University of Brasília – UnB
Faculty of Gama – FGA
Software Engineering

The Importance of (Exponentially More) Computing Power

Author: Gabriel Filipe Manso Araujo
Supervisor: Prof. Dr. Carla Silva Rocha Aguiar

Brasília, DF
2021

Gabriel Filipe Manso Araujo

The Importance of (Exponentially More) Computing Power

Monograph submitted to the undergraduate course in Software Engineering of the University of Brasília, as a partial requirement for obtaining the Bachelor's Degree in Software Engineering.

University of Brasília – UnB

Faculty of Gama – FGA

Supervisor: Prof. Dr. Carla Silva Rocha Aguiar

Co-supervisor: Dr. Neil C. Thompson

Brasília, DF

2021

Gabriel Filipe Manso Araujo
The Importance of (Exponentially More) Computing Power/ Gabriel Filipe Manso Araujo. – Brasília, DF, 2021-
51 p. : il. (algumas color.) ; 30 cm.

Supervisor: Prof. Dr. Carla Silva Rocha Aguiar

Undergraduate thesis – University of Brasília – UnB
Faculty of Gama – FGA , 2021.

1. Computing power. 2. Moore's Law. I. Prof. Dr. Carla Silva Rocha Aguiar. II. University of Brasília. III. Faculty of Gama. IV. The Importance of (Exponentially More) Computing Power

CDU 02:141:005.6

Gabriel Filipe Manso Araujo

The Importance of (Exponentially More) Computing Power

Monograph submitted to the undergraduate course in Software Engineering of the University of Brasília, as a partial requirement for obtaining the Bachelor's Degree in Software Engineering.

Undergraduate thesis approved. Brasília, DF, 2021, November 16th:

Prof. Dr. Carla Silva Rocha Aguiar
Supervisor

Prof. Dr. Renato Coral Sampaio
Guest 1

Prof. Dr. Paulo Meirelles
Guest 2

Brasília, DF
2021

To my siblings Daniel, Matheus, Miguel, Julia, and to all children who live in the "moon world" and dream of conquering the world.

Acknowledgements

I would first like to thank my supervisor, Prof. Dr. Carla Rocha, for all her support, kind words, advice, and wisdom that have helped me overcome many obstacles over the past five school years. Professor Carla, as I always tell you, I will always be grateful for everything you have done for me.

I want to thank my thesis co-supervisor, Dr. Neil Thompson at MIT. As a summer student just out of his second year of undergraduate, Dr. Neil Thompson was generous enough to give me the opportunity to work in his laboratory and introduce me to laboratory research. The example he sets as a leader is one I will aspire to if I ever find myself in a managerial role. I thank him for letting me try my strengths, for giving me the opportunity to fail at times, and for his constant support as my career goals have evolved. Professor Neil, I promise that I will always take all of our "teaching moments" with me.

I would also like to thank Prof. Dr. Vinicius Rispoli, Prof. Dr. Rejane Maria, Prof. Dr. Fabrício Braz, Prof. Dr. Renato Coral, Prof. Dr. Fernando William, Prof. Dr. John Gardengui, Prof. Dr. Nilton Correia da Silva, Prof. Dr. Maurício Serrano, Prof. Dr. Milene Serrano, Prof. M.Sc Hilmer Neri, Prof. M.Sc Cristiane Ramos and Prof. M.Sc Ricardo Ajax for always showing me the right directions, for believing in my potential, for giving me opportunities, and for always being willing to help me when I needed it.

In addition, I would also like to thanks my labmate, M.Sc Shuning Ge for all her patience, for supporting me and always making me believe in myself in many of the most chaotic moments of my University days. Shuning, "you are god!".

Finally, I would like to acknowledge my parents, my grandparents, and my siblings for always being there for me. My wife, Larissa, who is always by my side when times I needed her most, for her kind ear and for always taking care of me. And my friends, Andrew and João, for always holding hands with me when I am freaking out, for all our philosophical moments and sleepless nights talking about random things.

Thank you, everyone.

Abstract

Denizens of Silicon Valley have called Moore's Law "the most important graph in human history," and economists describe the Moore's Law-powered I.T. revolution as one of the most important sources of national productivity. But data substantiating these claims tend to either be abstracted – for example by examining spending on I.T., rather than I.T. itself – or anecdotal. In this work, we assemble direct evidence of the impact that computing power has had on two computing bellwethers: Computer Chess and Computer Go. Computing power explains 38%-94% of the performance improvements in these domains. Moreover, in line with economic theory, we find that exponential increases in computing power are needed to get linear improvements in outcomes, which helps clarify why Moore's Law has been so important. We also discuss how this dependence on computation means that performance improvements in these domains (and presumably many others) are becoming economically tenuous as Moore's Law breaks down.

Key-words: Computing power; Moore's Law; Production function; Computer chess; Computer Go;

Resumo

Habitantes do Vale do Silício chamaram a Lei de Moore de "o gráfico mais importante da história humana", e economistas descrevem a revolução do TI como uma das fontes mais importantes de produtividade nacional. Mas os dados que comprovam essas afirmações tendem a ser abstraídas - por exemplo, examinando os gastos com TI, ao invés de TI em si - ou anedótico. Neste artigo, reunimos evidências diretas do impacto que o poder da computação teve em dois termômetros da computação Computadores de Xadrez e de Go. O poder de computação explica 38 % - 94 % das melhorias de desempenho nesses domínios. Além disso, em linha com a teoria econômica, descobrimos que aumentos exponenciais no poder de computação são necessários para obter melhorias lineares nos resultados, o que ajuda a esclarecer por que a Lei de Moore tem sido tão importante. Também discutimos como essa dependência da computação significa que as melhorias de desempenho nesses domínios (e presumivelmente em muitos outros) estão se tornando economicamente tênues à medida que a Lei de Moore entra em colapso.

Key-words: Computing power; Moore's Law; Production function; Computer chess; Computer Go;

Contents

1 INTRODUCTION . . . 15
1.1 Objectives . . . 17
1.2 Work Structure . . . 17

2 BACKGROUND . . . 19
2.1 Production Functions Theory . . . 19
2.2 Computer Chess . . . 20
2.2.1 History . . . 20
2.2.2 Measuring Performance in Computer Chess . . . 22
2.2.3 Measuring Computing Power in Computer Chess . . . 23
2.3 Computer Go . . . 24
2.3.1 History . . . 24
2.3.2 Measuring Performance in Computer Go . . . 24
2.3.3 Measuring Computing Power in Computer Go . . . 26
2.4 Go vs Chess – Why Go Is So Complex? . . . 26

3 METHODOLOGY . . . 29
3.1 Research Questions . . . 29
3.2 Methodology diagram . . . 29
3.3 Data Gathering . . . 30
3.3.1 Computer Chess . . . 30
3.3.2 Computer Go . . . 31
3.4 Data processing . . . 33
3.5 Data Analysis . . . 33
3.6 Technologies . . . 34
3.6.1 Python . . . 34
3.6.2 Jupyter . . . 34
3.6.3 Seaborn . . . 34
3.6.4 Pandas . . . 34
3.6.5 Scikit-learn . . . 34
3.6.6 Statsmodels . . . 34

4 RESULTS . . . 35
4.1 Computer Chess . . . 35
4.2 Computer Go . . . 37
4.3 Analysis and Discussion . . . 37

4.3.1 The Growth in Computing Power 39
4.3.2 Contributions of I.T. to Performance Improvement 40
4.3.3 Analysis of variance 41

5 CONCLUSION 43

BIBLIOGRAPHY 45

1 Introduction

Production functions, be them macroeconomic[1] models of the economy or microeconomic[2] models of individual agents or firms, attempt to model and parameterize how changes in key inputs produce changes in output. Traditional models, for example (ROMER, 1989; ROMER, 1990), focused on two key inputs: labor and capital, which are mutual complementary, but independently face decreasing marginal returns. But not all capital is the same, and so some modern models (ACEMOGLU, 2018b; ACEMOGLU, 2018a; CHENG; NAULT, 2007) split apart I.T. capital, making it complementary to other forms of capital (HITT, 1996; BRYNJOLFSSON, 1997), rather than substitutable.

Studies at the micro-level have found strong effects on the importance of I.T. on firm productivity[3] (BANKER; KAUFFMAN, 1991; GEROW et al., 2014; SABHERWAL; JEYARAJ, 2015; FOSTER; FLYNN, 1984; PINSONNEAULT; RIVARD, 1998). For example Brynjolfsson and Hitt (HITT, 1996) showed I.T. capital, as distinct from other types of capital, contributed significantly to margin firm output. Most studies, however, take an abstracted view of computing capital, measuring it in dollars rather than in productive capacity. As Devaraj and Kholi (DEVARAJ; KOHLI, 2003) have pointed out, this is inferior to measuring I.T. in the form of actual usage.

In this work [4], we study the micro-level foundations for how I.T. capital improves performance in the natural units for computer processors: computing power, as measured by the number of operations that it can perform. This is very much in the spirit of Thompson, Greenewald, Lee, and Manso (THOMPSON et al., 2020), which shows that many of the most-important improvements in machine learning are heavily dependent on improvements in computing power. In this work, we extend this type of analysis to other areas, and show that a much broader conclusion can be reached: progress across other areas of computing is dependent on exponential increases in computing power. To reach this conclusion, we gather detailed records on the performance and usage of computing power across the domains of Computer Chess and Computer Go.

---

[1] Branch of economics dealing with performance, structure, behavior, and decision-making of an economy as a whole (Wikipedia , 2021e).

[2] Branch of mainstream economics that studies the behavior of individuals and firms in making decisions regarding the allocation of scarce resources and the interactions among these individuals and firms (Wikipedia , 2021f).

[3] The efficiency with which firms, organisations, industry, and the economy as a whole, convert inputs (labour, capital, and raw materials) into output (Australian Government Productivity Commission, 2021)

[4] This document is an excerpt from a research carried out in conjunction with two other co-authors, Dr. Neil Thompson (MIT CSAIL) and M.Sc Shuning Ge (University of Pennsylvania), and will contain more detailed information about two case studies, Computer Chess and Computer Go, in which I was in charge as well as other analysis that I was also responsible for.

Chess and Go are important bellwethers for computing performance because both were traditionally viewed as areas of human expertise. Therefore, progress against human acumen could be used to track the development of programs that could play these games. For example, chess master David Levy said "Until 1977, there seemed to be no point in my playing a formal challenge match against any chess program because none of them were good enough, but when [the program] CHESS 4.5 began doing well... it was time for me to defend the human race against the coming invasion." (LEVY; NEWBORN, 1982). Of the two games, Go is much harder. In 1997, astrophysicist Piet Hut, from the Institute for Advanced Study in New Jersey, told the New York Times (in retrospect incorrectly) that "It may be a hundred years before a computer beats humans at Go — maybe even longer." (MUOIO, 2016). Because these two games are such bellwethers, they have attracted substantial attention from computer scientists which has led to a broad exploration of computational algorithms, hardware and software approaches for playing. As such, these bellwethers present a promising way to study the importance of computing power for progress.

We see large effects from increases in computing power on both domains. In Computer Chess, a 10× increase in computing power correlates with an increase of 242-point Elo – half the point difference between Masters from Grandmasters. In Computer Go, we find a similar result with a 10x increase in computing power leading to a 244-point Elo improvement.

Perhaps as interesting as our overall findings about the importance of computing, is our ability to contrast the contributions that computing power is making to improved performance to those arising from other sources. Using an analysis-of-variance approach, we find that computing power explains 48-88% of the variation in output. Put another way, for these applications, increases in computing power[5] are at least as important as all other factors put together.

Our findings are particularly important at a time when computer progress is slowing, and thus the ability to get improvements in performance is getting harder (LEISERSON et al., 2020; THOMPSON; SPANUTH, 2021). Perhaps even more worrisome, if sources of computer improvement (such as Moore's Law) are running out, then the cost of improvement will rise proportionally to computing power increases. But, since exponential increases in computing power would then come with exponential increases in cost, such improvements are likely to be economically unappetizing, and thus we would expect the rate of progress to diminish in many areas as increases in computing power slow.

[5] This includes both computing power increases directly, as well as other productivity increases that depend on computing power increases (e.g. a less efficient algorithm that nevertheless performs better with sufficient computing power).

1.1 Objectives

To reach the expected goal of this work, the following tasks needed to be accomplished:

- Understand domains of the case studies (Computer Chess and Computer Go);
- Study the most suitable computing power and performance metrics for each case;
- Gather detailed records on the performance and usage of computing power across these two domains;
- Create scripts for data cleaning and preprocessing making necessary adjustments to the metrics, if needed;
- Perform ordinary least squares regressions to better understand the existing relationship between computers' performance and their computing power;
- Interpret coefficients based on the production function theory;

1.2 Work Structure

This work is structured starting with a background chapter, which explains the context of meaningful topics addressed in the study as well as historical contextualization of the case studies, a proposal chapter that defines the study's proposal, research questions, methodology, and technologies used, a results and discussion chapter that presents the result of our analysis and some discussions about further topics, and a conclusion.

2 Background

2.1 Production Functions Theory

Production functions [1], of the type pioneered by Romer (ROMER, 1986), take the following form:

$$Y = AL^{\alpha}K^{\beta} \tag{2.1}$$

Where Y represents output (the quantity of a good produced), K represents capital, L labor, and A represents a productivity multiplier that is calculated as a residual. The marginal contribution as each input grows is parameterized by $0 < \alpha < 1$ and $0 < \beta < 1$. As Syverson (SYVERSON, 2011) has pointed out, this form has the advantage of being a linear approximation to any production function. Two key features of such production functions are that they have decreasing marginal returns in each input, and that they are complementary between inputs. Thus, adding ever more capital becomes less and less efficient, but growing capital and labor proportionally yields mutually reinforcing benefits. As I.T. capital becomes increasingly important and evidence mounts that it is complementary, rather than substitutive, to other forms of capital (HITT, 1996), it makes sense to add it separately as another input factor, to get:

$$Y = AL^{\alpha}K^{\beta}IT^{\gamma} \tag{2.2}$$

For the cases that we will be observing, the rate of change of these inputs will be dramatically different. In particular $\frac{\partial IT}{\partial t}$, $\frac{\partial A}{\partial t} \gg \frac{\partial L}{\partial t}, \frac{\partial K}{\partial t}$, that is, the rate of increase of I.T. capital and of productivity are much higher than those for labor and non-I.T. capital. This is because there has been large exponential growth in the provision of computing power (DANOWITZ et al., 2012; LEISERSON et al., 2020; THOMPSON; SPANUTH, 2021) and in the improvement of using that computing power more efficiently through algorithms (HOLDREN; LANDER; VARMUS, 2010; SHERRY; THOMPSON, 2020) but relatively little change in labor and non-IT capital[2].

These growth rates have two important implications. First, exponential growth in an input can be sufficient to overcome decreasing marginal returns and thus provide a

[1] A production function, in economics, is a function that shows the relationship between inputs such as capital and labor (and other factors) and the outputs of goods and services. (The Editors of Encyclopaedia Britannica, 2021a)

[2] As an example, the number of employees at the National Oceanographic and Atmospheric Association (NOAA) has decreased from nearly 13,000 in 1997 (ENERGY; ENVIRONMENT, 1997) to 11,000 in 2015 (Wikipedia , 2021g)

constant (or rising) share of the improvements to output. The second implication of faster rates of growth for I.T. capital and productivity is that other rate terms are likely to be negligible, i.e.

$$\frac{\partial \mathbf{Y}}{\partial \mathbf{t}} = \frac{\partial \mathbf{Y}}{\partial \mathbf{A}} \cdot \frac{\partial \mathbf{A}}{\partial \mathbf{t}} + \frac{\partial \mathbf{Y}}{\partial \mathbf{IT}} \cdot \frac{\partial \mathbf{IT}}{\partial \mathbf{t}} + \frac{\partial \mathbf{Y}}{\partial \mathbf{L}} \cdot \frac{\partial \mathbf{L}}{\partial \mathbf{t}} + \frac{\partial \mathbf{Y}}{\partial \mathbf{K}} \cdot \frac{\partial \mathbf{K}}{\partial \mathbf{t}} \approx \frac{\partial \mathbf{Y}}{\partial \mathbf{A}} \cdot \frac{\partial \mathbf{A}}{\partial \mathbf{t}} + \frac{\partial \mathbf{Y}}{\partial \mathbf{IT}} \cdot \frac{\partial \mathbf{IT}}{\partial \mathbf{t}} \quad (2.3)$$

In the case studies that follow, we estimate how performance (Y) and computing power (IT) have changed over time for these important areas of computing. We find that, as expected, IT has strong decreasing marginal effects on output, but that $\frac{\partial \mathbf{IT}}{\partial \mathbf{t}}$ has grown so rapidly that $\frac{\partial \mathbf{Y}}{\partial \mathbf{IT}} \cdot \frac{\partial \mathbf{IT}}{\partial \mathbf{t}}$ accounts for most of $\frac{\partial \mathbf{Y}}{\partial \mathbf{IT}}$. That is, growth in computing power explains most of the growth in output. In the analysis that follows, we estimate these parameters explicitly, test the robustness of these effects, and consider what these mean for the economic viability of improving outcomes using I.T..

2.2 Computer Chess

2.2.1 History

Chess is one of the most popular board games in the world and has been played since at least the 6th century A.D. (MURRAY, 2015). The first discussion of chess, from a computational perspective, came in 1950 when the mathematician Claude Shannon published a paper entitled “Programming a Computer for Playing Chess” (SHANNON, 1950). Shortly thereafter, Alan Turing created the first algorithm that a computer could use to play chess. Unfortunately, because of the state of computing at the time, Turing’s chess algorithm had to be executed manually. It took 15-30 minutes to calculate each move and the algorithm only considered the consequences of its actions two moves in advance. It could not play a full game, much less beat a professional chess player. According to Kasparov and Friedel, Turing’s algorithm would take less than five milliseconds to calculate each move in a modern computer in 2017 (KASPAROV; FRIEDEL, 2017).

Roughly speaking, all computer chess programs do two types of operations: (1) look ahead to potential future board states and (2) evaluate how likely any board position is to produce a win. For example, a program might look ahead 3 moves (called “plies”) and for each potential result it will evaluate whether it is in a good or bad position by counting when opponent’s pieces can be constrained or taken. In computer science, the resulting analysis is thought of as a tree, where nodes are the board position and edges are the possible moves for each player, as shown in Figure 1.

Each node has an associated probability of winning. In general, these guide the system towards good moves, but can be misleading if the computer can’t look far enough into

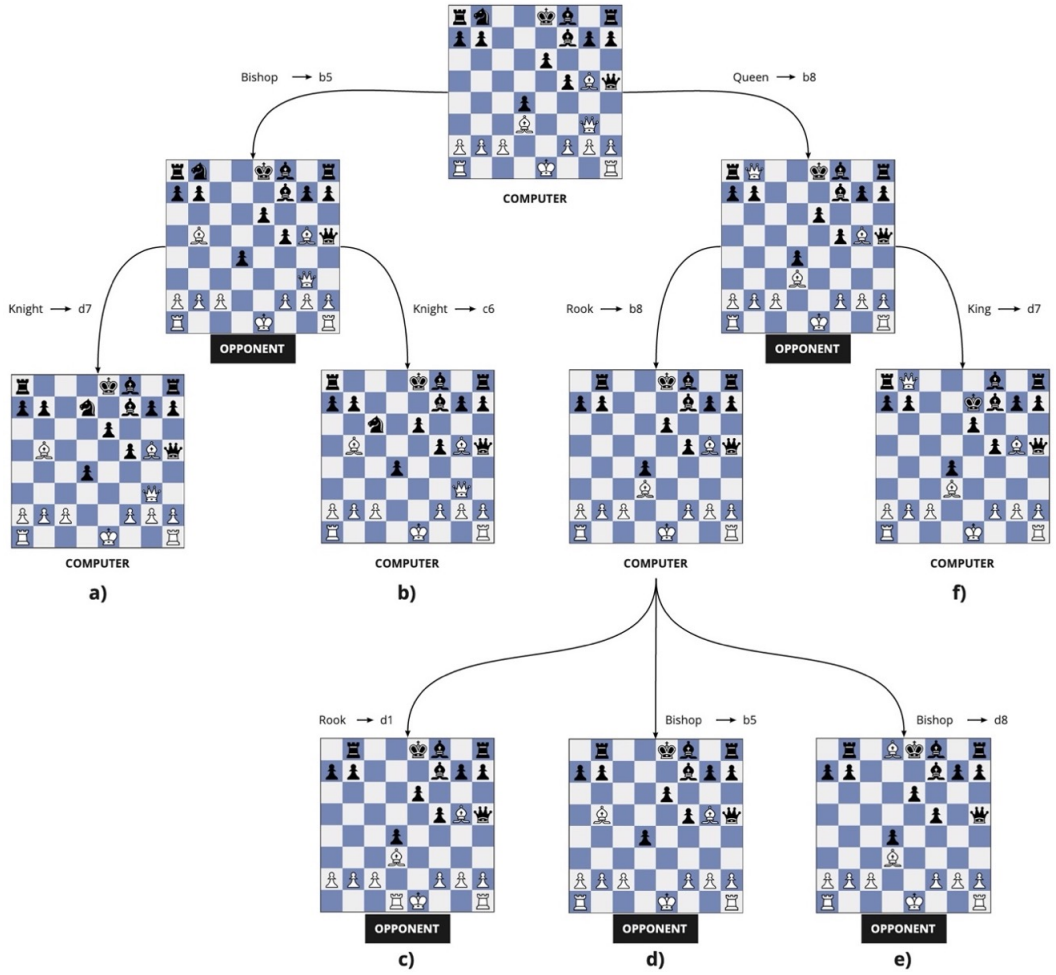

Figure 1 – Small subset of a decision tree for chess.

the future (and see a potential problem spot) or if it incorrectly evaluates how valuable different board configurations are. Figure 1 illustrates it showing that among the move options, only d) leads the computer (white player) to checkmate in 3 plies. Otherwise, if any other move is taken, the computer would take longer to reach the victory. Additional computing ameliorates these deficiencies by looking further ahead in the tree or adding more evaluation criteria to each position. Looking farther ahead, however, requires exponentially more computing power since the number of possible moves grows exponentially as you project forward. For example, if each player had 10 potential moves each ply, then there are 10,000 potential evaluations after four plies (10^{2+2}), whereas looking six plies into the future would take 1,000,000 evaluations (10^{3+3}). Smarter evaluation algorithms help this by decreasing the number of moves considered at each point, but at the cost of doing more computation to evaluate each board position. Most programs find a balance between spending time exploring farther ahead and evaluating positions more carefully - although there are exceptions that heavily favor one approach or another[3].

[3] One such example is AlphaZero, a 2017 program by DeepMind that focuses more on evaluating positions than depth of search.

By 1957, Alex Bernstein was able to develop a chess program running on an IBM 704 mainframe that was capable of playing a full game (BERNSTEIN; ROBERTS, 1958). Ten years later, the MacHack IV program was the first to play in a tournament. It ended up with a score of one draw and four losses against amateur J.Conroy (WALL, 2008). 30 years later, in 1996, after enormous work by the computer chess community Deep Blue, an IBM RS/6000 SP supercomputer[4] capable of calculating 200 million chess positions per second, beat the world chess champion, Garry Kasparov. This event was a major milestone in the history of computer chess and is known as the beginning of computer chess supremacy.

Algorithm and hardware improvements have continued to progress since Deep Blue. A study by "hippke" shows evidence that modern computer chess algorithms could achieve the same performance as Kasparov already in 1994 running on a single 486-DX4 100 MHz (HIPPKE, 2020). On the other hand, old chess engines could achieve much higher performance running on modern hardware (HIPPKE, 2021). As of July 2021, the best chess program was Stockfish 13. With an Elo of 3547, Stockfish 14 is 665 points better than the best rating ever achieved by a chess human player, Magnus Carlsen at 2882 in May 2014 (CCRL, 2021). This means that the best players in the world are as likely to win a game against Stockfish 13 as a top amateur player would be to win a game against a grandmaster.

2.2.2 Measuring Performance in Computer Chess

The most used performance metric in chess, and computer chess, is known as "Elo" (Wikipedia , 2021b). The Elo rating system system was designed to make the probability of a chess player to win a match, and the understanding of how Elo can vary in consequence of players' performance, easily understandable. It can be summarized by formulas 2.4 and 2.5.

$$Prob(A\ Wins) = \frac{1}{1 + 10^{\frac{-(ELO_A - ELO_B)}{400}}} \tag{2.4}$$

$$ELO_{new} = ELO_{old} + K(\sum_{i=1}^{n} S_i - \sum_{i=1}^{n} Prob(A\ Wins)_i) \tag{2.5}$$

Formula 2.4 calculates the winning likelihood of a player in a given match. The variables ELO_A and ELO_B represent the players' ratings. For instance, in a hypothetical game between players rated with a 50-point difference (e.g., 1550 and 1600), the winning likelihood for the lower-rated player could be estimated as $\frac{1}{1+10^{\frac{-(50)}{400}}} = 43\%$. If this rating

[4] In terms of comparison, NASA's Pathfinder used the same IBM RS/6000 technology for its onboard flight to Mars (COMPUTERWORLD, 1997)

difference among the players is bigger, say a 1200-amateur against a 2500-grandmaster, the winning likelihood for the 1200-player would decay to 0.06%.

Players' Elo are updated after they participate in tournaments. Formula 2.5 demonstrate who it works. In this formula, $Prob(A\ Wins)_i$ would be the estimated winning likelihood for each tournament match (Formula 2.4), S_i is the overall performance of the player registered in the tournament, K is a constant, ELO_{old} is the player's old rating (before the tournament), ELO_{new} is the adjusted player's rating and n is the number of games played in the tournament.

The chess rating division can be visualized in Table 1.

RATING RANGE	CATEGORY
2882 +	SUPER-HUMAN [5]
2500 - 2882 [6]	GRANDMASTER
2000 - 2499	EXPERT
1200 - 1999	AMATEUR
0 - 1199	NOVICE

Table 1 – Rating System in Chess

2.2.3 Measuring Computing Power in Computer Chess

When we are talking about measuring the computational power of a chess machine, 3 are the main metrics that can be used: positions/sec, search depth and MIPS.

Positions (or node) per second is the most widely used, and appropriated, metric when it comes to measuring the computational capability of a chess computer (Chessify,). It is defined as the number of positions that a chess program can calculate per second. *Search depth*, in turn, it concerns the depth, the level, maximum that a chess engine can traverse in the decision tree established by its used search algorithm. Finally, another metric that can be used to estimate the computing power of these computers is million instructions per second ($MIPS$). This one measures the microprocessor's speed looking at the number of million integer instructions that a computer can execute in one second.

For computer chess programs which data on $Positions/sec$ was not found, but instead, it was reported data for *search depth* or $MIPS$, we performed alternative regressions that would allow us to convert both *search depth* and $MIPS$ into positions/sec[7].

[6] This category was created to compose all chess programs that pursue a rating score higher than Magnus Carlsen, considered the best chess player of the world.

[6] This is the highest score ever achieved by a human Grandmaster as was recorded by Magnus Carlsen in May 2014.

[7] To convert $MIPS$ and *search depth* to "$Positions/sec$" we ran $Positions/sec = 10^{(\alpha+\beta log_{10}(MIPS))}$ and $Positions/sec = 10^{(\alpha+\beta\ SearchDepth)}$ respectively.

2.3 Computer Go

2.3.1 History

Computer Go began with Albert Zobrist's 1970 dissertation and programs – such as Interim.2, The Many Faces of Go and Go++ – that emerged shortly thereafter. However, only in 1984 did computer Go tournaments start to be held, and in 1986, the first computer Go congress ever took place (British Go Association, 2018). From 1986 to 2006, the progress in Computer Go suffered a long plod. This was mainly in consequence of the computational and algorithms limitations to deal with such complex game. In 2006, an algorithmic improvement[8] significantly improved performance (GELLY; SILVER, 2011; WALL, 2008) of Computer Go, allowing for the first time a Go program to achieve an advanced amateur (1 dan) rank on a smaller (9×9) board (GELLY et al., 2012). In 2015, AlphaGo defeated the European Go champion, Fan Hui, in a five-round game by 5-0 (BBCNEWS, 2016). This was the first time that an AI system had beaten a human professional player without a handicap. One year later AlphaGo sealed a 4-1 victory over Lee Sedol (BOROWIEC, 2016), considered by many to be the best Go player of all time (GIBNEY, 2016).

AlphaGo introduced to the world a creative approach for Go computers by addressing both the massive search problem and lack of knowledge problem. Its tree search algorithm could evaluate positions and select moves using deep neural networks. These neural networks (NNs) were trained by supervised learning (SL) from human expert moves, and by reinforcement learning (RL) from self-play (SILVER et al., 2017). Many grandmasters said that new moves and new types of strategies had been explored by some insightful moves from AlphaGo.

In November 27th, 2019, Lee Se-dol, the only human to ever beat AlphaGo, announced that he would no longer play Go professionally due to the AI invincibility. He said "With the debut of AI in Go games, I've realized that I'm not at the top even if I become the No. 1 through frantic efforts... Even if I become the No. 1, there is an entity that cannot be defeated.".

2.3.2 Measuring Performance in Computer Go

In the Edo period (Japan, 1603 – 1868), a rank system based on stones was developed aiming to offset the strength difference between go players. It was known as the handicap system(Wikipedia , 2021d). The handicap system was mainly responsible for supporting another rank system named kyu/dan (Table 2) (Wikipedia , 2021a).

[8] The introduction Monte-Carlo tree search (MCTS) applied to Computer Go (KOCSIS; SZEPESVáRI, 2006)

RANK	CATEGORY
9 pro - 1 pro	PROFESSIONAL
6 dan - 1 dan	ADVANCED AMATEUR
1 kyu - 9 kyu	INTERMEDIATE AMATEUR
10 kyu - 19 kyu	CASUAL PLAYER
20 kyu - 30 kyu	NOVICE PLAYER

Table 2 – Rank System in Go

Before the existence of computers that plays Go, the strength difference between players was enough to say that each rank difference, from 30 kyu to 9 pro, was equals to one handicap stone. For instance, if the difference of two players is equal to 2 ranks, two handicap stones are given to the weaker player as a way to offset the level disadvantages between the players.

Four centuries later, the advent of the internet allowed Go players to start playing games against machines. Online servers became widely used making practicing much more accessible. In consequence of this phenomena, human players could get better and better in this game. As an effect, the number of handicap stones for matches between humans against Go computer, in a 19×19 board, decreased from 29 (1998) to 0 (2016).

At this time, the handicap stone was already considered obsolete by many, due to the difficulty of measuring the professionals' rank differences which have become tighter and tighter over the years [9]. It has led many players to discredit this rank system and invest in other methods of measuring the strength difference between professional players. One of these alternative ways is adopted by the European Go Federation (EGF), where instead of using a rank system, the levels of players are distinguished by Elo (European Go Database, 2018). However, both in the literature and in other resources where it was possible to find data related to the performance of Go computers, the vast majority of data were reported using the rank kyu/dan system and handicaps. Thus, in this work we propose a formula (Formula 2.6) that, even with the decrease in the values of a handicap stone in the professional rank, it would still be possible to estimate the rank values using the handicap system.

$$r_i = \begin{cases} r_j - s_i, & r_j \leq 36 \\ (36 + (\frac{r_j - 36}{4}) - s_i, & r_j > 36 \end{cases} \tag{2.6}$$

In the formula, r_i is the player i's rank, r_j is the player j's rank, and s_i is the number of handicap stones given to the player i. When s_i is a negative value indicates that, instead of given to the player i, it was actually given to the player j [10].

[9] Currently, the gap of the eight professional ranks is approximately equal to two stones, instead of eight.

[10] Note that, the kyu/dan ranks were converted to the numerical scale to be expressed into the formula.

After estimating the rank by using the proposed formula, we recall to the European Go Federation (EGF) rating system to convert the ranks to equivalent Elo[11]. It is important to note that, in the EGF rating system, a 9-pro is equivalent to 2940 Elo. But, in consequence of the significant improvements in Computer Go, this Elo was already surpassed. One example is AlphaGo Zero (5185) (SILVER et al., 2017). Then, for a 9-pro Go program, it was used the Elo directly provided (if provided) by their developer teams.

2.3.3 Measuring Computing Power in Computer Go

In order to measure the computational power, we looked at the number of float points operations per second (Flops) provided by the processors used by Go computers over the years.

Processors used in computers range from CPUs to TPUs. Fortunately, data related to GPU and TPU flops is easily found. This type of data is usually available either on the official specification page of these pieces of hardware or on benchmark websites.

In contrast to that, calculating the number of flops on a CPU can be quite a bit of work at times. This is because this data is not available on hardware spec pages or easily found in other type of resource. Therefore, for CPUs, this data had to be manually calculated from Formula 2.7.

$$Flops = \#\ of\ cores * frequency * FP32 \quad (2.7)$$

[12]

Finally, having the number of flops of all computer processors, they are added together using the formula 2.8.

$$Total\ flops = \sum_{i=1}^{n} CPU flops * \sum_{i=1}^{n} GPU flops * \sum_{i=1}^{n} TPU flops \quad (2.8)$$

2.4 Go vs Chess – Why Go Is So Complex?

Go is the oldest board game played in the world, having been invented in China approximately four thousand years ago (ASSOCIATION, 2020). The basic rules of Go are simple: players take turns placing their stones on a 19×19 board and get points by completely surrounding the other player's stones. This simplicity, however, masks an enormous amount of computational complexity due to the large number of possible moves that players can make each turn. To put this into perspective, after each player makes a

[11] For calculation details, check the EGF's official website.

[12] FP32 is the number of 32-bit float point operations that the CPU can process per cycle given the microarchitecture that the CPU belongs to. Each microarchitecture has a specific number of FP32.

single move in chess, there are 400 possible board configurations, in Go, there are 130,000 (INSIDER, 2016).

The relative complexity of games is often described by their "state-space complexity", which is the number of game positions that are reachable in legal gameplay(Wikipedia , 2021c). In Go, the state-space complexity is estimated to be 10^{170}, a value higher than the number of atoms in the universe, computational capability of a supercomputer and of any other known board game Figure 2. This complexity is important for software programs that play the game because it limits how much brute-force can be used to strategize, for example by looking ahead to every move. For Go, the traditional algorithms used (minimax tree search and alpha-beta pruning) cannot look far ahead. MoGo, the first Go computer ever to achieve the dan (master) level in a 9×9 board in 2008, could look routinely only to 10 (or more) moves ahead (SILVER; SUTTON; MüLLER, 2012).

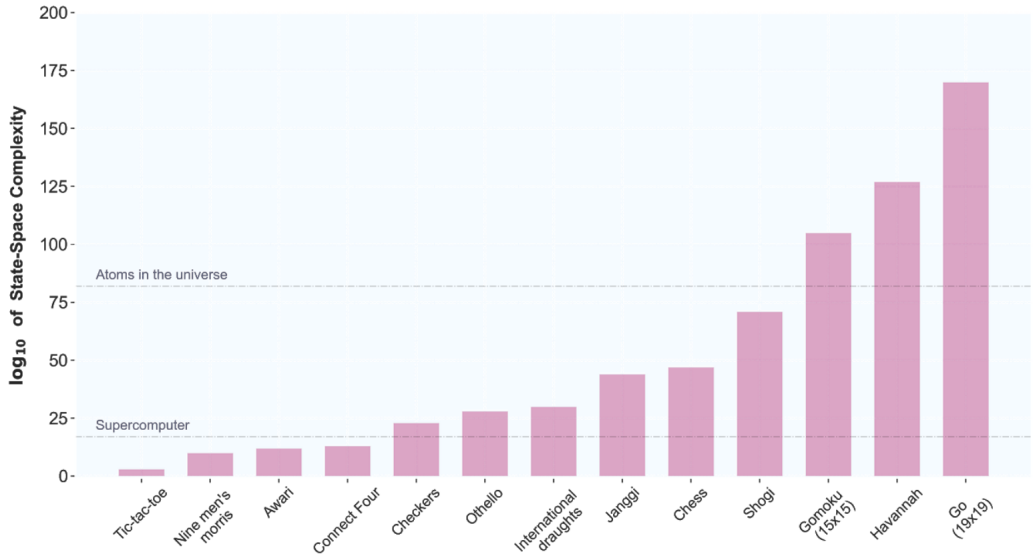

Figure 2 – State-space complexity of board games compared to the number of atoms in the universe and the computational burden of a supercomputer.

Another way of understanding this difference is by comparing the hardware that was used when programs beat the best humans in these two games. When Deep Blue, chess-playing specialized supercomputer developed by IBM, beat Garry Kasparov at chess, it used 30 chips containing 480 specialized processors. Twenty years later, when processor chips where 100× better (HENNESSY; PATTERSON, 2011), AlphaGo, program started in 2014 to test how well deep learning can compete at Go (Ribeiro, John, 2016), beat Lee Sedol using, roughly, 75× as many processors[13] . That is, using 7500×+ more computing

[13] The exact number of processors used by AlphaGo is unknown. Here we make the simplifying assumption that each modern chip has a similar number of processors as the IBM machines (16 processors/chip) and thus AlphaGo's 1920 CPUs, 280 GPUs, and 48 TPUs (THEECONOMIST, 2016; SILVER et al., 2017) represent at least 36,000 "processors-equivalents". In actual fact, the number may be much larger since, for example, a single TPU has 65,000 multiply units, which would make our number an underestimate.

power.

3 Methodology

The methodology of this work was established to better understand what role computational power has played in improving performance in the Computer Chess and Computer Go domains. Thus, this chapter aims to specify in more detail each of its steps.

3.1 Research Questions

This work focuses on creating a timeline of computer chess and Go since their beginnings, in order to map the contributions that computing power has had over the years in these domains, as well as understand their rate of growth over time. Thus, the main questions we seek to answer in this work are:

RQ. 1 *What are the in computing power growth rate (ψ) in these domains?*

RQ. 2 *What are the contributions of I.T. (γ) to performance improvement in these domains?*

RQ. 3 *What are the the share of variance in performance explained by computing power ($\frac{\partial Y}{\partial IT} \cdot \frac{\partial IT}{\partial t}$) in these domains?*

RQ. 4 *What are the the residual portion of growth* ***not*** *explained by increases in computing power ($\rho \equiv \frac{\partial Y}{\partial t} - \frac{\partial Y}{\partial IT} \cdot \frac{\partial IT}{\partial t}$) in these domains?*

For each of these questions, subtopics were created in the "Analysis and Discussion" section to answer them.

3.2 Methodology diagram

The diagram in Figure 3 was made to illustrate the methodology we chose to answer the research questions of this work.

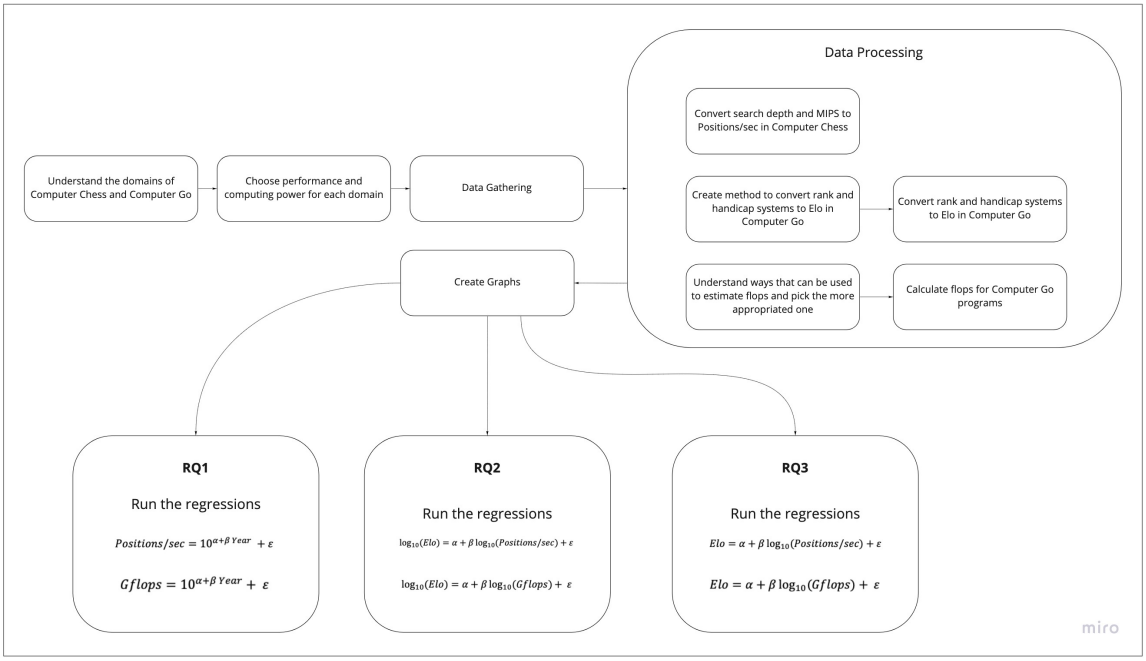

Figure 3 – Methodology Diagram

3.3 Data Gathering

This section specifies all resource sources that were visited in data collection in both Computer Chess and Computer Go. Data collection was done entirely manually due to the lack of a single data source.

3.3.1 Computer Chess

To assess the progress of Computer Chess programs since 1957 (Bernstein's program), we gathered data from:

- The International Computer Games Association;
- The Swedish Chess Computer Association;
- World Chess Federation;
- US Chess Federation;
- 16 Books (see Table 3);
- 8 Researcher Papers (see Table 3);
- 5 Conference and Tournament Reports (see Table 3);
- 4 Newspapers and Magazine Articles (see Table 3);

- 3 Databases provided by chess organizations (see Table 3);
- 7 Forum Articles and Community Discussions (see Table 3);
- 4 Official webpages (see Table 3);
- Personal interviews with Computer Chess program developers[1];

Source	Reference
Books	Computer Chess Compendium by David Levy, Claude E. Shannon p.372 (LEVY, 2013)
	Computer Games I by Hans Berliner (auth.), David Levy (eds.) p.206 (LEVY, 2012)
	Chess Skill in Man and Machine by Peter W. Frey p. 207 (FREY, 2012)
	"Concise Encyclopedia of Computer Science" by Edwin D. Reilly (book) (REILLY, 2004)
	Computer Chess, Then And Now: The Deep Blue Saga by Feng-hsiung Hsu p.154 (HSU, 1997)
	A.C.M monograph series by Monroe Newborn and Thomas A. Standish page 133 (BERLINER, 1976)
	All About Chess and Computers by David Levy p.36 (LEVY; NEWBORN, 2012)
	Deep Blue: An Artificial Intelligence Milestone (Page: 36, by Monty Newborn) (NEWBORN; NEWBORN, 2003)
	Digital at Work - Snapshots from the first thirty-five years by Jamie Parker Pearson p.54 (PEARSON, 1992)
	The Game of Chess by Nicolae Sfetcu (SFETCU, 2016)
	Kasparov versus Deep Blue by Monty Newborn p.58 (NEWBORN, 2012)
	Scalable Search in Computer Chess by Ernst A. Heinz p.127 (HEINZ, 2013)
	Beyond Deep Blue Chess in the Stratosphere-Springer-Verlag London page 150 (ROUGETET, 2019)
	The Quest for Artificial Intelligence page 593 (NILSSON, 2009)
	ROBOT, Moravec, Oxford, 1998, Chapter 3: Power and Presence, page 71 (MORAVEC, 2000)
	Computers, Chess, and Cognition by T. Anthony Marsland, Jonathan Schaeffer p.18 (SCHAEFFER; MARSLAND, 1990)
Research Papers	IBM'S DEEP BLUE CHESS GRANDMASTER CHIPS
	WCCC (Rating Computer Science Via Chess)
	MAC PROJECT (Artificial Intelligence Memo. No.178) - MIT
	23rd ACM Computer Chess Championship
	Deep Blue by Murray Campbell
	Chess computer - ChessGenius
	Communications of the ACM Vol 35, Num 11
Conference and Tournament Reports	http://rebel13.nl/misc/tournaments/yat.html
	http://www.anacadigital.com/historia/anaca5_1_89.htm
	Advances in Computer Vol.29 by Marshall C. Yovits
	Advances in Computer Chess 3, (eds.) M.R.B. Clarke
	Results of ACM's eighteenth computer chess championship, reported by Monty Newborn
Newspaper & Magazine Articles	Computerworld Vol.X
	Computerworld, 1987, No.49, Vol.XXI
	https://www.newscientist.com/article/dn3312-kasparov-flummoxes-chess-computer/
	the LINK - The magazine of Carnegie Mellon University's School of Computer Science
Databases	Computer Chess Rating Lists
	ChessBase
	https://frc.ri.cmu.edu/~hpm/book97/ch3/processor.list.txt
Forum Articles and Community Discussions	Computers and Chess - A History by Bill Wall
	http://billwall.phpwebhosting.com/articles/Kaissa.htm
	https://www.chess.com/article/view/machack-attack
	http://www.chessmaniac.com/kaissa-chess-program/
	https://www.reddit.com/r/chess/comments/8fe70l/how_many_positions_per_second_approximately_was/
	https://www.chess.com/news/view/updated-alphazero-crushes-stockfish-in-new-1-000-game-match
	http://billwall.phpwebhosting.com/articles/engines.htm
Official Webpages	http://www.chessgenius.com/
	https://www.ibm.com/ibm/history/documents/pdf/rs6000.pdf
	https://www.sjeng.org/indexold.html

Table 3 – Computer Chess Data Sources

3.3.2 Computer Go

To assess the progress of Computer Go Programs, we gathered data from:

- British Go Association;
- European Go Federation;
- Ontology Applications & Software Engineering Laboratory (OASE Lab.);

[1] We directly contacted Computer Chess programs developers, teams, and professional players to ask for data related to some programs that we could not find online. Members of the Chess community also generously provided significant help (e.g. Discord, Facebook). These interviews did not contain a specific structure and were performed randomly

- European Go database;
- KGS Go Server;
- Go forums (e.g. Sensei);
- 2 Personal reports and websites (see Table 4);
- 4 Conference and tournament reports (see Table 4);
- 4 Forum articles and community discussions (see Table 4)
- 2 Newspaper and magazine articles (see Table 4);
- 9 Brand and benchmark websites (see Table 4);
- 2 Tutorials and Manuals (see Table 4);
- 1 Technical Reports (see Table 4);
- Personal interviews with Computer Go programs' developers [2];

As our main source, we used a large database provided by Mr. Nick Weed on his website http://www.computer-go.info.

Source	Reference
Personal Reports	http://users.ics.aalto.fi/praiko/altparty2009/
	https://www.usgo.org/sites/default/files/bh_library/Supercomputer%20Go.pdf
Conference & Tournament Reports	http://www.altparty.org/2009/competition-rules.html#csc-compo
	https://www.usgo.org/sites/default/files/ejournal_archive/20080807/20080807.htm
	http://computer-go.info/events/na/2008/index.html
	https://www.eurogofed.org/index.html?id=89
Forum Articles & Community Discussions	https://www.lesswrong.com/posts/shnSyzv4Jq3bhMNw5/alphago-zero-and-the-foom-debate
	https://www.britgo.org/bgj/06316
	https://www.britgo.org/computergo/history
	https://blog.codecentric.de/en/2014/10/codecentric-go-challenge-2014-interviews-franz-josef-dickhut-remi-coulom/
Newspaper & Magazine Articles	https://www.theguardian.com/technology/2009/apr/30/games-software-mogo
	https://www.wired.com/2016/03/two-moves-alphago-lee-sedol-redefined-future/
Brand & Benchmark websites	https://www.top500.org/system/174848
	https://lowendmac.com/1991/mac-classic-ii/
	https://everymac.com/systems/apple/mac_pro/specs/mac-pro-eight-core-2.26-early-2009-nehalem-specs.html
	http://www.alternatewars.com/BBOW/Computing/Computing_Power.htm
	https://www.mobygames.com/game/bbc-micro_/microgo1
	http://www.gpuzoo.com/Compare/NVIDIA_Quadro_4000___vs___NVIDIA_Quadro_FX_5600/
	https://ja.wikipedia.org/wiki/FLOPS
	https://en.wikipedia.org/wiki/POWER6
	https://en.wikipedia.org/wiki/Motorola_68881
Tutorials & Manuals	https://www-903.ibm.com/kr/shop/pdf/IBM_Power_Systems_Facts_and_Features_April_2008.pdf
	http://www.prace-ri.eu/IMG/pdf/Best-Practice-Guide-IBM-Power.pdf
Technical Reports	http://staff.itee.uq.edu.au/janetw/Computer%20Go/CS-TR-339.html

Table 4 – Computer Go Data Sources

[2] We directly contacted Computer Go programs developers, teams, and professional players to ask for data related to some programs that we could not find online. Members of the Go community also generously provided significant help (e.g. Discord, Facebook). These interviews did not contain a specific structure and were performed randomly

3.4 Data processing

In this section, we will talk about how the data was processed so that, finally, we could achieve our expected goal (answer our research questions)[3].

There are specific nuances for each of the domains of this work. For Computer Chess, one of our concerns was, first of all, to find a method that would make it possible to convert the $SearchDepth$ and $MIPS$ variables to $Positions/sec$, taking into account that for some of the chess programs the $Positions/sec$ variable was not found. For this, the two regressions specified in subsection 2.2.3 (Formula 2.4 and Formula 2.5) were used. For Computer Chess, there were no setbacks regarding the performance variable, Elo, as it was accessible and reported in the resources used for data collection.

Regarding Go computers, the first challenge encountered was to find a way to convert data related to kyu/dan and handicap rank systems to Elo. This is due to the fact that in the vast majority of data sources visited, the existing data was not directly converted to Elo. Thus, Formula 2.6 was developed for this purpose. The computing power of Computer Go was measured using the flops metric, as mentioned earlier. To get the number of flops that Go's computers were able to calculate, we first looked at all the processors (CPU, GPU, TPU) that were used by these computers. The flop values related to GPUs and TPUs are easily found in the hardware specifications provided by their brands. In opposition to that, computing flops for CPU is a little more tricky, as this data is not reported in the technical specifications of the same. Therefore, the value of flops for all CPUs was calculated manually using the Formula 2.7.

With all the conversions and calculations carried out, the data is properly prepared for the execution of the analyses.

3.5 Data Analysis

After completing the data processing, they are ready for the analysis to be carried out. Initially, for each domain, 3 graphs were plotted with different approaches that allowed us to understand 3 different relationships:

- Performance x Time
- Computing Power x Time
- Performance x Computing Power

These graphs, in a way, corroborated so that we could finally answer the research questions of this work and directly understand the relationship between performance

[3] All phases of data processing are graphically displayed in the diagram in figure 3

improvements and computing power. The performed regressions can be seen from the diagram in Figure 3, as well as graphically from the graphs in Figures 4 and 5.

3.6 Technologies

In this section we describe the technologies adopted to develop this research.

3.6.1 Python

Python is an open-source programming language with an active community and multiple libraries. Python was used in the work to manipulate the data, create tables, create graphs and run the analysis (Python, 2021).

3.6.2 Jupyter

Jupyter is an open-software web application to make documents with live Python code cells. The notebook is used to make the all analysis of this work (Jupyter, 2021).

3.6.3 Seaborn

Seaborn is a Python data visualization library that provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn was used to generate graphs (Seaborn, 2021).

3.6.4 Pandas

Pandas powerful, flexible and easy to use open source data analysis and manipulation tool. It was used to manipulate the data (Pandas, 2021).

3.6.5 Scikit-learn

Scikit-learn is a free software machine learning library for the Python programming language. It was used to perform regressions.

3.6.6 Statsmodels

Statsmodels is a Python module that provides classes and functions for the estimation of many different statistical models, as well as for conducting statistical tests, and statistical data exploration. It was used to generate regression tables (scikitlearn, 2021).

4 Results

The Computer Chess dataset contains data of 46 different computers. Initially, 35 of these computers had data related to Positions/sec. The conversion from search depth and MIPS to Positions/sec allowed an increase of 18.7% of the data, which, finally, enabled the analyzes to be carried out for 43 chess programs. Our data contains observations from 1957 to 2019.

In parallel, for Computer Go, the number of observations that have performance variables and, at the same time, computing power, was equal to 109. Surprisingly, these programs, due to the late development of this domain compared to Computer Chess, belong to the range from 1990 to 2018.

The analysis was performed in Python and coded in the Jupyter notebook. The pandas library was used for all data management. The scikit-learn and statsmodels libraries were used to perform the regressions and statistical analysis of the data. Finally, the seaborn library was used for the creation and styling of the graphics.

The data is currently located in a private repository and will be opened shortly after this work is published, as well as incorporated into the computerprogress.com website.

4.1 Computer Chess

To assess the progress of chess computers over time, we developed an extensive history of computer chess programs from 1957 (Bernstein's program) to 2019 (Komodo 13.1 at the World Computer Chess Championship). From 1957 to 2006 we use matches between computers and humans, as gathered from records from international chess associations, research papers, books, databases provided by the community, and others (see Table 3). From 2006 on we use data from the World Computer Chess Championship, where computers face each other, since at that point computer performance is super-human. Figure 4 shows how the performance of computer chess has evolved.

Since Bernstein's 1957 model, computer chess performance went from a novice Elo score of 800 to a super-human Elo score of 3547 and increased their computing power usage by a factor of 10^8. On average, this means that chess programs improved by 37.6 Elo points per year and increased this computing power used by 38% per year ($=10^{0.14}$).

Analyzing the performance and the number of chess positions shows that an increase of $10\times$ in computing power is associated with an increase in Elo rating of 242 points (statistically significant at the 0.01 level), as shown in Figure 4c. This is similar to esti-

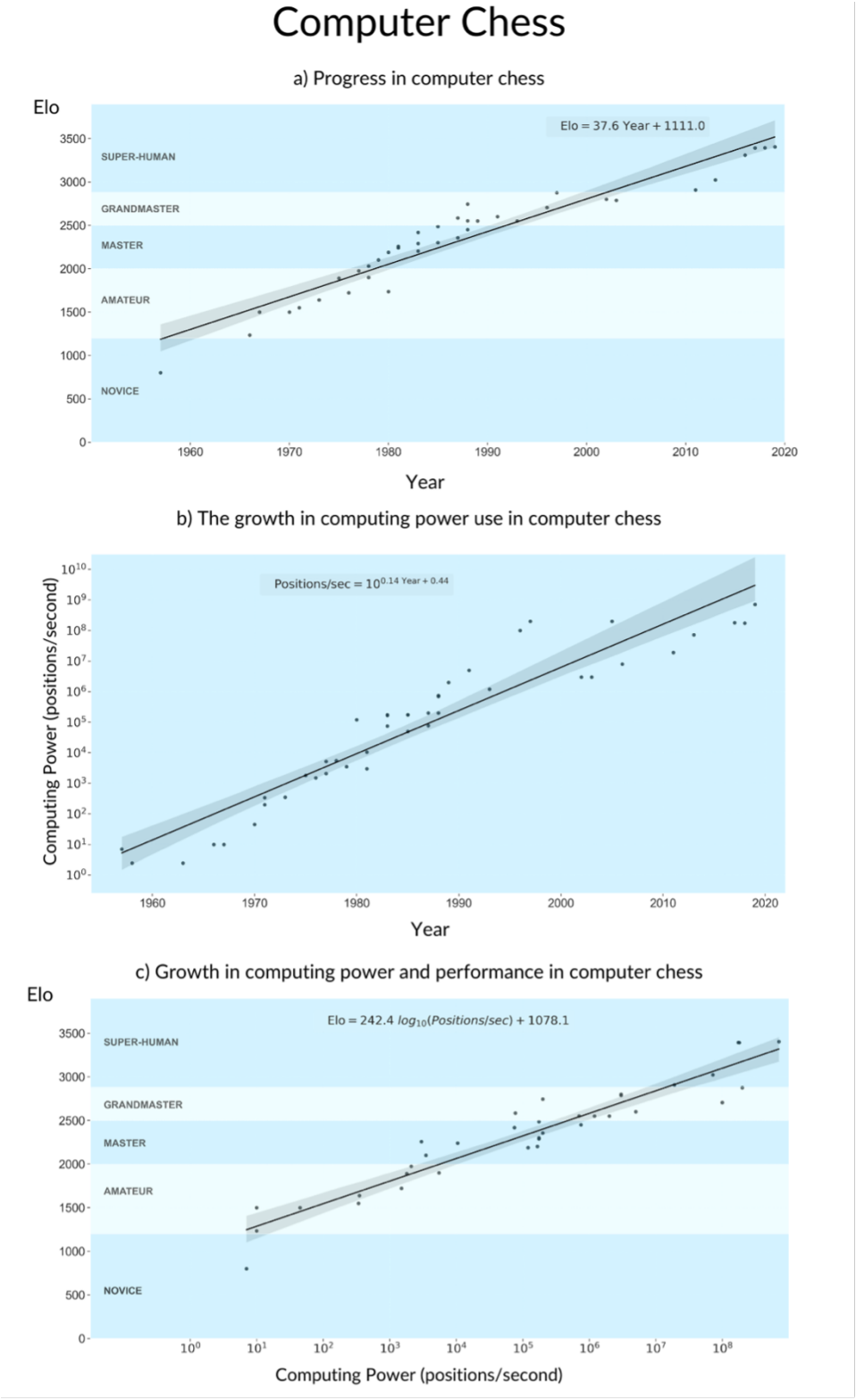

Figure 4 – Computer Chess: (a) Elo scores over time, (b) computing power used over time, and (c) Elo scores as computing power increases. In each subfigure, the dots are individual programs, the lines are linear regressions, and the shaded region is the 95% confidence interval for the regression.

mate made by the Deep Blue team when planning the hardware needed to beat Kasparov "there is a 200-point ELO rating improvement for each order of magnitude improvement

in computing speed of the chess machine platform" (TAN, 1995). The variation in computing power explains 88% of the variation in performance, while the residual variation (e.g. due to algorithmic improvement independent of computing power) only explains 12%.

4.2 Computer Go

To assess how Go programs use computing power and how it is impacting their performance we gathered data on computer versus human games. We sourced this data from the British Go Association, the European Go Federation, research papers, tournament and personal reports, books, databases provided by the community, and others (see Table 4). A key source in this work is the database assemble by Nick Wedd (WEDD, 2018). To fill in missing data, we directly contacted Go developers, teams, professional players and other members of the Go community.

As Figure 5a shows, there has been enormous improvement in computer Go from the 1970s until today, with an average of 84 Elo points being added to performance per year. The best performing systems are now much better than humans, with AlphaGo Zero achieving an Elo of 5135, compared to the human champion Shin Jinseo, who is a 3800 Elo player.

Unfortunately, very little data is available on the computing power used by the early Go systems, so we focus our analysis on the period since 1990, when better data is available. Since that time, the amount of computing power (as measured by floating point operations per second) used by Go programs has increased approximately one hundred billion-fold, a doubling every year ($=10^{0.3}$), as shown in Figure 5b.

Graph (c) compares the growth of computing power in Go with the performance of those programs, revealing a highly significant correlation between them (statistically significant at the p=0.01). In Go, a 10× increase in computing power increases Elo by 244 points on average. In Go, the variation in computing power explains 48% of the variation in system performance.

4.3 Analysis and Discussion

Having presented our case studies graphically, we now answer the research questions (Section 3.1) and explicitly estimating the parameters presented in our theory section, in particular: $\psi = \frac{\partial IT}{\partial t}$ the rate of increase of computing power, γ the exponent for I.T., $\frac{\partial Y}{\partial IT} \cdot \frac{\partial IT}{\partial t}$ the proportion of improvement in performance explainable from improve-

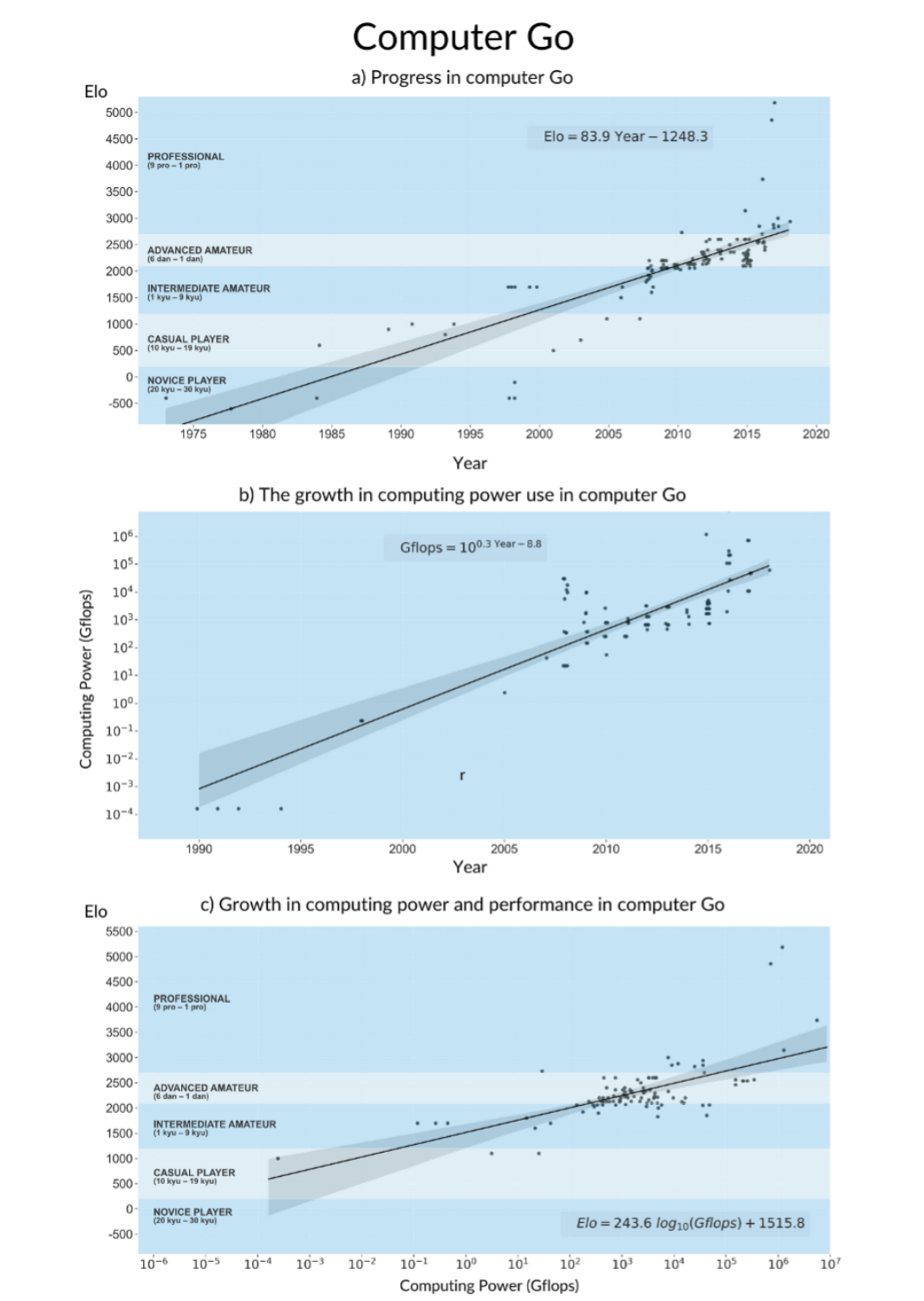

Figure 5 – Computer Go: (a) Elo scores over time, (b) computing power used over time, and (c) Elo scores as computing power increases. In each subfigure, the dots are individual programs, the lines are linear regressions and the shaded region represents the 95% confidence interval for the regression.

ment in I.T.[1], and $\rho \equiv \frac{\partial Y}{\partial t} - \frac{\partial Y}{\partial IT} \cdot \frac{\partial IT}{\partial t}$ the residual portion of growth not explained by

[1] This is proportion of variance in the dependent variable that can be explained by the independent variable or, in other words, the R^2.

increases in computing power.

Throughout this section, we shall use the language of causality in interpreting our results. Normally with time series data, this would be problematic without a natural experiment, instrumental variable or other means of providing statistical identification. But here, we rely not on statistical analysis of our data to get causality, but on the extensive scientific experimentation done by those in these fields. That is, in several areas, engineering and scientific understanding has been built using experiments that prove that computing power causally improves performance (e.g. (DIRECTOR, 2010; NEUMANN et al., 2019). For example, NOAA does test showing the weather forecast improvements it can achieve with more computing power by running a forecast that would need to be calculated in 1 day for a longer time. The results from this testing are then part of the approval process for getting the funding needed to get a computer that could calculate that better performance in the needed amount of time. Thus, it is the experimental testing in these fields, not after-the-fact statistical identification, that we use to get causality[2].

The economic cost of the computing systems used provide a second argument for why the causality runs from computing power to performance: revealed preference (See (The Editors of Encyclopaedia Britannica, 2021b) for more details). As is shown below, there has been an enormous increase in the cost of the computing power used for these problems. Literally, millions of dollars have been spent on these systems because their owners were convinced that causality runs in this direction. Had this not been true, we would have expected the owners of these systems to update their computers to get the benefit of newer hardware while maintaining or lowering costs.

4.3.1 The Growth in Computing Power

To show the growth rate in computing power over time we should look again at the regression $ComputingPower = 10^{\alpha+\beta\times Year}$ that was already graphically presented in Figures 4b and 5b. Table 5 shows the same regressions but now in a table format for easier comprehension and also our estimates for ψ, the rate of increase of computing power across our domains.

As this showed in Table 5, for example for Chess, $\frac{\partial log_{10}(ComputingPower)}{\partial Year} = 0.14$ (statistically significant at p-value $< 1\%$). Exponentiating shows that computer power usage in chess ($\psi = \frac{\partial ComputingPower}{\partial Year}$) is 1.38. That is, the amount of computing used for chess has grown by 38% per year, on average. In Computer GO, this effect is even stronger, with an increasing at 95%.

[2] One potential caveat to this is chess in the period since 1997. Since that time, the cost of computing being used has fallen, suggesting that there may have been less vigilance on the budget and thus on proving additional performance.

	Dependent variables	
	Computer Chess	Computer Go
	$\log_{10}(Positions/sec)$	$\log_{10}(Gigaflops)$
	(1)	(2)
Constant	0.44^{***}	-8.81^{***}
	(0.27)	(0.69)
Year	0.14^{***}	0.29^{***}
	(0.01)	(0.02)
Observations	43	109
R^2	0.88	0.74
Adjusted R^2	0.88	0.74
Residual Std. Error	0.80 (df = 41)	0.93 (df = 107)
F Statistic	324.97^{***} (df = 1; 41)	303.32^{***} (df = 1; 107)
ψ	**1.3842**	**1.9498**
Note:		*p<0.1; **p<0.05; ***p<0.01

Table 5 – Computational Power (Gigaflops) in logarithm scale over year.

4.3.2 Contributions of I.T. to Performance Improvement

Table 6 shows our estimates for γ, the coefficient that describes how increased computing power changes performance across these domains. Here we explicitly model the production function $Y = AL^{\alpha}K^{\beta}IT^{\gamma}$ by taking logs to get our estimating equation: $log(Y) = log(A) + \alpha\ log(L) + \beta\ log(K) + \gamma\ log(IT)$ which, again by our assumptions of the relative speeds of improvement simplifies to $log(Y) = log(A) + \gamma\ log(IT)$.

These show that computing power has, in general, very low exponent γ, ranging from 0.05 for Computer Chess to 0.11 for Computer Go.

As this makes clear, the returns to computing have rapidly decreasing marginal returns[3]. Adding a unit of computation to today's powerful machines has much less impact on outcomes that did ones when computers were brand new. Nevertheless, these results are a powerful statement of the extent to which the exponential increase in computing is needed for computing to contribute meaningfully to productivity improvements.

[3] The decrease in marginal (incremental) output of a production process as the amount of a single factor of production is incrementally increased, holding all other factors of production equal.(Wikipedia contributors, 2021)

	Dependent variables	
	Computer Chess	Computer Go
	$\log_{10}(ELO)$	$\log_{10}(ELO)$
	(1)	(2)
Constant	3.07***	1.51***
	(0.02)	(0.01)
$log_{10}(Computing\ Power)$	0.05***	0.03***
	(0.005)	(0.002)
γ	**0.05**	**0.03**
Observations	31	109
R^2	0.80	0.77
Adjusted R^2	0.80	0.76
Residual Std. Error	0.06 ($df = 29$)	0.03 ($df = 107$)
F Statistic	119.15*** (df = 1; 29)	350.44*** (df = 1; 107)
Note:		*p<0.1; **p<0.05; ***p<0.01

Table 6 – Performance as computation grows in Computer Chess and Computer Go

4.3.3 Analysis of variance

Table 7 consolidates the discussion from our case studies sections about the share of variance in performance explained by computing power. Figure 6 summarizes the R^2 results from these regressions, showing the fraction of the variation explained by computing power (dark) and that from all other factors (light).

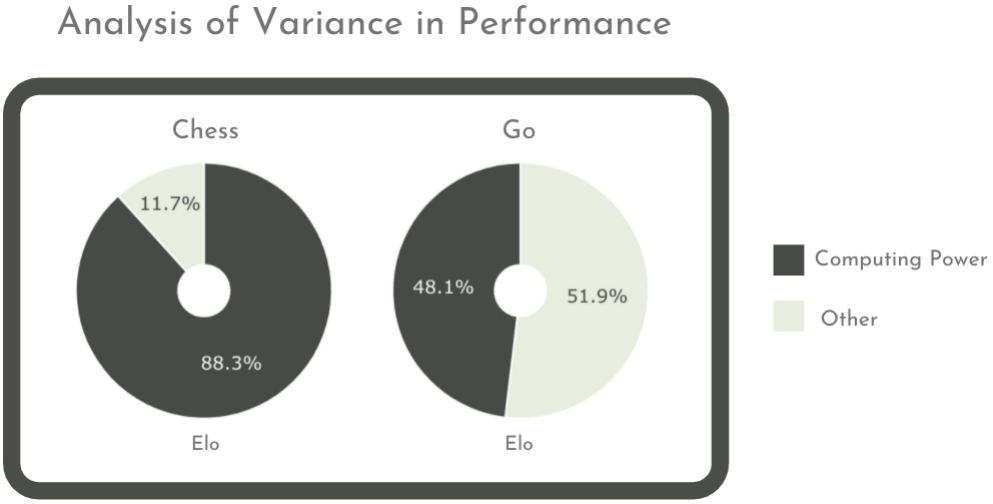

Figure 6 – Analysis of Variance in Computer Chess and Computer Go

As noted earlier, these results suggest that, in many areas, computing power has been overwhelmingly important as a source of gains in performance.

Dependent variables		
	Computer Chess	Computer Go
	ELO	ELO
	(1)	(2)
Constant	$1,078.15^{***}$	$1,515.75^{***}$
	(82.19)	(88.64)
$log_{10}(Computing\ Power)$	242.39^{***}	243.58^{***}
	(16.38)	(25.05)
Observations	31	104
R^2	0.88	0.48
Adjusted R^2	0.88	0.48
Residual Std. Error	180.33($df = 29$)	402.425($df = 102$)
F Statistic	218.94^{***} (df = 1; 29)	94.573^{***} (102)
Note:		*p<0.1; **p<0.05; ***p<0.01

Table 7 – Performance over computation in logarithm scale in Computer Chess and Computer Go

5 Conclusion

During this work, several challenges were overcome. Due to the vast amount of resources in which the data were found, it was impossible for their collection to be carried out automatically. This required a significant consummation of time, just invested in manual data collection. Subsequently, adequate solutions for the treatment of the data were essential for the continuity and achievement of the objective of this work.

Finally, this work shows that computing power has been central to performance improvement in the computing bellwethers of Computer Chess and Computer Go. In both cases, we find that computing power, and implicitly the concurrent improvements which depend on it, account for almost half of all improvement. The size of this contribution is remarkable since computing power in all these areas has only a tiny effect per unit. But with exponential increases in computing power, this can nevertheless be the dominant source of improvements. Besides that, the compound growth rates of computing power have for Computing Chess was equal to 38% and 95% for Computer Go.

Overall, this work paints a coherent picture of computing power improvements as a central driver of progress two computational areas over decades, quantifying long-held views about the centrality of I.T. in general.

Bibliography

ACEMOGLU, D. *Artificial Intelligence, Automation and Work*. [S.l.], 2018. (Working Paper Series, 24196). Disponível em: <http://www.nber.org/papers/w24196>. Citado na página 15.

ACEMOGLU, D. The race between man and machine: Implications of technology for growth, factor shares, and employment. *American Economic Review*, v. 108, n. 6, p. 1488–1542, June 2018. Disponível em: <https://www.aeaweb.org/articles?id=10.1257/aer.20160696>. Citado na página 15.

ASSOCIATION, B. G. *A Brief History of Go*. 2020. [Online; accessed 9-November-2021]. Disponível em: <https://www.britgo.org/intro/history>. Citado na página 26.

Australian Government Productivity Commission. *On Productivity: concepts and measurement*. 2021. [Online; accessed 17-November-2021]. Disponível em: <https://www.pc.gov.au/research/supporting/concepts-measurement>. Citado na página 15.

BANKER, R. D.; KAUFFMAN, R. J. Reuse and productivity in integrated computer-aided software engineering: An empirical study. *MIS quarterly*, JSTOR, p. 375–401, 1991. Citado na página 15.

BBCNEWS. *Google achieves AI 'breakthrough' by beating Go champion*. 2016. [Online; accessed 9-November-2021]. Disponível em: <https://www.bbc.com/news/technology-35420579>. Citado na página 24.

BERLINER, H. Computer chess (monroe newborn). *SIAM Review*, Society for Industrial and Applied Mathematics, v. 18, n. 3, p. 514, 1976. Citado na página 31.

BERNSTEIN, A.; ROBERTS, M. de V. Computer v. chess-player. *Scientific American*, JSTOR, v. 198, n. 6, p. 96–107, 1958. Citado na página 22.

BOROWIEC, S. *AlphaGo seals 4-1 victory over Go grandmaster Lee Sedol*. 2016. [Online; accessed 9-November-2021]. Disponível em: <https://www.theguardian.com/technology/2016/mar/15/googles-alphago-seals-4-1-victory-over-grandmaster-lee-sedol>. Citado na página 24.

British Go Association. *History of Go-playing Programs*. 2018. [Online; accessed 17-November-2021]. Disponível em: <https://www.britgo.org/computergo/history#:~:text=The%20first%20computer%20tournament%20that,Micro%20microcomputers%20on%2013x13%20boards.> Citado na página 24.

BRYNJOLFSSON, E. Paradox lost? firm-level evidence on the returns to information system spending. *Management Science*, v. 42, p. 541–558, 02 1997. Citado na página 15.

CCRL. *Computer Chess Rating Lists*. 2021. [Online; accessed 9-November-2021]. Disponível em: <https://www.computerchess.org.uk/ccrl/>. Citado na página 22.

CHENG, Z. J.; NAULT, B. R. Industry Level Supplier-Driven IT Spillovers. *Management Science*, v. 53, n. 8, p. 1199–1216, August 2007. Disponível em: <https://ideas.repec.org/a/inm/ormnsc/v53y2007i8p1199-1216.html>. Citado na página 15.

Chessify. *NPS - What are the*. Citado na página 23.

COMPUTERWORLD. *Computerworld*. [S.l.]: IDG Enterprise, 1997. Citado na página 22.

DANOWITZ, A. et al. Cpu db: recording microprocessor history. *Communications of the ACM*, ACM New York, NY, USA, v. 55, n. 4, p. 55–63, 2012. Citado na página 19.

DEVARAJ, S.; KOHLI, R. Performance impacts of information technology: Is actual usage the missing link? *Management science*, INFORMS, v. 49, n. 3, p. 273–289, 2003. Citado na página 15.

DIRECTOR, N. Increasing noaa's computational capacity to improve global forecast modeling. Citeseer, 2010. Citado na página 39.

ENERGY, U. S. C. H. C. on Science. Subcommittee on; ENVIRONMENT. *Budget Hearing on FY 1997 Request for DOE, NOAA, and EPA's Office of Research and Development (ORD); and Safe Drinking Water Act R&D Reauthorization: Hearing Before the Subcommittee on Energy and Environment of the Committee on Science, U.S. House of Representatives, One Hundred Fourth Congress, Second Session, March 21, 1996*. U.S. Government Printing Office, 1997. (Budget Hearing on FY 1997 Request for DOE, NOAA, and EPA's Office of Research and Development (ORD); and Safe Drinking Water Act R&D Reauthorization: Hearing Before the Subcommittee on Energy and Environment of the Committee on Science, U.S. House of Representatives, One Hundred Fourth Congress, Second Session, March 21, 1996, v. 2, no. 1). Disponível em: <https://books.google.com.br/books?id=-BeYyNv_W3cC>. Citado na página 19.

European Go Database. *EGF Official ratings system*. 2018. [Online; accessed 17-November-2021]. Disponível em: <https://www.europeangodatabase.eu/EGD/EGF_rating_system_old.php#System>. Citado na página 25.

FOSTER, L. W.; FLYNN, D. M. Management information technology: Its effects on organizational form and function. *MIS quarterly*, JSTOR, p. 229–236, 1984. Citado na página 15.

FREY, P. W. *Chess skill in man and machine*. [S.l.]: Springer Science & Business Media, 2012. Citado na página 31.

GELLY, S. et al. The grand challenge of computer go: Monte carlo tree search and extensions. *Communications of the ACM*, ACM New York, NY, USA, v. 55, n. 3, p. 106–113, 2012. Citado na página 24.

GELLY, S.; SILVER, D. Monte-carlo tree search and rapid action value estimation in computer go. *Artificial Intelligence*, Elsevier, v. 175, n. 11, p. 1856–1875, 2011. Citado na página 24.

GEROW, J. E. et al. Looking toward the future of it–business strategic alignment through the past. *MIS quarterly*, JSTOR, v. 38, n. 4, p. 1159–1186, 2014. Citado na página 15.

GIBNEY, E. Go players react to computer defeat. *Nature News*, 2016. Citado na página 24.

HEINZ, E. A. *Scalable search in computer chess: Algorithmic enhancements and experiments at high search depths*. [S.l.]: Springer Science & Business Media, 2013. Citado na página 31.

HENNESSY, J. L.; PATTERSON, D. A. *Computer architecture: a quantitative approach*. [S.l.]: Elsevier, 2011. Citado na página 27.

HIPPKE. *Measuring hardware overhang*. 2020. [Online; accessed 9-November-2021]. Disponível em: <https://www.lesswrong.com/posts/75dnjiD8kv2khe9eQ/measuring-hardware-overhang>. Citado na página 22.

HIPPKE. *A closer look at chess scalings (into the past)*. 2021. [Online; accessed 9-November-2021]. Disponível em: <https://www.lesswrong.com/posts/4MLBK7iCW3vYd93Mn/a-closer-look-at-chess-scalings-into-the-past>. Citado na página 22.

HITT, L. Productivity, business profitability, and consumer surplus: Three different measures of information technology value. *MIS Quarterly*, Management Information Systems Research Center, University of Minnesota, v. 20, n. 2, p. 121–142, 1996. ISSN 02767783. Disponível em: <http://www.jstor.org/stable/249475>. Citado 2 vezes nas páginas 15 and 19.

HOLDREN, J. P.; LANDER, E.; VARMUS, H. Report to the president and congress: Designing a digital future: federally funded research and development in networking and information technology. *Executive Office of the President and President's Council of Advisors on Science and Technology*, p. 148, 2010. Citado na página 19.

HSU, F.-h. Computer chess, then and now: The deep blue saga. In: IEEE COMPUTER SOCIETY. *International Symposium on VLSI Technology, Systems, and Applications*. [S.l.], 1997. p. 153–154. Citado na página 31.

INSIDER. *Why Go is so much harder for AI to beat than chess*. 2016. [Online; accessed 17-November-2021]. Disponível em: <https://www.businessinsider.com/why-google-ai-game-go-is-harder-than-chess-2016-3>. Citado na página 27.

Jupyter. *Jupyter*. 2021. Disponível em: <https://jupyter.org/>. Citado na página 34.

KASPAROV, G.; FRIEDEL, F. Reconstructing turing's "paper machine". *EasyChair Preprint*, n. 3, 2017. Citado na página 20.

KOCSIS, L.; SZEPESVáRI, C. Bandit based monte-carlo planning. In: . [S.l.: s.n.], 2006. v. 2006, p. 282–293. ISBN 978-3-540-45375-8. Citado na página 24.

LEISERSON, C. E. et al. There's plenty of room at the top: What will drive computer performance after moore's law? *Science*, American Association for the Advancement of Science, v. 368, n. 6495, 2020. Citado 2 vezes nas páginas 16 and 19.

LEVY, D. Computer chess compendium. Springer Science & Business Media, 2013. Citado na página 31.

LEVY, D.; NEWBORN, M. *All About Chess and Computers*. Springer Berlin Heidelberg, 1982. Disponível em: <http://dx.doi.org/10.1007/978-3-642-85538-2>. Citado na página 16.

LEVY, D.; NEWBORN, M. *All About Chess and Computers: Chess and Computers and More Chess and Computers*. [S.l.]: Springer Science & Business Media, 2012. Citado na página 31.

LEVY, D. N. *Computer Games I*. [S.l.]: Springer Science & Business Media, 2012. Citado na página 31.

MORAVEC, H. P. *Robot: Mere machine to transcendent mind*. [S.l.]: Oxford University Press on Demand, 2000. Citado na página 31.

MUOIO, D. *AI experts thought a computer couldn't beat a human at Go until the year 2100*. 2016. Accessed: 2021-11-09. Disponível em: <https://www.businessinsider.com/ai-experts-were-way-off-on-when-a-computer-could-win-go-2016-3#:~:text=AI%20experts%20thought%20it%20could,thread%20by%20user%20Yull%2DBan.> Citado na página 16.

MURRAY, H. *A History of Chess: The Original 1913 Edition*. Skyhorse, 2015. ISBN 9781632207708. Disponível em: <https://books.google.com.br/books?id=dNSBCgAAQBAJ>. Citado na página 20.

NEUMANN, P. et al. Assessing the scales in numerical weather and climate predictions: will exascale be the rescue? *Philosophical Transactions of the Royal Society A*, The Royal Society Publishing, v. 377, n. 2142, p. 20180148, 2019. Citado na página 39.

NEWBORN, M. *Kasparov versus Deep Blue: Computer chess comes of age*. [S.l.]: Springer Science & Business Media, 2012. Citado na página 31.

NEWBORN, M.; NEWBORN, M. *Deep Blue: an artificial intelligence milestone*. [S.l.]: Springer Science & Business Media, 2003. Citado na página 31.

NILSSON, N. J. *The quest for artificial intelligence*. [S.l.]: Cambridge University Press, 2009. Citado na página 31.

Pandas. *Pandas*. 2021. Disponível em: <https://pandas.pydata.org/>. Citado na página 34.

PEARSON, J. P. *Digital at work: Snapshots from the first thirty-five years*. [S.l.]: digital press Burlington, MA, 1992. Citado na página 31.

PINSONNEAULT, A.; RIVARD, S. Information technology and the nature of managerial work: From the productivity paradox to the icarus paradox? *MIS quarterly*, JSTOR, p. 287–311, 1998. Citado na página 15.

Python. *Python*. 2021. Disponível em: <https://www.python.org/>. Citado na página 34.

REILLY, E. D. *Concise encyclopedia of computer science*. [S.l.]: John Wiley & Sons, 2004. Citado na página 31.

Ribeiro, John. *AlphaGo's unusual moves prove its AI prowess, experts say*. 2016. Disponível em: <https://www.pcworld.com/article/420054/alphagos-unusual-moves-prove-its-ai-prowess-experts-say.html>. Citado na página 27.

ROMER, P. *Endogenous Technological Change*. [S.l.], 1989. (Working Paper Series, 3210). Disponível em: <http://www.nber.org/papers/w3210>. Citado na página 15.

ROMER, P. *Are Nonconvexities Important For Understanding Growth?* [S.l.], 1990. (Working Paper Series, 3271). Disponível em: <http://www.nber.org/papers/w3271>. Citado na página 15.

ROMER, P. M. Increasing returns and long-run growth. *Journal of Political Economy*, University of Chicago Press, v. 94, n. 5, p. 1002–1037, 1986. ISSN 00223808, 1537534X. Disponível em: <http://www.jstor.org/stable/1833190>. Citado na página 19.

ROUGETET, L. Aux origines de deep blue: une histoire de la programmation du jeu d'échecs. *Ligeia*, Éditions Ligeia, n. 1, p. 183–197, 2019. Citado na página 31.

SABHERWAL, R.; JEYARAJ, A. Information technology impacts on firm performance. *MIS quarterly*, JSTOR, v. 39, n. 4, p. 809–836, 2015. Citado na página 15.

SCHAEFFER, J.; MARSLAND, T. A. *Computers, Chess and Cognition*. [S.l.]: Springer, 1990. Citado na página 31.

scikitlearn. *scikitlearn*. 2021. Disponível em: <https://scikit-learn.org/>. Citado na página 34.

Seaborn. *Seaborn*. 2021. Disponível em: <https://seaborn.pydata.org/>. Citado na página 34.

SFETCU, N. *Gaming Guide-Gambling in Europe*. [S.l.]: Nicolae Sfetcu, 2016. Citado na página 31.

SHANNON, C. E. Xxii. programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, Taylor & Francis, v. 41, n. 314, p. 256–275, 1950. Citado na página 20.

SHERRY, Y.; THOMPSON, N. *How fast do algorithms improve*. [S.l.], 2020. Citado na página 19.

SILVER, D. et al. Mastering the game of go without human knowledge. *nature*, Nature Publishing Group, v. 550, n. 7676, p. 354–359, 2017. Citado 3 vezes nas páginas 24, 26, and 27.

SILVER, D.; SUTTON, R.; MüLLER, M. Temporal-difference search in computer go. *Machine Learning*, v. 87, 05 2012. Citado na página 27.

SYVERSON, C. What determines productivity? *Journal of Economic literature*, v. 49, n. 2, p. 326–65, 2011. Citado na página 19.

TAN, C. Deep blue: computer chess and massively parallel systems. In: *Proceedings of the 9th international conference on Supercomputing*. [S.l.: s.n.], 1995. p. 237–239. Citado na página 37.

The Editors of Encyclopaedia Britannica. *Production functions*. 2021. Disponível em: <https://www.britannica.com/topic/production-function>. Citado na página 19.

The Editors of Encyclopaedia Britannica. *Revealed Preference Theory*. 2021. Disponível em: <https://www.britannica.com/topic/revealed-preference-theory>. Citado na página 39.

THEECONOMIST. *Showdown*. 2016. [Online; accessed 9-November-2021]. Disponível em: <https://www.economist.com/science-and-technology/2016/03/12/showdown>. Citado na página 27.

THOMPSON, N. C. et al. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 2020. Citado na página 15.

THOMPSON, N. C.; SPANUTH, S. The decline of computers as a general purpose technology. *Communications of the ACM*, ACM New York, NY, USA, v. 64, n. 3, p. 64–72, 2021. Citado 2 vezes nas páginas 16 and 19.

WALL, B. B. *The MacHack Attack by Bill Wall*. 2008. [Online; accessed 9-November-2021]. Disponível em: <http://billwall.phpwebhosting.com/articles/machack.htm>. Citado 2 vezes nas páginas 22 and 24.

WEDD, N. *A list of Computer Go Tournaments*. 2018. [Online; accessed 9-November-2021]. Disponível em: <http://www.computer-go.info/>. Citado na página 37.

Wikipedia . *Dan (rank) — Wikipedia, The Free Encyclopedia*. 2021. [Online; accessed 17-November-2021]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Dan_(rank)&oldid=1051122370>. Citado na página 24.

Wikipedia . *Elo rating system — Wikipedia, The Free Encyclopedia*. 2021. [Online; accessed 17-November-2021]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Elo_rating_system&oldid=1055352041>. Citado na página 22.

Wikipedia . *Game complexity — Wikipedia, The Free Encyclopedia*. 2021. [Online; accessed 17-November-2021]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Game_complexity&oldid=1051866381>. Citado na página 27.

Wikipedia . *Handicapping in Go — Wikipedia, The Free Encyclopedia*. 2021. [Online; accessed 17-November-2021]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Handicapping_in_Go&oldid=1050990909>. Citado na página 24.

Wikipedia . *Macroeconomics — Wikipedia, The Free Encyclopedia*. 2021. [Online; accessed 17-November-2021]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Macroeconomics&oldid=1050943806>. Citado na página 15.

Wikipedia . *Microeconomics — Wikipedia, The Free Encyclopedia*. 2021. [Online; accessed 17-November-2021]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Microeconomics&oldid=1055146100>. Citado na página 15.

Wikipedia . *National Oceanic and Atmospheric Administration — Wikipedia, The Free Encyclopedia*. 2021. [Online; accessed 9-November-2021]. Disponível em: <https://en.wikipedia.org/w/index.php?title=National_Oceanic_and_Atmospheric_Administration&oldid=1051327326>. Citado na página 19.

Wikipedia contributors. *Diminishing returns — Wikipedia, The Free Encyclopedia*. 2021. [Online; accessed 17-November-2021]. Disponível em: <https://en.wikipedia.org/w/index.php?title=Diminishing_returns&oldid=1054880613>. Citado na página 40.