UC Berkeley
Berkeley Scientific Journal

Title
How to Win: Optimizing Overcooked

Permalink
https://escholarship.org/uc/item/4sq3h354

Journal
Berkeley Scientific Journal, 24(2)

ISSN
1097-0967

Author
Nolan, Nick

Publication Date
2020

DOI
10.5070/BS3242049344

Copyright Information
Copyright 2020 by the author(s). All rights reserved unless otherwise indicated. Contact the author(s) for any necessary permissions. Learn more at https://escholarship.org/terms

Undergraduate

eScholarship.org Powered by the California Digital Library
University of California

How to Win: Optimizing Overcooked

BY NICK NOLAN

Overcooked—the sensational group cooking video game—made its smash appearance in 2016. The premise is simple: prepare as many of the requested meals as possible before the time runs out by cooking meat, chopping vegetables, or boiling soup; combining the ingredients; and cleaning the plates you just put out for consumption (Fig. 1). The innately chaotic nature of the game unfurls in the floor plan: some levels are coated with ice that make it easy to slip off of the stage; others, a rotating center stage. When four people gather to try flipping burgers, it becomes difficult to navigate the unruly layouts while following occasionally-complex cooking instructions in time to satisfy the customer. Strategy is crucial to fulfill orders quickly and proceed to the next level.

Naturally, this idea prompts the question: how can we develop the best strategy to play Overcooked? How can we find the way to make the greatest number of food items in the time allotted for each level?

First and foremost, it seems the solution to our quest may not lie in our own hands—recent advances in computer science have demonstrated that machine learning algorithms have been able to outplay human experts. Specifically, these methods have bested human players on some of the first games developed on the Atari 2600, a gaming console from 1977.[1,2]

Machine learning is likely the way to go to play the best game of Overcooked, then. These Atari algorithms, developed in 2013 by DeepMind Technologies' Volodymyr Minh, were able to learn how to play a host of Atari games, eventually exceeding the highest scores of some of the most skilled human players. To accomplish this task, Minh utilized a class of machine learning known as reinforcement learning, in which the computer is a sort of "player," learning about the environment with which it interacts. The "player" eventually encounters good or bad rewards, and learns how to act to maximize the good rewards it receives.

Figure 1: Gameplay footage of a standard level of Overcooked. *The dropoff location, on the left, is where players need to submit completed meal orders, in the top left. The rest of the level is home to treadmills that move the player around, and resources are spread far apart from each other to make the level more challenging.*

This amounts to a massive amount of trial-and-error-type learning, and takes minutes to simulate a single second's worth of gameplay; the Atari system was slowed down by multiple orders of magnitude, and it was not until late 2014 when IBM's Xiaoxiao Guo developed a more efficient, real-time algorithm that—while slightly less effective—was able to play live.[3]

We are met at a crossroads: we can either develop an algorithm that takes a long time and plays a slowed-down version of the game, *or* we can prioritize an algorithm that plays real-time, albeit possibly suboptimally. Here, we will focus on the former to develop a "best" strategy, due to the generally messy nature of real-time strategy development.

"Naturally, this idea prompts the question: how can we develop the best strategy to play Overcooked?"

In order to properly tackle this problem, we must first consider our approach. To procure the greatest number of meals in a set time frame, we will make some simplifying assumptions, and establish *how* we will frame the data to be inputted into this system:

The game board will be split into discrete tiles in a grid over the entire board, small enough that only one unit—either player, plate, or other utility on the board—can fit into each tile.

The board will be split into layers of the above grid, corresponding to the distinct types of entities on the board: one layer corresponding to tabletops; one for non-walkable hazards; one for each player; one for tomatoes; and so on. For each of these items, its corresponding grid copy will contain 1's in every position in which the item is contained; every other position, will be filled in with 0's (Fig. 2).

One time step will be defined as the amount of time it takes for a player to move from one grid tile to any adjacent tile.

So, great. We've created a model for the system which speaks to where everything is on the board, and we've discretized the system such that players have a finite number of moves until the game is finished. However, the computer does not know what the objective of Overcooked is—it has been preprogrammed exclusively with knowledge of how to move and use other moveable items. So how can a computer choose the best ordering of actions, when it doesn't even know what it should be doing?

Simple: it won't. Not for a *long* time.

We have to start by making a guess of

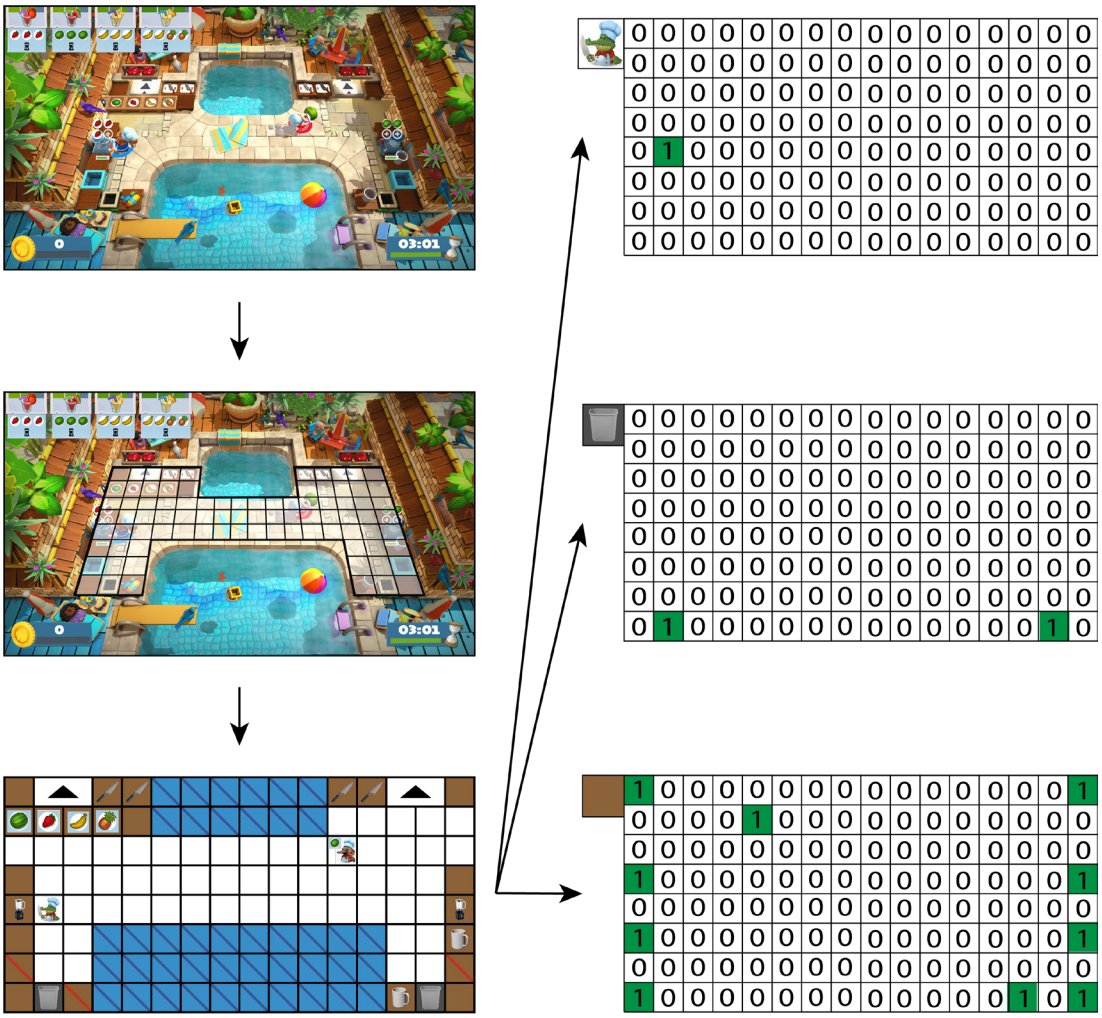

Figure 2: Each game board can be roughly modeled as a grid of tiles; to format this for input into our machine learning algorithm, we simply need to break the game down into its gridlike structure*. From here, all that's left is to create a new copy of the grid for each type of tile, and label each tile that has the item with a "1," and each that doesn't with a "0".*

"To accomplish this task, Minh utilized a class of machine learning known as reinforcement learning."

how to proceed from each possible configuration of positions, called a state, and updating our guesses as we proceed through simulations. Enter Q-Learning, an algorithm proposed in 1989 which does precisely this.[4,5] More recent advances from 2002 by the University of York's Spiros Kapetanakis expanded Q-Learning to work with multiple players, allowing for this approach to work with Overcooked.[6]

Q-Learning is a standard reinforcement learning algorithm which identifies and determines the *value*, denoted with a *Q*, of every *action*, *a*, at every possible *state*, *x*. From here, we can determine the optimal action to take at any state, which is known as a *policy*. The value of each state-action pairing for a policy π is defined as follows:

$$Q^{\pi}(x,\ a)\ =\ R_x(a)\ +\ \gamma \sum_y P_{xy}[\pi(x)]V^{\pi}(y).$$

By no means is this pleasant to look at, but the general takeaway here is as follows: the first term, $R_x(a)$, speaks to the reward obtained, purely from taking an action *a* at a state *x*. For example, dropping a filled order down into the dropoff location would yield a large reward, while cutting carrots or cooking meat would not achieve the same reward. This is not to say that these latter actions aren't valuable, though—their value shines in the second term of the equation. This term—which is admittedly complex—equals the potential of any given action to result in a downstream success. Note that this success is not guaranteed; each order that comes in is random, and so each action may be more or less successful based on the randomly generated requests. Regardless, this is how preparing food is useful—it sets up for future completed dishes to get even more points.

This is all splendid, but how do we use this? In essence, it amounts to making random or slightly-educated guesses of how valuable each state-action pairing is, testing these out, and updating them according to whether we found them to be as useful as we initially thought they would be. For example, we might have initially thought that it would be very valuable to chop lettuce, but if our simulations reveal that there's only one very uncommon dish on that level that requires lettuce, we can safely say that we overestimated the value of chopping lettuce, and update accordingly.

So, there we have it—under the model we have constructed, we will be able to play the optimal game of Overcooked. It will take some time, but by playing several thousands of simulations of a given level, a computer will be able to roughly determine the values of each state-action pairing for the optimal policy. From here, we simply need to choose what Q-Learning believes to be the most valuable action for each state we're in until the end of the game.

This concludes our quest; from constructing our model, we have simulated several thousand rounds of play, developed an approximation for the value of every move, and used these approximations to make the best strategy in Overcooked. Armed with that knowledge, replacing your own player with the optimal computer may even leave your teammates happier without you.

REFERENCES

1. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. *arXiv*. https://arxiv.org/abs/1312.5602
2. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236
3. Guo, X., Singh, S., Lee, H., Lewis, R. L., & Wang, X. (2014). Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning. In Ghahramani Z., Welling M., Cortes C., Lawrence N. D., & Weinberger, K.Q., (Eds.), *Advances in Neural Information Processing Systems*: 27 (pp. 3338-3346).
4. Watkins, C. J. C. H. (1989). *Learning from delayed rewards.* [Doctoral thesis, King's College]. ResearchGate.
5. Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*, 279-292. https://doi.org/10.1007/BF00992698
6. Kapetanakis, S., & Kudenko, D. (2002). Reinforcement learning of coordination in cooperative multi-agent systems. In Dechter, R., Kearns M. S., & Sutton, R. S., (Eds), *Eighteenth National Conference on Artificial Intelligence*, 2002, 326-331. https://doi.org/10.5555/777092.777145

IMAGE REFERENCES

1. *Banner*: (2016). *Characters from Overcooked* [jpg image]. IGN. https://www.ign.com/wikis/best-of-2016-awards/Best_Multiplayer.
2. *Figure 1*: (2018). *Gameplay footage of level in Overcooked 2* [jpg image]. Gamers of the World. http://gamersoftheworld.com/overcooked-2-review-a-great-second-course.
3. *Figure 2*: (2018). *Gameplay footage of level in Overcooked 2* [jpg image]. Steam. https://store.steampowered.com/app/909720/Overcooked_2__Surf_n_Turf.