Depth in Strategic Games

Frank Lantz[*], Aaron Isaksen[†], Alexander Jaffe[‡], Andy Nealen[*†], Julian Togelius[†]

[*]NYU Game Center [†]NYU Game Innovation Lab [‡]Spry Fox

Abstract

This paper explores the question of whether it's possible to discover a well-defined property of game systems that corresponds to what game designers and players mean by the term "depth." We propose a measurable property of a game's formal system, which we call '*d*', that corresponds to the capacity of a game to absorb dedicated problem-solving attention and allow for sustained, long-term learning. To define this property we develop a formal model that measures how susceptible a game is to partial solutions under conditions of steadily increasing computational resources. We then sketch out several directions for using the model to investigate questions about the structural properties of games that produce these effects.

Introduction

Game designers and players often make reference to the concept of depth. This term has a broad, general meaning that expresses the idea that something is absorbing and profound. But there is a narrow application of the term which refers specifically to the formal system of strategic games. Our goal is to examine this particular meaning of the term as it is used to describe the abstract system of choices and outcomes within games of this type. In this context, depth refers to a game's capacity to provide a lifetime of study, learning, and improvement. A game with great depth is one that seems to unfold into an endless series of challenging problems and responds to serious thought by continually revealing surprising and interesting things to think about.

Games like Chess, Bridge, Go, StarCraft, Hearthstone, and League of Legends are able to absorb the dedicated efforts of a large community of expert players over many generations of serious competition and collaborative analysis while continually producing fascinating strategic problems. Does this capacity correspond to a property of game systems that can be objectively observed and measured? After all, there are many well-defined formal properties of game systems that can be analyzed and quantified, features such as state space and branching factor. We often talk about depth as if it were a property like this, but is it? Is there some precisely definable, objectively observable property of a game's underlying structure that allows it to exhibit this quality?

One challenge with the informal use of the word "depth" is that it is often used as a binary term, implying a quality that games either do or don't have. The real picture is much more complex; every game exists along a spectrum of depth. Moreover, the same game may exhibit different levels of depth in relation to different player communities, or in relation to the same community at different times. We are interested in examining depth as a quality that all games have to various degrees, understanding how this quality is related to a game's formal structure, and developing conceptual tools that allow us to explore this relationship with greater precision.

Depth is often referred to by game developers (Pulsipher and Others 2011; Kiley 2013; Ghostcrawler 2016) and in scholarly research (Browne 2008; Nielsen et al. 2015; Abbott 1975) but to our knowledge no attempts have been undertaken to make a thorough and rigorous investigation into the property to which it refers. The purpose of this paper is to lay the groundwork for such an investigation. We are attempting to establish a foundation, clarify the important questions, and suggest directions for further study. We are not at this time proposing final answers to the central question.

Goals and Clarifications

It is not our goal with this project to make normative claims about how games *should* be designed or what makes a *good* game. The term depth can be used casually as a general superlative but that's not the way we are using it here. We aren't claiming that this quality is the most important feature for judging a game's overall value; there are many ways for a game to be good that aren't related to the kind of depth this paper investigates. In addition, even though we are attempting to analyze precise and quantifiable properties of a game's formal system, we are not attempting to reduce aesthetic judgments to objective empirical claims. Instead, we wish to establish clarity regarding features of game systems which *can* be observed and measured in order to better inform aesthetic analysis and discussion. In this way, our project is analogous to research into color theory (Albers 1971), which doesn't claim to distinguish between good and bad paintings, but provides a powerful, consistent, fine-grained conceptual tool for artists and critics to use in analyzing painting's aesthetic qualities. We believe that this conceptual tool can be used by game designers to understand some of the effects that rule and parameter changes have on their games.

Similarly, we are not proposing a threshold that distinguishes "deep" from "non-deep" games. Our goal is to discover a formal property of game systems, which we refer to as d, which would be present in all games to different degrees. Our methodology can apply to a wide variety of games, from *Snakes and Ladders*, where there are no strategic decisions and therefore a minimal value of d, all the way up to the most complex strategic games played by humans, and beyond to hypothetical games more complex than anything humans are capable of playing.

We aim to develop a precise definition of d that is psychology-independent. It should not make special reference to how humans learn or what humans find interesting or challenging (Gobet, Retschitzki, and de Voogt 2004). We want to describe structural features of game systems that hold regardless of the particular details of human cognition, and understand how these features produce certain kinds of complex problems, similar to how we can measure the spectral distribution of light sources, and then see how their interactions produce specific visual effects (Albers 1971). These structural features would be consistent under any conditions and would remain true for any intelligent, problem-solving process, whether human, non-human, or mechanical.

We recognize that the *perception* of depth is a function of the interaction between a game's formal properties and the cognitive capabilities of the perceiving player. For example, young players and novices may subjectively experience a great deal of depth where expert players find little. However, in this paper we are interested in the other side of this interaction - what are the objective formal properties to which these different kinds of players are reacting?

To simplify matters we will limit ourselves throughout this paper to looking at two-player, turn-based games; the term "game" in this paper refers to this category. Nonetheless, we hope that many of our arguments and concepts will be valid for or easily adaptable to to other types of games, such as one-player games (many videogames fall into this category) or real-time strategy games. Also, for simplicity's sake, we consider the rules of the game to be fixed. For example, in a collectible card game (CCG) we would consider the addition of new cards, or the modification of existing cards, to constitute a new game requiring new strategies and therefore potentially giving a different value for d.

At this stage in our research we do not yet have a system built to evaluate our proposed technique. This paper documents our initial forays into calculating the d property and outlines our thoughts and justifications regarding the requirements of such a metric.

Outcomes

We believe that the model we sketch out in this paper can enrich and inform our discussions about game design, suggesting directions for design exploration and expanding our palette of analytical tools and techniques.

Beyond the practical applications of this project, there is the intrinsic value of pure knowledge. We want to know if there is some specific structural quality that is shared by Chess, Bridge, Go, and games like them, some recognizable topological feature of their systems. And if there is then we would like to illuminate it in order to see it, and talk about it, and reason about its ramifications.

In addition, we believe this project may provide some insight into more general questions about artificial intelligence and machine aesthetics. If we can describe what makes certain games interesting in this particular way then we may learn something about "interestingness" in general, about the relationship between information, systems, and meaning. Deep games require knowledge, explanation, creativity, and cleverness. Modeling depth may help us better understand how to think about these things.

Existing Properties to Consider

To begin, we want to consider existing well-defined properties of game systems such as state space and branching factor. Could it be that one of these features *is* the property we're looking for? Or some simple combination of the two?

State Space

State space is the set of all possible arrangements of a game's elements. The set of valid arrangements which could be reached in actual games through the legal operation of the game's rules is a subset of the game's state space called *state space complexity* (Allis 1994). A related concept is *game tree size*, which refers to the total number of leaf nodes in the game tree. The game tree can be larger than the state space because one position can occur in many different branches.

Intuitively, any game that humans consider to be very deep will have an sufficiently large state space. However, state space by itself cannot explain the qualities of depth that we are looking to isolate. To see why, imagine any existing game and then add a new rule that allows either player, after their turn, to flip a token from side A to side B. This immediately doubles the size of the game's state space without affecting its depth in the slightest.

This thought experiment shows how one could artificially inflate a game's state space without increasing its depth at all. While it may be a necessary component of d, state space by itself cannot be sufficient. What matters is *meaningful* state space. The question of what makes state space meaningful in this way is exactly what our project seeks to understand.

Branching Factor

A game's *branching factor* (or average branching factor) refers to the number of choices presented to a player at each point in the game tree. Intuitively, this seems like it could be an important ingredient of strategic depth. One of the qualities of a deep game experience is the challenging process of weighing one's options and considering possible moves. A game with fewer options to consider would seem to offer less challenge and interest.

However, once again we can imagine inflating a game's branching factor artificially by adding any arbitrarily large number of "dud" alternatives at each node of the tree. Every dud branch leads in an obvious way to a losing position. In this way we can increase the branching factor as much as we want without affecting the game's depth.

Conversely, it's easy to imagine two games, one that presents the player with a highly-interesting binary choice at each node; the other that presents a large number of choices, one of which is obviously superior. Intuitively, the first game is much deeper, despite having the substantially smaller branching factor.

While it may be strongly correlated to d, branching factor cannot, by itself, be d. Nor, does it seem, can any combination of state space and branching factor by themselves. Again, it is a particular kind of *meaningful* state space and branching factor we are looking for.

Computational Complexity

Computational complexity is a concept used to describe how difficult a problem is to solve algorithmically (Papadimitriou 2003; Cormen et al. 2001). Traditionally, when applied to games this technique involves transforming the game system into a generalized problem and then measuring how the solution difficulty increases as that problem is made larger by scaling up variable quantitative factors such as size of board, number of pieces, etc. The difficulty is measured in terms of the amount of computational resources required: the time and/or (memory) space needed to find the solution. The result of this analysis is to place the game into one of several large classes of problems.

Computational complexity is a key concept for our project. Understanding depth in games means understanding them as problems and observing the specific nature of the challenges they present and the kind of solutions they demand.

However, classical computational complexity by itself can't serve as a proxy for d. For one thing, the traditional categories into which it classifies games are far too broad to be useful for our purposes. We are interested in making fine-grained distinctions between similar games. Knowing whether a game is in PSPACE or EXP, for example, doesn't give us enough detail about the game to help us distinguish it from the myriad other games in the same category.

Moreover, we want to know more than just how hard a game is. We want to understand the particular *way* in which a game is hard. A problem can require lots of resources to solve without being interesting. Searching for a needle in a haystack is a difficult problem (in the sense of being resource intensive) but it's not the kind of problem that requires cleverness and creativity, the kind of problem that rewards life-long learning and can support a large, long-term community of serious, dedicated players. Those are the features we are interested in explaining. Looking at the game system as a problem and thinking about the type of algorithm needed to solve it is a powerful tool for this project, but traditional computational complexity by itself can't be the complete answer.

Skill Chains and Strategy Ladders

So what exactly are the game features that indicate depth? Can we get more precise about them?

A key concept in addressing this question is the idea of the skill chain, as articulated by Bill Robertie under the term *complexity number* (1992), and later expounded upon in Characteristics of Games (Elias et al. 2012). The size of a game's skill chain is the number of distinct steps in the ranking of all players, where players at each step beat all the players at lower steps some significant percentage of the time.

This concept is important because the presence of a skill chain with a large number of distinct steps is evidence of a game in which a player can improve through study, a game in which the more you think about it the better you get. Players of a game like this are on a journey of gradual and continuous improvement, ascending ever-upwards towards better understanding and stronger play. Trivially easy games can't support long skill chains, and neither can all-or-nothing puzzles, no matter how large and difficult they might be.

The Strategy Ladder Model

In order to make this quality more precise and measurable, we propose a formal model called a *strategy ladder*. Whereas a game's skill chain consists of a sequence of human players of ascending skill, a game's strategy ladder consists of a sequence of algorithms called *strategies*. Each strategy is a complete set of instructions for how to play the game. The more complex and powerful these strategies become, the more computational resources they require to execute – time and/or memory - and the better performance they exhibit compared to simpler strategies. There are several advantages to study algorithmic strategies instead of human players, including (1) algorithms are repeatable, testable, and measurable, (2) algorithms can be fully examined while human players can't always describe how they are thinking, (3) human players change over time and are not static in their tactics or performance, and (4) the distribution of possible strategies is likely different than those used by human players influenced by community, opponents, and conventions.

We can frame d as the capacity for a game system to allow for a ranked population of strategies that provide partial/approximate solutions. The more discrete ranks in such a population, the greater the degree of d within the game. In other words, the more d a game has, the more it is susceptible to a sequence of partial solutions, each one requiring more time and/or memory to execute, and providing a better approximate solution to the problem of the game.

In Figure 1, we see three example games with different strategy ladders. Each dot represents a complete, fully-defined algorithmic strategy for playing a particular game represented by each path. Each dot is the *best* strategy that can be achieved at that level of computational resources (a thorough definition of "strength" and "computational resources" is given below). On the left we see a game that yields quickly to perfect play. On the right we see a game that requires a great deal of computational resources to play perfectly, but has only a few intermediary strategies along the way. In the center we see a game that requires a great deal of computational resources to play perfectly, and also allows for many intermediary strategies along the way. In this game, adding more resources immediately and continuously results in improved performance, allowing for many discrete stages of incremental improvement.

For any game of reasonable complexity the practical difficulty of building an algorithmic strategy ladder of this type would be insurmountable, as the model assumes both com-

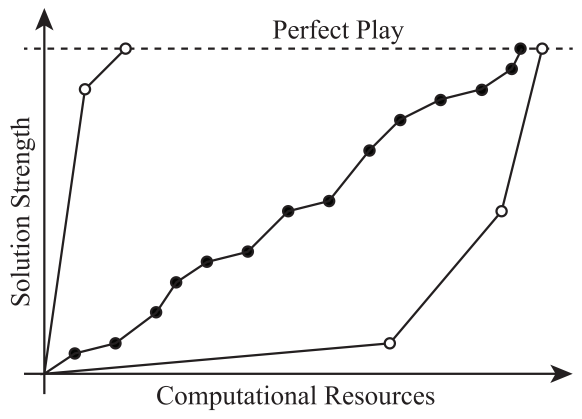

Figure 1: Strategy ladder steps and solution strength, as function of computational resources

plete knowledge of perfect play and a thorough comparative analysis of a sufficiently large sample of all possible strategies. Nonetheless, we believe this model can serve as a powerful conceptual tool for clarifying our thinking about the topic of depth. Furthermore, by building strategy ladders for simple games and observing how they respond to changes in the game's structure we hope to observe patterns that can be extrapolated to provide general insights about larger games as well.

The concept of the strategy ladder differs from the concept of the skill chain by highlighting how strategies with limited computational resources are *necessarily* imperfect, even for games of perfect information. Each dot in the model represents the *best possible strategy* available at that computational resource level. Even though the algorithms that populate the strategy ladder play the game sub-optimally, they all exhibit the strongest play for the resources available to them.

What is a Strategy?

In the strategy ladder model each algorithmic strategy is a complete set of instructions for playing the game. Each strategy determines how to play in every situation. They say when and how to look step-by-step through the branches of the game tree, when and how to evaluate board positions or apply other strategic rules of thumb, when to make a choice randomly or semi-randomly, when to follow a pre-determined sequence of moves, etc.

Before creating a sequence of algorithmic strategies that could be evaluated in a strategy ladder model one must first specify a language in which each of these algorithms is expressed. Because the choice of language will affect the possible strategies, any observations made about a game's depth based on this model must refer to the language selected.

To have the strategy ladder model applied to it, a game must have a well-defined time limit (on a reference machine). To be a valid strategy an algorithm must be able to output legal moves within the specified time-limit, given the computational resources available.

What are Computational Resources?

A specific level of computational resources is defined in terms of three factors: *speed* (operations per second), *size* (amount of memory available to store the algorithm and its data), and *working memory* (amount of memory available to the algorithm for use in carrying out its operations).

Any application of the strategy ladder model must specify how levels of computational resources ('CR-levels') are defined, and how many distinct levels to include. One approach would be to set the CR-levels needed for random-legal play and perfect play as lower and upper bounds, then divide the CR spectrum into, say, 100 evenly-distributed segments between them. On the other hand, it could be useful to observe how the strategy ladder changes when one or more CR factors are kept fixed while others increase.

What is Strength?

The strategy ladder looks at the relationship between a sequence of strategies, each of which is the strongest strategy at a particular level of computational resources. We refer to the strongest strategy at a given resource level CR_n as $A(CR_n)$.

The strength of a strategy refers to the quality of its performance along a spectrum that ranges from poor (performing no better than random legal moves) to perfect (optimal play from any position). We consider two primary ways to define this spectrum: *win rate* (WR) or *quality of move selection* (QM). Win rate looks at the results of a series of games in which the strategy is pitted against some set of other strategies at its own computational resource level or below. Quality of move selection looks at the frequency with which the strategy selects the best move, specifically how often its move corresponds to that selected by the perfect strategy.

Each of these methods has advantages and disadvantages. An approach based on win rate is especially sensitive to the question of which strategies it competes with to derive that win rate. Including every possible strategy adds yet another practical impossibility into the model. It also creates a situation in which an exploitative strategy might have a high overall WR because of its spectacular results against many weak strategies, even though it performs poorly against the other strongest strategies at its CR-level. Win rate can also be intransitive: strategy A can win against strategy B which wins against strategy C which, in turn, wins against A. Win rates are also highly dependent on factors such as game length, and can change based on game session duration or number of rounds. Finally, games with random elements have a looser correlation between strategic decisions and game outcome, which can obfuscate the signal win rate gives us about strategic strength. None of these problems are intractable, but they give some indication that win rate is not an unproblematic criteria for strategic strength.

A quality of move based approach can be more simply, clearly, and consistently defined. Another advantage is that QM offers a metric that increases as strategies gain access to more computational resources and improve in strength whereas, in most cases, WR does not. However, this approach has its own problems. A QM-based criteria limits our ability to apply the model in partial ways to games for which perfect play is unknown. It would also allow for counter-intuitive situations in which, for example, a strategy could have an extremely high rate of best move selection, but a very low win rate (because of some fatal weakness in end game play). More generally, a QM-based approach would force us to sometimes select a strategy as "best" even when there is another strategy that beats it most of the time. To the degree that the strategy ladder model is meant to capture and refine our intuitions about human players and real-world games, these issues make quality of movement selection highly problematic.

Suffice to say, the issue of how best to define strategy strength within our model is an unresolved question. Our belief is that we can defer this issue for future study. It seems likely that any reasonable method for determining strategy strength would yield *mostly* similar results, and we can proceed to use the strategy ladder model by assuming a clear and consistent method for determining strategy strength without yet knowing what that method will be.

What are Steps?

The final requirement for specifying a strategy ladder model for a particular game is to define some degree X of difference between each discrete "step" of strength level. However one ultimately decides to define strength, a new step is achieved whenever the best strategy at the current CR-level demonstrates improvement of X degree over the strategies of the previous step. Traditional skill chain models typically use some version of a win rate of 60-75 percent. As with all of the parameters that go into the strategy ladder model, it's less important to have a universal definition of step that applies across games than to have a well-specified definition within each application of the model.

The final output of the skill chain model is the number of these steps. The more steps in the chain, the more the game under consideration exhibits the characteristic of yielding a steady series of partial solutions as computational resources are increased, and therefore the more it demonstrates the quality we are looking for, the quality we call d.

Games where perfect play can be achieved at low levels of CR don't have sufficient capacity to support a high degree of d, and games that don't have interim strategies that produce significant jumps in performance also fail to exhibit this quality by not allowing incremental improvement.

Rolling Out a Strategy Ladder to Measure *d*

Putting these elements together, we can outline the steps one would go through in order to apply the strategy ladder model to a specific game:

1. Specify a complete definition of the game, including time limit.
2. Specify a language used to express algorithmic strategies.
3. Specify CR-levels by weighting or fixing constant each of:
   - operations per second
   - size of algorithm
   - working memory
4. Specify a performance metric for determining strategy strength (e.g. WR win rate or QM quality of move selection)
5. Specify a degree of strength increase to count as a step unit.
6. Find the best strategy at each CR-level, $A(CR_n)$.
7. Plot each $A(CR_n)$ as a point on a graph with computational resources as one axis and strategy strength as the other.
8. Start with $A(CR_1)$. Move to the next point on the curve, which is $A(CR_2)$. Compare it to the strength of $A(CR_1)$, if the difference in strength is a step unit or more then this point constitutes *step one*, otherwise proceed to $A(CR_3)$ and compare *it* to $A(CR_1)$ and so on until a point is found whose strength level is a step unit or more greater in strength than $A(CR_1)$. Whenever step one is found, start the process over with *that* point, looking for the next point at which the difference in strategy strength meets or exceeds a step unit, this point will be *step two*. Keep going until you reach the furthest point on the curve, which represents perfect play. The number of steps you have counted is the d for the this game *for the given settings*.

Note: Step 8 above presents one proposed way of measuring d, however other possible methods exist. For instance, one could count *only* the number of steps which occur between two adjacent points, thereby capturing how often a small change in computational resources produces a large change in strategy strength.

In either case, the main idea of the model is this: *the shape of the curve reveals something important about the game's structure*. The shape of the curve reflects how the game responds to the incremental application of computational resources.

Practical Limits and Theoretical Applications

As described, the complete model assumes knowledge, not only of a strategy for playing perfectly, but also of the minimal computational resources needed for such perfect play. This makes the model impossible to apply completely to complex, real-world games. Nonetheless, in our view, the model still has substantial value in the following ways:

- as a conceptual tool which allows us to speculate about what a complex game's algorithmic strategy ladder would look like *if* the model were applied completely
- as a practical tool for looking at the characteristics of a complex game's strategy ladder at the lowest CR-levels (by applying a partial version of the model)
- as a practical tool which can be applied completely to smaller games for which solutions and computationally-efficient strategies are known to some substantial degree of certainty (Silva et al. 2016)
- as a research tool for discovering the qualities of rule sets that *result* in depth, by finding correlations in small games, which persist as size increases.

Reverse Complexity

In some ways, the strategy ladder model we are proposing presents an inverse of the standard method of determining a game's computational complexity. Instead of looking at a particular game as one instance of a general class of problems and seeing how the computational requirements for solving that problem increase as the problem scales up, we analyze a game as a specific problem of a fixed size and observe how optimal strategies for solving this problem increase in effectiveness as computational resources scale up.

Search and Heuristics

The strategy ladder model allows us to observe directly, or speculate about with greater clarity, the *consequences* of depth, the *evidence* that a game has a high degree of this quality we call *d*. But can we use it to develop a greater understanding of the underlying structural properties of a game system that lead to it exhibiting these characteristics? Can the strategy ladder model help us explain why some problems lend themselves to partial and approximate solutions and some don't?

To begin with, let's consider the counterfactual, how might a game *fail* to produce many steps in our model?

We know that every game of the type we are looking at has some theoretical perfect strategy, a strategy that tells you exactly what move to make at any point. In fact, we know how to produce this solution in theory: you map out the entire game tree, find a winning node and work backwards looking for a path that your opponent can never force you off of.

Imagine a game in which this were the *only* way to play. Whatever your position, the only way to do better than choosing a move at random is to look ahead in the game tree, calculating move by move until you see the necessary outcome of each move. A game of this type would only have one type of strategy: raw look-ahead calculation.

Games of this type would display an increase in strategy performance at higher CR-levels but, intuitively, we can see that such games would allow for a much *smaller* degree of performance increase. Search isn't useful if there are no evaluation functions to tell us which moves to prefer. Strategies at lower CR-levels in such a game would be making random moves up until the point, late in the game, when they could start seeing win/loss outcomes at the end of the branches they were searching. There simply aren't enough different *types* of strategies to allow for a diverse, well-populated strategy ladder in such a game.

What's missing from this type of game is *heuristics* – the rules of thumb that players use as a shortcut through the intractable problem of fully searching the game tree. Here are examples of heuristics from various real-world games:

- Chess: *control the center*, *avoid doubling your pawns*
- Go: *hane at the head of two stones*, *avoid making empty triangles*
- Poker: *loosen up on the button*, *avoid cold-calling raises*

Heuristics take advantage of regularities in the game tree to guide the player towards areas of state space where winning paths are statistically denser. It's not true that every Chess position in which you control the center squares leads to a win, but it is true that, on average, these positions contain shorter and more certain paths to winning outcomes. In a sense the heuristic operates as a kind of compression function on the game tree – a map that reveals structural features in the underlying tree. A blurry, tattered map, but a map nonetheless.

In fact, in some ways, heuristics simply *are* the partial strategies we are talking about when we talk about the incremental solutions that comprise the strategy ladder. The more heuristics a game can support the more you can combine them into better and better strategies and the more you can improve with effort and study. In order to have a significant degree of *d*, a game must have a decision tree with some structural regularities that allow for the compression that heuristics provide.

But we can also imagine a game that fails in the opposite way. Consider a game in which a single heuristic allowed you to play perfectly. In such a game you would never need to search the game tree at all; whatever the situation, you would simply apply this one weird trick. This is how a game can fail to support a long strategy ladder by having a simple strategy that exploits an underlying regularity of the game tree to make the problem trivial.

Both of these theoretical depth-lacking games – one requiring pure search and one that allows for one simple heuristic – share an interesting property. In both cases the *description* of how to find the best move is short. In the first game, the description is "search the entire game tree and find the winning move", in the second game the description is "always move the smallest piece one square forward" (for example). Even though the process indicated by the first description is impossibly difficult and that of the second trivially easy, the descriptions themselves are both *short*.

But in deep games the best description for finding the best move is long and complicated. It will require a mix of heuristics (all things being equal, you usually want to move the smallest piece one square forward) and raw look ahead (actually in this position you need to move your largest piece backwards because otherwise this bad thing will happen).

We believe the quality we are looking for is a kind of semi-orderedness of the game's state space, a level of underlying structure that allows for some useful compression of the game tree but not so much that pure search is never needed.

We can frame this in terms of entropy. The best way of looking for something, like a best move, depends on how ordered the space you are searching through is. If the space is completely disorganized then one is forced to examine every possibility. This corresponds to high entropy. If the space is well-organized, precise instructions indicate where to look. This corresponds to low entropy.

The description of the best way to find something in a semi-ordered space is long and complicated. Deep games are semi-ordered. This is what the literature that a deep game produces is made of, all the strategy books and blog posts and training videos and forum arguments are part of this lengthy description of the most efficient method for finding the best move in a semi-ordered system under conditions of computational constraints.

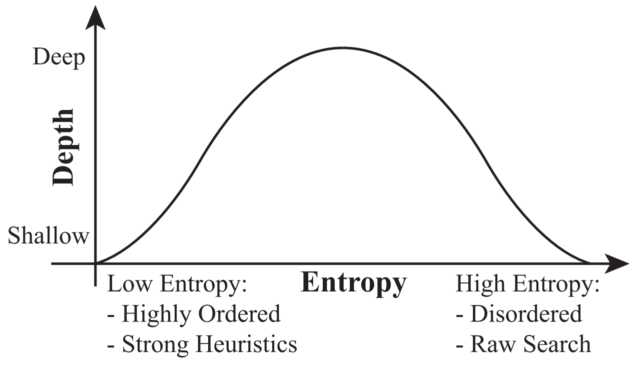

Figure 2: Maximizing depth requires a balance between heuristics and search.

Kolmogorov Complexity

In algorithmic information theory there is a precise term for referring to the size of the smallest algorithm that produces a desired output: Kolmogorov complexity (Li and Vitányi 2009; Kolmogorov 1968). By including algorithm size as one of the factors that determine CR-levels, the strategy ladder model presents a method for investigating how Kolmogorov complexity of optimal strategies is related to the concept of strategic depth.

Picture a "worst case" heuristic-resistant game, with no regularities that can be exploited as efficient shortcuts around pure search. Even in such a case, extra memory can be used as a *substitution* for operations by pre-computing a segment of the game tree and encoding it as a lookup table, in effect substituting size for speed. This global, base-level ability to transmute speed into size means that even a "search-only" game would make full use of all computational resources available to it - including both operational speed and algorithm size. However, we would expect a semi-ordered, non-search-only game to make *better* use of its combined resources by using algorithm size to gain performance increases that outpace the benefits gained from standard size for speed substitution.

One way we might apply our model to illuminate this topic is by taking a game and observing its strategy ladder under two sets of conditions:

- Speed Only: CR-levels have a fixed algorithm size and increasing speed
- Size Only: CR-levels have a fixed speed and increasing algorithm size

What would it mean if a game exhibited a substantially longer strategy ladder under the Size Only conditions? It would seem to suggest that the bigger strategies in this ladder are taking advantage of regularities in the game's structure to gain performance improvements that surpass the improvements gained by the faster strategies in the Speed Only model. An experiment of this type would allow us to directly observe how the incremental strategies of a deep game encode knowledge about that game's structure.

Search as Heuristic Arbitration

What about a situation in which a game's algorithmic strategy ladder is populated by strategies that don't use search at all, strategies that are just collections of heuristics? In any such case, there needs to be a way for the strategy to choose *when* to apply one heuristic vs another. This method itself must be some form of search. The source of a heuristic's efficiency is that it bundles together responses to many different situations that share some underlying structural similarity but are not identical. At some point, you can't improve this efficiency by slicing the situations into finer-grained categories; eventually one must use pure search to decide which heuristic to apply.

Search vs Heuristics in the Real World

If our approach is, in fact, the right direction for thinking about this issue then playing a deep game will involve a complex dance between heuristics and pure search. And sure enough, listening to the real-time thought process of an expert player often reveals just this – the compressed knowledge of proverbs, patterns, and rules of thumb alternating with periods of raw, if/then, move-by-move calculation.

Consider also the "killer move" – the unexpected, spectacular, high-impact play. What makes a killer move surprising and impressive is the way it deviates from a strong heuristic, indicating that heuristic's limitations. See, for example, Go expert Michael Redmond doing a triple take in response to AlphaGo's 4th line shoulder hit in the AI's second game against Lee Sedol (Silver et al. 2016; Redmond and Garlock 2016). The reason this move was so surprising was that it directly contradicts a well-known Go heuristic. Killer moves demonstrate how raw search can outperform even the most powerful rule of thumb.

Speaking of AlphaGo, it is interesting to note that the architecture of the world's best game-playing AI reflects this same search/heuristic balance. Its Monte Carlo tree search uses heuristics to channel the raw power of pure look-ahead, efficiently searching the game tree for directions with the greatest average value. Meanwhile, its board-evaluation functions are heuristics baked into its neural net, structural patterns discovered through millions of real and simulated games.

The human brain itself may demonstrate a similar structure. In Thinking Fast and Slow psychologist Daniel Kahneman (2011) outlines the two main modules of human cognition – System 1 (snap judgments, intuition, reflex) and System 2 (slow, step-by-step, logical calculation.) System 2 corresponds to the theoretically perfect but impractically slow process of pure search, while System 1 corresponds to the compression of reason into powerful but flawed heuristics.

Conclusion

In this paper we set out to examine the question of depth in strategic games. Can we define a property d that is precise, objectively measurable, and corresponds usefully with the concept of depth as it is broadly understood by game players and designers?

We considered existing concepts for measuring game size and complexity and discussed why, while they may be useful ingredients for constructing such a property, they could not by themselves suffice for our purposes.

We drew inspiration from the existing concept of the skill chain and then developed a new model that uses concepts from computational problem-solving to achieve a greater degree of precision and flexibility - the strategy ladder. We defined the strategy ladder model in detail, and suggested ways to apply it.

We then used our new conceptual model to speculate about questions of game structure, developing some initial thoughts about how the formal systems of deep games exhibit a semi-ordered structure that allows for strategies that balance pure search with powerful heuristics.

Next Steps

The project we've outlined in this paper is a substantial ongoing research program. What we have tried to do here is lay the foundation for this research, clarify the central questions, and define key terms. Our ongoing research agenda for this project includes the following:

- further refinement and clarification of the strategy ladder model
- development of "toy games" that illustrate key ideas of the model (all-search/no-heuristic games, games with the same game tree complexity but amounts of d)
- full application of the model to simple, real-world games (Tic Tac Toe, Blackjack)
- full application of the model to "toy versions" of complex real-world games (3x3 Go)
- partial application of the model to complex real-world games in order to observe the strategies that occur at the lowest computational resource levels
- further exploration of the entropy model of state space topology

Acknowledgments

We would like to thank Jason Rohrer, Keith Burgun, Eric Miles, Abe Friesen, and the anonymous reviewers for their thoughts and comments on this work. Special thanks to the Banff International Research Station for Mathematical Innovation and Discovery, for hosting the Computational Modeling in Games workshop, and to Marc ten Bosch, Seth Cooper, and Steve Swink for our helpful discussion in Banff.

References

Abbott, R. 1975. Under the strategy tree. *Games & Puzzles (reprinted in Game and Puzzle Design, 2016)* 36:4–5.

Albers, J. 1971. *Interaction of color*. Yale University Press.

Allis, L. V. 1994. *Searching for solutions in games and artificial intelligence*. Ponsen & Looijen.

Browne, C. 2008. *Automatic generation and evaluation of recombination games*. Ph.D. Dissertation, Queensland University of Technology.

Cormen, T.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2001. *Introduction to algorithms, 3rd ed.* MIT Press.

Elias, G. S.; Garfield, R.; Gutschera, K. R.; and Whitley, P. 2012. *Characteristics of games*. MIT Press.

Ghostcrawler. 2016. /dev: On Depth vs. Accessibility. `http://nexus.leagueoflegends.com/2016/10/dev-on-depth-vs-accessibility`. Online; accessed 12-Oct-2016.

Gobet, F.; Retschitzki, J.; and de Voogt, A. 2004. *Moves in mind: The psychology of board games*. Psychology Press.

Kahneman, D. 2011. *Thinking, fast and slow*. Macmillan.

Kiley, O. 2013. Searching the depths: Strategy, tactics, and the deception of complexity. `http://www.big-game-theory.com/2013/01/Searching-the-Depths.html`. Online; accessed 20-Dec-2015.

Kolmogorov, A. N. 1968. Three approaches to the quantitative definition of information*. *International Journal of Computer Mathematics* 2(1-4):157–168.

Li, M., and Vitányi, P. 2009. *An introduction to Kolmogorov complexity and its applications*. Springer Media.

Nielsen, T. S.; Barros, G. A.; Togelius, J.; and Nelson, M. J. 2015. General video game evaluation using relative algorithm performance profiles. In *European Conference on the Applications of Evolutionary Computation*, 369–380. Springer.

Papadimitriou, C. H. 2003. *Computational complexity*. John Wiley and Sons Ltd.

Pulsipher, L., and Others. 2011. What is depth in games? `http://www.gamasutra.com/blogs/LewisPulsipher/20111219/90810/What_is_Depth_in_Games.php`. Forum Post; accessed 13-Dec-2015.

Redmond, M., and Garlock, C. 2016. Match 4 - Google DeepMind Challenge Match: Lee Sedol vs AlphaGo. `https://www.youtube.com/watch?v=yCALyQRN3hw`. Online; accessed 13-Mar-2016.

Robertie, B. 1992. Letters to the editor. *Inside Backgammon* 2(1):2–4.

Silva, F. d. M.; Isaksen, A.; Togelius, J.; and Nealen, A. 2016. Generating heuristics for novice players. In *IEEE Computational Intelligence in Games*.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.