Theatre-wide automatic mission scheduling for a mine countermeasure force

Andreas Arnold-Bos, Israel Bacor, Jean-Philippe Brunet, Matthieu Lecouvez

Theatre-wide automatic mission scheduling for a mine countermeasure force

Andreas ARNOLD-BOS, Israel BACOR,
Jean-Philippe BRUNET
Thales Underwater Systems SAS
Route de Sainte Anne du Portzic
CS 43814
29238 BREST cedex, France
andreas.arnold-bos@fr.thalesgroup.com

Matthieu LECOUVEZ
École Nationale de Ponts et Chaussées ParisTech
6 et 8 avenue Blaise Pascal
77400 CHAMPS SUR MARNE, France

***Abstract*— Current combat management systems (CMS) for mine countermeasure (MCM) warfare mostly plan single tasks for single means at a time. Yet the current trend puts an increasing emphasis on unmanned vehicles (UxVs). Whereas individual task planning is feasible when only a few MCM vessels are used, this becomes intractable when a large number of UxVs need being coordinated over a wide theatre, sometimes for weeks. In our venture to develop a force-level MCM CMS, we are imagining ways to alleviate the burden of the operators in such a situation. This paper concerns the feasibility of automatic force-level mission scheduling for MCM warfare. This novel and ambitious task requires concurrently solving two complex problems: *i*) how to generate mission zones which are sub-zones of the operational theatre, fitting assets characteristics and constraints (environment, threat, etc); and *ii*) how to schedule the missions according to the availability of an asset and results of previous missions. We expose a formalization of the problem, then some of our solving methods and first results, which we validate through simulation.**

Mission planning, scheduling, mine warfare

I. INTRODUCTION

The current trend in mine counter-measure (MCM) warfare goes towards the massive use of unmanned vehicles (UxVs). Such vehicles allow standoff operations with reduced risks for humans; since the ships stay far from the danger they may also be built following less costly, civilian standards. Also, underwater vehicles for zone inspection are more discreet than surface vessels. These unmanned vehicles are specialized in one or several tasks: transport, sonar towing, mine detection using sonar, object identification, and finally mine disposal. In the scenario considered herein, a mother ship lays in the water several unmanned surface vessels (USVs). These USVs are designed to either deploy a towed synthetic aperture sonar (T-SAS), transport an explosive ordnance disposal (EOD) team, or to launch specialized UxVs of various sizes: either large unmanned underwater vehicles for mine detection and classification (D/C-UUVs) using sonar, large UUVs for mine identification using sonar/video means (I-UUVs), or expendable mine killers for neutralization (N-UUVs). The USVs are reconfigurable on the mother ship in a short time. The USVs also act as a communication relay between the UUVs and the mother ship. Figure 1 illustrates such an architecture.

This kind of configuration is more complex to use, than traditional MCM forces where mine-hunters are the base assets and few ships are employed at a time. In the near future, assets will be more numerous, and with several different capabilities and operational constraints, thus they will be harder to coordinate. Also, newer, more complex missions may be addressed, than what was previously doable with mine hunters: for instance, covertly clearing a coastal zone to prepare a beach attack, an operation that may take weeks. In this context, proper mission planning is critical but very hard, if not impossible, to do by hand.

Current combat management systems (CMS) for MCM operations typically focus on planning single tasks for single user-designated asset at one time. At this level, the CMS typically computes tracks and optimizes the sonar parameters to sweep a relatively small user-defined zone only a few nautical miles wide.

When the CMS is to manage a MCM force made of heterogeneous assets over a large zone, requiring many launch and recovery cycles, the desirable requirements are more numerous. First, it should be able to automatically (or at least help an operator to) divide the theatre into sub-zones that are suitable for a given asset in terms of area, communication requirements and ranges, environmental conditions and threat level. This is spatial allocation problem. Second, the CMS should also assist the user in sequencing the order in which the missions should be executed. Ideally, the zones partition and the mission planning should minimize the total duration of the time on the theatre and be robust to failures or asset losses. Finally, once sub-zones and assets are assigned to the zones, the CMS should assist the user to perform the fine planning such as precise track determination. Also, the system should be easily extensible to new types of assets.

In this paper, we begin by showing the limits of the Operational Research framework when solving the MCM asset/area allocation problem (section II). After some mathematical formalization (section III) we first show how to select assets and determine how many launches are required to process a polygonal zone (section IV) before showing how to

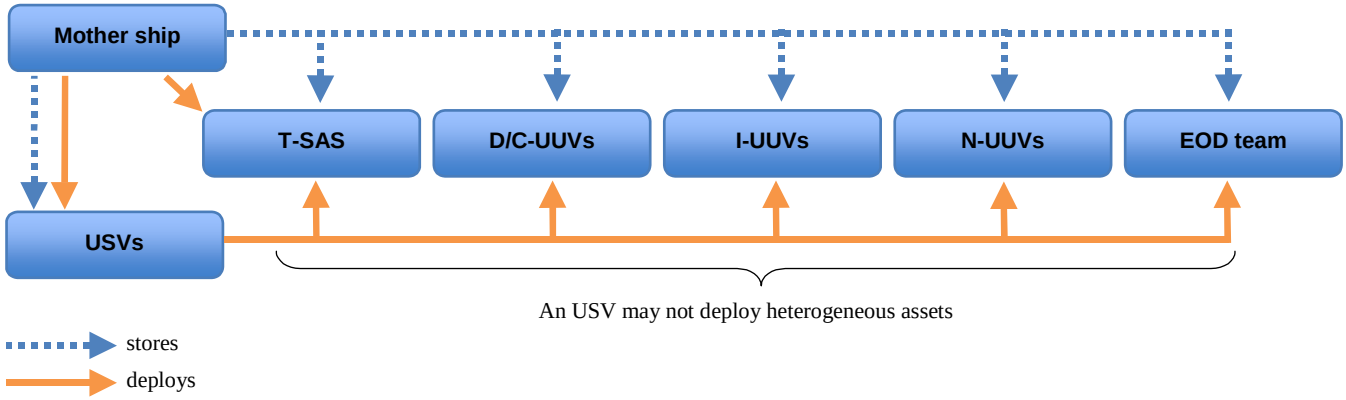

Figure 1 – A dummy ship/UxV architecture. The MCM mother ship may use several USVs simultaneously, one of them being for instance configured to deploy D/C-UUVs and another configured to deploy N-UUVs. USVs may be reconfigured on the mother ship to deploy other assets.

cut the polygon into sub-polygons so that each of these sub-polygons may be assigned to a single asset compatible with it (section V). This dual approach is then demonstrated in section VI on a simple example.

II. MCM Mission Scheduling Seen Through The Operational Research Lens

Graham *et al.* [1] introduced a well-known notation that has become the norm to categorize most classes of problems studied in Operational Research. According to this notation, the job scheduling component of our problem could be described as a (JMPM|pmtn,prec,r_j|$\Sigma w_j U_j$). In layman's terms, the zones are processed in a *job-shop* fashion (J), *i.e.* they undergo a series of operations: detection, classification, identification, neutralization, in this order. For one or several multi-purpose machines (MPM) are used: the UxVs. The jobs may be *preempted* (pmtn) and resumed later possibly on another machine. The *precedence* (prec) relation between jobs is here generic, as an oriented graph corresponding to the adjacency graph of the zones. The jobs may have to be finished at given dates r_j though this constraint may be relaxed. Finally, the objective function to minimize depends on the military nature of the scenario. For instance, it may be necessary to minimize the delay between the beginning and the end of the operations on the theatre (the *makespan*); this does not allow to set priorities. On the other hand, it is also possible to minimize the number of late operations, each late job yielding a unit penalty U_j weighted by a priority w_j which may be zero if the sub-zone must only be optionally cleared.

To the best of our knowledge, no solution to this problem is available off the shelf. Our problem generalizes the (J_2| | C_{max}) problem (flexible job shop with 2 machines with no precedence relations and minimum makespan) which has already been studied and shown to be NP-hard or strongly NP-hard. For this reason a brute-force approach is impossible since there are already $O(n!^m)$ possibilities to solve a flexible job-shop with m machines and n jobs [2].

Besides, Graham's notation does not take geographic constraints into account, *i.e.* it is supposed that the polygonal zones to process are available *a priori* and more importantly, that these zones have areas compatible with the energy reserve of the assets. Yet the real problem is different: the user supplies large zones that cannot necessarily be processed by a single asset at one time. Thus these large zones must be subdivided (automatically) into sub-zones, depending on the maximum area an asset may cover, and this partition may be recomputed during the execution. This spatial dimension is an additional layer of complexity added to the MCM scheduling problem.

For this reason we decided to break our problem into more manageable sub-problems: how to deal assets on a zone and how to divide macro-zones into zones.

III. Problem Formalization

This paragraph details the problem at hand from a mathematical point of view. All the notations we introduce are summed up in an appendix at the end of the paper.

When describing the scenario, the operator supplies a polygon T corresponding to the total zone to clear. This polygon is called herein the "theatre" as it is too wide to be cleared at once.

A. *Environment*

We assume that environmental conditions such as the bathymetric profile, tide schedule, currents, etc., are known on the theatre and supplied as maps. The mine-warfare characteristics of the zone (densities of non-mine bottom objects, sedimentary profile, etc) are also assumed to be known.

B. *List of operations to clear the theatre*

For the theatre T to be cleared, three operations must be done sequentially on each subset of T. First, mine detection and classification is done, typically using sonar-based means. This operation yields a list of mine-like contacts (MILCOs). The identification operation consists in determining whether a MILCO is a mine or not. Typically this involves high-resolution imagery, either acoustic or optical cameras, or direct human intervention. Also typically a human will be in the loop, though human intervention may not necessarily be done in

real-time. The final operation is mine neutralization: either by detonating a charge near the mine, or defusing it.

C. Assets

A set of assets A = {a_k, k=1..m} (corresponding to "USV", "D/C UUV", *etc.*) is stored in a database. An asset a_k is described, among other parameters, by its environment adaptation (minimum and maximum depth, current, sea state, etc), but also the operations they can perform (Detection, Identification, Neutralization), and the associated performance parameters such as the total available energy E_k, transit speeds V_k^t, *etc*. Another parameter of interest is the cycle time T_k^c necessary for one launch-retrieval cycle: that is, the time taken to lay the asset in the water, retrieve it, recharge the batteries and perform other upkeep tasks; T_k^c is here assumed to be constant.

D. Macro-zone and zones

The theatre is then cut into polygonal macro-zones Z_i, i=1..N, either automatically or with the help of the operator, such that $T=\cup_{i=1..N} Z_i$. A macro-zone Z_i is a subset of the theatre with homogenous environmental and mine warfare characteristics, such that there are assets in the database able to clear this zone. A macro-zone may still be too large to be processed at one single time. Thus each macro-zone Z_i may be divided into N_i zones z_{ij} such that $Z_i=\cup_{j=1..Ni} z_{ij}$. The macro-zones and the zones are also described as polygons. The area of macro-zone Z_i (resp. zone z_{ij}) is denoted by $|Z_i|$ (resp. $|z_{ij}|$). It is up to a human operator, with the help of the software, to define the macro-zones. The software may issue warnings, *e.g.* when a macro-zone may be covered only for D/C operations but not I or N operations.

E. What this paper solves

We describe a new algorithm where the input is one polygonal macro-zone Z_i and a set of assets A = {a_k, k=1..m}. The algorithm finds a series of N_i zones z_{ij} and assigns to each zone z_{ij} one asset a_k. Each asset may be re-used several times if it is not an expendable asset; in this case the number of launches for each asset is determined as a side-product of the algorithm. The following constraints must be respected:

- the whole macro-zone is processed: $Z_i=\cup_{j=1..Ni} z_{ij}$;
- each asset assigned to a zone must be compatible with it, in particular the area of the zone must be less than the maximum area coverable by the asset;
- the shape of the zone should be "smooth", *i.e.* a "simple" polygon as close as possible to a rectangle, oriented along locally privileged, user-supplied orientations such as a current field; this is to facilitate the track computation for each zone, asset couple.

The algorithm as described here works under the hypothesis that the assets may be launched a potentially infinite number of times (no expendable assets).

IV. Dealing Assets To Macro-Zones

In this part we wish to determine which assets to use on a given macro-zone Z_i and if so, how many times they should be launched. Although we also considered the I/N case, we consider in this paper only the Detection operation, which is the easiest to describe. In this operation, the asset uses a regular sweeping pattern, as is typically the case with sonar where the asset works its way on the zone by following parallel tracks. To begin with, a subset $A_i = \{a_k, k=1..m_i \le m\}$ of A is determined: these are the Detection assets compatible with the environmental conditions on Z_i. This is easy. Then, we determine the maximum area, which can be processed with each asset in A_i. The third step is to determine the optimum number of launches for each asset $a_k \in A_i$, assuming the assets can be re-used an infinite number of times. This is equivalent to determining which area $S_{k,i}$ of Z_i will be processed by asset a_k. Those two last steps are harder and detailed here.

A. Maximum area $S_{k,i}^{max}$ an asset can cover on zone Z_i

The exact area cannot be computed yet since it depends on several parameters including the exact configuration of the zone. It is however possible to compute a "guesstimate" of an upper bound of this value. We assume the k-th asset of A_i has a battery reserve E_k (in W.h), the power used during the hunt is P_k^h, the transit speed is V_t^k, the power used during transit is $P_k^t(V_k^t)$ which is a function of the speed, the coverage rate (in km^2/h) is $C_k(Z)$; the battery slack is denoted by E_k^s (in W.h). We also found based on real trials that time lost for end-of-track manoeuvres accounted was usually proportional to the useful time spent hunting on the zone; p_k is the coefficient. Also, $D(Z_i)$ is the maximum distance from the launch point to any point of Z_i. Then:

- the energy required for the ingress and egress trip is:

$$E_k^t = 2\, P_k^t(V_k^t) \times D(Z_i)/\, V_k^t \quad (1)$$

- the energy required for the hunt on Z_i is:

$$E_k^h = P_k^h \times \left(S_{k,i}^{max} / C_k(Z_i)\right)(1+p_k) \quad (2)$$

- the total energy budget is:

$$E_k = E_k^t + E_k^h + E_k^s \quad (3)$$

- hence:

$$S_{k,i}^{max} = \left(E_k - 2\, P_k^t(V_k^t) \times D(Z_i)/\, V_k^t - E_k^s\right)\ldots \times C_k(Z_i) \,/\, \left(P_k^h \times (1+p_k)\right) \quad (4)$$

B. Estimating the number of launches

We want to select which asset to launch and how many times. If an asset $a_k \in A_i$ processes an area $S_{k,i} \le |Z_i|$, the asset is launched $n_{k,i} = \text{ceil}(S_{k,i}/\, S_{k,i}^{max})$ times. An upper bound for the time $T_{k,i}$ necessary to process this area $S_{k,i}$ is:

$$T_{k,i} = \left(S_{k,i} / C_k(Z_i)\right)(1+p_k) + T_k^c \times \text{ceil}(\, S_{k,i} / S_{k,i}^{max}) \quad (5)$$

Several cost functions are possible; the easiest is the maximum total time spent by any asset on the zone. Said otherwise, we want to find the areas $S_{k,i}$ and, as a vital side-product, the value of $n_{k,i} = \text{ceil}(S_{k,i}/\, S_{k,i}^{max})$, maximizing the following function:

$$g(S_{1,i}, S_{2,i}, \ldots, S_{m_i,i}) = \max_{k=1..m_i} T_{k,i} \quad (6)$$

We work under the constraint:

$$(\Sigma_k S_{k,i}) - |Z_i| \geq 0,\ k=1..m_i \quad (7)$$

which can be easily integrated into g by letting

$$S_{m_i,i} = |Z_i| - \sum_{k=1}^{m_i-1} S_{k,i} \quad (8)$$

Minimizing this function can be done using standard numerical methods provided that an initial estimate of the areas $S^0_{k,i}$ is known. This initial estimate can be obtained by dropping the "ceil" function in (5). Then:

$$T'_{k,I} = S^0_{k,i} \times \left((1+p_k) / C_k(Z_i) + T_k^c / S_{k,i}^{max}\right) = S^0_{k,i} \times \alpha_{k,i} \quad (9)$$

Now let g' be the equivalent of g in (6), but using times $T'_{k,i}$. It is evident that g' is minimized when terms $T'_{k,I}$'s are equal for all k. In this case:

$$\forall\ k=1..m_i \quad S^0_{k,i} \times \alpha_{k,i} = S^0_{1,i} \times \alpha_{1,i} \quad (10)$$

Adding constraint (7) yields:

$$S^0_{1,i} \times \alpha_{1,i} \times \Sigma_k (1/\alpha_{k,i}) = |Z_i| \quad (11)$$

Hence, $S^0_{k,i}$ is a weighted average of $|Z_i|$, with weight $1/\alpha_{k,i}$:

$$\forall\ k=1..m_i\ ,\ S^0_{k,i} = |Z_i| \times [\ 1/\alpha_{k,i}] \ / \ \Sigma_k (1/\alpha_{k,i}) \quad (12)$$

V. CUTTING A MACRO-ZONE INTO ZONES

A. Constraints

The input consists in a macro-zone Z_i, assets $a_{k,i}$, $k=1..m_i \leq m$ compatible with this macro-zone, and an optional vector field F covering Z_i. Field F is typically a map of currents on Z_i. Indeed, if a track is in the direction of the current, drift is minimized and side-scan sonar coverage is maximized. The user may alternatively input a vector field defining locally privileged directions, taking into consideration other operational considerations than currents, which the survey tracks should ideally follow. This vector field may be degenerated to a single direction everywhere. Also, the areas $S_{k,i}$ assigned to each asset a_k are known. For each macro-zone Z_i, we need a list of sub-zones/asset couples $L_i=\{(z_{ij},\ a_{kj})\}$, $j = 1..N_i$. These are the constraints:

- the zone is compatible with the asset, *i.e.* if z_{ij} is associated to asset $a_{k,i}$, then $|z_{ij}| \leq S_{k,i}^{max}$;
- the whole macro-zone is covered: $\Sigma_j |z_{ij}| = |Z_i|$;
- the zones z_{ij} are as regular as possible, *i.e.* "similar" to rectangles, parallelograms, trapezes, this to facilitate track computation on z_{ij} and limiting the number turns at the end of tracks;
- the zones z_{ij} are oriented according to the field F.

B. Privileged Direction Curve

If F is supplied, a privileged direction curve (PDC) is computed. The PDC corresponds to the trajectory followed by a particle carried by the stream defined by vector field F. The PDC should be as long as possible while being on (an approximation of) the medial axis of the polygon, as far as possible from any edge and closest to the centre of its bounding box; also the derivatives of the PDC must be continuous. Many strategies are possible to find the PDC. One of them begins by computing[1] several stream lines with different start points; the best of them is chosen as the PDC. If the stream line goes out of the polygon at some point A and comes back in at some other point B, then the vertices from A to B are substituted to the stream line between A and B and the resulting line is smoothed by a low-pass filter, ensuring the curve is derivable (*i.e.* has an orthogonal line everywhere).

If F is not supplied, then the PDC is the longest axis of the oriented bounding box of the zone, typically found using the rotating calipers algorithm [3].

C. Recursive cutting

Let O_i be a list of area/assets objectives$\{\ (S_{k,j,i},\ a_{k,i})\ \}$, where $k = 1..m_i$, and for all values of $j = 1..\ n_{k,i}$, term $S_{k,j,i}$ is equal to $S_{k,i}^{max}$. The terms $n_{k,i}$ are those obtained after the optimization stage exposed in section IV.B.

The process is recursive, where polygon Z_i is cut in two, then each sub-polygon is also cut into two, *etc.*, until the final sub-polygon has an area less than one of the areas $S_{k,j,i}$. The input of the recursive process is:

- a sub-polygon z of Z_i;
- a sublist l of O_i with (area, asset) couples;

The output is:

- a list L_i made of (zones of Z_i, asset) couples;
- at most two sub-polygons z_0 and z_1 such that $z_0 \cup z_1 = z$;
- at most two sublists l_0 and l_1 such that $l_0 \cup l_1 = l$;

The process is initialized with $z = Z_i$, $l = O_i$, and $L_i = \varnothing$. First, split l into sub-lists l_a and l_b so that the sum of the areas in l_a, denoted by $|l_a|$, is as close as possible to the sum of the areas in l_b, which we write $|l_b|$. A cutting line C, orthogonal to the PDC of z, is then found so that it cuts z into two parts z_0 and z_1. Part z_0 must be adjusted to target either area $|l_a|$ or $|l_b|$ and both possibilities must be tried because one of them often does not work out. Assuming $|z_0| = |l_a|$, we set list l_0 to l_a; the area $|z_1| = |z| - |z_0|$ will necessarily be less than $|l_b|$, due to constraint (7). The converse situation occurs if $|z_i| = |l_b|$. Finding C may be efficiently done by dichotomy, iterating on the curvilinear abscissa of C on the PDC.

Now, if l_0 contains only one item $(S_{k,j,i},\ a_{k,i})$, then add $(z_0,\ a_k)$ to list L_i. Do the same for l_1, if necessary. On the other hand, if l_0 or l_1 contains more than one item, then reiterate recursively the same algorithm as used for *(z, l)*. It is possible to alternate cutting directions: in this variant, every two iteration, use the orthogonal field $F^{\perp}$ to F, instead of F.

VI. A WORKED EXAMPLE

In this section we consider the case where the macro-zone Z_i is described by the coordinates given in table I (see also figure 2). The total area $|Z_i|$ of this polygon is 4 units.

[1] Using MATLAB, this can be done using the `stream2` function.

TABLE I. POLYGON COORDINATES

x	-2	-1	1	2	2	1	-1	-2
y	-2	-1	-1	-2	-1	0	0	1

The field F is described by table II:

TABLE II. PREFFERED DIRECTION FIELD F

x	-2	0	2
y	-1.5	-0.5	-1.5
θ (deg)	45	90	135

A dummy list of assets $a_{k,i}$ compatible with Z_i is supplied in table III:

TABLE III. ASSETS COMPATIBLE WITH POLYGON

Asset $a_{k,i}$	1	2	3	4
Max. coverable area per launch $S_{k,i}^{max}$	0.5	0.3	0.4	0.35

The algorithm described in paragraph IV.B returns:

TABLE IV. NUMBER OF LAUNCHES FOR POLYGON

Asset $a_{k,i}$	1	2	3	4
Number of launches $n_{k,i}$	4	2	2	2

As a result of table III and IV the assets/area list is the following:

O_i = {(1, 0.5), (1, 0.5), (1, 0.5), (1, 0.5), (2, 0.3), (2, 0.3), (3, 0.4), (3, 0.4), (4, 0.35), (4, 0.35) }

It is easy to see that the area covered by the assets during all their missions is 4×0.5+2×0.3+2×0.4+2×0.35 = 4.1 area units, i.e. just a bit more than $|Z_i|$, as expected to respect constraint (7). We initialize the algorithm: $l \leftarrow O_i$.

Following the method exposed above, list l is divided at the first iteration into two sub-lists with approximately the same total area:

l_a = {(1, 0.5), (1, 0.5), (1, 0.5), (1, 0.5) }; $|l_b| = 2$;

l_b = {(2, 0.3), (2, 0.3), (3, 0.4), (3, 0.4), (4, 0.35), (4, 0.35) }; $|l_b| = 2.1$.

The PDC is found by interpolating F on a vector grid of 10×10 vectors evenly distributed over x = [-2, 2] and y = [-2, 0], then computing the stream line going through the center of Z_i which is found at (0, -0.5). The cutting line is parameterized by curvilinear abscissa s between 0 (left) and 1 (right). The PDC for the first iteration is shown in figure 2.

Now let us try to target either $|l_a|$ or $|l_b|$ as the area for the "left" polygon (where the stream line has abscissas below s):

- Target: $|l_a| = 2$: $s = 0.5$, $|z_0| = 2$, $|z_1| = 2 \leq |l_1|$;
- Target: $|l_b| = 2.1$: $s = 0.58$, $|z_0| = 1.24$, $|z_1| = 2.75 > |l_1|$;

Clearly the first case is the correct one, so we set $l_0 = l_a$. The process goes on recursively, until the list l is reduced to one element or empty. The final partition is given by table V and illustrated in figure 3.

In table V, the total areas and the allocation follow the constraint set by list L_i^0. There are some numerical deviations from the ideal value caused by the dichotomy process when finding the curvilinear abscissa s; these deviations are of negligible importance. In figure 3, notice how the cutting lines follow the user-supplied field F and how the partition yields regular polygons. Note that in this case the cutting directions were alternated every two iteration, as can be seen on the right for the zones processed by asset 2 and asset 4.

TABLE V. FINAL RESULT (SEE ALSO FIGURE 3)

Zone z_{ij}: value of j	**Asset $a_{k,i}$: value of k**	**Zone area**	**Maximum area for asset $S_{k,i}^{max}$, as per table III**
1	1	0.500005	0.5
2	1	0.500002	0.5
3	1	0.500000	0.5
4	1	0.499984	0.5
5	3	0.399998	0.4
6	3	0.400006	0.4
7	4	0.349990	0.35
8	4	0.350007	0.35
9	2	0.247365	0.3
10	2	0.252643	0.3
TOTAL AREA		**4.000000**	**4.1**
TRUE POLYGON AREA		**4.000000**	

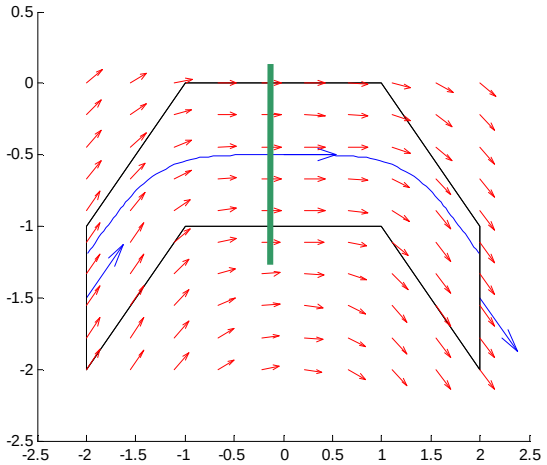

Figure 2: polygon Z_i described in table I. The field F (table II) is given in large blue arrows; the interpolated vector field is given with small red arrows; the stream line is in blue; the cutting line C for the first iteration is the fat green line.

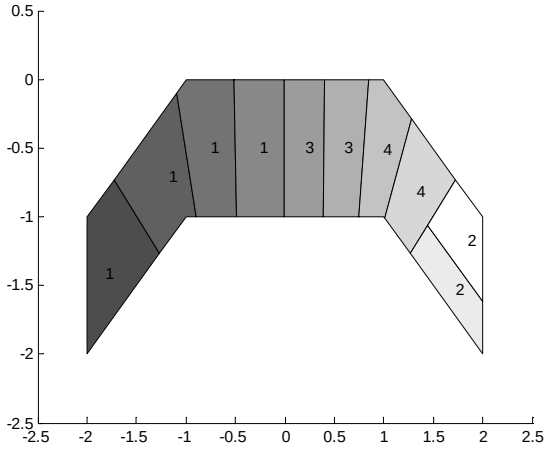

Figure 3: result of the cutting as supplied by the algorithm. The numbers correspond to asset types $a_{k,i}$.

VII. CONCLUSIONS

Mine warfare mission planning on a theater combined with assets assignment to perform elementary mine warfare tasks is a tough job because the problem mixes space and time constraints. As such, existing operational research methods cannot solve the problem and new approaches must be devised. In this paper, we formalized the MCM asset assignation and zone-cutting problem from a mathematical point of view for non-expendable assets (the expendable case can be adapted). We then proposed a method working in two steps, which first determines which assets are to be used and how many times they should be launched; then shows how to cut a polygon into a series of zones, which can be associated to assets. These zones have a smooth shape respecting local constraints expressed as a vector field, such as the current direction. Once implemented in a CMS, these tools will help mine warfare officers to prepare various missions by optimizing the assets to deploy and bring the capability to manage fleet of different UxVs on large theaters.

NOTATIONS

$\lvert\bullet\rvert$	Total area (of a polygon, or a list of polygons)
T	Theatre
A={a_k, k=1..m}	List of assets
Z_i, i=1..N	i-th macro-zone such that T=$\cup_{i=1..N}$ Z_i
z_{ij}, j=1..N_i	j-th zone of Z_i such that Z_i=$\cup_{j=1..Ni}$ z_{ij}
E_k^t	Energy needed for asset a_k for the ingress/egress trip
E_k^h	Energy needed for asset a_k for hunting
E_k^s	Battery slack for security purposes, asset a_k
E_k	Total energy reserve of asset a_k
P_k^h	Power used when hunting for asset a_k
$P_k^t(V_k^t)$	Power used when transiting for asset a_k
$C_k(Z_i)$	Coverage rate for asset a_k
$D(Z_i)$	Maximum distance from launch point to any point of Z_i
$S_{k,i}$	Total area processed on Z_i by asset a_k (several launches may be required)
$S_{k,i}^{max}$	Maximum area which can be processed by asset a_k on zone Z_i for one launch/retrieval cycle
$n_{k,i}$	Number of launches required for asset a_k to process Z_i
$T_{k,i}$	Upper bound for the total time needed to process zone Z_i using only asset a_k
F	Vector field describing preferred directions
$F^{\perp}$	Vector field orthogonal to F
O_i = { ($S_{k,j,i}$, $a_{k,i}$) }, k=1..m_i, j=1.. $n_{k,i}$	Objective area/asset distribution on macro-zone Z_i
C	Cutting line
s	Curvilinear abscissa of the cutting line
L_i= { ($z_{j,i}$, $a_{k,i}$) }, k=1..m_i, j=1.. $n_{k,i}$	Asset / zone distribution for macro-zone Z_i (“what we want”)

REFERENCES

[1] R.L Graham., E.L Lawer., J.K Lenstra., A.H.G. Rinnooy Kan, “Optimisation and approximation in deterministic sequencing and scheduling: A survey”. *Ann. Discr. Math.*, 5:287-326, 1979.

[2] H. Liu., A. Abraham, Z. Wang. “A multi-swarm approach to multi-objective flexible job-shop scheduling problems”. *Fundamenta Informaticae*, 95:1-25, 2009.

[3] G. Toussaint, “Solving Geometric Problems with the Rotating Calipers”. *Proc. IEEE MELECON 1983*, Athens, Greece, May 1983. http://www-cgrl.cs.mcgill.ca/~godfried/publications/calipers.ps.gz