RAWR: Algorithm for Dungeons and Dragons Strategy Optimization

Valerie Pietrasz[[1]](#footnote-1) and Thomas White[[2]](#footnote-2)

*Stanford University, Stanford, CA, 94305*

**This paper investigates a problem of critical importance for engineers and computer scientists everywhere – the optimal strategy for 5th Edition Dungeons and Dragons combat encounters. Dungeons and Dragons is an excellent form for investigating decision-making strategy, since it is an environment that is uncertain, but also operates according to known rules. Although much of the game is based on storytelling and therefore out of the scope of decision-making algorithms, it is founded on a robust system for the resolution of combat encounters. Randomized Actions With Rewards (RAWR) attempts to algorithmically solve the optimal combat strategy using an online planning approach. In addition, by creating an easily-extensible framework that implements these encounter rules, it seeks to create a structure that can be used easily for further investigations.**

1. **Nomenclature**
2. **Introduction**

Dungeons and Dragons (DnD) is a pen-and-paper role-playing game with a great deal of complexity that is impossible to model, because it relies on storytelling, social dynamics and worldbuilding for much of its appeal. However, within the broader system is a very clearly defined mechanism for simulating combat encounters. Characters fight foes, making choices about a predetermined set of actions they can take and how each one of them affects a set of metrics governing the stamina and capabilities of their foe. This project seeks to simulate a very simplified model of a combat encounter in the DnD 5e ruleset, beginning with an encounter in between an agent and a single foe, in which the agent seeks to identify a strategy that minimizes damage to itself while still eliminating its foe.

This problem is of interest for both practical and academic reasons. Practically, DnD is a common pastime the world over. Simulation of the optimal strategy given a certain framing of an encounter can provide perspective for players looking to hone their game; in addition, although this project only provides a foundation for such an effort, it can ultimately be used as a resource by people attempting to decide if a character or weapon they have created for DnD’s system is appropriately challenging, by providing a quick way to determine if an average player would be suitably challenged by it. This could one day be a great resource for game designers looking to iterate quickly.

Academically, DnD combat is an excellent example of a decision-making problem, because of the several levels of uncertainty lead to complex optimization and the large scope of both possible decisions and possible states prioritize online planning and other algorithms which can save computation time. As such, it is an interesting way to investigate the generalized process of solving Markov Decision Process problems.

1. **Problem Statement**

This project seeks to find an optimum policy for a combat encounter. The encounter is framed by combat in between an agent – representing the player and thus the decision-maker and the foe – representing an antagonist usually controlled by the dungeon master (DM). Success or failure in the encounter is tracked by both party’s health points (HP) – when these reach zero the party dies. As such, the optimum policy is framed such that the agent does as much damage as possible while sustaining as little damage as possible, with a large reward for killing the foe and a large penalty for dying itself.

In pursuit of this goal, both parties can take a variety of actions. Some actions directly lower the HP or their antagonist or raise their own HP, but others impact the battle more indirectly, by raising or lowering each side’s potential to do damage in the future. Such trade-offs in between present and future reward are common in DnD – for example, a player might be able to either lower the damage that they take, or increase the damage that they give out, but not both. Crucially, this also means that the list of available actions for both parties actually changes throughout the simulation.

Once both parties have settled on an action, they are settled according to a slightly randomized process. Both parties generate a random number (in an actual game, through the use of throwing dice), boost the random numbers with modifiers derived from their attributes or actions, and then compare the random numbers. If the imposing force is larger, it then makes a slightly random impact on its target’s status. Thus, there are two main sources of randomness within the action resolution – whether the target is able to resist the action entierly, and then the magnitude of the effect. This outcome uncertainty constitutes the first mains source of uncertainty in the simulation. In addition, because the agent does not know the ability scores of the foe, upon which its modifiers are predicated, there is a small element of parameter uncertainty as well.

The second source of uncertainty comes from the state of the foe. In both DnD and in this simulation, the foe’s state is unknown to the player, but sometimes communicated through very noisy measurements. In a normal game, this is in the form of verbal description by the DM, which provides a qualitative sense of the foe’s status. In this simulation, we have chosen to implement this knowledge as a noisy measurement.

Finally, the actions of the foe are not deterministic. In traditional DnD, they would be made based on the DM’s discretion. Although they are at least somewhat focused on eliminating the agent, the DM may have other narrative motivations unknown to the player, making it difficult for the player to always assume that their foe will take the action of maximum utility. As such, we have implemented a weighted function for the foe’s actions – it will sometimes take the action of maximum utility, but other times will choose an action randomly regardless of utility.

1. **Approach**

From previous experience, we chose to design the programming environment in Python, which provided a useful optimum in between broad commonality and accessibility and power.

To solve the problem, the first goal was to implement a complete model of the encounter that can be easily accessed by optimization algorithms. This had four key elements.

First was the creation of agent and foe classes. This allowed the tracking of a single agent or foe entity and their states, determining both their current status and what actions are available to them. Since all of that information is available in the function, the classes also implement functions that check their available actions, interface with their state, and construct their functions. As such, they are framed such that each and foe is fully replaceable – future implementations of the code could easily try out alternative agents and foes to see how they behave while trusting that their implementation throughout the code works smoothly. This is key to the long-term applicability of code like this one to experimentation with different configurations.

Secondly, we implemented a generalized function for handling actions. As discussed in the introduction, DnD actions follow an extremely general recipe, as above: a party takes an action directed at a target, the target attempts to resist it, both generate a random number and compare them, and then a weighted random effect impacts on of the target’s stats. We created a class for actions that can be initialized with the details of the implementation – such as the actor, the target, the particular dice to be rolled, and so on. This way, actions can be called using a single line of details inside the agent and foe, allowing all of the specific details about what they can or can’t do to remain inside their class functions, while preserving an extensible structure for actually handling the outcomes of those actions.

Third was a system for uniquely parameterizing states. During a DnD encounter, the state is uniquely described by the state of the agent and foe combined, each of which have a number of different state variables, which may even be of different types. To facilitate the ultimate parametrization of a value function, we created an algorithm to hash that information into a single integer, which can then be repeatably unpacked into an agent and foe state. By allowing the state of the entire simulation to be more easily stored and implemented, it became easier to work with the code.

Finally, a top-level set of functions automated the process of taking turns. In real DnD rules, there is an element of randomness in whether the agent or foe can act first, but in this simulation, for simplicity, the agent always does. Our structure made it easy to call a turn and have both parties resolve their actions. Because the agent does not actually know the state of the foe, an extra detail was added – a second foe object to represent the agent’s belief, named the faux foe for clarity and alliterative pleasantness, which is updated with noise based on the state of the foe. As mentioned above, in a real game, the player’s understanding of the foe’s state would be based on qualitative descriptions by the DM – one of the only intrusions of such qualitative reasoning into the otherwise very rules-bound approach for DnD combat. To add a small amount of color, a knowingly slightly inefficient approach to communicating this information was implemented – upon being damaged, the foe responds with printed cries of pain, which the agent can noisily interpret to identify the foe’s remaining HP.

Once this structure was complete, the next step was to figure out how to implement decision-making and optimize over it. The first approach we considered was Monte-Carlo optimization across the space of actions, ultimately converting to a deterministic policy with the maximum utility. However, ultimately this was deprioritized against a branch-and-bound based online planning approach. Online planning has the advantage of considerably reduced computing time, especially in a game where the number of possible states is extremely large but many of them are unreachable. It also more accurately simulates the process of DnD play. Given the very high uncertainty, which piles on during play, as well as constantly changing state itself, DnD is not especially conducive to a single deterministic ‘optimum’ policy in any case. A program that could more easily adapt to the decisions made by the actors within it was judged much more useful and accurate as a simulation.

Finally, a top-level implantation of an encounter rounded out the package – this encounter structured the optimization process and how it interacted with the underlying structure of the problem. Jupyter notebooks were used to graph the results and served as a high-level interface.

1. **Detailed Implementation**

As is usually the case, the code required a number of changes of direction during implementation. For instructive reference, some of the important ones are detailed below.

After some thought, it turned out that branch and bound was not as useful for this application. Traditionally, the goal with and bound is to lower the computation time with respect to forwards search by pruning some tree directions that fall below the minimum utility of a different branch. However, setting constructive bounds on the utility or action value function proved difficult, especially because of the different models of utility that different actions implemented – some actions were not even available some states, and when they were, many did not directly provide utility but instead just gave other actions additional utility. Given these concerns, the branch and bound algorithm was unlikely to be optimum and unlikely to help very much in computation, leading to a scoping change to a simpler forwards search algorithm.

The dungeon state became a point of contention. It was originally included to make it possible to parameterize the state. However, the transition away from Monte Carlo methods made the need to create a parametric state vector much less pressing. And since every opportunity to use it tended to immediate require either the agent or foe attributes to be unzipped in any case, it seemed to largely only add computation time. Future extensions could find this capability very useful – for example, if past runs were to be stored or transmitted easily. However, in the current implementation, the code ultimately turned out to not have much purpose.

At this point, it became clear that a lot of iteration was needed on the hyperparameters, many of which hasn’t seemed to be hyperparameters at the time. For example, the depth of the forwards search and many of the specifics of the agent and foe became things to be played with. To facilitate easy access to all of this work from the top-level function, pandas data frames and a structure of keyword arguments were implemented, so that all the hyperparameters can easily be set from the top level without a requirement to dig through the code.

1. **Results**
2. **Future Work**

The authors are very excited about the current outcome of the work. The current program simulates the realistic behavior of an agent and foe, and appears to do so approximately as well as an extremely mediocre human player. This is an important first step to a complete analysis of the DnD problem. The main future step is to create capabilities for handling complexity in DnD not currently modeled.

Most crucial are two elements: movement and multiagent behavior. The first is structural. Movement is a key element of DnD – not to mention any semblance of real combat –and composes some of the most interesting strategy. By moving about a play field, both agents and foes can restrict the actions available to their antagonists, or boost their own options. Given the complexity of implementing geometrical movement of this type, it was scrapped in the current version of RAWR, but its inclusion in a future version will greatly increase the efficacy of the simulation.

Secondly, and even more crucially, DnD is a game that thrives on group, not solo play. The current implementation has only a single foe, while real DnD usually has several. But, even more crucially, it also only has a single agent, whereas much of the strategy of real DnD lies in the interactions between several agents, and how, working as a team, they might be able to emphasize their relative strengths and make up for their weaknesses. A multi-agent model could serve as a crude model for this behavior, although obviously without the players having an ability to explicitly communicate about strategy, and the basic structure of our model of DnD is extensible to this possibility. However, like movement rules, it was judged ultimately too complex to implement in the current implementation.

Finally, future work could be useful in further exploring the space of optimization algorithms. More advanced forms of online learning and optimization could provide an even more utility-maximizing result, potentially in less time. Thankfully, the structure is robust enough that other approaches could be built atop it.

1. **Conclusion**

Our program’s first implantation, named Random Action with Rewards (RAWR), is currently capable of simulating a single simulated encounter with a single agent and foe. It provides the fundamental proof of concept and building blocks for much better future implementations. The authors are excited for the forthcoming field of DnD research that they hope will develop from their work, and can’t wait to experience future researcher’s findings.

**Acknowledgments**

Many thanks to Mykel Kochenderfer, the teaching team, and everybody else who helped the authors understand decision making uncer uncertainty. They would also like to thank the creators of Stack Overflow and some of the contributors, who helped the authors understand everything else. And many thanks to Gary Gygax and David Arneson for developing the system that has entranced so many of the authors’ weekends.

**References**

I left the references section the way that it is so that we can refer to it when formatting stuff later.

The following pages are intended to provide examples of the different reference types. All references should be in 9-point font, with the first line flush left and reference numbers inserted in brackets. You are not required to indicate the type of reference; different types are shown here for illustrative purposes only. The DOI (digital object identifier) should be incorporated in every reference for which it is available (see Ref. 1 sample); for more information on DOIs, visit www.doi.org or [www.crossref.org](http://www.crossref.org).

*Periodicals*

[1] Vatistas, G. H., Lin, S., and Kwok, C. K., “Reverse Flow Radius in Vortex Chambers,” *AIAA Journal*, Vol. 24, No. 11, 1986, pp. 1872, 1873.

doi: 10.2514/3.13046

[2] Alyanak, E. J., and Pendleton, E., “Aeroelastic Tailoring and Active Aeroelastic Wing Impact on a Lambda Wing Configuration,” Journal of Aircraft, published online 10 Nov. 2016.

doi: 10.2514/1.C033040

[3] Dornheim, M. A., “Planetary Flight Surge Faces Budget Realities,” *Aviation Week and Space Technology*, Vol. 145, No. 24, 9 Dec. 1996, pp. 44–46.

[4] Terster, W., “NASA Considers Switch to Delta 2,” *Space News*, Vol. 8, No. 2, 13–19 Jan. 1997, pp. 1, 18.

All of the preceding information is required. The journal issue number (“No. 11” in Ref. 1) is preferred, but the month (Nov.) can be substituted if the issue number is not available. Use the complete date for daily and weekly publications. Transactions follow the same style as other journals.

*Books*

[5] Peyret, R., and Taylor, T. D., *Computational Methods in Fluid Flow*, 2nd ed., Springer-Verlag, New York, 1983, Chaps. 7, 14.

[6] Oates, G. C. (ed.), *Aerothermodynamics of Gas Turbine and Rocket Propulsion*, AIAA Education Series, AIAA, New York, 1984, pp. 19, 136.

[7] Volpe, R., “Techniques for Collision Prevention, Impact Stability, and Force Control by Space Manipulators,” *Teleoperation and Robotics in Space*, edited by S. B. Skaar and C. F. Ruoff, Progress in Astronautics and Aeronautics, AIAA, Washington, DC, 1994, pp. 175–212.

Publisher, place, and date of publication are required for all books. No state or country is required for major cities: New York, London, Moscow, etc. A differentiation must always be made between Cambridge, MA, and Cambridge, England, UK. Note that series titles are in Roman type.

*Proceedings*

[8] Thompson, C. M., “Spacecraft Thermal Control, Design, and Operation,” *AIAA Guidance, Navigation, and Control Conference*, CP849, Vol. 1, AIAA, Washington, DC, 1989, pp. 103–115

[9] Chi, Y. (ed.), *Fluid Mechanics Proceedings*, NASA SP-255, 1993.

[10] Morris, J. D., “Convective Heat Transfer in Radially Rotating Ducts,” *Proceedings of the Annual Heat Transfer Conference*, edited by B. Corbell, Vol. 1, Inst. of Mechanical Engineering, New York, 1992, pp. 227–234.

*Reports, Theses, and Individual Papers*

[11] Chapman, G. T., and Tobak, M., “Nonlinear Problems in Flight Dynamics,” NASA TM-85940, 1984.

[12] Brandis, A. M., Johnston, C. O., and Cruden, B. A., “Nonequilibrium Radiation for Earth Entry,” AIAA Paper 2016-3690, June 2016.

[13] Steger, J. L., Jr., Nietubicz, C. J., and Heavey, J. E., “A General Curvilinear Grid Generation Program for Projectile Configurations,” U.S. Army Ballistic Research Lab., Rept. ARBRL-MR03142, Aberdeen Proving Ground, MD, Oct. 1981.

[14] Tseng, K., “Nonlinear Green’s Function Method for Transonic Potential Flow,” Ph.D. Dissertation, Aeronautics and Astronautics Dept., Boston Univ., Cambridge, MA, 1983.

Government agency reports do not require locations. For reports such as NASA TM-85940, neither insert nor delete dashes; leave them as provided. Place of publication *should* be given, although it is not mandatory, for military and company reports. Always include a city and state for universities. Papers need only the name of the sponsor; neither the sponsor’s location nor the conference name and location is required. *Do not confuse proceedings references with conference papers*.

*Electronic Publications*

Regularly issued electronic journals and other publications are permitted as references. Include the DOI if provided; otherwise provide the full URL. Archived data sets also may be referenced as long as the material is openly accessible, and the repository is committed to archiving the data indefinitely. References to electronic data available only from personal websites or commercial, academic, or government ones where there is no commitment to archiving the data are not permitted in the reference list.

[15] Atkins, C. P., and Scantelbury, J. D., “The Activity Coefficient of Sodium Chloride in a Simulated Pore Solution Environment,” *Journal of Corrosion Science and Engineering* [online journal], Vol. 1, No. 1, Paper 2, URL: <http://www.cp/umist.ac.uk/JCSE/vol1/vol1.html> [retrieved 13 April 1998].

[16] Vickers, A., “10-110 mm/hr Hypodermic Gravity Design A,” *Rainfall Simulation Database* [online database], URL: <http://www.geog.le.ac.uk/bgrg/lab.htm> [retrieved 15 March 2006].

Break website addresses after punctuation, and do not hyphenate at line breaks.

*Computer Software*

[17] TAPP, Thermochemical and Physical Properties, Software Package, Ver. 1.0, E. S. Microware, Hamilton, OH, 1992.

Include a version number and the company name and location of software packages.

*Patents*

Patents appear infrequently. Be sure to include the patent number and date.

[18] Scherrer, R., Overholster, D., and Watson, K., Lockheed Corp., Burbank, CA, U.S. Patent Application for a “Vehicle,” Docket No. P-01-1532, filed 11 Feb. 1979.

*Private Communications and Websites*

References to private communications and personal website addresses are not permitted. They may, however, be incorporated into the main text of a manuscript or may appear in footnotes.

*Unpublished Papers and Books*

Unpublished works can be used as references as long as they are being considered for publication or can be located by the reader (such as papers that are part of an archival collection). If a journal paper or a book is being considered for publication, choose the format that reflects the status of the work (depending upon whether it has been accepted for publication):

[19] Doe, J., “Title of Paper,” *Name of Journal* (to be published).

[20] Doe, J., “Title of Chapter,” *Name of Book*, edited by…, Publisher’s name and location (to be published).

[21] Doe, J., “Title of Work,” Name of Archive, Univ. (or organization), City, State, Year (unpublished).

Unpublished works in an archive *must* include the name of the archive and the name and location of the university or other organization where the archive is held. Also include any cataloging information that may be provided.

1. Student, Aerospace Engineering [↑](#footnote-ref-1)
2. Student, Aerospace Engineering [↑](#footnote-ref-2)