Deployment of Mobile Wireless Sensor Networks for Crisis Management: a Constraint-Based Local Search Approach

Cédric Pralet, Charles Lesire

ONERA – The French Aerospace Lab, F-31055, Toulouse, France
{firstname.lastname}@onera.fr

Abstract. In this paper, we consider a problem of management of crisis situations (incidents on nuclear or chemical plants, natural disasters...) that require remote sensing, using a set of ground and aerial robots. In this problem, sensed data must be transmitted in real-time to an operation center even in case of unavailability of traditional communication infrastructures. This implies that an ad hoc wireless communication network must be deployed, for instance using a fleet of UAVs acting as communication relays. From a technical point of view, we tackle a scheduling problem in which activities of mobile sensing robots and mobile relays must be synchronized both in time and space. Schedules produced must also be flexible and robust to the uncertainty about the duration of robot moves at execution time. The problem is modeled and solved using constraint-based local search, with some calls to graph algorithms that help defining good communication networks.

1 Problem Description

The first step in crisis management consists in performing sensing operations, to assess the situation before choosing appropriate measures. In many cases, sensing cannot be directly performed by humans, due either to the difficulty to reach some areas, *e.g.* in case of natural disasters, or to the dangerousness to do so, *e.g.* in case of incidents on nuclear or chemical plants. As a result, the use of Unmanned Ground Vehicles (UGVs) and Unmanned Aerial Vehicles (UAVs) is more and more considered to help rescue forces in such situations. These vehicles can be equipped with various sensors, allowing to measure radioactivity, to measure the concentration of a chemical element, to take pictures of damaged buildings, or to capture audio/video streams on critical areas.

Data collected may have to be transmitted to an operation center in real-time (or at least quite fast), first because the amount of memory available onboard each vehicle may be limited, and second because providing immediate feedback allows the operators to instantaneously analyze the situation. However, traditional communication infrastructures may be unavailable during crises. A solution for maintaining communication links is to deploy an ad hoc wireless communication network, using a fleet of UAVs acting as communication relays. These relays also allow operators to take the remote control of a vehicle.

In this paper, we gather all these features and consider a system composed of mobile sensors and mobile relays, respectively in charge of realizing acquisitions and transmitting acquisition data in real-time. Some vehicles may change roles dynamically during the mission, such as UAVs equipped with both a camera and a wireless router. The goal is to decide on the sequence of activities of each vehicle so that a set of requested acquisitions is performed as fast as possible, and sensors-operators communication links are maintained. The main constraint is that activities of UAVs and UGVs must be synchronized both in time and space, since communication between two vehicles is possible only at a certain distance. We must also manage cumulative resource consumptions, since each relay can transmit data coming from several sensors only if the total amount of data to be transmitted simultaneously does not exceed a given relay capacity. The problem obtained is a combinatorial scheduling problem, potentially hard to solve for human operators, and we propose to use automatic optimization tools [1].

Fig. 1 illustrates the kind of deployment strategies which we obtain, on a mission involving seven acquisitions (a1 to a7) and five UAVs (r1 to r5). Fig. 1(a) shows the trajectories of vehicles. Four specific relay positions are used (positions p1 to p4). For each relay position p, the large dotted circle around p represents the communication range of a relay placed at p. Fig. 1(b) gives the schedule, which involves two kinds of activities: acquisitions and communications. The setup durations between activities are induced by the durations of moves between locations. Acquisitions such as a4 do not require any communication relay, because they are near enough from the operation center. Acquisitions such as a1 require the simultaneous use of several relays (r2 at position p3, r4 at position p2, r5 at position p1). The communication network is dynamic because relays are mobile, such as robot r2 which moves from position p3 to position p4 as soon as it is no more needed in p3. Last, a communication relay can receive data from several robots simultaneously: see the example of robot r5, which receives data from both r3 and r4 when acquisitions a6 and a8 are performed.

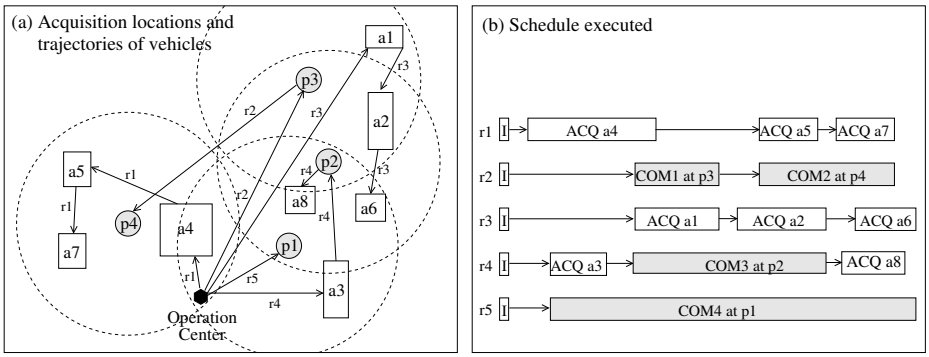


Fig. 1. Example of deployment of sensors and communication relays

The paper is organized as follows. We first describe some related work (Section 2). We then introduce a constraint-based model (Section 3). Next, a local search procedure is defined (Section 4). Last, experimental results are presented (Section 5). This work was performed during the French-German ANR ANCHORS project, whose goal is the definition of UAV-assisted ad hoc networks for crisis management and hostile environment sensing.

2 Related Work

The mobile sensor and relay deployment problem considered can be decomposed into two subproblems: (1) *exploration*, consisting in allocating acquisition activities to sensors and in ordering these acquisition activities; (2) *communication*, consisting in maintaining a communication network for transmitting acquisition data to the operation center.

The exploration subproblem can be seen as a kind of *multiple Traveling Salesman Problem* (mTSP [2]). This problem involves a set of salesmen and a set of cities, and the goal is to find minimum cost tours for the salesmen so that each city is visited exactly once. In our case, cities correspond to acquisitions and salesmen correspond to mobile sensors. In the robotics field, viewing the realization of a set of tasks by a set of robots as an mTSP is not new [3]. Constraint programming approaches for solving mTSPs also exist, with an emphasis on the flexibility of constraints for modeling additional specifications [4].

The communication subproblem is related to the literature on *wireless sensor networks* (WSNs [5, 6]). WSNs are composed of two types of nodes: sensor nodes, which collect data, and relay nodes, which transmit data. Each node is usually placed at a static position, and two nodes can communicate when the distance between them is within range. A major problem on WSNs is to place relay nodes so that in the network, there is a path between any two sensor nodes. When the number of relays must be minimized, the problem obtained is called the *Steiner Minimum Tree with Minimum Steiner Points*, which is NP-hard [7]. Another problem on WSNs is to build networks robust to relay failures. This has already been tackled in [8] using a constraint programming approach. Deployment aspects can also be considered, *e.g.* when minimizing the length of a tour which deploys new relays to repair a broken WSN [9].

The two above subproblems have been combined in the robotics field, where strategies were defined to explore an unknown environment while maintaining connectivity to a base station [10]. These strategies consist in maintaining an exploration frontier and in extending this frontier progressively, with some exploration robots taking the role of relays when the frontier to base station distance becomes too big [11]. In other contexts, static relays are deployed using one relay-deployment node, and the objective is both to place relays and minimize the length of the path for the relay-deploying robot [12, 13].

Contribution In previous approaches mixing exploration and communication maintenance, sensors and relays are deployed during successive rounds. At each

round, the planning process decides on how to extend the exploration frontier and on how to place relays. Once relays are placed, sensing and data transmission occur. When planning acquisitions for the current round, future rounds are not considered. Such a greedy deployment strategy can lead to suboptimal plans, therefore we propose to reason over a larger horizon by viewing the problem as a scheduling problem. As scheduling is one of the most successful application area of constraint programming, we explore the use of a constraint-based approach for deploying the team of cooperative vehicles. To the best of our knowledge, no prior constraint programming or scheduling approach has been proposed for solving the mixed exploration/communication problem.

Another drawback of existing exploration strategies with communication maintenance is that they synchronize actions of all mobile robots at each exploration round. This induces executions in which at each round, each vehicle waits for the placement of all relays, even if it could start its exploration task earlier, and then each vehicle waits for the end of the exploration of all other vehicles, even if it could perform another exploration task. One benefit offered by the scheduling approach we propose is that it has the capacity to avoid synchronizing activities which do not need to be synchronized. Schedules produced only require an ordering between conflicting activities, and not between all activities. This may improve reactivity during crises.

Operationally speaking, we consider that schedules are produced at the operation center, in a centralized way, before being dispatched and executed on-board each vehicle, in a distributed way. For this reason, schedules produced must also be flexible and robust to the uncertainty about the durations of robot moves. These durations may be shorter or longer than expected, especially for ground robots which may encounter unforeseen terrain conditions.

3 Modeling

Good deployment strategies must be generated quickly, in order to be reactive during the crisis. As problems considered may involve numerous acquisitions and possible positions for communication relays, we define a model in the framework of Constraint-Based Local Search [14].

3.1 Constraint-Based Local Search (CBLS)

In CBLS, models are defined by decision variables, constraints, and criteria, as in classical constraint programming. One specificity of CBLS models is the use of so-called *invariants*, which correspond to one-way constraints $x \leftarrow exp$, where x is a variable and exp is a functional expression of other variables of the problem, such as $x \leftarrow sum(i \in [1..N])\, y_i$. The set of invariants in a model must be acyclic, so that a variable is not a function of itself. Fig. 2 shows a CBLS model together with the Directed Acyclic Graph (DAG) of invariants associated with it.

In CBLS, the search space is explored more freely than with standard tree search with backtrack. When searching for a solution to a given CBLS model, all

Decision variables:
$\mathbf{var}\{bool\}\, b$
$\mathbf{var}\{int\}\, x \in [0..10]$
$\mathbf{var}\{int\}\, y \in [2..5]$

Invariants:
$\mathbf{var}\{int\}\ \ z \leftarrow ite(b, x, y)$
$\mathbf{var}\{int\}\ \ t \leftarrow (x - y)$
$\mathbf{var}\{bool\}\ \ u \leftarrow (z < t)$
$\mathbf{var}\{int\}\ \ v \leftarrow (t + 2)$

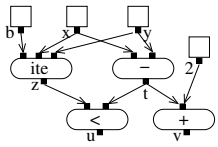


Fig. 2. Example of a CBLS model ($ite(b, x, y)$ stands for "if b then x else y")

decision variables are always assigned, *i.e.* the approach manipulates complete variable assignments. At each step during search, a local move is performed by reassigning some decision variables. Afterwards, all invariants impacted by the local move are reevaluated, following a topological order of the DAG of invariants. A specific procedure is attached to each type of invariant, so that the reevaluation is performed as fast as possible. On previous example $x \leftarrow sum(i \in [1..N])\, y_i$, in case of change of y_k for some $k \in [1..N]$, x can be incrementally reevaluated by adding to it the difference between the current and previous values of y_k, instead of recomputing the sum from scratch (reevaluation in constant time). More generally, invariants allow combinatorial constraints, temporal constraints, resource constraints, and criteria to be very quickly evaluated from a variable assignment and reevaluated from a small change in this assignment.

Several CBLS solvers were developed in the past few years, from the seminal work on Localizer [15] to solvers like COMET [14], iOpt [16], LocalSolver [17], Kangaroo [18], OscaR.cbls [19], or InCELL [20]. In this paper, we use InCELL, which offers flexibility for modeling complex scheduling problems, *e.g.* involving time-dependent scheduling aspects or continuously evolving states. In InCELL, invariants are formally defined as triples (I, O, f) with I and O sequences of variables called the input and output variables respectively, and f a function mapping assignments of I to assignments of O.

3.2 Data of the Mobile Sensor and Relay Deployment Problem

In the following, $\mathbf{R}$ denotes the number of robots involved in the mission, $\mathbf{A}$ denotes the number of acquisitions to be performed, and $\mathbf{P}$ denotes the number of 3D-positions (x, y, z) used in the modeling. $[\mathbf{Hs}, \mathbf{He}]$ denotes the scheduling horizon: every activity must start after Hs and end before He.

Each robot $r \in [1..R]$ is available from time $\mathbf{TimeIni}[r]$, and located at position $\mathbf{PosIni}[r]$ at that time. The duration required by r to move between two positions $p, p' \in [1..P]$ is given by $\mathbf{DuTrans}[r](p, p')$. This duration depends on the robot, because robots may have different motion capabilities. $DuTrans[r]$ is defined implicitly by a specific code: for a UAV, $DuTrans[r](p, p')$ can return the Euclidean distance between p and p' divided by the speed of robot r; for a UGV, $DuTrans[r](p, p')$ can be computed by a path-planning algorithm.

Each acquisition $a \in [1..A]$ can be realized following a certain number of acquisition modes $\mathbf{Nmodes}[a]$. The latter correspond to different ways of scanning the acquisition area. For instance, an acquisition between two points p and p'

can be performed from p to p' or from p' to p. With each acquisition mode m are associated positions $\mathbf{AcqPosSta}[a, m]$ and $\mathbf{AcqPosEnd}[a, m]$ at the start and end of a, and an acquisition duration $\mathbf{AcqDu}[a, m]$. Each acquisition generates a data flow with rate $\mathbf{Qos}[a]$ (quality of service requested for a, in Mb/s).

For each acquisition $a \in [1..A]$ and each robot $r \in [1..R]$, boolean data $\mathbf{AcqFeas}[a, r]$ takes value *true* iff robot r is equipped with the instrument required for realizing a. Boolean data $\mathbf{IsRelay}[r]$ takes value *true* iff robot r is equipped with a wireless router and can serve as a relay. The maximum capacity of each relay in terms of data transmission (in Mb/s) is denoted by $\mathbf{RelCap}$. We assume that $Qos[a] \leq RelCap$ holds for every acquisition a.

As for communications, we consider that relays can be placed only at predefined locations called *candidate communication nodes*. To define these nodes, one can discretize the environment into a certain number of cells and put one candidate communication node at the center of each cell. In the following, $\mathbf{N}$ denotes the number of candidate communication nodes and $\mathbf{NodePos}[n]$ denotes the position of a node n. We assume that one particular node denoted by $\mathbf{OpNode}$ is associated with the operation center. We also associate with each acquisition $a \in [1..A]$ a node $\mathbf{AcqNode}[a]$ such that a relay placed at this node is able to receive data sensed during the whole realization of a. If an acquisition is too wide to be covered by a unique node, it is always possible to split it into smaller acquisitions. Boolean function $\mathbf{Linked}(n, n')$ returns *true* iff a relay placed at node n can communicate with a relay placed at node n'. For communication between UAVs, this function checks whether the distance between the two nodes is not greater than the communication range. Last, the length of each communication path from a sensor node to the base station (number of relays on this path) must not be greater than a given limit, denoted by $\mathbf{NhopsMax}$. We assume that for every acquisition a considered individually, a valid communication path from $AcqNode[a]$ to $OpNode$ can be built.

3.3 Decision Variables

Decision variables are given in Eq. 1 to 6. Similarly to IBM ILOG CpOptimizer or to the CAIP framework [21], InCELL represents activities based on the notion of *interval*. An interval itv is defined by a boolean presence variable $\mathbf{pres}(itv)$, indicating whether the activity is present, and two time-points denoted by $\mathbf{start}(itv)$ and $\mathbf{end}(itv)$, representing respectively the start and the end of the activity.

Eq. 1 defines one acquisition interval $acqItv[a]$ per acquisition $a \in [1..A]$, and decision variable $acqMode[a]$ introduced in Eq. 2 represents the realization mode chosen for a. Next, intervals $comItv[k]$ are introduced in Eq. 3 for representing communication activities. Performing a communication activity consists in placing a robot at one of the N candidate communication nodes and in using this robot as a data transmission relay. The communication node in which the kth communication interval is placed corresponds to decision variable $comNode[k]$ given in Eq. 4. Through this choice of variables, we impose that during a single communication activity, the relay robot used must stay at the chosen node. As

they are at most A acquisitions and as each communication path can contain at most *NhopsMax* relays, we bound the number of possible communication activities by $\mathbf{K} = A \cdot NhopsMax$. The schedule provided in Fig. 1(b) involves eight acquisition activities (ACQ(a1) to ACQ(a8)) and four communication activities (activities "COM1 at p3" and "COM2 at p4" for robot r2, activity "COM3 at p2" for robot r4, and activity "COM4 at p1" for robot r5).

In order to synchronize acquisition and communication intervals, we introduce, for each acquisition $a \in [1..A]$ and for each communication interval index $k \in [1..K]$, one integer decision variable $useCom[a, k]$ representing the transmission rate (in Mb/s) reserved by a in communication interval k (Eq. 5).

The last set of decision variables (Eq. 6) represents the choice in the sequences of activities *activitySeqs* performed by robots. We use here a type of InCELL called $DisjointIntSequences(M, T)$. A variable of this type allows to compactly represent M sequences $s_m = [i_{m,1}, \ldots, i_{m,k_m}]$ composed of integers belonging to $[1..T]$, and such that any integer appears at most once over all sequences. By viewing integers as task indices and sequences as machines, a variable of type $DisjointIntSequences(M, T)$ represents, for each of the M machines, the sequence of indices of tasks which are successively performed on this machine. For the mobile sensor and relay deployment problem, we need to consider R machines (one per robot) and $A{+}K$ tasks (one task per acquisition and communication interval), hence we use a variable of type $DisjointIntSequences(R, A{+}K)$. An integer $i \in [1..A]$ corresponds to the ith acquisition, and an integer $i \in [A{+}1..A{+}K]$ corresponds to the $(i - A)$-th communication interval. The ordering of activities in Fig. 1 would be represented by the five sequences of integers $r_1 = [4, 5, 7]$, $r_2 = [9, 10]$, $r_3 = [1, 2, 6]$, $r_4 = [3, 11, 8]$, $r_5 = [12]$. Variables of type $DisjointIntSequences(M, T)$ support several local moves, including the insertion of an integer $i \in [1..T]$ in the sequence of a machine $m \in [1..M]$, or the removal of an integer $i \in [1..T]$ from sequences. Internally, as shown in Eq. 6, they are implemented using three variables per integer $i \in [1..T]$: two integer variables $prev[i]$ and $next[i]$ representing the integer preceding and following integer i in some sequence, and one integer variable $seq[i]$ representing the sequence in which integer i appears, with value 0 when i does not appear in any sequence. Specific indices and values not detailed here are also added to represent the start and end of each sequence.

$$\forall a \in [1..A],\ \mathbf{Interval}\ acqItv[a] \in [Hs, He] \quad \texttt{//acquisition intervals} \tag{1}$$

$$\forall a \in [1..A],\ \mathbf{var}\{int\}\ acqMode[a] \in [1..Nmodes[a]] \quad \texttt{//acquisition modes} \tag{2}$$

$$\forall k \in [1..K],\ \mathbf{Interval}\ comItv[k] \in [Hs, He] \quad \texttt{//communication intervals} \tag{3}$$

$$\forall k \in [1..K],\ \mathbf{var}\{int\}\ comNode[k] \in [1..N] \quad \texttt{//communication node} \tag{4}$$

$$\forall a \in [1..A], \forall k \in [1..K],\ \mathbf{var}\{int\}\ useCom[a, k] \in [0, RelCap] \quad \texttt{//relay use} \tag{5}$$

$$\mathbf{DisjointIntSequences}(R, A{+}K)\ activitySeqs \quad \texttt{//sequences of activities} \tag{6}$$

$$\begin{bmatrix} \forall i \in [1..A{+}K{+}R],\ \mathbf{var}\{int\}\ prev[i] \in [1..A{+}K{+}R] \\ \forall i \in [1..A{+}K{+}R],\ \mathbf{var}\{int\}\ next[i] \in [1..A{+}K{+}R] \\ \forall i \in [1..A{+}K],\ \mathbf{var}\{int\}\ seq[i] \in [0..R] \end{bmatrix}$$

3.4 Invariants, Constraints, and Criterion

Invariants and constraints, as well as the criterion, are defined in Eq. 7 to 20.

Constraint 7 expresses that every acquisition a must be performed and assigned to a robot. Constraint 8, expresses that a communication interval is present iff it appears in one of the sequences of activities. Constraint 9 enforces that each acquisition is performed by an appropriate robot (we assume that $AcqFeas[a, 0] = true$). Similarly, Constraint 10 enforces that the sequence which contains the kth communication interval must correspond to a robot capable of being a relay (we assume that $IsRelay[0] = true$). Constraint 11 expresses that relay capacities can only be reserved on communication intervals which are present. Invariants given in Eq. 12 to 14 define the start and end positions of each acquisition, function of the acquisition mode, as well as the position of each communication interval, function of the communication node chosen.

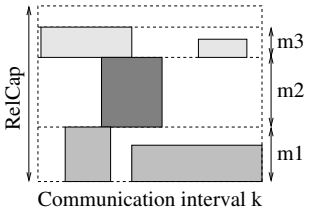
Next, Eq. 15 to 17 specify the temporal constraints of the model. These constraints are all simple precedence constraints between start/end time-points of activities. Constraint 15 expresses that the duration of an acquisition interval must be equal to the duration associated with the chosen realization mode. Constraint 16 expresses that if acquisition a uses communication interval k for transmitting data (boolean condition $useCom[a, k] > 0$), then acquisition interval $acqItv[a]$ must be included in communication interval $comItv[k]$. Constraint 17 imposes that for each robot r, there is no overlap between activities assigned to r when these activities are sequenced as specified in $activitySeqs$. Constraint $\mathbf{noOverlap}(TimeIni, PosIni, DuTrans, Itvs, PosSta, PosEnd, activitySeqs)$ used in Eq. 17 is a generic temporal constraint of InCELL. It has seven inputs: (1) a table $TimeIni$ defining the initial availability time of each machine usable for realizing tasks, (2) a table $PosIni$ defining the configuration of each machine at that time, (3) a table $DuTrans$ of functions giving the setup time required from one configuration to another, (4) a table of intervals $Itvs$ which may be placed on machines, (5) a table $PosSta$ such that $PosSta[i]$ defines the configuration required at the start of interval $Itvs[i]$, (6) a table $PosEnd$ such that $PosEnd[i]$ gives the configuration obtained at the end of interval $Itvs[i]$, (7) an element $activitySeqs$, of type $DisjointIntSequences$, which defines the successive indices of intervals to be realized on each machine. In InCELL, the $noOverlap$ constraint is implemented using several invariants, and formally it ensures that:

- for an activity j placed just after activity i on machine r,
  $\mathbf{start}(Itvs[j]) \geq \mathbf{end}(Itvs[i]) + DuTrans[r](PosEnd[i], PosSta[j])$;
- for the first activity j on a machine r,
  $\mathbf{start}(Itvs[j]) \geq TimeIni[r] + DuTrans[r](PosIni[r], PosSta[j])$.

Constraint 18 enforces that for every acquisition a, there must exist a communication path, *i.e.* a sequence of communication nodes $[n_1, \ldots, n_l]$, such that: (1) n_1 is node $AcqNode[a]$ associated with a, (2) n_l is node $OpNode$ associated with the operation center, (3) the length l of the path is not greater than $NhopsMax$, (4) every two successive nodes are such that $Linked(n_i, n_{i+1})$ holds. Also, it imposes that the total capacity reserved by a on a node n, defined

by $\sum_{k\in[1..K]\,|\,comNode[k]=n} useCom[a,k]$, is equal to $Qos[a]$ if n belongs to the path, and to 0 otherwise. The constraint takes as an input the nodes in which communication intervals are placed as well as all capacities reserved by a on communication intervals. It is expressed in a rather global form because it is handled using specific graph algorithms (see Sect. 4.3). Note that we do not forbid several relays to be placed side by side at the same node. More generally, we consider that possible conflicts on trajectories of robots are handled at execution time, using online collision avoidance techniques.

Constraint 19 imposes that for every communication interval k, the sum of the maximum resource usages of robots on k does not exceed the relay capacity. For example, in the figure on the right, if m_1, m_2, m_3 are the maximum resource usages in interval k by robots r_1, r_2, r_3 resp., then $m_1{+}m_2{+}m_3 \leq RelCap$ must hold. This guarantees that relay capacity is not exceeded whatever the real activity dates are at execution time. The approach may be suboptimal, but it is robust and does not require any synchronization between users of a communication interval.



The criterion given in Eq. 20 corresponds to the makespan, defined as the earliest end time of the last acquisition in the schedule.

$$\forall a \in [1..A], \mathbf{pres}(acqItv[a]) \wedge (seq[a] \neq 0) \tag{7}$$

$$\forall k \in [1..K], \mathbf{pres}(comItv[k]) \leftrightarrow (seq[k+A] \neq 0) \tag{8}$$

$$\forall a \in [1..A], AcqFeas[a, seq[a]] \tag{9}$$

$$\forall k \in [1..K], IsRelay[seq[k+A]] \tag{10}$$

$$\forall a \in [1..A], \forall k \in [1..K],\ (useCom[a,k] > 0) \rightarrow \mathbf{pres}(comItv[k]) \tag{11}$$

$$\forall a \in [1..A],\ \mathbf{var}\{int\}\ spos[a] \leftarrow AcqPosSta[a, acqMode[a]] \tag{12}$$

$$\forall a \in [1..A],\ \mathbf{var}\{int\}\ epos[a] \leftarrow AcqPosEnd[a, acqMode[a]] \tag{13}$$

$$\forall k \in [1..K],\ \mathbf{var}\{int\}\ comPos[k] \leftarrow NodePos[comNode[k]] \tag{14}$$

$$\forall a \in [1..A],\ \mathbf{durationEq}(acqItv[a], AcqDu[a, acqMode[a]]) \tag{15}$$

$$\forall a \in [1..A], \forall k \in [1..K],\ \mathbf{during}(useCom[a,k] > 0, acqItv[a], comItv[k]) \tag{16}$$

$$\mathbf{noOverlap}(TimeIni, PosIni, DuTrans, Itvs, PosSta, PosEnd, activitySeqs) \tag{17}$$

$$\begin{aligned} with:\ & Itvs = (\mathbf{all}(a \in [1..A])\ acqItv[a]) \cdot (\mathbf{all}(k \in [1..K])comItv[k]) \\ & PosSta = (\mathbf{all}(a \in [1..A])\ spos[a]) \cdot (\mathbf{all}(k \in [1..K])comPos[k]) \\ & PosEnd = (\mathbf{all}(a \in [1..A])\ epos[a]) \cdot (\mathbf{all}(k \in [1..K])comPos[k]) \end{aligned}$$

$$\forall a \in [1..A],\ \mathbf{ComPathConstraint}(AcqNode[a], OpNode, Linked, NhopsMax, Qos[a], comNode, \mathbf{all}(k \in [1..K])useCom[a,k]) \tag{18}$$

$$\forall k \in [1..K],\ (\sum_{r\in[1..R]} \max_{a\in[1..A]\,|\,seq[a]=r} useCom[a,k]) \leq RelCap \tag{19}$$

$$\mathbf{minimize}\ \max_{a\in[1..A]} \mathbf{earliestTime}(\mathbf{end}(acqItv[a])) \tag{20}$$

4 Local Search Algorithm

We now describe a local search procedure which produces schedules satisfying all constraints of the model, as the schedule given in Fig. 1.

A possible strategy could be to solve first the exploration problem and then the communication maintenance problem. Solving the exploration problem would consist in choosing sequences of acquisition activities, while solving the communication maintenance problem would consist in adding communication relay activities in the sequences found at the first step. The drawback of such a decomposition approach is that synchronization constraints between acquisition and communication activities are taken into account too late, and poor quality schedules may be produced. See Fig. 3 for an example.

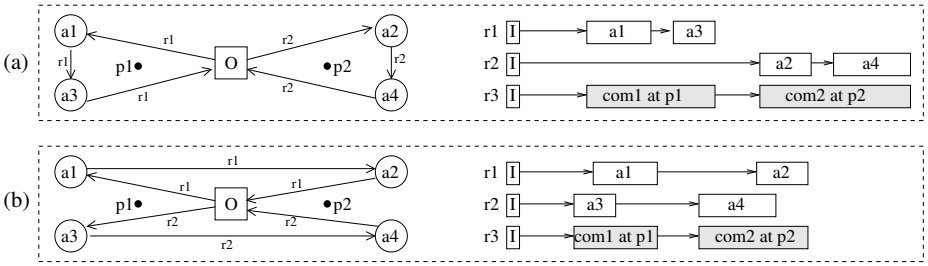


Fig. 3. An example involving two robots r1, r2 capable of making acquisitions and one relay robot r3: (a) schedule obtained by first computing optimal exploration tours, and then adding communication relay activities; (b) a better schedule, which uses longer exploration tours but synchronizes the accesses to relay r3 by robots r1 and r2

The strategy we propose uses two phases: (1) *a constructive phase*, which produces an initial schedule containing all acquisitions; this phase iteratively adds activities at the end of the robot schedules, using a greedy randomized heuristics; (2) *a local search phase*, during which the makespan of the schedule found at the previous step is improved; unlike the constructive phase, the local search phase can modify the schedule in a non chronological way. The two phases are iterated: when the local search phase does not create any new improvement, a restart from an empty schedule occurs, as in the GRASP metaheuristics [22].

4.1 Constructive Phase

The constructive phase starts from an empty schedule. While there exist acquisitions not scheduled yet, we select a pair (r, a) composed of a robot $r \in [1..R]$ and an acquisition $a \in [1..A]$. Robot r is selected randomly among robots which are capable of realizing an acquisition not performed yet, with a probability function of the current end time of the schedule of r (earliest idle robot heuristics).

Acquisition a is an acquisition which is (1) feasible by r, (2) not performed yet, and (3) as near as possible from the position of r at the end of its current schedule (nearest neighbor heuristics). These selection operations are implemented with the help of *set invariants*, such as $candidateRobots \leftarrow \{r \in [1..R] \,|\, \exists a \in [1..A]\, \neg\mathbf{pres}(acqItv[a]) \wedge AcqFeas[a, r]\}$. The latter CBLS invariant efficiently maintains the set of robots which are candidates for selection.

Acquisition a is then inserted at the end of the schedule of robot r. To do this, interval $acqItv[a]$ is marked as present, and integer a is added at the end of the rth sequence in $activitySeqs$. Next, the communication network allowing a to be covered is built following the graph-based procedure described in Section 4.3. This procedure computes a communication path from a to the operation center, and adds a set of communication intervals $comItv[k]$ at the end of plans of some robots $r' \neq r$. The procedure also chooses the resource usage of a in each communication interval (decision variables $useCom[a, k]$).

The schedule obtained after the constructive phase satisfies all constraints and contains all acquisitions (provided that the horizon end He is large enough).

4.2 Local Search Phase

To improve the schedule generated by the constructive phase, we use local moves that try to relocate acquisitions belonging to the critical path, that is to the list of successive activities justifying the value of the makespan.

The main issue is to avoid considering local moves which create cycles in the temporal precedence graph, and which are therefore trivially inconsistent. To solve this issue, we maintain an ordered list containing all acquisitions. This ordered list is denoted by **NetAccess**, and the meaning of this list is that if an acquisition a_1 appears before an acquisition a_2 in **NetAccess**, then it is guaranteed that no communication interval reserved for a_2 on a relay r is placed strictly before a communication interval reserved for a_1 on r.

Then, each step of the local search works as follows:

1. we randomly select an acquisition a on the critical path; a is removed from the schedule by removing interval $acqItv[a]$, as well as all capacity usages reserved by a; if a was the only user of a communication interval $comItv[k]$, the latter is also removed from the schedule;
2. we then choose a permutation of robots capable of realizing a; as long as a better insertion position has not been found for a, we select the next robot r in the permutation and go to point 3 below; if all robots of the permutation have already been considered, the schedule before the local move is restored and a is marked as currently not relocatable;
3. we try to perform a local move of addition of a into the schedule of r, for each position in the **NetAccess** order; assume that **NetAccess** corresponds to list $[a_1, \ldots, a_n]$; the insertion of a in the schedule of r, between a_i and a_{i+1} in **NetAccess**, is tested as follows; first, we determine, for each robot r', the ongoing activity $lastItv[r']$ on r' after the realization of $[a_1, \ldots a_i]$; if r' is free after the realization of $[a_1, \ldots a_i]$, new activities can be added to the plan

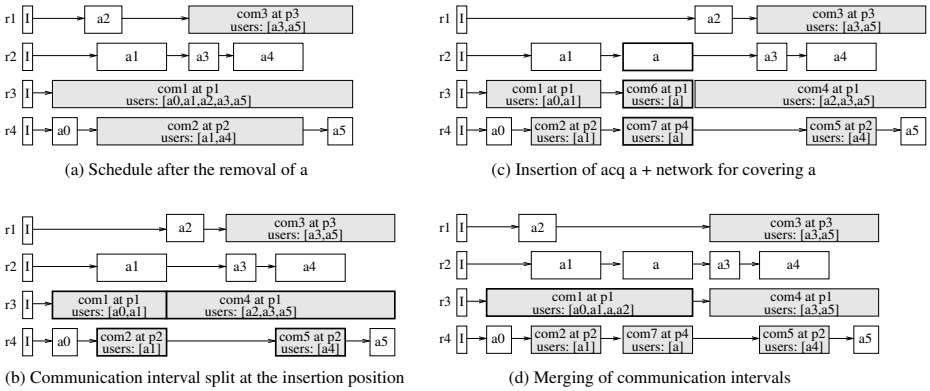


Fig. 4. Example of a local move: addition of acquisition a in the schedule of robot $r2$, between $a1$ and $a2$ in the **NetAccess** order given by $[a1, a2, a3, a4, a5]$

of r' just after $lastItv[r']$ without creating precedence cycles; the only case in which r' may not be free is when activity $lastItv[r']$ is a communication interval k, and this communication interval is also used by an acquisition in $[a_{i+1}, \ldots, a_n]$; in this case, inserting new activities for r' between a_i and a_{i+1} may create cycles; to avoid this, we create a new communication interval k' just after k in the schedule of r', and all acquisitions in $[a_{i+1}, \ldots, a_n]$ that use capacity on k are redirected to k'; an example is given in Fig. 4(b), which explicitly mentions the list of acquisitions which use a given communication interval; in this figure, the insertion of acquisition a between a_1 and a_2 induces a split of communication intervals com1 and com2 compared to the schedule given in Fig. 4(a); the two new intervals created, com4 and com5, contain all users of com1 and com2 placed after a_2 in **NetAccess**; note that splitting communication intervals may postpone acquisition tasks;

4. acquisition a is added to the schedule of r at the chosen position; from the state of the communication network after the realization of $[a_1, \ldots a_i]$, a communication network is built for covering a, using the procedure described in Sect. 4.3; communication activities corresponding to this new network are also added to the schedules, as in Fig. 4(c);
5. to reduce the makespan, we try to merge communication intervals that were split at step 3 for avoiding the creation of cycles, or that have become contiguous following the removal of a at step 1; merging two intervals k, k' means transferring from k' to k as many relay capacity usages as possible; transfers are performed following the **NetAccess** order and they stop as soon as one transfer fails; for instance, in Fig. 4(d), intervals com1 and com6 are merged, and capacity usage of a_2 over com4 is transferred to com1;
6. insertions are tested at all positions in the **NetAccess** order, with the best possible acquisition mode; the best insertion on robot r is kept provided that it improves the makespan; ties are broken by keeping the option that

minimizes the sum of the durations of the robot schedules, so as to occupy resources as least as possible; if the relocation of a succeeds, all acquisitions marked as non-relocatable are marked as relocatable again.

Local search ends when all acquisitions of the critical path are marked as not relocatable. A restart is performed if there is still some computing time left.

4.3 Building Communication Paths

To build a communication path for covering an acquisition a, we first build a sequence of connected communication nodes starting at $AcqNode[a]$ and ending at $OpNode$. This point is tackled using EQAR [23], an algorithm capable of building wireless sensor networks with quality of service requirements, when sensors produce data with a certain rate and when relays have a limited capacity. In short, EQAR is an iterative algorithm which considers each sensor in turn. For covering a sensor, it adds a set of relays at some nodes of the communication grid. At each step of the algorithm, each node n of the grid may already contain some relays and have a so-called *residual capacity* $resCap[n]$, which corresponds to the amount of Mb/s that are still available on relays placed at n. EQAR then computes a good communication path by solving a shortest-path problem (using Dijkstra's algorithm) in the graph where there are arcs between any two connected communication nodes, and each arc pointing to node n is weighted by 0 if the residual capacity $resCap[n]$ of n is greater than the quality of service qos required for the sensor (traversing n is free in this case), and by $1-resCap[n]/qos$ otherwise. An illustration of EQAR is given in Fig. 5.

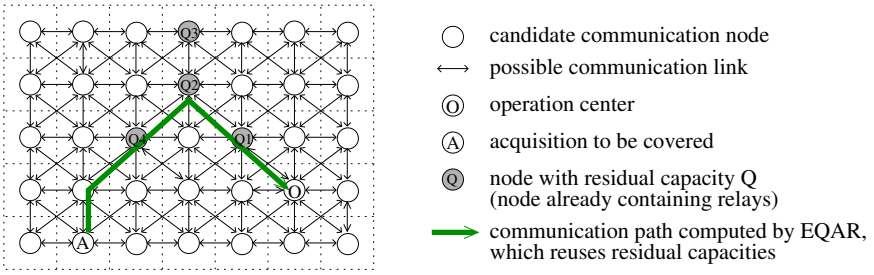


Fig. 5. Illustration of the EQAR algorithm [23]

Using EQAR for covering an acquisition a is quite straightforward. When a must be added to the schedule of robot r, the ongoing activity on each robot $r' \neq r$ is first determined. If the ongoing activity on r' is a communication activity in node n, then we compute the residual capacity offered by r' in n. Using all residual capacities of all robots, except for robot r, we compute a good communication path following the EQAR procedure. This communication path

may require the use of new relays at some communication nodes. Deciding on which robot to send to which node in order to build the communication path as fast as possible can be seen as a *Linear Bottleneck Assignment Problem* [24]. Polynomial algorithms do exist for such problems, but we use here a simple greedy procedure which successively sends to each node n a relay robot r able to reach n as fast as possible. An associated communication interval k is added to the schedule of r, and the capacity consumed by acquisition a on k is chosen as high as possible, in order to use the minimum number of relays for covering $Qos[a]$. The communication intervals introduced are merged with previous communication intervals when possible. These steps guarantee the satisfaction of all constraints given in Eq. 18.

5 Experiments

We consider here a 1km×1km crisis area. A communication grid containing 100 cells of size 100m×100m is built, and candidate communication nodes are placed at the center of these cells. The range of a communication relay is 150m, hence each cell can communicate with its eight neighbors. The operation center is placed at a corner of the environment. Two third of the robots can relay communications, the capacity of a relay is 45Mb/s, and we do not limit the length of communication paths. Acquisitions are performed using two types of instruments. Instruments of the first (resp. second) type generate 3Mb (resp. 6Mb) of data per second. Each robot has zero, one or two instrument(s).

As we do not dispose of real data, we build instances by defining sets of acquisition strips of a certain length. These strips are specified by end points chosen randomly, and each strip is split into acquisitions which can be covered by a unique communication node. Table 1 reports some statistics on such instances, concerning the CBLS model and the search phases. Results are obtained on an Intel i5-520 1.2GHz, 4GBRAM. They show that the approach scales quite well when the number of tasks involved in schedules increases (column *nItvs*).

Fig. 6 details results for two specific scenarios involving 14 UAVs. The first one involves 20 acquisitions generated as previously described. The second one

A	R	*nVars*	*nInvariants*	*nItvs*	*tCreateModel* (sec.)	*tGreedySol* (sec.)	*nMovesPerSec*
20	14	69550	12343	200	0.26	0.002	8295
50	14	238778	30433	500	0.45	0.03	1626
100	14	700827	60583	1000	1.09	0.093	637
20	30	161178	33465	357	0.49	0.008	1532
50	30	632050	97287	1050	1.21	0.06	573
100	30	1760500	193987	2100	2.21	0.197	351

Table 1. Statistics on the CBLS approach; column *nMovesPerSec* counts one local move each time an acquisition is added to a robot using a particular position in the **NetAccess** order (one local move corresponds to the set of operations given in Fig. 4)

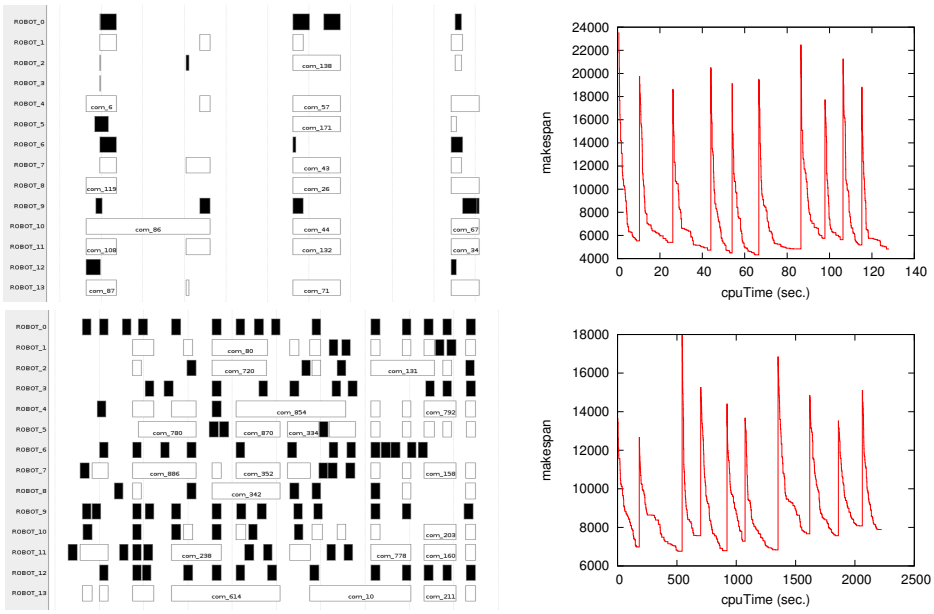


Fig. 6. Schedules produced by the local search and evolutions of the makespan; upper part: scenario involving 20 acquisitions; lower part: scenario involving 100 acquisitions; acquisitions (resp. communications) are depicted in black (resp. white)

involves 100 acquisitions positioned regularly at the center of the 100 cells of the communication grid, which simulates an exploration of the whole environment. Fig. 6 shows the best schedules generated after 50 restarts, as well as the evolution of the makespan during the first 10 restarts. It appears that restarts help in escaping local minima, and that local search quickly improves the value of the makespan given by the greedy constructive phase.

6 Conclusion

This paper introduced a CBLS approach for deploying mobile wireless sensor networks. This approach computes efficient schedules which remain executable despite the uncertainty about the duration of robot moves. In terms of modeling, additional resource constraints such as energy limitations could be taken into account, and we could build a CP model in which relays can move during communications. The latter point requires to interleave more finely scheduling with the planning of robot paths, which raises new modeling issues. The next step will be to tackle real scenarios and perform real demonstrations. On this point, we are currently developing a supervision layer able to manage plan execution and to request plan repairs/optimizations when robot moves are longer/shorter than expected, or when operators request new acquisitions during the mission.

References

1. Hentenryck, P.V.: Computational disaster management. In: Proc. of IJCAI-13. (2013) 12–18
2. Bektas, T.: The multiple traveling salesman problem: an overview of formulations and solution procedures. OMEGA: The International Journal of Management Science **34**(3) (2006) 209–219
3. Sariel-Talay, S., Balch, T., Erdogan, N.: Multiple traveling robot problem: A solution based on dynamic task selection and robust execution. IEEE/ASME Transactions on Mechatronics, Special Issue on Mechatronics in Multirobot Systems **14**(2) (2009) 198–206
4. Pesant, G., Gendreau, M., Potvin, J.Y., Rousseau, J.M.: On the flexibility of constraint programming models: From single to multiple time windows for the traveling salesman problem. European Journal of Operational Research **117** (1999) 253–263
5. Akyildiz, I., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks **38** (2002) 393–422
6. Younis, M., Akkaya, K.: Strategies and techniques for node placement in wireless sensor networks: A survey. Ad Hoc Networks **6**(4) (2008) 621–655
7. Hwang, F., Richards, D., Winter, P.: The Steiner tree problem. Volume 53 of Annals of Discrete Mathematics. Elsevier (1992)
8. Quesada, L., Brown, K., O'Sullivan, B., Sitanayah, L., Sreenan, C.: A constraint programming approach to the additional relay placement problem in wireless sensor networks. In: Proc. of ICTAI-13. (2013) 1052 – 1059
9. Thuy T. Truong, K.N.B., Sreenan, C.J.: Repairing wireless sensor network connectivity with mobility and hop-count constraints. In: Proc. of ADHOC-NOW-13. (2013) 75–86
10. Pal, A., Tiwari, R., Shukla, A.: Communication constraints multi-agent territory exploration task. Applied Intelligence **38**(3) (2013) 357–383
11. Mukhija, P., Krishna, K.M., Krishna, V.: A two phase recursive tree propagation based multi-robotic exploration framework with fixed base station constraint. In: Proc. of IROS-10. (2010)
12. Pei, Y., Mutka, M.W.: Steiner traveler: Relay deployment for remote sensing in heterogeneous multi-robot exploration. In: Proc. of ICRA-12. (2012) 1551–1556
13. Pei, Y., Mutka, M.W., Xi, N.: Connectivity and bandwidth-aware real-time exploration in mobile robot. Wireless Communications and Mobile Computing **13**(9) (2013) 847–863
14. Hentenryck, P.V., Michel, L.: Constraint-based local search. The MIT Press (2005)
15. Michel, L., Hentenryck, P.V.: Localizer. Constraints **5**(1-2) (2000) 43–84
16. Voudouris, C., Dorne, R., Lesaint, D., Liret, A.: iOpt: A software toolkit for heuristic search methods. In: Proc. of CP-01. (2001) 716–719
17. Benoist, T., Estellon, B., Gardi, F., Megel, R., Nouioua, K.: Localsolver 1.x: a black-box local-search solver for 0-1 programming. 4OR: A Quarterly Journal of Operations Research **9**(3) (2011) 299–316
18. Newton, M.H., Pham, D., Sattar, A., Maher, M.: Kangaroo: An efficient constraint-based local search system using lazy propagation. In: Proc. of CP-11. (2011) 645–659
19. Landtsheer, R.D.: OscaR.cbls: a Constraint-Based Local Search Engine. (2012)
20. Pralet, C., Verfaillie, G.: Dynamic online planning and scheduling using a static invariant-based evaluation model. In: Proc. of ICAPS-13. (2013)

21. Frank, J., Jónsson, A.: Constraint-based attribute and interval planning. Constraints **8**(4) (2003) 339–364
22. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization **6** (1995) 109–133
23. Lee, S., Younis, M.F.: EQAR: Effective QoS-aware relay node placement algorithm for connecting disjoint wireless sensor subnetworks. IEEE Transactions on Computers **60**(12) (2011) 1772–1787
24. Burkard, R.E., Dell'Amico, M., Martello, S.: Assignment Problems. SIAM (2009)