Max-Plus Enabled Dynamic Programming for Sensor Platform Tasking

Ali Oran and William M. McEneaney

Abstract—In a military context, the value of a future operation is obtained as the expectation of the payoff, based on the current probability distributions describing our knowledge of the battlespace state. Sensing vehicles may be employed to improve this value function through control of the observation-conditioned distribution. We consider the problem of tasking sensing vehicles so as to optimize the future value. Max-plus based dynamic programming was employed, exploiting the special form of the value function as a pointwise maximum of linear functionals, where we note that this form is preserved as one propagates. This approach avoids the curse-of-dimensionality, but encounters another computational complexity difficulty. This complexity growth is attenuated through projection onto lower-dimensional subspaces. An error analysis is developed for these approximations.

I. INTRODUCTION

During the last decade UAVs and other autonomous vehicles have demonstrated their capabilities in numerous situations. Different researchers have approached different aspects of autonomous UAV control, focusing on topics such as optimal fuel usage, collision avoidance and cooperative operations under limited communication (c.f. [6], [7], [8]). One of the most important topics being addressed is feedback-based task planning and replanning for reconnaissance operations. Solution of such problems is essential for UAVs used in adversarial environments under imperfect information, such as in military operations.

We consider a problem where the UAVs will engage in intelligence preparation of the battlefield prior to ground troop movement. We refer to the players as Blue and Red. Suppose that over a series of time-steps, from \( t = 0 \) to \( T \), Blue sensing actions will occur, and that immediately after, the ground troop action will begin. At time \( t = 0 \), Blue knowledge is described by distribution \( q_0 \). Given a series of sensing actions, \( u^o_{[0,T-1]} \), there will be an associated set of possible observations, \( y^o_{[0,T-1]} \). These observations are modeled as random variables since the actual observations will be corrupted by noise. Given a set of controls and observations, one may update the initial distribution, resulting finally in \( q_T \). Notice that \( q_T \) will itself be a random variable. The proper payoff for the use of the UAV actions is not the entropy reduction (although that is often closely related), but the resulting reduction in losses to Blue during the ensuing ground operations. One may formulate the sensing operation as an optimization problem where we maximize:

\[
J(u^o_{[0,T-1]}) = \mathbb{E}\{V_T(q_T) - C(u^o_{[0,T-1]})\},
\]

where \( V_T(q_T) \) represents the resulting expected ground operation payoff given \( q_T \), and \( C(u^o_{[0,T-1]}) \) is a random variable describing the possible UAV losses.

In [4] and [5] following the optimal control methodology discussed above, and using Bayes’ rule propagation of the conditional distribution, the authors were able to improve the expected ground troop survivability by 15% over a more traditional heuristic controller. However, in these two studies it was also noticed that the numerical analysis for both the open and the closed loop controllers was subject to the curse-of-dimensionality. This would lead to unfeasible computation times for analysis of a large number of UAVs or locations. A possible solution to this was the exploitation of the special form of the value function during backward dynamic programming (DP). Although introduction of that approach led to computationally more efficient schemes, it was found that further research was still needed to attain computational feasibility for larger problems.

This special form of the value function, is a pointwise maximum of affine functionals over a probability simplex. The computational problem arises from the growth of the number of elements in this set of affine functionals at each iteration of the DP. The first step in tackling this problem is isolating those functionals which do not contribute to the pointwise maximum— the inactive functionals. We refer to this as the refining step. Importantly, it is found that any inactive member of the set at the current step would be inactive for all further iterations in the dynamic program, and thus could be eliminated without loss of accuracy. This refining algorithm has a considerable effect on the computation speed while preserving the original optimal solution. Nonetheless, the computational burden is still too high for real-world application. An additional technique, pruning, is also applied. By pruning, we refer to the elimination of functionals which do make some contribution to the solution, but whose contribution is relatively small. Thus, in pruning, we do generate some inaccuracies in our solution. Here, in addition to introducing these computational improvements, we also consider their effects on the accuracy of the resulting value function approximation. Two different pruning methods are developed by introducing two different error functions based on \( L_\infty \) and \( L_1 \) norms. Both methods are...
analyzed with respect to the error propagation and other properties they possess.

II. THE OPTIMAL CONTROLLER

A. Stochastic Modeling of Ground Operations & the Value of Information

We use a Blue ground troop action control problem to obtain the value of information. This is a necessary precursor to solving the UAV control problem, as it indicates the proper form for the value of information as a piece-wise linear convex function over a probability simplex. This form is quite general. The discussion will be a bit terse, as the details of this subproblem are not the focus of the current effort. We use the following simple example to motivate the mathematical model, but the approach applies to a larger variety of Command and Control (C2) problems. We suppose that a single Blue force is operating in (moving through) hostile urban territory, and that Red forces might be hiding in any of the buildings located in this region. (One might think of a military convoy passing through an opponent-infested urban region – see Fig. 1) We continue to use this simple example problem in defining the underlying value of information/observation control cost criterion. There are a finite number of locations in the region, where we denote the region by \( \mathcal{R} \). For example, the set of locations might be simply the set of buildings. Given \( N_B \) distinct Red combat teams in \( N_R \) buildings, the number of possible states would be \( N = N_B N_R \). Let \( N_{\mathcal{R}} \trianglerighteq \{1, \ldots, N\} \). In the example depicted in Fig. 1, if there was only one Red team, then we would have \( N = 7 \). We denote the each possible Red state as \( x_R = n \in N_{\mathcal{R}} \). The Blue information on the Red state in \( \mathcal{R} \) is described by probability distribution \( q \). Note that distribution \( q \) lies in simplex \( S^N \) where

\[
S^N = \left\{ q \in R^N \mid q_n \in [0, 1], \forall n \in \{1, \ldots, N\}, \sum_{n=1}^{N} q_n = 1 \right\}.
\]

In particular, in an example where there was only one Red team, the \( n \)th component of \( q \), \( q_n \), would be the probability that the Red entity was at location \( n \). Continuing the development using our motivational example, we suppose that the allowable local, combat Blue controls are \( I = \{0\} \cup \mathcal{N} \). In our example, \( i \in \mathcal{N} \) implies that Blue is laying cover fire on potential Red location \( n \) (if possible), while proceeding with its Blue COA (course of action). Further, \( i = 0 \) implies that Blue is “tight” during this step, meaning that the local Blue entity does not fire unless fired upon during this step. We will make the assumption that Blue is more well-protected if firing upon a correct potential Red location. Further, we will assume that Blue is somewhat better able to defend itself from an attack from location \( n' \) if in stance “tight” \( (i = 0) \), rather than in stance \( i \neq n' \) (with \( n > 0 \)). The inclusion of control \( u = 0 \) is not necessary to the approach, but is included for additional realism.

We suppose the local Blue force have a health state \( h \in \{0, 1\} \), where \( h = 1 \) represents healthy, and \( h = 0 \) represents destroyed. This simple model is sufficient for our purposes here; our goal is the development of the sensing-platform control. The Blue force wishes to maximize its probability of survival through \( \mathcal{R} \). Let \( \rho: \mathcal{N} \times I \to [0, 1] \) where specifically, \( \rho(x_R, i) \) will be the probability that Blue is destroyed \( (h \text{ transitioning from } 1 \to 0) \) given true Red position state \( x_R \) and Blue local combat control \( i \). Under perfect information, Blue would choose its control to maximize \( \gamma(x_R, i) = 1 - \rho(x_R, i) \), over possible \( i \in I \). In general, one must consider multiple steps in the Blue troop actions, but we do not include the details here, as the form is unchanged, and our goal is development of the UAV controls. Now, given an information state in the form of probability distribution, \( q \in S^N \) (and continuing to ignore the details of multiple ground troop steps here), the Blue ground force control, \( i \), would be chosen to maximize payoff

\[
\mathcal{J}(q, i) \equiv \sum_{x \in N_{\mathcal{R}}} \gamma(x, i) q_x = v^i \cdot q,
\]

where \( v^i \) is the vector of length \( N \) with elements \( \gamma(x, i) \). Now, we define the value of information as the value of this control subproblem:

\[
\mathcal{V}(q) \equiv \max_{i \in I} \mathcal{J}(q, i) = \max_{i \in I} \{ v^i \cdot q \}
\]

It is worth noting that \( \mathcal{V} \) is a piecewise linear function of its argument, Blue’s probability distribution regarding Red; this form will be exploited in section III. An example \( \mathcal{V}(q) \) is depicted in Fig. 2.

B. Observation Control Problem

Now, that the value of information is defined through combat team dynamics, one can relate this quantity to the sensing operation. Since the information is a function of UAV observations, the goal of the sensing operation should be to maximize the expected value \( \mathcal{V}(q) \) through the tasking control of the UAV’s sensor platforms.

As stated earlier, the estimator that defines the dynamics of the information state is based on Bayes’ rule. Here, for simplicity, we suppose that the sensing
platforms can move from any position to any other position in $R$ in one time-step. With this freedom, any $x \in N_R$ is an admissible sensor-platform control, and so we take the control set to be $U = N_R$. Having defined the control set, suppose the sensor is at $x \in N_R$ at time $t$. The observation $y = y_t$ will take a value in $Y = \{0, 1\}$. We form the observation matrix with diagonal elements $[D(R^{u_t,y_t})]_{i,j} = R^{u_t,y_t}$, and such that $[D(R^{u_t,y_t})]_{i,j} = 0$ for $i \neq j$. Then, given any sensing control action $u_t = x \in N_R$ and resulting (random-variable) observation $y_t$, one has:

$$q_{t+1} = \frac{1}{R^{u_t,y_t} \cdot q_t} D(R^{u_t,y_t}) q_t \cdot \beta^{u_t,y_t}(q_t)$$

(3)

which defines the stochastic information state dynamics. For the closed loop controller, the state at time $t$ consists of the current sensor position and the current information state, $q_t$. As the sensor can move from any location to any other in one time-step, we will suppress the sensor-position as state component. Now let $A_s$ be the set of nonanticipative feedback controls from time $s$ to terminal time, $T$. That is, we let $A_s = \{a_{[s,T-1]}: [SN]_{T-s} \rightarrow U_{T-s}\}$ if $q_t = \hat{q}_t$ for all $r \leq t$ then $a_{[s,T-1]} = a_{[s,T-1]}(\hat{q}_t)$ for all $r \leq t$

where $U_{T-s}$ denotes the outer product of $U$, $T-s$ times, and similarly with $[SN]_{T-s}$. The payoff for information state $q_s = q$ and non-anticipative control $a \in A_s$ is

$$J(s,q,a) = \mathcal{E}\{V(q_T)\}$$

(4)

where the propagation of the state from $q_s = q$ to $q_T$ follows (3) with control $a_t = a_t[q_t]$ at each time, $t$. The corresponding value function is:

$$V(s,q) = \sup_{a \in A_s} J(s,q,a)$$

(5)

III. UTILIZATION OF DYNAMIC PROGRAMMING

To solve the state feedback problem backward DP was utilized. The following theorem formulates the dynamic programming principle for the state feedback problem:

Theorem 3.1: For $t \in \{0, 1, \ldots T-1\}$,

$$V(t,q) = \max_{u_t \in U} \mathcal{E}_{y_t} \left\{ V(t+1, \beta^{u_t,y_t}(q_t)) \right\}$$

(6)

where the expectation is over the set of possible observations.

Proof: The proof is standard.

One can notice that for larger values of $N = \# N_R$, the dynamic programming computations would become computationally infeasible when performed over the discretized probability simplex (grid-based methods), even for short time spans. However, the special form of $V(t,q)$ inherited from $V(q)$ can be exploited to avoid this problem. That is, from (2), $V(T,q) = \max_{i \in \mathcal{I}} (v^i.q)$. If we can show that this form is retained under the dynamic programming propagation, then we will be able to work with the $v^i$ vectors instead of a discretized form of $V(t,q)$ over the probability simplex. In order to demonstrate this, we first introduce the following notation. For any set, $\mathcal{I}$, and positive integer $M$, let $P^M(\mathcal{I})$ be the set of all sequences of length $M$ with elements from $\mathcal{I}$ (note that the cardinality of $P^M(\mathcal{I})$ is $(\# \mathcal{I})^M$). Also in order to generalize our problem statement, we relabel $Y$ as $Y = \{1, 2, \ldots N_y\}$ and denote the general control set as $U = \{1, 2, \ldots N_u\}$. Note that in our motivational example we had $Y = \{0,1\}$ and $U = N_R = [1, N]$.

Theorem 3.2: Suppose $V(t+1, q)$ takes the form

$$V(t+1, q) = \max_{i \in \mathcal{I}_{t+1}} v^i_{t+1} \cdot q$$

where $\mathcal{I}_{t+1} = \{1, 2, \ldots I_{t+1}\}$. Then,

$$V(t,q) = \max_{i \in \mathcal{I}_t} v^i_t \cdot q$$

(7)

where $\mathcal{I}_t = \{1, 2, \ldots I_t\}$, $I_t = N_u(I_{t+1})^{N_y}$,

$$v^i_t = \sum_{y_t \in Y} D(R^{u^i_t,y_t}) v^i_{t+1}$$

(8)

where $(u^o, \{j_{yt}\}) = M^{-1}(i)$, and $M$ is a one-to-one, onto mapping from $U \times P^{N_y}(I_{t+1})$ to $\mathcal{I}_t$ (i.e., an indexing of $U \times P^{N_y}(I_{t+1})$).

Proof: The proof requires application of Theorem 3.1 on the assumed form and interchange of the max and the $\sum$ operators (using the max-plus distributive property (c.f. [1], [2]). A complete proof is given in [5].

We now develop some helpful notation. For any $t$, let $\mathcal{V}_t = \{v^i_t| i \in \mathcal{I}_t\}$. Then, by Theorem 3.2, the dynamic program can equivalently be given as $\mathcal{V}_t, \mathcal{I}_t) = \mathcal{D}^{U}_{\mathcal{V}_t, \mathcal{I}_t}$, where the operator, $\mathcal{D}^{U}$ is defined by
the propagation (8). Also, we can denote the reconstruction of \( V(t, \cdot) \) from the pair \( (V_t, \mathcal{I}_t) \) as \( V(t, \cdot) = \mathcal{C}[(\nabla_t, \mathcal{I}_t)] \), where the reconstruction operator is given by (7).

With this finding the numerical burden of grid-based analysis of \( V(t, q) \) on the probability simplex is avoided, and one only needed propagate the vectors \( v^i_t \) backwards in time using (7). This yields a significant reduction in the computation times. However, when the DP is performed for large time spans, the computation speed is observed to still be too slow relative to what would be required for real-time UAV operations. The root cause is the growth of the set \( \mathcal{I}_t \) at each iteration. We address the means that may be used to attenuate this problem in the following section.

IV. REFINING AND PRUNING METHODS

In this section, we introduce two approaches to increase the computation speed of the algorithm. Briefly, the first one, “refining”, will involve the elimination of inactive members of the set \( \mathcal{I}_t \), i.e., the elimination of those that nowhere achieve the maximum. On the other hand, “pruning” will refer judicious elimination of active elements of \( \mathcal{I}_t \).

A. Refining the Index Set, \( \mathcal{I}_t \)

When backward dynamic programming was employed to obtain the vectors \( v_t \) out of the set of \( v_{t+1} \), it was noticed that the newly generated set of \( \mathcal{I}_t \) was not generally the minimal set containing the necessary information at time \( t \). Some vectors in set \( \mathcal{I}_t \) were inactive (suboptimal) in the simplex and thus they never influenced the supremum. That is, an inactive vector yields hyperplanes which are everywhere below the supremum of the other hyperplanes; an example can be seen in Fig. 2 where purple hyperplanes are inactive. From this observation, one might also wonder whether the progeny of such vectors (through the DP propagation) would remain inactive during subsequent steps. If so, then these vectors can be instantly eliminated at the first time of non-influence from the index set, and in this way the growth of the size of \( \mathcal{I}_t \) would be slower, thereby greatly speeding computation. The following theorem indicates this propagation of inactivity.

Theorem 4.1: Let \( \mathcal{R}_{t+1} \) be the refined subset of \( \mathcal{I}_{t+1} \), i.e:

\[
\mathcal{R}_{t+1} = \{i \in \mathcal{I}_{t+1} : \exists q \in S^N, v^i_{t+1} \cdot q > v^j_{t+1} \cdot q \forall j \in \mathcal{I}_{t+1} \setminus \{i\}\},
\]

and let the corresponding refined vectors be \( \nabla^R_{t+1} = \{v^i_{t+1} \in \nabla_{t+1} : i \in \mathcal{R}_{t+1}\} \). Let \( (\mathcal{V}_t, \mathcal{I}_t) = \mathcal{D}^R[(\nabla^R_{t+1}, \mathcal{I}^R_{t+1})] \), i.e., the backward propagation of the refined set of vectors. Then, \( \mathcal{C}[\mathcal{V}_t, \mathcal{I}_t] = \mathcal{C}[(\nabla^R_t, \mathcal{I}^R_t)] \). In other words, it is sufficient to work with \( \mathcal{I}_t \).

Proof: For the sake of presentation we define a slightly different notation for mapping \( \mathcal{M} \). For each \( u^o \in \mathcal{U} \) and \( \{i_{y_t}\} \in \mathcal{P}^N_y(\mathcal{I}_{t+1}) \), we let \( \mathcal{M}_t[u^o_{t+1}](\{i_{y_t}\}) = \mathcal{M}(u^o_{t+1}, \{i_{y_t}\}) \). Now, consider \( j \in \mathcal{I}_t \) but \( j \notin \mathcal{I}_t \). Then by the definition of \( \mathcal{I}_t \), \( \exists u^o_{t+1} \in \mathcal{U} \), and \( \{i_{y_t}\} \in (\mathcal{I}_{t+1})^N_y \) such that \( \mathcal{M}_t[u^o_{t+1}](\{i_{y_t}\}) = j \). Then for \( q \in S^N \) following the formulation (7) one can write:

\[
v^j_t \cdot q = \left( \sum_{y_t} D(R^{u^o_{t+1}, y_t})v^i_{t+1} \right) \cdot q = \sum_{y_t} \left( v^i_{t+1}(y_t) \right)^T D(R^{u^o_{t+1}, y_t})q = \sum_{y_t} (v^i_{t+1}, q^{u^o_{t+1}, y_t} R^{u^o_{t+1}, y_t} \cdot q),
\]

where \( q^{u^o_{t+1}, y_t} = \frac{D(R^{u^o_{t+1}, y_t})q}{R^{u^o_{t+1}, y_t} \cdot q} = \beta^{u^o_{t+1}, y_t}(q) \)

and it should be noticed that \( q^{u^o_{t+1}, y_t} \in S^N \). Then following the definition of \( \mathcal{R}_{t+1} \), \( \exists k_{y_t} \in \mathcal{R}_{t+1} \) (the maximizer index set for \( q^{u^o_{t+1}, y_t} \)) such that \( v^k_{t+1} \cdot q^{u^o_{t+1}, y_t} \geq v^j_{t+1} \cdot q^{u^o_{t+1}, y_t} \). Then

\[
v^j_t \cdot q \leq \sum_{y_t} (v^k_{t+1}, \beta^{k_{y_t}, y_t} R^{k_{y_t}, y_t} \cdot q) = \sum_{y_t} \left( D(R^{k_{y_t}, y_t})v^k_{t+1} \right) \cdot q
\]

Now, define a functional \( \mathcal{M}_t[u^o_{t+1}] : \mathcal{R}_{t+1}^N \rightarrow \mathcal{I}_t^N \), 1-1 and onto, where \( \mathcal{I}_t^N \) are partitions of \( \mathcal{I}_t \) for different \( u^o_{t+1} \); i.e:

\[
\hat{\mathcal{I}}_t = \bigcup_{u^o_{t+1}} \mathcal{I}_t^N \\
\hat{\mathcal{I}}_t \bigcap \mathcal{I}_t^N = \emptyset
\]

Then

\[
v^j_t \cdot q \leq \sum_{y_t} \left( D(R^{u^o_{t+1}, y_t})v^k_{t+1} \right) \cdot q = v^j_t \cdot q,
\]

where,

\[
v^j_t \cdot q \leq \sum_{y_t} \left( D(R^{u^o_{t+1}, y_t})v^k_{t+1} \right) \cdot q = \mathcal{M}_t[u^o_{t+1}](\{k_{y_t}\})
\]

This analysis shows that for any \( j \in \mathcal{I}_t \) but \( j \notin \mathcal{I}_t \), \( \exists r \in \mathcal{I}_t \) such that \( v^j_t \cdot q \leq v^r_t \cdot q \). Thus it is sufficient to propagate \( \mathcal{R}_{t+1} \).

Now that we have established that suboptimal vectors do not influence the subsequent value functions, the only thing that remains is to choose a method to eliminate these vectors. The simple way chosen for this purpose is the well documented linear programming algorithm. Suppose that a number of \( v^j_t \) vectors are already present in our index set \( \mathcal{I}_t \) and we want to determine if \( v^j_t \) contributes anything to \( V(t, q) \). The following optimization scheme handles this problem.

\[
\max_{q \in S^N} : v^j_t \cdot q - z
\]

subject to :

\[
v^i_t \cdot q - z \leq 0, \quad \forall j \neq i
\]

\[
z \geq 0
\]

\[
q_k \geq 0 \quad \forall k \quad \text{and} \quad \sum_k q_k = 1
\]
Now, writing the inequalities and equalities in the all-
inequality form one formulates:

$$\min \ x \ : \ c^i \cdot x, \ x = [q^T z]^T, \ c^i = [-v^T_1 1]^T$$

subject to : $Ax \geq 0,$

for proper choice of $A,$ and we do not include the details. If this last optimization scheme results in a negative value (positive in the first scheme) then the $v^i_1$ vector should be kept in the memory (contributing to $V(t,q)).$ Otherwise, it should be eliminated. Repeating this scheme $\forall \ i \in I_t$ one gets the the minimal set $R_t$ which still yields $V(t,q).$

The reader should notice the extreme growth of the size of $I_t$ in Theorem 3.2. Overall, the refining of $I_t$ gave us a significant boost to computation speeds by reducing this growth. Monte Carlo simulations were done with randomly created $V_T$ sets that were still conforming to our previous assumptions about the Blue fire teams. For a sensor tasking problem problem defined on $S^2,$ it was found that the refining algorithm reduced the size of the set $I_{T-3}$ to an average $1/3$ of its original size. The sizes of subsequent sets were even reduced by higher factors. By the end of the third iteration (with observation and refining at each step) the size of $I_{T-3}$ was reduced by an average factor of more than 30,000.

Besides this improvement, it was noticed that the simplex method was also giving us a quantitative value about the contribution of individual vector in $(V_T, R_t)$ to our analysis of $V(t,q),$ the quantity $\max \|v^i_1\cdot q - z\).$ Exploiting this value, one can think about eliminating vectors with very small contributions to $V(t,q)$ to keep the size of $I_t$ more manageable. This idea forms the basis for the next analysis.

B. Pruning the Refined Set, $R_t$

As mentioned above, some vectors defining $V(t,q)$ at time $t$ might have very small contributions to $V(t,q).$ To improve the computation speed these vectors might be omitted. To identify which vectors to eliminate from the refined set, $R_t,$ we first need to define an error function at time $t,$ $\epsilon_t,$ over the probability simplex, $S^N,$ for the pruning analysis. Two candidates chosen for this purpose were the functions defined by $L_{\infty}$ and $L_1$ norms. We first present our results with the $L_{\infty}$ norm and later with the $L_1$ norm.

Suppose that at time $t,$ a number of vectors were pruned out of the refined set, $R_t,$ leaving us the remaining set, $P_t.$ We define the the error, occurring by omitting those vectors out of the refined set, using the $L_{\infty}$ norm as:

$$\epsilon_{\infty}(t) \doteq \|V(t,q) - W(t,q)\|_{\infty} = \max_{q \in S^N} \{V(t,q) - W(t,q)\}$$

where

$$V(t,q) = \max_{i \in R_t} \{v^i \cdot q\} \quad \text{and} \quad W(t,q) = \max_{i \in P_t} \{v^i_1 \cdot q\}$$

Since $\epsilon_{\infty}(t)$ is defined as the maximum over the simplex $S^N,$ calculation of this error can be considered as an optimization problem over that simplex. Thus, the linear programming algorithm used in the previous subsection may be employed for here as well. Running the optimization scheme over the refined set for each vector, $v^i_1,$ $i \in R_t,$ one can find:

$$\epsilon_{\infty}^i(t) = \max_{q \in S^N} \{V(t,q) - W_i(t,q)\}$$

where

$$W_i(t,q) = \max_{j \in R_t, j \neq i} (v^i_1 \cdot q)$$

Here, $\epsilon_{\infty}^i(t)$ is the error induced by omitting vector $v^i_1$ from $V(t,q)$ computations. The maximum of these errors over the set of pruned vectors would yield $\epsilon_{\infty}(t):

$$\epsilon_{\infty}(t) = \max_{i \in R_t \setminus P_t} \epsilon_{\infty}^i(t)$$

Relevantly, the pruned set $P_t$ can be defined as: $P_t \doteq \{i \in R_t \mid \epsilon_{\infty}^i(t) < \tilde{\epsilon}(t)\},$ where $\tilde{\epsilon}(t)$ is an upper error bound for the time $t.$

At this point, one might wonder about the existence of a possible error bound during DP iterations. The following theorem highlights the boundedness property of the error, $\epsilon_{\infty}(t).$

Theorem 4.2: Let $V(t+1,q)$ and $W(t+1,q)$ be the functions defined above. If

$$\epsilon_{\infty}(t+1) = \|V(t+1,q) - W(t+1,q)\|_{\infty} = \epsilon$$

then

$$\epsilon_{\infty}(t) = \|V(t,q) - W(t,q)\|_{\infty} \leq \epsilon$$

Proof: By the definition of $V(t+1,q)$ and $W(t+1,q),$

$$V(t,q) = \max_{u_t} \left\{ \sum_{y_t} V(t+1, \beta^{u_t,y_t}(q)) P(y_t) \right\}$$

and

$$W(t,q) = \max_{u_t} \left\{ \sum_{y_t} W(t+1, \beta^{u_t,y_t}(q)) P(y_t) \right\}.$$
the error in pruning. However, $\epsilon_\infty$ also possesses two disadvantages. First, and most importantly, it was found that pruning was not optimal for approximating $V(t, q)$ with a smaller set using the $L_\infty$ norm. That is, the optimal set of, say $\bar{n}$, vectors for approximating $V(t, \cdot)$ (where $\bar{n} < \#\mathcal{V} - \#\mathcal{V}_t$) may not consist of a subset of the elements of $\mathcal{V}_t$. Second, the authors are concerned that using the $L_\infty$ norm might not be an accurate way to measure the pruning error. Because of these drawbacks an error function based on the $L_1$ norm, $\epsilon_1(t)$, was also considered.

We define $\epsilon_1(t)$ as below:

$$
\epsilon_1(t) = \int_{S_N} V(t, q) - W(t, q) dq
$$

where $V(t, q)$ and $W(t, q)$ are as defined earlier. The main advantage of using $\epsilon_1(t)$ was highlighted in [3], where it was proven that an error function based on the $L_1$ norm would be convex, and moreover, when approximating a set of functions with another smaller set of functions the optimal reduced complexity representation would be comprised of a subset of the original set of functions.

That is, with an error metric based on the $L_1$ norm, pruning does, in fact, yield the optimal solution. This is the superiority of the $L_1$ norm over the $L_\infty$ norm. Having an optimal set of pruned vectors over the refined set $\mathcal{R}_t$ we are encouraged us to use $\epsilon_1(t)$ over $\epsilon_\infty(t)$. However, contrary to these fine properties $\epsilon_1(t)$ did not posses the boundedness property of $\epsilon_\infty(t)$ during DP. A counterexample is given below.

In the two-dimensional simplex, $S^2$, consider $\mathcal{R}_{t+1} = \{1, 3\}$ with $v_{t+1}^1 = [0.95 \ 0.25]'$, and $v_{t+1}^3 = [0.7 \ 0.65]'$. Suppose that out of these 2 vectors we need to prune the vector that will result to the the minimal pruning error, $\epsilon_1(t+1)$. Similar to $\epsilon_\infty(t)$ defined earlier, we define the pruning error induced by pruning a vector $i$ from the refined set, $\mathcal{R}_t$, but this time based on the $L_1$ norm as:

$$
\epsilon_1(t+1) = \int_{S_N} V(t+1, q) - W_i(t+1, q) dq
$$

where $W_i(t+1, q)$ was defined in the previous page. Now, since we are trying to prune out one single vector that would lead to least error in our analysis, the error $\epsilon_1(t+1)$ would be $\epsilon_1(t+1) = \min_{i \in \mathcal{R}_{t+1}} \epsilon_1^i(t+1)$ Following the definition of $\epsilon_1^i(t+1)$ we find, $\epsilon_1^1(t+1) = 0.0481$, and $\epsilon_1^3(t+1) = 0.1231$. Then, $\epsilon_1(t+1) = 0.0481$ and vector 1 should be pruned out to give us the pruned set of functions $\mathcal{R}_{t+1} = \{3\}$. Using (6), (7), and (8) the piecewise linear functions defining $V(t, q)$ and $W(t, q)$ is found. Integrating the difference between $V(t, q)$ and $W(t, q)$, one finds $\epsilon_1(t) = 0.1058$.

Because of this unboundedness one might be worried about error growth during the propagation process. However, even tough $\epsilon_1(t)$ might grow during subsequent steps in DP, an upper bound for the error growth is always maintained because of the relationship between the $L_\infty$ and $L_1$ norms. The following theorem highlights this fact.

Theorem 4.4: Suppose at some time $t \in \{0, ..., T\}$, $\epsilon_\infty(t) = \epsilon$. Then during DP, for any $0 \leq \tau \leq t$, $\epsilon_1(\tau) \leq \epsilon$.

Proof: Following the definition of $\epsilon_1(\tau)$ and $\epsilon_\infty(\tau)$ we have: $\epsilon_1(\tau) = \int_{S_N} V(\tau, q) - W(\tau, q) dq \leq \int_{S_N} \epsilon_\infty(\tau) dq = \epsilon_\infty(\tau)$. Now, from Corollary 4.3, we have $\epsilon_\infty(\tau) \leq \epsilon_\infty(t) = \epsilon$ for any $0 \leq \tau \leq t$. Combining these, one finds that $\epsilon_1(\tau) \leq \epsilon$.

Although this theorem eases our worries about unbounded growth of $\epsilon_1(t)$, it can be easily seen that it is actually a conservative bound. Here, one needs to trade off between computation speed and error bounding carefully. Another issue that needs attention is the numerical computation methods needed to calculate $\epsilon_1(t)$. Unlike $\epsilon_\infty(t)$ one cannot use linear programming, and numerical integration methods are required for complex piecewise functions that define $V(t, q)$. For probability simplices with dimensions lower than four, volume calculation algorithms not subject to the curse-of-dimensionality can be developed with the help of visual aids. For higher dimensions, the construction of such volume computation algorithms appears to be a difficult task.

V. CONCLUSIONS

In this paper we followed the earlier works [4], [5] to find the optimal tasking controller for the UAVs on reconnaissance missions in support of ground operations. We showed how a Max-Plus approach to DP could be utilized to solve the tasking control problem. The curse-of-dimensionality encumbered grid-based methods were bypassed with the exploitation of the special form of $V(t+1, q)$ as a pointwise maximum of linear functionals. Additional numerical improvement was achieved with the refining method, which allowed us to eliminate unnecessary information at any time without degradation of the solution in subsequent steps. Finally, two different pruning methods were introduced to further improve computation speed.

References