a) quickly obtaining high-level schedules based on a coarse grain CP model which approximates navigation tasks as setup times between observations, and (b) generating more accurate schedules based on a fine grain CP model which takes into account all resource usage conflicts during traversals of the shared network. The low-level layer also contains an explanation module able to generate constraints holding on high-level decision variables. These constraints (or cuts) account for interferences found in the low-level solutions and which the high-level scheduler should take into account to minimize the makespan. The proposed variants of the cut generation strategy are incomplete, the aim being to obtain good quality solutions in a short time, and they differ in the way they allow to diversify search. Experiments show the efficiency of this approach and the complementarity of the cut generation schemes proposed.

Keywords: Multi-Robot Missions · Constraint-based scheduling · Problem decomposition

1 Problem Description

We consider a Multi-Robot Deployment (MRD) problem where a fleet of robots must perform, as quickly as possible, a set of observations on specific areas of a field. Each candidate observation must be allocated to a robot, and for each robot the sequence of its observation tasks must be scheduled. Between two successive observations, a robot must also navigate through a network composed of waypoints and links, as illustrated in Fig. 1. Several candidate paths can be considered to navigate between pairs of observation areas, and each alternative path can be broken down into successive movements through links and waypoints. One specificity is that the network is shared between all robots. To avoid collisions during traversals of the network, or at least to reduce the need to deal with collision situations online, we consider that each link and each waypoint

can be occupied by at most one robot at a time (disjunctive resources). A finer version could be used by considering a non-unit capacity, but this would require handling cumulative resources which are left for future work. The only locations which are considered as sharable are those associated with observation areas (see Fig. 1 again). The rationale for this assumption is that the robots have a smaller (or even null) speed when performing observations at these locations, so the online management of collisions is easier and less hazardous in this case. According to the time frame during which each move monopolizes the shared link and waypoint resources, two different approaches can be distinguished:

- *Minimum Handover*. In this first approach, a robot r successively consumes the network resources involved in a path, as illustrated in Fig. 2 for a transition between observations o and o' successively using link l_1 , waypoint wp , and link l_2 . One specificity though is that there exists a positive time lapse, called the *handover duration*, during which a robot switches between network resources (move between a link and a waypoint for example). During each handover period, the robot moves to the next resource on its path, but must also remain “connected” to the previous one, to prevent robots from “jumping” between resources. The goal is to forbid inconsistent solutions where two robots would instantaneously cross each other over the network (*e.g.* solutions where at a given time t , one robot instantaneously moves for link l to waypoint w while another one instantaneously moves from w to l).
- *Path Isolation*. In this second approach, for a transition of robot r between two observations o and o' , all path resources used during the transition are reserved for the whole move duration, as illustrated in Fig. 3. The path monopolization starts just before the robot leaves observation o and ends just after the robot arrives to observation o' .

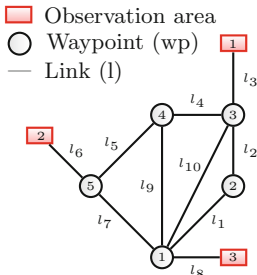


Fig. 1. Observation field

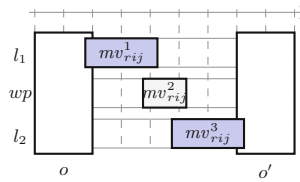


Fig. 2. Minimum handover

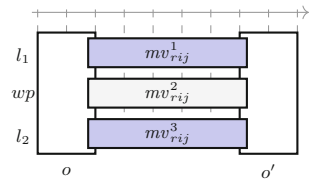


Fig. 3. Path isolation

The two approaches differ in terms of robustness and in terms of required synchronization between the robots at execution time, where duration of robot moves can be shorter or longer than expected. With the minimum handover strategy, the usage of resources is more finely optimized but there is a need

to synchronize the robots at each basic move. On the contrary, the path isolation strategy takes more margins to get a collision-free deployment, but it only requires synchronizing the robots at the start and at the end of the global moves between observation tasks, that is when the speed of robots is low. The path isolation approach is inspired by works dealing with inter-core interferences due to shared hardware resources in multi-core processors [4, 16], to temporally isolate hard real-time applications [12].

In the mission specifications, robots cannot perform more than one observation at a time, and they must transfer observation data in real time to a mission center, using a specific emission frequency. To avoid communication interferences, two robots that use the same frequency cannot transfer observation data simultaneously. Redundancy is also useful in this kind of application to be robust to robot failures at execution time, therefore each observation area must be observed by several different robots at times spaced by a defined lapse. Last, each robot is initially located at a given location and must come back at the end of the mission to a predefined goal location.

The rest of the paper is organized as follows. Section 2 describes a first possible approach for dealing with the MRD problem based on a global CP model. Section 3 describes a two-layer approach which decomposes the global problem into a coarse-grain scheduling layer L1 used to compute sequences of observations, and a fine-grain scheduling layer L2 responsible for detailing navigation paths through the shared network of waypoints. Section 4 then introduces an iterative process of interaction between layers L1 and L2, based on four different *cut generation strategies* that allow L2 to provide L1 with new relevant constraints. This strategy can be seen as a kind of Logic-Based Benders Decomposition [5] (more details later on the relationship with LBDD). Section 5 presents the results of the decomposed approach and shows the complementarity of the cut generation techniques introduced. Finally, Sect. 6 concludes and gives some perspectives.

Note that the MRD application has been widely addressed to tackle real-world problems related to situation awareness issues, such as cooperative sensing using air-ground teams [3], disaster response [13], and exploration-rescue systems in hostile environments [14]. The aim of this paper is to propose more generic solutions that can be applied to such industrial and academic applications. Also, one of our goals is to exploit existing CP solvers to the best of their potential, and not to compare them with other resolution techniques such as MILP [7], PDDL [15] or Greedy algorithms. Finally, the proposed approach seeks to decrease the computational complexity of the MRD application, by considering the navigation interferences of a detailed solution plan. In most of the existing references, these interferences are not taken into consideration at a planning stage, and the use of anti-collision mechanisms is supposed at execution time.

2 A First Global Constraint-Based Scheduling Approach

Input Data. A first possible approach to solve the MRD problem is to develop a global CP model covering all specifications of the mission. To define such a model, we consider the following input data:

- a set of *frequencies* \mathcal{F} available for communicating observation data in real time to the mission center;
- a set of *robots* \mathcal{R} ; for each robot $r \in \mathcal{R}$, $freq_r \in \mathcal{F}$ is the (unique) frequency used by r to emit data during observations;
- a set of *observation areas* \mathcal{A} , corresponding to areas of the field that must be observed; for each area $a \in \mathcal{A}$, $duObs_a \in \mathbb{N}$ denotes the duration required to observe a .
- a number *NobsPerArea* of observations required over each observation area; all observations of a given area must be performed by distinct robots for redundancy issues;
- a set of *observations* \mathcal{O} to be performed, which contains as many elements as the number of (a, k) pairs in $\mathcal{A} \times [1..NobsPerArea]$; for each observation $o \in \mathcal{O}$, $ar_o \in \mathcal{A}$ denotes the area associated with o ;
- for each robot $r \in \mathcal{R}$, two specific observations denoted by α and β which represent virtual observations that must be performed at the beginning and at the end of the plan of r respectively; fictitious observations α and β allow us to model the initial and goal locations of r ;
- a set of connected shared *waypoints* \mathcal{W} ; for each robot $r \in \mathcal{R}$ and each waypoint $w \in \mathcal{W}$, $duMv_{r,w} \in \mathbb{N}$ is the minimum duration spent in w during a navigation of r through w ; the observation areas are not considered as shared waypoints;
- a set of *links* \mathcal{L} , which correspond to direct connections between adjacent waypoints or between an observation area and a waypoint; for each robot $r \in \mathcal{R}$ and each link $l \in \mathcal{L}$, $duMv_{r,l} \in \mathbb{N}$ is the minimum duration required by r to traverse link l ;
- a minimum temporal distance d between two observations of the same area;
- a *temporal horizon* $H \in \mathbb{N}$ available for the whole mission.

Constraint-Based Scheduling Model. From the previous input data, we can define a constrained-based scheduling model. For space limitation reasons, and because this global approach does not scale well compared to the decomposed approach detailed later, we only give the main lines of the model.

Basically, the global CP model is built upon so-called *interval variables* used in CP Optimizer. If $[Ts, Te]$ denotes the time frame available for realizing the associated task, each interval variable itv is characterized by a start value $start(itv) \in [Ts, Te]$, an end value $end(itv) \in [Ts, Te]$, and a presence $pres(itv) \in \{0, 1\}$ expressing whether the task is present in the solution schedule. The global CP model involves intervals representing observation activities and navigation activities through the shared network. In particular, it contains a huge number of optional intervals $mv_{r,o,o',p,q}$ to model the move of robot r on the

q th network resource (link or waypoint) of the p th path available to travel from observation o to observation o' . To boost constraint propagation, the model also contains a coarse-grain no overlap constraint taking into account the minimum temporal distance between observation tasks. Also, a so-called *sequence variable* seq_r is associated with each robot $r \in \mathcal{R}$. The value of this variable corresponds to a total ordering of all observation tasks realized by r . The CP model contains constraints over such sequences to ensure that the start and end locations of a path chosen between two observations o, o' is consistent with the locations of o and o' . Finally, a solution is optimal if it minimizes the makespan C_{\max} , which corresponds to the maximum end time of the fictitious last observation tasks realized by the robots to reach their goal locations.

3 Decomposition into a Two-Layer Scheduling Problem

To decrease the computational complexity, the MRD problem can be split into two parts: (1) one part which decides on the successive observations realized by each robot based on a coarse grain model of navigation operations (so-called layer L1), and (2) one part responsible for detailing the navigations of robots within the network of shared waypoints (so-called layer L2).

3.1 Coarse-Grain Scheduling Model: Layer L1

In the high-level scheduling model of layer L1, the navigation between two given observations o and o' is abstracted in a very coarse way as a simple integer setup time required between the end of the realization of o and the start of the realization of o' . In other words, it considers the robots as disjunctive resources with setup times, as in Sequence Dependent Setup Time Job Shop Scheduling Problems [8].

Inputs. In addition to some inputs already mentioned in Sect. 2 (the set of *frequencies*, the set of *robots*, the set of *observation areas*, the set of *observations* to be performed, and the *temporal horizon*), an additional input of the high-level MRD scheduling problem considers:

- a constant setup time $\text{setup}_{r,o,o'} \in \mathbb{N}$ for each robot $r \in \mathcal{R}$ and each pair of observations (o, o') successively realized by r ; this setup time represents the minimum duration required for r between the end of o and the start of o' ; it is obtained from the length of the shortest path available to go from the location of o to the location of o' ; this length is computed in polynomial time in preprocessing, and it corresponds to an optimistic evaluation assuming that r is alone over the network during its navigation from o to o' .

We also define $\text{setup}_{r,\alpha,o}$ as the shortest duration required to move from the initial location of r to observation o , and $\text{setup}_{r,o,\beta}$ as the shortest duration required to move from observation o to the goal location of r . For each robot $r \in \mathcal{R}$, the associated function to obtain the setup time between each pair of observations, is denoted setup_r .

Scheduling Problem. To define a CP model for layer L1, we use several scheduling constructs available in the CP Optimizer tool. More precisely, we introduce the following decision variables:

- for each observation $o \in \mathcal{O}$, one *interval variable* obs_o which must be placed during time frame $[0, H]$ and whose duration is $duObs_{ar_o}$, that is the observation duration of the area associated with o ;
- for each observation $o \in \mathcal{O}$ and each robot $r \in \mathcal{R}$, one *optional interval variable* $obs_{o,r}$ used to represent the realization of observation o by robot r ;
- for each robot $r \in \mathcal{R}$, two (non-optional) interval variables $obs_{\alpha,r}$ and $obs_{\beta,r}$ representing fictitious observations that r must realize at the beginning and end of its plan respectively; we recall that these fictitious observations allow us to model the initial and goal locations of the robots; interval $obs_{\alpha,r}$ has a null duration and must be placed at time 0, and interval $obs_{\beta,r}$ has a null duration and must be placed during time frame $[0, H]$;
- for each robot $r \in \mathcal{R}$, one *sequence variable* \mathbf{seq}_r which represents an ordering over all present intervals associated with r , *i.e.* over all present intervals in set $\{obs_{o,r} \mid o \in \mathcal{O} \cup \{\alpha, \beta\}\}$.

The set of decision variables of the high-level model is therefore

$$V^1 = (\cup_{o \in \mathcal{O}} \{obs_o\}) \cup (\cup_{o \in \mathcal{O} \cup \{\alpha, \beta\}, r \in \mathcal{R}} \{obs_{o,r}\}) \cup (\cup_{r \in \mathcal{R}} \{\mathbf{seq}_r\}) \quad (1)$$

Constraints 2 to 7 are imposed over these variables. Constraint (2) imposes that the first and last observations in a robot sequence must correspond to the initial and final fictitious observations. Constraints (3) uses the *alternative* constraint of CP Optimizer and expresses that each observation is realized by a unique robot. Constraint (4) imposes that each robot can realize at most one observation for a given area (redundancy requirement). Constraint (5) expresses that observation tasks using the same frequency cannot overlap. Constraint (6) expresses that observations of a given area cannot overlap, taking into account the minimum delay d defined in the input data. Constraint (7) enforces that observation tasks using the same robot must not overlap, taking into account the approximated setup durations required to move from one observation to the next for each robot. Symmetry breaking constraints could also be added.

$$\forall r \in \mathcal{R}, first(\mathbf{seq}_r, obs_{\alpha,r}) \wedge last(\mathbf{seq}_r, obs_{\beta,r}) \quad (2)$$

$$\forall o \in \mathcal{O}, alternative(obs_o, \{obs_{o,r} \mid r \in \mathcal{R}\}) \quad (3)$$

$$\forall a \in \mathcal{A}, \forall r \in \mathcal{R}, \sum_{o \in \mathcal{O} \mid ar_o = a} pres(obs_{o,r}) \leq 1 \quad (4)$$

$$\forall f \in \mathcal{F}, noOverlap(\{obs_{o,r} \mid o \in \mathcal{O}, r \in \mathcal{R}, freq_r = f\}) \quad (5)$$

$$\forall a \in \mathcal{A}, noOverlap(\{obs_{o,r} \mid o \in \mathcal{O}, r \in \mathcal{R}, ar_o = a\}, d) \quad (6)$$

$$\forall r \in \mathcal{R}, noOverlap(\{obs_{o,r} \mid o \in \mathcal{O} \cup \{\alpha, \beta\}, setup_r\}) \quad (7)$$

The objective is to minimize the makespan, defined as the time at which each robot reaches its goal position:

$$\text{minimize } \max_{r \in \mathcal{R}} end(obs_{\beta,r}) \quad (8)$$

Output. A high-level *solution schedule* σ^1 for layer L1 is an assignment of the values of all the decision variables in V^1 that satisfies all the problem constraints. A solution schedule σ^1 is said to be optimal if it minimizes the makespan.

3.2 Fine-Grain Scheduling Model: Layer L2

The low-level scheduling model of layer L2 takes into consideration all the navigation paths available in the waypoint graph representing the observation field (see Fig. 1), to detail the routing of robots in the shared network and manage navigation conflicts. This decision layer only considers the navigations between the observation tasks which are present in the coarse-grain solution σ^1 produced by L1 (much less navigation options compared to the global CP model).

Inputs. In addition to the inputs mentioned in Sect. 2 for the MRD problem, the additional inputs of the low-level multi-robot scheduling problem are:

- the high-level solution schedule σ^1 produced by layer L1; in this solution, we keep for each robot r the value of sequence \mathbf{seq}_r , which defines the successive observations planned for r ; to have flexibility in L2, we do not keep the exact dates found by layer L1 for present intervals in σ^1 ; in the following, to make some notations easier, we denote by Tr the set of all triples (r, o, o') such that in solution σ^1 , observation o' is realized just after observation o for r ;
- for each robot r and each pair of successive observations (o, o') realized by r (i.e. $(r, o, o') \in Tr$), a set of candidate paths $P_{r,o,o'}$ which can be used by r to go from o to o' ; this set contains all paths whose length is not longer than the duration between the end of $obs_{o,r}^{L1}$ and the start of $obs_{o',r}^{L1}$ in the plan generated by layer L1; each path p in $P_{r,o,o'}$ is specified by the sequence $[p_1, \dots, p_Q]$ of successive network resources (waypoints or links) traversed by the path ($p_q \in \mathcal{W} \cup \mathcal{L}$ for every $q \in [1..Q]$); for instance, in Fig. 1, several paths are available to move from observation area 3 to observation area 1; according to the solution obtained in σ^1 , Fig. 4 details the setup operations between observation tasks $O_{3,2}$ and $O_{1,2}$ performed by robot 2, which can use either path $[l_8, wp_1, l_1, wp_2, l_2, wp_3, l_3]$ or path $[l_8, wp_1, l_{10}, wp_3, l_3]$.

Scheduling Problem. In the scheduling problem built for layer L2, the detailed routing between observation tasks must be defined. For space limitation reasons, we give the model only for the minimum handover case (Fig. 2). The model for the path isolation case (Fig. 3) is a bit simpler. For each robot $r \in \mathcal{R}$ and each observation o realized by r , the CP model contains one interval variable $obs_{o,r}$ as in L1. For each robot $r \in \mathcal{R}$ and each pair of observations o, o' successively realized by r in the solution found by L1, we also consider the following variables:

- one (mandatory) interval variable $mv_{r,o,o'}$ representing the global move of r from o to o' ;
- for each candidate path $p \in P_{r,o,o'}$, one optional interval variable $mv_{r,o,o',p}$ representing a move along path p ;

- for each candidate path $p = [p_1, \dots, p_Q] \in P_{r,o,o'}$ and each index $q \in [1..Q]$, one optional interval variable $mv_{r,o,o',p,q}$ representing the usage of the q th network resource of path p .

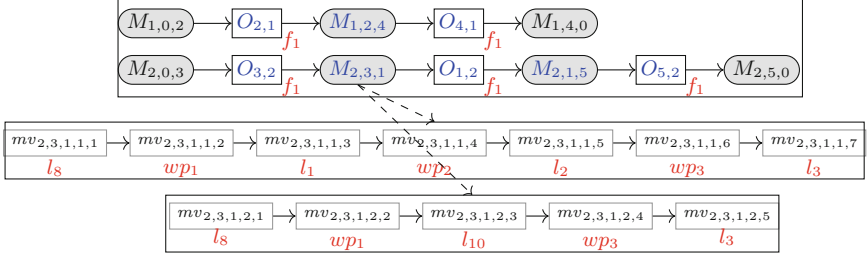


Fig. 4. Possible global move decompositions for layer L2

Together, these interval variables make up the set of decision variables V^2 of layer L2. Fine-grain constraints associated with the minimum handover case are given below. Constraint (9) forbids the temporal overlapping of tasks that use the same link or waypoint. Constraint (10) ensures that exactly one path is used between each pair of successive observations. Constraint (11) expresses that this path spans all its elementary moves. Constraint (12) states that the presences of elementary moves must be consistent with the presences of the selected paths. Constraints (13) and (14) define the start and end times of the moves from and to the first and last fictitious observations respectively. Constraints (15) to (17) enforce a handover period between the successive intervals involved in a chosen navigation path. Constraint (18) defines the minimum duration of each elementary move interval, taking into account the handover period. We consider an inequality here since in the minimum handover configuration, a robot is allowed to wait on a link or a waypoint. Constraint (19) forbids the temporal overlapping of tasks that use the same frequency (the ordering of observations over frequency resources is not transferred from L1 to L2 to keep more flexibility in L2). Finally, the goal is still to minimize the makespan (same expression as in Eq. (8)).

$$\forall \gamma \in \mathcal{W} \cup \mathcal{L}, \text{ noOverlap}(\{mv_{r,o,o',p,q} \mid \quad (9)$$

$$((r, o, o') \in Tr) \wedge (p \in P_{r,o,o'}) \wedge (q \in [1..|p|]) \wedge (p_q = \gamma)\}$$

$$\forall (r, o, o') \in Tr, \text{ alternative}(mv_{r,o,o'}, \{mv_{r,o,o',p} \mid p \in P_{r,o,o'}\}) \quad (10)$$

$$\forall (r, o, o') \in Tr, \forall p \in P_{r,o,o'}, \text{ span}(mv_{r,o,o',p}, \{mv_{r,o,o',p,q} \mid q \in [1..|p|]\}) \quad (11)$$

$$\forall (r, o, o') \in Tr, \forall p \in P_{r,o,o'}, \forall q \in [1..|p|], \quad (12)$$

$$\text{pres}(mv_{r,o,o',p}) = \text{pres}(mv_{r,o,o',p,q})$$

$$\forall (r, \alpha, o') \in Tr, \text{ endAtStart}(\text{obs}_{\alpha,r}, mv_{r,\alpha,o'}) \quad (13)$$

$$\forall (r, o, \beta) \in Tr, \text{ endAtStart}(mv_{r,o,\beta}, \text{obs}_{\beta,r}) \quad (14)$$

$$\forall(r, o, o') \in Tr, \forall p \in P_{r,o,o'}, \forall q \in [2..|p|], \quad (15)$$

$$pres(mv_{r,o,o',p}) \rightarrow (start(mv_{r,o,o',p,q}) = end(mv_{r,o,o',p,q-1}) - 1) \\ \forall(r, o, o') \in Tr, \forall p \in P_{r,o,o'}, \quad (16)$$

$$pres(mv_{r,o,o',p}) \rightarrow (start(mv_{r,o,o',p,1}) = end(obs_{o,r}) - 1) \\ \forall(r, o, o') \in Tr, \forall p \in P_{r,o,o'}, \quad (17)$$

$$pres(mv_{r,o,o',p}) \rightarrow (end(mv_{r,o,o',p,|p|}) = start(obs_{o',r}) + 1) \\ \forall(r, o, o') \in Tr, \forall p \in P_{r,o,o'}, \forall q \in [1..|p|], \quad (18)$$

$$pres(mv_{r,o,o',p,q}) \rightarrow (end(mv_{r,o,o',p,q}) - start(mv_{r,o,o',p,q}) \geq duMv_{r,p,q} + 2) \\ \forall f \in \mathcal{F}, noOverlap(\{obs_{o,r} \mid o \in \mathcal{O}, r \in \mathcal{R}, freq_r = f\}) \quad (19)$$

Output. A low-level *solution schedule* σ^2 for layer L2, is an assignment of all variables in V^2 that satisfies all the problem constraints. It corresponds to a solution of the global MRD problem. A solution schedule σ^2 is said to be optimal if it minimizes the makespan (end time of the fictitious last observation tasks performed by the robots).

4 Iteration Resolution and Cut Generation Strategies

4.1 Iterative Resolution Approach

When using a top-down approach such as the one described in the previous section, the highest quality solutions may be missed since high-level decisions are computed from a coarse-grain model. This is why we use an iterative resolution strategy related to Logic-Based Benders Decompositions (LBBD), where a master solver iteratively proposes solutions to a slave solver which generates new constraints called *cuts*. Iterations between the master and the slave solvers are realized until convergence or until a maximum CPU time is reached. In our case, layer L1 first transfers to layer L2 the sequence of tasks realized by each robot. Then, layer L2 obtains a consistent solution schedule σ^2 for the low-level scheduling problem. In L2, we introduce an explanation module which detects interferences between tasks consuming the shared network resources. As shown in Fig. 5, this explanation module synthesizes cuts which are sent as a feedback to L1.

Compared to standard LBBD, one specificity of the technique proposed is that, as shown later, the explanation module generates cuts that are not necessarily valid in the sense that they might prune optimal solutions. The purpose of these cuts, which could be called *heuristic cuts*, is not to converge towards an optimal solution, but to speed the search for good solutions by forbidding in a coarse way some observation sequence patterns which might lead to interferences on detailed navigation activities. These patterns can be more or less precise and the generated cuts range from cuts usable to intensify search around the best known solution to cuts usable for exploring completely different regions of the search space. We emphasize again that the purpose of this process is not

to obtain an optimal solution for the global problem but to get good quality solutions within a short computation time, which is more crucial than finding optimality in most MRD problems. Last, to perform several iterations between L1 and L2, we do not solve each problem in L1 or L2 to optimality. Instead, each run of L1 and L2 has a maximum allocated CPU time, which depends on the problem instance considered.

The approach proposed also differs from a strategy introduced in a previous work [10]. First, from an application point of view, [10] considers a simpler problem where the only disjunctive network resources are the links, and where the robots can freely cross each other at waypoints, in contrast to the approach followed in our “continuous” minimum handover strategy. The latter makes the model of layer L2 more complex but has the advantage of being more collision-safe. Also, [10] does not consider the path isolation configuration, which makes the moves of robots more constrained but which can be useful for robustness reasons. From a technical point of view, [10] also considers a two-layer approach but without any generation of cuts. Instead, the feedback from L2 to L1 corresponds to a simple update of the abstract setup durations of L1 by formula $setup_{r,o,o'} \leftarrow (1 - \mu) \cdot setup_{r,o,o'} + \mu \cdot du$, where μ corresponds to a learning rate and du corresponds to the duration of transition $o \rightarrow o'$ obtained for robot r in layer L2. Doing so, layer L1 learns a setup duration model from L2 and is close to work on surrogate models for optimization [6]. On the opposite, our approach exploits more detailed information (see Sect. 4.2) and is closer to LBDD. Other works already addressed similar real-world applications using two-stage decompositions involving CP models (Decomposed-CP), such as the deployment of multiple robots to assist the residents in a retirement home environment [15]. This last work also involves robots moving through the environment and a number of tasks that potentially increases with the number of robots and locations, and it also sets the value of certain decision variables for the sub-problem (layer L2), using the solution of the master problem (layer L1). Their Decomposed-CP approach may not find the optimal solution, but one distinctive feature is that no feedback loop from L2 to L1 is used. On this point, the authors of [15] state that it’s not straightforward to determine whether their problem structure allows for a decomposition such as LBDD to be implemented.

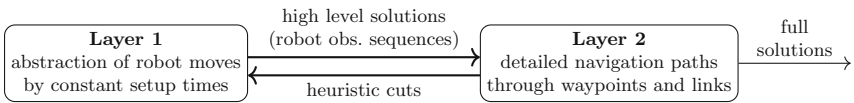


Fig. 5. Interactions between the decision layers

4.2 Cut Generation in the Explanation Module of Layer L2

The explanation module of L2 returns information about interferences found in the low-level solution and that have a negative impact on makespan minimization. These interferences are detected by examining conflicts related to the

usage of network resources during path traversals. More precisely, for a given robot r_1 , if the duration required to traverse a path between two successive observations o_1, o'_1 is strictly greater than the duration obtained in the solution of L1, this means that there is a resource precedence constraint which creates an interference at some point during the transition from o_1 to o'_1 . The goal of the explanation module of L2 is to detect interferences related to network resource usages in the obtained sequence from σ^2 . For instance, Fig. 6 illustrates a scenario where two robots are in conflict for using waypoint wp_1 and wp_2 , and link l_2 to traverse the paths needed to perform the sequences of observations shown in Figs. 7 and 8 (handover duration not represented). In this case, the duration of the transition is longer for robot r_1 since it must wait for some network resources to be released by r_2 . The explanations of these longer transition durations are depicted in red in Figs. 7 and 8. In the general case, the explanation module of L2 detects through critical path analysis all triples (r_2, o_2, o'_2) such that there is a transition from observation o_2 to observation o'_2 for robot r_2 and such that at some point between o_1 and o'_1 , robot r_1 waits for a network resource to be released by r_2 during its transition from o_2 to o'_2 . In terms of scheduling, we identify the critical resource precedence constraints associated with the network resources. In the end, each interference produced by the explanation module is defined by a 6-tuple $(r_1, o_1, o'_1, r_2, o_2, o'_2)$. In the following, the set of all interferences synthesized from low-level solution σ^2 is denoted by $\mathbf{Itf}(\sigma^2)$.

Four categories of cuts that can be generated through the explanation module are introduced below, by increasing order of refinement. In the following, we respectively denote by x^{L1} and x^{L2} the variables manipulated by layers L1 and L2. For instance, $obs_{o,r}^{L1}$ denotes the observation interval of o by robot r manipulated by L1, while $obs_{o,r}^{L2}$ denotes the observation interval $obs_{o,r}$ manipulated by L2 for representing the same task. Moreover, to get more concise expressions, we denote by $next_{r,o,o'}^{L1} \in \{0,1\}$ the variable taking value 1 if interval $obs_{o,r}^{L1}$ is the predecessor of interval $obs_{o',r}^{L1}$ in the sequence of intervals \mathbf{seq}_r^{L1} associated with robot r in layer L1. Also, $\sigma^2(start(obs_{o,r}^{L2}))$ and $\sigma^2(end(obs_{o,r}^{L2}))$, denote respectively the start and the end date of $obs_{o,r}^{L2}$ in the low-level solution σ^2 .

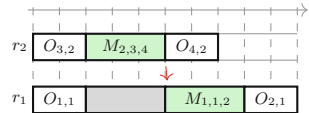
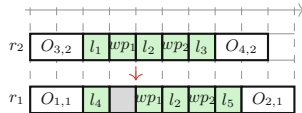
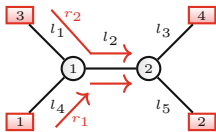
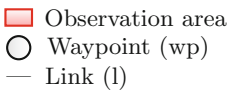


Fig. 6. Interference between robots

Fig. 7. Minimum handover interferences

Fig. 8. Path isolation interferences

Broad Cuts: Setup Times (Cuts C1). From the set of interferences $\mathbf{Itf}(\sigma^2)$, temporal constraints holding on high-level decision variables can be added to the scheduling problem of layer L1. A first possible approach is to return the following cuts:

$$\forall(r_1, o_1, o'_1, r_2, o_2, o'_2) \in \mathbf{Itf}(\sigma^2), \quad (20)$$

$$start(obs_{o'_1, r_1}^{L1}) - end(obs_{o_1, r_1}^{L1}) \geq \sigma^2(start(obs_{o'_1, r_1}^{L2})) - \sigma^2(end(obs_{o_1, r_1}^{L2}))$$

Such cuts are not valid since the initial abstract setup duration between o_1 and o'_1 for r_1 (the setup duration considered by L1) could be met by updating the sequences of observations realized by other robots. However, these cuts can allow to quickly diversify search by penalizing, at the level of L1, a transition $o_1 \rightarrow o'_1$ which *might* lead to an interference at the level of network resources.

Note that cuts C1 are equivalent to use the solution σ^2 from layer L2 to update the inputs of layer L1 (the coarse-grain duration of the setup operations between locations for each robot). Remember that $setup_{r,o,o'} \in \mathbb{N}$ corresponds to the high-level approximation of the duration required by r to move from the location of observation o to the location of observation o' over all possible paths of the waypoint network. The previous cuts amount to update $setup_{r,o,o'}$ by:

$$\forall(r_1, o_1, o'_1, r_2, o_2, o'_2) \in \mathbf{Itf}(\sigma^2), \quad (21)$$

$$setup_{r_1, o_1, o'_1} \leftarrow \max(setup_{r_1, o_1, o'_1}, \sigma^2(start(obs_{o'_1, r_1}^{L2})) - \sigma^2(end(obs_{o_1, r_1}^{L2})))$$

Moderate Cuts: Setup Times and Sequencing (Cuts C2). In contrast to the previous cut generation strategy, we can consider another category of cuts which take into consideration the precise transitions creating the interference in σ^2 . Such cuts are defined by:

$$\forall(r_1, o_1, o'_1, r_2, o_2, o'_2) \in \mathbf{Itf}(\sigma^2), \quad (22)$$

$$(next_{r_1, o_1, o'_1}^{L1} \wedge next_{r_2, o_2, o'_2}^{L1} \rightarrow$$

$$(start(obs_{o'_1, r_1}^{L1}) - end(obs_{o_1, r_1}^{L1}) \geq \sigma^2(start(obs_{o'_1, r_1}^{L2})) - \sigma^2(end(obs_{o_1, r_1}^{L2}))))$$

and can be added to the scheduling problem of layer L1. These cuts impose longer setup times in the high-level approximation whenever the successive observations involved in the interferences are successive again in a new sequence considered by L1. Cuts of type C2 are weaker than cuts of type C1, meaning that C2 prunes less solutions than C1.

Refined Cuts: Setup Times, Sequencing, and Temporal Positioning (Cuts C3). More refined cuts coming from the solution analysis of layer L2 can be sent to layer L1. Unlike the previous cut generation strategies, these new cuts consider the time frame during which the setup tasks between observations are performed. Basically, they add high-level constraints which impose longer coarse-grain setup times only in case of temporal overlapping between the transitions

involved in the interference. More precisely, let $overlap^{L1}(r_1, o_1, o'_1, r_2, o_2, o'_2)$ denote an expression taking value true when transitions $o_1 \rightarrow o'_1$ and $o_2 \rightarrow o'_2$ overlap in time, that is:

$$overlap^{L1}_{r_1, o_1, o'_1, r_2, o_2, o'_2} = (end(obs^{L1}_{o_1, r_1}) > start(obs^{L1}_{o'_2, r_2})) \wedge (end(obs^{L1}_{o_2, r_2}) > start(obs^{L1}_{o'_1, r_1})) \quad (23)$$

The detailed cuts are then given by:

$$\begin{aligned} & \forall (r_1, o_1, o'_1, r_2, o_2, o'_2) \in \mathbf{Itf}(\sigma^2), \quad (24) \\ & (next^{L1}_{r_1, o_1, o'_1} \wedge next^{L1}_{r_2, o_2, o'_2} \wedge overlap^{L1}_{r_1, o_1, o'_1, r_2, o_2, o'_2}) \rightarrow \\ & (start(obs^{L1}_{o'_1, r_1}) - end(obs^{L1}_{o_1, r_1}) \geq \sigma^2(start(obs^{L2}_{o'_1, r_1}) - \sigma^2(end(obs^{L2}_{o_1, r_1})))) \end{aligned}$$

which means that if transitions $o_1 \rightarrow o'_1$ and $o_2 \rightarrow o'_2$ appear again in a solution for L1 and if these transitions overlap in time, then a higher setup time must be used at the level of L1. Cuts of type C3 are weaker than cuts of type C2, meaning that C3 prunes less solutions than C2.

Valid Global Cut (Cut C4). A quite simple valid cut consists in forbidding the entire sequence obtained for L1 at the previous step. This cut is defined by:

$$\neg \left[\bigwedge_{(r, o, o') \in Tr} next^{L1}_{r, o, o'} \right] \quad (25)$$

It can be added to the scheduling problem of L1 as a global scheduling constraint. This cut will only force to seek for a different high-level solution, bypassing the synthesized information about the interferences found.

5 Experiments

Benchmarks. The two-layer approach and the four cut generation strategies proposed were evaluated over several MRD problem instances containing from 1 to 15 observation areas, connected through a network of shared links and waypoints. Several randomly generated observation scenarios were tested, considering from 1 to 3 frequencies available to transfer observation data, and from 2 or 3 homogeneous robots available to carry out the observations. The fields generated are regular grids of size $N \times M$ containing waypoints which are connected to their 4 adjacent neighbors. Random fields such as the one in Fig. 1, and other grid configurations were also tested, leading to the same experimental conclusion. Observation areas are randomly positioned so as to be connected to one waypoint of the grid, and for most observation pairs o, o' there are several navigation paths of minimum length from o to o' . Each area requires observations from 1 to 3 robots (redundancy). The generated instances were all tested on IBM ILOG CP Optimizer 12.5 on an Intel Xeon CPU E5-1603, 2.80 GHz 8 GB RAM, setting an adequate number of iterations depending on the problem size

and on `cpuMax`. Experiments were performed for both the minimum handover and path isolation configurations, to test the algorithms on instances which are more or less constrained in terms of usage of the shared network.

Representative results of the tested configurations are given in Figs. 9 and 10, where two different robots must observe each observation area. For nearly all problem instances, the four proposed strategies for the two-layer approach achieve better makespan results than the global CP approach, in a significantly shorter computation time. They provide good quality solutions in just a few seconds, even for the largest instances for which the global CP approach is not able to reach any solution with a CPU time of 30 min. For the smallest instances, most of the strategies of the two-layer approach manage to find the optimal solution, but without proving its optimality. As shown in Table 1, the results also demonstrate that over the set of benchmarks tested, there is not a single winner among the four cut generation strategies for the two-layer approach. One explanation is that for some instances, it may be more advantageous to diversify the exploration of the search space by generating moderate cuts (strategy C2) or coarse-grain cuts updating the entire set of setup times (strategy C1), while for other instances it may be more convenient to explore a search space not so far from the current problem by generating fine-grain cuts (strategies C3 and C4). In other words, there is a kind of exploration/exploitation trade-off depending on the instance, leading to a disparity in the number of added cuts and in the elapsed time until the best solution is found, averaging between 1 and 2 min for the different strategies. To take advantage of all cuts, the next step would be to define a portfolio solver exploiting the different kinds of cuts, the goal being to outperform each individual cut generation strategy. Portfolio approaches combine different solvers to get a globally better one, and their efficiency was already shown in the CP field [1,2,9].

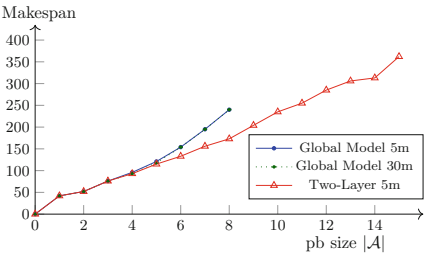


Fig. 9. Minimum handover

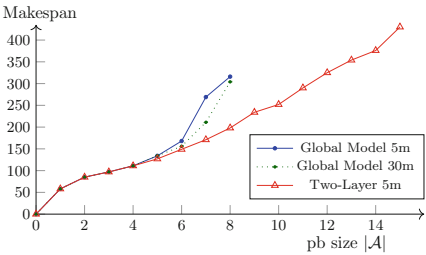


Fig. 10. Path isolation

Table 1. Makespan found along with the number of cuts added until the best solution is found (in brackets) for different sizes of the set of observation areas \mathcal{A} and for both configurations (Minimum Handover and Path Isolation); results are given for cut generation strategies C1 to C4, with 5-min time limit (cpuMax), and for the global CP model, with 5 and 30-min time limits

	Minimum Handover						Path Isolation								
	Cut Strategies					Global		Cut Strategies						Global	
A	C1	C2	C3	C4		5m	30m	C1	C2	C3	C4		5m	30m.	
1	42 [0]	42 [0]	42 [0]	42 [0]		42	42	58 [0]	58 [0]	58 [0]	58 [0]		58	58	
2	52 [0]	52 [0]	52 [0]	52 [0]		52	52	96 [0]	85 [2]	96 [0]	96 [0]		85	85	
3	76 [2]	77 [0]	76 [2]	76 [1]		76	76	97 [0]	97 [0]	97 [0]	97 [0]		97	97	
4	93 [0]	93 [0]	93 [0]	93 [0]		96	93	111 [4]	111 [8]	178 [0]	178 [0]		111	111	
5	115 [3]	115 [3]	115 [3]	115 [1]		121	118	130 [4]	130 [11]	129 [6]	130 [7]		134	134	
6	133 [13]	133 [5]	133 [8]	133 [0]		154	154	150 [11]	153 [10]	153 [10]	150 [13]		168	156	
7	156 [37]	162 [8]	156 [31]	159 [11]		195	195	174 [11]	174 [4]	177 [19]	187 [17]		269	211	
8	185 [17]	173 [42]	185 [34]	182 [1]		240	240	212 [6]	212 [21]	198 [18]	208 [15]		316	304	
9	205 [38]	204 [13]	211 [9]	205 [8]		-	-	240 [16]	237 [24]	234 [10]	238 [15]		-	-	
10	235 [41]	235 [6]	235 [9]	235 [10]		-	-	288 [5]	270 [13]	268 [24]	252 [9]		-	-	
11	256 [40]	260 [55]	255 [25]	257 [7]		-	-	306 [14]	290 [20]	317 [12]	309 [7]		-	-	
12	296 [0]	286 [51]	296 [0]	291 [3]		-	-	326 [13]	325 [26]	335 [10]	343 [5]		-	-	
13	312 [33]	306 [12]	320 [13]	312 [7]		-	-	354 [4]	374 [16]	380 [7]	364 [5]		-	-	
14	326 [5]	333 [3]	332 [3]	313 [4]		-	-	389 [0]	376 [11]	389 [0]	389 [0]		-	-	
15	373 [11]	362 [11]	362 [7]	377 [4]		-	-	430 [4]	430 [16]	432 [9]	430 [4]		-	-	

6 Conclusion

We proposed four strategies to generate cuts in a two-layer approach for solving Multi-Robot Deployment Problems. The generated cuts account for the interferences found in the low-level solutions, related to conflicts in resources of a shared network that have a negative impact on makespan minimization. The results obtained demonstrate the efficiency and complementary of these cuts. Even for large size problems, in which the global CP approach we developed has difficulties to produce a first solution, the cut generation strategies show a superior performance. The complementary of the cuts leads to the idea of merging them in a portfolio of cuts. This idea will be further refined in upcoming works, for which we could consider restart strategies when solutions found by the two layers cannot be improved. Last, the proposed approach can be extended to other scheduling problems involving complex setup operations between the main tasks. An example of such problems is the placement of embedded functions on a many-core processor [11], where the functions placed on the different cores interact through data exchanges over a shared network. Similarly, logistic in warehouses involves object transfers between locations and requires the utilization of shared resources whose activities must also be scheduled.

References

1. Amadini, R., Gabbrielli, M., Mauro, J.: An extensive evaluation of portfolio approaches for constraint satisfaction problems. *Int. J. Interact. Multimed. Artif. Intell.* **3**(7), 81–86 (2016)
2. Amadini, R., Gabbrielli, M., Mauro, J.: SUNNY-CP and the Minizinc challenge. *CoRR* (2017)
3. Chaimowicz, L., et al.: Deploying air-ground multi-robot teams in urban environments. In: Parker, L.E., Schneider, F.E., Schultz, A.C. (eds.) *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*, pp. 223–234. Springer, Dordrecht (2005). https://doi.org/10.1007/1-4020-3389-3_18
4. Girbal, S., Jean, X., Rhun, J.L., Pérez, D.G., Gatti, M.: Deterministic platform software for hard real-time systems using multi-core cots. In: 2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC) (2015)
5. Hooker, J., Ottosson, G.: Logic-based Benders decomposition. *Math. Program.* **96**(1), 33–60 (2013)
6. Vu Khac, K., D'Ambrosio, C., Hamadi, Y., Liberti, L.: Surrogate-based methods for black-box optimization: surrogate-based methods for black-box optimization. *Int. Trans. Oper. Res.* **24**, 393–424 (2016)
7. Koes, M., R. Nourbakhsh, I., Sycara, K.: Heterogeneous multirobot coordination with spatial and temporal constraints. In: *International Conference on Artificial Intelligence (AAAI)*, pp. 1292–1297 (2005)
8. Oddi, A., Rasconi, R., Cesta, A., Smith, S.F.: Applying iterative flattening search to the job shop scheduling problem with alternative resources and sequence dependent setup times. In: *Proceedings of the Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems*, pp. 15–22 (2011)
9. O'Mahony, E., Hebrard, E., Holland, A., Nugent, C., O'Sullivan, B.: Using case-based reasoning in an algorithm portfolio for constraint solving. In: *Irish Conference on Artificial Intelligence and Cognitive Science* (2008)
10. Pacheco, A., Pralet, C., Roussel, S.: Constraint-based scheduling with complex setup operations: an iterative two-layer approach. In: *International Joint Conference on Artificial Intelligence (IJCAI)* (2019)
11. Perret, Q., Maurère, P., Noulard, E., Pagetti, C., Sainrat, P., Triquet, B.: Mapping hard real-time applications on many-core processors. In: *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, pp. 235–244. ACM (2016)
12. Perret, Q., Maurère, P., Noulard, E., Pagetti, C., Sainrat, P., Triquet, B.: Temporal isolation of hard real-time applications on many-core processors. In: *RTAS: Real-Time Embedded Technology & Applications Symposium* (2016)
13. Ren, Q., Man, K.L., Lim, E.G., Lee, J., Kim, K.K.: Cooperation of multi robots for disaster rescue. In: *2017 International SoC Design Conference (ISOCC)*, pp. 133–134 (2017)
14. Sugiyama, H., Tsujioka, T., Murata, M.: Real-time exploration of a multi-robot rescue system in disaster areas. *Adv. Robot.* **27**, 1313–1323 (2013)
15. Tran, T.T., Vaquero, T.S., Nejat, G., Beck, J.C.: Robots in retirement homes: applying off-the-shelf planning and scheduling to a team of assistive robots. *J. Artif. Intell. Res.* **58**, 523–590 (2017)
16. Wilhelm, R., Reineke, J.: Embedded systems: many cores - many problems. In: *7th IEEE International Symposium on Industrial Embedded Systems*, pp. 176–180 (2012)