Long-Term vs. Short-Term Strategy in the Game of Monopoly

Mitchell Eithun
eithunm@ripon.edu

McKenzie Lamb
lambm@ripon.edu

Andrea Young
younga@ripon.edu

Dept. of Mathematics
Ripon College
Ripon, WI 54971

Abstract

We use a computer model to analyze and compare a variety of strategic paradigms in the game of Monopoly®, with an emphasis on strategies that can be implemented by humans.

The strategies that we consider can be divided into two categories: dynamic and static. Dynamic strategies seek to maximize short-term gains by analyzing the current state of the board. Static strategies, by contrast, work toward fixed, long-term goals determined by rankings of the property groups.

We first use hill-climbing on a graph and a ranking scheme based on the Google PageRank algorithm to optimize static strategy. We then conduct head-to-head competitions between our best static strategies, our dynamic strategy, and static strategies suggested by results from other papers, and use the results to produce a ranking of all of these strategies. We also use a blended static/dynamic strategy to show that it is seldom a good idea to depart from a strong static strategy, even for potentially large short-term gains.

Keywords: Hill-climbing on graphs, optimization, PageRank algorithm, combinatorial games

Introduction

The popular board game Monopoly® is a "fast-dealing property trading game" Hasbro [2007]. Players take turns moving around the board (see

The UMAP Journal 38 (1) (2017) 35–57. ©Copyright 2017 by COMAP, Inc. All rights reserved. Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice. Abstracting with credit is permitted, but copyrights for components of this work owned by others than COMAP must be honored. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior permission from COMAP.

Figure 1), according to throws of two dice. They buy and sell properties and collect rent from (or pay rent to) their fellow players. Properties are arranged into 10 groups: 8 color groups, Railroads, and Utilities. A player who owns all of the properties in a color group has a *monopoly* on that color group and can start building houses and hotels on its properties. (Owning the entire Railroad group or the entire Utilities group also constitutes a monopoly, although players cannot build on these properties.) Players are charged "rent" when they land on a property owned by another player. The amount of rent charged depends on the number of properties in a group held by the owner as well as on the number of houses or hotels.

The game ends when one player has bankrupted all of the other players.

As experienced players know, luck plays a substantial role in the game. Even expert strategies are sometimes confounded by the luck of the dice. Indeed, establishing the effectiveness of some strategies can take many thousands of games, as we will see. As a consequence, it may be very difficult for human players to draw useful conclusions about strategy from observations of their own games. The practical issue, of course, is that it takes a long time to play the game.

Figure 1 shows a diagram of the game board of the 2008 edition of the game Hasbro [2007]. (We note that in some earlier editions, the Brown group was colored Purple.)

Previous Research

Starting in the 1970s, mathematicians and computer scientists have used a variety of techniques to smooth the effects of random chance in Monopoly. Limits of Markov processes were used in Ash and Bishop [1972], Abbott and Richey [1997], and Collins [2005] to approximate the frequencies with which players land on the various board spaces. In Friedman et al. [2009], Markov chains, Brownian motion approximations, and computer simulations were used to estimate the probability that the game will never end. Computer simulations were used in Darzinskis [1975] to determine a variety of rules for effective play. In recent years, genetic algorithms have been used in Frayn [2005] and Mukhar [2013] to rank the properties and to approximate optimal strategy. There are other algorithms that could potentially be run on a Monopoly simulation, including Monte Carlo tree search, expectimax, bandit-based methods, etc. (See Brown et al. [2012] for a survey of these and other methods.)

Our Approach

We use computer simulations to analyze two-player Monopoly. The strategies that we study here are less complicated than those in Frayn [2005]

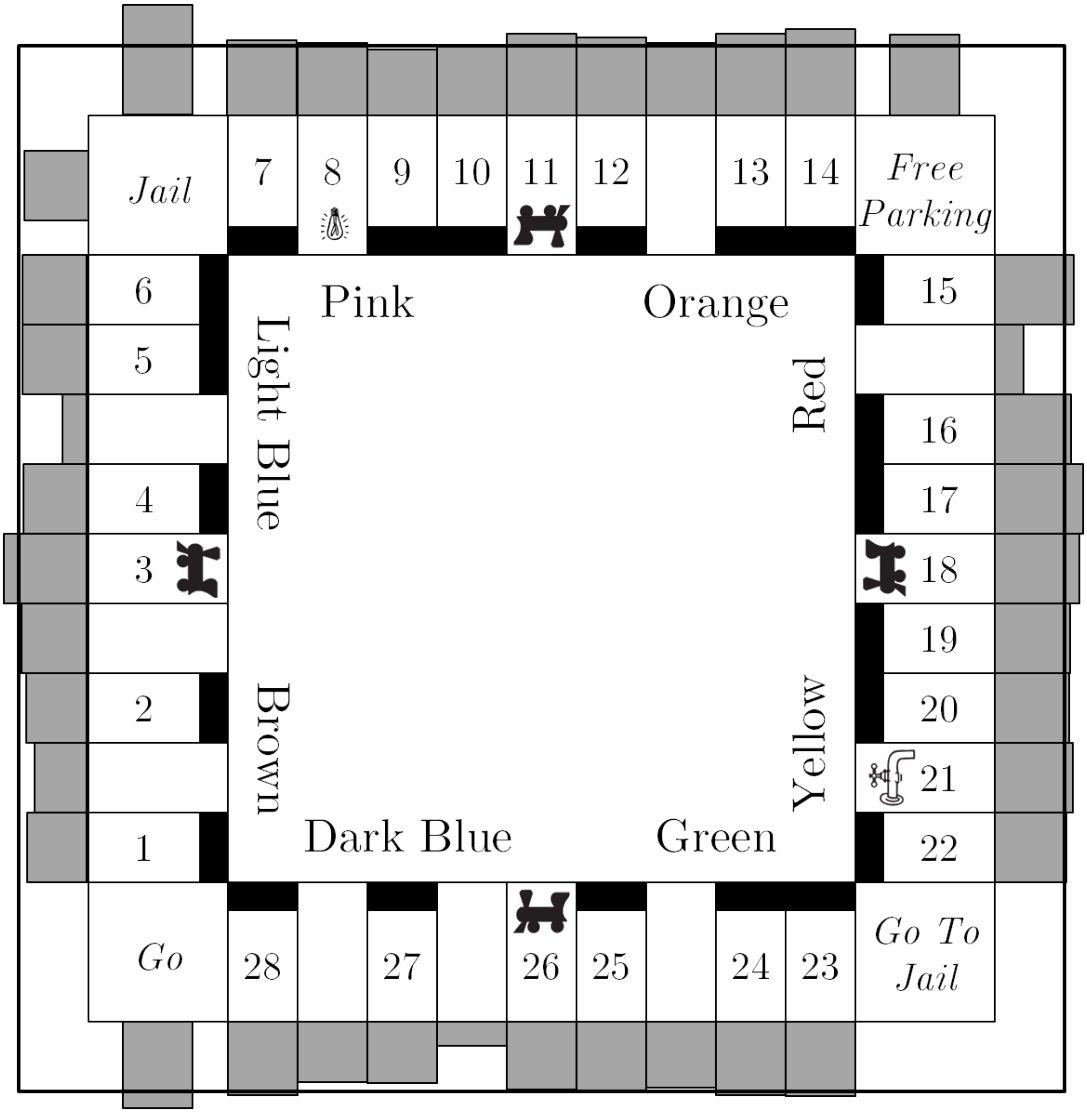


Figure 1. Diagram of the Monopoly board. The outer spikes indicate the relative frequencies with which players land on each property, assuming a long jail stay. The shorter spike on Jail indicates "Just Visiting" and the taller spike indicates "In Jail." The surrounding outer box marks the average frequency over all spaces.

and Mukhar [2013]. In contrast with the evolutionary approach in Mukhar [2013], for example, in which each strategy was determined by a chromosome of 160 real numbers, each of our strategies is determined by a set of 11 parameters (or fewer). Moreover, our parameters correspond directly to calculations or observations that could be approximated by human players. As a consequence, our results lead to concrete and useful recommendations for successful game play.

In a broader sense, we simplify a complex system that involves specific aspects of human decision-making. By doing so, we provide a context for novel applications of algorithms such as hill-climbing on graphs and Google PageRank.

We investigate and compare a variety of Monopoly strategies divided into two broad classes: dynamic and static.

- A *dynamic strategy* analyzes the current state of the board—primarily the rents that players expect to pay and receive over the next few turns—and makes decisions based on its net expected outcomes.
- *A static strategy* makes decisions based on fixed goals that are independent of the current state of the board; e.g., the goal of acquiring the Railroad group is more desirable than acquiring the Dark Blue group.

In other words, a dynamic strategy seeks to optimize short-term success, while a static strategy focuses on long-term goals.

The Plan of Attack

We begin by analyzing a simple class of static strategies. Each static strategy essentially corresponds to an ordering (permutation) of the 10 property groups in the game: Brown, Light Blue, Pink, Orange, Red, Yellow, Green, Dark Blue, Railroads, and Utilities. A strategy's ordering ranks the property groups according to preference. These preferences influence decisions about purchases, property development, and trades. We identify several effective orderings using a ranking scheme based on the Google PageRank algorithm and by running a hill-climbing algorithm on the space of all orderings. It is interesting to note that the preferred properties in the most effective orderings that we found are not always valued highly in other analyses (or in the minds of many human players).

Then we construct a dynamic strategy that utilizes an expected value computation to make decisions. The expected values are based on the rents that each player is likely to pay and collect during one circuit of the board. In these calculations, the rents are weighted by the probabilities that the players will land on the associated board spaces over the next few rolls of the dice.

Our primary results compare the strongest static strategies, the dynamic strategy, and two other static strategies suggested by results from Abbott

and Richey [1997]. We use two different schemes to rank the strategies, one based on success rates against a broad range of opponents, and the other based on head-to-head competition.

Finally, we use a composite strategy to determine if and when the short-term gains of a dynamic strategy outweigh the long-term goals of a static strategy. Our results strongly suggest that working toward long-term goals is far more important than maximizing short-term gains.

The details of our computer model, which was written in Python, can be found in the **Appendix**. All of the rules of the game are incorporated, but we make a few simplifying assumptions about how and when players make decisions.

Properties and Property Groups

Players can buy properties as they move around the board and land on unowned spaces. There are 28 properties, arranged into 10 "property groups." The properties, their groups, their board positions, and their prices are given in **Table A1** in the **Appendix**.

A player who owns all the properties of a color group has a monopoly on that color group and can begin building houses or hotels on its properties. Doing so causes a large increase in rent that must be paid by other players who land on the spaces.

Rent paid when landing on a Railroad is determined by how many Railroads the owner owns: \$25 for one, \$50 for two, \$100 for three, and \$200 for all four. If a player lands on a Utility owned by an opponent, the player rolls the dice and pays 4 or 10 times the value rolled, depending on whether the opponent owns one or two Utilities.

Without obtaining monopolies, a player is unlikely to bankrupt other players and win the game. Thus, a player's primary objective is to buy and trade properties in order to build monopolies. We therefore treat each property group as a single unit when it comes to creating and testing strategies. **Figure 1** shows the distribution of color groups on the game board.

Static Strategy

A *static* strategy is an 11-tuple consisting of a ranking of the 10 property groups and the maximum buying threshold, to be defined below. For example, one static strategy might have the following ranking, in order from most to least desirable (from rank 1 to rank 10):

Pink, Railroad, Utility, Green, Brown, Yellow, Dark Blue, Red, Orange, Light Blue.

A static strategy's ranking of the groups determines how it trades and spends money. We describe the details later.

In some sense, the static strategies described here are discrete approximations of the strategies in Frayn [2005]. Instead of assigning numerical values to the properties, we simply rank them. Since the outcome of the game is largely determined by which monopolies are formed, we further simplify the model by grouping the properties by color (the railroads are also grouped together, as are the utilities) and ranking the groups. As we will see, even this simplified model is sufficient to reveal significant and useful differences between strategies. Our simple model has the advantage that it allows us to replace the genetic algorithms used in Frayn [2005] and Mukhar [2013] with more-elementary techniques.

Trading and Buying

The scheme that we use for trades between static players is based on the following principles:

- Trades do not involve cash: Only properties change hands.
- Only one-for-one property trades are allowed.
- Any trade either gives both players a monopoly or neither player a monopoly. The idea is that no reasonable player would agree to a trade that gave the opponent a monopoly without receiving one in return.
- In every trade, both players think that they are getting the better deal according to their rankings. For example, if Player 1 prefers Green to Brown and Player 2 prefers Brown to Green, then a trade is possible in which Player 1 gives Player 2 a Brown property and Player 2 gives Player 1 a Green property. If both players prefer Green to Brown, then no trade is possible between these two color groups.
- Players do not break or trade away complete monopolies.

For simplicity and speed, we look for possible trades only immediately after a property has been purchased: If Player 1 has just purchased Property A, we search Player 2's inventory for a Property B such that the Property A $\leftrightarrow$ Property B trade satisfies the trading principles listed above. Searching for all possible trades on every turn would produce more trades, but this approach is considerably faster.

In addition to affecting trading decisions, the ranking in a static strategy also influences decisions about properties (buying a property, unmortgaging it, or building on it). To avoid being bankrupted, a player should keep some cash in reserve rather than have spent it all on properties. Hence, we associate a *buying threshold* to each decision involving a property—the amount of cash that the player should have on hand after executing the

decision. If the player would not have that much cash left, the player will not buy, unmortgage, or build on that property.

With a static strategy and a property in a group with rank r, we set the buying threshold for that property to be

$$\text{buying threshold for property with rank } r = m \cdot \frac{r}{10},$$

where m is the *maximum buying threshold* (MBT), a parameter of the strategy. The buying threshold is lowest for the most desirable properties (rank 1) and highest—equal to the MBT—for the least desirable properties (rank 10). This accords with the principle that a player should be more willing to purchase more-desirable properties. The basis for the mathematical formulation is that though a ranking does not quantify gaps between adjacent elements, it seems reasonable to impose a uniform linear relationship between the threshold for a property group and its place in the ranking.

For example, suppose that a static strategy has an MBT of $m = \$1500$ and ranks the Dark Blue group 5th. Then the strategy has a buying threshold of $m = \$1500 \cdot \frac{5}{10} = \$750$ for any property in the Dark Blue group, in particular for the Boardwalk property. If the player using this strategy has \$1000 in cash and lands on unowned Boardwalk, the player will not buy the property at the full price of \$400, because spending \$400 would drop the cash reserve to \$600, below $m = \$750$. In this scenario, according to the rules of the game, the property would go to auction, and the player would be willing to bid up to \$250 for the property.

The scheme described above is called the *step-wise buying threshold*. We find that **static strategies do significantly better with a step-wise buying threshold than with a fixed buying threshold** (see the **Appendix** for a description of a fixed buying threshold).

Hill-Climbing

An elementary approach to finding strong static strategies is to use a discrete hill-climbing algorithm.

As noted above, we encode each static strategy as an 11-tuple. The first 10 elements are a permutation of the 10 property groups. There are 10! such permutations. The 11th element is the player's MBT, which we allow to be one of \$500, \$1000, \$1500, or \$2000. Thus, the space of all static strategies has $4 \times 10! = 14{,}515{,}200$ elements.

The naïve approach would be simply to calculate a winning percentage for each of these strategies against a random set of opponents and choose the best. However, we found that to get statistically significant differences between winning percentages, we had to play each strategy against approximately 10,000 random opponents. On the hardware available (and with explicitly-programmed parallel processing), playing 10,000

games took about 6.4 seconds. At this rate, it would take approximately 3 years to do the complete computation.

Instead, we try to traverse the space of static strategies in such a way that our success rate consistently improves. We represent this space as a graph with $4 \times 10!$ vertices, each representing a strategy. Two vertices are connected by an edge if and only if one of the corresponding strategies can be transformed into the other by

- swapping two property groups in the ranking, or
- adjusting the MBT by $\pm\$500$ (to one of the other MBT values allowed).

We begin the hill climb at a random strategy and compute its winning percentage in 10,000 games against random (static) opponents. Then we randomly select a neighboring strategy and compute its winning percentage.

- If that percentage is worse than the initial strategy's winning percentage, we consider another neighbor.
- If that percentage is better than the initial strategy's winning percentage, we move to that strategy and consider its neighbors.

We repeat this process until we arrive at a strategy for which all of its neighbors have smaller winning percentages.

The Hill-Climb Strategy

After running this algorithm for about 18 hours and finding 60 local maxima, we identified a collection of strategies that won just under 82% of their games. Most of the following characteristics were present in each of these strategies:

- Railroad, Orange, and Light Blue come first, in that order.
- The 4th and 5th slots are Pink and Brown.
- The 6th slot is Utility.
- The 7th slot is Dark Blue.
- The last three slots are Red, Yellow, and Green.
- The MBT is \$1500.

Therefore, we can say that the optimal static strategy according to the hill-climbing algorithm has a ranking that is very similar to

Railroad, Orange, Light Blue, Brown, Pink, Utility, Dark Blue, Red, Yellow, Green,

and an MBT of \$1500. We call this strategy the *hill-climb strategy*.

Google PageRank

In this section, we describe two other approaches to optimizing a static player's ranking. Both are based on the public portion of the Google PageRank algorithm. Although this algorithm is typically used in a network setting, the linear algebra involved can be used more generally to rank any set of objects that can be compared.

Given objects and head-to-head comparisons between them, we generate a directed graph as follows. Imagining the objects as vertices in the graph, the head-to-head comparisons produce weights on the edges between each pair of vertices. These weights are rescaled and recorded as entries in a column-stochastic matrix A, i.e., a matrix with positive entries such that all of the columns sum to 1 [1]. Any such matrix is guaranteed to have eigenvalue 1. If the matrix has all positive entries, the eigenspace corresponding to eigenvalue 1 is one-dimensional and is generated by an eigenvector with all positive entries. Each entry in this eigenvector gives a weight to one of the objects that we are comparing, and these weights determine the ranking. For more details on this general ranking scheme, see Bryan and Leise [2006].

Our first computation ranks the property groups according to their power at the beginning of the game. We start by conducting head-to-head competitions between the groups. Suppose, for example, that we wish to compare the Red group to the Railroad group. We play 10,000 games in which Player 1 starts the game owning the Red group and Player 2 starts the game owning the Railroad group. Each player uses a static strategy determined by a random ordering (a new one for each game) and an MBT of \$1500. Suppose Player 1 wins 39% of these games and Player 2 wins 60% of these games (with 1% ties: games still active after 1000 turns are ended and labeled as ties). In our graph, we create an edge from the Red vertex to the Railroad vertex with weight 60 and an edge from the Railroad vertex to the Red vertex with weight 39.

After applying this process to every pair of groups, we have a graph with a pair of directed, weighted edges between each pair of vertices. These weights generate entries in a matrix, and the linear algebra described above produces a ranking of the groups. The resulting weights—rescaled between 0 and 1—are shown in **Figure 2**.

Thus, the ranking determined by this approach is

Railroad, Light Blue, Brown, Orange, Pink, Red, Utility, Dark Blue, Yellow, Green.

We can do a similar sort of analysis at the end of the game. In this case, the objects that we compare using PageRank are the individual properties.

[1] It can happen that some columns of the original matrix consist entirely of 0s and therefore cannot be rescaled to sum to 1. In this case, we get a column-*sub*stochastic matrix. However, this did not happen in our analyses.

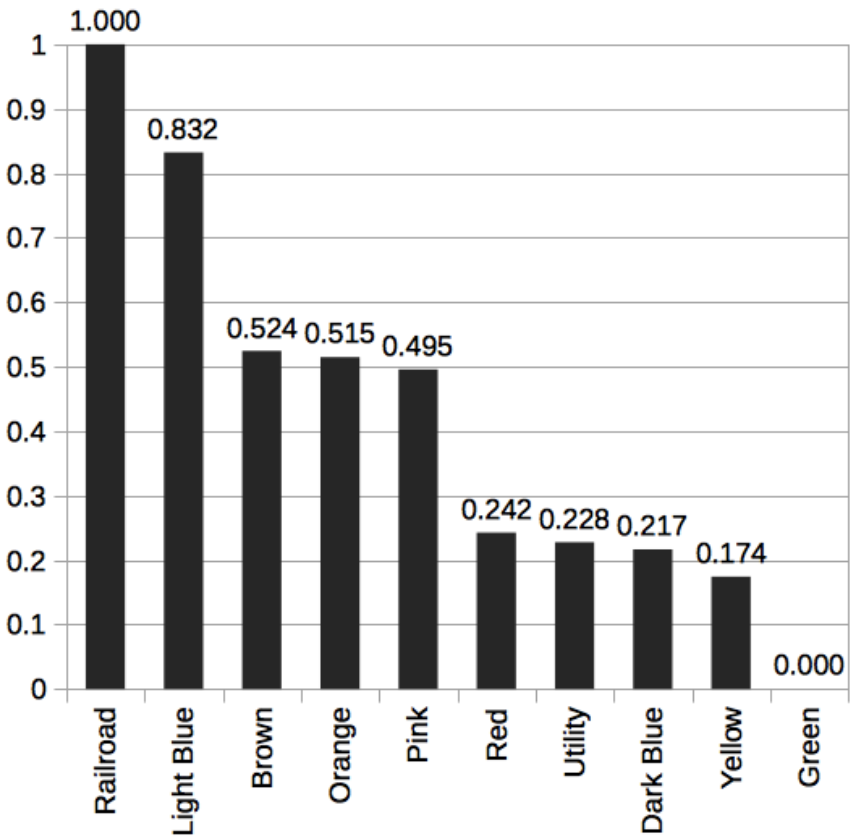


Figure 2. PageRank property group weights from the beginning of the game.

We use the same mathematics, but the edges in the graph—of which there are now $28 \times 27 = 756$—are determined by which properties have been acquired by the winners and by the losers.

We play 100,000 games between randomly-generated static strategies and record the following data in our graph. The weight on the edge pointing from Property A to Property B is the number of games in which the winning strategy had Property A in its inventory and the losing strategy had Property B. As before, the resulting graph produces a matrix, which in turn yields weights for each of the properties. These weights essentially measure the extent to which each property tends to be on the winning side. If we average the weights in each property group and rank the groups, we have the group weights shown in **Figure 3**, which yields the ranking

Railroad, Orange, Light Blue, Pink, Dark Blue, Red, Utility, Yellow, Green, Brown.

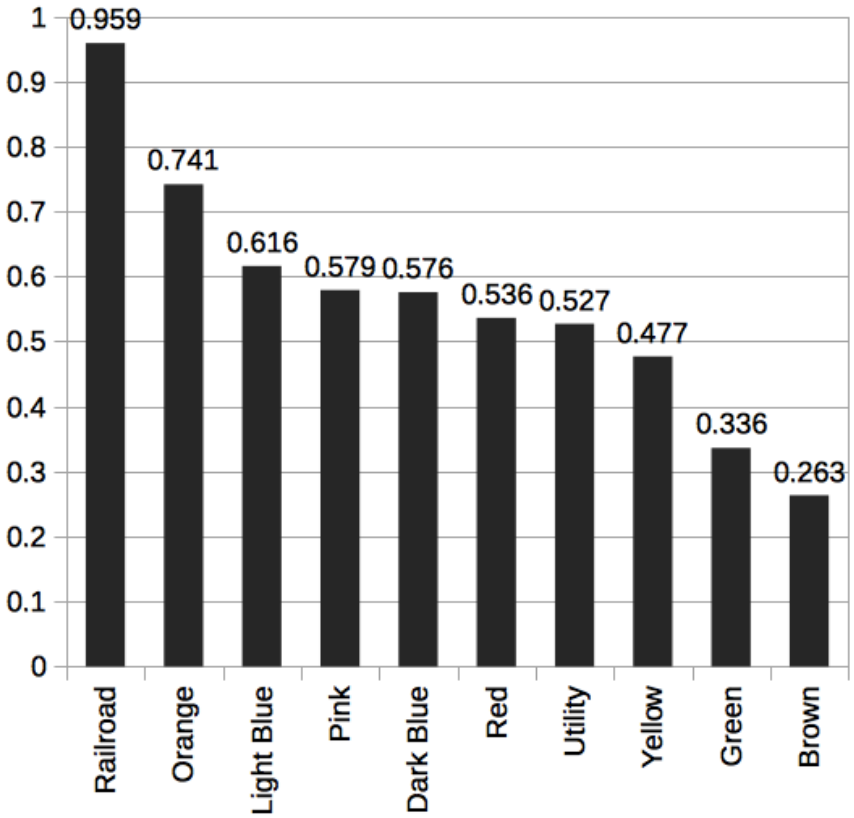


Figure 3. PageRank property group weights from the end of the game.

Dynamic Strategy

In this section, we construct a strategy that makes decisions based on the expected change E in money over the next six rolls of the dice, which move a player on average of roughly one circuit around the board. The expected change is based on incoming and outgoing rents. We call this strategy the *dynamic strategy*.

The rents for the properties are weighted by the probabilities that the players will land on them over the next six rolls. These probabilities are determined using a Markov chain-style calculation—i.e., by exponentiating a transition matrix. (We based our transition matrix on that in Collins [2005], which models the per-roll movement presuming a "long jail stay," an assumption that we make for all players.)

We modify rent values in some situations to reflect predictions about how players develop their properties (the rules provide for doubled rent on properties of a color for which the owner owns the entire color group). These predictions are based on how much money the players have and which complete groups they own. The dynamic strategy considering a trade knows how much money it has now and how much it will spend to develop any properties that it gets in trade. The expected value calcula-

tion can therefore predict exactly the extent to which properties received in a trade will be developed immediately, and those predictions are taken into account. Since the dynamic strategy does not know exactly how much money the opponent is willing to spend to develop a given property received in trade, it assumes that the opponent will spend all of their money on development (saving \$1 to stay in the game). No predictions are made about which properties will be purchased by either player.

The decisions dictated by these expected values are trading decisions[2]. As in trades between static strategies, trades between dynamic strategies consist of one-for-one property exchanges. They do not break complete groups; and if one player receives a complete group, then the other must as well.

If the dynamic strategy can expect its money to change by E over the next n rolls without doing the trade, and by E' if the trade is performed, then the *expected benefit* of the trade is $\alpha = E' - E$. The dynamic strategy will execute the trade with the maximum α such that $\alpha > 0$ and the trade is agreeable to the other player. This strategy will accept a trade offered if $\alpha > 0$ and decline otherwise[3].

Static Approximation of Dynamic Strategy

The PageRank algorithm (described in Bryan and Leise [2006]) can also be used to rank property groups that the dynamic strategy favors in trades to give an approximation of what we call the *effective ranking* of the dynamic strategy. The resulting ranking of property groups yields a static strategy that makes decisions similar to those of the dynamic strategy.

First, we play a large number of games between the dynamic strategy and a random population of static strategies[4]. We collect trading information to populate a weighted directed graph with 10 vertices, for the 10 property groups. The edge pointing from group i to group j is the number of trades in which the dynamic player gave a property in group i in exchange for a property in group j from its static opponent. This graph is then encoded into a 10×10 matrix, and the PageRank algorithm can be applied to produce a ranking value for each property group. **Figure 4** shows the resulting weights (rescaled between 0 and 1). Thus, the effective ranking of the dynamic player is

Railroad, Dark Blue, Utility, Green, Red, Yellow, Orange, Light Blue, Pink, Brown.

[2]Although it would also be reasonable for short-term expected values to affect a dynamic strategy's buying threshold, we found that our dynamic strategy did significantly better with a fixed buying threshold of \$1.

[3]Other expected gains and losses, such as getting \$200 for passing Go, would cancel in the expression $E' - E$, so they are excluded from our calculations.

[4]For a trade to take place, the dynamic strategy must see a positive expected benefit, and the static strategy must rank the property received above the property traded away.

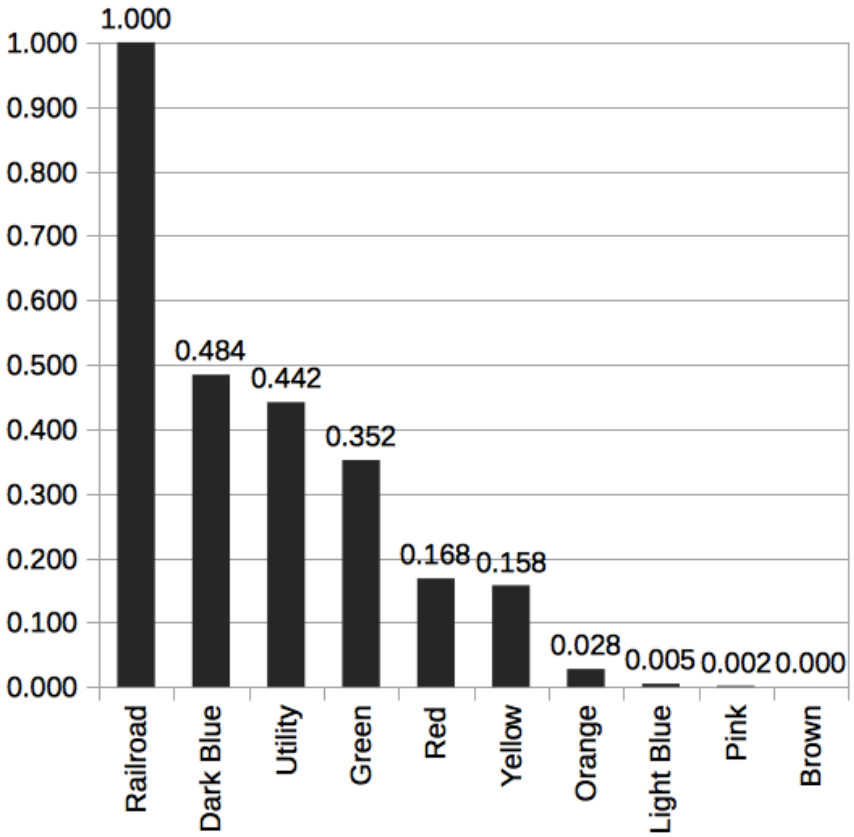


Figure 4. Effective ranking of the dynamic strategy.

Comparing Strategies

Our ultimate goal is to compare the effectiveness of a variety of human-usable strategies. This section presents the results of a number of trials using both kinds of strategies.

For a baseline comparison, we employ a simple strategy that we call *Newbie*, intended to approximate a novice Monopoly player. Newbie is a static strategy that ranks the property groups from most expensive to least expensive and has a fixed buying threshold of $500. Newbie is not a sophisticated strategy, but it does avoid making some especially egregious errors. For example, Newbie will not trade away or break a monopoly, and Newbie will not agree to a trade that gives its opponent a monopoly without getting one in return.

We also consider two static strategies that are based on results from Abbott and Richey [1997], who use a Markov chain analysis to calculate the average frequency of players landing on each property group. If one were to rely solely on this data, the property group ranking would be

Railroad, Orange, Red, Yellow, Green, Pink, Light Blue, Utility, Dark Blue, Brown.

They also calculate the expected value per roll of each fully-developed color group. The expected value per roll of the Railroads and Utilities can be calculated with the data given. If one were to rank properties based on expected value, the property group ranking would be

Green, Yellow, Red, Dark Blue, Orange, Pink, Light Blue, Railroad, Brown, Utility.

The static strategies that we compare in this section are summarized in **Table 1**. Newbie has a fixed buying threshold of \$500, while the rest of the static strategies use the step-threshold with an MBT of \$1500.

Table 1.
Static strategies.

Hill Climb	PageRank Start	PageRank End	Frequency	Expected Value	Newbie
Railroad	Railroad	Railroad	Railroad	Green	Dark Blue
Orange	Light Blue	Orange	Orange	Yellow	Green
Light Blue	Brown	Light Blue	Red	Red	Yellow
Brown	Orange	Pink	Yellow	Dark Blue	Red
Pink	Pink	Dark Blue	Green	Orange	Orange
Utility	Red	Red	Pink	Pink	Railroad
Dark Blue	Utility	Utility	Light Blue	Light Blue	Utility
Red	Dark Blue	Yellow	Utility	Railroad	Pink
Yellow	Yellow	Green	Dark Blue	Brown	Light Blue
Green	Green	Brown	Brown	Utility	Brown

We employ two different schemes for comparing strategies:

- The first scheme involves playing each strategy against a large random selection of strategies. The resulting winning percentages induce a ranking on the examined strategies.
- The second scheme involves playing each pair of strategies against each other over and over. The results of these head-to-head comparisons can then be used to generate a ranking.

Success Indicators

We define a player's *success indicator* as the percentage of games won after playing a 100,000 games against random opponents. The population from which these opponents are selected is the set of static strategies. Based on the results of the hill-climb, each of these static opponents uses a step-wise buying threshold with an MBT of \$1500.

Success indicators for all of the static strategies that we are comparing are shown in **Table 2**. Note that three groups of strategies emerge from this comparison scheme. The top three—Hill Climb, PageRank Start, PageRank End—are separated from the rest by a significant gap, and the bottom two—Newbie and Expected Value—are noticeably weaker than the rest. The middle two strategies—Frequency and Dynamic—are both ultimately based on the likelihood of landing on each property, so it is not surprising that their success indicators are similar.

Table 2.
Success indicators.

Strategy	Success Indicator
Hill Climb	79.6
PageRank Start	76.8
PageRank End	75.9
Frequency	65.8
Dynamic	61.0
Newbie	33.3
Expected Value	22.3

Head-To-Head Comparisons

Table 3 shows the results from a set of head-to-head comparisons. We played each pair of strategies against each other 10,000 times and recorded the number of wins for each strategy. For each pair (Strategy A, Strategy B), the entry in the row labeled Strategy A and column labeled Strategy B is the percentage of games in which Strategy A won against Strategy B. For example, the number 3.1 in the Newbie row and the Hill Climb column indicates that Newbie won just 3.1% of its games against Hill Climb.

Table 3.
Head-to-head results.

	Hill Climb	PageRank Start	PageRank End	Dynamic	Newbie	Frequency	Expected Value
Hill Climb		62	41	70	97	72	93
PageRank Start	38		43	64	97	73	94
PageRank End	56	56		65	95	68	91
Dynamic	29	36	34		75	40	81
Newbie	3	3	5	21		9	77
Frequency	28	27	32	59	91		89
Expected Value	7	6	9	18	23	11	

Applying to this matrix the linear algebra behind Google's PageRank algorithm (viz., normalizing the columns and finding an eigenvector with eigenvalue 1) gives the weights in **Table 4**. For comparison, we also include the success indicators for these strategies.

Table 4.
Head-to-head rankings.

Strategy	Weight	Success Indicator
PageRank End	1.00	75.9
Hill Climb	0.97	79.6
PageRank Start	0.90	76.8
Dynamic	0.72	61.0
Frequency	0.70	65.8
Expected Value	0.20	22.3
Newbie	0.17	33.3

These results indicate that **an optimized static strategy is much more effective than a dynamic strategy**. Although our dynamic strategy won against random static strategies about 61% of the time—and a fairly impressive 79% of the time against Newbie—our hill-climb strategy beat the dynamic strategy 70% of the time and beat Newbie in a whopping 97% of their games. The PageRank Start and End strategies also did very well against both Newbie and the dynamic strategy.

It is worth noting that the relation on strategies determined by head-to-head competition is not an ordering, because it is not transitive. For example, consider the static strategies—all with MBT \$500—determined by the property group rankings shown in **Table 5**:

In head-to-head competition,

- Strategy B beats Strategy A 58% of the time,
- Strategy A beats Strategy C 55% of the time, and
- Strategy C beats Strategy B 56% percent of the time.

In other words, $B > A > C > B$.

By contrast, the success indicator *does* provide a transitive relation (at least, up to the accuracy of the calculation) on the space of Monopoly strategies: Each strategy is mapped to a number between 0 and 1. For example, the success indicators for these three strategies are

- Strategy A: 49.3%
- Strategy B: 45.5%
- Strategy C: 51.0%

Table 5.
A 3-cycle of static strategies.

A	B	C
Dark Blue	Utility	Light Blue
Red	Green	Utility
Light Blue	Yellow	Yellow
Pink	Railroad	Brown
Railroad	Light Blue	Green
Utility	Pink	Railroad
Yellow	Orange	Red
Green	Brown	Dark Blue
Brown	Dark Blue	Pink
Orange	Red	Orange

Thus, Strategy C is the most effective of these three strategies, in the sense that it does the best of the three against a broad array of static strategies.

Composite Players

The static and dynamic strategies that we have constructed mimic some aspects of human decision-making. However, most human players would try to balance short-term and long-term strategy rather than relying on one or the other entirely. For example, a player might have the long-term goal of obtaining the Orange monopoly, but an analysis of the board might suggest that a decision running counter to that goal—e.g., trading away an Orange property—is likely to produce a large short-term gain. For a sufficiently large short-term benefit, it might be worthwhile to temporarily ignore long-term goals. In this section, we attempt to quantify how large a short-term benefit must be to justify overruling long-term strategy.

We call a strategy that balances static and dynamic decision-making a *composite strategy*. Composite strategies use group rankings to make some trade decisions and short-term expected values to make others. The *decision threshold*, a non-negative dollar value T, determines which of the two metrics is used to make a given decision:

- If the short-term expected benefit α of a trade lies between $-T$ and T, then the group ranking is used to make the decision.
- If α lies outside of this interval, the expected benefit is extreme enough to warrant ignoring the long-term strategy goals determined by the ranking, and E is used instead:

– If α is positive, the action is taken; and

– if α is negative, it is not taken.

(Buying decisions are determined using the step-threshold approach; thus, they are not influenced by T.)

Our goal is to search for an optimal T-value. A T-value that maximizes success would represent a threshold beyond which long-term goals should be temporarily ignored so as to maximize gains over the next few rolls of the dice. For this analysis, we fix a single group ranking—the ranking of our hill-climb strategy—and calculate success indicators for a range of T-values. Success indicators are plotted as a function of T in **Figure 5**.

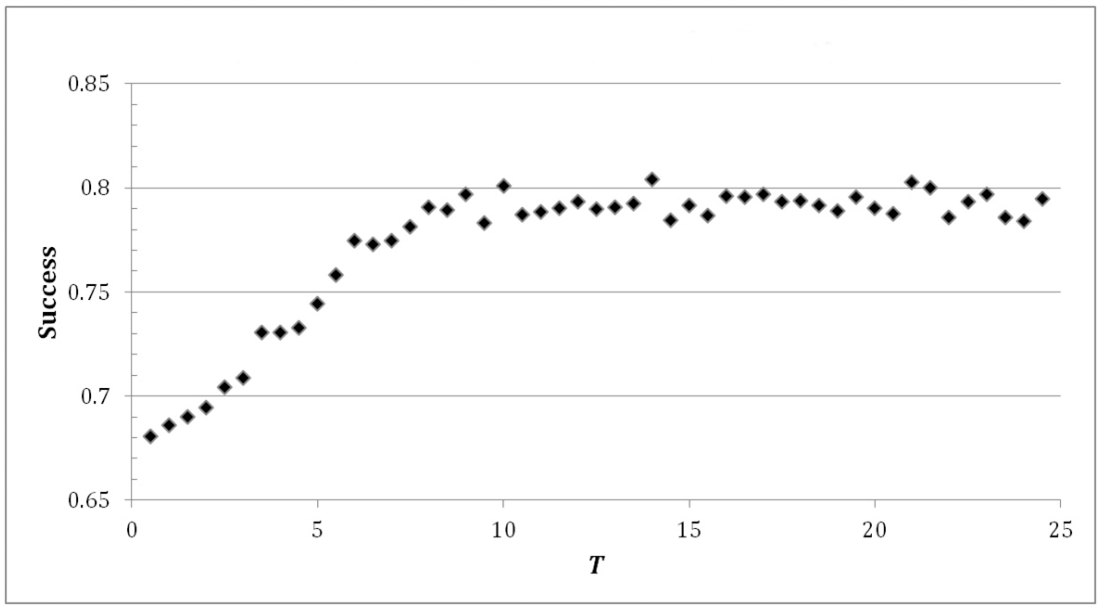


Figure 5. The success of a composite player as a function of T.

Figure 5 shows that success increases with T until T reaches about 10, at which point success appears to level off. More importantly, as T increases, the success rate of our composite player approaches the success rate of our best static strategy *from below*. In practical terms, this means that **(on average) there is no advantage to allowing the expectation of short-term gains to overrule long-term strategy**.

Implications for Practical Play

Our analysis suggests focusing on acquiring the Railroads above all, followed by the Orange and Light Blue groups, and deprioritizing the Red, Yellow and Green groups. In particular, the most successful strategy (at least in terms of the success indicator) is the static strategy with the ranking

Railroad, Orange, Light Blue, Brown, Pink, Utility, Dark Blue, Red, Yellow, Green,

together with a Maximum Buying Threshold of $1500. It is interesting to note that a player employing this strategy needs very little information about the state of the board to make decisions.

Of course, our model makes a few simplifying assumptions; and in real play, these priorities should be complemented by various short-term tactical decisions. For example, in real play, trades may take place at any time, not just immediately after properties are purchased, and careful timing of trades might allow an astute player to gain an advantage. However, the composite player analysis suggests that one should be very hesitant to make decisions that are inconsistent with the long-term goals determined by the group ranking.

Appendix

Constructing the Model

Our Monopoly model was written in Python 3.4.3 (the Anaconda 2.2.0 distribution). The code is available at

`https://github.com/RiponCollegeMaCS/Monopoly.git`

Games and players were represented as programmatic objects. The `Game` class runs game-play mechanics such as player movements, dice rolls, and trades and the `Player` class allows players to make strategy decisions, such as when to buy and build.

We adopted the official rules in the 2008 edition of the game Hasbro [2007] and followed them closely to construct the model. All established game-play elements, including trades and auctions were implemented. More modern (and optional) game-play elements, such as the "speed die," were not used. Where the rules allowed for differing interpretations, we followed the suggestions given by Orbanes [2007] and Darzinskis [1975].

The Buying Threshold

Each player's decisions about buying and developing properties are largely guided by a *buying threshold*, which is the amount of money that the player wishes to keep in reserve. This parameter is used in buying, bidding, building, mortgaging and trading. For example, if a player has $300 with buying threshold is $200, the player is willing to spend up to $100 to buy a given property. Thus, the player will not buy a $110 property and instead will put it up for auction. If the other player is willing to spend more than $100 on the property, that player will buy it at $101. If not, the first player will buy it at $100. Similarly, players will build houses and hotels

Table A1.
The 28 properties in Monopoly, organized into 10 property groups, with board position and purchase price.

	Brown	
1	Mediterranean Ave.	$60
2	Baltic Ave.	$60
	Light Blue	
4	Oriental Ave.	$100
5	Vermont Ave.	$100
6	Connecticut Ave.	$120
	Pink	
7	St. Charles Place	$140
9	States Ave.	$140
10	Virginia Ave.	$160
	Orange	
12	St. James Place	$180
13	Tennessee Ave.	$180
14	New York Ave.	$200
	Red	
15	Kentucky Ave.	$220
16	Indiana Ave.	$220
17	Illinois Ave.	$240
	Yellow	
19	Atlantic Ave.	$260
20	Ventnor Ave.	$260
22	Marvin Gardens	$280
	Green	
23	Pacific Ave.	$300
24	North Carolina	$300
25	Pennsylvania Ave.	$320
	Dark Blue	
27	Park Place	$350
28	Boardwalk	$400
	Railroad	
3	Reading Railroad	$200
11	Pennsylvania Railroad	$200
18	B & O Railroad	$200
26	Short Line Railroad	$200
	Utility	
8	Electric Company	$150
21	Water Works	$150

on complete property groups in accordance with their buying thresholds. A player's buying threshold can be no less than $1, because players need money to remain in the game.

Trading

After a player buys a property, the player can offer the property to the opponent for any of the opponent's properties. The opponent can accept or reject the offer. If accepted, the trading session ends. If rejected, the player can propose another trade. The order in which players traverse possible trade offers is based on their strategy parameters.

Building

The official rules allow players to "buy and erect [buildings] at any time." For efficiency, we allow players to build only between turns. The rules also state that if there is a limited supply of buildings, they should be "sold at auction to the highest bidder." To avoid this conflict, we allow the player who will be moving next to build first, followed by the player who will move after that, and so on. Unlike Frayn [2005], we do adopt the housing shortage rule, in which there are only 12 hotels and 32 houses available for

players to buy. If all houses or hotels are in use, players must wait until some are sold back to the bank before they can build on their properties.

Making Funds

If a player does not have enough money to pay a rent, the player will first mortgage properties not in completed groups. Then the player will sell any buildings back to the bank, at half their cost. Finally, the player will mortgage any remaining properties.

Jail

We assume a "long jail stay," meaning that players will attempt to get out of jail by rolling doubles before using a "Get Out of Jail Free" card or paying $50.

Ties

Games that are still active after 1000 turns are ended and labeled as ties. None of our trials generated a significant number of ties—always fewer than 1% of the games played—which fact suggests that this is a reasonable cutoff point.

References

Abbott, Stephen D., and Matt Richey. 1997. Take a walk on the Boardwalk. *College Mathematics Journal* 28: 162–171.

Ash, Robert B., and Richard L. Bishop. 1972. Monopoly as a Markov process. *Mathematics Magazine* 45: 26–29.

Browne, Cameron, Edward Powley, Daniel Whitehouse, Simon Lucas, Peter Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A Survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4 (1) (March 2012): 1–47; `http://www.cameronius.com/cv/mcts-survey-master.pdf`

Bryan, Kurt, and Tanya Leise. 2006. The $25,000,000,000 eigenvector: The linear algebra behind Google. *SIAM Review* 48: 569–581. `https://www.math.ksu.edu/courses/syllabi/math551/math551.s07/billioneigenvector.pdf`.

Collins, Truman. 2005. Probabilities in the game of Monopoly. `http://www.tkcs-collins.com/truman/monopoly/monopoly.shtm`.

Darzinskis, Kaz. 1975. *Winning Monopoly*. New York: Harper & Row.

Frayn, Colin M. 2005. An evolutionary approach to strategies for the game of Monopoly. In *IEEE 2005 Symposium on Computational Intelligence and Games*, edited by Graham Kendall and Simon Lucas, 66–72. `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.127.9609&rep=rep1&type=pdf`.

Friedman, Eric J., Shane G. Henderson, Thomas Byuen, and Germán Gutiérrez Gallardo. Estimating the probability that the game of Monopoloy never ends. In *Proceedings of the 2009 Winter Simulation Conference (WSC)*, 380–391.

Hasbro. 2007. Monopoly: Property trading game from Parker Brothers. `http://www.hasbro.com/common/instruct/00009.pdf`.

Mukhar, Kevin. 2013. Using evolutionary computing to develop game strategy for a non-deterministic game. Master's thesis, University of Colorado at Colorado Springs.

Orbanes, Philip. 2007. *The Monopoly Companion*. New York: Sterling.

About the Authors



Mitchell Eithun is a senior undergraduate at Ripon College, majoring in Mathematics and Computer Science. Aside from doing mathematics, Mitchell enjoys playing saxophone and clarinet and has published several pieces of music for handbell ensemble. He also makes memes for the Ripon College Math Club.



McKenzie Lamb is an Associate Professor of Mathematical Sciences at Ripon College. He earned his B.A. in mathematics at Beloit College and his Ph.D. in mathematics at the University of Arizona. He enjoys rock climbing and avoiding hard work by getting computers to do it for him. He is especially interested in using Monte Carlo simulations to analyze board games.



Andrea Young is an Associate Professor of Mathematical Sciences at Ripon College. She earned her B.S. in mathematics at Penn State University and her Ph.D. in mathematics at the University of Texas at Austin. While she enjoys playing board games, Monopoly is not one of her favorites; so she is happy that this project has resulted in strategies that will make the game shorter!