2012 IEEE 19th International Conference and Workshops on Engineering of Computer-Based Systems

Decision Support Using Deterministic Equivalents of Probabilistic Game Trees

Michael L. Valenzuela, Liana Suantak, Jerzy W. Rozenblit
Electrical and Computer Engineering Department
University of Arizona
Tucson, AZ 85721
{*mvalenz,liana,jr*}*@ece.arizona.edu*

***Abstract*—We have developed a game-theory driven decision-support tool that builds probabilistic game trees automatically from user-defined actions, rules, and states. The result of evaluating the paths in the game tree is a series of decisions which forms a decision-path representing an ε-Nash-Equilibrium. The algorithm uses certainty-equivalents to handle trade-offs between expected rewards and risks, effectively modeling the probabilistic game tree as deterministic. The resulting decision-paths correspond to player actions in the scenario. These sets of actions can be used as search patterns against a real-world database. A match to one of these patterns indicates an instance of novel behavior patterns generated by the game-theory driven decision support tool. This particular paradigm could be applied in any domain that requires anticipating and responding to adversarial agents with uncertainty, from mission planning to emergency responders to systems configuration.**

***Keywords*-decision support; repeated game theory; certainty equivalents; game simulation; risk aversion**

I. Introduction

Decision Support Systems (DSS) have been developed for many fields [1], used for medical purposes [2], [3], and even addressed issues in agriculture [4]. They have been applied to stability and support operations [5], [6], produced for network security [7]–[9], created to aid intelligence analysts [10], and provided analysis of web traffic [11]. Decison Support Systems are characterized by handling vast amounts of information, taking many factors into account, and providing expert advice all in a relatively short time. They explore alternatives and learn from and refine models. An important characteristic of a DSS is to provide consistent analysis, allowing users to refine their own decision-making processes in response to the analysis of the system.

The decision support system introduced in this paper is a game theoretic system that is both configurable and consistent. The user configures the game by specifying actions, rules, effects of actions, and desired outcomes. By allowing the user to configure these parameters, the game can take the form of an adversarial competition, a cooperative advancement toward a shared goal, or some mix of the two. Furthermore, the algorithm that chooses among alternatives models probabilities using certain equivalents, resulting in consistent and reproducible outcomes.

This paper continues with Section II briefly reviewing game theory (including game trees and the handling of uncertainty). Section III explains the use of deterministic equivalents of probabilistic game trees and refinements over previous methods. Section IV describes the DSS in detail. Finally, we conclude in Section V with a brief summary and future research directions.

II. Background

We demonstrate our approach on ATRAP (Asymmetric Threat Response and Analysis Program), an application developed at the University of Arizona. We built into ATRAP a framework for building games, finding optimal responses, exploring alternative responses, and inspecting the results to allow for human validation of the game. Thus, using our framework in an iterative process, games can be refined to include more strategies, more accurate outcomes of strategies, etc. Furthermore, the results from this decision support tool can be processed by other tools within ATRAP [10].

A. Game Theory

This section presents a brief review of game theory that is relevant to our application. In game theory there are payoffs, strategies, and players. The payoffs describe the desirability of a certain outcome for each player and are defined for all combinations of strategies. The decisions made by a player can be modeled as a tree of possible moves with alternating branches being turns, known as a game tree.

1) Game Trees: As each player takes turns evaluating the payoffs of their strategies and choosing the respective action, the turn can be represented as choosing one from a set of alternatives. A turn in a game tree can be simultaneous, sequential, or some other hybrid. Typically, each node in the game tree represents a state and each edge represents a possible action. For most interesting games, the game tree is too large to store in computer memory. Therefore the game tree is often pruned and only expanded out according to the system's resources.

This leads to each player's outcome being dependent on all the other players' decisions. In other words, the decisions at every turn determine a path through the tree of alternatives. This path can be termed the *decision path*. The DSS described in this paper produces a decision path that can be expanded and collapsed to show the result path as

978-0-7695-4664-3/12 $26.00 © 2012 IEEE
DOI 10.1109/ECBS.2012.22

well as the paths not taken. This can be useful for explaining, forecasting, and aiding decisions.

2) Game Theory Criticisms: There are some common criticisms of game theory. One of the most well known criticism of game theory is the traditional, unrealistic assumption that players are rational, choosing the optimal strategy. However, as long as there is a well-defined decision process (e.g., bounded-rationality), even if it is not optimal, then the game can still be anatomized [12]. Thus this criticism is unsound.

Another criticism which cannot be refuted, but may be mitigated, is the considerable amount of information needed to construct such as game; one rarely knows perfectly all the possible actions (strategies), their related payoffs to all the players, and how the players make decisions. However, many possible actions can be determined by observing historical outcomes of similar games. Additionally, the payoffs are usually known to some degree and according to game theory, only relative payoffs matter.

The last objection to game theory that will be addressed here is that game theory often suggests it is rational to take advantage of other players when possible, but ignores the long range consequences. There are two responses to this criticism. First, the payoff function should reflect the total utility of the situation including, for example, long-term damage to the reputation of a player who exploits another. The other response is that realistic scenarios, if viewed from the stand point of a game, is not a single stage game, but a repeated game. In order to analyze repeated games, game trees are used.

III. Deterministic Equivalents

Real-world situations always have some degree of uncertainty; players may not always realize all their available moves, they may be acting in only a near optimal fashion, or there may be some random elements. Probabilistic game trees can model this uncertainty. One technique to model "moves by nature" (random effects), is to introduce a player without any payoff and whose choices are stochastic. To address players acting in a heuristic fashion, adversaries can be modeled as taking the best move probabilistically. This is the approach taken in [9]. However, due to the probabilistic nature, each possible strategy has a distribution of payoffs (outcomes). To rapidly and deterministically analyze the uncertain outcomes, certainty equivalents (also called the deterministic equivalents) are computed from the expected value and variance.

Deterministic equivalents provide an efficient method for approximating expected utility. A utility function $u_i(x_a)$ describes the usefulness of the payoff. For example, one dollar is worth more to a college student than to a millionaire. Thus, maximizing utility is a more realistic and long-range method of making decisions.

Distributions can be described by their expected value as well as higher order central moments [13]. While only the expected value of the payoff matters with a linear utility function, risk may also be an important factor to consider. The expected utility of outcomes can then be approximated as

$$E[u_i(x_a)] \approx E[x_a] - \alpha_i E[(x_a - E[x_a])^2] = \mu_a - \alpha_i \sigma_a^2, \quad (1)$$

where x_a is the payoff distribution for the action a, μ_a and σ_a^2 are the expected value and variance of the payoff distribution, and α_i is the player's risk aversion. This approximation allows for a deterministic analysis of risks and rewards. Players who are risk adverse have $\alpha_i > 0$. Similarly, a player who is risk seeking would have $\alpha_i < 0$. $\alpha = 0$ describes a player who is risk neutral.

A. Overview of the Proactive Defense Strategy

We built upon the deterministic equivalent of probabilistic game trees for network security by Luo et al. [14]. This work focuses on finding optimal strategies in a probabilistic repeated game. There are two players, the defender and the attacker, each with their own risk aversion factor α. In this game, the attacker and defender have different actions available to them. The variable τ represents a sliding window of the number of moves the players are looking ahead. After the defender and attacker determine their moves, the game is advanced to a new stage. The current stage of the game is denoted as d for decision node.

According to the algorithm from Luo et al, let k_a be one possible path (of length τ) through the game tree originating with action a. Also let m represent the number of interactions into the future that are currently being examined. Moreover, let $p_{k_a}^{d+2m-1}$ be the prior probability that the defender chooses a response along k_a at stage $d+2m-1$. Likewise, $p_{k_a}^{d+2m}$ is the probability that the attacker chooses a subsequent action along the path k_a at stage $d+2m$. In Luo et al.'s work $p_{k_a}^{d+2m-1}$ (attacker's assumptions of the defender's probabilities) can be determined from historical data and $p_{k_a}^{d+2m}$ (attacker's moves) are determined by (5).

One potential problem with using deterministic equivalents as described by (1) is that a large risk aversion and large variance can lead to the selection of dominated solutions. An action is dominated when its best possible outcome is worse than another action's worst possible outcome (i.e. $a1$ is dominated when $max(x_{a1}) < min(x_{a2})$). Hence to avoid accidentally selecting dominated solutions, dominated actions are pruned.

The first step in calculating the probability of each of the attacker's possible actions is to calculate the expected value and variance. The expected value is obtained from

$$E[x_a] = \sum_{k=1}^{K} y_{k_a} Q_{k_a}, \quad (2)$$

where K is the total number of possible outcomes given the action a, y_{k_a} is the payoff for the specific path k_a, and

$$Q_{k_a} = \begin{cases} \frac{\prod_{m=1}^{\tau} p_{k_a}^{d+2m-1}}{\sum_{k=1}^{K} \prod_{m=1}^{\tau} p_{k_a}^{d+2m-1}} & \text{if attacker's turn} \\ \frac{\prod_{m=1}^{\tau} p_{k_a}^{d+2m}}{\sum_{k=1}^{K} \prod_{m=1}^{\tau} p_{k_a}^{d+2m}} & \text{if defender's turn} \end{cases}. \quad (3)$$

The variance is calculated from

$$V_a = \sum_{k=1}^{K} y_{k_a}^2 Q_{k_a} - E[x_a]^2. \quad (4)$$

Then the expected value and variance are combined according to (1) to produce U_a. Finally the probability of attacks at the current decision node is given by:

$$pp_a = \frac{U_a}{\sum_{a=1}^{A} U_a}, \quad (5)$$

Where A is the total number of available actions. The above steps with minor modifications can also be applied to find the defender's best moves. The current player then makes a move reflecting his/her optimal action. This advances the current decision, d, down the game tree one step. The game then continues for the other player, each side taking turns.

The above algorithm can be adapted to find ε-Nash equilibria by calculating the pp_a for both the defender and attacker. This would remove the dependence on historical data. Due to the nature of Q_{k_a}, the pp_a would have to be calculated from the leaves of the game tree up. Naturally there are many interesting variations and possible extensions.

B. Algorithm Extensions

Before discussing extensions to the above algorithm, we will show how it is related to the traditional AI minimax algorithm. The minimax algorithm can be rederived through three steps. First we use the Nash equilibria extension as discussed earlier. This will make all probabilities dependent on analysis only, removing the dependence on historical data. Then, we add some positive constant C to U_a such that $U_a + C > 0$. Lastly, we replace (5) with

$$pp_a = \lim_{\beta \to \infty} \frac{(U_a + C)^\beta}{\sum_{a=1}^{A} (U_a + C)^\beta}. \quad (6)$$

As $\beta \to \infty$, an action will have nonzero probability if and only if

$$U_a = \max_a U_a.$$

Note that the payoff is now deterministic so the risk attitudes α have no effect. This rederives the minimax algorithm except that it randomly chooses one of the best choices when there are multiple options.

We extend and modify the algorithm in [14] in the following ways. First, the games are not confined to zero-sum games (the payoffs are equal and opposite). This method allows the players to not necessarily be in direct conflict. This configuration is achieved by letting each player have its own payoff which it tries to maximize. Another minor extension was to rectify some numerical instability. Equation (4) replaced with the more numerically stable single pass algorithm suggested by West [15] for more accuracy in calculating the variance.

We also addressed the possibility that U_a could be negative due to a negative expected value or large risk. A negative U_a in (5) would produce a negative output, which cannot be used as a probability. Thus we decided to modify how the probabilities are calculated. There are two common approaches for removing negativity. The first approach is to use a softmax method using a Gibbs/Boltzman distribution [16]:

$$pp_a = \frac{\exp(U_a/T)}{\sum_{a}^{A} \exp(U_a/T)}, \quad (7)$$

where T is the temperature. The lower the temperature, the more optimal the player's strategy. Another approach is to normalize all the U_a using

$$U'_a = \frac{U_a - \min_a \{U_a\} + T}{\max_a \{U_a\} - \min_a \{U_a\}}, \quad (8)$$

where T provides behavior similar to temperature in (7). Then use the new U'_a in (5) to produce the path probabilities.

We also introduced a state-space to allow for a higher level description of the state of the game. This state-space can be any number of dimensions. For example, the state-space could describe resources such as player money, information, and reputation. This allows each action to impact multiple variables. As a consequence of this state-space, payoff functions for each player must be specified as a function of the state-space. In an n person game with r resources (r-dimensional state-space) there are $r \times n$ variables each player could consider.

In the next sections, we will describe the decision support tool that implements the algorithm and modifications described above.

IV. Implementation

Within the ATRAP framework, resides the Game Builder (see Fig. 1). This is where games can be specified, saved, and loaded.

A. Game Setup

The tool allows for construction of games via specifying:

- action sets (which moves are available),
- rule sets (when are moves valid),
- two players (initial conditions, payoff function, and risk aversion),
- a cost/benefit model (the state-space in which actions interact),
- player assumptions (prior probabilities from historical data) and
- the number of moves to look ahead.

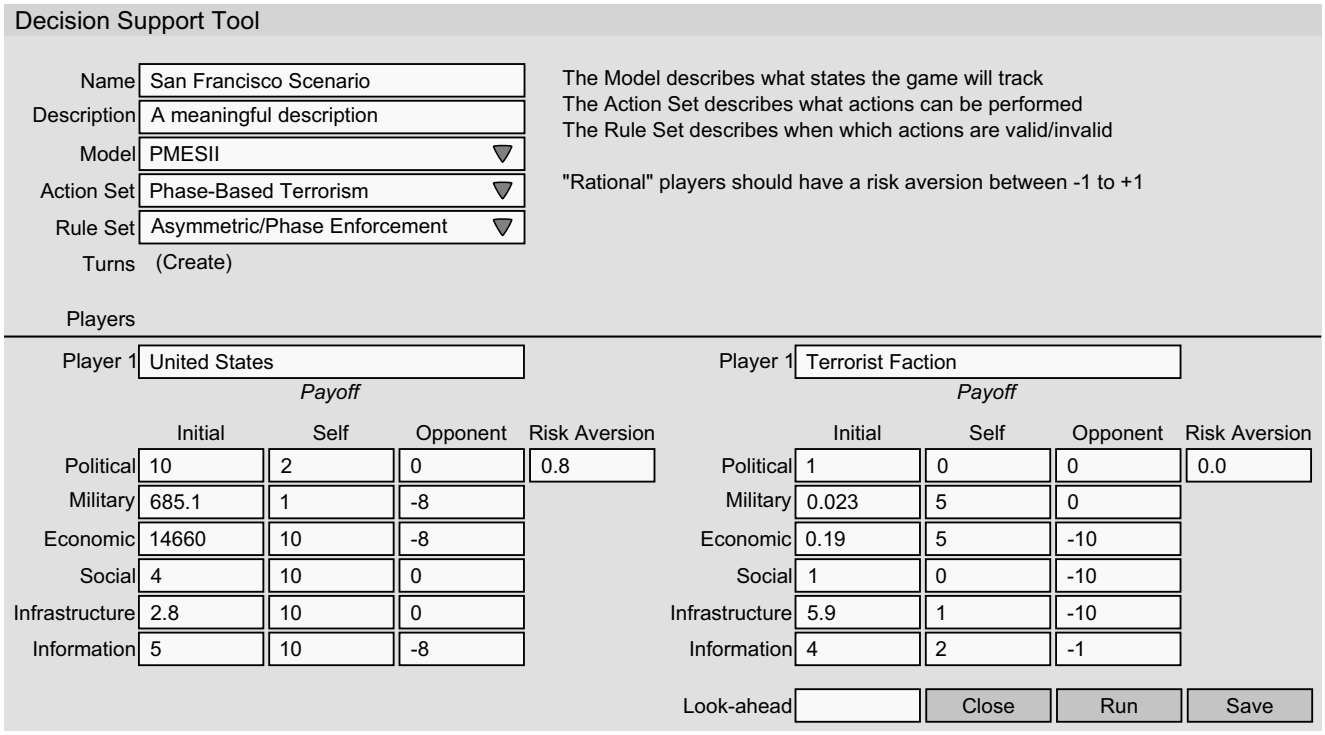


Figure 1. Game Builder home screen

1) Cost/Benefit Models: In evaluating moves, a player must weigh the costs and benefits of every possible action. They must be interpreted in a model that encompasses all of the important factors impacting a decision. The Game Builder tool allows for the selection of typical six-ring (six-dimensional) analyses, which include the PMESII (political, military, economic, social, infrastructure and information systems) [17] and ASCOPE (areas, structures, capabilities, organizations, people, events) models [18]. However, this could easily be extended to several other domains and more dimensions. Mathematically, the cost/benefit model is simply the space of $\mathbb{R}^6$ and names for each dimension.

2) State-Space: The state-space describes the state of the game at a particular turn. The state-space is a collection of real numbers describing the cost/benefit resources for each player. Formally, the state space is $(s|s \in \mathbb{R}^{n\times 6})$, where n is the number of players. Our tool fixes n at two.

3) Action Sets: The action set describes the actions or moves in the game. Each action set operates on a specific cost/benefit model. Precisely, action sets are a 2-tuple: the set of actions and an associated cost/benefit model. The action set is built from the tool shown in Fig. 2.

Actions are the individual moves. Each action has an impact on the state of the game, taking a previous state-space and transforming it into a new state-space. Each action is represented by a 12×13 matrix (more generally $2r\times 2r+1$). The matrix represents an affine transformation of

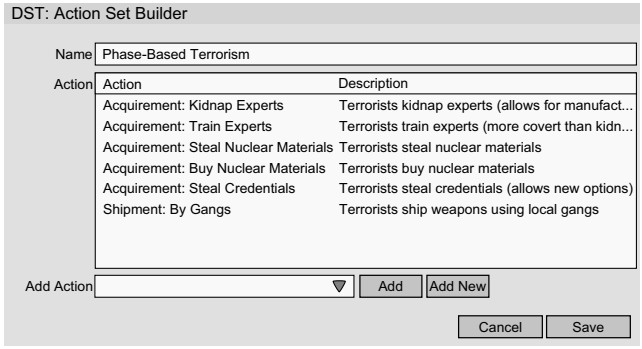


Figure 2. Constructing an action set

the previous state-space. In other words, the first 12 columns perform a linear transformation on the state-space and the last column is added to the result. The matrix is applied to the left-hand side of a column vector describing the state-space (see Fig. 3). While a 12×13 matrix may seem like it requires a lot of data entry, these matrices are typically sparse with ones on the main diagonal.

4) Rule Sets: Rules describe when actions are valid or invalid, and may be triggered when a player performs some action. The rule may allow or disallow other actions for one or more players, for some number of turns. A rule may also be initially active, not requiring a player to trigger it. Initial rules are useful for specifying asymmetric actions (only one

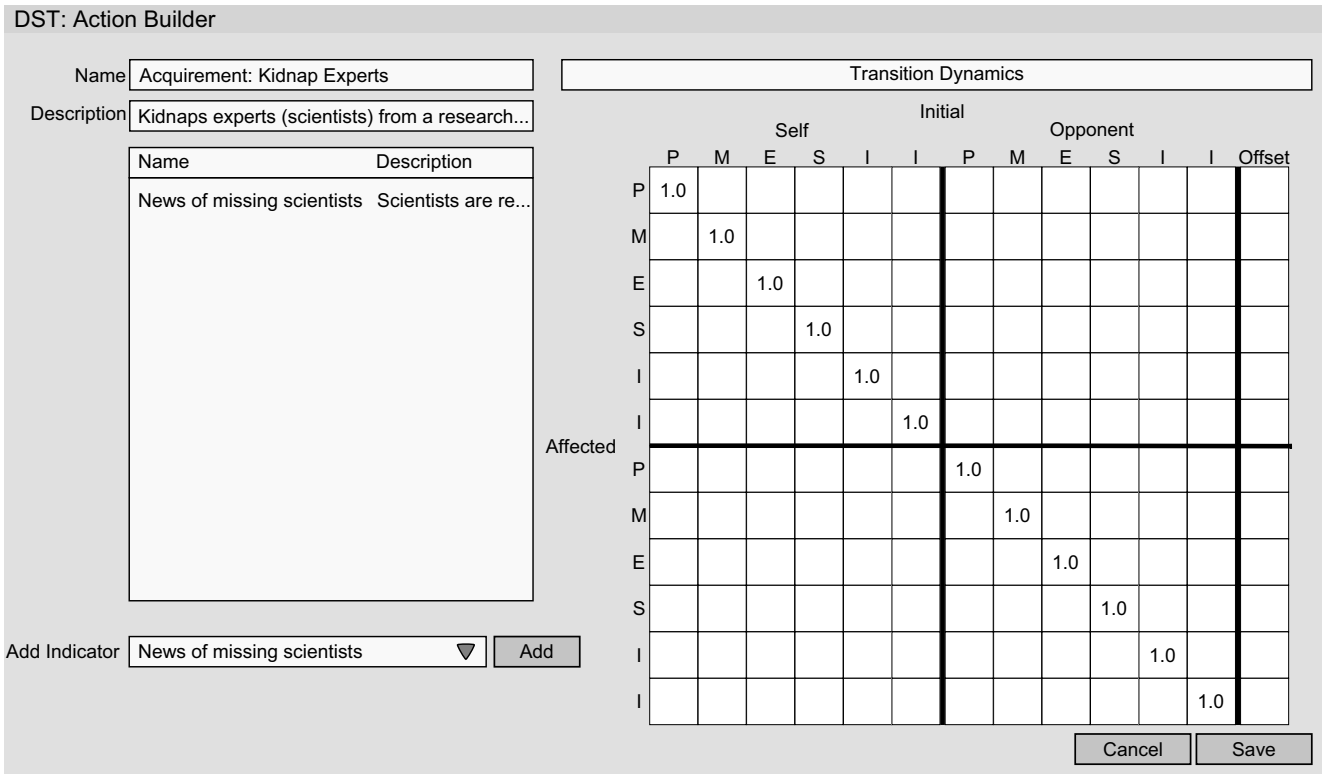


Figure 3. Constructing an individual action

player may perform them). Mathematically, a rule is a seven-tuple:

- set of triggering players,
- set of triggering actions,
- add/remove operator,
- set of add/remove actions,
- set of affected-players,
- rule's life span in number of turns and
- a boolean describing if the rule is an initially active.

If a rule is initially active then the set of triggering players and triggering actions should be empty. The tool for constructing rules has more options (like a replacement operator for compactness), but those rules can be reduced to the above seven-tuple. The rule builder is shown in Fig. 4. The rule set is a set of rules (see Fig. 5).

The rule set together with the action set describe the game tree's structure. At each step as the game tree is being built, the rule set is consulted to determine which rules were triggered by the current move. Additionally, a stack is kept to determine for how many more turns previously triggered rules will be valid. Thus the rules combined with the actions effectively defines the possible strategies.

5) Players: The players are the decision makers. Formally each player is described by a five-tuple: a vector of initial resources (which resources are defined by the cost/benefit model), a payoff matrix, a risk aversion, and a set of time-varying strategies (as defined by the rule set and action set). The payoff matrix is a 2×6 matrix describing how much the player values each component of the state-space (i.e., how much they value their resources and their opponent's resources). The risk aversion is a real number describing the α in (1). Recall that the more positive α the more risk-adverse the player. There is one more additional component for one of the players: assumptions (prior probabilities). The assumptions are simply the prior probabilities $p_{k_a}^{d+2m-1}$ of the defender.

6) Look ahead: The last step before analyzing the game is to determine the number of moves the players look ahead, τ. It is difficult to determine a good τ when there are complex rule interactions changing the number of possible actions at each stage of the game. For simple or no rules, there is a method which determines the maximum number of moves (such that memory is not exhausted) to look ahead for constant fan-out. Thus, the user can provide a positive integer to manually specify the τ otherwise the method will attempt to automatically estimate the maximum τ.

B. Results

To analyze a game, the user clicks the "run" button in Fig. 1. The optimal moves (for the first player) and optimal moves under assumptions (for the second player) are expanded in stage of five moves. Then the system prompts

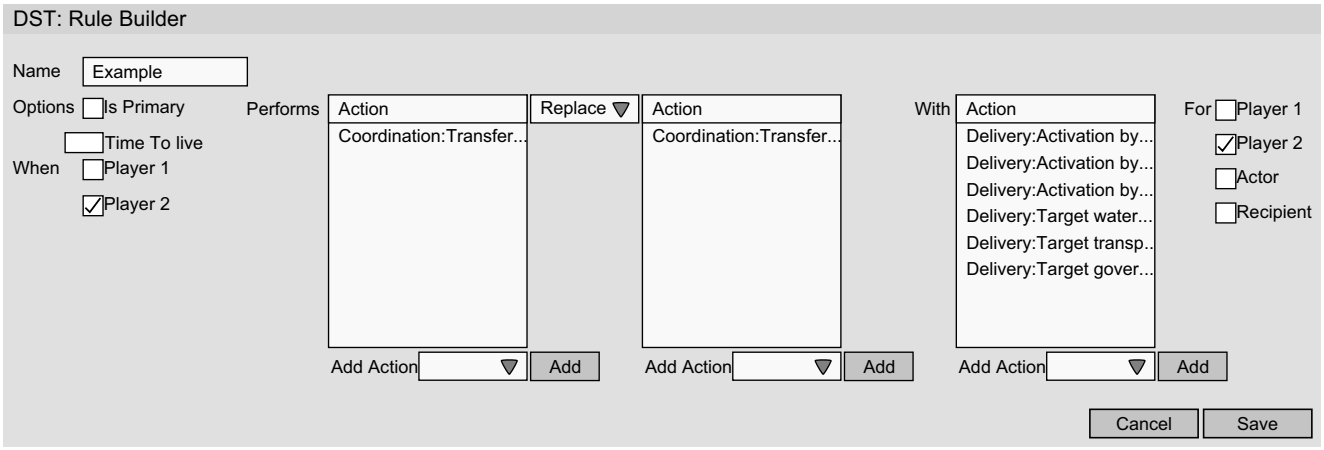


Figure 4. Constructing an individual rule

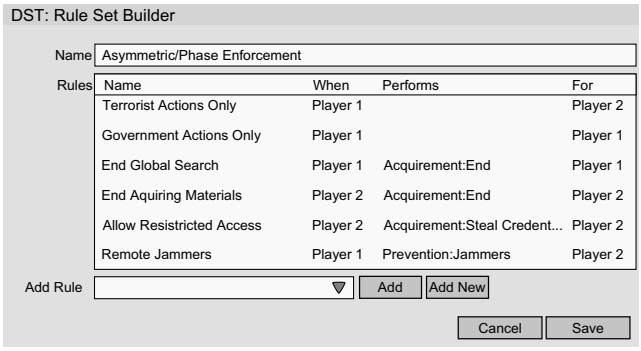


Figure 5. Constructing a rule set

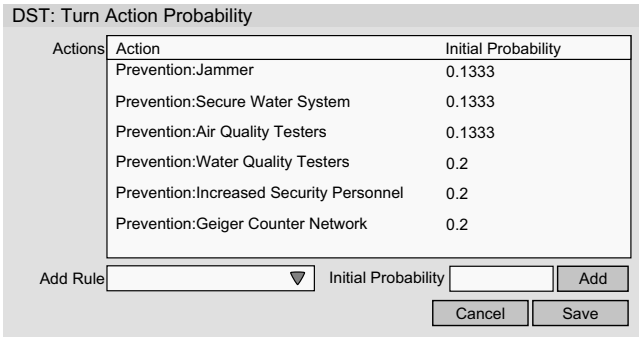


Figure 6. Specifying Priors

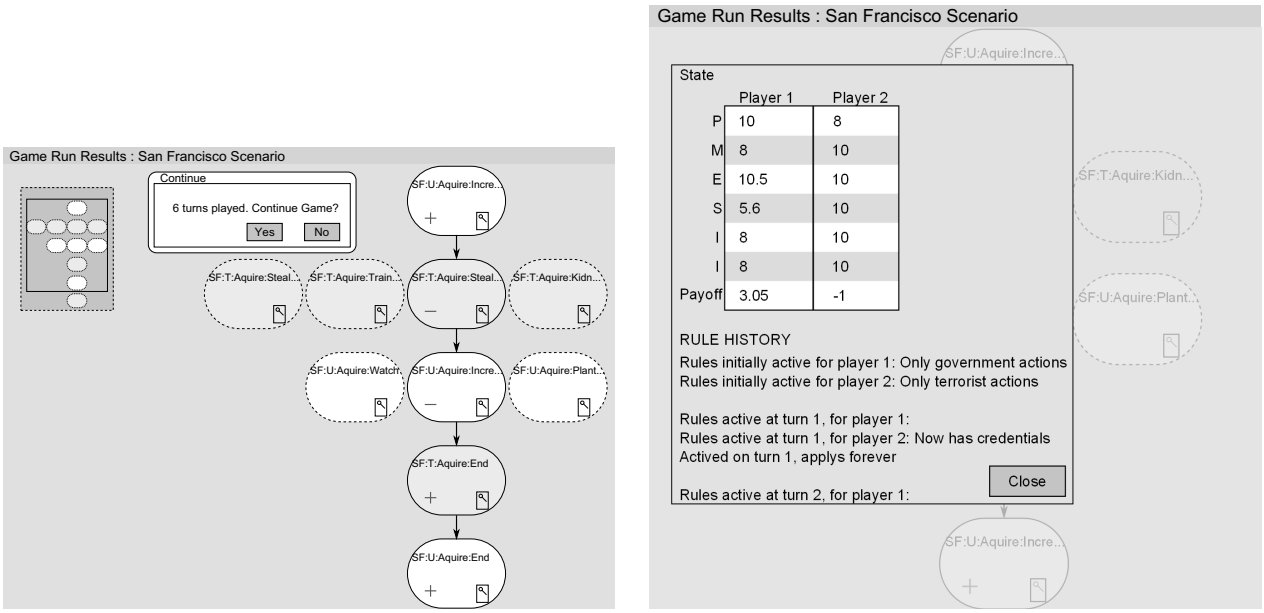
the user if the game should continue. This is repeated until the user is satisfied with the number of turns played. The result is similar to that in Fig. 7a. After the game has been expanded enough turns, the user is free to look at alternative actions by clicking the "+" button at any stage. This will reveal the other possible permitted actions. Any node can be inspected to see the state-space of the game as well as any active rules by clicking the document icon (see Fig. 7b).

Together these tools allow for a user to refine and validate the outcomes as well as explore alternative options. If the results are unexpected, the user can trace the issue, then refine the model. This allows for iterative model building, which encourages refining/expanding the action and rule sets. Furthermore the action and rule sets can be reused on different players with different initial conditions. Thus users can have confidence in their games by taking advantage of this human-in-the-loop feedback.

The resulting path through the game can be used as input into a tool that makes queries that ATRAP can then use to search a database for empirical support. These queries can be chained together in what is called a template, allowing for the result of one query be an input to another query. Instantiations (queries) of each player's moves (abstractions) go into different templates to permit querying for empirical support for either player.

V. Conclusion

We have reviewed game theory, deterministic equivalents, and how the two can be combined to produce a decision support system. The decision support system allows for the modeling of players who consider risk when making decisions. The resulting methodology's relation to the traditional minimax algorithm was explored. We analyzed, extended and improved on prior work, which included providing superior numerical stability and resolving an issue associated with negative deterministic equivalents. The work was extended to allow for non-zero sum games and the introduction of a state-space allows for more complicated scenarios to be simulated. Furthermore we built a framework to aid in the creation, refinement, and inspection of probabilistic multistage games. A brief walk-through of the tool demonstrated the specification of action sets, rule sets, player initial conditions and player risk aversion. It was shown how an opponent's assumptions could be taken into account. The



(a) Exploration of results

(b) Introspection of result

Figure 7. Exploration and Introspection of results

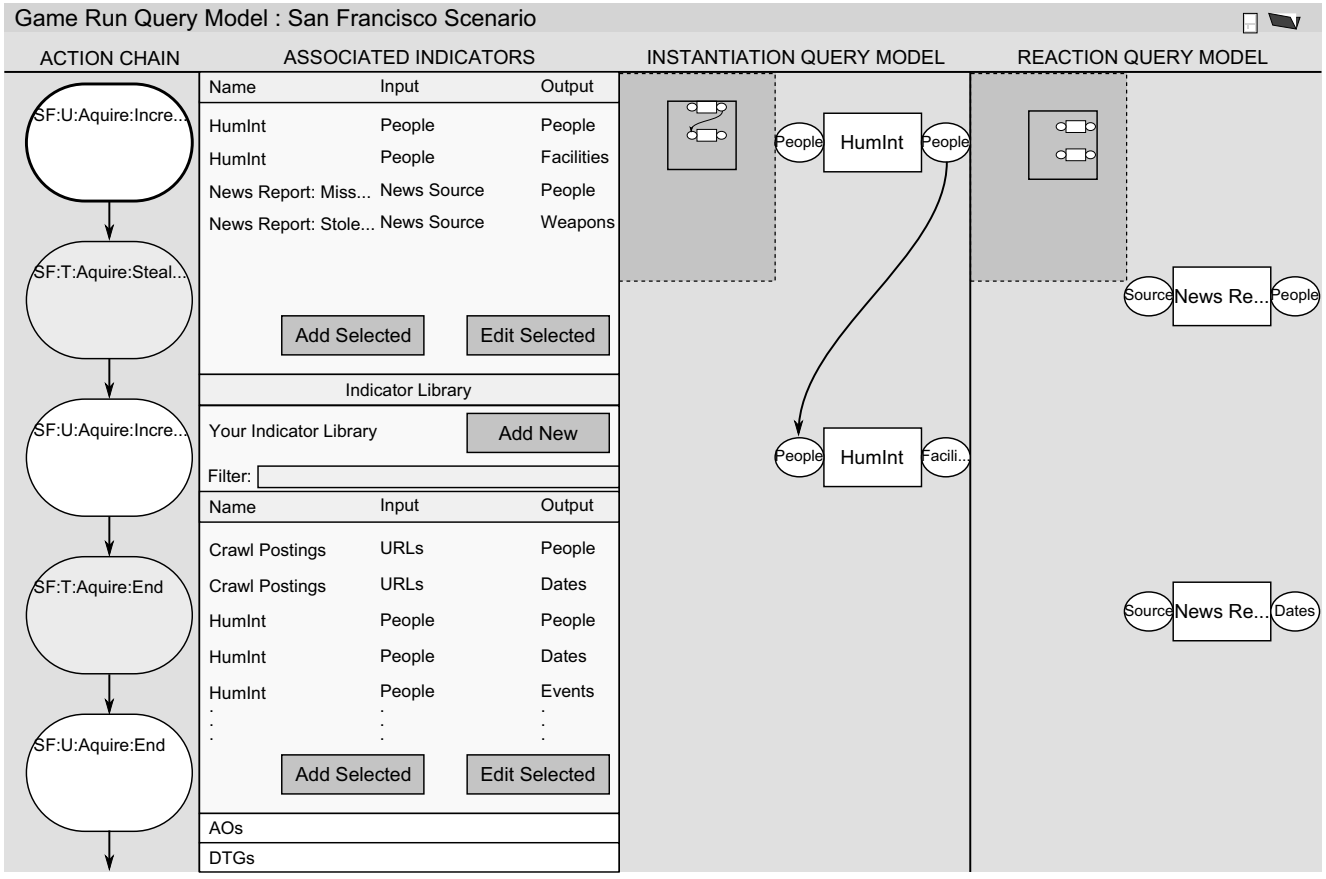


Figure 8. Conversion of results

results of the analysis were fed into another tool to allow for the exploration of alternatives and introspection of the model. The results were then translated into queries that could be used to search a database.

There are several possible extension for future work. The most interesting games need more than two players. Arbitrary dimensional models would allow for even more advanced simulation of situations. To further aid in robust decision making, multiple paths through the game tree should be returned. Assumptions could be refined through self-play to represent the evolution of the game as the other players begin to adapt to an optimal strategy. A more advanced rule system, such a production system, would allow for certain actions to be disabled or enabled depending on the current resources held by the players. Alternative decision-making models, such as bounded-rationality, for the opponents could be integrated to reflect more realistic opponents.

ACKNOWLEDGMENT

We would like to thank Dr. Ferenc Szidarovszky of the SIE department of the University of Arizona for access to his algorithm, and Ft Huachuca, Battlefied Element for funding this work. Ephibian, Inc. worked on the user interface and integration of this system with ATRAP. ATRAP is based upon work supported by the G9 Prototype under Task Order Numbers 9T7ZDAIS705, 9T8ZDAIS803, 9Q9SDAIS903, and 9Q0SDAIS003. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s), and do not necessarily reflect the views of the G9 Prototype or the US Army Intelligence Center of Excellence.

REFERENCES

[1] P. Keen and M. Morton, *Decision support systems: an organizational perspective*. Addison-Wesley Pub. Co., 1978.

[2] M. Johnston, K. Langton, R. Haynes, and A. Mathieu, "Effects of computer-based clinical decision support systems on clinician performance and patient outcome: a critical appraisal of research," *Annals of internal medicine*, vol. 120, no. 2, p. 135, 1994.

[3] D. Hunt, R. Haynes, S. Hanna, and K. Smith, "Effects of computer-based clinical decision support systems on physician performance and patient outcomes," *JAMA: the journal of the American Medical Association*, vol. 280, no. 15, p. 1339, 1998.

[4] M. Freer, A. Moore, and J. Donnelly, "Grazplan: Decision support systems for australian grazing enterprises–ii. the animal biology model for feed intake, production and reproduction and the grazfeed dss," *Agricultural Systems*, vol. 54, no. 1, pp. 77–126, 1997.

[5] L. Suantak, D. Hillis, J. Schlabach, J. W. Rozenblit, and M. Barnes, "A coevolutionary approach to course of action generation and visualization in multi-sided conflicts," in *Systems, Man and Cybernetics, IEEE International Conference on*, vol. 2, October 2003, pp. 1973–1978 vol.2.

[6] L. Suantak, F. Momen, J. W. Rozenblit, D. Hillis, M. Barnes, and J. Schlabach, "Modeling and simulation of stability and support operations (sasо)," in *Engineering of Computer-Based Systems, 2004. Proceedings. 11th IEEE International Conference and Workshop on the*. Los Alamitos, CA, USA: IEEE Computer Society, 2004, pp. 21–28.

[7] Y. Luo, F. Szidarovszky, Y. Al-Nashif, and S. Hariri, "A game theory based risk and impact analysis method for intrusion defense systems," in *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*. IEEE, 2009, pp. 975–982.

[8] Y. Luo, Y. Al-Nashif, F. Szidarovszky, and S. Hariri, "Game tree based partially observable stochastic game model for intrusion defense systems (ids)," in *IIE Annual Conference and EXPO (IERC)*, 2009.

[9] Y. Luo, F. Szidarovszky, Y. Al-Nashif, and S. Hariri, "Game theory based network security," *Journal of Information Security*, vol. 1, pp. 41–44, 2010.

[10] M. Valenzuela, C. Feng, P. Reddy, F. Momen, J. Rozenblit, B. ten Eyck, and F. Szidarovszky, "A non-numerical predictive model for asymmetric analysis," in *Engineering of Computer Based Systems (ECBS), 2010 17th IEEE International Conference and Workshops on*, march 2010, pp. 311 –315.

[11] N. Papadakakis, E. Markatos, and A. Papathanasiou, "Palantir: A visualization tool for the world wide web," in *Proceedings INET*, vol. 98, 1998.

[12] C. Camerer, "Does strategy research need game theory," *Strategic Management Journal*, vol. 12, no. S2, pp. 137–152, 1991.

[13] P. Samuelson, "The fundamental approximation theorem of portfolio analysis in terms of means, variances and higher moments," *The Review of Economic Studies*, vol. 37, no. 4, pp. 537–542, 1970.

[14] Y. Luo, F. Szidarovszky, Y. Al-Nashif, and S. Hariri, "A novel fictitious play approach for multi-stage intrusion defense systems," *International Journal of Information Security*, 2011, submitted, not yet published.

[15] D. West, "Updating mean and variance estimates: An improved method," *Communications of the ACM*, vol. 22, no. 9, pp. 532–535, 1979.

[16] R. Sutton and A. Barto, *Reinforcement learning: An introduction*. Cambridge Univ Press, 1998, vol. 28.

[17] U. S. J. C. Forces, "Pmesii tool index — dm2research," 2011, [Online; accessed 20-November-2011]. [Online]. Available: \url{http://pmesii.dm2research.com/wiki/index.php/Main_Page}

[18] P. Mansoor, "Linking doctrine to action: A new coin center-of-gravity analysis," DTIC Document, Tech. Rep., 2007.