Proceedings of the Fifteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-19)

Towards Usable Level PCG

Eric Lang
University of Utah
201 Presidents' Cir
Salt Lake City, Utah 84112
ewlang@cs.utah.edu

Abstract

My proposed dissertation work attempts to make procedural content generation (PCG) for game levels easier to use and more expressive for designers. This can be split into three rough layers: improving the expressive range of a dungeon PCG genetic algorithm by using multiobjective optimization, providing a more intuitive way for designers to create fitness variables, and letting them train the PCG system to give them the output they expect. While this work will focus on PCG algorithms for generating game levels, ideally much of the evaluation concept in this paper will be applicable to other forms of PCG. I will hopefully also be able to show that having the designer interact with an expressive PCG system as their "apprentice" is effective.

I choose search-based PCG for this work because these algorithms take an unknown solution space and move generated solutions toward optimal solutions as they run. Integral to this is the idea that variables have an optimization direction; all else being equal, a solution with value 1 in variable A is defined to be better or worse than another solution with value 2 in variable A depending on whether the variable is minimized or maximized. Not only is this approach useful for keeping variables easy to understand without requiring strict constraints, but in a multiobjective approach, it also allows the algorithm to explore the tradeoff between variables and find solutions that would be hard to discover without optimization.

Search-based PCG methods are a relatively well researched area (Togelius et al. 2011). Other research has been done investigating the effectiveness of using genetic algorithms to search for optimal solutions in generating dungeon levels (Valtchanov and Brown 2012). Many operate in mixed-initiative modes with the designer in the loop to improve the designer's control (Liapis, Yannakakis, and Togelius 2013) (Alvarez et al. 2018) (Baldwin et al. 2017). Researchers acknowledge that genetic algorithms are difficult for designers to work with directly because designing a fitness function is not trivial (van der Linden, Lopes, and Bidarra 2014). As such, one group has already attempted to incorporate multiobjective evolutionary algorithms in level PCG, though they evaluated their output primarily by inspecting the relationships between the objectives or, in other words, the shape of the Pareto front (Togelius et al. 2010) (Togelius, Preuss, and Yannakakis 2010). Another group has explored using a multiobjective evolutionary algorithm for game balance (Volz, Rudolph, and Naujoks 2016).

In this summary, I describe my proposed doctoral work to give designers control over PCG algorithms by using multiobjective optimization to avoid some of the burden of knowledge required to invent a fitness function and to improve the expressive range of the algorithm when compared to a single-objective approach. I also suggest a method by which designers can create fitness variables more intuitively by building them from layers of simpler terms. Finally, I plan to determine whether having the designer teach the genetic algorithm how best to optimize the levels is an effective approach.

Plan of Work

The problem can be split into three stages: the creation of an expressive level PCG genetic algorithm, giving designers generalized tools to create fitness variables from the output data, and creating the "apprentice" system whereby the designer trains the system to produce their desired level.

PCG System

In order to address this problem initially, I believe an end-to-end approach is necessary. I must have a PCG system as flexible as possible to provide the designers control over the system.

A multiobjective evolutionary algorithm should be capable of having a greater expressive range than an approach using a simple fitness function. While an approach with a single fitness function may produce a single result that has higher fitness in fewer runs than in a multiobjective approach, the multiobjective approach will investigate a wider range of solutions as it targets the entire Pareto front for optimization. The output of a multiobjective algorithm can present many options that optimize the given variables rather than only a single solution of the highest fitness.

One other candidate approach would be a quality-diversity (QD) algorithm (Gravina et al. 2019) (Preuss, Liapis, and Togelius 2014). While QD algorithms are very similar to multiobjective genetic algorithms in many ways, they require some form of diversity measurement and break up the solution space based on this diversity to generate multiple solutions that are each guaranteed to satisfy some minimum distance from other solutions. My primary rationale for using a multiobjective genetic algorithm rather than a QD algorithm is to try to explore the entire range of optimal solutions to present them to the designer, but once the designer has taught the system what it wants, the system may only provide one level per run for its final purpose. However, I acknowledge that training the system will be difficult if it produces solutions which are too similar, so I will need to ensure that my proposed method is effective at finding diverse solutions.

I have already created a PCG method based on Valtchanov and Brown's (2012) work with a genetic algorithm, though my approach uses multiobjective optimization. Like Togelius et al. (2010), I believe that multiobjective optimization allows for an algorithm to handle conflicting variables while also reducing the burden on the designer to combine variables together to form a rigid fitness function, though it does require additional designer evaluation as not all members of the Pareto front will be valuable. This additional evaluation is discussed briefly in the "PCG as an Apprentice" section.

Creating Fitness Variables

For designers to be able to create a wide variety of fitness variables, they need to be presented with the abstract data that the PCG system uses for evaluation and be able to manipulate that data to measure whatever they desire. I believe I can leverage the designer's existing perspective on classifying levels to aid in this step. While the designer will still need to create the fitness variables, the system should let the designer be granular in their approach to keep individual variables fairly simple.

Preferably, this system would use a tiered "vocabulary". The designer would be able to specify, for example, a "Connectivity" variable that would simply average the number of occupied doors across some given set of tiles. They may also be able to make a "Linearity" variable that penalizes dead ends, puts a bonus on hallway rooms, but applies a small penalty for doors past two. Then, the designer could specify a "Clustered" variable which is sections of high "Connectivity" interspersed between sections of high "Linearity", resulting in a level with clusters of dense tiles connected by more linear pathways. The necessity of making this a tiered or layered system are clear: in order for the designer to create the "Clustered" variable, it should be built on the "Connectivity" and "Linearity" variables. My eventual goal is to allow the designer to create a variable for an abstract concept like "Pace" or "Tension" in a PCG system by using multiple layers of other, less abstract variables, and be able to see an evaluation of levels based on their defined variables.

Ultimately, the idea of creating and layering variables could be useful for automatic evaluation in many contexts. However, the actual variable creation system created in this specific work may not be useful outside of the limited scope of PCG algorithms that generate content represented by graphs. If the PCG output can be described by units that have relationships with other units, then an evaluation system which simply takes abstracted units, relationships, and property sets as input could work for any of these systems.

PCG as an Apprentice

While a multiobjective algorithm will present many non-dominated solutions which optimize the given variables, the entire range of the Pareto front may not be suitable to the designer. Namely, simply because a level exhibits a high value of one variable despite exhibiting a low value of some other variable doesn't mean the level is necessarily what the designer desires, even though such cases normally happen in multiobjective optimization. Clearly, the designer needs to be able to specify their own expectations. Once these expectations are understood by the system, the system can produce individual levels that meet these expectations.

When the algorithm runs, the designer trains the multiobjective optimization to find the right balance of fitness variables. The designer would begin my simply defining the fitness variables and expressing whether they were positive or negative. Upon running the system, it would output a set of levels across the Pareto front for all of the designer's variables. The designer could then express what they don't like about certain output levels in terms of their own variables, informing the PCG algorithm of how to be more targeted in its multiobjective search. There is further work to be done to determine exactly how the algorithm should adjust based on the designer's feedback, but in its simplest and most basic form, the designer could point out levels where certain variables were too high or too low and the algorithm could constrain the allowed ranges of those variables. Variables could then be automatically weighted and combined based on the feedback from the designer. Eventually, if the designer's feedback is specific enough, the algorithm may become similar or identical to a single-objective approach (i.e. using a fitness function). Even if this is the final result, the value in using multiobjective optimization would be to find a function without the designer needing intricate knowledge of how the variables interact.

This proposed system would hopefully give the designer a better understanding of how the evaluation works. In essence, this apprenticeship step makes the system more intuitive. A level designer is likely not trained in how to arbitrarily combine different or conflicting fitness values that are defined by an abstract evaluation function, but they should be able to determine whether a level is within expectations. I should be able to determine the effectiveness of this approach by running a study with the system on designers. Ultimately, if I can leverage a level designer's preexisting expertise on evaluating levels, I should have more success in creating levels that they would consider good.

Research Contributions

- Using multiobjective optimization to improve expressivity
- Allowing designers to create an evaluation vocabulary
- Creating a system where designers use PCG as an "apprentice"

References

Alvarez, A.; Dahlskog, S.; Font, J.; Holmberg, J.; Nolasco, C.; and Österman, A. 2018. Fostering creativity in the mixed-initiative evolutionary dungeon designer. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, FDG '18, 50:1–50:8. New York, NY, USA: ACM.

Baldwin, A.; Dahlskog, S.; Font, J. M.; and Holmberg, J. 2017. Towards pattern-based mixed-initiative dungeon generation. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, FDG '17, 74:1–74:10. New York, NY, USA: ACM.

Gravina, D.; Khalifa, A.; Liapis, A.; Togelius, J.; and Yannakakis, G. N. 2019. Procedural content generation through quality diversity.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2013. Sentient sketchbook: Computer-aided game level authoring. In *FDG*.

Preuss, M.; Liapis, A.; and Togelius, J. 2014. Searching for good and diverse game levels. In *2014 IEEE Conference on Computational Intelligence and Games*, 1–8.

Togelius, J.; Preuss, M.; Beume, N.; Wessing, S.; Hagelbäck, J.; and Yannakakis, G. N. 2010. Multiobjective exploration of the starcraft map space. In *Proceedings of the 2010 IEEE Conference on Computational Intelligence and Games*, 265–272.

Togelius, J.; Yannakakis, G. N.; Stanley, K. O.; and Browne, C. 2011. Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games* 3(3):172–186.

Togelius, J.; Preuss, M.; and Yannakakis, G. N. 2010. Towards multiobjective procedural map generation. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, PCGames '10, 3:1–3:8. New York, NY, USA: ACM.

Valtchanov, V., and Brown, J. A. 2012. Evolving dungeon crawler levels with relative placement. In *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering*, C3S2E '12, 27–35. New York, NY, USA: ACM.

van der Linden, R.; Lopes, R.; and Bidarra, R. 2014. Procedural generation of dungeons. *IEEE Transactions on Computational Intelligence and AI in Games* 6(1):78–89.

Volz, V.; Rudolph, G.; and Naujoks, B. 2016. Demonstrating the feasibility of automatic game balancing. *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference - GECCO '16*.