1 Introduction

Argument dialogues provide a principled way of structuring rational interactions between participants (be they human or machine), each arguing over the validity of certain claims, with each agent aiming for an outcome that achieves their dialogue goal (e.g., to persuade the other participant to accept their point of view [8], or to reach agreement on an action to perform [2]). Achievement of an agent’s dialogue goal typically depends on both the arguments that the agent chooses to make during the dialogue, determined by its strategy, and the arguments asserted by its interlocutor. The strategising agent—the proponent—thus has the difficult problem of having to consider not only which arguments to assert but also the possible responses of its opponent. This problem is compounded since the opponent may exploit knowledge inferred from those arguments asserted by the proponent to construct new arguments. Hence, the proponent must take care not to divulge information that is advantageous to its opponent.

The important challenge of how to generate strategies for such a proponent has not been widely explored [10]. Notable exceptions are the work of Hadoux et al. [7], which employs mixed observability Markov decision processes to generate optimal policies for the proponent to follow; the work of Rienstra et al. [9], which applies a variant of the minimax algorithm to determine an effective proponent strategy; and the work of Black et al. [3], which employs heuristic planning techniques to determine an optimal proponent strategy for a simple asymmetric persuasion setting.

We highlight two types of uncertainty in the strategic argumentation problem: uncertainty over the arguments initially known to the opponent, captured by the opponent model, and uncertainty over how the opponent chooses to argue, given their initial knowledge base. Both [9] and [3] deal with uncertain models of the opponents initial knowledge base, where the opponent’s strategy is known (i.e., optimal [9] or deterministic [3]); while [7] considers the case in which we have only a stochastic model of how the opponent will behave.

The key novelties in our approach are that we deal with both of these types of uncertainty simultaneously the former through use of an uncertain opponent model and the latter by generating conformant strategies, that is strategies that are effective regardless of the opponent strategy. Further, our work is the first to generate strategies in a setting where the opponent may exploit information obtained during the dialogue to construct arguments unknown to it at the start of the dialogue, necessitating more cautious strategies.

2 Strategic Argumentation Problem

We consider a strategic argumentation setting in which both agents exchange arguments, with the proponent aiming to convince its opponent of some topic argument. Our problem comprises the following:

- An argumentation framework \((A, \rightarrow)\), comprising a set of arguments \(A\) and a binary attack relation \(\rightarrow \subseteq (A \times A)\),
- The function \(\text{Acc}: 2^A \rightarrow \emptyset\) describes the set of acceptable arguments \(\text{Acc}(B)\), for each subset \(B \subseteq A\), subject to some agreed semantics (see [5], for details).
- A proponent model \(K_A \subseteq A\), comprising a set of arguments available to the proponent.
- An uncertain opponent model \((E, K_O, p)\), comprising a finite set of labels \(E = \{O_1, \ldots, O_m\}\), a function \(K_O : E \rightarrow 2^A\) associating each label \(O_i \in E\) with a possible opponent model \(K_O \subseteq A\), and a probability distribution \(p: E \rightarrow \Omega\), assigning to each \(O_i \in E\) the likelihood that the opponent initially knows \(K_O\).
- A closure operator \(\mu: 2^A \rightarrow 2^A\) provides a description of how each agent may derive new arguments from knowledge acquired during the dialogue. i.e., \(\mu(B) \subseteq A\) is the set of all arguments that can be derived by \(B \subseteq A\).

A dialogue is a sequence of moves \(D = [M_0, M_1, \ldots, M_n]\) in which each move \(M_k \subseteq A\) is a finite set of arguments. A dialogue terminates when \(M_{k+1} \cup M_k = \emptyset\), and is successful for the proponent w.r.t. a given topic \(t \in A\), if \(t \in \text{Acc}(M_0 \cup \cdots \cup M_n)\).

A (general) strategy for \(A_g \in \{P, O\}\) is a function \(\sigma_g: 2^A \rightarrow 2^A\) such that \(\sigma(B) \subseteq \mu(K_g \cup B)\), for all \(B \subseteq A\), that determines which move \(A_g\) should make, given the arguments asserted thus far. A simple strategy is a special case of strategy specified by a sequence of moves \(S = [S_0, \ldots, S_n]\), with \(S_k \subseteq K_P\), representing sets of arguments to be asserted in turn by the proponent, unless the opponent accepts the topic, or the sequence is exhausted, in which case we assert \(\emptyset\). A pair of strategies \((\sigma_P, \sigma_O)\) generates the dialogue \(D_{(\sigma_P, \sigma_O)} = [M_0, \ldots, M_n]\) given by \(M_k = \sigma_P(M_0 \cup \cdots \cup M_{k-1})\), where \(A_g = P\) whenever \(k\) is even and \(A_g = O\) whenever \(k\) is odd.

A strategy \(\sigma_P\) is effective against \(O_i \in E\) if \(D_{(\sigma_P, \sigma_O)}\) is successful, for every possible opponent strategy \(\sigma_O\) for \(O_i\).

3 Persuasion Dialogues as Planning Problems

We provide a translation of a strategic argumentation problem into a propositional planning problem with (bounded) numerical variables, such that a solution to the planning problem yields an effective strategy for our proponent. Encoding the problem in the standard planning domain language PDDL2.1 [6] allows us to use an implementation of the planner Poppy [4] to generate appropriate simple strategies. A key challenge in using a propositional planner to solve these problems is capturing the uncertainty about the opponent’s initial beliefs. We use techniques inspired by the current state-of-the-art approach for solving conformant planning problems by compilation to classical planning [1]. The planning problem \(P\) is described as follows:

Variables We require the following numerical variables: (stage), (probSuccess) and (prob()], for each \(O_i \in E\). The variable (stage) regulates the order in which actions may be added to the plan, while
the variables (probSuccess) and (prob(i)) are responsible for calculating the probability that a given strategy will be successful. In addition to these, we have the following propositional variables: canAssertP(a) and canAssertO(M, i, D) govern the arguments that \( \mathcal{P} \) and \( \mathcal{O} \) can assert, respectively, for a \( a \in \mathcal{K}_\mathcal{P} \) and \( \mathcal{O}_i \in \mathcal{E} \) and \( D \subseteq \mathcal{K}_\mathcal{P}; \) dialogueP(D) and dialogueO(i, D) describe the set of arguments that have been asserted by \( \mathcal{P} \) and \( \mathcal{O}_i \), respectively; temp(i, D) a temporary `storage' variable for dialogueO(i, D); successful(D, D') if \( (D \cup D') \) is an acceptable set of arguments, where \( D_A \) are the arguments asserted by \( A \in \{\mathcal{P}, \mathcal{O}\} \); effective(i) says that the strategy is effective against \( \mathcal{O}_i \in \mathcal{E} \); addP(a, D, D') says that \( D' = D \cup \{a\} \), and addO(M, D, D') says that \( D' = D \cup M \).

**Initial Conditions** Our set of initial conditions comprises the following numerical conditions: \( \text{stage} = 0 \), \( \text{probSuccess} = 0 \), and \( \text{prob}(i) = \text{prob}(\text{O}(i)) \), for all \( \mathcal{O}_i \in \mathcal{E}. \) We also require the following propositional initial conditions: canAssertP(a), for all \( a \in \mathcal{K}_\mathcal{P}, \text{dialogueP}(\emptyset), \) and dialogueO(i, \( \emptyset \)), for all \( \mathcal{O}_i \in \mathcal{E}, \) together with the following conditions that, once set, remain unaltered by the effects of any of the actions: canAssertO(M, i, D) iff \( M \subseteq \mu(\mathcal{K}_\mathcal{O} \cup D_P) \); successful(D, D') iff \( t \in \text{Acc}(D_P \cup D_O) \); addP(a, D, D') iff \( D' = D \cup \{a\} \); addO(M, D, D') iff \( D' = D \cup M \).

**Goal Condition** Our goal is a single condition \( \text{probSuccess} > \lambda \), where \( \lambda \in [0, 1] \) is the required probability of success.

**Actions** Our set of actions comprises three types of action whose preconditions and effects are described in Figures 1–2, where all free parameters are universally quantified. For each argument \( a \in \mathcal{K}_\mathcal{P} \) known to the proponent, there is a \( \text{prop}(\text{proponent}(a)) \) action, which emulates the act of the proponent asserting \( a \). A single move is built up by iterated application of these actions (simulating the assertion of a set of arguments by the proponent). The opponent’s move is captured by a single (opponent) action. This action assigns all possible responses for each possible opponent model \( \mathcal{O}_i \in \mathcal{E} \), adding them to a ‘pool’ of possible dialogue states associated with that opponent model. Finally the action (probCount) must be applied after each (opponent) action and sums the total probability of guaranteed success, against each of the possible opponent models \( \mathcal{O}_i \in \mathcal{E} \).

A solution to the planning problem \( \mathcal{P} \), generated by a planner, is a sequence of actions that transforms the initial state into a state that satisfies the goal. Such a solution corresponds to a simple strategy that will be successful with probability \( \lambda > 0 \), and then iteratively seek solutions with strictly larger \( \lambda \) values.

**4 Results**

Owing to the lack of existing approaches for proponent strategy generation that do not depend on knowledge of the opponent’s strategy, we instead benchmarked our approach against a naive depth-first search, that exhaustively searches all simple strategies and evaluates their effectiveness with respect to the given opponent model. Our analysis showed that for smaller examples the performance two approaches in comparable; however, for larger examples, comprising more than nine arguments, the planning approach outperforms the naive search by several orders of magnitude.

We performed further experiments to compare to the most closely related approach by Hadoux et al. [7] on the problems described in their paper. The two approaches are not directly comparable: we do not rely on knowledge of the opponent’s strategy, whilst they do; they generate policies, dependent on the opponent’s response, whereas we generate simple strategies. We did however make two observations: first our approach finds solutions an order of magnitude faster than theirs; and second in all of the problems they examine, we were able to observe by inspection of the problems that it is not possible to outperform the optimal simple strategies generated by our approach, meaning that our approach is equally strong in those particular settings. Our problems, the planner we used, and the implementation of our translation and of our naive algorithm are all available from: http://tinyurl.com/jfxotsg.

**REFERENCES**