Deriving Persuasion Strategies Using Search-Based Model Engineering

Josh MURPHY [1], Alexandru BURDUSEL, Michael LUCK, Steffen ZSCHALER, and Elizabeth BLACK
Department of Informatics, King's College London

Abstract. We consider a one-to-many persuasion setting, where a persuader presents arguments to a multi-party audience, aiming to convince them of some particular goal argument. The individual audience members each have differing personal knowledge, which they use, together with the arguments presented by the persuader, to determine whether they are convinced of the goal. The persuader must, therefore, carefully consider its strategy, i.e., which arguments to assert, in order to maximise the number of convinced audience members. Here, we use evolutionary search to find (near-)optimal strategies for the persuader. We implement our approach using search-based model engineering, which provides a natural and efficient encoding for such problems. We investigate the performance of our approach on a range of settings, considering different structures and sizes of argumentation frameworks (representing the underlying knowledge available to the persuader and audience members), and varying the size of audience and of the audience members' personal knowledge bases. We show that we can find effective strategies for problems with more than 200 arguments and more than 100 audience members. Further, we show that the approach supports multiple persuader objectives, finding persuader strategies that aim to minimise arguments to assert while still maximising the number of convinced audience members.

Keywords. argumentation, persuasion, strategy, search-based model engineering

1. Introduction

Persuasion is the task of inducing the acceptance of a belief in other agents. A political speech is an example of persuasion, in which the politician attempts to persuade the public that their party is the one to vote for at the next election. This paper focuses on such a one-to-many persuasion setting, where a single persuader broadcasts arguments to a multi-party audience with the aim of convincing them of some goal argument. Since each individual audience member reasons with its own set of personal knowledge (which we assume is known to the persuader) any particular set of persuader arguments may be convincing to some audience members but not others, and so the persuader must carefully select which arguments they should assert in order to maximise the number of audience members they convince. This is a challenging problem because of the number of potential solutions and the number of audience members to evaluate against: to exhaustively

[1]Corresponding author: Josh Murphy, Department of Informatics, King's College London, 30 Aldwych, London, WC2B 4BG, UK; E-mail: josh.murphy@kcl.ac.uk

explore the solution space, for each subset of the persuader's arguments one must consider each audience member and determine whether they would be convinced by those arguments. The number of possible solutions is exponential in the number of arguments known by the persuader, so an exhaustive search for the optimal set is not practical.

Much of the recent work looking at strategic argumentation settings focuses on one-to-one persuasion, e.g., [3,8,21,9,12,19]. Notably, Hunter and Thimm [14] also consider how to determine which set of arguments to present to an audience, using probabilistic argumentation to capture uncertainty about the audience members' beliefs. However, in contrast to our approach, they do not allow for a range of audience members each with different beliefs. Furthermore, their approach has been applied to settings with up to 7 arguments, while our approach scales to more than 200 arguments. In earlier work [10,11], Hunter looks at how one can select arguments to resonate with a particular audience, but this similarly assumes a typical audience member, while our approach allows representation of distinct audience members. Bench-Capon *et al.* present a framework that can be used to describe audiences comprised of members with different values [2], but do not address the strategic considerations of the persuader in such a domain.

To efficiently determine the arguments the persuader should assert, we use evolutionary search to find a near-optimal strategy for the persuader, that maximises the number of convinced audience members. We use techniques from search-based model engineering (SBME) [4] to encode the problem. We ran experiments over a range of settings, varying both the size of the problem and the structure of argumentation framework representing the underlying knowledge available to the persuader and audience members, and show that our approach:

C1 produces strategies that are effective in convincing members of the audience;
C2 finds strategies efficiently, in that it scales well with increasing numbers of arguments in the domain, and increasing numbers of audience members; and
C3 can efficiently find strategies that satisfy multiple objectives (in particular, maximising convinced audience members while minimising arguments asserted).

To the best of our knowledge, this work is the first to apply evolutionary search to strategic argumentation. McBurney and Parsons [16] propose an application of an evolutionary algorithm to automate a chance discovery dialogue, where individuals exchange knowledge with the aim of discovering unknown risks and opportunities, but, while they outline their proposed approach, it has not been specified in detail. While the focus here is on a one-to-many persuasion setting, where a persuader uses its knowledge of the audience members to select a set of arguments to assert, the approach we present is sufficiently flexible to capture a range of argument dialogue settings, and we discuss in Section 6 our plans to extend this work to account for uncertainty in the persuader's knowledge of its audience and to allow dialogues in which each party may present arguments.

This paper is set out as follows. In Section 2 we formally define multi-audience persuasion and introduce search-based model engineering (SBME) in Section 3. Section 4 explains how we use techniques from SBME to search for persuader strategies. We evaluate our approach in Section 5 and finish with a discussion in Section 6.

2. Multi-Audience Persuasion Games (MAPGs)

We can represent arguments as abstract entities in an argumentation framework (AF) [6]. AFs are comprised of a set of arguments and the attacks between them.

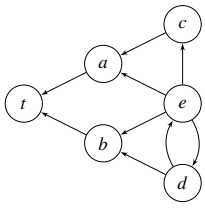

Figure 1. An argumentation framework.

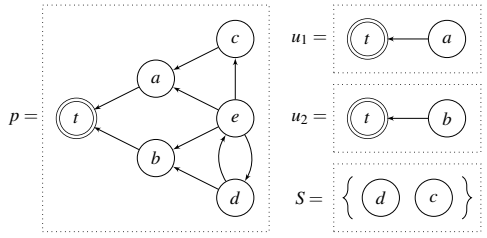

Figure 2. A multi-audience persuasion game, with persuader p, persuadees u_1, u_2, and strategy S.

Definition 1. *An **argumentation framework** is a pair $\langle A, R \rangle$ s.t. A is a finite set of arguments and $R \subseteq A^2$ is a set of attacks. $(a,b) \in R$ denotes a attacks b.*

Given an AF, we can ask which arguments are *justifiable*. Various argumentation semantics have been defined that seek to capture different notions of what it means for an argument to be justified. Our approach does not assume any particular semantics but for our evaluation we use the *preferred credulous* semantics, which is well-suited to practical reasoning about what to do [18]. An argument is justified under the preferred credulous semantics if it is part of a maximal (under set inclusion) subset of arguments that is conflict-free and that defends all of its members (i.e., if an argument attacks some member of the set, there is some member of the set that attacks that argument).

Definition 2. *Let $\langle A, R \rangle$ be an AF and $S \subseteq A$.*

- *S is* **conflict-free** *iff $\forall a, b \in S$: $(a,b) \notin R$.*
- *$a \in A$ is* **acceptable** *w.r.t. S iff $\forall b$ s.t. $(b,a) \in R$: $\exists c \in S$ s.t. $(c,b) \in R$.*
- *S is* **admissible** *iff S is conflict-free and each argument in S is acceptable w.r.t. S.*
- *S is* **maximally admissible** *iff $\nexists e \in (A - S) : S \cup \{e\}$ is admissible.*

*We denote the **justified arguments under the preferred credulous semantics** as $\sigma(AF) = \{a \mid \exists S \subseteq A$ s.t. S is maximally admissible and $a \in S\}$*

We consider a *multi-audience persuasion game* (MAPG), in which a persuader seeks to convince a set of persuadees, known as the audience, that a particular topic argument is justified. The persuader's knowledge is represented by an AF, from which each persuadee's knowledge is an AF induced from a subset of arguments in the persuader's AF. The audience captures each persuadee's knowledge, thus we assume that the persuader has exact knowledge of the audience members; we discuss in Section 6 how our approach can be adapted to allow for uncertain knowledge of the persuadees. The persuader's strategy is a subset of the persuader's AF, which are the arguments the persuader will assert to the audience. We assume *w.l.o.g.* that persuadees each know the topic argument before the persuader presents their arguments.

Definition 3. *A **multi-audience persuasion game** is a tuple $g = \langle p, t, U, S \rangle$, such that:*

- *$p = \langle A_p, R_p \rangle$ is the argumentation framework belonging to the **persuader**.*
- *$t \in A_p$ is the **topic**, the argument the persuader tries to convince the audiences of.*
- *$U = \{u_1, ..., u_n\}$ is the **audience**, where $u_i = \langle A_i, R_i \rangle$ is the argumentation framework belonging to persuadee i, s.t. $A_i \subseteq A_p$, $R_i \subseteq R_p$, and $t \in A_i$.*
- *$S \subseteq A_p$ is the persuader's **strategy**.*

An example MAPG is shown in Figure 2. Note, the persuader's strategy is asserted to all persuadees at once; the persuader cannot choose to assert an argument to only a subset of persuadees. A persuadee is convinced if, under their framework combined with the strategy, the topic is justified under the preferred credulous semantics. Note that, in an MAPG, the topic does not have to be justified *w.r.t.* the persuader's AF, who may wish to persuade the audience of an argument they do not themselves find justified.

Definition 4. *In a multi-audience persuasion game $g = \langle p, U, t, S \rangle$ with the persuader's framework $p = \langle A_p, R_p \rangle$, a persuadee i with AF $\langle A_i, R_i \rangle \in U$ is **initially convinced** in g iff $t \in \sigma(\langle A_i, R_i \rangle)$. The function $\gamma(g, u_i) \rightarrow [0,1]$ returns 1 iff u_i is initially convinced in g, 0 otherwise. Similarly, i is **convinced** in g iff $t \in \sigma(\langle A_i \cup S, R_p \cap (R_i \cup (A_i \cup S)^2) \rangle)$. The function $\hat{\gamma}(g, u_i) \rightarrow [0,1]$ returns 1 iff u_i is convinced in g, 0 otherwise.*

A persuader is typically interested in convincing as many persuadees as they can. We measure *effectiveness* of a strategy as the increase in the number of convinced persuadees from those that are initially convinced. By asserting arguments, the persuader may dissuade audience members of the topic; a persuader that dissaudes more audience members than they persuade will have a negative effectiveness. As well as trying to convince as many persuadees as possible, the persuader may also wish to minimise some *cost* associated with asserting a strategy. In this paper, we assume the cost of a strategy is the proportion of the persuader's arguments put forward in the strategy. The persuader wants to minimise the number of arguments they present, since more arguments may lead to audience disengagement [13]. We refer to this cost as the *efficiency* of a strategy.

Definition 5. *The **effectiveness** of the strategy in a multi-audience persuasion game $g = \langle p, U, t, S \rangle$, denoted $\varepsilon(g)$, is: $\sum_{u \in U} \hat{\gamma}(g, u) - \sum_{u \in U} \gamma(g, u)$.*

Definition 6. *The **efficiency** of the strategy in a multi-audience persuasion game $g = \langle p, U, t, S \rangle$ with persuader's framework $p = \langle A_p, R_p \rangle$, denoted $\kappa(g)$, is: $\frac{|A_p| - |S|}{|A_p|}$.*

Example 1. *Consider the example multi-audience persuasion game in Figure 2. The* effectiveness *of the strategy S is 2, as both persuadees will find the topic acceptable once the arguments in S are added to their respective frameworks. The* efficiency *of the strategy S is $\frac{6-2}{6} = \frac{4}{6}$ as two argument are asserted in the strategy. Note that had the persuader chosen strategy $\{e\}$ instead, then the effectiveness would remain the same but the efficiency would be improved to $\frac{6-1}{6} = \frac{5}{6}$.*

We use evolutionary search to find an effective and efficient strategy of a multi-audience persuasion game. We implement the problem using SBME, which provides a natural and efficient encoding.

3. Search-Based Model Engineering (SBME)

Search-based methods have long been used to solve optimisation problems [7]. Here, we give an overview of search-based methods, before examining SBME in more detail.

Meta-heuristic search. Many optimisation problems can be solved by dedicated algorithms or using specialised heuristics. However, as problems become more complex, it often becomes more efficient to find (near-)optimal solutions using meta-heuristic search techniques. These techniques start from one (or a population of) randomly generated *feasible candidate solutions* (i.e., solutions that satisfy all relevant constraints) and incrementally change these to explore the solution search space. The quality of any candidate solution is indicated by one or more *objective functions*—functions that take a solution and provide a numeric value indicating relative quality. A meta-heuristic algorithm then evolves the population of candidate solutions by:

1. creating a set of new candidate solutions derived from the existing solutions;
2. ranking old and new candidate solutions according to their objective values; and
3. keeping only the highest-ranked n candidate solutions for the next round.

The algorithm ends either when a pre-defined number of evolutions have been explored or when another stopping criterion has been reached (e.g., when the objective values of candidate solutions no longer change significantly).

Different meta-heuristic algorithms use different techniques for encoding solutions and deriving new ones, as well as for ranking solutions. Here, we focus on evolutionary search techniques, which derive a new candidate solution from each existing candidate solution by applying a *mutation operator* randomly picked from a pre-defined set.

Search-based model engineering. SBME [24,15] aims to apply meta-heuristic search techniques in the context of model-driven engineering (MDE). Specifically, SBME techniques search for models that are optimal as defined by some objective functions.

To understand SBME, we first need to briefly introduce key notions of MDE, such as model, meta-model, and model transformation. MDE's central tenet is that software should be developed using high-level models, expressed in domain-specific modelling languages, rather than by directly writing programs in general-purpose modelling languages such as Java or C. Key to this is the ability to define modelling languages and automatically and efficiently manipulate models expressed in these languages. Meta-models support this by providing a formalised representation of a modelling-language's abstract syntax; that is, the concepts of the language and their interactions. Typically in MDE, meta-models are expressed as class diagrams. Models are considered valid iff they are an instance of the meta-model; that is, if every model element is an instance of a corresponding meta-model element and all connections between model elements are specified according to the associations defined in the meta-model. Model transformations, finally, are programs that take models as input and produce new models as outputs (possibly instances of different meta-models).

By employing SBME techniques for specifying optimisation problems we benefit from three main advantages: 1. we can use the concept of model transformations to simplify the definition of complex search operators that can ensure consistency of the generated offspring; 2. the use of models allows us to use the user's domain expertise to consistently encode complex problems and solutions and, we can ensure that the search space exploration is done without generating inconsistent solutions; 3. this approach does not require the step of genotype to phenotype mapping that would otherwise be required in traditional genetic programming approaches.

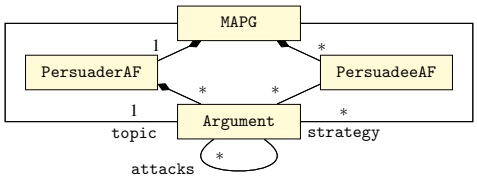

Figure 3. Metamodel for multi-audience persuasion games, represented as a class diagram

4. Multi-Audience Persuasion as a Search-Based Model Engineering Problem

To represent a multi-audience persuasion game (MAPG) as a SBME problem, we must first define a metamodel that encodes the space of possible solutions. This is shown in Figure 3 and we explain now how this corresponds to our MAPGs (Definition 3). The persuader's AF ($\langle A_p, R_P \rangle$ in Def. 3) and the persuadees' AFs ($\langle A_i, R_i \rangle$ in Def. 3) are represented by the `PersuaderAF` class and the `PersuadeeAF` class respectively. An `MAPG` has exactly one persuader and multiple persuadees, captured by the multiplicity constraints in Figure 3 (1, resp. * for many). The persuader framework contains all `Arguments`, denoted by the composition link between `PersuaderAF` and `Argument`, while persuadee frameworks contain some subset of the arguments. Arguments may attack one another (captured by the `attacks` edge in Figure 3), and exactly one argument is distinguished as the `topic`. Multiple arguments can be identified as forming the strategy of an MAPG (S in Def. 3), captured via the `strategy` link between the `MAPG` and `Argument` classes.

For a particular persuader, audience and topic argument, we are interested in finding a strategy that is effective and also, perhaps, efficient. To do this with our SBME approach, we mutate the strategy using two mutation operators: the first adds a new argument to the strategy; the second removes an argument from the strategy. The rest of the model does not change. Applying these mutations to the solution candidates allows exploration of any strategy in the search space. Two objective functions are used to evaluate any strategy found: one determines its effectiveness and one determines its efficiency.

5. Evaluation of Application of SBME to Find Strategies for MAPGs

We ran experiments to investigate performance of our approach, looking both at the quality of solutions found and the time taken to find them. We used the SBME tool MDEOptimiser (MDEO)[2] to run the NSGA-II [5] algorithm over models that instantiate the metamodel we define in the previous section, where only the strategy is mutable. The NSGA-II algorithm uses a crowding distance comparator to ensure a diverse population and improve the fitness landscape exploration for problems that have more than one objective. The algorithm is run for 250 generations with population size 30. We used Tweety [22] together with the *argmat-sat* [1] argument solver to determine whether a particular persuadee is convinced by a strategy. The performance of the solver used to evaluate a strategy has a significant impact on the runtime of our implementation. In order to find the fastest solver, we have evaluated several solvers [20,22,1] for our use case using runtime. We found *argmat-sat* to be the fastest for our use case.

[2] https://mde-optimiser.github.io/

Across experiments, we varied: the number of arguments in the persuader's framework (`af-size`); the structure of the persuader's framework (`struct`); the number of persuadees (`p-num`); and the number of arguments known to each persuadee, expressed as a proportion of the number of arguments in the persuader's framework (`p-size`).

We performed experiments with a range of argumentation framework structures:
Ladders and **Cycles** These were used in Black et al.'s evaluation of their strategies for one-to-one persuasion dialogues [3]. We do not redefine the structures here but note they were designed to be especially challenging, due to the existence of arguments that may be both beneficial or detrimental for a persuader, depending on the persuadee's beliefs.
Trees These are standard directed trees, whose root is the topic argument. As a bipartite AF, these are expected to be less challenging for the persauder than ladder or cycle AFs, since asserting an argument supporting the topic does not risk dissuading persuadees.
Competition Frameworks We randomly selected three AFs from the set used in the 2017 argumentation solver competition[3], specifically one derived from a planning problem (with 490 arguments), one based on a Barabási-Albert network (with 160 arguments), and one translated from assumption-based argumentation (with 691 arguments).

For Ladders, Cycles, and Trees we can vary the size of the AF, but for the competition frameworks this is fixed. The framework is used as the persuader's AF. The `p-num` persuadees' AFs are uniformly random sub-graphs of the persuader's AF, each composed of `p-size` $\times$ `af-size` arguments (recall, `p-size` is a proportion), one of which is ensured to be the topic argument.

As no existing work allows generation of strategies for multi-audience persuasion games, we benchmarked our approach against the two naive approaches below.
Brute-force (BF) searches through all possible assertions (that is, the power set of the arguments in the persuader's AF) to find a strategy that maximises the number of persuadees that are convinced. If, during the search, a strategy is found that convinces all persuadees then the search terminated, otherwise the search is exhaustive. This approach is computationally intractable for large games, as shown in Table 3, but for smaller AFs it is feasible to use this approach to determine an optimal solution.
Random asserter (RA) first selects a uniformly random number of arguments to assert, from 0 to the size of the persuader's AF. Then a uniformly random subset of this size is selected from the arguments in the persauder's AF to assert.

We ran our experiments on Amazon Web Services Elastic Compute Spot instances. We used *c4.large* instances, running Amazon Linux. The experiments have been configured to run inside a Docker container running Java 1.8.0 and Amazon Linux version 2017.12.0.20180330. Each experiment has been performed on an individual machine, with 2 CPU cores and 2.5GB RAM allocated to the container. For each experiment, we ran MDEO 10 times and RA 10 times, so as to consider both average and best performance. The complete implementation and results are available online[4].

C1: MDEO finds strategies that are effective We compare the performance of our approach to RA and BF, considering here the single objective to maximise the number of persuadees who are convinced. We use three settings: (1) small games where `struct` $\in \{$`cycle, ladder, tree`$\}$ of `af-size` $\in \{21, 51, 101\}$ (+1 argument for trees), with `p-num` $\in \{1, 2, 5\}$ persuadees, with `p-size=` $\{.25, .5, .75\}$; (2) larger games where

[3] http://argumentationcompetition.org/2017
[4] https://github.com/mde-optimiser/comma-18-mapg

`struct` $\in$ {`cycle, ladder, tree`} of `af-size` $\in \{51, 101, 201\}$, with `p-num` $\in \{10, 50, 100\}$ persuadees, with `p-size` $\in \{.25, .5, .75\}$; (3) games using the competition frameworks, with `p-num` = 50 and `p-size` = .5.

As BF is an exhaustive search, the strategies it returns are guaranteed to be optimal. However, BF is computationally intractable and so we are unable to compute the best outcome for larger games. For the games where we were able to use BF to determine the best outcome (where `af-size` ≤ 21) *MDEO always found an optimal solution.*

Tables 1 and 2 compare performance of RA and MDEO, showing average effectiveness of each solution found (Ma for MDEO, Ra for RA) and effectiveness of the best solution found (Mb for MDEO, Rb for RA). For smaller games (Table 1) both approaches generally found the best solutions, but average effectiveness of the strategies found using MDEO is significantly better than for RA. For larger games (Table 2) MDEO produces better average solutions than RA, and the best solutions of MDEO are better than the best of RA. Cycle AFs proved difficult for both approaches, often resulting in failure to find a strategy that increases the number of convinced persuadees. We plan to investigate whether by giving MDEO a larger population of solutions, or more evolutions, we may be able to find solutions for cycle AFs at the cost of additional computational resources.

Results for the competition frameworks are shown in Table 4. Our approach was unable to cope with the largest of these frameworks (with 691 arguments), timing out after 24 hours, but was able to find effective strategies for the smaller competition frameworks.

C2: MDEO can find solutions to large problems For a single objective to maximise the number of convinced persuadees, we compare average time taken by MDEO to find a strategy with time taken by BF. Table 3 shows the results for small games. For games with `af-size` > 11, the MDEO approach is almost always faster than BF search. Exceptions to this (e.g. Ladder-21, with `p-num` and `p-size` 25%) are when BF gets 'lucky' and quickly finds a solution that convinces all persuadees. For games with `af-size` of 11, BF is faster. However, closer observation of the MDEO search reveals that the best solution is actually found in earlier generations. Therefore, for these scenarios, MDEO runtime can be improved by specifying a lower number of generations or by specifying an additional termination condition that stops after there is no improvement in solutions quality for a number of algorithm steps, without an effect on the quality of solution produced.

For larger games, with `af-size` up to 201 arguments and number of persuadees up to 50, MDEO returns results within 90 minutes. (Full results are omitted here for space reasons but can be found in our repository. [4]) This demonstrates the scalability of MDEO, both to the number of arguments in the domain, but also with increasing numbers of persuadees. Table 4 shows results for the competition frameworks: MDEO took more than 24 hours to run for the largest of these, just over an hour for the framework with 480 arguments, and less than 16 minutes for the smallest competition framework.

C3: MDEO can find strategies that satisfy multiple objectives Here we seek strategies that aim to both maximise the number of convinced persuadees and minimise the number of arguments asserted. We compare both efficiency and effectiveness of the strategies produced by MDEO and RA for this multi-objective case. To compare the quality of search solutions with two objectives we use the hypervolume (HV) unary quality indicator. [23]. The HV measures the volume of objective space dominated by a set of objectives that form a Pareto front. A Pareto front with higher HV value is considered better. To use RA to determine a Pareto front, each run consisted of a batch of 10 appli-

Table 1. Average and best effectiveness of solutions found by MDEO (respectively, Ma, top left, Mb, bottom left) and by RA (respectively, Ra, top right, and Rb, bottom right) for small games. The results in bold are the better performing approach for a game. Asterisks show results where all persuadees are convinced.

	p-num	1		1		1		2		2		2		5		5		5	
	p-size	25%		50%		75%		25%		50%		75%		25%		50%		75%	
struct	af-size	Ma	Ra	Ma	Ra	Ma	Ra	Ma	Ra	Ma	Ra	Ma	Ra	Ma	Ra	Ma	Ra	Ma	Ra
		Mb	Rb	Mb	Rb	Mb	Rb	Mb	Rb	Mb	Rb	Mb	Rb	Mb	Rb	Mb	Rb	Mb	Rb
Ladder	11	**1.0***	0.2	**1.0***	0.08	0.0	0.0	**0.0***	-1.03	**0.0**	-0.5	**0.0**	-0.47	**2.0**	-0.61	**1.0**	-1.29	0.0	0.0
		1.0*	1.0*	1.0*	1.0*	0.0	0.0	0.0*	0.0*	0.0	0.0	0.0	0.0	2.0	2.0	1.0	1.0	0.0	0.0
Ladder	21	**0.0***	-0.58	0.0	0.0	0.0	0.0	**1.0***	-0.48	**0.0***	-1.08	0.0	0.0	**0.0***	-2.87	**2.0**	-0.19	**0.0**	-0.56
		0.0*	0.0*	0.0	0.0	0.0	0.0	1.0*	1.0*	0.0*	0.0*	0.0	0.0	0.0*	0.0*	2.0	2.0	0.0	0.0
Ladder	51	**0.0***	-0.54	**0.0***	-0.63	**0.0***	-0.53	**2.0***	0.28	**0.0**	-0.48	**0.0***	-0.92	**2.0***	-1.32	**1.0**	-0.45	0.0	0.0
		0.0*	0.0*	0.0*	0.0*	0.0*	0.0*	2.0*	2.0*	0.0	0.0	0.0*	0.0*	2.0*	2.0*	1.0	1.0	0.0	0.0
Cycle	11	**1.0***	0.01	0.0	0.0	0.0	0.0	**1.0**	0.04	0.0	0.0	0.0	0.0	**1.0**	-1.35	0.0	0.0	0.0	0.0
		1.0*	1.0*	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	**1.0**	0.0	0.0	0.0	0.0	0.0
Cycle	21	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.0**	0.0	0.0	0.0	**1.7**	-0.85	0.0	0.0	0.0	0.0
		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.0**	0.0	0.0	0.0	**2.0**	0.0	0.0	0.0	0.0	0.0
Cycle	51	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Tree	12	**0.0***	-0.12	**0.0***	-0.5	0.0	0.0	**2.0***	0.51	**1.0***	-0.49	0.0	0.0	**2.0***	0.99	**2.0**	0.1	**1.0**	0.16
		0.0*	0.0*	0.0*	0.0*	0.0	0.0	2.0*	2.0*	1.0*	1.0*	0.0	0.0	2.0*	2.0*	2.0	2.0	1.0	1.0
Tree	22	**0.0***	-0.88	0.0	0.0	**0.0***	-0.75	**0.0**	-0.63	**2.0***	0.52	**0.0**	-0.44	**0.0**	-1.56	**2.0***	1.1	**0.0**	-2.19
		0.0*	0.0*	0.0	0.0	0.0*	0.0*	0.0	0.0	2.0*	2.0*	0.0	0.0	0.0	0.0	2.0*	2.0*	0.0	0.0
Tree	52	**0.0***	-0.67	0.0	0.0	0.0	0.0	**1.0**	0.16	0.0*	0.0*	0.0	0.0	**1.0***	0.34	**3.0**	0.17	0.0	0.0
		0.0*	0.0*	0.0	0.0	0.0	0.0	1.0	1.0	0.0*	0.0*	0.0	0.0	1.0*	1.0*	**3.0**	2.0	0.0	0.0

cations of RA (so we ran RA 10×10 times for each experiment). For space reasons, we consider only games based on the larger AFs, including competition frameworks, and do not consider cycles, for which it is hard to find a solution that satisfies a single objective.

For the larger tree and ladder problems, we compare the hypervolumes over 10 runs, included as box plots in Figure 4. In almost all cases, the average hypervolume obtained by MDEO is higher than that obtained with RA, indicating that MDEO outperformed RA. Furthermore, MDEO performance is more consistent than that of RA (in the box plots, vertically smaller plots indicate a smaller variance in the individual Pareto fronts). For competition-based games, MDEO was able to find a solution for frameworks with 160 and 480 arguments in a reasonable time (Table 4).

In two experiments (indicated on Figure 4 with asterisks) RA found a better solution than MDEO. We repeated these experiments, adding two additional mutation operators that can assign and remove 10 arguments each time (instead of 1). With these new operators, MDEO outperformed RA, indicating that only mutating a single argument may not always be sufficient to allow the search to escape a local maximum.

Across all experiments, the average time taken for MDEO to find the strategy for the two objective case was not statistically longer than the time for the single objective case. Indeed, due to the non-deterministic search of MDEO, there were many scenarios in which the two objective cases were faster. This demonstrates that there is no computational overhead for adding the additional objective.

6. Conclusions and Future Work

We have shown that we can use techniques from SBME to represent the multi-audience persuasion setting as a meta-model, to which we can apply evolutionary search to find persuader strategies that maximise the number of convinced persuadees. Our evaluation demonstrates that the approach produces strategies that are effective, and that it does so

Table 2. Average and best effectiveness of solutions found by MDEO (respectively, Ma, top left, Mb, bottom left) and by RA (respectively, Ra, top right, and Rb, bottom right) for large games. The results in bold are the better performing approach for a game. Asterisks show results where all persuadees are convinced.

	p-num	10		10		10		50		50		50		100		100		100	
	p-size	25%		50%		75%		25%		50%		75%		25%		50%		75%	
struct	af-size	Ma	Ra	Ma	Ra	Ma	Ra	Ma	Ra	Ma	Ra	Ma	Ra	Ma	Ra	Ma	Ra	Ma	Ra
		Mb	Rb	Mb	Rb	Mb	Rb	Mb	Rb	Mb	Rb	Mb	Rb	Mb	Rb	Mb	Rb	Mb	Rb
Ladder	51	**1.0**	-4.59	**2.6**	-1.55	0.0	**0.06**	**7.0**	-15.93	**8.8**	-10.25	**4.0**	-4.61	**12.9**	-28.44	**15.8**	-21.28	**12.8**	-10.0
		1.0	1.0	**3.0**	2.0	0.0	**1.0**	7.0	7.0	**10.0**	7.0	**5.0**	1.0	13.0	13.0	**17.0**	10.0	**14.0**	5.0
Ladder	101	**1.0**	-4.23	**3.2**	0.03	**4.1**	-0.81	**10.6**	-16.16	**10.0**	-8.85	**4.8**	-8.26	**19.9**	-27.27	**7.7**	-21.74	**12.4**	-5.78
		1.0	1.0	**4.0**	3.0	**5.0**	4.0	**11.0**	10.0	**10.0**	9.0	**6.0**	2.0	**22.0**	21.0	**9.0**	7.0	**13.0**	11.0
Ladder	201	**3.0**	-2.81	**0.1**	-2.09	**1.0**	-1.28	**7.4**	-17.34	**7.8**	-8.73	**0.2**	-6.13	**20.0**	-31.48	**18.7**	-17.39	**9.9**	-8.75
		3.0	3.0	1.0	1.0	1.0	1.0	**8.0**	7.0	**10.0**	8.0	**2.0**	1.0	20.0	20.0	**21.0**	16.0	**12.0**	7.0
Cycle	51	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**1.9**	0.0	0.0	0.0	0.0	0.0
		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	**2.0**	0.0	0.0	0.0	0.0	0.0
Cycle	101	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Cycle	201	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
		0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Tree	52	**4.0**	-0.18	**1.0**	-2.68	**1.0**	-0.63	**4.0**	-14.78	**6.2**	-15.66	**1.0**	-7.37	**35.0**	-17.85	**27.0**	-4.8	**0.0**	-0.67
		4.0	4.0	1.0	1.0	1.0	1.0	4.0	4.0	8.0	8.0	1.0	1.0	35.0	35.0	27.0	27.0	0.0	0.0
Tree	102	**1.0**	-4.44	**2.0**	-3.5	**4.0***	1.75	**11.0**	-11.69	**5.0**	-1.3	**1.0**	-5.56	**20.0**	-20.23	**36.0**	-22.96	**7.0**	-9.68
		1.0	1.0	**2.0**	1.0	4.0*	4.0*	11.0	11.0	**5.0**	3.0	1.0	1.0	**20.0**	7.0	**36.0**	25.0	7.0	7.0
Tree	202	**5.0**	-2.71	**5.6**	2.71	**6.2**	1.11	**25.0**	-9.37	**8.0**	-4.98	**8.3**	-5.11	**50.0**	-12.13	**42.0**	-11.87	**7.0**	-3.98
		5.0	4.0	6.0*	6.0*	**7.0**	6.0	**25.0**	21.0	**8.0**	4.0	**9.0**	3.0	**50.0**	23.0	**42.0**	29.0	**7.0**	3.0

Table 3. Comparison of average time taken by MDEO (M Avg Time, in HH:MM:SS:ms) with time taken by BF for small games. The faster approach is in bold. N/A indicates a solution could not be found in <24 hours.

	p-num	1	1	1	2	2	2	5	5	5
	p-size	25%	50%	75%	25%	50%	75%	25%	50%	75%
struct	af-size	M Avg Time	M Avg Time	M Avg Time	M Avg Time	M Avg Time	M Avg Time	M Avg Time	M Avg Time	M Avg Time
		BF Time	BF Time	BF Time	BF Time	BF Time	BF Time	BF Time	BF Time	BF Time
Cycle	11	00:00:11.395	00:00:12.381	00:00:13.313	00:00:22.299	00:00:21.723	00:00:23.961	00:00:52.838	00:00:51.912	00:00:58.086
		00:00:01.953	**00:00:07.727**	**00:00:05.871**	**00:00:13.061**	**00:00:15.044**	**00:00:15.816**	**00:00:41.273**	**00:00:36.301**	**00:00:41.116**
Cycle	21	**00:00:11.940**	**00:00:12.976**	**00:00:13.281**	**00:00:22.576**	**00:00:22.388**	**00:00:24.030**	**00:00:53.318**	**00:00:55.401**	**00:00:56.283**
		03:36:53.909	03:32:46.163	03:56:49.568	06:45:16.808	07:17:00.997	07:40:23.322	10:02:09.465	18:41:30.431	13:20:00.271
Cycle	51	**00:00:13.071**	**00:00:14.075**	**00:00:15.338**	**00:00:24.370**	**00:00:26.274**	**00:00:29.230**	**00:00:57.730**	**00:01:03.490**	**00:01:03.469**
		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Ladder	11	00:00:12.052	00:00:11.967	00:00:13.071	00:00:20.877	00:00:23.162	00:00:22.691	00:00:52.066	00:00:55.853	00:00:52.202
		00:00:00.009	**00:00:02.037**	**00:00:06.517**	**00:00:00.009**	**00:00:13.927**	**00:00:13.533**	**00:00:31.977**	**00:00:32.643**	**00:00:32.826**
Ladder	21	00:00:12.597	**00:00:12.502**	**00:00:12.262**	00:00:22.761	00:00:22.284	**00:00:22.852**	00:00:53.578	**00:00:54.698**	**00:00:58.530**
		00:00:00.257	01:54:49.975	01:56:35.045	**00:00:00.720**	**00:00:00.254**	03:54:35.445	**00:00:00.839**	18:36:18.219	18:58:09.416
Ladder	51	**00:00:13.840**	**00:00:14.349**	**00:00:15.588**	**00:00:24.631**	**00:00:27.745**	**00:00:26.854**	**00:00:54.616**	**00:00:59.969**	**00:01:05.558**
		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Tree	11	00:00:12.045	00:00:12.512	**00:00:11.860**	00:00:22.131	00:00:21.805	**00:00:23.251**	00:00:52.180	**00:00:52.359**	**00:00:52.013**
		00:00:00.011	**00:00:00.011**	00:00:25.352	**00:00:01.117**	**00:00:03.519**	00:00:50.224	**00:00:04.924**	00:02:10.301	00:02:10.088
Tree	21	00:00:12.011	**00:00:12.817**	00:00:13.271	**00:00:22.640**	00:00:23.136	**00:00:25.649**	**00:00:53.472**	00:00:55.301	**00:00:53.260**
		00:00:01.621	07:12:59.317	**00:00:00.550**	07:24:38.857	**00:00:06.757**	13:57:47.948	19:45:43.663	**00:00:05.547**	18:54:44.007
Tree	51	**00:00:12.976**	**00:00:14.427**	**00:00:15.116**	**00:00:24.586**	**00:00:25.851**	**00:00:27.659**	**00:00:56.856**	**00:00:57.988**	**00:01:04.171**
		N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

efficiently even for large and complex scenarios. Further, we have shown how MDEO can be adjusted to a multi-objective problem, in which the persuader minimises the number of arguments asserted, while maximising the number of convinced persuadees.

A key advantage of this SBME approach is that the high-level metamodel which encodes multi-audience persuasion games is easy to interpret and to adjust to other types of strategic argumentation problems. Having demonstrated here the potential of SBME for solving strategic argumentation problems, we plan to apply SBME techniques to other settings, such as those in which persuadees are able to respond to assertions of the

Table 4. Results for MDEO and RA on competition frameworks (BA:Barabási-Albert, PP:Planning-problem, AB:Assumption-based). *HV* shows average hypervolume, *eff* shows effectiveness (best and average). Times shown are averages. N/A indicates the solution took longer than 24 hours to find.

		Single Objective			Multi Objective		
		MDEO time	MDEO eff	RA eff	MDEO time	MDEO HV	RA HV
BA(160)	Avg	00:15:31.438	**10**	-2.64	00:15:04.175	**0.20**	0
	Best		**10**	1			
PP (480)	Avg	01:04:00.672	**34**	9.86	01:00:16.931	0.640	**0.649**
	Best		34	34			
AB(691)	Avg	N/A	N/A	0	N/A	N/A	0
	Best		N/A	0			

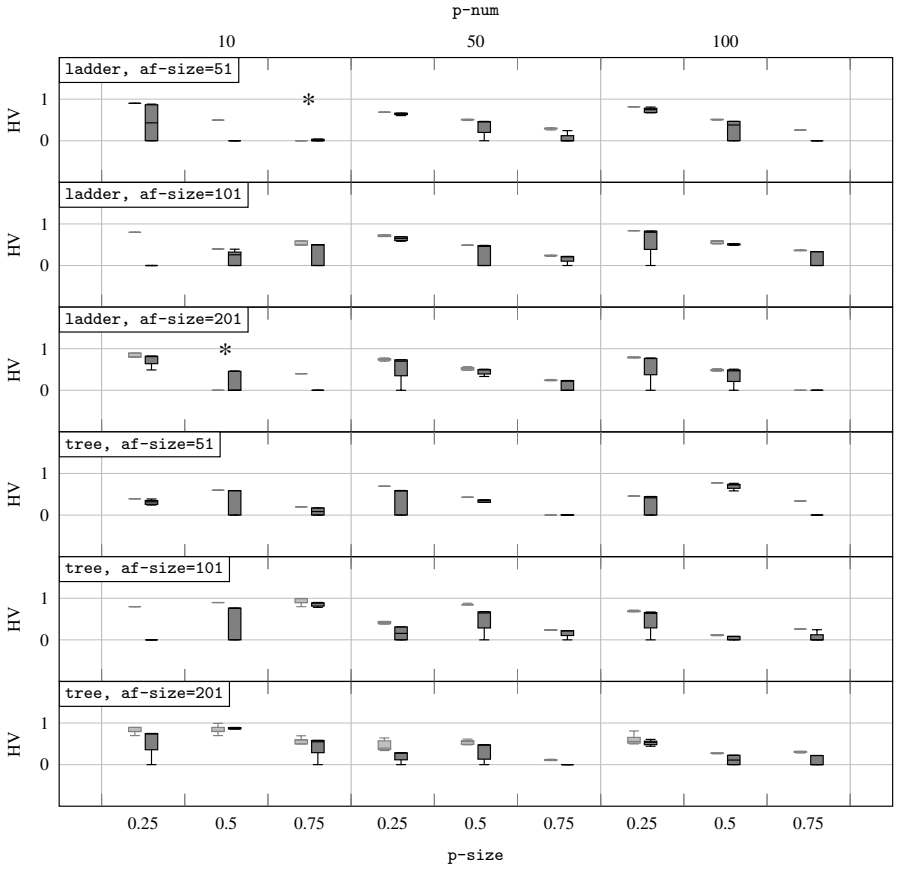

Figure 4. Multi-objective performance of MDEO and RA. Ticks on the top *x* axis shows number of persuadees in the scenario; bottom *x* axis shows number of arguments known to each persuadee (as a proportion of `af-size`). The size of the persuader's AF and the graph structure are included in the top left corner of each row. For each comparison, the light gray box plot on the left shows the spread of HVs obtained by MDEO and the dark gray box plot on the right shows the spread of HVs obtained by RA.

persuader, and in which the persuader has a probabilistic model of the persuadees' AFs.

Another avenue of future work is the implementation of more specific mutation operators that can select arguments which have a higher chance of increasing the strategy effectiveness. For example, we could use a heuristic that estimates the utility of assert-

ing a particular argument and specify the total utility as an additional objective to be maximised [17]. We are also interested in exploring if we can reduce the search time by trimming from the search space the arguments that do not support the topic.

References

[1] argumatrix argmat-sat. https://sites.google.com/site/argumatrix/argmat-sat. Accessed: 2018-04-02.

[2] T. J.M. Bench-Capon, S. Doutre, and P. E. Dunne. Audiences in argumentation frameworks. *Artificial Intelligence*, 171(1):42 – 71, 2007.

[3] E. Black, A. J. Coles, and C. Hampson. Planning for persuasion. In *Proc. of the 16th International Conference on Autonomous Agents and MultiAgent Systems*, pages 933–942, 2017.

[4] I. Boussaïd, P. Siarry, and M. Ahmed-Nacer. A survey on search-based model-driven engineering. *Automated Software Engineering*, 24(2):233–294, 2017.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[6] P. M. Dung. On the Acceptability of Arguments and its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and *n*-Person Games. *Artificial Intelligence*, 77(2):321–357, 1995.

[7] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Springer, 2nd edition, 2015.

[8] E. Hadoux, A. Beynier, N. Maudet, P. Weng, and A. Hunter. Optimization of probabilistic argumentation with Markov decision models. In *Proc. of the 24th International Joint Conference on Artificial Intelligence*, pages 2004–2010, 2015.

[9] E. Hadoux and A. Hunter. Strategic sequences of arguments for persuasion using decision trees. In *Proc. of the 31st AAAI Conference on Artificial Intelligence*, pages 1128–1134, 2017.

[10] A. Hunter. Making argumentation more believable. In *Proc. of the 19th AAAI Conference on Artificial Intelligence*, pages 269–274, 2004.

[11] A. Hunter. Towards higher impact argumentation. In *Proc. of the 19th AAAI Conference on Artificial Intelligence*, pages 275–280, 2004.

[12] A. Hunter. Probabilistic strategies in dialogical argumentation. In *Proc. of the 8th International Conference on Scalable Uncertainty Management*, pages 190–202. Springer, 2014.

[13] A. Hunter. Towards a framework for computational persuasion with applications in behaviour change. *Argument and Computation*, 9(1):15 – 40, 2018.

[14] A. Hunter and M. Thimm. Optimization of dialectical outcomes in dialogical argumentation. *International Journal of Approximate Reasoning*, 78:73–101, 2016.

[15] M. Kessentini, P. Langer, and M. Wimmer. Searching models, modeling search: On the synergies of SBSE and MDE. In *Proc. of the 1st International Workshop Combining Modelling and Search-Based Software Engineering*, pages 51–54, 2013.

[16] P. McBurney and S. Parsons. Chance discovery using dialectical argumentation. In *Proc. of JSAI 2001 International Workshop on Chance Discovery*, pages 414–424, 2001.

[17] J. Murphy, E. Black, and M. Luck. A heuristic strategy for persuasion dialogues. In *Proc. of the 6th International Conference on Computational Models of Argument*, pages 411 – 418. IOS Press, 2016.

[18] H. Prakken. Combining sceptical epistemic reasoning with credulous practical reasoning. In *Proc. of the 1st International Conference on Computational Models of Argument*, pages 311–322, 2006.

[19] T. Rienstra, M. Thimm, and N. Oren. Opponent models with uncertainty for strategic argumentation. In *Proc. of the 23rd International Joint Conference on Artificial Intelligence*, pages 332–338, 2013.

[20] O. Rodrigues. Eqargsolver system description. In *Proceedings of the 4th International Conference on Theory and Applications of Formal Argumentation*, pages 150–158, 2018.

[21] A. Rosenfeld and S. Kraus. Strategical argumentative agent for human persuasion. In *Proc. of the 22nd European Conference on Artificial Intelligence*, pages 320–328, 2016.

[22] M. Thimm. Tweety - A comprehensive collection of Java libraries for logical aspects of artificial intelligence and knowledge representation. In *Proc. of the 14th International Conference on Principles of Knowledge Representation and Reasoning*, pages 528–537, 2014.

[23] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[24] S. Zschaler and L. Mandow. Towards model-based optimisation: Using domain knowledge explicitly. In *Proc. Workshop on Model-Driven Engineering, Logic and Optimization*, 2016.