Proceedings, The Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-16)

Best-Fit Action-Cost Domain Model Acquisition and Its Application to Authorship in Interactive Narrative

Thomas Hayton, Peter Gregory, Alan Lindsay, and **Julie Porteous**
Digital Futures Institute
School of Computing
Teesside University, UK

Abstract

Domain model acquisition is the problem of learning the structure of a state-transition system from some input data, typically example transition sequences. Recent work has shown that it is possible to learn action costs of PDDL models, given the overall costs of individual plans. In this work we have explored the extension of these ideas to narrative planning where cost can represent a variety of features (e.g. tension or relationship strength) and where exact solutions don't exist. Hence in this paper we generalise earlier results to show that when an exact solution does not exist, a best-fit costing can be generated. This approach is of particular interest in the context of plan-based narrative generation where the input cost functions are based on subjective input. In order to demonstrate the effectiveness of the approach, we have learnt models of narratives using subjective measures of narrative tension. These were obtained with narratives (presented as video in this case) that were encoded as action traces, and then scored for subjective narrative tension by viewers. This provided a collection of input plan traces for our domain model acquisition system to learn a best-fit model. Using this learnt model we demonstrate how it can be used to generate new narratives that fit different target levels of tension.

Introduction

In our work we were interested in extending the *NLOCM* domain model acquisition system, which learns action costs in planning domains, to learn models of narrative planning domains in order to develop tools to assist in the task of narrative authoring. However, a feature of narrative planning domains is that cost models are based on subjective input which don't necessarily yield exact action cost models and thus required extensions to *NLOCM*. In this paper we present our new system $NLOCM_{BF}$, standing for *Numeric LOCM Best-Fit*, which extends the *NLOCM* domain model acquisition system, to learn action cost models in domains in which no exact action cost model can be found.

Domain model acquisition can be thought of as *automated modelling*. Modelling is the activity of formally specifying a problem that can then be reasoned about using automated techniques. Modelling in the context of interactive narrative is more accurately called authorship. Within the interactive narrative setting, numeric variables can represent varied structures, such as strength of relationships in social networks (Porteous, Charles, and Cavazza 2013; 2015) and the level of tension (Porteous et al. 2011) within a certain scene. In addition to evaluating on standard benchmarks, we apply $NLOCM_{BF}$ in the context of interactive narrative. Specifically, we apply our system to the task of automated story authorship, allowing variations of stories to be created from existing ones. We learn cost models for a narrative planning representation of the Scooby-Doo[1] cartoon, where the cost approximates subjective narrative tension, and then demonstrate their use for generating new Scooby-Doo episodes. We show that $NLOCM_{BF}$ can approximate challenging benchmarks with a typical error of around 10% to 15%, and can accurately model subjective narrative tension with around 10% error.

Background

This work builds upon two different strands of research: that of domain model acquisition and plan-based interactive narrative. We now consider each of these areas.

Domain Model Acquisition

In this work, what we refer to as domain model acquisition, is the learning of a planning domain from example data that includes sequences of state transitions. This work builds on the *LOCM* family (Cresswell, Mccluskey, and West 2009; Cresswell and Gregory 2011; Gregory and Cresswell 2015) of domain model acquisition systems. These algorithms are all developed with the aim of learning from the smallest amount of input data possible, preferably only the transition sequences. The *LOCM* algorithms learn models of domains in which each object type is represented by one or more finite state machines. Figure 1 shows an example of the state machines learnt by *LOCM* in the Transport domain. Each state represents a particular state that an object type can hold. Within each state, state parameters (shown in square brackets) record temporary associations that an object has. For example, the package state machine has a state parameter of a location to record its position. One weakness of *LOCM* and

[1] SCOOBY-DOO and all related characters and elements are copyright and trademark Hanna-Barbera

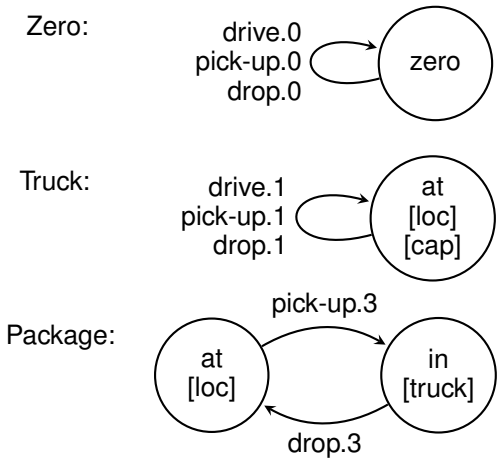

Figure 1: The finite state machines derived by *LOCM* in the transport domain for the two interesting object types: package and truck. The truck state machine has a single state, with two state parameters for the location of truck and truck capacity.

LOCM2 is that they only learn the dynamic aspects of domains. *LOP* learns the static relations in planning domains, such as road networks in logistic-style domains.

NLOCM (Gregory and Lindsay 2016) addresses the problem of learning domain models in the presence of action costs. In addition to the transition sequences, *NLOCM* also has the complete cost of each plan as input. The system then learns the numeric action costs using a constraint programming approach. *NLOCM* works by first learning a planning model using *LOP* without action costs, and then each plan can be seen as a linear equation over the transitions within that plan. Importantly, it uses the information gained from the *LOCM2* state machines and *LOP* static relations in order to filter the choices for which operator parameters are hypothesised to participate in the cost function.

Interactive Narrative

Planning has been widely applied to the problem of story generation in Interactive Narrative (IN) systems e.g. (Young 2000; Aylett, Dias, and Paiva 2006; Riedl and Young 2010; Gilroy et al. 2013). However, these systems pose interesting challenges since planning goals need not equate to the end of the narrative (they can evolve over time), re-planning is frequently required in response to user interaction and sub-optimal trajectories may be needed in order to shape a trajectory to display narrative concepts. One of particular interest to us is narrative tension as in the classic Aristotelian arc. In this work we were motivated to explore the use of a best-cost fit model to drive generation of different narrative variants.

In order to provide some degree of author control over narrative shape a number of declarative mechanisms have been proposed that help bridge the gap between authors on the one hand and the planner on the other. These include the approach of (Riedl 2009) which allowed for the specification of *author goals* to *complexify* the planning problem (Riedl 2009), HTN approaches where the information is embedded in method decompositions (Hoang, Lee-Urban, and Munoz-Avila 2005; Kelly, Botea, and Koenig 2007) and meta-level control via authored *constraints* (Porteous et al. 2011).

For our experiments with learnt cost models, which we report in section , we have adapted the approach of (Porteous et al. 2011) and use authored constraints as pseudo-landmarks which are used as intermediate goals to guide generation of plans that display the required narrative shape, i.e. visit interesting narrative states (situations). As illustration, for our Scooby-Doo domain there are certain elements which are frequently required in episodes, such as introducing the monster and the team starting investigating, represented as:

(introducedMonster ?sc-storyControl)
(startInvestigating ?sc-storyControl)

Then in order to enforce their inclusion, and any required orders, in a generated narrative plan they can be modelled in the narrative planning problem using the PDDL3 modal operators *sometime* and *sometime-before*. For example, in the Scooby-Doo domain, to force the introduction of the monster to occur at some stage in the narrative before the team start investigating then this can be represented as:

(sometime-before (startInvestigating ?sc-storyControl)
(introducedMonster ?sc-storyControl))

During narrative generation the partially ordered authored constraints are used to decompose the process of narrative generation into a sequence of sub-problems, where each sub-problem has a constraint as its goal and the planner generates a narrative for each decomposed sub-problem in turn. A final narrative can be assembled by composition of the sub-narratives in turn, however, since planning is embedded in an IN system a complete plan is not output in the traditional sense: instead sub-narratives are presented to a user as they are generated (mapped to pre-recorded video segments).

The $NLOCM_{BF}$ Domain Model Acquisition System

The *NLOCM* system (Gregory and Lindsay 2016) is a domain model acquisition system for domains which include action costs. The *NLOCM* system learns action costs by modelling the task as a constraint programming problem. Underlying the approach is the assumption that a plan's cost can be explained by costs associated with each action. In this work, we relax this assumption, aiming instead to find an allocation of cost that minimises the error in the model. This is necessary for our application to subjective measures of tension, however, it can also allow generating approximate models in domains with large search spaces (as demonstrated in Subsection). In this section we present the construction of the problem model in *NLOCM* and its relaxation to the best-fit model.

NLOCM Domain Model Acquisition System

In order to discuss the constraints model generated by *NLOCM*, we introduce a formalisation of the system's input. The system operates from a collection of plans, denoted

by Π, and a cost function over these plans, based on the input costs. For P input plans, we have:

$$\Pi = [\pi^1, ..., \pi^P]$$

$$c(\pi^i) = \text{input cost} \qquad (1 \leq i \leq P)$$

We also define a length function, based on the number of ground actions in a plan.

$$l(\pi^i) = \text{input length} \qquad (1 \leq i \leq P)$$

We use the subscript to refer to ground actions within each plan, in the following way:

$$\pi^i = [\pi^i_1, ..., \pi^i_{l(\pi^i)}]$$

And finally, we can define the set of all observed ground actions as the union of all of the actions seen in all plans:

$$\mathcal{A} = \bigcup_{\pi \in \Pi} \{\pi_1, ..., \pi_{l(\pi)}\}$$

We also denote by $\mathcal{A}_O \subseteq \mathcal{A}$ the set of ground actions observed of the operator O. Using this formalism, we will now describe the constraints model generated by *NLOCM*.

Base Model In the base model, ground action costs are encoded and constraints are posed over the input plans. Therefore, for each ground action, an integer variable is defined:

$$A_{1..|\mathcal{A}|} : Integer \tag{V1}$$

Then for each input plan, a linear constraint is encoded, such that the sum of the costs of its individual actions is equal to the cost of the entire plan:

$$\sum_{i=1}^{l(\pi^p)} (\pi^p_i) = c(\pi^p) \qquad (\pi^p \in \Pi) \tag{C1}$$

Any two identical ground actions in the input plans are represented by the same variable in the constraint model. This is true even between different input plans, so long as they are drawn from the same state space (i.e. the same planning problem). The base model can be used to learn a model of the ground action costs for a specific set of problem instances.

Encoding Operator and Ground Action Templates The second layer of the model allows the cost of the ground actions to be further broken down into its specific contributors. For each operator, O, a set of templates, T_O, is defined over the arguments that have been determined as interesting (see Subsection). The number of possible templates makes a complete exploration impractical, so different subsets are explored in *NLOCM*. The choice of templates in *NLOCM* was driven pragmatically from the structure of target formalism, and was also informed by the output of *LOCM2* and *LOP*. The templates can be defined generally as follows:

$$T_O \subseteq \mathbb{P}(\text{args}(O)) \qquad (O \in \mathcal{O})$$

It is useful to define $T_A = T_O$, where A is an instantiating action of operator O. Each template within T_O defines a set of arguments that are assigned a constant cost in each problem instance. The system uses variables to represent ground template values in order to discover the contribution of the template to the specific costs of actions in the input plans. For each operator, O, the following variables are used:

$$\text{Ground}^A_\tau : Integer \quad (A \in \mathcal{A}_O, \tau \in T_O) \tag{V2}$$

For example, $\text{Ground}^{(\text{GangLoseMember Shaggy Museum})}_{\{2\}} = 50$ defines a variable that represents the model's cost associated with the second parameter of the GangLoseMember action. In this case associating the specific parameter, Museum, with a cost of 50. Following the assumption that these contributors are the only contributors to the cost of the operator, the ground action cost is equated with the sum of the corresponding action templates. For each operator, O:

$$A = \sum_{\tau \in T_A} \text{Ground}^A_\tau \qquad (A \in \mathcal{A}_O) \tag{C2}$$

Given an adequate selection of templates, this model will support searching for satisfying action cost configurations, as was the intention in *NLOCM*.

Best-Fit Cost Model

We now consider how this model can be relaxed so that action costs can be learnt for domains where plan costs cannot be accurately explained. Our approach is to relax the *NLOCM* model and instead encode the best fit model. This is supported with the introduction of an error variable for each input plan:

$$E_{1..|\Pi|} : Integer \tag{V3}$$

And replace (C1) with the constraint stating that the sum of the ground action costs plus the new error term, is equal to the plan cost:

$$E_p + \sum_{i=1}^{l(\pi^p)} (\pi^p_i) = \pi^p \qquad (\pi^p \in \Pi) \tag{C3}$$

The error terms can therefore be summed, and the total error minimised. The final solution returned represents the action costs that best explain the input plan costs given the templates that were used as input.

Selection of Contributing Templates

As mentioned previously, there are different templates available to select what contributes to the action cost of each operator in the domain. Although we are exploring best-fit models and therefore there will be a solution for any set of templates, we have explored several layers of increasingly complex template sets. This allows us to explore the balance between model error and model complexity. The ordering of the templates is as follows:

1. Operator costs: Operator costs are those fixed costs associated to an operator in all problem instances. The most obvious example is the unit operator cost in pure STRIPS domains.

2. Action costs: Action costs are instance-specific costs assigned to operators. These are less common in the benchmark domains.
3. State Parameters: As discussed in the background section, *LOCM* and *LOCM2* identify state parameters by finding object associations within operator argument lists. Certain operators change the object identified in the state parameter. For example, the location state parameter in the truck machine of the Transport domain (Figure 1) is changed by arguments 2 and 3 of the drive action. Because these relationships are known to be significant, we add their corresponding templates as potential contributors to the action costs.
4. Static Parameters: Similarly to state parameters, we now consider adding templates based on the static relations identified by the *LOP* system. Recall that *LOP* identifies a static relation for each operator. The scope of these static relations thus forms a new template for each operator.
5. Single Transition Costs: This step allows costs to be associated with individual operator arguments (imagine, for example, that the Museum makes a far more tense setting of a scene than the Mines). In *NLOCM* the inclusion of single parameter templates was made using a series of steps, where each step allowed increasing numbers of single parameter templates to contribute to a single action cost. Our approach is to allow their combination in one layer; therefore using those that best combine to explain the cost.

Empirical Evaluation

We now provide a discussion of our empirical evaluation on several planning benchmarks and an application from interactive narrative. All of our experiments are run on CentOS Linux using a 16-core Intel Xeon E5-2665 with 64 GB system memory. *NLOCM*$_{BF}$ is implemented in Java (version 1.7.65) using the Choco constraint library (Prud'homme, Fages, and Lorca 2014) version 3.3.3. We use the domain over weighted degree variable selection heuristic, with a geometric restart policy. We use an hour time cutoff for any one constraint search.

Hard Action Cost Domains

Empirical analysis in (Gregory and Lindsay 2016) shows that the majority of action costs in benchmark domains can be learnt exactly using the *NLOCM* system. Three domains, however, are currently out of reach for this system using the default parameterisation. These are the Transport domain, the Woodworking domain and the Elevators domain. The domains all have an exact action costing, but involve multiple costing templates. Since these domains have exact solutions, they form a useful benchmark set for the *NLOCM*$_{BF}$ algorithm.

Table 1 shows the result of running *NLOCM*$_{BF}$ on increasingly large input sets in these three challenging domains. For each domain, 100 random walks of length 10 were generated for the first five instances of the IPC benchmark instances. The column labelled 'OC' shows the absolute error of *NLOCM*$_{BF}$ on the entire plan set alongside the percentage error. In several cases, *NLOCM*$_{BF}$ finds the exact best-fit operator costs for the problem instances. The 'BF' column shows the best result gained using further templates than operator costs. Although *NLOCM*$_{BF}$ fails to find the overall best-fit action costs (these domains have zero-error costings), learning an approximate costing (between 8% and 16% here) provides a better solution than having no costing at all, and using more complex templates can lead to smaller errors.

Domain	#	OC	Err	BF	Err
Transport	1	2344*	11%	1687	8%
	2	5341*	13%	5341	13%
	3	7575*	12%	7575	12%
	4	10744*	13%	10744	13%
	5	12965*	12%	12965	12%
Woodworking	1	2540	14%	2473	13%
	2	5794	14%	5738	14%
	3	8920	15%	8752	14%
	4	12271	16%	12045	15%
	5	15384	16%	15384	16%
Elevator	1	1055*	11%	1055	11%
	2	2372	12%	2372	12%
	3	4033	13%	4033	13%
	4	5549	14%	5549	14%
	5	6943	14%	6943	14%

Table 1: Results of *NLOCM*$_{BF}$ on challenging benchmark instances. This table reports the absolute error and the percentage error when running *NLOCM*$_{BF}$ on between one and five plan sets in each domain, each with 100 plans. Asterisks denote when the result was proven optimal. OC refers to the Operator Cost level of the search strategy, BF refers to the best found value using any selection on contributing templates. Err in both cases refers to the percentage error in the solution.

Scooby-Doo: A Case-Study

In order to test *NLOCM*$_{BF}$ on data without an exact solution, we applied it to an interactive narrative application, and we built a domain model based on the cartoon Scooby-Doo. The rationale for choosing this narrative domain is that the formulaic nature of the show means that a large number of actions will be shared across the episodes. This allows us to model multiple episodes of the show whilst maintaining a relatively small modelling domain. Scooby-Doo is targeted at children which is ideal for two reasons: scenes that are intended to be tense have been created so that they convey this in an obvious manner. To help in gathering data it should be clear which actions increase the tension for the audience. Secondly, episodes of the show are fairly short, approximately 20 minutes, and contain a large number of actions. Since input to *NLOCM*$_{BF}$ will be generated by people subjectively rating the tension of a story, using Scooby-Doo allows a large amount of actions to be sampled in a relatively short time.

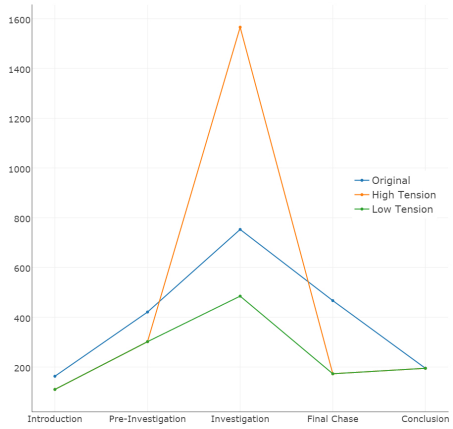

Figure 2: The narrative tension arcs for the three different version of the Scooby-Doo episode shown in Figure 4

Data Collection

A sample of episodes were selected for creation of the domain model (episodes 1-4, 21 from Scooby-Doo, Where Are You! (1969-1970)). From analysis of the episodes a general formula, i.e. baseline, was identified as shown in Figure 3. It consists of 5 phases and within these the narrative is structured around a number of sub-states. Despite its formulaic nature the domain still allows for considerable variation around the baseline since there are multiple sequences of action paths through the different phases. For example, alternative ways that the the first story event can unfold include the gang discovering a crime scene, an early encounter with the monster, or the gang getting lost somewhere. Similarly, during the investigation phase some of the alternatives include: the gang splitting up and reuniting, members becoming lost or trapped, and the gang meeting new characters.

Since we collect data per scene in the story, we can use the $NLOCM_{BF}$ system with a cost for each ground action in the training data. Training data from the ScoobyDoo domain was gathered by subjective ratings of narrative tension of selected episodes. Ratings were collected from 10 participants who watched 2 episodes, resulting in 4 ratings per episode. While watching episodes, participants moved a mouse along a scale of 0-100. No clicking or dragging was required so participants could remain focused on the episodes with minimal distraction. Ratings were recorded every half second for the duration of the episode. The start and end 10% of each scene was ignored so that ratings recorded in transitions between scenes weren't taken into account.

Narrative Generation

Whereas (Porteous et al. 2011) relied on manual assignment of tension levels to constraint factss, in our model the tension levels have been learnt from approximation to the subjective estimates obtained during our user study. During narrative generation the constraint-based decomposition approach is augmented to use a target accumulated tension level for the individual story phases, specified for each story phase as:

$$\text{lower\_bound} < \text{tension} < \text{upper\_bound}$$

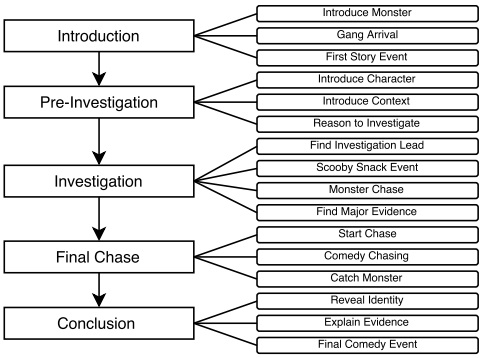

Figure 3: A schematic diagram showing the 5 phases of a Scooby-Doo episode and the sub-goals for each phase

We investigated the impact of changes in the target accumulation bounds impacted on the narrative generation and show that changes in the target tension enabled the generation of alternative narratives, as shown in Figure 2.

Table 2 shows the result of the learning Scooby-Doo models for each phase of each episode. The table shows the users' subjective rating for each episode, alongside the learnt model both for the operator cost level of the search strategy, and the complete search strategy. Note that for the full stories, including more templates always improves the accuracy of the result. Once learnt, the planning model can be used in order to generate new episodes of Scooby-Doo, or to present the existing episodes in an alternative way.

Related Work

This work builds on the rich literature on domain model acquisition in planning. $NLOCM_{BF}$ improves upon the ideas in the *NLOCM* system, providing a way of learning action costs in planning domains. *NLOCM* itself inherits ideas and the general philosophy of learning using minimal input from the *LOCM* systems (Cresswell, Mccluskey, and West 2009; Cresswell and Gregory 2011; Gregory and Cresswell 2015), and before those, OpMaker (McCluskey et al. 2009; Richardson 2008). These all target the OCL (McCluskey and Porteous 1997) language, either directly or indirectly, as the structure of the state-machines underlying the formalism provides constraints crucial to learning with sparse input. Other domain model acquisition systems require more input information (examples include ARMS (Wu, Yang, and Jiang 2007) and LAMP (Zhuo et al. 2010), which can target part of the ADL fragment of PDDL, or learn models in the presence of noisy and incomplete data (Mourao et al. 2012).

This work also contributes to the emerging body of work on planning for interactive narrative which challenges received wisdom about plan quality criteria and shifts the focus to properties of plan trajectories which determine narrative structure. Examples include the manipulation of narrative tension to drive plot development (Porteous et al. 2011; Gilroy et al. 2013) and manipulation of discourse level mechanisms to maintain user interest via the introduction of suspense (Cheong and Young 2014; Bae and Young 2008) and cinematic representation (Jhala and Young 2010).

Figure 4: Storyboard representations of Scooby-Doo Variants: (1) the baseline Scooby-Doo episode; (2) and (3) are episodes generated using the learnt planning model representing high and low tension, respectively. For the high tension episode the high cost narrative content is highlighted.

Episode	Phase	User Ave.	Op. Cost	Best Fit
Episode 1	Intro	131.25	160	163
	Pre-Inv.	357.25	432	421
	Investi.	882	747	753
	Chase	544	392	467
	Concl.	130.75	140	195
	Total	2045.25	1871	1999
Episode 2	Intro	99.25	154	101
	Pre-Inv.	332.5	429	386
	Investi.	1374.5	1458	1463
	Chase	138.5	111	111
	Concl.	60.5	140	108
	Total	2005.25	2292	2169
Episode 3	Intro	116.25	154	137
	Pre-Inv.	318.5	386	355
	Investi.	989.75	1112	1073
	Chase	369.75	276	312
	Concl.	60	140	102
	Total	1854.25	2068	1979
Episode 4	Intro	33	154	102
	Pre-Inv.	314.25	501	437
	Investi.	1060.25	1107	1099
	Chase	136.25	90	90
	Concl.	107.75	140	140
	Total	1651.5	1992	1868
Episode 5	Intro	116.25	154	132
	Pre-Inv.	480.25	512	512
	Investi.	1160	1062	1081
	Chase	244	162	168
	Concl.	116.75	140	143
	Total	2117.25	2030	2036

Table 2: Learnt tension levels over the story phases of the five Scooby-Doo episodes. The 'Op Cost' column is the value learnt in the operator cost level of the search strategy. 'Best Fit' refers to the best cost for any template set. Considering more templates improves the accuracy of $NLOCM_{BF}$.

Conclusions

We have introduced the $NLOCM_{BF}$ domain model acquisition system that extends a previous approach to learning domain models with action costs, to learn approximations of cost functions when no exact cost function exists. Narrative planning provided our motivating application area. $NLOCM_{BF}$ has been shown as effective, providing models with roughly 10% - 15% error in benchmark instances previously out of range for domain model acquisition systems.

We have also demonstrated the utility of $NLOCM_{BF}$ in learning models of subjectively scored plans (narratives presented as videos and rated by users), which will have natural variation, thus providing new ways to aid automatic authorship of interactive narrative.

Acknowledgements

This work is supported by EPSRC Grant EP/N017447/1.

References

Aylett, R.; Dias, J.; and Paiva, A. 2006. An Affectively Driven Planner for Synthetic Characters. In *Proc. of 16th Int. Conference on Automated Planning and Scheduling (ICAPS)*.

Bae, B., and Young, R. 2008. A Use of Flashback and Foreshadowing for Surprise Arousal in Narrative Using a Plan-Based Approach. In Spierling, U., and Szilas, N., eds., *Proc. of 1st Int. Conf. on Interactive Digital Storytelling (ICIDS 2008)*, 26–29.

Cheong, Y.-G., and Young, R. M. 2014. Suspenser: A Story Generation System for Suspense. *Transactions on Computational Intelligence and Artificial Intelligence in Games*.

Cresswell, S., and Gregory, P. 2011. Generalised Domain Model Acquisition from Action Traces. In *International Conference on Automated Planning and Scheduling*, 42 – 49.

Cresswell, S. N.; Mccluskey, T. L.; and West, M. M. 2009. Acquisition of Object-Centred Domain Models from Planning Examples. In *ICAPS*, 338 – 341.

Gilroy, S.; Porteous, J.; Charles, F.; Cavazza, M.; Soreq, E.; Raz, G.; Ikar, L.; Or-Borichov, A.; Ben-Arie, U.; Klovatch, I.; and Hendler, T. 2013. A Brain-Computer Interface to a Plan-based Narrative. In *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence (IJCAI-13)*.

Gregory, P., and Cresswell, S. 2015. Domain Model Acquisition in the Presence of Static Relations in the LOP System. In *International Conference on Automated Planning and Scheduling*, 97–105.

Gregory, P., and Lindsay, A. 2016. Domain Model Acquisition in Domains with Action Costs. In *International Conference on Automated Planning and Scheduling*.

Hoang, H.; Lee-Urban, S.; and Munoz-Avila, H. 2005. Hierarchical Plan Representations for Encoding Strategic Game AI. In *Proceedings of Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE-05)*. AAAI Press.

Jhala, A., and Young, R. M. 2010. Cinematic visual discourse: Representation, generation, and evaluation. *Computational Intelligence and AI in Games, IEEE Transactions on* 2(2):69–81.

Kelly, J. P.; Botea, A.; and Koenig, S. 2007. Planning with Hierarchical Task Networks in Video Games. In *Proceedings of the ICAPS-07 Workshop on Planning in Games*.

McCluskey, T. L., and Porteous, J. 1997. Engineering and compiling planning domain models to promote validity and efficiency. *Artificial Intelligence* 95(1):1–65.

McCluskey, T. L.; Cresswell, S. N.; Richardson, N. E.; and West, M. M. 2009. Automated acquisition of action knowledge. In *International Conference on Agents and Artificial Intelligence (ICAART)*, 93–100.

Mourao, K.; Zettlemoyer, L.; Petrick, R. P. A.; and Steedman, M. 2012. Learning STRIPS Operators from Noisy and Incomplete Observations. In *Uncertainty in Artifical Intelligence*, 614 – 623.

Porteous, J.; Teutenberg, J.; Pizzi, D.; and Cavazza, M. 2011. Visual programming of plan dynamics using constraints and landmarks. In *International Conference on Automated Planning and Scheduling*, 186–193.

Porteous, J.; Charles, F.; and Cavazza, M. 2013. NarrativeING: using character relationships for interactive narrative generation. In *International Conference on Autonomous Agents and Multi-agent Systems*, 595–602.

Porteous, J.; Charles, F.; and Cavazza, M. 2015. Using Social Relationships to Control Narrative Generation. In *AAAI*, 4311–4312.

Prud'homme, C.; Fages, J.-G.; and Lorca, X. 2014. *Choco3 Documentation*. TASC, INRIA Rennes, LINA CNRS UMR 6241, COSLING S.A.S.

Richardson, N. E. 2008. *An Operator Induction Tool Supporting Knowledge Engineering in Planning*. Ph.D. Dissertation, School of Computing and Engineering, University of Huddersfield, UK.

Riedl, M. O., and Young, R. M. 2010. Narrative Planning: Balancing Plot and Character. *Journal of Artificial Intelligence Research* 39:217–267.

Riedl, M. 2009. Incorporating Authorial Intent into Generative Narrative Systems. In *Proc. of AAAI Spring Symposium on Intelligent Narrative Technologies*.

Wu, K.; Yang, Q.; and Jiang, Y. 2007. ARMS: An automatic knowledge engineering tool for learning action models for AI planning. *The Knowledge Engineering Review* 22(2):135–152.

Young, R. M. 2000. Creating Interactive Narrative Structures: The Potential for AI Approaches. In *AAAI Spring Symposium in Artificial Intelligence and Entertainment*. AAAI Press.

Zhuo, H. H.; Yang, Q.; Hu, D. H.; and Li, L. 2010. Learning complex action models with quantifiers and logical implications. *Artificial Intellgence* 174:1540–1569.