A Canonical Experiment on System Complexity Metric and Its Impact on Engineering Management

by

Ricardo Bortot Hopker

BS in Mechanical Engineering, University of Texas Pan-American (2014)

Submitted to the System Design and Management Program
in partial fulfillment of the requirements for the degree of

Master of Science in Engineering and Management

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author .....
System Design and Management Program
May 10, 2022

Certified by .....
Olivier L. de Weck
Apollo Program Professor
Professor of Astronautics and Engineering Systems
Thesis Supervisor

Accepted by .....
Joan Rubin
Executive Director, System Design and Management Program

A Canonical Experiment on System Complexity Metric and Its Impact on Engineering Management

by

Ricardo Bortot Hopker

Submitted to the System Design and Management Program
on May 10, 2022, in partial fulfillment of the
requirements for the degree of
Master of Science in Engineering and Management

Abstract

Systems are constantly increasing complexity. Being able to quantify the system complexity and how it relates to human effort and cognition can bring numerous benefits for product development and project management. In this thesis, 25 people were part of an experiment using the travel salesperson problem, they completed 13 problems each with varying complexity. The results were summarized and through a series of statistical analysis it was found that the human effort scales super-linear with complexity in the form $e = AC^{1.47} + d$, where A and d are constants. Additionally, based on the results in this study and previous, it is proposed an objective function for optimization of system architecture decomposition which uses the heuristics learned to reduce the human effort to understand the system.

Thesis Supervisor: Olivier L. de Weck
Title: Apollo Program Professor
Professor of Astronautics and Engineering Systems

Acknowledgments

This thesis and degree would not be possible without the support and help of many people. I am lucky to be surrounded by an amazing people. I would like to thank them all. Firstly, I would like to thank Professor Oli de Weck, who has guided me through the ups and downs of this thesis, gave me energy and was a source of inspiration to keep going.

I also want to thank Dr. Bryan Moser for believing in me, for always being patient, and for encouraging me throughout the year. Also, I would like to thank the SDM program for providing the cameras needed to collect the data.

I want to thank the whole 2021 SDM TA team for the support and having each other's back. For the friendship that we built during the year.

Additionally, I want to thank Ignacio Vazquez Rodarte for the countless interactions and brainstorm sessions we had on Chapter 4, his insights were invaluable in the creation and development of this chapter.

Last but not least, I want to thank my mom Susy, my brother Henrique, his wife Carol, my girlfriend Isabele, and her parents Luiz and Cibele. I would not be able to accomplish anything without their support, love and patience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Throughout the years, due to the increase in safety, environmental, and customer requirements the systems are getting more and more complex, leading to greater development costs and longer development times even with the aid of better and more efficient technology, techniques, and methodologies. A study from 2008[1], shows that approximately half of the annual price escalation rate from the development of the aircraft F-15A in 1975 to F-22A in 2005 was attributed to the complexity of the system as shown in Figure 1-1. In this case, complexity was loosely defined as "performance characteristics and airframe material", driven by customer requirements.

In another study[4], it was analyzed the development costs in relationship to mission complexity, here complexity is measured as an index that uses a broad set of parameters based on performance, mass, power and technology choices. Results shown in Figure 1-2 depict a clear upward trend, as the Complexity Index increases, development costs increase exponentially. This is an important insight to the value of understanding and managing complexity.

In software systems lines of code are often used to measure complexity. Also, from the space industry, Figure 1-3 shows the increase from less than 100 to over one mullion lines of codes from space missions from the 1960's to the 2010's. This increase is attributed to the increased performance requirements, including increasing autonomy to deal with issues with latency, and improving efficiency and resiliency by having self-evaluating diagnosing and correction tools.[22]

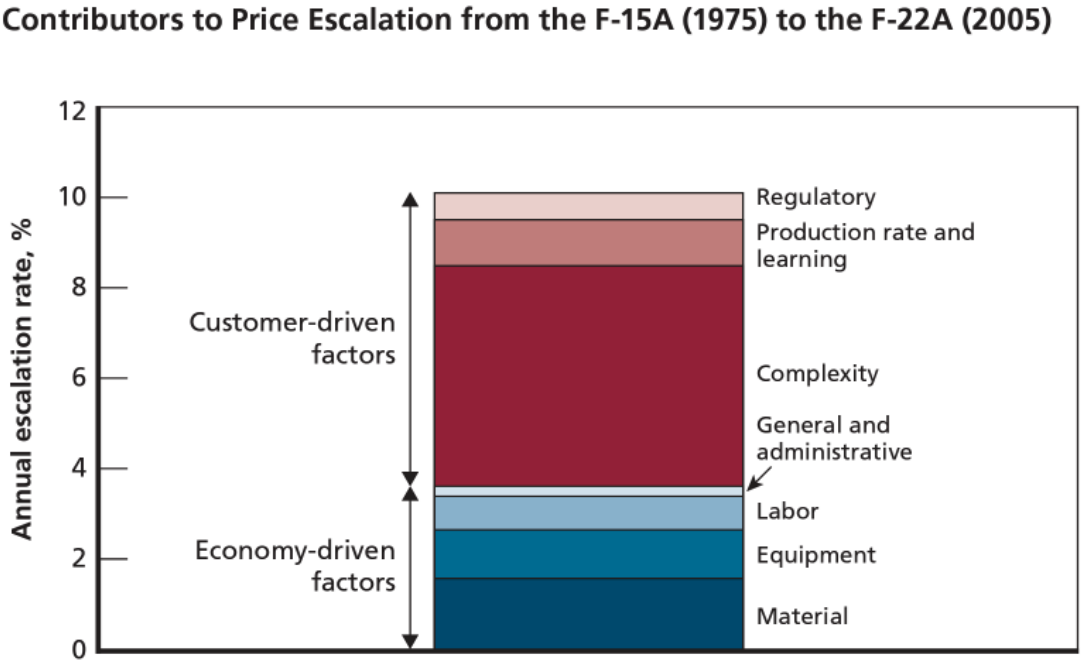


Figure 1-1: Contributors to the price escalation from the F-15A to the F-22A[1]

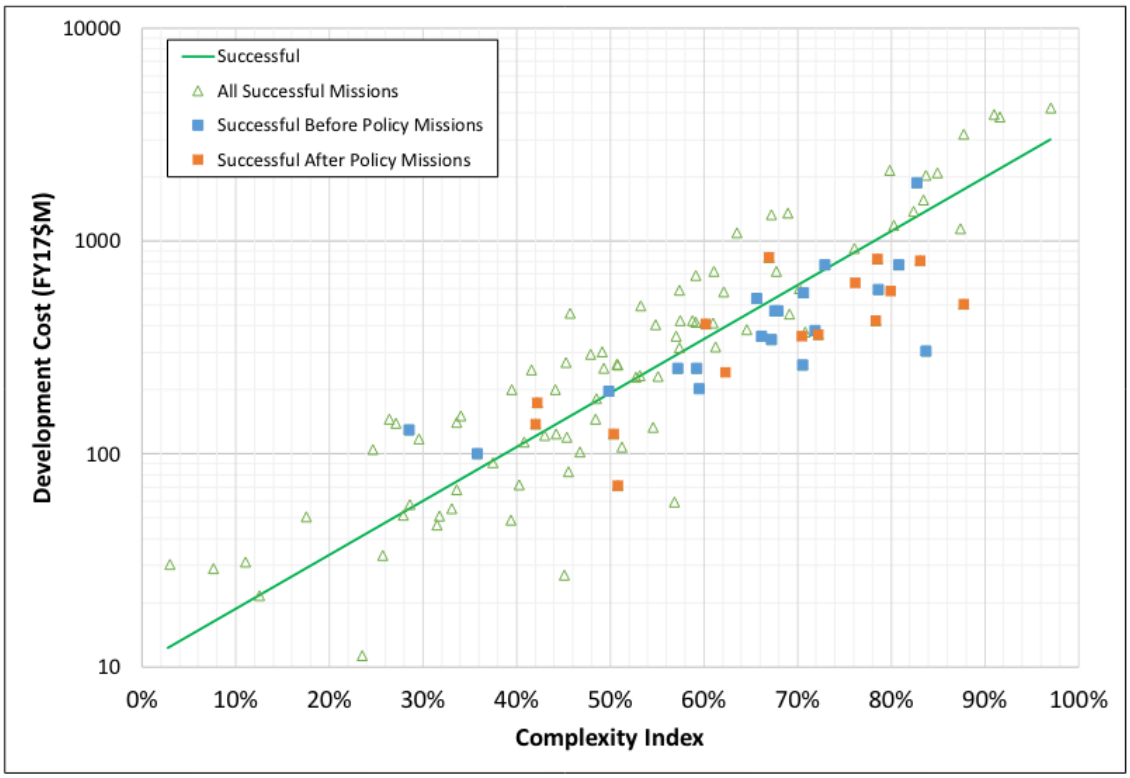


Figure 1-2: Development cost vs Complexity index [4]

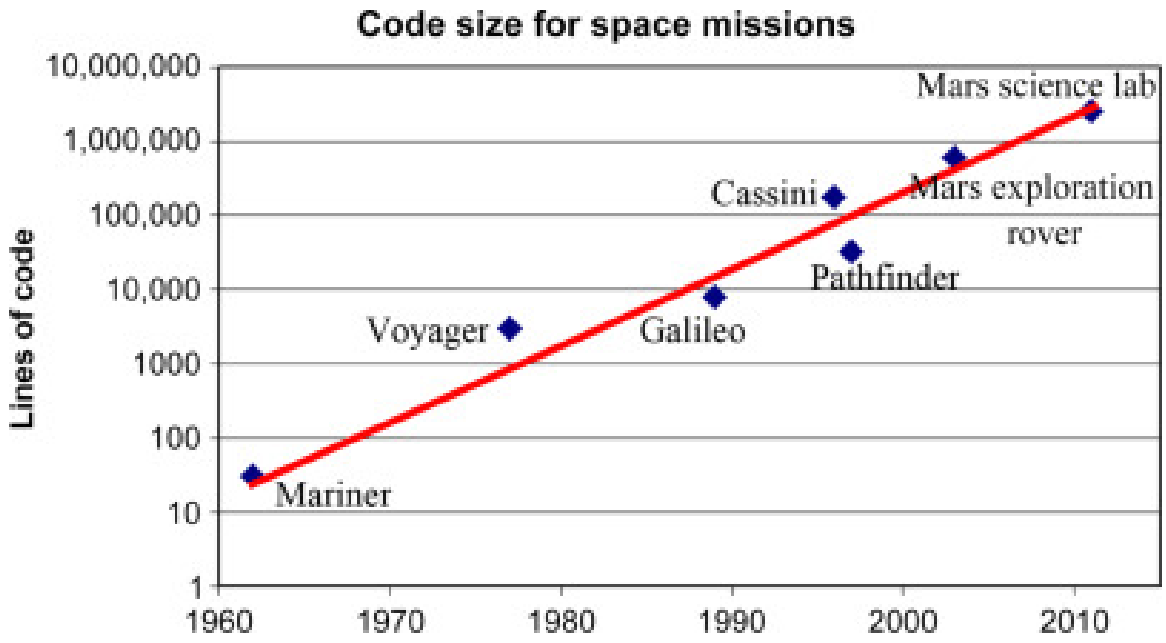


Figure 1-3: Lines of code through time in space mission [22]

Measuring complexity and being able to compare system complexities is a challenge. Several metrics have been introduced with no particular metric being unanimously used across multiple disciplines. What the studies have in largely in common is that system complexity is growing quickly. Each metric has its strengths and weaknesses, and therefore are used in different contexts, usually within the same discipline. Having a single metric that is universal would allow system architects, system engineers and project managers to further understand and optimize their processes, forecasts and better match them with reality.

1.1 Literature Review

There are many types of metrics of system complexity, in this section several are summarized, with their strengths and weaknesses, typical uses and pitfalls. This will give a large overview of the helm of possibilities followed by the reason why structural complexity is used throughout this thesis. The types described here are size, that can be represented in lines of code, number of functions, number of function calls, number of interfaces, or even weight, cyclomatic complexity, Big "O" notation,

coBRA, molecular complexity, and finally structural complexity.

1.1.1 Measuring complexity by size

This is a low-level, often easy to measure and compute. It can be used as a first indicator for software systems, by measuring the lines of code [12], or by calculating the number of functions, and how many times they are called. Additionally, for physical systems it can be used by calculating the number of components, or by weight. To illustrate one of the pitfalls of this method, Figure 1-4 shows the cumulative weight change of vehicles in the US from 1975 to 2010, and there are almost 200 kg added to vehicles attributed to the addition of new features. However, the total average weight of the vehicles remained much more constant since the 1980's due to the improvements in technology and architecture, such lighter and strong materials and a front wheel drive architecture and the unibody construction, depicted in Figure 1-5.[10] As in this example, when using a measure of size, often it is needed more context to understand and to take conclusions.

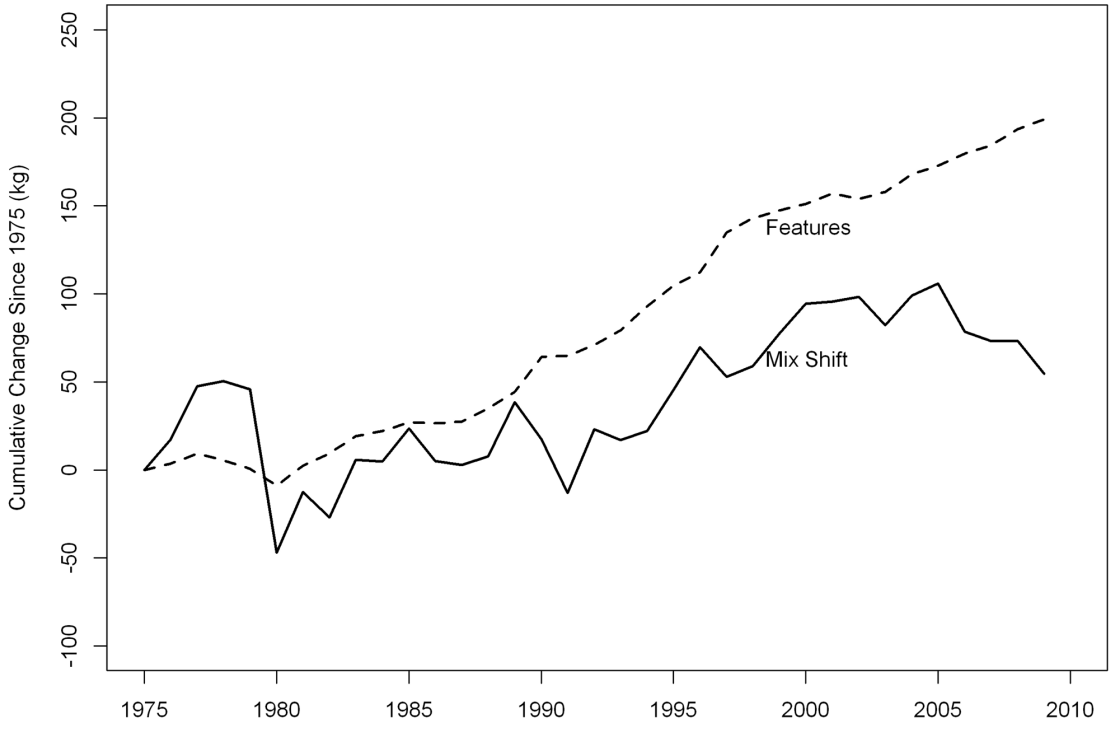


Figure 1-4: Cumulative weight effects of feature changes and mix shifting. [10]

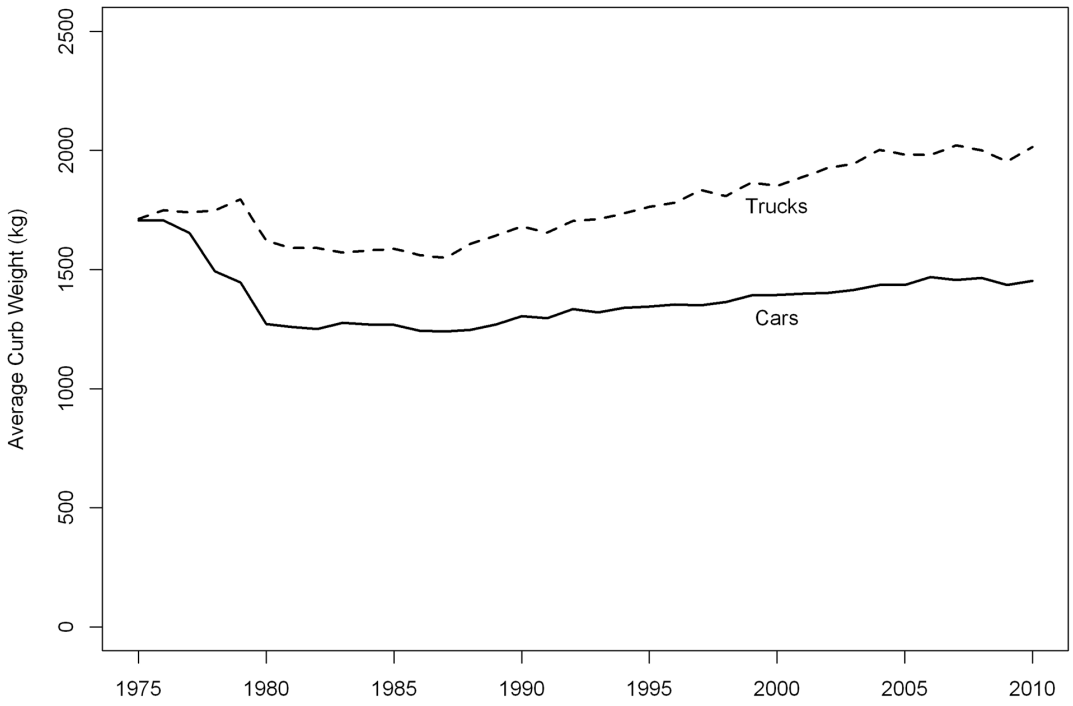


Figure 1-5: Average weights of new U.S. vehicles since 1975 [10]

1.1.2 Cyclomatic Complexity

Specifically created to be applied in software, the cyclomatic complexity measure was introduced by Thomas J McCabe in 1976. It is based on a network, based on graph theory. This approach seeks to measure and control the number of paths through a coding script. The more possible *paths* taken by the script, larger the complexity. More specifically the "cyclomatic number $V(G)$ of a graph G with. n vertices, e edges and p connected components is":[11]

$$V(G) = e - n + p. \tag{1.1}$$

Although this metric is used in software development, it is hard to generalize to other domains or systems. Additionally, it does not account for the interface, or code complexity, focusing solely a simplified measure of the structure of the code. Also, it is a metric used mostly for individual scripts and not for the whole software

development.

1.1.3 Big "O" Complexity

The Big "O" notation is commonly used for computational complexity. It refers to floating point operation, those being: addition, subtraction, division or multiplication. It is extremely useful to evaluate algorithms, as the input size n increases.[13] Although this is extremely valuable for many applications, systems often do not need to solved mathematically but understood, to measure a system complexity by the number of floating operations is counter-intuitive and often impossible.

1.1.4 CoBRA

Complexity Based Risk Assessment (CoBRA), as per the authors knowledge, was first intruducted in 1997 by Sarsfield to be used to determine the complexity of spacecrafts. CoBRA takes into consideration, the design life or required operating life, the destination target, the spacecraft density, the instrument mass fraction, the pointing accuracy of the bus, solar array efficiency, power system efficiency, downlink data rate, central processing power, mass memory, and software code lines.[17] This metric have been used exclusively for spacecraft since it was first introduced, as it is very specific to this domain it cannot be related to any system in any discipline.

1.1.5 Molecular Complexity

In chemistry and drug development complexity is an important characteristic of organic molecules. However, how to calculate it have been a topic in scientific literature for decades without a clear consensus. Bonchev and Trinajstic introduced a method in 1977[5] to measure one important aspect of the molecular complexity, the structural and topology of the connections. However, in these systems self-similarity is also important.[23] Some aspects of these measurements are shared with systems, however how to calculate them is often too laborious or too specific to molecules and cannot be transferred directly.

1.1.6 Structural Complexity

Structural complexity will be covered in more detail in the next chapter. It was developed in 2012 by Sinha and De Weck to measure specifically the complexity of systems. This metric as opposed to those aforementioned in this study and many others, was developed with the intent to address component, local interface and global topological complexity.[19] Throughout this document, unless specifically mentioned, the word complexity will be used to describe the system structural complexity.

1.2 Cognitive performance in the TSP

1.2.1 Travel Salesperson Problem

For this study it was used the Travel Salesperson Problem (TSP). The TSP was formulated in 1930 by Karl Menger and and since then has became one of the most famous and classical problems in optimization. The TSP is a Non-Deterministic Polynomial-time hard (NP-Hard), which means it has arithmetic complexity (Big "O") of $O(\frac{(n-1)!}{2})$.[2] The problem formulation is as follows: a sales person is based on a city and has to travel multiple cities to perform sales and comeback at the starting city at the minimum possible cost, which is often, and in this study, measured by distance. The salesperson should visit each city once and only once. The TSP is widely used in practice. It's applications range from circuit-board drilling, computer wiring, X-Ray Crystallography, order-picking in warehouses, vehicle routing, mask plotting in printed circuit boards (PCBs) [7]. The TSP in this study follows the traditional symmetric problem, where going from point A to point B have the same cost as going from point B to point A.

1.2.2 Previous experiment on Travel Salesperson Problem

In 1996, MacGregor and Ormerod [9] run two experiments with 58 and 29 test subjects, and 6 and 7 problems, and 10 and 20 total points respectively. One hypothesis they had was that the complexity of the TSP is a function of inner points rather

than total points, which the results corroborated. In both experiments, the variability in the tests increased with the number of inner points and the overall quality of the answers declined. Also, the test subjects were consistently producing very good results (within 5-10% of the optimal solution) in relatively short time (2 minutes per problem).

1.3 Problem Statement

The goal of this thesis is to explore the relationship between system complexity, effort, and quality of the solutions answered in an experiment by human test subjects. Furthermore, it will be analyzed behaviors, techniques and strategies employed by the subjects to solve the problems. The effort will be measured as the time to complete a problem, quality will be measured as a percentage above the best possible solution. The problems are a set of different travel salesperson problems with an array of different structural complexities. The TSP was chosen because of it's many possible solutions and how humans cope with these options, for the simplicity and easiness to understand the problem question, because it is relatively easy and low cost to the test subjects and run the experiment.

Chapter 2

Complexity metric

The word complexity is used a lot in many different disciplines and contexts. However it is a very hard concept to measure it and without an universal metric, thus almost impossible to compare the complexity of systems. Sinha and de-Weck[19] tackled this with a system product architecture structural approach. The authors understand that a system complexity arises from the combination of three aspects, these are: component complexity, interface complexity and the topology complexity.

2.1 Definition

In this thesis, it was adopted the system structural complexity metric created by Sinha and de-Weck[19]. Unless specified, the word complexity will be used as defined in their work. The equation that defines the metric is shown in equation 2.1.

$$Complexity, C = C_1 + C_2 * C_3 \tag{2.1}$$

Where C_1 is the component complexity, C_2 is the interface complexity, and C_3 is the topology complexity. For convenience, all the information necessary to calculate the system complexity is given through a n^2 matrix, called the complexity design structure matrix, DSM_c, shown in equation 2.2.

$$DSM_c = \begin{bmatrix} \alpha_1 & \dots & \beta_{n1} \\ \vdots & \ddots & \vdots \\ \beta_{1n} & \dots & \alpha_n \end{bmatrix} \tag{2.2}$$

Where:

$$C_1 = \sum_{i=1}^{n} \alpha_i \tag{2.3}$$

α_i is the complexity of component i.

$$C_2 = \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij}, where\ i \neq j \tag{2.4}$$

β_{ij} is the interface complexity between i and j components. It is worth mentioning that β_{ij} is not necessarily equal to β_{ji}, as not all interfaces are bi-directional or symmetric. Also that, α_i and β_{ij} are greater or equal to 0. Finally we have,

$$C_3 = \frac{1}{n} \sum S[SVD(A(DSM_c - trace(DSM_c)))] \tag{2.5}$$

In words, C_1 is the sum of the diagonal terms of the DSM_c, C_2 is the sum of all the values in the DSM_c minus the diagonal values (C_1). Finally, C_3 is calculated by setting the diagonal to zero, and taking the binary true for any value above zero, and false to values equal to zero, which called the adjency matrix of the DSM_c. The singular value decomposition SVD is then taken, and the sum of the scaling (S) of the SVD is normalized by the matrix size, n.

2.2 Calculation Example

For example, in another study from Sinha and de-Weck[20], it was used the sample controlled pump system shown in Figure 2-1. If the controller, pump, valve, filter and motor have a component complexity (α_i) of 5, 2, 1, 1 and 3 respectively, and the physical, material, energy and infomation flow have $\beta = 0.5, 1, 1, 1$ respectively, the system has the DSM_c as shown in Table 2.1.

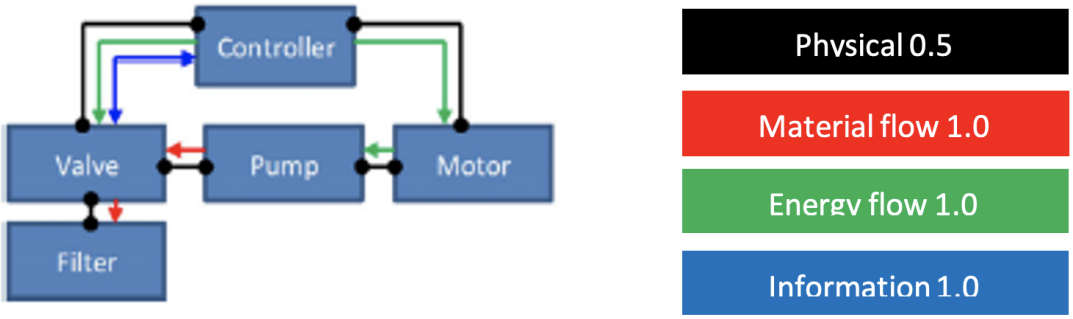


Figure 2-1: Sample problem. Controlled Pump System.

Table 2.1: DSM_C for problem in Figure 2-1

		C	P	V	F	M
Controller	C	5	0	1.5	0	0.5
Pump	P	0	2	0.5	0	1.5
Valve	V	2.5	1.5	1	0.5	0
Filter	F	0	0	1.5	1	0
Motor	M	1.5	0.5	0	0	3

Simplifying the DSM_C to calculate C_1 of the system, as shown in Table 2.2. And summing all the values, we have a $C_1 = 5 + 2 + 1 + 1 + 3 = 12$.

Table 2.2: Matrix for equation 2.3 for C_1 calculation, for problem in Figure 2-1

		C	P	V	F	M
Controller	C	5	0	0	0	0
Pump	P	0	2	0	0	0
Valve	V	0	0	1	0	0
Filter	F	0	0	0	1	0
Motor	M	0	0	0	0	3

Similarly, adapting the DSM_C to calculate C_2 of the system, as shown in Table

2.3. Summing all the values, we have a $C_2 = 12$. Note that C_1, and C_2 are accidentally the same and this is not true for every system.

Table 2.3: Matrix for equation 2.4 for C_2 calculation, for problem in Figure 2-1

		C	P	V	F	M
Controller	C	0	0	1.5	0	0.5
Pump	P	0	0	0.5	0	1.5
Valve	V	2.5	1.5	0	0.5	0
Filter	F	0	0	1.5	0	0
Motor	M	1.5	0.5	0	0	0

Finally, transforming the DSM_C to calculate C_3 of the system, as shown in Table 2.4. Summing the values of the scaling values of the SVD of the Table 2.5 and normalizing it, we get $C_3 = 1.12$. Now, it is possible to use Equation 2.1 to calculate the total system Complexity.

Table 2.4: Matrix for equation 2.5 for C_3 calculation, for problem in Figure 2-1

		C	P	V	F	M
Controller	C	0	0	1	0	1
Pump	P	0	0	1	0	1
Valve	V	1	1	0	1	0
Filter	F	0	0	1	0	0
Motor	M	1	1	0	0	0

$$C = 12 + 12 * 1.12 = 25.44 \tag{2.6}$$

Chapter 3

Canonical experiment on complexity

3.1 Complexity and Human Effort Relationship

As discussed in the Chapter 1 the main purpose of this study it to further analyze the relationship between human effort and system complexity as well as to build on knowledge from Sinha and De-Weck[20]. Where they analyzed the time it took subjects to correctly build physical models of chemical molecules with different complexities. For their experiment, only correctly built molecules were accepted, if there was a mistake, then re-work was necessary until it was correctly built. Their results suggests a super linear relationship between human effort and complexity in the form $e = AC^{1.47}$. Since this is only one experiment in a very specific discipline, using specific cognitive and dexterity skills, it is necessary to expand the study and the experiment to multiple domain such as natural social problems, path optimization, organization structure, and to other disciplines, such as mechanical and electronic to validate the versatility and cross-discipline of the structural complexity metric.

3.2 Experiment methodology

The experiment had in total 25 test volunteers, that were subjected to 13 TSP problems each. Although more participants is always better for statistical analysis, 25 participants is enough to provide meaningful results, assuming a normally distributed

data set, the impact of additional participants is smaller and smaller, in the confidence and standard deviation of the model is smaller in the form of $\frac{1}{\sqrt{n-1}}$. Also, the average values should not change dramatically for every new data point and it will add an impact of $\frac{1}{n}$. The problems were sampled from 30 problems varying in pairs for 15 different levels complexity, each test had at most one problem from each level. In total, it was created 210 unique problem combinations (2 times choose 13 from 15 levels – $2 * {}_{15}C_{13}$). All the test subjects had at least bachelor's degree, predominantly in STEM fields, with a mix of 84% male to 16% female, and 84% were students in the System Design and Management masters program at MIT. Each test subject received a brief explanation of what was expected from them and how to complete the problems. A sample problem was demonstrated, with an optimal solution, a sub-optimal and valid solution, and an invalid solution, where not every city was visited. For each problem the volunteer had 2 plots available of size 2.9 x 4.5 inches, one at the top used exclusively for draft, which they could use either a pen, or a black sharpie and one at the bottom, reserved for the final answer using a red sharpie. Figure 3-1 exemplifies how each problem for a test looked like. There was no time limit to complete the test, nor individual problems. Their objective was to find the optimal path or to get as close as possible to it. The total time to complete each problem was recorded along with the total path taken in each problem. Upon prior agreement, the experimental subjects were recorded using two cameras, a front view and a top view which were used to measure the time of each problem and to observe any techniques, strategies and methods used by the experimental subjects.

3.2.1 Creating problems and summary

To create the 30 problems, firstly it was defined how many inner and outer points each problem would have. Outer points are defined as the points that compose the convex hull of all the points in the TSP and the inner points are those not in the convex hull. Based on the study by, MacGregor and Ormerod [9] the quantity of inner points is the main driver for complexity of the TSP, while outer points add relatively less complexity. For the base scenario it was assumed that the component complexity, α_i,

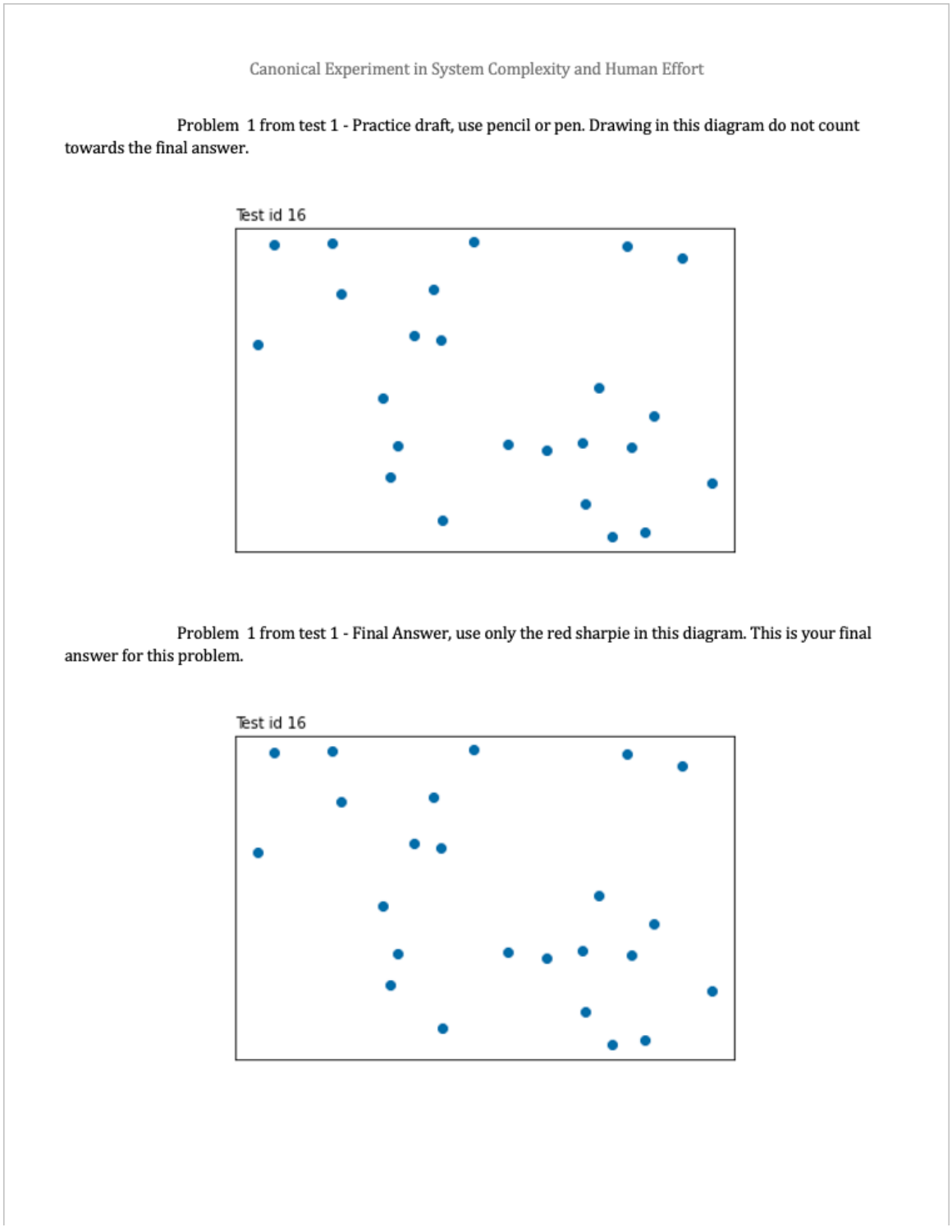


Figure 3-1: Example of experiment problem

Table 3.1: Base complexity for points and interfaces

	Point	Complexity type	Complexity
Component	Inner point	α_i	2
	Outer point	α_i	1
Interface	Inner-Inner	β_{ij}	3
	Inner-Outer	β_{ij}	2
	Outer-Outer	β_{ij}	1

for inner points was 2, while for outer points it was 1. For the interface complexity, β_{ij}, there are 3 possible connections, between inner and inner points, between inner and outer points, and between outer and outer points, these had complexity of 3,2 and 1 respectively. Or more intuitively, for each inner point in the interface β_{ij} added 1 to interface complexity that started at a base of 1. A summary is shown in Table 3.1. With this information is simple to calculate C_1 and C_2 for each problem as shown in equations 3.1 and 3.2. Where, N_{inner} and N_{outer} are the number of inner and outer points respectively.

$$C_1 = 2N_{inner} + N_{outer} \tag{3.1}$$

$$C_2 = 3N_{inner} + N_{outer} \tag{3.2}$$

Every TSP problem has the same connectivity structure or adjacency matrix, needed to calculate topology complexity, C_3, which can be computed following the pseudo-code shown in algorithm 1 and shown in Equation 3.3. This means that the complexity of the TSP is independent of the path, and is only a function of how many inner and outer points it has.

$$A_{TSP} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 & 1 \\ 1 & 0 & 1 & \dots & 0 & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \dots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & 0 & \dots & 1 & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 1 & 0 \end{bmatrix} \tag{3.3}$$

Algorithm 1 Code to calculate C_3 in TSP

Require: Total number of points (n)
  m = Initialize n^2 matrix with zeros
  for integers i in (1 to n) **do**
    $m[i, mod(i+1, n)] = 1$
    $m[mod(i+1, n), 1] = 1$
  end for
  $C_3 = \Sigma(SVD(m))/n$

As previously mentioned, 15 distinct levels with 2 problems in each were created. They were create to have 2 problems that had different number of inner and outer points but had similar complexities. Table 3.2 has all the problems with their level, problem ID, number of inner and outer points, along with the C_1, C_2, C_3 and total complexity.

The problem creation was based the data in Table 3.2, specifically the number of inner and outer points. The basic principle of the creation of the problem was to have randomized X and Y values and checking how many were part of the convex hull and if that matched the problem outer points number. Additionally, it was set a minimum distance of 5% of the total range between points to ensure meaningfulness of each point. A more detailed pseudo-code is shown in Algorithm 2.

3.2.2 Ensuring correct path distances

The experiment was designed in Python in X, Y coordinates using algorithm 2, then plotted using matplotlib, then copied to a word document and finally printed. There

Table 3.2: Problem level, ID, number of inner and outer points, and Complexity.

Level	Problem ID	Inner points	Outer points	C_1	C_2	C_3	C
1	id 0	1	5	7	8	1.33	18
	id 1	2	3	7	9	1.29	19
2	id 2	2	4	8	10	1.33	21
	id 3	2	5	9	11	1.28	23
3	id 4	4	3	11	15	1.28	30
	id 5	4	4	12	16	1.21	31
4	id 6	4	6	14	18	1.29	37
	id 7	4	7	15	19	1.28	39
5	id 8	7	5	19	26	1.24	51
	id 9	6	8	20	26	1.28	53
6	id 10	6	12	24	30	1.28	62
	id 11	7	10	24	31	1.28	64
7	id 12	10	10	30	40	1.26	81
	id 13	9	13	31	40	1.28	82
8	id 14	15	4	34	49	1.27	96
	id 15	12	13	37	49	1.27	99
9	id 16	13	11	37	50	1.27	100
	id 17	15	10	40	55	1.27	110
10	id 18	15	11	41	56	1.28	112
	id 19	16	10	42	58	1.28	116
11	id 20	17	10	44	61	1.27	122
	id 21	17	11	45	62	1.27	124
12	id 22	18	11	47	65	1.27	130
	id 23	18	12	48	66	1.28	132
13	id 24	19	12	50	69	1.27	138
	id 25	19	13	51	70	1.27	140
14	id 26	20	11	51	71	1.27	141
	id 27	20	12	52	72	1.27	143
15	id 28	22	12	56	78	1.28	155
	id 30	28	8	56	80	1.27	158

Algorithm 2 Creating a TSP problem

Require: Number of points (n), Number of outer points (out points), minimum distance between points, (min dist)
  XY = Initialize (n x 2) matrix with random values between 0 and 1
  hull points = how many points are part of the hull
  while hull points not equal to out points **do**
    Z = abs(hull points - out points)
    Zp = Generate Z points
    substitute Z random points in XY with points Zp
    Calculate hull points
  end while
  $d = n^2$ matrix with distance between points
  while d smaller than minimum distance between points **do**
    Nd = Calculate number of points smaller than min dist
    Ndp = Generate Nd random points
    cXY = copy of XY
    substitute Ndp in cXY in the points that were smaller than min dist
    if hull points in cXY equals out points **then**
      XY = cXY
      Will exit loop
    else
      Continue loop
    end if
  end while
  return XY

are many conversions happening in this process and it is important to ensure that the X, Y coordinates of points designed in Python are reflected in in the final printed experiment. This is important because the definition of optimal path uses the designed X,Y coordinates not the printed plot where the experiment happened. The step taken to ensure this was forcing the the scaling of the plot to have a ratio of 1. Additionally, 3 problems were manually checked and the distances between points matched to those in the Python calculations.

3.2.3 Defining problems optimal path

One of the important analysis aspects of the TSP is optimality, and since the test subjects are not expected to find the optimal values, in this experiment it was used as a measure of the quality of their answers. The optimal path and cost for each

problem was determined using Gurobi in Julia. Gurobi is a powerful optimization software for linear problems. The mathematical TSP optimization problem is defined in Equation 3.4:[7]

$$\begin{aligned}
\min \quad & \sum d_{ij}x_{ij} \\
\text{s.t.:} \quad & \sum_{j=1}^{n} x_{ij} = 1 && i = 1, 2, ..., n \\
& \sum_{i=1}^{n} x_{ij} = 1 && j = 1, 2, ..., n \\
& \sum_{i,j \in S} x_{ij} \leq |S| - 1 \quad 2 \leq |S| \leq n-2,\ S \subset \{1, 2, ..., n\} \\
& x_{ij} \in \{0, 1\} && i, j = 1, 2, ..., ni \neq j
\end{aligned} \tag{3.4}$$

Where d_{ij} is the distance between city i and city j, and $x_{ij} = 1$ means a path taken by the salesperson while $x_{ij} = 0$, means a path not taken. The first constraint determines that there will be only one departure per city, the second determine only one entrance per city. This does not rule out the possibility of multiple loops. Therefore, the third constraint is needed to assure that no loop in "any city subset should be formed", and "$|S|$ means the number of elements included in the set S"[7].

3.2.4 Data collection method

Mainly, there were two values collected from each experiment, the time it took the person to complete each problem and the path taken to solve the problem. The time it took the person to complete the test was recorded using the cameras subtracting the completion time to the initial time of the problem. The completion time was defined by when the person flipped the page to start the next problem and the initial time was defined once the problem was set on the table, ready to be solved. The path for each problem was collected by assigning each point an identification number and recording the order in which the cities were visited. For example in Figure 3-2, starting arbitrarily from point 6 and going clock-wise, the path taken is [6,1,7,5,4,2,3] and implicitly it comes back to 6. Each point location and distance to other points

was easily identifiable in Excel tables generated from Python.

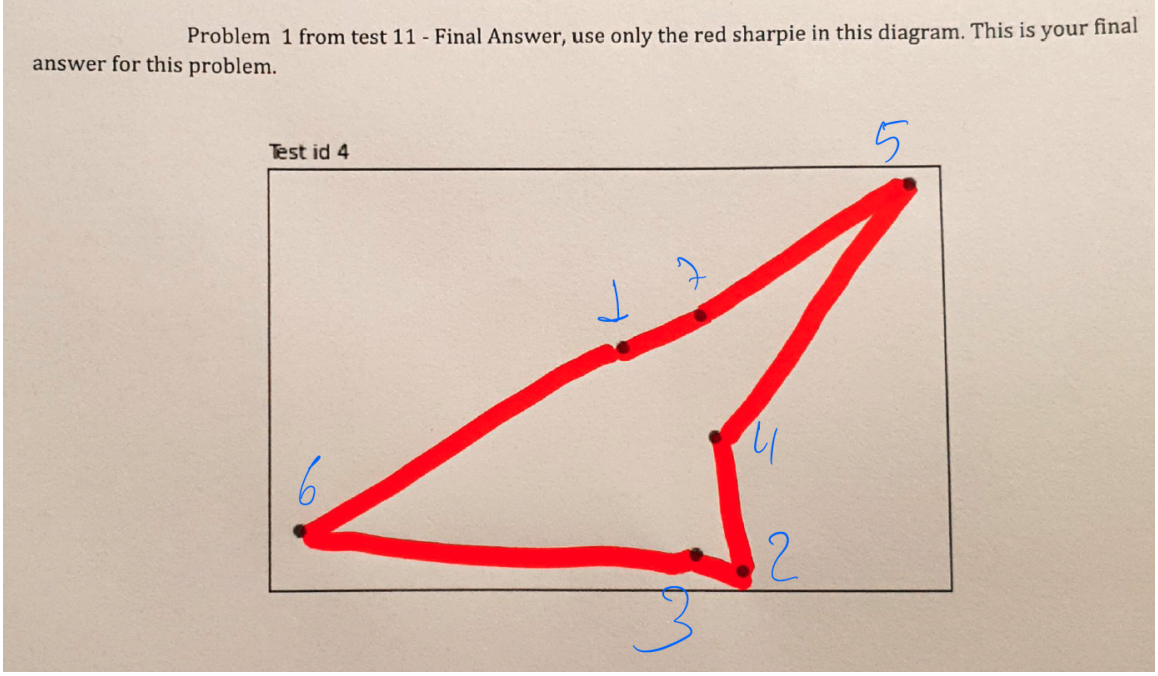


Figure 3-2: Sample from test 11, problem 1, for path recording

3.3 Results and Analysis

There were 25 total volunteers for the experiment, each completed 13 tests, one full experiment and other 2 problems were lost due to technical failure with the camera, totaling 310 valid tests completed. Results are summarized in Table 3.3, with how many times each problem was completed, the average time of completion (t), the average cost, the normalized average cost and a multiplication between the average time and the normalized average cost. The average normalized cost (NC) is the average cost normalized by the optimum path for the each problem that was determined by the method mentioned earlier in this chapter. The argument to use the $NC * t$ is based on the expectation that they are inversely proportional variables, the longer someone spends working on a problem the better the results they deliver (closer to optimal). And the shorter the time dedicated to the problem, worse the results are (larger costs).

Table 3.3: Experiment result summary

Test ID	Complexity	Test count	Av. time (s)	cost	Av. NC	Time * Av. NC
id 0	18	12	53.58	1.98	1.02	56.22
id 1	19	12	66.08	2.21	1.00	66.15
id 2	21	9	58.56	2.66	1.01	58.95
id 3	23	15	86.47	3.23	1.02	88.44
id 4	30	13	63.77	2.13	1.03	65.66
id 5	31	11	83.64	3.20	1.01	84.20
id 6	37	8	57.63	3.29	1.00	57.63
id 7	39	16	70.75	4.01	1.02	72.50
id 8	51	16	94.13	3.75	1.03	96.20
id 9	53	7	82.00	3.85	1.02	83.24
id 10	62	17	83.47	4.78	1.02	85.22
id 11	64	7	156.86	3.74	1.02	159.35
id 12	81	10	112.50	4.54	1.08	121.08
id 13	82	14	112.79	5.04	1.01	114.14
id 14	96	9	132.22	4.77	1.05	139.80
id 15	99	11	131.55	6.26	1.04	137.50
id 16	100	11	135.82	5.23	1.04	140.47
id 17	110	7	169.57	5.51	1.02	173.77
id 18	112	10	89.10	5.41	1.03	92.84
id 19	116	8	225.75	5.75	1.12	238.82
id 20	122	7	149.00	5.78	1.06	158.21
id 21	124	11	143.64	5.63	1.08	153.99
id 22	130	7	177.86	5.37	1.04	184.47
id 23	132	10	144.70	4.22	1.03	150.44
id 24	138	7	179.29	5.87	1.02	183.45
id 25	140	11	167.18	6.14	1.03	173.43
id 26	141	9	191.44	5.32	1.06	205.74
id 27	143	8	213.00	6.35	1.13	230.62
id 28	155	9	216.78	6.35	1.07	230.08
id 30	158	8	200.75	5.32	1.11	213.30

3.3.1 Initial Hypothesis

Using the data collected, it was performed on it several regression analysis using least square method. Since data collected consists of the time needed to complete the problem and the NC, it was firstly analyzed each relationships individually with respect to the problem complexity. Followed by the analysis of the relationship time needed to complete the test, times NC as a function of problem complexity. Finally these 3 analysis were performed using every data point, and aggregating it using the simple average or the weighted average. The averages are shown in Table 3.3, the weighted average used the test count to give a larger weight for tests that were taken more times. In the initial analysis there were 9 regressions. A summary of the possible variable analysis is shown below, in the left, what kind of data aggregation was performed in the data. And in the right, what types of regression formulas were used. Where t is the time to complete the problem, n_{opt} is the normalized cost, C is the problem complexity and a, b, and d are the variables to be found using the regression model.

Data used	Regression formula
• Every point	• Time as function of Complexity in the form $t(C) = aC^b$
• Average	• Percentage of optimal solution as function of Complexity in the form $n_{opt}(C) = aC^b + d$
• Weighted Average	• Time times Percentage of optimal solution as function of Complexity in the form $t * n_{opt}(C) = aC^b + d$

A overall summary of the results, with the hypothesis, the coefficeint, its value, t-test value, 70% confidence interval and R^2 is shown in the Table 3.4.

Regression on all points

The results of every data point and the time taken from each experiment are shown in Figure 3-3 and Table 3.5. On the contrary to the results shown by Sinha et Al.[20],

Table 3.4: Experiment analysis summary

Every Point

Regression	Coef	value	t-test	70% Confidence Interval	R^2
$t(C) = aC^b$	a	7.85	3.46	(5.50, 10.21)	0.321
	b	0.63	10.13	(0.56, 0.69)	
$n_{opt}(C) = aC^b + d$	a	1.6e-5	0.19	(-7.0e-5, 1.0e-4)	0.072
	b	1.66	1.60	(0.59, 2.74)	
	d	1.01	78.23	(1.00, 1.02)	
$t * n_{opt}(C) = aC^b + d$	a	0.06	0.52	(-0.06, 0.18)	0.356
	b	1.56	4.11	(1.16, 1.95)	
	d	61.09	5.17	(48.82, 73.37)	

Average

Regression	Coef	value	t-test	70% Confidence Interval	R^2
$t(C) = aC^b$	a	7.55	2.62	(4.51, 10.59)	0.775
	b	0.64	7.90	(0.56, 0.73)	
$n_{opt}(C) = aC^b + d$	a	1.8e-5	0.19	(-8.4e-5, 1.0e-4)	0.464
	b	1.64	1.55	(0.53, 2.76)	
	d	1.01	69.00	(0.99, 1.02)	
$t * n_{opt}(C) = aC^b + d$	a	0.14	0.42	(-0.22, 0.49)	0.802
	b	1.40	3.05	(0.91, 1.88)	
	d	56.64	3.11	(37.39, 75.90)	

Weighted Average

Regression	Coef	value	t-test	70% Confidence Interval	R^2
$t(C) = aC^b$	a	7.86	9.50	(7.00, 8.71)	0.780
	b	0.63	27.81	(0.61, 0.65)	
$n_{opt}(C) = aC^b + d$	a	1.6e-5	0.65	(-9.5e-6, 4.1e-5)	0.476
	b	1.66	5.47	(1.35, 1.98)	
	d	1.01	266.9	(1.01, 1.01)	
$t * n_{opt}(C) = aC^b + d$	a	0.06	1.48	(0.02, 0.10)	0.819
	b	1.56	11.76	(1.42, 1.70)	
	d	61.09	14.77	(56.80, 65.39)	

where they had a super-linear relationship between complexity and time, with the exponent $b = 1.47$, here it is shown a sub-linear relationship, $b = 0.629$. This can be attributed to time of re-work, as in Sinha and De Weck's study, the total time to compute was the sum of the time to construct their model and the time of re-work, as it was not accepted any incorrect built model and the correct solution was given to them to mimic it. In this study, there are $(n-1)!/2$ [7] possible solutions, it is unfeasible to expect every human to be able to find the optimal solution for any n above 10 (more than 180 thousand possible paths). Also, another hypothesis why this happens will be discussed later in the chapter.

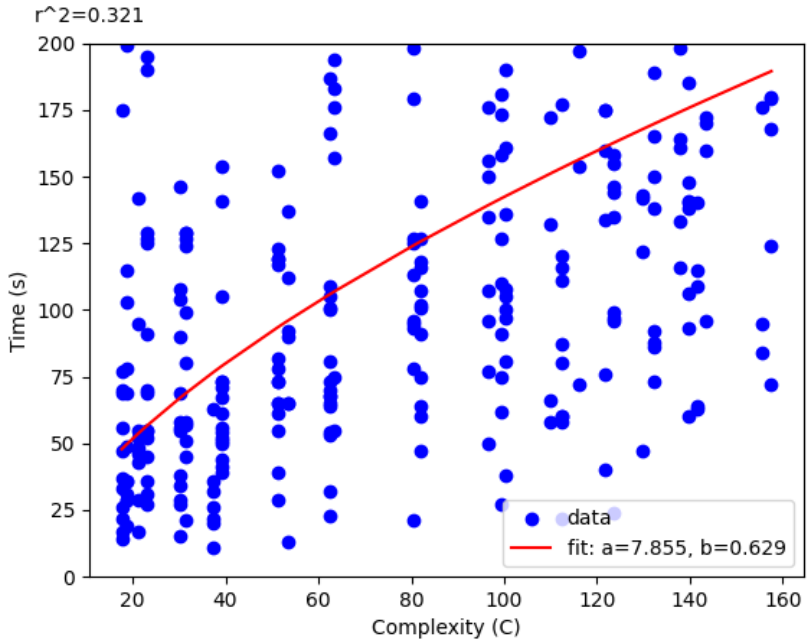


Figure 3-3: Regression Analysis, $t(C) = aC^b$, for every point, $R^2 = 0.321$

Table 3.5: Regression Analysis summary, $t(C) = aC^b$, for every point, $R^2 = 0.321$

	value	standard error	t-test	p-value	70% Confidence Interval
a	7.85	2.27	3.46	0.00	(5.50, 10.21)
b	0.63	0.06	10.13	0.00	(0.56, 0.69)

The resulting analysis from using every data point and taking the NC from each experiment are shown in Figure 3-3 and Table 3.5. Although the R^2 is low, there is a clear upward trend in the complexity and the distance from the optimal value. The offset $d = 1.01$ is necessary because the results are limited to the optimal path and it shows that even at very low complexities it is not expected to have perfect optimal solutions from the test subjects.

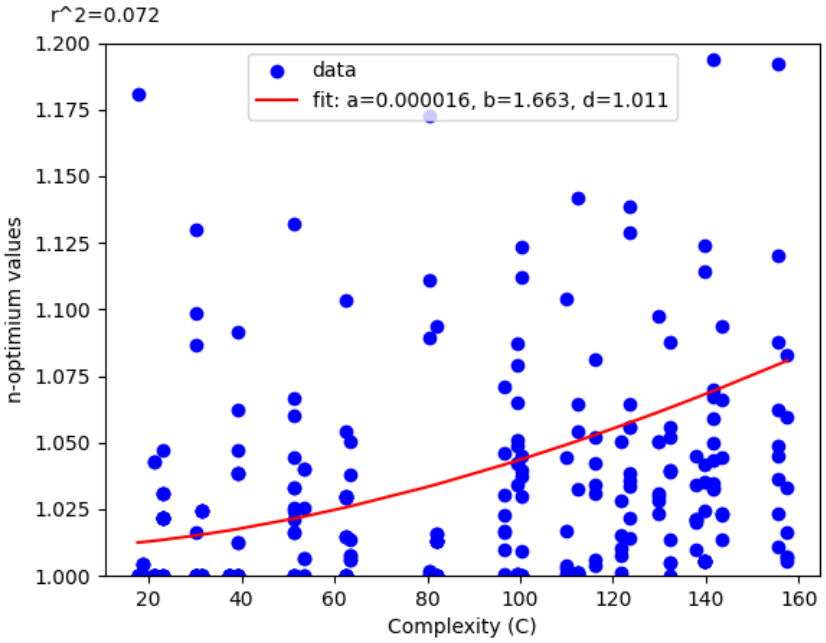


Figure 3-4: Regression Analysis, $n_{opt}(C) = aC^b + d$, for every point, $R^2 = 0.072$

Table 3.6: Regression Analysis summary, $n_{opt}(C) = aC^b + d$, for every point, $R^2 = 0.072$

	value	standard error	t-test	p-value	70% Confidence Interval
a	1.6e-5	8.2e-5	0.19	0.85	(-7.0e-5, 1.0e-4)
b	1.66	1.04	1.60	0.11	(0.59, 2.74)
d	1.01	0.01	78.23	0.00	(1.00, 1.02)

Finally, as previously mentioned, since in this study the test subjects were allowed

to find sub-optimal solutions, another hypothesis was tested. It was multiplied NC by the time to complete. The goal function is $t * n_{opt} = aC^b + d$. This hypothesis implies a tradeoff between time spent in the problem and the search to finding the optimal solution for each problem. The subjects that spend more time in the problem are expected to find results closer to the optimal value. The results are shown in Figure 3-5 and Table 3.7. This hypothesis generated better, more relevant statistical metrics than the previous analysis, with a $R^2 = 0.356$, and t value of 1.6 for the exponent $b = 1.56$.

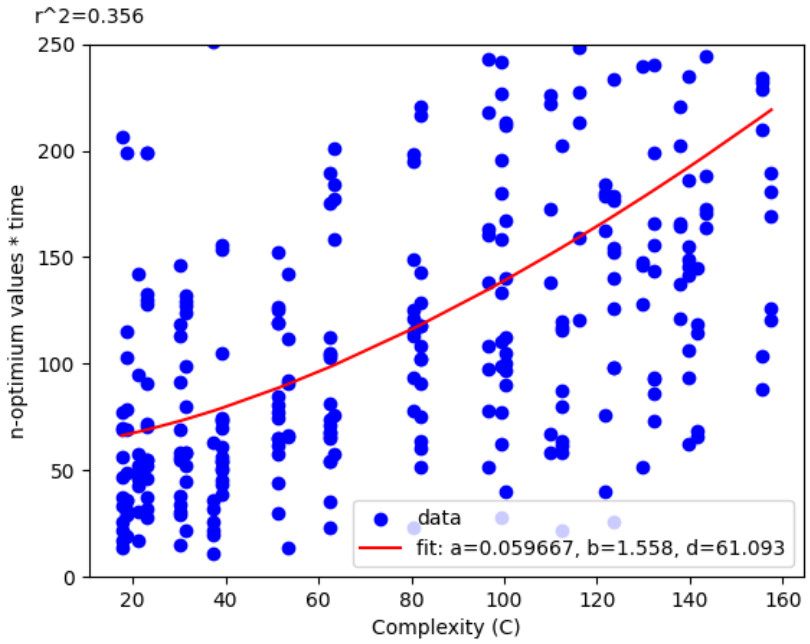


Figure 3-5: Regression Analysis, $n_{opt} * t(C) = aC^b + d$, for every point, $R^2 = 0.356$

Table 3.7: Regression Analysis summary, $n_{opt} * t(C) = aC^b + d$, for every point, $R^2 = 0.356$

	value	standard error	t-test	p-value	70% Confidence Interval
a	0.06	0.12	0.52	0.61	(-0.06, 0.18)
b	1.56	0.38	4.11	0.00	(1.16, 1.95)
d	61.09	11.82	5.17	0.00	(48.82, 73.37)

Regression on average based on problem complexity

The next set of analysis were performed using the average time, and average NC. Again, it will be presented in the order of time as function of complexity ($t(C) = aC^b$), NC as function of complexity ($n_{opt}(C) = aC^b + d$), and lastly, time multiplied by the NC as a function of complexity($n_{opt} * t(C) = aC^b + d$).

Firstly, the results of the regression of time as function of complexity ($t(C) = aC^b$) are shown in Figure 3-6 and Table 3.8. Similar to the previous section, the relationship is sub-linear. However, the statistical relevance increased substantially, achieving a $R^2 = 0.775$ and a t value of 7.90 for the exponent $b = 0.64$.

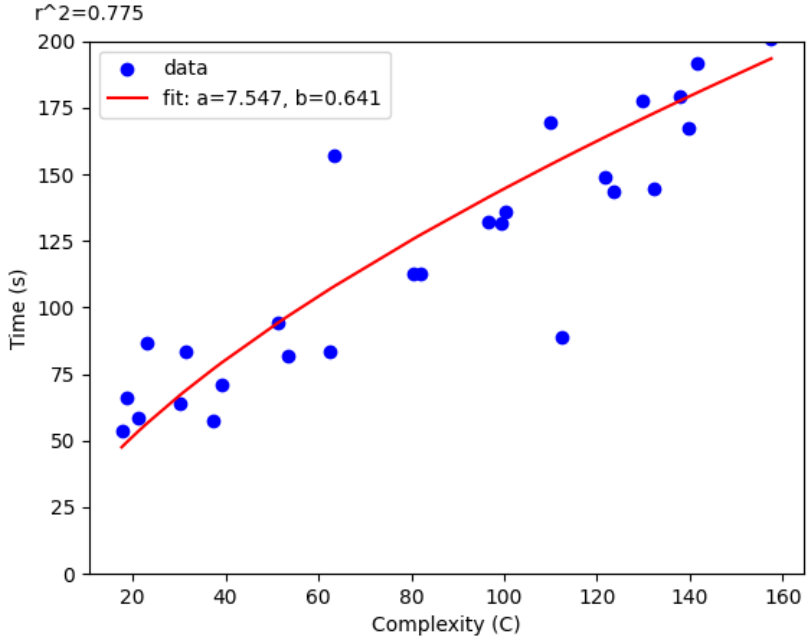


Figure 3-6: Regression Analysis, $t(C) = aC^b$, averaged, $R^2 = 0.775$

Table 3.8: Regression Analysis summary, $t(C) = aC^b$, averaged, $R^2 = 0.775$

	value	standard error	t-test	p-value	70% Confidence Interval
a	7.55	2.88	2.62	0.01	(4.51, 10.59)
b	0.64	0.08	7.90	0.00	(0.56, 0.73)

Secondly, the results of the regression of time as function of complexity ($n_{opt}(C) = aC^b + d$) are shown in Figure 3-7 and Table 3.9.

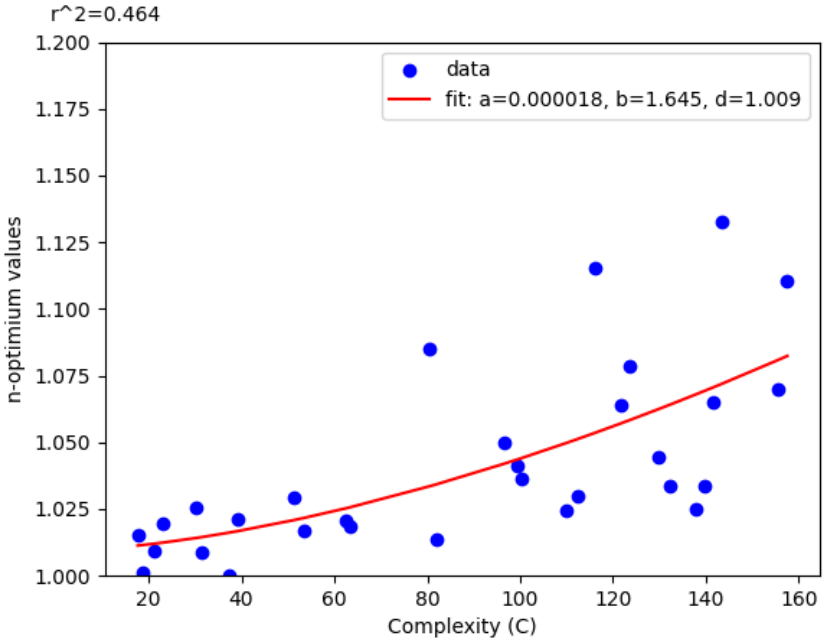


Figure 3-7: Regression Analysis, $n_{opt}(C) = aC^b + d$, averaged, $R^2 = 0.464$

Table 3.9: Regression Analysis summary, $n_{opt}(C) = aC^b + d$, averaged, $R^2 = 0.464$

	value	standard error	t-test	p-value	70% Confidence Interval
a	1.8e-5	9.6e-5	0.19	0.85	(-8.4e-5, 1.0e-4)
b	1.64	1.06	1.55	0.13	(0.53, 2.76)
d	1.01	0.01	69.00	0.00	(0.99, 1.02)

Finally, the results of the regression using time multiplied by NC as a function of complexity($n_{opt} * t(C) = aC^b + d$) are shown in Figure 3-8 and Table 3.10. The results show a super-linear relationship with the exponent $b = 1.40$, with a statistically significant t-value of 3.05 and $R^2 = 0.802$. This is very similar to the results found by Sinha and De Weck in their study, increasing the hypothesis that effort has a super-linear relationship with complexity.

Table 3.10: Regression Analysis summary, $n_{opt} * t(C) = aC^b + d$, averaged, $R^2 = 0.802$

	value	standard error	t-test	p-value	70% Confidence Interval
a	0.14	0.33	0.42	0.68	(-0.22, 0.49)
b	1.40	0.46	3.05	0.00	(0.91, 1.88)
d	56.64	18.24	3.11	0.00	(37.39, 75.90)

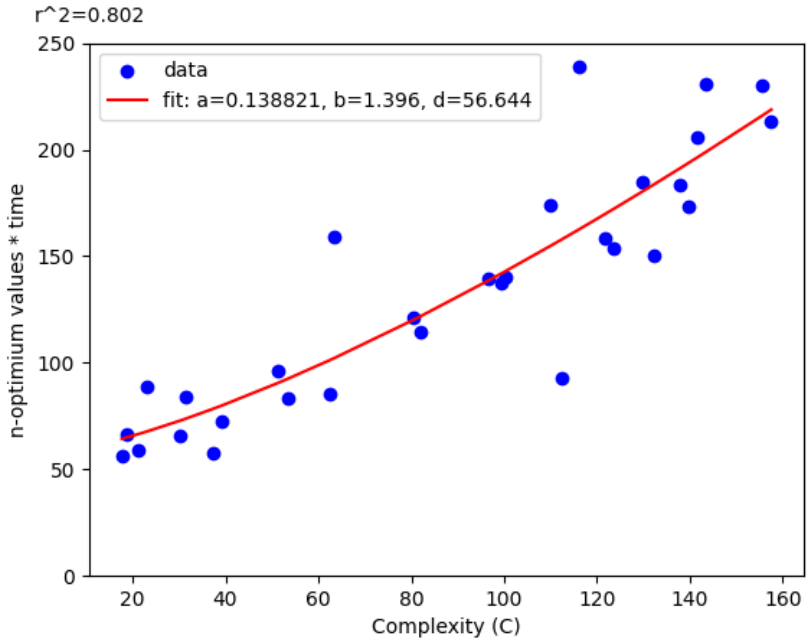


Figure 3-8: Regression Analysis, $n_{opt} * t(C) = aC^b + d$, averaged, $R^2 = 0.802$

Regression on weighted average based on problem complexity

Using the same order from the previous section, the analysis in this section were performed using a weighted average. This means that each value of the 310 tests were substituted by the average of the respective complexity. This will give problems that were conducted more times higher weight in regression. The results are shown in order, regression on time, Figure 3-9 and Table 3.11, Figure 3-10 and Table 3.12, and Figure 3-11 and Table 3.13 respectively. The most relevant analysis here is that due to the increased number of data points when compared to the simple average is the improved t value (from 3.05 to 11.76) and a much narrower confidence interval(from

[0.91,1.88] to [1.42,1.70]) for the exponent $b = 1.56$ in the time multiplied by the NC when compared to the average results.

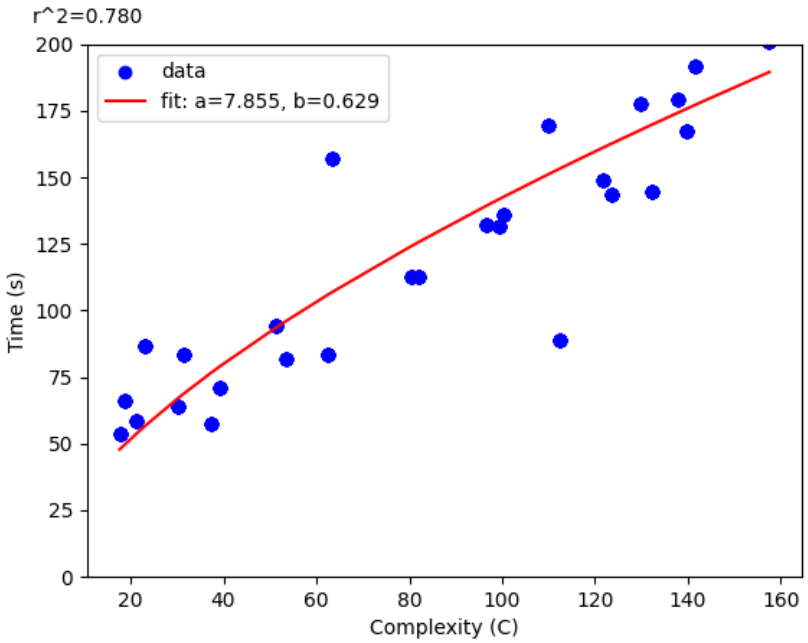


Figure 3-9: Regression Analysis, $t(C) = aC^b$, weighted average, $R^2 = 0.780$

Table 3.11: Regression Analysis summary, $t(C) = aC^b$, weighted average, $R^2 = 0.780$

	value	standard error	t-test	p-value	70% Confidence Interval
a	7.86	0.83	9.50	0.00	(7.00, 8.71)
b	0.63	0.02	27.81	0.00	(0.61, 0.65)

Table 3.12: Regression Analysis summary, $n_{opt}(C) = aC^b + d$, weighted average, $R^2 = 0.476$

	value	standard error	t-test	p-value	70% Confidence Interval
a	0.00	0.00	0.65	0.52	(-9.5e-6, 4.1e-5)
b	1.66	0.30	5.47	0.00	(1.35, 1.98)
d	1.01	0.00	266.92	0.00	(1.01, 1.01)

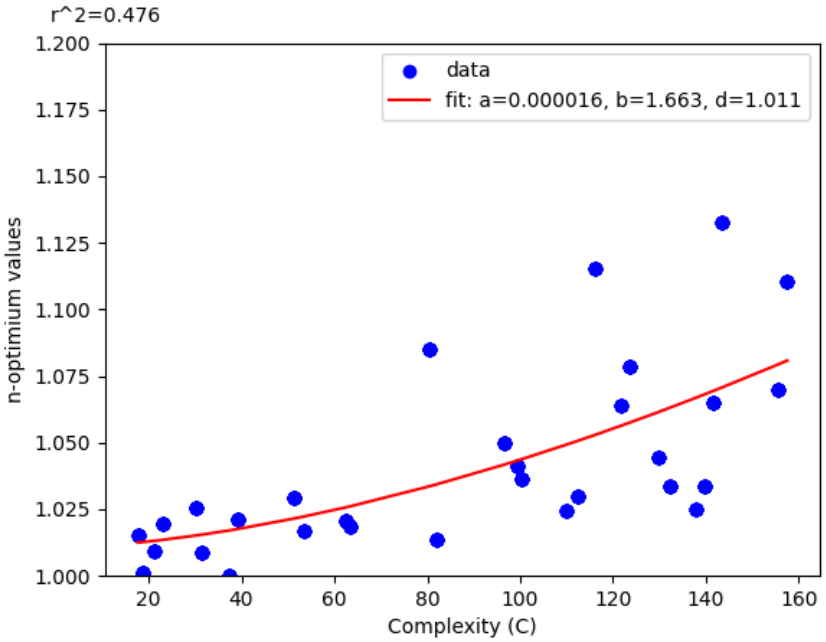


Figure 3-10: Regression Analysis, $n_{opt}(C) = aC^b + d$, weighted average, $R^2 = 0.476$

Table 3.13: Regression Analysis summary, $n_{opt} * t(C) = aC^b + d$, weighted average, $R^2 = 0.819$

	value	standard error	t-test	p-value	70% Confidence Interval
a	0.06	0.04	1.48	0.14	(0.02, 0.10)
b	1.56	0.13	11.76	0.00	(1.42, 1.70)
d	61.09	4.14	14.77	0.00	(56.80, 65.39)

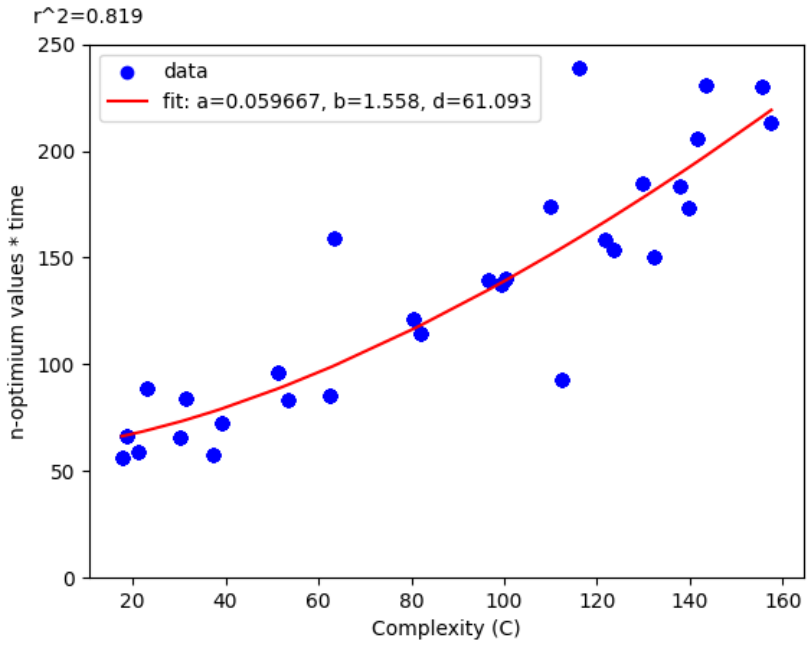


Figure 3-11: Regression Analysis, $n_{opt} * t(C) = aC^b + d$, weighted average, $R^2 = 0.819$

Residuals

To further analyze and validate the results, the residuals for each regression was created and shown in Figure 3-12. In the x-axis, it is shown the sample number, this is ordered based on the problem complexity and in the y-axis its the residual. Ideally, for a least square method the samples are normally randomly distributed around the zero value. In the experiment, values with larger complexity have larger residuals. This can be seen better in the first column when every point is taken into consideration. Although, this is an important measure, the skewness in this analysis

is not enough to discredit the analysis mentioned in the previous sections.

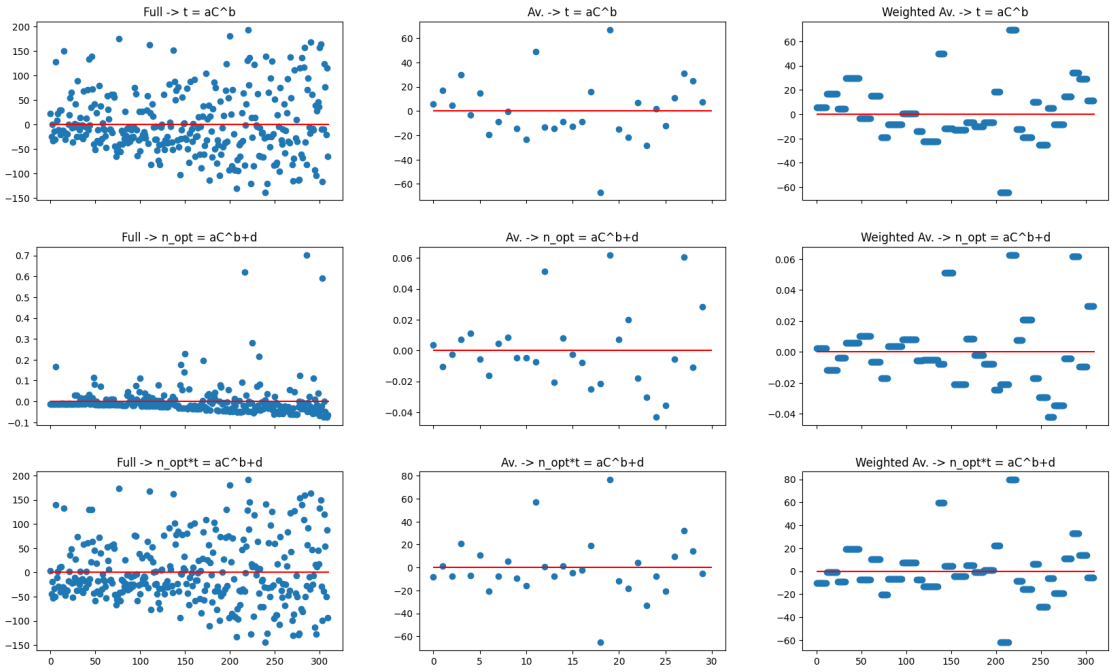


Figure 3-12: Residuals for the regressions. Columns: 1. Using every point, 2. Averaged, 3. Weighted Average. Rows: 1. $t(C) = aC^b$, 2. $n_{opt}(C) = aC^b + d$, 3. $n_{opt} * t(C) = aC^b + d$

3.3.2 Further analysis

Once the analysis in the previous section was completed, it was noticed something interesting, specially the three plots based on the regression where NC and time were multiplied ($n_{opt} * t(C) = aC^b + d$). These had better statistics results, with better R^2, p-values and t-test. These were also the only analysis that allowed the time of completion to have an offset or a Y intercept different than zero. So another hypothesis was created: $t(C) = aC^b + d$. The results are summarized in Table 3.14 and shown in Figures 3-13, 3-14, and 3-15. For the weighted average, the b coefficient is 1.46, with a 70% interval confidence between 1.33 and 1.60, with very high t-tests for the exponent b and intercept d, of 11.4 and 13.9 respectively. All these regressions perform very closely to the $n_{opt} * t(C) = aC^b + d$ option. Since the NC were very close to optimal, varying in average from 0% to 13% above the optimal, it has a

much smaller impact when comparing to the time for completion. Additionally, this results are different from that shown in previous studies mentioned earlier, where time was expected to approach zero as complexity approaches zero. Here, it is shown that even for the least complex problems there is an overhead cognitive load where the test subjects take to first understand the problems before solving it. Once that initial hurdle has passed, then the complexity of the task increases the time to solve super-linearly.

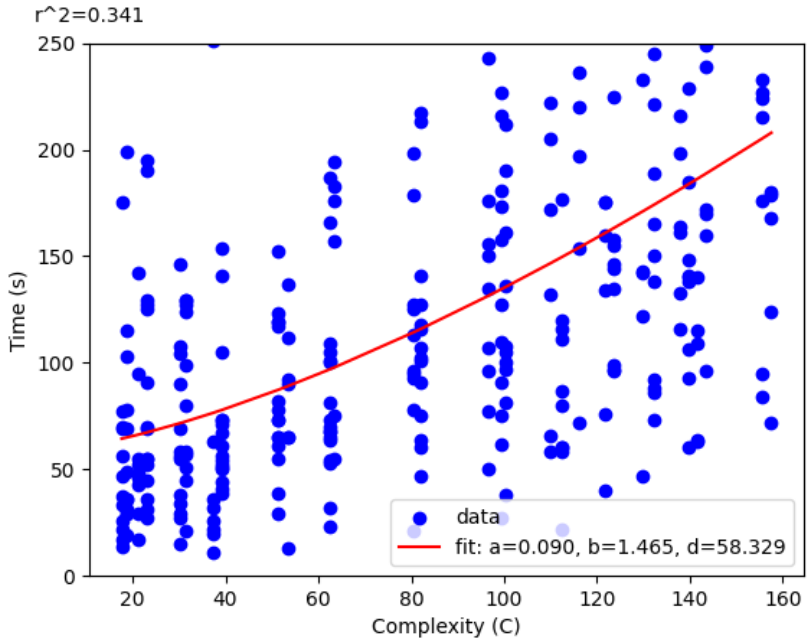


Figure 3-13: $t(C) = aC^b + d$ using all points

3.3.3 Uncertainty on inner and outer point values

Up to this point, it was assumed the values in Table 3.1 to calculate complexity, however these numbers are not absolute and at this stage of research there is a good amount of uncertainty. To address, and mitigate this issue, a Monte-Carlo simulation was performed in the α and in the β coefficients to calculate problem complexity and regressions in the form $n_{opt} * t(C) = aC^b + d$ were performed. It was chosen the triangular uncertainty for every coefficient as it is only needed 3 values, a pessimistic,

Table 3.14: Summary results for $t(C) = aC^b + d$

Every point

Regression	Coef	value	standard error	t-test	p-value	70% Confidence Interval
$t(C) = aC^b + d$	a	0.09	0.18	0.51	0.61	(-0.09, 0.27)
	b	1.46	0.38	3.84	0.00	(1.07, 1.86)
	d	58.33	12.49	4.67	0.00	(45.4, 71.3)

Average

Regression	Coef	value	standard error	t-test	p-value	70% Confidence Interval
$t(C) = aC^b + d$	a	0.21	0.48	0.43	0.67	(-0.30, 0.72)
	b	1.31	0.45	2.93	0.01	(0.84, 1.78)
	d	53.65	18.74	2.86	0.01	(33.88, 73.42)

Weighted Average

Regression	Coef	value	standard error	t-test	p-value	70% Confidence Interval
$t(C) = aC^b + d$	a	0.09	0.06	1.52	0.13	(0.03, 0.15)
	b	1.46	0.13	11.43	0.00	(1.33, 1.60)
	d	58.33	4.19	13.91	0.00	(54.0, 62.7)

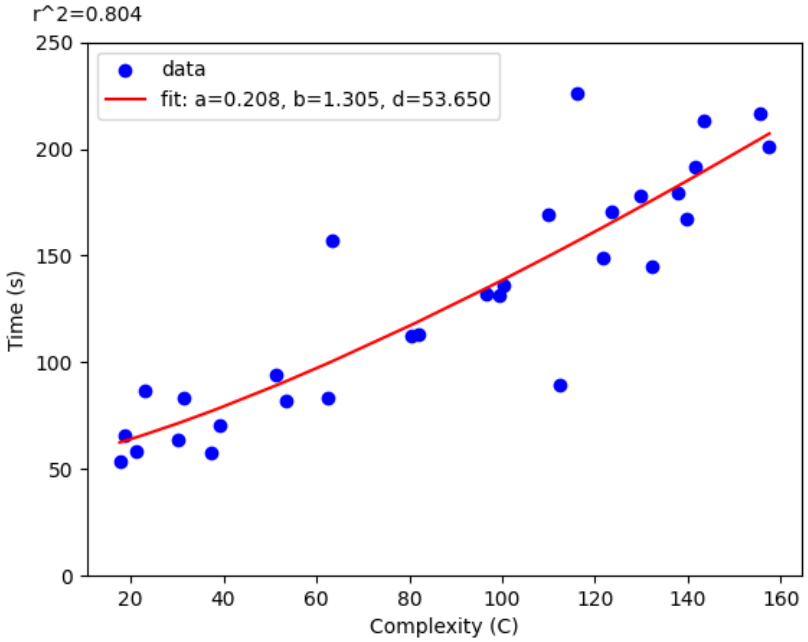


Figure 3-14: $t(C) = aC^b + d$ using point average

Table 3.15: Base complexity with uncertainty for points and interfaces

	Point	Complexity type	Complexity	Uncertainty Distribution (min/max/mode)
Component	Inner point	α_i	2	Triangular(0,3,2)
	Outer point	α_i	1	Triangular(0,2,1)
Interface	Inner point	β	3	Triangular(0,3,2)
	Outer point	β	1	Triangular(0,2,1)

an optimistic, and a most probable value as shown in Table 3.15. Since the coefficients are relative to one another, I believe that with enough Monte-Carlo samples these distributions will yield meaningful results.

The most important variable that is being tracked in this study is the exponent b, which is of interest to understand the relationship between effort and complexity. The coefficient a is a scaling factor, which is highly dependent of the α and β values, by multiplying them by a factor of x, it would increase the complexity by a factor of y, and that would be reflected in coefficient a. d is a necessary offset because of the initial overhead of analyzing the problem. Thus, b was the value chosen to be tracked in

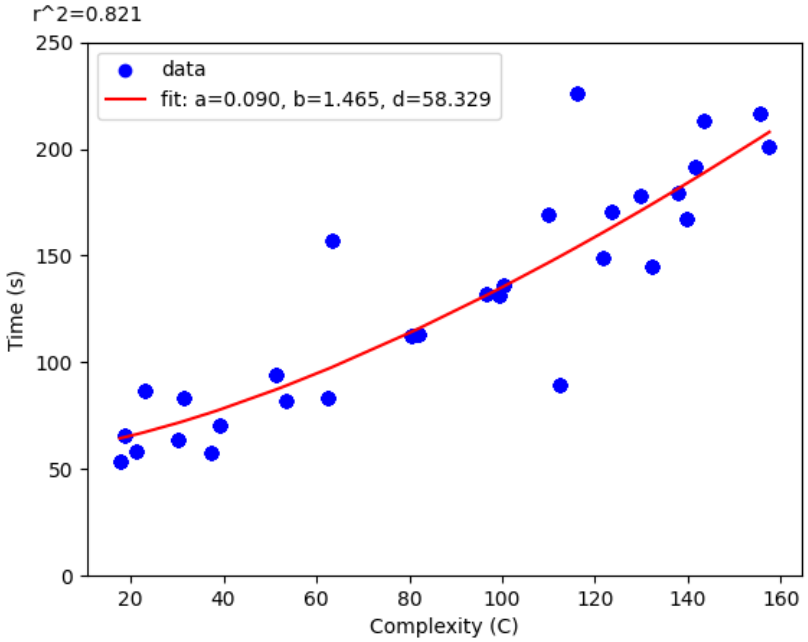


Figure 3-15: $t(C) = aC^b + d$ using weighted point average

the Monte-Carlo simulation. Figure 3-16 shows the cumulative distribution function of this variable across 10 thousand simulations. In the figure, it is also shown the deterministic value, in a vertical dashed line, the Monte-Carlo median and average, in a red dot, and a red vertical line respectively. In the simulation, the minimum value recorded was 1.1 and the maximum was 1.9, with the average and median being slightly over 1.5. In comparison, the deterministic value was 1.4. This result increases the confidence that the complexity and effort relationship is super-linear.

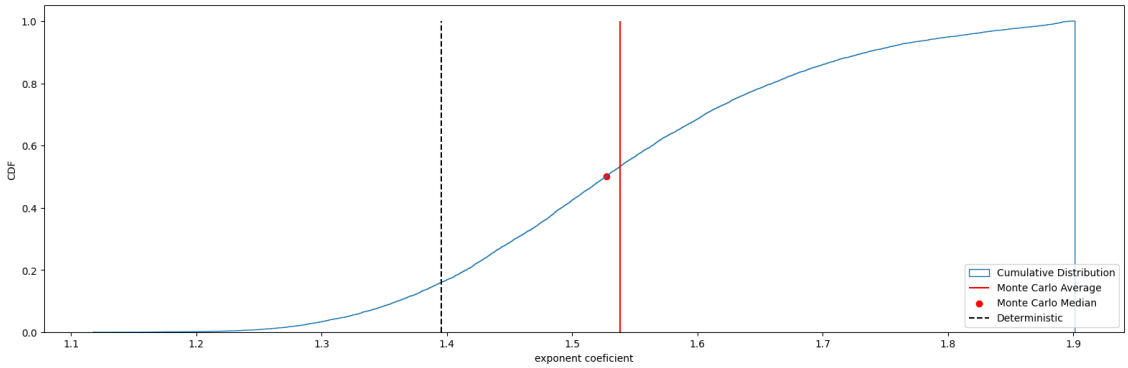


Figure 3-16: Monte-Carlo simulation for exponent b

Also, for each problem it was recorded the new complexity for each iteration of the Monte Carlo and is shown in Figure 3-17 in boxplot format. Although, the variability seems to be large, the upward complexity from the simpler problems in the left to the more complex problems is still valid. However, there were problems that at first, in the deterministic analysis, were more complex than other, but depending on the iteration had lower complexity. Two examples are problems Id14 and Id13, and problems Id28 and Id30.

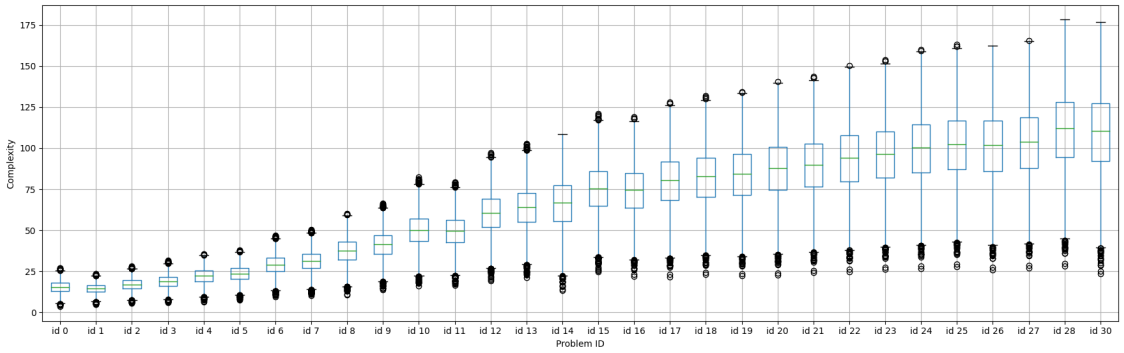


Figure 3-17: Monte-Carlo problem complexity variability

3.3.4 Normalized Cost and Normalized Time

One assumption made in the study was that the normalized cost and time were inversely proportional. So this section is dedicated on challenging this assumption. to do that, it was analyzes the normalized cost versus the normalized time (NT). NT was defined as the time to solve a problem divided by the average of time to solve that problem. Figure 3-18 shows the inverse relationship between NC and NT, however is it arguably a very slow relationship as the fit suggests $NC = -9e{-}3NT + 1.05$, when doubling the time taken in the problem, the NC is only increased by 0.9%. Also, Figure 3-19 shows that simpler problems have larger variations in NT, this happens mainly because the average time taken in those are smaller. Figure 3-20 was created using only the data collected for problem ids equal or larger than 11 or a complexity value above 63 to understand the more complex problems. This

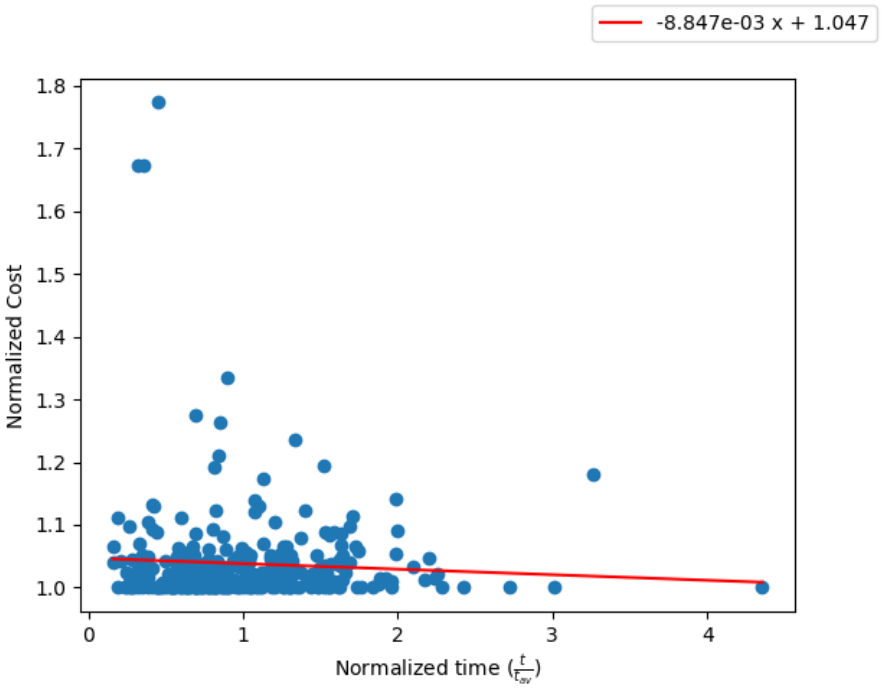


Figure 3-18: Normalized cost vs. Normalized time

plot shows a stronger relationship, accounting for 3.6% improvement in NC for every additional average time incremented in the problem. From both Figures 3-18 and 3-20, it can be argued that lower NT yields to more inconsistent results, leaving the possibility for large errors while taking longer does not eliminate the possibility of sub-optimal solutions, but reduces the possibility of having large errors. For example, the standard deviation in NC for NT smaller than 1 was 0.1, and for NT larger than 1 was 0.04 which corroborate with this analysis. The difference is even larger when taking into consideration only the problems with complexity over 63, with a standard deviation of 0.13 versus 0.05 for NT smaller than 1 and over 1 respectively.

Moreover, Figure 3-21 shows the average NT and average NC for each test subject. Here, each data point is a different test subject. Clearly there is a downward trend, where more time is spent better results surface.

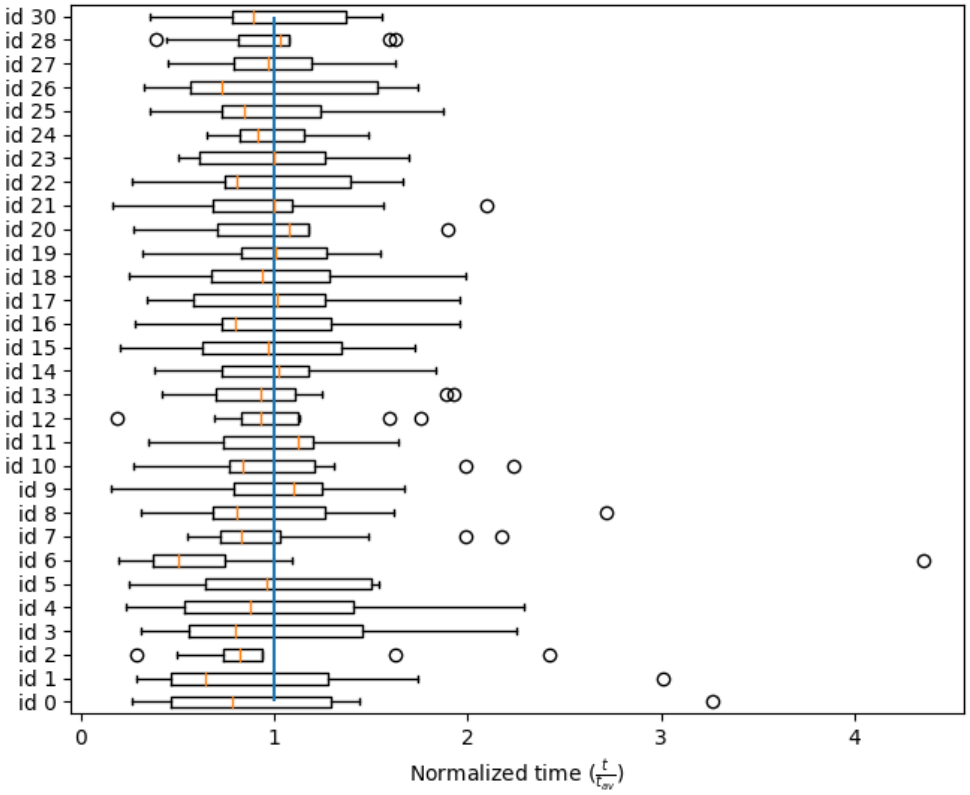


Figure 3-19: Normalized time, boxplot by problem id

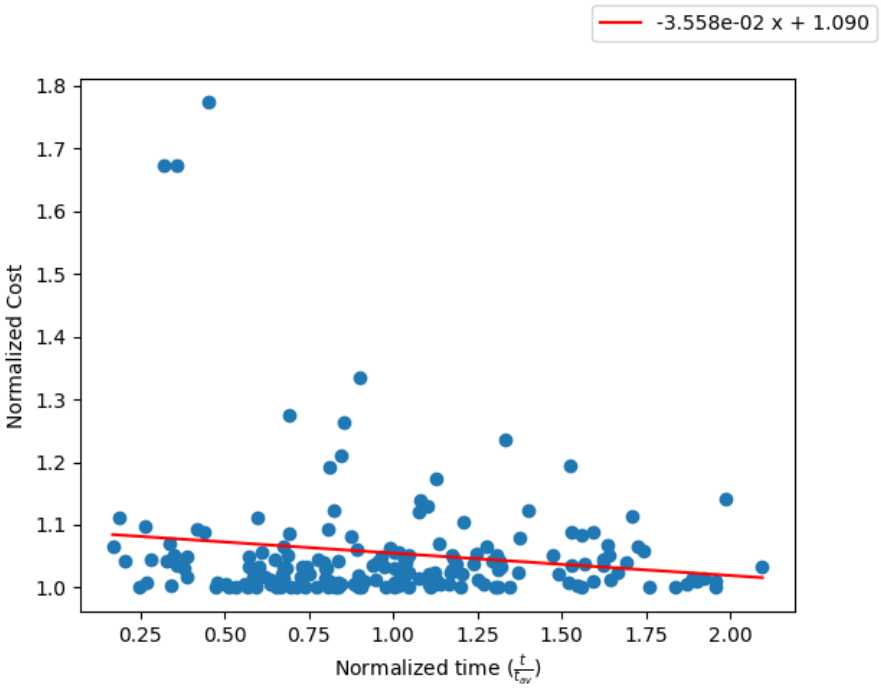


Figure 3-20: Normalized cost vs. Normalized time for complexities larger than 63 (id 11 and above)

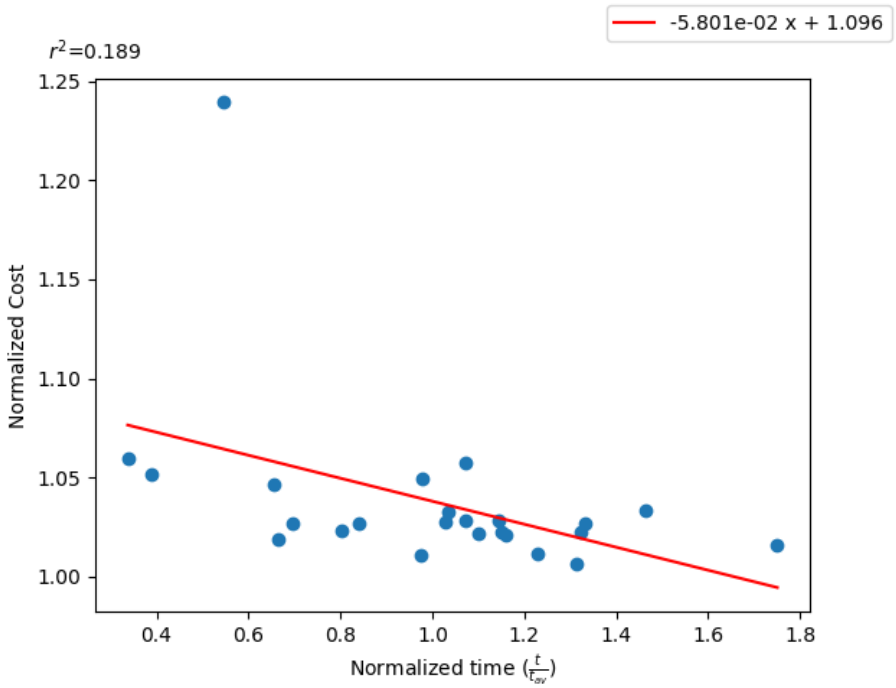


Figure 3-21: Average normalized time versus Average normalized cost for each test subject

.

Table 3.16: Base complexity and 10% sensitivity

	Point	Complexity type	Complexity	Sensitivity test
Component	Inner point	α_i	2	[1.8, 2.2]
	Outer point	α_i	1	[0.9, 1.1]
Interface	Inner point	β	3	[2.7, 3.3]
	Outer point	β	1	[0.9, 1.1]

3.3.5 Sensitivity

Finally, a sensitivity analysis was performed in the component and interface complexity values as shown in Table 3.16. By varying 10% in each of the complexity type, it was analyzed the overall average percentage complexity change across all 30 problems. The results are shown in Figure 3-22. Since the problems in this experiment have in average more inner points (12.0) than outer points (8.7), it was expected the inner points to have a larger impact in the sensitivity. Also, inner points have larger complexity values, thus changing 10% will lead to a larger change in the total Complexity. Similarly, the interface component β has a larger impact in both inner and outer points. This is due to the multiplier C_3 term, which in every problem is above 1, and with an average of 1.28. The C_3 term does not change, as it is independent of α and β, and dependent only whether there is a connection between two cities.

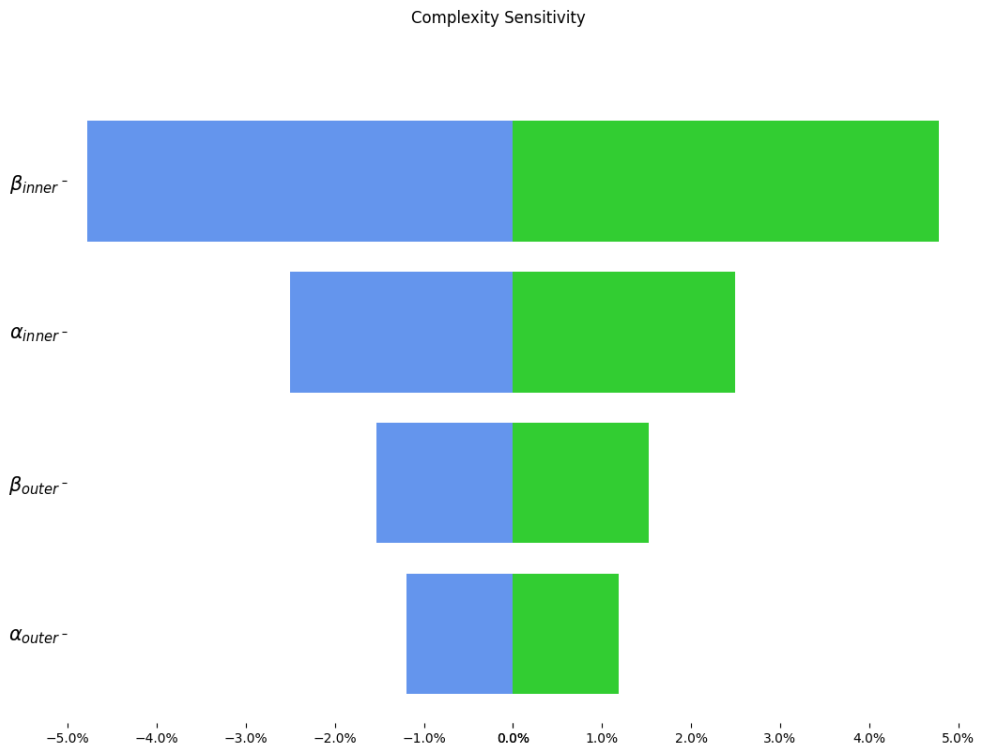


Figure 3-22: Sensitivity analysis, impact on average Complexity by a change in 10% in coefficient

.

Additionally, the impact of the ratios between inner and outer component and interface complexity on the exponent b was analyzed using the average results for each problem. As mentioned previously, although there is a precedent and strong reasoning that inner points contribute more to the complexity of the problems, it is not know exactly by how much, or what are the values. Figure 3-23, shows this analysis for ratios over one, where inner points contribute more the complexity. In this figure, both x and y axis are in the range from 1 to 10, and the b coefficient is always over one, in the range from 1.7 to 1.2. The red dot represents the deterministic data point used in the regressions showed earlier, where the ratio for $\frac{\alpha_{inner}}{\alpha_{outer}} = 2$ and $\frac{\beta_{inner}}{\beta_{outer}} = 3$ yielding $b = 1.396$

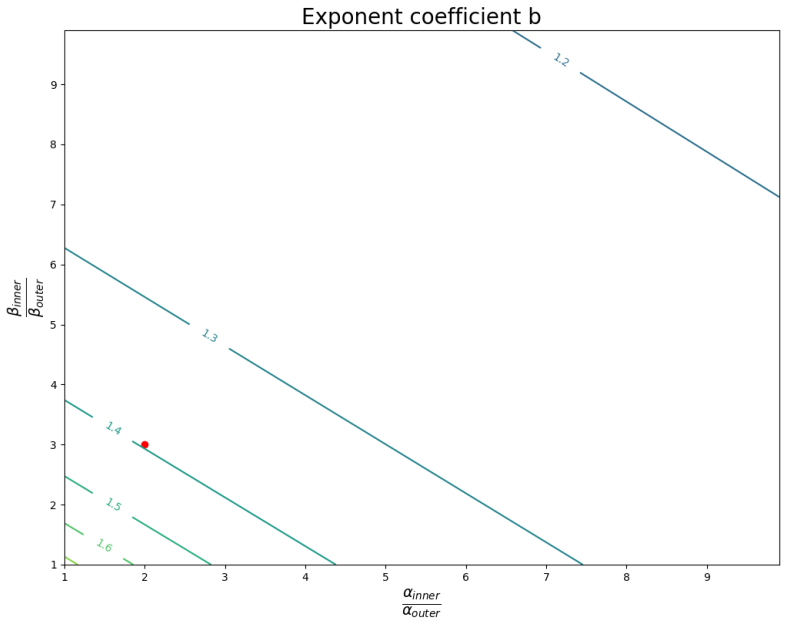


Figure 3-23: $\frac{\alpha_{inner}}{\alpha_{outer}}$ vs. $\frac{\beta_{inner}}{\beta_{outer}}$ impact on exponent coefficient b, the red dot represents the deterministic data point used in the analysis, for ratios over 1

.

Similarly, Figure 3-24 shows the same relationship between the ratios of inner and outer points, however in this case the outer points have a larger impact than inner points. In this case, there is a faster change in the exponent coefficient, reaching almost at 1.9, and below 1 for very small ratios of both α and β, around 0.3 and 0.21 respectively. Based on previous studies it is possible to eliminate the possibility that inner points will have a much lower impact in the complexity than outer points, increasing the confidence of the super-linear relationship attached to the b coefficient.

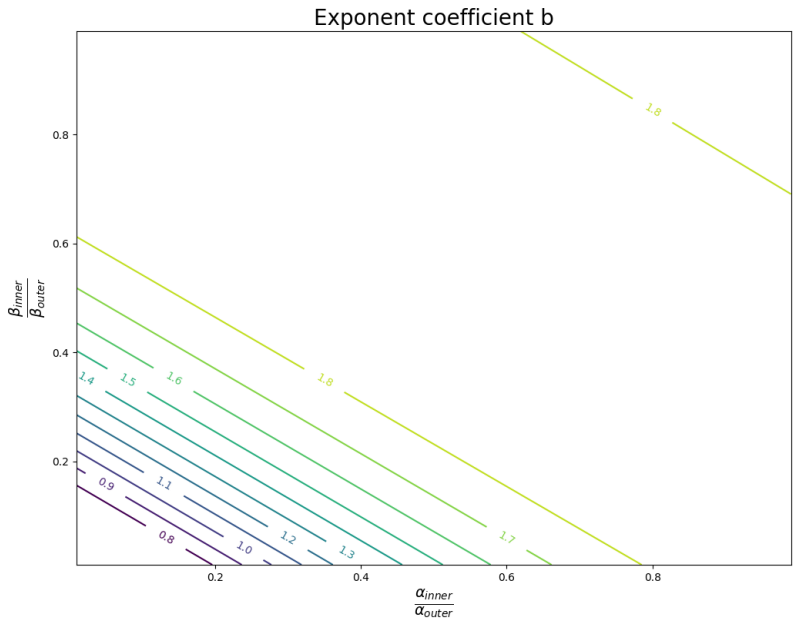


Figure 3-24: $\frac{\alpha_{inner}}{\alpha_{outer}}$ vs. $\frac{\beta_{inner}}{\beta_{outer}}$ impact on exponent coefficient b, for ratios under 1

.

3.3.6 Variability

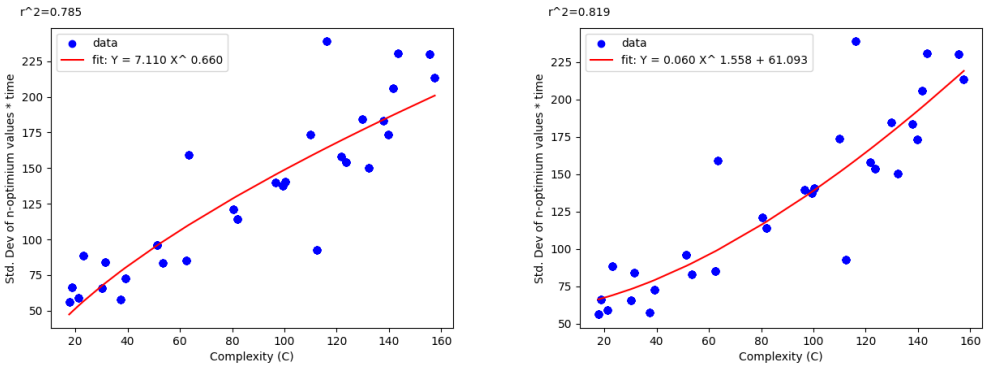
The variability, or the standard deviation for each problem was also calculated and is shown in Table 3.17. It interesting to measure the how the variability in the results vary across different complexities. Similar to the previous analysis, here the hypothesis, visually corroborated by residuals shown in Figure 3-12, is that variability increases as complexity increases. Two regressions were created in the following forms:

$$Y = aC^b \tag{3.5}$$

$$Y = aC^b + d \tag{3.6}$$

Where Y is the expected variability in the NC multiplied by the time for the system complexity C, and a, b and d are constants. A summary of the results is shown in Table 3.18. Also depicted in Figures 3-25a and 3-25b. Both models perform relatively well, with fairly high t-values and nearly zero p-values, however Equation 3.5 performs

slightly worse, and seems to underestimate at higher complexities while Equation 3.5 seems to over and underestimate randomly across all complexities, as expected. It also implies that even at when approaching problem with zero complexity, there is a overhead human variability to assimilate and solve the problem, different people will solve in different speeds even the simplest of the problems which does not happen at Equation 3.5. Thus, Equation 3.6 seems to be a better approximation. After the initial overhead, as complexity increases the variability of the results increase super-linearly.



(a) (b)

Figure 3-25: Variability model:
(a) $Y = 7.110X^{0.660}$, $R^2 = 0.785$,
(b) $Y = 0.060X^{1.558} + 61.093$, $R^2 = 0.819$

3.3.7 Learning effect

To minimize the learning effect, ordering of the problems within a test was randomized, this was intended to avoid having an ascending or descending complexity order which could have skewed the results. To better understand the impact of learning in the experiment, three plots were generated, the time taken in each problem, the normalized time and the normalized cost, as a function of the solving sequence. First, Figure 3-26a does not show evidence of more time being spent in either a specific problem or periods of the test. Figure 3-26b, in the other hand shows a clearly an relatively high value in the normalized time spent in the first problem solved by the

Table 3.17: Experiment analysis standard deviation summary

		Time (s)		Cost		NC		Time * NC	
id	C	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.	Mean	St. Dev.
id 0	17.67	53.58	43.91	1.98	0.10	1.02	0.05	56.22	52.06
id 1	18.65	66.08	52.11	2.21	0.00	1.00	0.00	66.15	52.11
id 2	21.33	58.56	37.85	2.66	0.05	1.01	0.02	58.95	37.71
id 3	23.12	86.47	54.82	3.23	0.04	1.02	0.01	88.44	56.86
id 4	30.26	63.77	38.42	2.13	0.10	1.03	0.05	65.66	40.17
id 5	31.31	83.64	39.65	3.20	0.04	1.01	0.01	84.20	39.75
id 6	37.30	57.63	79.65	3.29	0.00	1.00	0.00	57.63	79.65
id 7	39.27	70.75	34.09	4.01	0.11	1.02	0.03	72.50	35.96
id 8	51.34	94.13	54.80	3.75	0.13	1.03	0.03	96.20	54.42
id 9	53.38	82.00	39.64	3.85	0.07	1.02	0.02	83.24	40.66
id 10	62.39	83.47	42.44	4.78	0.13	1.02	0.03	85.22	43.85
id 11	63.53	156.86	70.40	3.74	0.07	1.02	0.02	159.35	71.24
id 12	80.51	112.50	50.13	4.54	0.42	1.08	0.10	121.08	52.13
id 13	82.10	112.79	50.86	5.04	0.12	1.01	0.02	114.14	51.56
id 14	96.46	132.22	57.92	4.77	0.33	1.05	0.07	139.80	63.35
id 15	99.43	131.55	64.96	6.26	0.18	1.04	0.03	137.50	69.28
id 16	100.30	135.82	65.88	5.23	0.22	1.04	0.04	140.47	67.69
id 17	110.07	169.57	95.73	5.51	0.21	1.02	0.04	173.77	96.81
id 18	112.48	89.10	43.63	5.41	0.24	1.03	0.05	92.84	49.45
id 19	116.03	225.75	86.98	5.75	1.17	1.12	0.23	238.82	76.00
id 20	121.71	149.00	78.38	5.78	0.66	1.06	0.12	158.21	80.13
id 21	123.61	143.64	72.49	5.63	0.40	1.08	0.08	153.99	74.72
id 22	129.80	177.86	88.34	5.37	0.13	1.04	0.03	184.47	90.36
id 23	132.19	144.70	60.47	4.22	0.11	1.03	0.03	150.44	65.22
id 24	137.89	179.29	51.81	5.87	0.07	1.02	0.01	183.45	52.22
id 25	139.84	167.18	79.31	6.14	0.26	1.03	0.04	173.43	84.22
id 26	141.44	191.44	113.77	5.32	0.25	1.06	0.05	205.74	125.67
id 27	143.38	213.00	78.53	6.35	1.46	1.13	0.26	230.62	70.43
id 28	155.45	216.78	93.05	6.35	0.34	1.07	0.06	230.08	93.44
id 30	157.53	200.75	86.22	5.32	1.10	1.11	0.23	213.30	82.85

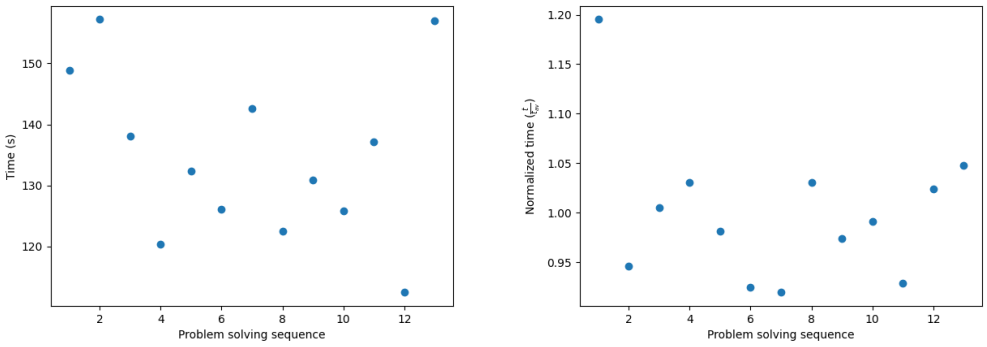
Table 3.18: Results of standard deviation analysis - summary

$R^2 = 0.785$	fit: $Y = 7.110X^{0.660}$				
	value	standard error	t-test	p-value	70% Confidence Interval
a	7.11	0.78	9.07	0.00	(6.30, 7.92)
b	0.66	0.02	27.93	0.00	(0.64, 0.69)
$R^2 = 0.819$	fit: $Y = 0.060X^{1.558} + 61.093$				
	value	standard error	t-test	p-value	70% Confidence Interval
a	0.06	0.04	1.48	0.14	(0.02, 0.10)
b	1.56	0.13	11.76	0.00	(1.42, 1.70)
d	61.09	4.14	14.77	0.00	(56.80, 65.39)

test subjects. This was observed during the tests as well, the subjects often took longer to process the task and asked clarifying questions during the first problem. As for the remaining of the problems, the data does not show high correlation. With the data points varying within ± 5% of the normalized time. The NC as a function of the solving sequence plot shown in Figure 3-26c does not have a clear trend or pattern. The fourth problem has a much higher normalized cost, this is due to 2 problems which had very high 17% and 26% above optimum that coincidentally where every other sequence had at most one. This is also observed in the standard deviation of each sequence, where sequence 4 has a standard deviation of 7.4% while every other sequence falls between 1.8% and 5.1%. This easily could be a sampling issue of an outlier and is not necessarily a pattern.

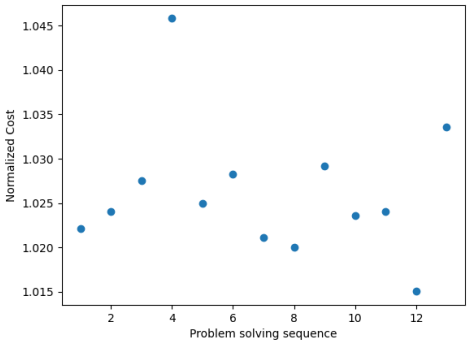
3.4 Video insights

Each test subject was free to choose whichever strategy they thought was best to find the optimum solutions. As previously mentioned, the TSP is a NP hard problem with $\frac{(n-1)!}{2}$ possible solutions. For a problem with 10 points there are more than 180 thousand distinct solutions. We, humans, are not capable of process and store this amount of information, much less in a short period of time with multiple problems being done in a short period of time (the average time of completion for all the problem was 2 minutes). Thus, humans tend to rely on heuristics and experience



(a)

(b)

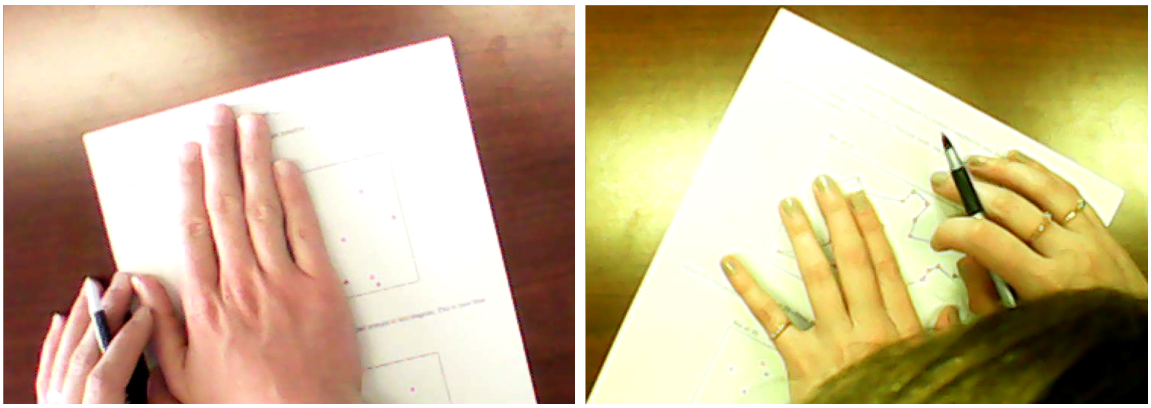


(c)

Figure 3-26: Sequence analysis:
(a) Problem Time as a function of problem sequence,
(b) Problem normalized time as a function of problem sequence,
(c) Problem normalized cost as a function of problem sequence

to solve the TSP. The TSP is a common problem in most people's life, for example: Bob is at home working and has to go in no particular order, to grocery shopping, to pick up his daughter at soccer practice, to go to the pharmacy and to drop his mom's dinner at her house before returning home. In this scenario, Bob will most likely to plan his route based on his previous experience, time traveled and distance between places. Throughout the years, we develop a heuristic on how to approach this problem. In this experiment it was observed in several occasions the same heuristics being applied. Figure 3-27a and 3-27b are two examples of different test subjects physically hiding part of the problem to reduce the problem space and be able to better grasp and search for a solution. They are abstracting the problem to reduce the number of possible solutions, and they are able to do this with a high level of

confidence due to the heuristic they created along their life. They are confident that searching for smaller sub-clusters and then linking the sub clusters will give them good enough results. This is also shown in Figure 3-28 where the test subject marked the sub-sections he was confident with a black sharpie on the draft session while he was looking for the best way to connect the other points with a pen where his path confidence was lower. These techniques, although do not guarantee the optimal solution, they reduce the search space significantly and give good enough results for most of the applications we see in an ordinary day to day life.



(a) (b)

Figure 3-27: Video excerpt - Human abstraction examples, (a) Test 15 (b) Test 25

Additionally, for the results of 15 test subjects, it was recorded the time spent in making the drafts as a percentage of the total time spent in the problem. On average the test subjects spent 72% of the time in the draft section of the problems. The amount of time spent in the draft portion does not have an obvious relationship to the NC or the problem complexity as shown in Figures 3-29a and 3-29b. These are interesting findings, it was expected the ratio to increase as complexity increases as there were more points to be searched and calculated. However, the results suggest that with the increase of problem complexity is linearly proportional to the time needed to copy the solution from the draft to the final answer plot. Moreover, the time ratio cannot be correlated to the normalized cost, one behavior that was observed in the experiment was that test subjects sometimes made changes from the draft directly in the final answers, so they were still actively solving the problem, even after they

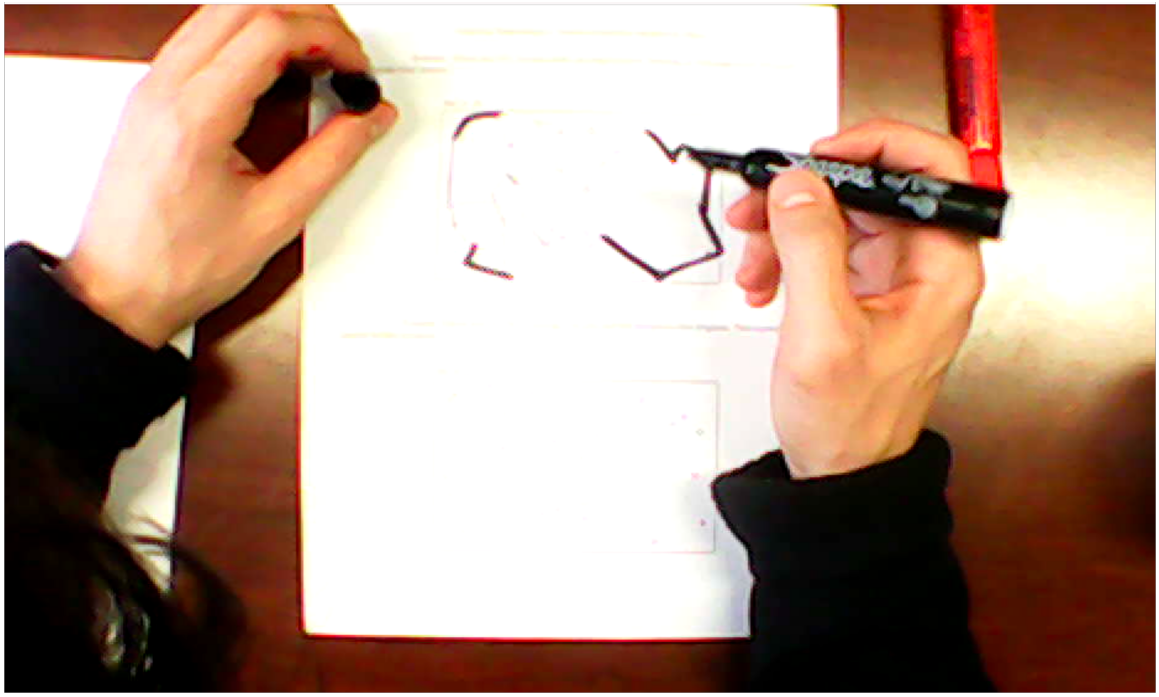


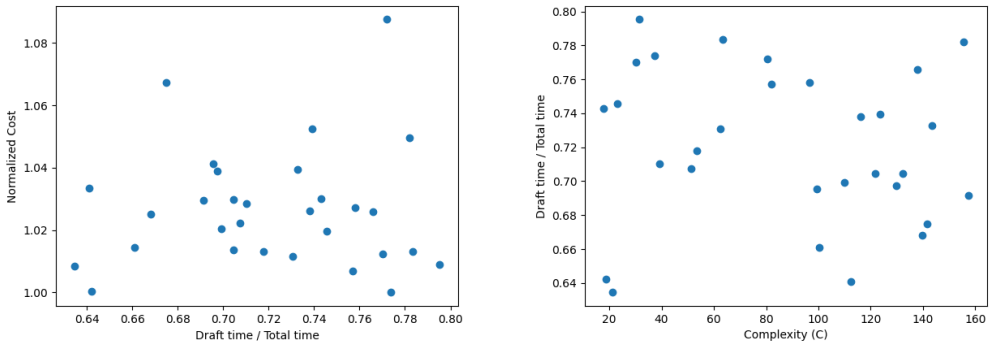
Figure 3-28: Video excerpt - Piecewise solving, Test 19

.

had stopped working on the draft solution.

3.5 Discussion

One of the biggest findings from this experiment is that for the TSP there is an overhead of cognitive load. The data suggests that regardless of the complexity of the problem there is an initial time where the test subjects take to understand it, after that, the complexity has a super-linear relationship with the time taken to solve the problem. This is different from the results previously by Sinha and de Weck[20], where their results suggests that as complexity approaches zero, the time to solve it also approaches zero. Additionally, in this experiment, sub-optimal solutions are allowed whereas in the previous experiment it was not. The sub-optimality has some influence in the differences in the results, however are very minor as shown in the previous section. Thus, there must be something else that explains the need of the inclusion of an offset for this problem. One hypothesis based on the video analysis and Figures 3-27 and 3-28 is that the test subjects involuntarily cluster points before starting to



(a) (b)

Figure 3-29: (a) Time ratio, Draft time over Total time versus Normalized cost, (b) Problem Complexity versus Time ratio, Draft time over Total time

solve the problems, Figure 3-30 was created to exemplify this phenomenon, starting at step (a) with the problem to be solved, at (b) the test subjects visually inspect and create mental clusters, where points are close together and seem to aggregate well, next at step (c) they solve a one way path trying to find a solution that could link the clusters as well as possible and finally at (d) they link the clusters. The argument here is that it takes a set amount of time to go from step (c) to step (d), the simplest of the problems do not have clusters and internal paths to be optimized. So the overhead is due to linking of clusters. The super-linearity relationship is a reflection of the increasingly difficulty to come up with meaningful clusters and to find the best path within each cluster. This hypothesis, also supports the super-linear growth as opposed to $O(\frac{(n-1)!}{2})$, the clustering reduces the number of possible solutions substantially. And humans are able to do this based on their heuristics developed throughout the years where achieving an the absolute optimal path is not necessary, specially because searching for the optimal solution may take longer than the benefit since often use it only a few times, as opposed to industry where the same path is taken thousand times a day which justifies the investment.

Another interesting finding is that the longer the time spent in the problem ,as expected, improves the results, but it also improves the variability of the outcome. So by spending longer in a problem, it is expected to get better and more consistent results. The test subjects that spent less time in the problems were more erratic,

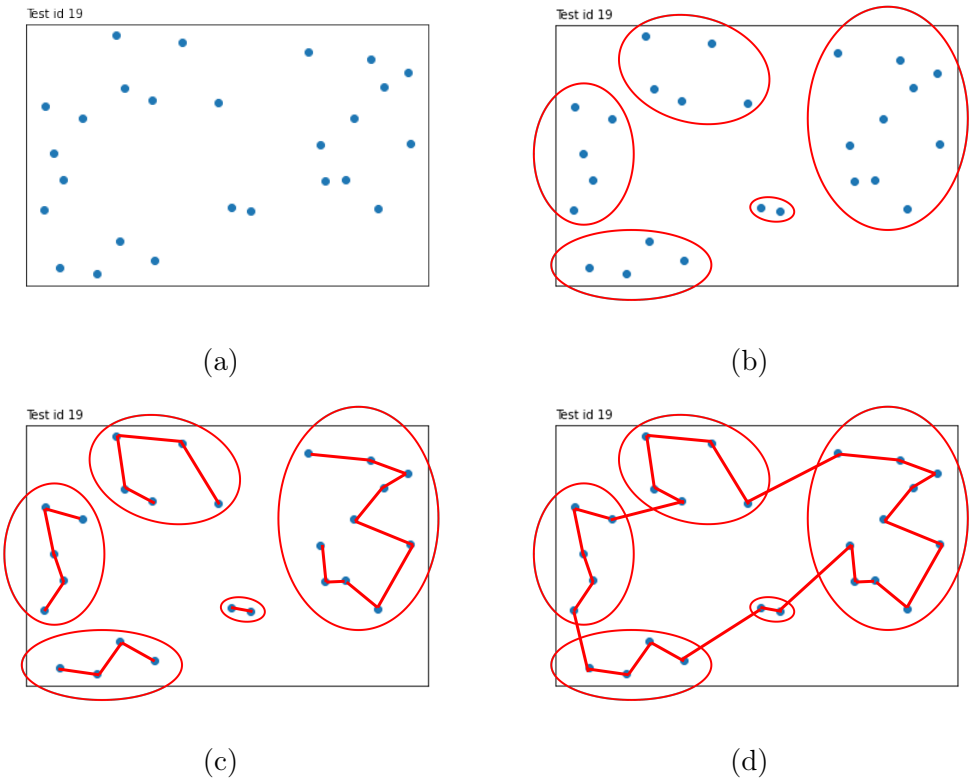


Figure 3-30: Hypothesis of solving steps for humans, exemplified in test id 19
(a) Unsolved problem
(b) Step 1 - Clustering points
(c) Step 2 - Solving one way within the clusters
(d) Step 4 - Connect clusters and solve problem

prone to large errors, reaching over 60% of the optimum values. Also, similar to the time to complete the problem, the variability of the time spent also increases super-linearly after an offset. This can be attributed to the different strategies chosen by the test subjects, the variability seems to vary proportionally based on the problem complexity. So the person who was solving the problems quickly, also solved the more complex problems faster, and the person who decided to take longer did it so in the same proportion.

3.6 Code base

All of the code used in this analysis is available at https://github.com/rbhopker/Thesis_complexity. It includes the methods to calculate complexity, complexity manipulation functions. the code to create the TSP problems, how the TSP problems were solved, all the analysis were performed, along with the code for a an implementation of the TSP experiment online using Streamlit library.

Chapter 4

System Decomposition

4.1 Introduction

To better understand systems, their behaviors, interaction and emergencies, it is a common practice to reduce the details and focus on the primary focus of the system. Setting the level of abstraction allows the architect to hide the details and while allowing us to reason about the function of the system[6]. Therefore setting the level of abstraction is crucial for the understanding of the system, however I argue that the complexity of the system is unaffected by it, because it is a inherited system property. As highlighted in the "Principle of Decomposition" in Crawley et al.[6] system decomposition is possibly the most important decision in system architecting to "minimize the apparent complexity of the system" and that it "is an active choice made by the architect". Figure 4-1 is a high level depiction of this rationale. The more detailed the level of abstraction the higher it is the perceived system complexity, in the other hand, the more coarse the level of abstraction is the smaller the system complexity is perceived. In contrast the system complexity remains unchanged. Ronnie Thebeau, in his thesis[21] tackled this problem by modifying an algorithm that creates decompositions given an architecture, which uses a randomized approach, and the objective function is based on the sum of interactions within a cluster and outside the cluster, and there is a penalty based on the cluster size and the total DSM size.

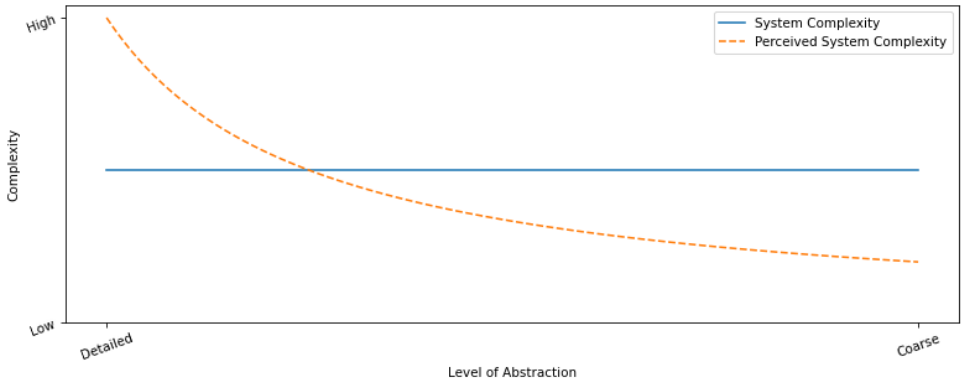


Figure 4-1: Relationship between Level of abstraction and System complexity

4.2 "Conservation of Complexity"

To exemplify how this can be applied to the metric we defined in Section 2.1, we will continue using the same example problem shown in Figure 2-1. This system is composed of five sub-systems, the Controller, Pump, Valve, Filter and Motor. Each sub-system have their own internal interfaces and components. Thus it is possible to calculate their complexities. In this example, it was given them a value of 5, 2, 1, 1 and 3 respectively, based on expertise of the paper authors'. The level of abstraction was arbitrarily chosen to exemplify the complexity metric. But what happens if the level of abstraction was increased even more? For example, it is possible to combine the sub-systems Valve and Filter to be a part of a single subsystem. If the complexity of the system remains constant what modifications Table 2.1 have to go through to depict the new system architecture level of abstraction. Table 4.1 shows the values that are unknown for this transformation of the system. Firstly, it is simple to calculate the C_1 for the VF sub-system. By calculating the complexity of the VF slice from Table 2.1,or in short $C_1^{VF} = C(\left[\begin{smallmatrix} 1 & 0.5 \\ 1.5 & 1 \end{smallmatrix}\right]) = 4$, and substituting in Table 4.1 yields to Table 4.2.

The new C_1 is easily calculated by adding 5,3 4 and 3 which is 14. From Equation 2.6 the total system complexity is unchanged at 25.44, which means that the C_2C_3 term must be equal to 11.44. To calculate the C_3 term all it is needed is the binary value of the adjacency matrix, whether there is an interface between the components

Table 4.1: DSM_C for problem in Figure 2-1 with increased level of abstraction

		C	P	VF	M
Controller	C	5	0	?	0.5
Pump	P	0	2	?	1.5
Valve and Filter	VF	?	?	?	?
Motor	M	1.5	0.5	?	3

Table 4.2: DSM_C for problem in Figure 2-1 with increased level of abstraction

		C	P	VF	M
Controller	C	5	0	?	0.5
Pump	P	0	2	?	1.5
Valve and Filter	VF	?	?	4	?
Motor	M	1.5	0.5	?	3

which is available to us from Table 2.4, if either the Valve or Filter have interface with the other sub-systems it means that the new abstraction will also have interfaces between them. The new matrix for C_3 calculation is shown in Table 4.3, which yields $C_3 = 1$. Thus, $C_2 = 11.44$. This C_2 term has two components, the known interface values from Table 2.4, and the unknown values from the change in abstraction level. From equation 2.4 the known values (C_2^k) add up to 4, so it is possible to determine that the unknown values (C_2^u) must be equal to 7.44. The issue is to determine how much it will be attributed for each of the interfaces identified while calculating C_3 as shown in Table 4.4. In this study, it was used the weighted average, based on the sum of the interfaces that got abstracted. For example, $C_{VF(C)} = \frac{2.5}{2.5+1.5+1.5+0.5} 7.44 = 3.1$. Doing this for all unknown values yields to Table 4.5.

Table 4.3: Modified Table 2.4 for new level of abstraction

		C	P	VF	M
Controller	C	0	0	1	1
Pump	P	0	0	1	1
Valve and Filter	VF	1	1	0	0
Motor	M	1	1	0	0

Table 4.4: C_2 first estimate for problem in Figure 2-1 with increased level of abstraction

		C	P	VF	M
Controller	C	0	0	1.5?	0.5
Pump	P	0	0	0.5?	1.5
Valve and Filter	VF	2.5?	1.5?	0	0
Motor	M	1.5	0.5	0	0

Table 4.5: DSM_C for problem in Figure 2-1 with increased level of abstraction

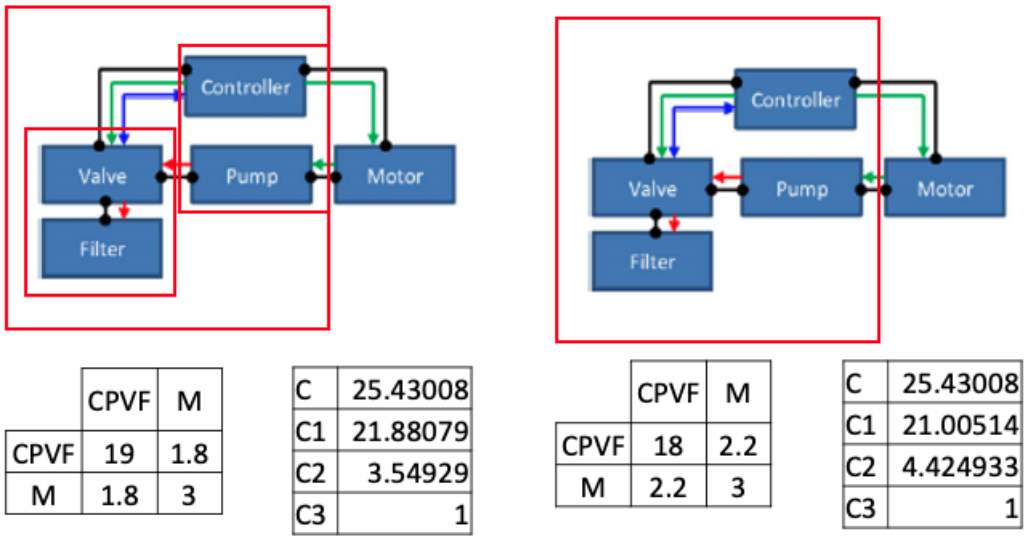
		C	P	VF	M
Controller	C	5	0	1.9	0.5
Pump	P	0	2	0.6	1.5
Valve and Filter	VF	3.1	1.9	4	0
Motor	M	1.5	0.5	0	3

4.2.1 Component and interface complexity

Unfortunately, it is possible to increase the level of abstraction, but it is not possible to reduce it with knowledge from the expanded system architecture. Neither is possible to reverse the calculation without using the original complexity DSM. Increasing the abstraction is possible because information gets merged, but to accurately unmerge it, it is necessary to know the system's architecture.

By using the calculation method described in this section, the breakdown and the level of abstraction matters. It is specially important in the distribution of the complexity. We can demonstrate it using the same sample system in Figure 2-1. By using different approaches (Figure 4-2) to achieve the apparently the same level of abstraction. As expected, both systems have the same overall complexity, however the system in Figure 4-2a, which is a level 2 decomposition has a higher C_1 value while having a lower C_2 value when comparing to the system in Figure 4-2b which is a level 1 decomposition. This analysis provides an interesting insight. In Figure 4-2a, because some interfaces are within the level 2 decomposition part of the interface complexity (C_2) is transferred to internal complexity (C_1). In the other hand, the system in Figure 4-2b retains more interface complexity because it does not have

further decomposition. Therefore, it is an architectural choice to trade complexity from components to the interfaces and vice-versa.



	CPVF	M
CPVF	19	1.8
M	1.8	3

C	25.43008
C1	21.88079
C2	3.54929
C3	1

(a)

	CPVF	M
CPVF	18	2.2
M	2.2	3

C	25.43008
C1	21.00514
C2	4.424933
C3	1

(b)

Figure 4-2: (a) Multiple abstraction level (b) Single abstraction level

4.3 Optimal level of abstraction to manage complexity

As mentioned in Chapter 3, the human effort has a super-linear relationship with system complexity in the approximated in the form $e = AC^b + d$ which means that not only decomposition level but also the architecture of the decomposition directly impact the effort to manage a system. This also means that there is at least one optimal decomposition that minimizes human effort. Equation 4.1, proposed a objective function to minimize the effort taken by humans to understand the system.

$$
\begin{aligned}
\min \quad & e_t \\
\text{s.t.:} \quad & \\
& e_t = e_i + \sum_{n=1}^{j} e_n \\
& e_n = AC_n^b \\
& C_i = C - \sum_{n=1}^{j} C_n \\
& e_i = AC_i^b + d
\end{aligned}
\tag{4.1}
$$

Substituting:

$$
e_t = AC_i^b + d + \sum_{n=1}^{j} AC_n^b
$$

Where e_t is the total human effort, e_i is the integration effort, e_n is the individual decomposition effort to understand(grasp, build) the system and C_n is the complexity for each decomposition n, C_i is the integration complexity and C is the total system complexity. b is super-linear, d is a possible overhead.

4.4 Example application

Using the air conditioner (AC) system shown in Figure 4-3, created by Prof. Steve Eppinger, modified from [15], also from his authorship, although it looks simple, there are over 27 million possible different ways of decompose this system. From the possible solutions some are better level of abstractions than other, while others may take a similar amount of effort to understand it. The system's connections along with Eppinger's favorite solution, based on his experience is shown in Figures 4-4a and 4-4b, respectively. The components are clustered into 3 different subsystems, the heater, the blower and the AC, which is much more intuitive for engineers and those familiar to AC systems. However, an expert is not always available to decompose a system, or it is too complicated to do it. Using the intuition described in Chapter 3 and in Equation 4.1, with the intent to minimize the the total effort to understand

the system, calculating all the possible abstractions it is possible to determine good system architectures.

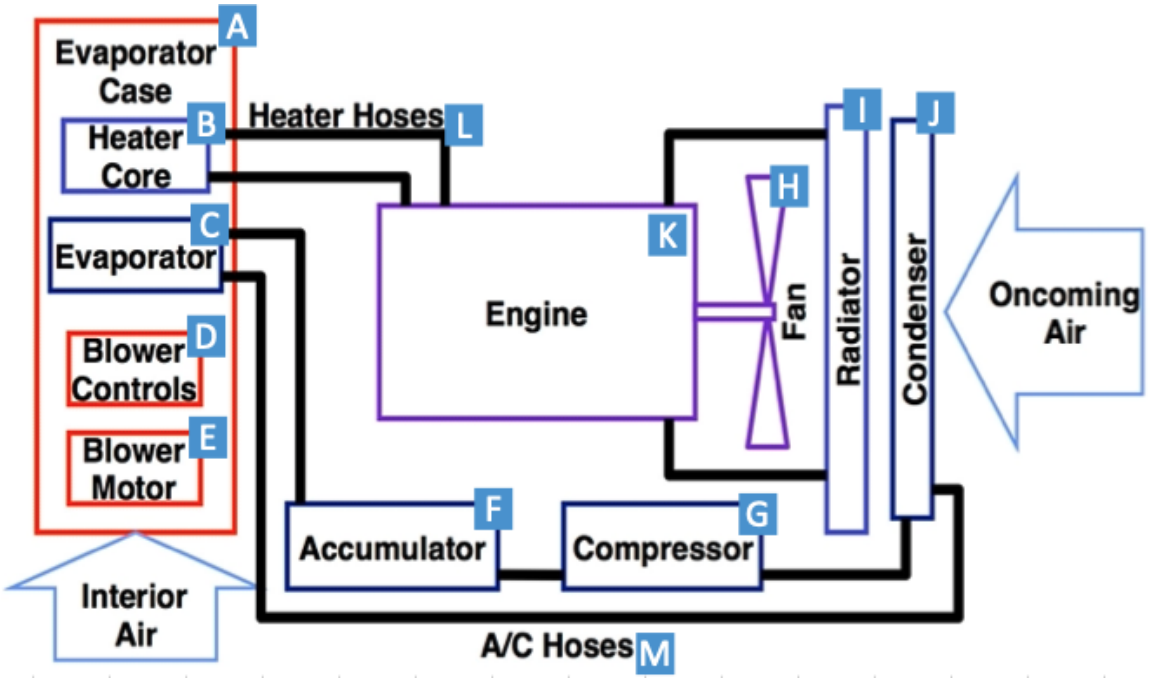


Figure 4-3: Air conditioning system

		A	B	C	D	E	F	G	H	I	J	K	L	M
Evaporator Case	A	A	x	x	x	x								
Heater Core	B	x	B	x								x	x	
Evaporator	C	x	x	C			x							x
Blower Controls	D	x			D	x								
Blower Motor	E	x			x	E								
Accumulator	F			x			F	x						x
Compressor	G						x	G			x	x		x
Fan	H								H		x	x		
Radiator	I									I	x	x	x	
Condenser	J							x	x	x	J			x
Engine	K		x					x	x	x		K	x	
Heater Hoses	L		x							x		x	L	
A/C Hoses	M			x			x	x			x			M

(a)

		F	G	J	M	C	A	D	E	B	L	K	H	I
Accumulator	F	F	x		x	x								
Compressor	G	x	G	x	x		A/C					x		
Condenser	J		x	J	x								x	x
A/C Hoses	M	x	x	x	M	x			Blower					
Evaporator	C	x			x	C	x			x				
Evaporator Case	A					x	A	x	x	x				
Blower Controls	D						x	D	x					
Blower Motor	E						x	x	E			Heat		
Heater Core	B					x	x			B	x	x		
Heater Hoses	L									x	L	x		x
Engine	K		x							x	x	K	x	x
Fan	H			x								x	H	
Radiator	I			x								x	x	I

(b)

Figure 4-4: (a) AC system marks (b) AC expert decomposition

For this system, it was calculated the effort for nearly 20 million possible different architectural decompositions, where the system was divided into 1, 2, 3, 4, 5 and 6 clusters. Every possible solution using these clusters was evaluated. Figure 4-5 summarizes all these possible options, the Y-axis is a normalized effort, where the denominator is the system without an abstraction, as seen in Figure 4-4a. Firstly,

by analysing the median values, it shows that by having a decomposition, in average the effort to understand the system decreases between 20% and 30%, regardless of how many clusters there are. Even the worse decompositions, have a benefit from 8% to 20%. The optimal decomposition, shown in Figure 4-6a, has a normalized effort of 52%, while the modified expert solution is shown in Figure 4-6b has a normalized effort of 53%. It was needed to modified the abstraction because the method presented in this chapter does not allow for overlapping abstractions. Interestingly, the expert solution is the best decomposition for decompositions with 3 clusters, and only 1% worse than the global optimal. This difference is minimal and the architect can argue for either option based on other information not included in the model. The only differences between the two decompositions were components H - Fan and J - Condenser, which combined into a new cluster, previously they were in the clusters Heater and AC respectively. Although there are a lot of assumptions boiled into this model, it clearly yielded a meaningful decomposition that could be used as a baseline. The denominator of the normalized cost is the effort calculated for the whole system as seen in Figure 4-4a. In this solution the exponent b used was equal to 1.4775 based on values found in Chaper 3 and in the study by Sinha and de Weck[20].

4.5 Abstraction of the TSP

Applying this method to the TSP is a non-trivial task, mainly because the clustering for the TSP should not be based on the complexity and connection if the points, but instead in their distances and relative distances between clusters, before the path between the points are drawn, as shown in Figure 3-30, step (a) to (b). It can be argued that for each problem, there is a cluster that reduces the human effort to solve the problem, and it is the architect's job to decipher and determine them. The different times and normalized costs seen in the experiment mentioned earlier could be partially attributed to this argument, where each test subject either chose a different abstraction or chose to not have abstractions at all. In the TSP, the tradeoff of introducing clusters is the risk of not finding the optimal path, however, good enough

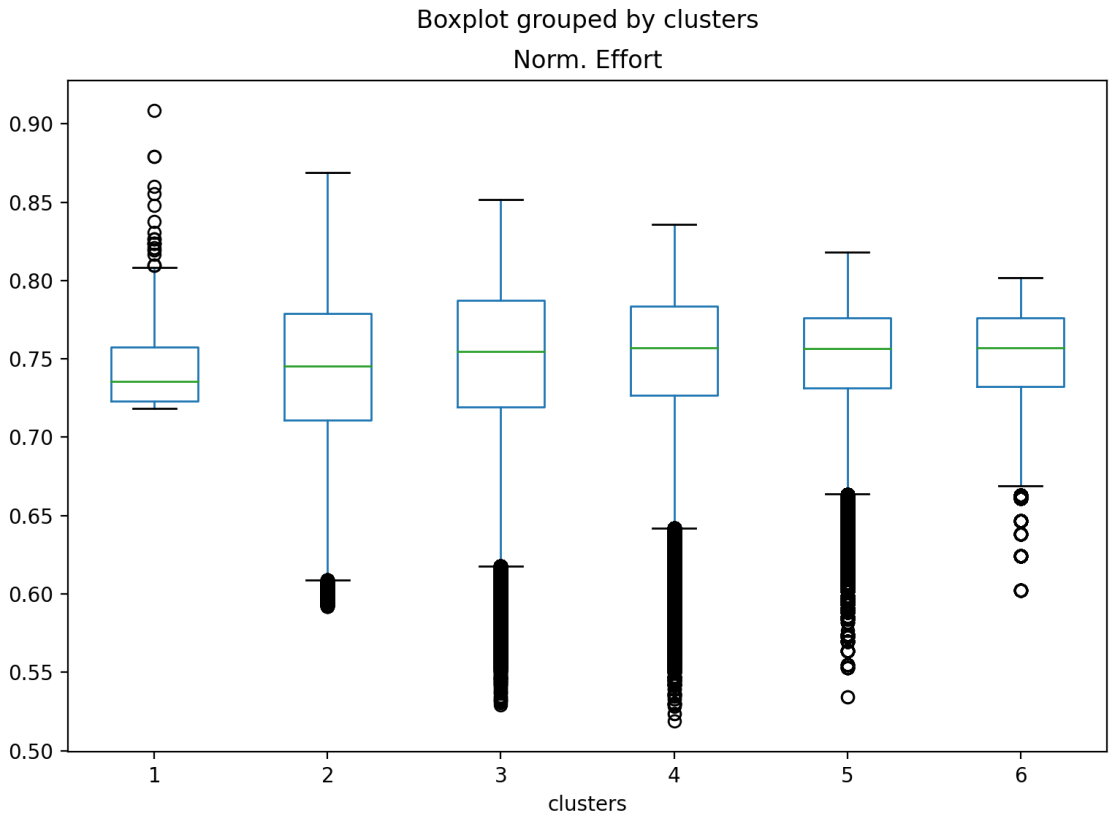
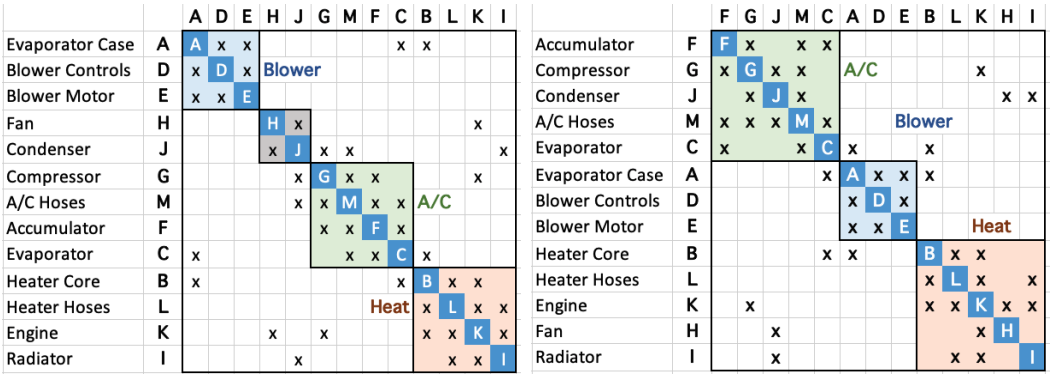


Figure 4-5: Evaluating system decomposition options, Normalized effort in Y-axis

results often arise in relatively much shorter time. To test this hypothesis, the same problem could be given to two groups, where in one set of problems the clusters are given, and for the other group is not. If proven true, the group with the clusters will perform better, faster and more consistently.

4.6 Discussion

In this Chapter it was introduced a method to reduce the human effort to comprehend a system based on the complexity metric and on system decomposition. The objective function is based on the effort it takes humans to comprehend and understand the system as a function of system and decomposition structural complexities. The system complexity metric still needs to be further validated, Chapter 3 is a step forward in this direction, while the system decomposition is an established method used widely in system design. The combination of the two in order to find optimal values is



(a) (b)

Figure 4-6: (a) Optimal level of the decomposition (b) AC modified expert decomposition

computational expensive but the usual method is hard and requires deep system and discipline knowledge from the system architect. Different algorithms can be implemented to reduce the search space, one example would be to use randomized optimization algorithms such as simulated annealing and genetic algorithm, however these do not guarantee optimality. Finding a good architecture decomposition has a compounding positive impact, it often shape the development of systems, from team formation and interactions, subsystem testing, verification and validation to supplier interaction.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Measuring complexity across different systems and disciplines is hard. Being able to do it is important for comparison and to take conclusions based on the system complexity, such as estimation of cost and time for completion. Using the structural complexity metric, this study reinforces the hypothesis that effort increases super-linearly with the respect of system complexity. In this study it was a found an exponent of 1.56 with a 70% confidence interval between 1.42 and 1.70. Although the exact number or range still has to be further developed, it was shown that even with great variation in the component and interface complexity the super-linearity was still true. However, unlikely the previous study, here it was observed an overhead needed to understand the system before the increase of complexity played a role in the increase in effort to solve the problems. Additionally, test subjects that spent more time than the average in the respective problems showed better quality answers but also much smaller variability, yielding better answers more consistently. Also, with the exception of the first problem, the data did not support any trend in learning effect or typical behavior while solving the problems. It is speculated that the human effort to solve the TSP is not only a function of interior points but also as a function of the distance and of how it is possible to cluster points.

It is the job of the architect to decompose the system. They should strive to

make it as simple as possible for the human mind. In this study, it was introduced a method to evaluate it based on human cognitive understanding. There are infinitely many possible level of abstractions for a system, and the architect using experience and the heuristic shown in this study should be able to create a decomposition that is close to optimum, while retaining the most important details.

5.2 Future Work

Further studies are still needed to validate the results shown in this paper. The TSP is an interesting problem, however humans have a good heuristic to solve it which led to answers close to optimum constantly. Removing some constraints and modifying some rules could yield interesting results to analyze the complexity of path finding problems. For example, it could be allowed multiple loops as if there were multiple salespersons starting in different cities that had to visit a set of cities as a group, or multiple salespersons starting at the same city. I believe these can increase the range of the NC by the test subjects. To validate the hypothesis generated in Section 3.5 and shown in Figure 3-30, it can be done an AB testing where group A receives the problems with the clusters given to them as in Figure 3-30b, while group B receives only the points as shown in Figure 3-30a. Other experiments, in different disciplines are required to further increase the confidence of the super-linearity relationship between complexity and effort. Additionally, more information is needed to determine what types of systems have an overhead for understanding as seen in this experiment. Moreover, a validation experiment in the search for optimality is system decomposition is needed to strengthen the argument posed in Chapter 4.

Appendix A

COUHES Approval

The Committee on the Use of Humans as Experimental Subjects (COUHES) approval for the experiments mentioned in this study is shown in the next page. The study was exempted under the Exempt Category 3, Benign Behavioral Intervention.

Massachusetts Institute of Technology
Committee on the Use of Humans as Experimental Subjects
77 Massachusetts Avenue Building E25-143B Cambridge, MA 02139-4307

Submission Date: Sep-13-2021

Title: E-3401, A Canonical Experiment on System Complexity Metric and Its Impact on Engineering Management
Principal Investigator: Hopker, Ricardo Bortot
Department: System Design and Management Program (SDM)
Faculty Sponsor: de Weck, Olivier L
Start Date: Sep-15-2021
End Date: Apr-15-2022

Determination: Exempt

Your research activities meet the criteria for exemption as defined by Federal regulation 45 CFR 46 under the following:

Exempt Category 3 - Benign Behavioral Intervention

Research involving benign behavioral interventions where the study activities are limited to adults only and disclosure of the subjects' responses outside the research could not reasonably place the subjects at risk for criminal or civil liability or be damaging to the subjects' financial standing, employability, educational advancement, or reputation. Research does not involve deception or participants prospectively agree to the deception. 45 CFR 46.104(d)(3)

All members of the research team must adhere to the policies as outlined in the Investigator Responsibilities for Exempt Research. If the facts surrounding your evaluation change, you are required to submit a new Exempt Evaluation. Research records may be audited at any time during the conduct of the study.

Bibliography

[1] Why has the cost of fixed-wing aircraft risen? a macroscopic examination of the trends in u.s. military aircraft costs over the past several decades. OCLC: ocn232257684.

[2] Haider A Abdulkarim and Ibrahim F Alshammari. Comparison of algorithms for solving traveling salesman problem. 4(6):5.

[3] Adaptive make radical advances in system design & manufacturing paul eremenko fmr. deputy director/acting director tactical technology. - ppt download.

[4] Robert Bitten, Charles Hunt, Debra Emmons, Robert Kellogg, Eric Mahr, and Sarah Lang. The effect of policy changes on NASA science mission cost & schedule growth. In *2018 IEEE Aerospace Conference*, pages 1–10. IEEE.

[5] D. Bonchev and N. Trinajstić. Information theory, distance matrix, and molecular branching. 67(10):4517–4533.

[6] E.F. Crawley, D. Selva, B. Cameron, and an O'Reilly Media Company Safari. *System Architecture: Strategy and Product Development for Complex Systems, First Edition*. Pearson.

[7] Donald Davendra. *Traveling Salesman Problem Theory and Applications.* OCLC: 1286422565.

[8] Paul T. Grogan and Olivier L. de Weck. Collaboration and complexity: an experiment on the effect of multi-actor coupled design. 27(3):221–235.

[9] J. N. Macgregor and T. Ormerod. Human performance on the traveling salesman problem. 58(4):527–539.

[10] Donald MacKenzie, Stephen Zoepf, and John Heywood. Determinants of US passenger car weight. 65(1):73.

[11] T.J. McCabe. A complexity measure. SE-2(4):308–320.

[12] Tom Mens. Research trends in structural software complexity. 08 2016.

[13] Austin Mohr. Quantum computing in complexity theory and theory of computation. page 6.

[14] Teguh Narwadi and Subiyanto. An application of traveling salesman problem using the improved genetic algorithm on android google maps. page 020035.

[15] Thomas U. Pimmler and Steven D. Eppinger. Integration analysis of product decompositions. In *6th International Conference on Design Theory and Methodology*, pages 343–351. American Society of Mechanical Engineers.

[16] Matthew W. Potts, Angus Johnson, and Seth Bullock. Evaluating the complexity of engineered systems: A framework informed by a user case study. 23(6):707–723. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sys.21558.

[17] Liam Sarsfield. *The cosmos on a shoestring: small spacecraft for space and earth science*. RAND, Critical Technologies Institute.

[18] Kaushik Sinha and Olivier de Weck. A network-based structural complexity metric for engineered complex systems. pages 426–430.

[19] Kaushik Sinha and Olivier de Weck. Structural complexity metric for engineered complex systems and its application. pages 181–192.

[20] Kaushik Sinha and Olivier L. de Weck. Empirical validation of structural complexity metric and complexity management for engineering systems. 19(3):193–206. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sys.21356.

[21] Ronnie E. Thebeau. Knowledge management of system interfaces and interactions from product development processes.

[22] author. Vega, Augusto. *Harsh computing in the space domain*. Morgan Kaufmann,, Amsterdam, [Netherlands] :, first edition. edition, 2017.

[23] Modest von Korff and Thomas Sander. Molecular complexity calculated by fractal dimension. 9(1):967.