

# Search-based Planning: A Method for Character Behaviour

Miguel Lozano<sup>1</sup>, Steven Mead<sup>2</sup>, Marc Cavazza<sup>2</sup>, Fred Charles<sup>2</sup>

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

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

**Keyword:** Heuristic Search Planning, Search-Based AI, Virtual Actors.

## Abstract

In this paper we present experiments with search-based planning as the most generic description of intelligent behaviour for virtual actors. Using a MinMin heuristic search-based planner (HSP), we demonstrate how this can be used to generate efficient plans (in terms of plan length), and how the production of these minimum length plans is performed in times suitable for the real-time 3D virtual environments. Using an extension to the classical 'dinner-date' problem, we illustrate the planning and action execution undertaken by the virtual actor.

## 1. Introduction

Planning is the most generic AI technique to generate intelligent behaviour for virtual actors, in computer animation or computer games. In such domains, planning capabilities consist in finding a suitable sequence of actions that let an agent achieve pre-defined goals. Each action generated can be played in the environment to produce animation. Hence, entire animations can be generated from first principles, by defining a set of actions and allocating high-level goals to the character. It is not only possible 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][9]. Funge has used situation calculus to generate intelligent behaviours for virtual actors [4], and Cavazza has approached this problem with Hierarchical Task Networks (HTNs) for storytelling [2][3], 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:

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

This paper will present the domain representation and the behavioural animation problem proposed to integrate planning to drive character behaviour. Then a review of the key points associated to HSPs, and finally a

discussion of the results obtained by this integration describing a classical planning problem, which is also relevant to a character animation.

## 2. Planning for virtual actors

In animation domains, planning capabilities will consist in finding a right sequence of actions that let an agent achieve its goals, with the ‘added value’ of seeing the solution-plan carried out by it. Considering this, planning systems will provide actors with a general method to drive their intelligent behaviours, and visualized within the virtual environment in order to see how the agent can solve their virtual planning problems. For instance, intelligent agents in simulation systems could compute solution plans in response to user's instructions.

In the context we describe, plan optimality will not be an essential requirement, however we will be normally interested in minimum length plans, that is, the minimum sequence of actions that let the virtual actors to achieve their goals. The visualization of these minimum-length plans will display efficient and intelligent behaviour.

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
Faint	quiet	fun, not clean-hands
computer-work	clean-hands	not fun, not clean-hands, not work

**Table 1 - Planning problem example**

Experimenting with a Heuristic Search Planner (HSP), we provide an example using an extension to the classical dinner-date planning problem (funny-dinner-date - see Table 1). The actor must undertake a set of tasks in order to prepare a dinner date, such as removing the garbage, wrapping a present, etc. We have extended this

problem with more operators (*watch-tv, computer-work...*) but also with new goals and preconditions, such as to having the house clean and to be in appropriate mood for cooking (in our example fun).

Moreover, this scenario has similarities with the storytelling application as described in [3] and gives us an opportunity to investigate with a (non-decomposable, non-empty delete-lists) planning problem on a similar application.

HSP domains are mainly represented by three elements:

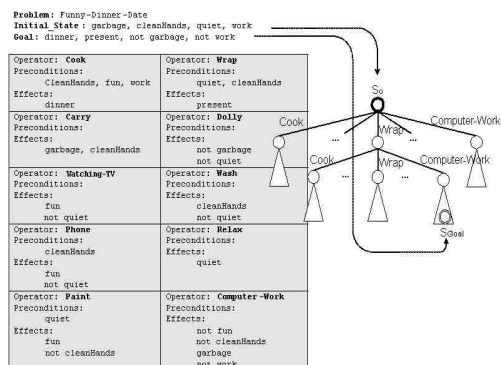
- i. The domain representation of the problem.
- ii. The search algorithm.
- iii. The heuristic function: In the next subsections, we will review the integration of these tree key elements in our behavioural animation domains.

### 2.1 The domain representation

Our agent-centred approach is based on the typical state-model representation for planning domains [1]. Each state contains 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. The states 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, such that, longer plans are often non-optimal in their action sequence. For instance, an agent who washes his hands before carrying out the garbage will have to

wash his hands again. 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 virtual actors 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 1. Initialisation of the start and goal states**

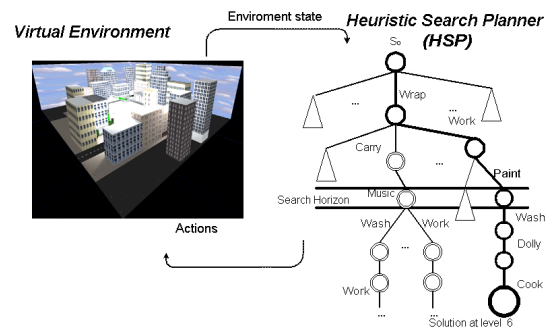
Taking into account the domain representation introduced, the next subsection will present MinMin as an adequate search algorithm to supply the planning requirements for our virtual actors.

## 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 [1], 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.

MinMin is capable of refining its solutions during the search using a dynamic depth-bounding criterion. As the plan-search progresses, a bounding factor  $d$  is maintained to keep track of the last best plan extracted (i.e. that with the minimum plan length). This bounding is also useful to overcome the main problem of MinMin, that of cycling. A secondary bounding criterion has been introduced to MinMin in order to improve its efficiency. This second bounding (2-B) simply detects the 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.



**Figure 2. Environment actions as operators in the MinMin search**

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. Figure 2 demonstrates the feedback from the environment as operators are carried out on stage by the virtual actor.

We are using the independent domain heuristics presented by Bonet&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. The Unreal™ engine performs low-level animation (movement, orientation, etc...) and visualization, however the animation system is under direct control of the planner. The planner and Unreal communicates via UDP, interfaced by the engine's scripting language UnrealScript™.

In 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 ( $d = 13$ ), MinMin finds 6 plan length solutions in a suitable time frame for the real-time performance requirements.

The agent will start searching from its initial state  $S_0$  using MinMin, and will obtain solutions or plans from 13 to 8-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.,  $S_0 - wrap - S_I$ ).



**Figure 3. Integrating search-based planning in 3D virtual environments**

As shown in Figure 3, once the virtual actor has executed an action, it should update its own internal state (eg.  $S_I = (S_0) + dolly$ ) performing future searches from this ( $S_I$ ), interleaving in this way planning and action execution, and achieving finally an intelligent autonomous behaviour able to reduce the distance to its goals. Figure 4 illustrates the search-plan carried out by MinMin to 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.

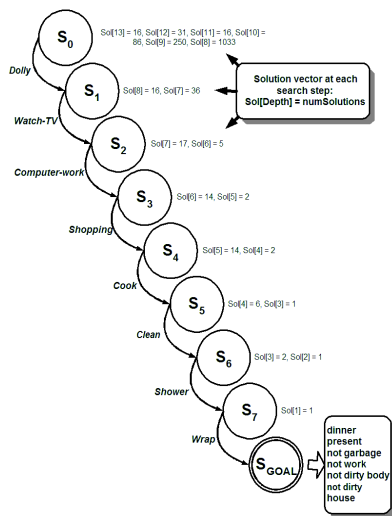


Figure 4 – Solution vector for plan

#### 4. Conclusions

We have described a specific approach to integrate fully search based planning behaviour for virtual actors. Performance of the planning system has shown good potential for scaling-up on simulation tests. Our future work will be oriented to include enlarging the set of operators available and uncertain information from the environment in a more complex visual planning problem, so that, a complete intelligent virtual agent architecture could be tested in 3D virtual environmental simulations.

#### 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. Pemberton, J.C. and Korf, R.E., *Incremental Search Algorithms for Real-Time Decision Making*. Proceedings of the 2nd Artificial Intelligence Planning Systems Conference (AIPS-94).
8. Tsuneto, R., Nau, D. and Hendler, J., *Plan-Refinement Strategies and Search-Space Size*. Proceedings of the European Conference on Planning, pp. 414-426.
9. 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.