Nationaal Lucht- en Ruimtevaartlaboratorium
National Aerospace Laboratory NLR

NLR-TP-98260

On vehicle allocation to targets in mission planning

R. Choenni

Nationaal Lucht- en Ruimtevaartlaboratorium

National Aerospace Laboratory NLR

NLR-TP-98260

On vehicle allocation to targets in mission planning

R. Choenni

This report is based on an article to be published in the Proc. RTA SCI 9th Symposium on the Application of Information Technologies to Mission Systems, 1998, by NATO RTA.

Division:	Informatics
Issued:	June 1998
Classification of title:	unclassified

Summary

A military mission consists of a large number of activities, such as launching weapons to targets, deploying vehicles to targets and pointing out an efficient route for each vehicle, etc. Planning plays an important role in many of these activities. Despite the apparently different nature of a plan in each activity, we feel that many of these plans can be captured in one formalism. In this paper, we present such a formalism and exploit it to solve a planning problem. Our formalism is based on the well-known notions of sets and lists. A plan can be regarded as a list of operations on sets of objects. The problem to deploy vehicles to targets with acceptable costs will be expressed in this formalism. To solve this problem, we propose a two step approach. In the first step, we generate lists of targets that are feasible for each vehicle. In the second step, we select for each vehicle at most one list, such that, finally, all objectives for the mission concerning the allocation of vehicles to targets are attained. In our approach, we have taken advantage of the fact that the cost function for the chosen problem is monotonically non-decreasing as well as from heuristics that are used by experts.

Contents

1 Introduction 5

2 Problem definition 7

3 Mathematical model 9

4 Approach 12

4.1 Plan selection 12

4.2 Plan updating 16

5 Application 19

6 Conclusions & further research 21

(22 pages in total)

1 Introduction

Planning is an essential requisite to fulfill (complex) military missions successfully [Ref 1]. The purpose of planning is to generate and maintain efficient plans in order to guide a mission. Important planning activities are, among others, determination of an efficient route for a vehicle and allocation of a set of vehicles to a number of targets. Although the nature of these activities is apparently different, many of these activities can be captured in one formalism. In this paper, we present such a formalism and exploit it to solve a problem in the field of mission planning.

The problem addressed in this paper is the following: *given a set of vehicles, e.g., aircraft, located on geographically different bases, and a number of targets with relevant characteristics; deploy these vehicles to the targets in an efficient way.* In the following we refer to this problem as Vehicle Mission Problem (VMP). While we assume that relevant information with regard to the vehicles and bases is precisely known, we do not make this assumption for the targets. Information with regard to a target is the product of intelligence gathering, and may be updated frequently. As a consequence, a plan should be revised in an efficient way whenever an update occurs.

Since the VMP is NP-complete, there exists no technique that solves the problem in polynomial time. To attack the problem, we have developed a formal model in terms of a cost function, which takes as input a plan and computes its associated cost. We were able to prove that this function is monotonically non-decreasing. We exploit this property in combination with domain knowledge to search efficiently for a solution for VMP, resulting into a two step approach. In the first step, we generate for each vehicle the lists of targets that are appropriate for this vehicle. In the second step, we select for each vehicle at most one list such that, finally, all objectives for the mission are attained with acceptable cost. Theoretically, both steps have an exponential complexity. Therefore, we have derived a number of rules to control this complexity. Furthermore, we present a procedure to update a plan adequately whenever previously unknown information becomes available.

A number of efforts has been reported to solve problems similar to the one we have addressed, see among others [Ref 1, 2]. In these efforts, more or less the following approach is followed for this type of problem. A problem is typically formulated as an optimization problem in a ”black box” way, i.e., in the form of *optimize a function f(.), which is subjected to a set of constraints*. Then, an algorithm is selected or developed to search for a solution. In [Ref 4], these algorithms are categorized as algorithms that guarantee an optimal solution and algorithms that do not guarantee this. The algorithms in the former category are mainly based on (mathematical) classical techniques, such as dynamic programming, gradient methods, etc., while the algorithms of the latter category are mainly based on more novel techniques, such as neural networks, knowledge base technology, etc.

Our approach differs on three main points from above-mentioned approach. First, we have chosen a formalism to express a problem that is better accessible for domain experts than the black box formulation. This allows to add domain knowledge and modifications to a plan easily. For example, in our formalism we can easily express relationships as ”Action A should be followed/preceded by action B”, while the black box does not explicitly support this kind of relationships. Second, often, a planning problem is divided into two subproblems, namely an allocation and a scheduling problem [Ref 3]. In our approach, we do not make this difference since allocation and scheduling may strongly related to each other. Third, we have integrated the strong points of classical and more novel techniques to solve VMP. We have taken advantage of the fact that the cost function for VMP is non-decreasing as well as from heuristics that are used by experts in this field. In most efforts, either a classical or a more novel technique is used to solve the problem.

The remainder of this paper is organized as follows. In Chapter 2, we describe VMP in more detail, and in Chapter 3, we derive a mathematical model for the problem. Then, in Chapter 4, we discuss our approach to select plans and to update plans according to previously unknown information. In Chapter 5, we show how our approach can be applied to solve a problem in the naval domain. Preliminary results based on this application will be discussed as well. Finally, Chapter 6 concludes the paper.

2 Problem definition

In this chapter, we define VMP in more detail. In a military setting, vehicles, e.g., aircraft, are located on geographically different bases. These vehicles should be deployed to targets. The allocation of vehicles to targets and in which order they are deployed to targets are expressed in a plan.

While relevant data with regard to each vehicle are precisely known, this may be not true for targets. Information with regard to a target is the product of continuous intelligence gathering, and may be frequently updated. As a consequence, whenever information with regard to a target is updated, the plan should be updated according to the new information. Our goal is to contribute to an efficient selection and update of plans.

For the time being, we assume that a vehicle can be deployed for at most one list of targets in a plan, i.e., a vehicle appears at most once in a plan. Before elaborating our problem, we define the notion of a plan more precisely. For this purpose, we use the well-known set and list notations. Perhaps unnecessarily, we note that the order of elements in a set is irrelevant and that a list can be regarded as an ordered multi-set (thus allowing duplicates).

Definition 1: Let $\mathcal{V} = \{v_1, v_2, v_3, ..., v_m\}$, be a set of vehicles, $\mathcal{T} = \{t_1, t_2, t_3, ..., t_n\}$, a set of targets, $m, n \in I\!N$, and $X_h = \{(V_{h,1}, \mathbf{T}_{h,1}), (V_{h,2}, \mathbf{T}_{h,2}), (V_{h,3}, \mathbf{T}_{h,3}), ..., (V_{h,k}, \mathbf{T}_{h,k})\}$, $h \in I\!N$;, in which $V_{h,i} \subseteq \mathcal{V}$ and $\forall i, j; i \neq j : V_{h,i} \cap V_{h,j} = \emptyset, 1 \leq i, j \leq k$, and $\mathbf{T}_{h,i} = [t^1_{h,i}, t^2_{h,i}, t^3_{h,i}, ..., t^l_{h,i}]$, $t^f_{h,i} \in \mathcal{T}$, $1 \leq l \leq n$.

Then, a plan P is defined as $P = [X_1, X_2, X_3, ..., X_s]; \forall X_p, X_q; 1 \leq p, q \leq s; p \neq q : x \in X_p, y \in X_q \Rightarrow x.V_{p,i} \cap y.V_{q,j} = \emptyset$.

In the following, an X_h will be called an *action* set.

Example 1 Let us consider a plan $P = [\{(\{v_1, v_2\}, [t_1, t_2])\}, \{(\{v_3\}, [t_3])\}, \{(\{v_4\}, [t_5, t_6])\}]$. Note that P consists of three action sets namely, $X_1 = \{(\{v_1, v_2\}, [t_1, t_2])\}$, $X_2 = \{(\{v_3\}, [t_3])\}$, and $X_3 = \{(\{v_4\}, [t_5, t_6])\}$. According to this plan, v_1 and v_2 (simultaneously) should successively destroy targets t_1 and t_2. Then, v_3 should destroy t_3. Finally, v_4 should destroy targets t_5 and t_6, successively. If it is not necessary that v_4 starts its actions after that v_3 has completed its actions, this can be expressed by taking the union X_2 and X_3 in plan P. So, $P = [\{(\{v_1, v_2\}, [t_1, t_2])\}, \{(\{v_3\}, [t_3]), (\{v_4\}, [t_5, t_6])\}]$. □

Two main factors can be distinguished in the selection of plans namely, the effectiveness of a plan and the cost of a plan. The effectiveness of a plan depends on the extent that a defined goal is met. A plan that totally meets a goal is called a *complete* plan.

We set ourselves as goal to select complete plans with acceptable cost with regard to a given set of targets. Furthermore, whenever information with regard to a target is updated, the selected plan is updated according to the new information.

The cost of a plan is determined by parameters, such as fuel consumption, types and number of vehicles involved in a plan, etc. However, in general, a plan P according to which a target t should be destroyed by vehicle v will be cheaper than any plan P' that includes plan P. This will be used to control the search space of plans. In the following chapters, we formalize this observation and exploit it in searching for plans.

3 Mathematical model

In this chapter, we define a mathematical model for the problem introduced in the previous chapter, and prove some properties for this model. On the basis of these properties, we introduce, in the next chapter, a solution for the problem.

As discussed in the foregoing, our goal is to select complete plans with acceptable cost. In order to fulfil this task, we define a cost function to compare different plans. The cost function is defined in such a way that incomplete plans get ∞ as cost value. Before defining the cost to move to a target t by vehicle v, we define under which conditions a target can be attained by a vehicle.

Definition 2: Let $\mathcal{B} = \{b_1, b_2, b_3, ..., b_m\}$ be a set of bases. Then, a list of targets $\mathbf{T} = [t_1, t_2, t_3, ..., t_n]$ is *attainable* by a vehicle v_j, which is located at a base $b_j, j \leq m$, if

1. v_j has the capacity to bridge the distance between b_j and all targets of $\mathbf{T}$ in the given sequence (without any stopovers), and afterwards
2. There exists a $b_h \in \mathcal{B}$, such that $C_{\text{base}}(t_n, b_h) = min_{b_l \in \mathcal{B}} C_{\text{base}}(t_n, b_l)$, in which $C_{\text{base}}(t, b) \geq 0$ represents the *minimal* cost that is required by a vehicle to move from target t to base b, and v_j has the capacity to bridge the distance between t_n and b_h.

Note, b_h is the base that can be reached with the lowest cost by a vehicle from the last target in a target list.

The cost (which assumes values ≥ 0) involved in attaining a list of target T by a vehicle v starting from a base b is defined as follows:

$$C_{\text{att}}(v, b, \mathbf{T}) = \begin{cases} f_{v,b,\mathbf{T}} & \text{if } \mathbf{T} \text{ is attainable by } v \\ \infty & \text{otherwise} \end{cases}$$

Once a vehicle attains a target, some actions should be taken to destroy the targets. This cost depends on the actions and weapons used for these actions. If a vehicle is unable to perform these actions, e.g., it is not able to carry the suitable weapons, the cost is defined as ∞. Let us define the cost (assuming values ≥ 0) involved with performing actions on a target t by a vehicle v as follows:

$$C_{\text{action}}(v, t) = \begin{cases} c_{v,t} & \text{if } v \text{ is able to perform the} \\ & \text{defined actions above } t \\ \infty & \text{otherwise} \end{cases}$$

The general cost function for a plan P is defined as:

$$C(P) = \sum_{X_i \in P} \sum_{x \in X_i} \Big(\sum_{v_k \in x.V} (C_{\text{att}}(v_k, b_k, \mathbf{T}_i) + \sum_{t_l \in \mathbf{T}_i} C_{\text{action}}(v_k, t_l)) \Big)$$

Before showing that function C is monotonically non-decreasing, we note the following with regard to this function. The cost of a plan does not depend on the order in which the action sets of a plan are performed. We feel that the order of action sets becomes significant if a vehicle is allowed in more than one action set of a plan. In that case a vehicle has various alternatives with probably different cost to perform its action sets. Since we have assumed that a vehicle appears at most once in a plan, we feel that the order of action sets can be neglected. However, the extension of our cost function, such that it is able to take the order of action sets into account, is straightforward. For example, one may introduce a penalty function for two action sets that are not in a proper order in a plan. A penalty function C_{pen} takes two action sets X_i and X_j of a plan as input, and produces a penalty in terms of cost if they are in an inappropriate order, i.e., if X_i precedes X_j while the reverse is desired, and zero otherwise. Let the penalty due to undesired orders on a plan be $S(P) = \sum_{X_i \in P} \sum_{X_j \in P, i \neq j} C_{\text{pen}}(X_i, X_j)$. Then, function C can be extended to a function $\tilde{C}$ that takes the order of action sets into account as follows: $\tilde{C}(P) = C(P) + S(P)$.

Although a plan can be regarded as a set of action sets instead of a list since we consider function C, we still regard a plan as a list. As illustrated above, function C can be easily extended such that the order of action sets has its impact on the cost of a plan. Therefore, we anticipate on this situation and regard a plan as a list of action sets in solving VMP. Note that regarding a plan as a set of action sets simplifies VMP.

Let us return to function C. In the following, we prove some properties for this function.

Proposition 1: The cost function C is monotonically non-decreasing with regard to a target list **T**.

Proof: We prove consecutively that C_{att} and C_{action} are monotonically non-decreasing. Let $\mathbf{T} = [t_1, t_2, ..., t_i]$ be a list of targets attainable by a vehicle v that starts from a base b, and b_j^k the cost to move from target t_k to the j-th base. The cost involved in attaining the last element of **T**, i.e., t_i, is given by $C_{\text{att}}^{-}(v, b, \mathbf{T})$. Then,

$$C_{\text{att}}^{-}(v, b, \mathbf{T}) = C_{\text{att}}(v, b, \mathbf{T}) \Leftrightarrow b_h^i,$$

in which $b_h^i = \min_{b_l \in \mathcal{B}} C_{\text{base}}(t_i, b_l)$ as discussed in Definition 2.

Let $\mathbf{T}'$ be a list containing **T** to which a target t_k has been added, i.e., $\mathbf{T}' = [t_1, t_2, ..., t_i, t_k]$. The cost to move from t_i to t_k is represented by $C_{\text{att\_ptp}}(t_i, t_k)$. Then,

$$C_{\text{att}}(v, b, \mathbf{T}') = C_{\text{att}}^{-}(v, b, \mathbf{T}) + C_{\text{att\_ptp}}(t_i, t_k) + b_p^k$$

in which $b_p^k = \min_{b_l \in \mathcal{B}} C_{\text{base}}(t_k, b_l)$.

Then, $C_{\text{att}}(.)$ is monotonically non-decreasing if

$$\begin{aligned} C_{\text{att}}(v, b, \mathbf{T}') \geq C_{\text{att}}(v, b, \mathbf{T}) \quad &\Leftrightarrow \\ C_{\text{att\_ptp}}(t_i, t_k) + b_p^k \geq b_h^i \quad &\Leftrightarrow \\ C_{\text{att\_ptp}}(t_i, t_k) + b_p^k \geq b_p^i \end{aligned} \tag{1}$$

Then, by Definition 2, equation (1) holds. Hence, C_{att} is monotonically non-decreasing. (Note, if the cost to move from t_i, via t_k, to a base would be smaller than b_p^i, this would be in contradiction with Definition 2 item 2, since b_p^i is the minimal cost to base b_p from target t_i.)

Let $C_{\mathbf{T}} = \sum_{t_l \in \mathbf{T}} C_{\text{action}}(v, t_l)$ be the cost involved in performing the actions on each target in the target list **T** by vehicle v. Then, $C_{\mathbf{T}'} = C_{\mathbf{T}} + C_{\text{action}}(v, t_k)$. It should be clear that $C_{\mathbf{T}'} \geq C_{\mathbf{T}}$, since $C_{\text{action}} \geq 0$. Hence, C_{action} is monotonically non-decreasing.

Since C_{att} and C_{action} are both monotonically non-decreasing, cost function $C(.)$ is monotonically non-decreasing. $\square$

Corollary 1: Let L and L' be the lists of $X_i's$ corresponding to a plan P and P', respectively. Plan P' subsumes plan P, $P \sqsubseteq P'$, if L is a sublist of L'. Then, $\forall P \sqsubseteq P' : C(P) \leq C(P')$

Proof: Trivial $\square$.

In the next chapter, we exploit this result in solving the before-mentioned problem.

4 Approach

This chapter is devoted to the generation and maintenance of plans. In Section 4.1, we introduce a two step approach to select complete plans, in which vehicles are deployed to targets, with acceptable cost. It is up to the expert to decide whether the cost of a plan is acceptable or not. For example, an expert may define a threshold value for the cost of a plan, i.e., all plans that have a lower cost than the threshold value are acceptable. Since the information with regard to a target may be uncertain and/or incomplete, a plan may become invalid or to expensive whenever new information becomes available. In Section 4.2, we address this issue.

4.1 Plan selection

As noted already, we propose a two step approach to select plans that deploy vehicles to targets. In the first step, we generate lists of targets attainable by each vehicle. In the second step, we select for each vehicle at most one list such that the result will be a complete plan with acceptable cost. Theoretically, both steps have an exponential complexity. Therefore, we introduce a number of rules to control the complexity. Let us elaborate the steps in more detail.

Step 1: To generate lists of attainable targets by a vehicle, we rely on the following observations. First, not all vehicles will be suited to each target. Depending on the target characteristics, one can decide whether a vehicle is a candidate to be deployed to a target or not. This information is provided by an expert or by a database. Once this information is available, we generate lists of targets that is longer than 1, taking into account the following principles:

P1: If a list **T** is *not* attainable by a vehicle v, then each $\mathbf{T}'$ that contains **T** is also not attainable by vehicle v.

P2: If a list **T** is attainable by a vehicle v, then each list $\mathbf{T}^-$ that is a sublist of **T** is attainable by vehicle v.

The rationale behind these principles follows directly from Definition 2.

P3: If a vehicle v is not able to perform the actions on a list **T**, i.e., $\sum_{t \in \mathbf{T}} C_{\text{action}}(v, \mathbf{T}) = \infty$, then v is *not* able to perform the actions on any list $\mathbf{T}'$ that contains **T**.

P4: If a vehicle v is able to perform the actions on a list **T**, then v is able to perform the actions on any list list $\mathbf{T}^-$ that is a sublist of **T**.

The rationale behind these principles follows from the fact that C_{action} is monotonically non-decreasing.

Let us illustrate how these principles are used to identify attainable lists for a vehicle that is able to perform the actions on the lists. In the following, such a list is called a *feasible* list for a vehicle.

Example 2 Consider a vehicle v that is deployable to the targets t_1, t_2, and t_3. In Figure 1, it is illustrated how all lists may be generated with regard to the three targets[1]. In general, the number of lists for n targets is $\sum_{k=1}^{n} k! \begin{pmatrix} n \\ k \end{pmatrix}$. Suppose that $[t_1], [t_2]$, and $[t_3]$ are feasible lists for

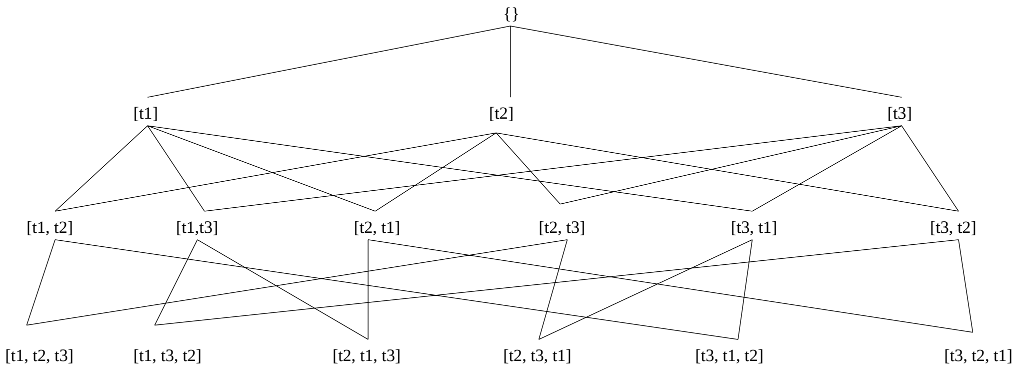

Fig. 1 Generation of target lists

vehicle v. Then, we choose two other lists $\mathbf{T}_1$ and $\mathbf{T}_2$ from Figure 1, such that neither $\mathbf{T}_1$ is a sublist of $\mathbf{T}_2$, nor $\mathbf{T}_2$ is a sublist of $\mathbf{T}_1$, and decide whether the lists are feasible for v or not. Let assume that the chosen lists are $[t_1, t_2]$ and $[t_3, t_2, t_1]$. Suppose that $[t_1, t_2]$ appears not to be attainable by vehicle v, (and, therefore, not feasible for v), and $[t_3, t_2, t_1]$ appears to be feasible for v. This means that all lists that have $[t_1, t_2]$ as sublist, i.e., $[t_1, t_2, t_3]$ and $[t_3, t_1, t_2]$, are not attainable according to principle P1, and, therefore, not feasible for v.

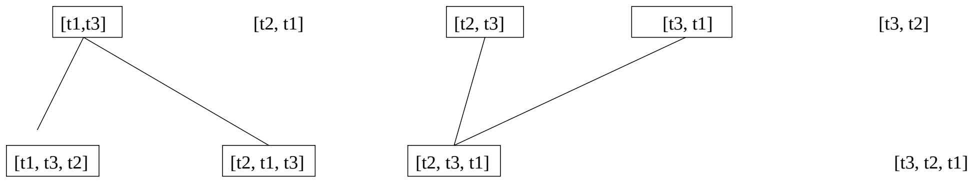

Fig. 2 Reduced set of candidate target lists

According to principle P2, all sublists of $[t_3, t_2, t_1]$, i.e., $[t_3, t_2]$ and $[t_2, t_1]$ are attainable lists and according to P4 vehicle v is also able to perform the defined actions on these lists. Hence, the lists $[t_3, t_2]$ and $[t_2, t_1]$ are feasible for v. The lists that are candidate for further consideration are surrounded by a box in Figure 2.

[1]The notation in the figure slightly differs from notation in the text. t*i*, in which *i* is a number, in the figure corresponds to t_i in the text.

Let us investigate the lists $[t_1, t_3]$ and $[t_2, t_3, t_1]$. Suppose that it appears that v is not able to perform the defined actions on $[t_1, t_3]$ and that $[t_2, t_3, t_1]$ is feasible. Then, according to P3, v will not be able to perform the defined actions on the lists $[t_1, t_3, t_2]$ and $[t_2, t_1, t_3]$. Hence, these lists are not feasible for v. Since $[t_2, t_3, t_1]$ is feasible, the lists $[t_2, t_3]$ and $[t_3, t_1]$ are feasible as well according to P2 and P4.

So, feasible lists for v are: $[t_1], [t_2], [t_3], [t_2, t_1], [t_2, t_3], [t_3, t_1], [t_3, t_2], [t_2, t_3, t_1]$, and $[t_3, t_2, t_1]$.
$\square$

Once all feasible lists of targets have been generated for each vehicle, we check if all targets are handled by the available vehicle. If this is the case, we execute Step 2 (see below). Otherwise, we select for each target that can not be handled by a single vehicle, the sets of vehicle that are capable to handle this target. Again, we will use variants of principle P1 and P2 to identify these sets. Namely, the following holds: if a set of vehicles V is able to handle a target t, then each superset V' of V is able to handle target t, and if a set of vehicles V is not able to handle a target t, no subset V^- of V is able to handle target t. However, we will consider in our approach for each target the sets with the minimal number of vehicles. In Table 1, an example of the output of Step 1 is given (for further clarification see Step 2).

We note that the variant of P2 helps in concluding that no complete plan exists. Suppose that the set $\mathcal{V} = \{v_1, v_2, v_3\}$ of vehicles is available, and that the set V is not capable in handling one of the targets t. Then, target t can not be handled by any set of vehicles. Hence, there exists no complete plan.

Step 2: In Step 2, a tree is generated on the basis of the results of Step 1. Each node in the tree represents an X_i, and paths in a tree represents a plan. Since Proposition 1 holds, the paths corresponding to complete plans is generated according to the branch and bound technique. Let us describe the actions involved in generating the tree.

Consider a set $\{X_1, X_2, X_3, ..., X_n\}$, in which $X_i = \{(V_i, T_i) | i = 1, 2, .., \}$ and T_i is a feasible list for A_i.

1. A random plan that satisfies to Definition 1 is generated, and its cost is computed. This cost will be used to bound parts of the tree and will be referred as *b_cost*.
2. The root of the tree is $X_0 = (\{\}, [])$, and $X_1, X_2, X_3, ..., X_n$ represents the nodes on level 1 of the tree. Then, for each node X_i, we generate a subtree according to action 3.
3. A node X_k is added to a node X_j of a plan $P = [X_0, X_1, ..., X_j]$ as long as the new plan is

v_1 : $[t_1], [t_2], [t_3], [t_2, t_1], [t_2, t_3], [t_3, t_1], [t_3, t_2], [t_2, t_3, t_1], [t_3, t_2, t_1]$	$\rightarrow X_1, X_2, X_3, ..., X_9$
v_2 : $[t_1], [t_2], [t_1, t_2], [t_2, t_1]$	$\rightarrow X_{10}, X_{11}, X_{12}, X_{13}$
v_3 $[t_1], [t_3], [t_3, t_1]$	$\rightarrow X_{14}, X_{15}, X_{16}$
v_4 : $[t_1], [t_2], [t_3], [t_5], [t_2, t_5], [t_3, t_5]$	$\rightarrow X_{17}, X_{18}, X_{19}, ...,$ X_{22}
$t_4 : \{v_1, v_3\}, \{v_2, v_3\}$	$\rightarrow X_{23}, X_{24}$

Table 1 Feasible target lists for vehicles v_1 to v_4 as output of Step1

not in conflict with Definition 1. Nodes are added according to the following rule:

$$P \cup X_k = \begin{cases} [X_0, X_1, ..., X_j, X_k] & \text{if } X_j \text{ should be followed by } X_k \\ [X_0, X_1, ..., \{X_j \cup X_k\}] & \text{otherwise} \end{cases}$$

If the addition of X_k to P violates Definition 1, we do not expand plan P further, and this part of the tree is bound. Otherwise, the cost of the plan is computed and checked whether the plan is complete or not. If the plan is complete and the cost c is less than *b_cost*, then *b_cost* takes the value of c and the tree is bound from X_k . If the cost c of the plan is greater or equal than *b_cost*, then the tree is bound from X_k as well. Otherwise, action 3 is repeated[2].

4. The procedure terminates if the whole tree has been searched for, or a user defined criterion has been met, e.g. a maximum amount of time.

In the following, we illustrate the steps of the procedure by means of an example.

Example 3 Consider a target set $\{t_1, t_2, t_3, t_4, t_5\}$ and a set of vehicles $\{v_1, v_2, v_3, v_4\}$. In Table 1, the lists of targets that are feasible by a set of vehicles are given as a result of Step 1. Note that X_1 corresponds to $(\{v_1\}, [t_1])$, X_9 corresponds to $(\{v_1\}, [t_3, t_2, t_1])$, and so on. X_{23} and X_{24} correspond to $(\{v_1, v_3\}, [t_4])$ and $(\{v_3, v_4\}, [t_4])$, respectively. Suppose that the cost yielded by a plan $P = [X_{12}, \{X_{22} \cup X_{23}\}] = [\{(\{v_2\}, [t_1, t_2])\}, \{(\{v_4\}, [t_3, t_5]), (\{v_1, v_3\}, [t_4])\}]$ is 30, the initial value of *b_cost*.

In Figure 3, a part of the tree has been generated and the cost associated with each plan is surrounded by a box. For example, the cost of plan $[X_{24}]$ is 15. Adding X_9 to this plan, which results into $[X_{24}, X_9]$, increases the cost to 30. Since this cost is equal to the cost of $P = [X_{12}, \{X_{22} \cup X_{23}\}]$, this part of the tree can be bound as a result of Proposition 1. Expand-

[2]A user or a knowledge base may decide whether an action set X_j should precede or follow a set X_k.

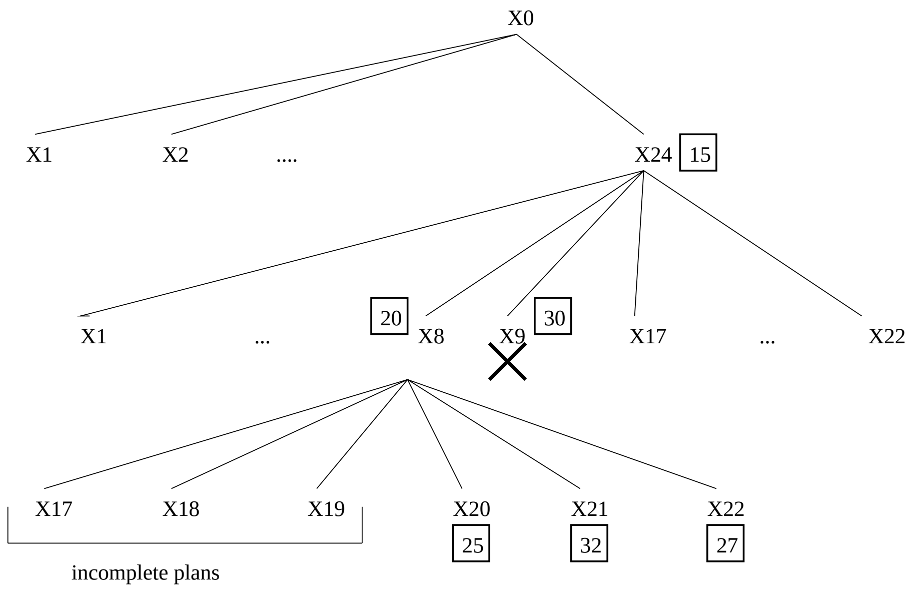

Fig. 3 Plan generation

ing plan $[X_{24}]$ with X_8 increases the cost to 20, which justifies the further exploration from X_8 of the tree. The result is given in Figure 3. As we can see the plans $[X_{24}, X_8, X_{17}]$, $[X_{24}, X_8, X_{18}]$, and $[X_{24}, X_8, X_{19}]$ are incomplete. Since the cost of plan $[X_{24}, X_8, X_{20}]$ is 25, this is the cheapest plan until now, and *b_cost* gets the value 25.

We note that the expansion of X_{24} with the pairs X_{10} to X_{16} are omitted, since it is in conflict with Definition 1. $\square$

In the next section, we discuss the impact on a plan when previously unknown information with regard to a target becomes available. Furthermore, we discuss how to revise a plan such that it will be in agreement with the new situation.

4.2 Plan updating

As stated in the foregoing, information with regard to targets is the product of continuous intelligence gathering. It may happen that previously unknown information becomes available or that some information becomes invalid in the course of time. For example, a previously unknown target has been discovered or the location of a target that has been assumed appears not to be correct. This may has as consequence that our selected plan may become incomplete or that the actual cost of the plan is more expensive than computed. Therefore, we revise a selected plan in accordance with new relevant information.

For the sake of plan revisions, we store the results of steps 1 and 2, i.e., the attainable list of targets by each vehicle and the tree generated in Step 2. For the time being, we assume that there is enough storage space to store the tree. Furthermore, we assume that the release of previously unknown information may lead to the following operations in our model.

- deletion of a target t
- insertion of a target t
- update of recorded information with regard to a target t

Let us study the effect of these operations on the results of Step 1. If a target t is deleted, then each attainable list in which t appears should be deleted from the results of Step 1. If a target t is inserted, then we generate, for each vehicle, all attainable lists that includes t. Update of information with regard to a vehicle can be considered as a deletion of a target followed by an insertion of the same target with the appropriate information.

Let X be the set of X_i's that is the result of Step 1. Then, application of a number of above-mentioned operations induce to the following situations:

1. X remains the same
2. a non-empty set of X_i's has been removed from X
3. a non-empty set of X_i's has been added to X
4. a combination of 2 and 3 occurs

If X remains the same or a non empty set of X_i's has been removed, then the selected plan is still complete. In the other cases, the plan may become incomplete. However, in all cases it may occur that the cost of the selected plan is changed. In the following, we discuss how the tree generated in Step 2 may updated for the different situation.

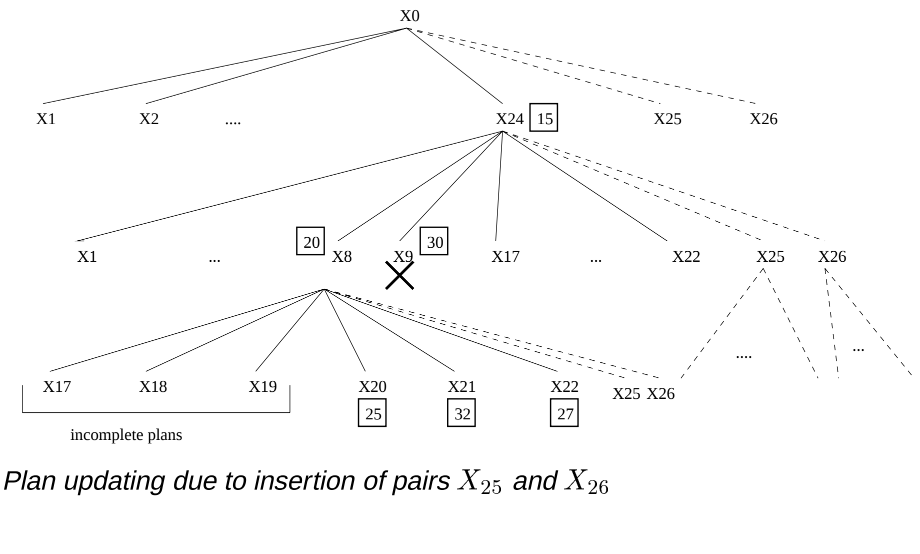

Fig. 4 Plan updating due to insertion of pairs X_{25} *and* X_{26}

In the first situation, we recompute the cost at all relevant nodes, i.e., nodes in which target t is involved. This may have as consequence that at nodes where we had decided to bound the tree should be expanded. Suppose that the location of t_2 in Example 3 has been changed and due to this change the cost of plan $[X_{24}, X_8, X_{20}]$ becomes 28 and the cost of plan $[X_{24}, X_9]$ becomes 25. Then, if *b_cost* = 28, we have no justification to bound the tree at X_9.

To update the tree due to the second situation, we mark all plans, in which at least one of the X_i's appear, as not applicable and the cost of remaining plans are recomputed.

If new X_i's are added in Step 1, then the tree is expanded with the new X_i's as illustrated in Figure 4 and the cost at the relevant nodes are recomputed. In Figure 4, the tree of Figure 3 is expanded with X_{25} and X_{26}. We note that if X_i's are added due to insertion of a new target, then the new added nodes are the relevant ones.

Finally, the tree can be updated properly due to situation 4 by first marking the relevant nodes as not applicable, and then adding the new relevant nodes in the tree.

5 Application

Although our approach is applicable in many domains, we have selected to tackle the so-called resource allocation problem for the naval domain. In this chapter, we describe how our approach can be applied to solve this problem. Although we are currently implementing our approach for this problem, we are able to provide some preliminary results. These results are based on a preliminary implementation and evaluation. After introducing the resource allocation problem, we will report our preliminary results.

In the naval domain, forces at sea, consisting of different ships, may require support from aircraft to fulfil their mission. Air support is realized by assigning a number of aircraft with weapons to a sea force. We focus on the distribution of ships and aircraft —equipped with weapons— to the different targets, such that each target is damaged to a desirable extent and with acceptable cost. To determine candidate vehicles to a target, an expert applies a number of rules. The antecedent of these rules takes relevant target characteristics, such as size of a target, goal of a target, etc., and relevant information with regard to the environment, such as weather conditions, etc., into account. In [Ref 6], a number of useful rules are reported. These rules can be used to build up the target lists for each vehicle.

In the following, two types of aircraft are distinguished, namely helicopters and fixed-wing aircraft. So, in this application, the set of vehicles is defined as follows: $\mathcal{V} = \{f_1, f_2, ..., f_{n_f}, h_1, h_2, ..., h_{n_h}, s_1, s_2, ..., s_{n_s}\}$, in which f_i is a fixed-wing aircraft, h_i is a helicopter, and s_i is a ship.

The deployment of each vehicle to a target entails some cost, which depends on the type of the vehicle and the weapons carried by this vehicle. In the following, this cost is defined as follows: $F : \mathcal{V} \times \mathcal{W} \rightarrow \Re^+$, in which $\mathcal{W}$ is a set of weapons. A database is available that contains the cost associated with each relevant pair (v, w), in which $v \in \mathcal{V}$ and $w \in \mathcal{W}$.

Let W be a set of pairs (w, q), in which w represents a weapon type and q the number of this type carried by a vehicle v. So, W is defined as $W = \{(w, q) | w \in \mathcal{W} \wedge q = 1, 2, ..\}$.

$$C_{\text{att}}(v, b, [t_1, t_2, ..., t_n]) = \begin{cases} \sum_{z \in W} z.qF(v, z.w) + d(b, t_1) + & \text{if } [t_1, t_2, ..., t_n] \\ \sum_{i=1}^{n-1} d(t_i, t_{i+1}) + d(b_h, t_n) & \text{is attainable by } v \\ \infty & \text{otherwise,} \end{cases}$$

in which $d(t_i, t_j)$ is the distance between two targets and $C_{\text{base}}(t_n, b_h) = \min_{b_l \in \mathcal{B}} C_{\text{base}}(t_n, b_l)$.

The damage of each target is measured by a value between 0 and 1, and a target is considered as damaged whenever a threshold value is obtained or exceeded. The damage function D is defined as follows: $D : 2^{\mathcal{V}} \times 2^{\mathcal{W}} \times \mathcal{T} \to [0,1]$. The damage value of a triple $(V \subseteq \mathcal{V}, W \subseteq \mathcal{W}, t)$ is derived from a database as well. Plans that contain targets that are not damaged are not complete.

In this application an additional constraint is put upon plans, namely a plan should be completed within a time interval. It should be clear that this constraint may be used in bounding parts of the plan tree, which is discussed in the previous chapter. If a plan P is not completed in a certain interval, then no plan that subsumes P can be completed in this time interval. So, all plans containing P can be ignored for investigation.

We note that, for the time-being, the cost to perform actions on a target is not explicitly specified but included in the function C_{att}.

As noted already, a preliminary implementation has been made for the resource allocation problem, called RACAS, which has been described in [Ref 5]. RACAS takes as input a number of vehicles and targets, and produces as output plans with decreasing costs. We have offered RACAS various combination of vehicles and targets and have monitored its results. Two conclusions could be drawn from these results. RACAS had no problems in generating lists of attainable targets, i.e., step 1 of our approach. In performing Step 1, some of the available domain knowledge has been applied. Second, the first implementation of RACAS ran into memory problems while generating plans (step 2), since we had decided to store the whole tree generated in Step 2 in order to facilitate updates.

To solve the memory problem during Step 2, strategies are being devised to remove parts of the tree. Two obvious strategies are: 1) remove those parts of the tree that consist of plans whose costs are higher than a threshold cost value, 2) remove those parts of the tree that have not been used for a while or that is not expected to be used frequently. For example, if two action sets in a plan are not desired, then paths in the tree that contains those sets can be removed. We feel that domain knowledge may play a major role in removing large parts of the tree, and to use the available memory efficiently. These issues are currently under investigation.

6 Conclusions & further research

We have presented a novel approach to tackle VMP. We have integrated mathematical results, which have been derived by formalizing VMP, with domain knowledge to solve the problem. Furthermore, we have chosen a more accessible formalism to express plans. This has as advantage that a plan is better understood by users, which in turn makes it easier to add domain knowledge in the planning process. Traditionally, mission planning problems are formulated as: Optimize a function that is subjected to a set of constraints. Often, the user does not have any insight how this problem is solved, which makes it difficult to add useful domain knowledge. Finally, we have proposed a strategy to update plans according to new information. This strategy keeps replanning efforts to a minimum.

Although currently our approach is being implemented, we were able to perform a preliminary evaluation on the basis of an application in the naval domain. Therefore, a preliminary implementation of some vital parts of our approach has been realized. From this preliminary evaluation, we conclude that Step 1 of our approach could be performed successfully for this application without extensively using domain knowledge. Storing the whole tree generated during Step 2 led to memory problems, simply because this tree is too large. We have proposed several strategies to tackle this problem.

In the first half of 1998, the preliminary implementation will be extended to a tool for resource allocation in the naval domain that will be equipped with domain knowledge and strategies for an efficient use of memory space.

Acknowledgments The author thanks Yves van de Vijver from NLR and Julio Rives from INDRA DTD (Spain) for their useful comments on earlier drafts of this paper. Niels Basjes from NLR is thanked for his implementation efforts.

References

1. New Advances in Mission Planning and Rehearsal Systems, AGARD Lecture Series 192, NATO, October 1993.
2. Dockery, J.T., Woodcock, A.E.R., The Military Landscape, Wood head Publishing, Cambridge, England, 1993.
3. Donker, J.C., Artificial Intelligence Methods and Systems for Planning and Scheduling, Overview of the State of the Art, Technical report TP 97356 L, National Aerospace Lab., Amsterdam, 1997.
4. Kolitz, S.E.,Computing Techniques in Mission Planning, in [1], pp. 4.1-4.20, 1993.
5. Euclid RTP 6.1 - Grace, Working Paper W420, To be published as NLR Technical Report.
6. Euclid RTP 6.1 - Grace, Working Paper W404.2, To be published as NLR Technical Report.