FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Minimizing Wiggles in Storyline Visualizations

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Masterstudium Software Engineering & Internet Computing

eingereicht von

Theresa Fröschl, BSc
Matrikelnummer 00826841

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Assoc.Prof. Dipl.-Inform. Dr.rer.nat. Martin Nöllenburg

Wien, 4. September 2018

Theresa Fröschl   Martin Nöllenburg

Minimizing Wiggles in Storyline Visualizations

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Masterstudium Software Engineering & Internet Computing

by

Theresa Fröschl, BSc
Registration Number 00826841

to the Faculty of Informatics

at the TU Wien

Advisor: Assoc.Prof. Dipl.-Inform. Dr.rer.nat. Martin Nöllenburg

Vienna, 4th September, 2018

Theresa Fröschl   Martin Nöllenburg

Erklärung zur Verfassung der Arbeit

Theresa Fröschl, BSc
Ernsdorf 108, 2134 Staatz-Kautendorf

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 4. September 2018

Theresa Fröschl

Kurzfassung

Storyline Visualizations sind einfache 2-dimensionale Illustrationen über die Handlungen von Filmen oder Büchern die nur aus den horizontal verlaufenden Lebenslinien der Charaktere der Geschichte und den Interaktionen der einzelnen Charaktere miteinander bestehen. Diese Interaktionen stehen im Fokus einer Storyline Visualization.

Aktuelle wissenschaftliche Arbeiten haben ihren Fokus auf der Optimierung von Storyline Visualizations durch die Minimierung von Kreuzungen zwischen den Linien der Charaktere. Diese Diplomarbeit knüpft an diese Vorarbeit an und behandelt vertikale Sprünge entlang der Charakterlinien. Solche Höhensprünge werden im Zusammenhand mit Storyline Visualizations als *Wiggles* bezeichnet. Ziel ist es die Qualität und Lesbarkeit von Storyline Visualizations zu verbessert indem Wiggles minimiert werden. Um diese Wiggles zu reduzieren werden zuerst Optimierungs-Metriken definiert, um die Qualität einer Storyline Visualization in Zahlen ausdrücken zu können. Insgesamt haben wir vier solcher Metriken: *die Gesamthöhe aller Wiggles (total wiggle height)*, *die Anzahl aller Wiggles (number of wiggles)*, *die Höhe des höchsten Wiggles (highest wiggle)* und *die Anzahl aller paarweisen Kreuzungen (number of pairwise crossings)*. Diese Metriken bilden die Grundlage für die Zielfunktionen von Wiggle-Minimierungen.
Mit einer kombinatorischen Sichtweise auf unser Optimierungsproblem beschreibt diese Arbeit zwei Lösungsmodelle um die vertikale Anordnung der Charakterlinien für jeden Zeitpunkt zu variieren um die beste Lösung mit den kleinsten möglichen Werten für unsere Metriken zu produzieren. Das erste Modell beschreibt ein ganzzahliges lineares Programm (ILP) für Wiggle-Minimierung, das zweite Modell beinhaltet die Formulierung eines maximalen Erfüllbarkeitsproblems (Max-SAT) zur Wiggle-Minimierung.

Basierend auf unseren beiden Lösungsmodellen wurden Experimente durchgeführt um die unterschiedlichen Variationen der möglichen Zielfunktionen bezüglich Wiggle-Minimierung auszutesten. Die Evaluierung dieser Experimente zeigt deutlich, dass die Minimierung der Gesamthöhe aller Wiggles die besten Resultate bezüglich unserer Metriken erzielt. Die Minimierung von Wiggles reduziert auch die Anzahl der Kreuzungen innerhalb einer Storyline Visualization. Durch eine kombinierte Optimierung von Wiggles und Kreuzungen kann die Anzahl der Kreuzungen weiter reduziert werden, allerdings meist auf Kosten der anderen Wiggle-Metriken.

Abstract

Storyline visualizations are abstract 2-dimensional drawings of storylines, which shows the lifelines of the characters of the storyline as horizontal lines and interactions between those characters by bundles of the corresponding character lines. The focus lies on illustrating the interactions of the characters within the storyline.

Recent scientific work had its focus on optimizing storyline visualizations by minimizing crossings. This thesis examines line wiggles as another important quality criteria of storyline visualizations and how such wiggles can be minimized to generate storyline visualizations with good legibility. To minimize wiggles we start defining optimization metrics for *total wiggle height*, *number of wiggles*, *highest wiggle* and *number of pairwise crossings* which can be used to formulate objective functions for optimizations of storyline visualizations.
Considering our optimization problem as a combinatorial problem we formulate two solution models for optimizing storyline visualizations based on changing the vertical order of the character lines to minimize one or more of our optimization metrics. The first solution model contains an Integer Linear Programming (ILP) approach; the second solution model describes a Max-SAT formulation.

Following experiments based on our two solution models with different objective formulations show that the minimization of the total wiggle height produces the best results regarding our four optimization metrics and that wiggle minimizations also reduces crossing within a storyline visualization. With multi-objectives for combined minimization of wiggles and crossings the number of crossings can be further reduced, but very likely at the expense of at least one of the wiggle metric values.

Contents

CHAPTER 1

Introduction

The aim of storyline visualizations is to give an overview of all interactions between the entities of a storyline. The plot of the storyline comes second and arises only from the context of the visualization.
Recent scientific research about storyline visualizations was mainly inspired by a webcomic by Munroe about movie narrative charts [26] shown in Figure 1.1.

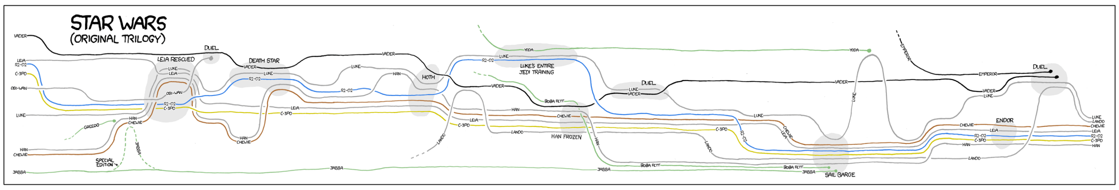


Figure 1.1: Movie narrative chart of *Star Wars* (xkcd 657)

Following the style of the webcomic by Monroe such storyline visualizations are 2-dimensional drawings of a set of x-monotone curves going from left to right along a timeline. Every character of the storyline is represented by one of the horizontal lines. The focus lies on the interactions between the characters, which we refer to as meetings of characters. A meeting of a subset of all characters is illustrated by vertical proximity of the corresponding character lines.

To obtain storyline visualizations with good legibility where it is easy to follow every character line the vertical ordering of the lines along the timeline is crucial. The current literature names three quality criteria for storyline visualizations to identify and generate storyline visualisations with good legibility. These quality criteria are crossings, line wiggles and white-space gaps [23, 36]. Therefore optimal storyline visualizations should contain minimal amounts of crossings, wiggles and white space.

After several attempts in the literature about optimizing only the number of crossings in storyline visualizations [17, 21, 37, 38] this thesis focus on minimizing the wiggles in storyline visualizations. In other words, instead of focusing on the minimization of crossings between two or more character lines we focus on minimizing unnecessary vertical jumps of the character lines between two successive time points within a storyline visualization.

In the following thesis we start by defining storyline visualizations and wiggles in more detail, followed by the definition of four optimization metrics for *total wiggle height*, *number of wiggles*, *highest wiggle* and *number of crossings* which we will use to formulate concrete objective functions for wiggle minimization and for evaluating the quality of resulting storyline visualizations.

By looking at our optimization problem from a combinatorial point of view we create two solution models for minimizing wiggles. Our first solution model contains an *Integer Linear Programming (ILP)* model and for our second optimization method we create a *Maximum Satisfiability (Max-SAT)* formulation. Both solution models include comparable variables and constraints to describe correct storyline visualizations and enable us to formulate objective functions for at least two different wiggle minimization objectives. Our first wiggle minimization objective contains the minimization for the total wiggle height, which is simply the sum of the height of all wiggles in the complete storyline visualization. Our second wiggle minimization objective contains the minimization of the number of wiggles, which is represented by the sum of all wiggle occurrences in the storyline visualization.
Furthermore the ILP formulation extends these two wiggle minimization objectives with a third wiggle objective function for minimizing the maximum wiggle height and the additional possibility of formulating multi-objectives. Such a multi-objective combines one of the three wiggle minimization objectives with the minimization of the number of pairwise crossings within storyline visualizations.

By analysing the results from experiments based on our two solutions models we evaluate the generated storyline visualizations from both solution models. Our performed experiments comprise all our different objective alternatives for wiggle minimization, which enables us to compare the quality of the resulting storyline visualizations regarding our optimization metrics produced by different objective variations.

The thesis is structured the following way:

- Chapter 2 is about the basic definition of the research problem this thesis examines and provides a detailed definition of storyline visualizations and wiggles in storyline visualizations. The chapter also contains the definition of our optimization metrics we use for formulating objective functions for wiggle minimization and for evaluating the quality of storyline visualizations.

- Chapter 3 summarizes previous scientific research work about storyline visualizations in the style of the webcomic by Monroe [26] and comparable visualizations which have their focus on illustrating relations between entities along a timeline. The chapter also examines the current state of the art in the literature regarding optimizations of storyline visualizations.
- Chapter 4 gives a short overview of the theoretical foundation needed for our two solution models in the next chapter.
- Chapter 5 starts with a general section about all required variables and constraints for both of our solution models so that they produce correct and comparable storyline visualizations. Furthermore this chapter contains also a detailed description of the concrete ILP and Max-SAT formulations.
- Chapter 6 names all technologies used for the implementation of both solution models and gives a short overview of the created software project.
- Chapter 7 contains the description and analysis of all experiments performed for both of our solutions models. The chapter contains also an evaluation of the storyline visualization results from different alternative objective functions for optimizing storyline visualizations.
- Chapter 8 summarizes the evaluation of wiggle minimization in storyline visualizations based on our two solution models.

CHAPTER 2

Problem Definition

2.1 Definition of Storyline Visualizations

The kind of storyline visualizations we examine in this thesis is inspired by the webcomic of movie narrative charts by Monroe [26]. Figure 1.1 shows a part of this webcomic. Such a storyline visualization is a 2-dimensional drawing with focus on the interactions of the entities of the storyline over time. These entities are illustrated by x-monotone curves. The interactions are represented by vertical proximity of the corresponding entity curves.

In the webcomic these entities are the characters of the movie and the interactions are the scenes the associated characters share. To distinguish between different character lines, colours of different shades and labels at the beginning of the lines are used.

We will follow this style of storyline visualizations in the following chapters and refer to the x-monotone curves of the visualization as *character lines* and to the interactions as *meetings* between a set of characters which we refer to as *members* of the meeting. An example of storyline visualizations within this thesis is shown in Figure 2.1.

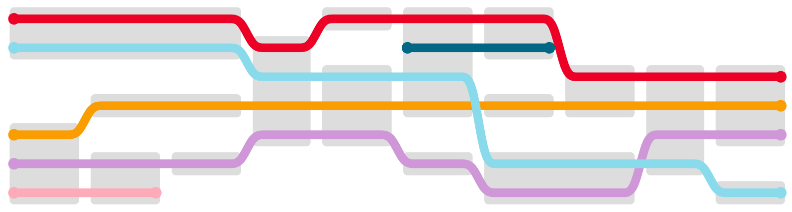


Figure 2.1: Example of a storyline visualization

Because we use storylines of movies for our experiments in Chapter 7 we will refer to movie data as instances for generating storyline visualizations where a meeting represents a scene in the movie containing at least one character.

Meetings are identifiable by bundles of character lines and by the light grey coloured boxes we draw behind the bundled character lines. Parallel meetings which share time points are possible and allowed, because although only one scene can be shown at once on a screen, all other characters which are not shown in the current scene don't just stop existing. Most of the time it is known from the context of a storyline where all characters are currently and which other characters are with them. To visually separate such parallel meetings we require at least one blank line between the meetings. A character is only allowed to be member of at most one meeting per time point.

The start time point of a character can be derived by the start time point of the first meeting along the timeline the character is a member of, and the end time point of a character by the last meeting the character is a member of. We allow gaps along the lifeline of the characters for storylines where the whereabouts of certain characters is not known from the context of the storyline. During these gaps the character line ends and starts again when the corresponding character restarts appearing in meetings.

In optimized storyline visualizations the character lines are driven to be as straight as possible. An optimal character line is represented as a straight horizontal line. It is natural in a movie or any other narrative that characters interact with various other characters. Also a meeting can't continue forever and has to end at a certain point along the timeline. Then the bundled characters have to separate from each other to form new bundles for new meetings. Thus after every end of a meeting at least one character has to break out from its horizontal flow to join one ore more other characters for the start of a new meeting. Such a beak out forms a *wiggle*.

2.2 Definition of Wiggles

A wiggle is a change of direction along a character line which start at a specific time point and ends at the very next time point. For this short period the horizontal character line becomes a rising or dropping line. After this quick jump along the y-axis the line finds back into it's horizontal flow.

Such wiggles can be observed for one line alone, or for a bunch of parallel lines together. In both cases the behaviour stays the same. Storyline visualizations with many wiggles can degrade the legibility of the visualizations because it gets harder to follow single lines when they constantly change their directions.

In the literature different perspectives and presentations can be found for wiggles. Not always is there a visual difference between straight horizontal line segments and the rising or dropping line segments depending on the goal of the corresponding literature work.

A high contrast in visual characteristics between wiggles and horizontal line segments can be found in Tanahashi and Ma [36] and Liu et al. [23] where wiggles are drawn as thin and lightly coloured lines connecting horizontal static line segments of the storyline visualization. With this different representation wiggles are more seen as connecting elements of interaction sessions and less part of the character lifelines.

A different point of view of line wiggles where the focus lies more on emphasizing wiggles is described in the works of Bereg et al. [9, 10] where wiggles are named as moves within a line of a tangle. A tangle describes a sequence of permutations where each element is only allowed to change its position by at most 1 between two consecutive permutations. The goal is to get all elements ordered at the end of the sequence. The moves get illustrated as thickened lines and both the beginning and the ending corner of a move got marked with circles around them.

Another important point to mention is that it is usually recommended in line visualizations to smooth corners to enhance the readability of the visualization. Then the corners appear more rounded and the border between the horizontal line and the wiggle becomes a blur. With this it is easier for the human eye to follow the flow of the lines despite the changes of directions [9, 36, 40].

For our storyline visualizations we will keep the style of the line during the rising or dropping line pieces and round the corners at the start and end of a wiggle. During a meeting we forbid the appearance of wiggles, which reduces wiggles for a bundle of lines together.

2.3 Optimization metrics

To be able to evaluate storyline visualizations and compare different solutions for the same storyline instance with each other we have to define metrics to describe the quality of storyline visualizations in numbers. These metrics should describe the properties of storyline visualizations and enfold the optimization criteria about crossings, line wiggles and white-space gaps from Tanahashi and Ma [36] and Liu et al. [23] and further criteria used for wiggle minimizations in this thesis.

For our optimization metrics we use the same properties of storyline visualizations as we will use for defining concrete objective functions for our optimizations later. Thus we define the following four metrics on which we will determine the quality of the experiment results in Section 7:

twh *total wiggle height*: sum of the height of all wiggle of all characters

hw *height of highest wiggle*: the height of the highest wiggle in the complete visualization

wc *wiggle count*: number of all wiggles of all characters

cc *pairwise crossings count*: sum of all crossings between two characters

In the following chapters we will focus only on this four metrics to describe and evaluate storyline visualizations.

2.4 Expected Results

The aim of this thesis is to investigate the minimization of line wiggles in storyline visualizations, which can be formulated into the following research questions:

Q1 *Which methods can be used to optimize storyline visualizations by minimizing wiggles and how can an appropriate solution model be formulated to automatically generate storyline visualizations with minimized wiggles?*

Q2 *What effect has the minimization of wiggles in storyline visualization on the other optimization metrics and how does the quality of such visualizations differ from similar ones with additional crossings minimization?*

To answer the first research question we will formulate two different solution models based on the same theoretical variables and constraints. For the choice of the solution models we will take two different methods, which were already used for optimizing storyline visualizations by minimizing crossings.
For the second research question we will discuss the results from experiments based on the two solution models. In addition we will use our optimization metric cc to construct a multi-objective function were we minimize also crossings next to wiggles. With such a multi-objective we want to investigate if it is necessary to optimize multiple metrics at once, or if it is sufficient to focus only on one metric to obtain good quality results of storyline visualizations.

2.5 Contribution

The contribution of this thesis to scientific research is to investigate the minimization of line wiggles in storyline visualizations. The minimization of wiggles complements the state of the art literature regarding the minimization of crossings in storyline visualizations, the theoretical discussion of wiggle minimization and the combined optimizations of storyline visualizations for multiple optimization metrics.

A poster on our work describing the ILP model for total wiggle height minimization from Section 5.2 has been published at the 25th International Symposium on Graph Drawing and Network Visualization (GD 2017).[1]

[1] Theresa Fröschl and Martin Nöllenburg. Minimizing Wiggles in Storyline Visualizations. In *International Symposium on Graph Drawing and Network Visualization (GD 2017)*, volume 10692 of LNCS, pages 585—587. Springer, 2017.

CHAPTER 3

Literature Review

There are different approaches in the literature to define, compute, optimize and represent storyline visualizations. A new boom of scientific research regarding storyline visualizations was started by a webcomic by Monroe [26] shown in Figure 1.1. Research work inspired by this narrative movie chart had its focus mainly on creating optimal visualizations of data containing relationships between entities connected with a timeline.

Before the webcomic by Monroe there has already been scientific research about drawings of temporal data containing a timeline as base component. *LifeLines* by Plaisant et al. [30] was probably one of the first with focus on visualizing dynamic temporal data for a static diagram. The goal with *LifeLines* was to visualize personal history of medical records and to provide a possibility to present and explore this data within one diagram. With this kind of drawings it should also be possible to visualize different personal history data like court records, biographical data or professional history. The authors continued their work and developed an interactive tool written in Java to enable the user to explore real clinical data in more detail [31].

A few years later it was the webcomic by Monroe [26] that inspired research work to focus on optimizing storyline visualizations detached from a specific temporal data set or a specific context of the data set. The definition of such storyline visualizations got more generic and although storyline data from movies and books were still used to draw result images, it was no longer the goal to find a suitable visualization technique for a specific context. The new goal was to optimize the visualization of information that was based on points in time and where entities exist and interact with each other along this timeline. Certain papers had their focus more on visualization criteria and others more on computing optimized storyline visualizations based on defined metrics.

Ogawa and Ma [28] picked up the idea of storyline visualization from Monroe to create a static picture of version control system data, more specifically the interactions between the developers over time and how they worked together in their software projects. The authors defined general rules for the aesthetic layout of their visualizations. This rules included the clustering of developers during the interactions, the visual separation of different clusters, the reduction of position changes along the y-axis of the tubes representing the developers and the reduction of crossings between the tubes. The authors also got inspirations for their visualizations from metro map drawings and therefore decided to use bold colors and thick lines for the developers.

In the same year Kim et al. [20] created *TimeNets* to visualize relationships in families over time, also partly motivated by Monroe. Here the goal was to have a different view of genealogical data than with ancestor charts. With an additional timeline complex relationships like divorce, remarriage, out-of-wedlock births, and polygamy in families and their development could be visualized. *TimeNets* showed families more as networks of relationships than as trees. The individual members of the family were visualized as horizontal lines, which converged to show marriage and diverged again if they divorce. Vertical connections between the entities were used to illustrate parent-child-relationships.

Based on the work of Ogawa and Ma [28], Tanahashi and Ma [36] continued the discussion of general aspects of storyline visualizations. The authors discussed the main aspects of well-designed storyline visualizations and defined quality criteria for storyline visualizations which lead to the formulation of the three main optimization metrics for crossings, line wiggles and white-space gaps. All later attempts of optimizing storyline visualizations in other papers are leaned upon this optimization metrics. Furthermore the paper describes the data of a storyline as a set of interaction sessions. Each of this interaction sessions is described by a starting time point, a concrete duration and a set of characters. These interaction sessions were then used as input for a genetic algorithm that was composed of three computation steps. First the interaction sessions were laid out, followed by rearranging the lines. The last step included the removal of white space. The result contained the computed layout of the interaction sessions.

In a successor work Tanahashi et al. [34] used the theoretical foundation from Tanahashi and Ma [36] and continued their research work by constructing a framework for creating storyline visualizations from streaming data where the constant data flow was dealt with and the storyline visualization was updated accordingly to the data flow.

Meanwhile Chen et al. [13] followed a very different approach for visualizing storylines. Like the movie narrative chart by Monroe [26] the goal here was to summarize the complete storyline of a movie with only one static picture which should be visually pleasing and informative. But differently from the storyline visualizations discussed in this thesis the authors used video shots to summarize the content of the movie and merged them together into one picture. The results of such an automatic storyline extraction looks very different to our storyline visualizations.

StoryFlow by Liu et al. [23] describes a storyline visualization system for a stepwise generation of storyline visualizations with regards to the optimization metrics from Tanahashi and Ma [36]. For the optimizations *StoryFlow* followed a hybrid strategy of discrete and continuous optimization methods in a layout pipeline containing four steps. In the first step a relationship tree got generated, then the relationships and lines got ordered following by an alignment and the last steps included a layout compaction. Moreover *StoryFlow* allowed real-time user interactions to manipulate the storyline visualization manually and enabled the users to add and delete entities, interaction sessions or locations, change the position of entities and interaction sessions, straightening the character lines and bundle a group of interaction sessions together. With these user interactions the user could influence the automatically generated storyline visualizations.

Muelder et al. [25] used storyline representations for visualizing large dynamic network graphs. The authors focused here on a bottom-up approach together with a static timeline so that the user could explore the dynamic data and select different foci for the visualization.

Later works started laying their focus mainly on minimizing crossings for optimizing storyline visualizations and neglected the other optimization criteria of line wiggles and white-space gaps in favor of minimizing crossing. Furthermore the optimization problem was more considered from a combinatorial optimization point of view. Kostitsyna et al. [21] sought to find the minimum number of crossings necessary in a particular storyline visualization. Therefore an event graph illustrating the interaction sessions was used where the vertices represented the characters and the edges represented the interactions between the characters. With this event graph the authors defined a lower bound for crossings and provided a formal prove for the NP-hardness of the crossing minimization problem in storyline visualizations.

Gronemann et al. [17] followed the approach of Kostitsyna et al. [21] of focusing on the minimization of crossings and formulated an integer linear programming (ILP) model for crossing minimization of storyline visualizations. A base for this ILP model was formed with the multi-layer crossing minimization (MLCM) problem under tree constraints (MLCM-TC) where every time point was constructed as layer for all characters active at this time point. The goal of this ILP model was to find a permutation for every layer such that the crossings of the lines connecting consecutive layers were at its minimum.

A slightly different problem definition for crossing minimizations can be found in van Dijk et al. [37] where the authors examined the minimization of block crossings. A block crossing in storyline visualizations is defined as a crossing of two arbitrary large sets of parallel character lines. The paper describes the structure of a storyline as hypergraph and defines a group hypergraph for groups of characters, which have at least one meeting together. The authors defined a greedy algorithm and an approximation algorithm to find permutations for the groups of characters over all time points with a minimized amount of block crossings.

Van Dijk et al. [38] continued the work from van Dijk et al. [37] for minimizing block crossings with a SAT-based algorithm and improved their results. For the SAT-based approach they described the optimization goal for a storyline S with an integer λ for which a solution with a permutation sequence of exactly λ elements had to be found. By minimizing the number for λ an optimal solution with a minimal amount of block crossings could be found.

Dang et al. [14] developed the visualization technique *TimeArcs* that also had its focus on relationships between entities during a specific time period. With *TimeArcs* users should be enabled to recognize patterns over time with regards to changing relationships between the entities. The evolutions of the entities over time were visualized and clustering of the corresponding entities also highlighted the relationships. These two characteristics were included in the storyline visualizations discussed in this thesis as well. *TimeArcs* supported also user interactions so that one data set could be viewed with different levels of detail. One main difference of *TimeArcs* to storyline visualizations after Monroe [26] was that *TimeArcs* enabled concurrent relationships of entities, thus one entity could be included in multiple relationships at the same time point.

Ogawa and Ma [28] and Tanahashi and Ma [36] named wiggles as optimization criteria for storyline visualizations. But aside from the theoretical description and the discussion about suitable layouts for wiggles there is still research work missing for optimizing storyline visualizations by minimizing wiggles only. If wiggles were incorporated in the optimizations of storyline visualizations before, they were always combined with the minimization of crossings and white-space gaps.

Considering wiggle minimization from a combinatorial optimization point of view the optimization problem can be related to minimizing corners or moves in permutation diagrams in Bereg et al. [9, 10]. Such permutations diagrams visualize tangles, a sequence of permutations for an unordered series of natural numbers until the identity permutation is reached and the numbers are ordered in an ascending sequence. From one permutation to the next one every element is only allowed to change its position by 1, e.g. there are only swaps by two neighboring elements allowed. Illustrating the path of each of the numbers through the complete tangle results in a drawing comparable to storyline visualizations. In both works [9, 10] the authors name as research goal the minimization of moves within a tangle diagram. These moves relate to line wiggles in storyline visualizations and can therefore be used as base to formulate a definition for wiggles in storyline visualizations.

CHAPTER 4

Preliminaries

This chapter contains basic definitions and explanations needed for the solution models in Chapter 5.2 and Chapter 5.3. The intention of this chapter is to give a short introduction in the theoretical concepts and solution approaches and to provide additional references to further readings, which go more into detail.

4.1 Integer Linear Programming (ILP)

This section gives a short overview of the Integer Linear Programming (ILP) problem and related problems. For more details see [12, 27, 41] on which the following description and definitions are based on.

In general, the linear programming problem deals with finding a maximal or minimal solution for a linear cost function or objective function where the potential values of the variables in the cost functions are restricted by certain equality and inequality constraints. Because this thesis concerns a minimization problem and any minimization problem can be transformed into a maximization problem by a negation of the objective function this section only contains explanations to minimization problems.

Given a m by n matrix A, an n-dimensional row vector (or *1* by n matrix) c which is our cost vector, an m-dimensional column vector (or a m by *1* matrix) b, and an n-dimensional column vector (or a m by *1* matrix) x for our decision variables or unknowns. The formulation in 4.1 shows a general linear program with the objective to minimize the linear cost function $\sum_{i=1}^{n} c_i x_i$. This linear cost function is subjected to the linear equality and inequality constraints expressed by $Ax \geq b$ and $x \geq 0$.

$$min \ \{cx : Ax \geq b, x \geq 0\} \tag{4.1}$$

Without specific restrictions a linear program contains continuous variables. If we require some but not all variables to be integers, we get the problem formulation for a (Linear) Mixed Integer Program (MIP) shown in 4.2 with a m by n matrix A, a m by p matrix G, an n-dimensional row cost vector c, a p-dimensional row vector h, an n-dimensional column vector x of continuous variables and a p-dimensional column vector y of integer variables.

$$\begin{aligned} &min\ cx \ + \ hy \\ &Ax \ + \ Gy \ \geq \ b \\ &x \ \geq \ 0, \ y \ \geq \ 0 \ \text{ and } y \in \mathbb{Z}^p \end{aligned} \tag{4.2}$$

By adding the restriction that all variables have to be integers we get an Integer Linear Program (ILP) shown in 4.3, again with A as m by n matrix, c as n-dimensional cost vector, b as m-dimensional vector and an n-dimensional vector x of integer variables.

$$\begin{aligned} &min\ cx \\ &Ax \ \geq \ b \\ &x \ \geq \ 0 \ \text{ and } x \in \mathbb{Z}^n \end{aligned} \tag{4.3}$$

If the integer variables are used to represent logical relationships they can be further restricted to 0-1 values. With this further restriction we obtain a 0-1 or Binary Integer Program (BIP, 0-1 MIP or 0-1 IP) shown in 4.4 with the same matrix A and vector b from before and the decision variable vector x which is only allowed to contain values of 0 or 1.

$$\begin{aligned} &min\ cx \\ &Ax \ \geq \ b \\ &x \ \in \ \{0,1\}^n \end{aligned} \tag{4.4}$$

A concrete assignment for all decision variables in x that satisfies all constraints is called a feasible solution or feasible vector. The set of all feasible solutions is called feasible set. A feasible solution where the cost function or objective function is at its minimum is called an optimal (feasible) solution, this means for a optimal solution the costs are minimized. The value of the cost function corresponds then to the optimal cost.

4.2 Maximum Satisfiability Problem (Max-SAT)

In this section the Maximum Satisfiability Problem (Max-SAT) and its difference to the SAT problem is explained. For further details have a look at [7, 18, 19, 22] which were used as foundation for this section.

In short, the satisfiability problem, called SAT, can be described with only one sentence from Kroening and Strichman [22]: "A formula is satisfiable if there exists an assignment of its variables under which the formula evaluates to true."
An assignment describes a mapping of all variables in a formula to concrete values of the corresponding domain. Because we use propositional logic in this thesis, a variable can either be true (1) or false (0).

The Maximum Satisfiability Problem is based upon the SAT problem but takes into account that certain real world problems don't necessarily require all constraints to be satisfied. For example a combinatorial optimization problem where the objective requires the minimization of a cost function has most likely many acceptable solutions where the objective is not at its optimum. Thus the goal for a Max-SAT formulation is not to find a solution that satisfies all constraints but to find one where the maximum amount of satisfied constraints is reached.

To be still able to define feasible solutions for a Max-SAT formulation we distinguish between hard and soft constraints. The set of clauses belonging to hard constraints are necessary to be satisfied to obtain a feasible solution. Clauses belonging to soft constraints on the other hand can be violated by some solutions. An optimal solution satisfies all hard constraints and a maximum amount of soft constraints.

Max-SAT formulations are usually expressed in conjunctive normal form (CNF). A formula in CNF can be seen as a set of clauses that are connected by conjunction. A clause is defined as a disjunction of literals and a literal are either simply Boolean variables or negated Boolean variables.

CHAPTER 5

Solution Models

In this section we will describe two concrete solution models for optimizing storyline visualizations by minimizing wiggles. This section also serves as answer to research question **Q1**.
At first, in Section 5.1 we get a general overview of all variables and constraints necessary to meet all requirements of correct storyline visualization, which have to be incorporated into both solution models.

Next, in Section 5.2 an Integer Linear Programming (ILP) formulation is described with three different objectives for wiggle minimization: total wiggle height minimization, maximum wiggle height minimization and the minimization of the number of wiggles. Furthermore the ILP solution model contains also the possibility of minimizing crossings as second objective by formulating a multi-objective together with one of the three wiggle minimization objectives.

Lastly, Section 5.3 treats our second solution model, a Maximum Satisfiability (Max-SAT) formulation for minimizing the total wiggle height and minimizing the number of wiggles in storyline visualizations. This formulation is build upon the ILP model and should illustrate the solution approach from the ILP model with a different solution method to allow the comparison of the results produced by both solution models.

5.1 General

Before we start creating our solution models it is necessary to formally describe the structure of storyline visualizations we want to optimize. In this section we will describe the variables and constrains for both solution models so that both models produce results with the same properties and restrictions.

For knowing the height of a wiggle the absolute position of each character is needed within the storyline visualization. Therefore we will build our solution models upon a matrix representation of the storyline. Corresponding to the currently used storyline data the concrete height and width of the storyline visualization matrix can be computed at the beginning of the optimization computations. The width of the matrix matches the number of time points of the storyline, so we have one column for every time point. The height of the matrix corresponds to the needed slots by the characters of the storyline. By knowing how many meetings are active at any time point and by knowing how many members each meeting contains the number of necessary slots can be easily determined. The result is a m by p matrix for a storyline with m slots and p time points. By comparing the occupied slots of a specific character from one time point to the next, wiggles can be detected and the height of such a potential wiggle can be calculated.

Every storyline visualization with n characters, m slots and p time points can be defined as a quadruple *(I,J,T,M)* where $I = \{i_1, i_2, ..., i_n\}$ stands for a set of *characters*, $J = \{j_1, j_2, ..., j_m\}$ describes a set of *slots*, $T = [1, p[$ is the interval of all *time points* and $M = \{(I_e, J_e, T_e)\}$ is a set of triples containing all meetings.

For a *meeting* $e = (I_e, J_e, T_e)$ with k characters and l time points is $I_e = \{i_1, i_2, ..., i_k\}$ a set of k characters, $J_e = \{j_1, j_2, ..., j_k\}$ a set for k consecutive slots and $T_e = [t_1; t_l[$ an interval of l consecutive time points where the meeting takes place. We call all characters in I_e *members* of meeting e and all time points in which character i is member in any meeting *active time points* of i.

For both solution model descriptions we will stick with this denominations of the storyline data variables.

The following list of properties outlines all necessary variables and constraints for both solution models:

P1 matrix for storing the storyline visualization

P2 variables for storing the position of the characters

P3 variables for potential occurrences of wiggles regarding a specific character and time point or variables holding the exact height of a potential wiggle regarding a specific character and time point

P4 constraint to make sure that a character occupies exactly one slot at every active time point

P5 constraint to make sure that a slot can only be occupied by at most one character at once at every time point

P6 representation of a meeting as a set of characters

P7 constraint to make sure that all members of a meeting keep their position from the start of the meeting till the end of the meeting, no character is allowed to change its position over the lifetime of the meeting

P8 constraint to make sure that all members of a meeting occupy consecutive slots, no empty lines between the members are allowed

P9 constraint to make sure that there is at least one blank line between different meetings that share at least one time point along their lifetime such that the slot above and below any meeting is held empty and cannot be occupied by characters from other meetings (containing the exception that this only applies when the meeting is not positioned at the very top or very bottom of the matrix)

P10 formulation for minimizing the total wiggle height over all characters and all time points of the storyline visualization

P11 formulation for minimizing the total number of wiggles over all characters and all time point of the storyline visualization

To prevent unnecessary many variables the movie data get condensed so that only time points are included into the visualization matrix at which at least one meeting starts or ends. With the constraint that members of a meeting are not allowed to change their position during the lifetime of the corresponding meeting it is not needed to consider time points where nothing is allowed to change and therefore wiggles can not occur. Thus such time points can get excluded from the optimizations at the very beginning without an effect on the solution.

5.2 ILP Formulation

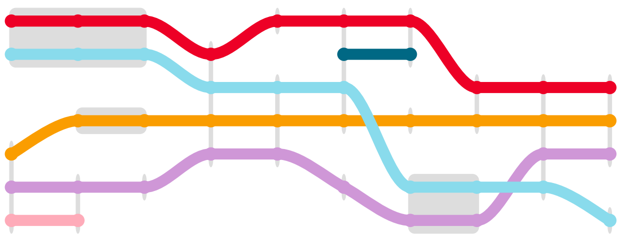
As first optimization approach for minimizing wiggles in storyline visualizations we construct an Integer Linear Programming (ILP) model. The decision to use ILP to optimize storyline visualizations was inspired by a paper of Gronemann et al. [17] where an ILP model was used for minimizing pairwise crossings in storyline visualizations.

5.2.1 Character position variables and constraints

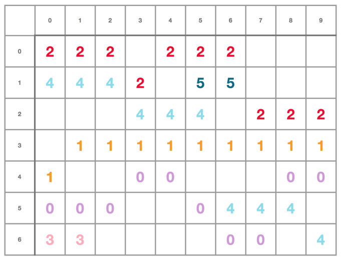
For creating the ILP model we start with the position variables of the characters within the visualization matrix. The result of the optimizations should be represented by one *m by p* matrix for m slots (rows) and p time points (columns) which contains all concrete positions of the characters at every time point and corresponds to Property **P1**. We assign each character a unique natural number as identifier starting by the number zero. Figure 5.1 shows an example of a resulting storyline visualization with condensed time points and the associated matrix representation.

Because a character requires the possibility to occupy one of all slots at all its active time points, it is necessary to have position variables for all this potential positions of the character. To make it easier we take the assumption that every character has all time points of the storyline as active time points, thus we can use the index values of our matrix as time point values.

We receive n layers of binary variables of our matrix, one for each of our n characters. One layer contains all possible positions for the corresponding character. This variables denoted with the letter x are shown in Equation 5.1 and correspond to Property **P2**. A character i (with $0 \leq i < n$) occupies a specific slot j (with $0 \leq j < m$) at a specific time point t (with $0 \leq t < p$) if the corresponding position variable $x^t_{i,j}$ is set to 1, otherwise it holds value 0. An example of a matrix with concrete position variables is shown in Figure 5.2. Figure 5.2a shows the positions of all characters for the example storyline visualization and Figure 5.2b shows the position variables layer for character *4*.

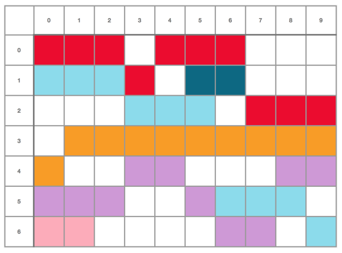


(a) Example of an optimized storyline visualization



	0	1	2	3	4	5	6	7	8	9
0	2	2	2		2	2	2			
1	4	4	4	2		5	5			
2				4	4	4		2	2	2
3		1	1	1	1	1	1	1	1	1
4	1			0	0				0	0
5	0	0	0			0	4	4	4	
6	3	3					0	0		4

(b) Taken positions of all characters



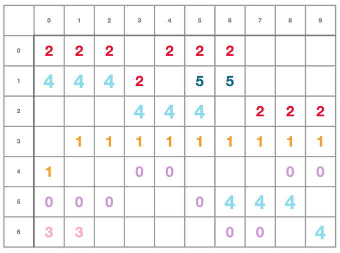
(c) Colored matrix representation of the storyline

Figure 5.1: Different views of one storyline visualization

$$x^t_{i,j} = \begin{cases} 1, & \text{if character } i \text{ uses slot } j \text{ at time point } t \\ 0, & \text{otherwise.} \end{cases} \tag{5.1}$$

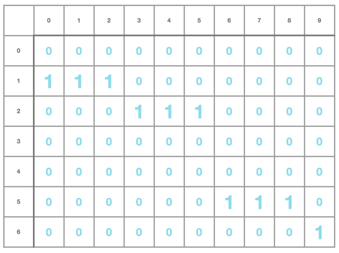
To comply with the condition of Property **P5** that a slot j can only be occupied by at most one character i for one time point t we add the constraint shown in Equation 5.2 which makes sure that for slot j and time point t there is at most one corresponding character position variable $x^t_{i,j}$ evaluating to 1 over all n characters. The constraint is also satisfied if no character occupies a specific slot j at a specific time point t.

$$\sum_{i=0}^{n-1} x^t_{i,j} \leq 1, \qquad \text{for } 0 \leq j < m,\ 0 \leq t < p \tag{5.2}$$



	0	1	2	3	4	5	6	7	8	9
0	2	2	2		2	2	2			
1	4	4	4	2		5	5			
2				4	4	4		2	2	2
3		1	1	1	1	1	1	1	1	1
4	1			0	0				0	0
5	0	0	0			0	4	4	4	
6	3	3					0	0		4

(a) Positions of all characters with highlighted character 4



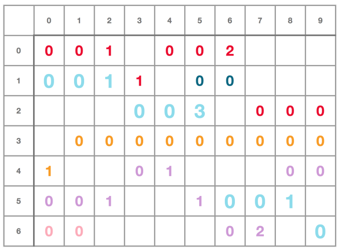
	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0
2	0	0	0	1	1	1	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	1	1	1	0
6	0	0	0	0	0	0	0	0	0	1

(b) Maxtrix of position variables of character 4

Figure 5.2: Matrix representation of the positions of character 4

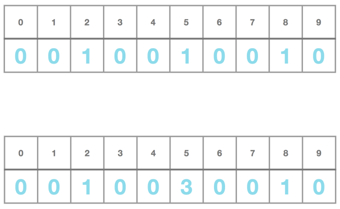
5.2.2 Wiggle variables

Next we have to define variables for retaining knowledge about wiggle occurrences and the height of a wiggle to conform Property **P3**. To hold the information of a wiggle occurrence binary variables are sufficient. To retain the height of a wiggle we require integer variables. Every character can have for every time point at most one wiggle, because a character is only allowed to occupy at most one slot at every time point. Thus for counting the number of all potential wiggles of a character we need an array of binary variables with the size of all time points of the storyline visualization. The height of a potential wiggle can be captured into a similar array of the same length but for integer variables. Of course we only need one of this variable types according to our current optimization objective. We denote such a wiggle variable with the letter z and according to our current objective function this variable z_i^t (for $0 \leq i < n$ and $0 \leq t < p - 1$) is a binary variable or an integer variable.



	0	1	2	3	4	5	6	7	8	9
0	0	0	1		0	0	2			
1	0	0	1	1		0	0			
2				0	0	3		0	0	0
3		0	0	0	0	0	0	0	0	0
4	1			0	1				0	0
5	0	0	1			1	0	0	1	
6	0	0					0	2		0

(a) Wiggle height variables of all characters with highlighed character 4



0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	1	0	0	1	0

0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	3	0	0	1	0

(b) Variable array for wiggle occurrences and variable array for wiggle heights of character 4

Figure 5.3: Variables for describing wiggles

By comparing the occupied slots of character i from one time point t to the next time point $t + 1$ we can find out if the character changes its position or stayed in the same slot and save this information in variable z_i^t. If z_i^t is a binary variable and a wiggle is found for character i we set it to 1, if no wiggle is found for character i we set it to 0.

If z_i^t is an integer variable it holds the difference of the indices of the occupied slots at time point t and time point $t+1$ for character i. Figure 5.3 shows an example of wiggle variables within a storyline visualization and the difference of our two variable types for one character. Figure 5.3a shows the integer wiggle variables for all characters mapped into our visualization matrix and Figure 5.3b shows the corresponding wiggle variable arrays for character 4.

5.2.3 Meetings representation

The set of members of a meeting and the positions of these members within the visualization matrix represent a meeting. In Figure 5.1a meetings are illustrated by light gray areas behind the member lines over the complete lifetime of the meeting.

When it comes to meetings it is more about the constraints, which restrict the possibilities of potential positions of the members. We remember the constraints regarding meetings from Section 5.1. Regarding Property **P6** meetings are seen as sets of characters, which belong together during the lifetime of the meeting, which we use as starting point for our following concrete constraint formulations. No member is allowed to change its position during the lifetime of the meeting such that a meeting can always be illustrated as a rectangle surrounding all character lines representing the members over the complete lifetime of the meeting (Property **P7**). All members of the meeting have to occupy consecutive slots and no empty lines are allowed inside this block of slots describing the position of the meeting (Property **P8**). The next slot above and below should be held empty to mark the border of the meeting (Property **P9**). Without these empty lines it would be hard to distinguish two different meetings from each other and this two meetings could be misinterpreted as one meeting.

To create appropriate formulas for this constraints we take an arbitrary meeting e with k members and a lifetime of l time points: (I_e, J_e, T_e) with $I_e = \{i_1, i_2, ..., i_k\}$ as set of k members, $J_e = \{j_1, j_2, ..., j_k\}$ as set of k consecutive slots and $T_e = [t_1; t_l[$ as interval for l time points.

First we have to make sure that every member of meeting e occupies exactly one slot per time point in T_e. We already have defined in Equation 5.2 that an arbitrary slot j can be occupied by at most one character per time point. But this constraint does not include the condition that character i has to occupy a slot at all for time point t. The constraint in Equation 5.2 would also be satisfied if no character reserves slot j in which case the sum would evaluate to 0.

However for meeting e we need all members so show themselves. Therefore we need the additional formula shown in Equation 5.3 which makes sure that every member of meeting e occupies exactly one slot for all time points in T_e. This constraint complies Property **P4**.

$$\sum_{j=0}^{m-1} x_{i,j}^t = 1, \qquad \text{for all } t \in T_e,\ i \in I_e \tag{5.3}$$

In order to implement the next constraints for meetings, which include the borders of the slot block reserved by the members of the meeting we have to know more about this borders. In other words we need information about the first slot (i.e. the slot with the lowest index value) and the last slot (i.e. the slot with the highest index value) of this block of slots representing the meeting position. We call the first slot of a meeting position the minimal slot j_{min}^e and the last slot of the meeting position the maximal slot j_{max}^e of meeting e and add them as new integer variables to our ILP model. Because it is forbidden for all members of e to change their position during the lifetime of meeting e we only need two integer variables for the borders of e, they have to stay the same for all time points in T_e.

All members of meeting e have to take a slot within the boundaries created by j_{min}^e and j_{max}^e of e. Thus every slot j_i^e taken by an arbitrary member i of meeting e has to have a higher or equal index value than the value of the minimal slot j_{min}^e of e. At the same time member i also has to have a lower or equal index value than the value of the maximal slot j_{max}^e of e. A member i has taken slot j_i^e if $x_{i,j_i^e}^t$ evaluates to 1 for all time points t in T_e. This two conditions are shown in Equation 5.4 and in Equation 5.5.

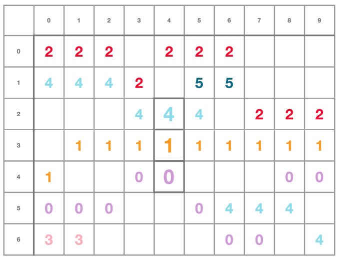
$$j_{min}^e \ \leq \ \begin{cases} j_i^e, \text{ if } x_{i,j_i^e}^t = 1, \quad 0 \leq j_i^e < m, \forall i \in I_e, \forall t \in T_e \\ undefined, \text{ otherwise.} \end{cases} \tag{5.4}$$

$$j_{max}^e \ \geq \ \begin{cases} j_i^e, \text{ if } x_{i,j_i^e}^t = 1, \quad 0 \leq j_i^e < m, \forall i \in I_e, \forall t \in T_e \\ undefined, \text{ otherwise.} \end{cases} \tag{5.5}$$

With these restrictions we have defined the borders of a meeting position and connected them with the members of the meeting so that the members occupy slots between the minimal slot and the maximal slot. But we are still missing the constraint that no empty lines are possible inside of this block so that the members are forced to take consecutive slots within the boundaries of the meeting position. We can forbid empty lines within a meeting position by restricting the distance between the maximal slot and the minimal slot of a meeting, which has to be the amount of all members (which is k in our arbitrary meeting) minus one. This condition is shown in Equation 5.6 and completes Property **P8**.

$$j_{max}^e \ - \ j_{min}^e \ = \ k - 1 \tag{5.6}$$

Figure 5.4 shows an example of a meeting and all associated variables of the meeting.



(a) Highlighted meeting e at time point 4

$$\begin{aligned} \text{meeting } e &= (I_e, J_e, T_e) \\ &= (\{0,1,4\}, \{2,3,4\}, [4,5[) \\ j^e_{min} &= 2 \\ j^e_{max} &= 4 \end{aligned}$$

(b) Concrete values for variables of meeting e

Figure 5.4: Example meeting e within a storyline visualization and concrete variables to describe the meeting and the meeting position

In Section 5.1 we mentioned to remove all time points from the storyline visualization input data where neither a meeting starts nor ends and therefore no changes within the visualization are expected or welcomed. This means that in the best case T_e of meeting e can be reduced to the start and end time point and because the end time point is considered exclusive we only have to create the meeting constraints for the start time point of the meeting.

We only need for a meeting e additional time points between the start time point and the end time point in the compressed version of T_e if another meeting starts or ends at that additional time points during the lifetime of e. This reduces the amount of necessary meeting variables. For such longer meetings with time points between start time point and end time point in the compressed T_e we have to create the additional constraint in Equation 5.7 which describes that during the complete lifetime of meeting e all members of e have to occupy the same slot for all time points in T_e. With this formula the members of e have no longer the possibility to swap their position among each other along the time points in T_e and remain as horizontal lines until the meeting ends, which complies Property **P7**.

$$x^{t_1}_{i,j} = x^{t_2}_{i,j} \quad , \qquad \forall t_1,\, t_2 \in T_e,\, \forall i \in I_e,\, 0 \leq j < m \tag{5.7}$$

We remain to create the constraint for Property **P9** that adds at least one blank line between different meetings such that these meetings are visualized separately. For that we can again use our defined borders of two different meetings that share at least one time point. By comparing the minimal slots and maximal slots of these two meetings we can easily find out which one is placed above or below the other one.

Lets assume that meeting e_1 is placed above meeting e_2 at time point t, than the maximal slot $j^{e_1}_{max}$ of e_1 and the minimal slot $j^{e_2}_{min}$ of e_2 have to be at least two slots apart. We save this information about the comparison of the relative position of the two meetings e_1 and e_2 in the addition binary variable v_{e_1,e_2}. The variable v_{e_1,e_2} therefore has value 1 if meeting e_1 is above meeting e_2 and value 0 if e_1 is placed below e_2. This new comparison variable is shown in Equation 5.8.

$$v_{e_1,e_2} \;=\; \begin{cases} 1, \text{ if } \; j^{e_2}_{min} - j^{e_1}_{max} \;\geq\; 2 \\ 0, \text{ otherwise.} \end{cases} \tag{5.8}$$

For one time point t we have to compare all distinct pairs of meetings which are present at t so that for two meetings e_1 and e_2 we end up with two binary comparison variables v_{e_1,e_2} and v_{e_2,e_1}. The formula in Equation 5.9 adds the restriction that exactly one of these two variables has to be set to 1 which completes Property **P9**.

$$v_{e_1,e_2} \;+\; v_{e_2,e_1} \;=\; 1 \tag{5.9}$$

This completes all constraints needed for our ILP model for minimizing the total wiggle height and to minimize the number of wiggles in storyline visualizations.

5.2.4 ILP objective formulation

For our ILP objective formulation we only need our character position variables we denoted with the letter x and our wiggle variables we denoted with the letter z. Because the only difference between minimizing the total wiggle height and minimizing the number of all wiggles is the variable type of the wiggle variables we can describe both objectives with only one formula. The objective function can be formulated by creating the sum over all z variables of all characters. This resulting objective formulations correspond to Property **P10** and Property **P11** and are illustrated in Equation 5.10 and Equation 5.11 with the following bounds: $0 \leq i < n$, $0 \leq j < m$, $0 \leq t < p$. The constraints in Equation 5.11 make sure that the wiggle variables are set with the correct values if a wiggle occurs.

$$minimize \;\; \sum_{i=0}^{n-1} \sum_{t=0}^{p-2} z_i^t \tag{5.10}$$

$$\begin{aligned} \text{subject to} \quad \sum_{j=0}^{m-1} j\left(x_{i,j}^{t} - x_{i,j}^{t+1}\right) \;&\leq\; z_i^t \\ \sum_{j=0}^{m-1} j\left(x_{i,j}^{t+1} - x_{i,j}^{t}\right) \;&\leq\; z_i^t \\ 0 \;&\leq\; z_i^t \end{aligned} \tag{5.11}$$

5.2.5 Maximal wiggle height

Additionally, our ILP model allows us to easily formulate another alternative objective for wiggle minimization, which has its focus on minimizing the global maximum height for wiggles in storyline visualizations.

This maximum height over all wiggles is easily represented by an integer variable w_{max} with $0 \leq w_{max} < m - 1$ and holds the value for the highest wiggle in a storyline visualization. Every wiggle height variable z_i^t for a character i (for $0 \leq i < n$) at a time point t ($0 \leq t < p - 1$) has to be lower or equal to w_{max}. This constraint is shown in Equation 5.12.

$$w_{max} \;\; \geq \;\; z_i^t \;, \quad 0 \leq i < n, \; 0 \leq t < p - 1 \tag{5.12}$$

The alternative objective formulation only contains the minimization of this maximum wiggle height variable w_{max} described in Equation 5.13.

$$minimize \quad w_{max} \tag{5.13}$$

By minimizing the maximum wiggle height variable the height of all wiggles of the visualization get reduced as well which should produce a result of a storyline visualization which doesn't contain high jumps along the character lines.

5.2.6 Crossing minimization as second objective

Beside the minimization of wiggles the ILP formulation contains also the basis for adding a second objective of minimizing pairwise crossings in storyline visualizations. By minimizing crossings together with wiggles we want to prevent the creation of crossings in favor of smaller or fewer wiggles in storyline visualizations.

Gronemann et al. [17] already created an ILP model for minimizing pairwise crossings. In their ILP model crossings are detected by comparing the relative positions of two characters to each other for two consecutive time points t and $t + 1$. If the position of the first character is above the position of the second character at time point t and below at time point $t + 1$ a crossing is found.

We can take this approach of finding crossings between character lines and use our character position variables to create also binary comparison variables for the position of two characters for a specific time point. Gronemann et al. [17] used the letter x for denoting their position comparison variables, but because we already use the letter x for our character position variables we denote our new variables with the letter y.

Thus, one binary variable $y_{a,b}^t$ (for $0 \leq a, b < n$ and $0 \leq t < p$) evaluates to 1 if character a is placed above character b at time point t, otherwise it evaluates to 0. Formula 5.14 describes how to determine if character a is above character b. Therefore we use the character position variables to find the occupied slots of the particular character by summing up all multiplications of the slot index value with the value of the corresponding position variable, which results in the index value of the occupied slot. The value of the difference of this two slot values tells us which character is above the other one and enables us to set variable $y^t_{a,b}$ accordingly.

$$y^t_{a,b} \;=\; \begin{cases} 1, \text{ if } \left(\sum\limits_{j=0}^{m-1} j\, x^t_{b,j}\right) \;-\; \left(\sum\limits_{j=0}^{m-1} j\, x^t_{a,j}\right) \;\geq\; 1 \text{ holds} \\ 0, \text{ otherwise.} \end{cases} \tag{5.14}$$

By comparing the positions of two characters one character always has to be above the other one, i.e. only $y^t_{a,b}$ or $y^t_{b,a}$ can hold value 1, the other one has to have value 0. Thus we add the constraint in Equation 5.15. It is similar to the constraint in Equation 5.9 we added for the meetings position variables.

$$y^t_{a,b} \;+\; y^t_{b,a} \;=\; 1 \tag{5.15}$$

With these new variables we can now detect crossings between two characters. To do that we first have a look at all possible cases that occur by creating the sum of the two variables $y^t_{a,b}$ and $y^{t+1}_{a,b}$ for the same character pair a and b but two different time points t and $t+1$. If the result of this sum is 1 we found a crossing between the corresponding character pair. Is the result 0 or 2, the relative position of the character pair to each other does not change. All three cases are illustrated in Equation 5.16.

$$\begin{aligned} y^t_{a,b} \;+\; y^{t+1}_{a,b} \;&=\; 1, \text{ if } a \text{ and } b \text{ cross from time point } t \text{ to } \textit{t+1} \\ y^t_{a,b} \;+\; y^{t+1}_{a,b} \;&=\; 0, \text{ if } b \text{ is above } a \text{ in both time points} \\ y^t_{a,b} \;+\; y^{t+1}_{a,b} \;&=\; 2, \text{ if } a \text{ is above } b \text{ in both time points} \end{aligned} \tag{5.16}$$

To be able to formulate our second objective which is the sum of all crossings we need another set of binary variables shown in Equation 5.17 where $c^t_{a,b}$ (with $0 \leq a, b < n$ and $0 \leq t < p-1$) evaluate to 1 if for characters a and b a crossing is found from time point t to time point $t+1$, and 0 otherwise.

$$c^t_{a,b} \;=\; \begin{cases} 1, \text{ if } \; y^t_{a,b} \;+\; y^{t+1}_{a,b} \;=\; 1 \\ 0, \text{ otherwise.} \end{cases} \tag{5.17}$$

Similar to the objectives regarding wiggle minimization our second objective can now be formulated by minimizing the sum over all crossing variables for all time points. Earlier we took the assumption that each character is present at all time points. But with our ILP model we do not explicitly prohibit gabs along a character line. This means a character does not have to be present from the very beginning of the storyline or has to stay to the very end of the storyline. Also a character can disappear and come back a few time points later, i.e. the character line of the corresponding character ends and starts again. These concessions are necessary to preserve the plot of the movie on which the visualization is based on, e.g. if a character dies in a movie it is necessary that the character line ends as well.

Thus we can only create y variables of a character pair where each of the two characters occupies a slot for the corresponding time point. With the constraint in Equation 5.3 that makes sure that every member of a meeting is present during the complete lifetime of the meeting we can derive all active time point of a character from all meetings the character is a member of. For every time point t we only need to compare the position of two characters where both have t in their set of active time points. We can describe all characters which are active at time point t with the set I^t and are now able to formulate our second objective for crossing minimizations shown in Equation 5.18.

$$minimize \sum_{t=0}^{p-2} \sum_{\forall (a,b) \in I^t,\, a \neq b} c_{a,b}^{t} \tag{5.18}$$

5.3 Max-SAT Formulation

Next to the ILP approach in Gronemann et al. [17] for optimizing storyline visualizations a different method was used in van Dijk et al. [38] where the authors developed a SAT formulation for minimizing block crossings. This work motivated the creation of a Max-SAT formulation as second solution model for wiggle minimization.

This section describes all components of the Max-SAT formulation whose objectives for minimizing wiggles in storyline visualizations are similar to the objectives in the ILP formulation. Using our ILP model as foundation we create a Max-SAT formulation for which a Max-SAT solver can be used to minimize the total height of all wiggles or to minimize the number of all wiggles.

The Max-SAT formulation takes over the character position variables as they are from the ILP formulation we denoted with the letter x. Because these position variables are binary variables this take-over is totally fine for SAT.

Since we used in our ILP model integer variables for our wiggle height variables we have to describe the height of wiggles differently in our Max-SAT formulation by only using binary variables. Furthermore all properties described in Section 5.1 regarding a correct storyline visualization based on our matrix representation have to be formulated again by following a SAT-based approach. This requires also a different representation of the meetings, which we will discuss in more detail in the following.

All clauses emerging from formulas needed for securing a correct result of storyline visualizations are to be seen as hard clauses for the Max-SAT solver. All clauses belonging to the current objective form the set of soft clauses. To create an objective for minimizing the total wiggle height or the number of all wiggles a formulation has to be found which describes a set of clauses where the maximal amount of satisfied clauses correspond to a minimized objective.

5.3.1 Wiggle height

Because we take over the character position variables denoted with the letter x from the ILP model shown in 5.1, which conforms Property **P1** and Property **P2** we can move on to our wiggle height variables that need a different representation for the Max-SAT formulation.

We recall the wiggle height variables in the ILP model which are denoted with the letter z. This variables are integer variables, thus they cannot be used in the Max-SAT formulation like in the ILP model. But we can encode these integer variables into arrays of binary variables.

In the literature several approaches can be found for encoding integer variables into binary variables. One of them is called *order encoding* [3, 6, 16] where the domain of the integer variable is used to introduce an array of binary variables which contains one binary variable for each possible value of this domain. The order encoding is sometimes also called *ladder* or *regular* encoding.

In our case the domain of a integer wiggle variable z_i^t for a character i (with $0 \leq i < n$) and a time point t (with $0 \leq t < p$) is the interval $[0, m-2]$ of all possible wiggle height values for a storyline visualization with m slots. By doing order encoding we create for every integer variable z_i^t an array of m binary variables where one binary variable $z_{i,f}^t$ (with $0 \leq f < m$) evaluates to 1 if the index value f is smaller than the primary integer variable z_i^t, i.e. $z_{i,f}^t$ is 1 iff $f < z_i^t$. By summing up all $z_{i,f}^t$ variables for one specific i and t we get the value for our associated integer variable z_i^t from our ILP model.

A slightly different encoding approach is *direct encoding* in Walsh [39] where each possible value f in $[0, m-1]$ gets again a corresponding propositional variable $z_{i,f}^t$. The difference here is that for a specific i and t only one $z_{i,f}^t$ (for $0 \leq f < m$) where $f = z_i^t$ holds is set to 1. Thus the index value f holds the value of our former integer variable.

Both approaches would require the same amount of variables and clauses for our following SAT formulas. Because with *order encoding* after Abío and Stuckey [3] the calculation of the total wiggle height only requires a sum over all binary wiggle variables $z_{i,f}^t$ we will follow this approach for binary encoding of our wiggle variables. Our new binary z variables are described in Equation 5.19 and comply Property **P3**.

$$z_{i,f}^t = \begin{cases} 1, \text{if } \ f < z_i^t, \ \text{ for } 0 \leq i < n, \ 0 \leq t < p, 0 \leq f < m-1, 0 \leq z_i^t < m \\ 0, \ \text{otherwise.} \end{cases} \tag{5.19}$$

With the transferred position variables of the characters (denoted by the letter x in Equation 5.1) from the ILP model to the Max-SAT formulation and our binary encoded wiggle variables we can start creating a propositional formula for describing the character lines along the timeline of storyline visualizations. Considering a position of character i at time point t for slot j the corresponding position variable $x^t_{i,j}$ can be either true or false. To find a wiggle we have to find the occupied slots j_1, j_2 by character i for time point t (x^t_{i,j_1}) and time point $t+1$ (x^{t+1}_{i,j_2}). Because the formula we want to create is a hard formula (i.e. the formula always has to evaluate to true for a correct storyline visualization) we have to include all possible cases regarding the position for a character from one time point to the next one.

To simplify matters we start describing the formula by considering only one arbitrary slot j. For this slot the formula has to evaluate to true for all combinations of truth-values for the position variables of character i for the time points t and $t+1$.

This means the formula we want to build has to evaluate to true for character i, slot j and the two consecutive time points t and $t+1$ for the following four cases:

C1 i occupies j at time point t and $t+1$
C2 i occupies j only at time point t
C3 i occupies j only at time point $t+1$
C4 i occupies j neither at time point t nor at time point $t+1$

All four cases represent acceptable states for the position of a character along the timeline. **C1** and **C2** describe the positive position for our character i at time point t, because if one of this two cases apply then we have found the occupied slot j at time point t. For all other slots h ($h \neq j$) at time points t for which we know that character i does not occupy h we have **C3** and **C4**. Thus for slot h were $x^t_{i,h}$ evaluates to 0, we can just negate this position variable to obtain an acceptable formulation for **C3** and **C4**: $\neg x^t_{i,h}$. But for **C1** and **C2** we have to differ between the position of i at time point $t+1$. For both cases $x^t_{i,j}$ evaluates to true. For **C1** the position of character i stays the same, so we can say $x^t_{i,j} \wedge x^{t+1}_{i,j}$ evaluates to true as well. **C2** includes a position change which can be expresses with the formula $x^t_{i,j} \wedge \neg x^{t+1}_{i,j}$.

By creating a disjunction of all four cases we obtain the following formula shown in 5.20.

$$\neg x^t_{i,j} \;\vee\; \left(x^t_{i,j} \;\wedge\; x^{t+1}_{i,j} \right) \;\vee\; \left(x^t_{i,j} \;\wedge\; \neg x^{t+1}_{i,j} \right) \tag{5.20}$$

With **C2** we have found the start of a wiggle. The next step is to find the end of a wiggle for slot h with $h \neq j$ for time point $t+1$. To correctly describe a wiggle we have to distinct between an ascending and a descending wiggle. For an ascending wiggle it is only necessary to consider slots with an index smaller than slot j, thus $h \in [0, j-1]$. For a descending wiggle only slots with a higher index than j are necessary to consider, so that $h \in [j+1, m-1]$. Aside from the different bounds of slot h the formula for both cases has the same form, hence we will describe only the case for an ascending wiggle in detail. For the formula describing a descending wiggle only the bounds for slot h at time point $t+1$ have to be changed.

Again, to simplify matters we start by taking an arbitrary slot h with $h \in [0, j-1]$. We have now again two cases to cover. In the first case $\neg x_{i,h}^{t+1}$ evaluates to true, i.e. character i does not occupy slot h at time point $t+1$. In the second case $x_{i,h}^{t+1}$ evaluates to true, i.e. i occupies h at $t+1$. If the second case applies we have found the end of the wiggle at time point $t+1$. But we are not only interested in finding the end position of the wiggle; we also want to know the exact height of it. Thus, if $x_{i,h}^{t+1}$ evaluates to true, then all binary wiggle variables $z_{i,f}^{t}$ ($\forall f \in [0, j-h-1]$) for character i and time point t have to evaluate to true. This can be expressed with the conjunction of the corresponding set of wiggle variables: $\bigwedge_{f=0}^{j-h-1} z_{i,f}^{t}$.

The result of the sum for all wiggle variables $\sum_{f=0}^{m-2} z_{i,f}^{t}$ holds the integer value of the corresponding wiggle.

Thus for time point $t+1$ we get formula 5.21 for an ascending wiggle starting at slot j at time point t and end at slot h at time point $t+1$.

$$\neg x_{i,h}^{t+1} \;\vee\; \left(x_{i,h}^{t+1} \;\wedge\; \left\langle \bigwedge_{f=0}^{j-h-1} z_{i,f}^{t} \right\rangle \right) \tag{5.21}$$

Now we can set all our little formulas parts together to one formula to describe the flow of a character line. For that we start by considering all possible slots h for time point $t+1$ if a wiggles is found for character i from time point t to time point $t+1$. This is shown in 5.22 which also contains both cases for an ascending wiggle and a descending wiggle.

$$\begin{aligned} & \left[x_{i,j}^{t} \wedge \neg x_{i,j}^{t+1} \wedge \left\langle \bigwedge_{h=0}^{j-1} \left(\neg x_{i,h}^{t+1} \vee \left(x_{i,h}^{t+1} \wedge \left\langle \bigwedge_{f=0}^{j-h-1} z_{i,f}^{t} \right\rangle \right)\right) \right\rangle \right] \\ \vee \; & \left[x_{i,j}^{t} \wedge \neg x_{i,j}^{t+1} \wedge \left\langle \bigwedge_{h=j+1}^{m-1} \left(\neg x_{i,h}^{t+1} \vee \left(x_{i,h}^{t+1} \wedge \left\langle \bigwedge_{f=0}^{h-j-1} z_{i,f}^{t} \right\rangle \right)\right) \right\rangle \right] \end{aligned} \tag{5.22}$$

Finally, we are now able to describe our character lines in Formula 5.23 including all four cases we specified above for a character position going from one time point to the next. This formula also contains the conjunction over all characters and all time points and represents therefore all character lines of the complete storyline visualization.

$$
\begin{aligned}
&\bigwedge_{i=0}^{n-1} \bigwedge_{t=0}^{p-1} \bigwedge_{j=0}^{m-1} \Bigg[\neg x_{i,j}^{t} \vee \left[x_{i,j}^{t} \wedge x_{i,j}^{t+1} \right] \\
&\vee \left[x_{i,j}^{t} \wedge \neg x_{i,j}^{t+1} \wedge \left\langle \bigwedge_{h=0}^{j-1} \left(\neg x_{i,h}^{t+1} \vee \left(x_{i,h}^{t+1} \wedge \left\langle \bigwedge_{f=0}^{j-h-1} z_{i,f}^{t} \right\rangle \right)\right) \right\rangle \right] \\
&\vee \left[x_{i,j}^{t} \wedge \neg x_{i,j}^{t+1} \wedge \left\langle \bigwedge_{h=j+1}^{m-1} \left(\neg x_{i,h}^{t+1} \vee \left(x_{i,h}^{t+1} \wedge \left\langle \bigwedge_{f=0}^{h-j-1} z_{i,f}^{t} \right\rangle \right)\right) \right\rangle \right] \Bigg]
\end{aligned} \tag{5.23}
$$

As last step two limit cases have to be considered regarding ascending and descending wiggles. An ascending wiggle is only possible if it starts at a slot $j > 0$ and a descending wiggle is only possible if it starts at a slot $j < m - 1$. This means for slot $j = 0$ only descending wiggles are possible and for slot $j = m - 1$ only ascending wiggles are possible. Adjustments according to this limit cases and further simplifications results in Formula 5.24, which is our final formula for wiggle height calculation.

$$
\begin{aligned}
\bigwedge_{i=0}^{n-1} \bigwedge_{t=0}^{p-1} \Bigg[&\left\langle \bigwedge_{j=1}^{m-1} \left(\neg x_{i,j}^{t} \vee x_{i,j}^{t+1} \vee \left\langle \bigwedge_{h=0}^{j-1} \left(\neg x_{i,h}^{t+1} \vee \left\langle \bigwedge_{f=0}^{j-h-1} z_{i,f}^{t} \right\rangle \right) \right\rangle \right) \right\rangle \\
\wedge &\left\langle \bigwedge_{j=0}^{m-2} \left(\neg x_{i,j}^{t} \vee x_{i,j}^{t+1} \vee \left\langle \bigwedge_{h=j+1}^{m-1} \left(\neg x_{i,h}^{t+1} \vee \left\langle \bigwedge_{f=0}^{h-j-1} z_{i,f}^{t} \right\rangle \right) \right\rangle \right) \right\rangle \Bigg]
\end{aligned} \tag{5.24}
$$

To be able to use a Max-SAT solver the formula is needed in conjunctive normal form (CNF). Formula 5.25 shows the transformed formula of 5.24 in CNF.

$$
\begin{aligned}
\bigwedge_{i=0}^{n-1} \bigwedge_{t=0}^{p-1} \Bigg[&\left\langle \bigwedge_{j=1}^{m-1} \bigwedge_{h=0}^{j-1} \bigwedge_{f=0}^{j-h-1} \left(\neg x_{i,j}^{t} \vee x_{i,j}^{t+1} \vee \neg x_{i,h}^{t+1} \vee z_{i,f}^{t} \right) \right\rangle \\
\wedge &\left\langle \bigwedge_{j=0}^{m-2} \bigwedge_{h=j+1}^{m-1} \bigwedge_{f=0}^{h-j-1} \left(\neg x_{i,j}^{t} \vee x_{i,j}^{t+1} \vee \neg x_{i,h}^{t+1} \vee z_{i,f}^{t} \right) \right\rangle \Bigg]
\end{aligned} \tag{5.25}
$$

5.3.2 Character position constraints

Although we took over the character position variables from our ILP model we still need propositional formulas for the Max-SAT formulation regarding our character position constraints described in section 5.1. We are still missing the constraint of Property **P5** that a slot can only be occupied by at most one character at once and the constraint of Property **P4** that a character occupies exactly one slot per active time point. Again we can take our ILP model as inspiration to add both restrictions to the Max-SAT formulation.

For Property **P5** we have to make sure that we only have at most one character per slot and per time point. We call such a constraint an at-most-one (AMO) constraint from Gent and Nightingale [16]. Therefore we need a formula which forbids that for a character i_1, a second character i_2 (with $i_1 \neq i_2$ and $0 \leq i_1, i_2 < n$), a slot j (with $0 \leq j < m$) and a time point t (with $0 \leq t < p$) the corresponding position variables $x^t_{i_1,j}$ and $x^t_{i_2,j}$ both evaluate to true, i.e. if one of the two position variables is set to true the other one has to be set to false to satisfy the formula.

This restriction is shown in Formula 5.26 and has to be applied to all time points t, all slots j and all distinct character pairs i_1 and i_2. Unnecessary clauses can be eliminated by only considering character pairs for a time point t when both characters have t in their set of active time points, i.e. both characters are member of a meeting at time point t. This set of characters is represented by the set I^t for time point t.

$$\bigwedge_{t=0}^{p-1} \bigwedge_{j=0}^{m-1} \bigwedge_{\forall i_1, i_2 \in I^t, i_1 \neq i_2} \left(\neg x^t_{i_1,j} \vee \neg x^t_{i_2,j} \right) \tag{5.26}$$

For the constraint of Property **P4** we have to make sure that a character i is present whenever i is a member in a meeting. So again, we need for every character the set of active time points T_i. For this property we need an exactly-one (EO) constraint which is a combination of an at-least-one (ALO) and an at-most-one (AMO) constraint from Gent and Nightingale [16]. An ALO constraint is build by a disjunction of all position variables $x^t_{i,j}$ for all $j \in J$. For the AMO constraint we take the same approach from before. For every pair of slots j_1 and j_2 only one corresponding position variable for one specific i and t is allowed to evaluate to true, i.e. $\neg x^t_{i,j_1} \vee \neg x^t_{i,j_2}$ has to hold. Both constraints have to be combined by conjunction to obtain our EO constraint shown in Formula 5.27.

$$\bigwedge_{t \in T_i} \left[\bigwedge_{\forall j_1, j_2 \in J, j_1 \neq j_2} \left(\neg x^t_{i,j_1} \vee \neg x^t_{i,j_2} \right) \wedge \left(\bigvee_{j \in J} x^t_{i,j} \right) \right] \tag{5.27}$$

5.3.3 Meetings representation

Again, we describe meetings in the Max-SAT formulation as set of positions occupied by the members of a meeting (Property **P6**). To guarantee a correct representation of storyline visualizations we have to create all meeting constraints for the Max-SAT formulation, which we also defined for the ILP model. These constraints include that all members of a meeting have to be positioned one below the other, such that there is no empty line and no other character from another meeting between them (Property

P8). Furthermore it has to be ensured that every member of the meeting occupies a slot and keeps this position at every time point the meeting takes place (Property **P7**). For virtual distinction between different meetings an empty line directly above and below a meeting position is needed (Property **P9**).

Because the meeting representation in our ILP model contains again integer variables it is not possible to just take over all meeting variables to the Max-SAT formulation to use them in propositional formulas. To prevent unnecessary many new binary variables by binary encoding all integer variables for describing meetings from our ILP model we will follow a slightly different approach for describing meeting positions in the Max-SAT formulation.

Thus we introduce new binary variables for all possible meeting positions and denote them with the letter s. One variable $s^t_{e,w}$ (with $1 \leq w \leq m-k+1$ and $0 \leq t < p$) for a meeting e describes a set $J^t_{e,w}$ of length k of specific slots which can get occupied by the k members of e at time point t. The time point t stands here for an arbitrary time point during the lifetime of the meeting. Such a position variable $s^t_{e,w}$ can be compared to a character position variable $x^t_{i,j}$ with the difference that a character position variable contains only one slot while a meeting position variable describes a set of k slots. The index value w of variable $s^t_{e,w}$ helps to identify a set of specific slots for which the meeting position variable stands for.
For any active time point t of meeting e with k members and a visualization matrix with m slots ($k \leq m$ holds) we have $m-k+1$ different set of slots and therefore $m-k+1$ different possibilities to position e inside the matrix. Thus for meeting e we can now define the set $S^t_e = \{s^t_{e,1}, .., s^t_{e,m-k+1}\} = \{J^t_{e,1}, .., J^t_{e,m-k+1}\}$ which contains all possible meeting positions for e at time point t. This set of meeting position variables complies Property **P6**.

These meeting position variables can be used to create all necessary restrictions regarding meetings. Additionally we also need to make sure that only one $s^t_{e,w} \in S^t_e$ evaluates to true, i.e. the meeting is only allowed to have exactly one position.

But before we can start creating meeting constraints it remains to connect the position variables of all members of e to the meeting position variable $s^t_{e,w}$. Hence for every variable $s^t_{e,w}$ we have to create a propositional formula which includes the position variables of all members of the meeting and then create an equivalence relation between this formula and the corresponding meeting position variable.

For one slot $j_r \in J^t_{e,w}$ we need exactly one member $i \in I_e = \{i_1, .., i_k\}$ for which the position variable x^t_{i,j_r} evaluates to true. This can be done with the same approach used in 5.27 where we created an exactly-one (AO) constraint. The clause $(x^t_{i_1,j_r} \vee x^t_{i_2,j_r} \vee ... \vee x^t_{i_{k-1},j_r} \vee x^t_{i_k,j_r})$ has to evaluate to true which means that at least one character has to occupy slot j_r. By creating a conjunction over all clauses for all $j_r \in J^t_{e,w}$ we have our constraint that every slot of a specific meeting position has to be occupied by at least one member of the meeting.

With 5.27 we make sure that a character i occupies exactly one slot for every active time point of i corresponding to Property **P4** and with 5.26 we ensure that a slots can only be occupied by at most one character for any time point, which complies Property **P5**. Hence we can say that if the conjunction of our new clauses evaluates to true, then every member of the meeting occupies exactly one slot within the corresponding set of slots. This formula for one $s^t_{e,w}$ is shown in 5.28 where the symbol $=$ is used for expressing equality between the position variable and the formula describing the position of the meeting. With this equality relation between them we ensure that both always have to evaluate to the same truth value.

$$\begin{aligned} s^t_{e,w} &= \left(x^t_{i_1,j_1} \vee \ldots \vee x^t_{i_k,j_1}\right) \wedge \left(x^t_{i_1,j_2} \vee \ldots \vee x^t_{i_k,j_2}\right) \wedge \ldots \wedge \left(x^t_{i_1,j_k} \vee \ldots \vee x^t_{i_k,j_k}\right) \\ &= \bigwedge_{j \in J^t_{e,w}} \left(\bigvee_{i \in I_e} x^t_{i,j} \right) \end{aligned} \tag{5.28}$$

For the constraint that every line above and below a meeting position has to be held empty we can extend the formula in 5.28. An empty line above and below meeting e at an arbitrary time point t during the lifetime of e means that no character is allowed to take slot j_{1-1} or slot j_{k+1} at time point t. Thus, for all characters $i \in I$ the position variables $x^t_{i,j_{1-1}}$ and $x^t_{i,j_{k+1}}$ have to evaluate to false. Because we already know that the members of e cannot occupy the slots j_{1-1} and j_{k+1}, since they already have to occupy a position within the boundaries of these two slots, it is sufficient for our current constraint that we only consider characters which are not a member of meeting e. All not-members of meeting e form the set $I^{'}_e$ which contains all characters of the storyline visualization which are not a member of meeting e. We can describe a not-member of e as $i' \in I^{'}_e$ or $i' \notin I_e$.
For an empty line above $s^t_{e,w}$ we add the condition for every character $i' \in I^{'}_e$ that x^t_{i',j_1-1} has to evaluate to false for all $t \in T_e$. For an empty line below $s^t_{e,w}$ the variable x^t_{i',j_k+1} has to evaluate to false for all $i' \in I^{'}_e$ and for all $t \in T_e$.

If the meeting is positioned at the very top of the visualization matrix the constraint for an empty line above the meeting position does not apply, as well as for an empty line below the meeting position if the meeting is positioned at the very bottom of the matrix.

These two restrictions can be simply added by conjunction to formula 5.28, which is shown in Formula 5.29.

$$s^t_{e,w} = \left[\bigwedge_{j \in J^t_{e,w}} \left(\bigvee_{i \in I_e} x^t_{i,j} \right) \right] \wedge \left[\bigwedge_{\forall i' \in I^{'}_e} \left(\neg x^t_{i',j_1-1} \wedge \neg x^t_{i',j_k+1} \right) \right] \tag{5.29}$$

Because we need all our formulas in CNF to be able to use Max-SAT solvers we have to replace our equality operator in 5.29 with the alternative form where the equality $x = y$

gets replaced by the expression $(\neg x \vee y) \wedge (x \vee \neg y)$. The construction in 5.30 shows this alternative form for equality from Formula 5.29.

$$\begin{aligned}
&\left[\neg s^t_{e,w} \;\vee\; \left(\left\langle \bigwedge_{j \in J^t_{e,w}} \left(\bigvee_{i \in I_e} x^t_{i,j} \right) \right\rangle \wedge \left\langle \bigwedge_{\forall i' \in I'_e} \left(\neg x^t_{i',j_1-1} \wedge \neg x^t_{i',j_k+1} \right) \right\rangle \right) \right] \\
\wedge &\left[\; s^t_{e,w} \;\vee\; \neg \left(\left\langle \bigwedge_{j \in J^t_{e,w}} \left(\bigvee_{i \in I_e} x^t_{i,j} \right) \right\rangle \wedge \left\langle \bigwedge_{\forall i' \in I'_e} \left(\neg x^t_{i',j_1-1} \wedge \neg x^t_{i',j_k+1} \right) \right\rangle \right) \right]
\end{aligned} \tag{5.30}$$

Next we translate Formula 5.30 to CNF and end up with our result shown in 5.31 for connecting our meeting position variables to our character position variables.

$$\begin{aligned}
&\left[\bigwedge_{\forall j \in J_{e,w}} \left\langle \left(\bigvee_{\forall i \in I_e} x^t_{i,j} \right) \vee \neg s^t_{e,w} \right\rangle \right] \\
\wedge &\left[\bigwedge_{\forall i' \in I'_e} \left(\left(\neg x^t_{i',j_1-1} \vee \neg s^t_{e,w} \right) \wedge \left(\neg x^t_{i',j_k+1} \vee \neg s^t_{e,w} \right) \right) \right] \\
\wedge &\left[\bigwedge_{\forall i' \in I'_e} \left\langle \bigwedge_{i_1 \in I_e} \cdots \bigwedge_{i_k \in I_e} \left(\neg x^t_{i_1,j_1} \vee \ldots \vee \neg x^t_{i_k,j_k} \vee x^t_{i',j_1-1} \vee s^t_{e,w} \right) \right\rangle \right] \\
\wedge &\left[\bigwedge_{\forall i' \in I'_e} \left\langle \bigwedge_{i_1 \in I_e} \cdots \bigwedge_{i_k \in I_e} \left(\neg x^t_{i_1,j_1} \vee \ldots \vee \neg x^t_{i_k,j_k} \vee x^t_{i',j_k+1} \vee s^t_{e,w} \right) \right\rangle \right]
\end{aligned} \tag{5.31}$$

If we can neglect the conditions that at least one line above and below the meeting position has to be held empty we get for Formula 5.31 a much simpler version in 5.32. But this formula only applies if there is no line left above and below the current meeting position or if the meeting contains all characters of the storyline.

$$\begin{aligned}
&\left[\bigwedge_{\forall j \in J_{e,w}} \left(\bigvee_{\forall i \in I_e} x^t_{i,j} \right) \vee \neg s^t_{e,w} \right] \\
\wedge &\left[\bigwedge_{i_1 \in I_e} \cdots \bigwedge_{i_k \in I_e} \left(\neg x^t_{i_1,j_1} \vee \ldots \vee \neg x^t_{i_k,j_k} \vee s^t_{e,w} \right) \right]
\end{aligned} \tag{5.32}$$

Now, we have complied Property **P8** with our new meeting representations and the meeting constraint that all members occupy consecutive slots without gaps between them and Property **P9** with the constraint regarding the empty lines above and below a meeting. But we are still missing two meeting constraints for which the meeting position variables come in handy and which would be much more difficult to formulate if we would have held on to the meeting variables from the ILP formulation.

For the first constraint we have to make sure that the meeting has exactly one position variable $s^t_{e,w} \in S^t_e$ evaluating to true for an arbitrary active time point $t \in T_e$. The second constraint contains the condition that a meeting has to hold its position along the complete lifetime of the meeting. This also includes that all members have to keep their position as well to meet Property **P7**.

The first constraint can now be easily realized by a conjunction of an at-least-one (ALO) and an at-most-one (AMO) formula shown in 5.33. This formula is similar to 5.27 which we have created for ensuring that each character occupies exactly one slot for each active time point.

$$\left[\bigvee_{\forall s \in S^t_e} s \right] \quad \wedge \quad \left[\bigwedge_{\forall s_1, s_2 \in S^t_e, s_1 \neq s_2} \left(\neg s_1 \quad \vee \quad \neg s_2 \right) \right] \tag{5.33}$$

To make sure the position of meeting e is always the same for all time points $t \in [t_s, t_e[$ of e we enforce that all position variables which are associated with the same set $J_{e,w}$ of slots but different time points have to evaluate to the same truth value. We recall the compression of the time points for the visualization described in section 5.1 where all time points got removed where neither at least one meeting starts or ends. This also reduced the amount of meetings for which this constraint is needed. We are only left with meetings containing at least one addition compressed time point between start and end time point of the meeting (the end time point is exclusive).

Because of the transitivity of an equality relation it is sufficient for one specific $J_{e,w}$ to enforce the same truth value for meeting position variable $s^{t_s}_{e,w}$ for the start time point t_s of e with all other variables $s^t_{e,w}$ for $t_s < t < t_e$. Formula 5.34 shows this constraint in CNF.

$$\bigwedge_{\forall t \in]t_s, t_e[} \left(\left(\neg s^{t_s}_{e,w} \quad \vee \quad s^t_{e,w} \right) \quad \wedge \quad \left(s^{t_s}_{e,w} \quad \vee \quad \neg s^t_{e,w} \right) \right) \tag{5.34}$$

5.3.4 Max-SAT Objective formulation

All formulas described before are required to get a correct storyline visualization similar to the ones generated with the ILP model. For the objective of our Max-SAT formulation we need clauses that don't necessarily have to evaluate to true for a correct storyline visualization. For total wiggle height minimization the number of positive z variables which hold the height of each wiggle has to be minimized. Therefore we can formulate this objective of wiggle height minimization for the Max-SAT formulation the following way:

$$\bigwedge_{t=0}^{p-1} \bigwedge_{i=0}^{n-1} \bigwedge_{j=0}^{m-1} \neg z_{i,j}^{t} \tag{5.35}$$

Every negated z variable forms one soft clause within the formulation and the more of this soft clauses evaluate to true the better is the solution for the storyline visualization. This completes Property **P10** for our Max-SAT formulation.

Next to wiggle height minimization we require a second wiggle objective regarding Property **P11**, which describes the minimization of the number of all wiggles in storyline visualizations. This second objective can easily be expressed in our Max-SAT formulation by small changes for our wiggle variables we denoted with the letter z.
Because we are no longer interested in the actual height of a wiggle, we can reduce the number of z variables. Thus, we only need one binary variable for every character at every time point instead of an array of binary variables of the length of the highest possible wiggle. This one variable should hold the information if a character i changes its position from time point t to time point $t+1$. If this is the case we have a wiggle at time point t for character i. With this reduction of the wiggle height variables we loose the information about the exact height of a wiggle, but this also means simplifications of our formula, which describes the detection of wiggles.

An arbitrary wiggle variable z_i^t evaluates to true if the storyline visualization has a wiggle at time point t for character i, otherwise it evaluates to false, shown in 5.36.

$$z_i^t = \begin{cases} 1, \text{if a wiggle occurs for character } i \text{ at time point } t \\ 0, \text{ otherwise.} \end{cases} \tag{5.36}$$

We can use our previous Formulas 5.23 and 5.24 as base to create a slightly different formula for detecting wiggles. After inserting our changed wiggle variables we receive Formula 5.37, which misses the calculation of the height of a potential wiggle. Because we only want to know the sum of all wiggles we can neglect the height of wiggles and also don't have to distinguish between ascending and descending wiggles, or pay attention to our previous limit cases of wiggles which start or end at the very top or the very bottom of the visualization matrix.

$$\begin{aligned} &\bigwedge_{i=0}^{n-1} \bigwedge_{t=0}^{p-1} \bigwedge_{j=0}^{m-1} \Bigg[\neg x_{i,j}^{t} \lor \left(x_{i,j}^{t} \land x_{i,j}^{t+1} \right) \\ &\lor \left(x_{i,j}^{t} \land \neg x_{i,j}^{t+1} \land \left\langle \bigwedge_{\forall h \in [0,m[, h \neq j} \left(\neg x_{i,h}^{t+1} \lor \left(x_{i,h}^{t+1} \land z_i^t \right) \right) \right\rangle \right) \Bigg] \end{aligned} \tag{5.37}$$

With further transformation we get Formula 5.38, which shows our new wiggle detection formula in conjunctive normal form (CNF). It's easy to see that this formula contains fewer clauses as our previous CNF formula in 5.25 for wiggle detection.

$$\bigwedge_{i=0}^{n-1} \bigwedge_{t=0}^{p-1} \bigwedge_{j=0}^{m-1} \bigwedge_{\forall h \in [0,m[, h \neq j} \left(\neg x_{i,j}^{t} \lor x_{i,j}^{t+1} \lor \neg x_{i,h}^{t+1} \lor z_i^t \right) \tag{5.38}$$

As last step for our second objective it remains to adapt our objective formula in 5.35 regarding the reduced number of wiggle variables. These adjustments are very easy. We just have to remove the conjunctions over all slots. Formula 5.39 shows our altered soft formula describing our alternative objective of minimizing the number of wiggles.

$$\bigwedge_{t=0}^{p-1} \bigwedge_{i=0}^{n-1} \neg z_i^t \tag{5.39}$$

All other formulas remain unchanged for the second objective.

CHAPTER 6

Implementation

Both solution models were programmed in Java into separated subprojects of one software project. With the support tool Maven as build tool and for dependency management the project can easily be compiled, packaged and executed. The complete project consists of four subprojects. The first subproject contains the basic code required in all other three subprojects to read and process movie data files from Tanahashi and Ma [36] provided at [35]. The second subproject contains the implementation of the ILP model from Section 5.2 of this thesis. The third project consists of the implementation of the Max-SAT formulation from Section 5.3 of this thesis. The fourth project contains a web interface for viewing storyline visualizations generated by the subprojects of the ILP model or the Max-SAT formulation. The complete code is provided at a public git repository at Github.[1]

The implementation of the ILP model uses the Gurobi Java library to create all concrete variations of the ILP model and objective functions we described in Section 5.2 and Section 7.2. With Maven the subproject can be packaged into a JAR file for running experiments on the ILP model and generate storyline visualizations. For performing optimizations on storyline visualizations the parameters for defining the specific objective can be added when the JAR gets executed.

The implementation of the Max-SAT formulation was mainly developed to serve as adapter between the Max-SAT formulation in Section 5.3 and Max-SAT solvers used for the experiments in Section 7.3. The subprojects supports the creation of input files in standard WCNF format [1] for Max-SAT solvers and creates storyline visualizations from the resulting variable assignments computed by a Max-SAT solver.

To be able to visualize storyline visualizations generated during the experiments in Chapter 7 a simple web interface was included into the software project, which was

[1] https://github.com/theresa77/wiggle-minimization

written in JavaScript. The visualizations were realized using the D3 JavaScript library. Furthermore the interface enables the user to download the image of particular storyline visualizations as PNG or SVG.

CHAPTER 7

Experiments and Result Analysis

This section contains the description and analysis of all performed experiments based on the solution models in Section 5.

The goal of the experiments and the analysis of the resulting solutions is to evaluate all possible objectives for wiggles minimization and combinations of wiggle and crossing minimization which serves as answer to research question **Q2**. The performed experiments should gain insight about which specific objective function produces the best solutions regarding wiggle minimization.

The quality of each resulting storyline visualization solution from an experiment can be determined by the concrete values of the corresponding optimization metrics *twh*, *hw*, *wc* and *cc* which were defined in Section 2.3. All four metrics are equally meaningful for the evaluation of the solution quality and for the comparison with other solutions, thus none of the metrics can outrank one of the other three, e.g. the number of wiggles is not more important that the number of crossings. The smaller the metric values the better is the quality of a solution. By comparing two solutions with each other one solution is superior to the second solution if it has at least three smaller metric values than the other solution. If both solutions have each two smaller metric values than the other solution then none of them is superior or inferior to the other solution.

The bigger part of the following experiments is based on the ILP model from Section 5.2, because the implementation of the ILP model allows more alternative objective functions than the implementation of the Max-SAT formulation.
The experiments based on the Max-SAT formulation from Section 5.3 will be mainly used for the comparison and evaluation of the two solutions models and to identify pros and cons between the two optimization methods.

Table 7.1 gives an overview of all concrete objective functions for the experiments. We have three single-objectives for wiggle minimization and three multi-objective variations with different weighting strategies. By combining every wiggle objective with every multi-objective variation we obtain in total twelve objective variations for every movie data instance we use. Due to the extended length of the nominations of all objective functions, we use abbreviations which are also shown in Table 7.1. The ILP model allows the formulation of all specified objective functions in the table. The Max-SAT formulation allows only the two single-objectives *mtwh* and *mnrw* for wiggle minimization marked by *.

concrete objective	short cut
single wiggle minimization	
minimize total wiggle height*	mtwh
minimize maximum wiggle height	mmwh
minimize number of wiggles*	mnrw
multi-objective with same weighting (sw)	
same weighting for total wiggle height and crossing count	mtwh sw
same weighting for maximum wiggle height and crossing count	mmwh sw
same weighting for wiggle count and crossing count	mnrw sw
multi-objective with higher wiggle weighting (hww)	
higher weighting for total wiggle height than for crossing count	mtwh hww
higher weighting for maximum wiggle height than for crossing count	mmwh hww
higher weighting for wiggle count than for crossing count	mnrw hww
multi-objective with higher crossing weighting (hcw)	
higher weighting for crossing count than for mtwh	mtwh hcw
higher weighting for crossing count than for mmwh	mmwh hcw
higher weighting for crossing count than for mnrw	mnrw hcw

Table 7.1: Overview of all objective functions supported by the ILP model. Objective formulations supported by the Max-SAT formulation are marked by *.

Combined wiggle and crossing minimization objectives are only possible within the ILP model, which we will refer to as multi-objectives. The concrete formulation of our multi-objectives are inspired by the weighted sum method [24]. A multi-objective formulation is represented by a linear function where our two single-objectives, one for wiggle minimization and one for crossing minimization, get multiplied by an integer weight. The minimization of the sum of both weighted single-objectives represents the complete multi-objective. A general formulation of our multi-objective functions is shown in Equation 7.1 where w_{obj} represents one of the wiggle minimization objective variations from Equation 5.10 or Equation 5.13 and c_{obj} represents the objective for crossing minimization from Formula 5.18.

$$minimize \quad a \cdot w_{obj} + b \cdot c_{obj} \tag{7.1}$$

For the concrete values of the weights a and b we use three different weighting variations.

The first multi-objective weighting contains an equal weighting between the wiggle and the crossing objective, i.e. both objectives are weighted by one. In the following we will refer to this uniform weighting of both single-objectives as *same weighting* with the abbreviation *sw*. For the second weighting variation for a multi-objective the single-objective for wiggle minimization receives a higher weighting than the single-objective for crossing minimization which we will call *higher wiggle weighting* with the abbreviation *hww*. The third and last multi-objective variation contains a higher weighting of the crossing objective over the wiggle objective which we will refer to as *higher crossing weighting* with the abbreviation *hcw*.

To ensure that a different effect is received for multi-objectives with a higher weighting for of the two single-objectives, we calculate the worst case for the current wiggle objective and use this as weight for the higher weighted single-objective. This worst case objective value depends on the currently used movie instance and gets therefore calculated during the initialization of the objective function. If our current wiggle objective is the minimization of the number of wiggles the worst case is calculated with the assumption that whenever a character is allowed to change its position than it produces a wiggle. Because a character is only allowed to change its position if it is a member at a new meeting for the current time point, we can just sum up the number of members for every new meeting for all time points. A different worst case gets calculated if our current wiggle objective considers the height of wiggles. Here we start with the same assumption that every characters is driven to always make the highest possible jump if a jump is allowed. Thus, for the worst case regarding the wiggle heights we sum up all highest possible wiggles for all characters of a new meeting at every time point.

	Inception snippet	Inception	Star Wars snippet	Star Wars
character count	7	10	10	14
time points	197	490	33	200
compr. time points	27	71	8	50
min required slots	7	12	16	17
meetings count	41	116	20	93

Table 7.2: Key data of used movie instances

We use two different movie data instances for our experiments from Tanahashi and Ma [36] which are provided at [35]. The first instance contains the movie *Inception* (2010) and the second instance contains the movie *Star Wars* (1977). Because both movie instances require too many variables and constraints within our two solution models to obtain a solution for every objective function in Table 7.1, we add two small snippets from the beginning of each movie as additional movie instances for our experiments which require much less resources during the optimizations. Table 7.2 gives an overview of the key data of our four movie instances. The table also illustrates the main distinctions between the movies *Inception* and *Star Wars*. The movie *Inception* has much more time points than the movie *Star Wars* and storyline visualizations for *Inception* will be wider than visualizations for *Star Wars*. On the other hand requires the movie *Star Wars* more slots and therefore more height for the resulting storyline visualization.

To embrace the evaluation of the results from the experiments and to support the conclusion of this thesis in Section 8 we formulate the following list of evaluation questions which will be answered during the experiments analysis in Section 7.2 and Section 7.3:

E1 Considering all four optimization metrics individually for a particular movie data instance, what are the best and worst solutions (i.e. for which solution has the considered metric the smallest or highest value)?

E2 Considering all metrics at once, what are the best and worst solutions (i.e. are there solutions where more than one metric has its smallest or highest value over all solutions for the current movie instance)?

E3 By comparing the results after one hour and after ten hours for every objective, are there observable effects (positive of negative) for the three (or two for multi-objectives) metric values not included into the objective while minimizing one (or two for multi-objectives) of the metrics (e.g. has the minimization of the number of wiggles a negative effect on the number of crossing)?

E4 By comparing the results per objective with and without an initial instance, which approach produces better results regarding the metrics?

E5 Which of the alternative objectives for wiggle minimization did produce the best or worst solutions regardless of a possible combination with crossing minimization?

E6 By comparing the solutions for minimizing only wiggles and minimizing wiggles combined with the number of crossings, which approach produces better solutions regarding the resulting values for the optimization metrics and did the combined objectives produce solutions with less crossings?

E7 By comparing the results for the different alternatives for multi-objectives, is there an observable difference in the quality of the results between the different weighting approaches for wiggle and crossing minimizations (e.g. did the higher weighting of wiggles or crossings produce solutions with lower metric values than uniform weighting)?

E8 Considering the individual objective definitions over all movie instances, does the quality of the corresponding solutions differ between the movie instances?

E9 By comparing the solutions for one movie instance over all used Max-SAT solvers, how much does the quality of the solutions differ between the solvers?

E10 By comparing the solutions from the experiments based on the ILP formulation and the solutions based on the Max-SAT formulation, how does the quality differ between the results from these two solution models?

7.1 Setup

To be able to compare the solutions from different experiments we used a consistent setup for all our experiments. Therefore all experiments where executed using the grid engine at the Algorithms and Complexity Group institute [4] where version 6.2u5 of the Sun Grid Engine (SGE) is currently installed. Furthermore all experiments used the same node with an Intel Xeon E5-2640 v4 (2.40GHz 10-core) processor.

All experiments for both solution models got a timeout of ten hours. If no solution was found after these ten hours the calculation got terminated without a solution.

For the experiments based on the ILP model we used as linear programming solver the Gurobi Optimizer version 7.0 [29]. To improve the runtime of the ILP solving and the quality of the solutions with Gurobi a specific set of Gurobi parameters were set for the implemented *RGBModel* which were identified in advance during pre experiments. The integer parameter *MIPFocus* was set to 3 to lay more focus on moving the objective bound during the optimization. The integer parameter *Method* for selecting different algorithms for the root was set to 3 to select the concurrent solver. The integer parameter *Presolve* was set to value 2 to select an aggressive presolve level. All ILP experiments were started using five threads for a job at the grid engine and to control the number used threads during the ILP solving the Gurobi integer parameter *Threads* was set to 5. For all other *RGBModel* parameters the default values were used. Furthermore we used Gurobi callbacks whenever a new solution during the calculations was found.

All experiments based on the ILP model were also executed twice, at first without pre-initialization of the model variables and the second time with an initial solution for the particular movie instance. As initial instance the solution from the genetic algorithm after 1500 generations from Tanahashi and Ma [36] was used. The implementation of this genetic algorithm can be found at [35]. Because we only needed the initial solutions to create a different starting point for the optimizations the implementation of the genetic algorithm without optimizing pre-computations was used for generating the initial solutions.

For the experiments without initial solution we computed the height of our visualization matrix by calculating the minimal amount of necessary slots. With such a minimal amount of slots we can still place all meetings at all time points below each other with one blank line between them and without too much empty lines above, below or between the meetings. For the experiments with an initial solution the height of our visualization matrix is given by the initial solution. Because all initial solutions require more slots than the minimal amount of necessary slots we also require more variables and constraints for the corresponding ILP model instances.

The concrete values for the required number of variables and constraints for the different objective functions are summarized in several tables. Table 7.4 lists the key values for the Inception snippet instance, Table 7.5 for the complete Inception movie instance, Table 7.6 for the Star Wars snippet instance and Table 7.7 for the complete Star Wars movie instance. The different weighting variations of multi-objectives have no effect on the number of variables and constraints.

For the experiments based on the Max-SAT formulation the four Max-SAT solvers *LMHS* [32, 33], *Loandra* [11], *MaxHS* [8, 15] and *Maxino* [5] from the *2017 Max-SAT Evaluation* [2] were used. The decision of choosing these four solvers was made after pre-experiments with small movie instances where the four selected solvers were the most promising ones. All four Max-SAT solvers accept the same input files containing all required variables and constraints regarding the specific movie instance in the standard WCNF format [1]. An overview of the different input files with the number of variables and clauses can be found at Table 7.3.

wiggle obj	no. of variables	no. of hard clauses	no. of soft clauses
Key data of concrete SAT instance for Inception snippet			
mtwh	1796	28284	687
mnrw	1208	20836	99
Key data of concrete SAT instance for Star Wars snippet			
mtwh	2831	3168876	1025
mnrw	1871	3096236	65

Table 7.3: Key data of concrete Max-SAT instances

wiggle obj	constraints	variables
wiggle minimization - no initial variable assignment		
mtwh	1656	2494 int (2223 binary)
mmwh	1656	2495 int (2223 binary)
mnrw	1656	2305 int (2223 binary)
wiggle minimization - with initial variable assignment		
mtwh	6066	8332 int (8061 binary)
mmwh	6066	8333 int (8061 binary)
mnrw	6066	8143 int (8061 binary)
multi-objective (sw/hww/hcw) - no initial variable assignment		
mtwh	2266	3138 int (2867 binary)
mmwh	2266	3139 int (2867 binary)
mnrw	2266	2949 int (2867 binary)
multi-objective (sw/hww/hcw) - with initial variable assignment		
mtwh	6676	8976 int (8705 binary)
mmwh	6676	8977 int (8705 binary)
mnrw	6676	8787 int (8705 binary)

Table 7.4: Concrete Gurobi ILP model instance of Inception snippet instance

short cuts	constraints	variables
wiggle minimization - no initial variable assignment		
mtwh	11018	14290 integer (13348 binary)
mmwh	11018	14291 integer (13348 binary)
mnrw	11018	13580 integer (13348 binary)
wiggle minimization - with initial variable assignment		
mtwh	42172	51697 integer (50755 binary)
mmwh	42172	51698 integer (50755 binary)
mnrw	42172	50987 integer (50755 binary)
multi-objective (sw/hww/hcw) - no initial variable assignment		
mtwh	15400	18778 integer (17836 binary)
mmwh	15400	18779 integer (17836 binary)
mnrw	15400	18068 integer (17836 binary)
multi-objective (sw/hww/hcw) - with initial variable assignment		
mtwh	46554	56185 integer (55243 binary)
mmwh	46554	56186 integer (55243 binary)
mnrw	46554	55475 integer (55243 binary)

Table 7.5: Concrete Gurobi ILP model instance complete Inception instance

short cuts	constraints	variables
wiggle minimization - no initial variable assignment		
mtwh	2620	2468 integer (2348 binary)
mmwh	2620	2469 integer (2348 binary)
mnrw	2620	2388 integer (2348 binary)
wiggle minimization - with initial variable assignment		
mtwh	4840	4418 integer (4298 binary)
mmwh	4840	4419 integer (4298 binary)
mnrw	4840	4338 integer (4298 binary)
multi-objective (sw/hww/hcw) - no initial variable assignment		
mtwh	3754	3692 integer (3572 binary)
mmwh	3754	3693 integer (3572 binary)
mnrw	3754	3612 integer (3572 binary)
multi-objective (sw/hww/hcw) - with initial variable assignment		
mtwh	5974	5642 integer (5522 binary)
mmwh	5974	5643 integer (5522 binary)
mnrw	5974	5562 integer (5522 binary)

Table 7.6: Concrete Gurobi ILP model instance of Star Wars snippet instance

short cuts	constraints	variables
wiggle minimization - no initial variable assignment		
mtwh	17172	18931 integer (18045 binary)
mmwh	17172	18932 integer (18045 binary)
mnrw	17172	18231 integer (18045 binary)
wiggle minimization - with initial variable assignment		
mtwh	55712	58988 integer (58102 binary)
mmwh	55712	58989 integer (58102 binary)
mnrw	55712	58288 integer (58102 binary)
multi-objective (sw/hww/hcw) - no initial variable assignment		
mtwh	25060	26979 integer (26093 binary)
mmwh	25060	26980 integer (26093 binary)
mnrw	25060	26279 integer (26093 binary)
multi-objective (sw/hww/hcw) - with initial variable assignment		
mtwh	63600	67036 integer (66150 binary)
mmwh	63600	67037 integer (66150 binary)
mnrw	63600	66336 integer (66150 binary)

Table 7.7: Concrete Gurobi ILP model instance of complete Star Wars instance

7.2 Experiments with ILP Formulation

In this section the evaluation questions from above will be answered for all experiments based on our ILP model from Section 5.2. We start with the answering of the Questions **E1** to **E7** for each movie instance individually, followed by the answer of Evaluation Question **E8** concerning all four movie instances at once.

7.2.1 Evaluation of Inception snippet

The resulting metric values of the Inception snippet instance are shown in Table 7.8. We will consider mainly the solutions after ten hours, because we expect the solutions after ten hours to be superior to the solutions after one hour.

*Answer to **E1***: The best value for metric *twh* is 39 and for metric *wc* it is 26, both belong to the solution of single-objective *mtwh* without initial solution (shown in Figure 7.1). The best value for metric *hw* is 2 and is associated with the multi-objectives *mmwh sw* (shown in Figure 7.2) and *mmwh hcw*, both without initial solution. The best value for metric *cc* is 5 and corresponds to the multi-objectives *mmwh sw* (shown in Figure 7.2) and *mtwh hcw* (shown in Figure 7.3) without initial variable assignment. The worst value 733 for metric *twh* as well as the worst value 27 for metric *hw* and the worst value 51 for metric *cc* are corresponding to the single-objective *mnrw* with initial variable assignment which is shown in Figure 7.4. The highest result for metric *wc* holds value 65 and belongs to single-objective *mmwh* with initial variable assignment shown in Figure 7.5.

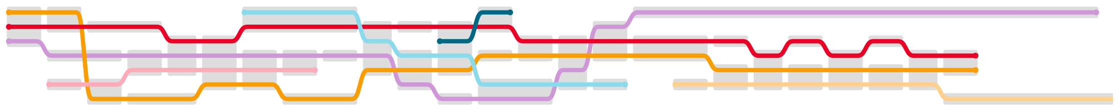


Figure 7.1: Inception snippet solution for objective *mtwh* with best results of metrics *twh* and *wc* (twh:39, hw:6, wc:26, cc:10)

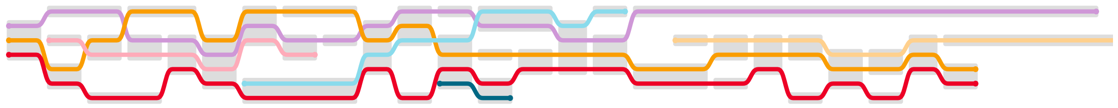


Figure 7.2: Inception snippet solution for objective *mmwh sw* with best results of metrics *hw* and *cc* (twh:72, hw:2, wc:52, cc:5)

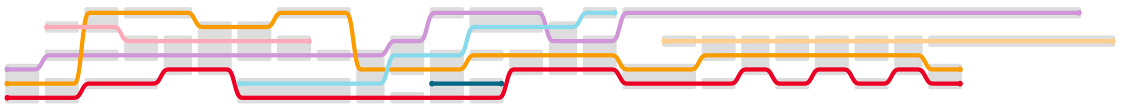


Figure 7.3: Inception snippet solution for objective *mtwh hcw* with best result of metric *cc* (twh:42, hw:5, wc:28, cc:5)

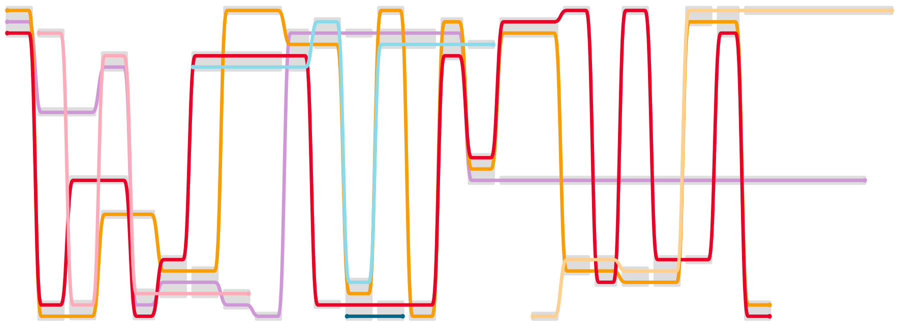


Figure 7.4: Inception snippet solution for objective *mnrw* with worst results of metrics *twh*, *hw* and *cc* (twh:733, hw:27, wc:47, cc:51)

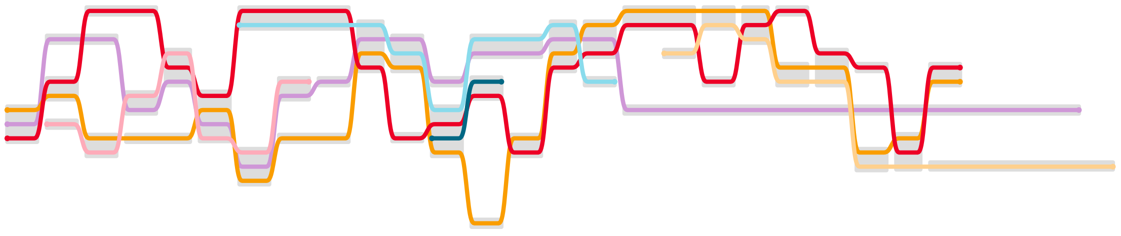


Figure 7.5: Inception snippet solution for objective *mmwh* with worst result of metric *wc* (twh:225, hw:6, wc:65, cc:36)

*Answer to **E2***: We have two solutions where each holds two smallest metric values over all ten hour solutions. The first is the solution of single-objective *mtwh* (Figure 7.1) and the second of multi-objective *mmwh sw* (Figure 7.2). Both of these two objectives didn't have an initial variable assignment. The worst solution for all metrics together is the solution for single-objective *mnrw* with initial variable assignment (Figure 7.4). The solution has the highest values for three metric values.

*Answer to **E3***: Altogether we have twelve cases where an effect for metrics not included into the objective function is observable between the solution after one hour and the solution after ten hours. For five of these cases there are only negative effects, i.e. the not included metric values increased. For one case we have a combination of a positive effect and a negative effect, i.e. one not included metric increased while another not included metric decreased. The remaining six cases contain only positive effects for metrics not included into the objective function, i.e. the not included metrics only decreased together with the metrics corresponding to the specific objective function.

*Answer to **E4***: It is clearly recognizable that almost all objectives connected to experiments without initial variable assignment produced better solutions than the experiments for the same objectives but with an initialization of the variables. The only exceptions are multi-objective *mmwh hww*, where we didn't get any solutions, and multi-objective *mnrw hww* where none of the two solutions (one with and one without initialization) is superior to the other solution.

*Answer to **E5***: We consider first every objective function block on its own to find the best wiggle objective for the specific block. Over all experiment blocks we obtain wiggle objective *mtwh* as most efficient, i.e. *mtwh* produced more best solutions within each block than the other two wiggle objectives. The other two wiggle objectives *mmwh* and *mnrw* are equally efficient and therefore there is no obvious worst wiggle objective for the Inception snippet instance.

*Answer to **E6***: Both, the best solution and the worst solution over all solutions are corresponding to single-objectives. The quality of the solutions between single-objectives and multi-objectives does not differ very much, the quality of the solutions depends more on a potential initial solution used at the beginning of the optimization. The smallest values for *cc* are all corresponding to multi-objectives. Comparing each single-objective with the associated multi-objectives, once with initial variable assignment and once without initialization, it is clear that the additional minimization of crossings had a positive effect on the number of crossing for this movie instance without a serious negative effect for the corresponding wiggle metric values.

*Answer to **E7***: Regarding metric *cc* there aren't essential differences between the solutions for the multi-objectives with uniform weighting and with a higher crossing weighting. The values for *cc* corresponding to the multi-objectives with highest wiggle weighting were slightly higher as for higher crossing weighting, but still lower as for the associated single-objectives. Thus, for the Inception snippet a higher weighting of wiggles or crossings doesn't produce essentially better results for the corresponding higher weighted metric.

	1 hour					10 hours				
	time	twh	hw	wc	cc	time	twh	hw	wc	cc
wiggle minimization - no initial variable assignment										
mtwh	*1048s*	40	6	28	9	*5431s*	**39**	6	**26**	10
mmwh	*2146s*	114	3	62	30	*2146s*	114	3	62	30
mnrw	N/A	N/A	N/A	N/A	N/A	*8561s*	62	6	29	24
wiggle minimization - with initial variable assignment										
mtwh	*2837s*	50	5	29	12	*8118s*	47	6	**26**	12
mmwh	*936s*	251	8	70	47	*31381s*	225	6	65	36
mnrw	*629s*	733	27	47	51	*629s*	733	27	47	51
multi-objective with same weighting (sw) - no initial variable assignment										
mtwh	N/A	N/A	N/A	N/A	N/A	*14821s*	47	4	34	6
mmwh	*337s*	63	**2**	47	10	*9502s*	72	**2**	52	**5**
mnrw	*3047s*	55	3	31	8	*3047s*	55	3	31	8
multi-objective with same weighting (sw) - with initial variable assignment										
mtwh	*1899s*	78	5	40	12	*29770s*	77	5	42	12
mmwh	*752s*	262	7	62	12	*7078s*	219	6	64	12
mnrw	*952s*	278	21	31	10	*952s*	278	21	31	10
multi-objective with higher wiggle weighting (hww) - no initial variable assignment										
mtwh	*2205s*	47	6	32	10	*26769s*	42	5	28	8
mmwh	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
mnrw	*2893s*	81	5	36	21	*16920s*	73	5	35	15
multi-objective with higher wiggle weighting (hww) - with initial variable assignment										
mtwh	*2135s*	80	5	41	12	*33643s*	79	5	42	12
mmwh	*457s*	171	5	59	12	*457s*	171	5	59	12
mnrw	*1571s*	408	22	36	11	*10924s*	176	19	31	10
multi-objective with higher crossing weighting (hcw) - no initial variable assignment										
mtwh	N/A	N/A	N/A	N/A	N/A	*7652s*	42	5	28	**5**
mmwh	*1845s*	65	**2**	50	6	*1845s*	65	**2**	50	6
mnrw	N/A	N/A	N/A	N/A	N/A	*20203s*	51	4	31	7
multi-objective with higher crossing weighting (hcw) - with initial variable assignment										
mtwh	*3277s*	85	7	39	11	*34256s*	62	5	34	7
mmwh	*2109s*	124	5	58	9	*2109s*	124	5	58	9
mnrw	*624s*	248	20	33	8	*25057s*	255	23	31	8

Table 7.8: Solutions of Inception snippet instance from experiments based on the ILP model. Smallest metric values are displayed in bold. *N/A* indicates that no solution was found after the associated computation time.

7.2.2 Evaluation of Inception

Table 7.9 shows all results for the complete Inception movie instance which we will use in the following to answer the evaluation questions for this instance.

*Answer to **E1***: The smallest value for metric *twh* is 249, the smallest value for metric *wc* is 95 and the smallest value for metric *cc* is 92, all three correspond to the solution for single-objective *mtwh* with initial variable assignment which is shown in Figure 7.6. The best result for metric *hw* has value 7 which appears in three solutions. The first solution corresponds to single-objective *mmwh* without initial solution shown in Figure 7.8, the second solution to the multi-objective *mmwh sw* without initial solution and the third solution to the multi-objective *mmwh hww* without initial solution shown in Figure 7.7. The highest result after ten hours for metric *twh* has value 2831 and the highest result value for metric *wc* is 265, both correspond to the same solution for multi-objective *mmwh hcw* with initial variable assignment which is shown in Figure 7.9. The worst solution for metric *hw* holds value 47 and belongs to the solution of single-objective *mnrw* with initial variable assignment shown in Figure 7.10. The highest value of metric *cc* is 247 and corresponds to the solution of single-objective *mmwh* without initial variable assignment which is shown in Figure 7.8.

*Answer to **E2***: The best solution for all metric values together is the result for single-objective *mtwh* shown in Figure 7.6, because it contains the smallest values for three metrics. The worst solution for the complete Inception movie is with two highest metric values the solution for multi-objective *mmwh hcw* shown in Figure 7.9.

*Answer to **E3***: By comparing the results after one hour and after ten hours for every objective, only two cases with one negative effect exist where metrics increased which were not included into the corresponding objective function. All other ten cases with observable effects were all positive effects, which means that not included metrics decreased together with the metrics belonging to the current objective function.

*Answer to **E4***: For the complete Inception movie the experiments with an initial variable assignment did produce much more solutions. A lot of experiments without an initial solution didn't even find one solution after ten hours. Relating to the quantity of solutions for this movie instance an initial variable assignment is the better choice. However, comparing the quality of the solutions, most of the resulting storyline visualizations from the experiments without initial variable assignment are superior to the solutions of the experiments, which started with initial variable assignments.

*Answer to **E5***: The selection of the best wiggle objective depends on what's more important, quantity and quality of solutions. If the quantity of the solutions is more important, than wiggle objective *mmwh* was obviously the best one over all experiment blocks, just because it was the only wiggle objective where at least one solutions was found after ten hours. If the focus lies more on the quality of the solutions and we only consider experiment blocks for which at least one solution was found for each wiggle objective, the wiggle objective *mtwh* produced the best solutions.

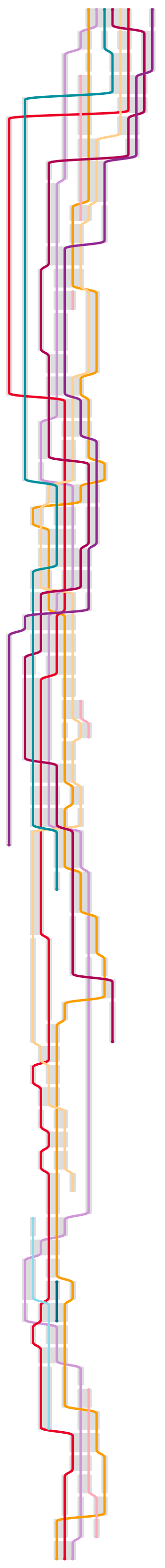


Figure 7.6: Inception solution for objective *mtwh* with best result of metrics *twh*, *wc* and *cc* (twh:249, hw:15, wc:95, cc:92)

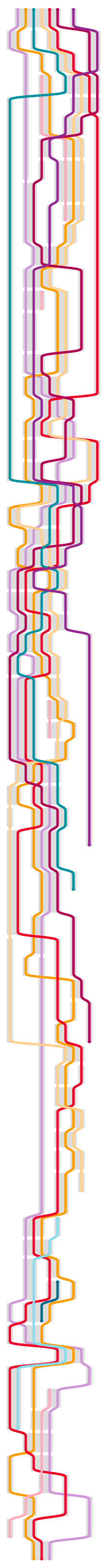


Figure 7.7: Inception solution for objective *mmwh hww* with best result of metric *hw* (twh:463, hw:7, wc:208, cc:101)

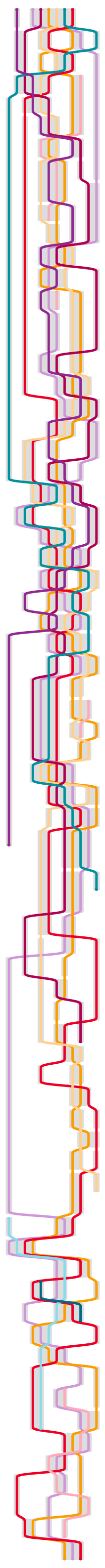


Figure 7.8: Inception result for objective *mmwh* with worst result of metric *cc* (twh:675, hw:7, wc:231, cc:247)

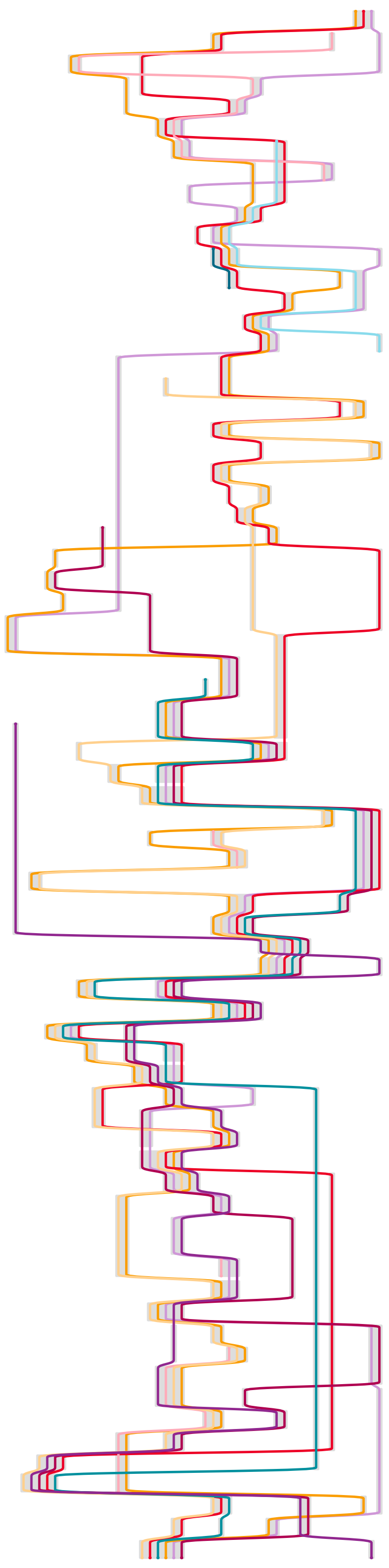


Figure 7.9: Inception solution for objective *mmwh hcw* with worst results of metrics *twh* and *wc* (twh:2831, hw:34, wc:265, cc:106)

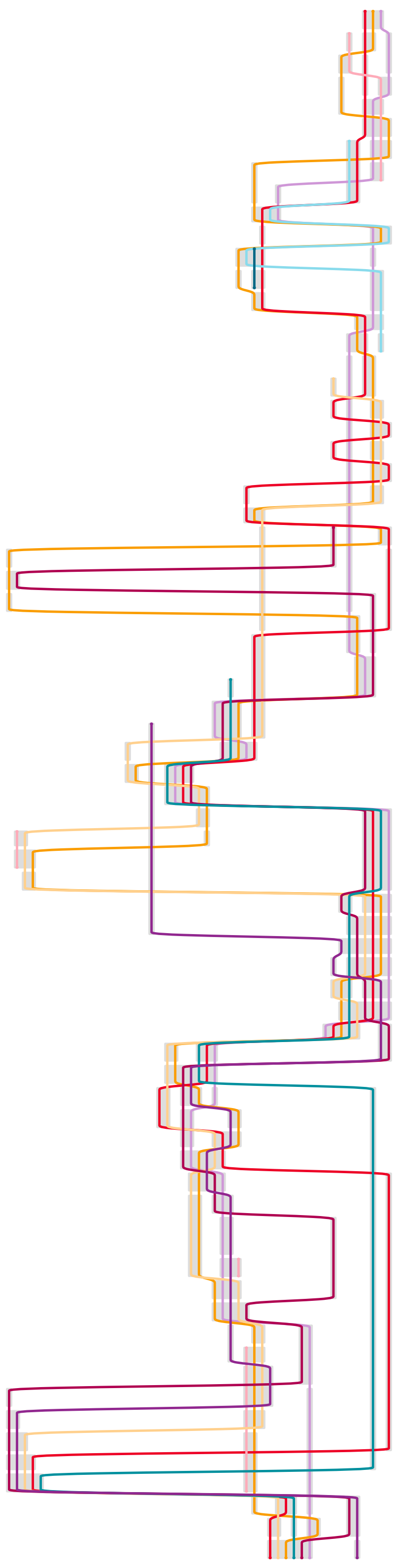


Figure 7.10: Inception solution for objective *mmrw* with worst result of metric *hw* (twh:1639, hw:47, wc:115, cc:243)

	1 hour					10 hours				
	time	twh	hw	wc	cc	time	twh	hw	wc	cc
wiggle minimization - no initial variable assignment										
mtwh	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
mmwh	N/A	N/A	N/A	N/A	N/A	*5811s*	675	**7**	231	247
mnrw	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
wiggle minimization - with initial variable assignment										
mtwh	*2220s*	581	28	112	129	*35703s*	**249**	15	**95**	**92**
mmwh	*0s*	2482	38	261	121	*0s*	2482	38	261	121
mnrw	*3583s*	2213	47	159	223	*32666s*	1639	47	115	243
multi-objective with same weighting (sw) - no initial variable assignment										
mtwh	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
mmwh	*3048s*	675	**7**	242	186	*32986s*	607	**7**	226	128
mnrw	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
multi-objective with same weighting (sw) - with initial variable assignment										
mtwh	*3556s*	1247	34	211	111	*35227s*	746	28	181	100
mmwh	*1810s*	2619	33	264	110	*30103s*	2492	24	260	96
mnrw	*3465s*	2144	41	180	107	*24767s*	1669	39	143	98
multi-objective with higher wiggle weighting (hww) - no initial variable assignment										
mtwh	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
mmwh	N/A	N/A	N/A	N/A	N/A	*29219s*	463	**7**	208	101
mnrw	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
multi-objective with higher wiggle weighting (hww) - with initial variable assignment										
mtwh	*3562s*	1075	31	196	108	*28594s*	565	13	161	101
mmwh	*720s*	2615	33	263	113	*21196s*	2488	27	259	106
mnrw	*3225s*	2021	44	184	107	*32868s*	1610	38	149	102
multi-objective with higher crossing weighting (hcw) - no initial variable assignment										
mtwh	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
mmwh	N/A	N/A	N/A	N/A	N/A	*35633s*	583	9	237	113
mnrw	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
multi-objective with higher crossing weighting (hcw) - with initial variable assignment										
mtwh	*3512s*	1562	34	231	111	*29050s*	929	24	183	107
mmwh	*2311s*	2866	35	267	111	*20464s*	2831	34	265	106
mnrw	*2891s*	2059	30	191	98	*34203s*	1537	36	139	95

Table 7.9: Solutions of Inception instance from experiments based on the ILP model. Smallest metric values are displayed in bold. *N/A* indicates that no solution was found after the specific computation time.

*Answer to **E6***: By looking only at the best and the worst solution from Question *E2* the single-objective experiments produced better solutions than the multi-objective experiments. But the quality of the best solutions from multi-objectives is altogether not worse than the solutions from the single-objective experiments. Even though the smallest value for metric *cc* belongs to a solution of a single-objective experiment the average value for metric *cc* is lower for the multi-objective solutions than for the single-objective solutions.

*Answer to **E7***: There are no significant differences between the qualities of the solutions between the alternative multi-objectives. The average values for metric *cc* between the multi-objective experiment blocks are quite equal, thus higher weighting of the crossing count did not result in better results regarding metric *cc*. Also the higher weighting of the specific wiggle objective didn't produce solutions with much smaller wiggle metric values.

7.2.3 Evaluation of Star Wars snippet

All results for the optimization metrics of the experiments based on the Star Wars snippet movie instance are summarized in Table 7.10. This section contains the answers for Evaluation Questions **E1** to **E7** for this movie instance.

*Answer to **E1***: The smallest result value for metric *twh* is 19 and the smallest value for metric *wc* is 10, both correspond to the solution of the single-objective *mtwh* without initial variable assignment which is shown in Figure 7.11. Value 10 for metric *wc* also appears within the solution for single-objective *mnrw* with initial variable assignment shown in Figure 7.14. The best result for metric *hw* is 3 and appears in 6 out of 8 experiments for wiggle objective *mmwh*. Considering all other metrics as well the solution regarding the multi-objective *mmwh sw* without initial variable assignment can be seen as best solution over all 6 solutions which is shown in Figure 7.12. The best result of *cc* has also value 3 and corresponds to the two multi-objectives *mtwh hcw* and *mnrw hcw*, both with initial variable assignment. Again, by considering all other metric values as well the first of this two solutions has smaller metric values than the second which is shown in Figure 7.13. The worst solution for metric *twh* holds value 141 and the worst value for metric *hw* is 25, both belong to the solution for single-objective *mnrw* with initial variable assignment and the corresponding visualization is shown in Figure 7.14. The highest value for metric *wc* is 30 and is associated to the single-objective *mmwh* with initial variable assignment shown in Figure 7.14. Lastly, the highest value for metric *cc* is 37 and corresponds to single-objective *mnrw* without initial variable assignment and is shown in Figure 7.16.

*Answer to **E2***: The best solution for all four metric values together is again the solution for the single objective *mtwh* without initial variable assignment with the smallest values for the two metrics *twh* and *wc*, the corresponding image is shown in Figure 7.11. The worst solution with the highest values for metric *twh* and metric *hw* corresponds to the single-objective *mnrw* with initial variable assignment and is shown in Figure 7.14.

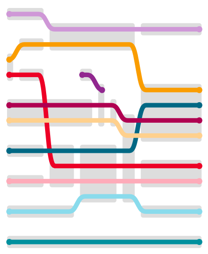


Figure 7.11: Star Wars snippet solution for objective *mtwh* with best result of metrics *twh* and *wc* (twh:19, hw:6, wc:10, cc:5)

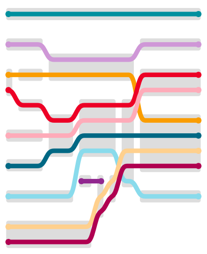


Figure 7.12: Star Wars snippet solution for objective *mmwh sw* with best result of metric *hw* (twh:31, hw:3, wc:20, cc:4)

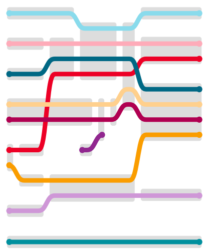


Figure 7.13: Star Wars snippet solution for objective *mtwh hcw* with best result of metric *cc* (twh:21, hw:5, wc:12, cc:3)

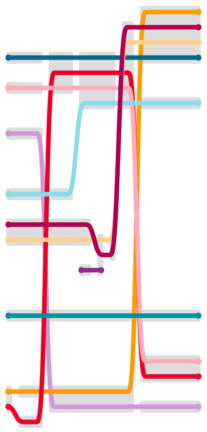


Figure 7.14: Star Wars snippet solution for objective *mnrw* with worst result of metrics *twh* and *hw* (twh:141, hw:25, wc:10, cc:34)

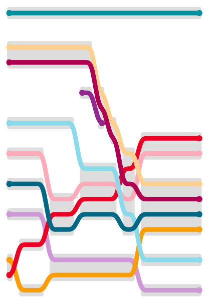


Figure 7.15: Star Wars snippet solution for objective *mmwh* with worst result of metric *wc* (twh:63, hw:3, wc:30, cc:13)

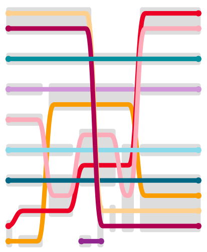


Figure 7.16: Star Wars snippet solution for objective *mnrw* with worst result of metric *cc* (twh:79, hw:13, wc:11, cc:37)

	1 hour					10 hours				
	time	twh	hw	wc	cc	time	twh	hw	wc	cc
wiggle minimization - no initial variable assignment										
mtwh	*832s*	24	6	11	8	*26630s*	**19**	6	**10**	5
mmwh	*527s*	44	**3**	26	10	*527s*	44	**3**	26	10
mnrw	*2422s*	76	11	13	37	*12046s*	79	13	11	37
wiggle minimization - with initial variable assignment										
mtwh	*806s*	25	8	11	9	*7519s*	20	5	11	6
mmwh	*150s*	63	**3**	30	13	*150s*	63	**3**	30	13
mnrw	*14s*	141	25	**10**	34	*14s*	141	25	**10**	34
multi-objective with same weighting (sw) - no initial variable assignment										
mtwh	*1731s*	28	6	15	5	*1731s*	28	6	15	5
mmwh	*505s*	31	**3**	20	4	*505s*	31	**3**	20	4
mnrw	*826s*	58	12	16	21	*826s*	58	12	16	21
multi-objective with same weighting (sw) - with initial variable assignment										
mtwh	*1010s*	27	7	15	5	*1010s*	27	7	15	5
mmwh	*147s*	73	7	27	5	*147s*	73	7	27	5
mnrw	*123s*	64	12	13	5	*123s*	64	12	13	5
multi-objective with higher wiggle weighting (hww) - no initial variable assignment										
mtwh	*2113s*	33	6	17	5	*6103s*	31	6	19	5
mmwh	*1740s*	48	5	21	9	*15835s*	38	**3**	21	8
mnrw	*2358s*	31	5	12	5	*5096s*	29	5	11	6
multi-objective with higher wiggle weighting (hww) - with initial variable assignment										
mtwh	*183s*	30	7	16	5	*11928s*	27	7	15	5
mmwh	*1629s*	80	6	27	5	*13769s*	41	**3**	23	4
mnrw	*569s*	69	13	13	5	*569s*	69	13	13	5
multi-objective with higher crossing weighting (hcw) - no initial variable assignment										
mtwh	*1283s*	29	6	14	5	*1283s*	29	6	14	5
mmwh	*550*	48	**3**	26	7	*550s*	48	**3**	26	7
mnrw	*1228s*	30	7	12	7	*1228s*	30	7	12	7
multi-objective with higher crossing weighting (hcw) - with initial variable assignment										
mtwh	*275s*	30	7	16	5	*8165s*	21	5	12	**3**
mmwh	*47s*	107	7	29	5	*47s*	107	7	29	5
mnrw	*383s*	63	10	13	5	*19379s*	57	11	13	**3**

Table 7.10: Solutions of Star Wars snippet instance from experiments based on the ILP model. Smallest metric values are displayed in bold.

*Answer to **E3***: For more than half of the experiments we didn't get a different solution after ten hours which we hadn't already after one hour for the current movie instance. Therefore there are less recognizable effects on result values for the optimization metrics where the corresponding metric was not included into the objective function. Altogether we have ten experiments with effects on metric values not included into the current objective. Seven effects are only positive with decreasing metric values. For one experiment we have a combination of a positive effect and a negative effect on another metric. The remaining two experiments with effects included only increasing values for metrics not included into the objective function.

*Answer to **E4***: There are not very essential differences in the solutions for the individual objective functions with and without initial variable assignments. For most cases the results without an initial variable assignment have slightly smaller metric values, but the differences are mostly rather minor.

*Answer to **E5***: The best wiggle objective over all objective function blocks is *mtwh* which produced the best solutions for 6 out of 8 blocks. The other two wiggle objectives *mmwh* and *mnrw* are equally efficient.

*Answer to **E6***: The Star Wars snippet instance is the first movie instance where the smallest values for metric *cc* is connected to a solution for a multi-objective with higher crossing weighting. All in all is the average value for *cc* definitely lower for the multi-objective solutions than for the results of single-objective experiments.

*Answer to **E7***: The higher weighting of crossings had clearly improved the values for metric *cc*. A positive effect on the wiggle metrics for higher wiggle weighting is not that obvious, here we have only minor improvements of the wiggle metric values. Apart from that is the quality between the solutions corresponding to multi-objectives rather similar.

7.2.4 Evaluation of Star Wars

Table 7.11 summarizes all results for the complete Star Wars movie instance which we will use to answer the Evaluation Questions **E1** to **E7** for the movie instance.

*Answer to **E1***: The best result for metric *twh* is 271, the best result for metric *hw* is 9 and the best solution for metric *cc* is 76, all three correspond to the solution of multi-objective *mtwh hcw* without initial variable assignment which is shown in Figure 7.17. The smallest value for metric *wc* is 93 and belongs to the solution of single-objective *mnrw* with initial variable assignment with the associated image in Figure 7.18. The worst result for metric *twh* after ten hours has value 2402 and belongs to single-objective *mmwh* with initial variable assignment, shown in Figure 7.19. The highest value of metric *hw* is 54 and the highest value of metric *cc* is 259 which both correspond to the same solution of single-objective *mnrw* with initial variable assignment shown in Figure 7.18. Lastly, the worst result for metric *wc* is 231 and belongs to the solution for multi-objective *mmwh sw* with initial variable assignment and corresponds to the image in Figure 7.20.

*Answer to **E2***: The solution for multi-objective *mtwh hcw* without initial variable assignment can be named as the best solution for the complete Star Wars instance which contains three smallest metric values. The associated image is illustrated in Figure 7.17. Although the solution for the single objective *mnrw* with initial variable assignment shown in Figure 7.18 holds the smallest values for metric *wc* it also corresponds to the highest results for the two metrics *hw* and *cc* and is therefore the worst solution for all metric values together for the movie instance.

*Answer to **E3***: The bigger part of the solutions for the Star Wars movie instance is corresponding to experiments with initial variable assignment. Thus, changes of the metric values from one hour to ten hours can only be observed for experiment with initial solutions. All in all is the number of positive effects on metrics, which were not included into the specific objective function predominant. From eleven experiments with at least one observable effect, only one experiments contained a negative effect where a metric values increased and two experiments, which contained one decreasing and one increasing metric. The remaining eight experiments included only positive effects where the other metric values stayed the same or decreased together with the metrics which where incorporated into the corresponding objective function.

*Answer to **E4***: Since there are only two solutions found by experiments without an initial solution this question cannot be answered as it should be. For big movie instances it is definitely advisable to apply initial variable assignments, even though the best solution is connected to an experiment without the use of an initial solution. With the right choice for the objective function a result with an acceptable quality can still be found like the solution after ten hours for single-objective *mtwh* with initial variable assignment shown in Figure 7.21.

*Answer to **E5***: Considering only experiment blocks which used initial solutions at the beginning of the computations the two wiggle objectives *mtwh* and *mnrw* are both connected to the best results and are therefore equally efficient. The worst solutions per block are produced by the third wiggle objective *mmwh*.

*Answer to **E6***: The quality of the solutions for single-objectives and multi-objectives is quite equal. The average values for metric *twh* and metric *wc* are smaller for solutions corresponding to single-objectives and the average value for metric *cc* is smaller for solutions of multi-objective experiments.

*Answer to **E7***: Comparing the solutions for the different multi-objective weighting strategies no significant difference in the quality of the solutions is recognizable.

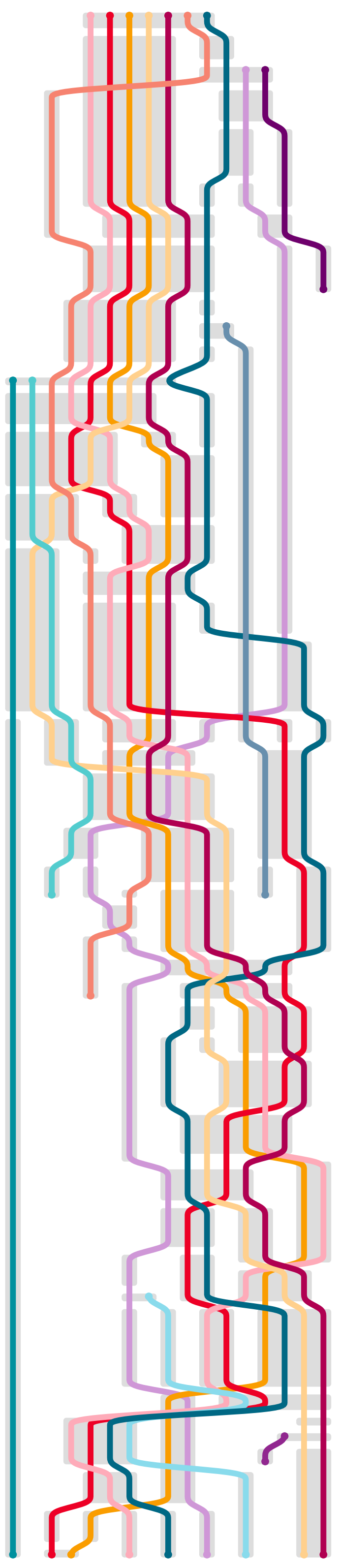


Figure 7.17: Star Wars solution for objective *mtwh hcw* with best result of metrics *twh*, *hw* and *cc* (twh:271, hw:9, wc:142, cc:76)

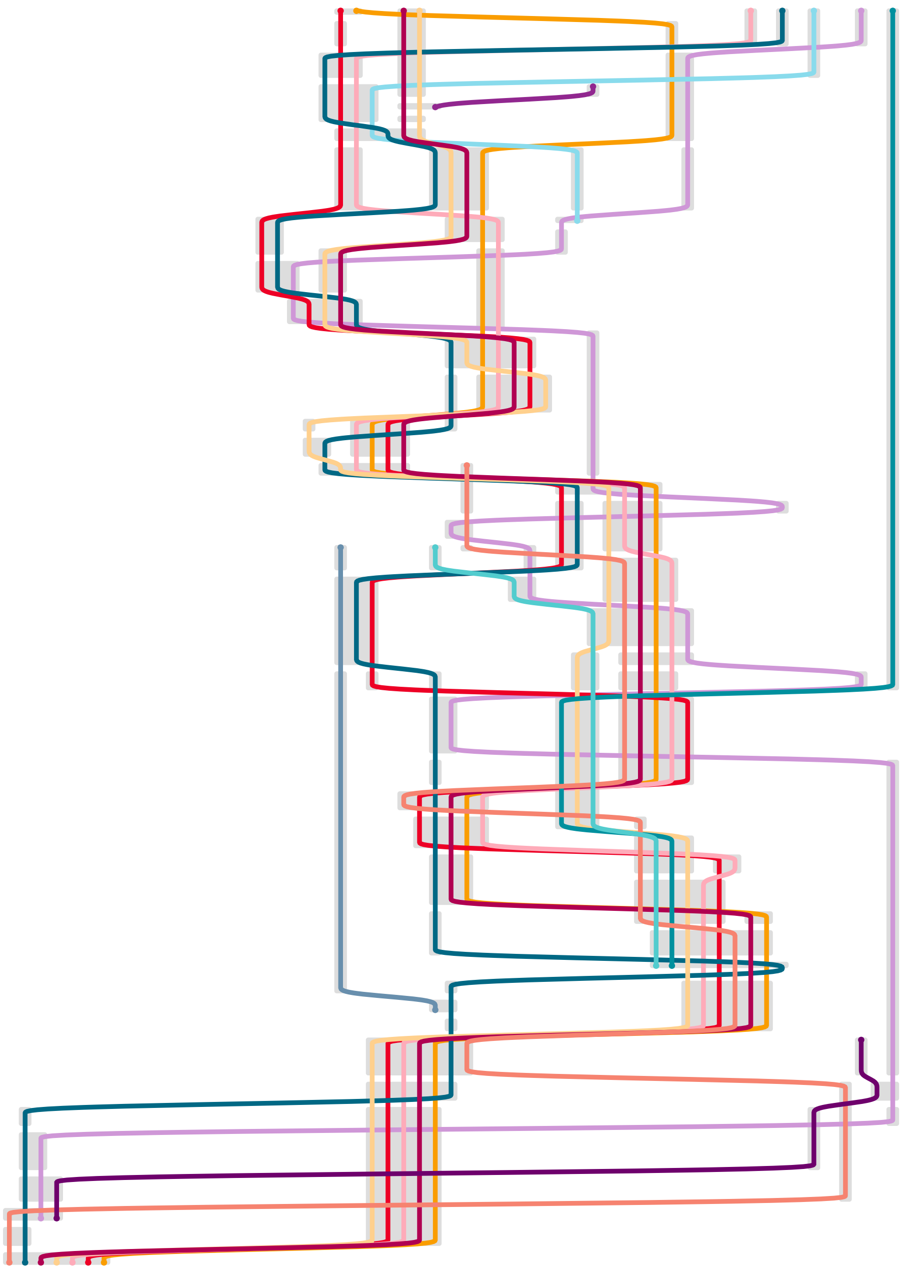


Figure 7.18: Star Wars solution for objective *mnrw* with best result of metric *wc* and worst result of metrics *hw* and *cc* (twh:1324, hw:54, wc:93, cc:259)

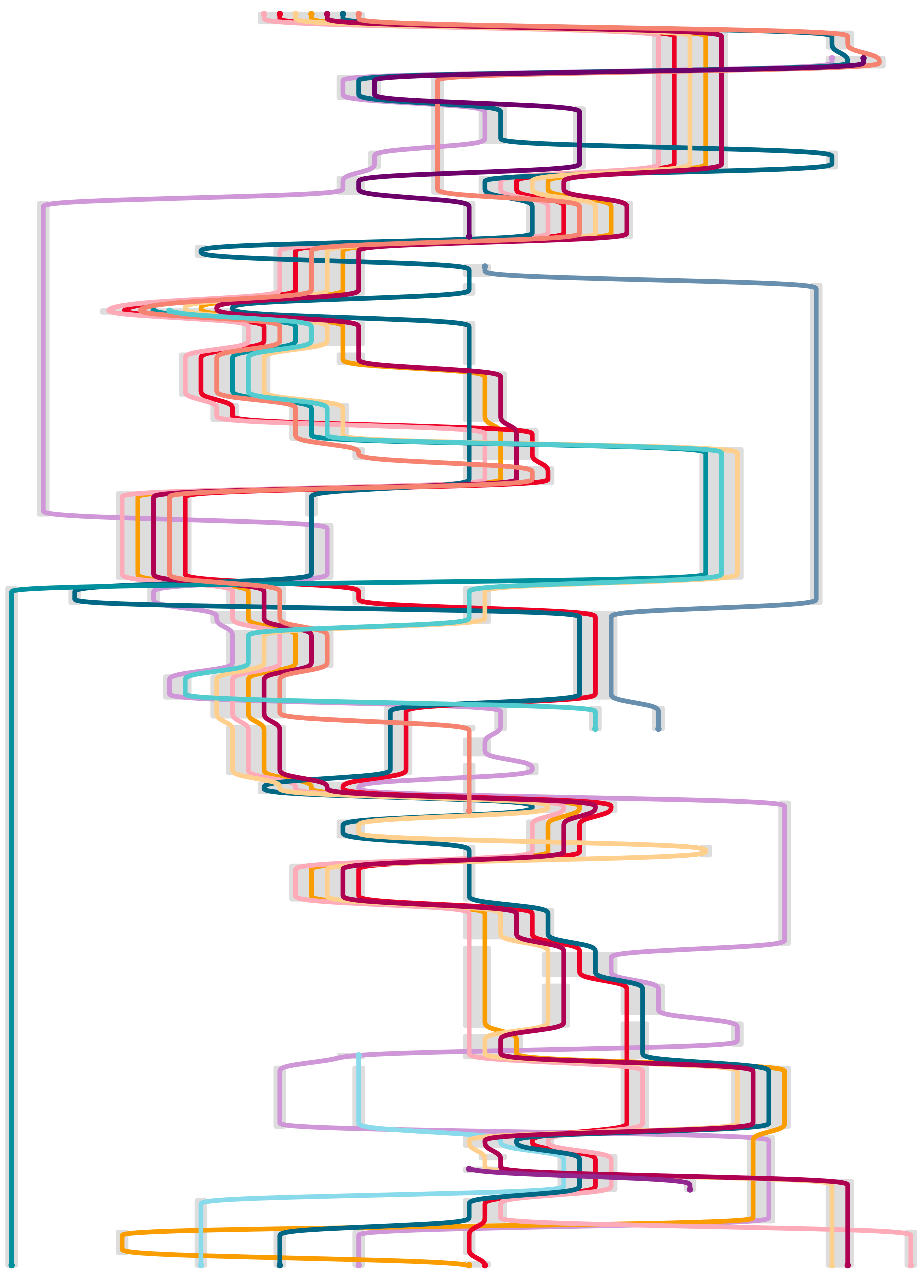


Figure 7.19: Star Wars solution for objective *mmwh* with worst result of metric *twh* (twh:2402, hw:44, wc:214, cc:217)

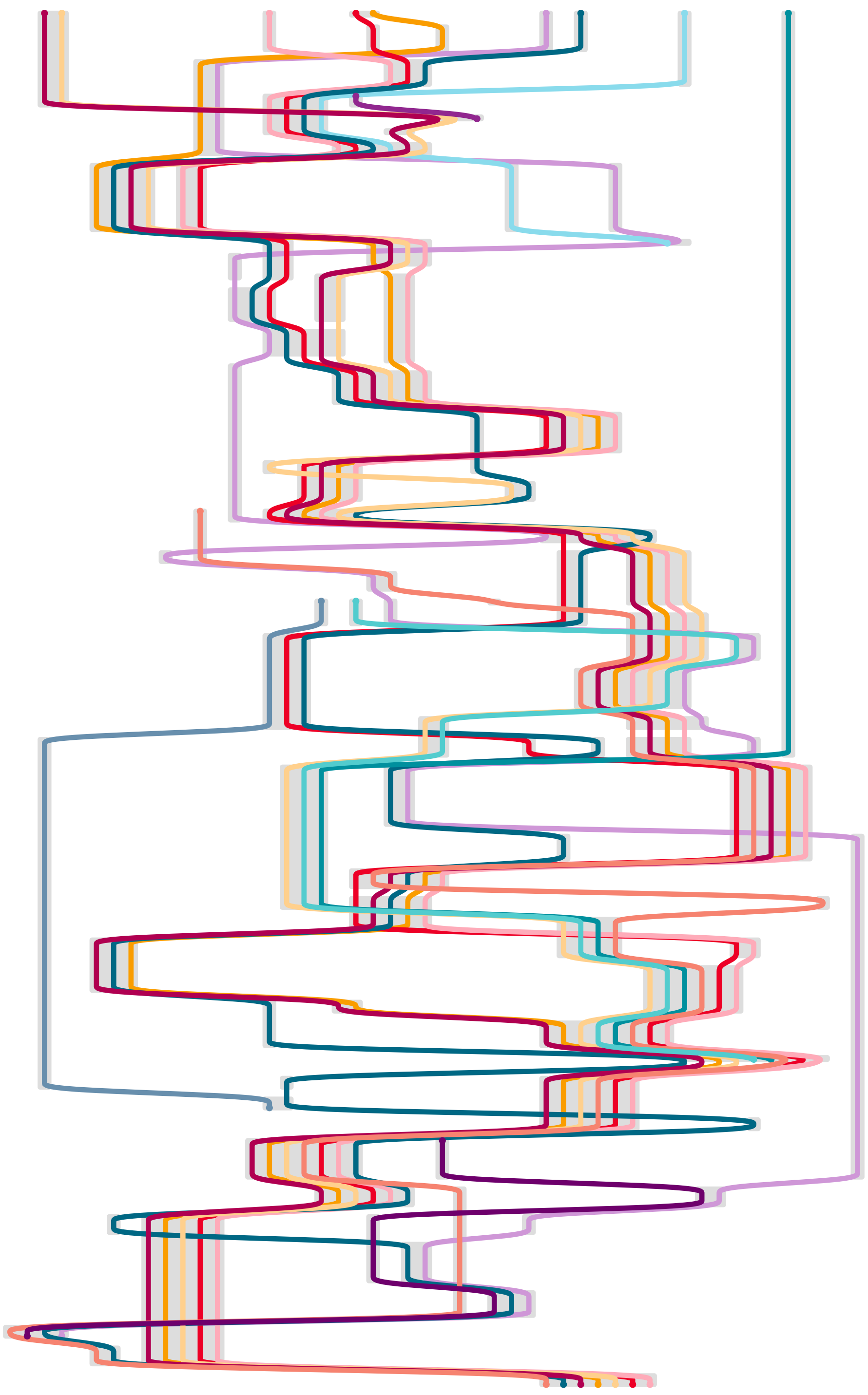


Figure 7.20: Star Wars solution for objective *mmwh sw* with worst result of metric *wc* (twh:2376, hw:27, wc:231, cc:184)

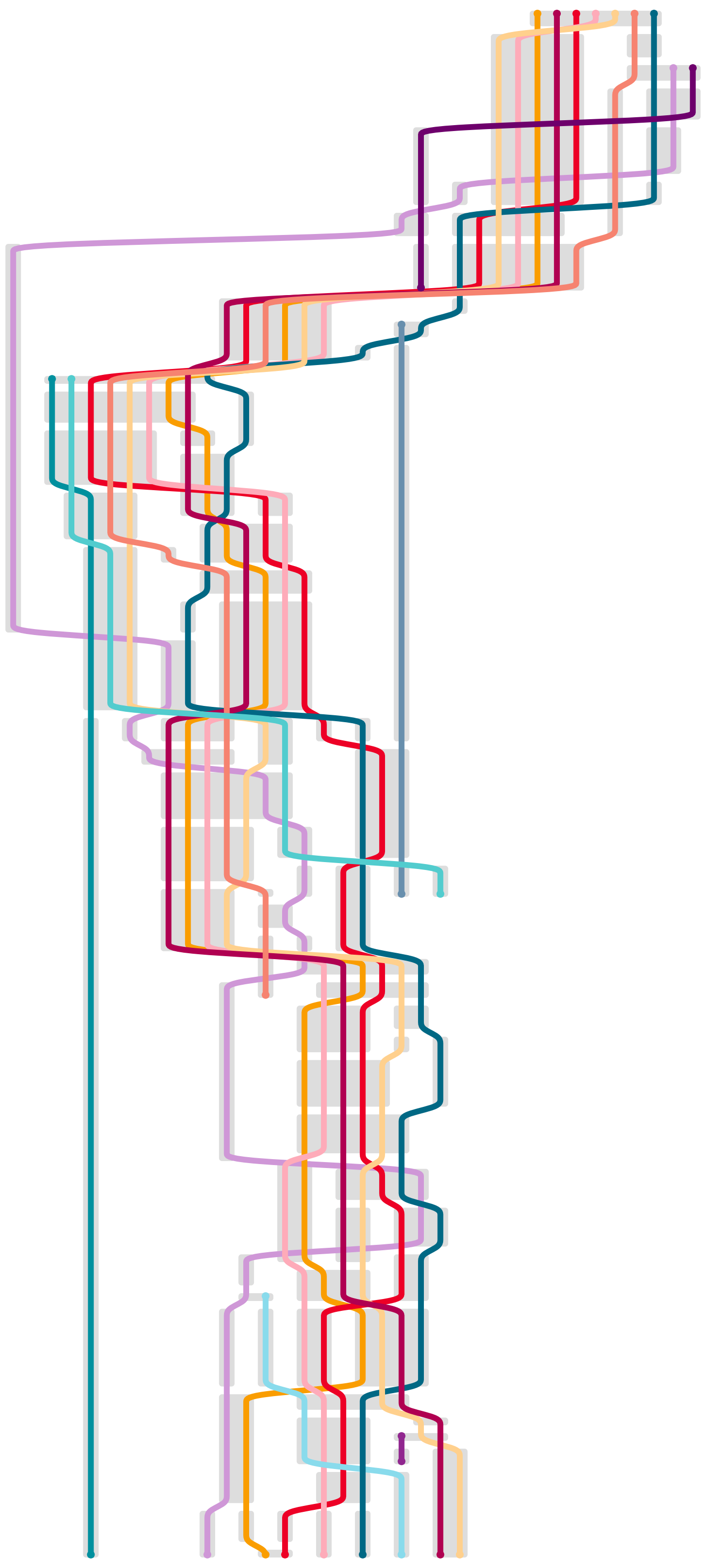


Figure 7.21: Star Wars solution for objective *mtwh* (twh:445, hw:20, wc:100, cc:167)

	1 hour					10 hours				
	time	twh	hw	wc	cc	time	twh	hw	wc	cc
wiggle minimization - no initial variable assignment										
mtwh	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
mmwh	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
mnrw	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
wiggle minimization - with initial variable assignment										
mtwh	*3253s*	670	22	119	211	*34097s*	445	20	100	167
mmwh	*2s*	2402	44	214	217	*2s*	2402	44	214	217
mnrw	*3598s*	1699	54	119	294	*24694s*	1324	54	**93**	259
multi-objective with same weighting (sw) - no initial variable assignment										
mtwh	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
mmwh	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
mnrw	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
multi-objective with same weighting (sw) - with initial variable assignment										
mtwh	*3419s*	1976	32	206	213	*35542s*	1286	26	183	199
mmwh	*2426s*	2667	36	225	202	*32816s*	2376	27	231	184
mnrw	*3533s*	2331	48	188	186	*35504s*	2010	45	149	175
multi-objective with higher wiggle weighting (hww) - no initial variable assignment										
mtwh	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
mmwh	N/A	N/A	N/A	N/A	N/A	*35894s*	673	12	199	226
mnrw	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
multi-objective with higher wiggle weighting (hww) - with initial variable assignment										
mtwh	*3242s*	2141	35	207	215	*35607s*	1415	28	198	213
mmwh	*3523s*	2098	34	206	215	*36000s*	1392	27	191	201
mnrw	*3401s*	2222	44	190	207	*34250s*	2225	47	164	199
multi-objective with higher crossing weighting (hcw) - no initial variable assignment										
mtwh	N/A	N/A	N/A	N/A	N/A	*35779s*	**271**	**9**	142	**76**
mmwh	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
mnrw	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
multi-objective with higher crossing weighting (hcw) - with initial variable assignment										
mtwh	*3491s*	2117	36	209	211	*36000s*	1444	25	192	192
mmwh	*2910s*	2503	30	226	198	*32310s*	2216	28	228	180
mnrw	*3457s*	2710	52	200	200	*28642s*	2184	52	159	172

Table 7.11: Solutions of Star Wars instance from experiments based on the ILP model. Smallest metric values are displayed in bold. *N/A* indicates that no solution was found after the specific computation time.

7.2.5 Evaluation over all movie instances

Answer to **E8**: The best solutions over all movie instances got produced by the wiggle objective *mtwh* for minimizing the total wiggle height. Regarding the minimization of the highest wiggle *mmwh* the experiments without an initial variable assignment did generate better results for the corresponding metric *hw*. The initial solutions used for the initial variable assignments contained much more slots than the associated ILP model instances without the use of an initial solution. Thus the experiments with the initial variable assignments allowed higher character jumps within the visualization matrix which got reflected in the resulting values of metric *hw*. For the two smaller movie instances of Inception and Star Wars the use of initial variable assignments didn't result in better solutions. However, for the complete movie instances we got much more solutions for the experiments by applying an initial solution. Lastly, mostly the multi-objective did produce better results for the corresponding metric *cc* of crossings count. By minimizing two metrics instead of one we got more solutions because if the minimization of one objective got stuck we still had a second objective to focus on.

7.3 Experiments with Max-SAT Formulation

In the following subsections we will analyze the results from the experiments based on our Max-SAT formulation from Section 5.3 for the Inception snippet instance and the Star Wars snippet instance and compare them to the solutions from the experiments based on our ILP formulation. We will analyze our results from our Max-SAT experiment again by answering our evaluation questions from the beginning of this chapter. Because the experiments based on the Max-SAT formulation didn't allow the extensive variations for wiggle objectives, multi-objectives, initial variable assignments and multiple solutions for every experiment we can only answer the Evaluation Questions **E1**, **E2**, **E5** and **E9** per movie instance and Question **E8** for both instances together in this section.

7.3.1 Evaluation of Inception snippet

The results for all Max-SAT experiments for the small Inception movie instance are summarized in Table 7.12. Based on this table we will answer the evaluation questions from above.

Answer to **E1**: For all four Max-SAT solvers we got for both objectives the optimal solutions which also means that our smallest result values for the metrics *twh* and *wc* are the optimal values for the two metrics of the Inception snippet instance. The smallest value for metric *twh* is 39, which is also the smallest value of the metric from the ILP experiments. The smallest value for metric *hw* is 3 which is slightly higher as the corresponding smallest result for the metric from the ILP experiments. For metric *wc* the smallest and also optimal value is 23 which is lower than the best result for the metric from the ILP experiments. For metric *cc* the smallest result from the Max-SAT experiments is 8 which is higher than the best result for the metric from the ILP experiments. All in all

is the difference between the smallest metric values from the Max-SAT experiments and the smallest metric values from the ILP experiments quite small.
The difference between the highest metric values from Max-SAT and ILP are much higher. The highest result from Max-SAT for metric *twh* is 60, for metric *hw* it is 6, for metric *wc* it is 27 and for metric *cc* is is 24. All this four worst metric results are much lower than the worst metric values from the ILP experiments.
Comparing the average metric values for the Inception snippet instance from both solution models we get three out of four average metric values from the Max-SAT experiments which are smaller than the corresponding average values from the ILP experiments. Only for the metric *cc* we got a better average result for the ILP model.

Answer to **E2**: The best storyline visualization solution with three smallest metric values was produced by the Max-SAT solver *Loandra* with the objective *mtwh* and is shown in Figure 7.22. The worst solution got produced by Max-SAT solver *Maxino* for objective *mnrw* with three highest metric values over all Max-SAT experiment results and is shown in Figure 7.23.

Answer to **E5**: Over all solutions of the Max-SAT solvers the wiggle objective *mtwh* did produce better results for the other three metrics *hw*, *wc* and *cc* which were not included into the objective than the objective *mnrw* which mostly only provided the smallest value for its objective metric *wc*.

Answer to **E9**: All used Max-SAT solvers did calculate for both objectives the same optimal value for the corresponding metric which was directly incorporated into the current objective. The minimal value of metric *twh* with the corresponding objective *mtwh* is 39. For objective *mnrw* the associated minimized metric *wc* holds value 23. All metrics not part of the current objective did vary between the different Max-SAT solvers. The biggest variations are recognizable for metric *twh* and metric *cc* while the other two metrics *hw* and *wc* vary very little. Comparing the metric values of all solutions and the time for the different solvers with each other none of the Max-SAT solver can be named as the best or worst solver for the Inception snippet instance.

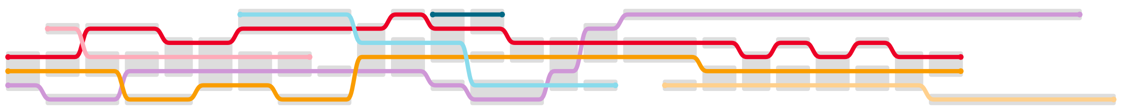


Figure 7.22: Best solution of Inception snippet for Max-SAT with objective *mtwh* from Max-SAT solver *Loandra* (twh:39, hw:3, wc:27, cc:8)

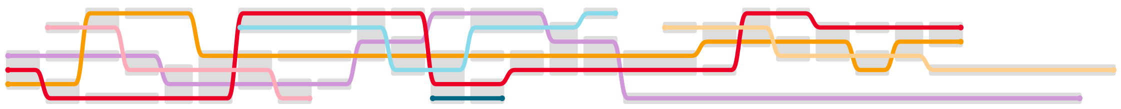


Figure 7.23: Worst solution of Inception snippet for Max-SAT with objective *mnrw* from Max-SAT solver *Maxino* (twh:60, hw:6, wc:23, cc:24)

wiggle obj	time	twh	mwh	wc	cc
max-sat solver	*LMHS*				
mtwh	*1m 45s 559ms*	**39**	5	27	**8**
mnrw	*2s 205ms*	60	6	**23**	23
max-sat solver	*Loandra*				
mtwh	*25m 39s 913ms*	**39**	**3**	27	**8**
mnrw	*388ms*	58	5	**23**	24
max-sat solver	*MaxHS*				
mtwh	*54s 412ms*	**39**	5	25	**8**
mnrw	*9s 483ms*	56	6	**23**	20
max-sat solver	*Maxino*				
mtwh	*1m 8s 900ms*	**39**	5	25	**8**
mnrw	*578ms*	60	6	**23**	24

Table 7.12: Solutions of Inception snippet movie instance from experiments based on the Max-SAT formulation. Smallest metric values are displayed in bold.

7.3.2 Evaluation of Star Wars snippet

Table 7.13 contains all results of the Max-SAT experiments for the Star Wars snippet instance. *N/A* in the table indicates that no solution was found after ten hours. With this table of metric values we are again able to answer the evaluation questions from above.

Answer to **E1**: Again we got for almost all Max-SAT solvers and objectives the same optimal metric value for the metric currently incorporated into the objective. We got for metric *twh* the value 19 as smallest value which is also the smallest value for the ILP experiments. The best result for metric *hw* holds value 6 which is higher than the smallest value for the metric from the ILP experiments. Metric *wc* has value 8 as best result value which is smaller than the best result for *wc* produced by the corresponding ILP experiments. The best value for metric *cc* is 3 which is equal to the best result for the metric from the ILP experiments.
The highest result for metric *twh* has value 41, for metric *hw* the worst solution holds value 12, for metric *wc* the highest value is 12 and for metric *cc* it is 22. All of these four worst results for the metrics are still much better than the highest metric values from the ILP experiments.
By the comparison of the average values for all four metrics between the two solution models we have smaller average values for the two metrics *twh* and *wc* from the Max-SAT experiments. For the remaining two metrics *hw* and *cc* we got better average result values from the ILP experiments.

Answer to **E2**: The best solution over all experiments for the Star Wars snippet instance got produced by Max-SAT solver *MaxHS* containing three smallest metric values with *mtwh* as objective. The corresponding image is shown in Figure 7.24. As worst solution over all solutions for the movie instance we got two solutions for objective *mnrw* from the Max-SAT solvers *Loandra* and *Maxino* which have exactly the same values for all four metrics. Because the solver *Loandra* needed slightly longer to produce the solution we name it as the worst solution with three highest metric values. Figure 7.25 shows the associated image of the worst solution.

Answer to **E5**: It is clear to see that the objective for minimizing the total wiggle height *mtwh* produced better solutions than the objective for minimizing the number of wiggles *mnrw*. Objective *mtwh* delivered three out of four best metric values. Only for metric *wc* objective *mnrw* produced a better result.

Answer to **E9**: With the exception of the experiment of Max-SAT solver *LMHS* with objective *mtwh* where we didn't get a solution within ten hours, all other experiments delivered the same minimal value for the metric currently directly incorporated into the objective formulation. For the solutions of objective *mtwh* the corresponding minimized metric *twh* has value 19 and for the solutions of objective *mnrw* the corresponding minimized metric *wc* has value 8. For the metrics, which were not incorporated directly in the current objective formulation, we got more variations for the resulting metric values. None of the four solvers can be named as the best choice for the Star Wars snippet instance, only the Max-SAT solver *LMHS* is clearly the worst solver for the current movie instance.

wiggle obj	time	twh	hw	wc	cc
max-sat solver	*LMHS*				
mtwh	N/A	N/A	N/A	N/A	N/A
mnrw	*15m 1s 196ms*	33	10	**8**	17
max-sat solver	*Loandra*				
mtwh	*24m 12s 276ms*	**19**	**6**	10	5
mnrw	*9s 418ms*	41	12	**8**	22
max-sat solver	*MaxHS*				
mtwh	*1h 41m 35s 375ms*	**19**	**6**	12	**3**
mnrw	*2m 12s 595ms*	33	10	**8**	17
max-sat solver	*Maxino*				
mtwh	*1h 29m 24s 676ms*	**19**	**6**	11	4
mnrw	*9s 309ms*	41	12	**8**	22

Table 7.13: Solutions of Star Wars snippet movie instance from experiments based on the Max-SAT formulation. Smallest metric values are displayed in bold. *N/A* indicates that no solution was found after ten hours.

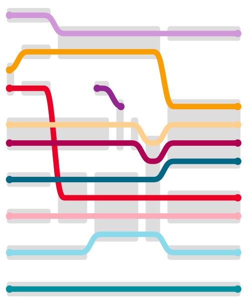


Figure 7.24: Best solution of Star Wars snippet for Max-SAT with objective *mtwh* from Max-SAT solver *MaxHS* (twh:19, hw:6, wc:12, cc:3)

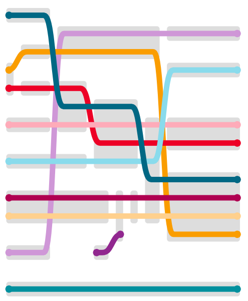


Figure 7.25: Worst solution of Star Wars snippet for Max-SAT with objective *mnrw* from Max-SAT solver *Loandra* (twh:41, hw:12, wc:8, cc:22)

7.3.3 Evaluation over all movie instances

Answer to **E8**: For both movie instances the resulting solutions for objective *mtwh* are superior to the solutions for objective *mnrw* regarding the optimization metrics.

7.3.4 Overall comparison of ILP and Max-SAT experiments

Finally for this chapter it remains to *answer Evaluation Question* **E10**:

The best wiggle objective for both solution models was *mtwh* for minimizing the total wiggle height which produced the smallest metric values irrespective of the solution model. By comparing the best solutions for the individual movie instances between the ILP formulation and the Max-SAT formulation, none of the two formulations produced storyline visualizations where the best results from both solution models have a distinct difference regarding the quality. Therefore both optimization approaches can produce solutions of similar quality.

By comparing the worst solutions from both formulations the difference in the solution qualities is more apparent. This difference is even then visible if we only consider solutions from the ILP experiments connected to the single objectives *mtwh* and *mnrw* without initial variable assignment. This can be explained by the fact that all solutions we got from our Max-SAT experiments finished the computation with an optimal value of the metric incorporated into the current objective. For none of the ILP experiments we have the certainty that we computed the optimal solution for a specific objective function. Therefore it is understandable that ILP produces more solutions, which are inferior to the solutions for the Max-SAT experiments.

Each of the formulations has an advantage over the other solution model. All solutions produced by Max-SAT solvers have a much lower runtime than the solutions from the ILP experiments. The ILP model on the other hand enables to formulate multi-objectives and initial variable assignments for big movie instances.

CHAPTER 8

Conclusion

Both solutions models for ILP and Max-SAT from Chapter 5 are suited for minimizing wiggles in storyline visualizations.

Our wiggle objective variations for minimizing the total wiggle height (*mtwh*), minimizing the number of wiggles (*mnrw*) and minimizing the maximum wiggle height (*mmwh*) can all be used for producing good quality solutions of storyline visualizations. But the quality of a found solution depends on more factors than just the choice of the wiggle minimization objective. The currently used movie instance and a potential combination with crossing minimization can also be significant for the quality of a solution.

Though, without considerations of instance properties or potential multi-objective combinations, we can name the wiggle minimization objective *mtwh* for minimizing the total wiggle height as the best choice for wiggle minimization in storyline visualizations. This wiggle minimization objective produced good quality storyline visualizations for all of our four movie instances and both solution models.

The comparison of the solutions for the wiggle objective for minimizing the maximum wiggle height (*mmwh*), with initial variable assignment and without initial variable assignment, has shown that it is more efficient to reduce the maximum height of wiggles by reducing the number of slots of the visualization matrix instead of formulating an objective for minimizing the maximum wiggle height. If only the minimal number of necessary slots is used then it is no longer possible for characters to make high jumps. Our experiments with initial variable assignments required much more slots, because our initial solutions contained high wiggles, which was a disadvantage for maximum wiggle height minimization just because we had the possibility for high wiggles. By reducing the slots at the beginning we erase the possibility for high wiggles.

A big disadvantage of wiggle minimization is the necessary knowledge about the absolute positions of all characters and meetings within the visualization matrix. For wiggle minimization we need the absolute position of the characters for determining the exact height of wiggles so that we are able to minimize the total wiggle height or minimize the maximum wiggle height. If we would only consider relative positions of characters and meetings we would require much less variables and constraints as with absolute positions, but we would also only be able to calculate the number of all wiggles. The knowledge about the exact heights of the wiggles would be lost.

Comparisons of absolute positions of characters and meetings at every time points entail also disadvantages for movie instances with a higher number of parallel meetings. Especially in our Max-SAT formulation we have much more variables and clauses if a movie instance has more characters and parallel meetings and requires therefore more height in the visualization matrix. This gets evident by comparing the concrete Max-SAT instances in Table 7.3 for our two snippets of the movies Inception and Star Wars from Table 7.2. The Max-SAT instance of the Star Wars snippet has much more variables and constraints, because it has more parallel meetings and more meeting with a higher number of members than the Inception snippet instance, even though the Star Wars snippet has less time point than the Inception snippet.

During the analysis of our experiments based on our two solution models we also observed that the number of crossings get reduced along with the wiggles during the minimization of wiggles in storyline visualizations. More direct crossing minimizations could be achieved with a multi-objective of a combined wiggle and crossing minimization. With such a multi-objective we obtained solutions with even less crossings than the solutions of single-objectives for wiggle minimization only. Nonetheless did the single-objectives for wiggle minimization calculate more solutions which were superior to the solutions from multi-objectives.

For small movie instances the Max-SAT formulation did produce the solutions much faster than the ILP formulation. But for big movie instances the required number of variables and constraints would get too high for the Max-SAT formulation. The ILP formulation with the possibilities of initializing all variables and constraints at the beginning of the computation makes it easier to generate solutions for big movie instances.

Although we weren't able to calculate the optimal solutions in the ILP experiments within ten hours for none of our movie instances, we still generated good quality solutions for large movie instances by applying initial solutions at the beginning of the computations. With our Max-SAT formulation we could only use small snippets of our movie instances for which we got optimal solutions for almost all used Max-SAT solvers.

For future work about optimizing storyline visualizations further development for wiggle minimization with focus on more efficiency regarding big movie instances can be done. Optimization approaches for minimizing wiggles based on algorithms are still missing as well. Also the question if the wiggle minimization problem is NP-hard has not yet been answered in the current literature. Furthermore the realization of a user study is still open which incorporates storyline visualizations produced by different optimization methods and different objective functions. Such a user study should enable the comparison and evaluation of the legibility of different storyline visualization solutions.

List of Figures

List of Tables

Bibliography

[1] MaxSAT Evaluation 2017. Input format (accessed 2018-09-04). `http://mse17.cs.helsinki.fi/rules.html#input`.

[2] MaxSAT Evaluation 2017. Participating solvers (accessed 2018-09-04). `http://mse17.cs.helsinki.fi/descriptions.html`.

[3] Ignasi Abío and Peter J Stuckey. Encoding Linear Constraints into SAT. In *International Conference on Principles and Practice of Constraint Programming*, volume 8656 of *LNCS*, pages 75–91. Springer, 2014.

[4] Algorithms and Complexity Group TU Wien. Grid Engine (accessed 2018-09-04). `https://www.ac.tuwien.ac.at/students/grid-engine/`.

[5] Mario Alviano. Maxino. *MaxSAT Evaluation 2017: Solver and Benchmark Descriptions*, page 10.

[6] Carlos Ansótegui and Felip Manyà. Mapping Problems with Finite-Domain Variables to Problems with Boolean Variables. In *International conference on theory and applications of satisfiability testing*, volume 3542 of *LNCS*, pages 1–15. Springer, 2004.

[7] Josep Argelich and Felip Manyà. Exact Max-SAT solvers for over-constrained problems. *Journal of Heuristics*, 12(4-5):375–392, 2006.

[8] Fahiem Bacchus. MaxHS v3. 0 in the 2017 MaxSat Evaluation. *MaxSAT Evaluation 2017: Solver and Benchmark Descriptions*, page 8.

[9] Sergey Bereg, Alexander E Holroyd, Lev Nachmanson, and Sergey Pupyrev. Drawing Permutations with Few Corners. In *International Symposium on Graph Drawing*, volume 8242 of *LNCS*, pages 484–495. Springer, 2013.

[10] Sergey Bereg, Alexander E Holroyd, Lev Nachmanson, and Sergey Pupyrev. Representing Permutations with Few Moves. *SIAM Journal on Discrete Mathematics*, 30(4):1950–1977, 2016.

[11] Jeremias Berg, Tuukka Korhonen, and Matti Järvisalo. Loandra: PMRES Extended with Preprocessing Entering MaxSAT Evaluation 2017. *MaxSAT Evaluation 2017: Solver and Benchmark Descriptions*, page 13.

[12] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

[13] Tao Chen, Aidong Lu, and Shi-Min Hu. Visual storylines: Semantic visualization of movie sequence. *Computers & Graphics*, 36(4):241–249, 2012.

[14] Tuan Nhon Dang, Nick Pendar, and Angus Graeme Forbes. TimeArcs: Visualizing Fluctuations in Dynamic Networks. In *Computer Graphics Forum*, volume 35, pages 61–69. Wiley Online Library, 2016.

[15] Jessica Davies and Fahiem Bacchus. MaxHS: A fast and robust MaxSAT solver (accessed 2018-09-04). `http://www.maxhs.org`.

[16] Ian P Gent and Peter Nightingale. A New Encoding of AllDifferent into SAT. In *International Workshop on Modelling and Reformulating Constraint Satisfaction*, pages 95–110, 2004.

[17] Martin Gronemann, Michael Jünger, Frauke Liers, and Francesco Mambelli. Crossing Minimization in Storyline Visualization. In Yifan Hu and Martin Nöllenburg, editors, *Graph Drawing and Network Visualization (GD 2016)*, volume 9801 of *LNCS*, pages 367–381. Springer, 2016.

[18] Pierre Hansen and Brigitte Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44(4):279–303, 1990.

[19] Yuejun Jiang, Henry Kautz, and Bart Selman. Solving Problems with Hard and Soft Constraints Using a Stochastic Algorithm for MAX-SAT. In *1st International Joint Workshop on Artificial Intelligence and Operations Research*, page 20, 1995.

[20] Nam Wook Kim, Stuart K Card, and Jeffrey Heer. Tracing genealogical data with TimeNets. In *Proceedings of the International Conference on Advanced Visual Interfaces*, pages 241–248. ACM, 2010.

[21] Irina Kostitsyna, Martin Nöllenburg, Valentin Polishchuk, André Schulz, and Darren Strash. On Minimizing Crossings in Storyline Visualizations. In *International Symposium on Graph Drawing and Network Visualization*, volume 9411 of *LNCS*, pages 192–198. Springer, 2015.

[22] Daniel Kroening and Ofer Strichman. *Decision procedures.* Springer, 2016.

[23] Shixia Liu, Yingcai Wu, Enxun Wei, Mengchen Liu, and Yang Liu. StoryFlow: Tracking the Evolution of Stories. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2436–2445, 2013.

[24] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[25] Chris W Muelder, Tarik Crnovrsanin, Arnaud Sallaberry, and Kwan-Liu Ma. Egocentric storylines for visual analysis of large dynamic graphs. In *Big Data, 2013 IEEE International Conference on*, pages 56–62. IEEE, 2013.

[26] R Munroe. Xkcd# 657: Movie narrative charts (accessed 2018-09-04). `https://xkcd.com/657/`, 2009.

[27] George L Nemhauser and Laurence A Wolsey. *Integer and Combinatorial Optimization.* Wiley, 2014.

[28] Michael Ogawa and Kwan-Liu Ma. Software evolution storylines. In *Proceedings of the 5th international symposium on Software visualization*, pages 35–42. ACM, 2010.

[29] Gurobi Optimization. Gurobi Optimizer version 7.0 (accessed 2018-09-04). `http://www.gurobi.com`.

[30] Catherine Plaisant, Brett Milash, Anne Rose, Seth Widoff, and Ben Shneiderman. Lifelines: Visualizing Personal Histories. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 221–227. ACM, 1996.

[31] Catherine Plaisant, Richard Mushlin, Aaron Snyder, Jia Li, Dan Heller, and Ben Shneiderman. Lifelines: Using Visualization to Enhance Navigation and Analysis of Patient Records. In *The Craft of Information Visualization*, pages 308–312. Elsevier, 2003.

[32] Paul Saikko, Jeremias Berg, and Matti Järvisalo. Lmhs: A SAT-IP Hybrid MaxSAT Solver. In *International Conference on Theory and Applications of Satisfiability Testing*, volume 9710 of *LNCS*, pages 539–546. Springer, 2016.

[33] Paul Saikko, Tuukka Korhonen, Jeremias Berg, and Matti Järvisalo. LMHS in MaxSAT Evaluation 2017. *MaxSAT Evaluation 2017: Solver and Benchmark Descriptions*, page 16.

[34] Yuzuru Tanahashi, Chien-Hsin Hsueh, and Kwan-Liu Ma. An Efficient Framework for Generating Storyline Visualizations from Streaming Data. *IEEE transactions on visualization and computer graphics*, 21(6):730–742, 2015.

[35] Yuzuru Tanahashi and Kwan-Liu Ma. Design Considerations for Optimizing Storyline Visualizations: Data and Resources (accessed 2018-09-04). `https://old.datahub.io/dataset/vis-storyline-visualizations`.

[36] Yuzuru Tanahashi and Kwan-Liu Ma. Design Considerations for Optimizing Storyline Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2679–2688, 2012.

[37] Thomas C van Dijk, Martin Fink, Norbert Fischer, Fabian Lipp, Peter Markfelder, Alexander Ravsky, Subhash Suri, and Alexander Wolff. Block Crossings in Storyline Visualizations. In *International Symposium on Graph Drawing and Network Visualization (GD 2016)*, volume 9801 of *LNCS*, pages 382–398. Springer, 2016.

[38] Thomas C van Dijk, Fabian Lipp, Peter Markfelder, and Alexander Wolff. Computing Storyline Visualizations with Few Block Crossings. In *International Symposium on Graph Drawing and Network Visualization (GD 2017)*, volume 10692 of *LNCS*, pages 365–378. Springer, 2017.

[39] Toby Walsh. SAT v CSP. In *International Conference on Principles and Practice of Constraint Programming*, volume 1894 of *LNCS*, pages 441–456. Springer, 2000.

[40] Colin Ware. *Information visualization: perception for design.* Elsevier, 2012.

[41] Laurence A Wolsey. *Integer Programming.* Wiley, 1998.