A column generation based label correcting approach for the sensor management in an information collection process

Duc Manh Nguyen *, Frédéric Dambreville, Abdelmalek Toumi, Jean-Christophe Cexus, and Ali Khenchaf

Lab-STICC UMR CNRS 6285, ENSTA Bretagne
2 rue François Verny, 29806 Brest Cedex 9, France
{nguyendu,dambrefr,toumiab,cexusje,khenchal}@ensta-bretagne.fr

Abstract. This paper deals with problems of sensor management in a human driven information collection process. This applicative context results in complex sensor-to-task assignment problems, which encompass several difficulties. First of all, the tasks take the form of several information requirements, which are linked together by logical connections and priority rankings. Second, the assignment problem is correlated by many constraint paradigms. Our problem is a variant of Vehicle Routing Problem with Time Windows (VRPTW), and it also implements resource constraints including refuelling issues. For solving this problem, we propose a column generation approach, where the label correcting method is used to treat the sub-problem. The efficiency of our approach is evaluated by comparing with solution given by CPLEX on different scenarios.

Keywords: Sensor management, Information collection, Vehicle Routing Problem, Column generation, Mixed integer linear programming.

1 Introduction

Sensor planning is a research domain that treats the problem of how to manage or coordinate the usage of a suite of sensors or measurement devices in a dynamic, uncertain environment, to improve the performance of data fusion and ultimately that of perception [24]. It is also beneficial to avoid overwhelming storage and computational requirements in a sensor and data rich environment by controlling the data gathering process such that only the truly necessary data are collected and stored [18]. The literature on sensor planning closely followed the appearance of the first significant sensor capacity, and its history tracks back to the seminal works of Koopman during World War II [13, 23]. Nowadays, because sensors are becoming more complex with the advances in sensor technology and also due to the perplexing nature of the environment to be sensed, sensor planning has evolved out of the need for some form of assigning and scheduling tasks to the sensors [16].

* Corresponding author. Email: nguyendu@ensta-bretagne.fr

Sensor planning has been studied extensively and is becoming increasingly important due to its practical implementations and applications. Besides several military applications, sensor planning currently deals with the general domain of search and surveillance [9, 10], and also is one of the key points to optimize the performance of a sensor network [4, 12]. In sensor planning, the global issue is to optimize an implementation of sensors in order to maximize the positive effect of subsequent data processing in regards to mission objectives. Therefore, we have to deal with both the optimization of implementation of sensors and the information processing (typically data fusion). From this point of view, sensor planning is also related to difficult topics in robotic - *e.g.* Partially Observable Markov Decision Process [22, 5].

In this paper, we will consider the planing of sensors, which are monitored by human teams. This problem is reduced to a generalization of Vehicle Routing Problem with Time Windows (VRPTW). Therefore, it is a NP-complete problem. For solving this problem, we introduce an approach based on the column generation method [6, 7], which is one of the most famous methods in the literature for solving VRPTW. In order to successfully apply the column generation method, we propose an suitable integer programming formulation of this problem, and then develop a label correcting method [8] for treating the sub-problem. The numerical results will show the efficiency of our approach.

The rest of paper is organized as follows. In Section 2, we introduce the considered sensor planing problem and its formulation. Our column generation based label correcting approach for solving this problem is presented in Section 3. Numerical experiments are reported in Section 4 while some conclusions and perspectives are discussed in Section 5.

2 Problem formulation

When sensors are planned by human teams, the planning process is typically divided into two stages: the first is purely human driven, and results in the definition of an assignment problem with time and travel constraints; the second is based on optimization processes and results from the formalization of the first step. In such case, the human interaction with the optimization process is fundamental. Therefore, the human operators should be highly skilled in their domain, and may provide useful information to the optimization processes. Moreover, the human operators need to know, to understand and to interact with the optimization processes. These requirements quite often lead to the intricate sensor planning problems, for instance, the sensor-to-task assignment problems [14, 19], or the variants of the vehicle routing problem with time constraint satisfaction [11], etc.

In this work, we are interested by the second step of the planning. Our problem is to design the trajectories for a set of sensors in order to perform a set of missions with maximum performance. Besides taking into account several constraints (trajectory constraints, time windows constraints) like those in the Vehicle Routing Problem with Time Windows (VRPTW), we have to deal with both refuelling steps and a plan evaluation doctrine. Our problem could be

considered as a generalization of the VRPTW.
This sensor planning problem is characterized by the following objects:

Formal Information Requirements: we have a set F of formal information requirements (FIR) needed to be satisfied. For each requirement $u \in F$, we have a set of missions corresponding $\mu(u)$ to satisfy this requirement. Here, we suppose that $\mu(u) \cap \mu(v) = \emptyset$ if $u \neq v$, and denote $M = \bigcup_{u \in F} \mu(u)$ the set of all missions for all requirements.
Sensors: A sensor is a resource unit which may be used for some FIR acquisition. We denote K the set of all sensors.
Starting points: A starting point is a possible state from which a sensor have to start. S is the set of all starting points.
Refuelling centres: A refuelling centre is a possible state where a sensor will reset its autonomy levels. R is the set of all refuelling centres.
Arrival points: An arrival point is a possible state where a sensor have to end. E is the set of all arrival points (endpoints).
Sensor states: In our model, starting points, refuelling centres and ending points could be considered as particular cases of missions, and represent a possible *state* of the sensor. For this reason, we denote $N = S \cup M \cup R \cup E$ the set of all possible states (also called tasks, or points).
Some states may be incompatible with some sensors. Thus, for each state $i \in N$, we define $K(i) \subset K$ the set of all sensors being compatible with state i. Also, the following definitions will be useful:

$S(k) \subset S$ is the set of all starting points for sensor $k \in K$;
$E(k) \subset E$ is the set of all arrival points for sensor $k \in K$.

Variables for trajectories and affectations: The boolean variables x and y are used for modelling edges and vertices of the sensors trajectories.

$$y_{ik} = \begin{cases} 1 & \text{if sensor } k \text{ performs task } i, \\ 0 & \text{otherwise,} \end{cases}$$

$$x_{ijk} = \begin{cases} 1 & \text{if sensor } k \text{ performs task } j \text{ after task } i, \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, the following instrumental variable will be used in order to prevent any cyclic trajectory:

$\omega_{ik} \in \mathbb{R}$ is a counting variable for the passed states of the trajectories.

Constant parameters for performances and costs: Performances are evaluated by means of the degrees of importance of the FIR and by means of precomputed evaluations of the efficiency of any sensor in performing a mission:

- p_u is the weight of requirement $u \in F$ with respect to its priority;
- g_{ik} is the efficiency of sensor $k \in K$ in performing mission $i \in M$;
- c_{ijk} evaluates the resources expended by sensor $k \in K$ while performing state $j \in N$ after state $i \in N$;
- d_{ijk} evaluates the distance travelled by sensor $k \in K$ while performing state $j \in N$ after state $i \in N$.

The following corrected cost is defined by weighting the actual cost and the distance:

- $\widetilde{c}_{ijk} = \epsilon_1(c_{ijk} + \epsilon_2 d_{ijk})$ is the corrected cost for $i, j \in N$ and $k \in K$.

Here $\epsilon_1, \epsilon_2 \in \mathbb{R}_+$ are small positive numbers.

Variables and constant parameters for resources: Depending on the nature of the state (*e.g.* is it a refuelling centre or not?), the resources of each sensor may be replenished or not after each state. We consider the following variables:

- α_{ik} is the level of autonomy of sensor $k \in K$ after performing state $i \in N$ and before a possible refuelling;
- β_{ik} is the level of autonomy of sensor $k \in K$ after a possible refuelling at state $i \in N$.

By the way, the levels of supply, after possible refuelling, are also defined as constant parameters:

- A_{ik} is the level of supply of sensor $k \in K$ after leaving state $i \in S \cup R$.

Variables and constant parameters for time:

- $[a_i, b_i]$ is the time windows related to the state $i \in N$;
- o_{ik} is the starting time of state $i \in N$ for sensor $k \in K$;
- Δ_{ik} is the necessary time period for sensor $k \in K$ to perform state $i \in N$; (execution time)
- t_{ijk} is the necessary time period for sensor $k \in K$ to move from state $i \in N$ to state $j \in N$. (transition time)

As a conclusion: the variables of the problem are x, y, ω, α and β. Next paragraphs will present the relationship between these parameters and variables, under the form of constraints and optimization criterion.

Trajectory constraints: We consider the constraints which link variables x, y and ω, and which state that the sensors perform non cyclic states trajectories, from starting points to arrival points:

$$y_{ik} + y_{jk} \geq 2x_{ijk}, \forall i, j \in N, k \in K, \tag{1}$$

$$1 + \sum_{i,j \in N} x_{ijk} = \sum_{i \in N} y_{ik}, \forall k \in K, \tag{2}$$

$$\sum_{i \in N} x_{ihk} = \sum_{i \in N} x_{hjk}, \forall h \in M \cup R, k \in K, \tag{3}$$

$$\omega_{jk} \geq \omega_{ik} + 1 + \infty \times (1 - x_{ijk}), \forall i, j \in N, k \in K, \tag{4}$$

$$x_{ijk} = 0, \forall i \in N, k \in K, j \in S, \tag{5}$$

$$x_{ijk} = 0, \forall j \in N, k \in K, i \in E, \tag{6}$$

$$\sum_{i \in S(k)} y_{ik} = 1; \sum_{i \in E(k)} y_{ik} = 1, \forall k \in K. \tag{7}$$

Time windows constraints:

$$o_{ik} + \Delta_{ik} + t_{ijk} - \infty \times (1 - x_{ijk}) \leq o_{jk}, \forall i, j \in N, k \in K, \tag{8}$$

$$a_i \leq o_{ik}, o_{ik} + \Delta_{ik} \leq b_i, \forall k \in K, \forall i \in S \cup R \cup E, \tag{9}$$

$$a_i \leq o_{ik}, o_{ik} + \Delta_{ik} \leq b_i, \forall k \in K, \text{ for all reconnaissance mission } i, \tag{10}$$

$$o_{ik} \leq a_i, b_i \leq o_{ik} + \Delta_{ik}, \forall k \in K, \text{ for all surveillance mission } i. \tag{11}$$

Resource constraints:

$$\alpha_{jk} \leq \beta_{ik} - c_{ijk} + \infty \times (1 - x_{ijk}), \forall i, j \in N, k \in K, \tag{12}$$

$$\beta_{ik} = A_{ik}, \forall i \in S \cup R, \qquad \text{(fuelled/refuelled)} \tag{13}$$

$$\beta_{ik} = \alpha_{ik}, \forall i \in M \cup E, \qquad \text{(not refuelled)} \tag{14}$$

$$\alpha \geq 0, \beta \geq 0. \tag{15}$$

Also we consider the following constraint:

$$\sum_{i \in \mu(u)} \sum_{k \in K} y_{ik} \leq 1, \forall u \in F. \tag{16}$$

Our purpose is to maximize a global criterion which is a balance between the satisfaction of the FIR and the expense. Thus, we have the following optimization problem :

$$\begin{cases} \max\limits_{x,y,u,o,\alpha,\beta} \left(\sum\limits_{u \in F} p_u \sum\limits_{i \in \mu(u)} \sum\limits_{k \in K} y_{ik} g_{ik} - \sum\limits_{i \in N} \sum\limits_{j \in N} \sum\limits_{k \in K} \widetilde{c}_{ijk} x_{ijk} \right) \\ \text{subject to: from (1) to (16).} \end{cases} \tag{17}$$

This is a linear mixed 0-1 programming. This problem is NP-complete, since it is a generalization of VRPTW. Therefore, our considered problem is very hard to solve, even for reasonably sized cases.

3 A column generation approach

While several successful methods for solving several VRPTW variants have been proposed in the literature [1–3, 6, 7, 20, 21], one of the most famous approach is column generation. The embedding of column generation techniques within a linear-programming-based branch-and-bound framework, introduced by Desrosiers et al. [6] for solving the VRPTW, became classic. It contributed as the key step in the design of exact algorithms for a large class of integer programs [15]. Nowadays, column generation is a prominent method to cope with a huge number of variables, and numerous integer programming column generation applications have been developed (see e.g. [15] for an overview). As a generalization of the VRPTW, our sensor planning has some good properties (for instance, trajectory constraints and time windows constraints) for a column generation based approach. Therefore, we will investigate the column generation approach for solving the problem (17) in this section.

3.1 Column generation

Applying the methodology described in [6], the column generation approach will be based on the notion of feasible routes for the sensors. A feasible route of a sensor $k \in K$ is a route starting from a compatible departure, going to a compatible endpoint, satisfying all constraints and visiting *at least one mission* $i \in M$. We denote by Ω_k the set of all feasible routes for sensor k, and $\Omega = \bigcup_{k \in K} \Omega_k$ the set of all feasible routes.

Let $r = (r_1, r_2, ..., r_m) \in \Omega_k \subset \Omega$ be a route, where $r_1, \cdots, r_m \in N$ are the states visited by the sensor k. The performance of this route, denoted by $f(r)$, is computed as follows:

$$f(r) = \underbrace{\sum_{u\in F:\mu(u)\cap\{r_2,\ldots,r_{m-1}\}=\{r_h\}} p_u g_{r_h,k}}_{g(r)} - \underbrace{\sum_{i=1}^{m-1} \widetilde{c}_{r_i,r_{i+1},k}}_{c(r)}. \tag{18}$$

In this formula, $g(r)$ is the gain of route r, and $c(r)$ is the cost of route r.

Now we define the parameter $a_{ru}, u \in F$ by:

$$a_{ru} = \begin{cases} 1 & \text{if } \mu(u) \cap \{r_2, ..., r_{m-1}\} \neq \emptyset\ , \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

The sensor planning problem (17) is reformulated as:

$$\begin{cases} \max \sum\limits_{k\in K} \sum\limits_{r\in\Omega_k} f(r).\theta_r \\ \text{s.t.} \sum\limits_{k\in K} \sum\limits_{r\in\Omega_k} a_{ru}\theta_r \leq 1, \forall u \in F, \\ \quad \sum\limits_{r\in\Omega_k} a_{ru}\theta_r \leq 1, \forall k \in K, \\ \quad \theta_r \in \{0,1\}, \forall r \in \Omega. \end{cases} \tag{20}$$

The variable $\theta_r \in \{0,1\}$ is a decision variable which describes whether a route r is chosen or not. The first constraint specifies that each requirement $u \in F$ is satisfied at most one time while the second constraint ensures that each sensor $k \in K$ does at most one feasible route.

Because of the first constraint, the condition $\theta_r \in \{0,1\}, \forall r \in \Omega$ can be replaced by $\theta_r \in \mathbb{N}, \forall r \in \Omega$. The linear relaxation of problem (20), i.e., with $\theta_r \geq 0, \forall r \in \Omega$, is called Master Problem (MP), which is an instrument for evaluating the feasible route generated at each iteration. The methodology of column generation approach can be described as follows.

Let $\Omega_k^1 \subset \Omega_k, k \in K$, and $\Omega^1 = \bigcup_{k\in K} \Omega_k^1$. We consider the restriction of the Master Problem, denoted MP(Ω^1):

$$\begin{cases} \max \sum\limits_{k\in K} \sum\limits_{r\in\Omega_k^1} f(r).\theta_r \\ \text{s.t.} \sum\limits_{k\in K} \sum\limits_{r\in\Omega_k^1} a_{ru}\theta_r \leq 1, \forall u \in F, \\ \quad \sum\limits_{r\in\Omega_k^1} a_{ru}\theta_r \leq 1, \forall k \in K, \\ \quad \theta_r \geq 0, \forall r \in \Omega^1. \end{cases} \tag{21}$$

The dual program of (21), denoted by D(Ω^1), is:

$$\begin{cases} \min \sum\limits_{u\in F} \lambda_u + \sum\limits_{k\in K} \mu_k \\ \text{s.t.} \sum\limits_{u\in F} a_{ru}\lambda_u + \mu_k \geq f(r), \forall r \in \Omega_k^1, k \in K, \\ \quad \lambda_u \geq 0, \forall u \in F, \\ \quad \mu_k \geq 0, \forall k \in K. \end{cases} \tag{22}$$

Now suppose that $(\bar{\lambda}, \bar{\mu}) = (\bar{\lambda}_1, ..., \bar{\lambda}_F, \bar{\mu}_1, ..., \bar{\mu}_K)$ is an optimal solution of the dual problem D(Ω^1). Then, we have:

$$\sum_{u\in F} a_{ru}\bar{\lambda}_u + \bar{\mu}_k \geq f(r), \forall r \in \Omega_k^1, k \in K.$$

It is clear that if this condition holds for all $r \in \Omega_k, k \in K$, then $(\bar{\lambda}, \bar{\mu})$ is also the optimal solution of the dual program of (MP). Otherwise, we will look for a route $r \in \Omega_k \backslash \Omega_k^1$, for a $k \in K$ such that:

$$\sum_{u \in F} a_{ru} \bar{\lambda}_u + \bar{\mu}_k < f(r). \tag{23}$$

This is called the *sub-problem*.

The column generation algorithm for solving the problem (20) can be described as follows:

Column generation algorithm for solving (20):
Step 1. Generate initial sets Ω_k^1 for $k \in K$,
Step 2. Solve the problem (21) in order to obtain the optimal solution and its dual solution $(\bar{\lambda}, \bar{\mu})$,
Step 3. For each $k \in K$, find a route $r \in \Omega_k \backslash \Omega_k^1$ satisfying the condition (23) and update $\Omega_k^1 := \Omega_k^1 \cup \{r\}$,
Step 4. Iterate step 2-3 until there is no route satisfying the condition (23),
Step 5. Solve (20) with $\Omega := \Omega^1$.

3.2 A label correcting method for solving the sub-problem

In [17], we have proposed an approach using CPLEX for the MILP formulation of the sub-problem in Step 3. In this section, we investigate another method for solving the sub-problem: the label correcting method. This method is based on the ideas of Feillet et al. (2004) [8] developped for treating the Elementary Shortest Path Problem with Resource Constraints. The principle of this method is to use the dynamic programming.

For a sensor k fixed, we consider $F(k) = \{FIR_1, \ldots, FIR_m\}$ the set of associated requirements. For each requirement $FIR_u \in F(k)$, we denote $\mu_k(FIR_u)$ the set of missions which can be performed by the sensor k in order to satisfy this requirement. Additionally, we denote $R(k) = \{R_1, ..., R_n\}$ the set of compatible refuelling centres and $E(k)$ the set of compatible endpoints corresponding to this sensor k. Since the position of sensor k is known, we also use the notation k to represent its position, and call $V^k = F(k) \cup R(k) \cup \{k\}$ the set of nodes.

Definition 1. *Each path P_{kv} from the position of sensor k to a node $v \in V^k \backslash \{k\}$ associates a state $H_v = (T_v^1, T_v^2, X_v^1, ..., X_v^m, Y_v^1, ..., Y_v^n, Z_v)$ and a performance $f_v = f(P_{kv})$. Here, the two first parameters T_v^1, T_v^2 correspond to the quantity of time and fuel resources used by the path. The parameters $X_v^1, ..., X_v^m$ represent the visitation of requirement ($X_v^u = i \neq 0$ if the path visits the requirement FIR_u by performing the mission $i \in \mu_k(FIR_u)$, 0 otherwise), and the parameters $Y_v^1, ..., Y_v^n$ represent the visitation of refuelling centre ($Y_v^i = 1$ if the path visits the refuelling centre R_i, 0 otherwise). The last parameter Z_v shows the ability to reach an endpoint, i.e., $Z_v = 1$, if after visiting node v, the sensor k can go to some endpoint, 0 otherwise. The couple $\lambda_v = (H_v, f_v)$ is said to be a label on the node v.*

Definition 2. *Let P_{kv} and $\bar{P}_{kv}$ be two paths from the position of sensor k to a node v with associated labels $(H_v, f_v), H_v = (T_v^1, T_v^2, X_v^1, ..., X_v^m, Y_v^1, ..., Y_v^n, Z_v)$ and $(\bar{H}_v, \bar{f}_v)$, $\bar{H}_v = (\bar{T}_v^1, \bar{T}_v^2, \bar{X}_v^1, ..., \bar{X}_v^m, \bar{Y}_v^1, ..., \bar{Y}_v^n, \bar{Z}_v)$. We say that P_{kv} dominates $\bar{P}_{kv}$ if:*

$$T_v^i \leq \bar{T}_v^i, \forall i = 1, 2, id(X_v^i) \geq id(\bar{X}_v^i), \forall i = 1, 2, ..., m,$$
$$Y_v^i \leq \bar{Y}_v^i, \forall i = 1, 2, ..., n, Z_v \geq \bar{Z}_v, f_v \geq \bar{f}_v.$$

Here, $id(x) = 1$, *if* $x \neq 0; id(x) = 0$ *otherwise.*

We use the following notations to describe the algorithm:

- Λ_v: List of labels on node v.
- $Succ(v)$: Set of successors of node v.
- L: List of nodes waiting to be treated.
- $Extend(\lambda_v, \tilde{v})$: Multi-value function that returns the labels resulting from the extension of label $\lambda_v = (H_v, f_v) \in \Lambda_v$ towards node $\tilde{v}$ (with respect to the missions at $\tilde{v}$) when the extension is possible, nothing otherwise. More precisely, suppose that $\lambda_v = (H_v, f_v) \in \Lambda_v$ is a label on v, with $H_v = (T_v^1, T_v^2, X_v^1, ..., X_v^m, Y_v^1, ..., Y_v^n, Z_v)$. We will distinguish two cases of $\tilde{v}$ as follows:
  - If $\tilde{v}$ is a FIR, and $\mu_k(\tilde{v}) = \{m_1, ..., m_j\}$ are the set of missions corresponding, then we extend this label with respect to each mission $m_i, i = 1, ..., j$ in order to obtain the new label $\lambda_{\tilde{v}} = (H_{\tilde{v}}, f_{\tilde{v}})$ as follows.

    + If m_i is a reconnaisance mission

$$T_{\tilde{v}}^1 = \begin{cases} T_v^1 + t_{v,m_i,k} + \Delta_{m_i,k} & \text{if } a_{m_i} < T_v^1 + t_{v,m_i,k} < T_v^1 + t_{v,m_i,k} + \Delta_{m_i,k} \leq b_{m_i} \\ a_{m_i} + \Delta_{m_i,k} & \text{if } T_v^1 + t_{v,m_i,k} \leq a_{m_i}, \end{cases} \tag{24}$$

    + If m_i is a surveillance mission

$$T_{\tilde{v}}^1 = T_v^1 + t_{v,m_i,k} + \Delta_{m_i,k} \text{ if } T_v^1 + t_{v,m_i,k} \leq a_{m_i}, \quad b_{m_i} \leq T_v^1 + t_{v,m_i,k} + \Delta_{m_i,k}, \tag{25}$$

$$T_{\tilde{v}}^2 = T_v^2 + c_{v,m_i,k} \text{ if } T_v^2 + c_{v,m_i,k} \leq A_k, \tag{26}$$

$$X_{\tilde{v}}^{\tilde{v}} = m_i, \tag{27}$$

$$Z_{\tilde{v}} = 1 \text{ if the sensor can go to an endpoint after performing the mission } m_i, \text{ otherwise 0}, \tag{28}$$

$$f_{\tilde{v}} = f_v + g_{m_i,k} - \epsilon_1(c_{v,m_i,k} + \epsilon_2 d_{v,m_i,k}). \tag{29}$$

    Here, A_k is the capacity of sensor k. Of course, if the conditions in (24), or (25) or (26) are violated, there is no extension. Therefore, from a label λ_v, after extension procedure we get at most $|\mu(\tilde{v})|$ new labels on node $\tilde{v}$.
  - If $\tilde{v}$ is a refuelling centre, we extend the label $\lambda_v = (H_v, f_v)$ to obtain a new label $\lambda_{\tilde{v}} = (H_{\tilde{v}}, f_{\tilde{v}})$ on $\tilde{v}$ by updating the following parameters:

$$T_{\tilde{v}}^1 = \begin{cases} T_v^1 + t_{v,\tilde{v},k} + \Delta_{\tilde{v},k} & \text{if } a_{\tilde{v}} < T_v^1 + t_{v,\tilde{v},k} < T_v^1 + t_{v,\tilde{v},k} + \Delta_{\tilde{v},k} \leq b_{\tilde{v}} \\ a_{\tilde{v}} + \Delta_{\tilde{v},k} & \text{if } T_v^1 + t_{v,\tilde{v},k} \leq a_{\tilde{v}}, \end{cases} \tag{30}$$

$$T_{\tilde{v}}^2 = 0 \text{ if } T_v^2 + c_{v,\tilde{v},k} \leq A_k, \tag{31}$$

$$Y_{\tilde{v}}^{\tilde{v}} = 1, \tag{32}$$

$$Z_{\tilde{v}} = 1 \text{ if the sensor can go to an endpoint after refuelling at } \tilde{v}, \text{ otherwise 0}, \tag{33}$$

$$f_{\tilde{v}} = f_v - \epsilon_1(c_{v,\tilde{v},k} + \epsilon_2 d_{v,\tilde{v},k}). \tag{34}$$

    If the conditions in (30) or (31) are violated, then there is no extension.
- $F_{v,\tilde{v}}$: Set of labels extended from node v to node $\tilde{v}$.
- $EFF(\Lambda)$: Procedure that keeps only non-dominated labels in the list of labels Λ.

The label correcting procedure for solving the subproblem can be described as follows.

LabelCorrecting(k):
Set $\Lambda_k = (0, 0, ..., 0)$ and $\Lambda_v = \emptyset$ for all $v \in V^k \backslash \{k\}$
Set $L = \{k\}$
Repeat
Choose $v \in L$
for all $\tilde{v} \in Succ(v)$
Set $F_{v,\tilde{v}} = \emptyset$
for all $\lambda_v = (H_v, f_v) \in \Lambda_v$, with $H_v = (T_v^1, T_v^2, X_v^1, ..., X_v^m, Y_v^1, ..., Y_v^n, Z_v)$
if $X_v^{\tilde{v}} = 0$ or $Y_v^{\tilde{v}} = 0$ **then**
$F_{v,\tilde{v}} := F_{v,\tilde{v}} \cup \{Extend(\lambda_v, \tilde{v})\}$
endif
endfor
$\Lambda_{\tilde{v}} = EFF(\Lambda_{\tilde{v}} \cup F_{v,\tilde{v}})$
if $\Lambda_{\tilde{v}}$ has changed **then**
$L = L \cup \{\tilde{v}\}$
endif
endfor
Set $L = L \backslash \{v\}$
Until finding a label $\lambda_v = (H_v, f_v)$ satisfying the following condition:

$$f_v \text{ satisfies (23) and } Z_v = 1.$$

Remark 1. In practice, to prevent the explosion of number of labels, we should limit the number of labels on each node at each iteration. We denote l_v the maximum labels on node v. After the step "$\Lambda_{\tilde{v}} = EFF(\Lambda_{\tilde{v}} \cup F_{v,\tilde{v}})$", if $\text{card}(\Lambda_{\tilde{v}}) > l_{\tilde{v}}$ then we only remain $l_{\tilde{v}}$ labels which have more requirements visited.

4 Experiments and Numerical Results

Our algorithm is written in MATLAB 2010, and is tested on a PC 64 bits Windows 7, Intel(R) Xeon (R) CPU X5690 @ 3.47 GHz 3.47 GHz, 24G of RAM. CPLEX 12.4 is used for solving the linear program (21), and the problem (20) in Step 5. In order to evaluate the performance of this approach, we compare the results obtained by our approach with a purely CPLEX-based approach (applying directly to the problem (17)).

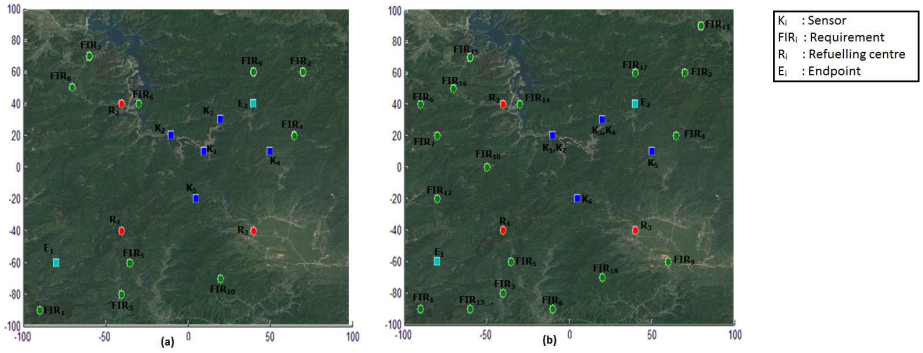

Fig. 1. Plans.

In our scenarios, we assume that the sensors are starting from unique starting points, *i.e.* $\#S(k) = 1$, and that the sensors are endowed with the same autonomy level $A_{ik} = A$ after (re-)fuelling. The costs are also identically valued by $c_{ijk} = 20$. Therefore, if $A = 100$, then each sensor can visit less or equal to 5 states without refuelling. The priority of requirement is determined as follows: if the requirement u has the priority 1 (resp. 2), then $p_u = 100$ (resp. $p_u = 1$). We also define $\Delta_{ik} = 20$ (minutes), $t_{ijk} = 20$ (minutes) and $\epsilon_1 = \epsilon_2 = 10^{-4}$.

The set of initial routes for the column generation method is generated as follows: for each requirement $u \in F$, we choose a mission $i \in \mu(u)$ and a compatible sensor $k \in K$ that performs the maximum gain. Then, we choose an arrival point e which implies the smallest corrected cost, thus obtaining the route: "$s \to i \to e$".

4.1 The first data

We have $|F| = 10$ requirements, $|M| = 15$ missions, $|R| = 3$ refuelling centres, $|E| = 2$ arrival points and $|K| = 5$ sensors (see Fig. 1 (a)). Tables 1-3 present the parameters of missions, refuelling centres and arrival points respectively. Table 4 presents the gains of missions performed by the sensors. In this case, we have MILPs with 3250 binary variables, 465 continuous variables and 10730 constraints. The maximum number of labels $l_v = 100, \forall v \in V^k, \forall k \in K$. The computational time of label correcting algorithm for each sensor is limited to 10 seconds.

Table 1. Parameters of missions in Data 1

Requirement	Mission	Type of mission	Priority	Compatible type of sensor	Start	End
1	1	R	1	2, 4	110	360
	2	R	1	2, 4	110	360
2	3	R	1	2, 4	250	375
	4	R	1	2, 4	250	375
3	5	R	2	2, 4	425	540
	6	R	2	2, 4	425	540
4	7	R	1	4	320	370
	8	R	1	2, 4	320	370
5	9	R	1	2, 4	120	470
	10	R	1	2, 4	120	470
6	11	S	1	4	280	300
7	12	S	2	1, 4	320	340
8	13	S	1	4	360	380
9	14	S	1	4	400	420
10	15	S	2	4	445	465

Table 2. Parameters of refueling centers in Data 1

Refuelling centre	Compatible type of sensor	Start	End
1	2, 4	100	600
2	1, 2, 3, 4	100	600
3	1, 2, 3, 4	100	600

Table 3. Parameters of endpoints in Data 1

Endpoint	Compatible type of sensor	Start	End
1	4	100	600
2	1, 2, 3	100	600

Table 4. The gains of missions given by sensors in Data 1

Mission	K_1 (Type 1)	K_2 (Type 2)	K_3 (Type 3)	K_4 (Type 4)	K_5 (Type 1)
1	0	10	0	6	0
2	0	12	0	5	0
3	0	18	0	15	0
4	0	17	0	17	0
5	0	7	0	17	0
6	0	8	0	16	0
7	0	0	0	14	0
8	0	20	0	13	0
9	0	18	0	13	0
10	0	18	0	13	0
11	0	0	0	12	0
12	4	0	0	14	5
13	0	0	0	11	0
14	0	0	0	9	0
15	0	0	0	18	0

Table 5. Numerical results for Data 1

Test	A	CPLEX		Column Generation			
		Objective value	CPU time (s)	Objective value	CPU time (s)	Iteration	Column
1	120	10039.9760	41.19	10039.9740	52.94	65	87
2	100	10039.9720	38.42	10030.9720	31.48	46	80
3	80	10030.9700	39.61	10030.9700	27.88	53	85

Table 5 gives the comparative results between the column generation method and CPLEX for different values of parameter A. From Table 5, we see that the column generation method produced very good solutions. The relative error of

objective value between the two methods varies from 0.00% to 0.09% (0.03% in average). Moreover, the column generation method is slightly faster than the pure CPLEX approach: the average of CPU time of column generation method is 37.43 seconds while this of CPLEX is 39.74 seconds.

4.2 The second data

In this section, we tested the performance of our approach on a large scale scenario. We have $|F| = 18$ requirements, $|M| = 34$ missions, $|R| = 3$ refuelling centres, $|E| = 2$ arrival points and $|K| = 6$ sensors (see Fig. 1 (b)). The refuelling centres and arrival points are the same as in the first data set. Tables 6 presents the parameters of missions. Table 7 presents the gains of missions performed by the sensors. In this case, we have MILPs with 12420 binary variables, 1032 continuous variables and 44106 constraints.

Table 6. Parameters of missions in Data 2

Requirement	Mission	Type of mission	Priority	Compatible type of sensor	Start	End
1	1	R	1	2, 4	110	360
	2	R	1	2, 4	110	360
2	3	R	1	2, 4	250	375
	4	R	1	2, 4	250	375
3	5	R	2	2, 4	425	540
	6	R	2	2, 4	425	540
4	7	R	1	4	320	370
	8	R	1	2, 4	320	370
5	9	R	1	2, 4	120	470
	10	R	1	2, 4	120	470
6	11	R	1	1, 2	110	360
	12	R	1	2, 3	110	360
7	13	R	1	2, 3	250	375
	14	R	1	1, 2	250	375
8	15	R	2	1, 2, 4	425	540
	16	R	2	2, 3, 4	425	540
9	17	R	1	4	320	370
	18	R	1	1, 2, 4	320	370
10	19	R	1	1, 3, 4	120	470
	20	R	1	1, 2, 4	120	470
11	21	R	1	2, 4	220	300
	22	S	1	1, 2, 4	220	240
	23	R	1	2, 3, 4	220	280
12	24	R	2	1, 2, 4	260	360
	25	R	2	1, 3, 4	260	340
	26	R	2	1, 3, 4	260	300
13	27	R	1	1, 2, 3, 4	280	360
	28	R	1	1, 4	280	390
	29	R	1	1, 4	280	350
14	30	S	1	1, 4	280	300
15	31	S	2	1, 4	320	340
16	32	S	1	2, 4	360	380
17	33	S	1	1, 4	400	420
18	34	S	2	2, 4	445	465

Table 7. The gains of missions given by sensors in Data 2

Mission	K_1 (Type 1)	K_2 (Type 1)	K_3 (Type 2)	K_4 (Type 2)	K_5 (Type 3)	K_6 (Type 4)
1	0	0	17	17	0	16
2	0	0	17	17	0	18
3	0	0	15	15	0	14
4	0	0	13	13	0	15
5	0	0	14	14	0	14
6	0	0	16	16	0	12
7	0	0	0	0	0	16
8	0	0	13	13	0	16
9	0	0	9	9	0	13
10	0	0	10	10	0	14
11	11	11	9	9	0	0
12	0	0	11	11	15	0
13	0	0	12	12	14	0
14	17	17	13	13	0	0
15	17	17	14	14	0	13
16	0	0	15	15	14	15
17	0	0	0	0	0	19
18	12	12	18	18	0	18
19	4	4	0	0	8	7
20	5	5	8	8	0	9
21	0	0	11	11	0	12
22	17	17	9	9	0	11
23	0	0	10	10	13	9
24	4	4	13	13	0	11
25	5	5	0	0	7	10
26	5	5	0	0	8	12
27	11	11	15	15	15	15
28	10	10	0	0	0	14
29	12	12	0	0	0	16
30	14	14	0	0	0	10
31	8	8	0	0	0	14
32	0	0	15	15	0	9
33	11	11	0	0	0	13
34	0	0	7	7	0	8

Table 8. Numerical results for Data 2

Test	A	CPLEX		Column Generation			
		Objective value	CPU time (s)	Objective value	CPU time (s)	Iteration	Column
1	120	19751.9480	5292.98	19632.9540	605.86	38	145
2	100	19761.9500	1496.26	19532.9540	377.62	42	141
3	80	19760.9500	1479.12	19432.9520	405.47	48	156

Here, the sensors K_1, K_2 are of the same type (Type 1) and located in the same position (depot), and so are the sensors K_3, K_4 (Type 2). As is done classically, same-type sensors have been solved by only one sub-problem. The maximum number of labels $l_v = 200, \forall v \in V^k, \forall k \in K$. The computational time of label correcting algorithm for each sensor is limited to 20 seconds. Also, we use a stopping criteria (gap $\leq 1\%$) when implementing the purely CPLEX-based approach.

From Table 8, we see that the column generation method once again produces quite good solutions in acceptable time. The relative error of objective value between the two methods varies from 0.61% to 1.69% (1.16% in average).

5 Conclusion

In this paper, we have proposed a column generation approach for solving the optimal sensors management in an information collection process, where a label correcting algorithm has been developped for treating the sub-problem. The comparative results with CPLEX have demonstrated the efficiency of our proposed approach. It found a near-optimal solution within acceptable time for even large-scale problems. In the future, we plan to study some dedicated heuristics and meta-heuristics for the search of column candidate. Also, we intend to parallelize the Step 3 so as to speed up our algorithm.

Acknowledgement

The authors would like to thank the DGA (Délégation Générale pour l'Armement) for its support to this research.

References

1. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. Transport. Sci. 39(1), 104–118 (2005)
2. Bräysy, O., Gendreau, M.: Vehicle routing problem with time windows, Part II: Metaheuristics. TTransport. Sci. 39 (1), 119–139 (2005)
3. Campbell, A., Savelsbergh, M.: Efficient insertion heuristics for vehicle routing and scheduling problems. Transport. Sci. 38 (3), 369–378 (2004)
4. Chakrabarty, K., Iyengar, S.S., Qi, H., Cho E.: Grid coverage for surveillance and target location in distributed sensor networks. IEEE Transactions on Computers 51, 1448–1453 (2002)
5. Dambreville, F.: Cross-entropic learning of a machine for the decision in a partially observable universe. Journal of Global Optimization 37(4), 541–555 (2007)
6. Desrosiers, J., Soumis, F., Desrochers, M.: Routing With Time Windows by Column Generation. Networks 14, 545–565 (1984)
7. Desrosiers, J., Soumis, F., Sauvé, M.: Lagrangian Relaxation Methods for Solving the Minimum Fleet Size Multiple Traveling Salesman Problem With Time Windows. Mgmt. Sci. 34, 1005–1022 (1988)
8. Feillet, D., Dejax, P., Gendreau, M., Gueguen, C.: An exact algorithm for the elementary shortest path problemwith resource constraints: application to some vehicle routing problems. Networks 44(3), 216–229 (2004)
9. Frost, J.R.: Principle of search theory. Technical report, Soza & Company Ltd (1999)
10. Haley, K.B., Stone L.D.: Search Theory and Applications. Plenum Press, New York (1980)
11. Janez, F.: Optimization method for sensor planning. Aerospace Science and Technologie 11, 310–316 (2007)
12. Jayaweera, S.K.: Optimal node placement in decision fusion wireless sensor networks for distributed detection of a randomly-located target. IEEE Military Communications Conference, pp. 1–6 (2007)
13. Koopman, B.O.: The theory of search. iii. the optimum distribution of searching effort. Operations Research 5(5), 613–626 (1957)
14. Le Thi, H.A., Nguyen, D.M., and Pham, D.T.: A DC programming approach for planning a multisensor multizones search for a target. Computers & Operations Research, Online first (July 2012)
15. Lübbecke M., Desrosiers, J.: Selected Topics in Column Generation. Operations Research 53 (6), 1007–1023 (2005)
16. Ng, G.W., Ng, K.H.: Sensor management – what, why and how. Information Fusion 1(2), 67–75 (2000)
17. Nguyen, D.M., Dambreville, F., Toumi, A., Cexus, J.C., Khenchaf, A.: A column generation method for solving the sensor management in an information collection process. Submitted to Optimization (October 2012)
18. Schaefer, C.G., Hintz, K.J.: Sensor management in a sensor rich environment. In: Proceedings of the SPIE International Symposium on Aerospace/Defense Sensing and Control, Orlando, FL, vol. 4052, pp. 48–57 (2000)
19. Simonin, C., Le Cadre, J.-P., Dambreville, F.: A hierarchical approach for planning a multisensor multizone search for a moving target. Computers and Operations Research 36(7), 2179–2192 (2009)
20. Solomon, M.: Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints. Operations Research 35, 254–365 (1987)
21. Toth P., Vigo, D.: An exact algorithm for the vehicle routing problem with backhauls. Transport. Sci. 31, 372–385 (1997)
22. Tremois, O., Le Cadre, J.-P.: Optimal observer trajectory in bearings-only tracking for maneuvering sources. Sonar and Navigation 146(1), 1242–1257 (1997)
23. Washburn, A.R.: Search for a moving target : The FAB algorithm. Operations Research 31(4), 739–751 (1983)
24. Xiong, N., Svensson, P.: Multi-sensor management for information fusion : issues and approaches. Information Fusion 3(2), 163–186 (2002)