Abstract

Author's note: This version of my bachelor thesis was subsequently revised, primarily to correct spelling mistakes – it is therefore not the version that was submitted. Changes have also been made in some places to make the content easier to understand.

Alexander Kutscheid gained experiences with the minimization of crossings in point-set embeddings of graphs within a previous study. In particular, a combined application of force-directed algorithms and the global optimization method Simulated Annealing reliably yielded satisfactory results. The Graph Drawing Contest, a traditional event within the Symposium on Graph Drawing and Network Visualization, is dedicated to the same problem in this year. With the goal of a competitive participation, this bachelor thesis adapts the combined approach, as described above, and investigates it further. Therefore, new sub-strategies are introduced and compared with those already investigated. These comparisons are based on a series of tests, which were carried out on the available drawings from the last competition. During these tests, the force-directed algorithm of Fruchterman and Reingold has achieved significantly better intermediate results than the original spring embedder of Eades. In the context of Simulated Annealing, different incremental modification techniques were compared. For some drawings, good results were obtained much faster when neighbouring vertices were strictly placed close to another. In other cases, this rule caused significantly poorer results. In contrast to this technique, the application of random modifications to single vertices has consistently generated satisfactory results. Further improvements were achieved for both refactoring methods through a dynamically calculated probability, with which the vertices, that are to be modified, were selected. These calculations were based on the number of crossings, in which a vertex's adjacent edges are involved. A side product of this thesis is an object-oriented C++ console application, in which all considered algorithms are implemented. Even though the library exhibits runtime deficits, it was still able to contribute to our team achieving the second place in the competition of 2024.

Contents

1. Introduction

1.1. Motivation

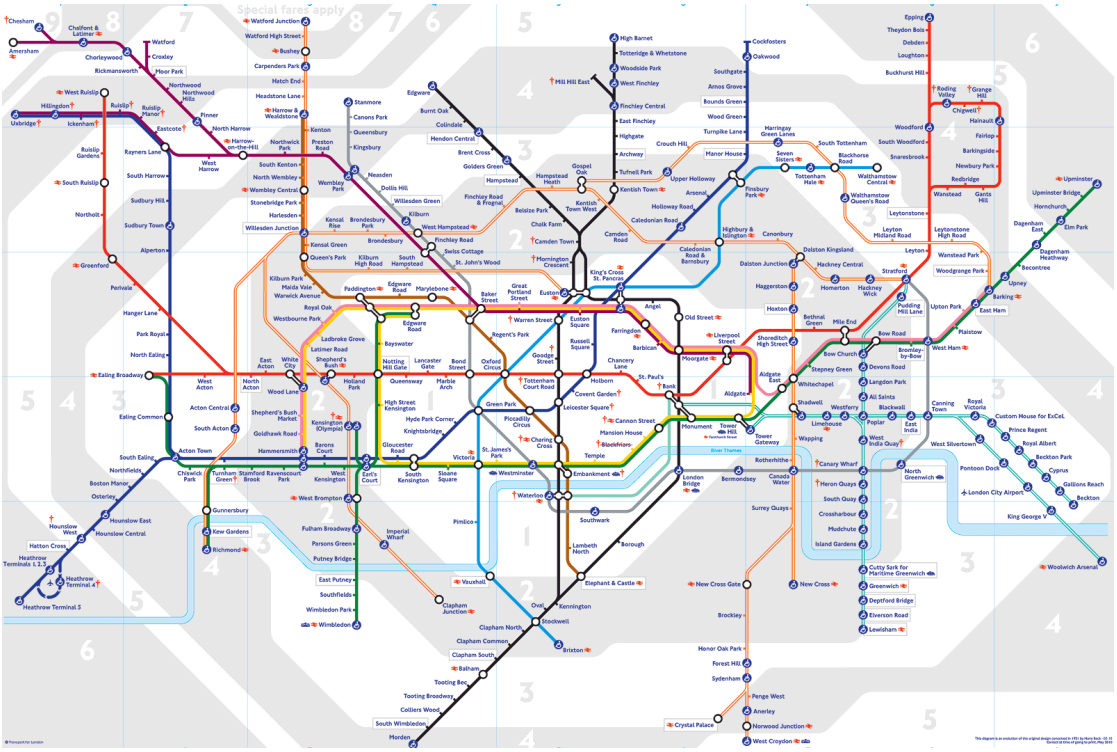


Fig. 1.1.: Map of London's famous underground railway network: the Tube.

Graphs are mathematical structures, that are present in everyday life. Most commonly, we encounter them in their visual representation and unconsciously. For instance, the typical maps of regional transport networks can be understood as graphs [Wol07], such as shown in Figure 1.1. Besides that, graphs also play a crucial role in addressing a wide range of difficulties in the domain of application development. Language processing, routing in computer networks and search engines are only three examples of many services that strongly depend on graph problems [PP13]. All these areas in which we encounter them share one similarity: the concept of graphs and their visual depictions help to improve the examination and understanding of complex topics. Therefore, it is equally important for both customers and developers to visualize graphs in an aesthetically pleasing manner, that effectively conveys all essential information.

Out of this motivation, an entire sub-discipline became established within the Graph Theory: *Graph Drawing* – the (algorithmic) transformation of formally defined graphs into visual representations [BKMW24]. In fact, many interesting and versatile problems can be assigned to the discipline. Some of these are of such complexity, that they serve as inspiration for the annually *Graph Drawing Contest* (GDC), which is a traditional part of the *Symposium on Graph and Network Visualization* [BKMW24]. This year's challenge demands the contestants to minimize the number of crossings in *point-set embeddings* (PSE) of graphs [BKM+24a]. Therefore, a graph's vertex-set has to be assigned on a set of points, such that the, as straight lines drawn, edges have as few crossings as possible.

This bachelor thesis outlines all the necessary preparations made to take part in the last GDC, which took place on September 18th, 2024 [BKM+24a]. Finally, its culmination was to achieve a successful participation in the contest.

1.2. Objectives

As part of the preparations, a selection of conventional strategies capable of solving the defined problem will be investigated. Especially interesting for addressing the GDC are algorithms, which can be scaled up and consistently produce favourable outcomes. Thus, a further prerequisite for the approaches in question is the ability to process very large and complex graphs as well. Alexander Kutscheid already gained experiences in this task during a prior bachelor thesis. He demonstrated, that the combination of *force-directed algorithms* (FDA) and *Simulated Annealing* (SA) is a promising and memory-efficient methodology, which satisfies the specified requirements [Kut24]. Another comfortable aspect is, that the point-sets do not have to fulfil any special characteristics for it to be applicable. Consequently, this thesis adheres to this particular approach in essence.

Moreover, Kutscheid's study offers a good starting point, since his implementations were also tested with the drawings from last year's competition. Although he was generally able to achieve above-average solutions, some of the results were significantly worse than those of the previous year's winner [Kut24, BKK+23b]. This suggests, that there is a potential for further improvement. In regard to this, the present thesis examines further related sub-strategies, optimizes the already assessed ones and compares the new results with those of Kutscheid. An particularly prominent edge-case among last year's tasks was, that the point-set was arranged in a linear manner [Kut24]. Consequently, this should be given special consideration in the elaborations of this thesis.

Finally, the discussed approaches were combined into a comprehensive toolbox, which then was utilized during the GDC 2024. Therefore, a C++ header-only library is developed along with this thesis. The practical objective was to implement a flexible toolbox, that enables a quick selection and sequential execution of various strategies. This helped to respond quickly on the GDC's drawings and to achieve some good results.

2. Definitions and background

On the following pages, all concepts and vocabulary necessary for the understanding of the contents are described and defined. Moreover, the leitmotif of this thesis is outlined on the basis of previous works.

2.1. Definitions

Definition 1. *A graph $G = (V, E)$ is an ordered pair consisting of two finite sets V and E, such that $E \subseteq V \times V$. Typically, $v \in V$ are referred to as vertices and $e \in E$ as edges. Another common shorthand notation is $G(V, E)$.* [Pat13]

Definition 2. *$G(V, E)$ is called undirected, if $(e_1, e_2) \in E \Rightarrow (e_2, e_1) \in E$ always holds. Otherwise, the graph is directed.* [Pat13]

Definition 3. *A graph $G(V, E)$ is simple, if $(v_1, v_2) \in E \Rightarrow v_1 \neq v_2$ applies.* [Pat13]

Definition 4. *Two vertices $v_1, v_2 \in V$ of a Graph $G(V, E)$ are called neighbours or adjacent, if an edge $e \in E$ exists, such that $e = (v_1, v_2)$. The neighbourhood-function $N : V \to 2^V$ returns all neighbours of a vertex.* [Die00]

This thesis focuses only on simple and undirected graphs, similar to the GDCs of 2023 and 2024 [BKK+23a, BKM+24a]. Therefore, the edges of a graph $G(V, E)$ can be interpreted as unordered tuples [Pat13], meaning that $e = (v_1, v_2) = (v_2, v_1)$ applies for each edge $e \in E$. Parallel to Definition 4, an edge $(v_1, v_2) \in E$ will also be called adjacent to $v \in V$, if $v_1 = v$ or $v_2 = v$ applies.

Definition 5. *The degree of a vertex is defined as* $\deg(v) = |N(v)|$. [Die00]

Definition 6. *"A curve is a* [multi-]*subset of $\mathbb{R}^2$ of the form $\alpha = \{\gamma(x) : x \in [0, 1]\}$, where $\gamma : [0, 1] \to \mathbb{R}^2$ is a continuous mapping from the closed interval $[0, 1]$ to the plane. $\gamma(0)$ and $\gamma(1)$ are called the endpoints of curve α."*[Sza12]

The set $\gamma(\{0, 1\}) = \{\gamma(0), \gamma(1)\}$ provides a short-hand notation of a curve's endpoints.

Definition 7 (Straight line)**.** *A curve $\alpha = \{\gamma(x) : x \in [0, 1]\}$ is a straight line segment or shorthand a straight line, if it can be defined as $\gamma(x) = (1-x)\cdot\gamma(0)+x\cdot\gamma(1)$. Thus, the function $\gamma(x)$ is a linear interpolation between the endpoints $\gamma(\{0, 1\})$.* [Smi24, ayu24]

In accordance with this definition, straight lines can be defined with vectors between the corresponding endpoints. This allows several of the following assumptions and methods

to be adopted from the Analytical Geometry. In this context, a straight line segment's mapping $\gamma : [0,1] \to \mathbb{R}^2$ can be considered as follows, whereby $x \in [0,1]$:

$$\gamma(x) = (1-x) \cdot \gamma(0) + x \cdot \gamma(1) = \gamma(0) + x \cdot (\gamma(1) - \gamma(0)) = \gamma(0) + x \cdot \overrightarrow{\gamma(0)\gamma(1)}$$

Definition 8 (Drawing)**.** *"A drawing* Γ *of a graph* G *maps each vertex* v *to a distinct point* $\Gamma(v)$ *of* [a plane in $\mathbb{R}^2$] *and each edge* (u,v) *to a* [...] *curve* $\Gamma(u,v)$ *with endpoints* $\Gamma(u)$ *and* $\Gamma(v)$*."*[Pat13] *For the purposes of this thesis, it is assumed, that the vertex-drawing function* $\Gamma : V \to \mathbb{R}^2$ *is always injective.*

In the following, $\Gamma(E) = \{\Gamma(e) : e \in E\}$ contains all curves of a graph's drawing Γ and $\Gamma(V) = \{\Gamma(v) : v \in V\}$ is defined as the set of all positions, to which the vertices are mapped to. According to Definition 8, $\gamma(\{0,1\}) = \{\Gamma(u), \Gamma(v)\}$ applies for each curve $\Gamma((u,v)) = \{\gamma(x) : x \in [0,1]\}$. The inversion $\Gamma^{-1} : \mathbb{R}^2 \to V \cup \{\emptyset\}$ is another helpful function to determine, which vertex is placed on a position. Consequently, $\Gamma^{-1}(p) = \emptyset$ applies, if no vertex is positioned on $p \in \mathbb{R}^2$. Moreover, the vector between two drawn vertices $u \in V$ and $v \in V$ in $\mathbb{R}^2$ is described by the shorthand notation $\overrightarrow{uv} = \Gamma(v) - \Gamma(u)$. The length of these vectors is given by $\|\overrightarrow{uv}\| = \|\Gamma(v) - \Gamma(u)\|$. In consideration of these definitions, $\overrightarrow{e_{uv}} = \overrightarrow{uv} \cdot \|\overrightarrow{uv}\|^{-1}$ describes the unit vector pointing from $\Gamma(u)$ to $\Gamma(v)$.

Definition 9 (Point-set embedding)**.** *A drawing* Γ *of a graph* $G(V,E)$ *is a point-set embedding (PSE) of* G *on* P*, if a finite point-set* $P \subseteq \mathbb{R}^2$ *exists, such that the vertices' mapping can be restricted to* $\Gamma : V \to P$ *and all curves are drawn exclusively as straight lines* [NMR12]*. This also restricts the edge-drawing function to* $\Gamma : P \times P \to 2^{\mathbb{R}^2}$*.*

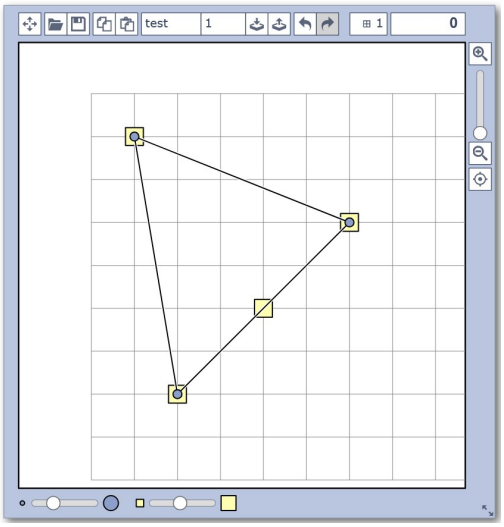


Fig. 2.1.: The yellow squares illustrate the positions derived from the point-set, while the blue circles show the current positions of the vertices. Each edge is illustrated by a straight line between the corresponding vertices. The PSE's score is displayed at the top. If the counter is highlighted in red, the solution is invalid (see Chapter 2.2). [Kin24]

The challenges are limited to drawings in the two-dimensional $\mathbb{R}^2$-plane. In addition, all edges must be drawn as straight line segments and each graph is provided with a finite set of points $P \subseteq \mathbb{N}^2$, to which the vertices are required to be assigned to [BKM+24a]. The points in these sets can be arranged in any pattern, as long as the condition $|P| \geq |V|$ is fulfilled [BKM+24a]. This enables the aspired drawings to be classified as PSEs.

Illustrations will be often utilized to demonstrate and deliberate about PSEs. The visuals displayed were generated using Philipp Kindermann's Graph Drawing Tool, which is integrated into the GDC submission system [Kin24]. A first example, which includes some instructions in the caption, is provided in Figure 2.1.

Definition 10. *$\mathbb{T}_{G,P}$ is going to describe the smallest set, which contains all existing PSE's of the graph $G(V, E)$ within a finite point-set $P \subseteq \mathbb{R}^2$.*

Lemma 11. *Let $G(V, E)$ be a graph and $P \subseteq \mathbb{R}^2$ be a finite arbitrary point-set. Then, the following equation applies for the size of $\mathbb{T}_{G,P}$:*

$$|\mathbb{T}_{G,P}| = \frac{|P|!}{(|P| - |V|)!} = |V|! \cdot \binom{|P|}{|V|}$$

Proof. The linear interpolations of straight line segments are uniquely determined by the endpoints $\gamma(\{0, 1\})$. This means, that each assignment of the vertices in the point-set can only be drawn in a single manner. Consequently, the cardinality of $\mathbb{T}_{G,P}$ corresponds to the number of existing assignments or, in other words, the number of injective mappings $\Gamma : V \to P$. The formula to be proven is a direct consequence of this [Mat24]. □

Finally, the term "edge" will be employed to refer to both a drawn straight line and its corresponding edge, particularly in explanations of drawings. The likelihood of producing dual clarity can be dismissed, because of the distinct manner in which each straight line is drawn. The same applies to the vertices.

2.2. Graph Drawing Contest 2024

According to the previous explanations, the objective of this year's GDC can be formally classified by the minimization of crossings in PSEs. Next, the succeeding paragraphs provide a detailed overview of the contest's standard procedures. The thesis does not describe any non-essential content, however more information can be accessed under this link: https://mozart.diei.unipg.it/gdcontest/2024/.

In general, there are two categories of participants, who take part in the competition. On one hand, participants can choose to minimize crossings of PSEs exclusively by hand. The tasks for this group include fairly small graphs. On the other hand, competitors can solve the tasks semi-automatically with the help of their own toolboxes. Since these resources usually contain scalable algorithms, these participants are additionally challenged with large and complex PSEs. Yet, they also have to solve the smaller tasks and

are especially penalized for bad solutions of these. Two different categories of drawings can be derived from these manners of participation. The term "Automatics" will refer to those drawings, which are assigned to the second category. Analogously, "Manuals" will describe drawings, which are smaller and suitable for manual editing. As soon as the competition began, the tasks were made available to the participants. Historically, a total of about 7 drawings are assigned to each of the categories. Immediately after the distribution of tasks, a 60-minute time window started, in which the participants had to optimize the PSEs. In practice, computational calculations were given approximately 50 minutes to provide satisfactory solutions, leaving enough time to submit all results using the official submission system [Kut24]. [BKM+24a]

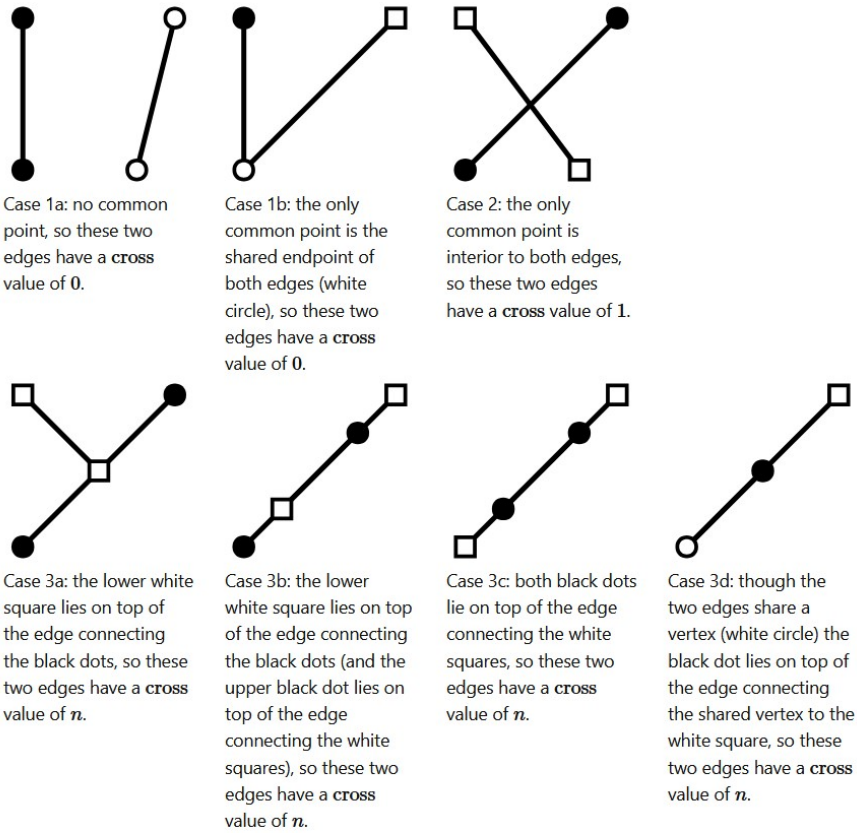


Fig. 2.2.: Visualized explanation of the Cross(e_1, e_2) function [BKM+24a].

All tasks are supplied and submitted in a pre-defined JSON format. The files are composed of the sets of points and vertices. Each point and vertex is identified by an unique ID and associated coordinates in the two-dimensional $(\mathbb{N} \times \mathbb{N})$-plane, which is additionally limited by a specified height and width. Furthermore, the edges are represented as unordered pairs of vertex IDs. The endpoints required for the drawing of edges can accordingly be taken from the referenced vertices. Finally, the only required and permitted modifications of the submissions are those to the vertices' coordinates.

A solution is considered as valid, if it complies to the predefined format requirements and if each vertex is located on a distinct position within the provided set of points. If the graph is not valid, the PSE is rated with an infinite score. Otherwise, the graph will be evaluated using the following formula, for which $\Gamma \in \mathbb{T}_{G,P}$ is the submitted PSE of the graph $G(V, E)$ and for any $i, j \in [1, |E|]$ it holds that $e_i, e_j \in E$:

$$\text{Score}(\Gamma) = \sum_{i=1}^{|E|} \sum_{j=i+1}^{|E|} \text{Cross}(e_i, e_j)$$

Let e_1 and e_2 be two different drawn edges. Then, the pair will be evaluated according to the following: If the drawn straight line segments do not cross or if they only share one endpoint, then $\text{Cross}(e_1, e_2) = 0$. If they cross, so that their cut includes more then one endpoint, then $\text{Cross}(e_1, e_2) = |V|$. If all of these statements are false, meaning that the edges have a distinct intersection point, which is not one of the edges' endpoints, then $\text{Cross}(e_1, e_2) = 1$. Figure 2.2 illustrates this description even better. [BKM+24a]

2.3. A combined approach

The article "Experimental studies on graph drawing algorithms", authored by Luca Vismara et al., offers a detailed examination of frequently employed graph drawing strategies. Thereby, his paper systematically categorized multiple approaches according to their characteristics and use cases [VBG+00]. Considering the GDC's framework conditions, two approaches remain of interest in particular: FDA and randomized strategies. The first methodology summarizes algorithms, that simulate forces between vertices and let these interact iteratively until a stable drawing is created (see Chapter 3.3) [VBG+00]. Next, randomized techniques remain. Typically, these algorithms are implemented using the SA technique (see Chapter 3.5) [VBG+00]. Other mentioned, but in this thesis unconsidered, algorithms cannot be applied to arbitrary arranged point-sets and require, for instance, grid-like or layered arrangements [VBG+00].

Kutscheid successfully integrated the outstanding methodologies described by Vismara in his previous bachelor's thesis and merged them in a combined approach [Kut24]. Moreover, he conducted a comparative assessment of several sub-strategies and simulated the circumstances of the GDC 2023 in his experiments [Kut24], which set the same challenge as this year's contest [BKK+23a, BKM+24a]. Due to this, many assumptions and decisions made can be justified referencing his work. In summary, his combined approach consists of the following sequential steps:

- First of all, the crossings of edges are to be minimized without consideration of a point-set. One way to solve this is the application of FDAs. Kutscheid specifically compared the *Spring Embedder by Eades* with the *Tutte Embedding*.
- Next, it is necessary to assign the vertices to appropriate points. In the prior thesis, two variants of a *greedy assignment* were compared to the *Hungarian Algorithm*.

- To take advantage of the remaining time, his approach proceeds with SA to further improve the drawing. Because of time limitations, Kutscheid was only able to implement the *Random Walk* as modification technique.

Considering the convincing success of this approach [Kut24], his research will serve as an initial basis. Thus, this thesis integrates the most effective algorithms, namely the Spring Embedder by Eades and SA with Random Walk, in order to carry out a comparative examination of other related sub-strategies. Greedy assignments are used to convert the results of FDA's into valid PSE's, because Kutscheid's assessments show, that the increased effort required for Hungarian embedding cannot be justified by superior outcomes [Kut24]. Nevertheless, the combined approach produces disappointing outcomes for embeddings within point-sets, which are arranged in a linear manner [Kut24]. A further objective aims to develop appropriate improvements for this edge-case. In addition, Kutscheid suggests, that employing *Fruchterman and Reingold's* (FR) further development of Eades' FDA could generally yield favourable outcomes [Kut24].

3. Algorithmic approaches

This chapter is going to explain multiple approaches and sub-strategies, which are capable of solving the contest's challenges within the combined approach. All of them will be included in the final toolkit and examined in the chapter on testing.

3.1. Score calculation

Some of the following algorithms require a constant reevaluation of the criterion to be minimized: Score(Γ). These are especially those approaches, that can only exploit their potential if they work quickly and explore as many drawings as possible during the specified time. An efficient implementation of the score calculation is therefore equally important than the optimisation algorithms themselves, in order to avoid bottlenecks. For this reason, two consecutive methods for the calculation of scores are presented and discussed in the following sections.

3.1.1. Exhaustive counting

Algorithm 1: Score(Γ)

Input: Drawing Γ of a graph $G(V, E)$
Output: Score of Γ

```
1  score ← 0
2  foreach {e1, e2} in (Γ(E) 2) do
3      if e1 and e2 have no endpoint in common then
4          if an endpoint of e1 lies on e2 or vice versa then
5              score ← score + |V|
6          else if e1 and e2 cross then
7              score ← score + 1
8      else
9          if the unshared endpoint of e1 lies on e2 or vice versa then
10             score ← score + |V|
11 return score
```

An intuitive method for counting crossings consists of the excessive comparison of all edges with each other. The supervising Professor Philipp Kindermann provided a simple algorithm along with the necessary procedures, which adheres this idea and is shown in Algorithm 1. According to the pseudocode, a loop over all two elementary subsets of the edge-set is necessary for this approach, even if in practice this is more likely to be solved by two separate loops, which yield the same effect.

For an explanation of the calculations, consider $e_1 = \Gamma((v_1, v_2)) = \{\gamma_1(x) : x \in [0,1]\}$ and $e_2 = \Gamma((v_3, v_4)) = \{\gamma_2(x) : x \in [0,1]\}$. As the loop is constructed via the two-element subsets $\binom{\Gamma(E)}{2}$, the case of two edges being equal can be ruled out. Consequently, it is possible to immediately check whether the different edges have endpoints in common. A straightforward comparison of the sets $\gamma_1(\{0,1\})$ and $\gamma_2(\{0,1\})$ suffices for this purpose. In both cases, it must further be determined whether one of e_1's endpoints lies on the other edge e_2 or vice versa, as can be seen in line 4 respectively 9. This can be determined, for example for a $p \in \gamma_1(\{0,1\})$, with vector mathematics: "If the x-coordinates of $[e_2]$'s endpoints are either both lower or both higher than $[p]$'s x-coordinate, [then p] cannot lie on the edge. The same comparison is made for the y-coordinates."[Kut24] If any of these considerations make a further investigation necessary, the perp dot product, also known as two-dimensional cross product, $\times : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ is utilized [Mol24]. Two different vectors $\vec{a}$ and $\vec{b}$ course parallel to each other, if $\vec{a} \times \vec{b} = 0$ holds [ekt24]. If this is the case for the vectors $\gamma_2(0)$ to p and p to $\gamma_2(1)$, then p must be on the edge e_2 [Kut24]. Consequently, the crossing must be penalised in accordance with case 3a-3c (in line 5) or cases 3d (in line 10) of Figure 2.2.

$$\vec{a} \times \vec{b} = \vec{a}^{\perp} \cdot \vec{b} = a_0 \cdot b_1 - b_0 \cdot a_1$$

Finally, the only remaining case is, that both edges are unequal and none of the endpoints lie on the other edge. For this case, it remains to be checked in line 6 whether the edges cross such as in case 2 of Figure 2.2. Therefore, the algorithm proceeds with the orientation operator orient : $\mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$. If for three pairwise different $a, b, c \in \mathbb{R}^2$ $\text{sign}(\text{orient}(a,b,c)) \neq \text{sign}(\text{orient}(a,b,d))$ applies, it can be concluded, that the positions c and d are not on the same side of the straight line segment between a and b [ING24]. If $\gamma_1(\{0,1\})$ are inserted for a and b as well as $\gamma_2(\{0,1\})$ for c and d, this and the previous considerations would imply, that a simple crossing must be panellised.

$$\text{orient}(a,b,c) = \overrightarrow{ab} \times \overrightarrow{ac} = (b_0 - a_0)(c_1 - a_1) - (b_1 - a_1)(c_0 - a_0)$$

Due to the outer loop, the expectable runtime is within $\mathcal{O}(\binom{|E|}{2}) = \mathcal{O}(\frac{|E| \cdot (|E|-1)}{2})$. Even though all subroutines have a low computational effort, the optimization process will be negatively impacted by a frequent use of this methodology for score calculation. The significance of this impact grows even greater when dealing with large PSEs, such as these in the GDC's Automatic category. It is therefore necessary to further develop and optimize this methodology.

3.1.2. Lazy counting

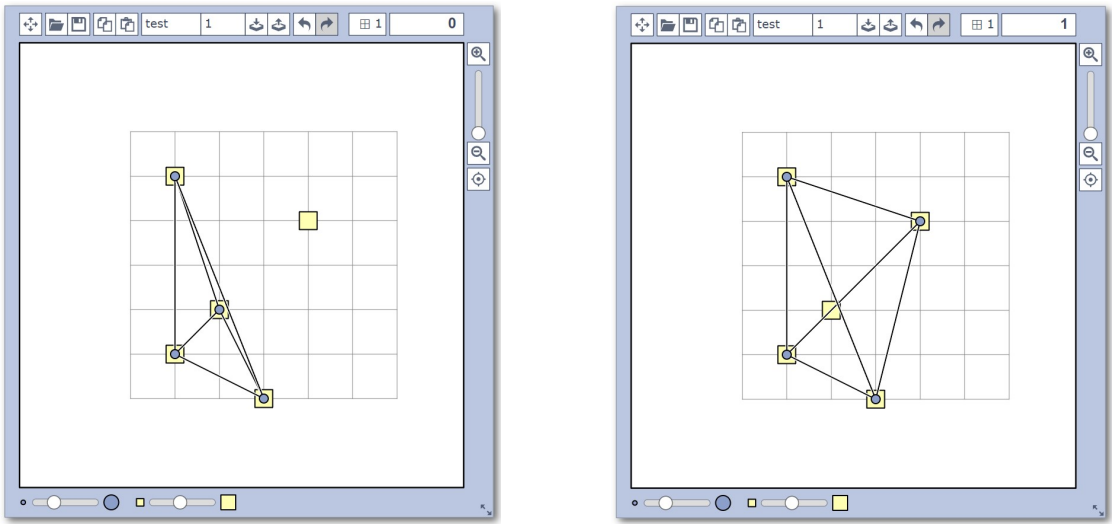


Fig. 3.1.: The drawing on the right was generated from the left by a repositioning of the central vertex. It can be seen, that only adjacent edges were affected by this change.

The algorithms, which require a frequent recalculation of the current score, solely manipulate a small amount of vertices at the same time. This provides the idea, that it is not needed to compare each edge with another. Instead, it is sufficient to determine the impact of a modification by comparing only the adjusted edges to all the others [Kut24]. Specifically, only the edges, that are adjacent to the vertex being repositioned, are affected [Kut24], as illustrated in Figure 3.1. To describe a vertex's impact formally, the penality operator $\text{pen}(v) = \sum_{(v,u)\in E} \sum_{e\in E\setminus\{(v,u)\}} \text{Cross}((v,u),e)$ is introduced at this point. Note, that the operator utilizes $\text{Cross}(e_1, e_2)$, which was already introduced in Chapter 2.2. Thus, it correctly sums up the penalties of all edges adjacent to a $v \in V$ within an expected runtime of $\mathcal{O}(\deg(v) \cdot |E|)$.

To calculate the score lazily, a variable must be initialized with the correct score at first. Consequently, an expensive calculation of the score is required at least once. This variable will then be employed to track the current score. Whenever a vertex $v \in V$ is moved, the tracker gets adjusted with the difference of $\text{pen}(v)$ between before and after the modification. This implies, that a runtime of $\mathcal{O}(2 \cdot (\deg(v) \cdot |E|))$ is needed for each update. Moreover, the penalties are also required for subsequent approaches. Since it is necessary to iterate over all changing edges anyway, it is possible to track these as well and to save them for instance in the vertex data-structure.

Lemma 12. *Let Γ be a drawing of $G(V, E)$ in which $v \in V$ is to be repositioned. The expectable runtime for a lazy update of a tracked score is faster than exhaustive counting, if $\deg(v) < \frac{|E|-1}{4}$ applies.*

Proof. The first inequality was determined according to the previously estimated runtimes for the two score calculation routines. Then, the term can be simplified:

$$2 \cdot \deg(v) \cdot |E| < \frac{|E| \cdot (|E| - 1)}{2} \Rightarrow 2 \cdot \deg(v) < \frac{|E| - 1}{2} \Rightarrow \deg(v) < \frac{|E| - 1}{4}$$

According to the chain of implications, the assertion to be proven is therefore valid. □

Lemma 13. *Let Γ be a drawing of $G(V, E)$ in which $v_1, v_2 \in V$ are to be swapped. The expectable runtime for updating a tracked score lazily is faster than exhaustive counting, if $\deg(v) < \frac{|E|-1}{8}$ applies.*

Proof. When two vertices are swapped, the tracked score must be updated for the adjacent edges of both. The formulated inequality is therefore deduced from Lemma 12. □

The Lemmas 12 and 13 suggest, that the use of tracked scoring should be especially rewarding for large graphs. In these, the vertices are most likely to have a much smaller degree as the edge-set's size. However, if a larger sequence of modifications can be made without requiring a recalculation of Score(Γ), it may be beneficial to employ the former method instead. Nevertheless, a *sweep line algorithm* could further enhance the score calculation and count all crossings much more efficiently [dBea08]. Due to the time constraints of this bachelor thesis, the completion of the debugging was too time-consuming for this methodology. It may be worth pursuing this further for future work.

3.2. Brute-force algorithm

Algorithm 2: BruteForce(G, P)

Input: Graph $G(V, E)$, point-set P
Output: PSE $\Gamma \in \mathbb{T}_{G,P}$ (optimal solution as long as not aborted in line 6)

```
1 Γ_min ← null
2 foreach Γ in T_{G,P} do
3     if Score(Γ_min) > Score(Γ) then
4         Γ_min ← Γ
5     if not time remains then
6         break
7 return Γ_min
```

The naive *brute-force* strategy investigates all possible PSEs of a graph in the given point-set, as demonstrated in Algorithm 2. During each iteration, the score must be recalculated and compared to the preceding minimum. If a better drawing is discovered, the interim result must be overwritten accordingly. In order to guarantee termination within the stated time without any loss of data, it is additionally necessary to introduce

an abortion criterion, which establishes a maximum runtime. In summary, this approach is the most intuitive, but also the least efficient, of all the strategies mentioned in this thesis. By utilizing Lemma 11 and the explanations of Score, this runtime is assessable:

$$\mathcal{O}\left(\frac{|P|!}{(|P|-|V|)!}\cdot\binom{|E|}{2}\right)$$

For this estimation, the multipliers result from the size of $\mathbb{T}_{G,P}$ and the expectable runtime for the slow score recalculation. However, the latter multiplier can be improved for large graphs according to the considerations regarding lazy score calculation. But for both variants it is quite obvious, that the required iterations are not appropriate for large PSEs. Nevertheless, the competition's time-limits may suffice to optimize smaller graphs with this strategy. Otherwise, the necessity of the other approaches will be underlined. Despite all of that, bruteforce remains particularly interesting since it is the only considered strategy, that guarantees optimal outputs. Of course, this assumes that all existing embeddings can be examined without a forced termination.

To avoid the necessity of maintaining a taboo-list of already investigated embeddings, new candidates must be generated in a predetermined sequence. This resulted in a separate issue, which emerged during the actual implementation: It became challenging to create a suitable sequence, that manipulates as few vertices as possible at once. A suitable solution would have a significant impact on the potential benefit resulting from the utilization of the lazy score calculation.

3.3. Force-directed algorithms

Algorithm 3: ForceDirected(Γ)

Input: Drawing Γ of a graph $G(V, E)$
Output: Beautified variant of Γ

```
1 t ← 1
2 while t < T and max_{v∈V} ||δ_t(F(v))|| > ε do
3     foreach v in V do
4         F(v) ← Σ_{u∈V} f_rep(v, u) + Σ_{(v,u)∈E} f_attr(v, u)
5     foreach v in V do
6         Γ(v) ← Γ(v) + δ_t(F(v))
7     t ← t + 1
8 return Normalize(Γ)
```

The core principle of FDAs centres on the interaction between attracting and repulsive forces, as shortly described before. In this paper, the so-called *spring embedders* are considered in particular, whose basic idea is based on an analogy from the real world:

"To embed a graph we replace the vertices by steel rings and replace each edge with a spring to form a mechanical system [...]. The vertices are placed in some initial layout and let go so that the spring forces on the rings move the system to a minimal energy state"[Ead84]. In consequence, adjacent vertices are attracted to each other, which typically leads to the formation of clusters. Non-adjacent ones on the other hand, are pushed away from each other. As Kutscheid has demonstrated, this general behaviour of FDAs contributes significantly to the reduction of crossings in drawings [Kut24].

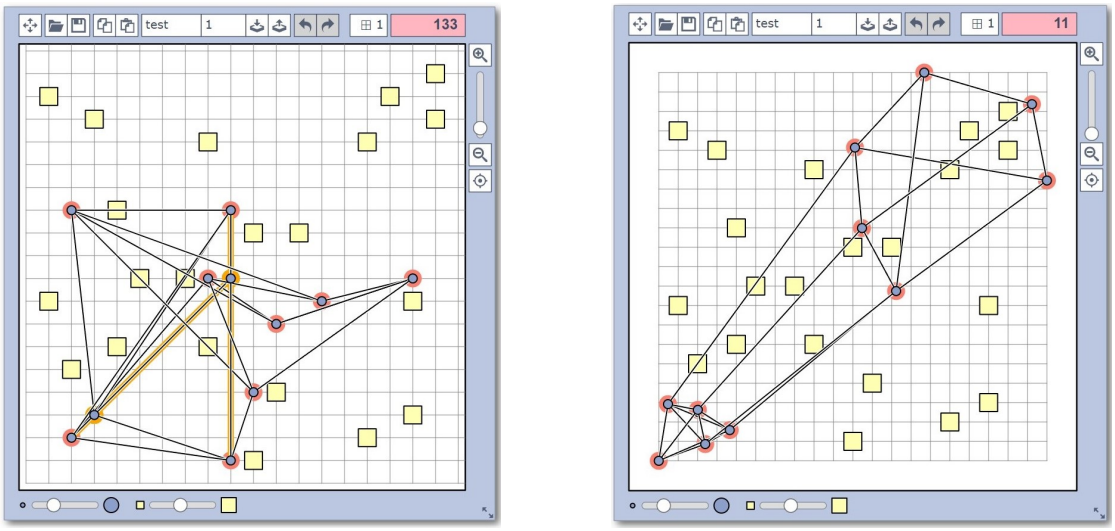
In practice, these forces can be simulated with an attract-function $f_{attr} : V \times V \to \mathbb{R}^2$ and a repel-function $f_{repl} : V \times V \to \mathbb{R}^2$, which are summed up over all vertices and edges in a vector movement [Kin21a]. This process can be seen in line 4 of Algorithm 3 and needs to be repeated for each vertex. Prior to the application of these forces, they are subsequently modified by a cooling function $\delta_t : \mathbb{R}^2 \to \mathbb{R}^2$, whereby $t \in \mathbb{N}$ corresponds to the current iteration. A proper cooling guarantees, that the forces continuously decrease. This causes the drawing to remain relatively stable in late iterations, in which a good solution should already be present [FR91]. Finally, the forces are applied to each vertex in line 6. These steps are repeated, until a maximum number of iterations $T \in \mathbb{N}$ is reached or the calculated forces remain marginal, which implies, that a stable drawing has been produced. To perform the second check, the algorithm inquires whether a vertex was moved further than $\varepsilon = 0.001$ length units in the last iteration.

A comprehensive comparison will be made between the spring embedding suggested by Eades and an alternative approach by Fruchterman and Reingold. Both spring embedders only differ in the definition of the force calculating and cooling functions. Although they are closely related in this regard, they tend to produce very different drawings [Kin21c]. This can also be observed in Figures 3.2b and 3.3b, which both illustrate the embedding results of the same initial drawing. Kutscheid's previous research demonstrated, that the so-called *Tutte Embedding*, which iteratively positions all vertices in a centred position between their neighbours, does not offer any benefits compared to the typical spring embedders [Kut24]. In context of the GDC, it would therefore not be expedient to pursue this approach further.

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

During the execution of both FDAs, the vertices are expected to exceed the width and height boundaries stated by the GDC. For the most drawings, it proved to be suitable to subsequently perform a traditional *min-max normalization* as described in [MP21] and can be seen above. Thus, this formula firstly normalizes the vertices' coordinates to $[0, 1]^2$. The normalized drawing can afterwards be scaled to the required ratios via a multiplication with the specified heights and widths. Occasionally, some of the vertices are strongly repelled from all others, so that the normalisation may result in a large centralised cluster with a few lonesome vertices at the drawing's borders. It might be interesting for further studies to counteract these effects with alternative normalization methods, that relativize strong differences between the distances.

3.3.1. Eades



(a) Initial drawing.

(b) Final drawing.

Fig. 3.2.: Example run of the spring embedder by Eades. The following parameters have been used: $T = 64$, $\varepsilon = 0.0001$, $c_{attr} = 1$, $c_{repl} = 20$, $c = 0.95$ and $l = 1$.

In Eades' implementation of spring embedding, the attractive forces between adjacent vertices have a logarithmic influence on the summed forces [Ead84, Kin21b]. This ensures, that distant but adjacent vertices are strongly attracted to each other, while the acting forces are prevented from becoming excessive. Two parameters assist in fine-tuning: Parameter $c_{attr} \in \mathbb{R}^+$ is used as a multiplier for the attracting forces and $l \in \mathbb{R}^+$ describes an ideal length for the drawn edges. If the vertices are closer than the length parameter, the attracting forces are negative and thus behave pushing [Kin21b].

$$f_{attr}(u, v) = c_{attr} \cdot \log \frac{\|\overrightarrow{uv}\|}{l} \cdot \overrightarrow{e_{uv}}$$

An inverse-quadratic repel function, which utilizes the vertices' euclidean distance as base, is intended to work as a compensatory force [Kin21b]. As a result, the repulsive forces decrease when the vertices are distanced from each other. If they are close, the opposite behaviour will be generated. Parameter c_{repl} is the only introduced tunable variable for this calculation and serves as a multiplier of the repulsive forces.

$$f_{repl}(u, v) = \frac{c_{repl}}{\|\overrightarrow{uv}\|^2} \cdot \overrightarrow{e_{vu}}$$

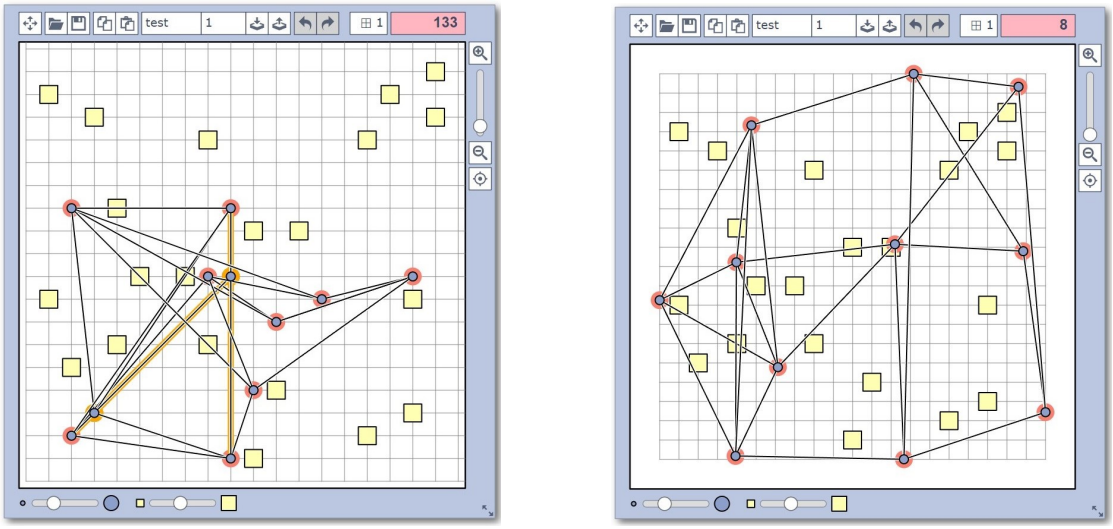
In Eades' original paper, the repulsive forces are computed solely based on vertices, that are not adjacent to each other [Ead84]. This has not been adopted here, so that

the concept of Algorithm 3 can also be applied to the second sub-strategy without any modifications. In order to prevent falsified results, f_{attr} can be expanded by an additional subtraction, which corrects the over-calculated repulsive forces [Kin21b].

$$f_{attr}(u,v) = c_{attr} \cdot \log \frac{\|\overrightarrow{uv}\|}{l} \cdot \overrightarrow{e_{uv}} - f_{repl}(u,v)$$

Finally, the cooling function must be carefully adjusted: If the temperature decreases too quickly, the drawing may freeze without any significant modifications [Kin21b]. On the other hand, if the cooling process is too slow, the algorithm may take too long to reach a stable state [Kin21b]. Parallel to Kutscheids implementation [Kut24], the cooling function is defined as a scalar multiplication: $\delta_t(\vec{v}) = c^{(t)} \cdot \vec{v}$, whereby $c \in [0,1)$ is an additional tunable parameter and $t \in \mathbb{N}$ is the current iteration. In order to avoid early freezing, a value closer to the upper interval boundary should be selected.

3.3.2. Fruchterman and Reingold



(a) Initial drawing.

(b) Final drawing.

Fig. 3.3.: Example run of the spring embedder by Fruchterman and Reingold. The following parameters have been used: $T = 64$, $\varepsilon = 0.0001$, $c_{attr} = 1$, $c_{repl} = 20$, $l = 50$.

A significant disadvantage of Eades' spring embedder is its tendency to converge towards local minima [Kin21b]. FR's further development of the embedder is designed to counter this behaviour. Through the generation of much stronger attractions, the vertices are prevented from being trapped in non-optimal locations by the acting repulsive forces [FR91]. This should enable local minima to be overcome and left behind. Another noticeable difference of FRs algorithm is, that the utilized functions are much simpler [FR91] and use a single tunable parameter: the optimal length $l \in \mathbb{R}^+$. Apart from that,

the attractive forces behave equal to the former variant: “Vertices connected by an edge should be drawn near each other [, but] not be drawn too close to each other.”[FR91]

$$f_{attr}(u,v) = \frac{\|\overrightarrow{uv}\|^2}{l} \cdot \overrightarrow{e_{uv}}$$

In contrast to the stronger attractive forces, the curve behaviour of the repulsive function, and thus also it's impact, is comparable to the variant defined by Eades [Kin21c]. Different is, that repulsive forces are also simulated for adjacent vertices [FR91].

$$f_{repl}(u,v) = \frac{l^2}{\|\overrightarrow{uv}\|} \cdot \overrightarrow{e_{vu}}$$

Due to the extreme growth of the acting forces, the practical realization of this FDA quickly encountered the risk of number overflows in early iterations. The cooling function therefore plays another important role in this sub-strategy: It has to ensure, that the vertices cannot move too far away from each other. Based on the experiences gained during the tests, this also resulted in a more even distribution of the vertices, which also improved the subsequently min-max normalized results. In this implementation, a tunable cooling factor $c \in [0,1)$ is used to determine a maximal movement $m_t = 2 \cdot l \cdot c^{(t)}$, which is allowed to act on each vertex in the t-th iteration. If the summed forces yield larger adjustments, the calculated vector is simply trimmed to the desired length.

$$\delta_t(\vec{v}) = \begin{cases} m_t \cdot e_{\vec{v}}, & m_t < \|\vec{v}\| \\ \vec{v}, & \text{otherwise} \end{cases}$$

3.4. Greedy assignment

As the figures 3.2 and 3.3 exemplify, FDAs do not produce valid PSEs. Instead, the algorithms position vertices freely in the plane and do not consider a set of allowed points. Consequently, the interim results must be converted into valid embeddings. To ensure, that the previous calculations were not in vain, the assignment must produce results, which are as similar as possible to the previous drawing. Regarding this matter, Kutscheid has already been able to demonstrate, that a simple and distance-based greedy approach produces satisfactory outcomes [Kut24]. For this reason, two different implementations will be adapted from the prior thesis.

The first variant is displayed in Algorithm 4 and essentially consists of two nested loops. The inner loop determines for each vertex, which point is the closest according to the euclidean distance and has not yet been occupied. The latter check is intended to prevent double occupancies and thus ensures, that the assignment's result is a valid PSE. Once a target is determined, the vertex will be immediately repositioned and the outer loop continues with the next vertex until all are assigned.

Algorithm 4: FastAssign(Γ, P)

Input: Drawing Γ of a Graph $G(V, E)$, point-set P
Output: PSE $\Gamma \in \mathbb{T}_{G,P}$, which strongly resembles Γ

```
1  foreach v in V do
2      d_min ← ∞
3      p_min ← null
4      foreach p in P do
5          if Γ(p)^{-1} = ∅ then
6              d ← ‖Γ(v) − p‖
7              if d < d_min then
8                  d_min ← d
9                  p_min ← p
10     Γ(v) ← p_min
11 return Γ
```

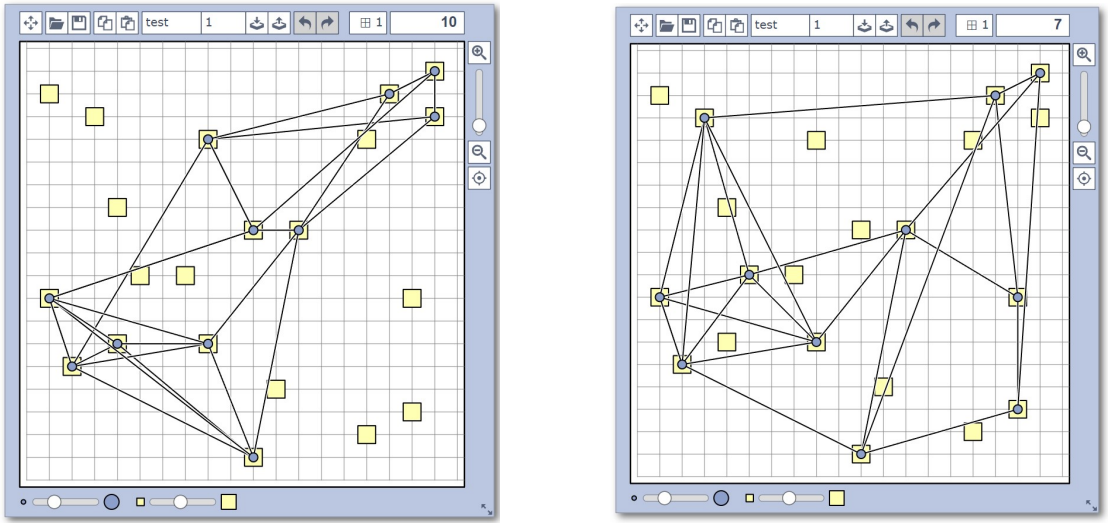
During the practical realisation, this approach was further investigated to determine on whether the vertices' order has a significant influence. Thus, additional calculations were added to arrange the vertices in descending order based on their degree. This was supposed to prioritise highly connected vertices. However, during the first tests it quickly became apparent, that this modification had no influence on the score and was consequently not investigated further (see Appendix A.1).

Algorithm 5: SlowAssign(Γ, P)

Input: Drawing Γ of a graph $G(V, E)$, point-set P
Output: PSE $\Gamma \in \mathbb{T}_{G,P}$, which strongly resembles Γ

```
1  while ∃v ∈ V : Γ(v) ∉ P do
2      d_min ← ∞
3      p_min ← (null, null)
4      foreach v in V do
5          if Γ(v) ∉ P then
6              foreach p in P do
7                  if Γ(p)^{-1} = ∅ then
8                      d ← ‖Γ(v) − p‖
9                      if d < d_min then
10                         d_min ← d
11                         p_min ← (v, p)
12     Γ(p_min1) ← p_min2
13 return Γ
```

An issue arising from the immediate assignment is, that a vertex might be moved to a point even though other vertices are closer. To prevent this, the loops can be modified to determine a pair of an unassigned vertex and an unoccupied point, whose distance is the current global minimum [Kut24], as can be seen in Algorithm 5. But then, the assignment must be outsourced into another outer loop over all vertices, which assembles these optimal pairs until no vertex remains unassigned. Ultimately, a trade-off between better accuracy and a longer runtime must be made. Since each of the two variants has it's strengths and interesting exceptions occurred during the testing, this thesis proceeds with both. Accordingly, a conceptual distinction is required. Therefore, the second approach will be referred to as the slow greedy algorithm. Any reference to a fast assignment, will refer to the variant from Algorithm 4.



(a) Embedded version of drawing 3.2b.

(b) Embedded version of drawing 3.3b.

Fig. 3.4.: Example runs of the fast greedy assignment. The left figure was previously drawn by Eades' spring embedder and the right one by FR's approach.

3.5. Simulated Annealing

During the description of the brute-force approach, it has already been emphasised, that it is insufficient to generate and examine all embeddings sequentially. Instead, an approach must be adopted, which improves the drawing incrementally, while already made process is preserved. In other words, only embeddings, which slightly differ from a recently obtained solution, should be examined further. During this process, it is important, that the algorithm does not get trapped in local minima [Kut24]. It may require several iterations, which yield poorer results, to finally obtain a drawing with fewer crossings. A commonly used and interdisciplinary methodology for global optimisation problems, which addresses exactly this problem, is Simulated Annealing [HJJ03].

Algorithm 6: SimulatedAnnealing(Γ)

Input: PSE $\Gamma \in \mathbb{T}_{G,P}$ with $G = (V, E)$
Output: Variant of Γ with minimized crossings.

```
1  Γ_min ← Γ
2  while time remains do
3      Γ ← Γ_min
4      while less than s seconds have passed do
5          Γ' ← Refactor(Γ)
6          if Score(Γ') < Score(Γ_min) then
7              Γ_min ← Γ'
8          if random x ∈ [0,1) < A_{δ_t(θ)}(Score(Γ') − Score(Γ)) then
9              Γ ← Γ'
10 return Γ_min
```

The naming of SA also originates from a real world analogy: "[...] because of its analogy to the process of physical annealing with solids, in which a crystalline solid is heated and then allowed to cool very slowly until it achieves its most regular possible crystal lattice configuration (i.e., its minimum lattice energy state), and thus is free of crystal defects"[HJJ03]. If a PSE Γ is optimized according to this methodology, the energy state corresponds to Score(Γ). During the process, this PSE is randomly modified to generate a slightly deviating drawing Γ'. If Score(Γ) > Score(Γ') applies, the modification is an improvement and will be accepted deterministically. This means, that the next iteration proceeds with the newly created PSE Γ'. If the modification is a deterioration, meaning that Score(Γ) $\leq$ Score(Γ') applies, it can still be accepted with a certain probability. A high temperature should therefore make acceptance more likely. Over time, the temperature drops, which results in a tendency to reject deteriorations. This behaviour needs to be mathematised with a so-called *acceptance criterion* – one for positive inputs asymptotically and monotonous falling function $A_{temp} : \mathbb{R} \to [0, 1]$ [Sch89], which also satisfies $x \leq 0 \Rightarrow A_{temp}(x) = 1$. The smaller a temperature $temp \in \mathbb{R}$ is, the faster should A_{temp} converge towards zero. Parallel to Kutscheid's work, the acceptance criterion introduced by Metropolis will be used for this purpose [Kut24, Sch89, MRR$^+$53]:

$$A_{temp}(x) = \begin{cases} exp(\frac{-x}{temp}), & x \geq 0 \\ 1, & x < 0 \end{cases}$$

The employed Algorithm 6 utilizes the criterion within lines 8-9. The score's difference after and before a modification, more formally Score(Γ') − Score(Γ), is used as input. A monotonically decreasing temperature will be discussed later, which adjusts the parameter $temp \in \mathbb{R}$ and causes the criterion to determine a probability according to the desired conditions. Beforehand, the drawing is modified as indicated by Refactor(Γ) in line 4.

This placeholder will be replaced by two different refactoring methods: *Random Walk* (see 3.5.1) and *Rebuild Neighbourhood* (see 3.5.2). In addition, a hybrid approach was temporarily considered, which combines both of these techniques and yielded promising results in a short series of tests. However, this approach was not tested extensively prior to the GDC 2024. Because of this, it is only outlined in the Appendix A.2.

Another special feature of Kutscheid's and also of this project's implementations can be found in lines 3-4 [Kut24]. Immediately after modification, the PSE is saved, if it is the best explored so far. This intermediate result will be used to overwrite the currently examined drawing, when the inner loop aborts after s seconds. At the same time, the iteration counter $t \in \mathbb{R}$ is reset, which sensitises the acceptance criterion again. Consequently, the complete annealing process will be reset and restarted based on the best solution found so far. As a result, the initial temperature $\theta \in \mathbb{R}^+$ does not have to be set extraordinarily high in order to resolve local minima even in late phases. During the previous study, the inner loop's execution time was always specified with one minute [Kut24]. A critical situation may arise, in which a high temperature accepts a very poor modification, so that this deterioration cannot be compensated for a long time. Consequently, almost a minute can be consumed without the generation of a new minimal solution. Fine-tuning of the reset frequency s could therefore be worthwhile.

Finally, a monotonously sinking temperature curve $\delta_t : \mathbb{R}^+ \to \mathbb{R}^+$ needs to be determined. Two different approaches are widespread for this purpose: *exponential cooling* with $\delta_t(\theta) = \theta \cdot c_{exp}^t$ and *linear cooling* with $\delta_t(\theta) = \theta - t \cdot c_{lin}$ [NA98], whereby $t \in \mathbb{N}$ is the current iteration of the inner loop and $c_{exp} \in [0, 1)$ respectively $c_{lin} \in [0, \infty)$ are tunable constants. In addition, both functions utilize the initial temperature $\theta \in \mathbb{R}^+$ as input. In order to illustrate the interaction between cooling functions and acceptance criterion, a short example is given here: Suppose the parameters $c_{lin} = 10$, $c_{exp} = 0.85$ and $\theta = 2000$. Let the drawing have worsened by 15 crossings in the tenth iteration. For $t = 10$, the linearly calculated temperature is $\delta_{10}(\theta) = 1000$, so that the deterioration is accepted with a probability of $A_{\delta_{10}(\theta)} \approx 0.985$. For the exponential temperature, $\delta_{10}(\theta) \approx 393.748$ results in an acceptance probability of $A_{\delta_{10}(\theta)}(15) \approx 0.963$.

3.5.1. Random Walk

The first discussed modification technique is the Random Walk, as it is shown in Algorithm 7. It's underlying logic is very simple: A vertex is randomly selected and then moved to a random point [Kut24]. Consequently, an if-else-clause between the lines 3-8 has to handle two different cases, in order to preserve the PSE's validity. If the designated point is already occupied, then both involved vertices must be swapped. Otherwise, the vertex is simply repositioned. In the next refactoring technique, this procedure will be referred to as the operator MoveOrSwap(v, p), whereby $v \in V$ is to be moved to $p \in P$. The greatest advantage of this refactoring technique is its simplicity, which enables to examine as many modifications as possible within a short period of time. Ultimately, the only bottleneck is the frequently needed recalculation of the current score.

Algorithm 7: RandomWalk(Γ)

Input: PSE $\Gamma \in \mathbb{T}_{G,P}$ with $G = (V, E)$
Output: Refactored variant of Γ.

```
1 p ← random p ∈ P
2 v ← random v ∈ V
3 if Γ(p)^{-1} = ∅ then
4 |  Γ(v) ← p
5 else
6 |  v' ← Γ(p)^{-1}
7 |  Γ(v') ← Γ(v)
8 |  Γ(v) ← p
9 return Γ
```

Nevertheless, a completely random selection of vertices makes the generation of worsening modifications probable. To counteract this, a significant difference to Kutscheid's implementation will be tested for line 2 [Kut24]: Dynamically determined probabilities for the randomized selection of vertices. For this purpose, additional runtime is invested to increase the probability of selecting vertices with a large influence on the score. It can be assumed, that especially these are currently poor positioned. Therefore, the *probability function* $d_p : V \to [0, 1]$, which is shown below, will be employed with the parameters $p \in \{0, 1, 2\}$. An application of this function causes, that the penalties determined by the penality operator $\text{pen} : V \to \mathbb{N}$ are considered in a linear manner, when $p = 1$ is set. A selection of $p = 2$ additionally amplifies the desired behaviour by squaring. Parametrization $p = 0$ results in $d_0(v) = (\sum_{u \in V} 1)^{-1} = |V|^{-1}$, and thus generates uniform probabilities, as were used by Kutscheid [Kut24]. In order to prevent zero divisions, the function is applicable as long as one $v \in V$ fulfils $\text{pen}(v) \neq 0$. This is the case, if at least one crossing exists and therefore $\text{Score}(\Gamma) > 0$ holds.

$$d_p(v) = \frac{\text{pen}(v)^p}{\sum_{u \in V} \text{pen}(u)^p}$$

Lemma 14. *Let $\Gamma \in \mathbb{T}_{G,P}$ with $\text{Score}(\Gamma) > 0$ be the drawing, on which d_p operates. Then is $d_p : V \to [0, 1]$, as it is defined above, a valid probability function for all $p \in \mathbb{N}$.*

Proof. Based on [Bar24], it is sufficient to show $\sum_{v \in V} d_p(v) = 1$. $\text{Score}(\Gamma) > 0$ implies, that at least one crossing and hence at least two $v \in V$ with $\text{pen}(v) \neq 0$ exist. In consequence must $\sum_{u \in V} \text{pen}(u)^p \neq 0$ also hold, which means, that zero divisions can be ruled out. This enables the following generally valid simplification:

$$\sum_{v \in V} d_p(v) = \sum_{v \in V} \frac{\text{pen}(v)^p}{\sum_{u \in V} \text{pen}(u)^p} = \frac{\sum_{v \in V} \text{pen}(v)^p}{\sum_{u \in V} \text{pen}(u)^p} = 1$$

These equations can therefore show, that d_p is a valid probability function. □

That the case $\text{Score}(\Gamma) = 0$ is not supported by the proposed functions is not a problem, because no better solution can be found at this point. Consequently, the algorithm can terminate and return an optimal solution. Moreover, the time costs for these additional calculations are relatively low, since the frequent lazy score reevaluation is capable of keeping all penalties tracked. Thus, the final realisation of this functionality was achieved within an additional runtime of $\mathcal{O}(2 \cdot |V|)$ for each iteration, which is acceptable, especially in relation to the already expensive score calculation.

3.5.2. Rebuild Neighbourhood

Algorithm 8: RebuildNeighbourhood(Γ)

Input: PSE $\Gamma \in \mathbb{T}_{G,P}$ with $G = (V, E)$
Output: Refactored variant of Γ.

```
1  v ← random v ∈ V
2  move ← N(v) ∪ {v}
3  near ← {|N(v)|-nearest points to Γ(v)} ∪ {Γ(v)}
4  Shuffle(near)
5  for i = 0 to |near| do
6  |   v ← move[i]
7  |   p ← near[i]
8  |   if random x ∈ [0, 1] < κ then
9  |   |   p ← random p ∈ P
10 |   MoveOrSwap(v, p)
11 return Γ
```

Rebuild Neighbourhood addresses the weakness of Random Walk in another way. Firstly, this sub-strategy modifies entire neighbourhoods, instead of individual vertices. It can accordingly be classified as a *Ruin and Recreate* approach. In addition, it introduces a heuristic, whose idea has already been mentioned for the FDA: It assumes, that neighbouring vertices should be placed close to one another [FR91].

At first, Algorithm 8 randomly selects a vertex $v \in V$ and allocates the whole neighbourhood $N(v)$. Since these pieces of information are required repeatedly, the neighbourhoods should be determined beforehand and kept in *adjacency lists*. As part of the tests, all previously defined probability functions are also taken into account for the selection in line 1. The point $\Gamma(v)$ will be utilized as the centre, on which and the $|N(v)|$-nearest points the neighbourhood will be rearranged. It is also practicable to prepare ordered lists for this purpose, because at most $\max_{v \in V}(\deg(v))$-nearest points are required. As a result, both allocations in lines 2-3 can be realised within a constant runtime. If the latter set is stored in an array-like structure, it can be shuffled to get a random permutation of the nearby points. In the subsequent loop, the vertices are repositioned according to their indices in the reordered structure. Furthermore, $\kappa \in [0, 1]$

is introduced as a tunable parameter and determines the probability with which distant points are also permitted for the reconstruction in lines 8-9. To be more precise, each vertex is assigned to a random point with this probability, instead of the one specified by the permuted data-structure. This parameter therefore helps to recognise the extent to which the heuristic generates modifications too selectively, that could possibly lead to rapid convergence to poor local minima.

This methodology is introduced to consider the edge-case of linearly arranged point-sets in a specialised manner. Long drawn edges are particularly expensive in these instances, because many vertices could potentially be located on the straight line segment between the corresponding endpoints. The crossings created in this manner must always comply to cases 3c, 3b or 3d from Figure 2.2 and will therefore be severely penalised with $|E|$ crossings according to the GDC rules [BKK+23a, BKM+24a]. Figure 3.5 illustrates this situation with a simple example. According to these thoughts, closely positioned neighbours, and thus a small value for the parameter $\kappa \in [0, 1]$, should be beneficial for the optimization of PSEs with linearly arranged point-sets. The upcoming tests will demonstrate, that the considered neighbourhood heuristic can also accelerate the generation of desirable results for other arrangements.

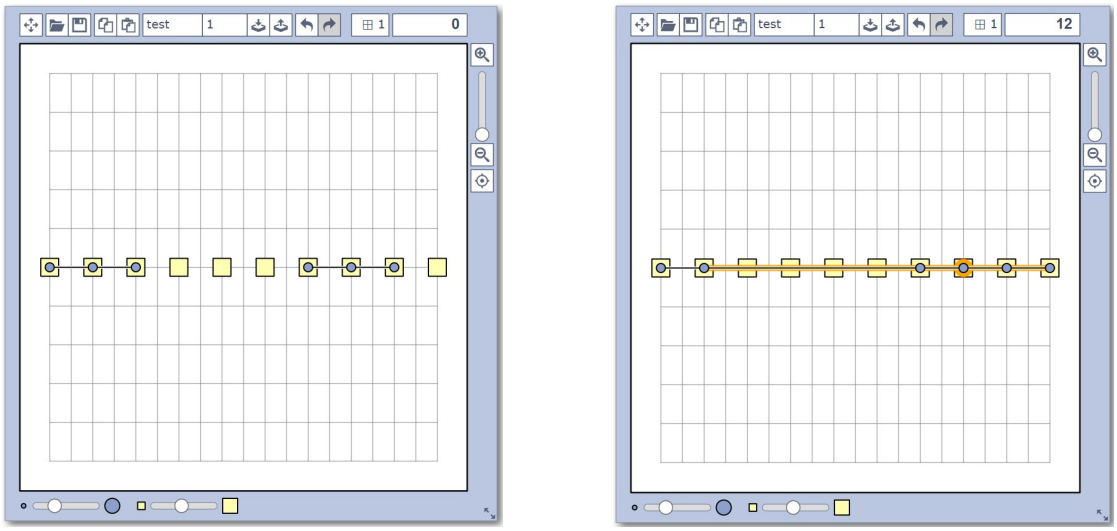


Fig. 3.5.: The drawing on the right side was created from the left side by a repositioning of a single vertex. The score has deteriorated significantly as a result of this modification.

The disadvantage of this approach is, that it is much slower than the Random Walk and can therefore explore fewer modifications – especially when the score is tracked. An additional runtime of $\mathcal{O}((\deg(v) + 1) \cdot 2 \cdot (\deg(v) \cdot |E|))$ is only required for the lazy score calculation, since $\deg(v) + 1$ vertices are moved and the tracker must be updated equally often. However, this should still be faster for the Automatics than if the slower score calculation would be applied. Ultimately, the crucial factor will be whether the heuristic's effect can outweigh the exploration of fewer PSEs.

4. Application

As mentioned earlier, this thesis is complemented by the implementation of a C++ header-only library. The language was chosen in particular to obtain the best possible performance, ensuring that the SA can explore as many drawings as possible. At the same time, care was taken to ensure, that the codes remain maintainable and easily expandable, so that they can potentially be further developed by subsequent work. Consequently, a strict object-orientation was adhered to, although some performance compromises had to be made. This chapter provides a brief overview of the code architecture and the considerations, that were made during the implementation. In addition, a user manual explains how the console API works.

At this point, I want to acknowledge the utilized open source projects: Firstly, the console interface was realised with the cxxopts framework. Secondly, the JSON-library by Nils Lohmann simplified the parsing and handling of JSON files. In addition, the framework doctest provided all necessary tools for testing. Within the project folders, the corresponding licenses can be found enclosed with the respective source codes.

4.1. Code Architecture

"We want our systems to be composed of many small classes, not a few large ones. Each small class encapsulates a single responsibility, [...] and collaborates with a few others to achieve the desired system behaviours."[Mar09] In accordance with this principle, a strict separation between the management of drawings and the execution of algorithms is made. This separation is also reflected in the directory structure, since the realised algorithms are managed in a separate folder. Moreover, two central classes have emerged, that handle these tasks: Strategy and PSE. A description of the resulting architecture is explained below and can be traced in parallel with the UML diagram 4.1.

Each instance of Drawing is responsible for the management of a single graph's vertices and edges. Because all positions are stored in the contained data-structures, it indirectly manages the vertex-drawing mapping $\Gamma : V \to \mathbb{R}^2$. Consequently, the chosen naming is more suitable as if it would be named as graph only. In accordance with the different definitions of drawings and PSEs in Chapter 2.1, the point-set's logic is also introduced separately. Therefore, the PSE class acts as a, through an aggregation created, *decorator* [Gur24], which complements all functions associated with points and the evaluation logics. Ultimately, the decorator offers such a comprehensive interface, that none of the optimisation approaches need to modify data-structures directly.

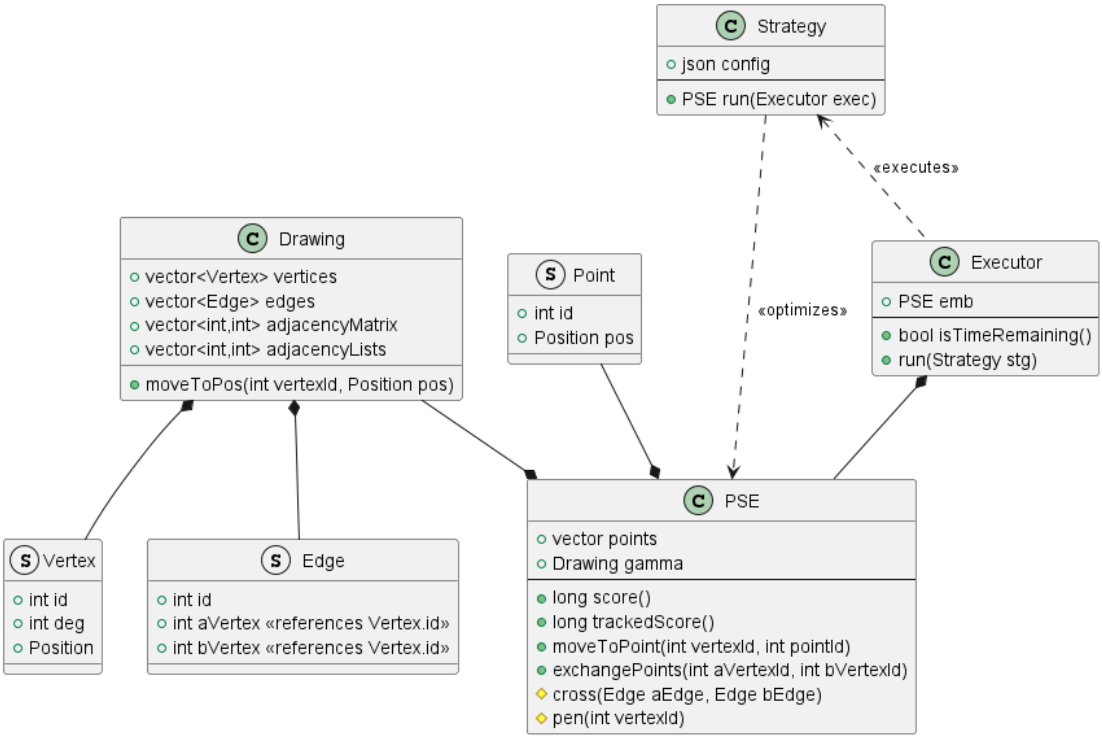


Fig. 4.1.: UML class diagram of the final application's structure. Note, that it is incomplete and only covers basic functionalities and relationships. The approaches discussed in the previous chapters are realised as subclasses of Strategy. The main control flow employs the executor to apply several strategies sequentially.

The defined task, namely the exploration of different methodologies addressing the same problem, implies the use of the *strategy* pattern in natural fashion [Gur24]. For this reason, the term "strategy" is deliberately equated with "approach" throughout the whole thesis. In the application, this pattern is realised with the Executor class. It's run method accepts subclasses of Strategy, which contain the algorithms described in the previous chapter. On execution, the executor itself and therefore also the PSE to be optimised are passed via a *dependency injection* to the strategy's run method [Mar09]. If several optimisation methods are to be executed in sequence, the strategy can simply be replaced. Since the executor is maintained throughout the run of several strategies, it is ideally suited for the management of time information and interim results.

The main program performs the correct initialisation and multithreaded execution of strategies based on a string map. Due to the pairing of this solution with the strategy pattern, new approaches can be added in just three steps: Implementing a subclass of Strategy, depositing the dependencies and finally adding them to the map. Furthermore, lambda expressions can be applied to avoid unnecessary code duplicates. For instance, the sub-strategies for SA and FDA are not defined with separate sub-classes. Instead, global lambda functions are introduced, which are assigned during the construction.

Almost all public functions of the PSE class work with IDs in order to avoid unclear referencing effects and errors. Internally within the classes, vertices and points are therefore organised in vectors according to their IDs, which are introduced by the JSON format. In parallel, an ID-sorted vector is utilized for the edges, whereas the access usually takes place via an intermediate adjacency matrix or instances of vertices. As a result, each data-structure can be retrieved in a constant runtime. Due to this design decision, getters are frequently used, especially in loops. To ensure, that the compiler resolves these as efficiently as possible, compilation flags, which enable aggressive transformations of loops, should be set urgently [Mis23]. Thus, setting the flags to `-O3` or even `-Ofast` led to enormous performance improvements during the tests (see Figure 5.1).

4.2. Operating instructions

Parameter	Explanation
`-m`	Flag to indicate whether multiple files are to be processed. Defaults to `false`.
`-i`	Relative path to the input directory or file.
`-o`	Relative path to the output directory or file.
`-c`	Relative path to the configuration directory. Defaults to `./config/`.
`-t`	Number of minutes after which termination is forced. Defaults to `50`.
`-s`	Specification of the strategies to be applied. If a sequence is to be executed, multiple can be listed "+"-separated.

Tab. 4.1.: Parameters that are required to run the console application.

The published project consists of four different folders. The directory `./source/` contains all relevant classes and the file `./source/dependencies.h`, which includes all source files in bulk. Additionally, the subdirectory `./source/external/` contains all employed open-source libraries. The algorithms themselves are, as already explained, located separately and can be found in `./strategies/`. In order to consider various parametrizations without causing complex console commands, it was decided to save the configurations in JSON files. The standard directory, from which these are imported, is `./config/`, which therefore contains blueprints for further configurations. Finally, the application should be compiled within `./production/`. This ensures, that all default paths can be used without any problems: `cmake -B . -S .. && make`.

To optimize PSEs, the `main` executable must be called within the production folder. Inputs and outputs are organized with additional paths, which must be specified according to the instructions of Table 4.1. If the flag `-m` is set, the program expects a directory path for parameter `-i`, in which each contained JSON file will be read and then optimized in a separate thread. Otherwise, the main routine searches for a single

file, which is then optimized single-threaded. Scanned files must comply with the JSON format of the GDC. Because detailed exception handling has been omitted, the parser would otherwise throw errors. Generated (intermediate) results are stored in the specified output directory in a manner, that allows the solutions to be recognized based on the previously scanned file names. Moreover, the implemented algorithms were executed with multiple parameter configurations during the competition of 2024. To cover this requirement, it is also possible to specify a custom folder with the flag `-c`, which contains differing configuration files. The default settings correspond to those parameter combinations, which are identified as the best in the subsequent chapter or have already been recommended by Kutscheid. The strategies to be applied sequentially can be specified with a "+"-separated string for `-s`. In addition, a notation with square brackets was introduced for the selection of sub-strategies. The examples below should illustrate this even better and provide a template for the most common use cases at the same time.

Executes bruteforce for a single PSE and forces termination after 30 minutes.

```
./main -s bruteforce -i ./input/ -o ./output/ -t 30
```

Executes greedy assignment for a single PSE with non-standard configurations.

```
./main -s greedy -i ./input/ -o ./output/ -c ./config/
```

Executes Eades' spring embedding and greedy embedding in a sequence.
Because the flag `-m` is set, several PSEs will be processed in parallel.

```
./main -s fda[spring]+greedy -i ./input/ -o ./output/ -m
```

Executes the combined approach utilizing SA with random walk and FR.
Because the flag `-t` is not set, termination will be forced after 50 minutes.

```
./main -s fda[fr]+greedy+sa[walk] -i ./input/ -o ./output/ -m
```

Prints a short report about the most important key data of a PSE.

```
./main -s analysis -i ./input/ -o ./output/
```

5. Testing

In this chapter, the different optimization strategies and the complete combined approach are examined extensively. Since Kutscheid has already carried out systematic tests for some of the employed methodologies, his results will be referenced and applied without further testing. This makes it possible to focus on other aspects and to examine certain patterns in more detail. Nevertheless, for those strategies, that have not yet been investigated, a favourable parametrization must be found at first. Based on the most promising parameter combinations, all sub-strategies were ultimately compared with Kutscheids final results (see Chapter 5.4).

ID	$\|V\|$	$\|E\|$	$\|P\|$	size	avg deg(v)	max deg(v)	arrangement
m_1	11	23	21	20x20	4.182	5	arbitrary
m_2	12	28	12	48x48	4.667	7	circle
m_3	24	33	25	12x14	2.75	14	grid
m_4	16	31	16	25x25	3.875	6	arbitrary
m_5	25	46	25	13x13	3.68	9	symmetric
m_6	31	147	31	138x108	9.484	17	arbitrary
m_7	10	25	10	15x15	5	8	grid
a_1	551	8544	651	1000x1000	31.013	47	arbitrary
a_2	1200	3344	1200	144000x144000	5.574	13	circle
a_3	2400	4450	2475	92x122	3.708	4	grid
a_4	2950	8717	2950	1896x1694	5.911	16	grid
a_5	4761	4761	4761	90x90	2	1,974	grid
a_6	1589	2743	1589	5555x5516	3.453	34	arbitrary
a_7	3000	2999	3000	5999x1	1.999	9	linear

Tab. 5.1.: An overview of the PSEs to be optimised from the GDC 2023, which were adopted as datasets for the following series of tests.

Traditionally, most of the PSEs addressed in a GDC are not randomly generated and correspond to a variety of special cases or manually manipulated patterns. A randomized PSE generator would have difficulties to cover them completely and satisfactorily. Since this thesis especially aimed to perform well at the GDC 2024, it was decided to employ the repertoire of tasks from last year's competition. Important key information about these can be retrieved from Table 5.1. Due to the diverse arrangements of the point-sets, it is still possible to make reliable statements about the conditions under which the strategies work particularly well or poorly. Furthermore, the utilization of these drawings ensures, that the last competition's conditions are exactly reproduced. With this in mind, all algorithms are also forced to terminate after 50 minutes.

The testing was performed on an Ubuntu 22.04.4 LTS machine with two Intel Xeon Gold 6342 Processors, each with 24 cores and a 36 MB Cache, operating at a base clock frequency of 2.80 GHz. In addition, a total RAM of 239 GB was available. The same server was used during the participation at this year's GDC.

5.1. Brute-force and score calculation

Firstly, the performance of the brute-force methodology was examined. The analytical runtime estimation already revealed, that this approach is impractical for large drawings. The test runs revealed, that the performance is indeed so poor, that only the smallest Manual m_7 was solved within the defined time constraints. Nevertheless, Appendix B.1 lists the results obtained within the brute-force test series, whereas the numerical entries refer to the scores. However, the results are not very meaningful, because for most PSEs only a fraction of all existing solutions were explored. A discussion of the test results is therefore not worthwhile, even though they can be regarded as negative reference values for the other approaches. Yet, the results of drawings a_5 with 10,322 and a_7 with 15,165 are remarkable, as they compete well with the results of more expedient approaches. Rather than indicating efficiency, this situation is more likely to be explained by a fortunate sequence in which new PSEs are explored.

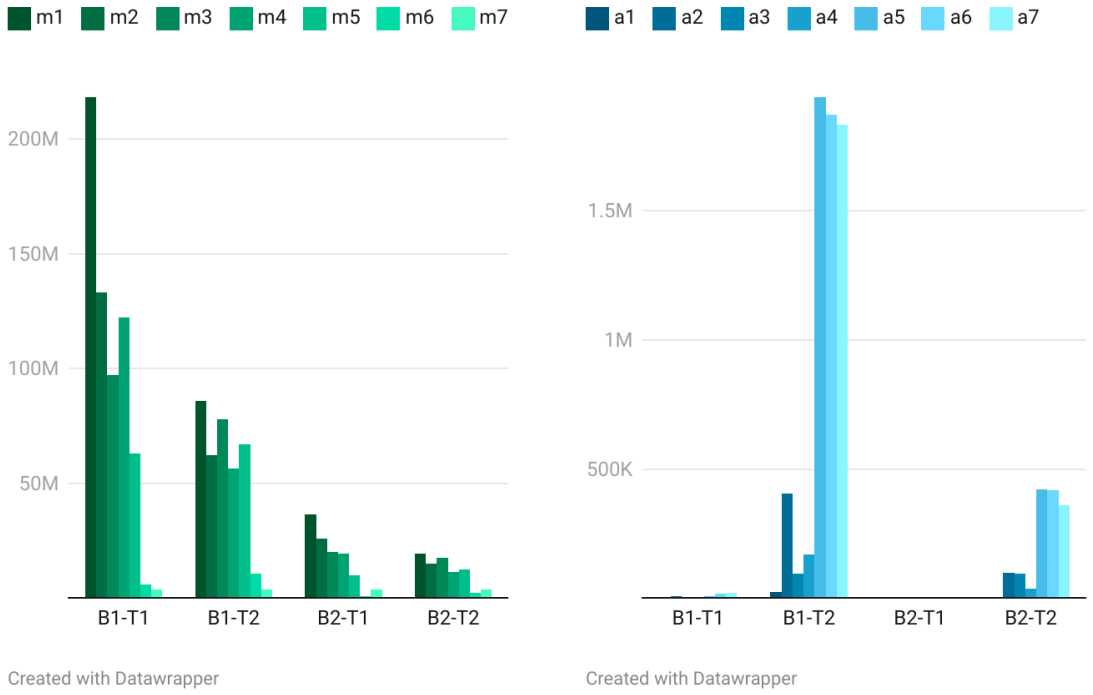


Fig. 5.1.: Quantity of PSEs examined by the brute-force algorithm within a time limit of 50 minutes. The number corresponds to the outer loop's iterations of Algorithm 2.

Despite the poor efficiency, other important observations can be made regarding the necessity of lazy scoring. Therefore, the inner loop of Algorithm 2 can be observed, whose executions correspond to the number of explored drawings. The following two sections will therefore refer to Figure 5.1, which illustrates the achieved iterations. A tabular overview is provided in Appendix B.2. Results with the suffix T2 refer to runs, that applied tracked scoring, while the slow variant was used for the others. In addition, datasets with the prefix B1 are those, which have been compiled with the optimization flag `-Ofast`. In contrast, B2 identifies those, that have been compiled with default flags.

At first, these statistics legitimise the utilization of tracked scoring as standard methodology for score calculation. Considering the results for large drawings, the number of iterations for the tracked calculation was increased by a factor of approximately 100 on average, which is indicated by the comparisons of B1-T1 and B1-T2 as well as B2-T1 and B2-T2. A remarkable improvement can be observed especially for a_5, for which only 6,602 iterations were performed in B1-T1 and 1,936,682 in B2-T2. On the other hand, the number of iterations decreases when tracked scoring is applied to smaller PSEs, as can be seen for the left pairs. However, the inequalities given in Lemmas 12 and 13 provide sufficient explanations for this situation. Nevertheless, the average number of explorations for the smaller PSEs exceeds one million per minute in B1-T2, even with the tracked variant in use. Because helpful on-the-fly calculations can be carried out alongside the tracking process, this disadvantage is taken into account in the following.

Furthermore, the importance of applying optimization flags during compilation is demonstrated. As already discussed, these help to resolve disadvantages, which are caused by some design decisions regarding the object-oriented programming. For instance, this can be observed by comparing B1-T2 and B2-T2, whereas the first was compiled with `-Ofast`. The utilization increased the number of completed iterations within the same amount of time by an average factor between 4 and 5. Ultimately, this leads to the decision, that all further test cases are also compiled in this way.

5.2. Force-directed algorithms

Next, the two FDA's are compared with each other. Since FR's sub-strategy is newly considered for the combined approach, it is at first necessary to find favourable parameter combinations for $T \in \mathbb{N}$, $l \in \mathbb{R}^+$ and $c \in [0, 1)$. To summarize once again: l describes the optimal length of the drawn edges, c the base for exponential cooling and T the maximum number of iterations. For this purpose, the parameter combinations of Table 5.2 were tested. Tabular overviews of all obtained results are provided in the appendices B.3 and B.4. Before taking a closer look at the length parameter $l \in \mathbb{N}$, it makes sense to evaluate the impact of other tunable variables at first. This might exclude some cases, for which no further consideration would be worthwhile. Accordingly, the effects of the selected cooling and the number of iterations will be discussed first. Therefore, Figure 5.2, which visualizes the results of parameter combinations F4-T1 to F4-T9 for the Automatics, will be considered in the following paragraphs.

label	l	T	c
F1-T1	1	32	0.992
F1-T2	1	64	0.992
F1-T3	1	128	0.992
F1-T4	1	32	0.9999
F1-T5	1	64	0.9999
F1-T6	1	128	0.9999
F1-T7	1	32	0.85
F1-T8	1	64	0.85
F1-T9	1	128	0.85
F2-T1	5	32	0.992
F2-T2	5	64	0.992
F2-T3	5	128	0.992
F2-T4	5	32	0.9999
F2-T5	5	64	0.9999
F2-T6	5	128	0.9999
F2-T7	5	32	0.85
F2-T8	5	64	0.85
F2-T9	5	128	0.85

label	l	T	c
F3-T1	50	32	0.992
F3-T2	50	64	0.992
F3-T3	50	128	0.992
F3-T4	50	32	0.9999
F3-T5	50	64	0.9999
F3-T6	50	128	0.9999
F3-T7	50	32	0.85
F3-T8	50	64	0.85
F3-T9	50	128	0.85
F4-T1	100	32	0.992
F4-T2	100	64	0.992
F4-T3	100	128	0.992
F4-T4	100	32	0.9999
F4-T5	100	64	0.9999
F4-T6	100	128	0.9999
F4-T7	100	32	0.85
F4-T8	100	64	0.85
F4-T9	100	128	0.85

Tab. 5.2.: Parameter combinations with which FR was tested.

When taking a look at the next page's diagram, tests F4-T7 to F4-T8 are immediately apparent: For each drawing, the parameter $c = 0.85$ seems to cause only marginal improvements, if any at all, after the 32nd iteration. Closer analysis even revealed, that this is the case already before the 16th iteration. Moreover, the intermediate results achieved up to this point are unsatisfactory, as can be gathered from a comparison with F4-T1. For example, the parametrization F4-T1 generated only 14,080 crossings for a_5, while F4-T7 was unable to resolve a total of 662,321 more crossings. Consequently, a scenario occurred, which should actually be prevented: The drawing freezes too quickly. Considering this, it is reasonable to exclude this parameter from further investigations.

For the other datasets, it can be concluded, that most improvements are indeed made before the 32nd iteration. Only the drawings a_3, a_6 and a_7 are improved decisively after this timestamp. Considering the non-illustrated test cases, this regularity applies in particular, if the length parameter's value l is high. This suggests, that unfavourable arrangements of vertices are resolved extraordinarily fast in these cases. Thus, prior the first measuring point, most of the drawings are already relatively stable, which causes the positive influences to converge. Nevertheless, unlike in the exceptional cases F4-T7 to F4-T8, stagnation is not the case. Accordingly, the number of crossings tends to decrease between the 62nd and 124th iteration. Nor is it a disadvantage to execute 128 times, since the execution of FDAs was always completed within two minutes, leaving sufficient time for the SA. A noticeable exception to this conclusion is, that drawing a_5 deteriorates by 402 crossings during 96 further iterations in test F4-T3, although a very satisfactory result of 4 crossings was already found in F4-T1. Similar exceptional events occasionally happen for other lengths and drawings as well. However, at least in the

context of this thesis, no systematic correlation between a particular base c, the number of iterations T and the frequency of unexpected deteriorations was identified.

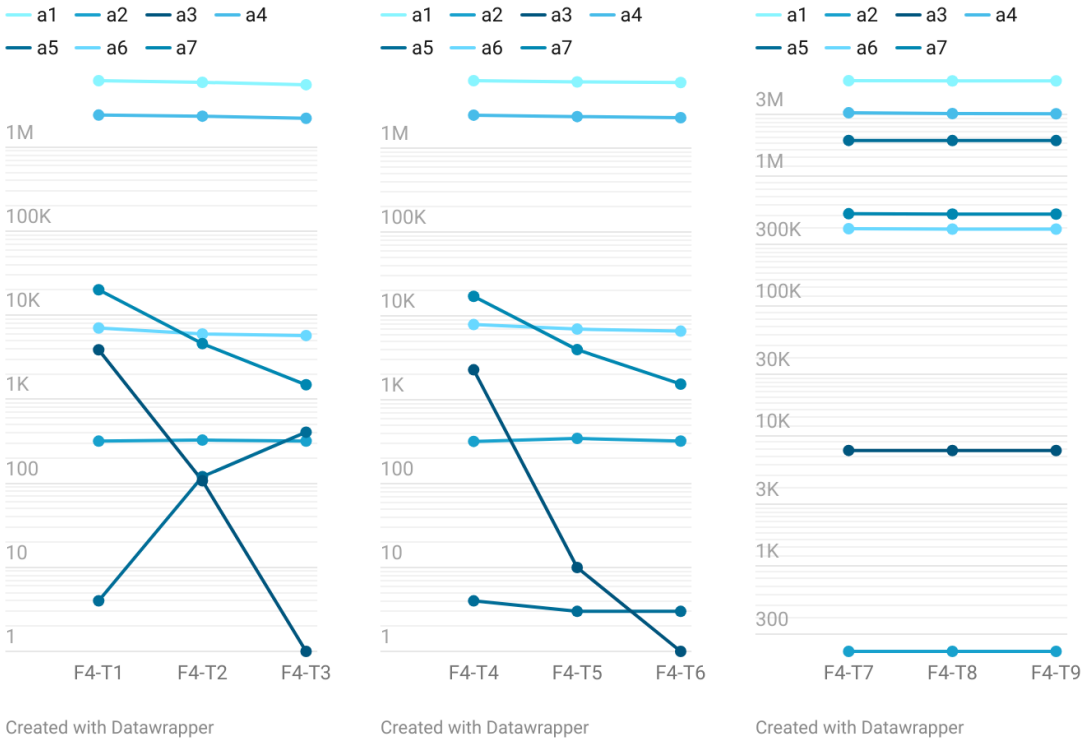


Fig. 5.2.: Scores obtained from FR for the parameters F4-T1 to F4-T9. Grouped tests utilize the same base for cooling and are ordered according to the maximal count of iterations, so that the iterative development of the score is also visualized.

To summarize, it is reasonable to further consider all parameter combinations with $T = 124$ and a cooling tuned by $c \in \{0.992, 0.9999\}$. Consequently, the optimum length parameter must now be found, for which Figure 5.3 will be of assistance.

A closer look at the Manuals' data indicates an upward and downward shift in the obtained scores. This scenario is caused by the cooling parameter, which is $c = 0.992$ for the T3 and $c = 0.9999$ for the T4 datasets. Because lower scores are consistently yielded for the first choice, a systematic difference between the two bases can be observed. In contrast, the replacement of parameter l only had a strong influence on m_6. For this drawing, F2-T6 delivered the worst result with 906 and F3-T6 the best with 509. However, the parameter combination F4-T3 delivered the best results on average.

Similar definite observations regarding the optimal cooling cannot be made, if the Automatics are considered next. For example, tests for the PSEs a_5 and a_7 achieved better scores with $c = 0.9999$, although the others obtained preferable results with $c = 0.992$. Furthermore, it can be seen, that significant improvements can generally be achieved

with a large length parameter. Therefore, the scores for a_4 develop in an exemplary manner. The scores obtained for this drawing reached a value of 5,981,872 with $l = 1$ in F1-T3, while $l = 100$ in F4-T3 achieved 2,199,384 – an improvement of more than half. Similar results also apply to a_6 and a_7. The only exception to this pattern is a_2, whose values deteriorated for increasing choices of l from 66 in F1-T3 to 318 in F4-T3. If this deterioration is set in relation to the enormous improvements of a large length, this difference can be neglected. Overall, the parameter combination F4-T3 also achieved the best on average results for the more complex drawings.

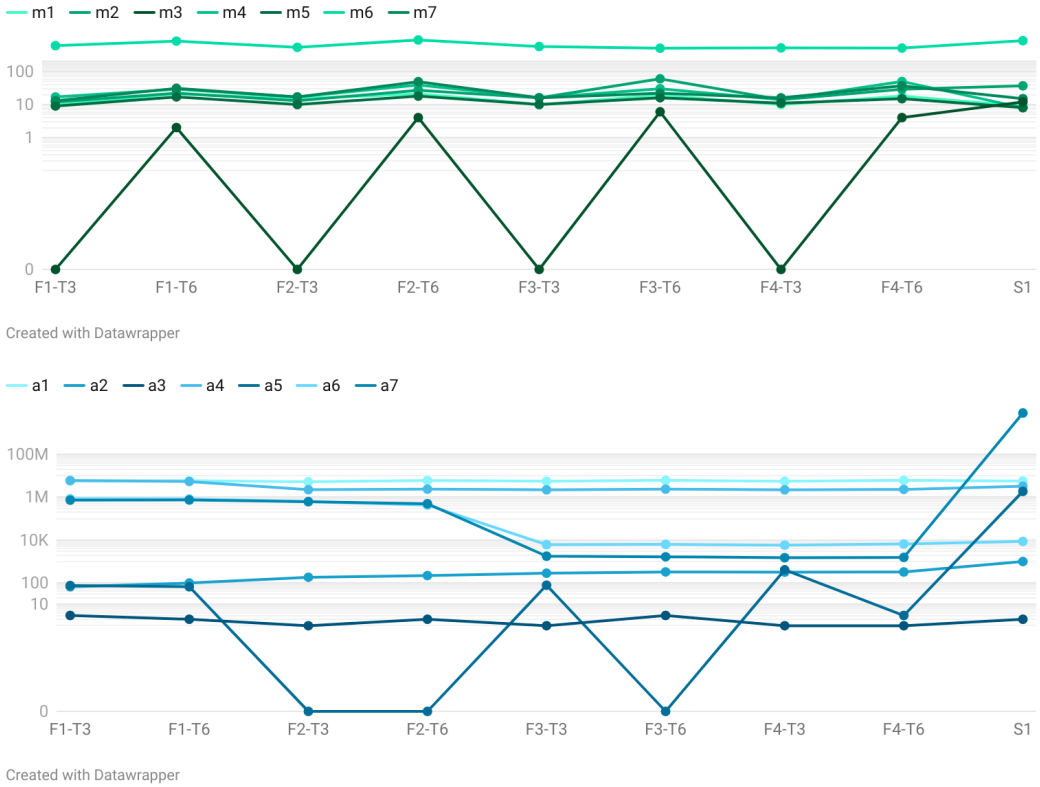


Fig. 5.3.: Scores obtained from FR for all parameter combinations with $T = 124$, except of the discarded base $c = 0.85$. In addition, S1 shows the results of Eades' spring embedder with the parameters $c_{attr} = 10$, $c_{repl} = 20000$, $c = 0.992$ and $l = 5$.

Based on the previous observations and conclusions, it is evident, that the parameter combination F4-T3 expectedly achieves the best outcomes for FR. Now, the results of this FDA must finally be compared with Eades' spring embedding. Since Kutscheid already conducted detailed testing to discover favourable parameter combinations, no further analysis for Eades' FDA are required nor repeated in this thesis. Instead, the parameter combination recommended by Kutscheid will be utilized: $c_{attr} = 10$, $c_{repl} = 10000$, $l = 5$ and $c = 0.992$ [Kut24]. The results of Eades' spring embedder with these parameters and $T = 124$ will be referenced below with the label S1.

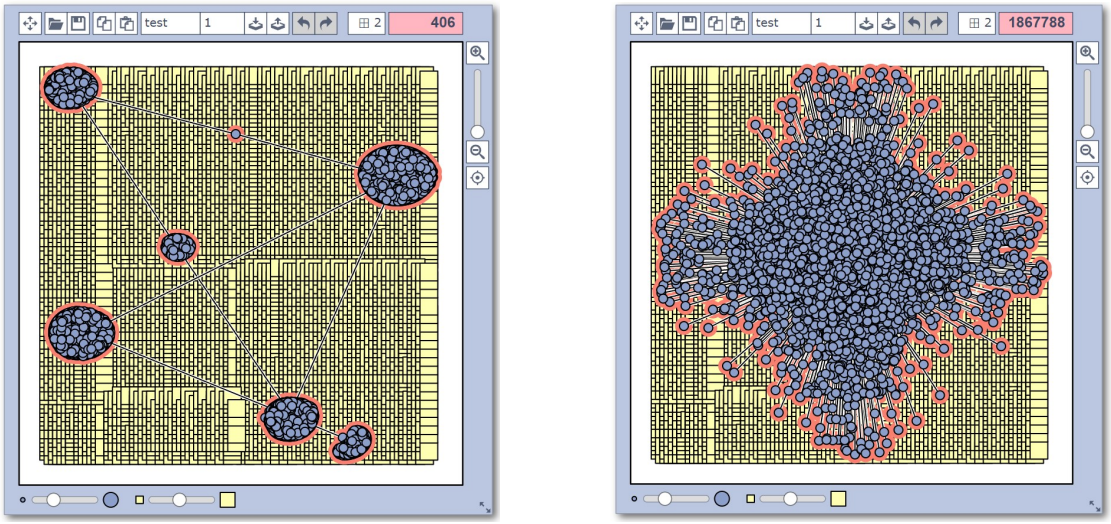
label	m_1	m_2	m_3	m_4	m_5	m_6	m_7
S1	9	37	12	8	8	865	15
F4-T3	10	14	0	14	11	520	16

label	a_1	a_2	a_3	a_4	a_5	a_6	a_7
S1	5,672,562	987	2	3,231,928	1,867,788	8,680	8,529,555,000
F4-T3	5,520,554	318	1	2,199,384	406	5,725	1,485

Tab. 5.3.: Comparison of the scores obtained from the best parametrisations of both FDA's. All algorithms were tested with a maximum of $T = 124$ iterations.

Finally, a comparison of both FDAs can be carried out, for which an overview is provided in Table 5.3. It is immediately apparent, that the FDA by FR achieves significantly better results for the more complex and larger drawings. In fact, there is not a single drawing in the category of the Automatics, for which Eades' embedder performs better. Particularly significant are the improvements of a_7, where F4-T3 even achieves an improvement in the billions. For the drawings a_2, a_4 and a_6 the scores were reduced by approximately two thirds each. The sole Automatic for which only relatively small improvements were achieved is a_1, so that the score improves by only about 8%. However, the relativity of the latter comparison must be emphasized, since a_3 was only improved by 1, which is still a relative improvement of a half.

On the other hand, the Spring Embedder by Eades seems to achieve equally good results for the Manuals. It even brings marginal improvements for m_1, m_4, m_5 and m_7. Nevertheless, FR can also be considered to be the superior approach here, because the large improvements for m_2 and m_6 outweigh the marginal deteriorations.



(a) Result of F4-T3.

(b) Result of S1.

Fig. 5.4.: Comparison of the two FDAs applied with optimal parametrizations on drawing a_5. A maximum of $T = 124$ iterations was defined.

The results of PSE a_5, which can be seen in Figure 5.4, exemplify how FR's key ideas have successfully yielded improvements. On the left side, the better solution shows, that there are strongly connected subsets within the vertex-set, which cause fewer crossings when distributed across clusters. This is not the case on the right-hand side, where all the vertices are located in a large and central accumulation. It therefore seems, that the forces calculated with S1 cause local minima, from which the vertices cannot escape, while the extremely high attractive forces of FR assisted in resolving these.

As already mentioned, these interim results are invalid PSEs and subsequently have to be embedded utilizing the presented greedy assignments. During this process, a considerable deterioration in the score must be expected. The magnitude of this deterioration should additionally depend on how even the drawn vertices are distributed in the plane. It is therefore expedient to compare the results of both FDAs again, but now after the assignments have been applied. For this purpose, Table 5.4 presents a compact overview of the embedded intermediate results, which are the starting points for the SA. The results of Eades' FDA and the assignments were additionally verified with Kutscheid's program, because a definite tendency is indicated. However, these samples generated similar results, while insignificant differences are plausibly explainable by the additional min-max normalization and slightly differing values for T and ε. Consequently, the displayed results confirm, that FR yields significantly better outcomes for the Automatics, even when embedded by greedy assignments. The solely exception to consider is drawing a_4, for which the fast assignment of Eades' outcomes achieves a better score than both assignments of F4-T3. For the manual drawings on the other hand, both FDA's achieve equally satisfactory results. As was assumed, it is important to emphasize the extent to which the assignments cause deteriorations, especially for the Automatics.

label	m_1	m_2	m_3	m_4	m_5	m_6	m_7
S1_f	14	52	670	53	859	1,572	320
S1_s	14	69	600	37	1,453	1,422	204
F4-T3_f	12	55	284	23	946	1,383	331
F4-T3_s	9	69	209	18	864	703	332

label	a_1	a_2	a_3	a_4
S1_f	8,554,575	401,786	49,609,340	10,445,229
S1_s	7,837,526	139,061	28,565,914	118,313,253
F4-T3_f	5,580,218	342,506	16,688,138	18,165,757
F4-T3_s	5,634,345	106,037	7,670,227	89,568,674

label	a_5	a_6	a_7
S1_f	49,495,213	275,714	8,951,028,000
S1_s	98,626,741	181,507	8,572,500,000
F4-T3_f	32,690,218	31,023	3,218,733,000
F4-T3_s	31,572,197	33,983	2,600,058,000

Tab. 5.4.: Comparison of the scores obtained from the best parametrizations of both FDA's. All algorithms were tested with a maximum of $T = 124$ iterations. Letter "f" identifies PSEs embedded by the fast and "s" those embedded by the slow greedy assignment.

Moreover, the differences caused by the two greedy algorithms must be discussed. During his work, Kutscheid came to the conclusion, that utilizing the slow greedy assignment is preferable. The considered table confirms this in general, because the improvements of F4-T3$_\text{f}$ for a_2, a_3 and a_7 outweigh the slight deterioration for a_1, a_5 and a_6. However, the extraordinary exception is a_4, whose enormous deterioration due to the slow variant cannot be ignored for both FDA's. As a consequence, the greedy approach will be manipulated for the GDC 2024. Hence, the final implementation applies both assignments multi-threaded and returns the better result.

Taking all these observations and conclusions together, the application of FR's spring embedding variant with the parameters $l = 100$, $T = 124$ and $c = 0.992$ continuously yields superior results for large and complex drawings. These advantages remain, even if the intermediate results are subsequently converted into valid PSEs by greedy assignments. In contrast, the original variant of Eades performs slightly better for smaller drawings, although the marginal differences should be quickly eliminated during SA.

5.3. Simulated Annealing

In the preliminary work, the focus of the investigations regarding SA was placed in particular on the determination of suitable parameter combinations for exponential cooling. Unfortunately, other interesting aspects could therefore not be pursued further. For this reason, the next experiments will concentrate on other aspects. In addition to the Random Walk, the Rebuild Neighbourhood approach will be investigated as a second refactoring technique. Furthermore, the influence of the SA's reset frequency, which is specified in s seconds, will be fine-tuned for both refactoring approaches. Similarly, three different probability functions are considered for the randomized selection of vertices, which can be controlled with the parameter $p \in \{0, 1, 2\}$. In order to allocate the required time capacities for these plans, exponential cooling parametrized with $\theta = 2000$ and $c_{exp} = 0.85$ as well as $\theta = 1$ and $c_{exp} = 0.94$ will be applied without further comparisons. Both of these parametrizations have made it into Kutscheid's final recommendations for SA [Kut24]. The first one stands out, because of it's high starting temperature and fast cooling. As a result, the probability, that degradations are accepted in early iterations of the inner loop, is very high, but then decreases rapidly. Contrary to this, the second introduces moderate cooling with consistently lower acceptance probabilities.

5.3.1. Random Walk

At first, the Random Walk will be investigated using the parameter combinations specified in Table 5.5. Since the employed cooling functions are strongly differing, it is appropriate to consider them separately. For this reason, the next few sections will begin to discuss the fast cooling variant using Figure 5.5. Afterwards, the moderate function is considered according to Figure 5.7. The discussion will focus on noteworthy findings, while all results are listed in the appendices B.5 and B.6.

label	s	p	c_{exp}	θ
W1-T1	60	0	0.85	2,000
W1-T2	30	0	0.85	2,000
W1-T3	15	0	0.85	2,000
W1-T4	10	0	0.85	2,000
W1-T5	5	0	0.85	2,000
W2-T1	60	2	0.85	2,000
W2-T2	30	2	0.85	2,000
W2-T3	15	2	0.85	2,000
W2-T4	10	2	0.85	2,000
W2-T5	5	2	0.85	2,000
W3-T1	60	1	0.85	2,000
W3-T2	30	1	0.85	2,000
W3-T3	15	1	0.85	2,000
W3-T4	10	1	0.85	2,000
W3-T5	5	1	0.85	2,000

label	s	p	c_{exp}	θ
W4-T1	60	0	0.94	1
W4-T2	30	0	0.94	1
W4-T3	15	0	0.94	1
W4-T4	10	0	0.94	1
W4-T5	5	0	0.94	1
W5-T1	60	2	0.94	1
W5-T2	30	2	0.94	1
W5-T3	15	2	0.94	1
W5-T4	10	2	0.94	1
W5-T5	5	2	0.94	1
W6-T1	60	1	0.94	1
W6-T2	30	1	0.94	1
W6-T3	15	1	0.94	1
W6-T4	10	1	0.94	1
W6-T5	5	1	0.94	1

Tab. 5.5.: Parameters for which SA with Random Walk and exponential cooling was tested.

Considering the results from the diagram 5.5, it can be seen, that rapid cooling in this parameter constellation yielded constant scores for many of the Manuals. Especially positive is, that Score(m_1) = 3, Score(m_2) = 25, Score(m_4) = 4, Score(m_5) = 4 and Score(m_7) = 19 were achieved without exception. Similarly, Score(m_3) = 0 was always achieved, except for one case. These results correspond to each case to the best solutions from the previous year [BKK$^+$23b]. Thus, they are an indicator, that SA and the combined approach achieve very satisfactory results, at least for smaller drawings. The only drawing whose scores do not remain constant in the upper diagram is m_6, which is also the largest PSE among the Manuals. However, a closer look at the differences, which can hardly be recognized in the diagram, due to their marginality, reveals first impacts of the different probability functions. For example, an average score of 420.8 was achieved for m_6 with $p = 0$, while tests with $p = 2$ or $p = 1$ yielded 412.2 respectively 409.4 crossings on average and therefore tended to perform better. Statements about a systematic influence of the frequency s cannot be made for the Manuals.

Many more differences can be observed for the Automatics, whose results are illustrated in the lower diagram of 5.5. A steadily decreasing trend can be recognized between the tests T1 and T5 of each prefix. This curve behaviour proceeds parallel to the reduction of parameter s. This trend is so absolute, that the best outcomes of almost all drawings were achieved with a frequency of $s = 5$ seconds. The only exception is a_5, for which the best results were systematically found with a reset frequency between 10 and 15 seconds. The positive influence, that was achieved for a_7, is particularly noteworthy. The solution generated in W1-T1 and a frequency of $s = 60$ seconds was rated with 1,847,403,000, while W1-T5 and $s = 5$ was rated with 1,111,602,000. This systematically noticeable situation is additionally interesting, since a_7 corresponds to the edge-case of linearly arranged point-sets. One conclusion, that a high reset frequency brings advantages in

this case, must be, that deteriorating adjustments are frequently accepted shortly after a reset, when the temperature is still high. These modifications can be so poor, that they cannot be recovered until the next reset. According to the arrangement, this is the case, if an edge is drawn in a manner, that an extraordinary large number of points is placed on the straight line between the corresponding endpoints. This situation has already been illustrated in Figure 3.5. Especially with the completely arbitrary placements of the Random Walk, this should happen frequently. A high reset frequency counters this, because it quickly discards such unrecoverable deteriorations.

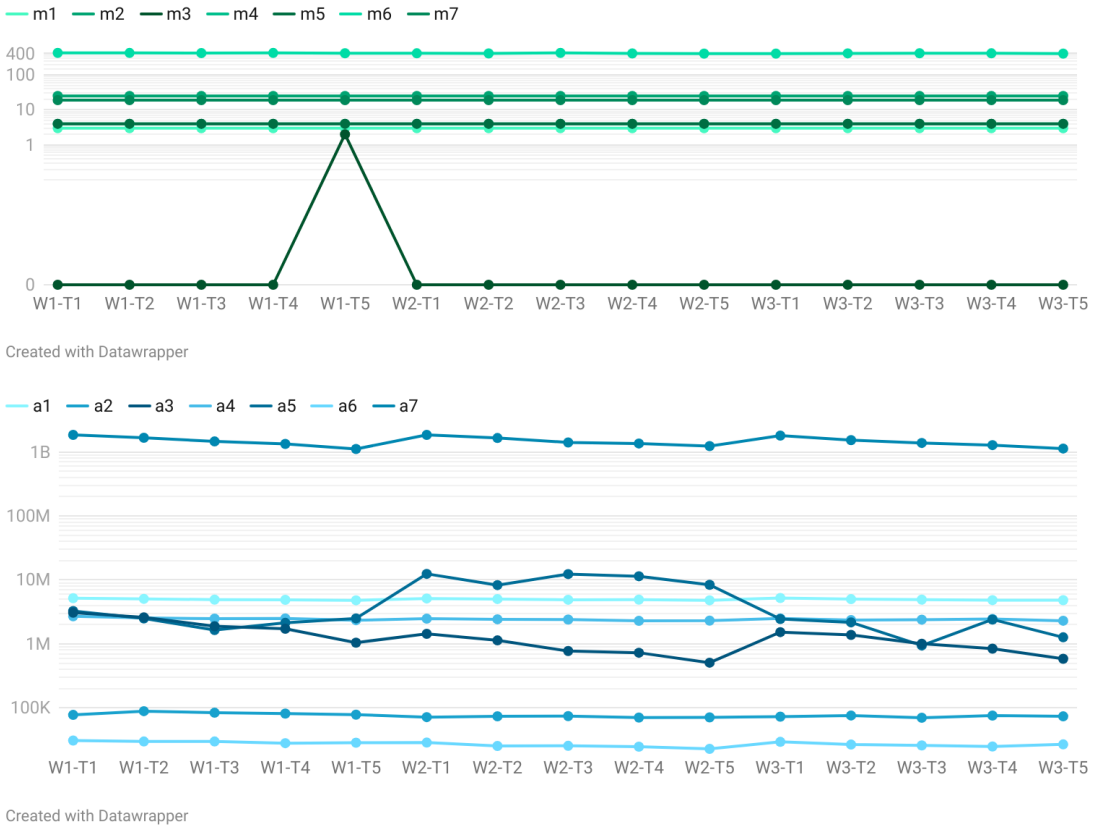


Fig. 5.5.: Scores obtained from SA with Random Walk and a rapid exponential cooling. The parameter combinations are provided in Table 5.5. Previously, the drawings had been optimized with FR and combined greedy assignment.

The positive influences, that the different probability functions have on the PSE a_3 are especially remarkable. For instance, a score of 1,046,040 was achieved for this drawing in W1-T5, which set $p = 0$. In contrast, the test W2-T5 with quadratically influenced probabilities found a solution with only 508,475 crossings – less than half. Also the test W1-T5, which set $p = 1$, performed significantly better with 586,639 crossings. For most of the other PSEs, the different probabilities do not show significant differences, although slightly better solutions tend to be found with $p = 1$ or $p = 2$.

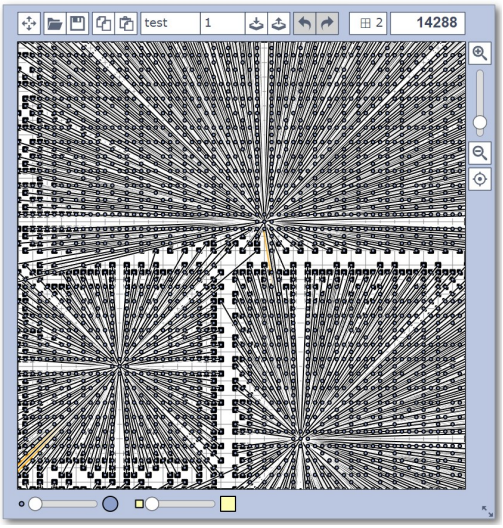


Fig. 5.6.: A cut-out of the best solution for a_5 from the GDC 2023 [BKK+23b]. Some vertices have extraordinarily high degrees, while the average degree is rather low.

Again, the PSE a_5 is an exception to the last conclusion, for which significant poorer results were yielded with the probability function generated by $p = 2$. An explanation for this situation can be found in a special characteristic of a_5, which is illustrated in Figure 5.6. A closer examination of Table 5.1 reveals, that some vertices have an extraordinarily high degree of up to 1,974, while the average is only 2. As a consequence, the penalty operator continuously outputs very large values for the strongly connected vertices. This effect is then further amplified by the squaring. Ultimately, this leads to the situation, that especially the strongly connected vertices are continuously shifted and probably always without significant improvements. This observation immediately suggests, that this particular case could be avoided, if the summation of the penalties would be adjusted by averaging. Since the implementation at hand does not yet do this, the probabilities generated with $p = 2$ cause more edge-cases than they solve. This means, that $p = 2$ should be dismissed in principle, although it caused improvements in some cases. Interestingly, the same effect does not occur for a_5, when the probabilities are generated with $p = 1$. The best solutions obtained for this PSE were therefore found with fast cooling and the parameter combinations W3-T3 and W3-T5.

Considering the results obtained with moderate cooling, as shown in Figure 5.7, mainly parallels become apparent. Once again, the Manuals were solved consistently well, although this is less due to adequate cooling. Instead, this favourable results are caused by an average of 3,170,874 modifications, which can be examined for small drawings per minute. Nevertheless, the results of m_3 oppose static and uniform probabilities for the selection of vertices to be repositioned. For this drawing, optimal solutions were not found with the parameter $p = 0$ twice, which suggests more than an occasional exception. Similarly, $\text{score}(m_6) = 398$ was achieved with $p \neq 0$ in the tests W5-T5 and W6-T3, which is only marginally worse than the contests best solution [BKK+23b].

Fig. 5.7.: Scores obtained from SA with Random Walk and a moderate exponential cooling. The parameter combinations are provided in Table 5.5. Previously, the drawings had been optimized with FR and combined greedy assignment.

The absolute results achieved with the two coolings generally differ comparatively little. However, an interesting and systematic exception to this is drawing a_6. As an example, the outcomes of W1-T1 to W1-T5 and W4-T1 to W4-T5 are compared with each other. Firstly, the moderate cooling achieves a score of 23,686 with a reset frequency of 60 seconds, which is already an improvement of 4,850 crossings compared to W1-T5. Secondly, the score's development with an increasing s is interesting. From W1-T1 to W1-T5, only marginal improvements occurred. In contrast, if we look at the development between W4-T1 and W4-T5, the score fell by around a tenth in proportion to the gradual increase of the frequency. Ultimately, a score of 14,977 was achieved with W4-T5, which is even better than the best result found at the last competition [BKK+23b]. Since the same pattern can be observed between W6-T1 and W6-T5, an increased reset frequency and moderate cooling seems to have a mutually reinforcing effect for this PSE.

Summarizing these conclusions, $s = 5$ seconds can be determined as a suitable runtime for the SA's inner loop and $p = 2$ should be discarded, because it can cause unfavourable deviating probabilities for drawings with strongly unequal vertex degrees. According

to the exclusion principle, W1-T5, W3-T5, W4-T5 and W6-T5 remain as promising parameter combinations, which are to be compared in absolute values. Therefore, Table 5.6 provides a overview of the final results obtained in the corresponding tests. It can be seen, that apart from the solution for PSE a_1, all best solutions were found in the tests W3-T5 or W6-T5, which both set $p = 1$. Some of them even with significant margin. However, it should be emphasized, that the differences for a_1 are negotiable. The comparison of the utilized cooling functions shows only marginal differences for a_1, a_3, a_4 and a_7. In contrast, the systematic differences for the drawings a_2 and a_6 are significant enough, to introduce moderate cooling, controlled by $\theta = 1$ and $c_{exp} = 0.94$, as the default setting for the toolbox. This result is also conform with the cooling, which Kutscheid highlighted as the best of his recommendations [Kut24]. In addition, it is expedient to proceed with the probability function, which considers the penalties in a linear manner and is generated by the parametrization $p = 1$.

label	m_1	m_2	m_3	m_4	m_5	m_6	m_7
W1-T5	3	25	2	4	4	414	19
W3-T5	3	25	0	4	4	404	19
W4-T5	3	25	0	4	4	395	19
W6-T5	3	25	0	4	4	411	19

label	a_1	a_2	a_3	a_4	a_5	a_6	a_7
W1-T5	4,806,314	78,381	1,046,040	2,347,978	2,504,333	28,536	1,111,602,000
W3-T5	4,820,138	73,641	585,639	2,301,006	1,266,689	26,877	1,130,448,000
W4-T5	4,784,304	64,101	1,088,192	2,182,424	1,803,207	14,977	1,138,653,000
W6-T5	4,797,353	59,505	618,957	2,174,089	2,665,358	12,305	1,069,290,000

Tab. 5.6.: Comparison of outcomes from the best parametrizations for SA with Random Walk. The PSEs were previously optimized with FR and combined greedy assignment.

5.3.2. Rebuild Neighbourhood

In the next step, the alternative Rebuild Neighbourhood refactoring technique will be examined. This requires a further parameter to be considered: $\kappa \in [0, 1]$. The variable determines the probability with which vertices may not be placed on nearby points, but on distant ones instead. In other words, it indicates how strictly the rule is adhered to. Accordingly, the neighbourhood heuristic is employed without exceptions, if $\kappa = 0$ is set. The first considered tests will be focused on this parameter, which helps to find out whether the heuristic is helpful at all. Subsequently, the most promising combinations will be tested in a second series of tests with varying reset frequencies. Therefore, only variations of the parameters p and κ as listed in Table 5.7 are considered at first, while the SA's reset timer is fixed to $s = 30$ seconds. As in the previous discussions of the Random Walk, rapid cooling will be initially discussed alongside Figure 5.8, while results obtained with moderate cooling are considered separately. Similarly, all scores can be found in a tabular overview within the appendices B.7 and B.8.

label	s	p	κ	c_{exp}	θ
R1-T1	30	0	0.3	0.85	2,000
R1-T2	30	2	0.3	0.85	2,000
R1-T3	30	1	0.3	0.85	2,000
R2-T1	30	0	0.5	0.85	2,000
R2-T2	30	2	0.5	0.85	2,000
R2-T3	30	1	0.5	0.85	2,000
R3-T1	30	0	0.7	0.85	2,000
R3-T2	30	2	0.7	0.85	2,000
R3-T3	30	1	0.7	0.85	2,000
R4-T1	30	0	0	0.85	2,000
R4-T2	30	2	0	0.85	2,000
R4-T3	30	1	0	0.85	2,000

label	s	p	κ	c_{exp}	θ
R5-T1	30	0	0.3	0.94	1
R5-T2	30	2	0.3	0.94	1
R5-T3	30	1	0.3	0.94	1
R6-T1	30	0	0.5	0.94	1
R6-T2	30	2	0.5	0.94	1
R6-T3	30	1	0.5	0.94	1
R7-T1	30	0	0.7	0.94	1
R7-T2	30	2	0.7	0.94	1
R7-T3	30	1	0.7	0.94	1
R8-T1	30	0	0	0.94	1
R8-T2	30	2	0	0.94	1
R8-T3	30	1	0	0.94	1

Tab. 5.7.: Parameters for which SA with Rebuild Neighbours was tested.

In contrast to the simple refactoring method, Rebuild Neighbourhood was only able to continuously achieve the best results from the previous year for m_1 and m_2. Consequently, a deterioration can be seen for most of the Manuals, although these are minor for m_2 and m_7. This observation is not surprising. On the one hand, the application of heuristics in this manner causes, that potentially good embeddings are explored faster. At the same time, patterns deviating from the suggested heuristics are less likely or even prevented to be explored. The Random Walk is capable of examining a huge amount of modifications for smaller PSEs. Because of that, the heuristic's first effect becomes irrelevant and the second even counterproductive. This is particularly apparent in the scores obtained for PSE m_2. All tests found optimal solutions for this drawing, except for R4-T1 to R4-T3, which set $\kappa = 0$. Consequently, strict adherence to the heuristic seems to block purposeful modifications for this PSE – the optimal solution is not supported. Drawing m_6 indicates, that this is not always the case. For this instance, the parameter combinations R3-T1 to R3-T3, and accordingly $\kappa = 0.7$, achieved a score of 703. This result is 175 crossings worse than the best solution, which was found in R4-T1 with $\kappa = 0$. A relaxation of the heuristic has consequently a significantly worsening effect for this PSE. Nevertheless, both of these solutions are considerably worse than those obtained with the Random Walk, whose worst for m_6 was evaluated with 424 crossings.

In comparison to the tests with $\kappa \neq 0$, a consequent application of the heuristic, which is given in the tests R4-T1 to R4-T3, has a better or non-deteriorating effect on most of the Manuals. Hence, apart from the already mentioned case m_2, the best results were consistently achieved within the R4 prefixed test series. A further remarkable example of this situation is provided with m_3, whose best solutions of the GDC were only nearly missed in the tests R4-T1 to R4-T3, while $\kappa \neq 0$ led to significantly worse outcomes.

Based on these findings, it must be concluded, that SA with Rebuild Neighbourhood and a rapid cooling is not a suitable optimization method for small drawings. But if it is to be applied, a tendency is in favour of $\kappa = 0$.

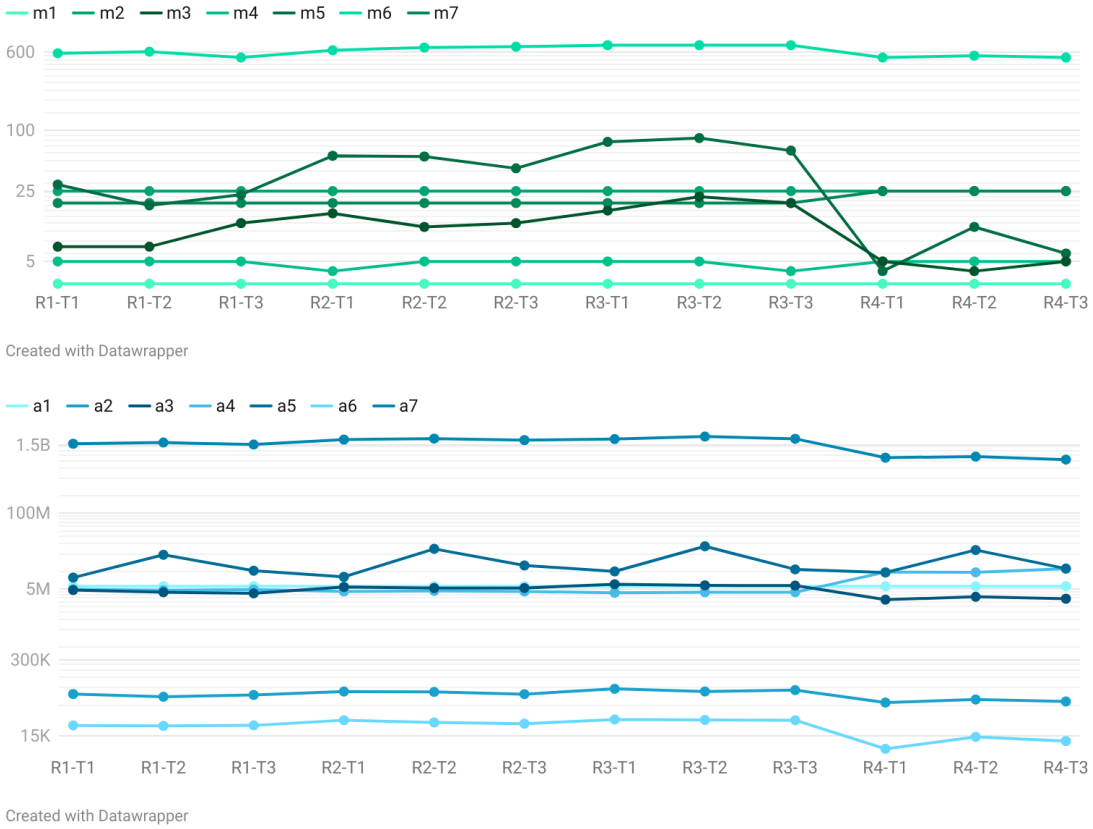


Fig. 5.8.: Scores obtained from SA with Rebuild Neighbourhood and a rapid exponential cooling. The parameter combinations are provided in Table 5.7. Previously, the drawings had been optimized with FR and combined greedy assignment.

Considering the Automatics as displayed in the lower diagram, negative deflections can be seen more frequently in experiments with the suffix T2. These can be attributed to the selection of parameter $p = 2$, which particularly influences the PSE a_5. However, this particular pattern has already been revealed for a_5 in the discussions about Random Walk. In contrast, these discrepancies now also affect other drawings. This is reflected in the fact, that tests with the suffix T2 often exhibit deteriorations, especially between R4-T1 and R4-T3. This provides an initial indication, that the quadratic probability function seems to be unsuitable, if it is applied with Rebuild Neighbourhood.

Parallel to the Manuals, the crossings in large drawings tend to be reduced when the heuristic is fully applied. Thus, the results of a_3, a_2, a_6 and a_7 have steadily deteriorated as the value of κ increases and the best results were usually obtained in the tests R4-T1 or R4-T3. Once again, the edge-case a_5, which was already visualized in Figure 5.6, delivers an interesting exception. This drawing's best score of 8,085,395 was achieved with $\kappa = 0.5$ in R2-T1, which is still much worse than the best solution achieved with the Random Walk. A clue for this extraordinary discrepancy is, that a good solution

for a_5 consists of a star-shaped distribution of vertices. Additionally complicating is, that the high degreed vertices must be placed into exactly the right spots. Achieving such an arrangement with fully automated and randomized methods, such as SA, is unlikely and even less probable with the applied heuristic. A semi-automatic approach would be more expedient here, with which the central vertices could be placed and fixed manually. Desirable solutions for a_1 and a_4 also seem to be not covered by the heuristic. As a result, 50 minutes pass for these three drawings without any notable improvements being achieved. PSE a_1, which is visualized in Figure 5.9, is especially noticeable in this context, because it was constantly scored with 5,580,218 in each test about Rebuild Neighbourhood. According to Table 5.4, this corresponds to the intermediate result of the spring embedding variant of FR, meaning that not a single improvement was generated for this drawing in the complete time span of 50 minutes.

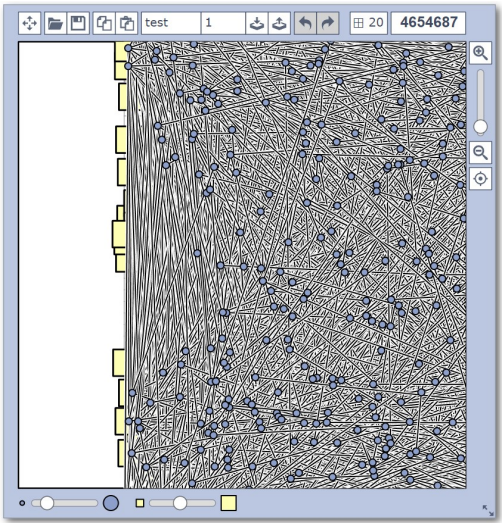


Fig. 5.9.: A cut-out of the best solution for a_1 from the GDC 2023. A heuristic, that generates comparable solutions, must also consider long edges, unlike the one employed.

When the results obtained with moderate cooling are considered next, which are illustrated in the diagram 5.10, similar conclusions can be drawn as for the rapid cooling. Thus, negative deflections can especially be found, where $p = 2$ is set. In consequence, all parameter combinations, that generate a quadratic probability function, are also discarded for Rebuild Neighbourhood. Furthermore, the best results for drawings, that are supported by the heuristics, are also achieved, when $\kappa = 0$ is fixed.

An interesting situation is observable, if the results gained by the different cooling functions are compared directly with each other. Within the test series about the Random Walk, a relatively clear tendency emerged, that moderate cooling systematically produces better scores. This improvement was particularly noticeable for solutions of the drawings a_2 and a_6. In contrast, a provable tendency cannot be observed, when the neighbourhood heuristic is applied. One example of many is test R4-T1, which was able to reach a score of 8,864 for a_6 with fast cooling, while the result of a test with

same parametrization, but moderate cooling, was 10,374. A different situation arises for parameter combination R4-T3, which generated 12,042 crossings with fast cooling, while 10,511 were achieved with a slowly decreasing temperature curve. This contrast should be partly explainable by the significantly lower number of iterations, which can be completed. On average, Rebuild Neighbourhood was only able to examine half of the embeddings, which Random Walk was able to examine within 50 minutes. Consequently, discrepancies caused by fortunate modifications are more likely to occur. Executions of Rebuild Neighbourhood for PSE a_1 are particularly slow, for which only 2,094 instead of 78,407 modifications were examined. This is caused by the PSE's extraordinarily large edge-set, which considerably slows down the score recalculations.

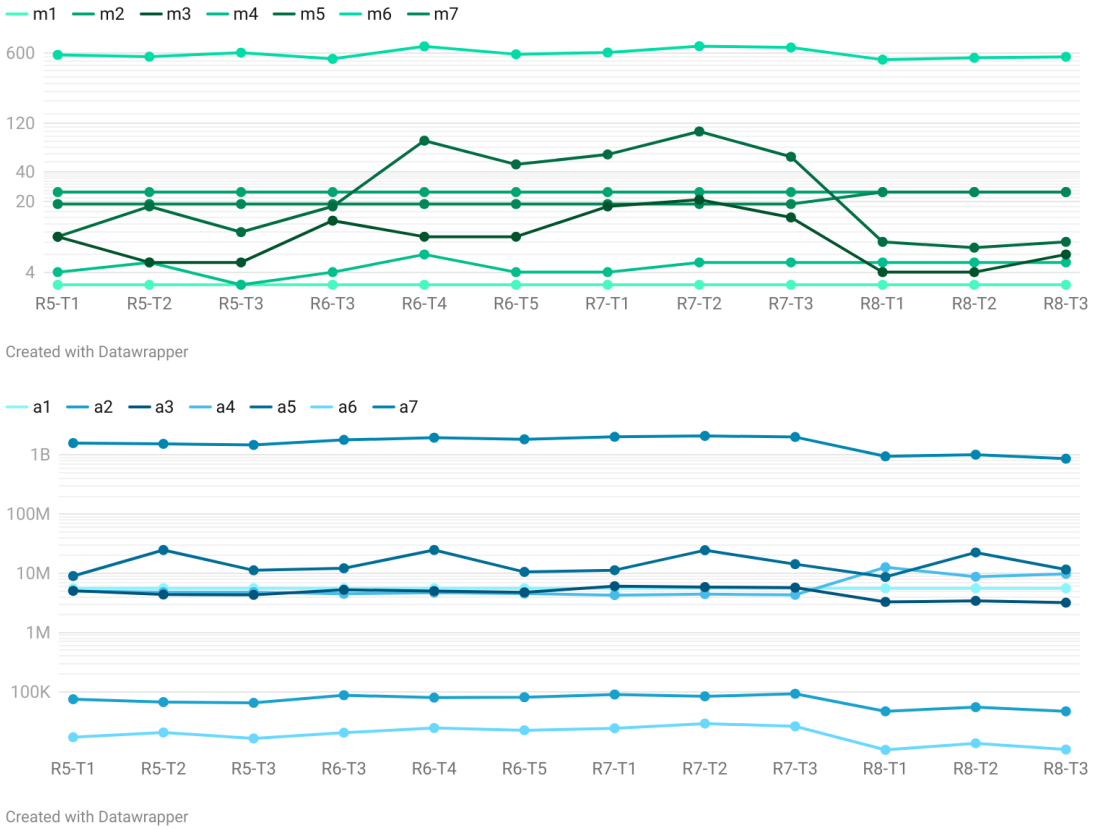


Fig. 5.10.: Scores obtained from SA with Rebuild Neighbourhood and a moderate exponential cooling. The parameter combinations are provided in Table 5.7. Previously, the drawings had been optimized with FR and combined greedy assignment.

Despite some negative conclusions, significant improvements in comparison to the Random Walk frequently occurred for the PSEs a_2, a_6 and a_7. For instance, if the solution of R4-T3 for the considered edge-case a_7 is compared to the best solution, that was found with the Random Walk, the crossings have already decreased from 1,069,290,000 to 841,614,000. This is the case, even though parameter s is not yet fine-tuned.

label	s	p	κ	c_{exp}	θ
N1-T1	60	0	0	0.85	2,000
N1-T2	30	0	0	0.85	2,000
N1-T3	15	0	0	0.85	2,000
N1-T4	10	0	0	0.85	2,000
N1-T5	5	0	0	0.85	2,000
N2-T1	60	1	0	0.85	2,000
N2-T2	30	1	0	0.85	2,000
N2-T3	15	1	0	0.85	2,000
N2-T4	10	1	0	0.85	2,000
N2-T5	5	1	0	0.85	2,000

label	s	p	κ	c_{exp}	θ
N3-T1	60	0	0	0.94	1
N3-T2	30	0	0	0.94	1
N3-T3	15	0	0	0.94	1
N3-T4	10	0	0	0.94	1
N3-T5	5	0	0	0.94	1
N4-T1	60	1	0	0.94	1
N4-T2	30	1	0	0.94	1
N4-T3	15	1	0	0.94	1
N4-T4	10	1	0	0.94	1
N4-T5	5	1	0	0.94	1

Tab. 5.8.: Parameters for which Rebuild Neighbourhood was tested in the second phase.

Motivated by these findings, the reset frequency for the SA will now be adjusted in order to achieve even better results for these drawings. Another series of tests was carried out for this purpose, for which the considered parameter combinations can be taken from Table 5.8. According to the previous considerations, $\kappa = 0$ is fixed and $p = 2$ is discarded. As for the previous series of tests, tabular overviews of all results are provided in the appendices B.9 and B.10. At this point, it is anticipated, that an adjustment of the reset frequency did achieve improvements for the Manuals, although the results are still poorer than those of the Random Walk. For this reason, it must be concluded, that the Random Walk is more appropriate for smaller drawings than this sub-strategy. Accordingly, conclusions in the subsequent paragraphs will be exclusively based on solutions of the Automatics. Figure 5.11 provides another useful visualization for this purpose and will be discussed.

At first glance, the diagram on the next page again appears to show improvements in favour of an increased reset frequency. With a few exceptions, the score's reduction accompanied by the increase of the frequency can be seen in particular for the PSEs a_3, a_4 and a_5. However, these are the drawings, which can be solved much better with the Random Walk. If only those drawings are considered, which are supported by the heuristic, two different patterns emerge. The first pattern, which corresponds to the main trend, can be recognized in the test series with the prefixes N1 and N4. Thus, the best results for a_6 and a_7 were found with the parametrizations N1-T5 and N4-T5. For PSE a_2, the same also applies for N4-T5, although no definitive tendency was recognizable in the series N1. The second pattern behaves oppositely and can be seen in the test series with the prefixes N2 and N3. For these, the best results for the considered drawings were mostly found with a large reset frequency of 60 seconds. Yet, exactly one drawing always falls out of the pattern, such that one minimum is again found according to the opposite pattern. Based on these findings, the interaction between rapid exponential cooling and a selection of vertices according to uniform probabilities is the only one, for which an adjustment of the reset frequency causes predictable effects for all tested PSEs. Repetitions of the tests did not lead to any other conclusions.

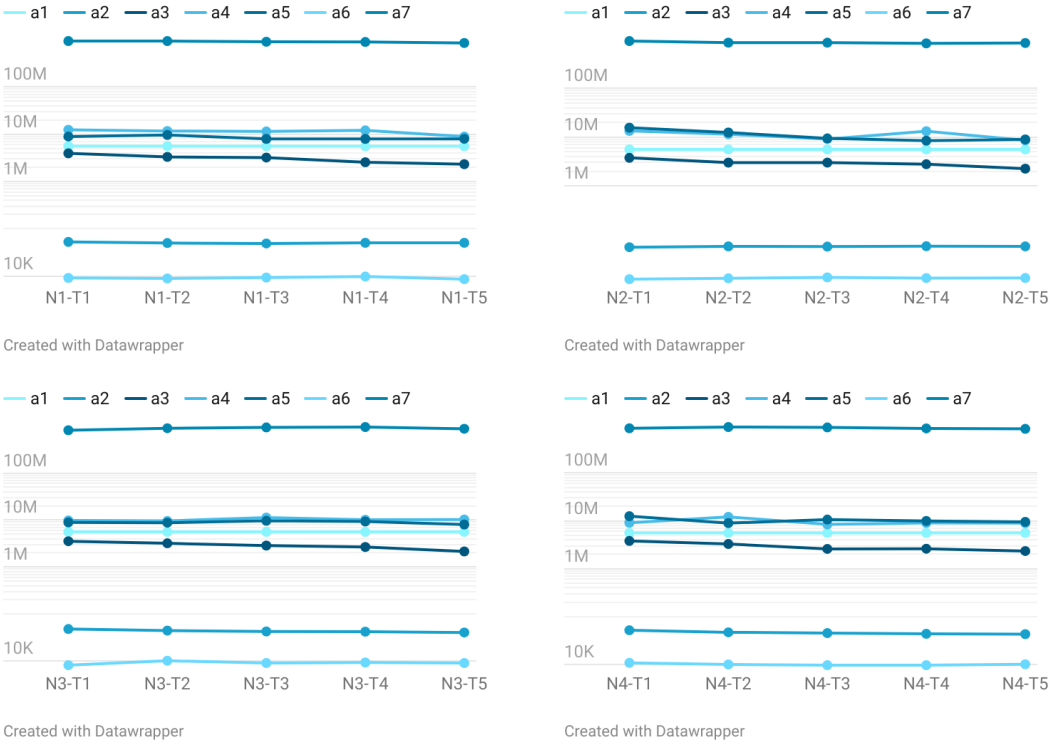


Fig. 5.11.: Scores obtained from SA with Rebuild Neighbourhood and a moderate exponential cooling. The parameter combinations are provided in Table 5.8. Previously, the drawings had been optimized with FR and combined greedy assignment. Grouped tests are ordered in descending order according to the utilized reset frequency.

Consequently, a selection of the most suitable parameter combination should be detached from a common frequency in this regard. Therefore, the highlighted parametrizations N1-T5, N2-T1, N3-T1 and N4-T5 are compared in Table 5.9 on the next page. As already recognized, the results show, that N1-T5 achieved the best solutions for the drawings a_3, a_4 and a_5. On the other hand, N3-T1 found the best solution for a_6 and a_7, while the solution for a_2 differs only marginally from the best solution found by N4-T5. For N3-T1 and N4-T5, it can be generally concluded, that their results are quite close to each other. Particularly encouraging is the fact, that these parameter combinations, almost without exception, beat the last GDC's solutions for a_2 and a_6 [BKK$^+$23b].

In summary, N1-T5 and N3-T1 are the tests, whose parameters are worthwhile for Rebuild Neighbourhood. The first uses fast cooling and a very high reset frequency, while the other establishes moderate cooling and a slow reset frequency. Even though the second parametrization achieved better results for supported drawings, N1-T5 was able to achieve better scores for more drawings. Similarly, the differences for supported ones are acceptably small. This has led to the decision, that N1-T5 with $s = 60$, $p = 0$, $\kappa = 0$ and $\theta = 2000$ will be used as the default setting for the toolbox.

However, the results indicate, that the alternative methodology should not be used exclusively during a GDC. Too many of the Automatics were therefore solved far better with the Random Walk. Nevertheless, for the drawings a_2, a_6 and a_7, which are supported by the heuristic, better results were consistently achieved. Rebuild Neighbourhood should therefore be seen as a complementary approach instead, which is expected to produce better results for drawings, whose number of crossings benefit from closely positioned neighbours – such as PSEs with linearly arranged point-sets.

label	a_1	a_2	a_3	a_4	a_5	a_6	a_7
N1-T5	5,580,218	50,365	2,316,571	8,932,152	7,976,241	8,555	850,587,000
N2-T1	5,580,218	55,160	3,746,059	13,376,200	15,639,734	12,180	933,318,000
N3-T1	5,580,218	47,768	3,535,908	9,761,329	8,886,244	8,137	815,661,000
N4-T5	5,580,218	42,730	2,334,993	8,947,071	9,524,353	9,994	834,693,000

Tab. 5.9.: Comparison of the scores obtained for the Automatics with the best parametrizations for SA with Rebuild Neighbourhood. The PSEs were previously optimized with FR and combined greedy assignment.

5.4. Performance issues

In a final step, the results of this work must be combined with those of the previous bachelor thesis. Therefore, the SA test cases W4-T5, W6-T5 and N1-T5 were carried out again on a machine, which is similar to Kutscheid's computer. More specifically, a PC with the Windows 11 operating system, an AMD Ryzen 7 5800X 8-core processor running at a base speed of 3.8 GHz and 32 GB RAM was chosen. Therefore, the next paragraphs solely refer to Table 5.10, which provides a better comparable overview of the new results. Additionally, it also contains the best solutions of the GDC 2023 and those, that Kutscheid achieved with his program. Only the Automatics are included in this comparison, as they were the only PSEs considered in the previous work.

label	a_1	a_2	a_3	a_4	a_5	a_6	a_7
W4-T5$_w$	4,804,362	76,350	549,214	2,191,078	943,235	24,942	1,044,753,000
W6-T5$_w$	4,797,740	56,029	544,827	2,073,971	2,095,189	12,172	1,045,719,000
N1-T5$_w$	5,580,218	43,614	2,074,219	9,550,345	7,928,623	8,591	872,394,000
N3-T1$_w$	5,580,218	41,658	2,764,094	8,997,012	7,928,623	8,928	782,643,000

label	a_1	a_2	a_3	a_4	a_5	a_6	a_7
KUT	4,948,788	34,584	246,254	1,592,751,925	142,354	4,443	768,501,000
GDC	4,654,687	51,178	3,994	-	14,288	18,943	209,532,000

Tab. 5.10.: Final results obtained with the combined approach of FR, combined greedy assignment and SA with Random Walk or Rebuild Neighbourhood. Utilized parameter combinations can be taken from 5.5 and 5.8. The solutions are compared to the best of the GDC 2023 [BKK+23b] and Kutscheid's combined implementation of Eades's spring embedder, slow greedy assignment and SA with Random Walk [Kut24].

In direct comparison with Kutscheid's results, it has to be admitted, that the toolbox's solutions are quite disappointing. Figure 5.12 demonstrates, that these differences are caused by serious runtime deficits, which occur during the SA process. This is the case, even if C++ is generally much faster than Java, which is the language employed by Kutscheid. Thus, it was only possible to examine more modifications than the old application for PSE a_1, which has the fewest vertices and points of the Automatics. As a consequence, a_1 is the only drawing for which continuously better results were found with the Random Walk, besides a_4. Especially disappointing are the solutions for a_5, for which only 13,242 modifications were explored during test W4-T5$_w$, instead of 2,666,958 examinations with the Java application – a difference, that is large enough to quickly offset the benefits achieved with the previous application of FR. The situation is very similar for drawing a_3, for which in W4-T5$_w$ only 264,300 compared to 1,019,270 iterations were achieved. Despite this situation, a five-hour test series, which can be found in Appendix B.11, demonstrates, that comparable results can be achieved, if the number of iterations would be similarly large. Hidden but serious bugs are hence not responsible for these discrepancies.

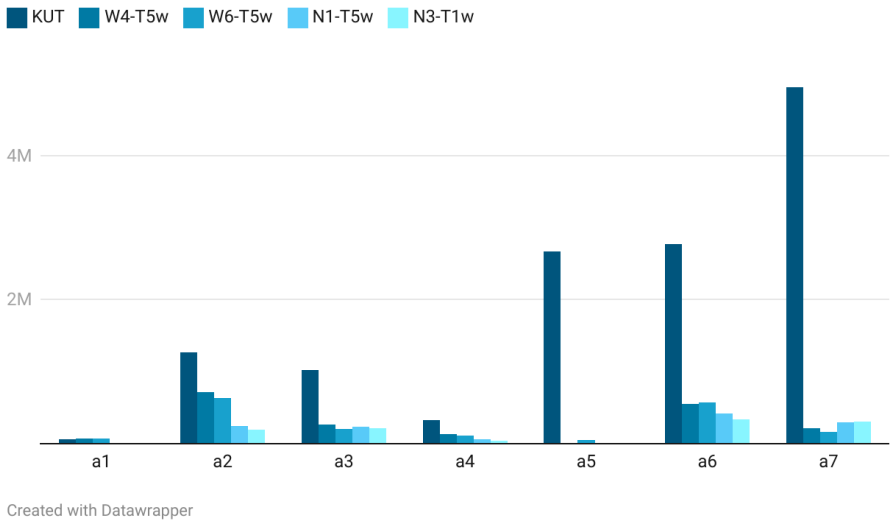


Fig. 5.12.: Quantity of PSEs examined via SA within a limit of 50 minutes. The information on Kutscheid's application was taken from a trial run with slightly modified code, so that the inner loop's executions were output. KUT, W4-T5$_w$ and W6-T5$_w$ applied the Random Walk, while N1-T5$_w$ and N3-T1$_w$ employed Rebuild Neighbourhood.

A critical section, which is mainly responsible for the differences, is located in the inner loop of the implemented SA algorithm. Due to the object-oriented decision to strictly separate all data-structures from algorithms and executors, a copy of the PSE under consideration must be created in each iteration. The modification is then applied to this working copy, which allows an easy rollback to the previous state on rejection. Once the reason for the discrepancies was found, an additional quick copy operator was introduced, which only copies necessary data and retains never-changing edge information as

well as adjacency-lists and -matrices. This increased the average number of iterations for the Automatics by around 85%. But the operator was still not an adequate solution to resolve the deficits. The reason for this is, that even so all vertices and points have to be duplicated, because they contain continuously tracked data for other partial functionalities. Note, that after the introduction of the quick copy operator, all tests were repeated in order to take its influence into account.

The last finding is further supported by the observation, that in particular the PSE a_1, which has the fewest points and vertices, is investigated the fastest. Thus, it can be stated, that the consequences of some object-oriented design decisions start to outweigh the C++ performance benefits once the PSEs point- or vertex-set exceeds 500 elements. In contrast to the duplication process, the score calculation depends in particular on the number of edges, which is remarkably large for drawing a_1. Therefore, the computationally complex score recalculation seems to be less critical in this regard. Kutscheid solved this pragmatically efficient, because he manages all data-structures and routines in a central thread object. In addition, he was not required to carry out as many on-the-fly calculations as are needed for the dynamic probability functions. Unfortunately, it was not possible to resolve this critical section prior to the GDC 2024, without damaging other essential features or discarding the entire project's architecture.

At least for the PSEs, that are supported by the neighbourhood heuristic, Rebuild Neighbourhood offers a successful compensation of these deficits. Only with this alternative refactoring method was it possible to continuously find better solutions than the best results of the GDC 2023 for the drawings a_2 and a_6. Consequently, the final toolbox has at least met the requirement of beating as many solutions of last year's GDC as Kutscheid's application did. Also noteworthy is drawing a_4, for which no valid solution was submitted in the 2023 competition [BKK+23b]. Although the Java application found a solution for this PSE, it is significantly worse than the newly found solution. In accordance with Table 5.3, this score can be attributed to the favourable solutions of FR and a dynamic application of both greedy assignments, rather then improvements caused by the SA. Thus, the results are disappointing, but not a complete failure.

label	a_1	a_2	a_3	a_4	a_5	a_6	a_7
W1-T1$_w$	5,117,621	84,981	3,535,410	2,595,468	2,357,947	30,697	1,913,322,000
W3-T1$_w$	5,170,320	77,322	3,573,002	2,633,078	2,220,068	23,848	1,769,469,000

Tab. 5.11.: Final results obtained with the combined approach of FR, greedy assignment and SA with Random Walk. Utilized parameter combinations can be taken from 5.5 and correspond to these considered by Kutscheid for the SA [Kut24].

Nevertheless, the results of the Random Walk are only competitive, because the increased reset frequency and dynamic probabilities for the selection of vertices positively affect the optimization process. Table 5.11 shows in this regard, that the solutions would be significantly worse without these adjustments. Therefore, the results of W1-T1$_w$ and W3-T1$_w$, whose parametrizations represent the starting point of this thesis, were continuously poorer than the scores obtained with W4-T5$_w$ and W6-T5$_w$.

6. Participation protocol

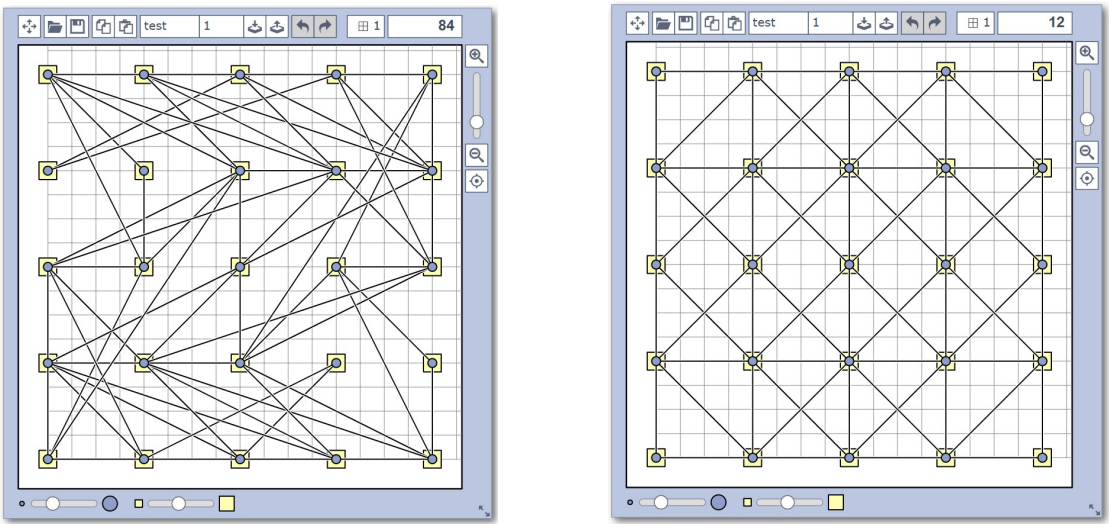
Alexander Kutscheid, Philipp Kindermann and myself participated in a team of three during the GDC 2024. In total, we competed against 7 other teams, that were categorized in the semi-automatic participant group. We managed to achieve the second place among these, which can therefore be considered as a general success [BKM+24b]. The key to this result was, that we did not rely on just one approach. Instead, we took advantage of the large number of CPU cores and executed multiple applications, algorithms and parameter combinations simultaneously.

The toolbox's individual target was to reliably generate average results with the SA and Random Walk for the Automatics, while satisfactory outcomes were expectable for the Manuals. In addition, applications of the neighbourhood heuristic should achieve particularly good results in a few cases. For this purpose, the SA was executed multiple times for each PSE, while a combination of the previously outlined parametrizations were employed. Beforehand, the drawings were prepared by FR's FDA and assigned with the combined greedy assignment. In order to enable a coordinated and quick initialization of all processes, bash scripts were prepared. At the same time, Kutscheid's program was employed to compensate the toolbox's deficits for larger PSEs. Furthermore, Kindermann implemented a specialized ILP solver, for which the time-limits were sufficient to calculate exact solutions for some of the Manuals. Both complementary programs provide interesting reference values, which will be discussed in the next few sections. For an overview of our team's results and the best overall solutions, refer to the Table 6.1. A statistical summary of the PSEs' key data is provided in Appendix B.12.

label	m_1^*	m_2^*	m_3^*	m_4^*	m_5^*	m_6^*	m_7^*
W1-T5	11	0	2	8	84	24	15
KIN	11	0	2	-	12	-	-
GDC	11	0	2	8	12	24	15

label	a_1^*	a_2^*	a_3^*	a_4^*	a_5^*	a_6^*	a_7^*	a_8^*
W6-T5	528,821	307,724	38,588	10,424	3,826,113	908,423	3,602,316	24,727
N1-T5	192,211	614,229	47,515	6,277	3,705,859	1,147,840	1,536,688	12,619
KUT	183,516	-	25,309	5,450	2,113,030	598,224	831,078	-
GDC	4,468	307,742	3,961	4	65,486	598,224	65,947	3,583

Tab. 6.1.: Results obtained during the GDC 2024. Scores labelled by KIN refer to Kindermann's ILP solver, while KUT indicates outcomes of Kutscheid's implementations. The overall best submitted solutions are indicated by GDC [BKM+24b]. The other parameter combinations can be found in the tables 5.5 and 5.8.



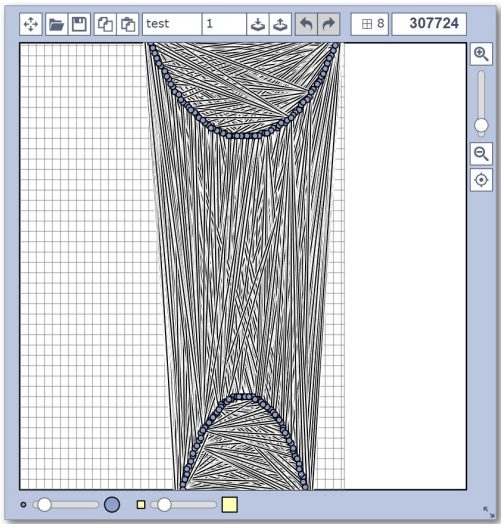
(a) Solution of the toolbox.

(b) Solution of the ILP solver.

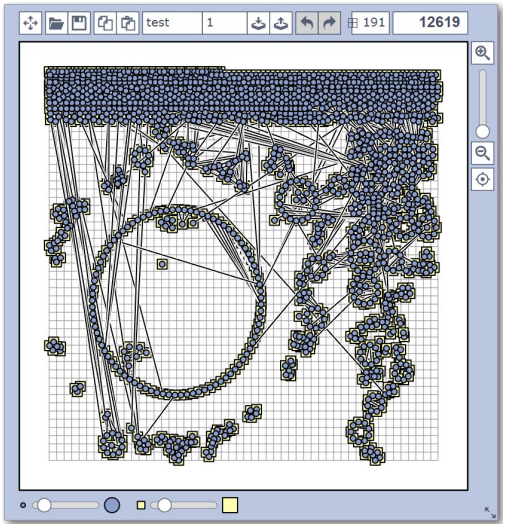
Fig. 6.1.: Comparison of outcomes for the drawing m_5^*.

When the results of the Manuals are considered, the toolbox was able to find the best solutions for almost all tasks. Only for the PSE m_5^* a considerably worse score was obtained, in comparison to the optimal solution found by the ILP solver. Among the Manuals, this one task is particularly special, because it has the most points and vertices. However, this drawing generally caused serious problems, because an ID was duplicated in the provided JSON file. Due to a lack of exception handling in the file import process, a segmentation error was thrown. Once the cause had been identified and fixed, the processes started as planned, but only with a time frame of 30 minutes left. In addition, only the parameter combination W1-T5 was executed for the Manuals, because the available CPU resources were already nearly exhausted.

Again, many more interesting observations can be made for the Automatics. Particularly positive is, that for many drawings the differences between Kutscheid's solutions and those of the toolbox are not as great as they were for last year's tasks. Thus, the improvements made during this thesis were able to fully compensate the runtime deficits for a_2^* and a_8^*. Therefore, the result for drawing a_2^* was found with the Random Walk and is even the best found among all participants. This result should be additionally attributable to a high number of explored modifications, because the instance has only 120 points and vertices. Accordingly, the size of this drawing is below the limit past which the performance problems dominate. On the other hand, the solution for a_8^* was generated using Rebuild Neighbourhood. Figure 6.2a illustrates, that the applied heuristic should have had a significantly positive effect on the score, especially in the upper and dense area. Note, that the corresponding entries for Kutscheid's results in Table 6.1 are empty, because we continuously compared our results and deleted worse solutions in order to maintain the overview during this hectic situation.



(a) Solution of the toolbox for a_2^*.

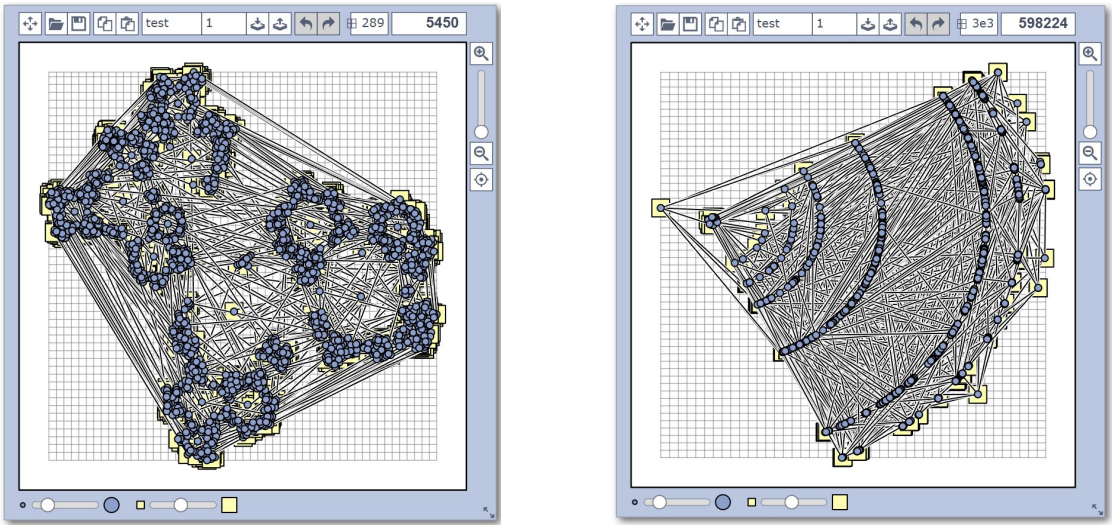


(b) Solution of the toolbox for a_8^*.

Fig. 6.2.: Comparison of outcomes for the drawings a_2^* and a_8^*. The first is even the best solution, that was found among all participants. The other result was beaten by the winners.

For the drawings a_1^*, a_3^* and a_4^*, the results within our team also differ only slightly. In the solutions for a_1^* as well as a_4^*, this comparability was mainly achieved through the application of Rebuild Neighborhood. Whereas for a_3^* on the other hand, the toolbox yielded it's best result with the parametrization W6-T5. Counter-tests with a uniform selection of vertices and rapid cooling according to the parameter combination W1-T5 only achieved a score of 70,621. Therefore the improvements, which this thesis has introduced for the SA, are particularly responsible for the situation, that the obtained result is below the average solution of the GDC 2024 [BKM+24b]. Nevertheless, also the results of Kutscheid's program, which achieved slightly better outcomes for the currently considered drawings, are far behind those of this year's winning team [BKM+24b].

The differences between scores obtained with Kutscheid's program and the toolbox are notably large for the drawings a_5^*, a_6^* and a_7^*. This is again caused by the performance issues, whose effects are especially significant when the Random Walk is applied. Also the neighbourhood heuristic was not able to generate satisfactory improvements for these PSEs. Particularly significant are the caused discrepancies for drawing a_7^*. Kutscheid was able to find a solution, which is less than a quarter as large as the solution of W6-T5. This significant difference can be explained with the drawing's characteristic, that it has 5000 points and vertices. Therefore, it is larger than any other instance examined so far and most comparable with the drawing a_5^* from the previous year. For this tested instance, a similar situation was identified in the previous chapter. Kutscheid's implementations even achieved the best solution of the entire competition for PSE a_6^*. The obtained score is 2,644,762 crossings ahead of this year's winner, who submitted a exceptionally poorly solution in comparison to his other results [BKM+24b].



(a) Solution of Kutscheid's program for a_4^*.

(b) Solution of Kutscheid's program for a_6^*.

Fig. 6.3.: Comparison of outcomes for the drawings a_4^* and a_6^*. The second is even the best, that was found among all participants. The other result was beaten by the winners.

To summarize, it is mainly attributable to Kutscheid's implementations, that we were able to perform so well at the Automatics. However, the implementation of the toolbox was not in vain, as it contributed to the success achieved in at least two of the large drawings. Moreover, both of the implementations show, that the combined approach, as discussed in this and the previous thesis, was able to achieve above average results, even in a competitive situation. Nevertheless, the final scores of this year's winning team, which had already won last year, are extraordinarily distant from our solutions for the tasks a_1^*, a_5^* and a_7^*. The toolbox would certainly be able to resolve a significant higher number of crossings with a better performance, although it is considered to be unlikely, that it would be possible to catch up completely.

7. Conclusion

Within a previous bachelor thesis, Alexander Kutscheid was already able to demonstrate, that a combined application of force-directed algorithms and Simulated Annealing is a promising approach to minimize crossings in point-set embeddings of graphs. However, a direct comparison based on the datasets from last year's Graph Drawing Contest led to the conclusion, that some of the solutions are nevertheless considerably worse than last year's winner. In order to participate in the Graph Drawing Contest of 2024, this thesis therefore adapted Kutscheid's approach and aimed to further improve it to potentially achieve even better results and to reduce the discrepancies for these drawings.

For this purpose, a direct comparison of the force-directed algorithms of Eades as well as Fruchterman and Reingold was first carried out. This comparison indicated, that solutions of the second algorithm yielded an average of 35% less crossings for large drawings, even after being embedded in the point-set. Consequently, these intermediate solutions provide a much better basis for the subsequently applied method. In this regard, a parametrization with $l = 100$ and a maximum number of iterations of $T = 124$ was particularly purposeful. In contrast, Eades' original variant performed better for small drawings, although these differences are negligible. The Simulated Annealing approach was further investigated in consideration of two different refactoring techniques. For the already proven Random Walk, systematically verifiable improvements were achieved through the introduction of a dynamically calculated probability, of which vertex should be processed next. Hereby, the probabilities of all vertices were continuously updated, based on the number of crossings, which are caused by the adjacent edges. Moreover, it was worthwhile to increase the reset frequency to $s = 5$ seconds. It was once again possible to verify, that a moderate cooling as controlled by $\theta = 1$ and $c_{exp} = 0.94$ tends to yield better results. In parallel, Rebuild Neighbourhood was introduced as a second refactoring methodology, which uses a strict heuristic to reposition a complete neighbourhood at points, that are close to one another. As a result, it was able to quickly generate better solutions for certain drawings, despite significantly fewer iterations were executed on average. At the same time, it has caused enormous disadvantages for other drawings, whose approximate optimum solutions are not supported by the heuristic. Consequently, this approach should be applied as a complementary algorithm and not be solely utilized during a competition.

As a by-product of this thesis, an easily expandable and maintainable C++ console application was developed, which was also employed during the GDC 2024. Nevertheless, it missed the target of this thesis in the sense, that it generates poorer solutions for more complex and larger point-set embeddings than Kutscheid's program did. These

deficits are attributed to some object-oriented decisions, which caused serious runtime deficits during the Simulated Annealing and were not resolved in time. As a result, the improvements did not lead to better solutions and instead only compensated the existing runtime problems. The applied suggestions for improvement are the only reason, that the results remained comparable at all. However, this does not change the validity of gathered findings, meaning that these can be transferred to other implementations without any problems. The discrepancies simultaneously demonstrate, that the success of Simulated Annealing is particularly dependent on the quantity of modifications being explored within the defined period of time. Hence, the score calculation remains a particular bottleneck, which could be solved more efficiently, for instance with the application of a sweep line approach.

Together with Kutscheid and our supervising professor Philipp Kindermann, our team was able to achieve the second place during the 2024's competition – a great success, especially for newcomers. Even though the toolbox as a whole failed to meet the expectations, it still contributed to this achievement. That above-average results were achieved in relation to the number of participants, is a final confirmation of the combined approach as a target-oriented resolution of the defined problem. However, large differences to the winners reveal, that even without the runtime deficits, the made improvements are not crucial enough to catch up. Should the toolbox be expanded in the future, for example if the Graph Drawing Contest addresses this problem again at some point, completely different strategies should be integrated. These can then also be specialized for individual edge-cases, because the combined approach is already capable of reliably achieving above-average solutions for arbitrary arranged point-sets.

Bibliography

[ayu24] ayushutwb: Line segment, 2024. https://www.geeksforgeeks.org/line-segment/, visited on 30.09.2024.

[Bar24] Ingo Bartling: Mathematik, physik, informatik: Wahrscheinlichkeitsverteilung und -dichte, 2024. https://www.ingo-bartling.de/mathe/klasse12/html/stochastik/verteilung/verteilungen.html, visited on 06.10.2024.

[BKK+23a] Sara Di Bartolomeo, Philipp Kindermann, Fabian Klute, Tamara Mchedlidze, Debajyoti Mondal, Wouter Meulemans, and Jules Wulms: Contest 2023: Call for participation, 2023. https://mozart.diei.unipg.it/gdcontest/2023/, visited on 26.09.2024.

[BKK+23b] Sara Di Bartolomeo, Philipp Kindermann, Fabian Klute, Tamara Mchedlidze, Debajyoti Mondal, Wouter Meulemans, and Jules Wulms: Graph drawing contest report. 2023. https://mozart.diei.unipg.it/gdcontest/assets/2023/2023-gd-contestreport.pdf, visited on 26.09.2024.

[BKM+24a] Sara Di Bartolomeo, Fabian Klute, Debajyoti Mondal, Wouter Meulemans, and Jules Wulms: Contest 2024: Call for participation, 2024. https://mozart.diei.unipg.it/gdcontest/2024/, visited on 26.09.2024.

[BKM+24b] Sara Di Bartolomeo, Fabian Klute, Debajyoti Mondal, Wouter Meulemans, and Jules Wulms: Graph drawing contest report. 2024. https://mozart.diei.unipg.it/gdcontest/assets/2024/2024-gd-contestreport.pdf, visited on 26.09.2024.

[BKMW24] Sara Di Bartolomeo, Fabian Klute, Debajyoti Mondal, and Jules Wulms: What is graph drawing, 2024. https://mozart.diei.unipg.it/gdcontest/, visited on 29.09.2024.

[dBea08] Mark de Berg and Otfried Cheong et al.: *Computational Geometry: Algorithms and Applications*. Springer-Verlag Berlin Heidelberg, 3rd edition, 2008, ISBN 978-3-540-77973-5. https://link.springer.com/book/10.1007/978-3-540-77974-2, visited on 26.09.2024.

[Die00] Reinhard Diestel: *Graph Theory: Second Edition*, chapter 1: The Basics, pages 1–6. Graduate Texts in Mathematics. Springer-Verlag

New York, electronic edition, 2000, ISBN 0-387-98976-5. https://diestel-graph-theory.com/index.html, visited on 26.09.2024.

[Ead84] Peter Eades: A heuristic for graph drawing. *Congressus numerantium*, 42(11):149–160, 1984. https://www.cs.ubc.ca/~will/536E/papers/Eades1984.pdf, visited on 26.09.2024.

[ekt24] ektamaini: Dot and cross products on vectors, 2024. https://www.geeksforgeeks.org/dot-and-cross-products-on-vectors/, visited on 30.09.2024.

[FR91] Thomas M. J. Fruchterman and Edward M. Reingold: Graph drawing by force-directed placement. *Software: Practice and experience*, 21(11):1129–1164, 1991. https://doi.org/10.1002/spe.4380211102, visited on 26.09.2024.

[Gur24] Refactoring Guru: The catalog of design patterns, 2024. https://refactoring.guru/design-patterns/catalog, visited on 26.09.2024.

[HJJ03] Darrall Henderson, Sheldon H. Jacobson, and Alan W. Johnson: The theory and practice of simulated annealing. In Fred Glover and Gary A. Kochenberger (editors): *Handbook of metaheuristics*, chapter 10, pages 287–319. Springer, 2003. https://dx.doi.org/10.1007/0-306-48056-5_10, visited on 26.09.2024.

[ING24] INGInious: Competitive programming: Geometry - point orientation. Université catholique de Louvain, 2024. https://inginious.org/course/competitive-programming/geometry-orient, visited on 26.09.2024.

[Kin21a] Philipp Kindermann: Force-directed drawings (1/3): Algorithmic framework | visualization of graphs - lecture 3. YouTube, 2021. https://youtu.be/WWm-g2nLHds?si=QeEx1tXbQwqNZ7Dy, visited on 26.09.2024.

[Kin21b] Philipp Kindermann: Force-directed drawings (2/3): Eades spring embedder | visualization of graphs - lecture 3. YouTube, 2021. https://youtu.be/gJiSvGbH0CA?si=3uFNbJAcoTxKVmsZ, visited on 26.09.2024.

[Kin21c] Philipp Kindermann: Force-directed drawings (3/3): Fruchterman & reingold | visualization of graphs - lecture 3. YouTube, 2021. https://youtu.be/JAe7Oscsp98?si=byxGhA3ddf8cTBJw, visited on 26.09.2024.

[Kin24] Philipp Kindermann: Graph drawing tool, 2024. https://graphdrawingcontest.appspot.com/tool.jsp, visited on 26.09.2024.

[Kut24] Alexander Kutscheid: Minimizing crossings in point-set embeddings, 2024.

[Mar09] Robert C. Martin: *Clean Code - a Handbook of Agile Software Craftsmanship*, chapter 10: Classes, pages 135–170. Pearson Addison-Wesley, 2009, ISBN 978-0-13-235088-4. https://amzn.eu/d/hhKDSR7, visited on 26.09.2024.

[Mat24] Mathepedia: Injektive Abbildungen, 2024. https://mathepedia.de/Injektion.html, visited on 26.09.2024.

[Mis23] Durganshu Mishra: Compiler optimizations: Boosting code performance without doing much. Nerd For Tech, 2023. https://medium.com/nerd-for-tech/search?q=boosting+performance+without+doing+much, visited on 26.09.2024.

[Mol24] Bastian Molkenthin: Notes about the (two-dimensional) perp dot product, 2024. https://www.sunshine2k.de/articles/Notes_PerpDotProduct_R2.pdf, visited on 30.09.2024.

[MP21] Matteo Mazziotta and Adriano Pareto: Data normalization for aggregating time series: The constrained min-max method. *Rivista Italiana di Economia Demografia e Statistica*, 75(4):144–151, 2021. http://www.sieds.it/listing/RePEc/journl/20217549_01441RV_Mazziotta.pdf.

[MRR+53] Nicholas Metropolis, Rosenbluth Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H., and Edward Teller: Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953. https://bayes.wustl.edu/Manual/EquationOfState.pdf, visited on 26.09.2024.

[NA98] Yaghout Nourani and Bjarne Andresen: A comparison of simulated annealing cooling strategies. *Journal of Physics A: Mathematical and General*, 31:8373–8385, 1998. https://www.fys.ku.dk/~andresen/BAhome/ownpapers/perm-annealSched.pdf, visited on 26.09.2024.

[NMR12] Rahnuma Islam Nishat, Debajyoti Mondal, and Md. Saidur Rahman: Point-set embeddings of plane 3-trees. *Computational Geometry*, 45(3):88–98, 2012, ISSN 0925-7721. https://www.sciencedirect.com/science/article/pii/S0925772111000757, visited on 26.09.2024.

[Pat13] Maurizio Patrignani: Planarity testing and embedding. In Roberto Tamassia (editor): *Handbook of Graph Drawing and Visualization*, pages 1–42. CRC press, 2013. https://api.pageplace.de/preview/DT0400.9781420010268_A37879766/preview-9781420010268_A37879766.pdf, visited on 26.09.2024, (Preview link).

[PP13] Pranav Patel and Chirag Patel: Various graphs and their applications in real world. *International Journal of Engineering Research and Technology (IJERT)*, 2(12), 2013. https://www.ijert.org/various-graphs-and-their-applications-in-real-world-2, visited on 26.09.2024.

[Sch89] P. C. Schuur: *Classification of acceptance criteria for the simulated annealing algorithm*, volume 8929 of *Memorandum COSOR*. Technische Univer-

siteit Eindhoven, 1989. https://pure.tue.nl/ws/portalfiles/portal/2116564/338267.pdf, visited on 26.09.2024.

[Smi24] Julius O. Smith: *Physical Audio Signal Processing*. University of Stanford, Online Book, 2024. https://ccrma.stanford.edu/~jos/pasp/Linear_Interpolation.html, visited on 26.09.2024.

[Sza12] Tibor Szabó: Lectureship discrete mathematics ii: Jordan curves and planar graphs. Freie Universität Berlin, 2012. http://discretemath.imp.fu-berlin.de/DMII-2011-12/planar-graphs.pdf, visited on 26.09.2024.

[VBG+00] Luca Vismara, Giuseppe Di Battista, Ashim Garg, Giuseppe Liotta, Roberto Tamassia, and Francesco Vargiu: Experimental studies on graph drawing algorithms. *Software: Practice and experience*, 30(11):1235–1284, 2000. https://onlinelibrary.wiley.com/doi/abs/10.1002/1097-024X%28200009%2930%3A11%3C1235%3A%3AAID-SPE339%3E3.0.CO%3B2-B, visited on 26.09.2024.

[Wol07] Alexander Wolff: Drawing subway maps: A survey. *Informatik-Forschung und Entwicklung*, 22:23–44, 2007. https://link.springer.com/content/pdf/10.1007/s00450-007-0036-y.pdf, visited on 26.09.2024.

A. Discarded approaches

This appendix contains a brief overview of smaller approaches and suggestions for improvements, which were not pursued further. They were therefore neither discussed in detail nor employed during the GDC 2024.

A.1. Queued assignment

As discussed in the previous chapters, the faster greedy assignment usually leads to poorer results. The reason for this is, that vertices are potentially assigned to points, although others are closer to these. The queued assignment invests additional runtime to order the vertices in a queue according to their degree, prior to the assignment loop. Consequently, vertices with higher degree are then prioritized. However, Table A.1 indicates, that this had no impact on the assignment process. Another reason why the approach can be discarded is, that a similar situation no longer needs to be considered, if the slower greedy assignment is applied.

label	$S2_f$	$S2_q$	F4-$T2_f$	F4-$T2_q$
m_1	18	18	17	17
m_2	36	36	72	72
m_3	770	770	901	901
m_4	32	32	32	32
m_5	1,550	1,550	779	779
m_6	1,542	1,542	1,223	1,223
m_7	306	306	308	308
a_1	8,715,135	8,715,135	5,714,446	5,714,446
a_2	384,484	384,484	391,046	391,046
a_3	52,982,782	52,982,782	22,313,393	22,313,393
a_4	19,574,479	19,574,479	12,827,198	12,827,198
a_5	50,526,914	50,526,914	30,712,274	30,712,274
a_6	586,318	586,318	45,399	45,399
a_7	8,813,910,000	8,813,910,000	3,057,663,000	3,057,663,000

Tab. A.1.: Evaluations of fast greedy assignment applied on intermediate results of the two investigated FDAs. Label S2 describes outcomes of Eades' spring embedder parametrised with $attr = 10$, $repl = 20000$, $l = 100$, $T = 64$ and $c = 0.992$. Test case F4-T2 is defined in Table 5.2. The letter "q" marks assignments, for which the vertices were distributed in descending order according to their degree.

A.2. Hybrid refactoring

During the tests of Rebuild Neighbourhood, it was observed, that the heuristic causes significant improvements at the beginning. It's positive impact then quickly converged for some of the drawings. In consequence, a large amount of the time passed without purposeful improvements being achieved. Out of this emerged the idea to optimize the neighbourhood of randomly selected vertices, until the positive effect stagnates. As soon as this point is reached, the modification method switches to the Random Walk. In the toolset, this behaviour is achieved with a check before each inner loop, to determine when the current minimum result was overwritten for the last time. If the last improvement occurred more than two minutes ago, a switch is triggered and all subsequent loops use the next methodology instead.

label	H1	H2	N1-T5	N3-T5	W4-T5	W6-T5
m_1	3	3	3	3	3	3
m_2	25	25	25	25	25	25
m_3	0	0	1	1	0	0
m_4	4	4	5	5	4	4
m_5	4	4	8	8	4	4
m_6	405	407	537	531	404	411
m_7	19	19	25	25	19	19
a_1	4,780,548	4,799,467	5,580,218	5,580,218	4,820,138	4,797,353
a_2	59,585	44,420	50,365	57,118	73,641	59,505
a_3	450,477	497,089	2,316,571	2,255,877	585,639	618,957
a_4	2,154,387	2,072,427	8,932,152	8,581,900	2,301,006	2,174,089
a_5	1,405,182	1,017,950	7,976,241	8,949,537	1,266,689	2,665,358
a_6	13,498	9,010	8,555	12,877	26,877	12,305
a_7	845,925,000	804,738,000	850,587,000	850,392,000	1,130,448,000	1,069,290,000

Tab. A.2.: Overview of scores obtained by SA with the three refactoring techniques. The hybrid approach H1 initially employs Rebuild Neighbourhood with the parameter combination N1-T5 and then Random Walk with SW4-T5. H2 on the other hand, applied N3-T5 and SW6-T5. The parameter combinations for the other approaches can be found in tables 5.5 and 5.8.

According to experiences made during this series of tests, the success of this methodology is closely related to the number of iterations, which can be executed within the 50 minute time constraint. An initial investigation of the hybrid approach falsely indicated, that it is a compromise of the other two approaches. Only after the introduction of a quick copy command, which was introduced shortly before the GDC 2024, it was possible to achieve the improvements shown in Table A.2. Unfortunately, there was not enough time left after this discovery to investigate the approach in detail.

B. Results

On the subsequent pages, the results of all discussed series of tests are provided in a complete and tabular form. The corresponding parameter combinations and characteristics of the PSEs to be optimized are outlined in the chapter 5.

label	B1-T1	B1-T2	B2-T1	B2-T2
m_1	3	3	3	3
m_2	25	25	25	28
m_3	195	195	195	195
m_4	6	8	9	9
m_5	1584	1584	1584	1548
m_6	609	609	612	609
m_7	19	19	19	19
a_1	8,347,246	8,333,723	8,359,898	8,345,796
a_2	1,875,180	1,872,537	1,875,180	1,873,025
a_3	74,101,653	73,995,158	74,110,172	74,001,748
a_4	42,260,737	41,995,123	42,261,138	42,242,209
a_5	15,165	15,165	15,165	15,165
a_6	10,390	10,322	10,397	10,349
a_7	8,981,418,000	8,981,418,000	8,981,418,000	8,981,418,000

Tab. B.1.: Final scores obtained from the testing series of the brute-force algorithm. Results with the suffix T2 refer to runs, that applied tracked scoring, while the slow variant was used for the others. In addition, datasets with the prefix B1 are those, which have been compiled with the optimization flag `-Ofast`.

label	B1-T1	B1-T2	B2-T1	B2-T2
m_1	218,055,443	86,011,573	36,496,518	19,266,737
m_2	133,062,306	62,217,943	25,849,851	14,925,424
m_3	97,196,879	78,050,201	20,108,035	17,500,706
m_4	122,407,136	56,621,151	19,324,972	11,449,945
m_5	63,096,080	66,891,899	9,791,127	12,626,024
m_6	5,949,523	10,772,355	946,252	2,265,756
m_7	3,628,800	3,628,800	3,628,800	3,628,800
a_1	1,301	24,204	292	5,742
a_2	8,712	405,193	1,663	99,151
a_3	4,206	94,774	1,055	94,774
a_4	1,354	169,171	277	37,806
a_5	6,602	1,936,682	1,064	421,837
a_6	17,311	1,871,456	2,921	418,579
a_7	19,905	1,832,791	2,819	361,164

Tab. B.2.: Number of PSEs, that were explored during the testing series of the brute-force algorithm. Results with the suffix T2 refer to runs, that applied tracked scoring, while the slow variant was used for the others. In addition, datasets with the prefix B1 are those, which have been compiled with the optimization flag `-Ofast`.

label	m_1	m_2	m_3	m_4	m_5	m_6	m_7
F1-T1	17	37	2	40	13	665	36
F1-T2	14	28	0	23	15	663	17
F1-T3	10	12	0	17	9	607	13
F1-T4	13	28	2	30	16	872	24
F1-T5	21	37	5	35	20	985	27
F1-T6	21	22	2	29	17	837	31
F1-T7	9	3	0	18	9	1,696	21
F1-T8	7	3	0	18	9	1,694	17
F1-T9	7	3	0	16	9	1,694	17
F2-T1	12	20	3	38	21	754	27
F2-T2	6	24	0	24	17	561	25
F2-T3	15	13	0	17	10	542	17
F2-T4	27	20	4	43	24	939	41
F2-T5	17	58	4	45	27	883	22
F2-T6	21	27	4	39	18	906	49
F2-T7	11	13	0	13	14	649	20
F2-T8	11	13	0	13	14	681	18
F2-T9	11	13	0	13	14	681	18
F3-T1	16	16	0	17	9	516	21
F3-T2	13	16	0	16	8	490	16
F3-T3	10	16	0	16	10	572	16
F3-T4	14	42	4	29	18	515	22
F3-T5	13	18	6	37	9	534	22
F3-T6	19	60	6	30	16	509	22
F3-T7	7	12	0	14	9	523	15
F3-T8	7	13	0	14	9	530	15
F3-T9	7	13	0	14	9	530	15
F4-T1	17	18	0	20	18	539	18
F4-T2	13	30	0	17	11	523	14
F4-T3	10	14	0	14	11	520	16
F4-T4	18	47	4	58	27	496	20
F4-T5	15	14	2	54	14	523	31
F4-T6	18	29	4	50	15	513	37
F4-T7	11	13	0	16	9	583	25
F4-T8	11	13	0	16	9	578	25
F4-T9	11	13	0	16	9	578	25

Tab. B.3.: Overview of all results obtained by the application of FR on the Manuals. Corresponding parameter combinations can be found in Table 5.2. Note, that these are not valid PSEs, because the vertices are not yet assigned to the point-set. Therefore, these solutions would be penalized with $\text{Score}(\Gamma) = \infty$ during the GDC.

label	a_1	a_2	a_3	a_4	a_5	a_6	a_7
F1-T1	6,495,141	7	2	6,235,663	6,212	883,500	776,596
F1-T2	6,076,945	18	1	6,161,706	110	870,383	762,001
F1-T3	5,777,254	66	3	5,981,872	75	851,601	730,576
F1-T4	6,409,357	7	3	6,224,380	5,029	881,268	872,087
F1-T5	5,913,721	63	0	6,098,284	375	861,834	811,766
F1-T6	5,906,244	98	2	5,406,007	66	821,516	738,839
F1-T7	7,144,545	0	0	6,338,622	1,076,227	896,551	830,658
F1-T8	7,143,042	0	0	6,338,129	1,071,865	896,533	830,529
F1-T9	7,143,022	0	0	6,338,140	1,071,844	880,449	830,519
F2-T1	5,968,099	104	10	5,118,780	5	812,007	771,641
F2-T2	5,378,716	148	6	3,121,554	3	736,974	703,778
F2-T3	5,230,094	183	1	2,232,062	0	623,437	619,045
F2-T4	6,168,925	128	77	4,755,829	3	800,443	739,207
F2-T5	6,343,927	170	1	2,529,045	432	687,504	660,995
F2-T6	6,033,201	220	2	2,373,039	0	428,268	493,055
F2-T7	6,405,525	7	0	6,221,206	461,412	880,520	882,728
F2-T8	6,402,733	7	0	6,220,180	880,892	880,449	880,892
F2-T9	6,402,680	7	0	6,220,750	451,412	880,449	880,884
F3-T1	6,232,313	283	1,816	2,478,384	137	14,080	49,327
F3-T2	5,887,370	284	55	2,306,075	1	6,824	6,840
F3-T3	5,595,245	281	1	2,213,757	78	6,089	1,768
F3-T4	6,441,406	280	1,518	2,505,602	100	12,653	34,808
F3-T5	6,284,302	319	3	2,409,364	1	7,670	4,964
F3-T6	6,095,586	324	3	2,381,795	0	6,281	1,651
F3-T7	5,520,859	170	863	3,094,466	1,827,857	677,708	652,129
F3-T8	5,503,136	170	861	3,056,108	1,820,475	676,400	651,002
F3-T9	5,502,684	170	861	3,081,992	1,820,437	676,401	651,009
F4-T1	6,164,462	317	3,869	2,400,807	4	7,047	20,057
F4-T2	5,891,725	326	107	2,330,801	120	5,978	4,580
F4-T3	5,520,554	318	1	2,199,384	406	5,725	1,485
F4-T4	6,380,905	319	2,276	2,478,988	4	7,884	17,091
F4-T5	6,174,709	346	10	2,380,329	3	6,958	3,958
F4-T6	6,058,049	322	1	2,309,480	3	6,577	1,538
F4-T7	5,464,994	220	7,761	3,090,394	1,894,140	396,046	516,004
F4-T8	5,448,564	220	7,745	3,049,664	1,888,963	392,891	512,900
F4-T9	5,446,782	220	7,745	3,038,798	1,888,932	392,871	512,881

Tab. B.4.: Overview of all results obtained by the application of FR on the Automatics. Corresponding parameter combinations can be found in Table 5.2. Note, that these are not valid PSEs, because the vertices are not yet assigned to the point-set. Therefore, these solutions would be penalized with $\text{Score}(\Gamma) = \infty$ during the GDC.

label	m_1	m_2	m_3	m_4	m_5	m_6	m_7
W1-T1	3	25	0	4	4	424	19
W1-T2	3	25	0	4	4	423	19
W1-T3	3	25	0	4	4	420	19
W1-T4	3	25	0	4	4	423	19
W1-T5	3	25	2	4	4	414	19
W2-T1	3	25	0	4	4	413	19
W2-T2	3	25	0	4	4	409	19
W2-T3	3	25	0	4	4	424	19
W2-T4	3	25	0	4	4	410	19
W2-T5	3	25	0	4	4	405	19
W3-T1	3	25	0	4	4	404	19
W3-T2	3	25	0	4	4	412	19
W3-T3	3	25	0	4	4	414	19
W3-T4	3	25	0	4	4	413	19
W3-T5	3	25	0	4	4	404	19
W4-T1	3	25	0	4	4	418	19
W4-T2	3	25	3	4	4	414	19
W4-T3	3	25	2	4	4	413	19
W4-T4	3	25	0	4	4	408	19
W4-T5	3	25	0	4	4	395	19
W5-T1	3	25	0	4	4	397	19
W5-T2	3	25	0	4	4	416	19
W5-T3	3	25	0	4	4	409	19
W5-T4	3	25	0	4	4	410	19
W5-T5	3	25	0	4	4	395	19
W6-T1	3	25	0	4	4	410	19
W6-T2	3	25	0	4	4	414	19
W6-T3	3	25	0	4	4	395	19
W6-T4	3	25	0	4	4	398	19
W6-T5	3	25	0	4	4	411	19

Tab. B.5.: Overview of all results obtained by the subsequent application of FR, combined greedy assignment and SA with Random Walk on the Manuals. Corresponding parameter combinations can be found in Table 5.5.

label	a_1	a_2	a_3	a_4	a_5	a_6	a_7
W1-T1	5,172,807	77,651	3,080,041	2,703,244	3,258,850	30,895	1,847,403,000
W1-T2	5,062,314	88,820	2,606,198	2,546,197	2,500,877	29,815	1,667,580,000
W1-T3	4,912,075	83,970	1,886,875	2,483,523	1,644,596	29,808	1,459,053,000
W1-T4	4,888,671	81,401	1,726,384	2,497,326	2,131,469	28,095	1,335,684,000
W1-T5	4,806,314	78,381	1,046,040	2,347,978	2,504,333	28,536	1,111,602,000
W2-T1	5,126,688	71,703	1,439,090	2,475,541	12,428,144	28,668	1,848,804,000
W2-T2	5,029,479	73,788	1,134,502	2,424,630	8,294,244	25,383	1,650,297,000
W2-T3	4,896,639	74,270	773,653	2,398,554	12,338,836	25,518	1,408,833,000
W2-T4	4,913,466	70,613	727,803	2,289,117	11,409,576	24,716	1,355,439,000
W2-T5	4,803,402	71,011	508,475	2,312,199	8,384,307	22,891	1,236,426,000
W3-T1	5,214,650	72,896	1,526,927	2,498,524	2,448,462	29,427	1,798,350,000
W3-T2	5,002,246	75,976	1,380,183	2,356,798	2,170,844	26,696	1,528,233,000
W3-T3	4,911,097	70,252	1,005,083	2,393,593	943,721	25,938	1,383,171,000
W3-T4	4,832,490	75,777	841,707	2,446,302	2,407,535	24,915	1,278,303,000
W3-T5	4,820,138	73,641	585,639	2,301,006	1,266,689	26,877	1,130,448,000
W4-T1	5,158,996	74,340	3,320,006	2,627,462	2,637,721	23,686	1,874,286,000
W4-T2	5,012,041	75,077	2,594,993	2,556,852	2,555,627	22,397	1,603,158,000
W4-T3	4,938,838	67,654	1,905,190	2,391,475	1,775,502	19,775	1,415,367,000
W4-T4	4,874,421	66,081	1,673,536	2,343,720	2,377,791	17,294	1,336,038,000
W4-T5	4,784,304	64,101	1,088,192	2,182,424	1,803,207	14,977	1,138,653,000
W5-T1	5,144,402	68,138	1,367,129	2,497,653	10,558,935	20,990	1,773,354,000
W5-T2	5,058,711	66,021	1,067,829	2,361,363	10,292,787	18,354	1,646,163,000
W5-T3	4,890,371	63,550	789,701	2,337,435	11,059,570	16,923	1,484,496,000
W5-T4	4,859,666	61,503	691,038	2,239,593	8,688,001	15,645	1,410,699,000
W5-T5	4,785,846	57,724	451,385	2,193,252	10,647,439	13,927	1,237,773,000
W6-T1	5,110,796	70,178	1,773,391	2,447,674	2,653,158	19,462	1,713,723,000
W6-T2	5,098,009	65,769	1,570,073	2,429,862	2,592,841	18,403	1,533,672,000
W6-T3	4,939,750	63,083	940,552	2,475,466	1,990,092	16,259	1,338,873,000
W6-T4	4,862,542	62,234	794,844	2,282,299	2,425,847	14,550	1,268,058,000
W6-T5	4,797,353	59,505	618,957	2,174,089	2,665,358	12,305	1,069,290,000

Tab. B.6.: Overview of all results obtained by the subsequent application of FR, combined greedy assignment and SA with Random Walk on the Automatics. Corresponding parameter combinations can be found in Table 5.5.

label	m_1	m_2	m_3	m_4	m_5	m_6	m_7
R1-T1	3	25	7	5	29	584	19
R1-T2	3	25	7	5	18	606	19
R1-T3	3	25	12	5	23	529	19
R2-T1	3	25	15	4	56	626	19
R2-T2	3	25	11	5	55	666	19
R2-T3	3	25	12	5	42	678	19
R3-T1	3	25	16	5	77	703	19
R3-T2	3	25	22	5	84	703	19
R3-T3	3	25	19	4	63	703	19
R4-T1	3	25	5	5	4	529	25
R4-T2	3	25	4	5	11	553	25
R4-T3	3	25	5	5	6	529	25
R5-T1	3	25	9	4	9	575	19
R5-T2	3	25	5	5	18	553	19
R5-T3	3	25	5	3	10	606	19
R6-T3	3	25	13	4	18	525	19
R6-T4	3	25	9	6	81	700	19
R6-T5	3	25	9	4	47	584	19
R7-T1	3	25	18	4	59	607	19
R7-T2	3	25	21	5	100	703	19
R7-T3	3	25	14	5	56	681	19
R8-T1	3	25	4	5	8	516	25
R8-T2	3	25	4	5	7	538	25
R8-T3	3	25	6	5	8	551	25

Tab. B.7.: Overview of all results obtained by the subsequent application of FR, combined greedy assignment and SA with Rebuild Neighbourhood on the Manuals. Corresponding parameter combinations can be found in Table 5.7.

label	a_1	a_2	a_3	a_4	a_5	a_6	a_7
R1-T1	5,580,218	77,727	4,782,566	4,930,850	7,833,875	22,396	1,575,435,000
R1-T2	5,580,218	69,747	4,382,588	4,766,963	19,377,652	21,980	1,653,105,000
R1-T3	5,580,218	75,110	4,194,343	4,887,521	10,298,516	22,422	1,527,381,000
R2-T1	5,580,218	85,900	5,402,835	4,516,945	8,085,395	27,600	1,860,666,000
R2-T2	5,580,218	84,513	5,176,708	4,613,244	24,448,820	25,186	1,934,136,000
R2-T3	5,580,218	77,362	5,195,626	4,517,913	12,708,064	24,016	1,816,590,000
R3-T1	5,580,218	95,571	6,042,376	4,304,769	10,027,820	28,362	1,894,401,000
R3-T2	5,580,218	85,811	5,758,488	4,385,295	27,113,022	27,901	2,094,018,000
R3-T3	5,580,218	90,751	5,713,615	4,381,330	10,874,806	27,567	1,919,727,000
R4-T1	5,580,218	55,430	3,270,699	9,762,364	9,594,412	8,864	909,348,000
R4-T2	5,580,218	62,692	3,663,265	9,676,788	23,387,537	14,266	947,040,000
R4-T3	5,580,218	57,949	3,387,276	11,127,719	11,199,391	12,042	841,614,000
R5-T1	5,580,218	74,281	5,067,149	5,028,606	9,004,997	16,924	1,589,025,000
R5-T2	5,580,218	66,613	4,371,710	4,801,672	24,808,501	20,395	1,544,142,000
R5-T3	5,580,218	64,493	4,333,543	4,809,067	11,227,686	16,191	1,478,835,000
R6-T3	5,580,218	86,834	5,255,118	4,508,764	12,166,993	20,196	1,798,200,000
R6-T4	5,580,218	79,091	5,010,350	4,703,024	24,757,084	24,350	1,959,042,000
R6-T5	5,580,218	80,456	4,768,564	4,528,540	10,542,129	22,261	1,840,299,000
R7-T1	5,580,218	89,249	6,063,798	4,274,557	11,305,042	24,036	2,026,668,000
R7-T2	5,580,218	83,240	5,852,625	4,438,175	24,551,871	28,887	2,105,181,000
R7-T3	5,580,218	91,963	5,734,808	4,325,291	14,265,329	25,876	2,011,371,000
R8-T1	5,580,218	46,483	3,280,593	12,669,909	8,700,038	10,374	948,189,000
R8-T2	5,580,218	54,678	3,424,587	8,765,722	22,510,067	13,399	1,014,384,000
R8-T3	5,580,218	46,414	3,184,480	9,794,398	11,588,505	10,511	864,255,000

Tab. B.8.: Overview of all results obtained by the subsequent application of FR, combined greedy assignment and SA with Rebuild Neighbourhood on the Automatics. Corresponding parameter combinations can be found in Table 5.7.

label	m_1	m_2	m_3	m_4	m_5	m_6	m_7
N1-T1	3	25	4	5	5	507	25
N1-T2	3	25	4	5	8	521	25
N1-T3	3	25	3	5	7	525	25
N1-T4	3	25	2	5	8	526	25
N1-T5	3	25	1	5	8	537	25
N2-T1	3	25	5	5	10	525	25
N2-T2	3	25	1	5	4	553	25
N2-T3	3	25	3	5	6	526	25
N2-T4	3	25	7	5	7	554	25
N2-T5	3	25	1	5	8	531	25
N3-T1	3	25	5	5	8	535	25
N3-T2	3	25	4	5	6	477	25
N3-T3	3	25	1	5	8	509	25
N3-T4	3	25	0	5	10	522	25
N3-T5	3	25	0	8	8	504	25
N4-T1	3	25	3	5	4	545	25
N4-T2	3	25	3	5	8	521	25
N4-T3	3	25	3	5	5	533	25
N4-T4	3	25	0	5	8	518	25
N4-T5	3	25	0	5	5	513	25

Tab. B.9.: Overview of all results obtained by the subsequent application of FR, combined greedy assignment and SA with Rebuild Neighbourhood on the Automatics. Corresponding parameter combinations can be found in Table 5.8.

label	a_1	a_2	a_3	a_4	a_5	a_6	a_7
N1-T1	5,580,218	52,444	3,937,478	12,421,037	8,933,349	9,111	934,881,000
N1-T2	5,580,218	49,724	3,306,282	11,676,363	9,641,923	8,836	934,710,000
N1-T3	5,580,218	48,736	3,207,165	11,399,607	7,928,799	9,245	901,686,000
N1-T4	5,580,218	50,244	2,545,926	12,008,576	7,933,398	9,713	892,785,000
N1-T5	5,580,218	50,365	2,316,571	8,932,152	7,976,241	8,555	850,587,000
N2-T1	5,580,218	55,160	3,746,059	13,376,200	15,639,734	12,180	933,318,000
N2-T2	5,580,218	57,392	2,990,498	11,447,282	12,431,598	12,655	863,244,000
N2-T3	5,580,218	56,618	2,994,825	9,139,810	9,391,888	13,278	862,584,000
N2-T4	5,580,218	57,590	2,767,650	13,216,885	8,454,379	12,849	838,248,000
N2-T5	5,580,218	57,118	2,255,877	8,581,900	8,949,537	12,877	850,392,000
N3-T1	5,580,218	47,768	3,535,908	9,761,329	8,886,244	8,137	815,661,000
N3-T2	5,580,218	44,331	3,188,877	9,564,036	8,766,787	10,112	898,350,000
N3-T3	5,580,218	42,334	2,864,204	11,303,226	9,589,550	9,033	933,243,000
N3-T4	5,580,218	42,137	2,663,343	10,127,792	9,279,811	9,271	952,287,000
N3-T5	5,580,218	40,352	2,139,183	10,232,073	8,000,032	9,045	876,186,000
N4-T1	5,580,218	51,364	3,787,771	9,121,007	12,508,591	10,702	858,042,000
N4-T2	5,580,218	46,654	3,268,446	12,059,623	8,947,940	9,926	912,204,000
N4-T3	5,580,218	44,879	2,588,860	8,426,780	10,579,797	9,594	893,301,000
N4-T4	5,580,218	43,420	2,608,863	9,008,231	9,922,983	9,603	855,291,000
N4-T5	5,580,218	42,730	2,334,993	8,947,071	9,524,353	9,994	834,693,000

Tab. B.10.: Overview of all results obtained by the subsequent application of FR, combined greedy assignment and SA with Rebuild Neighbourhood on the Automatics. Corresponding parameter combinations can be found in Table 5.8.

label	a_1	a_2	a_3	a_4	a_5	a_6	a_7
W1-T5$_l$	4,573,185	76,577	382,111	1,793,158	645,627	25,961	610,578,000
W3-T5$_l$	4,526,327	45,569	324,808	1,651,746	890,916	7,621	575,949,000
W4-T5$_l$	4,557,427	70,250	245,785	1,713,403	627,733	20,415	573,843,000
W6-T5$_l$	4,546,839	43,412	212,165	1,566,672	424,459	6,884	537,843,000
N1-T5$_l$	5,580,218	43,614	2,074,219	9,550,345	7,928,623	8,591	872,394,000
N3-T1$_l$	5,580,218	41,658	2,764,094	8,997,012	7,928,623	8,928	782,643,000

Tab. B.11.: Results obtained by the subsequent application of FR, combined greedy assignment and SA with Rebuild Neighbourhood on the Automatics. The time available for these tests was extended to 300 minutes. Corresponding parameter combinations can be found in the tables 5.5 and 5.8.

ID	$\|V\|$	$\|E\|$	$\|P\|$	size	avg deg(v)	max deg(v)	arrangement
m_1^*	8	16	8	20x20	4	7	square
m_2^*	20	19	20	10x10	1.9	10	grid
m_3^*	12	24	12	48x48	4	6	circle
m_4^*	20	40	20	20x20	4	5	symmetric
m_5^*	25	64	25	20x20	5.12	8	grid
m_6^*	20	46	20	64x194	4.6	8	symmetric
m_7^*	20	54	20	39x20	5.4	6	arbitrary
a_1^*	1500	4494	1500	1000x1000	5.992	6	grid
a_2^*	160	2486	160	699x543	31.075	44	symmetric
a_3^*	1200	3500	1200	2275x2382	5.834	16	arbitrary
a_4^*	2000	2003	2000	15017x14330	2.003	16	structures
a_5^*	900	3454	900	161833x191373	5	13	grid
a_6^*	2000	5000	2000	5555x5516	3.453	34	layered
a_7^*	5000	10000	5000	198634x208135	4	11	structures
a_8^*	1589	2742	1589	9920x9031	3.451	34	structures

Tab. B.12.: An overview of the PSEs to be optimised from the GDC 2024.

Erklärung

Hiermit versichere ich die vorliegende Abschlussarbeit selbstständig verfasst zu haben, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben, und die Arbeit bisher oder gleichzeitig keiner anderen Prüfungsbehörde unter Erlangung eines akademischen Grades vorgelegt zu haben.

Trier, den 10. October 2024

. . . . . .
Jan-Niclas Loosen