

Search-based Planning for Character Animation

Miguel Lozano¹, Steven J. Mead², Marc Cavazza², Fred Charles²

¹University of Valencia
Dr. Moliner 50 (Burjassot) Valencia, Spain.
miguel.lozano@uv.es

²School of Computing and Mathematics
University of Teesside, TS1 3BA Middlesbrough, United Kingdom.
{m.cavazza, f.charles, steve.j.mead}@tees.ac.uk}

1 Introduction

Planning is the most generic AI technique to generate intelligent behaviour for virtual actors, in computer animation or computer games. In the animation domain, planning capabilities will consist in finding a right sequence of actions that let an agent achieve pre-defined goals. Each action generated can be played in the environment to produce an animation. Hence entire animations can be generated from first principles, by defining a set of actions and allocating high-level goals to the character. This makes possible not only to generate intelligent behaviour, but also to explore the diversity of courses of action. In recent years, several researchers have described the use of planning systems to control characters' behaviours.

Geib [5] has proposed the use of refinement planning following a detailed study of animation requirements [5][8]. Funge has used situation calculus to generate intelligent behaviours for virtual actors [4], and Cavazza et al. [2][3] have approached this problem with Hierarchical Task Networks (HTNs) for storytelling, considering the *knowledge intensive* nature of this kind of applications.

The planning requirements for virtual actors depend on the specific application, however we can identify these essential requirements:

- a. The domain representation should be appropriate to embodied agents in virtual environments and identify both goals and physical actions.
- b. Solution plans should be computed efficiently, considering the time scale of a virtual agent.
- c. In some cases when the agent evolves in a dynamic environment, there is need to interleave planning and execution as well.

2 Planning for animation

Novel planning techniques have been developed in recent years, especially search-based planning techniques [1], which have the potential to meet the above requirements. We have thus decided to experiment with a Heuristic Search Planner (HSP). To support these experiments, we provide two example problems: Firstly, to explain our use of HSP for animation we provide an extension of the classical *dinner-date* planning problem described by Weld, where an agent must compute a plan to achieve all the necessary to prepare a dinner, such as removing the garbage, wrapping a present, etc. We have extended this problem with more operators (*music*, *computer-work...*) but also with new goals and preconditions, such as to finish the work or to require fun for cooking. This provides us an opportunity to investigate with a non-decomposable, non-empty delete-lists planning problem on a similar application unlike the second example. The second example provided introduces the potential of HSP as a technique for story dramatisation, which is based upon earlier work into interactive storytelling [3]. In the story presented, the lead male character is attempting to ask the lead female character on a date - his plan should take him through the appropriate set of actions to achieve this goal (e.g. acquiring her gift preferences, getting the gift, getting her alone, etc.) Because the various actions can be associated to different locations in a 3D environment and the sequence of actions explored by the planner has an impact on the character's motion into the environment, these examples are well suited to illustrate differing animation applications.

Heuristic Search Planners can be characterised by three elements:

- a) the domain representation of the problem,
- b) the search algorithm and
- c) the heuristic function. In the next sub-sections, we will review the integration of these elements in our behavioural animation domains.

2.1 The domain representation

Our approach is based on the typical state-model representation for planning domains [1]. Basically, each state will contain a set of atoms representing the agent state (see Figure 1, e.g. (*cleanHands, not garbage, not work...*)). To complete the problem formulation the agent will require a set of operators that will represent its 'effector' capacity, mapping states to successor states according to its preconditions. These operators can be represented using a STRIPS-like formulation, which will also be used in the computation of heuristics for the search process. As we introduced before, the quality of the agent plans will be directly related to their lengths, and the monotonic progression towards the goal (at least in a static environment). For instance, an agent who washes his hands before carrying out the garbage will have to wash his hands again.

Problem: funny-dinner-date		
Initial State garbage, clean-hands, quiet, work		
Goal State dinner, present, not garbage, not work		
Operator	Preconditions	Effects
Cook	clean-hands	dinner
Watch-TV		fun, not quiet
Wash		clean-hands, not quiet
Carry		garbage, clean-hands
Phone	clean-hands	fun, not quiet
Relax		quiet
Paint	quiet	fun, not clean-hands
computer-work	clean-hands	not fun, not clean-hands, not work

Figure 1 – The Planning Operators

To achieve this, we are managing at search time a depth bounding criteria, which will prune all the plans beyond the maximum length plan allowed d . Considering that the embodied agents should achieve their goals through plans with no actions repeated, we have initialised d as the total number of operators the agent can apply. In this way, the depth level reached by a goal state of any plan-solution will represent its plan length and this will be the necessary information to consider in the final agent decision taking. Figure 2 shows an scheme of the system architecture, where 3D embodied agents can carry out their plans in the 3D virtual environment.

Taking into account the domain representation introduced, the next subsection will present MinMin as an

adequate search algorithm to supply the planning requirements for our 3Dembodied agents.

2.2 Planning with MinMin

MinMin [6] has been proposed as a search algorithm for real time decision taking. It has the advantage of searching forward from the current state to a fixed depth horizon and then computes the heuristics values for the frontier nodes. Furthermore, MinMin provides a forward search method able to interleave planning and action execution, and to extract the minimum-length plans required. As Geffner pointed out, the heuristics calculation associated to every node in classical HSPs, is the most expensive computational step associated to HSPs, and MinMin reduces this calculation to the search horizon nodes [1].

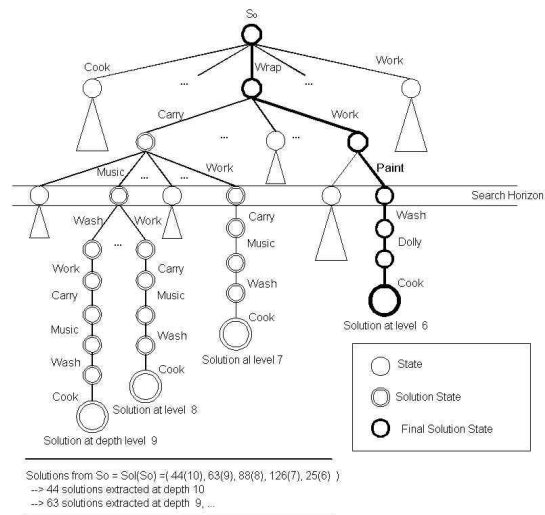


Figure 2. MinMin search

Figure 2 shows how MinMin is capable of refining its solutions during the search using a dynamic depth-bounding criterion. According to this, during the plan-search this bounding factor d should be decreased to the length of the last best plan extracted (the minimum one). This bounding is also useful to overcome the main problem of MinMin, that of cycling. A secondary bounding criterion introduced to MinMin in order to improve its efficiency. This second bounding (2-B) simply detects if creation of a new state with no new effects and thus prunes it (e.g. S_0 -Carry - $S_{useless}$, S_0 -Relax - $S_{useless}$). The performance of the whole planning system at the *funny-dinner-date* problem introduced will be shown, as the rest of tests, in the results section.

MinMin control is also adequate to extract the shortest-length plans, though not always the optimal one, as each node will select the child with the minimum cost (i.e. the node which could be part of a minimum length-plan

solution). In this way, at the root node tree the agent can perform an informed action selection mechanism, deciding at each plan step the shortest strategy or sub-plan which let him to achieve his goals.

We are using the independent domain heuristics presented by Bonet and Geffner in [1] which can be easily adequate to MinMin search domains. Heuristics are computed from the horizon nodes by ignoring *delete-list* and expanding the atomic facts that belong to post-conditions until all the atomic facts corresponding to the goal are met. Then the depth-level reached by this goal node will be treated as the necessary information to help MinMin in its decision taking.

3 Results

The system has been fully implemented and tested over a number of initial configurations, in a graphic environment corresponding that to the *funny* dinner-date problem, for which corresponding solutions can be produced in the form of 3D animations, which is visualized with the Unreal™ engine. The planner outputs this sequence of actions to the engine via UDP, which is interfaced by UnrealScript™, the engine's scripting language.

Figure 3 illustrates the search carried out by MinMin to generate a plan solve the *funny dinner-date* problem as presented previously, where the solution-vector associated to each search state indicates the total number of solutions or plans extracted by MinMin in several depth levels.

The agent starts searching from its initial state S_0 using MinMin, and will obtain solutions or plans from 10 to 6 length. Then at the top of the tree it will try to apply the first operator associated to the last minimum plan calculated (e.g., $S_0 - wrap - S_1$). As shown in Figure 3, once the agent has executed an action, it should update its own internal state (eg. $S_1 = (S_0) + present$) performing future searches from this (S_1), interleaving in this way planning and action execution, and achieving finally an intelligent autonomous behaviour able to reduce the distance to its goals.

On this problem, the overall performance obtained by MinMin (*search horizon* = 3) has been adequate to 3D real time graphics environments. Furthermore, restricting at S_0 the maximum plan length ($d = 7$), MinMin finds 6 plan length solutions in 0.082 seconds. The system is thus fast enough to support real-time animation, with a good potential for scaling up.

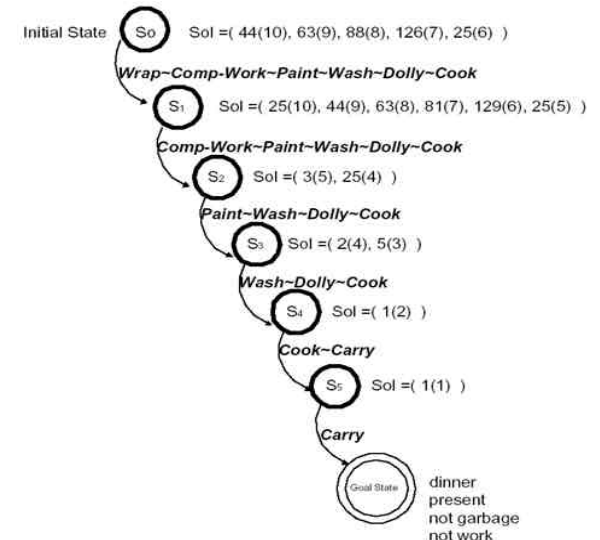
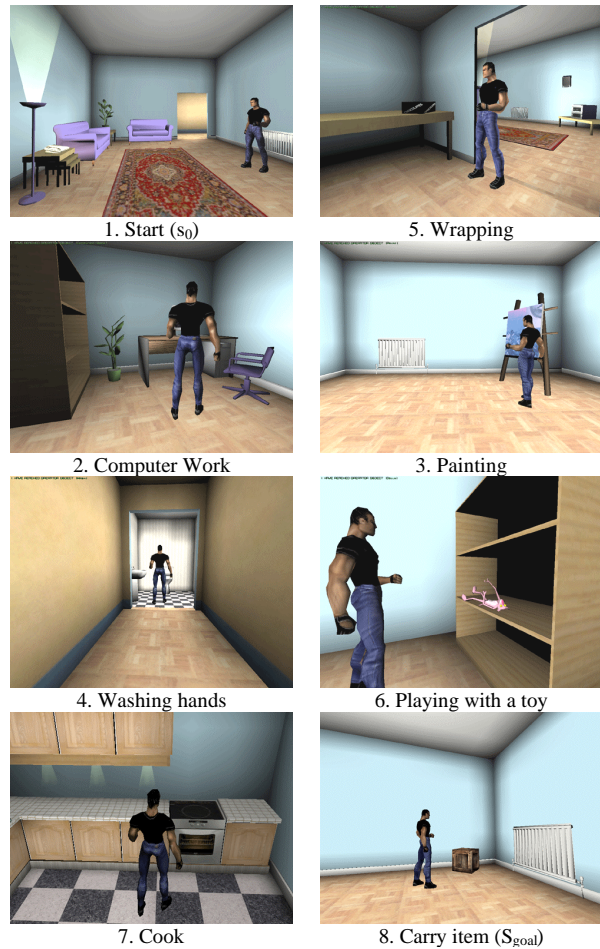


Figure 3. Generating Character's Behaviour with Search-based Planning

4 Integrating MinMin in storytelling domains

Storytelling systems are mainly interested in the emergence of story variants of a global theme, which arise from the interactions of virtual characters. As mentioned before, HTNs have been successfully introduced in these domains as an explicit representation for the set of all possible solutions or plans managed by the characters [3][7]. In order to manage the associated knowledge of the storytelling problem, it can be split in independent subtasks that will be managed by HTNs for the necessary decision taking of the virtual actors, which finally create the story.

We have presented the MinMin search domain in a similar problem in terms of the story created by the actor, although the funny dinner date problem introduces non-decomposability in its definition, which we consider an interesting feature for storytelling domains. Furthermore, as an autonomous behavioural system, MinMin give us the opportunity to evaluate different plans according with its order¹.

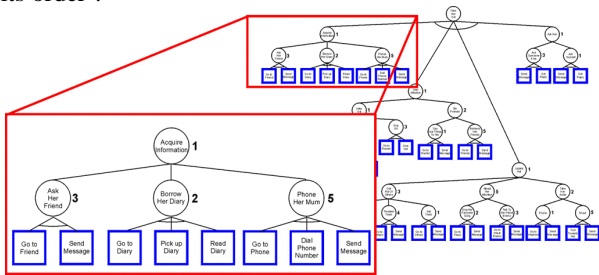


Figure 4. HTN representation for character behaviour

On the other hand, storytelling domains requires several features, such as, action failures and interactivity, still not covered by the MinMin domain presented. This section focuses on the design of a MinMin a ‘upgrade’ to be integrated in storytelling domains.

In order to help with this integration we use the sitcom example presented in [3]. In this example, the problem to be solved by the main character “Ross”, is to seduce another character “Rachel”. In order to this, “Ross” needs to acquire information about her, to gain her friendship, find a way to talk to her in private, etc. All the independent tasks are organised in a HTN, which represents the character behaviour (Figure 4).

Searching from the current agent state, the transition function of MinMin should provide the possible future

states as a result of the application of any operator (*success* or *fail*). To achieve believable behaviour in terms of a logical sequence of actions, a partial order can be easily introduced using the preconditions of the operators. For instance, we can guarantee that the final task will be to **ask her** (waiting for taking her out) including the necessary preconditions, such as, gain his attraction, isolate her, etc.

On the other hand, in STRIPS domains all the subtasks should be defined through the necessary operators. In this way, to acquire information we could define the operators showed in the Figure 5.

Operator::	ask-her-friend	
prec::	see-friend	1
effects::	acquire-info	1
actions::	goto-friend,	
	send-msg	
internal::	mood +3	
Operator::	borrow-her-diary	
prec::	see-rachel	0
effects::	acquire-info	1
actions::	goto-diary,	
	pick-up-diary,	
	read-diary	
internal::	mood +2	
Operator::	phone-her-mum	
prec::	nil	
effects::	acquire_info	1
actions::	go-to-phone,	
	dial-phone#,	
	send-msg	
internal::	mood +5	

Figure 5. Necessary operators to acquire info

To be able to animate the full story with MinMin we should require 19 operators to cover all the necessary motor actions. Furthermore, a list of *fluents* should be associated to each operator in order to manage the internal variables of the character, such as, mood, jealous, etc (Figure 5). For instance, to apply the operator *sing-her-favourite-song* the female character could be required isolated (preconditions), achieving her attraction (effect) and increasing its *mood* in *k* units. Once an operator is applied, the necessary internal variables will be updated, so that, they could be used at the horizon search level of MinMin together with the heuristics cited before. The search strategy introduced should provide an intelligent decision taking mechanism for storytelling characters. Figure 6 is a partially completed graph that would be created by the HSP, which would be equivalent to the explicit HTN representation.

¹ In storytelling this order is fixed off-line by the *author* and normally no alteration is allowed during execution, as an adequate technique for designing a manageable set of actor plans.

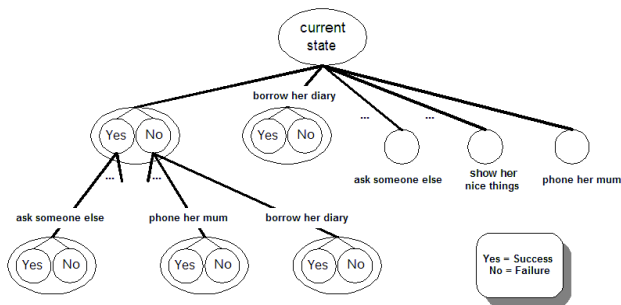


Figure 6. MinMin search design for storytelling domains

5 References

- [1] Bonet B, Geffner H. *Planning as Heuristic Search: New results*. Proceedings of ECP'99, pp.360-372.
- [2] Cavazza, M., Charles F. Mead, S. J. *Agents's interaction in virtual storytelling*. Proceedings of Third International Workshop on Intelligent Virtual Agents 2001. Madrid Spain.
- [3] Cavazza M., Charles F., Mead, S. J. *Interacting with virtual characters in Interactive Storytelling*. Proceedings of the Autonomous Agents Conf., AAMAS'02. Bologna, Italy, 2002.
- [4] Funge, J. *Cognitive Modeling for games and Animation*. Communications of the ACM, Vol 43. no.7, 2000.
- [5] Geib, C. *The intentional planning system: Itplans*. Proceedings of the 2nd Artificial Intelligence Planning Systems Conference, pp. 55-64, 1994
- [6] Korf, R.E. *Real-time heuristic search*. Artificial Intelligence, 42:2-3, pp. 189-211, 1990.
- [7] Nau, D.S., Smith, S.J.J., and Erol, K., 1998. Control Strategies in HTN Planning: Theory versus Practice. Proceedings of AAAI/IAAI-98, pp. 1127-1133.
- [8] Webber, B., Badler, N., Di Eugenio, B., Geib, C., Levison, L., and Moore, M., *Instructions, intentions and expectations*. Artificial Intelligence Journal. 73, pp. 253-269.