2001-11-01

JOIST (Joint Optimizing Informational Strike Tool)

Washburn, Alan R.

Monterey, California. Naval Postgraduate School

NPS-OR-02-001-PR

NAVAL POSTGRADUATE SCHOOL
Monterey, California

JOIST
(Joint Optimizing Informational Strike Tool)

by

Alan Washburn

November 2001

Prepared for: Air Force Office of Scientific Research

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943-5000

RADM David R. Ellison
Superintendent

Richard Elster
Provost

This report was prepared for and funded by the Air Force Office of Scientific Research.

This report was prepared by:

ALAN WASHBURN
Professor of Operations Research

Reviewed by:

R. KEVIN WOOD
Associate Chairman for Research
Department of Operations Research

JAMES N. EAGLE
Chairman
Department of Operations Research

Released by:

DAVID W. NETZER
Associate Provost and Dean of Research

JOIST (Joint Optimizing Informational Strike Tool)

by Alan Washburn

Abstract

JOIST is a prototype computer program that deals with the optimal tasking of both strike and information assets in a protracted war. Phenomena modeled include damage assessment, surveillance, and target tracking. The optimization is constrained by limits on weapon stocks, attrition, and the capacity of platforms to undertake assigned tasks. The model carefully follows the laws of Probability, using Bayes' Theorem to process information by revising the marksman's perception of target state. Realistically scaled problems can be solved in minutes or hours, depending on the degree of precision required.

Executive Summary

The technological possibilities that are emerging in the Information Age make many things possible militarily that were not possible just a few years ago, but many of the new information systems are expensive or in short supply. Therefore, these observational assets need to be carefully tasked, just as firepower assets need to be carefully tasked. Passive sensors must be told where and when to look and on what frequencies. Active sensors must be similarly guided, and must be employed especially carefully because of the risk of giving up more information than is gained. Furthermore, although this has not always been the case historically, strike and information assets should be tasked *jointly*. Information has no value in itself, but only as a guide to subsequent action. Using a shoot-look-shoot strategy with an expensive munition such as a cruise missile does not make sense if the "look" capability is ineffective or urgently employed elsewhere. JOIST attempts to task both kinds of asset jointly; in fact, the "Joint" part of JOIST's name refers as much to jointness between strike and information asset tasking, as it does to inter-service jointness.

JOIST is a tasking method dedicated to the idea that strike and information assets should be tasked *optimally* as well as jointly. The word "optimal" is used here in its mathematical sense; that is, an objective function whose meaning is "expected net value gained" is maximized over a specified time period, subject to constraints on the availability of assets such as aircraft sorties, aircraft losses, weapons and sensors. The reader should not be overly impressed by this optimality, since computational necessity forces JOIST to deal only with a simplified version of reality. The number of possible target states and sensor reports is very small, for example. Nonetheless, optimal systems have several important advantages, among them not being at the mercy of arbitrary rules about when information should be sought or how it should be used. Surprisingly complex joint optimal strategies emerge even in JOIST's simplified setting.

JOIST has been tested with realistic data, but not with real data. Realistically scaled problems can be solved in several minutes on a 450 MHz Wintel computer. An example is given below.

Since JOIST is optimizing and considers both information and firepower resources, it is potentially useful for making tradeoffs between those resources. Early indications are that the Information Age is still one in which it pays to know how to shoot—results are generally more sensitive to strike resources than to information resources in the (unclassified) scenarios considered so far. JOIST might also possibly be used operationally as part of an automatic Air Tasking Order construction package, since it deals with optimal asset allocation in a time-extended scenario. The main impediment to accomplishing either of these goals will be the acquisition of accurate data about sensors.

Background

The optimal use of information is difficult to model for two fundamental reasons. First, any policy for using information must take the mathematical form of a function, since the essence of the problem is to determine the best action as a function of the information. In most practical cases there are an extremely large number of functions, so the problem of finding the best one tends to be difficult except in special circumstances. Second, the effect of actions on the system to be controlled must also be modeled, since the meaning of the word "optimal" must involve the effect of decision making on the controlled system. For these reasons, models of optimal decision making with information tend to be crude. In practice, many modelers adopt reasonable but non-optimal decision rules to specify which action is to be taken; in other words, they adopt a particular one of the above mentioned functions, rather than searching for the best one.

One of the special circumstances where optimal decision policies can be found as a practical matter is the Partially Observable Markov Decision Process (POMDP). As long as the state of the process is not too complicated and evolves in a Markov manner over time, optimal policies can be found without having to list all the possibilities. POMDPs are still difficult to solve, but, with continual advances in computer power and the topical importance that comes with the Information Age, applications have become worth considering. One natural application area is the problem of Bomb Damage Assessment (BDA), since the Markov process representing the state of the target has only a small number of states (two, if one thinks of the target as being either live or dead), and also since the weapons employed to change the state of the target can be very expensive. Castenon (1997) describes one implementation of this idea. Yost (1998) develops a

similar model, except that the action costs required in solving a POMDP are replaced by constraints on resources enforced by a Linear Program (LP) in an iterative scheme. Yost and Washburn (2000) describe this as the LP/POMDP marriage. These efforts have been criticized for omitting the possibilities of lost track (targets sometimes move or submerge before they can be struck) and surveillance (military sensors are used to find new targets, as well as deal with old ones). An additional criticism is that even targets that are either "live" or "dead" can be in multiple states as far as observation is concerned. Dead targets sometimes appear to be still alive, for example, thereby inviting further unnecessary strikes unless the truth is exposed by observation. JOIST is intended to be responsive to these concerns.

Overview of the Marksman's Problem

JOIST models a situation where a collection of targets is to be destroyed or at least reduced in effectiveness by a "marksman." Targets are of multiple types, each of which is in either state 1, state 2, or state 3 at all times. The states are arbitrary as far as the method is concerned, but have been given specific meanings in what will be described as the "standard scenario." In the standard scenario state 1 is a virgin live target, state 2 is a target that is dead but not obviously so, and state 3 is a target that is more or less clearly dead, and a strike can in no case decrease a target's state. In addition, some targets are "tracked," and some are not. Actions can be taken only against tracked targets, with each action potentially having multiple effects. An action can change the state of a target, and may also provide information ("live," "dead," or "no report") about the new state. An action may also have the effect of causing a tracked target to become untracked, in which case no further actions against it may be taken in the future, or the surveillance effect of

discovering previously untracked targets that are then added to the list of targets against which actions may be taken in the future. In principle, a single action could have all of these effects.

Time proceeds in discrete periods, with the marksman choosing some action (possibly null) against each tracked target in each period. For each tracked target, at the end of each period, any information provided by the chosen action is incorporated into the Marksman's estimate of its state, possibly including the knowledge that the target has become untracked. There is a final period after which no further actions can be taken against any target, some limit on the total number of periods being a computational necessity. In practice, we would expect JOIST to be used with a "rolling horizon" where the last period is always a fixed distance from the time of the next action.

The object of JOIST is to provide *optimal* actions against every target at every opportunity. An objective function is therefore necessary. The JOIST objective is to maximize the expected net gain from the whole campaign. Each terminal state of each target has a reward associated with it, and the campaign reward is the sum of the individual rewards, one for each target that is tracked at the end. In the standard scenario the reward is the target's value if the terminal state is either 2 or 3, otherwise 0. Each action has a cost associated with it, and the campaign cost is the sum of the costs of all actions taken. The campaign net gain is just the difference between the campaign reward and the campaign cost.

JOIST is a constrained optimizer; that is, the choice of actions is limited by constraints on the availability of resources such as aircraft, weapons and sensors. These constraints are joint in the sense that they simultaneously influence the feasible action

choices for the entire collection of tracked targets, possibly including targets to be discovered in the future. These constraints are a significant computational complication, since otherwise an independent solution could be made at each target. The basic LP/POMDP solution method is to construct an independent POMDP solution for each target using prices for the resources obtained from the LP, incorporate the solution as a new column of the LP, and then solve the LP again in an iterative loop (see Yost and Washburn (2000)). JOIST uses this method.

Although JOIST is not a simulation, it does incorporate a post-optimization Monte Carlo simulator for purposes of conducting sanity checks on the policies determined to be "optimal." A sample output for a target worth 30 points is given in Table 1. The marksman is at each stage required to choose an action that is either null or a strike or a look. The three numbers in parentheses are the state probabilities just before the action is taken, as updated by the rules of Probability (Bayes' Theorem). The true state is also given for reference in Table 1, so both truth and the marksman's perception are available for the analyst. The marksman, of course, is not allowed to know the true state.

Look actions do not transform the target state, but do report on it and have the side effect of possibly discovering new targets. Thus, the first two actions in Table 1 (looks by Sensor 1) are not foolish, even though the state is known for certain to be 1 initially, but rather represent failed gambles to discover new targets. The third action (Mission 157 out of a long list of legal strikes) sends the target to state 3, but the marksman does not know that so he employs Sensor 9. Sensor 9 eventually provides a Dead report, but not before discovering a new target of the same type (the indented text), etc. In this

replication two targets are killed, but the same policy could result in a different outcome (possibly much more complicated than the one shown) in a different replication. JOIST averages over all of these possible outcomes in the process of determining the optimal strategy.

```
Target value 30
State 1, Period 1 (1.000 0.000 0.000), Sensor 1, Cost 0.10
Live Report
State 1, Period 2 (1.000 0.000 0.000), Sensor 1, Cost 0.10
Live report
State 1, Period 3 (1.000 0.000 0.000), Mission 157, Cost 0.11
State 3, Period 4 (0.135 0.015 0.850), Sensor 9, Cost 0.24
  Found target 1, value 30
  State 1, Period 5 (1.000 0.000 0.000), Mission 163, Cost 0.19
  State 3, Period 6 (0.115 0.035 0.850), Mission 148, Cost 0.88
  State 3, Period 7 (0.016 0.027 0.957), No Action
  Net gain from target 1 is 28.94
Dead Report
State 3, Period 5 (0.028 0.010 0.962), Sensor 2, Cost 0.10
Dead Report
State 3, Period 6 (0.009 0.007 0.984), No Action
State 3, Period 7 (0.009 0.007 0.984), No Action
Overall Net Gain 58.2851
```

Table 1: Output from JOIST's post-optimization simulator. The indented text is events related to a second target of type 1 found by Sensor 9.

A Small Constrained Example

JOIST manipulates *policies* for dealing with targets. Policies are difficult to illustrate, even in small examples, because each policy must deal with all possibilities for what the information might be. The same policy that produced the replication shown above, for example, is also prepared to deal with the first use of Sensor 9 producing a Live report, rather than a Dead report, or not finding a second target of type 1, or any of the other possibilities in a potentially large tree. In order to avoid this branching, in this section we construct a small example involving "sensors" that do not provide information. Even

though they provide no information about the current target, they still have an important surveillance effect (in fact, the surveillance effect is the only reason for employing them).

Suppose there are only two target types, initially 400 of type A and none of type B, with three periods available for attacking them. Each target is worth 100 points, and is either live or dead at all times. A strike against an A target will kill it with probability .5, while a strike against a B target will certainly kill it. All strikes cost 30, so a strike against either target type is profitable. Either target may also be looked at, rather than struck. The entire effect of a look is to expose one additional target of each type, with no associated BDA report as to the state of the examined target. Each look costs 1. Since B targets are as valuable as A targets and easier to kill, the marksman has a conflict as to whether to immediately set to work striking the A targets, or to "look" in order to expose some more lucrative B targets that can be struck at a later time. Consider three possible policies for handling one of the 400 A targets:

- **Policy 1:** Look at the A target in period 1, thereby exposing one additional A target and one B target. In period 2, kill the exposed B and look at the two A's, the newly exposed A and the original A. In period 3, kill the two B's exposed by the two looks in period 2 and ignore the four A's. The net effect is to kill three B's at the cost of three strikes and three looks, with an associated net gain of 207.

- **Policy 2:** Look at the A target in period 1, kill the exposed B in period 2, and take no further action. The net gain is 100-30-1=69.

- **Policy 3:** Strike the A in period 1 and take no further action. The net gain is 100(.5)-30=20.

With no further constraints on the marksman's actions, Policy 1 is superior to the other two, and the campaign's total net gain is 400(207)=81,400. However, consider the effect of adding a constraint that there can be no more than 300 strikes in any period. Constraints of this form are realistic, and JOIST has been formulated specifically to deal with them. Policy 1 is now infeasible if applied to all 400 initial targets, since it results in 800 strikes in period 3. With this constraint, the optimal overall policy from JOIST turns out to be to employ Policy 1 on 150 targets, Policy 2 on 150 targets, and Policy 3 on the remaining 100, for a total gain of 43,400. Note that this results in 300 strikes in both periods 2 and 3. The dual values for these two constraints are each 20, so the effective cost of a strike in either period is 50, rather than 30, and it is this value that is used in solving the POMDP in the last iteration.

A more practical example would have effective BDA and much less effective surveillance, but the policies would be much harder to describe and illustrate.

Mathematical Formulation

The central part of JOIST is a POMDP involving decisions about what kind of action to take at each stage of a decision process extended in time. Time is counted backwards from the last stage, at which time no further actions are possible. At the last stage, the JOIST objective function is simply a sum over all tracked targets of the target values appropriate to the target state. At all other stages, the price of the action taken is subtracted; that is, the objective function at all stages has the meaning "expected net gain over this and all succeeding stages." The action prices include the dual variables of the resource constraints. The mechanism for accomplishing this conversion of constraints to prices is exactly as in Yost (1998) (see also Yost and Washburn (2000)), so it will not be

repeated here.

Let π be a probability vector over all of the possible states for some tracked target, the "state vector" of the target, and let $V_t(\pi)$ be the optimal expected gain obtainable from stage t onward. In the usual POMDP formulation, $V_t(\pi)$ is initialized for t=0 and then obtained for larger values of t through the relationship

$$V_{t+1}(\pi) = \max_a \sum_i \pi_i \{-c(i,a) + \sum_j p_{ij}^a \sum_k r_{jk}^a V_t(\pi'(\pi,a,k)\}, \text{ where} \qquad (1)$$

- a is the action chosen, with the set of possibilities possibly depending on t
- i and j are the target states before and after the action at stage t
- p_{ij}^a is the probability that state j succeeds state i if action a is taken, summing to 1 on j for all i and a
- r_{jk}^a is the probability that message k is received after the action at stage t+1 and before the action at stage t, summing to 1 on k for all j and a
- $\pi'(\pi,a,k)$ is the state vector, obtained by conditioning on the observation k and applying Bayes' Theorem. Conditioning on k corresponds to the assumption that the observation k is known when the decision at time t is made.

Equation (1) states that the total expected gain at stage t+1 is the sum of two parts: the present cost $c(i,a)$ and the optimal expected profit starting from the new state π' at time t. Since $c(i,a) \geq 0$, selection of the best action can be viewed as making a present sacrifice in order to obtain a better future state vector. The set of feasible actions always includes the null action a* for which $c(i,a^*) = 0$. This action leads to the null observation k* for which $\pi'(\pi,a^*,k^*) = \pi$, i.e., the null action produces neither information nor a state

change, and therefore no change in the state vector.

Equation (1) actually permits considerable generalization. The equation would remain true if r_{jk}^{a} were also subscripted for the initial state i; indeed, the only important object in (1) is the joint probability $p_{ij}^{a} r_{jk}^{a}$, so only one triply subscripted array is needed. All arrays could also be time-dependent, although this would make hopeless the quest for stationary policies that has motivated most theoretical work on POMDPs. These generalizations are not made in JOIST for lack of supporting data, but one further generalization is central to JOIST's ability to handle surveillance and tracking.

Although p_{ij}^{a} is normally assumed to be a stochastic probability matrix, as stated above, JOIST instead interprets p_{ij}^{a} to be the expected number of tracked targets in state j in the next stage, if action a is applied to one tracked target in state i. Thus, p_{ij}^{a} is not necessarily a probability, and it is only required that $p_{ij}^{a} \geq 0$. Equation (1) is now justified by the Conditional Expectation Theorem instead of the Theorem of Total Probability, but remains unchanged in form. This interpretation of p_{ij}^{a} is actually not new, since Bellman (1957) had already allowed for it. See also Rothblum (1984). Although this change of interpretation is almost trivial computationally, it provides considerable additional modeling flexibility. If p_{ij}^{a} is substochastic (sums on j to less than 1), then the defect from 1 can be interpreted as a track-loss probability. In this way, "overt" actions might cause a target to disappear, or at least to disappear with greater likelihood than "covert" actions. If a particular action has a "surveillance" effect of discovering new targets, then p_{ij}^{a} may also sum on j to more than 1.

It is a cheap, but useful, generality to permit each target to have a "class" m, in which case an action taken on a target in state m may, through surveillance, discover targets in other classes, possibly including class m itself. JOIST assumes that newly discovered targets, whatever the class, always have a state vector π_0 that corresponds to state 1. Let s^a_{imn} be the average number of targets of type n discovered by action a taken on target m in state I. Then the generalized iterative formula employed by JOIST is

$$V_{m,t+1}(\pi) = max_a \sum_i \pi_i \{-c(i,m,a) + \sum_j p^a_{ijm} \sum_k r^a_{jkm} V_{mt}(\pi'(\pi,a,k)) + \sum_n s^a_{imn} V_{nt}(\pi_0)\}. \quad (2)$$

Introducing the surveillance array s^a_{imn} permits p^a_{ijm} to refer only to the state of the target under consideration, so p^a_{ijm} is substochastic when summed on j.

In summary, to get from equation (1) to equation (2), add a target subscript to the indicated arrays and functions, and add a surveillance term to account for newly discovered targets.

JOIST Database

The JOIST database consists of five comma-delimited input files. For historical reasons there are two files for actions, with mission.csv corresponding to "strike" actions that can potentially modify the state of the target, and sensor.csv corresponding to "look" actions that can only provide information. Only looks can have a surveillance effect, and the surveillance array s^a_{imn} depends only on a and n, specifically $s^a_{imn} = (\text{surv}_a)(\text{duty}_n)$. The surveillance concept is essentially random search, with surv_a being a sensor property having to do with area coverage, and duty_n being a target property having to do with target density and tendency to be exposed.

The entirety of the sensor.csv file in the standard scenario is shown in Table 2:

Name	Number	Rate	Live/1	Live/2	Live/3	Delay	Prep	BasCst	Surv
1	4	240	0.7926	0.5	0.2585	0.33	1	0.1	0.1
2	4	293.04	0.778	0.5	0.2585	0.33	1	0.1	0.1
3	20	1.5	0.7927	0.5	0.2793	0.33	1	0.1	0.1
4	6	33	0.7821	0.5	0.3194	0.33	1	0.1	0.1
8	12	180	0.7724	0.5	0.2933	0.33	1	0.1	0.1
9	8	60	0.8466	0.5	0.1669	0.33	1	0.1	0.1
11	2	21.12	0.68	0.5	0.3246	0.33	1	0.1	0.1
12	4	100	0.809	0.5	0.2119	0.33	1	0.1	0.1
15	4	105	0.8349	0.5	0.2133	0.33	1	0.1	0.1
18	96	0.16	0.5	0.5	0.5	0.33	0	0.1	0.2

Table 2: The standard sensor input file sensor.csv.

The meanings of the columns in Table 2 are as follows:

Name An arbitrary name or number for the sensor type.

Number The number of sensors available.

Rate The number of looks per day.

Live/1 The probability of a "live" report given that the target is in state 1.

Live/2 The probability of a "live" report given that the target is in state 2.

Live/3 The probability of a "live" report given that the target is in state 3.

Delay The time delay in days until the sensor report is available.

Prep The probability that a report is actually made.

BasCst The basic cost of the action is part of $c(i,m,a)$ in equation (2).

Surv The surveillance parameter as explained above.

Columns 2 and 3 are used in computing the number of looks available in any given stage, thus determining the right-hand side of one constraint in the linear program. The dual variable for this constraint is the other part of the action cost $c(i,m,a)$ in equation (2). Note that there are actually three report possibilities: "live," "dead," and "no report (null)." The probabilities in columns 4-6 are conditional on a report being made, and

column 8 is the probability of a report. State 1 is the "live" state for a target, so the possibility of error is apparent in column 4. Column 7 is not currently used; instead, each look is assumed to require a single stage to process.

Table 3 shows the beginning of the standard mission input file. Each mission applies only to a single target, so there are many more mission types than look types. The first three columns give the target, aircraft, and weapon type involved in the mission (the number of weapons is shown in the "Load" column). The next five columns give transition probabilities from state i to state j, with j always at least as large as i because attacking a target cannot decrease its state. Attacking a target may also provide information about its state after the attack, as shown in the next four columns. The number of sorties involved ("Sorts") is usually 1, but may be less than 1 for missions that attack multiple targets. The probability of losing the aircraft per sortie is shown in the last column, and plays a role in the attrition constraint for the aircraft type.

Missions are not directly constrained, nor do they have an associated basic cost. The reason for this is that constraints and basic costs are imposed on weapon usage, sortie usage, and attrition. Mission costs are then inferred depending on the use of those scarce resources.

Tgt	Acft	Wpn	p11	p12	p13	p22	p23	Prep	Live/1	Live/2	Live/3	Load	Sorts	Attrit
1	1	1	0.56	0.3	0.14	0.68	0.32	1	0.8	0.5	0.2	6	1	0.062
1	1	3	0.755	0.105	0.14	0.44	0.56	0	0	0	0	6	1	0.013
1	1	7	0.395	0.375	0.23	0.52	0.48	0	0	0	0	2	1	0.062
1	1	40	0.91	0.02	0.07	0.28	0.72	0	0	0	0	4	1	0.06
1	3	1	0.615	0.245	0.14	0.36	0.64	0	0	0	0	6	1	0.008
1	3	3	0.64	0.22	0.14	0.68	0.32	0	0	0	0	6	1	0.009

Table 3: The first 6 of 2891 missions in the standard mission.csv file.

Table 4 shows the target file. Each target has an arbitrary name (a simple number in the standard scenario), an initial quantity, and a value that constitutes the marksman's reward if its terminal state is either 2 or 3. After every action, even the "null" action, a target in state 1 vanishes (track is lost) with probability (1- existence), in which case all subsequent opportunities to kill it also vanish. Since the existence parameter is set to 1 for all targets in the standard database, targets do not vanish in the standard case. The "Duty" column is as described earlier.

Name	Number	Value	Existence	Duty
1	360	22	1	0
2	167	30	1	0
3	126	30	1	0
4	49	30	1	0
5	11	7	1	0.05

Table 4: The first 5 of 40 target types in the standard target.csv file.

All missions (strikes) involve the use of scarce aircraft and weapons. The number of aircraft sorties used in any period is limited by the rate at which sorties can be generated, and the total attrition over all stages is also constrained not to be too large for each aircraft type. Table 5 shows the aircraft file aircraft.csv.

Name	Sortrate	Number	Maxloss	Bassort	Basattrit
1	1.84	96	4	0.1	10
3	0.23	16	1	0.1	10
4	0.23	12	1	0.1	10
5	1.2	24	1	0.1	10
6	0.9	36	2	0.1	10
7	0.99	102	4	0.1	10
8	1.29	114	4	0.1	10
9	1.42	111	4	0.1	10
10	0.23	72	3	0.1	10

Table 5: The file aircraft.csv in the standard case.

The sortie generation rate for the type is the product of the *Number* of aircraft available and the *Sortrate* per aircraft. The *Maxloss* column is the total number of aircraft that can be lost over the planning horizon. The next two columns are the direct or "basic" costs per sortie and per lost aircraft.

Finally, Table 6 shows the data in weapon.csv. Each weapon type is constrained by the number available, and, in addition, has a basic cost per weapon for employment.

Name	Number	Basiccost
1	223412	0
2	223412	0
3	223412	0
4	223412	0
7	1	1
8	1	1
9	27255	0
10	27255	0
11	1	1

Table 6: The first 10 of 42 weapon types in the weapon.csv file.

Analysis of the Standard Scenario

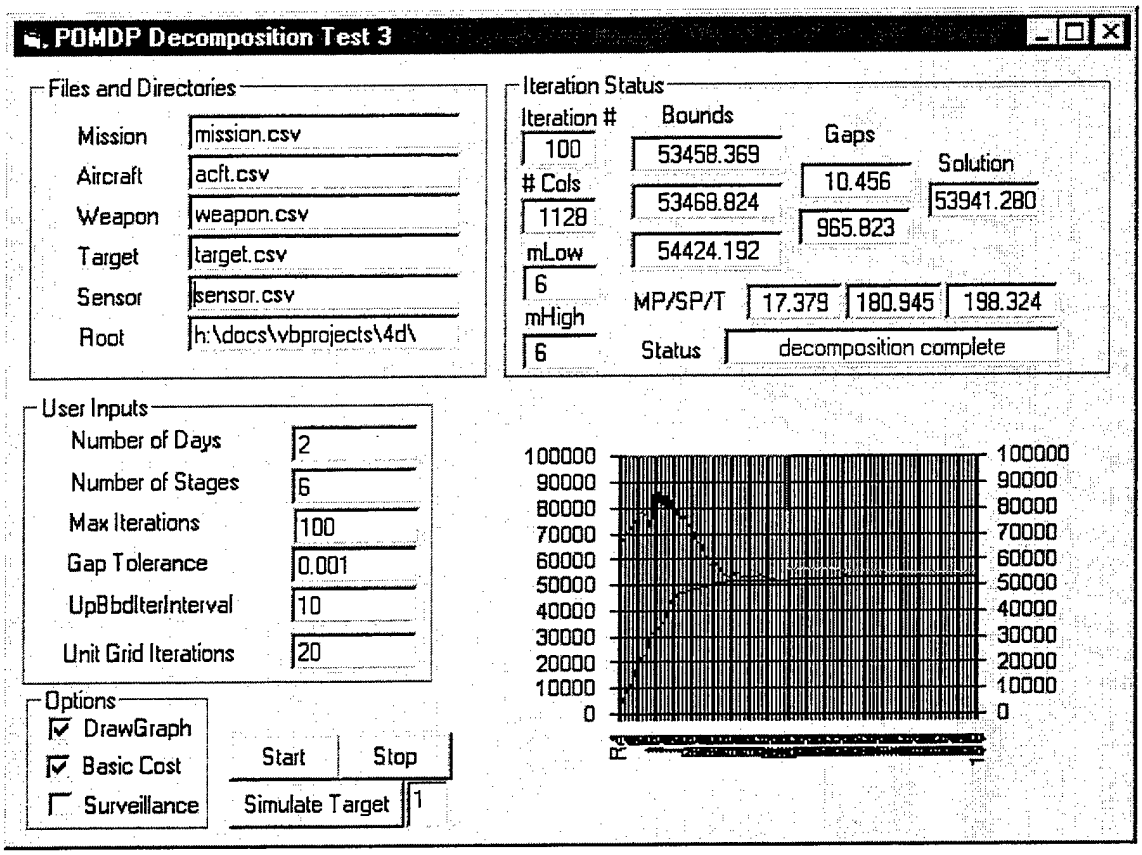

Figure 1: The JOIST user interface after completion of computations.

Figure 1 shows the JOIST Visual Basic GUI after the completion of 100 iterations of a six-stage scenario. All surveillance has been turned off in this scenario (note the unchecked box). There are 40 target types, 5019 targets, and a total target value of 67441. After 100 iterations, the optimal net gain is known to be somewhere between 53458 and 54425. The total computation time for the 100 iterations is 198 seconds on a 450 MHz Wintel computer, most of which (181 seconds) is taken up by POMDP computations.

The graph in Figure 1 shows that 100 iterations are actually more than necessary. The lower curve is the LP objective function, a lower bound on the total net gain. This curve starts near zero because the LP starts with only a single policy (column), and then

increases because successive POMDP solutions generate increasingly numerous and effective policies, each with a corresponding column for the LP. The upper curve is an upper bound based on a Freudenthal grid of three-dimensional probabilities as suggested by Lovejoy (1991). The probability interval [0,1] is subdivided into mHigh equal intervals, with the initial value of mHigh being 1 and (in Figure 1) the final value being 6. The upper bound is not evaluated at every iteration because the evaluation is time consuming, hence the staircase nature of the graph in Figure 1, which descends sharply from 100,000 after about 50 iterations. The gap between lower and upper bounds is already small after only 50 iterations, and most of the higher iterations are devoted to marginal reductions. This is typical of column-generation schemes for solving large linear programs. Columns are generated by solving the POMDP at a grid of state vectors, with mLow playing the same role as mHigh. Each solution of the POMDP also generates a local upper bound on what the LP can achieve, the word "local" being needed because the grid does not exhaust all state vectors. The middle curve shows this local upper bound.

The inclusion of surveillance is an important feature of JOIST, but unfortunately it comes at a significant computational cost. Turning on surveillance in this scenario eventually results in establishing that the net gain is somewhere between 55317 and 55913—only a small gain because most of the *surv* and *duty* numbers in the database are either zero or small. Nonetheless, establishing these bounds requires about 700 iterations and 1500 seconds, which is a large increase compared to the time needed when surveillance is turned off. Future development of JOIST will address this issue.

References

Bellman, R. 1957. *Dynamic Programming*, Princeton University Press, page 318.

Castenon, David A. 1997. "Approximate dynamic programming for sensor management," in *Proceedings of the 36th IEEE Conference on Decision and Control*, Vol. 2, IEEE Control Systems Society, Danvers, MA, pp. 1202-1207.

Lovejoy, W. "Computationally feasible bounds for Partially Observable Markov Decision Processes," *Operations Research* Vol. 39, No. 1, pp. 162-175.

Rothblum, U. 1984. Multiplicative Markov Decision Chains, *Mathematics of Operations Research,* Vol. 9, No. 1, pp. 6-24.

Yost, K., *Solution of large-scale allocation problems with partially observable outcomes*, PhD thesis, Naval Postgraduate School, September 1998.

Yost, K. and Washburn, A. 2000. "Optimizing Assignment of Air-to-Ground assets and BDA sensors," Military Operations Research, Vol. 5, No. 2, pp. 77-91.

Initial Distribution List

1. Research Office (Code 09)..... 1
   Naval Postgraduate School
   Monterey, CA 93944-5000
2. Dudley Knox Library (Code 013).....2
   Naval Postgraduate School
   Monterey, CA 93943-5002
3. Defense Technical Information Center.....1
   8725 John J. Kingman Road
   Ft. Belvoir, VA 22060-6218
4. Professor Alan Washburn (Code OR/Ws).....2
   Department of Operations Research
   Naval Postgraduate School
   Monterey, CA 93943-5000
5. Richard Mastowski (Editorial Assistant).....2
   Department of Operations Research
   Naval Postgraduate School
   Monterey, CA 93943-5000
6. Major Juan Vasquez.....1
   AFOSR/NM
   801 N. Randolph Street, Room 933
   Arlington, VA 22203-1977
7. Computer Science Department.....1
   Box 1910
   Brown University
   Providence, RI 02912-1210
8. Institute for Defense Analysis.....1
   ATTN: Philip J. Walsh
   1801 N. Beauregard Street
   Alexandria, VA 22311-1772
9. Steve Robinson.....1
   I.E. Department
   University of Wisconsin
   Madison, WI 53706-1539
10. Lt Col Kirk Yost, USAF.....1
    Headquarters USAF
    The Pentagon
    Washington, DC 20330