

PEDRO GOMES BRANQUINHO

Free software in industry and academia

Lorena

2021

PEDRO GOMES BRANQUINHO

Free software in industry and academia

This monograph is presented to the Engineer School of Lorena, the University of São Paulo, so to be obtained the title of Bachelor by the Graduation Program on Engineering Physics with emphasis on the Science of Materials.

University of São Paulo - USP

Engineer School of Lorena

Monograph, Conclusion Thesis

Supervisor: Dr. Wei-Liang Qian

Lorena

2021

PEDRO GOMES BRANQUINHO

Free software in industry and academia/ PEDRO GOMES BRANQUINHO. – Lorena, 2021-

Supervisor: Dr. Wei-Liang Qian

Dissertation – University of São Paulo - USP

Engineer School of Lorena

Monograph, Conclusion Thesis , 2021.

1. Free Software. 2. Open Source. 2. Academy and Industry. I. Wei-Liang Qian. II. University EEL-USP. III. Escola de Engenharia de Lorena. IV. Free Software on Industry and Academia.

*To those whom found me in their own path
and, by finding me, made part of my own.*

Acknowledgements

My acknowledgments wouldn't fit in a single page. But, for the purpose of conciseness, I will mention those who are closer to my work. I thank first my advisor, Wei-Liang Qian, who in his patience and kindness knew how to conduce me to produce the present work. I thank Juan Zapata for the support and enthusiasm on teaching Mathematics and showing me the way of how to study it myself. Last but not least, I thank Luiz Eleno who has been a role-model for me, and has teach me so much about computing through out the years.

And, in a big umbrella, I thank all those anonymous people who have contributed for my experience of communal sharing and understanding in the Open Source community. Specially, David Wilson, the founder of System's Crafter, from whom I derived the basis of my Emacs's system.

Abstract

In this work, It's shown how to build a series of application only upon a Free Software system - EXWM, Artix Linux OSS. I explain how the experience of participating in the Open Community can be significant for other Engineers; specially Physics Engineers. It's delineated what are the current trends on the adoption of Free and/or Open Source Software (FOSS). Furthermore, I put forward some of the tools I used in Academia, and in Industry, and some other not so well known software, which could be used in these two contexts - e.g., Freqtrade, OR-Tools, Git(hub) et cetera. I also argue why Linux is such a key software for someone shifting to the Open Source paradigm.

Key-words: trends. foss. academia. industry. linux. freqtrade. or-tools. git. github.

Resumo

Demonstrou-se como é possível construir uma série de aplicações baseada em softwares de licença livre, à partir de um sistema aberto, o Linux com interface EXWM - Emacs X Window Manager. Além disso, foi propiciado casos reais de aplicações na Indústria e no investimento privado, autônomo. Bem como, utilizações na Academia, à nível de lecionar, e pesquisa. Sustenta-se que a economia aberta possui similaridade estrutural ao movimento *Open Source* e seu desenvolvimento, o que aponta que essa é e continuará a ser, paulatinamente mais, o paradigma de desenvolvimento econômico tecnológico. Assim, imprescindível à formação do engenheiro.

Palavras-chaves: software livre. automação. freqtrade. indústria. academia.

List of Figures

Figure 1 – Schema of a tower of interpreters	13
Figure 2 – Categorization of the study of towers of interpreters	14
Figure 3 – Towers of interpreters and the Operational Systems.	14
Figure 4 – Genealogy of Linux’s Distributions	17
Figure 5 – EXWM - Emacs X Window Manager	18
Figure 6 – a simplified schema of client-server communication	21
Figure 7 – Schematic representation of client-server communication.	21

List of abbreviations and acronyms

FOSS	Free and Open Source Software
IDE	:Integrated Development System
abnTeX	ABsurdas Normas para TeX

Contents

1	INTRODUCTION	11
1.1	Objective	11
1.2	The interconnection between Applications and the OS	12
1.2.1	Why does GNU/Linux matter?	12
1.2.2	How high level applications benefit from an OS	12
1.3	On the influence of education in adoption	14
1.4	Performance and the current trend as reasons for adoption	15
1.5	The set-conjunction between physics engineer and FOSS's users	15
1.6	How to leverage the potencial of OSS in Industry and Academia	15
2	BIBLIOGRAPHY REVIEW	16
2.1	Open Source	16
2.1.1	Diversity	16
2.2	GNU/Linux	17
2.2.1	Historical Origin	17
2.2.2	Emacs	18
2.3	Performance comparison among OS adoption	18
2.3.1	Performance	19
2.3.2	FOSS adoption demography	19
2.4	Industry Application Demonstration	20
2.4.1	Freqtrade	20
2.4.2	OR-Tools	22
2.4.2.1	Application on Economics	23
2.4.2.2	Application to environmental policies	23
2.4.2.3	Application to other areas	24
2.5	Demonstration in Academic Applications	25
2.5.1	DifferentialEquations.jl	25
2.5.1.1	Julia portability	25
2.5.1.2	Python portability	26
2.5.1.3	R portability	26
2.5.2	L ^A T _E X	26
2.5.2.1	The canonical ABNT class for scientific production	27
2.6	Canonical works on Computer Science	28
2.6.1	Structure and interpretation of computer programs (SICP)	28
2.6.2	SCIMUtils - (SCIM) Portability in Clojure	28

3	MATERIALS AND METHODS	30
3.1	Initial Strategy	30
3.2	Theme delineation	30
3.3	Cronological Schedule	30
3.4	The monograph theme	31
3.5	Gathering various notes	31
3.6	Bibliographical Research	31
3.7	Continuous Review	31
3.8	Presentation Organization	32
4	RESULTS AND DISCUSSION	33
4.1	Advisor expertise and the theme	33
4.2	Theme delineation	33
4.2.1	Industry Applications	33
4.2.1.1	OR-Tools	33
4.2.1.2	Freqtrade	38
4.2.2	Academia Applications	38
	BIBLIOGRAPHY	39

1 Introduction

In training a Physics Engineer, of which, by definition, is a generalist professional. The use of Free and Open Sourced Softwares (FOSS), as well as the participation in the Open Source community (OS), are two detrimental experiences to have.

The variability, which open source software (OSS) brings to existence of applications and it's extension, can change altogether user's experience. Thus, taking him closer to acting as a developer. This experience of interloping user and developer roles doesn't require that you are a computer scientist or a Information Technology (IT) professional by training. For, programming can be seen as both a Science and an Art ([KNUTH, 1968](#)) - e.g., an exercise of self-expression.

OSS guarantees four fundamental liberties [section 2.1](#), the right to study, copy, modify and redistribute it.

Just as the scientific enterprise benefits, with it's rapid development, by means of the global community's participation, which holds space for individuals with a variety of different training. Also, the computation enterprise benefits from the variety of people's training, which constitute the body of the open source community.

1.1 Objective

We will demonstrate the vality of the hypothesis that a engineer professional, without strong training in computation, lags behind it's current potential. Furthermore, we present a range of different applications developed to the state of the art, which are distributed under open licenses (copy-left). Thus, reinforcing the uniqueness of open source phenomena and the need for it's use - the logic and dynamic FOSS brings has no parallel on the economy or scientific community ([HIPPEL; KROGH, 2003](#); [PETERS, 2009](#)).

There exits debate around the meaning of *Open Science*. This term recently became popular and which has a obvious reference to the *Open Source*. Although, in the social literature, it's a phenomena poorly depicted and rarely debated through the point view that the Open Source Movement has anything to do with it. We cite the most cited article on Google Scholar research on the topic "Open Science" - called "The future(s) of open science" - and which only uses trice the term "Open Source" and in dismissive way ([MIROWSKI, 2018](#)).

Therefore, I argue that a strong basic knowledge, for engineer training, is fundamental for understanding the movement and how it has been shaped, so to critically asses the validity of the current social-economic shift we live in. This understand can, in turn, shape

the carer path and formation of the working engineer.

1.2 The interconnection between Applications and the OS

The author used an **Application** as to comprise any end product of software development.

1.2.1 Why does GNU/Linux matter?

We will discuss, as a brief introduction, what is the Operational System (OS) GNU/Linux, and why it's the *de facto* opening door to the Open Source Community. Firstly, the GNU/Linux is the first and most successful project carried out in the Open Source paradigm (TU et al., 2000; WEST; DEDRICK, 2001). Therefore, it's use is a way to acquaintance, in practical terms, with how a business dependent on mainly using the open sourced development products might operate (FINK, 2003).

Furthermore, the new user-developer of the GNU/Linux framework must inherently learn about the accompanying software which comes with it's distribution - which increases it's chance of adoption (WEST; DEDRICK, 2001). This way, the user gradually becomes accustomed to participate on development and extend programs (HERTEL; NIEDNER; HERRMANN, 2003).

Beyond it's initial philosophical appeal to user liberty, GNU/Linux is today's standard in Technological Enterprises. And, although fundamentally opposed of many age-old *modus operandus* of the *status quo* of companies, there is no doubt left under analysis of the benefits it brings to the companies, general economy, and the society triad (MOODY, 2009). To make use of GNU/Linux, thus, is a way to be inserted in this new economic paradigm (HIPPEL; KROGH, 2003; PETERS, 2009).

1.2.2 How high level applications benefit from an OS

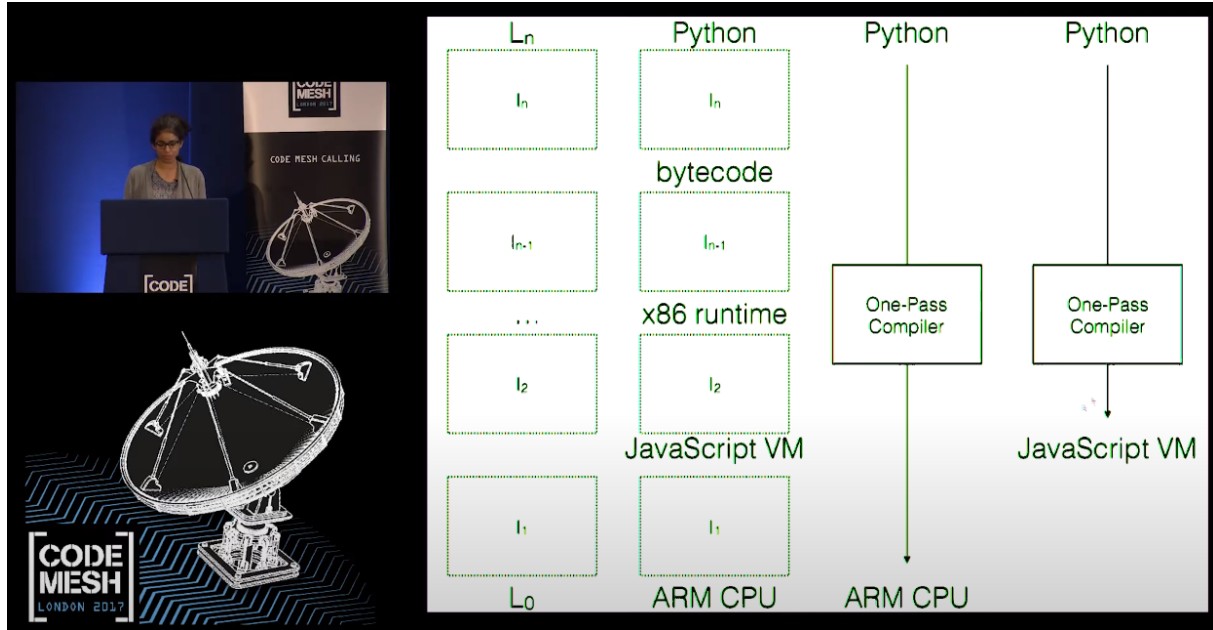
In the hierarchy of software and applications, the Operational Systems (OSes) can be seen as a meta-application or meta-software.

“The evaluator, which determines the meaning of expressions in a programming language, is just another program.” (ABELSON; SUSSMAN, 1996)

There exists levels, or layers, of abstractions in virtually any application. That is, the concept of meta-programming and Towers of Interpreters comprise a common situation, for which a devoted field of study exists. Thus this area has direct implication for software development practice, as it's a ubiquitous problem faced in computing.

Any OS, as the GNU/Linux, comprise an essential layer in this tower of interpreters. Particularly, an OS communicates with *firmwares* - low-expressivity and highly-performing software, which control *hardware*. Also, they communicate with high-expressivity software, among which contain the user-developer written or extended software. Therefore, the OS play a fundamental role, mediating between low and high level software. This function categorizes them as a *middleware* software.

Figure 1 – Schema of a tower of interpreters



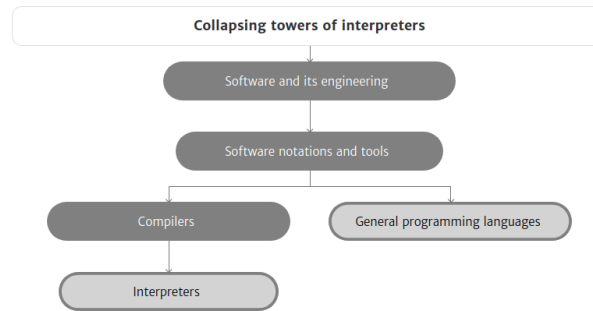
Code Mesh, presentation “Towers of Interpreters”, by Nada Amin

The characteristic problem of concatenate a system of software, one on top of the other, introduces complexity to maintaining compatibility among program’s versions and it’s performance. The study of these behavior and it’s theoretical solutions posses a field of it’s own. And, this field is autonomous, detached, for an example, from which languages compose the Tower of Interpreters; or which type of application we are dealing with (AMIN; ROMPF, 2017). The object of study is the final behavior of the system, and if it’s a collapsible system.

Finally, the OSES consist in a big tower-collapser of interpreters. They are subordinate to collapsing firmware, middleware and high-level software. Therefore, as well as the OS conduct this task, as much the user-developer experience is facilitated.

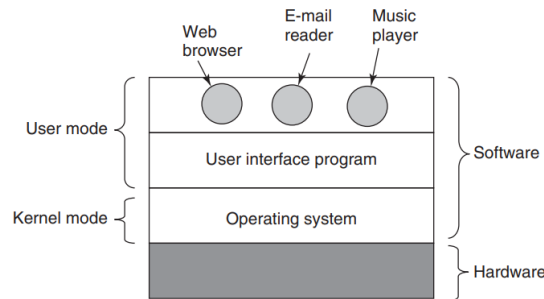
If every application programmer had to understand how all these things work in detail, no code would ever get written. Furthermore, managing all these components and using them optimally is an exceedingly challenging job. For this reason, computers are equipped with a layer of software called the operating system, whose job is to provide user programs with a better, simpler, cleaner, model of the computer and

Figure 2 – Categorization of the study of towers of interpreters



Reference: (AMIN; ROMPF, 2017)

Figure 3 – Towers of interpreters and the Operational Systems.



Reference: (TANENBAUM; BOS, 2015)

to handle managing all the resources just mentioned. (TANENBAUM; BOS, 2015)

1.3 On the influence of education in adoption

In the bibliographical literature, it's clear that the rate of OSS adoption in the Industrial sector - commonly refereed as "Production" - depends heavily on both: competency and the level of expertise a project require (LI; TAN; YANG, 2013; GALLEGO et al., 2015; SPINELLIS; GIANNIKAS, 2012).

At the same time, the adoption depends directly on the intrinsic inclination of the Informational Technology (IT) team (RACERO; BUENO; GALLEGO, 2021).

Therefore, regardless of how much a professionalizing course may increase the *intensity* and *quickness* of adoption. Data shows that, generally, those students and professionals positively correlated to seeking autonomy would adopt it (RACERO; BUENO; GALLEGO, 2020). This means that intrinsic motivation is key (GALLEGO et al., 2015). Nonetheless, it's important to notice that there exist a net-effect in adoption (SPINELLIS; GIANNIKAS, 2012). e.g., how much more peers adopt it, the more likely to any given

individual to adopt it.

1.4 Performance and the current trend as reasons for adoption

Research with different areas of benchmarks state a performance gain, when utilizing GNU/Linux, compared to Windows (SULAIMAN; RAFFI, 2021). Although, even more important, the key benefits are not in the differential performance per se, but in the training one naturally goes through upon utilizing a totally community-dependent Operational System.

Thereof, using GNU/Linux is a door for an individual professional shift. At the same time, this personal use increases the probability of adoption in whichever Industry carrier one may lead (HAUGE; SØRENSEN; CONRADI, 2008). Combined with that fact, the Industry per se has no observable effect on Open Source development of projects - as so far as measured in 2008 (HAUGE; SØRENSEN; CONRADI, 2008).

1.5 The set-conjunction between physics engineer and FOSS's users

We note that the deepness of training proposed, in graduation level, for a physics engineer makes them perfect candidates for the use of FOSS. Because, both trainings imply a *a priori* necessity for autonomy and purposeness (SCHRAPE, 2019; RACERO; BUENO; GALLEGO, 2020); imply a profound and will for generalistic technical knowledge (LI; TAN; YANG, 2013; GALLEGO et al., 2015).

1.6 How to leverage the potencial of OSS in Industry and Academia

In the present work, I utilized of many concept demonstrations, in which the autor has developed or/and extended applications, in a context of free and open source. Also, I present how can one collaborate in the community and how does that collaboration can imply significant professional connections. This way, both the so called “soft skills” and “hard skills” benefits have been elucidated, in practice.

2 Bibliography review

2.1 Open Source

Any program which permits the user-developer to have the following liberties:

1. The right to run the program, as seen fit, for any end.
2. The right to access the source code and study it.
3. The right to copy and redistribute it.
4. The right to modify the software.

Practically, the Open Source community fundamentally base itself upon the free distribution of it's tools and programs. One of the differential advantage of having innumerable other people extending the same software is that the advancement of the frontier of the program, in many directions, increases rapidly in relation to a program controlled by a limited number of programmers.

2.1.1 Diversity

Given that one fundamental right of OSS is the modification and propagation of new modified versions. This right implies in the observable wide range of maintained versions of these software, which doesn't have a parallel in any other technological enterprise.

For an example, one key application in any user's computer is a general Graphical User Interface (GUI)'s manager, commonly known as Window Manager (WM). These can be both Floating or Tilling, or mixed WM, e.g., Floating WM are those that the user must hover windows and adjust them manually; Tilling WM are those that a pre-defined program have a set of rules to resize automatically the windows in a screen.

While private Operational Systems (OS), as Windows and MacOS, have frequent releases - a total of twenty five releases for Windows. Generally, they've few *active* versions; Windows have currently four ([MICROSOFT...](#), 2021). MacOS also have four active versions ([MACOS...](#), 2021).

The fact there are only narrowly supported versions is due to, among many contributing factors, users lack the right to alter and extend the software's behavior. Therefore, they fall victims of discontinued support and restrictive access to the company's official upgrades.

On the other hand, there exists, in parallel, around two hundred seventy eight available distributions of Linux ([LINUX...](#), 2021). Of which, there are main/root distributions, which each embody a set of different principles; theoretical and practical philosophies of

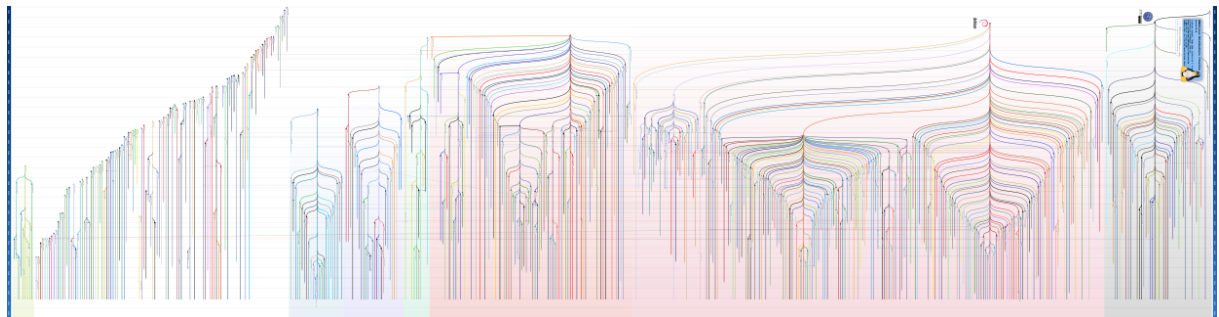
how to extend software.

Thus, just as with any other scope of software, the variability of FOSS always will be grater than monopolized ones.

2.2 GNU/Linux

There are root distributions of Linux, from which many other distributions emanate. Generically, these partitions are called families. We cite some of the most influential and popular ones, Red Hat Linux (☛), Debian (©), CentOS (⚙), Fedora (Ⓟ), Pacman-based (⚡/Ⓜ), OpenSUSE (☘), Gentoo-based (☹), Ubuntu-based (☺), Slackware (Ⓢ), Open Sourced-based and the Independent Distributions (☹/Ⓜ).

Figure 4 – Genealogy of Linux’s Distributions



Genealogical history of Linux Distributions ([LINUX...](#), 2021)

2.2.1 Historical Origin

The GNU/Linux began as two separeted and different directions. GNU stands for “GNU’s Not Unix”, a recursive name. And GNU initialy has been developed as a collaboration of revolted academics by the restrictive secure system of the MIT Lab (Laboratory of Compuer Science - LCS) ([STALLMAN](#), 2002; [EMACS...](#), 2021). Amongst them, there was the still active Richard Stallman, which heavily worked on the text editor of the time - already ten years into development. This editor became Emacs ([EMACS...](#), 2021).

Parallel to these events, Linus Torvalds had been developing an open portatible op-erational system, as his master’s thesis ([TORVALDS](#), 1997).

Finally, both projects united in a symbiotic system, of which the OS was Linux (the formentined portible kernel) and the GNU’s interface program whit all utilities one may have had in their computers at the time ([STALLMAN](#), 1997)

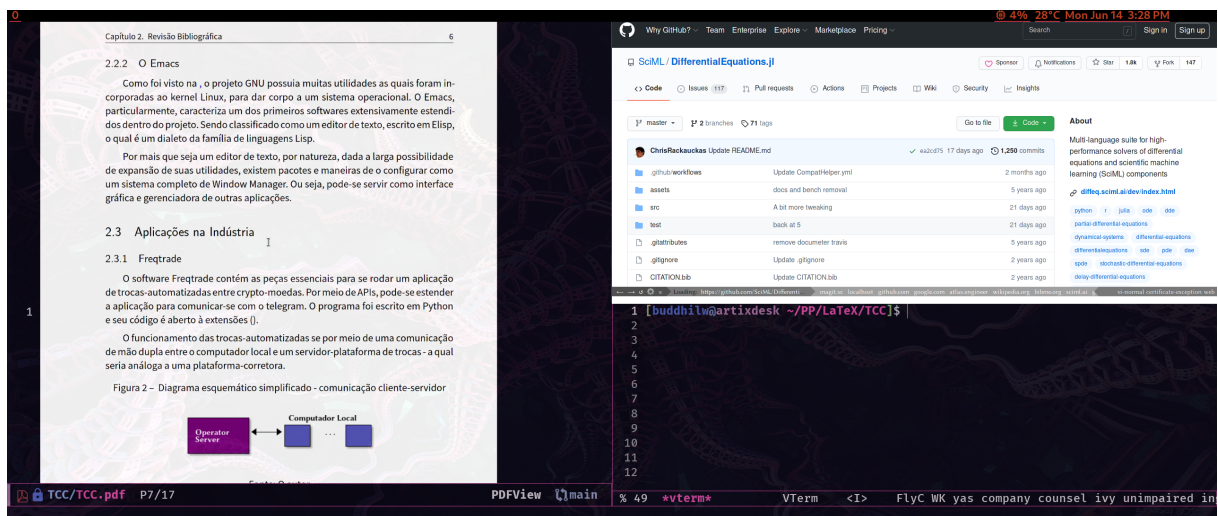
2.2.2 Emacs

As has been seen in , the GNU project already had developed a variety of applications, all of which, incorporated into Linux. This way, the OS gained a “body”. The Emacs, particularly, characterize one of the first software extensively extended, as a project, in a open community.

Although, the label given to Emacs as an “editor” covers it’s main function. Actually, the fundamental role of Emacs equates to evaluating Elisp expressions. Elisp as in a dialect of Lisp. Therefore, as an interpreter of a language, it has Turing complete capabilities. So it has a complete-system capability ¹.

Even though Emacs usual use has been of an Integrated Development System (IDE), by it’s unlimited potentiality and expressiveness, there exists packages and applications written in Elisp to become a fully featured Window Manager (WM). That is, Emacs, through Emacs’s X Window Manager (EXWM), can serve as a graphical and manager interface to other applications.

Figure 5 – EXWM - Emacs X Window Manager



Source: author’s WM ambient.

Figure 5 exemplifies a desktop environment which can render images, PDFs, Browsers et el, through Emacs.

2.3 Performance comparison among OS adoption

operacionais

¹ The GNU/Guix implementation of Linux, has been implemented in Scheme - a cousin with better performance than Elisp

2.3.1 Performance

In user-level applications, the efficiency and processing benchmarks favor Linux over Windows (SULAIMAN; RAFFI, 2021).

In the same hardware, the underground/underlying programs maintaining a stationary Windows consume around 5% of CPU and 41% of RAM. In the mean time, Linux Mint - a popular distribution of Linux - consumed 1.8% of CPU and 24% of RAM. A pronounced difference of more than 200% in CPU use efficiency and approximately 200% in RAM efficiency (SULAIMAN; RAFFI, 2021).

A VBS-program script execution ² amounts to a difference in time-execution of 4.249 seconds. In which, Linux Mint took 0.501 seconds, against a 4.75 seconds for Windows. E.g., there exist a roughly $\frac{4.75-0.501}{0.501} = 423\%$ in time-execution performance (SULAIMAN; RAFFI, 2021).

Lastly, there also exists other researches done with technical rigor in a broad spectrum of other applications and hardware. To name a few, in wireless (S.DEVAN, 2013); parallelism and management of server applications (AVELEDA et al., 2010); scientific programming (Fast Fourier Transform) et al (D'ELIA; PACIELLO, 2011); performance on Virtual Reality (VR) application (THUBAASINI; RUSNIDA; ROHANI, 2010) etc. All of which present different kinds of advantages for Linux-use compared to Windows. Although, against what is expected in view of Industry sector adoption, Linux has almost no difference, and sometimes can be worst depending of memory usage of the application (RISTOV; GUSEV, 2013), than Windows performance on multi-threaded and parallel systems, typically founded in cloud/cluster server use-case (AVELEDA et al., 2010).

2.3.2 FOSS adoption demography

Fritzgerald enunciated, in 2006, that the adoption profile of users of Open Source Software (OSS) passed through a phase “OSS 2.0”. This phase was characterized as a transcendence from the mentality that the only people who would be able to use the system would be “hackers” (FITZGERALD, 2006). And, furthermore that such OS were even becoming “mainstream”. That is, it became a common place in society and industry. He also professed the hypothesis that OSS project’s activities were significant influenced due to big companies.

In 2008, another research tried to quantitatively verify the hypothesis. The data was obtained by questioners and interviews of Finland software industries. In contrast to the Fritzgerald’s characterization, 50% of the companies utilized vastly FOSS, although, they had negligent impact of the development carried by the community. Lastly, more than 30% of these companies told that at least 40% of their companies value on software was

² An application concerning the labeled “Office” bundle.

due to services done by the community (HAUGE; SØRENSEN; CONRADI, 2008). Also relevant, these were mostly small to medium size enterprises.

In 2012, a study on big companies use of OSS - a thousand companies listed on US Fortune magazine - have derived some conclusions (SPINELLIS; GIANNIKAS, 2012):

- Adoption is directly related to IT's team need for expertise on the work they would partaken, as well as efficient cautiousness (GALLEGO et al., 2015; LI; TAN; YANG, 2013).
- Exponential growth on adoption, once the existence of an accumulation of users in the company.
- There exist a network effect and the effect is notorious in big companies.

Therefore, adoption in small, medium and big companies process vary, and there are no salient effects caused by companies on the expansion of open source projects. Nonetheless, these companies still benefit from the later (SPINELLIS; GIANNIKAS, 2012; HAUGE; SØRENSEN; CONRADI, 2008; FITZGERALD, 2006). Still, the most relevant positive-effect on the OSS and community is due to small enterprises (KSHETRI, 2004).

Regarding the expansion of the FOSS development paradigm and ecosystem, undoubtedly there is an grow and ubiquity of adoption (SCHMIDT, 2016). Moreover, FOSS has been considered in economical papers as the innovatory cradle of software industry (SCHRAPE, 2019; SCHMIDT, 2016).

More recent studies, dating to 2021 and 2020, determine that, independent of a company size, adoption depends directly on training of IT and autonomous inclination towards these tools (RACERO; BUENO; GALLEGO, 2020). Finally, individual adoption and belief on the effectiveness of these software depend primarily on intrinsic factors, and is independent of previous training. Even though, training on OSS notoriously escalates those users inclined for it (RACERO; BUENO; GALLEGO, 2021).

2.4 Industry Application Demonstration

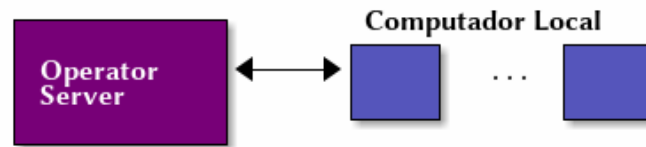
2.4.1 Freqtrade

The *Freqtrade* software contains the essential pieces for a working-application of automated trading of crypto-currencies. By means of APIs, it's possible to extend the application so to communicate with the Telegram app. The program has been written in Python and the code base is open, which rely on one hundred and forty seven contributors. It's existence dates back to may 2017 (FANG et al., 2020).

The functioning of automated trading goes through a two-way communication between the local computer and a platform-server of trading - which would be Angles to a

broker.

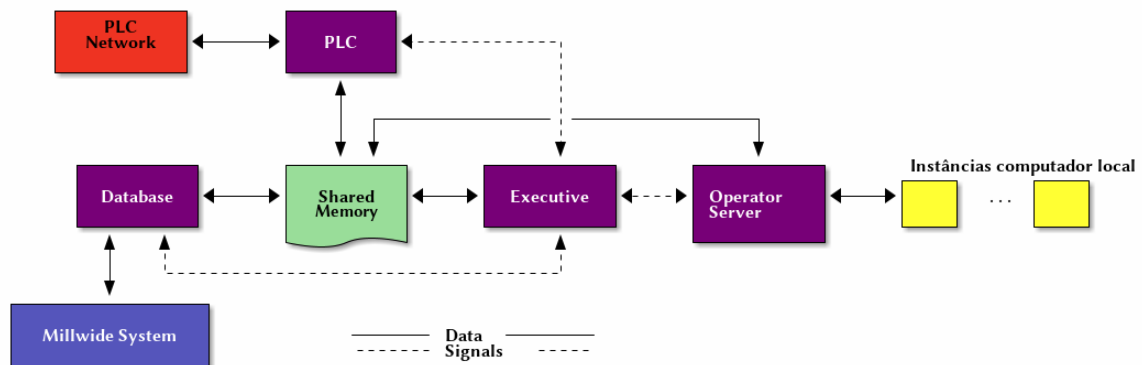
Figure 6 – a simplified schema of client-server communication



Resource: The author.

A great part of the work on writing an automated robot, for any application end, boils down to programming communication protocols with server. Because, the robot should be capable of telling the server which operations should happen, both as means of a response to the client - e.g., a “store data on local machine” request -, as well as internal operations that should take place on the server - e.g., execute buy and sell orders on crypto-current exchanges.

Figure 7 – Schematic representation of client-server communication.



Resource: the author.

Thus, a schematic representation of this complexity could be seen in [Figure 7](#).

By utilizing OSS, all this complexity ([Figure 7](#)) becomes close to the first schema ([Figure 6](#)). Because, all the structural abstraction has already been dealt of, by this application. To make use of this advantage, one just need to read and comprehend the documentation and extend the application to it's use.

2.4.2 OR-Tools

Linear Programming is viewed as a revolutionary development giving us the ability for the first time to state general objectives and to find, by means of the simplex method, optimal policy decisions for a broad class of practical decision problems of great complexity. (DANTZIG, 1983)

The OR-Tools project translates into a series of code libraries developed in an open sourced manner, initially developed by Google. Possible ports exist to C++, Python, Java, C# and .Net.

The name is given due to Operational Researches Tools acronym, ORT.

We list all category of tools which are under OR-Tools ³,

1. A constraint programming solver;
2. A linear programming solver;
3. Wrappers around commercial and other open source solvers, including mixed integer solvers;
4. Bin packing and knapsack algorithms;
5. Algorithms for the Traveling Salesman Problem and Vehicle Routing Problem;
6. Graph algorithms (shortest paths, min cost flow, max flow, linear sum assignment).

The use of this tool can help solve the common problem of Scheduling. Structurally, this class of algorithms relate to solving puzzles as Sudoku - for which there are dynamical constraints upon interaction (SIMONIS, 2005).

In general, these problem fall under a mathematical and computational field called Constrained Optimization (BERTSEKAS, 2014).

Not rarely, the typical problem in Constrained Optimization can fall under a NP-hard problem (a problem for which the Combinatorics of the system have an exponential ratio, not describable by a polynomial). A classical example is the “job shop problem”, for which a seemingly simple and practically ubiquitous task, actually is a NP-hard problem (ZHANG et al., 2019).

Just as in non-deterministic differential equations, or in differential equations which do not have an analytical solution, it’s common to use numerical solutions. When there one find a NP-hard problem, analogously, it’s used genetic-algorithms and other reinforced-learning algorithms (ZHANG et al., 2019).

P vs NP problem is one of the open problems in mathematics which has deep implications to computing. Furthermore, it’s listed among the ten Millennium Prize Problems, of which only one has been solved (COOK, 2006).

The Millenium Prize Problems:

³ <https://github.com/google/or-tools>

- Birch and Swinnerton-Dyer conjecture
- Hodge conjecture
- Navier–Stokes existence and smoothness
- P versus NP problem
- Poincaré conjecture (solved)
- Riemann hypothesis
- Yang–Mills existence and mass gap

2.4.2.1 Application on Economics

The job scheduling problem thus is central to many areas of science. For example, generalizing the problem of Operational Research in economics, we have

The problem of allocating scarce resources among competing ends is central to economic analysis.

(...) The problem of optimal allocation of scarce resources can thus be summarized as the optimization of some objective(s) subject to constraints. (LUPTACIK et al., 2010)

Which means that this field has great application for tools as OR-Tools.

The emphasis is given to the exposition of mathematical optimization as an instrument for qualitative analysis and to a wide range of applications in economics, including efficiency analysis, industrial economics (with focus on regulatory economics), international economics, input–output economics, quantitative economic policy and environmental economics. (LUPTACIK et al., 2010)

2.4.2.2 Application to environmental policies

Policies annually are pitched by Intergovernmental Panel on Climate Change (IPCC) on how business should constrain their business matrix of energy etc, so to meet environmental goals (KEYSSER; LENZEN, 2021).

There is an open field to transform these measures into actionable plans to enterprises. If sufficiently quantitative, such measures could be incorporated in programs is Operational Researches software (LUPTACIK et al., 2010).

There are some papers which gives Optimization Models for air pollution control (ABRAMS, 1975), countries comparative eco-efficiency (DYCKHOFF; ALLEN, 2001) etc. We cite a quotation from “Eco-efficiency analysis of industrial system in China: A data envelopment analysis approach” (ZHANG et al., 2008).

In this connection, strategies for optimizing the use of resources or environment as expressed in more efficient way play a particularly important role. Eco-efficiency, which is an instrument for sustainability analysis,

indicating an empirical relation in economic activities between environmental cost or value and environmental impact, has been proposed as a route to promote such a transformation (MICKWITZ et al., 2006).

And we can see that this is the case-use, at least in quantitative researches in ecology; we look into one result from US eco-efficiency research on a decade of data,

Although an increase in both the mean RR (0.3–0.4) and the mean E-RR (0.38–0.52) scores was recorded from 1995 to 2014, this was concluded to be unsatisfactory, since the majority of the industries' eco-efficiency results were still below 0.5. (EZICI; EĞILMEZ; GEDIK, 2020)

And, the conclusion is that,

The findings of this study suggest that substantial policy changes are required immediately to shift the negative trend in renewable energy use to comply with UN Sustainability Development Goals 7 and 13. (EZICI; EĞILMEZ; GEDIK, 2020)

We also emphasize, that this problem can seem unrelated to other things like grammar analysis on authorship, or ecological policies formulation. But, indeed they are related.

Research using eco-efficient measures upon constrains for making policies has matured. And plans utilizing it as an constrain can be pathways to saving the world from ecological decimation.

Urban-industrial symbiosis (UIS) is an important system innovation via sectors integration, and has been widely recognized as a novel pathway for achieving regional eco-industrial development. Eco-efficiency, as a mature approach and indicator, offers an effective tool to uncover both the status and trends of such a transformation (BIAN et al., 2020).

2.4.2.3 Application to other areas

The areas that such optimization constrains are applied are uncountable. We cite some not so intuitive areas:

- Natural Language analysis and evolution (POTTS et al., 2010; HEINZ, 2018)
- Plans to maintain production of rural areas in Brazil, with no need for further deforestation (COSTA et al., 2018).
- Planning upon corporate social performance (CHEN; DELMAS, 2011; JACOBS; KRAUDE; NARAYANAN, 2016).
- Public transportation (SCHIEWE et al., 2020)
- Traffic Control (MONACHE et al., 2019)

2.5 Demonstration in Academic Applications

2.5.1 DifferentialEquations.jl

The numerical library, DifferentialEquations, has one of the best performances among numerical software ([RACKAUCKAS; NIE, 2017b](#)). The performance is comparable to implementations in FORTRAN and C. Also, this library has many ports⁴, some of the languages in which you can use it are in Python, R and Julia itself.

Among the many different category of problems this library can deal with are ([RACKAUCKAS; NIE, 2019](#); [RACKAUCKAS; NIE, 2017a](#); [RACKAUCKAS; NIE, 2018](#); [SYKORA et al., 2020](#); [RACKAUCKAS et al., 2018](#); [RACKAUCKAS et al., 2019](#); [RACKAUCKAS et al., 2020](#); [GOWDA et al., 2019](#); [MA et al., 2021](#)),

- Discrete equations (function maps, discrete stochastic (Gillespie/Markov) simulations)
- Ordinary differential equations (ODEs)
- Split and Partitioned ODEs (Symplectic integrators, IMEX Methods)
- Stochastic ordinary differential equations (SODEs or SDEs)
- Stochastic differential-algebraic equations (SDAEs)
- Random differential equations (RODEs or RDEs)
- Differential algebraic equations (DAEs)
- Delay differential equations (DDEs)
- Neutral, retarded, and algebraic delay differential equations (NDDEs, RDDEs, and DDAEs)
- Stochastic delay differential equations (SDDEs)
- Experimental support for stochastic neutral, retarded, and algebraic delay differential equations (SNDDEs, SRDDEs, and SDDAEs)
- Mixed discrete and continuous equations (Hybrid Equations, Jump Diffusions)
- (Stochastic) partial differential equations ((S)PDEs) (with both finite difference and finite element methods)

2.5.1.1 Julia portability

The steps to use the library in Julia are,

```
# Using the package Pkg, so to manage packages.
using Pkg
# Add and/or download the DifferentialEquations.jl package in the project
Pkg.add("DifferentialEquations")
# Access all the functions defined in the package
```

⁴ Capability to interoperate besides the language it was written in.

using DifferentialEquations

2.5.1.2 Python portability

In a terminal using the package manager pip,

```
pip install diffeqpy
```

In a terminal, running the Python interpreter,

```
>>> import diffeqpy
>>> diffeqpy.install()
```

An optional step is to add numba, so to optimize the code performance,

```
pip install numba
```

Finally, import the package in a file and use in a program.

```
import diffeqpy
```

2.5.1.3 R portability

To install, use

```
install.packages("diffeqr")
```

In a first call,

```
diffeqr::diffeq_setup()
```

This way, the package will be downloaded, directly from DifferentialEquations.jl. An alternative is to use [the CRAN maintained subset](#).

2.5.2 L^AT_EX

The L^AT_EX has fundamentally differentiate the many tasks in the typography of a document. The language permits a separation of formatting-tasks, and the content-writing. This way, the user can focus exclusively on content, in a fundamental step in document writing. And in another moment, focus solemnly on format and aesthetics.

Therefore, there a quality gain in production. Just as one is rewarded with the total control of the document, because every graphical disposition and behavior is defined by

open code. Furthermore, one can extend already written templates and packages which do usual formatting. The typographical system in \LaTeX has been considered, also, one of the most sophisticated software of it's category, mostly because of this bottom-up paradigm one naturally uses ([HARALAMBOUS, 2007](#)).

The \LaTeX , technically, is the union between the typographical language \TeX , invented by Donald Knuth, for high quality documents ([KNUTH, 1986](#)). And, the powerful macro ecosystem, which extend the \TeX programming, for which we call \LaTeX .

Initially \LaTeX was developed by Leslie Lamport, with his personal use-case formatting programs ([LAMPORT, 1994](#)). Thus, \LaTeX is not just a language, but a conjunction of macros and standard environments, which is actively maintained, modified, and extended by the open community.

2.5.2.1 The canonical ABNT class for scientific production

What is called as canonical are a set of documents following the more general and robust ABNT (Associação Brasileira de Normas Técnicas) norms. These canonical forms exist for scientific articles, technical reports, academic works as thesis, dissertations and research projects, books and presentations ([ABNTEX, 2012](#)).

Os documentos indicados tratam-se de “Modelos Canônicos”, ou seja, de modelos que não são específicos a nenhuma universidade ou instituição, mas que implementam exclusivamente os requisitos das normas da ABNT, Associação Brasileira de Normas Técnicas. ([ARAUJO, 2018](#), Cap. 1)

The norms prescribed by the canonical model are:

- **ABNT NBR 6022:2018:** Informação e documentação - Artigo em publicação periódica científica - Apresentação.
- **ABNT NBR 6023:2002:** Informação e documentação - Referência - Elaboração.
- **ABNT NBR 6024:2012:** Informação e documentação - Numeração progressiva das seções de um documento - Apresentação.
- **ABNT NBR 6027:2012:** Informação e documentação - Sumário - Apresentação.
- **ABNT NBR 6028:2003:** Informação e documentação - Resumo - Apresentação.
- **ABNT NBR 6029:2006:** Informação e documentação - Livros e folhetos - Apresentação.
- **ABNT NBR 6034:2004:** Informação e documentação - Índice - Apresentação.
- **ABNT NBR 10520:2002:** Informação e documentação - Citações.
- **ABNT NBR 10719:2015:** Informação e documentação - Relatórios técnicos e/ou científico - Apresentação.
- **ABNT NBR 14724:2011:** Informação e documentação - Trabalhos acadêmicos - Apresentação.

- **ABNT NBR 15287:2011:** Informação e documentação - Projeto de pesquisa - Apresentação.

2.6 Canonical works on Computer Science

The scientific library of classical mechanics, written in Scheme (a Lisp dialect), has been used to teach master's degree MIT students. Accompanied to the library there is the book, which servers as didactic material for the course (SUSSMAN; WISDOM, 2015).

The library and the book are open sourced.

2.6.1 Structure and interpretation of computer programs (SICP)

The scheme course (SICP), which is teach as the foundations of computer science course on MIT, has accompanied as textbook one of the most world wide recognized books in computer history (ABELSON; SUSSMAN, 1996).

This course has been the pioneer on givin emphasis to the epistemological activity, when programming. The use of Scheme, which has a uniform notation (only one form), was revolutionary. This has shown to work well, instead of presenting the trending language in Industry, in a given moment - which is the standard approach.

The SCIM (section 2.6) course as a predecessor to SICP. The course and the text book still are in use in MIT and a number of other universities around the world. A non-exhaustive list can be found at <<https://mitpress.mit.edu/sites/default/files/sicp/adopt-list.html>>. At least twenty universities adopts, distributed in more than eight countries.

These materials, textbook and the MIT classes, are both free and open sourced.

2.6.2 SCIMUtils - (SCIM) Portability in Clojure

These are dedicated libraries for rewriting the SCIM programs in Clojure, which ports many functionalities mentioned in the scientific library SCIM. The library and the course's programs are not the same thing (SMITH; RITCHIE, 2016). This library, therefore, has been ported to Clojure, which is the most industry-used Lisp modern-dialect. For example, Nubank bought Cognitect which was the first and most successful application written in Clojure (HAMILTON, 2020).

The language has a double quality in it's program. Due to the language supporting multiple definitions of a function under a name (polymorphism), it's possible to very easily switch and integrate numerical and symbolical computations.

Changing inputs one would have one or the other output, by inference of the Clojure compiler. This is particularly interesting when solving, for example, Lagrange equations of motion. One can have both the symbolical results outputted in \TeX , and by a change in inputs, in the same procedure, have numerical data which can be used to simulate the behavior of physical systems.

3 Materials and Methods

The planning of the monograph has followed the steps:

1. Contact and develop strategies with the supervisor;
2. Deliniating the main theme;
3. Given the date constrains, schedule an fitting agenda;
4. Outline the structure of the TCC;
5. Gather of exploratory notes about the topic;
6. Biliographical reseach;
7. Writting each dedicated topic;
8. Reuse past knowledge and use personal practical examples;
9. Continuous revision, for each step of the monograph;
10. Organize the presentation;

3.1 Initial Strategy

The first steps taken were to determine what would be the best match of student-advisor, based on mutual interest on the topic. Then, began the first general schema of the work, and the schedules.

3.2 Theme delineation

Given that the theme OSS on Industry and Academia has indefinetly many branches to discuss, we urged to narrow down the boudaries of the topic to those applications of most family to the student.

In the continuous reviews, we further narrowed the topic by wheighthing if deepening the analyses wouldn't deviate from the main topics.

3.3 Cronological Schedule

We followed the Gantt diagram method to orient ourselves, which is a visual aid to help on following deadlines. At the same time, the diagram helps to situate oneself in relation to what should the team be doing.

3.4 The monograph theme

As this monograph mainly falls under an Exploratory Analysis, the main focus was the theme's literature. We show programs developed by the open community.

On the Academic Application's branch, the emphasis was given to auxiliary software for research. Also, we shown some software that may as well be used for teaching physical concepts.

In the short experience of the student on Industry, we found examples of software he already made use of.

3.5 Gathering various notes

One of the open sourced software of most use to the student is org-mode. In *.org* files, one can take notes, just as in note applications found in any device. A lot of material was gathered from those notes to compose the results.

3.6 Bibliographical Research

The bibliographical review intended to facilitate the comprehension of Software Industry. We gathered some cuts of the history of FOSS. Furthermore, we brought information about some open software which are considered state of the art, or iconic, on their field of application.

Most of the historical content about the FOSS movement and software in general can be found in websites. Particularly, GNU was the main source of FOSS history. In contrast, most sources on software development and comparative functionality was found on scientific articles.

We used Google Scholar and Association for Computing Machinery's (ACM) platform. The student had access due to being a member to the research engine and papers they have. The content of ACM is mainly about software and theoretical computation in general.

3.7 Continuous Review

A fundamental part of the research's organization conciseness was due to the frequent interaction between the student and the advisor.

3.8 Presentation Organization

The use of visual software coupled with org-mode¹ makes an presentation interactive. So, much of the work done in the monograph was reused for presentation.

¹ There are dedicated packages in Emacs to make presentations with org-mode files <https://github.com/howardabrams/demo-it>

4 Results and Discussion

4.1 Advisor expertise and the theme

One of the applications the student supposed could be done with OSS was with Lagrangians. SCIMUtils ([subsection 2.6.2](#)) exist in Clojure - the port of the famous work SCIM. and the advisor had given lectures on the subject of Classical Mechanics on the university courses. At the same time, the advisor itself is an user-developer, using Linux. So, the themes and expertise matched.

4.2 Theme delineation

The GNU and the OSS initiative have indefinitely many subjects that could be tackled. There are virtually FOSS written for application on any human enterprise.

We opted to work on applications most familiar to the student. Also, we the work focused on presenting the most significant aspects of FOSS, in geral, for Academia and Industry.

Thus, the delimitation on showing the framework for an working FOSS OS. The delimitation to the OR-Tools and Freqtrade in Industry. Finally, the use of Clojure, Julia, Python and R as means to present graphical and numerical applications to abstract ideas in mathematics.

4.2.1 Industry Applications

4.2.1.1 OR-Tools

The language used to write this application is C++¹. But, there are ports to Python, Java (thus, Clojure), .Net. All languages have this program licensed in *copy-left* terms.

OR-Tools has been designed to tackle conspicuous problems: one of them, scheduling. Virtually any social-economic enterprise use scheduling in some moment.

We offer an example in the following setting. A hospital needs to create a schedule for four nurses over a three-day period, subject to the following conditions:

1. Each day is divided into three 8-hour shifts.
2. Every day, each shift is assigned to a single nurse, and no nurse works more than one shift.

¹ <https://github.com/google/or-tools>

3. Each nurse is assigned to at least two shifts during the three-day period.
4. There are 10 nurses.
5. There are 7 days of work.

In this problem, the program would translate to², Importing the OR-Tools utilities

```
from ortools.sat.python import cp_model
```

Stating the constants, and standard input/output functions.

```
class NursesPartialSolutionPrinter(cp_model.CpSolverSolutionCallback):
    """Print intermediate solutions."""

    def __init__(self, shifts, num_nurses, num_days, num_shifts, sols):
        cp_model.CpSolverSolutionCallback.__init__(self)
        self._shifts = shifts
        self._num_nurses = num_nurses
        self._num_days = num_days
        self._num_shifts = num_shifts
        self._solutions = set(sols)
        self._solution_count = 0

    def on_solution_callback(self):
        if self._solution_count in self._solutions:
            print('Solution %i' % self._solution_count)
            for d in range(self._num_days):
                print('Day %i' % d)
                for n in range(self._num_nurses):
                    is_working = False
                    for s in range(self._num_shifts):
                        if self.Value(self._shifts[(n, d, s)]):
                            is_working = True
                            print(' Nurse %i works shift %i' % (n, s))
                    if not is_working:
                        print(' Nurse {} does not work'.format(n))
            print()
            self._solution_count += 1

    def solution_count(self):
        return self._solution_count
```

Giving fixed values to the program,

```
def main():
    # Data.
    num_nurses = 10
    num_shifts = 3
```

² This problem is a variation from the one on the project's documentation, https://developers.google.com/optimization/scheduling/employee_scheduling

```

num_days = 7
all_nurses = range(num_nurses)
all_shifts = range(num_shifts)
all_days = range(num_days)

```

Creating the model constraints,

```

# Creates the model.
model = cp_model.CpModel()

# Creates shift variables.
# shifts[(n, d, s)]: nurse 'n' works shift 's' on day 'd'.
shifts = {}
for n in all_nurses:
    for d in all_days:
        for s in all_shifts:
            shifts[(n, d,
                    s)] = model.NewBoolVar('shift_n{id}%is%i' % (n, d, s
                                                                ))

# Each shift is assigned to exactly one nurse in the schedule period.
for d in all_days:
    for s in all_shifts:
        model.Add(sum(shifts[(n, d, s)] for n in all_nurses) == 1)

# Each nurse works at most one shift per day.
for n in all_nurses:
    for d in all_days:
        model.Add(sum(shifts[(n, d, s)] for s in all_shifts) <= 1)

# Try to distribute the shifts evenly, so that each nurse works
# min_shifts_per_nurse shifts. If this is not possible, because the
# total
# number of shifts is not divisible by the number of nurses, some
# nurses will
# be assigned one more shift.
min_shifts_per_nurse = (num_shifts * num_days) // num_nurses
if num_shifts * num_days % num_nurses == 0:
    max_shifts_per_nurse = min_shifts_per_nurse
else:
    max_shifts_per_nurse = min_shifts_per_nurse + 1
for n in all_nurses:
    num_shifts_worked = 0
    for d in all_days:
        for s in all_shifts:
            num_shifts_worked += shifts[(n, d, s)]
    model.Add(min_shifts_per_nurse <= num_shifts_worked)
    model.Add(num_shifts_worked <= max_shifts_per_nurse)

```

Call the `cp_model.to_transform_the_constraints_in_a_solvable_format`,

```
# Creates the solver and solve.
solver = cp_model.CpSolver()
solver.parameters.linearization_level = 0
```

Finally, print a few of the solutions (5) and give some feedback on performance and analytics of the problem.

```
# Display the first five solutions.
a_few_solutions = range(5)
solution_printer = NursesPartialSolutionPrinter(shifts, num_nurses,
                                                num_days, num_shifts,
                                                a_few_solutions)

solver.SearchForAllSolutions(model, solution_printer)

# Statistics.
print()
print('Statistics')
print(' - conflicts      : %i' % solver.NumConflicts())
print(' - branches       : %i' % solver.NumBranches())
print(' - wall time        : %f s' % solver.WallTime())
print(' - solutions found : %i' % solution_printer.solution_count())

if __name__ == '__main__':
    main()
```

The result:

```
Solution 4
Day 0
Nurse 0 does not work
Nurse 1 does not work
Nurse 2 does not work
Nurse 3 does not work
Nurse 4 does not work
Nurse 5 does not work
Nurse 6 does not work
Nurse 7 works shift 0
Nurse 8 works shift 1
Nurse 9 works shift 2
Day 1
Nurse 0 does not work
Nurse 1 does not work
Nurse 2 does not work
Nurse 3 does not work
Nurse 4 does not work
```

```

Nurse 5 does not work
Nurse 6 does not work
Nurse 7 works shift 2
Nurse 8 works shift 1
Nurse 9 works shift 0
(...)
Day 6
Nurse 0 works shift 1
Nurse 1 works shift 0
Nurse 2 works shift 2
Nurse 3 does not work
Nurse 4 does not work
Nurse 5 does not work
Nurse 6 does not work
Nurse 7 does not work
Nurse 8 does not work
Nurse 9 does not work

Statistics
- conflicts      : 3540
- branches      : 67968438
- wall time     : 102.701620 s
- solutions found : 561425

```

When there are too many solutions to a constrain problem, that means there are too much resources to the given constrain. That is, the hospital could operate with less workers³. Let's use 3 nurses, keeping the other constrains.

This gives us, on termination,

```

Statistics
- conflicts      : 386
- branches      : 2683080
- wall time     : 15.481512 s
- solutions found : 279936

```

But, if we then change the constrains to number of shifts being 4 - that is, 6 hour-work.

```

Statistics
- conflicts      : 10
- branches      : 194
- wall time     : 0.002517 s
- solutions found : 0

```

³ We stopped the program before it found all solutions.

Of course, because one of the nurses would need to work more than once a day. This would break one of our constrained-condition. We note, though, that there is no continuity in-between the possibility of no solution and the most optimized one.

4.2.1.2 Freqtrade

Freq-trade automate crypto-trading. The software use is pretty straight forward, once one has developed an strategy, in Python.

The strategy using Boiling Bands (BB) and Relative Strength Index (RSI) can be understood in words as the principle: buy when prices are oversold and sell when it's on an average price.

4.2.2 Academia Applications

We opted to work with applications written in Python, Julia and Clojure(Script). Because, the student already is familiar to these languages. In fact, many of the applications brought in this work has been cuts of real problems the authors faced in Academia. These simulations are thus a deepening of how these solutions emerge.

Julia has been invented to work with scientific computation. The optimized computation of Julia compares to FORTRAN and C++. Thus, we chose to use a totally julia-written package, called *DifferentialEquations.jl*. This package can be used to solve a wide range of differential equations.

Bibliography

ABELSON, H.; SUSSMAN, G. J. *Structure and interpretation of computer programs*. [S.l.]: The MIT Press, 1996. Citado 2 vezes nas páginas 12 and 28.

ABNTEX, E. A classe abntex2: Modelo canônico de trabalhos acadêmicos brasileiros compatível com as normas abnt nbr 14724: 2011, abnt nbr 6024: 2012 e outras.[sl], 2012. <http://code.google.com/p/abntex2/>>. Citado, v. 2, p. 2, 2012. Citado na página 27.

ABRAMS, R. Optimization models for regional air pollution control. In: *Mathematical Analysis of Decision Problems in Ecology*. [S.l.]: Springer, 1975. p. 116–137. Citado na página 23.

AMIN, N.; ROMPF, T. Collapsing towers of interpreters. *Proc. ACM Program. Lang.*, Association for Computing Machinery, New York, NY, USA, v. 2, n. POPL, dez. 2017. Disponível em: <<https://doi.org/10.1145/3158140>>. Citado 2 vezes nas páginas 13 and 14.

ARAUJO, L. C. A classe abntex2. *Documentos técnicos e científicos brasileiros*, 2018. Citado na página 27.

AVELEDA, A. A. et al. Performance comparison of scientific applications on linux and windows hpc server clusters. *Mecânica Computacional*, v. 29, n. 30, p. 2985–2997, 2010. Citado na página 19.

BERTSEKAS, D. P. *Constrained optimization and Lagrange multiplier methods*. [S.l.]: Academic press, 2014. Citado na página 22.

BIAN, Y. et al. A sectoral eco-efficiency analysis on urban-industrial symbiosis. *Sustainability*, Multidisciplinary Digital Publishing Institute, v. 12, n. 9, p. 3650, 2020. Citado na página 24.

CHEN, C.-M.; DELMAS, M. Measuring corporate social performance: An efficiency perspective. *Production and operations management*, Wiley Online Library, v. 20, n. 6, p. 789–804, 2011. Citado na página 24.

COOK, S. The p versus np problem. *The millennium prize problems*, p. 87–104, 2006. Citado na página 22.

COSTA, M. P. et al. A socio-eco-efficiency analysis of integrated and non-integrated crop-livestock-forestry systems in the brazilian cerrado based on lca. *Journal of Cleaner Production*, Elsevier, v. 171, p. 1460–1471, 2018. Citado na página 24.

DANTZIG, G. B. Reminiscences about the origins of linear programming. In: *Mathematical Programming The State of the Art*. [S.l.]: Springer, 1983. p. 78–86. Citado na página 22.

D'ELIA, M.; PACIELLO, V. Performance evaluation of labview on linux ubuntu and window xp operating systems. In: IEEE. *2011 19th Telecommunications Forum (TELFOR) Proceedings of Papers*. [S.l.], 2011. p. 1494–1498. Citado na página 19.

- DYCKHOFF, H.; ALLEN, K. Measuring ecological efficiency with data envelopment analysis (dea). *European Journal of Operational Research*, Elsevier, v. 132, n. 2, p. 312–325, 2001. Citado na página 23.
- EMACS History. 2021. <<https://www.emacswiki.org/emacs/EmacsHistory>>. Citado na página 17.
- EZICI, B.; EĞILMEZ, G.; GEDIK, R. Assessing the eco-efficiency of us manufacturing industries with a focus on renewable vs. non-renewable energy use: An integrated time series mrio and dea approach. *Journal of Cleaner Production*, Elsevier, v. 253, p. 119630, 2020. Citado na página 24.
- FANG, F. et al. Cryptocurrency trading: a comprehensive survey. *arXiv preprint arXiv:2003.11352*, 2020. Citado na página 20.
- FINK, M. *The business and economics of Linux and open source*. [S.l.]: Prentice Hall Professional, 2003. Citado na página 12.
- FITZGERALD, B. The transformation of open source software. *MIS quarterly*, JSTOR, p. 587–598, 2006. Citado 2 vezes nas páginas 19 and 20.
- GALLEGO, M. D. et al. Open source software: The effects of training on acceptance. *Computers in Human Behavior*, Elsevier, v. 49, p. 390–399, 2015. Citado 3 vezes nas páginas 14, 15, and 20.
- GOWDA, S. et al. Sparsity programming: Automated sparsity-aware optimizations in differentiable programming. 2019. Citado na página 25.
- HAMILTON, A. *Brazilian challenger Nubank acquires firm behind Clojure and Datomic*. 2020. <<https://www.fintechfutures.com/2020/07/brazilian-challenger-nubank-acquires-firm-behind-clojure-and-datomic/>>. Citado na página 28.
- HARALAMBOUS, Y. *Fonts & encodings*. [S.l.]: "O'Reilly Media, Inc.", 2007. Citado na página 27.
- HAUGE, Ø.; SØRENSEN, C.-F.; CONRADI, R. Adoption of open source in the software industry. In: SPRINGER. *IFIP International Conference on Open Source Systems*. [S.l.], 2008. p. 211–221. Citado 2 vezes nas páginas 15 and 20.
- HEINZ, J. The computational nature of phonological generalizations. In: *Phonological typology*. [S.l.]: De Gruyter Mouton, 2018. p. 126–195. Citado na página 24.
- HERTEL, G.; NIEDNER, S.; HERRMANN, S. Motivation of software developers in open source projects: an internet-based survey of contributors to the linux kernel. *Research policy*, Elsevier, v. 32, n. 7, p. 1159–1177, 2003. Citado na página 12.
- HIPPEL, E. v.; KROGH, G. v. Open source software and the “private-collective” innovation model: Issues for organization science. *Organization science*, Informs, v. 14, n. 2, p. 209–223, 2003. Citado 2 vezes nas páginas 11 and 12.
- JACOBS, B. W.; KRAUDE, R.; NARAYANAN, S. Operational productivity, corporate social performance, financial performance, and risk in manufacturing firms. *Production and Operations Management*, Wiley Online Library, v. 25, n. 12, p. 2065–2085, 2016. Citado na página 24.

- KEYSSER, L. T.; LENZEN, M. 1.5 c degrowth scenarios suggest the need for new mitigation pathways. *Nature communications*, Nature Publishing Group, v. 12, n. 1, p. 1–16, 2021. Citado na página 23.
- KNUTH, D. E. The art of computer programming, vol 1: Fundamental. *algorithms*, p. 187, 1968. Citado na página 11.
- KNUTH, D. E. *TEX: the Program*. [S.l.]: Addison-Wesley, 1986. Citado na página 27.
- KSHETRI, N. Economics of linux adoption in developing countries. *IEEE software*, IEEE, v. 21, n. 1, p. 74–81, 2004. Citado na página 20.
- LAMPORT, L. *LATEX: a document preparation system: user's guide and reference manual*. [S.l.]: Addison-wesley, 1994. Citado na página 27.
- LI, Y.; TAN, C.-H.; YANG, X. It is all about what we have: A discriminant analysis of organizations' decision to adopt open source software. *Decision support systems*, Elsevier, v. 56, p. 56–62, 2013. Citado 3 vezes nas páginas 14, 15, and 20.
- LINUX, list of distributions. [S.l.]: Wikimedia Foundation, 2021. <https://en.wikipedia.org/wiki/List_of_Linux_distributions>. Citado 2 vezes nas páginas 16 and 17.
- LUPTACIK, M. et al. *Mathematical optimization and economic analysis*. [S.l.]: Springer, 2010. Citado na página 23.
- MA, Y. et al. *ModelingToolkit: A Composable Graph Transformation System For Equation-Based Modeling*. 2021. Citado na página 25.
- MACOS version history. [S.l.]: Wikimedia Foundation, 2021. <https://en.wikipedia.org/wiki/MacOS_version_history>. Citado na página 16.
- MICKWITZ, P. et al. Regional eco-efficiency indicators—a participatory approach. *Journal of Cleaner Production*, Elsevier, v. 14, n. 18, p. 1603–1611, 2006. Citado na página 24.
- MICROSOFT, list of operating systems. [S.l.]: Wikimedia Foundation, 2021. <https://en.wikipedia.org/wiki/List_of_Microsoft_operating_systems>. Citado na página 16.
- MIROWSKI, P. The future (s) of open science. *Social studies of science*, SAGE Publications Sage UK: London, England, v. 48, n. 2, p. 171–203, 2018. Citado na página 11.
- MONACHE, M. L. D. et al. Feedback control algorithms for the dissipation of traffic waves with autonomous vehicles. In: *Computational Intelligence and Optimization Methods for Control Engineering*. [S.l.]: Springer, 2019. p. 275–299. Citado na página 24.
- MOODY, G. *Rebel code: Linux and the open source revolution*. [S.l.]: Hachette UK, 2009. Citado na página 12.
- PETERS, M. A. Open education and the open science economy. *Yearbook of the National Society for the Study of Education*, v. 108, n. 2, p. 203–225, 2009. Citado 2 vezes nas páginas 11 and 12.

POTTS, C. et al. Harmonic grammar with linear programming: from linear systems to linguistic typology. *Phonology*, Cambridge University Press, v. 27, n. 1, p. 77–117, 2010. Citado na página 24.

RACERO, F. J.; BUENO, S.; GALLEGO, M. D. Predicting students' behavioral intention to use open source software: A combined view of the technology acceptance model and self-determination theory. *Applied Sciences*, Multidisciplinary Digital Publishing Institute, v. 10, n. 8, p. 2711, 2020. Citado 3 vezes nas páginas 14, 15, and 20.

RACERO, F. J.; BUENO, S.; GALLEGO, M. D. Can the oss-focused education impact on oss implementations in companies? a motivational answer through a delphi-based consensus study. *Electronics*, Multidisciplinary Digital Publishing Institute, v. 10, n. 3, p. 277, 2021. Citado 2 vezes nas páginas 14 and 20.

RACKAUCKAS, C. et al. Diffeqflux.jl—a julia library for neural differential equations. *arXiv preprint arXiv:1902.02376*, 2019. Citado na página 25.

RACKAUCKAS, C. et al. A comparison of automatic differentiation and continuous sensitivity analysis for derivatives of differential equation solutions. *arXiv preprint arXiv:1812.01892*, 2018. Citado na página 25.

RACKAUCKAS, C. et al. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020. Citado na página 25.

RACKAUCKAS, C.; NIE, Q. Adaptive methods for stochastic differential equations via natural embeddings and rejection sampling with memory. *Discrete and continuous dynamical systems. Series B*, NIH Public Access, v. 22, n. 7, p. 2731, 2017. Citado na página 25.

RACKAUCKAS, C.; NIE, Q. Differentialequations.jl—a performant and feature-rich ecosystem for solving differential equations in julia. *Journal of Open Research Software*, Ubiquity Press, v. 5, n. 1, 2017. Citado na página 25.

RACKAUCKAS, C.; NIE, Q. Stability-Optimized High Order Methods and Stiffness Detection for Pathwise Stiff Stochastic Differential Equations. *arXiv:1804.04344 [math]*, 2018. Disponível em: <<http://arxiv.org/abs/1804.04344>>. Citado na página 25.

RACKAUCKAS, C.; NIE, Q. Confederated modular differential equation apis for accelerated algorithm development and benchmarking. *Advances in Engineering Software*, Elsevier, v. 132, p. 1–6, 2019. Citado na página 25.

RISTOV, S.; GUSEV, M. Performance vs cost for windows and linux platforms in windows azure cloud. In: *2013 IEEE 2nd International Conference on Cloud Networking (CloudNet)*. [S.l.: s.n.], 2013. p. 214–218. Citado na página 19.

SCHIEWE, P. et al. *Integrated optimization in public transport planning*. [S.l.]: Springer, 2020. Citado na página 24.

SCHMIDT, C. Agile software development. In: *Agile Software Development Teams*. [S.l.]: Springer, 2016. p. 7–35. Citado na página 20.

- SCHRAPE, J.-F. Open-source projects as incubators of innovation: From niche phenomenon to integral part of the industry. *Convergence*, SAGE Publications Sage UK: London, England, v. 25, n. 3, p. 409–427, 2019. Citado 2 vezes nas páginas 15 and 20.
- S.DEVAN, S. Windows 8 v/s linux ubuntu 12.10 - comparison of the network performance. *International Journal of Research in Engineering and Technology*, v. 02, p. 577–580, 2013. Citado na página 19.
- SIMONIS, H. Sudoku as a constraint problem. In: CITESEER. *CP Workshop on modeling and reformulating Constraint Satisfaction Problems*. [S.l.], 2005. v. 12, p. 13–27. Citado na página 22.
- SMITH, C.; RITCHIE, S. *SICMUtils: Functional Computer Algebra in Clojure*. 2016. <<http://github.com/sicmutils/sicmutils>>. Citado na página 28.
- SPINELLIS, D.; GIANNIKAS, V. Organizational adoption of open source software. *Journal of Systems and Software*, Elsevier, v. 85, n. 3, p. 666–682, 2012. Citado 2 vezes nas páginas 14 and 20.
- STALLMAN, R. *Linux and GNU - GNU Project - Free Software Foundation*. 1997. <<https://www.gnu.org/gnu/linux-and-gnu.html>>. Citado na página 17.
- STALLMAN, R. My lisp experiences and the development of gnu emacs (transcript of richard stallman's speech, 28 oct 2002, at the international lisp conference). *URL (consulted December 2003): http://www.gnu.org/gnu/rms-lisp.html*, 2002. Citado na página 17.
- SULAIMAN, N. S.; RAFFI, A. S. H. A. Comparison of operating system performance between windows 10 and linux mint. *International Journal of Synergy in Engineering and Technology*, v. 2, n. 1, p. 92–102, 2021. Citado 2 vezes nas páginas 15 and 19.
- SUSSMAN, G. J.; WISDOM, J. *Structure and interpretation of classical mechanics*. [S.l.]: The MIT Press, 2015. Citado na página 28.
- SYKORA, H. T. et al. Stochasticdelaydiffeq.jl-an integrator interface for stochastic delay differential equations in julia. 2020. Citado na página 25.
- TANENBAUM, A. S.; BOS, H. *Modern operating systems*. [S.l.]: Pearson, 2015. Citado na página 14.
- THUBAASINI, P.; RUSNIDA, R.; ROHANI, S. Efficient comparison between windows and linux platform applicable in a virtual architectural walkthrough application. In: *Innovations in Computing Sciences and Software Engineering*. [S.l.]: Springer, 2010. p. 337–342. Citado na página 19.
- TORVALDS, L. Linux: a portable operating system. *Master's thesis, University of Helsinki*, 1997. Citado na página 17.
- TU, Q. et al. Evolution in open source software: A case study. In: IEEE. *Proceedings 2000 International Conference on Software Maintenance*. [S.l.], 2000. p. 131–142. Citado na página 12.

WEST, J.; DEDRICK, J. Open source standardization: the rise of linux in the network era. *Knowledge, Technology & Policy*, Springer, v. 14, n. 2, p. 88–112, 2001. Citado na página [12](#).

ZHANG, B. et al. Eco-efficiency analysis of industrial system in china: A data envelopment analysis approach. *Ecological economics*, Elsevier, v. 68, n. 1-2, p. 306–316, 2008. Citado na página [23](#).

ZHANG, J. et al. Review of job shop scheduling research and its new perspectives under industry 4.0. *Journal of Intelligent Manufacturing*, Springer, v. 30, n. 4, p. 1809–1830, 2019. Citado na página [22](#).