Universitat Rovira i Virgili

Master's Thesis

Masters of Artificial Intelligence and Computer Security

Behavior approximation using fuzzy-genetic systems

Mark Safronov

August 3, 2025

Contents

1 Introduction

In this section we'll set up the context of a problem to be solved and explain an important comparison we make in this work with the reinforcement learning. Readers which don't require such an introduction can proceed directly to the 2.

1.1 A game solver as a scheduling problem

In 1993 Studio Gainax in Japan released a computer game called *Princess Maker 2*. The game belongs to a so-called "life simulation" genre, the player takes the role of a guardian of a young girl, making decisions that affect her upbringing and future. *Princess Maker 2* is a narrative-heavy game, featuring a very complex storyline which depends on the choices the player makes, but under the plot and all the other art elements which normally constitute a computer game, lies very formal and routine gameplay loop:

1. The player estimates the current state of the girl;

2. The player chooses the day-to-day schedule for the girl;

3. The girl "performs" the scheduled actions;

4. The game changes the state of the girl according to the actions performed.

Story-wise, the game ends after ten years of upbringing the girl. At this point the game evaluates the state reached by the girl and tells the final "fate" she got as the result. The title of the game being the hint of the ultimate goal.

This is essentially a stateful agent planning its behavior according to the predefined goal to be reached. Thus, the question arises: can we solve it?

1.2 Comparison with Reinforcement learning

Reinforcement Learning [6] is used in this work as a benchmark. The choice of this optimization method is deceptively obvious, because at a glance, the problem looks like a perfect match for it. We have an agent, which has a state, and this agent can perform actions which change the state. Ultimately we want the agent to reach the goal state which will give it the best reward.

The issue with applying the RL methods to this problem is the issue of scale.

The origin problem of solving Princess Maker 2, described in the previous subsection, assumes 25 actions over a state space of 50 numeric characteristics each one having values between 0 and 500. Just enumerating the possible states of the agent caused by these actions in the state space of such a size is an intractable problem, which we will rigorously show in **??**. Just the rewards table for this has a size of 25×500^{50}, which is already practically intractable.

Moreover, the most important problem is the length of the process. Following the base example of *Princess Maker 2* gameplay, player makes 3 choices per virtual "month", and the game spans 10 "years", so the search space is a tree $3 \times 12 \times 10 = 360$ levels deep.

2 Formal problem statement

2.1 Actor behavior as a control problem

Formally we have a choice of whether to treat this as a planning problem or a control problem.

We will define "behavior" as the control problem.

Assuming we have a character described as a set of numeric characteristics

$$\mathbf{x} \in \mathbb{Z}^n \tag{1}$$

we have a set of possible actions

$$A = \{a_1, a_2, \ldots, a_m\} \tag{2}$$

which collectively form a transfer function

$$f(\mathbf{x}, a) = \mathbf{x}' \tag{3}$$

To describe the desired outcome, we first declare a fitness function mapping the state to a numerical value:

$$\Phi : \mathbf{x}' \to \mathbb{R} \tag{4}$$

a goal fitness value

$$\mathbf{G} \in \mathbb{R} \tag{5}$$

and a planning horizon

$$T \in \mathbb{Z} \tag{6}$$

We want to get an ordered actions sequence of length T which will lead $\mathbf{x}$ to some $\mathbf{x}^*$:

$$\mathbf{a} \in A^T, x_o = \mathbf{x} : \bigodot_{i=1}^{T} f(x, a_i) = \mathbf{x}^* \tag{7}$$

(where $\bigodot$ is a fold operator)
such as:

$$\Phi(\mathbf{x}^*) > \mathbf{G} \tag{8}$$

The transfer function f is assumed to be completely determined, and the whole process being non-stochastic. This is a significant restriction which cannot be lifted for the proposed solution to work.

2.2 Dimensionality explosion stemming from the original context

Normally, there is a number of classical approaches to this problem, including dynamic programming, reinforcement learning and state space searches using graph theory. However, in this work we'll focus specifically on the cases which lead to combinatorial explosion for classical solutions, that is, when we have sufficiently large amount of characteristics, actions to choose from and most importantly, very large planning horizon:

$$n > 50 \tag{9}$$
$$m > 20 \tag{10}$$
$$T > 300 \tag{11}$$

As an additional restriction, we'll state that it's not enough to reach the desired state $\mathbf{x}^*$ sometime before T, instead, it is a must that we perform T actions and only then evaluate the final state $\mathbf{x}'$ of the character. Having this distinction is important, as actions result in *arbitrary* changes in the character state, both positive and negative. So, it is possible, but unacceptable, to reach the desired state $\mathbf{x}^*$ before T and after that perform a number of actions which will lead to a final state $\mathbf{x}'$ which is not equal to $\mathbf{x}^*$.

We assume a fixed-length trajectory of actions, each of which transforms the state of the system according to a known deterministic transfer function. While the agent cannot avoid taking actions — and hence cannot avoid changes to the system — it is allowed to evaluate its progress toward the goal at every intermediate state. In this sense, the problem is not a pure planning task but an episode-based control problem with delayed evaluation.

We define the reward function as a continuous measure of proximity to the goal state, such as an ℓ_2-norm to a target vector or distance to a target region described by bounded inequalities. While the final reward is evaluated only after steps, the intermediate distances can serve as feedback for heuristic guidance or surrogate loss during optimization.

Importantly, the system does not support "no-op" actions or any idle transitions. Each step must result in a meaningful state transformation, reflecting the irreversible nature of time and action in real-world analogs such as life planning.

With these restrictions in place, a need in an heuristic arises to perform efficient search in the state space, as its size becomes unrealistically large.

3 Proposed solution approach

Classical reinforcement learning methods become intractable in this domain due to the high dimensionality of the state space, large action set, and long planning horizon. Moreover, the inability to halt or take neutral actions further exacerbates the combinatorial explosion of the trajectory space.

To address this, we introduce a heuristic dimensionality reduction via the concept of inclinations — latent behavioral parameters — and model the behavior policy as a fuzzy controller which maps the current state and inclinations to a concrete action.

This parametrization constrains the space of possible behaviors, making the optimization tractable. Instead of learning or searching over action sequences directly, we perform optimization in the significantly smaller space of inclinations, evaluating the final outcome after T steps. The resulting problem becomes an offline, black-box control task — suitable for evolutionary algorithms, rather than classical RL methods.

3.1 Using domain knowledge to reduce dimensionality

In this work we evaluate one possible approach which is defined as follows.

Let's assume that we can segment the set of possible actions to clusters with the following particularities:

1. actions in the same cluster lead to "similar" changes in the character state $\mathbf{x}$.
2. the cluster as a whole can be described symbolically

In this case we can synthesize a set of numeric characteristics which we'll call "inclinations":

$$\mathbf{I} \in \mathbb{Z}^q \tag{12}$$

$$q << n \tag{13}$$

From this, we can define a set of fuzzy rules[4] mapping the inclinations to action choices:

1. if an inclination I_i has a fuzzy value V_I,
2. and the current state $\mathbf{x}$ has fuzzy values V_i^x
3. then P_a, the priority of an action a, is a fuzzy set V_A.

After the defuzzification of all the inferred fuzzy values P_a we select an action with the highest priority.

The selection and design of fuzzy rules is a critical aspect of this approach. In this thesis, the fuzzy rule base is constructed manually, leveraging domain knowledge to define the mapping from inclinations to action priorities. Future research may investigate automated methods for generating fuzzy rules, such as clustering or machine learning techniques, to further improve scalability and reduce manual effort.

While it is theoretically possible to define fuzzy rules that map every possible inclination vector $\mathbf{I}$ or even every state $\mathbf{x}$ to action priorities, such exhaustive rule sets would quickly become infeasible due to combinatorial growth. This reinforces the importance of dimensionality reduction and clustering in making the fuzzy-genetic approach tractable for high-dimensional planning problems.

The assumption which we explore among others in this work is the practical possibility to write a coherent set of fuzzy rules which will be clustered around the clusters of actions, and each inclination will tend to map to its own cluster of actions.

Now, using such a fuzzy controller $\xi(I, \mathbf{x})$ we can construct the goal function:

$$g(I, \mathbf{x}) = \bigodot_{i=1}^{T} f(x, \xi(I, x_i)) \tag{14}$$

the above formula being subject to improvements in expressiveness, the main point of which being the fuzzy controller $\xi(I, x_i)$ selecting the action to perform on the step i according to the inclinations and (ideally) the current state x_i.

The argument $\mathbf{x}$ is essentially a constant for both (7) and (14). As that the transfer function f is non-stochastic, $\mathbf{I}$ uniquely maps to the actions sequence $\mathbf{a}$. Thus, given (13), we effectively performed dimensionality reduction on the original problem.

We can find $\arg\max(g)$ now using an appropriate optimization method. For this work, because of a strong biosocial analogies a genetic algorithm[2] was chosen, with the vector of inclinations $\mathbf{I}$ as a chromosome.

3.2 Hypothesis

The hypothesis explored in this work is that the combination of assumptions described above constructs an heuristic which allows solving the problem more efficiently than performing the reinforcement learning directly.

3.3 Objectives

The objective of this thesis is to investigate whether fuzzy-genetic heuristics can effectively solve high-dimensional deterministic planning problems through dimensionality reduction and symbolic reasoning.

In particular, we aim to:

1. Formalize the problem as an optimization task.
2. Implement a working solver.
3. Evaluate the performance of the solver on a set of test cases of increasing complexity.
4. Analyze the results to draw conclusions about the effectiveness of the approach.

4 Methodology

4.1 Encoding domain knowledge of actions as a fuzzy controller

The major benefit and the core reason for the fuzzy controller is the ability to encode the domain knowledge in a limited set of rules which will be formally processed.

Compared to, for example, some of the reinforcement learning methods, we don't need to specify the full table of rewards for every possible action-state combination. It is enough to

specify one rule for every available action and the controller will already become fully functional. With some configuration of rules it's possible to write even less of them.

This allows to simplify the implementation of the solver, because one of the main weaknesses of the proposed solution is writing the fuzzy rules by hand.

4.2 Control feedback loop as a fitness function

A single trajectory in the action space is explored using the following process.

1. We start with the initial state $\mathbf{x}_0$ and the given set of inclinations $\mathbf{I}^k$
2. We evaluate both $\mathbf{x}_0$ and $\mathbf{I}^k$ with the preconfigured fuzzy controller
3. The defuzzified output of the controller is the set of priorities for all the actions. We pick the action with the highest priority. Tiebreaker is the position of the action in the list.
4. Action is executed and if we haven't made T actions yet we return to the step 2
5. After T executed actions we apply the goal conditions predicate $\Phi(\mathbf{x}*)$ and calculate the fitness based on that.

4.3 Global optimization using an evolutionary algorithm

Strong biosocial analogies and the configuration of the control loop from 4.2 suggest us to use the evolutionary algorithms for optimization. This is what would be used in this work. However, any algorithm which is able to use the concept of fitness function would be applicable here.

The choice of the implementation for the genetic algorithm guided this section. The library Pagmo [1] includes a lot of already implemented different evolutionary algorithms apart from the simple genetic algorithm so it enables us easier exploration of possibilities in optimizing the full solver.

5 Implementation

The technical implementation of the method is performed in C++ [5] using the libraries FuzzyLite [3] and Pagmo [1]

5.1 Choice of a C++ language as foundation

5.2 Fuzzylite library for the fuzzy controller implementation

5.3 Pagmo library for evolutionary computations

6 Experiments and Results

In the scope of this work we'll use the original Princess Maker 2 problem but segmented in four different problems of increasing scale.

6.1 Trivial case

2 characteristics, 4 mutually exclusive actions, 3 steps.

This scenario represents a trivial case, with only 4^3 possible action sequences—a total of 64. The small state space allows for exhaustive enumeration and manual verification of results. This case serves to validate the correctness of the implementation and the fuzzy controller, as the system's behavior can be easily traced and analyzed by hand.

6.2 Base control case

4 characteristics, 12 actions, 100 steps.

This case is the base case, as it introduces enough complexity to test the proposed approach and at the same time compare it with classical approaches.

A decision tree of the size 12^{100} is already too large to be completely enumerated.

However, with 4 characteristics and 12 actions, the problem is still well within the range where Reinforcement Learning methods—especially those using function approximation—can be applied efficiently. The planning horizon of 100 steps is long enough to be non-trivial, but does not pose significant challenges for standard RL algorithms.

6.3 Origin case

The complete Princess Maker 2 case is a problem with 50 numeric characteristics of a character and 25 actions to choose from, with a planning horizon of 360 steps.

This case is an attempt to directly solve the original problem which started this work. It will be used as a benchmark for the proposed solution on a real-world problem.

7 Discussion

(TBD)

8 Conclusions and Future Work

Despite the context of the problem being a computer game, the problem itself is a general one, and the proposed approach can be applied to any high-dimensional deterministic planning problem. Which constitutes the core value of this work.

In the span of this work, only three distinct cases were explored, and the more thorough exploration of the parameter space is left for a dissertation-level research.

References

[1] Francesco Biscani and Dario Izzo. "A parallel global multiobjective framework for optimization: pagmo". In: *Journal of Open Source Software* 5.53 (2020), p. 2338. DOI: 10.21105/joss.02338. URL: https://doi.org/10.21105/joss.02338.

[2] Melanie Mitchell. *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press, 1999. ISBN: 9780262631853.

[3] Juan Rada-Vilela. *The FuzzyLite Libraries for Fuzzy Logic Control*. 2018. URL: https://fuzzylite.com.

[4] S. Kumar Ray. *Soft Computing and Its Applications, Volume II*. Boca Raton, FL: CRC Press, 2014. ISBN: 978-1-4822-5793-9.

[5] Bjarne Stroustrup. *Programming: Principles and Practice Using C++, 3rd Edition*. 3rd ed. Available online. Addison-Wesley Professional, 2024. ISBN: 978-0-13-830868-1. URL: https://www.informit.com/store/programming-principles-and-practice-using-c-plus-plus-9780138308681.

[6] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd ed. Available online. MIT Press, 2018. ISBN: 978-0-262-03924-6. URL: http://incompleteideas.net/book/the-book-2nd.html.