Dominion: A New Frontier for AI Research

Aron Sarmasi
UC Davis
asarmasi@ucdavis.edu

Danny Halawi
UC Berkeley
dhalawi@berkeley.edu

Siena Saltzen
UC Davis
scsaltzen@ucdavis.edu

Joshua McCoy
UC Davis
jamccoy@ucdavis.edu

***Abstract*—In recent years, machine learning approaches have made dramatic advances, reaching superhuman performance in Go, Atari, and certain poker variants. These games, and others before them, have served not only as a useful testbed, but have also helped to push the boundaries of AI research. Continuing this tradition, we examine the tabletop game Dominion and discuss its properties that make it well suited to benchmarking the next generation of reinforcement learning (RL) algorithms. In particular, we highlight the large variety of game setups, resulting in a novel ruleset for each game played; we argue this will force agents to approach Dominion with a greater degree of adaptability than other games require. We also present the Dominion Online Dataset, a collection of over 2,000,000 games of Dominion played by experienced players on the Dominion Online webserver. Finally, we introduce an RL baseline bot that uses existing techniques to beat common heuristic based bots, and shows competitive performance against the previously strongest bot Provincial.**

***Index Terms*—Dominion, reinforcement learning, benchmark, game, dataset**

I. Introduction

Games have long played a role in AI research, both as a test-bed, and as a moving goal-post, constantly driving innovation. From the heyday of chess agents, when Deep Blue beat Gary Kasparov, to more recent advances, like AlphaGo's dark horse ascent to fame, games have both assisted AI research, and provided something to aim for. As the AIs got better, the games they were applied to also got more complex. New game mechanics, such as the fog of war in StarCraft and the stochasticity of Poker, pushed researchers to adapt their methods to ever greater generality. In this paper, we argue that the deck-building strategy game Dominion [1] deserves to join the ranks of AI benchmark games, providing an RL-based bot in service of that benchmark.

Dominion has all of the abovementioned elements, but it also incorporates a mechanic that is not present in other popular RL benchmarks: every game is played with a different set of cards. Since each dominion card has a specific rule printed on it, and the set of 10 cards for a game are randomly picked from among hundreds of cards, no two games of Dominion can be approached the same way. Thus a key part of playing Dominion is adapting one's inductive bias of how to play to the specific cards on the table. Although a general knowledge of which card combos are powerful is certainly helpful, with over $\binom{350}{10}$ possible kingdom card setups, it is insufficient to rely solely on inductive bias. Players need to spend a considerable amount of mental effort at the beginning of the game to identify possible strategies (there are usually multiple competing strategies on most boards), and as the game progresses, players must pick up on what strategy the opponent is following and possibly adapt their own strategy to counter their opponent's. This game mechanic is going to require researchers to develop a unique approach if they are to beat today's top Dominion players. We describe all the other features that make Dominion attractive for study in section II.

In addition to its game mechanics, Dominion has an active player community, with regular tournaments, well documented strategy guides [2], and a popular online web server [3]. The size of the player community serves not only to increase the relevance and audience of Dominion related research, but it also increases the credibility of a bot who can beat the top players. The strategy guides on the other hand help researchers who are new to the game to become familiar with the kinds of behavior to look for in their bots. Finally, and most importantly, the developers of the web server have given us access to over 2,000,000 game records, of which we've already compiled 30,000 using cards from the base game. Section III gives a more detailed breakdown of the available data.

In this paper, we lay out the reasons why Dominion is a strong benchmark candidate for AI systems and we provide an RL player as a baseline for future research. We start by introducing the game and describing the various mechanics that are friendly to AI development. Next we describe a large dataset of human-played games which we have made easily available online. Finally, we introduce a deep RL based bot using existing techniques that can handily beat common heuristic based bots and and is competitive against the previous best AI, Provincial. The dataset and the code are available at *gitub.com/aronsar/domray*.

II. Dominion

Dominion is a turn-based, deck building strategy game for 2-6 players (in this paper we focus on the 2-player scenario). The object of the game is to own the most victory points (VPs) when the game ends, which typically means buying victory cards over the course of the game. When the Province victory card pile runs out, or any three piles run out, the game ends. In a turn, a player may play a single action card and any number of treasure cards, and then buy a single card, before discarding the rest of their hand and everything they played that turn and drawing 5 cards from their deck to form the next turn's hand. As cards are bought, they are placed directly in the discard pile, and are only drawn sometime later, i.e. once the deck runs out and the discard pile is shuffled to create a fresh deck.

Fig. 1. Typical game setup on the Dominion Online web server–this web interface closely models the analog version of the game. The "basic" cards in the blue rectangle are present in every game of Dominion, whereas the 10 "kingdom" cards in the magenta rectangle are chosen randomly from a set of up to 350 cards. A dominion card has the card cost in the bottom left, the number of copies in the top left, and the card art in the middle. The effect of the card pops up when the user hovers over the card; it is printed on the card in the analog version.

Managing the contents of one's deck is an important concept in dominion gameplay. Each player starts the game with 7 Coppers and 3 Estates—some of the weakest cards in the game. Over the course of the game, players strive to fill their decks with powerful cards, and remove weak cards, so as to ensure that every hand contains good cards. To remove a card from the deck, a player must "trash" it; the only way to do this is using certain action cards. Another important game concept is pacing: players will typically buy actions and treasures (thereby improving the quality of their decks) until the very end of the game, where they quickly try to buy as many victory cards as they can, thereby increasing their score. This is because victory cards usually have no beneficial effect during the game, and so it is best to leave off buying them until the player's deck is strong enough to support a large influx of weak cards. A full description of the rules is available at [4].

The vast majority of the complexity of Dominion is due not to its basic rules, described above, but rather to the large variety of cards that are available to play. During game setup, players randomly choose 10 different "kingdom" cards from among over 350 different available cards; 10 copies of each kingdom card is used to form the supply piles. In addition to the kingdom cards, every game is played with a set of base cards, as shown in Fig. 1. This represents a lower bound of $\binom{350}{10} = 6 \cdot 10^{18}$ different possible game setups, but due to additional mechanics released in later expansions (which we do not cover in this paper), an even larger variety is possible. Since each kingdom card has a unique rule printed on it, the ruleset, and thus the optimal strategy, are going to be different each game.

In addition to this unique mechanic, Dominion has a few other properties that make it especially attractive to AI researchers. Unlike Atari games where each pixel is a dimension, the dominion state space is quite small. Each game is played with only 17-25 different cards, and these cards can only be in a few different positions: these are the supply, a player's deck, their hand, or their discard (plus a few other locations that only come into play with certain cards). Based on our analysis of Dominion game logs (see section III), games usually consist of 15-40 turns, and a player usually has to make less than a few hundred decisions in a typical game. These properties allow researchers to solve a very interesting problem without the high computational resource requirements entailed by other baselines.

Furthermore, not every decision in Dominion needs to be given the same amount of thought. Specifically, the "buy" decisions, where players decide what cards to populate their decks with, are widely regarded as the most difficult decisions, requiring the most consideration [5]. "Action" decisions, where players decide what cards to play during their turn, can

TABLE I
STATISTICS FOR 5471 (OUT OF 30,000) RATED GAMES OF 2-PLAYER DOMINION USING ONLY 2ND EDITION BASE GAME CARDS, DOWNLOADED FROM THE DOMINION ONLINE WEB SERVER.

Statistic	Value
Avg game length	31
Avg VP totals/player	20
Avg margin of victory [VPs]	8
Avg gain decisions/player	19
Avg card plays/player	44
Avg mu [7] (skill)	0.65
Avg phi (deviation)	0.17
Ties	1.90%

often be scripted according to simple rules of thumb without significantly affecting game outcome. While there certainly exist card sets where the action decisions play a critical role in a player's strategy, in most games they do not. Since this scripting still preserves the game's most interesting features, this represents a natural affordance for AI researchers, who can test their techniques against an even smaller state space. Another example in which Dominion supports incremental improvements is that a researcher can pick and choose which cards to play with; restricting testing to just the simpler cards early on can help accelerate development. For example, the base game Dominion, and the expansions Seaside and Hinterlands, all contain cards that the game designer has specifically intended to be less complex and more beginner friendly [6]. Few other games allow testing with a reduced ruleset without grossly upsetting the game balance.

III. DOMINION ONLINE DATASET

One of the most popular ways to play dominion, especially given the high rate of self-isolation in the past year, is on the web server Dominion Online. Fortunately for the research community, the developers save game logs for all the games played on the site. It is these logs that we have downloaded, processed, and will now present.

When playing on the Dominion Online web server, players are given the option to play ranked or casual games, with 1 to 3 opponents. A player then enters the automatch queue, where after a short time they are matched with somebody close to their skill level. Then, a random set of cards is selected from among the expansions that the player has paid for and the game begins. To keep our project in scope, in this work we have restricted ourselves to ranked, 2-player games using the 2nd edition Dominion card set (commonly referred to as the "base game"). We've so far downloaded 5471 out of 30,000 such games (downloading the games is a bit of a tenuous process that we are working to improve), and present statistics on these games in table I.

To keep track of players' skill levels, Dominion Online implements a rating system [8] based on Glicko-2 [7]. Without going to deeply into the intricacies of the rating system, we suffice it to say that each player has an estimated skill μ, and a deviation ϕ, such that the system can be 95% confident that the players skill is between $(\mu - 2\phi)$ and $(\mu + 2\phi)$. A newly made account starts with a μ of 0, and when the player wins a game their rating improves and vice versa. Larger increments/decrements of skill are applied when the skill levels of the players are further apart. The ratings and deviations for the players of each game are made available along with the game data. From table I, we can see that on average, the players in our dataset have μ = .65 and ϕ = .17. For reference, the average player from our dataset has a 66% chance of beating a player with μ = 0, and a 20% chance of beating a top-10 player (using current ratings from the leaderboards). Please see [7] for details of these calculations. When compiling the dataset of base games, we included only games where the sum of the two players' μ scores was greater than 1, in order to filter out games with too low ranked players. As a point of reference, the author has a μ of -0.07 at the time of this writing, and is the best Dominion player they know.

Additionally, we present an excerpt of a game log in table II to familiarize the reader with the format of the data. Each game log lists the ids of the two players, followed by a list of

TABLE II
EXCERPT OF A GAME LOG FROM THE DOMINION ONLINE DATASET SHOWING TWO FULL TURNS OF GAMEPLAY.

```
<player_id1>∼∼<player_id2>
Supply: 10 Curse, 60 Copper, 40 Silver, 30 Gold, 14 Estate, 8 Duchy, 8
Province, 10 Bandit, 10 Bureaucrat, 10 Cellar, 10 Harbinger, 10 Militia,
10 Poacher, 10 Remodel, 10 Throne Room, 10 Village, 10 Workshop
0:P0 - GAME_META_INFO (<game_id>):
1:P1 - STARTS_WITH: 7 Copper
2:P1 - STARTS_WITH: 3 Estate
...
148:P1 - NEW_TURN (10, 0):
149:P1 - PLAY: 1 Harbinger
150:P1 - DRAW: 1 Cellar
151:P1 - GETS_ACTION (1):
152:P1 - LOOK_AT: 3 Copper, 1 Province, 1 Harbinger, 1 Militia, 1 Gold
153:P1 - TOPDECK: 1 Gold
154:P1 - PLAY: 1 Poacher
155:P1 - DRAW: 1 Gold
156:P1 - GETS_ACTION (1):
157:P1 - GETS_COIN (1):
158:P1 - PLAY: 1 Cellar
159:P1 - GETS_ACTION (1):
160:P1 - DISCARD: 2 Estate
161:P1 - DRAW: 1 Copper, 1 Remodel
162:P1 - PLAY: 1 Bandit
163:P1 - ”GAIN: 1 Gold
164:P2 - REVEAL: 2 Village
165:P2 - DISCARD: 2 Village
166:P1 - PLAY_TREASURES_FOR (4): 1 Copper, 1 Gold
167:P1 - BUY_AND_GAIN: 1 Throne Room
168:P1 - SHUFFLES
169:P1 - DRAW: 2 Copper, 2 Estate, 1 Bandit
170:P2 - NEW_TURN (10, 0):
171:P2 - PLAY: 1 Bandit
172:P2 - ”GAIN: 1 Gold
173:P1 - REVEAL: 1 Estate, 1 Throne Room
174:P1 - DISCARD: 1 Estate, 1 Throne Room
175:P2 - PLAY_TREASURES_FOR (3): 3 Copper
176:P2 - BUY_AND_GAIN: 1 Village
177:P2 - SHUFFLES
178:P2 - DRAW: 1 Copper, 1 Gold, 1 Estate, 1 Bandit, 1 Throne Room
...
```

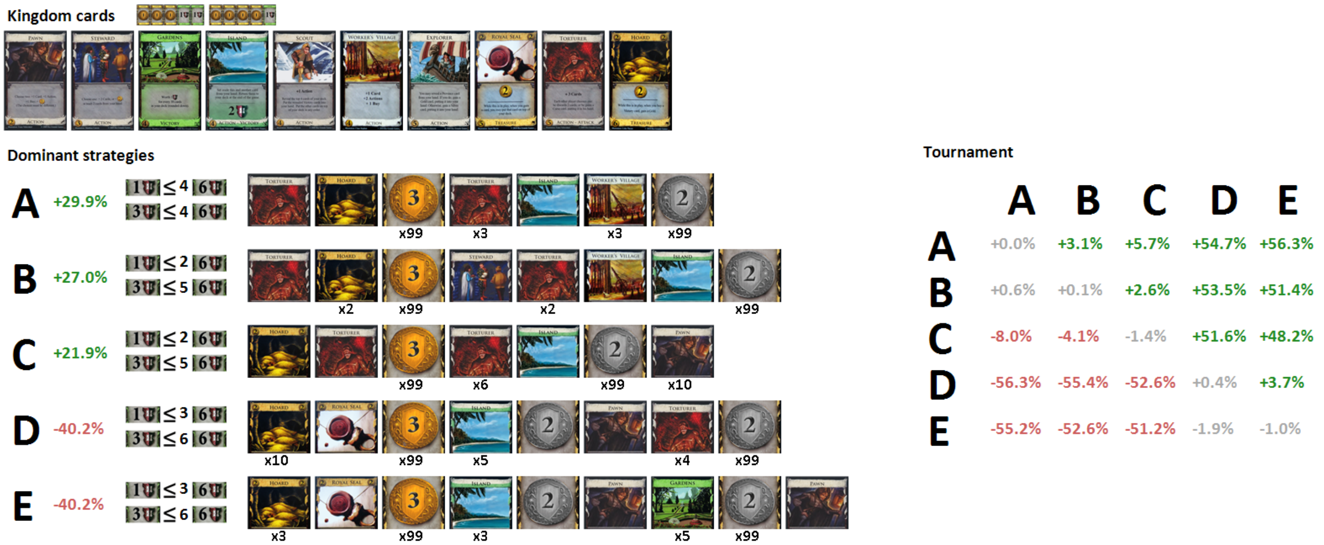

Fig. 2. A visualization of the leading strategies Provincial discovered given a random cardset (including cards from later expansions). From left to right, we see 1) the expected win ratio when each of the five strategies plays against each other strategy average over 10,000 games, 2) the victory card purchase thresholds, 3) the buy menus.

the supply pile cards, and the game id. Then each line of the log corresponds to some event that the game client processed, in the order the players caused those events. For example, the card Harbinger first draws the player a card, then increases the number of actions they can take this turn by 1, and finally allows them to look through their discard pile and optionally topdeck a card. This sequence of events is reflected by lines 149 - 153.

IV. Related Works

The literature on Dominion is somewhat sporadic, limited to a few masters theses, conference papers, and unpublished works. We focus this section on existing Dominion playing AIs, specifically the ones which we did not implement, and explain why we believe they are not suitable benchmarks.

Van der Heijden [9] develops a framework of definitions for reasoning about deck building games, focusing their analysis on the equivalency of different game states, and choosing the optimal buy decision in a certain state. However, they make assumptions that overly simplify and rigidify the game, rendering their AIs unusable to us.

Fynbo and Nellemann [5] develop a bot consisting of three parts: a neural network to estimate the number of turns remaining in a game, and two NEAT-powered networks to take actions and make buys. Their bot beat Big Money (a common buy-heuristic explained in section V) 55% of the time and a finite state machine bot 48% of the time; in section V we review the performance of Provincial and our DQN-bot against Big Money and other heuristic bots and show that this is not very impressive. Mahlmann, Togelius, and Yannakakis [10] use Dominion as a complex test-bed for their genetic search algorithm whose purpose is to aid game designers in coming up with more balanced rulesets. The authors implement a number of quite complex rule based agents that they use to test this algorithm. However, these bots demonstrate mixed performance when playing against Fynbo and Nellemann's AI, and do not seem strong enough in general to warrant reimplementing in our environment.

In addition to performance, we were looking for existing bots that had the potential to scale to the rest of the Dominion expansions beside the base game. Jansen and Tollisen [11] build an AI for Dominion using Monte Carlo methods that handily beat moderately-experienced human players, and do quite well against heuristic baselines. However, for their approach they limit all their tests to just 10 of the simplest action cards from the base game. Although they suggest that their approach could be scalable to the complete Dominion, it is not at all trivial to do so, and we considered it outside the scope of this work. Winder's [12] method uses a neural network to predict a value approximation of every possible next game state. This approach is not scalable for two reasons: their genetic training process is quickly overwhelmed by the totality of Dominion cards, and the value approximation requires an accurate prediction of the next state, which is not always possible. Fenner [13] applies a hybrid approach, combining heuristics, neural networks, and tree search to build a Dominion AI. The most interesting aspect of the approach is that it vectorizes cards by their effects. Because some cards reuse effects from earlier cards, the intent of this representation is to possibly generalize to previously unseen cards whose effects are a subset of previously seen effects. Although the specifics of the approach are a bit outdated and costly to scale, we believe this idea could make for interesting future work, perhaps using something like an autoencoder instead of hand engineering.

TABLE III
PLAY PERFORMANCE OF PROVINCIAL AND DQN-BOT AGAINST HEURISTIC BASELINES. THE SPECIFIC HEURISTIC IS EXPRESSED AS A BUY MENU, WHICH ARE EXPLAINED IN SECTION V-A. IN EACH GAME, THE HEURISTIC BASELINES ARE GIVEN THE FIRST TURN.

		Provincial			DQN-bot		
Bot	Bot Buy Menu	Wins	Ties	Losses	Wins	Ties	Losses
Big Money	(Gold, 99), (Silver, 99)	972	4	24	912	27	61
Double Witch	(Witch, 1), (Gold, 99), (Witch, 1), (Silver, 99)	596	69	335	561	63	376
Big Smithy	(Gold, 99), (Smithy, 1), (Silver, 99)	930	7	63	851	38	111
Big Militia	(Gold, 99), (Militia, 1), (Silver, 99)	946	9	45	850	31	119
Village/Smithy Engine	(Gold, 99), (Smithy, 5), (Militia, 1), (Village, 5), (Silver, 99)	994	2	4	418	24	558
Provincial	(Witch, 1), (Gold, 99), (Militia, 1), (Witch, 1), (Market, 3), (Silver, 99)	550	53	397	575	62	363

V. BENCHMARKS

To demonstrate that Dominion is ripe for reinforcement learning research we've gathered and implemented some of the best bots currently available, and compare them to a simple DQN bot. The simplest Dominion bots follow a set of easily articulable heuristics, while still regularly beating novice players; indeed beginners are often encouraged to manually play out some of these heuristics to get a feel for the game, before implementing their own strategies. The most simple of these heuristics is known as "Big Money". In essence, Big Money dictates the player to buy the most expensive money card available, unless they can afford a Province, in which case they should buy that. Many of the other common heuristic bots are simple variations on Big Money. For example, "Smithy Big Money" is the same as the original, but dictates that the player should buy a single Smithy when they are able to. Since Smithy draws 3 cards, as long as the player draws more than 2 coins' worth of money, Smithy is strictly stronger than Silver (a treasure card worth 2 coins), which is what the original Big Money strategy would have dictated be bought.

A. Buy Menus

In 2014, Fisher formalized this type of heuristic into a "buy menu" [14]. This formalism is so useful that we were able to cast most of the best known heuristic bots in terms of a buy menu, which we list in table III. A buy menu is simply an ordered list of card/count tuples. To play a buy menu, a bot buys the leftmost card it can afford, decrementing the count of that card. Cards with zero counts are skipped over, ensuring no more than the specified number of each card is bought. To ensure that the bots buy victory cards correctly, we use Fisher's convention: Provinces are bought any time the bot can afford it, Duchies are only bought if there are less than 4 Provinces left, and Estates are only bought if there are less than 2 Provinces left. As a note, none of the heuristic bots use common end-game rules of thumb such as the "penultimate province rule", which gives guidance on when to buy Provinces and Duchies once only 2 Provinces are left (so as to not end the game in a losing position).

B. Provincial

To answer the question of how to assemble the best buy menu given a card set, Fisher proposes the AI system Provincial [14]. This bot uses competitive coevolution, taking an arbitrary set of kingdom cards, and evolving a list of leading strategies over the course of a few million games. The core of the bot's strategy is a buy menu template with a fixed number of slots. During training, mutations can replace cards from the template, modify the purchase count, swap the order of cards, and change the victory card purchase thresholds. At the start of training, two pools are initialized randomly: a general pool of about 100 strategies and a set of 5 leading strategies. A single generation consists of mutating the leaders to form a new general pool, and then testing the resulting strategies against one another to pick new leaders. For most kingdom card sets, 32 generations are enough to produce a strong buy menu that even experienced players struggle against. We also test Provincial against some of the best known heuristics (also cast as buy menus), and show results in table III; Provincial clearly sweeps the board. This is despite the fact that the heuristic baselines always went first, and going first seems to carry a significant advantage. This is evidenced by the fact that in the Provincial-Provincial matchup, the first player won 55% of the time, whereas second player only won 40% of the time. In addition to this strong performance, Provincial's relatively simple strategy representation (ie. the buy menu) allows it to provide a very clear and explanative visual representation of the leading strategies, as pictured in Fig. 2. This property further increases the usefulness of Provincial as a benchmark, as it allows a researcher to quickly identify leading strategies on a new card set they wish to test their agents on.

C. DQN-bot

We implement a simple RL-based buyer bot using Rainbow DQN [15], trained to play Dominion via self-play. Our intention with this bot is not to break new ground in the field of RL, but rather simply to show that recent developments in RL can easily outpace the current state of the art in Dominion. For that reason, we present here only necessary detail and encourage the reader to refer to our code for the minutiae. The bot is a small fully connected neural network (with two hidden layers, both size 256) that takes in the game state representation and action mask, and returns which card to gain. The bot only makes gain decisions; action card decisions are handled by the same heuristics that Provincial uses. To train the bot, we use Rainbow DQN, but we don't use dueling networks or double Q due to technical difficulties. We train for 1000 steps, which comes out to about 7000 games of Dominion, and 45 minutes of train time on an NVIDIA 1080 Ti GPU. This is comparable to how long Provincial takes to train with 32 generations on a modern computer.

The game state representation we use is also quite simple, with just enough information to make buy decisions with. Similar to [12], we concatenate three vectors, one each for the player's deck, the supply, and the opponent's deck. Each vector consists of the counts of each card in that location. We also add an extra dimension for the number of empty supply piles. This representation is powerful enough to allow the bot to consider its opponent's strategy and to determine how close to the game is to ending, but not fine grained enough to allow reasoning about the next few hands (since no difference is made between cards in the discard pile, the hand, or the deck).

We see in table III that our DQN-bot beats or ties Provincial almost two thirds of the time. It also clearly beats all of the heuristics except for the engine, against which it struggles. We hypothesize that this is because the learner did not explore this particular combo during its training and so got confused when it saw its opponent take this strategy. Indeed when we examined the game logs, the bot was behaving strangely, buying Duchies too early, and even buying Curses. We believe this problem is solvable, likely with a combination of more training and a more sophisticated game representation, but we leave that to future work. In particular, we believe that this simple instance of a DQN-bot shows that existing RL based methods can crush existing bots.

VI. Conclusion

We started off by discussing some of the properties of Dominion that make it particularly well suited for AI research. Specifically, these include its huge variety of card combinations, a vibrant player community, and various affordances for making the state space more or less complex. Among these affordances are decisions of different complexity/importance, and a gradient of card complexity, which both serve to lower the barrier to entry for researchers trying out new ideas. Next we explored the Dominion Online Dataset, which consists of 30,000 high quality base game records, and up to 2 million total games. These games were played by experienced Dominion players, and can serve as a strong pretraining tool for RL based AI approaches. Finally, we explored the current state of the art in Dominion playing bots, and implemented a number of heuristic based bots, along with the genetically evolved bot Provincial. We compared these to a simple RL-based bot using recent innovations in RL, and showed that the RL-based bot generally outperformed the existing baselines.

Acknowledgment

We would like to thank Ben Zhang for helping us understand and extend the DomRL codebase, ceviri for giving us free server time and assistance with downloading games from Dominion, Markus for his help with extracting the base game ids, and Dan Brooks for his good advice.

References

[1] D. X. Vaccarino, “Dominion,” 2008. [Online]. Available: http://riograndegames.com/Game/ 278-Dominion

[2] “Dominion strategy wiki.” [Online]. Available: http://wiki.dominionstrategy.com/

[3] S. Meijer, “Dominion online,” 2017. [Online]. Available: dominion.games

[4] “Gameplay.” [Online]. Available: http://wiki.dominionstrategy.com/index.php/Gameplay

[5] R. B. Fynbo and C. S. Nellemann, “Developing an agent for dominion using modern ai-approaches,” Master's thesis, IT-University of Copenhagen, 2010.

[6] D. X. Vaccarino, “Donald x.'s guide to 12 dominion expansions,” 2 2019. [Online]. Available: https://dominionstrategy.com/2019/02/22/donald-x-s-guide-to-12-dominion-expansions/

[7] M. E. Glickman, “Example of the glicko-2 system,” Boston University, Tech. Rep., 11 2013, unpublished.

[8] S. Meijer, “Rating details,” 3 2017. [Online]. Available: http://forum.shuffleit.nl/index.php?topic=1679.msg5891

[9] R. R. van der Heijden, “An analysis of dominion,” Master's thesis, Leiden University, 8 2014.

[10] T. Mahlmann, J. Togelius, and G. N. Yannakakis, “Evolving card sets towards balancing dominion,” in *2012 IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

[11] J. V. Jansen and R. Tollisen, “An ai for dominion based on monte-carlo methods,” Ph.D. dissertation, University of Agder, 6 2014.

[12] R. K. Winder, “Methods for approximating value functions for the dominion card game,” *Evolutionary Intelligence*, vol. 6, no. 4, pp. 195–204, 2014.

[13] C. Fenner, “Ann assisted heuristic tree search in computer dominion,” Master's thesis, University of Oklahoma, 2014.

[14] M. D. Fisher, “Provincial: A kingdom-adaptive ai for dominion,” Stanford, Tech. Rep., 2014, unpublished.

[15] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, “Rainbow: Combining improvements in deep reinforcement learning,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.