Multi-Actor Planning for Directable Simulations

Mubbasir Kapadia, Shawn Singh, Glenn Reinman, Petros Faloutsos
University of California, Los Angeles
Email: mubbasir, shawnsin, reinman, pfal @cs.ucla.edu

***Abstract*—There has been growing academic and industry interest in the behavioral animation of autonomous actors in virtual worlds. However, it remains a considerable challenge to automatically generate complicated interactions between multiple actors in a customizable way with minimal user specification.**

In this paper, we propose a behavior authoring framework which provides the user with complete control over the domain of the system: the state space, action space and cost of executing actions. Actors are specialized using *effect* and *cost* modifiers – which modify existing action definitions, and *constraints* which prune action choices in a state-dependent manner. *Behaviors* are used to define goals and objective functions for an actor. Actors having common or conflicting goals are grouped together to form a *composite domain*, and a multi-agent planner is used to generate complicated interactions between multiple actors. We demonstrate the effectiveness of our framework by authoring and generating a city simulation involving multiple pedestrians and vehicles that interact with one another to produce complex multi-actor behaviors.

***Keywords*-behavior authoring; centralized planning; multi-objective constraint satisfaction;**

I. Introduction

Multi-actor simulation is a critical component of cinematic content creation, disaster and security simulation, and interactive entertainment. For example, a director may want to author massive armies in movies, autonomous characters interacting in games, or a panicked crowd in an urban simulation. Scenarios are authored by specifying objectives and constraints on one or more actors with actors having common or competing goals. In general, this can be formulated as a multi-objective constraint satisfaction optimization problem which must be solved to generate the simulation.

Generating such behaviors in crowds has been studied extensively from many different perspectives [1]. These methods represent different tradeoffs between the ease of user specification and the autonomy of behavior generation:

Scripted approaches [2], [3], [4] describe behaviors as predefined sequences of actions where small changes often require far-reaching modifications of monolithic scripts. Most crowd approaches [5], [6] use goals and parameters to add heterogeneity to their simulations. The work in [7] maps parameters to personality traits and examine the emergent behaviors in crowds. The work in [8] uses natural language instructions to define goals for *smart* avatars.

Approaches such as Improv [9], LIVE [10], SmartBody [11] and commercial systems like Massive [12] describe behaviors as rules which govern how actors act based on certain conditions. These systems are *reactive* in nature: i.e. they produce pre-defined behaviors corresponding to the current situation and are not equipped to generate complicated agent interactions that pan out over the course of an entire simulation. Cognitive approaches [13], [14], [15] use complex cognitive models such as decision networks and neural networks to model knowledge and action selection in virtual agents.

The use of domain-independent planners [16], [17], [18] is a promising direction for automated behavior generation. Planning approaches provide automation at the expense of computation. Also, collaboration among agents requires the overhead of a centralized planner or the modeling of agent communication. Hence, current systems [19], [20] using planners for behavior generation are restricted to simple problem domains (small state and action space) with a small number of agents exhibiting limited interaction.

To our knowledge, no prior work provides a flexible means of specification with little effort, while generating complex interactions between multiple actors. The far reaching goal that still remains a considerable challenge is this: to provide an animator with the ability to easily orchestrate complicated "stories" between multiple interacting actors that can be easily customized and is portable across scenarios, with minimal user specification.

In this paper, we extend existing domain definition languages to allow the *specialization* of existing actor definitions and present a multi-agent planner that works in the composite domain of interacting actors. Our framework allows complex behaviors between multiple interacting actors to be generated in a flexible way. This paper makes the following contributions:

- **Modular and Natural Specification:** Domain experts define the state and action space for different scenarios while end-users can modify and constrain existing definitions to add variation and purpose to their simulation. Specializations can focus on different levels of abstraction and can be as general or specific as necessary. Behaviors are specified as goals and objectives for actors that are triggered based on their current state.
- **Cooperative and Competitive Planning:** Complicated interactions between multiple actors can be authored by specifying common or contradicting goals for actors in the scenario. Our method automatically clusters actors

that have dependent goals to define a *composite* state and action space. This avoids the complexity of modeling communication between actors or the need for explicit scripting of cooperation schemes in actors. Collaborative behaviors arise as a solution found by the multi-agent planner which minimizes the combined cost of actions of all actors in the composite space.

II. Behavior Specification and Generation

Figure 1 presents an overview of our framework. A domain expert first defines the problem domain (state and action space) of the actors in the scene. Next, a director specializes the actors using modifiers, constraints, and behaviors. Actors with dependent goals or constraints enforcing their interaction are grouped together into a composite domain, forming a set of independent domains. For each of these domains, a multi-actor planner generates a trajectory of actions for all actors in that domain that satisfies the composite goal while optimizing the individual objective of each actor. The result of each search is combined into a global plan which is executed to generate the resulting simulation.

A. Domain specification

Domain specification entails defining the state space, the action space and, costs of executing actions for actors in a scene. An actor is an entity which has a state and can affect the state of itself or other actors by executing actions. Different actors (e.g., pedestrians, cars, and even objects in the environment) in the same scenario may have different state and action spaces.

We represent the state space of an actor using *metrics* – physical or abstract properties of an actor that are affected by the execution of actions. *Costs* are a numerical measure of executing an action (e.g., distance and energy). The action space of an actor is a set of actions which it can perform in any given state. Actions affect one or more metrics of an actor. An action has the following properties: (1) preconditions which determine if an action is possible in a given state, (2) the effect of the action on the state of the actor as well as target actors and, (3) the cost of executing an action.

B. Domain Specialization

End users can re-use existing actor definitions to specify vastly different simulations in an intuitive manner by using specializations.

Modifiers. Users can specialize the effects and costs of executing an action in a state-dependent manner using modifiers. For example, an *effect modifier* can be placed on elderly actors to reduce their normal speed of movement. *Cost modifiers* indicate what actions are in an actor's best interest at a particular state. For example, a cautious actor can be authored by increasing the cost of actions that may place the actor in danger (e.g. enter a burning building). Here, the notion of *danger* would be a user specified metric in the state space of these actors.

Constraints. Constraints are used to enforce strict requirements on actors in a scenario. Constraints can be used to prune the action choices of an actor in a particular state. For example, constraints can be used to prevent pedestrians from walking on the road and obey traffic signals. Constraints can also be placed on the trajectory of the simulation to author specific events (e.g. two cars must collide), generate complex interactions between actors, and direct the high-level story.

Behaviors. Behaviors defines the current goal and objective function of an actor. The goal of an actor is a desired state that the actor must reach, while the objective function is a weighted sum of costs that the actor must optimize. The objective function o of an actor is specified by setting the weights $\{w_i\}$ of the different cost metrics $\{c_i\}$, and is defined as $o = \min(\sum_i w_i \cdot c_i)$. A user can define multiple behaviors for an actor which are activated depending on the current state.

C. Composite Search Domain and Problem Definition

Actors with dependent goals or constraints enforcing their interaction are grouped together to form a set of independent composite domains. The composite state space is the cartesian product of the states of each actor and the composite action space is the union of the actions of each actor in the composite domain. The composite domain is denoted by Σ.

A particular problem instance $\mathtt{P} = (\Sigma, s_0, g, \{o_i\})$ is defined by determining the initial state s_0, composite goal g , and objectives $\{o_i\}$ of each actor in the composite domain. The composite goal, g, is the logical combination of the goals for all actors in the composite domain. Common goals are combined using an $\wedge$ operator, indicating that all actors must satisfy their goal. Contradicting goals are combined using an $\vee$ operator, indicating that any one of the actors must satisfy their goal.

A composite goal can thus be one of the following: (1) objective(s) for a single actor (e.g. get a hot dog and meet a friend at the park), (2) common objectives for a group of actors (e.g. two actors collaborating to lift a heavy load), (3) conflicting objectives between actors (e.g. the objective of the thief is to steal from the victim while the objective of the victim is to protect his money), or (4) combination of common and conflicting objectives (e.g. two actors collaborating to catch a third actor).

Since the planner works in the composite space of multiple actors, complicated interactions between actors that may be collaborating or competing with one another can be generated, without the need of global centralized planning across all actors in the scenario. Even though the actions of an actor only affect the state space of the composite domain it belongs to, the possibility of an action is determined by considering the global state space of all actors in the scenario. This is done to ensure collision-free trajectories

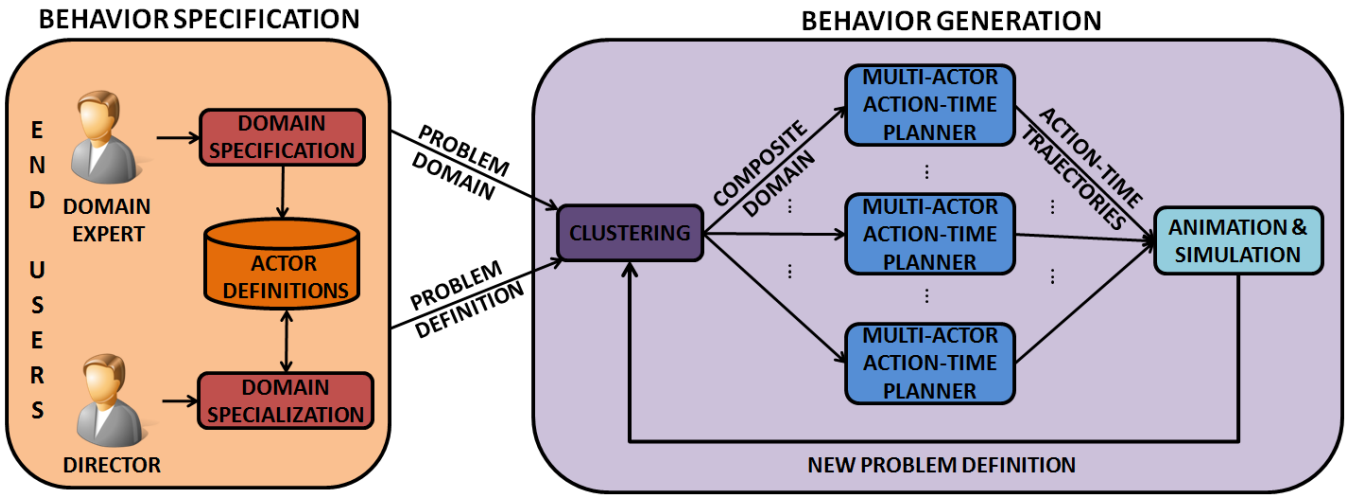

Figure 1. An overview of the framework.

between two independent plans. As a result, the action trajectories generated for actors in different groups can be overlayed to generate a complete simulation.

D. Multi-Actor Action-Time Planner

The input to the planner is the problem definition $\mathtt{P} = (\Sigma, s_0, g, \{o_i\})$, described above. The search process generates a trajectory of actions for all actors in the composite space which meets the composite goal, g while optimizing the individual objectives, $\{o_i\}$ of all actors in the group. Our planner extends traditional planning approaches [16], [17] as follows: (1) it works in the composite space of multiple actors with competitive or collaborative goals and, (2) it explicitly takes time into account with different actions taking variable amounts of time and actions of different actors overlapping.

Overview. Using a best-first planner, the search tree expands as follows: For the current state, a set of possible *transitions* is first generated. Each transition represents the forward simulation of the actions for all actors in the composite space by one time step, where actors are simultaneously executing actions. The planner chooses a transition by minimizing the sum of the *total cost* of the transition. The cost of a transition is computed such that the action chosen by an actor optimizes its own objective function. When the planner reaches a state which satisfies the composite goal g, it returns the generated plan.

Transitions. A transition represents the simultaneous execution of actions chosen by all actors in the composite domain by one time step. A transition is said to be *valid* and ready to simulate if all the actors have a valid action that they are executing or ready to execute. An action for a particular actor possible if all the following conditions are met: (1) the actor is currently not executing an action, (2) the preconditions of the action are satisfied and, (3) no constraints prohibit the action. For a valid transition, the actions of all actors are simulated for one time step in a random order. The explicit modeling of time in the action definition results in overlapping actions, actions being partially executed (action failure) and actors choosing to perform new actions while other actors are still performing their current action.

After the simulation of a transition, an actor may find itself in one of the following states: (1) *Success.* The action is successfully completed and the actor must chose a new action in the next time step. (2) *Executing.* The action is partially executed at the end of the simulation routine for that time step. (3) *Failure.* The preconditions of the action are negated as a result of the execution of actions of other actors. The action is said to have failed and the actor must choose a new action.

Cost Function. The cost of simulating a transition $\{a_i\}$, at state s, in the time interval $(t, t+1)$ where a_i is the action chosen by actor i in the composite domain is given by: $\sum_i o_i(\{c_j\})$, where $\{c_j\}$ are the values of the cost metrics for simulating action, a_i for actor i at state, s, from time, t to $t+1$, and o_i is the objective function of actor i.

III. City Simulation

We demonstrate the effectiveness of our framework by authoring a car accident in a busy city street and observing the repercussions of the event on other actors that are part of the simulation, such as the old man and his son, whose behaviors are automatically generated using our framework.

A. Actor Specification and Specialization

Three generic actors, each having their own state and action space are defined as follows:

- **Pedestrians:** Pedestrians have a position, orientation, speed of movement and radius to model their move-

(a) (b) (c) (d)

Figure 2. Snapshots of the city simulation authored using our framework: (a) Actors queue up at a hot dog stand while the vendors talk to one another. (b) Cars giving right of way to pedestrians. (c) Cautious actors run to a place of safety in the event of an accident. (e) Fire-fighters extinguish the fire while daring actors look on.

ment in the environment. The action, `Move` is defined to kinematically translate the actor and has an associated distance and energy cost. In addition, pedestrians have the following abstract metrics: hunger, safety, and amount of money. Actions such as `Eat` are modeled to affect particular abstract metrics and have an associated cost metric. Pedestrians are constrained to move on the sidewalks, use the crossings when the signal turns green and are given high-level behaviors (e.g. satisfy their hunger by getting a hot dog, meet a friend at the park).

- **Vehicles:** The state and action space of vehicles is defined similarly to simulate their movement. In addition, they have a metric `damage` which increases if a vehicle collides with another vehicle. Vehicles are constrained to stay on the roads, give right of way to pedestrians, and obey the traffic lights.
- **Traffic Signals:** A traffic signal represents an environment actor that models the simulation of the traffic signals at the intersection. It has a single metric `signal state` which is the current state that the traffic signals at the intersection are in. An action, `ChangeTrafficSignal` determines the state of the traffic signal based upon the current simulation time. The pedestrian and the vehicles query the signal state in order to follow the traffic signals.

These generic actors are specialized as follows:

- **Fire-fighters:** Fire-fighters have a common goal to extinguish any fires that may be present in the city block. These actors have proportionally lower weights for the safety metric, allowing them to move closer to a dangerous situation in performing their jobs.
- **Elderly:** An elderly person is specialized by reducing his walking speed and giving him a goal to follow his grandson.
- **Grandson:** The objective of the grandson is to escort his grandfather at all times and to keep him away from danger (e.g. car accidents, oncoming traffic and other pedestrians). We achieve this by changing the objective function of the grandson to include the safety cost metric of the grandfather as well.
- **Cautious and Daring Actors:** Cautious or daring actors are authored by simply increasing or decreasing the cost of actions that place them in danger. Here, `danger` is a user-defined cost metric that is associated with each actor in the scenario.
- **Reckless Vehicle:** A reckless vehicle is modeled by increasing its speed and relaxing the constraints of obeying traffic signals and collisions with other vehicles.

B. *Results*

We populate a city block with specialized pedestrians and vehicles using our framework. We observe pedestrians walking along the sidewalks in the city in a goal-oriented manner (satisfying hunger by getting a hot dog, going to the park to meet a friend, stopping to take a look at objects of interest) while obeying constraints and modifications (obey traffic lights, avoid collisions, stay off the streets etc).

In order to add drama to the simulation, we introduce constraints on the trajectory of the entire simulation. First, we introduce a constraint, that an accident must happen (i.e. two vehicles must collide). A simulation is generated where two reckless vehicles collide with one another, resulting in a fire that stops the traffic at the intersection (Figure 2(d)). Cautious pedestrians who are near the accident run away to a safe distance in panic or walk away calmly (depending on their specialization) while daring actors approach the scene of the accident.

The car accident triggers the activation of the behaviors in the fire-fighters who run to the location of the fires. They work together collaboratively to extinguish both fires (a result of the planner working in the composite domain).

Collisions and Animation. Since the planner takes time into account and always considers the global state of all actors (not just the actors in the composite domain) while generating a valid action trajectory, the resulting simulation is guaranteed to be collision-free. The output of the planner is input to a simple animation system which produces a visualization of animated characters as shown in the video.

Performance and Implementation Details. We demonstrate 106 actors in the city simulation, with 15 cars and 91 pedestrians. Based on constraints, goal definitions and spatial locality, the following composite domains are defined: (1) 15 cars and 4 fire-fighters, (2) old man and

son and, (3) generic pedestrians grouped together based on spatial locality. Dividing the problem domain into smaller composite domains reduces the branching factor of the search by two orders of magnitude, reducing an intractable search problem to smaller, more feasible searches. The plans for each of these domains is then overlayed to form the complete solution. The performance results are provided in Figure 3. The amortized performance of our behavior generation framework for the results shown in the video is 0.02 seconds per actor per second of simulation generated.

Number of actors	106
Number of composite domains	12
Max # of actors in a composite domain	19
Total generation time	219 sec
Max generation time for one domain	76 sec
Min generation time for one domain	8 sec
Generation time per actor	2.06 sec
Length of output simulation	95 sec
Amortized time per actor per second	0.02 sec

Figure 3. Performance Results.

IV. Conclusion and Future Work

In this paper, we present a multi-actor planning framework for generating complicated behaviors between interacting actors in a user-authored scenario. Users define the state and action space of actors and *specialize* existing actor definitions to add variety and purpose to their simulation. Actors with dependent goals are grouped together into a set of independent composite domains. For each of these domains, a multi-actor planner generates a trajectory of actions for all actors to meet the desired behavior. We author and demonstrate a simulation of more than one hundred actors (pedestrians and vehicles) in a busy city street and inject heterogeneity and drama into our simulation using specializations. With help from the community, we envision a growing open-source library of actor definitions that can be re-used to direct completely new simulations, with a few simple specializations.

Limitations. The behaviors generated by our framework are heavily dependent on the manner in which actors are grouped together and the weights of the objectives. For example, authoring interactions between the old man and firemen would necessitate the old man and firemen belonging to the same planning domain. For future work, we will adopt a dynamic clustering strategy in which actors may be re-grouped based on their current states.

One of the major design choices of this framework was the use of a multi-actor planner which prohibits its use in interactive applications such as games. The major reason for adopting a centralized approach was to facilitate the authoring of complex multi-actor interactions which would require a complex model of communication and prediction between actors in a decentralized system. We are currently investigating the use of anytime planners [21] as well as parallel search algorithms [22] for both centralized and agent-based planning in an effort to achieve real-time performance.

Acknowledgements

We wish to thank the anonymous reviewers for their comments. The work in this paper was partially supported by NSF grant No. CCF-0429983. We thank Intel Corp., Microsoft Corp., and AMD/ATI Corp. for their generous support through equipment and software grants.

References

[1] Norman Badler, *Virtual Crowds: Methods, Simulation, and Control (Synthesis Lectures on Computer Graphics and Animation)*, Morgan and Claypool, 2008.

[2] Aaron Bryan Loyall, *Believable agents: building interactive personalities*, Ph.D. thesis, Pittsburgh, PA, USA, 1997.

[3] Michael Mateas, *Interactive drama, art and artificial intelligence*, Ph.D. thesis, Pittsburgh, PA, USA, 2002.

[4] Hannes Vilhjálmsson, Nathan Cantelmo, Justine Cassell, Nicolas E. Chafai, Michael Kipp, Stefan Kopp, Maurizio Mancini, Stacy Marsella, Andrew N. Marshall, Catherine Pelachaud, Zsofi Ruttkay, Kristinn R. Thórisson, Herwin Welbergen, and Rick J. Werf, "The behavior markup language: Recent developments and challenges," in *Proceedings of IVA '07*, 2007, pp. 99–111.

[5] Mankyu Sung, Michael Gleicher, and Stephen Chenney, "Scalable behaviors for crowd simulation," *Computer Graphics Forum*, vol. 23, no. 3, pp. 519–528, 2004.

[6] Adriana Braun, Soraia R. Musse, Luiz P. L. de Oliveira, and Bardo E. J. Bodmann, "Modeling individual behaviors in crowd simulation," *Computer Animation and Social Agents*, vol. 0, pp. 143, 2003.

[7] Funda Durupinar, Jan Allbeck, Nuria Pelechano, and Norman Badler, "Creating crowd variation with the ocean personality model," in *Proceedings of AAMAS'08*, Richland, SC, 2008, pp. 1217–1220.

[8] Rama Bindiganavale, William Schuler, Jan M. Allbeck, Norman I. Badler, Aravind K. Joshi, and Martha Palmer, "Dynamically altering agent behaviors using natural language instructions," in *In Autonomous Agents*. 2000, pp. 293–300, ACM Press.

[9] Ken Perlin and Athomas Goldberg, "Improv: a system for scripting interactive actors in virtual worlds," in *Proceedings of ACM SIGGRAPH*, New York, NY, USA, 1996, pp. 205–216, ACM.

[10] Eric Menou, "Real-time character animation using multi-layered scripts and spacetime optimization," in *Proceedings of ICVS '01*, London, UK, 2001, pp. 135–144, Springer-Verlag.

[11] Marcus Thiebaux, Stacy Marsella, Andrew N. Marshall, and Marcelo Kallmann, "Smartbody: behavior realization for embodied conversational agents," in *Proceedings of AAMAS'08*, Richland, SC, 2008, pp. 151–158.

[12] Massive Software Inc., "Massive: Simulating life," 2010, www.massivesofware.com.

[13] Wei Shao and Demetri Terzopoulos, "Autonomous pedestrians," in *Proceedings of the ACM SIGGRAPH/EG Symposium on Computer Animation*, New York, NY, USA, 2005, pp. 19–28.

[14] Qinxin Yu and Demetri Terzopoulos, "A decision network framework for the behavioral animation of virtual humans," in *Proceedings of the ACM SIGGRAPH/EG Symposium on Computer Animation*, Aire-la-Ville, Switzerland, 2007, pp. 119–128.

[15] Bruce Mitchell Blumberg, *Old tricks, new dogs: ethology and interactive creatures*, Ph.D. thesis, 1997, Supervisor-Maes, Pattie.

[16] Richard E. Fikes and Nils J. Nilsson, "Strips: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, 1971.

[17] Avrim L. Blum and Merrick L. Furst, "Fast planning through planning graph analysis," *ARTIFICIAL INTELLIGENCE*, vol. 90, no. 1, pp. 1636–1642, 1995.

[18] Kutluhan Erol, James Hendler, and Dana S. Nau, "Htn planning: Complexity and expressivity," in *Proceedings of AAAI*. 1994, pp. 1123–1128, AAAI Press.

[19] John Funge, Xiaoyuan Tu, and Demetri Terzopoulos, "Cognitive modeling: knowledge, reasoning and planning for intelligent characters," in *Proceedings of ACM SIGGRAPH*, New York, USA, 1999, pp. 29–38.

[20] Vincent Decugis and Jacques Ferber, "Action selection in an autonomous agent with a hierarchical distributed reactive planning architecture," in *AGENTS '98: Proceedings of the second international conference on Autonomous agents*, New York, NY, USA, 1998, pp. 354–361, ACM.

[21] Maxim Likhachev, Dave Ferguson, Geoff Gordon, Anthony Stentz, and Sebastian Thrun, "Anytime search in dynamic graphs," *Artif. Intell.*, vol. 172, no. 14, pp. 1613–1643, 2008.

[22] A. Y. Grama and V. Kumar, "A survey of parallel search algorithms for discrete optimization problems," *Orsa Journal of Computing*, 1993.