Mixed Integer Programming Approaches for Group Decision Making

by

Hoi Cheong Iam

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2022

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Group decision making problems are everywhere in our day-to-day lives and have great influence on the daily operation of companies and institutions. With the recent advances in computational technology, it's not surprising that some companies would want to harvest that power to aid their decision-making procedures. Ethelo, the company that we partnered with in this project, developed an online platform that aids decision-making procedures by formulating the decision-making problem as a mixed integer nonlinear program (MINLP), providing feedback by solving the MINLP in real-time, and allowing the general public to contribute their opinions. Since an interactive component is involved, it is the goal of this thesis to attempt to reduce the solve time of their MINLP by applying tools from Operational Research. The main contribution in this thesis is threefold: first, we noticed that a big proportion of the MINLPs can be easily reposed as linear integer programs, and that a runtime reduction of at least 87.9% can be achieved by simply redirecting them to a linear solver. Second, we identified a knapsack-like polyhedral structure that, to the best of our knowledge, has not been studied before, and derived a sufficient condition to identify the cases for which all valid cuts can be derived by considering other knapsack or covering problems. Finally, for the more general case where the objective function is nonlinear and not continuous, we derived a few different formulations to get to different approximations of the nonlinear model, and tested all of the approximations computationally.

Acknowledgements

First, I would like to express my sincere gratitude to Prof. Ricardo Fukasawa and Prof. Joe Naoum-Sawaya, my two supervisors, for their patience, support, and guidance throughout the journey of my Master's degree. It would not have been possibe for me to make it this far without them. I would also like to thank Ethelo and its staffs, who made this thesis possible by granting us access to their business engine and also providing technical support in using it.

Last but not least, I would like to thank my thesis readers, Prof. Kanstantsin Pashkovich and Prof. Walaa Moursi, for their valuable comments that allowed for further improvement to this thesis.

Table of Contents

List of Figures vii

List of Tables viii

1 Introduction 1

2 Ethelo's Voting Engine 5

2.1 Terminology . . . 5

2.2 Ethelo's Survey and Vote Encoding . . . 6

2.3 The Scoring Functions . . . 7

2.3.1 The Ethelo Function . . . 8

2.3.2 Full Scoring Function . . . 10

2.4 Describing feasible solutions . . . 11

2.5 Full MINLP Formulation and Problem Sizes . . . 13

3 Single-Influence Cases 14

3.1 Computational Gains . . . 14

3.2 Partial Results . . . 17

3.2.1 Sufficient Conditions for general finite set X . . . 19

3.2.2 Implication on $X = \{0,1\}^n$ . . . 23

4 Multi-Influence Cases **26**

4.1 Reformulating Piecewise Linear Function 27

4.2 Approximating Bi-variate Functions 30

4.3 Best-fitting with Grid Triangulated Piecewise Linear Functions 33

4.3.1 Fixed Triangulation 34

4.3.2 Fixed Function Values 38

4.3.3 Dynamic Triangulation with Adjusted Function Values 46

4.4 Computational Results 50

4.4.1 Testing Environments 50

4.4.2 Machine Comparison 51

4.4.3 Best-fitting Ethelo function 51

4.4.4 Approximating multi-influence cases 54

4.4.5 Remark on Best Setting for Ethelo Function 56

5 Conclusion **65**

References **67**

APPENDICES **71**

A Relative Gap and CPU Runtime data for multi-influence tests **72**

List of Figures

1.1 Snapsort of survey 3

2.1 Unity function with $t = 1/3$ 9

2.2 Graph of Ethelo function with $t = 1/3, \Xi = 1/2$ (not to be confused with variable Σ) 10

4.1 A Grid Triangulation with $d_1 = 8, d_2 = 4$, where $\mathbb{T}$ is the collection of sets of extreme points of the triangles 30

4.2 Example for piecewise linear function constructed with **GTConstruct** . . 32

4.3 Diagonal (left) and skew-diagonal (right) split for a cell 39

List of Tables

3.1 Information on Problem Formulation for Single-influence cases 16
3.2 Average CPU Runtime Result for Single-Influence Cases 16

4.1 CPU Time comparison between server and PC environment 58
4.2 Wallclock Time and Gap for Approximating Ethelo Function 59
4.3 Table for Squared l_2 errors 60
4.4 Squared l_2-error reduction for approximating Ethelo Function 61
4.5 Average Percentage Relative Error over all instances 62
4.6 Worst Percentage Relative Error over all instances 63
4.7 Avg Runtime over all instances, Bonmin Avg = 4.60 64

A.1 Average Percent Rel. Error for buildbackbetter 73
A.2 Average Percent Rel. Error for carbon 74
A.3 Average Percent Rel. Error for citizen 75
A.4 Average Percent Rel. Error for granting 76
A.5 Average Percent Rel. Error for parks 77
A.6 Average Percent Rel. Error for stratford 78
A.7 Average CPU Time for buildbackbetter, Bonmin Avg = 15.68s 79
A.8 Average CPU Time for carbon, Bonmin Avg = 1.57s 80
A.9 Average CPU Time for citizen, Bonmin Avg = 7.09s 81
A.10 Average CPU Time for granting, Bonmin Avg = 0.60s 82
A.11 Average CPU Time for parks, Bonmin Avg = 0.34s 83
A.12 Average CPU Time for stratford, Bonmin Avg = 2.33s 84

Chapter 1

Introduction

Group decision making is practically everywhere in our day-to-day life. It affects everything from electing the next prime minister, to simply arranging a time slot for a regular Zoom meeting. With related works tracing back to Condorcet's Jury Theorem as early as 1785 [2], we can still see researchers investigating its related problems through the lenses of economics [25, 16], operational research (OR) [27, 23], psychology [6], and artificial intelligence [18, 9, 34] over the past century. In recent years, we are also seeing interests from companies who wanted to utilize the power of computers for making better decisions. For example, some companies are interested in using software to analyze the feedback they collected from a survey to gain better insight, and some take it one step further and look for software that also provide recommendation about the best options. In this work, we discuss the modern computer-based group decision-making platform developed by Ethelo, and present the work that we developed for improving that platform using OR tools.

Ethelo is a company that specializes on solving group-decision making problems that arise from budgeting, policy-making, and planing [12]. They have worked with over 200 institutions, including local governments, decentralized autonomous organizations (DAOs), and indigenous communities, to solve the group-decision making problems they face [12]. When a client approaches Ethelo with their group decision-making problem, which involves a list of available options to choose from and a set of practical constraints that needs to be satisfied, Ethelo creates an online survey for the problem dedicated to collecting public opinion on what should be done about it. On the survey, Ethelo provides necessary background of the problem to its participants, asks questions about basic democratic information for future analysis and, most importantly, allows participants to vote on which

of the provided options should be chosen when making the final decision [11]. While online surveys are commonly used for different purposes in modern days, there is one feature that sets Ethelo's surveys apart from others: Ethelo interactively reports the best solution, both during and after the participant vote.

To ensure that a participant's intention is correctly captured by the participant's vote, Ethelo interactively displays a closest "reasonable" solution - a solution that satisfies either all or a selected subset of provided constraints, depending on the client's preference - to the participant while he/she is voting. In the example provided in Fig. 1.1, the solution is displayed under the "My Ideal Budget" panel on the right. The displayed project's aim was to figure out a plan for reducing carbon footprint while staying under a given budget. As the participant votes, Ethelo finds a closest reasonable solution by solving a mixed integer non-linear program (MINLP) and then displays the returned solution, as well as a few of its characteristics (eg. "My adjusted tax bill"), on the panel. In this project, all constraints except the budget constraint were enforced by default when looking for the reasonable solution, and how well the budget constraint was satisfied was displayed on the ideal budget panel. In the cases where the budget constraint is violated by the selected plan (as shown in Fig. 1.1), the participant can either modify their input until all constraints are satisfied, or simply turn on the "Auto-Balance" feature and ask the system to show the closest feasible solution instead. That is, to ask the system to enforce all constraints in the project when looking for the solution to display. While these computed solutions are mostly for information and do not override the participants' votes, they allow the participants to modify their vote until the reported feasible solution looks acceptable, which then ensures that the participant's intention is correctly captured by the voting mechanism. At the end of the voting procedure, after all preferences are specified, the participant can re-weigh the importance of different sections of the survey before their vote is finalized, which then changes how Ethelo makes their trade-offs in case the voted solution is not feasible to the problem. Participants are free to modify any part of their votes before it is finalized and submitted.

After the participant submits his/her vote, Ethelo reports to the participant a short list of best solutions based on the votes collected thus far, and also where the other participants stand on supporting or rejecting each of the solutions. This short list allows the participants to get a preview of what the final decision may look like, and also get an idea of where they are standing among the other participants. This short list is computed by formulating and solving yet another MINLP that has similar constraints as the ones in the previous paragraph, except this time it considers all collected votes instead of just

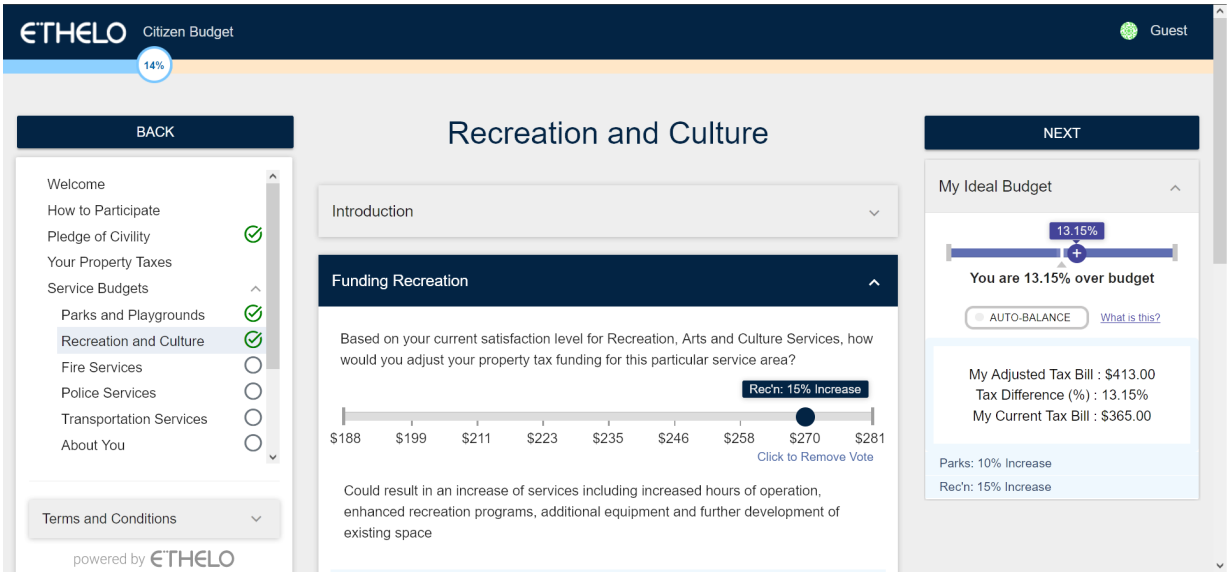

Figure 1.1: Snapsort of survey

one. To distinguish between the two, we call the MINLP concerning all collected votes the "multi-influence case" of the problem, and refer to the MINLP concerning only the voting participant as the "single-influence case" of the problem. While all the interactive components look engaging to the participants, the fact that a MINLP needs to be solved in real-time poses a challenge to Ethelo.

As popularly known, MINLPs are NP-hard to solve in general [24], and MINLP solvers are generally more expensive computationally when compared to mixed-integer linear program (MILP) solvers. In the cases where the MINLP problems are not solved fast enough, participants will have to wait for the required solution to be reported on the survey. This means that from the participants' point of view, there would be a noticeable lag on the supposedly interactive components, or even worse, that some components will not be responding. This is not desirable from Ethelo's point of view. In this research, we attempted to improve the runtime for solving the MINLPs that arises from both the single-influence and multi-influence case. Our contributions are 1) we identified that most of the single-influence cases can be re-posed as mixed-integer linear programs (MILP), which led to significant runtime improvement after redirecting the problems to a MILP solver; and 2) we studied different approximations the MINLP in multi-influence cases, and showed that they can potentially lead to observable runtime improvements of varying degree while

retaining a reasonably good solution quality.

The remaining of this thesis is organized as follows: in chapter 2, we introduce Ethelo's voting engine in further detail, and present how the MINLPs in both the single-influence and multi-influence cases are formulated. In chapter 3, we go over the computational results in single-influence cases obtained by switching to a MILP solver, and also present several theoretical results about a sub-case of the single-influence MINLPs. Then, in chapter 4, we present the reformulation techniques we used for approximating the multi-influence cases as well as its computational results. Finally, we give our conclusions in chapter 5.

Chapter 2

Ethelo's Voting Engine

In this chapter, we will introduce how Ethelo collects and encodes the votes collected from participants, and also how the MINLPs for both single- and multi-influence cases are formulated.

2.1 Terminology

We start our discussion by defining a few terms that will be used throughout this thesis. For our purpose, a "group decision-making problem" is a problem where we are provided with 1) a set of options $\mathbb{O} = [n] := \{1, 2, ..., n\}$, 2) a non-empty set $X \subseteq 2^{\mathbb{O}}$ of feasible solutions, and 3) a real-valued function $f : 2^{\mathbb{O}} \to \mathbb{R}$ that scores each solution in $2^{\mathbb{O}}$ based on how well they aligns with the collected votes. The goal is to find a feasible solution $x^* \in X$ such that the score $f(x^*)$ is maximized.

The scoring function f was designed by Ethelo, which we will describe more formally in section 2.3 and is a part of the US-patent owned by Ethelo [29]. We refer to any subset of the set $\mathbb{O}$ of available options interchangeably as a "solution" or "scenario". The set $2^{\mathbb{O}}$ of all subsets of available options $\mathbb{O}$ is also the set of all possible solutions. A solution $s \in 2^{\mathbb{O}}$ will be encoded as an indicator vector $x \in \{0, 1\}^{\mathbb{O}} = \{0, 1\}^n$, with $x_o = 1$ indicating that option o is contained in solution s and $x_o = 0$ otherwise. Therefore, we will also consider $\{0, 1\}^n$ as the set of all possible solutions. Whether we are considering the solutions as subset of $\mathbb{O}$ or binary vector in $\{0, 1\}^n$ in any part of this thesis will be clear from context.

We refer to the company or institution that approaches Ethelo with a group decision-making problem as a "client", and the individuals that provided inputs to Ethelo's online surveys as "participants". We use "Ethelo's engine" to refer to the code that Ethelo uses for formulating the MINLPs and passing them to the MINLP solvers, and use "project" to refer to the instances of group-decision making problems that Ethelo has solved.

2.2 Ethelo's Survey and Vote Encoding

To present the set $\mathbb{O}$ of available options for the participants to vote, Ethelo organizes the set of options into widgets, like slide bars and drop down lists, depending on their semantic meanings. For example, contradicting options like "build a green gate" versus "build a yellow gate" can be organized as a drop-down list so that participants can only choose one of them when voting, and options regarding a continuous quantity, like "Funding Recreation" shown in Fig. 1.1, can be organized as a slide bar. The widgets are set up in a way that every option in $\mathbb{O}$ is contained in precisely one widget. When presenting the widgets on the online survey, the widgets are further grouped into sections so that questions regarding similar issues can be presented together. At the end of the survey, the participants can assign a weight in $[1, 100]$ to each of the sections, with heavier weight means that the section is more important to them, and hence should avoid deviating from the selected options in that section when making trade-offs. Note that before a participant assigns a weight to each of the sections, all sections have an equal weight of 50. Also, participants can still change their vote in previous sections after assigning weights to sections.

A "vote" of a participant can be regarded as a partition (O_1, O_2, O_3) of the set $\mathbb{O}$ and a weight vector $w \in [1, 100]^{\mathbb{O}}$, where O_1 are the options that are "selected" by the participant, O_2 are the ones that are "not selected", and O_3 are the options that are "not voted". For each $o \in \mathbb{O}$, w_o is the weight the participant assigned to the section that contains the widget for o. An option o is called "voted" if the participant has clicked on, ie. selected an option on, the widget that contains o. In practice, the votes are encoded as a **preference matrix**, which we define as follows:

Definition 2.2.1. *When given a project, the preference vector of a participant is a vector $p \in [-1, 1]^n$ with $\|p\|_1 = 1$ that encodes the preference of the voter, with higher value of p_o indicating that the option $o \in \mathbb{O}$ is more preferred. When there are N participants, each with preference vector $p_1, p_2, ..., p_N \in \mathbb{R}^n$, we say that the influence matrix of the project is*

a matrix $P \in \mathbb{R}^{N \times n}$ *given by:*

$$P = [p_1, p_2, ..., p_N]^T$$

The complete procedure for creating a preference vector for a participant can be found in Ethelo's White Paper [28]. For our purpose, we can summarize it as follows:

1. Before the voter starts voting, define a vector $v = \epsilon \mathbf{1} \in \mathbb{R}^n$, where $\mathbf{1}$ denotes an all-one vector and $\epsilon \in \mathbb{R}$ is a small constant.

2. When a participant votes on an option $o \in \mathbb{O}$, update the value of v_o to 1; and for other options o' in the same widget as o, update $v_{o'}$ to a value in $[-1, 1]$ in a way such that options with closer semantic meaning to o has a value closer to 1. The precise values are not important for the purpose of this thesis, but interested readers can refer to their White Paper [28] for the details of how the values are assigned.

3. When a preference vector for a participant is required, we computed its value by

$$p = \frac{v \odot w}{\|v \odot w\|_1}$$

   where $\odot$ refers to the entry-wise multiplication.

The influence matrix P and the preference vectors it contains are only used for identifying whether one solution is more preferred over another, and the constraints are independent from the participants' votes. In other words, whether or not a solution is feasible is independent from the participants' votes.

2.3 The Scoring Functions

Here we define how the solutions $x \in \{0, 1\}^n$ are scored. There are two cases that need to be considered: the single-influence case, where there is only one vote (or, equivalently, one participant) to consider, and the multi-influence case, where we consider multiple votes.

In the single-influence case, if $p \in \mathbb{R}^n$ is the preference vector of the participant, a solution $x \in \{0,1\}^n$ can be scored simply by $p^T x$, which we refer to as the "satisfaction score" of the participant toward x. In the multi-influence, however, it is not desirable to simply take the sum or average of the participants' satisfaction scores as our scoring function, as Ethelo also want to avoid the controversial solutions where the participants' opinion are polarized [11]. To fix this issue, Ethelo designed a function, named the "Ethelo function", for scoring the multi-influence cases.

2.3.1 The Ethelo Function

The Ethelo function $\eth : [-1,1] \times [0,1] \to \mathbb{R}$ is designed to be a function of the average and the (sample) variance of the participants' satisfaction scores. Note that since the preference vector p satisfies $\|p\|_1 = 1$ by definition, for any solution $x \in \{0,1\}^n$ we have:

$$|p^T x| \leq \sum_{i=1}^{n} |p_i||x_i| \leq \sum_{i=1}^{n} |p_i| = \|p\|_1 = 1$$

In other words, satisfaction scores of any participant always falls within the interval $[-1,1]$. Hence, the average of satisfaction scores will fall in $[-1,1]$, and the variance will fall in the range $[0,1]$. Note that the above inequalities also applies to any $x \in [0,1]^n$.

Now, let's define the Ethelo function:

Definition 2.3.1. *The **Ethelo function** $\eth : [-1,1] \times [0,1] \to \mathbb{R}$ is given by:*

$$\eth(\mu, \Sigma) = \begin{cases} 0 & , \text{ if } \mu = 0 \\ \mu + \Xi \cdot U(\Sigma)\mu & , \text{ if } \Sigma > t \\ \mu & , \text{ if } \Sigma = t \\ \mu + \Xi \cdot U(\Sigma)(1-\mu) & , \text{ if } \Sigma < t, \mu > 0 \\ \mu + \Xi \cdot U(\Sigma)(-1-\mu) & , \text{ if } \Sigma < t, \mu < 0 \end{cases}$$

where $t, \Xi \in [0,1]$ are fixed parameters, and $U : [0,1] \to [-1,1]$ is called the "unity function", defined by:

$$U(\Sigma) = \begin{cases} \frac{t-\Sigma}{t} & , \text{ if } \Sigma \leq t \\ \frac{t-\Sigma}{1-t} & , \text{ if } \Sigma > t \end{cases}$$

Specially, when $\mu(x), \Sigma(x)$ are the average and variance of the participants' satisfaction scores toward a solution $x \in \{0,1\}^n$, we say that $\eth(\mu(x), \Sigma(x))$ is the "Ethelo Score" of x.

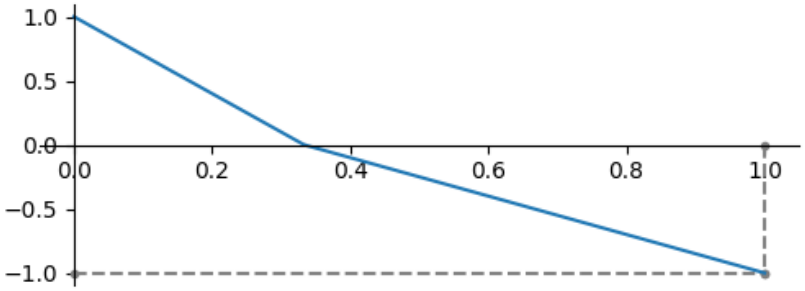

Figure 2.1: Unity function with $t = 1/3$

We included a plot of the Ethelo function in Fig 2.2 for reference. The unity function U is designed for measuring how "united" are the participants about their opinions on a particular solution. $U(\Sigma)$ is a decreasing function on $[0, 1]$ and satisfies $U(0) = 1$, $U(t) = 0$, and $U(1) = -1$. A graph of the unity function is shown in Fig 2.1. The Ethelo function is designed to satisfy the following properties:

1. For any fixed $\Sigma_0 \in [0, 1]$, $\eth(\cdot, \Sigma_0)$ is an increasing function on $[-1, 1]$.
2. For $\mu_0 > 0$, $\eth(\mu_0, \Sigma)$ is a decreasing function in Σ.
3. For $\mu_0 < 0$, $\eth(\mu_0, \Sigma)$ is increasing in Σ.
4. $\eth(\mu, \Sigma)$ always have the same sign (positive / negative) as μ.

Intuitively, solutions with positive average satisfaction score can be understood as being "supported" by general public, and the variance Σ is a measure of how divided the voters are about their opinions. Since we are looking for the solution with highest Ethelo score when solving our group decision-making problem, the above presuppositions can be interpreted as follows:

- When the voters' opinions are equally divided between two solutions, we prefer the ones that has a higher average satisfaction score.
- Of the solutions that have the same average in opinion scores and are supported by general public, we prefer the solutions where the participants are less divided about their opinions (ie. more united).

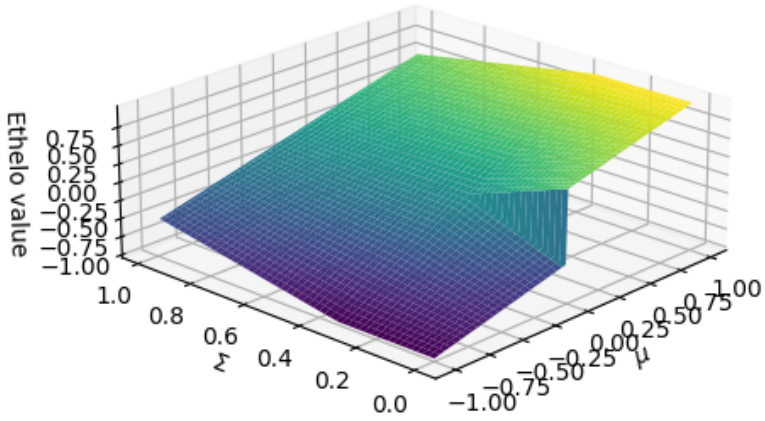

Figure 2.2: Graph of Ethelo function with $t = 1/3, \Xi = 1/2$ (not to be confused with variable Σ)

- In the case where no feasible solution is supported by the general public, of the feasible solutions that have the same (negative) average score, we prefer the ones where the public opinions are more diverse, in hope that a larger population will support the decision.
- In all cases, we prefer solutions that are supported by general public over those that are not supported.

The parameter Ξ is used as a weight that governs the variance-average trade-off, and t is referred to as the "tipping point" for deciding whether the general public is united about their opinions. In practice, the tipping point is usually set to $t = 1/3$, which is the variance of uniform distribution over interval $[-1, 1]$, and Ξ is usually set to $1/2$. Also, for any $\Xi > 0$, we note that the Ethelo function is discontinuous on $(\mu, \Sigma) \in \{0\} \times [0, t]$ but continuous everywhere else.

2.3.2 Full Scoring Function

To summarize the above, we note that when given an influence matrix $P \in \mathbb{R}^{N \times n}$, where N is the number of participants, the average of satisfaction scores toward a solution x can

be computed as:

$$\mu(x) = \frac{1}{N}\mathbf{1}^T Px$$

and the variance of satisfaction scores can be computed by:

$$\begin{aligned}\Sigma(x) &= Var[p^T x] \\ &= E[(p^T x)^2] - (E[p^T x])^2 \\ &= \frac{1}{N}x^T P^T Px - \frac{1}{N^2}x^T P^T \mathbf{1}\mathbf{1}^T Px \\ &= \frac{1}{N^2}x^T P(NI - \mathbf{1}\mathbf{1}^T)Px\end{aligned}$$

where $\mathbf{1}$ is an all-one vector, and I denotes the identity matrix of appropriate dimension, in this case $N \times N$. Thus, given influence matrix P, the complete scoring function $f : [0,1]^n \to \mathbb{R}$ can be given by:

$$f(x) = \begin{cases} p^T x & \text{, if } P \text{ is a row matrix } P = p^T \\ \eth(\mu(x), \Sigma(x)) & \text{, otherwise} \end{cases}$$

which means that in single-influence cases, the scoring function $f(x)$ is linear in x.

2.4 Describing feasible solutions

In practice, the set of feasible solutions $X \subseteq \{0,1\}^n$ will be given implicitly by a set of constraints:

$$X = \left\{ x \in \{0,1\}^n : \begin{array}{l} l^1 \leq x_{AB} \cdot g^1(x) \leq u^1 \\ l^2 \leq g^2(x) \leq u^2 \end{array} \right\}$$

where x_{AB} is a binary variable corresponding to the auto-balance option, $g^1 : \mathbb{R}^{\mathbb{O}\setminus\{AB\}} \to \mathbb{R}^{m_1}$, where $g^2 : \mathbb{R}^{\mathbb{O}} \to \mathbb{R}^{m_2}$, all entries on l_i^1, l_i^2 are in $\mathbb{R} \cup \{-\infty\}$, entries of u^1, u^2 are in $\mathbb{R} \cup \{+\infty\}$, and the constraints $l^1 \leq x_{AB} \cdot g^1(x) \leq u^1$ are the ones that will be ignored unless auto-balance is selected. $l^1 \leq x_{AB} \cdot g^1(x) \leq u^1$ and $l^2 \leq g^2(x) \leq u^2$ are collectively called the "constraints" of the project. The variable x_{AB} was introduced as above to allow for better interaction with the parts of Ethelo's engine that are outside the MINLP solver. There are cases where the variable x_{AB} is used in $g^2(x)$, which we will mention later in this section. We also note that depending on the values of l^1, u^1, there may not exist feasible solutions with $x_{AB} = 0$.

While there are no assumptions on the function g^1, g^2, when setting up the constraints, each entry g_i of both g^1, g^2 will be expressed as a compositions of the arithmetic operators $(+, -, \times, \div)$, absolute value operator $|\cdot|$, and square root operator $\sqrt{\cdot}$ in variables $x_i : i \in [n]$ due to how the online platform is set up. By looking at the previous projects that were solved by Ethelo, we observed that in all of the provided cases, each constraint falls into one of the following 5 categories:

1. Two-sided Knapsack constraints: $l \leq a^T x \leq u$ where $a \in \mathbb{R}^n_+$, $l, u \in \mathbb{R}$.
2. XOR constraints (also known as multiple-choice constraints [22, 4]): $\sum_{i \in S} x_i = 1$ for some $S \subseteq \mathbb{O}$.
3. Exclusion constraint: $\|x - \bar{x}\|_1 \geq 1$ for some $\bar{x} \in \{0, 1\}^n$.
4. Quotient constraints: $l \leq g_i(x) \leq u$ with $l, u \in \mathbb{R}$ and $g_i(x) = a(x)/b(x)$, where $a, b : \mathbb{R}^n \to \mathbb{R}$ are affine functions and $b(x) > 0$ for all $x \in [0, 1]^n$.
5. General Quadratic constraints: $l \leq g_i(x) \leq u$ with $l, u \in \mathbb{R}$ and $g_i(x) = x^T A x + b^T x + c$, where $A \in \mathbb{R}^{n \times n}, b \in \mathbb{R}^n, c \in \mathbb{R}$.

Also, the variable x_{AB} do not appear in any constraint in g^2 other than in exclusion constraints, which is used for excluding a chosen solution from the feasible region of the MINLP.

From the above description, readers may have noticed that all of the aforementioned nonlinear constraints, namely the quotient, exclusion, and general quadratic constraints, can be easily replaced with linear constraints when the variables $x \in \{0, 1\}^n$ are binary. The quotient constraints $l \leq a(x)/b(x) \leq u$ can be equivalently replaced by two linear constraints $a(x) - l \cdot b(x) \geq 0$ and $a(x) - u \cdot b(x) \leq 0$. For exclusion constraints, since $\bar{x} \in \{0, 1\}^n$ and variables $x \in \{0, 1\}^n$, the sign of $x_i - \bar{x}_i$ can be decided solely by the sign of $\bar{x}_i$, and so $\|x - \bar{x}\|_1 = \sum_{i=1}^n |x_i - \bar{x}_i|$ can be rewritten as a linear expression in x. Lastly, since all variables are binary, the general quadratic constraints can be easily linearized using any reformulation-linearization techniques (RLT) available in a rich body of literature ([1, 30, 33], for example). Further, in the cases where these constraints are multiplied by x_{AB}, we can simply solve the problem twice - once with $x_{AB} = 1$ and the other with $x_{AB} = 0$ - to avoid the non-linear terms that were introduced by multiplying x_{AB}. These reformulations were originally not performed in Ethelo's engine, and as we will see in section 3.1, a significant speedup can be obtained by simply carrying out the above reformulation and redirecting the originally-MINLP to a MILP solver.

2.5 Full MINLP Formulation and Problem Sizes

To summarize all the above sections, the full formulation of MINLP that Ethelo solves can be given as follows:

$$\begin{array}{lll} \max & f(x) & \\ s.t. & l_i \leq x_{AB} \cdot g_i^1(x) \leq u_i & \forall i \in \{1, 2, ..., m_1\} \\ & l_i \leq g_i^2(x) \leq u_i & \forall i \in \{1, 2, ..., m_2\} \\ & \|x\|_1 \geq 1 & \\ & x \in \{0, 1\}^n & \end{array} \tag{EthP}$$

where $n = |\mathbb{O}|$ is the number of available options, m is the number of constraints in the project. $f : [0, 1]^n \to \mathbb{R}$ is given by:

$$f(x) = \begin{cases} p^T x & \text{, if } P \text{ is a row matrix } P = p^T \\ \eth(\mu(x), \Sigma(x)) & \text{, otherwise} \end{cases}$$

as in section 2.3, $g^j : \mathbb{R}^n \to \mathbb{R}^{m_j}$, $l_i^j \in \mathbb{R} \cup \{-\infty\}$, $u_i^j \in \mathbb{R} \cup \{+\infty\}$ for all $i \in \{1, 2, ..., m_j\}$ and $j \in \{1, 2\}$. The $\|x\|_1 \geq 1$ constraint was added to exclude the solution $x = 0$. Semantically, the solution $x = 0$ means that "no action needs to be taken for the proposed problem", which is not very helpful for Ethelo's clients.

For most of the problems that Ethelo solved, we have $n \leq 100$ and $m \leq 30$, but there are larger problems with more constraints and $n \approxeq 250$. Also, when multiple best solutions are required in the multi-influence case, Ethelo obtains the sub-sequence solutions by solving (EthP) repeatedly, while adding a new constraint $\|x - \bar{x}\|_1 \geq 1$ to the program (more specifically, to constraint $l^2 \leq g^2(x) \leq u^2$) after each solve, with $\bar{x}$ being the just-obtained optimal solution. These cases are not explicitly considered in this thesis, and are also not considered in the computational results. However, we note that the exclusion constraints can be re-posed as linear constraints, so ignoring the added exclusion constraints does not affect the validity of our results.

Chapter 3

Single-Influence Cases

In this chapter, we will present the work we have done regarding the single-influence cases. These are the cases that arise when Ethelo is computing a closest feasible solution for their participants, which means that there is only one participant in consideration, and that only one optimal solution is needed. In these cases, the program ($EthP$) can be simplified as:

$$\begin{array}{lll} \max & p^T x & \\ s.t. & l_i \leq x_{AB} \cdot g_i^1(x) \leq u_i & \forall i \in \{1, 2, ..., m_1\} \\ & l_i \leq g_i^2(x) \leq u_i & \forall i \in \{1, 2, ..., m_2\} \\ & \|x\|_1 \geq 1 & \\ & x \in \{0, 1\}^n & \end{array} \tag{EthP-SI}$$

where p is the preference vector of the participant. As mentioned in section 2.5, while the constraints in (EthP-SI) can often be reposed as linear constraints, such reformulation procedures were not in place in Ethelo's engine before this project. In section 3.1, we will introduce the procedures we implemented in Ethelo's engine for reformulating the constraints, as well as the computational speedup by doing so. Then, in section 3.2, we present a few theoretical results regarding cut generation in a special case of ($EthP - SI$).

3.1 Computational Gains

Since all constraints in the past projects provided by Ethelo falls into one of the 5 categories of constraints mentioned in section 2.4, which can all be, but had not been, reposed as linear constraints, it is natural for us to want to redirect the cases that can be

reposed as MILP to a MILP solver. To do so, we implemented two procedures in Ethelo's engine. The first one, named **simple-reform**, proceed as follows when received a program (EthP) that does not contain auto-balance option x_{AB}:

1. If there is at least one constraint in (EthP) that does not fall into type 1-5 as described in section 2.4, send (EthP) to BONMIN and return the result.

2. Otherwise, for quotient constraints $l \leq a(x)/b(x) \leq u$, we replace it with two constraints $a(x) - l \cdot b(x) \geq 0$ and $a(x) - u \cdot b(x) \leq 0$. For exclusion constraints $\|x - \bar{x}\|_1 \geq 1$, replace it with a linear constraint $\sum_{i=1}^{n} (-1)^{\bar{x}_i}(x - \bar{x}_i) \geq 1$.

3. If there are general quadratic constraints, replace them with its McCormick's relaxation [1, 26]. That is, replace all terms x_i^2 with x_i (recall all x_i are binary) and each quadratic term $x_i x_j$, where $i < j$, with a new binary variable $y_{i,j}$. Then, for each new variable $y_{i,j}$, append constraints $y_{i,j} \leq x_i$, $y_{i,j} \leq x_j$, and $y_{i,j} \geq x_i + x_j - 1$ to the program.

4. Pass the resulting program to CBC, a MILP solver, and return the result.

We have argued in section 2.4 that the above procedure reformulates (EthP) to a MILP when x_{AB} is not used. CBC was chosen as the MILP solver instead of industrial solvers like CPLEX due to Ethelo's preference of open-source solvers, and also because CBC was already used in BONMIN as one of the sub-procedures. It is worth noting that restricting ourselves to packages already used in BONMIN also forbids the use of MIQCP solvers. For programs (EthP) that uses the auto-balance option, we use the following procedure:

1. Formulate two programs, say $(EthP - AB0)$ and $(EthP - AB1)$, by replacing all instances of x_{AB} by 0 and 1 respectively.

2. If either of $(EthP - AB0)$ and $(EthP - AB1)$ contains constraint that does not fall into type 1-5 described in section 2.4, pass (EthP) to BONMIN and return the result.

3. Otherwise, pass both $(EthP - AB0)$ and $(EthP - AB1)$ to **simple-reform**, and return the better solution of the two.

Below, we present the runtime reduction obtained by implementing the two procedures above. All tests are performed in window's subsystem for Linux (WSL) on windows 10, on a machine with 12GB RAM and a 2.50GHz Intel(R) Core(TM) i7-6500U processor. The test cases are generated by taking the pass projects that Ethelo granted us access to,

and modifying the votes such that only one of the provided votes is being considered. We generated up to 10 cases for each available project. For the projects where less than 10 votes have been collected, we generated a test case for each of the provided votes. Information about hte projects are presented in table 3.1, and the runtime results are shown in table 3.2. As can be seen from the tables, the aforementioned reformulations resulted in at least 99% reduction in CPU runtime for the projects that do not contain quadratic constraints. For the only project that used quadratic constraints, namely CCD, a runtime reduction of 87.9% was observed when compared to the original implementation that uses BONMIN.

Project	Problem Size			g2 cons type		g1 cons type			N_Cases
	n	m1	m2	Linear	Quotient	XOR	Linear	Quad	
parks	13	-	2	2	-	-	-	-	10
BBB	91	12	-	-	-	12	-	-	10
carbon	76	13	2	1	1	13	-	-	10
citizen	48	5	3	2	1	5	-	-	9
climate	76	13	2	1	1	13	-	-	10
granting	50	1	-	-	-	-	1	-	10
semistic	97	9	-	-	-	9	-	-	7
stratford	47	3	-	-	-	3	-	-	10
DAO	243	40	1	1	-	40	-	-	8
CCD	43	11	-	-	-	3	6	2	10

Table 3.1: Information on Problem Formulation for Single-influence cases

Project	BONMIN	CBC	Reduction
parks	1.201	0.002	99.9%
BBB	4.280	0.008	99.8%
carbon	30.600	0.007	>99.9%
citizen	4.802	0.005	99.9%
climate	30.332	0.009	>99.9%
granting	0.357	0.001	99.8%
semistic	7.650	0.007	99.9%
stratford	0.323	0.003	99.1%
DAO	165.820	0.025	>99.9%
CCD	3.870	0.467	87.9%

Table 3.2: Average CPU Runtime Result for Single-Influence Cases

3.2 Partial Results

After seeing the significant reduction in the preceding section, we also tried to identify other structures in Ethelo's previous projects and see if we can improve their runtime by introducing new cuts to the solver. At the time when the experiment in the preceeding section was first done, there were only 8 projects available: the projects BBB and DAO were not available at the time. Of the available 8 projects, we observed that 2 of them (semistic, stratford) used only XOR constraints and can be proven to be integral (that is, solving its continuous relaxation naturally yields the optimal solution of the integer program), and 2 of the remaining 6 (parks and citizen) contain constraints that are equivalent to $l \leq \alpha^T x \leq u$ for some $\alpha \in \mathbb{R}^n_+, l, u \in \mathbb{R}$, which we called the "two sided knapsack constraint", making it the most common constraint type after XOR constraints, covering constraints $\alpha^T x \geq l$, and knapsack constraints $\alpha^T x \leq u$. Further, seeing that the XOR constraints often arise from the commonly-used slider widget on the survey, and that in all of the available projects each variable is only used in at most one XOR constraint, we considered the following structure and attempted to find new cuts for it:

$$\begin{array}{ll} \max & p^T x \\ s.t. & l \leq \alpha^T x \leq u \\ & \sum_{i \in S} x_i = 1 \quad \forall S \in \mathbb{S} \\ & x \in \{0,1\}^n \end{array} \tag{2SKP}$$

where $l, u \in \mathbb{R}$, $\alpha \in \mathbb{R}^n_+$, and $\mathbb{S} \subseteq 2^{\mathbb{O}} = 2^{[n]}$ is a collection of disjoint sets such that each $S \in \mathbb{S}$ has a size of $|S| \geq 2$. When performing literature review, we have found related work regarding what we call the "multiple-choice covering problem" [32, 15, 14, 31]:

$$\begin{array}{ll} \max & p^T x \\ s.t. & l \leq \alpha^T x \\ & \sum_{i \in S} x_i = 1 \quad \forall S \in \mathbb{S} \\ & x \in \{0,1\}^n \end{array} \tag{MCC}$$

and also regarding the multiple-choice knapsack problem [22, 15, 3, 17]:

$$\begin{array}{ll} \max & p^T x \\ s.t. & \alpha^T x \leq u \\ & \sum_{i \in S} x_i = 1 \quad \forall S \in \mathbb{S} \\ & x \in \{0,1\}^n \end{array} \tag{MCK}$$

While cuts generated for 2-dimensional knapsack problems can also be applied to $(2SKP)$, we were unable to find works on cut-generation that consider precisely the formulation in

$(2SKP)$ to the best of our knowledge. Thus, we asked the question: is there any cut for $(2SKP)$ that cannot be derived from the formulation (MCC) or (MCK)?

Before attempting to answer the above question, we give some notations and basic definitions for completeness. Let $X = \{x \in \{0,1\}^n : \sum_{i \in S} x_i = 1, \forall S \in \mathbb{S}\}$, where $\mathbb{S}$ denotes the same collection of disjoint subsets of $[n]$ as before.

Definition 3.2.1. *We say that a set $C \subseteq \mathbb{R}^n$ is **convex** if $\forall a, b \in C$ and $\forall \lambda \in [0,1]$, we have $\lambda a + (1-\lambda)b \in C$.*

Definition 3.2.2. *For a non-empty finite set $\{x^1, x^2, ..., x^k\}$, we define its **convex hull** to be:*

$$conv\{x^1, x^2, ..., x^k\} = \left\{ \sum_{i=1}^{k} \lambda_i x^i : \lambda \in \mathbb{R}^k, \lambda \geq 0, \sum_{i=1}^{k} \lambda_i = 1 \right\}$$

Then, showing that all valid cuts to $(2SKP)$ are valid for one of (MCC) and (MCK) is equivalent to showing:

$$conv\{x \in X : l \leq \alpha^T x \leq u\} = conv\{x \in X : \alpha^T x \leq u\} \cap conv\{x \in X : \alpha^T x \geq l\} \qquad (*)$$

for any $\alpha \in \mathbb{R}^n_+, l, u \in \mathbb{R}$ with $l \leq u$. By defining:

- $P_I = conv\{x \in X : l \leq \alpha^T x \leq u\}$
- $P_L = conv\{x \in X : \alpha^T x \leq u\}$
- $P_U = conv\{x \in X : \alpha^T x \geq l\}$

we may rewrite $(*)$ as $P_I = P_L \cap P_U$.

In attempt to find new cuts for $(2SKP)$ that cannot be obtained from (MCK) or (MCC), we attempted to disprove $(*)$ and tried constructing counter-examples, both by hand and with the help of a computer program. We implemented a program that draws $\alpha \in \mathbb{R}^n_+, l, u \in \mathbb{R}_+$ randomly in a way that guarantees $P_I \neq \emptyset$, with dimension $n \leq 10$, and relaxed the XOR constraints to consider $X = \{0,1\}^n$. Then, we used PORTA ("POlyhedron Representation Transformation Algorithm", a publicly available software [5]) to verify whether the extreme points of P_I are same as those of $P_L \cap P_U$. However, after running the program for more than 24 hours, we were unable to find a counter-example with dimension $n \leq 10$. Hence, we conjectured that the statement $(*)$ is true for all $\alpha \in \mathbb{R}^n_+, l, u \in \mathbb{R}$. While we are also unable to prove that $(*)$ is correct in general, we will present in this section the intermediate results that were gathered during our attempt.

3.2.1 Sufficient Conditions for general finite set X

While the results in this subsection (section 3.2.1) were derived for $\alpha \in \mathbb{R}^n_+$, $l, u \in \mathbb{R}$ with $l \leq u$, and $X = \{x \in \{0,1\}^n : \sum_{i \in S} x_i = 1, \forall S \in \mathbb{S}\}$ with $\mathbb{S}$ being a collection of subsets of $[n]$, since all results can be applied to a general finite set $X \subseteq \mathbb{R}^n$ and any $\alpha \in \mathbb{R}^n$, we will present the results under the latter settings. For future reference, let the finite set $X \subseteq \mathbb{R}^n$, $\alpha \in \mathbb{R}^n$, and $l, u \in \mathbb{R}$ with $l \leq u$ be fixed. Following previous notations, let:

- $P_L = conv\{x \in X : \alpha^T x \leq u\}$
- $P_U = conv\{x \in X : l \leq \alpha^T x\}$
- $P_I = conv\{x \in X : l \leq \alpha^T x \leq u\}$

Then, the statement $(*)$ can be expressed as $P_I = P_L \cap P_U$. Observe from our definition that we always have $P_I \subseteq P_L$ and $P_I \subseteq P_U$, and hence $P_I \subseteq P_L \cap P_U$. To show that $P_I = P_L \cap P_U$, it suffices to show that $P_L \cap P_U \subseteq P_I$.

We first observe the following property:

Property 3.2.3. *For any $\bar{v} \in P_L \cap P_U$, we have $l \leq \alpha^T \bar{v} \leq u$.*

Proof. By definition of convex hull and P_L, we may write $\bar{v} = \sum_{i=1}^k \lambda_i v^i$ for some $v^1, v^2, ..., v^k \in X$ with $\alpha^T v^i \leq u$, and $\lambda \in \mathbb{R}^n$ with $\lambda > 0$. Therefore, we have $\alpha^T \bar{v} = \alpha^T (\sum_{i=1}^k \lambda_i v^i) = \sum_{i=1}^k \lambda_i (\alpha^T v^i) \leq \sum_{i=1}^k \lambda_i \cdot u = u$. Similarly, we can argue that $\alpha^T \bar{v} \geq l$ by considering $\bar{v} \in P_U$. □

We further note that any point $\bar{v} \in (P_L \cap P_U) \setminus P_I$ cannot be on the boundaries of $l \leq \alpha^T x \leq u$. That is:

Theorem 3.2.4. *For any $\bar{v} \in P_L \cap P_U$ with either $\alpha^T \bar{v} = l$ or $\alpha^T \bar{v} = u$, then $\bar{v} \in P_I$.*

Proof. Suppose that $\bar{v} \in P_L \cap P_U$ is such that $\alpha^T \bar{v} = l$. Then, since $\bar{v} \in P_U = conv\{x \in X : \alpha^T x \geq l\}$, we can write $\bar{v} = \sum_{i=1}^k \lambda_i v^i$ for some $v^1, v^2, ..., v^k \in X$ with $\alpha^T v^i \geq l$ and $\lambda \in \mathbb{R}^k$ with $\lambda > 0$. However, since:

$$l = \alpha^T \bar{v} = \alpha^T \sum_{i=1} \lambda_i v^i = \sum_{i=1}^k \lambda_i (\alpha^T v^i) \geq \sum_{i=1}^k \lambda_i \cdot l = l$$

we must have $\alpha^T v^i = l$ for all i. It then follows that $v^i \in X$ satisfies $\alpha^T v^i = l \leq u$, and hence $v^i \in \{x \in X : l \leq \alpha^T x \leq u\}$. Thus, we have $\bar{v} = \sum_{i=1}^{k} \lambda_i v^i \in conv\{x \in X : l \leq \alpha^T x \leq u\} = P_I$, as desired.

The case for $\alpha^T \bar{v} = u$ can be proven analogously by considering $v^1, v^2, ..., v^k \in P_L$ and showing $\alpha^T v^i = u$. □

Then, as a corollary, we see that:

Corollary 3.2.5. *When $l = u$, we have $P_I = P_L \cap P_U$.*

Proof. As noted before, it suffices to show that $P_L \cap P_U \subseteq P_I$. Let $\bar{v} \in P_L \cap P_U$ be arbitrary. Then, by property 3.2.3, we see that $l \leq \alpha^T \bar{v} \leq u$. Since $l = u$, we have $l = \alpha^T \bar{v} = u$; and since $\bar{v} \in P_L \cap P_U$ by construction, we may conclude by theorem 3.2.4 that $\bar{v} \in P_I$. □

We also noticed that since the set X is a finite set, $\{\alpha^T x : x \in X\}$ is a finite set of discrete values. Thus, it might be possible to perturb α, l, u by a small amount without affecting the sets P_L, P_U, P_I. Therefore, we generalize corollary 3.2.5 as follows:

Corollary 3.2.6. *Let $L^0 = \{x \in X : \alpha^T x < l\}$, $C^0 = \{x \in X : l \leq \alpha^T x \leq u\}$, $U^0 = \{x \in X : \alpha^T x > u\}$. If there exists $(\beta, \beta_0) \in \mathbb{R}^n \times \mathbb{R}$ such that:*

- $\forall x \in L^0, \beta^T x < \beta_0$; *and*
- $\forall x \in C^0, \beta^T x = \beta_0$; *and*
- $\forall x \in U^0, \beta^T x > \beta_0$

Then, $P_I = P_L \cap P_U$.

Proof. Let $L^1 = \{x \in X : \beta^T x < \beta_0\}$, $C^1 = \{x \in X : \beta^T x = \beta_0\}$, and $U^1 = \{x \in X : \beta^T x > \beta_0\}$. Notice that (L^1, C^1, U^1) partitions X, and hence are all finite sets. By assumption, we see that $L^0 \subseteq L^1, C^0 \subseteq C^1$, and $U^0 \subseteq U^1$. Further notice that (L^0, C^0, U^0) also partitions the set X. Thus, we have $L^0 = L^1$, $C^0 = C^1$, and $U^0 = U^1$. Observe from definition of P_L, P_I, P_U that:

- $P_L = conv(L^0 \cup C^0) = conv(L^1 \cup C^1) = conv\{x \in X : \beta^T x \leq \beta_0\}$
- $P_I = conv(C^0) = conv(C^1) = conv\{x \in X : \beta^T x = \beta_0\}$
- $P_U = conv(C^0 \cup U^0) = conv(C^1 \cup U^1) = conv\{x \in X : \beta^T x \geq \beta_0\}$

Therefore, by considering $\beta_0 \leq \beta^T x \leq \beta_0$ in place of $l \leq \alpha^T x \leq u$, we know from corollary 3.2.5 that $P_L \cap P_U = P_I$. □

From property 3.2.3 and theorem 3.2.4, we see that any $\bar{v} \in (P_L \cap P_U) \setminus P_I$ must satisfy $l < \alpha^T \bar{v} < u$, and so we may focus our attention on the points $\bar{v} \in P_L \cap P_U$ where $l < \alpha^T \bar{v} < u$. We observed that:

Observation 3.2.7. *If there exists $t \in \mathbb{R}$ with $l < t < u$ such that $\{x \in P_L : \alpha^T x = t\} \subseteq P_I$, then for any $\bar{v} \in P_L$ with $\alpha^T \bar{v} \geq t$, we have $\bar{v} \in P_I$.*

To see this, we will instead prove a more general property:

Property 3.2.8. *If there exists $(\beta, \beta_0) \in \mathbb{R}^n \times \mathbb{R}$ such that:*

1. $\{x \in P_L : \beta^T x = \beta_0\} \subseteq P_I$; *and*
2. $\forall x \in (P_L \setminus P_I) \cap X = \{x \in X : \alpha^T x < l\}$, $\beta^T x < \beta_0$

Then, $\{x \in P_L : \beta^T x \geq \beta_0\} \subseteq P_I$.

Proof. Let $H = \{x \in \mathbb{R}^n : \beta^T x = \beta_0\}$ and $\bar{v} \in P_L$ be such that $\beta^T \bar{v} \geq \beta_0$. Then, by condition (1), we see that $P_L \cap H \subseteq P_I$. If $\beta^T \bar{v} = \beta_0$, then we have $\bar{v} \in P_L \cap H \subseteq P_I$ and we would be done. Thus, it suffices to consider $\beta^T \bar{v} > \beta_0$.

Since $\bar{v} \in P_L$, we may write $\bar{v} = \sum_{i=1}^k \lambda_i v^i$ for some $v^1, v^2, ..., v^k \in X \cap P_L$, and $\lambda \in \mathbb{R}^k$ with $\lambda > 0$. Assume without loss of generality that $v^1, v^2, ..., v^k$ are ordered in decreasing order of $\beta^T v^i$. If $\beta^T v^i \geq \beta_0$ for all i, then by condition (2) we see that $v^i \in (P_L \cap X) \setminus ((P_L \setminus P_I) \cap X) = P_I \cap X$ for all i, and hence $\bar{v} \in conv(P_I \cap X) = P_I$. Otherwise, let $k_0 \leq k$ be the smallest index such that $\beta^T v^{k_0} < \beta_0$. Let $\lambda' = \sum_{i=1}^{k_0 - 1} \lambda_i$. Since $\lambda > 0$, $\sum_{i=1}^k \lambda_i = 1$, and $k_0 - 1 < k$, we see that $\sum_{i=k_0}^k \lambda_i = 1 - \lambda' > 0$, and that $0 < \lambda' < 1$. Let $q^1 = \frac{1}{\lambda'} \sum_{i=1}^{k_0 - 1} \lambda_i v^i$, and $q^2 = \frac{1}{1-\lambda'} \sum_{i=k_0}^k \lambda_i v^i$. Then, we have:

- $\bar{v} = \lambda' q^1 + (1 - \lambda') q^2$; and
- $q^1, q^2 \in conv\{v^1, v^2, ..., v^k\} \subseteq P_L$; and
- $\beta^T q^2 < \beta_0$ by choice of k_0.

Since $\beta^T q^2 < \beta_0 < \beta^T \bar{v}$, and $\beta^T \bar{v} = \lambda' \beta^T q^1 + (1 - \lambda') \beta^T q^2$, we see that $\beta^T q^1 > \beta^T \bar{v} > \beta_0$. Also consider $f : [0, 1] \to \mathbb{R}$ given by:

$$f(\mu) = \mu \beta^T q^1 + (1 - \mu) \beta^T q^2$$

Since $f(\lambda') = \beta^T \bar{v} > \beta_0$ and $f(0) = \beta^T q^2 < \beta_0$, by Intermediate Value Theorem we see that there exists $\mu_0 \in (0, \lambda')$ such that $f(\mu_0) = \beta_0$. Take:

$$\bar{q} = \mu_0 q^1 + (1 - \mu_0) q^2$$

Then, we have $\beta^T \bar{q} = f(\mu_0) = \beta_0$. Since $\mu_0 \in (0, \lambda') \subseteq (0, 1)$, we also have $\bar{q} \in conv\{q^1, q^2\} \subseteq P_L$. Thus, $\bar{q} \in P_L \cap H \subseteq P_I$. Now, note:

$$\begin{aligned}
\bar{v} =& \lambda' q^1 + (1 - \lambda') q^2 \\
=& \left(\lambda' - \frac{\mu_0 (1 - \lambda')}{1 - \mu_0} \right) q^1 + \frac{1 - \lambda'}{1 - \mu_0} \left(\mu_0 q^1 + (1 - \mu_0) q^2 \right) \\
=& \frac{\lambda' - \mu_0}{1 - \mu_0} q^1 + \frac{1 - \lambda'}{1 - \mu_0} \bar{q}
\end{aligned}$$

Since $\mu_0 < \lambda'$, we see that $1 - \mu_0 > 1 - \lambda'$; since $\mu_0 \in (0, \lambda') \subseteq (0, 1)$, we see that $\frac{\lambda' - \mu_0}{1 - \mu_0}, \frac{1 - \lambda'}{1 - \mu_0} \in (0, 1)$. Further, observe that $\frac{\lambda' - \mu_0}{1 - \mu_0} + \frac{1 - \lambda'}{1 - \mu_0} = 1$. Therefore, we have $\bar{v} \in conv\{q^1, \bar{q}\}$.

Recall that by construction, $q^1 \in conv\{x \in X : \beta^T x > \beta_0\} \subseteq P_I$, and we have shown that $\bar{q} \in P_I$. Thus, we may conclude that $\bar{v} \in conv\{q^1, \bar{q}\} \subseteq P_I$, as desired. □

Observation 3.2.7 can be treated as a special case of Property 3.2.8 where $(\beta, \beta_0) = (\alpha, t)$. Now, analogous to Property 3.2.8, we can show:

Property 3.2.9. *If there exists $(\beta, \beta_0) \in \mathbb{R}^n \times \mathbb{R}$ such that:*

1. $\{x \in P_U : \beta^T x = \beta_0\} \subseteq P_I$; *and*
2. $\forall x \in (P_U \setminus P_I) \cap X = \{x \in X : \alpha^T x > u\}$, $\beta^T x > \beta_0$.

Then, $\{x \in P_U : \beta^T x \leq \beta_0\} \subseteq P_I$.

Proof. Analogous to property 3.2.8. □

Then, combining properties 3.2.8 and 3.2.9, we get the following theorem:

Theorem 3.2.10. *If there exists $(\beta, \beta_0) \in \mathbb{R}^n \times \mathbb{R}$ such that:*

- $\{x \in P_L : \beta^T x = \beta_0\} \subseteq P_I$, $\{x \in P_U : \beta^T x = \beta_0\} \subseteq P_I$; *and*

- $\forall x \in (P_U \setminus P_I) \cap X, \beta^T x > \beta_0$; *and*
- $\forall x \in (P_L \setminus P_I) \cap X, \beta^T x < \beta_0$

Then, $P_L \cap P_U = P_I$.

Proof. Note that the 3 conditions for this theorem are precisely the conditions for Properties 3.2.8, 3.2.9 combined. Thus, we may write:

$$\begin{aligned} P_L \cap P_U =& \{x \in P_L \cap P_U : \beta^T x \geq \beta_0\} \cup \{x \in P_L \cap P_U : \beta^T x \leq \beta_0\} \\ \subseteq& \{x \in P_L : \beta^T x \geq \beta_0\} \cup \{x \in P_U : \beta^T x \leq \beta_0\} \\ \subseteq& P_I \cup P_I \\ =& P_I \end{aligned}$$

where the last (ie. second) containment follows from both properties 3.2.8, 3.2.9. □

Corollary 3.2.11. *If there exists* $(\beta, \beta_0) \in \mathbb{R}^n \times \mathbb{R}$ *such that:*

- $\{x \in conv(X) : \beta^T x = \beta_0\} \subseteq P_I$; *and*
- $\forall x \in (P_U \setminus P_I) \cap X, \beta^T x > \beta_0$; *and*
- $\forall x \in (P_L \setminus P_I) \cap X, \beta^T x < \beta_0$

Then, $P_L \cap P_U = P_I$.

Proof. Noticing that $P_L \subseteq conv(X)$, $P_U \subseteq conv(X)$, and $\{x \in conv(X) : \beta^T x = \beta_0\} \subseteq P_I$ necessarily implies $\{x \in P_L : \beta^T x = \beta_0\} \subseteq P_I$ and $\{x \in P_U : \beta^T x = \beta_0\} \subseteq P_I$, the conclusion follows from theorem 3.2.10. □

3.2.2 Implication on $X = \{0,1\}^n$

Before stating the implication of theorem 3.2.10 on $X = \{0,1\}^n$, we first introduce a few basic definitions from chapter 4 of a textbook by Conforti, Cornuéjols and Zambelli [7]:

Definition 3.2.12. *A convex set* P *is integral if* $P = conv(P \cap \mathbb{Z}^n)$.

Definition 3.2.13. *A matrix* $A \in \mathbb{R}^{m \times n}$ *is totally unimodular if all of its square sub-matrix has determinant* $0, 1,$ *or* -1.

Theorem 3.2.14 (theorem 4.5 of [7])**.** *Let* $A \in \mathbb{Z}^{m \times n}$. *The polyhedron* $Q = \{x \in \mathbb{R}^n : c \leq Ax \leq d, l \leq x \leq u\}$ *is integral for all integral vectors* c, d, l, u *if and only if* A *is totally unimodular.*

With the above definitions, we show that theorem 3.2.10 has the following implication:

Corollary 3.2.15. *Let $\alpha \in \mathbb{R}^n_+, l, u \in \mathbb{R}$ be such that $l \leq u$. Let:*

- $P_L = conv\{x \in \{0,1\}^n : \alpha^T x \leq u\}$;
- $P_U = conv\{x \in \{0,1\}^n : \alpha^T x \geq l\}$; *and*
- $P_I = conv\{x \in \{0,1\}^n : l \leq \alpha^T x \leq u\}$.

Then, if there exists integer $t \in [n]$ such that all $S \subseteq [n]$ with $|S| = t$ satisfies $l \leq \sum_{i \in S} \alpha_i \leq u$, then we have $P_I = P_L \cap P_U$.

Proof. Let $H = \{x \in \mathbb{R}^n : \sum_{i=1}^n x_i = t\}$ and $H_0 = H \cap [0,1]^n = \{x \in \mathbb{R}^n : t \leq \mathbf{1}^T x \leq t, 0 \leq x \leq \mathbf{1}\}$, where $\mathbf{1}$ denotes an all-one vector. Since $\mathbf{1}^T$ is a row matrix, the only square sub-matrices of $\mathbf{1}^T$ are the 1-by-1 sub-matrices, which consists of a single entry of 1. Thus, $\mathbf{1}^T$ is totally unimodular, and so by theorem 3.2.14 we see that H_0 is integral. Thus, we may write:

$$H_0 = conv\left\{x \in [0,1]^n \cap \mathbb{Z}^n : \sum_{i=1}^n x_i = t\right\} = conv\left\{x \in \{0,1\}^n : \sum_{i=1}^n x_i = t\right\}$$

Note that by the choice of t, we see that $\{x \in \{0,1\}^n : \sum_{i=1}^n x_i = t\} \subseteq \{x \in \{0,1\}^n : l \leq \alpha^T x \leq u\}$, and so $H_0 \subseteq P_I$. Now, since $P_L, P_U \subseteq [0,1]^n$, we have $P_L \cap H \subseteq [0,1]^n \cap H = H_0 \subseteq P_I$, and $P_U \cap H \subseteq [0,1]^n \cap H = H_0 \subseteq P_I$. Also, since $\alpha \geq 0$:

- If $\mathbf{1}^T \bar{x} \leq t$, then there exists $x' \in [0,1]^n$ with $x' \geq \bar{x}$ and $\mathbf{1}^T x' = t$, which gives $\alpha^T \bar{x} \leq \alpha^T x'$ and $x' \in H_0$. Since $H_0 \subseteq P_I$, it follows that $\alpha^T \bar{x} \leq \alpha^T x' \leq u$.
- Similarly, if $\mathbf{1}^T \bar{x} \geq t$, then there exists $x' \in [0,1]^n$ with $x' \leq \bar{x}$ and $\mathbf{1}^T x' = t$, which gives $\alpha^T \bar{x} \geq \alpha^T x'$ and $x' \in H_0$. Since $H_0 \subseteq P_I$, it follows that $\alpha^T \bar{x} \geq \alpha^T x' \geq l$.

In short, we argued that for any $\bar{x} \in \{0,1\}^n$, we have $\mathbf{1}^T \bar{x} \leq t$ implies $\alpha^T \bar{x} \leq u$, and $\mathbf{1}^T \bar{x} \geq t$ implies $\alpha^T \bar{x} \geq u$. Taking contrapositive gives us that:

- For any $\bar{x} \in \{0,1\}^n$ with $\alpha^T \bar{x} > u$, $\mathbf{1}^T \bar{x} > t$; and
- For any $\bar{x} \in \{0,1\}^n$ with $\alpha^T \bar{x} < l$, $\mathbf{1}^T \bar{x} < t$.

Since we have also showed that $H_0 := \{x \in [0,1]^n : \mathbf{1}^T x = t\} \subseteq P_I$, by considering Corollary 3.2.11 with $(\beta, \beta_0) = (\mathbf{1}, t)$, we may conclude that $P_I = P_L \cap P_U$ □

Loosely speaking, corollary 3.2.15 says that when the bounds l, u are sufficiently far apart, then P_I can be described by simply merging the outer descriptions (ie. inequality descriptions) of P_L and P_U. Unfortunately, we were unable to progress beyond this point. Noting the significant runtime reduction for single-influence cases that was obtained in section 3.1, we decided to switch our attention to the multi-influence cases.

Chapter 4

Multi-Influence Cases

Now we switch our attention to the multi-influence cases. From section 2.5, the MINLP for this case is given by:

$$\begin{array}{lll} \max & \eth(\mu(x), \Sigma(x)) & \\ s.t. & l_i \leq x_{AB} \cdot g_i^1(x) \leq u_i & \forall i \in \{1, 2, ..., m_1\} \\ & l_i \leq g_i^2(x) \leq u_i & \forall i \in \{1, 2, ..., m_2\} \\ & \|x\|_1 \geq 1 & \\ & x \in \{0, 1\}^n & \end{array} \tag{EthP-MI}$$

Since the Ethelo function $\eth$ is non-linear and not continuous, it is hard for us to reformulate (EthP) as an MILP, even when both g^1, g^2 are linear functions. However, by replacing the Ethelo function with a piecewise linear function that satisfies a few special properties, it is possible to approximate (EthP) with a mixed-integer quadratically constrained program (MIQCP), which can then be either passed to a MIQCP solver or further linearized using common RLT techniques. The remainder of this chapter will be divided into 4 parts. In section 4.1, we will present the procedure we use for remodeling a mathematical program with piecewise linear objective function using linear constraints, and the properties that the piecewise linear function needs to satisfy. In section 4.2, we present the procedure for applying the results in section 4.1 to (EthP) assuming that a proper piecewise linear function is used to approximate the Ethelo function. Then, in section 4.3, we present the necessary tools for constructing the piecewise linear function that satisfies the requirements in section 4.1, and from which the l_2-distance to the Ethelo function is minimized. Finally, in section 4.4, we present the computational results for the multi-influence cases.

4.1 Reformulating Piecewise Linear Function

In this section, we will present the procedures for reformulating $\max\{\bar{f}(x) : x \in X\}$ using linear constraints, where $\bar{f} : D \to \mathbb{R}$ is a piecewise linear function with $D \subseteq \mathbb{R}^2$ compact. Most results in this subsection are taken from [19, 20, 36].

We will start our discussion by defining piecewise linear functions formally:

Definition 4.1.1. *Let $D \subseteq \mathbb{R}^n$ be compact. We define:*

- *We say that a finite set of polytopes $(P_i)_{i=1}^k$ **partitions** D if 1) $\bigcup_{i=1}^k P_i = D$ and 2) for all distinct $i, j \in \{1, 2, ..., k\}$, $relint(P_i) \cap relint(P_j) = \emptyset$.*
- *A function $f : D \to \mathbb{R}$ is called a **piecewise linear function** if its graph $gr(f) = \{(x, f(x)) : x \in D\}$ can be partitioned by some finite collection of polytopes.*

Note that with the above definition, piecewise linear functions are necessarily continuous.

Let $D \subseteq \mathbb{R}^n$ and $X \subseteq D$ be compact, and let $\bar{f} : D \to \mathbb{R}$ be piecewise linear. By definition, we see that $gr(\bar{f})$ can be partitioned by a finite set of polytopes, say $\{P_i\}_{i=1}^k$. Let $v(P_i)$ denote the set of vertices of P_i, and let $V_P = \bigcup_{i=1}^k v(P_i)$. Then, by considering the inner descriptions of the polytopes P_i, ie. describing each P_i as a convex hull of its extreme points, we can model the graph $gr(\bar{f})$ using linear disjunctive constraints as follows:

Property 4.1.2. *(Formulas (16a), (16b) of [21])Let $\bar{f} : D \subseteq \mathbb{R}^n \to \mathbb{R}$ be a piecewise linear function, and let $\{P_i\}_{i=1}^k$ be polyhedrons that partitions $gr(\bar{f})$. Let $v(P_i)$ be the set of extreme points of P_i, $V = \bigcup_{i=1}^k v(P_i)$ be the set of all vertices, and $\mathbb{S} = \{v(P_i) : i \in [k]\}$ be the collection of the sets of extreme points for all P_i's. Then, we can write:*

$$gr(\bar{f}) = \left\{ (x, z) \in \mathbb{R}^n \times \mathbb{R} : \begin{array}{l} \begin{bmatrix} x \\ z \end{bmatrix} = \sum_{v \in V} \lambda_v v \\ \sum_{v \in V} \lambda_v = 1 \\ \lambda \in CDC(\mathbb{S}) \end{array} \right\}$$

where

$$CDC(\mathbb{S}) := \left\{ \lambda \in \mathbb{R}_+^V : \begin{array}{l} \sum_{v \in V} \lambda_v \leq 1 \\ \exists i \in [k] : \lambda_v = 0 \text{ for all } v \notin v(P_i) \end{array} \right\}$$

Note that while the constraint $\sum_{v \in V} \lambda_v \leq 1$ in definition of $CDC(\mathbb{S})$ is not necessary in the above formulation, it is necessary for the subsequent results that we use about $CDC(\mathbb{S})$.

Intuitively, the formulation in Property 4.1.2 says $gr(\bar{f})$ contains precisely the points that can be described as convex combination of points in V, such that all coefficients λ_v with non-zero value correspond to extreme points of a same polytope in $\{P_i\}_{i=1}^k$. The constraint $\lambda \in CDC(\mathbb{S})$ is referred to as "combinatorial disjunctive constraint" (CDC constraints) by Huchette and Vielma [20], hence the abbreviation. CDC constraints can be represented by Mixed-Integer Linear Program. One way of doing so is by using the "independent branching (IB) formulation", proposed by Vielma and Nemhauser [36]. To introduce this formulation, we will need a few extra definitions.

Definition 4.1.3. *Let V be a finite set and $\mathbb{S} = \{S_1, S_2, ..., S_k\} \subseteq 2^V$. We say that the **conflict graph** of $CDC(\mathbb{S})$ is an undirected graph $G = (V, E)$ where*

$$E = \{\{u, v\} \subseteq V : u \neq v, \nexists i \in \{1, 2, ..., k\}, \{u, v\} \subseteq S_i\}$$

In other words, for $\mathbb{S} \subseteq 2^V$, the conflict graph $G = (V, E)$ of $CDC(\mathbb{S})$ is a graph where two vertices u, v are adjacent if and only if at least one of λ_u, λ_v has to be zero under constraint $\lambda \in CDC(\mathbb{S})$. Now, following the definitions given by Huchette and Vielma [19], for any undirected graph G, we say:

Definition 4.1.4. *Let $G = (V, E)$ be an undirected graph.*

- *We say that H is a **biclique** of G if H is a complete bipartite subgraph of G. That is, $H = (A \cup B, A * B)$ for some non-empty disjoint $A, B \subseteq V$, where $A * B = \{\{a, b\} : a \in A, b \in B\}$.*
- *We say that $\{(A_i, B_i)\}_{i=1}^r$ is a **biclique cover** of G with r levels if $\bigcup_{i=1}^r A_i * B_i = E$ and $(A_i \cup B_i, A_i * B_i)$ is a biclique of G for all i.*

Then, the IB formulation for CDC constraints can be given as follows:

Property 4.1.5. *(Proposition 4 in [19]) Let V be a finite set, $\mathbb{S} \subseteq 2^V$ and G be the conflict graph of $CDC(\mathbb{S})$. Let $\{(A_i, B_i)\}_{i=1}^r$ be a biclique cover of G. Then:*

$$CDC(\mathbb{S}) = \left\{ \lambda \in \mathbb{R}_+^V : \begin{array}{ll} \sum_{j \in A_i} \lambda_j \leq y_i & \forall i \in [r] \\ \sum_{j \in B_i} \lambda_j \leq 1 - y_i & \forall i \in [r] \\ y \in \{0, 1\}^r & \end{array} \right\}$$

Essentially, the IB formulation enforces the CDC constraint by requesting that for each biclique (A_i, B_i) in a biclique cover of the conflict graph, at most one of A_i, B_i can contain variables with non-zero value. Now, putting together properties 4.1.2 and 4.1.5, we can reach the following theorem:

Theorem 4.1.6. *Let $D \subseteq \mathbb{R}^2$ be compact and $\bar{f} : D \subseteq \mathbb{R}^2 \to \mathbb{R}$ be piecewise linear. Let $gr(\bar{f})$ be partitioned by polytopes $\{P_i\}_{i=1}^k$, $v(P_i)$ be the set of extreme points of P_i, and $V = \bigcup_{i=1}^k v(P_i)$, and $\mathbb{S} = \{v(P_i) : i \in [k]\}$. Let $(A_i, B_i)_{i=1}^r$ be a biclique cover of conflict graph of $CDC(\mathbb{S})$. Then, the program $\max\{\bar{f}(x) : x \in X\}$ is equivalent to:*

$$\begin{cases} \max & z \\ st. & \begin{bmatrix} x \\ z \end{bmatrix} = \sum_{v \in V} \lambda_v v \\ & \sum_{v \in V} \lambda_v = 1 \\ & \sum_{v \in A_i} \lambda_v \leq y_i \qquad \forall i \in [r] \\ & \sum_{v \in B_i} \lambda_v \leq 1 - y_i \quad \forall i \in [r] \\ & \lambda \in \mathbb{R}_+^V, y \in \{0,1\}^r \\ & x \in X \end{cases} \tag{MIP}$$

Proof. Follows by replacing $\lambda \in CDC(\mathbb{S})$ in property 4.1.2 with formulation in property 4.1.5. □

Let's assume that the piecewise linear function $\bar{f}$ is given by its graph $gr(\bar{f}) = \bigcup_{i=1}^k P_i$, where all P_i are given by their inner description. To construct (MIP), it remains to find a procedure for constructing small biclique covers for conflict graphs. While a biclique cover always exists for any CDC constraints (for example, for $\mathbb{S} = \{S_1, S_2, ..., S_k\} \subseteq 2^V$, $\{(S_i, V \setminus S_i)\}_{i=1}^k$ is a biclique cover for its underlying conflict graph), finding the smallest biclique cover (in terms of number of levels) is a NP hard problem [13]. In [19], the authors provided an algorithm for finding biclique covers of $\Theta(\log n)$ levels when the conflict graph of $CDC(\mathbb{S})$ satisfies some special requirement. We summarize their results as follows:

Definition 4.1.7. *Let d_1, d_2 be positive integers and $V = [d_1] \times [d_2]$. A collection $\mathbb{T} = (T_i)_{i=1}^k \subseteq 2^V$ is said to be a **grid triangulation** if:*

1. *$|T_i| = 3$ for all $T_i \in \mathbb{T}$, ie. $conv(T_i)$ is a triangle for all $T_i \in \mathbb{T}$.*
2. *$\|u - v\|_1 \leq 1$ for all $u, v \in T_i$ for all $T_i \in \mathbb{T}$, ie. T_i is on a regular grid.*

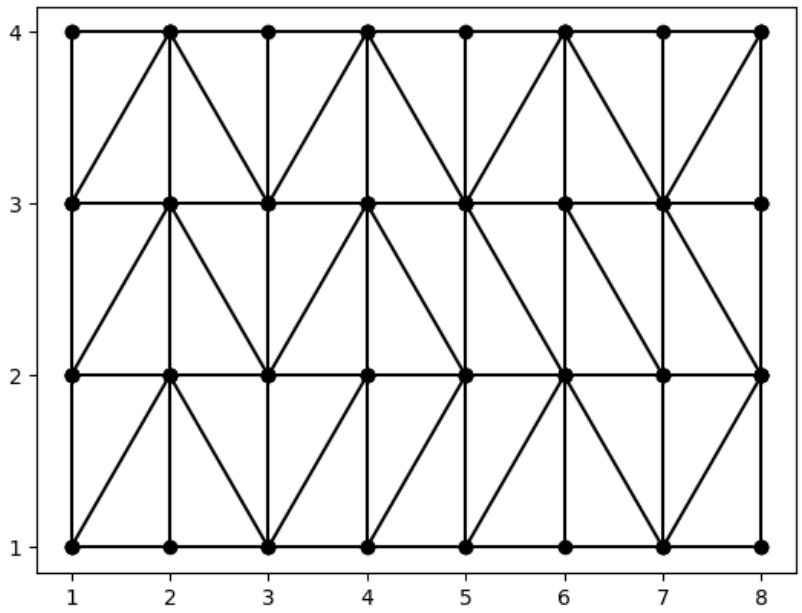

Figure 4.1: A Grid Triangulation with $d_1 = 8, d_2 = 4$, where $\mathbb{T}$ is the collection of sets of extreme points of the triangles

3. The triangles $\{conv(T_i) : T_i \in \mathbb{T}\}$ *partition* $D = [1, d_1] \times [1, d_2]$.

Note that in the above definition, $[d_1]$ refers to the set $\{1, 2, ..., d_1\}$ while $[1, d_1]$ denotes a closed interval on $\mathbb{R}$; same applies for $[d_2]$ and $[1, d_2]$.

Theorem 4.1.8. *Let* $\mathbb{T}$ *be a grid triangulation over* $V = [d_1] \times [d_2]$. *Then, there is a biclique cover for the conflict graph of* $CDC(\mathbb{T})$ *with* $\lceil \log(d_1+1) \rceil + \lceil \log(d_2+1) \rceil + 6$ *levels.*

Readers can refer to [19] for an algorithmic construction of such biclique cover.

4.2 Approximating Bi-variate Functions

To apply the formulation (MIP) to Ethelo's problem, we want to approximate the Ethelo function with a bivariate piecewise linear function $\bar{f}$ with grid triangulated domain, which we define as:

Definition 4.2.1. *A piecewise linear function $\bar{f}$ is said to have "grid triangulated domain" if there exists polytopes $\{P_i\}_{i=1}^k$ satisfying:*

- *$\{P_i\}_{i=1}^k$ partitions $gr(\bar{f})$; and*
- *The conflict graph of $CDC(\{v(P_i) : i \in [k]\}$ is isomorphic to that of some $CDC(\mathbb{T})$, where $\mathbb{T}$ is a grid triangulation.*

To construct grid triangulated piecewise linear functions $\bar{f} : D \to \mathbb{R}$ where $D = [a_l, a_u] \times [b_l, b_u] \subseteq \mathbb{R}^2$, we designed a procedure, named **GTConstruct**, that constructs the function $\bar{f}$ as follows:

1. Given $d_1, d_2 \in \mathbb{Z}$ with $d_1, d_2 \geq 2$, partition $[a_l, a_u]$ and $[b_l, b_u]$ with sequences $a_l = a_1 < a_2 < ... < a_{d_1} = a_u$ and $b_l = b_1 < b_2 < ... < b_{d_2} = b_u$.
2. Divide the region D into smaller squares, which we call "cells", with vertical lines $x = a_i : i \in [d_1]$ and horizontal lines $y = b_j : j \in [d_2]$. Let $v_{i,j} = (a_i, b_j)$ denote the grid points, and $V = \{v_{i,j} : i \in [d_1], j \in [d_2]\}$ be the set of all grid points.
3. Further divide each cell into two triangles by splitting them along one of their diagonals. Let $\mathbb{T}$ denote the set of all resulting triangles. For each $T \in \mathbb{T}$, let $v(T)$ denote its vertices. Let $v(\mathbb{T}) = \{v(T) : T \in \mathbb{T}\}$.
4. For each $v \in V$, choose $F_v \in \mathbb{R}$, and construct the graph of $\bar{f} : D \to \mathbb{R}$ as follows:

$$gr(\bar{f}) = \bigcup_{T \in \mathbb{T}} conv\{(v, F_v) : v \in v(T)\}$$

   In other words, the function values on gridpoint v is given by F_v, and for all other points $x \in D$, if $T \in \mathbb{T}$ is such that $x \in T$, the function value $\bar{f}(x)$ is defined by linear interpolation of function values on vertices (v, F_v) of vertices $v \in v(T)$.

Taking figure 4.1 as an example, the procedure **GTConstruct** works as follows:

1. We are given $d_1 = 8, d_2 = 4$, and $D = [1, 8] \times [1, 4]$. We choose the partition $\{a_i\}_{i=1}^8, \{b_j\}_{j=1}^4$ such that $a_i = i, b_j = j$ for all i, j.
2. Divide the region $D = [1, 8] \times [1, 4]$ into a collection $\mathcal{C}$ of cells, given by $\mathcal{C} = \{[a_i, a_{i+1}] \times [b_j, b_{j+1}] : i \in \{1, 2, ..., d_1 - 1 = 7\}, j \in \{1, 2, 3\}\}$, which are the squares in Fig. 4.1. The set of grid points are given by $V = \{1, 2, 3, ..., 8\} \times \{1, 2, 3, 4\}$, and $v_{i,j} = (i, j)$ for all $i = 1, 2, ..., 8$, $j = 1, 2, 3, 4$.

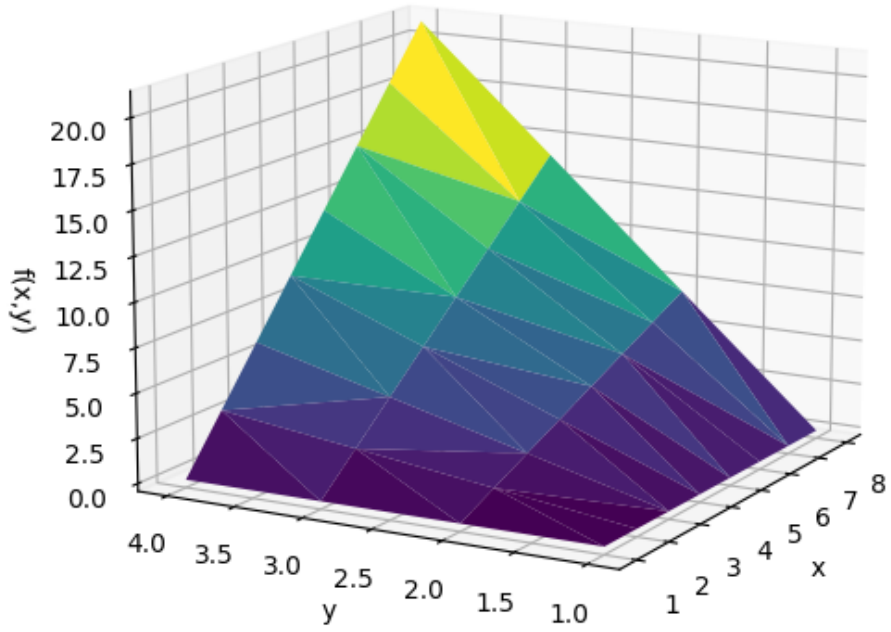

Figure 4.2: Example for piecewise linear function constructed with **GTConstruct**

3. Each cell $C \in \mathcal{C}$ is divided into two triangles to give the partition in Fig 4.1, $\mathbb{T}$ denotes the set of all triangles in Fig 4.1.

4. Finally, we need to pick the function values F_v for all $v \in V = \{1, 2, 3, ..., 8\} \times \{1, 2, 3, 4\}$, and take linear interpolation in each triangle $T \in \mathbb{T}$. If we take $F_{i,j} = (i-1)(j-1)$ for all i, j, then the resulting function will be as shown in Fig 4.2.

Property 4.2.2. ***GTConstruct*** *returns a proper piecewise-linear function with grid triangulated domain.*

Proof. To see that **GTConstruct** actually returns a proper function, we show that for any $x \in D$, $\bar{f}(x) = \{z : (x, z) \in gr(\bar{f})\}$ is a singleton. Let $x \in D$ be arbitrary. Since $\mathbb{T}$ partitions D by construction, we see that $\bar{f}(x) \neq \emptyset$. Also, if there is an unique $T \in \mathbb{T}$ such that $x \in T$, then it's easy to see that $\bar{f}(x)$ is unique. Otherwise, say there are two distinct $T_1, T_2 \in \mathbb{T}$ such that $x \in T_1$ and $x \in T_2$. Let $\hat{v}(T_i; F) = \{(v, F_v) : v \in v(T_i)\}$ for both $i = 1, 2$. To see that $\bar{f}(x)$ is a singleton, we want to show that $\hat{v}(T_1; F) = \hat{v}(T_2; F)$ and are both singletons.

Suppose for contradiction that there exists distinct $z_1, z_2 \in \mathbb{R}$ such that $(x, z_1) \in \hat{v}(T_1; F)$ and $(x, z_2) \in \hat{v}(T_2; F)$. From the construction of $\mathbb{T}$, we observe that for any $T_1, T_2 \in \mathbb{T}$, the intersection $T_1 \cap T_2$ can only be $\emptyset$, a singleton $\{v\} \subseteq V$, or a common edge of T_1, T_2. Since $x \in T_1 \cap T_2$, we see that $T_1 \cap T_2$ is not empty. If $T_1 \cap T_2 = \{v\}$ for some grid point $v \in V$, it must be that $x = v$, and so by construction of $\hat{v}(T_i; F)$ for $i = 1, 2$ we see that $z_1 = z_2 = F_v$, contradicting $z_1 \neq z_2$. If $T_1 \cap T_2$ is a common edge of T_1 and T_2, then since both T_1, T_2 are triangles we know that T_1, T_2 share exactly 2 common vertices, say v^1, v^2, and $T_1 \cap T_2 = conv(v^1, v^2)$. Since v^1, v^2 are distinct and $x \in T_1 \cap T_2 = conv(v^1, v^2)$, there exists an unique $\lambda \in [0, 1]$ such that $\lambda v^1 + (1-\lambda)v^2 = x_0$. Now, from construction of $\hat{v}(T_i; F)$ for $i = 1, 2$, we see that $z_1 = z_2 = \lambda F_{v^1} + (1 - \lambda)F_{v^2}$, which is again a contradiction. As such, we may conclude that $\bar{f}$ as returned by **GTConstruct** is a proper function. Further, since $gr(\bar{f})$ is a union of finitely many polytopes by construction, $\bar{f}$ is piecewise linear.

Also, by considering the mapping $id : V \to \mathbb{Z}^2, (a_i, b_j) \mapsto (i, j)$, we see that $\{id(v(T)) : T \in \mathbb{T}\}$ is a grid triangulation. Since $\{conv\{(v, F_v) : v \in v(T)\} : T \in \mathbb{T}\}$ is a collection of polytope that partitions $gr(\bar{f})$ by construction, we see that the returned function $\bar{f}$ is indeed a piecewise linear function with grid triangulated domain, as desired. □

We end this subsection by noting that **GTConstruct** can be used for approximating the Ethelo function by considering $D = [-1, 1] \times [0, 1]$.

4.3 Best-fitting with Grid Triangulated Piecewise Linear Functions

In **GTConstruct**, there are 4 parameters that needs to be inputted for constructing $\bar{f}$: the two partitions $\{a_i\}_{i=1}^{d_1}$ and $\{b_j\}_{j=1}^{d_2}$, triangulation $\mathbb{T}$, and the function values $F \in \mathbb{R}^V$. Thus, it is natural to ask how should we choose these parameters such that the resulting function $\bar{f}$ best approximates the Ethelo function, or any general bivariate function $f : [a_l, a_u] \times [b_l, b_u] \subseteq \mathbb{R}^2 \to \mathbb{R}$. In attempt to answer this question, in this section we will present the tools for constructing these parameters such that the l_2-distance between the resulting function $\bar{f}$ and the target function f is minimized, and also the computational results when the target function f is the Ethelo function.

For the remainder of this section, we assume that the partitions $\{a_i\}_{i=1}^{d_1}$ and $\{b_j\}_{j=1}^{d_2}$ are pre-determined, and that the target function $f : D = [a_l, a_u] \times [b_l, b_u] \to \mathbb{R}$ is fixed. We also

assume that $f(x,y)$, $xf(x,y)$, $yf(x,y)$, and $(f(x,y))^2$ are all integrable over any compact subset of D, but the function f itself need not be continuous.

4.3.1 Fixed Triangulation

We first consider a simpler case where the triangulation $\mathbb{T}$ is fixed, and attempt to decide only the function values $F \in \mathbb{T}^V$. For convenience, we write $\bar{f}^{(\mathbb{T},F)}$ to denote the function returned by **GTConstruct** that uses triangulation $\mathbb{T}$ and function values $F \in \mathbb{R}^V$ on grid points. Recall that the partitions $\{a_i\}_{i=1}^{d_1}$, $\{b_j\}_{j=1}^{d_2}$ are assumed to be fixed. With these notations, the problem we want to solve for finding F can be expressed as:

$$\min_{F\in\mathbb{R}^V} \|f - \bar{f}^{(\mathbb{T},F)}\|_2^2 := \min_{F\in\mathbb{R}^V} \iint_D (f(x,y) - \bar{f}^{(\mathbb{T},F)}(x,y))^2 dA$$

We claim that $\|f - \bar{f}^{(\mathbb{T},F)}\|_2^2$ is a convex quadratic function in F and can be evaluated either numerically or analytically.

For convenience, we will start with the following standard definitions.

Definition 4.3.1. *A set of vectors $v^1, v^2, ..., v^k \in \mathbb{R}^n$ is said to be **affinely independent** if $\lambda = 0$ is the unique solution to the system $(\sum_{i=1}^k \lambda_i v^i = 0, \sum_{i=1}^k \lambda_i = 0, \lambda \in \mathbb{R}^k)$.*

Definition 4.3.2. *Let $v^1, v^2, v^3 \in \mathbb{R}^2$ be affinely independent, and let $F_{v^1}, F_{v^2}, F_{v^3} \in \mathbb{R}$. Let $v^i = (v_x^i, v_y^i)$ for all $i = 1, 2, 3$. Then, the **linear interpolation** over points $\{(v^i, F_{v^i}) : i = 1, 2, 3\}$ is a linear function $L_{\{(v^i,F_{v^i}):i=1,2,3\}}(x,y) = q_1x + q_2y + q_3$ such that:*

$$L_{\{(v^i,F_{v^i}):i=1,2,3\}}(v^i) = q_1v_x^i + q_2v_y^i + q_3 = F_{v^i}$$

for all $i = 1, 2, 3$.

Property 4.3.3. *Let $v^1, v^2, v^3 \in \mathbb{R}^2$ be affinely independent and $F_{v^i} \in \mathbb{R}$ for $i = 1, 2, 3$. Let $S = \{(v^i, F_{v^i}) : i = 1, 2, 3\}$. Then, the linear interpolation $L_S(x,y)$ is uniquely defined. Further, if v^1, v^2, v^3 are fixed, and $q = (q_1, q_2, q_3)$ is such that $L_S(x,y) = q_1x + q_2y + q_3$, then the mapping $m : \mathbb{R}^3 \to \mathbb{R}^3$ which maps F to q (denoted $F \mapsto q$) is a linear mapping.*

Proof. By definition of linear interpolation, the vector $q = (q_1, q_2, q_3)$ is such that:

$$\begin{bmatrix} v_x^1 & v_y^1 & 1 \\ v_x^2 & v_y^2 & 1 \\ v_x^3 & v_y^3 & 1 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} F_{v^1} \\ F_{v^2} \\ F_{v^3} \end{bmatrix}$$

Since v^1, v^2, v^3 are affinely independent, we see that the following system in λ has only the zero solution:

$$\begin{bmatrix} v_x^1 & v_x^2 & v_x^3 \\ v_y^1 & v_y^2 & v_y^3 \\ 1 & 1 & 1 \end{bmatrix} \lambda = 0$$

which means that $\begin{bmatrix} v_x^1 & v_x^2 & v_x^3 \\ v_y^1 & v_y^2 & v_y^3 \\ 1 & 1 & 1 \end{bmatrix}$ is invertible, and hence so is its transpose $\begin{bmatrix} v_x^1 & v_y^1 & 1 \\ v_x^2 & v_y^2 & 1 \\ v_x^3 & v_y^3 & 1 \end{bmatrix}$.
It then follows that the vector $q = (q_1, q_2, q_3)$ is uniquely defined by:

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} v_x^1 & v_y^1 & 1 \\ v_x^2 & v_y^2 & 1 \\ v_x^3 & v_y^3 & 1 \end{bmatrix}^{-1} \begin{bmatrix} F_{v^1} \\ F_{v^2} \\ F_{v^3} \end{bmatrix}$$

thus the mapping $F \mapsto q$ is a linear mapping, which completes our proof. □

Now, for any $T \in \mathbb{T}$, let:

$$\delta_T(F) = \iint_T (f(x,y) - L_{\{(v,F_v):v\in v(T)\}}(x,y))^2 dA$$

be the squared l_2-distance between the target function f and linear interpolation $L_{\{(v,F_v):v\in v(T)\}}$ over region T. Notice from the construction of $\bar{f}^{(\mathbb{T},F)}$ that for any $(x,y) \in D$ and $T \in \mathbb{T}$ with $(x,y) \in T$, we have:

$$\bar{f}^{(\mathbb{T},F)}(x,y) = L_{\{(v,F_v):v\in v(T)\}}(x,y)$$

Further, since the triangulation $\mathbb{T}$ partitions D by construction, we have:

$$\begin{aligned} \|f - \bar{f}^{(\mathbb{T},F)}\|_2^2 &= \iint_D (f(x,y) - \bar{f}^{(\mathbb{T},F)}(x,y))^2 dA \\ &= \sum_{T\in\mathbb{T}} \iint_T (f(x,y) - \bar{f}^{(\mathbb{T},F)}(x,y))^2 dA \\ &= \sum_{T\in\mathbb{T}} \iint_T (f(x,y) - L_{\{(v,F_v):v\in v(T)\}}(x,y))^2 dA \\ &= \sum_{T\in\mathbb{T}} \delta_T(F) \end{aligned} \tag{1}$$

We claim that $\delta_T(F)$ is a convex quadratic function in F.

Property 4.3.4. *For a linear function $L : \mathbb{R}^2 \to \mathbb{R}$ with $L(x, y) = q_1x + q_2y + q_3$, and for any compact $R \subseteq D$, $\iint_R (f(x, y) - L(x, y))^2 dA$ is quadratic in q.*

Proof. Since the parameters q_1, q_2, q_3 are independent from variables x, y, by expanding:

$$\begin{aligned}
&\iint_R (f(x,y) - L(x,y))^2 dA \\
&= \iint_R (f(x,y) - q_1x - q_2y - q_3)^2 dA \\
&= \iint_R ((f(x,y))^2 + q_1^2x^2 + q_2^2y^2 + q_3^3 - 2q_1xf(x,y) \\
&\quad - 2q_2yf(x,y) - 2q_3f(x,y) + 2q_1q_2xy + 2q_1q_3x + 2q_2q_3y)dA \\
&= \left(\iint_R (f(x,y))^2 dA\right) + \left(\iint_R x^2 dA\right) q_1^2 + \left(\iint_R y^2 dA\right) q_2^2 + \left(\iint_R 1 \cdot dA\right) q_3^2 \\
&\quad - 2\left(\iint_R xf(x,y)dA\right) q_1 - 2\left(\iint_R yf(x,y)dA\right) q_2 - 2\left(\iint_R f(x,y)dA\right) q_3 \\
&\quad + 2\left(\iint_R xydA\right) q_1q_2 + 2\left(\iint_R xdA\right) q_1q_3 + 2\left(\iint_R ydA\right) q_2q_3
\end{aligned}$$

Note that all of the integrations above are independent from q, and can be evaluated either numerically or analytically when a description of f is given. As such, our result follows. □

Property 4.3.5. *For any $T \in \mathbb{T}$, $\delta_T(F)$ is convex in F.*

Proof. We first observe as a corollary of property 4.3.3 that for any fixed triangle $T \in \mathbb{T}$ and point $(x, y) \in T$, $L_{\{(v,F_v):v\in v(T)\}}(x, y)$ is linear in F. That is, for $F^1, F^2 \in \mathbb{R}^V$, $a, b \in \mathbb{R}$ and $T \in \mathbb{T}$, we have:

$$L_{\{(v,aF_v^1+bF_v^2):v\in v(T)\}}(x,y) = a \cdot L_{\{(v,F_v^1):v\in v(T)\}}(x,y) + b \cdot L_{\{(v,F_v^2):v\in v(T)\}}(x,y)$$

For simplicity, let $L_T(F; x, y) = L_{\{(v,F_v):v\in v(T)\}}(x, y)$ for any $T \in \mathbb{T}$, $F \in \mathbb{R}^V$, and $x, y \in \mathbb{R}$. Then, we know from above that $L_T(F; x, y)$ is linear in F, and so for any $\lambda \in (0, 1)$ and

$F^1, F^2 \in \mathbb{R}^V$:

$$
\begin{aligned}
&\delta_T(\lambda F^1 + (1-\lambda)F^2) \\
=&\iint_T [f(x,y) - \bar{f}^{(\mathbb{T},\lambda F^1+\lambda F^2)}(x,y)]^2 dA \\
=&\iint_T [f(x,y) - L_T(\lambda F^1 + (1-\lambda)F^2; x, y)]^2 dA \\
=&\iint_T [\lambda(f(x,y) - L_T(F^1; x, y)) + (1-\lambda)(f(x,y) - L_T(F^2; x.y))]^2 dA \\
\leq&\iint_T \left(\lambda[f(x,y) - L_T(F^1; x, y)]^2 + (1-\lambda)[f(x,y) - L_T(F^2; x, y)]^2\right) dA \\
=&\lambda\delta_T(F^1) + (1-\lambda)\delta_T(F^2)
\end{aligned}
$$

where the inequality follows from convexity of $x \mapsto x^2 : \mathbb{R} \to \mathbb{R}$. □

Now, combining properties 4.3.3,4.3.4, we see that $\delta_T(F)$ is a quadratic function of F with constant coefficients. Property 4.3.5 tells us that $\delta_T(F)$ is convex in F. Thus, $\delta_T(F)$ is a convex quadratic function of F.

Also recall that we have showed $\|\eth - \bar{f}^{(\mathbb{T},F)}\|_2^2 = \sum_{T\in\mathbb{T}} \delta_T(F)$ in (1). Thus, the program we want to solve can be rewritten as:

$$\min_{F\in\mathbb{R}^V} \|\eth - \bar{f}^{(\mathbb{T},F)}\|_2^2 = \min_{F\in\mathbb{R}^V} \sum_{T\in\mathbb{T}} \delta_T(F) \tag{FnCl}$$

For any $T \in \mathbb{T}$, since $\delta_T(F)$ is a convex quadratic function in F, so is $\sum_{T\in\mathbb{T}} \delta_T(F)$. Therefore, we see that $\min_{F\in\mathbb{R}^V} \sum_{T\in\mathbb{T}} \delta_T(F)$ is an unconstrained convex quadratic program.

In our experiments, we noticed that for optimal solution F^* obtained by solving the above program with the Ethelo function being the target function, the difference between the optimal value of (EthP) and that of (MIP) using $\bar{f}^{(\mathbb{T},F^*)}$ as approximation tends to be bigger if it happens that both optimal solutions $x^*, \bar{x}$ falls on a same triangle $T \in \mathbb{T}$ where the order of values $\{F^*_v : v \in v(T)\}$ on the vertices is different from that of $\{\eth(v) : v \in v(T)\}$. Hence, we also consider the following program that fixes some of the ordering of variables:

$$
\begin{cases}
\min & \sum_{T\in\mathbb{T}} \delta_T(F) & \\
s.t. & F_v + \epsilon_0 \leq F_{v'} & \forall (v,v') : \exists T \in \mathbb{T}, v, v' \in v(T), f(v) < f(v') \\
& F \in \mathbb{R}^V &
\end{cases}
\tag{FnCl-ϵ_0}
$$

where $\epsilon_0 \geq 0$ is some pre-determined constant. Note that the ordering does not have to be strict: by setting $\epsilon_0 = 0$, the above simply enforces that F_u is not greater than F_v for any $u, v \in V$ with $f(u) < f(v)$.

4.3.2 Fixed Function Values

After the considering the case where we only adjust function values, we also considered the case where we use the exact function values $F_v = f(v)$ for all $v \in V$, where $f : D \subseteq \mathbb{R}^2 \to \mathbb{R}$ is the target function, which is the Ethelo function $\eth$ in our case, and decide only the triangulation $\mathbb{T}$. Recall from **GTConstruct** that the triangulation $\mathbb{T}$ was defined by splitting each of the cells along one of its diagonals. Let $\mathcal{C}$ denote the set of cells in **GTConstruct**, note that the partitions $\{a_i\}_{i=1}^{d_1}, \{b_j\}_{j=1}^{d_2}$ were assumed to be fixed. For future reference, we define the following:

Definition 4.3.6. *For any square $C = [x_l, x_u] \times [y_l, y_u] \subseteq \mathbb{R}^2$ where $x_l, x_u, y_l, y_u \in \mathbb{R}$ with $x_l < x_u$ and $y_l < y_u$, let's label its 4 vertices $v_1(C) = (x_l, y_l), v_2(C) = (x_l, y_u), v_3(C) = (x_u, y_u), v_4(C) = (x_u, y_l)$ in clockwise order. Let:*

- $T_1(C) = conv\{v^2(C), v^3(C), v^4(C)\}$
- $T_2(C) = conv\{v^1(C), v^3(C), v^4(C)\}$
- $T_3(C) = conv\{v^1(C), v^2(C), v^4(C)\}$
- $T_4(C) = conv\{v^1(C), v^2(C), v^3(C)\}$

*We say that C is split **diagonally** if it is given as the union $C = T_1(C) \cup T_3(C)$, and we say that it is split **skew-diagonally** if it is given as $C = T_2(C) \cup T_4(C)$.*

*Specially, if C is a cell, then we say that C is split **diagonally** in triangulation $\mathbb{T}$ if $T_1(C), T_3(C) \in \mathbb{T}$, and we say that C is split **skew-diagonally** in triangulation $\mathbb{T}$ if $T_2(C), T_4(C) \in \mathbb{T}$.*

For any cell $C \in \mathcal{C}$, let $v(C)$ denote the set of its 4 vertices. Note from the above that the l_2 error contributed by cell C when being split diagonally can be given by $\delta_C^{(D)}(F) = \delta_{T_1(C)}(F) + \delta_{T_3(C)}(F)$, and that when C is being split diagonally can be split skew-diagonally is given by $\delta_C^{(SD)}(F) = \delta_{T_2(C)}(F) + \delta_{T_4(C)}(F)$, where $\delta_T(F)$ follows the same definition as

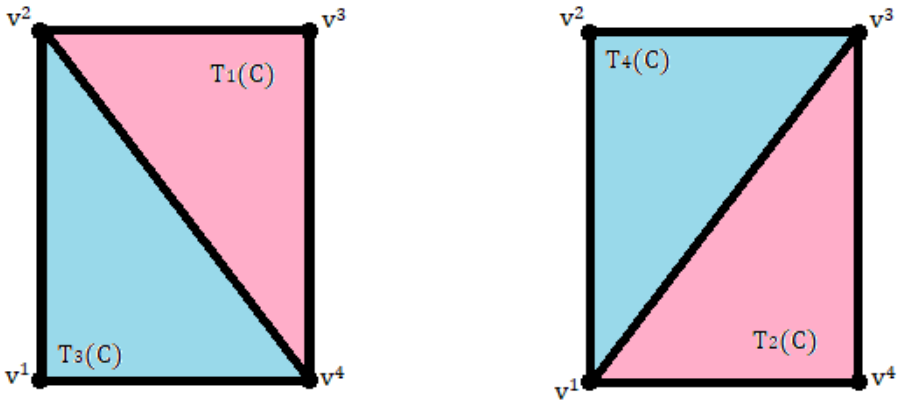

Figure 4.3: Diagonal (left) and skew-diagonal (right) split for a cell

in the preceding subsection. Note that the values $\delta_C^{(D)}(F), \delta_C^{(SD)}(F)$ are independent from how any other cell $C' \in \mathcal{C}$ is being split. Thus, when function values F are fixed to some $F^0 \in \mathbb{R}^V$, to minimize $\|f - \bar{f}^{(\mathbb{T},F^0)}\|_2^2$ we can decide the triangulation $\mathbb{T}$ greedily. That is, for each cell $C \in \mathcal{C}$, we split C diagonally if $\delta_C^{(D)}(F^0) \leq \delta_C^{(SD)}(F^0)$; otherwise we split C skew-diagonally. However, this part is not considered in the computational experiments due to the following observation:

Property. *For a cell $C \in \mathcal{C}$, if $f(x,y) = t_1xy + t_2x + t_3y + t_4$ for some $t \in \mathbb{R}^4$ on C, then $\delta_C^{(D)}(F^0) = \delta_C^{(SD)}(F^0)$, where $F^0 \in \mathbb{R}^V$ is such that $F^0_v = f(v)$ for all $v \in v(C)$.*

To see this, we will first show the following special case:

Lemma 4.3.7. *Let $C = [0,w] \times [0,h]$ for some $w, h \in \mathbb{R}_+$, let $f(x,y) = xy$, and $F^0 \in \mathbb{R}^{v(C)}$ be such that $F^0_{(u,v)} = f(u,v) = uv$ for all $(u,v) \in v(C) \subseteq \mathbb{R}^2$. Define $l_i(x,y) = L_{(u,v,f(u,v)):(u,v)\in T_i(C)}(x,y)$ for all $i = 1,2,3,4$. Then, $\delta_C^{(D)}(F^0) = \delta_C^{(SD)}(F^0)$.*

Proof. We will show that $\delta_{T_i(C)}(F^0) = h^3w^3/180$ for all i by evaluating each function $\delta_{T_i(C)}(F^0)$ one by one. We first consider $T_3(C)$.

Note that $T_3(C) = conv\{(0,0),(w,0),(0,h)\}$, and the linear interpolation l_3 is such that $(0,0) \mapsto 0, (w,0) \mapsto 0, (0,h) \mapsto 0$. Since the zero function $(x,y) \mapsto 0$ is a linear function that satisfies the requirement, and the linear interpolation is unique by property 4.3.3, we

see that $l_3(x,y) = 0$ for all x, y. Also, $T_3(C)$ can be written as:

$$\begin{aligned} T_3(C) =& conv\{(0,0),(w,0),(0,h)\} \\ =& \{(x,y) \in C = [0,w] \times [0,h] : hx + wy \geq hw\} \\ =& \{(x,y) : 0 \leq x \leq w, 0 \leq y \leq \frac{h}{w}(w-x)\} \end{aligned}$$

we can evaluate:

$$\begin{aligned} \delta_{T_3(C)}(F^0) =& \iint_{T_3(C)} (xy-0)^2 dA \\ =& \int_0^w \int_0^{\frac{h}{w}(w-x)} x^2 y^2 dy dx \\ =& \int_0^w x^2 \cdot \frac{1}{3} \cdot \frac{h^3}{w^3}(w-x)^3 dx \\ =& \frac{h^3}{3w^3} \int_0^w \left(w^3x^2 - 3w^2x^3 + 3wx^4 - x^5\right) dx \\ =& \frac{h^3}{3w^3} \left(\frac{1}{3}w^3 \cdot w^3 - \frac{3}{4}w^2 \cdot w^4 + \frac{3}{5}w \cdot w^5 - \frac{1}{6}w^6\right) \\ =& \frac{h^3}{3w^3} \cdot \frac{1}{60} w^6 \\ =& \frac{1}{180} h^3 w^3 \end{aligned}$$

Now, for $T_1(C)$, the linear interpolation l_1 is such that $(0,h) \mapsto 0, (w,0) \mapsto 0, (w,h) \mapsto wh$. Since $(x,y) \mapsto hx + wy - hw$ is a linear function satisfying the requirement, and linear interpolation is unique, we see that $l_1(x,y) = hx + wy - hw$ for all x, y. Also, $T_1(C)$ can be given by:

$$\begin{aligned} T_1(C) =& conv\{(0,h),(w,0),(w,h)\} \\ =& \{(x,y) \in [0,w] \times [0,h] : hx + wy \geq hw\} \\ =& \{(x,y) : 0 \leq x \leq w, \frac{h}{w}(w-x) \leq y \leq h\} \end{aligned}$$

we may write:

$$\begin{aligned}
\delta_{T_1(C)}(F^0) &= \iint_{T_1(C)} (xy - hx - wy + hw)^2 dA \\
&= \int_0^w \int_{\frac{h}{w}(w-x)}^{h} (w-x)^2(h-y)^2 dydx \\
&= \int_0^w \int_0^{h-\frac{h}{w}(w-x)} (w-x)^2(h-y)^2 d(h-y)d(w-x) \\
&= \int_0^w \int_0^{\frac{h}{w}(w-(w-x))} (w-x)^2(h-y)^2 d(h-y)d(w-x) \\
&= \int_0^w \int_0^{\frac{h}{w}(w-u)} u^2 v^2 dvdu \qquad \text{(Change of variable)} \\
&= \delta_{T_3(C)}
\end{aligned}$$

where the last equality comes from the observation that the integration is same as that of $\delta_{T_3(C)}(F^0)$ except with different names for variables.

Similarly, for $T_2(C)$, the linear interpolation l_2 is such that $(0,0) \mapsto 0, (w,0) \mapsto 0, (w,h) \mapsto wh$, so we observe that $l_2(x,y) = wy$. Also, $T_2(C)$ can be given by:

$$\begin{aligned}
T_2(C) &= conv\{(0,0),(w,0),(w,h)\} \\
&= \{(x,y) \in [0,w] \times [0,h] : hx - wy \geq 0\} \\
&= \{(x,y) : 0 \leq x \leq w, 0 \leq y \leq \frac{h}{w}x\}
\end{aligned}$$

Thus,

$$\begin{aligned}
\delta_{T_2(C)}(F^0) &= \int_0^w \int_0^{\frac{h}{w}x} (xy - wy)^2 dydx \\
&= \int_0^w \int_0^{\frac{h}{w}x} (w-x)^2 y^2 dydx \\
&= \int_0^w \int_0^{\frac{h}{w}(w-u)} u^2 y^2 dydu \qquad \text{(Change of variable, } u = w - x\text{)} \\
&= \delta_{T_3(C)}(F^0)
\end{aligned}$$

And for $T_4(C)$, the linear interpolation l_4 is such that $(0,0) \mapsto 0, (0,h) \mapsto 0, (w,h) \mapsto$

wh. We observe that $l_4(x,y) = hx$. $T_4(C)$ can be given by:

$$\begin{aligned} T_4(C) =& conv\{(0,0),(0,h),(w,h)\} \\ =& \{(x,y) \in [0,w] \times [0,h] : hx - wy \leq 0\} \\ =& \{(x,y) : 0 \leq x \leq w, \frac{h}{w}x \leq y \leq h\} \end{aligned}$$

Thus,

$$\begin{aligned} \delta_{T_4(C)}(F^0) &= \iint_{T_4(C)} (xy - hx)^2 dA \\ &= \int_0^w \int_{\frac{h}{w}x}^h x^2(h-y)^2 dy dx \\ &= \int_0^w \int_0^{h-\frac{h}{w}x} x^2 v^2 dv dx \qquad \text{(Change of variable, } v = h - y) \\ &= \int_0^w \int_0^{\frac{h}{w}(w-x)} x^2 v^2 dv dx \\ &= \delta_{T_3(C)}(F^0) \end{aligned}$$

In sum, we showed that $\delta_{T_i(C)}(F^0) = \delta_{T_3(C)}(F^0) = \frac{1}{180}h^3w^3$ for all $i = 1,2,3,4$. Thus, we have:

$$\delta_C^{(D)}(F^0) = \delta_{T_1(C)}(F^0) + \delta_{T_3(C)}(F^0) = \delta_{T_2(C)}(F^0) + \delta_{T_4(C)}(F^0) = \delta_C^{(SD)}(F^0)$$

as desired. □

With the above lemma, we can show the previously-mentioned property:

Property 4.3.8. *For a cell $C \in \mathcal{C}$, if $f(x,y) = t_1xy + t_2x + t_3y + t_4$ for some $t \in \mathbb{R}^4$ on C, then $\delta_C^{(D)}(F^0) = \delta_C^{(SD)}(F^0)$, where $F^0 \in \mathbb{R}^V$ is such that $F_v^0 = f(v)$ for all $v \in v(C)$.*

Proof. We will prove $\delta_C^{(D)}(F_0) = \delta_C^{(SD)}(F_0)$ by relating the quantities $\delta_C^{(D)}(F_0), \delta_C^{(SD)}(F_0)$ to those in lemma 1. Suppose the cell is $C = [x_l, x_u] \times [y_l, y_u]$ for some $x_l, x_u, y_l, y_u \in \mathbb{R}$ with $x_u > x_l, y_u > y_l$. Let $w = x_u - x_l > 0$ and $h = y_u - y_l > 0$, and let $C' = [0,w] \times [0,h]$. We also define:

- $l_i(x,y) = L_{\{(v,F_v^0):v \in v(T_i(C))\}}(x,y)$ for all $i = 1,2,3,4$; and

- $l'_i(x,y) = L_{\{(u,v,uv):(u,v)\in v(T_i(C'))\}}(x,y)$ for all $i = 1,2,3,4$.

Then, by lemma 1, we see that:

$$\sum_{j\in\{1,3\}} \iint_{T_j(C')} (xy - l'_j(x,y))^2 dA = \sum_{j\in\{2,4\}} \iint_{T_j(C')} (xy - l'_j(x,y))^2 dA$$

also, by definition of $\delta_C^{(D)}(F_0)$ and $\delta_C^{(SD)}(F_0)$, we have:

- $\delta_C^{(D)}(F_0) = \sum_{j\in\{1,3\}} \iint_{T_j(C)} (f(x,y) - l_j(x,y))^2 dA$
- $\delta_C^{(SD)}(F_0) = \sum_{j\in\{2,4\}} \iint_{T_j(C)} (f(x,y) - l_j(x,y))^2 dA$

To see that $\delta_C^{(D)}(F_0) = \delta_C^{(SD)}(F_0)$, it suffices to show that

$$\iint_{T_j(C)} (f(x,y) - l_j(x,y))^2 dA = K \iint_{T_j(C')} (xy - l'_j(x,y))^2 dA$$

for all $j = 1,2,3,4$ and for some constant K independent from j.

Let $j \in \{1,2,3,4\}$ be arbitrary. For any set $S \subseteq \mathbb{R}^2$ and vector $v \in \mathbb{R}^2$, we denote the set translations $S + v := \{u + v : u \in S\}$ and $S - v = S + (-v)$. Note that the region of integration can be translated as follows:

$$\iint_{T_j(C)} (f(x,y) - l_j(x,y))^2 dA = \iint_{T_j(C)-(x_l,y_l)} (f(x + x_l, y + y_l) - l_j(x + x_l, y + y_l))^2 dA$$

Also observe that:

- $v_1(C) - (x_l, y_l) = (x_l, y_l) - (x_l, y_l) = (0,0) = v_1(C')$
- $v_2(C) - (x_l, y_l) = (x_l, y_u) - (x_l, y_l) = (0, y_u - y_l) = (0,h) = v_2(C')$
- $v_3(C) - (x_l, y_l) = (x_u, y_u) - (x_l, y_l) = (x_u - x_l, y_u - y_l) = (w,h) = v_3(C')$
- $v_4(C) - (x_l, y_l) = (x_u, y_l) - (x_l, y_l) = (x_u - x_l, 0) = (w,0) = v_4(C')$

Thus, we see that $v_k(C) - (x_l, y_l) = v_k(C')$ for all $k = 1, 2, 3, 4$, and so $v(T_j(C)) - (x_l, y_l) = v(T_j(C'))$, where $v(T)$ denotes the set of the 3 vertices for a triangle T as before. It then follows that:

$$T_j(C) - (x_l, y_l) = conv(v(T_j(C)) - (x_l, y_l)) = conv(v(T_j(C'))) = T_j(C')$$

Therefore:

$$\begin{aligned}\iint_{T_j(C)} (f(x,y) - l_j(x,y))^2 dA &= \iint_{T_j(C)-(x_l,y_l)} (f(x+x_l, y+y_l) - l_j(x+x_l, y+y_l))^2 dA \\ &= \iint_{T_j(C')} (f(x+x_l, y+y_l) - l_j(x+x_l, y+y_l))^2 dA\end{aligned}$$

Let $g_j(x,y) = \frac{1}{t_1}(t_1 xy - f(x+x_l, y+y_l) + l_j(x+x_l, y+y+l))$. Then, we may write:

$$\begin{aligned}&\iint_{T_j(C)} (f(x,y) - l_j(x,y))^2 dA \\ =&\iint_{T_j(C')} (f(x+x_l, y+y_l) - l_j(x+x_l, y+y_l))^2 dA \\ =&\iint_{T_j(C')} (t_1 xy - (t_1 xy - f(x+x_l, y+y_l) + l_j(x+x_l, y+y_l))^2 dA \\ =&\iint_{T_j(C')} (t_1 xy - t_1 g_j(x,y))^2 dA \\ =&t_1^2 \iint_{T_j(C')} (xy - g_j(x,y))^2 dA \qquad (**)\end{aligned}$$

We claim that $g_j(x,y) = l'_j(x,y) := L_{\{(u,v,uv):(u,v)\in v(T_j(C'))\}}(x,y)$ for all x, y. To see this, it suffices to show that $g_j(x,y)$ is a linear function with $g_j(u,v) = uv$ for all $(u,v) \in v(T_j(C'))$.

To see that g_j is linear, we note that since $f(x,y) = t_1 xy + t_2 x + t_3 y + t_4$ by assumption, we have:

$$\begin{aligned}t_1 xy - f(x+x_l, y+y_l) =& t_1 xy - t_1(x+x_l)(y+y_l) - t_2(x+x_l) - t_3(y+y_l) - t_4 \\ =& t_1 xy - t_1(xy + x_l y + y_l x + x_l y_l) - t_2(x+x_l) - t_3(y+y_l) - t_4 \\ =& -t_1(x_l y + y_l x + x_l y_l) - t_2(x+x_l) - t_3(y+y_l) - t_4\end{aligned}$$

which is linear in (x,y), given that $t_1, t_2, t_3, t_4, x_l, y_l$ are all constants. Also, since $l_j(x,y)$ is linear in (x,y), so is $l_j(x+x_l, y+y_l)$. It then follows that the sum $(t_1 xy - f(x+x_l, y+$

$y_l)) + l_j(x + x_l, y + y_l)$ is linear in (x, y), and hence so is $g_j(x, y) := \frac{1}{t_1}(t_1xy - f(x + x_l, y + y_l) + l_j(x + x_l, y + y + l))$.

To see that $g_j(u, v) = uv$ for all $(u, v) \in v(T_j(C'))$, let $(u, v) \in v(T_j(C'))$ be arbitrary. Then:

$$\begin{aligned} t_1(uv - g_j(u, v)) =& t_1uv - t_1g_j(u, v) \\ =& t_1uv - (t_1uv - f(u + x_l, v + y_l) + l_j(u + x_l, v + y + l)) \\ =& f(u + x_l, v + y_l) - l_j(u + x_l, v + y + l) \end{aligned}$$

Recall we have shown that $v(T_j(C)) = v(T_j(C')) + (x_l, y_l)$, which implies that $(u + x_l, v + y_l) \in v(T_j(C))$. Further, recall from definition of l_j that $l_j(x, y) = f(x, y)$ for all $(x, y) \in v(T_j(C))$. Thus, we may conclude that $f(u + x_l, v + y_l) - l_j(u + x_l, v + y + l) = 0$, and so $g_j(u, v) = uv$ for all $(u, v) \in v(T_j(C'))$.

Since $g_j(x, y)$ is a linear function such that $g_j(x, y) = xy$ for all $(x, y) \in v(T_j(C'))$, we may conclude that $g_j = l'_j$. Therefore, continuing from $(**)$, we have:

$$\begin{aligned} \iint_{T_j(C)} (f(x, y) - l_j(x, y))^2 dA =& t_1^2 \iint_{T_j(C')} (xy - g_j(x, y))^2 dA \\ =& t_1^2 \iint_{T_j(C')} (xy - l'_j(x, y))^2 dA \end{aligned}$$

Since $j \in \{1,2,3,4\}$ was arbitrary, we may conclude that:

$$\begin{aligned}
\delta_C^{(D)}(F_0) &= \sum_{j\in\{1,3\}} \iint_{T_j(C)} (f(x,y) - l_j(x,y))^2 dA \\
&= \sum_{j\in\{1,3\}} t_1^2 \iint_{T_j(C')} (xy - l'_j(x,y))^2 dA \\
&= t_1^2 \sum_{j\in\{1,3\}} \iint_{T_j(C')} (xy - l'_j(x,y))^2 dA \\
&= t_1^2 \sum_{j\in\{2,4\}} \iint_{T_j(C')} (xy - l'_j(x,y))^2 dA && \text{(Lemma 1)} \\
&= \sum_{j\in\{2,4\}} t_1^2 \iint_{T_j(C')} (xy - l'_j(x,y))^2 dA \\
&= \sum_{j\in\{2,4\}} \iint_{T_j(C)} (f(x,y) - l_j(x,y))^2 dA \\
&= \delta_C^{(SD)}(F_0)
\end{aligned}$$

as desired. □

Notice from the definition of the Ethelo function that it can be described as $\eth(\mu, \Sigma) = c_1\mu\Sigma + c_2\mu + c_3\Sigma + c_4$ for some $c \in \mathbb{R}^4$ on the regions $[-1,1] \times (t,1]$, $[-1,0) \times [0,t)$, and $(0,1] \times [0,t)$ respectively. Thus, any cell that does not intersect with the line segments $[-1,1] \times \{t\}$, $\{0\} \times [0,t]$ will contribute the same error regardless whether they are split diagonally or skew-diagonally. Note that we can always choose the partitions $\{a_i\}_{i=1}^{d_1}$, $\{b_j\}_{j=1}^{d_2}$ in a way such that the total area of cells that intersects with at least one of $[-1,1] \times \{t\}$, $\{0\} \times [0,t]$ to be arbitrarily small, and hence the l_2-error contributed by these cells are arbitrarily small, which then means that the l_2-error for all triangulations $\mathbb{T}$ will be arbitrarily close to the minimum. Thus, we removed the part where we decide the triangulation from our experiments.

4.3.3 Dynamic Triangulation with Adjusted Function Values

Now we move on to the more general case where we decide both the triangulation $\mathbb{T}$ and the function values $F \in \mathbb{R}^V$ together. Following the notations in previous sections, we can

formulate the problem of finding $\bar{f}^{(\mathbb{T},F)}$ with minimal l_2-error as:

$$\begin{cases} \min & \sum_{C\in\mathcal{C}} z_C \\ s.t. & \{z_C = \delta_C^{(D)}(F), y_C = 0\} \vee \{z_C = \delta_C^{(SD)}(F), y_C = 1\} \quad \forall C \in \mathcal{C} \\ & z \in \mathbb{R}^{\mathcal{C}}, F \in \mathbb{R}^V, y \in \{0,1\}^{\mathcal{C}} \end{cases} \tag{AD_0}$$

where $\delta_T(F)$ follows the same definition as in the preceding subsections. The variables z_C encodes the squared l_2-error contributed by the cell C, and the binary variables y_C encodes whether the cell C is being split diagonally ($y_C = 0$) or skew-diagonally ($y_C = 1$). While (AD_0) is a disjunctive program, we note that if F can be restricted to a bounded region B and establish a bound $M_C \in \mathbb{R}$ for each cell $C \in \mathcal{C}$ such that $\delta_{T_1(C)}(F) + \delta_{T_3(C)}(F) \leq M_C$ and $\delta_{T_2(C)}(F) + \delta_{T_4(C)}(F) \leq M_C$ for all $C \in \mathcal{C}$, the above can be remodelled as a convex program using big-M constraints:

$$\begin{cases} \min & \sum_{C\in\mathcal{C}} z_C \\ s.t. & \delta_C^{(D)}(F) \leq z_C + M_C y_C & \forall C \in \mathcal{C} \\ & \delta_C^{(SD)}(F) \leq z_C + M_C(1 - y_C) & \forall C \in \mathcal{C} \\ & z \in \mathbb{R}^{\mathcal{C}}, F \in B \subseteq \mathbb{R}^V, y \in \{0,1\}^{\mathcal{C}} \end{cases} \tag{AD}$$

Now it remains to identify the set B and constants $M \in \mathbb{R}^{\mathcal{C}}$.

Recall from the properties 4.3.4, 4.3.5 that $\delta_T(F)$ is a convex quadratic function in F. From definition of $\delta_T(F)$, we see that $\delta_T(F)$ only depends on the values of $F_v : v \in v(T)$ when the triangle T is fixed. Thus, we can consider $\delta_T(F) = \delta_T(F_{v(T)})$ as a function of variables $F_v : v \in v(T)$.

Unlike previous subsections, we will assume in this subsection that $\delta_T(F_{v(T)})$ is *strictly* convex as a function of $F_{v(T)}$. To see that this assumption is reasonable, we note that since $\delta_T(F)$ is a convex quadratic function in F, $\delta_T(F_{v(T)})$ is also a convex quadratic function in $F_{v(T)}$. If $\delta_T(F_{v(T)})$ is convex but not strictly convex, there would be a line $\mathcal{L} \subseteq \mathbb{R}^{v(T)}$ in which all points are minimum points for $\delta_T(F_{v(T)})$, but this would be counter-intuitive since as $F_{v(T)}$ deviates away from the values of $\eth(v) : v \in v(T)$, the l_2-distance $\delta_T(F_{v(T)})$ between the target function f and the linear interpolation $L_{\{(v,F_v):v\in v(T)\}}$ over triangle T should approach infinity. Therefore, while we do not have a formal proof for the statement, it is reasonable for us to assume that $\delta_T(F) = \delta_T(F_{v(T)})$ is strictly convex for any triangle T in any triangulation $\mathbb{T}$. We also note that this assumption was not violated in any of our experiments.

Now, for strictly convex quadratic functions, we may observe the following properties:

Property 4.3.9. *Let $A \in \mathbb{R}^{n\times n}$ be positive definite, $b \in \mathbb{R}^n, c \in \mathbb{R}$, $M \in \mathbb{R}$ be arbitrary, and $\lambda_{min}(A)$ be the smallest eigenvalue of A. Then, we have $x^TAx + b^Tx + c \geq M$ in the following two cases:*

1. $\|b\|_1^2 - 4\lambda_{\min}(A) * (c - M) < 0$ *; or*
2. $x \in \mathbb{R}^n$ *and* $\|x\|_\infty \geq \frac{\|b\|_1+\sqrt{\|b\|_1^2-4\lambda_{\min}(A)*(c-M)}}{2\lambda_{\min}(A)}$

Proof. We note that $\lambda_{min}(A) > 0$ as A is positive definite, and:

$$\begin{aligned} x^TAx + b^Tx + c - M \geq& \lambda_{\min}(A)\|x\|_2^2 - \|b\|_1\|x\|_\infty + c \\ \geq& \lambda_{\min}(A)\|x\|_\infty^2 - \|b\|_1\|x\|_\infty + c - M \end{aligned}$$

By viewing the above as a quadratic polynomial in $\|x\|_\infty$, we see that if $\|b\|_1^2 - 4\lambda_{\min}(A) * (c - M) < 0$, then $x^TAx + b^Tx + c - M > 0$ for all x; otherwise, for any $\|x\|_\infty \geq \frac{\|b\|_1+\sqrt{\|b\|_1^2-4\lambda_{\min}(A)*(c-M)}}{2\lambda_{\min}(A)}$, we have $x^TAx+b^Tx+c-M > 0$. As such, the result follows. □

Property 4.3.10. *Let $A \in \mathbb{R}^{n\times n}$ be positive definite, $b \in \mathbb{R}^n, c \in \mathbb{R}$, and $r \in \mathbb{R}_+$ be arbitrary. Then, for any $x \in \mathbb{R}^n$ with $\|x\|_\infty \leq r$, we have:*

$$x^TAx + b^Tx + c \leq \sum_{i=1}^n \sum_{j=1}^n |A_{i,j}|r^2 + \|b\|_1 r + c$$

Proof.

$$\begin{aligned} x^TAx + b^Tx + c =& \sum_{i=1}^n \sum_{j=1}^n A_{i,j}x_ix_j + \sum_{i=1}^n b_ix_i + c \\ \leq& \sum_{i=1}^n \sum_{j=1}^n |A_{i,j}|r^2 + \sum_{i=1}^n |b_i|r + c \\ =& \sum_{i=1}^n \sum_{j=1}^n |A_{i,j}|r^2 + \|b\|_1 r + c \end{aligned}$$

□

To choose the region B and constants M_C in (AD_0), let $M_0 = \|f - \bar{f}^{(\bar{\mathbb{T}},\bar{F})}\|_2^2$ where $\bar{F}_v = f(v)$ for all $v \in V$ and $\bar{\mathbb{T}}$ is an arbitrary triangulation. Then, for triangulation $\mathbb{T}^*$ and function values F^* arising from optimal solution of (AD), we must have $\|f - \bar{f}^{(\mathbb{T}^*,F^*)}\|_2^2 \leq M_0$. Thus, we may choose B to be a box such that for any $F \in \mathbb{R}^V \setminus B$ there is always a cell $C \in \mathcal{C}$ such that both $\delta_C^{(D)}(F) \geq M_0$ and $\delta_C^{(SD)}(F) \geq M_0$, and then choose M_C to be an upper bound for both $\delta_C^{(D)}(F), \delta_C^{(SD)}(F)$ over $F \in B$.

Let $C \in \mathcal{C}$ be an arbitrary cell, and let $v(C)$ be the set of vertices for C. Then, since $\delta_{T_i(C)}(F_{v(T_i(C))})$ are strictly convex over $F_{v(T_i(C))}$ for all $i = 1, 2, 3, 4$, and $v(T_1(C)) \cup v(T_3(C)) = v(T_2(C)) \cup v(T_4(C)) = v(C)$, we see that $\delta_C^{(D)} = \delta_{T_1(C)} + \delta_{T_3(C)}$ and $\delta_C^{(SD)} = \delta_{T_2(C)} + \delta_{T_4(C)}$ are both strictly convex over $F_{v(C)}$. Also, since $\delta_{T_i(C)}(F)$ are quadratic functions in F given in closed form, so are $\delta_C^{(D)}$ and $\delta_C^{(SD)}$. As such, we can compute $A^{C,0}, A^{C,1} \in \mathbb{R}^{4\times4}, b^{(C,0)}, b^{(C,1)} \in \mathbb{R}^4, c_{(C,0)}, c_{(C,1)} \in \mathbb{R}$ such that:

$$\begin{aligned}\delta_C^{(D)}(F_{v(C)}) =& F_{v(C)}^T A^{C,0} F_{v(C)} + {b^{(C,0)}}^T F_{v(C)} + c_{(C,0)}\\ \delta_C^{(SD)}(F_{v(C)}) =& F_{v(C)}^T A^{C,1} F_{v(C)} + {b^{(C,1)}}^T F_{v(C)} + c_{(C,1)}\end{aligned}$$

Then, the program (AD) can be constructed with the following procedure:

1. Compute $M_0 = \|f - \bar{f}^{(\bar{\mathbb{T}},\bar{F})}\|_2^2$ where $\bar{F}_v = f(v)$ for all $v \in V$ and $\bar{\mathbb{T}}$ is an arbitrary triangulation.
2. For each cell $C \in \mathcal{C}$, compute $A^{C,0}, A^{C,1} \in \mathbb{R}^{4\times4}, b^{(C,0)}, b^{(C,1)} \in \mathbb{R}^4, c_{(C,0)}, c_{(C,1)} \in \mathbb{R}$ as defined above.
3. For $k = 0, 1$, let $r_{C,k} = real(\frac{\|b^{C,k}\|_1 + \sqrt{\|b^{C,k}\|_1^2 - 4\lambda_{\min}(A^{C,k})*(c-M)}}{2\lambda_{\min}(A^{C,k})})$, where $real(\cdot)$ denotes the real part of a complex number, and define $r_C = \max\{r_0, r_1\}$.
4. Compute $M_C = \max_{k=0,1} \sum_{i=1}^n \sum_{j=1}^n |A_{i,j}^{C,k}| r^2 + \|b^{C,k}\|_1 r + c_{(C,k)}$.
5. Construct program (AD) using M_C computed above and

$$B = \{F \in \mathbb{R}^V : \forall C \in \mathcal{C}, \forall v \in v(C), |F_v| \leq r_C\}$$

Similar to section 4.3.1, when using the Ethelo function $\eth$ as target function, we noticed that for triangulation $\mathbb{T}^*$ and function values F^* obtained by solving (AD), the error between optimal values of (EthP) and (MIP) tends to be larger when the optimal solutions

$x^*, \bar{x}$ to the two programs both falls in a same triangle $T \in \mathbb{T}^*$, of which the order of function values $\{F_v : v \in v(T)\}$ is different from that of the original values $\{\eth(v) : v \in v(T)\}$. In attempt to fix this, we also considered a program with extra constraints added to (AD) to enforce a partial order on the variables $F_v : v \in V$:

$$\begin{cases} \min & \sum_{C \in \mathcal{C}} z_C & \\ s.t. & \delta_C^{(D)}(F) \leq z_C + M_C y_C & \forall C \in \mathcal{C} \\ & \delta_C^{(SD)}(F) \leq z_C + M_C(1 - y_C) & \forall C \in \mathcal{C} \\ & F_u + \epsilon_c \leq F_v & \forall C \in \mathcal{C}, \forall u, v \in v(C), \eth(u) < \eth(v) \\ & z \in \mathbb{R}^{\mathcal{C}}, F \in B \subseteq \mathbb{R}^V, y \in \{0,1\}^{\mathcal{C}} \end{cases} \tag{AD-ϵ_0}$$

where $\epsilon_0 \geq 0$ is a pre-determined constant. The order of F_v between vertices in $v(C)$ for a cell C is strict when ϵ_0 is strictly bigger than 0; and when $\epsilon_0 = 0$, $F_u + \epsilon_0 \leq F_v$ simply requires that F_u is not greater than F_v.

4.4 Computational Results

4.4.1 Testing Environments

The test results in this section are obtained in the following 3 environments:

- "Server": The "server" is a Linux environment equipped with 256 GB RAM and 4 Intel(R) Xeon(R) Gold 6142 CPUs operating at 2.60GHz, which gives 64 cores in total. Tests carried out on this machine uses the Python interface of Gurobi 9.1.2, and are limited to use up to 4 threads at a time.
- "WSL": This environment is a "Windows Subsystem for Linux" in Windows 10, which operates on a machine equipped with 12GB RAM and a Intel(R) Core(TM) i7-6500U CPU operating at 2.50GHz. This gives 2 cores in total. The only tests that were ran on this environment are those that are done via Ethelo's engine, which uses BONMIN for solving MINLP.
- "PC": This is the Windows 10 environment on the same machine as WSL. This environment is only used for accessing the difference in performance of Server and the machine that hosts WSL.

While it would be ideal to have all tests running in the same environment, we were unable to do so for the tests in this section. Ethelo's engine needs to be ran under the WSL

environment, which the tests in this section cannot be ran on due to time constraints. We chose to run the tests under the Server environment for its capability of running several tests in parallel, but we could not run Ethelo's engine on the Server due to unresolved technical issue. The "PC" environment is used as a middle ground for comparing the performance of our program on the WSL environment, which is on the same machine as PC, and the Server environment.

4.4.2 Machine Comparison

Since the computational data came from different machines, it would be necessary to note the difference in performance of our code when being ran on different machines. In table 4.1, we re-ran one of our test from section 4.4.4 (setting `mid.cl.adj`) on both the Server and PC to compare the difference in average CPU Time. The "Ratio" rows are computed by dividing the average CPU runtime on server by that of the PC. The ratios that are less than 1 (ie. cases where the code runs faster on Server than on PC) are put in boldface.

As can be observed from the table, while there are cases with denser grid points that runs faster in terms of CPU Time on the server, the average CPU runtime tends to be slower on server.

4.4.3 Best-fitting Ethelo function

In this subsection, we present the results that were obtained by solving the programs presented in section 4.3, namely (FnCl), (FnCl-ϵ_0), (AD), (AD-ϵ_0), using the Ethelo function with parameters $\Xi = 0.5, t = 1/3$ as the target function. This means that we will be considering the domain $D = [-1, 1] \times [0, 1]$ in our formulations. The functions constructed in this subsection will also be used in subsequent tests for formulation (EthP-MI). The settings we tested are named in form `P.T.F`, where `P` defines the partitions $\{a_i\}_{i=1}^{d_1}, \{b_j\}_{j=1}^{d_2}$, `T` defines the triangulation $\mathbb{T}$, and `F` defines the construction of the function values $F \in \mathbb{R}^V$ when running procedure **GTConstruct**. We say that $(d_1 \times d_2)$ is the "grid size" of the approximation. We will define each of the 3 components of `P.T.F` below.

Recall that the partitions $\{a_i\}_{i=1}^{d_1}, \{b_j\}_{j=1}^{d_2}$ of $[a_l, a_u], [b_l, b_u]$ were assumed to be fixed throughout section 4.3, where the domain was $D = [a_l, a_u] \times [b_l, b_u] = [-1, 1] \times [0, 1]$.

For testing purpose, we considered the following two ways of generating the partitions $\{a_i\}_{i=1}^{d_1}, \{b_j\}_{j=1}^{d_2}$:

- eq: $a_i = -1 + 2(i-1)/(d_1-1)$, $b_j = (i-1)/(d_2-1)$ for $i \in [d_1], j \in [d_2]$. That is, $\{a_i\}_{i=1}^{d_1}, \{b_j\}_{j=1}^{d_2}$ partitions $[-1,1]$ and $[0,1]$ into intervals of equal length, respectively.
- mid: $b_j = (i-1)/(d_2-1)$ for $j \in [d_2]$. $d_1 = 2k$ for some integer $k > 1$, and

$$a_i = \begin{cases} -1 + (1-10^{-4}) \cdot \frac{i-1}{k-1} & \text{, if } i \le k \\ 10^{-4} + (1-10^{-4}) \cdot \frac{i-k}{k-1} & \text{, if } i > k \end{cases}$$

  That is, we construct $\{a_i\}_{i=1}^{d_1+1}$ by defining the small interval $[a_k, a_{k+1}] = [-10^{-4}, 10^{-4}]$ around 0, and partitions the two sides $[-1, -10^{-4}], [10^{-4}, 1]$ evenly. This is done to accommodate for Ethelo's discontinuity on $\{0\} \times [0,t]$.

For formulations (FnCl) and (FnCl-ϵ_0), the triangulation was also assumed to be fixed. We tested the two formulations under the following triangulation:

- "cl": Split a cell in a way that gives a closer approximation at the center of the cell. That is, given a cell $C = conv\left\{\begin{bmatrix} a_i \\ b_i \end{bmatrix}, \begin{bmatrix} a_i \\ b_{j+1} \end{bmatrix}, \begin{bmatrix} a_{i+1} \\ b_{j+1} \end{bmatrix}, \begin{bmatrix} a_{i+1} \\ b_j \end{bmatrix}\right\}$, we split it diagonally if

$$\begin{aligned} &\eth\left(\begin{bmatrix} a_i \\ b_{j+1} \end{bmatrix}\right) + \eth\left(\begin{bmatrix} a_{i+1} \\ b_j \end{bmatrix}\right) - \eth\left(\begin{bmatrix} (a_i + a_{i+1})/2 \\ (b_j + b_{j+1})/2 \end{bmatrix}\right) \\ \le &\eth\left(\begin{bmatrix} a_i \\ b_j \end{bmatrix}\right) + \eth\left(\begin{bmatrix} a_{i+1} \\ b_{j+1} \end{bmatrix}\right) - \eth\left(\begin{bmatrix} (a_i + a_{i+1})/2 \\ (b_j + b_{j+1})/2 \end{bmatrix}\right) \end{aligned}$$

  otherwise we split the cell skew-diagonally.

- "uj": An "union jack" [35] triangulation, where we split the bottom left cell (ie. the cell containing point $(a_1, b_1) = (-1, 0)$) diagonally, and then the remaining cells are split in a way that no two cells that share a common edge are split in the same way.

Then, regarding the ordering of vertices, we considered the following 4 settings when the triangulation is pre-determined:

- exact: Use exact function values $F_{(a_i,b_j)} = \eth(a_i, b_j)$ for all a_i, b_j instead of solving (FnCl) or (FnCl-ϵ_0).
- adj: Solve program (FnCl) for function values, without enforcing order between vertices.
- adj-O: Solve (FnCl-ϵ_0) with $\epsilon_0 = 0$, ie. enforce non-strict order between vertices.
- adj-S: Solve (FnCl-ϵ_0) with $\epsilon_0 = 10^{-5}$, ie. enforce strict ordering between vertices.

The "exact" setting is used for comparison only. Similar to above, when the triangulation is determined by the program (ie. for formulation (AD), (AD-ϵ_0)) we considered the following 3 settings:

- AT: "auto triangulation", solve program (AD) without enforcing ordering between vertices.
- AT-O: solve (AD-ϵ_0) with $\epsilon_0 = 0$.
- AT-S: solve (AD-ϵ_0) with $\epsilon_0 = 10^{-5}$.

For formulations (AD) and (AD-ϵ_0), the triangulation is not pre-determined, so we represented the triangulation with a backslash character "/".

The tests for the programs were run on the Server using Python interface of Gurobi 9.1.2, with parameters $ThreadCount = 4$, which limits the number of concurrent threads, and $TimeLimit = 7200$, which is the wallclock time limit in seconds. We presented the runtime data in table 4.2. For programs that did not terminate within time limit, we also included the relative gap on termination. The settings are named in form `P.T.F` as mentioned earlier. There is one entry (`mid./.AT`, grid size 40x20) where the relative gap could not be computed because the best bound upon termination is 0. In that case we presented the absolute gap instead.

From table 4.2, we observed that for settings where triangulation was pre-determined, ie. the settings `P.T.F` where `T` is either "cl" or "uj", can all be solved within 1 second wallclock time. These settings corresponds to the formulations (FnCl) and (FnCl-ϵ_0). The settings of form `P./.F`, which corresponds to the formulations (AD), (AD-ϵ_0), are much harder for Gurobi to solve. The runtime for solving the program can exceed time limit

for grid size as small as 12x6. Further, of the 19 cases that did not terminate within time limit, 11 has a gap of greater than 10% at termination, and 5 of them have final gap larger than 50%, both excluding the one case where the relative gap could not be computed.

To evaluate the effectiveness of (AD-ϵ_0), (AD), (FnCl-ϵ_0) and (FnCl) in reducing squared l_2-distance between the Ethelo function $\eth$ and the piecewise linear function $\bar{f}$ constructed by procedure **GTConstruct**, we denote $\bar{f}^{(d_1,d_2)}_{\texttt{P.T.F.}}$ to be the piecewise linear function constructed with setting `P.T.F.` and grid size $d_1 \times d_2$, and define the squared l_2-distances as $\|\bar{f}^{(d_1,d_2)}_{\texttt{P.T.F.}} - \eth\|_2^2$. The percentage reduction in squared l_2-error is given by $1-\bar{z}/z^*$, where $\bar{z} = \|\bar{f}^{(d_1,d_2)}_{\texttt{P.T.F.}} - \eth\|_2^2$ and:

- $z^* = \bar{f}^{(d_1,d_2)}_{\texttt{P.T.exact}}$, if `T` is not "/";
- $z^* = \bar{f}^{(d_1,d_2)}_{\texttt{P.cl.exact}}$, otherwise.

The numerical values of the squared l_2-errors are given in table 4.3, and the percentage reduction from table 4.4. We first note in table 4.3 that the squared distance for setting `mid./.AT` with grid size 22x11 is negative as reported by Gurobi, which is theoretically impossible. This shows that the computation for sufficiently small l_2-distances are susceptible to being dominated by feasibility tolerance in Gurobi. In table 4.3, all squared l_2-errors less than 10^{-6}, the default feasibility tolerance in Gurobi, are put in boldface. Then, from table 4.3, we noticed that the reduction percentages tend to decrease as the grid sizes get larger. This implies that the squared l_2-distances of `P.T.exact` tend to be closer to the optimal as the grids get finer.

4.4.4 Approximating multi-influence cases

In this subsection, we compare the performance of solving (MIP) with Gurobi against solving the original (EthP-MI) with BONMIN. There are 6 projects that were considered in this experiment, all of which involves only linear constraints and did not use auto-balance option (ie. x_{AB} was not involved, $m_1 = 0$):

- "BBB" : 91 variables, 12 XOR constraints.
- "carbon": 76 variables, 13 XOR constraints, 1 covering constraint, 1 knapsack constraint, and 13 XOR constraints.

- "citizen": 48 variables, 5 XOR constraints and 3 linear constraints.
- "granting": 50 variables, 1 knapsack constraint.
- "parks": 13 variables, 1 knapsack, 1 two-sided knapsack
- "stratford": 47 variables, 3 XOR constraints.

We generated 100 instances for each of the 6 projects by 1) selecting a random number N between 20 to 999 (inclusive), and then 2) from the votes collected for the project, randomly draw N times to form the new influence matrix, with possibly duplicated votes. We solve the instances under the following two settings, and compared their CPU Runtime:

1. Solve formulation (EthP-MI) with BONMIN, via Ethelo's engine under WSL environment.
2. Solve formulation (MIP) with Gurobi 9.1.2, under Server environment, with parameter $ThreadCount = 4$ and $TimeLimit = 600$. The piecewise linear function $\bar{f}$ is constructed in section 4.4.3 using the corresponding settings.

We note from section 4.4.2 that the CPU Time on server is not faster than the ones on PC environment, which is on the same machine as the WSL environment. Thus, we may treat the CPU Runtime on Server as an overestimate of the CPU runtime under WSL environment and compare it directly to BONMIN's CPU runtime on WSL.

In tables 4.5 and 4.6, we presented the average "relative error" for between Gurobi's solution and BONMIN's solution. The relative error for an instance is computed as follows: let x_{BON} be the solution returned by BONMIN, x_{GRB} be the solution returned by Gurobi, $z_{BON} = \eth(\mu(x_{BON}), \Sigma(x_{BON}))$ and $z_{GRB} = \eth(\mu(x_{GRB}), \Sigma(x_{GRB}))$ be the Ethelo scores of x_{BON}, x_{GRB} respectively. Then, the relative error is given by $(z_{BON} - z_{GRB})/z_{BON}$. The entries in table 4.5 are computed by taking average of the relative error over all 600 generated instances (100 instances for each of the 6 projects), and those in table 4.6 are given by taking the maximum over the 600 instances. The average relative errors for each of the projects are also attached in Appendix A.

From table 4.6, we noticed that the worst average relative error for settings `eq.T.F` tend to be greater than the ones for the same setting `mid.T.F` that uses "mid" partition instead. We also note that all settings `mid.cl.F` and `mid.uj.F` have a worst case average

relative error of less than 1%, and that settings `mid.cl.exact`, `mid.cl.adj` are the only two settings that obtained an worst case relative error of less than 0.2% on all tested grid sizes. Further, with regard to settings `mid./.F`, we noticed that by adding the constraints $F_u + \epsilon_0 \leq F_v$ to the programs (AD-ϵ_0) when computing the approximation, the tested instances with formulation (FnCl-ϵ_0) did not give significant improvement in worst-case relative error. In fact, the grid sizes 4x4, 8x4 are the only two where the settings with smallest worst-case error uses approximations obtained by (AD-ϵ_0), namely `mid./.AT` and `mid./.AT-O`.

From table 4.5, we see that the relative error of settings `mid.cl.F`, `mid.uj.F` are no greater than 0.06% when averaged over all 600 instances. In particular, `mid.cl.exact` and `mid.uj.exact` are the only two settings that achieves an average error of less than 0.01% over all tested grids. The average errors of settings `mid./.F` are no smaller than the worse of settings `mid.cl.exact` and `mid.uj.exact`.

In addition to the quality of solutions, we also compared the CPU runtime for computing the approximated solution with Gurobi versus computing the exact solution using BONMIN. In table 4.7, we presented the CPU Runtime of different settings averaged over all 600 instances. The settings with average CPU runtime longer than that of BONMIN are put in boldface, and the per-project average CPU runtime are attached in Appendix A. From table 4.7, we see that the settings `mid.cl.F`, `mid.uj.F` runs faster than BONMIN on average in all grid sizes. However, there is no single setting among the 8 that runs faster than one another on all grid sizes. The settings `mid./.F` also do not have a significant edge in runtime over the settings `mid.cl.exact` and `mid.uj.exact`. However, for the two grid sizes 4x4, 8x4 where `mid./.F` offers a better worst-case relative error than `mid.cl.exact` and `mid.uj.exact`, the average CPU runtimes are slightly faster than the latter settings, by up to 0.57 CPU seconds (comparing `mid./.AT-O` and `mid.cl.exact`).

4.4.5 Remark on Best Setting for Ethelo Function

In sum, from the above experiments, we believe that the settings `mid.cl.exact` and `mid.uj.exact` with grid sizes 4x4 to 32x16 are most suitable for generating the piecewise linear approximation that will be used in (EthP-MI), because 1) they reduced the average CPU runtime from 4.6 seconds to between 1.46s to 2.29s, which translates to 31.7% to 49.8% of the original; 2) they gave one of the lowest relative errors in our experiments, with an average error of $< 0.01\%$ and worst-case error of up to 0.17%; 3) they require minimal pre-processing when formulating (EthP-MI). This also indicates that l_2-error between the

original and approximated objective function does not serve as a good predictor for the quality of the approximated solution.

Proj		4x2	4x3	4x4	6x3	8x4	10x5	12x6	22x11	40x20	100x50	Average
BBB	Server	1.71	2.05	3.08	2.54	3.75	5.25	4.46	5.41	16.49	28.51	
	PC	1.64	2.06	2.64	2.07	2.68	2.71	3.57	3.78	9.18	28.88	
	Ratio	1.04	1.00	1.17	1.23	1.40	1.94	1.25	1.43	1.80	**0.99**	1.32
carbon	Server	0.55	0.59	0.60	0.58	0.69	0.60	0.62	1.00	1.97	6.65	
	PC	0.55	0.58	0.54	0.58	0.65	0.59	0.71	1.37	1.16	6.70	
	Ratio	1.00	1.02	1.11	1.00	1.06	1.02	**0.87**	**0.73**	1.70	**0.99**	1.05
citizen	Server	2.04	1.99	1.99	2.28	1.98	1.88	1.85	1.67	2.11	12.21	
	PC	0.55	0.57	0.58	0.58	0.61	0.60	1.02	1.77	2.99	12.22	
	Ratio	3.71	3.49	3.43	3.93	3.25	3.13	1.81	**0.94**	**0.71**	1.00	2.54
granting	Server	0.05	0.06	0.06	0.05	0.05	0.10	0.12	0.21	0.42	2.87	
	PC	0.02	0.02	0.02	0.02	0.02	0.08	0.16	0.24	0.41	2.88	
	Ratio	2.50	3.00	3.00	2.50	2.50	1.25	**0.75**	**0.88**	1.02	1.00	1.84
parks	Server	0.09	0.10	0.10	0.11	0.09	0.09	0.11	0.13	0.16	1.99	
	PC	0.05	0.04	0.05	0.05	0.05	0.05	0.06	0.12	0.23	1.98	
	Ratio	1.80	2.50	2.00	2.20	1.80	1.80	1.83	1.08	**0.70**	1.01	1.67
stratford	Server	1.87	1.71	1.65	1.86	1.73	2.08	1.67	1.03	0.94	8.55	
	PC	0.38	0.39	0.39	0.39	0.41	0.51	1.07	1.43	2.54	8.64	
	Ratio	4.92	4.38	4.23	4.77	4.22	4.08	1.56	**0.72**	**0.37**	**0.99**	3.02

Table 4.1: CPU Time comparison between server and PC environment

Setting		4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
mid.cl.adj	Time	0.03	0.02	0.01	0	0	0	0.01	0.01
mid.cl.adj-O	Time	0.03	0.03	0.03	0.01	0.02	0.03	0.06	0.10
mid.cl.adj-S	Time	0.02	0.02	0.01	0.01	0.02	0.03	0.08	0.10
mid.uj.adj	Time	0.01	0.02	0.02	0	0.01	0	0.01	0.02
mid.uj.adj-O	Time	0.03	0.03	0.01	0.01	0.02	0.04	0.07	0.11
mid.uj.adj-S	Time	0.03	0.03	0.02	0.01	0.02	0.06	0.07	0.13
mid./.AT	Time	0.49	1.20	8.74	7200.01	7200.02	1255.06	630.42	7200.06
	Gap	-	-	-	2.37	100	-	-	* 5.6e-08
mid./.AT-O	Time	0.24	0.43	1.55	7200	7200.01	22.63	9.50	5.40
	Gap	-	-	-	23.18	99.35	-	-	-
mid./.AT-S	Time	0.29	0.41	7.02	7200.01	7200.01	2573.34	9.99	6.41
	Gap	-	-	-	26.66	94.72	-	-	-
eq.cl.adj	Time	0	0.04	0	0	0	0	0.01	0.01
eq.cl.adj-O	Time	0	0.02	0.01	0.01	0.02	0.03	0.07	0.12
eq.cl.adj-S	Time	0.02	0.02	0.01	0.01	0.02	0.04	0.07	0.12
eq.uj.adj	Time	0.03	0.03	0.01	0	0	0	0.01	0.01
eq.uj.adj-O	Time	0.02	0.03	0.03	0.01	0.02	0.04	0.07	0.11
eq.uj.adj-S	Time	0.03	0.03	0.03	0.01	0.02	0.04	0.07	0.11
eq./.AT	Time	0.70	0.89	6.17	168.70	7200.01	7200.02	7200.06	7200.03
	Gap	-	-	-	-	16.75	60.50	18.54	59.69
eq./.AT-O	Time	0.37	0.58	1.31	4838.76	7200.01	7200.01	7200.01	7200.02
	Gap	-	-	-	-	1.25	1.41	8.67	6.81
eq./.AT-S	Time	0.31	0.66	0.91	1770.49	7200.01	7200.01	7200.04	7200.02
	Gap	-	-	-	-	1.59	3.03	11.73	21.56

Table 4.2: Wallclock Time and Gap for Approximating Ethelo Function

Settings	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
mid.cl.exact	3.71E-3	2.78E-3	3.11E-4	1.48E-4	6.48E-5	2.27E-5	2.37E-6	4.75E-6
mid.cl.adj	1.42E-3	7.89E-4	1.76E-4	8.09E-5	3.26E-5	1.19E-5	2.16E-6	3.40E-6
mid.cl.adj-O	2.04E-3	1.00E-3	1.83E-4	9.64E-5	3.37E-5	1.20E-5	2.16E-6	3.43E-6
mid.cl.adj-S	2.04E-3	1.00E-3	1.83E-4	9.65E-5	3.37E-5	1.20E-5	2.16E-6	3.43E-6
mid.uj.exact	5.30E-3	2.78E-3	3.11E-4	1.82E-4	6.48E-5	2.27E-5	2.37E-6	4.75E-6
mid.uj.adj	1.64E-3	7.27E-4	2.01E-4	1.12E-4	3.83E-5	1.35E-5	2.36E-6	3.38E-6
mid.uj.adj-O	1.76E-3	8.00E-4	2.09E-4	1.20E-4	3.95E-5	1.38E-5	2.36E-6	3.43E-6
mid.uj.adj-S	1.76E-3	8.00E-4	2.09E-4	1.20E-4	3.95E-5	1.38E-5	2.36E-6	3.43E-6
mid./.AT	1.23E-3	4.59E-4	9.47E-6	3.16E-5	1.01E-5	**-1.31E-7**	**0.00**	**5.61E-8**
mid./.AT-O	1.36E-3	4.94E-4	1.09E-5	4.62E-5	1.08E-5	1.36E-6	**0.00**	**0.00**
mid./.AT-S	1.37E-3	4.66E-4	9.13E-6	4.63E-5	1.16E-5	**1.06E-7**	**0.00**	**0.00**
eq.cl.exact	8.82E-3	7.87E-3	3.43E-3	1.91E-3	1.34E-3	9.21E-4	6.05E-4	4.81E-4
eq.cl.adj	7.46E-3	6.94E-3	2.99E-3	1.71E-3	1.20E-3	8.27E-4	5.45E-4	4.34E-4
eq.cl.adj-O	7.46E-3	6.94E-3	2.99E-3	1.71E-3	1.21E-3	8.38E-4	5.60E-4	4.47E-4
eq.cl.adj-S	7.46E-3	6.94E-3	2.99E-3	1.71E-3	1.21E-3	8.38E-4	5.60E-4	4.47E-4
eq.uj.exact	8.82E-3	7.87E-3	3.43E-3	1.94E-3	1.34E-3	9.21E-4	6.05E-4	4.81E-4
eq.uj.adj	7.27E-3	6.56E-3	2.91E-3	1.76E-3	1.21E-3	8.35E-4	5.49E-4	4.35E-4
eq.uj.adj-O	7.27E-3	6.56E-3	2.91E-3	1.77E-3	1.23E-3	8.47E-4	5.62E-4	4.49E-4
eq.uj.adj-S	7.27E-3	6.56E-3	2.91E-3	1.77E-3	1.23E-3	8.47E-4	5.62E-4	4.49E-4
eq./.AT	6.70E-3	6.08E-3	2.74E-3	1.64E-3	1.16E-3	8.06E-4	5.37E-4	4.21E-4
eq./.AT-O	6.73E-3	6.07E-3	2.74E-3	1.65E-3	1.17E-3	8.22E-4	5.51E-4	4.37E-4
eq./.AT-S	6.69E-3	6.08E-3	2.74E-3	1.65E-3	1.17E-3	8.21E-4	5.52E-4	4.36E-4

Table 4.3: Table for Squared l_2 errors
Entries less than 10^{-6} are put in boldface

Settings	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
mid.cl.adj	62%	72%	43%	45%	50%	48%	9%	28%
mid.cl.adj-O	45%	64%	41%	35%	48%	47%	9%	28%
mid.cl.adj-S	45%	64%	41%	35%	48%	47%	9%	28%
mid.uj.adj	69%	74%	35%	38%	41%	41%	0%	29%
mid.uj.adj-O	67%	71%	33%	34%	39%	39%	0%	28%
mid.uj.adj-S	67%	71%	33%	34%	39%	39%	0%	28%
mid./.AT	67%	83%	97%	79%	84%	101%	100%	99%
mid./.AT-O	63%	82%	96%	69%	83%	94%	100%	100%
mid./.AT-S	63%	83%	97%	69%	82%	100%	100%	100%
eq.cl.adj	15%	12%	13%	10%	10%	10%	10%	10%
eq.cl.adj-O	15%	12%	13%	10%	10%	9%	7%	7%
eq.cl.adj-S	15%	12%	13%	10%	10%	9%	7%	7%
eq.uj.adj	18%	17%	15%	9%	10%	9%	9%	10%
eq.uj.adj-O	18%	17%	15%	9%	8%	8%	7%	7%
eq.uj.adj-S	18%	17%	15%	9%	8%	8%	7%	7%
eq./.AT	24%	23%	20%	14%	13%	12%	11%	12%
eq./.AT-O	24%	23%	20%	14%	13%	11%	9%	9%
eq./.AT-S	24%	23%	20%	14%	13%	11%	9%	9%

Table 4.4: Squared l_2-error reduction for approximating Ethelo Function

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
mid.cl.exact	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
mid.cl.adj	0.05	0.02	0.00	0.01	0.00	0.00	0.00	0.00
mid.cl.adj-O	0.06	0.02	0.00	0.01	0.00	0.00	0.00	0.00
mid.cl.adj-S	0.06	0.02	0.00	0.01	0.00	0.00	0.00	0.00
mid.uj.exact	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
mid.uj.adj	0.06	0.01	0.00	0.01	0.01	0.00	0.00	0.00
mid.uj.adj-O	0.06	0.01	0.00	0.02	0.00	0.00	0.00	0.00
mid.uj.adj-S	0.06	0.01	0.00	0.02	0.00	0.00	0.00	0.00
mid./.AT	0.06	0.00	0.00	0.01	0.03	0.01	0.36	0.70
mid./.AT-O	0.05	0.00	0.00	0.00	0.01	0.53	**1.17**	**1.31**
mid./.AT-S	0.06	0.01	0.00	0.01	0.06	0.64	1.00	1.00
eq.cl.exact	0.24	0.20	0.25	0.04	0.04	0.03	0.00	0.00
eq.cl.adj	0.25	0.21	0.31	0.90	**5.96**	**5.50**	**1.44**	**5.82**
eq.cl.adj-O	0.25	0.21	0.31	0.93	**1.42**	0.67	0.90	**1.08**
eq.cl.adj-S	0.25	0.21	0.31	0.93	**1.42**	0.67	0.90	**1.08**
eq.uj.exact	0.24	0.20	0.25	0.04	0.04	0.03	0.00	0.00
eq.uj.adj	0.22	0.18	**1.22**	**5.08**	**5.99**	0.60	**5.49**	**5.80**
eq.uj.adj-O	0.22	0.18	**1.22**	**2.81**	**1.43**	0.52	0.91	**1.07**
eq.uj.adj-S	0.22	0.18	**1.22**	**2.37**	**1.42**	0.52	0.91	**1.07**
eq./.AT	0.24	0.64	0.99	**5.33**	**3.33**	**1.14**	**1.16**	**5.16**
eq./.AT-O	0.23	0.65	0.40	**3.15**	**1.83**	0.95	**2.82**	0.76
eq./.AT-S	0.60	0.19	0.40	**2.37**	**1.45**	0.67	**1.34**	**1.51**

Table 4.5: Average Percentage Relative Error over all instances

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
mid.cl.exact	0.18	0.17	0.17	0.11	0.03	0.16	0.05	0.07
mid.cl.adj	0.90	0.50	0.17	0.15	0.08	0.07	0.08	0.08
mid.cl.adj-O	0.90	0.50	0.17	0.15	0.08	0.07	0.08	0.08
mid.cl.adj-S	0.90	0.50	0.17	0.15	0.08	0.07	0.08	0.08
mid.uj.exact	0.18	0.17	0.17	0.11	0.09	0.11	0.02	0.03
mid.uj.adj	0.90	0.12	0.17	0.26	0.15	0.11	0.05	0.03
mid.uj.adj-O	0.90	0.13	0.17	0.29	0.15	0.11	0.05	0.03
mid.uj.adj-S	0.90	0.13	0.17	0.29	0.15	0.11	0.05	0.03
mid./.AT	0.90	0.08	0.08	0.12	0.60	0.50	**5.81**	**5.31**
mid./.AT-O	0.90	0.08	0.04	0.09	0.34	**6.53**	**5.28**	**7.93**
mid./.AT-S	0.90	0.16	0.12	0.12	**1.41**	**7.58**	**5.28**	**5.09**
eq.cl.exact	**3.62**	**2.91**	**3.62**	**1.89**	0.50	0.43	0.38	0.38
eq.cl.adj	**3.62**	**3.62**	**3.62**	**8.35**	**16.47**	**17.64**	**12.41**	**18.69**
eq.cl.adj-O	**3.62**	**3.62**	**3.62**	**8.35**	**10.49**	**6.98**	**7.04**	**7.79**
eq.cl.adj-S	**3.62**	**3.62**	**3.62**	**8.35**	**10.49**	**6.98**	**7.04**	**7.79**
eq.uj.exact	**3.62**	**2.91**	**3.62**	**1.89**	0.50	0.55	0.38	0.38
eq.uj.adj	**3.62**	**2.59**	**10.99**	**15.02**	**16.47**	**4.90**	**18.01**	**17.01**
eq.uj.adj-O	**3.62**	**2.59**	**10.99**	**14.20**	**10.75**	**4.10**	**8.03**	**7.64**
eq.uj.adj-S	**3.62**	**2.59**	**10.99**	**12.77**	**10.75**	**4.10**	**8.03**	**7.64**
eq./.AT	**3.62**	**5.60**	**8.63**	**15.02**	**12.48**	**11.53**	**8.59**	**15.80**
eq./.AT-O	**3.62**	**5.60**	**5.43**	**13.66**	**10.98**	**6.98**	**14.31**	**4.53**
eq./.AT-S	**5.60**	**2.81**	**5.43**	**12.77**	**10.75**	**6.98**	**10.31**	**7.79**

Table 4.6: Worst Percentage Relative Error over all instances

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
mid.cl.exact	1.34	1.60	1.60	1.72	1.75	1.78	2.29	3.24
mid.cl.adj	1.05	1.11	1.41	1.73	1.53	1.49	2.37	3.79
mid.cl.adj-O	1.05	1.12	1.43	1.80	1.49	1.46	2.30	3.76
mid.cl.adj-S	0.99	1.15	1.48	1.78	1.51	1.47	2.32	3.86
mid.uj.exact	1.28	1.46	1.58	1.71	1.85	1.83	1.92	3.13
mid.uj.adj	1.21	1.39	1.66	1.82	1.99	1.75	2.13	3.75
mid.uj.adj-O	1.02	1.26	1.46	1.91	1.78	1.68	2.16	3.91
mid.uj.adj-S	1.05	1.19	1.46	1.90	1.87	1.72	2.32	4.24
mid./.AT	1.13	1.14	1.15	1.26	1.23	1.47	2.14	**336.38**
mid./.AT-O	0.99	1.07	1.19	1.61	1.38	4.14	**339.59**	**339.54**
mid./.AT-S	0.89	1.15	1.34	1.35	2.47	3.36	**337.28**	**299.61**
eq.cl.exact	0.97	0.97	1.08	1.45	1.95	1.66	2.72	2.98
eq.cl.adj	0.82	0.87	0.86	1.52	**8.48**	4.33	**54.79**	**158.54**
eq.cl.adj-O	0.92	0.91	0.95	1.59	2.84	2.94	**45.37**	**155.16**
eq.cl.adj-S	0.86	0.94	0.92	1.64	2.77	3.09	**47.33**	**149.44**
eq.uj.exact	1.06	1.03	1.07	1.31	1.81	1.80	2.58	3.12
eq.uj.adj	1.11	1.11	2.22	**120.17**	**12.96**	3.27	**49.66**	**157.53**
eq.uj.adj-O	1.05	1.07	2.18	**10.41**	2.95	2.66	**44.14**	**147.94**
eq.uj.adj-S	1.04	1.06	2.17	**5.52**	3.19	2.79	**42.20**	**142.60**
eq./.AT	1.08	1.03	1.16	**85.48**	2.75	3.75	**57.26**	**167.41**
eq./.AT-O	0.97	0.93	0.95	**4.96**	3.08	**16.12**	**52.48**	**159.76**
eq./.AT-S	0.90	0.87	0.84	3.32	3.00	3.21	**54.82**	**137.78**

Table 4.7: Avg Runtime over all instances, Bonmin Avg = 4.60

Chapter 5

Conclusion

In this thesis, we attempted to improve the computational performance of Ethelo's group-decision making engine by applying tools from Operational Research. Ethelo used two MINLPs, namely the "single-influence" and "multi-influence" cases, for solving their group-decision making problem. For the single-influence cases, we made an observation that the formulations in all of the past projects provided by Ethelo can be re-posed as a MILP. By implementing the reformulation procedure and redirecting the resulting MILP to a specialized MILP solver, namely COIN-OR CBC, we reduced the average time spent in solving the single-influence MINLP by at least 87.9% in all of the provided projects. For the single-influence cases, we also identified a generalization of knapsack problem, which we named as two-sided multiple-choice knapsack problem, and attempted to prove the non-existence of new cuts in this problem. However, we only managed to derive a few sufficient conditions, and proved the statement to be true in one special case. On this front, more work can be done on proving or disproving our conjecture about the underlying polyhedral structure of the two-sided multiple-choice knapsack problem.

Regarding multi-influence cases, since the objective function was not continuous, we attempted to replace it with a piecewise linear function and apply results from the literature, mainly results by Huchette and Violma [19], and approximate the original MINLP with a MIQCP. We also derived two program formulations for finding piecewise linear functions with minimal l_2-distance to the target function, namely the objective function of the original MINLP, and that satisfies some special requirements so that the piecewise linear function to be used in formulating the MIQCP. We see from our computational experiment in section 4.4 that, while without theoretical guarantee, our MIQCP is capable of finding

a solution that is up to 0.17% worst than the solution provided by BONMIN, which is the original MINLP solver used by Ethelo, and has an average CPU runtime that is at least 50.2% faster than BONMIN on our testcases when using the settings recommended in section 4.4.5. This MIQCP approximation can be applied to any program of which the objective function lacks desirable properties for finding global optimal solutions, but more work is needed to derive a theoretical guarantee on quality of resulting approximated solution. To further improve the performance of Ethelo's engine, it may also be beneficial to study how the existing literature ([8], for example) on generating multiple optimal solutions in one branch-and-bound procedure can be generalized to Ethelo's multi-influence MINLPs.

References

[1] Warren P. Adams and Hanif D. Sherali. A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Science*, 32(10):1274–1290, 1986.

[2] David Austen-Smith and Jeffrey S. Banks. Information aggregation, rationality, and the Condorcet jury theorem. *The American Political Science Review*, 90(1):34–45, 1996.

[3] Egon Balas. Facets of the knapsack polytope. *Mathematical Programming*, 8(1):146 – 164, 1975.

[4] E. Beale and J. Tomlin. Special facilities in a general mathematical programming system for nonconvex problems using ordered sets of variables. *Operational Research*, 69:447–454, 01 1969.

[5] Zuse Institute Berlin. Polyhedron representation transformation algorithm. https://porta.zib.de/ (Accessed Sep 19, 2022).

[6] Pol Campos-Mercade. When are groups less moral than individuals? *Games and Economic Behavior*, 134:20–36, 2022.

[7] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Perfect Formulations*, pages 129–194. Springer International Publishing, Cham, 2014.

[8] Emilie Danna, Mary Fenelon, Zonghao Gu, and Roland Wunderling. Generating multiple solutions for mixed integer programming problems. In Matteo Fischetti and David P. Williamson, editors, *Integer Programming and Combinatorial Optimization*, pages 280–294, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[9] Ru-Xi Ding, Iván Palomares, Xueqing Wang, Guo-Rui Yang, Bingsheng Liu, Yucheng Dong, Enrique Herrera-Viedma, and Francisco Herrera. Large-scale decision-making: Characterization, taxonomy, challenges and future directions from an artificial intelligence and applications perspective. *Information Fusion*, 59:84–102, 2020.

[10] Raymond M. Duch and Albert Falcó-Gimeno. Collective decision-making and the economic vote. *Comparative Political Studies*, 55(5):757–788, 2022.

[11] Ethelo. Video "ethelo - park design". https://ethelo.com/videos/ (Accessed Sep 14,2022),.

[12] Ethelo. Website "what is ethelo?". https://ethelo.com/what-is-ethelo/ (Accessed Sep 14,2022).

[13] Michael R. Garey and David S. Johnson. *Computers and intractability*. A Series of Books in the Mathematical Sciences. W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness.

[14] Fred. Glover and Hanif D. Sherali. Second-order cover inequalities. *Mathematical Programming*, 114:207–234, 2008.

[15] Elif Ilke Gokce and Wilbert E. Wilhelm. Valid inequalities for the multi-dimensional multiple-choice 0–1 knapsack problem. *Discrete Optimization*, 17:25–54, 2015.

[16] Mark Gradstein, Shmuel Nitzan, and Jacob Paroush. Collective decision making and the limits on the organization's size. *Public Choice*, 66(3):279–291, 1990.

[17] P.L. Hammer, E.L. Johnson, and U.N. Peled. Facet of regular 0-1 polytopes. *Mathematical Programming*, 8(1):179 – 206, 1975.

[18] Xuan hua Xu, Zhi jiao Du, Xiao hong Chen, and Chen guang Cai. Confidence consensus-based model for large-scale group decision making: A novel approach to managing non-cooperative behaviors. *Information Sciences*, 477:410–427, 2019.

[19] Joey Huchette and Juan Pablo Vielma. Nonconvex piecewise linear functions: Advanced formulations and simple modeling tools. https://arxiv.org/abs/1708.00050, 2017.

[20] Joey Huchette and Juan Pablo Vielma. A geometric way to build strong mixed-integer programming formulations. https://arxiv.org/abs/1811.10409, 2018.

[21] George L. Nemhauser Juna Pablo Vielma. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Mathematical Programming*, 128:49–72, 2011.

[22] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *The Multiple-Choice Knapsack Problem*, pages 317–347. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[23] Soung Hie Kim and Byeong Seok Ahn. Interactive group decision making procedure under incomplete information. *European Journal of Operational Research*, 116(3):498–507, 1999.

[24] Leo Liberti. Undecidability and hardness in mixed-integer nonlinear programming. *RAIRO - Operations Research*, 53, 05 2018.

[25] Kenneth O. May. A set of independent necessary and sufficient conditions for simple majority decision. *Econometrica*, 20(4):680–684, 1952.

[26] Garth P. McCormick. Computability of global solutions to factorable nonconvex programs: Part i — convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.

[27] R. Ramanathan and L.S. Ganesh. Group preference aggregation methods employed in ahp: An evaluation and an intrinsic process for deriving members' weightages. *European Journal of Operational Research*, 79(2):249–265, 1994.

[28] John Richardson. An algorithm-based approach for resolving complex group decision problems fairly. https://ethelo.com/wp-content/uploads/2019/01/Ethelo-White-Paper.pdf=AOvVaw26YzYbFkiVORU9DrLTOvfl (Accessed Sep 14, 2022), 2019.

[29] John Richardson. Methods and systems for conducting surveys and processing survey data to generate a collective outcome, U.S. Patent 9 727 883, Aug 2017.

[30] Hanif D. Sherali and Warren P. Adams. *Reformulation-Linearization Techniques for Discrete Optimization Problems*, pages 479–532. Springer US, Boston, MA, 1998.

[31] Hanif D. Sherali and Fred Glover. Higher-order cover cuts from zero–one knapsack constraints augmented by two-sided bounding inequalities. *Discrete Optimization*, 5(2):270–289, 2008. In Memory of George B. Dantzig.

[32] Hanif D. Sherali and Youngho Lee. Sequential and simultaneous liftings of minimal cover inequalities for generalized upper bound constrained knapsack polytopes. *SIAM Journal on Discrete Mathematics*, 8(1):133–153, 1995.

[33] Hanif D. Sherali and Cihan H. Tuncbilek. New reformulation linearization/convexification relaxations for univariate and multivariate polynomial programming problems. *Operations Research Letters*, 21(1):1–9, 1997.

[34] Cédric Sueur, Christophe Bousquet, Romain Espinosa, and Jean-Louis Deneubourg. Improving human collective decision-making through animal and artificial intelligence. 1:e59, 12 2021.

[35] Michael J. Todd. Union Jack triangulations. In *Fixed points: algorithms and applications (Proc. First Internat. Conf., Clemson Univ., Clemson, S.C., 1974)*, pages 315–336. Academic Press, New York, 1977.

[36] Juan Pablo Vielma and George L. Nemhauser. Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Math. Program.*, 128(1-2, Ser. A):49–72, 2011.

[37] Laurence A. Wolsey. Valid inequalities for 0–1 knapsacks and mips with generalised upper bound constraints. *Discrete Applied Mathematics*, 29(2):251–261, 1990.

APPENDICES

Appendix A

Relative Gap and CPU Runtime data for multi-influence tests

We present the per-project results of our multi-influence tests from section 4.4.4 here. For the relative error tables, namely tables A.1,A.2,A.3,A.4,A.5, and A.6, an entry of "-" denotes an average error of less than 0.01%.

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
.cl.exact	0.01	-	-	-	-	-	-	-
mid.cl.adj	0.06	0.02	-	-	-	-	-	-
mid.cl.adj-O	0.07	0.02	-	-	-	-	-	-
mid.cl.adj-S	0.07	0.02	-	-	-	-	-	-
mid.uj.exact	0.01	-	-	-	-	-	-	-
mid.uj.adj	0.06	-	-	-	-	-	-	-
mid.uj.adj-O	0.07	0.01	-	-	-	-	-	-
mid.uj.adj-S	0.07	0.01	-	-	-	-	-	-
mid./.AT	0.07	-	-	0.01	0.03	-	0.07	**3.53**
mid./.AT-O	0.06	-	-	-	0.02	-	**3.47**	**3.48**
mid./.AT-S	0.06	0.01	-	-	0.03	-	**3.46**	**3.09**
eq.cl.exact	0.17	0.15	0.18	0.20	0.21	0.18	-	-
eq.cl.adj	0.17	0.16	0.18	0.20	0.20	0.23	0.69	**1.21**
eq.cl.adj-O	0.17	0.16	0.18	0.20	0.21	0.23	0.67	**1.20**
eq.cl.adj-S	0.17	0.16	0.18	0.20	0.21	0.23	0.67	**1.20**
eq.uj.exact	0.17	0.15	0.18	0.20	0.21	0.21	-	-
eq.uj.adj	0.16	0.14	0.17	0.18	0.20	0.26	0.69	**1.21**
eq.uj.adj-O	0.16	0.14	0.17	0.19	0.20	0.26	0.67	**1.20**
eq.uj.adj-S	0.16	0.14	0.17	0.19	0.20	0.25	0.67	**1.19**
eq./.AT	0.17	0.31	0.17	0.19	0.27	0.26	0.69	**1.21**
eq./.AT-O	0.17	0.31	0.29	0.20	0.21	0.23	0.67	**1.20**
eq./.AT-S	0.30	0.15	0.29	0.20	0.21	0.25	0.67	**1.20**

Table A.1: Average Percent Rel. Error for buildbackbetter

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
.cl.exact	-	-	-	-	-	-	-	-
mid.cl.adj	-	-	-	-	-	-	-	-
mid.cl.adj-O	-	-	-	-	-	-	-	-
mid.cl.adj-S	-	-	-	-	-	-	-	-
mid.uj.exact	-	-	-	-	-	-	-	-
mid.uj.adj	-	-	-	-	-	-	-	-
mid.uj.adj-O	-	-	-	-	-	-	-	-
mid.uj.adj-S	-	-	-	-	-	-	-	-
mid./.AT	-	-	-	-	-	-	0.12	0.05
mid./.AT-O	-	-	-	-	-	**2.42**	-	0.03
mid./.AT-S	-	-	-	-	-	0.04	0.01	**2.26**
eq.cl.exact	-	-	-	-	-	-	-	-
eq.cl.adj	-	-	-	-	**15.24**	**16.40**	0.16	0.08
eq.cl.adj-O	-	-	-	-	0.06	0.06	-	-
eq.cl.adj-S	-	-	-	-	0.06	0.07	-	-
eq.uj.exact	-	-	-	-	-	-	-	-
eq.uj.adj	-	-	0.37	**13.31**	**15.24**	-	0.16	0.16
eq.uj.adj-O	-	-	0.37	**5.55**	0.07	-	-	-
eq.uj.adj-S	-	-	0.38	**2.94**	0.07	-	-	-
eq./.AT	-	-	0.07	**13.31**	0.12	0.15	0.16	-
eq./.AT-O	-	-	-	**7.56**	0.06	**1.63**	0.09	**1.76**
eq./.AT-S	-	-	-	**2.86**	0.06	0.06	**1.76**	-

Table A.2: Average Percent Rel. Error for carbon

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
.cl.exact	-	-	-	-	-	-	-	-
mid.cl.adj	-	-	-	-	-	-	-	-
mid.cl.adj-O	-	-	-	-	-	-	-	-
mid.cl.adj-S	-	-	-	-	-	-	-	-
mid.uj.exact	-	-	-	-	-	-	-	-
mid.uj.adj	-	-	-	-	-	-	-	-
mid.uj.adj-O	-	-	-	-	-	-	-	-
mid.uj.adj-S	-	-	-	-	-	-	-	-
mid./.AT	-	-	-	-	-	-	-	-
mid./.AT-O	-	-	-	-	-	0.03	**1.39**	-
mid./.AT-S	-	-	-	-	-	**1.05**	**1.35**	-
eq.cl.exact	-	-	-	-	-	-	-	-
eq.cl.adj	-	-	-	0.05	**7.40**	0.74	-	**14.44**
eq.cl.adj-O	-	-	-	0.05	**1.47**	-	-	-
eq.cl.adj-S	-	-	-	0.05	**1.48**	-	-	-
eq.uj.exact	-	-	-	-	-	-	-	-
eq.uj.adj	-	-	**1.09**	**7.38**	**7.40**	-	**13.78**	**14.45**
eq.uj.adj-O	-	-	**1.09**	**2.21**	**1.32**	-	-	-
eq.uj.adj-S	-	-	**1.11**	**2.18**	**1.28**	-	-	-
eq./.AT	-	0.01	0.30	**7.40**	**7.38**	-	-	**14.45**
eq./.AT-O	-	0.01	-	**2.22**	**1.67**	-	**2.21**	-
eq./.AT-S	-	-	-	**2.21**	**1.44**	-	-	**1.36**

Table A.3: Average Percent Rel. Error for citizen

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
.cl.exact	-	-	-	-	-	-	-	-
mid.cl.adj	0.01	-	-	-	-	-	-	-
mid.cl.adj-O	0.02	-	-	-	-	-	-	-
mid.cl.adj-S	0.02	-	-	-	-	-	-	-
mid.uj.exact	-	-	-	-	-	-	-	-
mid.uj.adj	0.02	-	-	-	-	-	-	-
mid.uj.adj-O	0.02	-	-	-	-	-	-	-
mid.uj.adj-S	0.02	-	-	-	-	-	-	-
mid./.AT	0.02	-	-	-	-	-	0.06	0.42
mid./.AT-O	0.01	-	-	-	-	0.71	**1.05**	0.09
mid./.AT-S	0.02	-	-	-	-	**2.30**	0.95	0.41
eq.cl.exact	0.03	0.03	-	-	-	-	-	-
eq.cl.adj	0.04	0.03	0.36	**4.33**	**10.69**	**11.83**	**2.64**	**13.47**
eq.cl.adj-O	0.04	0.03	0.36	**4.53**	**4.53**	0.06	-	0.08
eq.cl.adj-S	0.04	0.03	0.36	**4.54**	**4.53**	0.06	-	0.08
eq.uj.exact	0.03	0.03	-	-	-	-	-	-
eq.uj.adj	0.03	0.03	**4.39**	**8.72**	**10.70**	-	**13.16**	**13.28**
eq.uj.adj-O	0.03	0.03	**4.39**	**8.08**	**4.65**	-	-	-
eq.uj.adj-S	0.03	0.03	**4.39**	**8.08**	**4.65**	-	-	-
eq./.AT	0.03	0.11	**4.06**	**8.76**	**8.62**	**2.59**	0.08	**9.58**
eq./.AT-O	0.03	0.11	-	**8.08**	**6.01**	0.17	**8.87**	0.42
eq./.AT-S	0.10	0.03	-	**8.08**	**4.65**	0.05	0.64	**1.23**

Table A.4: Average Percent Rel. Error for granting

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
.cl.exact	-	-	-	-	-	-	-	-
mid.cl.adj	-	-	-	-	-	-	-	-
mid.cl.adj-O	-	-	-	-	-	-	-	-
mid.cl.adj-S	-	-	-	-	-	-	-	-
mid.uj.exact	-	-	-	-	-	-	-	-
mid.uj.adj	-	-	-	-	-	-	-	-
mid.uj.adj-O	-	-	-	-	-	-	-	-
mid.uj.adj-S	-	-	-	-	-	-	-	-
mid./.AT	-	-	-	-	-	-	-	-
mid./.AT-O	-	-	-	-	-	-	0.79	0.12
mid./.AT-S	-	-	-	-	0.12	-	-	-
eq.cl.exact	-	-	-	-	-	-	-	-
eq.cl.adj	-	-	-	-	-	-	-	-
eq.cl.adj-O	-	-	-	-	-	-	-	-
eq.cl.adj-S	-	-	-	-	-	-	-	-
eq.uj.exact	-	-	-	-	-	-	-	-
eq.uj.adj	-	-	-	-	-	-	-	-
eq.uj.adj-O	-	-	-	-	-	-	-	-
eq.uj.adj-S	-	-	-	-	-	-	-	-
eq./.AT	-	-	-	**1.46**	**1.25**	0.01	0.99	-
eq./.AT-O	-	-	-	-	0.69	-	-	0.27
eq./.AT-S	-	-	-	-	-	-	-	-

Table A.5: Average Percent Rel. Error for parks

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
.cl.exact	0.04	0.01	0.01	0.01	-	0.01	-	-
mid.cl.adj	0.23	0.08	0.02	0.03	0.01	-	-	-
mid.cl.adj-O	0.27	0.09	0.01	0.05	0.01	-	-	-
mid.cl.adj-S	0.27	0.09	0.01	0.05	0.01	-	-	-
mid.uj.exact	0.04	0.01	0.01	0.01	0.01	-	-	-
mid.uj.adj	0.25	0.03	0.02	0.06	0.03	-	-	-
mid.uj.adj-O	0.27	0.03	0.01	0.08	0.03	-	-	-
mid.uj.adj-S	0.27	0.03	0.01	0.08	0.03	-	-	-
mid./.AT	0.26	0.02	0.01	0.03	0.12	0.07	**1.89**	0.20
mid./.AT-O	0.23	-	-	0.01	0.06	0.01	0.31	**4.15**
mid./.AT-S	0.25	0.03	-	0.02	0.20	0.44	0.22	0.23
eq.cl.exact	**1.24**	**1.00**	**1.29**	0.02	-	-	-	-
eq.cl.adj	**1.27**	**1.06**	**1.32**	0.79	**2.22**	**3.83**	**5.16**	**5.72**
eq.cl.adj-O	**1.27**	**1.06**	**1.32**	0.79	**2.22**	**3.67**	**4.70**	**5.24**
eq.cl.adj-S	**1.27**	**1.06**	**1.32**	0.79	**2.22**	**3.67**	**4.70**	**5.23**
eq.uj.exact	**1.24**	**1.00**	**1.29**	0.02	-	-	-	-
eq.uj.adj	**1.15**	0.90	**1.30**	0.87	**2.37**	**3.35**	**5.16**	**5.73**
eq.uj.adj-O	**1.15**	0.90	**1.30**	0.84	**2.33**	**2.85**	**4.79**	**5.23**
eq.uj.adj-S	**1.15**	0.90	**1.30**	0.84	**2.33**	**2.84**	**4.79**	**5.23**
eq./.AT	**1.23**	**3.41**	**1.31**	0.88	**2.37**	**3.85**	**5.05**	**5.73**
eq./.AT-O	**1.19**	**3.45**	**2.12**	0.84	**2.33**	**3.67**	**5.05**	0.91
eq./.AT-S	**3.23**	0.95	**2.13**	0.84	**2.33**	**3.68**	**4.94**	**5.25**

Table A.6: Average Percent Rel. Error for stratford

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
.cl.exact	2.95	4.26	4.25	4.71	4.54	5.44	9.36	14.83
mid.cl.adj	2.16	2.68	4.46	5.67	4.36	5.01	10.00	**18.00**
mid.cl.adj-O	2.01	2.60	4.39	5.72	4.42	4.95	9.82	**17.81**
mid.cl.adj-S	2.05	2.64	4.39	5.72	4.36	4.92	9.78	**18.42**
mid.uj.exact	2.90	4.23	4.26	4.84	5.39	6.14	7.38	14.11
mid.uj.adj	2.01	3.10	4.35	5.93	5.70	6.03	8.83	**17.21**
mid.uj.adj-O	2.00	3.15	4.36	6.58	6.09	6.28	9.19	**18.07**
mid.uj.adj-S	1.97	3.15	4.73	6.96	6.30	6.49	10.05	**19.83**
mid./.AT	2.30	3.51	3.71	3.91	3.34	5.33	3.86	**2007.17**
mid./.AT-O	2.14	3.65	3.79	5.35	3.70	15.21	**2025.94**	**2026.82**
mid./.AT-S	2.14	3.27	4.20	4.01	10.19	6.57	**2010.53**	**1774.09**
eq.cl.exact	1.27	1.30	1.23	2.95	5.66	4.82	11.38	12.17
eq.cl.adj	1.33	1.36	1.30	3.43	6.02	10.74	**315.37**	**922.31**
eq.cl.adj-O	1.31	1.30	1.29	3.40	5.95	9.22	**261.24**	**916.63**
eq.cl.adj-S	1.31	1.37	1.29	3.39	6.18	9.73	**273.27**	**882.83**
eq.uj.exact	1.31	1.33	1.25	2.36	5.49	5.51	10.69	13.39
eq.uj.adj	1.26	1.35	1.27	2.65	5.30	10.15	**273.81**	**918.67**
eq.uj.adj-O	1.29	1.38	1.29	2.77	6.17	9.02	**254.85**	**875.83**
eq.uj.adj-S	1.28	1.37	1.28	2.74	5.86	9.46	**243.09**	**843.70**
eq./.AT	1.29	1.43	1.28	3.92	6.74	13.73	**330.64**	**972.16**
eq./.AT-O	1.32	1.47	1.49	4.25	6.53	10.30	**290.01**	**920.78**
eq./.AT-S	1.42	1.39	1.52	3.38	6.56	10.58	**314.21**	**801.85**

Table A.7: Average CPU Time for buildbackbetter, Bonmin Avg = 15.68s

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
.cl.exact	0.60	0.60	0.68	0.74	0.79	1.17	0.87	1.16
mid.cl.adj	0.62	0.62	0.81	0.69	0.74	1.15	0.89	1.18
mid.cl.adj-O	0.61	0.61	0.79	0.69	0.75	1.15	0.89	1.19
mid.cl.adj-S	0.61	0.61	0.80	0.69	0.75	1.15	0.90	1.18
mid.uj.exact	0.59	0.57	0.71	0.64	0.75	0.90	0.93	1.11
mid.uj.adj	0.57	0.59	0.79	0.64	0.74	0.82	0.85	1.54
mid.uj.adj-O	0.60	0.62	0.81	0.65	0.77	0.84	0.90	**1.60**
mid.uj.adj-S	0.60	0.62	0.83	0.68	0.81	0.89	0.94	**1.68**
mid./.AT	0.62	0.62	0.71	0.67	0.75	0.76	**1.58**	**3.67**
mid./.AT-O	0.61	0.62	0.70	0.74	0.85	**6.58**	0.95	1.49
mid./.AT-S	0.62	0.63	0.79	0.64	0.88	**1.85**	1.18	**18.10**
eq.cl.exact	0.59	0.58	0.56	0.83	0.85	0.89	1.02	1.52
eq.cl.adj	0.62	0.62	0.59	1.20	**37.79**	**2.21**	1.25	**2.30**
eq.cl.adj-O	0.62	0.61	0.58	1.20	1.15	1.05	1.14	**2.46**
eq.cl.adj-S	0.62	0.62	0.59	1.22	1.15	1.04	1.14	**2.24**
eq.uj.exact	0.61	0.60	0.56	0.74	0.77	0.87	0.90	1.19
eq.uj.adj	0.58	0.58	**4.34**	**711.71**	**64.29**	1.00	**1.71**	1.54
eq.uj.adj-O	0.59	0.60	**4.43**	**49.98**	1.51	1.02	1.04	**1.91**
eq.uj.adj-S	0.59	0.58	**4.39**	**20.86**	1.52	1.00	1.02	**1.86**
eq./.AT	0.58	0.56	0.79	**503.61**	**2.51**	1.00	1.44	**2.64**
eq./.AT-O	0.61	0.58	0.59	**17.11**	1.29	**78.26**	1.20	**28.53**
eq./.AT-S	0.57	0.63	0.59	**8.28**	**2.10**	**1.92**	**3.08**	**2.70**

Table A.8: Average CPU Time for carbon, Bonmin Avg = 1.57s

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
.cl.exact	2.36	2.43	2.30	2.40	2.48	2.25	2.13	2.00
mid.cl.adj	1.73	1.69	1.64	1.92	1.93	1.57	1.86	2.14
mid.cl.adj-O	1.90	1.96	1.87	2.09	2.01	1.52	1.63	2.13
mid.cl.adj-S	1.52	1.87	1.83	2.12	1.89	1.50	1.81	2.13
mid.uj.exact	2.12	1.78	2.26	2.24	2.37	2.24	1.72	2.02
mid.uj.adj	2.32	2.42	2.45	2.04	2.70	2.11	1.82	2.19
mid.uj.adj-O	1.73	1.90	1.87	1.95	1.86	1.67	1.69	2.18
mid.uj.adj-S	1.84	1.68	1.53	1.66	1.94	1.65	1.74	2.27
mid./.AT	1.82	1.39	1.32	1.32	1.57	1.50	3.92	4.87
mid./.AT-O	1.57	0.87	1.42	1.76	1.79	1.64	**9.21**	4.86
mid./.AT-S	1.39	1.58	1.51	1.49	1.90	**9.87**	**9.76**	3.08
eq.cl.exact	1.79	1.81	2.34	2.35	2.43	2.42	2.51	2.70
eq.cl.adj	1.48	1.65	1.71	2.25	3.89	**9.61**	5.78	**20.20**
eq.cl.adj-O	1.79	1.83	1.99	2.52	6.72	4.29	4.84	5.59
eq.cl.adj-S	1.42	1.83	2.01	2.57	6.20	4.69	4.69	5.49
eq.uj.exact	2.14	2.06	2.33	2.22	2.04	2.62	2.32	2.61
eq.uj.adj	2.44	2.38	5.25	3.70	3.87	4.03	**16.57**	**18.30**
eq.uj.adj-O	2.25	2.19	5.05	6.83	**7.32**	3.16	4.36	4.89
eq.uj.adj-S	2.15	2.23	5.02	6.70	**8.01**	3.62	4.39	5.09
eq./.AT	2.27	2.04	3.16	2.83	3.60	3.50	5.52	**22.65**
eq./.AT-O	1.91	1.53	1.92	6.14	**7.47**	4.67	**18.23**	5.90
eq./.AT-S	1.44	1.54	1.55	6.00	6.35	3.55	5.93	**15.77**

Table A.9: Average CPU Time for citizen, Bonmin Avg = 7.09s

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
.cl.exact	0.04	0.04	0.05	0.13	0.15	0.21	0.26	0.35
mid.cl.adj	0.04	0.04	0.05	0.14	0.14	0.20	0.28	0.37
mid.cl.adj-O	0.05	0.05	0.05	0.14	0.14	0.21	0.28	0.37
mid.cl.adj-S	0.05	0.04	0.05	0.14	0.14	0.20	0.28	0.37
mid.uj.exact	0.05	0.05	0.05	0.12	0.14	0.20	0.26	0.37
mid.uj.adj	0.05	0.05	0.06	0.14	0.16	0.22	0.24	0.43
mid.uj.adj-O	0.04	0.04	0.05	0.13	0.16	0.22	0.25	0.44
mid.uj.adj-S	0.04	0.05	0.05	0.13	0.16	0.23	0.26	0.47
mid./.AT	0.04	0.04	0.05	0.10	0.12	0.17	0.35	0.48
mid./.AT-O	0.04	0.04	0.04	0.15	0.15	0.25	0.38	0.45
mid./.AT-S	0.05	0.04	0.04	0.11	0.17	0.39	0.33	0.44
eq.cl.exact	0.04	0.04	0.08	0.12	0.16	0.26	0.33	0.43
eq.cl.adj	0.04	0.04	0.10	0.18	0.24	0.33	0.47	**0.64**
eq.cl.adj-O	0.05	0.04	0.10	0.17	0.20	0.30	0.44	**0.62**
eq.cl.adj-S	0.04	0.04	0.09	0.18	0.20	0.31	0.44	0.59
eq.uj.exact	0.04	0.04	0.08	0.12	0.16	0.23	0.34	0.45
eq.uj.adj	0.05	0.05	0.11	0.20	0.25	0.29	**0.60**	0.52
eq.uj.adj-O	0.05	0.05	0.12	0.19	0.28	0.36	0.48	0.54
eq.uj.adj-S	0.05	0.05	0.12	0.19	0.22	0.30	0.48	0.55
eq./.AT	0.05	0.04	0.10	0.24	0.31	0.42	**0.65**	0.52
eq./.AT-O	0.04	0.04	0.06	0.17	0.26	0.34	**0.61**	**0.71**
eq./.AT-S	0.04	0.04	0.06	0.17	0.24	0.35	**0.67**	**0.64**

Table A.10: Average CPU Time for granting, Bonmin Avg = 0.60s

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
.cl.exact	0.09	0.09	0.10	0.10	0.11	0.12	0.13	0.15
mid.cl.adj	0.07	0.08	0.09	0.10	0.09	0.11	0.16	0.18
mid.cl.adj-O	0.08	0.08	0.09	0.10	0.09	0.11	0.16	0.18
mid.cl.adj-S	0.07	0.08	0.09	0.10	0.09	0.11	0.16	0.18
mid.uj.exact	0.08	0.08	0.10	0.11	0.10	0.10	0.11	0.15
mid.uj.adj	0.09	0.08	0.11	0.11	0.13	0.15	0.15	0.18
mid.uj.adj-O	0.07	0.08	0.09	0.10	0.11	0.13	0.14	0.17
mid.uj.adj-S	0.07	0.08	0.09	0.10	0.11	0.14	0.15	0.18
mid./.AT	0.08	0.07	0.09	0.10	0.10	0.10	0.11	0.21
mid./.AT-O	0.07	0.07	0.08	0.10	0.10	0.09	0.06	0.18
mid./.AT-S	0.07	0.08	0.08	0.11	0.09	0.09	0.14	0.16
eq.cl.exact	0.07	0.07	0.11	0.10	0.11	0.11	0.11	0.25
eq.cl.adj	0.07	0.08	0.09	0.08	0.10	0.09	0.15	0.23
eq.cl.adj-O	0.07	0.07	0.09	0.08	0.11	0.10	0.14	0.25
eq.cl.adj-S	0.07	0.08	0.10	0.10	0.11	0.10	0.14	0.25
eq.uj.exact	0.09	0.08	0.11	0.11	0.10	0.11	0.12	0.25
eq.uj.adj	0.09	0.08	0.11	0.11	0.13	0.10	0.14	0.25
eq.uj.adj-O	0.08	0.09	0.11	0.10	0.14	0.14	0.14	0.26
eq.uj.adj-S	0.09	0.09	0.11	0.10	0.13	0.11	0.14	0.27
eq./.AT	0.09	0.09	0.10	0.11	0.13	0.13	0.21	0.26
eq./.AT-O	0.07	0.08	0.10	0.10	0.07	0.09	0.14	**0.37**
eq./.AT-S	0.09	0.08	0.09	0.10	0.11	0.11	0.14	0.27

Table A.11: Average CPU Time for parks, Bonmin Avg = 0.34s

Setting	4x3	4x4	8x4	12x6	16x8	22x11	32x16	40x20
.cl.exact	1.97	2.20	2.22	2.27	**2.45**	1.47	1.01	0.95
mid.cl.adj	1.69	1.59	1.38	1.85	1.91	0.90	1.06	0.88
mid.cl.adj-O	1.67	1.43	1.39	2.04	1.55	0.84	1.04	0.87
mid.cl.adj-S	1.66	1.68	1.72	1.90	1.84	0.93	1.02	0.88
mid.uj.exact	1.99	2.05	2.07	2.32	**2.36**	1.42	1.10	0.99
mid.uj.adj	2.22	2.08	2.18	2.05	**2.51**	1.16	0.86	0.96
mid.uj.adj-O	1.66	1.79	1.56	2.02	1.68	0.96	0.79	0.99
mid.uj.adj-S	1.77	1.55	1.55	1.86	1.91	0.93	0.82	1.02
mid./.AT	1.93	1.23	1.04	1.49	1.52	0.95	**3.03**	1.87
mid./.AT-O	1.53	1.16	1.10	1.55	1.69	1.10	1.03	**3.46**
mid./.AT-S	1.03	1.34	1.42	1.76	1.63	1.37	1.71	1.81
eq.cl.exact	2.06	2.04	2.17	**2.35**	**2.49**	1.47	0.96	0.82
eq.cl.adj	1.39	1.50	1.36	1.96	**2.85**	**3.01**	**5.73**	**5.58**
eq.cl.adj-O	1.67	1.60	1.63	2.18	**2.90**	**2.67**	**4.44**	**5.43**
eq.cl.adj-S	1.68	1.69	1.43	**2.36**	**2.76**	**2.67**	**4.29**	**5.24**
eq.uj.exact	2.14	2.08	2.09	2.31	2.30	1.47	1.13	0.83
eq.uj.adj	2.25	2.22	2.20	**2.64**	**3.90**	**4.04**	**5.13**	**5.93**
eq.uj.adj-O	2.04	2.10	2.10	**2.57**	2.31	2.26	**3.96**	**4.21**
eq.uj.adj-S	2.09	2.02	2.10	**2.54**	**3.42**	2.27	**4.10**	**4.15**
eq./.AT	2.19	2.01	1.51	2.20	**3.21**	**3.73**	**5.12**	**6.24**
eq./.AT-O	1.88	1.88	1.55	2.01	**2.83**	**3.05**	**4.71**	2.29
eq./.AT-S	1.81	1.54	1.22	1.99	**2.64**	**2.72**	**4.91**	**5.43**

Table A.12: Average CPU Time for stratford, Bonmin Avg = 2.33s