Timo Poranen

Approximation Algorithms for Some Topological Invariants of Graphs

ACADEMIC DISSERTATION

To be presented, with the permission of the Faculty of Information Sciences of the University of Tampere, for public discussion in the B1096 Auditorium of the University on October 22nd, 2004, at 12 noon.

DEPARTMENT OF COMPUTER SCIENCES
UNIVERSITY OF TAMPERE

A-2004-7
TAMPERE 2004
Supervisors: Professor Erkki Mäkinen
Department of Computer Sciences
University of Tampere

Professor Jyrki Nummenmaa
Department of Computer Sciences
University of Tampere

Opponent: Professor Ville Leppänen
Department of Information Technology
University of Turku

Reviewers: Professor Robert Cimikowski
Department of Computer Science
Montana State University

Professor Tapio Elomaa
Institute of Software Systems
Tampere University of Technology

Department of Computer Sciences
FIN-33014 UNIVERSITY OF TAMPERE
Finland

ISBN 951-44-6098-7
ISSN 1459-6903

Tampereen yliopistopaino Oy
Tampere 2004
Abstract

Topological graph theory studies the embeddings of graphs on various surfaces and the properties of these embeddings. Topological properties of graphs have many applications in the fields of graph drawing, user interface design, circuit design, and resource location optimization.

This thesis studies approximation methods for the following $NP$-complete optimization problems: maximum planar subgraph, maximum outerplanar subgraph, and thickness of a graph. We also study the outerthickness problem which complexity status is not known. We compare the solution quality and computation times of a simulated annealing algorithm and several algorithms based on triangular cactus heuristic, including other heuristics taken from the literature, to approximately solve these problems.

Triangular cactus heuristic was the first non-trivial approximation algorithm for the maximum planar and outerplanar subgraph problems. We give a modified version of the triangular cactus heuristic that has at least equal performance ratio and asymptotic running time as the linear time version of the original algorithm. A large number of experiments show that the new algorithm achieves better approximations than the earlier methods.

We give two new theoretical results for the thickness and outerthickness of a graph. We prove a new upper bound for the thickness of complete tripartite graphs, and lower and upper bounds in the terms of the minimum and maximum degree of a graph for outerthickness. Also, the simulated annealing algorithm given in this work solves partially an open problem related to the thickness of the complete bipartite graphs. Our experiments show that the general formula also holds for some previously unknown cases.
Preface

The simulated annealing algorithm given in Chapter 4 for approximating thickness will appear in *Information Sciences* [108]. Experiments reported in Chapter 8 are mainly taken from the article, but also some new approximations for large complete graphs are given in this work. The algorithm is also applied for outerthickness in Chapter 8. In the same article a new upper bound for the complete tripartite graphs was given. This result is discussed in Chapter 3 and experimented in Chapter 8.

The simulated annealing algorithm for the maximum planar subgraph problem appeared in *International Journal of Computer Mathematics* [109]. The algorithm is studied in Chapter 4 and experimental results are given in Chapter 6. The same algorithm is applied for approximating the maximum outerplanar subgraph problem in Chapter 7.

The new lower and upper bounds for outerthickness as the function of the minimum and maximum degree of a graph are given in Chapter 3. This result is based on an unpublished manuscript written with Erkki Mäkinen. The verification that the general formula holds for the previously unknown cases of the thickness of complete bipartite graphs, discussed in Section 8.7, is given in the same manuscript.

The genetic algorithm for determining the thickness of a graph, introduced in Chapter 4 and experimented in Chapter 8, is originally published together with Erkki Mäkinen and Petri Vuorenmaa [88].
Acknowledgements

I’m very grateful to my supervisors, professors Erkki Mäkinen and Jyrki Nummenmaa, for these years. Especially I thank Erkki’s supervising method, he has always given me constructive feedback of my writings. Jyrki was the man who made me interested in the graph theory.

I warmly thank all my colleagues and the old and new staff personnel of the Department of Computer Sciences for making the working environment enjoyable.

I’m grateful to Isto Aho, Jorma Laurikkala, Jouni Mykkänen, Timo Tossavainen, and Petri Vuorenmaa for scientific assistance and to Arto Viitanen for giving me many technical advises.

This work was funded by the Tampere Graduate School in Information Science and Engineering (TISE) and the University of Tampere, by awarding me a research stipend that gave me possibility to finish this thesis. Also the Academy of Finland (project 51528) has supported this work.

The greatest joy in my life during these years have been my wife Pirjo, and our children Niko, Toni, and Jenni. Thank you for your encouragement and support for this work!

Tampere, April 2004
Timo Poranen
Contents

1 Introduction 1

2 Preliminaries 5
  2.1 Graphs ................................................. 5
  2.2 Algorithms and computational complexity ................. 11
  2.3 Testing planarity and outerplanarity .................. 15
  2.4 Combinatorial optimization .......................... 18
    2.4.1 Local search ...................................... 18
    2.4.2 Simulated annealing .............................. 19
    2.4.3 Genetic algorithms ............................... 20
    2.4.4 Branch-and-bound ................................. 21
  2.5 Experimental algorithmics ............................ 22

3 Topological Invariants 25
  3.1 Maximum planar subgraph ............................. 25
  3.2 Maximum outerplanar subgraph ....................... 27
  3.3 Thickness ............................................. 28
  3.4 Outerthickness ....................................... 31
  3.5 Other invariants ..................................... 34
    3.5.1 Arboricity ....................................... 34
    3.5.2 Book thickness .................................. 34
    3.5.3 Geometric thickness .............................. 35

4 Approximating invariants 37
  4.1 Algorithms for MPS .................................. 37
  4.2 Algorithms for MOPS ................................ 42
  4.3 Algorithms for thickness ............................. 44
  4.4 Algorithms for outerthickness ....................... 48
CONTENTS

B  Statistics for MOPS  
   B.1  CA and CA1 statistics ........................................ 129
   B.2  GCA and GCA1 statistics .................................... 130
   B.3  $SA_E$ and $SA_{CA1}$ statistics ............................ 132

C  Statistics for thickness  
   C.1  ST and CA statistics ........................................... 135
   C.2  CA1 and CA2 statistics ....................................... 137
   C.3  GCA statistics .................................................. 139
   C.4  GCA1 and GCA2 statistics .................................... 140

D  Statistics for outerthickness  
   D.1  GRE, GCA, GCA1, and $SA$ statistics ....................... 141

E  Decomposition of $K_{19,29}$ into six planar subgraphs  

Bibliography  

ix
Chapter 1

Introduction

Topological graph theory studies the embeddings of graphs on various surfaces and the properties of these embeddings [55]. Topological properties of graphs have many applications in the fields of graph drawing (information visualization) [38], VLSI circuit design [3], and resource location optimization [97].

An undirected, simple graph $G$ (shortly a graph), denoted by $G = (V, E)$, consists of a finite vertex set $V$ and a set of undirected edges $E \subseteq \{(u, v) \mid u, v \in V, u \neq v\}$. A graph is an abstract mathematical structure that models objects and connections between them. A graph can model any system involving binary relations. Given a graph $G = (V, E)$, a graph $G' = (V', E')$ is called a subgraph of $G$ if $V' \subseteq V$ and $E' \subseteq \{(u, v) \mid u \in V', v \in V' \text{ and } (u, v) \in E\}$.

A graph is usually visualized by representing each vertex through a point in the plane, and by representing each edge through a curve in the plane, connecting the points corresponding to the end vertices of the edge. Such a representation is called a drawing of the graph if no two vertices are represented by the same point, if the curve representing an edge does not include any other points representing a vertex (except its endpoints), and if two distinct edges have at most one common point. A drawing of a graph is plane if no two distinct edges intersect apart from their endpoints. A graph is planar if it admits a plane drawing, otherwise the graph is nonplanar. Such a drawing of a planar graph is called a plane embedding.

Two graphs are isomorphic if there exists a one-to-one correspondence between their vertex sets which preserves adjacency. An invariant of a graph $G$ is a number associated with a property of $G$ which has the same value for any graph isomorphic to $G$. For example, a simple topological invariant of a graph is the number of vertices.

If a graph $G' = (V, E')$ is a planar subgraph of $G$ such that every graph $G''$ obtained from $G'$ by adding an edge from $E \setminus E'$ is nonplanar, then $G'$ is
called a maximal planar subgraph of $G$. Let $G' = (V, E')$ be a maximal planar subgraph of $G$. If there is no planar subgraph $G'' = (V, E'')$ of $G$ with $|E''| > |E'|$, then $G'$ is a maximum planar subgraph. A maximal planar subgraph is maximal in the sense that adding edges is not possible and the maximum planar subgraph is maximal with respect to the cardinality of its edge set. A graph is outerplanar if it admits a plane drawing where all its vertices surround the same region and no two distinct edges intersect, otherwise the graph is non-outerplanar. Maximal outerplanar subgraph and maximum outerplanar subgraph are defined in the similar way as the maximal and maximum planar subgraphs.

The thickness of a graph is the minimum number of planar subgraphs into which the graph can be decomposed. Similarly, the outerthickness of a graph is the minimum number of outerplanar subgraphs into which the graph can be decomposed.

Determining the maximum planar subgraph (MPS), maximum outerplanar subgraph (MOPS), and thickness of a given graph are known to be $NP$-complete (that is, there is no polynomial time algorithm for them, unless $P = NP$ [51]). If a problem is $NP$-complete, we are unlikely to solve it exactly, but it might be possible to find near-optimal solutions in polynomial time using an approximation algorithm. The complexity status of determining the outerthickness of a graph is not known.

This work studies approximation methods and approximability results for determining MPS, MOPS, thickness, and outerthickness of a graph. We introduce several new algorithms that can be used to approximate these invariants. Also, a comprehensive experimental study of the new algorithms against each other and against earlier heuristics algorithms from the literature is given. One of the new algorithms solves partially an open problem of unknown cases of the thickness of complete bipartite graphs. This work also contains two new theoretical results for thickness and outerthickness.

In Chapter 2, we give standard definitions and operations for graphs and introduce theoretical background for approximation algorithms, planarity and outerplanarity tests. We also introduce commonly used optimization methods, including guidelines to design experimental comparison of algorithms.

Chapter 3 recalls known theoretical results for the studied invariants. In this chapter, two new theoretical results for the thickness and outerthickness of a graph are given. The first result gives a new upper bound for the thickness of complete tripartite graphs, and the second result gives lower and upper bounds for outerthickness in the terms of minimum and maximum degree of a graph. Further, an upper bound for outerthickness as a function of the number of
edges is conjectured.

Approximation algorithms for MPS, MOPS, thickness and outerthickness are presented in Chapter 4. We describe a new optimization method based on simulated annealing scheme to approximate MPS and MOPS. In addition, another simulated annealing algorithm for thickness and outerthickness is represented. Experiments given in Chapters 6, 7, and 8 show that the simulated annealing approach outperforms clearly the earlier heuristic algorithms.

The simulated annealing is quite time consuming approximation method and therefore Chapter 5 concentrates on methods that run in linear time for bounded degree graphs. We start with an algorithm based on triangular cactus heuristic, given by Călinescu et al. [26]. The algorithm was the first with non-trivial performance ratio for MPS and MOPS. Based on the triangular cactus approach, we give two new algorithms that have at least equal performance as the original algorithm and still run in linear time for bounded degree graphs. The first algorithm approximates MPS and MOPS, the second algorithm gives approximations only for MPS. As shown in the experimental part of this work (Chapters 6, 7 and 8), these algorithms achieve clearly better approximations than the original triangular cactus heuristic. The exact performance ratio of the new algorithms is left open, but we state a conjecture that is supported by the experiments.

Chapters 6 and 7 describe an experimental comparison of the studied algorithms for MPS and MOPS, respectively. The test graph set [112] that is common with a few exceptions for both problems, contains 50 graphs.

The total number of the compared algorithms is 12 for MPS and 7 for MOPS. The experiments show that the new algorithms based on the triangular cactus heuristic (given in Chapter 5) and the simulated annealing algorithm (given in Chapter 4) produce the best known approximations for MPS and MOPS. No earlier experimental results for MOPS are presented in the literature.

Chapter 8 describes the experimental comparison of the algorithms for thickness and outerthickness. Thickness comparison contains approximation results for 12 algorithms and outerthickness comparison for 7 algorithms. The test graph set contains in total 63 graphs. Again, the experiments show that the simulated annealing is a very efficient method to approximate the thickness and outerthickness of a graph. The efficiency of simulated annealing is shown by solving some previously unknown cases of complete bipartite graphs. These results verify that the general formula holds for the solved cases. No earlier experimental results for outerthickness are presented in the literature.

The last chapter summarizes the results and lists the open problems that
CHAPTER 1. INTRODUCTION

have arisen from the theoretical and experimental parts of this work.
Chapter 2

Preliminaries

In this chapter we introduce basic definitions related to graphs and topological graph theory. We also define some important concepts related to algorithms. After that, we give methods to recognize whether a graph is planar or outerplanar, and represent commonly used optimization methods. Finally, we describe guidelines to design experimental comparison of algorithms.

If some terms or notations related to graphs are unfamiliar, we recommend books written by Harary [60], Swamy and Thulasiraman [118] and Berge [14]. Moreover, Gross and Tucker [55] deal with topological graph theory and Liebers [84] surveys topics related to planarity and planarization of graphs. Standard algorithm textbooks are Cormen et al. [36] and Knuth [77], while Even [44] concentrates only on graph algorithms. All algorithms in this work are given in similar format as in the book of Cormen et al. [36]. Garey and Johnson [51] give a formal approach for $NP$-complete problems and an introduction to optimization heuristics can be found in the book by Reeves [110].

2.1 Graphs

A graph is an abstract mathematical structure that models objects and connections between them. A graph $G$, denoted by $G = (V, E)$, consists of a finite set $V$ of vertices and a set $E \subseteq \{(u, v) \mid u \in V, v \in V, u \neq v\}$ of edges. If $E = \emptyset$, the graph is empty. In the literature vertices are sometimes called points or nodes and edges are lines or arcs. Given an edge $e = (u, v)$, the vertices $u$ and $v$ are called the end vertices of $e$.

If the vertex pairs of the edge set are ordered, the graph is called directed and otherwise undirected. In undirected graphs, $(u, v) = (v, u)$ for all edges, and in directed, $(u, v) \neq (v, u)$. We mainly consider undirected graphs. Therefore, if not stated otherwise, the graphs under consideration are undirected.
Two vertices $u$ and $v$ in $V$ are said to be adjacent, if $(u, v) \in E$. The number of vertices $|V|$ and edges $|E|$ of a graph are often denoted by $n$ and $m$, respectively. The complement $\overline{G}$ of a graph $G$ has the same vertex set as $G$, but two vertices are adjacent in $\overline{G}$ if and only if they are not adjacent in $G$.

A path in a graph is a sequence of vertices $v_1, v_2, v_3, \ldots, v_k$, where vertices $v_i$ and $v_{i+1}$, $1 \leq i < k$ and $k \geq 2$, are adjacent. A path is closed, if $v_1 = v_k$ and open otherwise. A path is simple, if all vertices of the path are distinct, except possibly the end vertices. If a path is simple and closed, it is a cycle. The length of a simple and open path is $k - 1$, where $k$ is the number of vertices in it. The length of a cycle is the number of vertices in it. A cycle of length $n$ is denoted by $C_n$. A cycle of length three, $C_3$, is a triangle. A triangle with vertices $v_1, v_2,$ and $v_3$ is denoted by $(v_1, v_2, v_3)$. A simple open path with $n$ vertices is denoted by $P_n$.

Given a graph $G = (V, E)$, a graph $G' = (V', E')$ is called a subgraph of $G$ if $V' \subseteq V$ and $E' \subseteq \{(u, v) \mid u \in V', v \in V' \text{ and } (u, v) \in E\}$. If not stated otherwise, we assume that $V' = V$ when we consider subgraphs. Let $G = (V, E)$ be a graph and let $E' \subseteq E$. Graph $G' = (V', E')$ is called an edge induced subgraph of $G$ if $V' = \{v \mid (u, v) \in E'\}$. If $V' \subseteq V$, then graph $G' = (V', E')$, where $E' = \{(u, v) \in E \mid u \in V' \text{ and } v \in V'\}$, is a vertex induced subgraph of $G$. For a vertex induced subgraph $G' = (V \setminus v, E')$ of $G = (V, E)$, we say that $G'$ is obtained from $G$ by deleting or removing $v$ and that $G$ is obtained from $G'$ by inserting $v$.

If $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are subgraphs of $G = (V, E)$, then subgraph $G' = (V_1 \cup V_2, E_1 \cup E_2)$ of $G$ is called the union of $G_1$ and $G_2$, and denoted by $G_1 \cup G_2$. If $V_1 = V_2$ and $E_2 = \{(u, v)\}$, then we say that edge $(u, v)$ is inserted or added to $G_1$.

A graph is connected, if there is a path between every pair of vertices, otherwise the graph is disconnected. A maximal connected subgraph of a graph is a component. A cut vertex of a graph is a vertex whose removal increases the number of components. If at least $k$ vertices have to be deleted from the graph to make it disconnected, we say that the graph is $k$-connected. A $k$-connected component is a maximal $k$-connected subgraph. A 2-connected component is also called biconnected.

A graph is a tree if it does not contain any cycles. A tree is a spanning tree, if there is a unique path between every pair of vertices. For a graph with $n$ vertices, a spanning tree contains $n - 1$ edges.

The degree of a vertex $v$ is the number of adjacent vertices of $v$. The minimum degree of a graph $G$ is denoted by $\delta(G)$ and the maximum degree is denoted by $\Delta(G)$, respectively. If $r = \delta(G) = \Delta(G)$, $G$ is said to be regular of
degree $r$.

A graph $G = (V, E)$ is bipartite, if the vertex set $V$ can be partitioned into two subsets $V_1$ and $V_2$ such that for every edge $(u, v)$ it holds that $u \in V_1$ and $v \in V_2$. The join of graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, denoted by $G_1 + G_2$, is a graph with vertex set $V_1 \cup V_2$ and edge set $E_1 \cup E_2 \cup \{(u, v) \mid u \in V_1$ and $v \in V_2\}$.

For $n \geq 1$, the complete graph, denoted by $K_n$, consists of $n$ vertices with all possible edges, that is, with $n(n-1)/2$ edges. The complete bipartite graph, denoted by $K_{v_1,v_2}$, is the join of $\overline{K}_{v_1}$ and $\overline{K}_{v_2}$ and it has $v_1 + v_2$ vertices and $v_1v_2$ edges. The complete $k$-partite graph, denoted by $K_{v_1,v_2,\ldots,v_k}$, is defined as the iterated join $\overline{K}_{v_1} + \overline{K}_{v_2} + \cdots + \overline{K}_{v_k}$. The complete $k$-partite graph has $\sum v_i$ vertices and $\sum_{i<j} v_iv_j$ edges.

The product of graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, denoted by $G_1 \times G_2$, is a graph with vertex set $V_1 \times V_2$ and edge set defined as follows: let $u = (u_1, u_2)$ and $v = (v_1, v_2)$ belong to $V_1 \times V_2$. Vertices $u$ and $v$ are adjacent if $u_1 = v_1$ and $u_2$ is adjacent with $v_2$ in $G_2$ or $u_2 = v_2$ and $u_1$ is adjacent with $v_1$ in $G_1$. A $m \times n$ grid graph, where $m, n \geq 1$ is product of simple open paths $P_m$ and $P_n$ and it contains $mn$ vertices and $2mn - n - m$ edges.

The hypercube of order $n$, denoted by $Q_n$, is defined recursively by $Q_1 = K_2$ and $Q_n = K_2 \times Q_{n-1}$. Hypercubes are bipartite and $Q_n$ has $2^n$ vertices and $n2^{n-1}$ edges. Hypercubes of order 1, 2 and 3 determine a line, a square, and a cube, respectively.

A graph with $n$ vertices is Hamiltonian, if there is a cycle $C_n$ in $G$. If for two graphs $G = (V, E)$ and $G' = (V, E')$ it holds that $|E| < |E'|$, we say that $G$ is sparser than $G'$ and that $G'$ is denser than $G$. When the graphs are compared, it is often said that $G$ is a sparse graph and $G'$ is a dense graph.

Two graphs are isomorphic, if between their vertex sets there exists a one-to-one correspondence which preserves adjacency. Let $e = (u, v)$ be an edge of a graph. Edge $e$ is subdivided when we replace $e$ by adding a new vertex $w$ and two new edges $(u, w)$ and $(w, v)$ to the graph. Two graphs are homeomorphic if both can be obtained from the same graph by a sequence of subdivisions.

For a graph $G = (V, E)$ and an edge $(u, v) \in E$, the graph $G'$ obtained from $G$ by deleting $(u, v)$, identifying $u$ with $v$, and removing all edges $f \in \{(u, x) \mid x \in V, x \neq v, (u, x) \in E\}$ and $(v, x) \in E$, is said to have been obtained from $G$ by contracting the edge $(u, v)$. A graph obtained from a subgraph of $G$ by any number of edge contractions is said to be a minor of $G$.

A graph is usually visualized by representing each vertex through a point in the plane, and by representing each edge through a curve in the plane, connecting the adjacent points. Such a representation is called a drawing of
the graph if no two vertices are represented by the same point, if the curve representing an edge does not include any other points representing a vertex (except its endpoints), and if two distinct edges have at most one common point. A drawing of a graph is plane if no two distinct edges intersect. A graph is planar if it admits a plane drawing. Such a drawing of a planar graph is called a plane embedding. Regions in the plane that are defined by the drawing of a planar graph are called faces. Edges that form a region, are called the boundary edges of the region, or shortly a boundary. The unbounded region is an outerface or exterior face and all the other regions are innerfaces. Edges and vertices belonging to the outerface, are called exterior edges and exterior vertices, respectively. The other edges and vertices are interior edges and interior vertices. If every face of a plane embedding of a graph is a triangle, the graph is called triangulated. A chord is an interior edge in which the end vertices are exterior.

A combinatorial embedding for a planar graph is a cyclic list of adjacent edges for each vertex such that if the edges are drawn in this order, a plane embedding is obtained. A graph may have many different combinatorial embeddings and some different combinatorial embeddings may yield similar drawings.

The famous Euler’s polyhedron formula (see [60], pp. 102-104 or [55], pp. 27-28) gives a connection for the number of faces, edges and vertices of a plane embedding. Since the result of Euler plays an important role in topological graph theory, we also give the proof.

**Theorem 2.1.** Let $G$ be a connected planar graph with $n$ vertices, $m$ edges and $f$ faces, then we have for the plane embedding of $G$ that

$$n - m + f = 2.$$ 

**Proof.** The proof is by induction on the number of faces on the embedding. For the base case, if the number of regions is one, that is, if $f = 1$, the graph must be a spanning tree. The formula holds, since for a connected tree we have $m = n - 1$.

For the induction step, suppose that the equation holds when the number of regions is at most $n$, and suppose that for the graph $G$ we have $f = n + 1$. Now some edge $e$ lies in a cycle that is a boundary for two regions. Since the two regions are distinct, the subgraph $G'$ obtained from $G$ by removing the edge $e$ is connected. Denote the number of vertices, faces and edges of $G'$ by $n', f'$ and $m'$. Since $n' = n$, $m' = m - 1$ and $f' = f - 1$, it follows that $n - m + f = 2$. 

8
The right side of the equation is the Euler characteristic of the surface. For the plane and sphere the Euler characteristic is 2. For the usage of the Euler’s polyhedron formula for the other surfaces, see [55] and the references given there.

Instead of searching graph’s plane embedding, for example, by paper and pencil, another way to characterize planarity is to recognize subgraphs that violate planarity. The following result is due to Kuratowski.

**Theorem 2.2 ([79] (See also [60], pp. 108–112.)).** A graph is planar if and only if it has no subgraph homeomorphic to $K_5$ or $K_{3,3}$.

See Figure 2.1 for an illustration of $K_5$ and $K_{3,3}$.

![Basic nonplanar graphs $K_5$ and $K_{3,3}$](image)

Figure 2.1: Basic nonplanar graphs $K_5$ and $K_{3,3}$.

If a graph $G' = (V, E')$ is a planar subgraph of $G$ such that every graph $G''$ obtained from $G'$ by adding an edge from $E \setminus E'$ is nonplanar, then $G'$ is called a maximal planar subgraph of $G$. Let $G'' = (V, E'')$ be a maximal planar subgraph of $G$. If there is no planar subgraph $G'' = (V, E'')$ of $G$ with $|E''| > |E'|$, then $G'$ is a maximum planar subgraph. A maximal planar subgraph is maximal in the sense that adding edges is not possible and the maximum planar subgraph is maximal with respect to the cardinality of its edge set.

Figure 2.2 [84] illustrates the concepts of a planar subgraph, a maximal planar subgraph and a maximum planar subgraph of a nonplanar graph. Graph $G_1$ is a planar subgraph of $G$. $G_1$ is is not a maximal planar subgraph: edge $(1, 5)$ can be added to $G_1$ without violating planarity. The result is $G_2$ which is a maximal planar subgraph. Another maximal planar subgraph of $G$ is $G_3$ which is also a maximum planar subgraph.

A planar graph is outerplanar if it admits a plane drawing where all its vertices lie on the outerface and no two distinct edges intersect, otherwise the graph is non-outerplanar. Maximal outerplanar subgraph and maximum outerplanar subgraph are defined in the similar way as the maximal and maximum planar subgraphs.
A graph is outerplanar if and only if it has no subgraph homeomorphic to $K_4$ or $K_{2,3}$.

Figure 2.3 shows graphs $K_4$ and $K_{2,3}$ which violate outerplanarity.

A maximal planar graph is a planar graph with the property that the addition of any edge results in a nonplanar graph. A maximal outerplanar graph (mop) is an outerplanar graph with the property that the addition of any edge results in a non-outerplanar graph.

The number of triangular faces of a maximal planar graph can be derived from Theorem 2.1.

**Theorem 2.4.** Given a maximal planar graph with $n$ vertices and $m$ edges, the graph contains $2n - 4$ triangular faces.
2.2. ALGORITHMS AND COMPUTATIONAL COMPLEXITY

Definition 2.5. Mops having at least three vertices can be defined recursively as follows:

1. $K_3$ is a mop.

2. If $G$ is a mop which is embedded in the plane so that every vertex lie on the outer face, and $G'$ is obtained by joining a new vertex to two vertices of an edge on the outer face of $G$, then $G'$ is a mop.

3. Graph $H$ is a mop if and only if there is one application of statement (1) and a finite sequence of applications of statement (2) to obtain $H$.

There is also a straight connection between outerplanar graphs and planar graphs. The theorem below is mentioned by Wiegers [132], but probably it is older.

Theorem 2.6. A graph $G$ is outerplanar if and only if $G + K_1$ is planar.

2.2 Algorithms and computational complexity

An algorithm consists of a finite set of well defined instructions which gives a sequence of operations for solving a specific problem [77]. An algorithm has usually at least one input and at least one output. An algorithm transforms the input into the output by performing a sequence of computational steps. An algorithm is deterministic if for a fixed input, the algorithm produces always the same output. Otherwise it is non-deterministic. A randomized algorithm makes random choices during its execution.

Algorithms 2.1 (DFS) and 2.2 (Search) illustrate the concept. These algorithms perform depth-first search for the input graph, that is, all vertices and edges are explored. The name of the algorithm is given first (DFS) and the input is given in the parenthesis ($G = (V, E)$). Standard ways to give an encoding for a graph are adjacency lists and adjacency matrices. In the adjacency list there is an entry, a list, for each vertex, where the adjacent vertices are given. The adjacency matrix for a graph $G = (V, E)$ is a $|V| \times |V|$ matrix where row $i$ shows the adjacent vertices for the $i$th vertex as follows. The entry $i, j$ is 1, if there is an edge between vertices $i$ and $j$, otherwise the entry is 0. The adjacency matrix needs space proportional to $|V|^2$, while adjacency list needs space proportional to $|E|$.

The description of DFS is finite: its instructions are given as a finite list of numbered steps. In general, one step may contain more than one instruction.
DFS($G = (V, E)$)

1. $T \leftarrow \emptyset$
2. mark all vertices in $V$ unvisited
3. while there is an unvisited vertex $u$ in $V$
   4. do SEARCH($u$);
4. return $T$;

Algorithm 2.1: DFS.

SEARCH($u$)

1. mark $u$ visited;
2. for each $v$ that is adjacent to $u$
   3. do if $v$ is not visited
   4. $T \leftarrow T \cup \{(u, v)\}$;
5. SEARCH($v$);

Algorithm 2.2: SEARCH.

Algorithm SEARCH($u$) is a subalgorithm of DFS, since if the condition statement in Step 3 is true, DFS calls SEARCH. When we refer later in this work to DFS, we assume that also SEARCH belongs to the description of DFS.

Depth-first search has many applications in the field of graph algorithms, since it can be used as a skeleton for other algorithms. For example, we can construct a spanning tree for a given graph. Set $T$, where the edges of the spanning tree will be stored, is initialized as empty in Step 1 of DFS. If we arrive during the dfs-traversal from a visited vertex $u$ to an unvisited vertex $v$ in SEARCH, we add $(u, v)$ to $T$ (Step 4). These edges are called the tree-edges of the graph. The other edges are the back-edges. If the input graph is connected, then in the end of DFS edges in $T$ form a spanning tree. DFS can produce different spanning trees depending on the order in which the edges and vertices are searched. If the tree-edges found in a dfs-traversal are directed in the natural way, we get a dfs-tree for the graph. $T$ is returned in the last step of the algorithm.

DFS can also be used to check if a graph is connected: it is enough to count how many times SEARCH is called in Step 4 of DFS. If there are zero calls, the input graph is empty, otherwise the graph contains as many connected
components as there are calls for Search. DFS can also be used to find biconnected components of a graph [124].

Other applications of DFS in this work are planarity testing algorithms studied in the next section, a heuristic to make the graph connected in Chapter 5, and a spanning tree heuristic for approximating different topological invariants discussed in Chapter 4.

To classify different algorithms, a common way is to compare their running times. The running time can be “wall-clock” time, when we start the clock at the same time as the algorithm begins and stop the clock when the algorithm halts. The wall-clock time depends highly on the implementation of the algorithm and on the efficiency of the execution platform. A platform independent approach is to study the number of base operations of the algorithm as the function of the input size. This can be done using asymptotic notations. Consider functions \( f : \mathbb{N} \to \mathbb{R}^+ \) and \( g : \mathbb{N} \to \mathbb{R}^+ \). Function \( f \) is \( O(g) \), if there exists positive constants \( c \) and \( n_0 \) such that \( f(n) \leq cg(n) \), when \( n \geq n_0 \).

If the number of base operations, \( f(n) \), for all inputs of size \( n \) is \( O(g(n)) \), we say that the running time of the algorithm is \( O(g(n)) \). If \( g(n) = n^k \) for some positive constant \( k \), we say that the algorithm is polynomial. Especially when \( g(n) \) is a polynomial function of the first order, we say that the algorithm is linear.

As an example, consider again DFS. It takes a graph as input and then it chooses a vertex from where it starts exploring the other vertices and edges. Finally, all vertices and edges of the graph are explored. Since DFS gets a graph as input, it is reasonable to study the running time as a function of numbers of edges and vertices. We may assume that the input graph is represented by its adjacency list. We count how many assignments and searches for the adjacency lists are made during the execution of DFS for a graph with \( n \) vertices and \( m \) edges. Step 1 is performed once and each of Steps 2, 3 and 4 are performed at most \( n \) times. Algorithm Search is called in total \( n \) times in Step 4 of DFS and in Step 5 of Search. The adjacency lists of each vertex are explored once in Step 2 of Search, so this takes time proportional to \( m \). By adding these all together, we get an \( O(m + n) \) time algorithm.

The class of problems that are solvable in polynomial time is denoted by \( P \). If the time complexity of the algorithm cannot be bounded by any polynomial, we say that the algorithm is exponential. Some problems are such that we do not know, if they have a deterministic polynomial time algorithm. For these problems it might be possible to find the solution non-deterministically in polynomial time: we can guess the solution and verify the correctness of the solution in polynomial time. The class of problems that have a non-
deterministic polynomial time algorithm is denoted by $NP$. It is obvious that $P \subseteq NP$, but whether the inclusion is proper or not, is one of the main open questions in the theoretical computer science.

One of the reasons why most theoreticians believe that $P \neq NP$, is the existence of $NP$-complete problems. This class has the property that if any of its problems can be solved in polynomial time, then every problem in $NP$ can be solved in polynomial time. The reason behind this is that all problems in $NP$ are polynomially transformable to any $NP$-complete problem. Since no polynomial time algorithm for any $NP$-complete problem is known, this gives a good evidence for the intractability of the $NP$-complete problems.

Many computational problems, for example, determining the maximum planar or outerplanar subgraph of a graph, are $NP$-complete. To find optimal solution may require exponential time in the function of the input size: the whole set of possible solutions have to be checked before we can be sure that the optimal solution is found. This kind of method is called exhaustive search. Since the running time of the exhaustive search can be very high even for small problem instances, we do not usually require to solve the problem optimally in practical situation. It is enough to find solutions that are close to optima. Algorithms that return always some solution, but not necessarily optimal, are called approximation algorithms. Fast approximation algorithms can handle much larger input instances than the algorithms based on exhaustive search.

One way to compare approximation algorithms is to study the ratio of the worst possible solution that the approximation algorithm can return and the optimal solution. This is called the performance ratio of the algorithm. Let $G$ be a problem instance, $A(G) \geq 0$ the solution obtained by approximation algorithm $A$ and $OPT(G) \geq 0$ the optimal solution for input $G$. Next we define the performance ratio for maximization problems.

**Definition 2.7.** The performance ratio $R_A(P)$ of an approximation algorithm $A$ for a maximization problem $P$ is the minimum ratio of obtained solutions to the cost of optimal solution:

$$
R_A = \begin{cases} 
\min A(G) \quad \text{if } OPT(G) \neq 0 \\
1 \quad \text{if } OPT(G) = 0.
\end{cases}
$$

The performance ratio for minimization problems is similar, but now the optimal solution is the denominator and the worst solution is the numerator.

**Definition 2.8.** The performance ratio $R_A(P)$ of an approximation algorithm $A$ for a minimization problem $P$ is the maximum ratio of obtained solutions to the cost of optimal solution:
2.3. TESTING PLANARITY AND OUTERPLANARITY

\[ R_A = \begin{cases} 
\max \frac{A(G)}{OPT(G)}, & \text{if } A(G) \neq 0 \\
1, & \text{if } A(G) = 0.
\end{cases} \]

2.3 Testing planarity and outerplanarity

One way to test graph’s planarity is to apply Theorem 2.2 that gives a structural property of nonplanar graphs. By searching directly subgraphs homeomorphic to \( K_5 \) and \( K_{3,3} \) from the graph we obtain an \( O(n^6) \) algorithm [63]. Although the running time is polynomial, it is too slow for practical applications.

The other approach is to construct the plane embedding by an algorithm. We know that the graph is planar, if the construction succeeds and otherwise we can recognize forbidden subgraphs that violate planarity. The first such planarity testing algorithm was described by Auslander and Parter [8], correctly formulated by Goldstein [54] and implemented to run in \( O(n^3) \) time by Shirey [116]. Finally Hopcroft and Tarjan gave a linear time implementation for the algorithm [63]. Clearly any planarity testing algorithm needs at least linear time in the number of vertices and edges of the graph, so this algorithm is optimal up to a constant factor.

Next we introduce the basic ideas behind this path-addition approach to test planarity. We show, following [38], how cubic running time can be achieved. Linear time implementation is quite involved, a detailed description can be found from [111, pp. 364-385] and [93, 92]. The algorithm is based on divide-and-conquer approach. The following property of planar graphs is elementary and trivial. It shows a natural way to divide the problem into smaller subproblems.

**Lemma 2.9.** A graph is planar if and only if all its biconnected components are planar.

By Lemma 2.9, we may assume without loss of generality, that the graph \( G \), whose planarity will be tested, is biconnected (if the input graph is not biconnected, we can test the planarity of every biconnected component separately). Next we divide \( G \) into smaller subgraphs, called pieces. Let \( C \) be an arbitrary cycle of \( G \). Now the edges not in \( C \) are partitioned into classes \( E_1, E_2, \ldots, E_p \) as follows: two edges are in the same class if there is a path between them that does not contain any vertices of \( C \). Each edge class \( E_i \) induces a subgraph of \( G \), called piece \( P_i \). The vertices of a piece \( P_i \) that belong to \( C \) are called attachments of \( P_i \). We may assume that the number of pieces is greater than
or equal to two. If there are no pieces at all, then \( G \) is a cycle and therefore it is planar. If there is only one piece, the graph is planar if and only if the piece is planar (if a piece is planar, it can be embedded inside or outside of \( C \)).

There are two types of pieces with respect to \( C \):

1. an edge whose endpoints belong to \( C \), or

2. a connected subgraph with at least one vertex not in \( C \).

It is obvious that the whole piece should be drawn inside or outside of \( C \). The next task is to check that is it possible to divide the pieces into two sets such that the first set is embedded on the outside of \( C \) and the second set inside of \( C \). To check if there are embeddable pieces, we construct a so called \textit{interlacement graph}. Two pieces \textit{interlace}, if they cannot be drawn on the same side of \( C \) without violating planarity. This can be checked using the attachments and the types of the pieces. The interlacement graph \( I \) of pieces is the graph whose vertices are the pieces of \( G \) and there is an edge between the vertices if the corresponding pieces interlace. Now, if \( I \) is bipartite, pieces can be embedded inside and outside of \( C \), otherwise the planarity is violated.

The described procedure is applied recursively to all pieces. If all pieces are planar and they can be embedded inside and outside of the cycle, we recognize that the graph is planar. This leads to an \( O(n^3) \) time algorithm. The algorithm of Hopcroft and Tarjan [63] is based on using DFS to find the cycle and to recognize the pieces. The algorithm tries to embed pieces inside and outside of the cycle in a good order and using a complicated stack data structure to achieve the linearity.

The second planarity testing paradigm is \textit{vertex addition} approach. Rather than adding an edge or a path, the algorithm starts from a single vertex, and the remaining vertices are considered in some predefined order. A vertex with its adjacent edges to already embedded vertices are added to the planar subgraph. This process is continued until a plane embedding is achieved for the graph, or an edge that violates planarity is found. First vertex addition algorithm was given by Lempel et al. [82]. The algorithm was based on calculating first a suitable numbering for the vertices, and then adding vertices in this order. The authors did not give implementation or complexity analysis for the algorithm. Later Even and Tarjan [45] showed that the suitable numbering can be found in linear time and finally a linear time implementation was given by Booth and Lueker [20]. The algorithm uses a data structure called \( PQ \)-trees.

Later Shih and Hsu [115] gave a new simpler linear time planarity testing algorithm. The algorithm uses a PC-tree data structure [66] and it is based on vertex addition approach. The second new linear time planarity testing
algorithm is given by Boyer and Myrwold [23]. This algorithm is also based on vertex addition approach and it is similar to that of Booth and Lueker [20]. For the implementation details concerning the algorithms of Shih and Shu, see [65] and for the algorithm of Boyer and Myrwold, see [22].

A dynamic planarity testing algorithm [50] maintains a data structure to represent a planar graph $G = (V, E)$ in such a way that the algorithm can answer to the following question

1. For two vertices $u$ and $v$ in $V$ with $(u, v) \notin E$, is $(V, E \cup (u, v))$ planar?

and make the following updates to the data structure:

2. For two vertices $u$ and $v$ in $V$ with $(u, v) \notin E$, add $(u, v)$ to $E$ (in the case of positive answer for the question 1).

3. Delete an edge $e$ from $E$.

4. Add a new vertex to $V$.

5. Delete a vertex from $V$ with all its adjacent edges.

An algorithm is called semi-dynamic, if it supports only insertions or deletions, but not both. Dynamic planarity testing algorithms can be used to search maximal planar subgraphs of a given graph, and these methods are examined in Section 4.1.

Next we study methods to test the outerplanarity of a given graph. Theorem 2.6 gives a direct way to apply any planarity testing algorithm to recognize also outerplanar graphs. Since it takes only linear time to add one vertex $v$ to an input graph $G = (V, E)$ and to add $|V|$ edges to make $v$ adjacent with the vertices of $G$, it takes asymptotically the same time to apply a planarity testing algorithm to test outerplanarity. Since planarity testing algorithms are quite involved, there exists also pure outerplanarity testing algorithms.

In the end of 1970’s, three outerplanarity testing algorithms were given. Brehaut [24] gave a linear time algorithm that was based on the same idea as the Hopcroft-Tarjan planarity testing algorithm, without any need for complicated data structures. Also, an algorithm by Syslo and Iri [121] was based on a similar approach (see also an article by Syslo [120] for characterization, testing, coding, and counting of outerplanar graphs). The third algorithm was given by Mitchell [96], and it was designed to recognize maximal outerplanar subgraphs. The approach was simpler than in the previous two algorithms: for a biconnected graph with $n$ vertices, the algorithm deletes $n - 2$ vertices of
degree two from the graph. If the deletion of a degree two vertex was not possible or if an occurrence of an illegal element was added in an edge list during the execution, the graph was recognized to be non-maximal outerplanar. The algorithm recognizes also outerplanar graphs with minor modifications.

The previous outerplanarity testing algorithms required that the graph should be first divided into biconnected components. Wiegers [132] gave an algorithm that is based on deleting successively vertices of degree less than or equal to two from the input graph, without a need to divide the graph into biconnected components.

2.4 Combinatorial optimization

In this section we introduce commonly used optimization paradigms for computationally hard problems. The discussed methods include local search, simulated annealing, genetic algorithms and branch-and-bound heuristic. These methods produce only approximations if the running time is limited.

2.4.1 Local search

The basic idea behind all local search algorithms is to choose an initial solution, denoted by \( s_0 \), and then trying to improve the quality of this solution [110]. The improvements are done by choosing another solution that belongs to the neighborhood, denoted by \( N(s_0) \), of the current solution. The neighborhood of a solution is a set of other solutions that are close of the current solution (where the definition of “close” depends on the problem in question). This new solution is accepted to be the new current solution, if some requirements are fulfilled. Acceptance criterion could be that the new solution always improves the current solution (hillclimbing) or that it does not make the current solution too much worse (simulated annealing), or the whole neighborhood is searched and the best neighbor solution is chosen. All these algorithms have a stopping criterion (termination condition), that might be the number of iteration steps, or notifying that there are no possibilities to improve the current solution anymore. See Algorithm 2.3 for the general structure of a local search algorithm.

There are several problem dependent parameters to be decided when the algorithm is implemented: how the set of possible solutions is represented, what is a good definition for the neighborhood structure, what is the stopping criterion, and how the initial solution is constructed.
2.4. COMBINATORIAL OPTIMIZATION

Local-Search
1 select an initial solution \( s_0 \);
2 while not termination condition
3 \hspace{1em} do randomly select a solution \( s \in N(s_0) \)
4 \hspace{1em} if \( s \) is acceptable
5 \hspace{2em} then \( s_0 \leftarrow s \);
6 return \( s_0 \);

Algorithm 2.3: Basic structure of a local search algorithm.

2.4.2 Simulated annealing

Simulated annealing (SA) algorithm imitates the cooling process of material in a heat bath. SA was originally proposed by Kirkpatrick et al. [75] based on some ideas given by Metropolis et al. [94].

The search begins with initial temperature \( t_0 \) and ends when temperature is decreased to frozen temperature \( t_f \), where \( 0 \leq t_f \leq t_0 \). The equilibrium detection rate \( r \) tells when an equilibrium state is achieved, and temperature is decreased geometrically by multiplying current temperature by the cooling ratio \( \alpha \), where \( 0 < \alpha < 1 \). To determine good parameters for a given problem is often a hard task, and it needs experimental analysis. There are also adaptive techniques for SA [127].

The temperature of the simulated annealing algorithm gives the probability of choosing solutions that makes the current solution worse. If there are many bad solutions in a neighborhood of a “better” solution, and these bad solutions are accepted too often, the algorithm does not converge to a good solution. If the good solutions are rare and the bad solutions appear very often, the temperature should be low enough to ensure that the bad solutions are not accepted too often.

See Algorithm 2.4 for the general structure of the simulated annealing for a maximization problem. In Step 6, “Cost” denotes the goodness of the solution, and it is depended on the problem in question.

Often it is possible to improve the solution quality and to decrease the needed computation time of SA by finding a better initial solution than a randomly generated solution [5, 69, 110]. For more details concerning simulated annealing algorithms, see [1, 110, 127] and the references given there.
CHAPTER 2. PRELIMINARIES

SA

1. select a cooling ratio $\alpha$ and an initial temperature $t_0$;
2. select a frozen temperature $t_f$ and an equilibrium detection rate $r$;
3. select an initial solution $s_0$; set $t \leftarrow t_0$ and $e \leftarrow 0$;
4. while $t \geq t_f$
5. \hspace{1em} do while $e \leq r$
6. \hspace{2em} do $e \leftarrow e + 1$;
7. \hspace{2em} randomly select a solution $s \in \mathcal{N}(s_0)$;
8. \hspace{2em} $\delta \leftarrow Cost(s) - Cost(s_0)$;
9. \hspace{2em} generate a random integer $i$, $0 \leq i \leq 1$;
10. \hspace{2em} if $i \leq e^{-\delta/t}$
11. \hspace{3em} then $s_0 \leftarrow s$;
12. \hspace{1em} $t \leftarrow \alpha t$;
13. \hspace{1em} $e \leftarrow 0$;
14. return $s_0$;

Algorithm 2.4: Basic structure of a simulated annealing algorithm.

2.4.3 Genetic algorithms

The general principle underlying genetic algorithms is that of maintaining and improving a set of possible solutions, often called population, instead of only one solution as in SA. The population undergoes an evolutionary process which imitates the natural biological evolution. In each generation better solutions candidates have a greater chance to reproduce, while worse solutions are more likely to die and to be replaced by new solutions. To distinguish between “good” and “bad” solutions there is a need to define an evaluation function that maps each solution to a real number.

The general structure of a genetic algorithm is given in Algorithm 2.5 (GA).

The basic operations in recombination are the mutation and crossover operations. Mutation is a unary operation which increases the variability of the population by making pointwise changes in the representation of the solutions. Usually crossover combines the features of the two parents to form two new solutions by swapping corresponding segments of the parents’ representations. See Mitchell [95] for further details concerning genetic algorithms.

20
2.4. COMBINATORIAL OPTIMIZATION

GA
1  t ← 0;
2  create the initial population \(P(0)\);
3  evaluate the initial population;
4  while not termination condition
5     do  \(t \leftarrow t + 1\);
6     select solution candidates to be reproduced;
7     recombine (i.e. apply genetic operations to create the new
8     population \(P(t)\));
9  evaluate \(P(t)\);
10 return the best solution from \(P(t)\);

Algorithm 2.5: Basic structure of a genetic algorithm.

2.4.4 Branch-and-bound

One method to solve a discrete and finite optimization problem is to generate all possible solutions and then choose the best one of them. For \(NP\)-complete problems this exhaustive search method fails since the number of solutions is exponential in the size of the input. For example, given a graph \(G = (V, E)\), there exists \(2^{|E|}\) different subgraphs containing all \(|V|\) vertices. If the problem in question asks a subgraph of the given graph with some specific property, it is possible that all subgraphs need to be checked before the right one is found. Only small instances can be solved in this way.

Branch-and-bound is a method that can be used in the exhaustive search by recognizing partial solutions that cannot lead to an optimal solution. For example, suppose that we know that the optimal solution for a maximization problem is at least \(k\) (we have found a solution with cost \(k\)). If during the generation of the other solution candidates we recognize that the current solution cannot be augmented to any solution with cost \(\geq k\), we can stop generating these solution candidates and continue searching from more promising solutions.

Branch-and-bound is a method that can be used in the exhaustive search by recognizing partial solutions that cannot lead to an optimal solution. For example, suppose that we know that the optimal solution for a maximization problem is at least \(k\) (we have found a solution with cost \(k\)). If during the generation of the other solution candidates we recognize that the current solution cannot be augmented to any solution with cost \(\geq k\), we can stop generating these solution candidates and continue searching from more promising solutions.

The efficiency of the branch-and-bound techniques depends highly on the problem in question and the order in which the solution candidates are found. If the lower bound for the optimal solution is bad, there are little possibilities to reject any solutions. The worst case running time of branch-and-bound is still exponential, but often it decreases the computation time remarkably.
2.5 Experimental algorithmics

Analysis of algorithms concentrates on determining the theoretical properties of algorithms, for example, what is the running time of an algorithm, how much space an algorithm needs in the worst case, and if an algorithm is an approximation algorithm, what is its performance ratio.

In practice, the theoretical analysis of an algorithm is not always enough. The algorithm might be so complicated that it is almost impossible to implement it to run within the claimed asymptotic time and space bounds. The worst case performance ratio of the algorithm can be poor, but for all reasonable inputs the solutions are of good quality. The constant factor of the asymptotic running time could be so high that another algorithm with a polynomially higher running time can be faster for small inputs.

Yet another reason using non-theoretic approaches to study the properties of an algorithm could be the hardness of the theoretical analysis. For example, if the performance ratio of the algorithm is not known, the only way to test the efficiency of the algorithm is experimental analysis. We can run the algorithm for a number of inputs (test instances) and try to draw some conclusion about the outputs. For an algorithm with an unknown performance ratio it is possible to use the experimental analysis to get a good guess on the possible performance ratio. In the best case, the experimental analysis may lead to new theoretical results. This may also help to recognize parts of the algorithm which can be improved to make it more efficient.

In general, the experimental analysis of algorithms can be used to compare the running times and solution qualities of different algorithms. There are several guidelines to perform reliable comparison of algorithms, next we recall some suggestions given by Johnson [68], Moret [98], McGeoch [91], and Moret and Shapiro [99].

**Define the goal of the study clearly.** If the aim of the research is not clear enough in the beginning of the work, the study may go to totally wrong directions. The researcher should be familiar with his research area, mainly on the earlier results. When the goal of the study is clearly defined, it is easier to start asking right question to design the experimental setup well. This decreases the danger on devoting too much computation time to wrong questions. During the experimentation process, there might appear new subgoals, or the old goal may be changed to a better one.

**Use a reasonably large number of well chosen test instances.** When the goal is clear, it is easier to construct a testbed that can be used to find differences between the algorithms and can tie the experiments with the real world. The
testbed should contain instances from the real world applications and (randomly) generated instances. The generated instances can be used to test some specific properties of the algorithms, mainly their weaknesses and strengths. Often a testbed includes test instances with the known optima, and random instances whose optima is not known. Furthermore, the earlier testbeds should be taken into account to tie the experiments to the literature. At least some test instances should be large enough to support justified conclusions about the (asymptotic) running time of the algorithms.

**Draw justified conclusions.** Reporting just the solutions of the experimented algorithms is not enough. There should be drawn conclusions from the data that are related to the goal of the study. For example, can we sort the algorithms by their running time difference or by their solution quality? Is it possible to recognize test instances which are good for an algorithm, but bad for another algorithm? When the differences of the algorithms are not clear enough, there is often a need to apply statistical methods to justify the conclusions.

**Ensure reproducibility and comparability of the experiments.** The reported measures of the algorithms should be comparable with the earlier results, and it should also be possible to apply the results in the forthcoming experiments. Since the test platform affects highly on the running time of the algorithms, the exact description about the used computer should be given. All the test instances, source codes of the algorithms and instance generators should be available to the public. This ensures that other researchers can reproduce all experiments.
Chapter 3

Topological Invariants

An invariant of a graph $G$ is a number associated with a property of $G$ having the same value for any graph isomorphic to $G$. Topological invariants describe a graph’s embeddability on different surfaces. In what follows, the used surface is the plane.

In this chapter we define the following four topological invariants: maximum planar subgraph, maximum outerplanar subgraph, thickness, and outerthickness. If a graph is not (outer)planar, these invariants give an approximation how far away from (outer)planarity the graph is. We recapitulate known results and give some new results for the thickness and outerthickness of a graph.

3.1 Maximum planar subgraph

The maximum planar subgraph problem (MPS) is defined as follows.

Definition 3.1. MPS: given a graph $G = (V, E)$, how many edges are there in a maximum planar subgraph $G'$ of $G$?

MPS is one of the most studied topological invariants that can be found in the literature. The origin of MPS lies in so called facility layout problem, where different facilities are assigned on a plane in order to minimize distances and flows between them [97, 61]. MPS also plays an important role in graph drawing [70, 123], since a large planar subgraph can be used to obtain layouts fulfilling many aesthetic criteria [38].

Determining MPS is actually the same as determining the skewness of a graph $G$, i.e., the minimum number of edges whose removal leads to a planar subgraph. If we can determine a subgraph $G'' = (V, E'')$, whose extraction leads to a maximum planar subgraph $G' = (V, E')$, we know that skewness is
CHAPTER 3. TOPOLOGICAL INVARIANTS

$|E''|$ and that the solution of MPS is $|E'|$, since union of $G'$ and $G''$ is $G$. In this work we consider MPS only, but all results can be easily formulated for skeness.

The complexity status of MPS was solved independently by Liu and Geldmacher [85], and Yannakakis [135]. A few years later Watanabe et al. [129] also gave similar results. These works also include results for a wider class of edge deletion problems.

**Theorem 3.2 ([85, 135, 129]).** MPS is NP-complete.

The problem of determining a maximal planar subgraph is close to MPS but it belongs to $P$. Notice that a maximum planar subgraph is also a maximal planar subgraph, but not vice versa. MPS seeks for a global optima, and the maximal planar subgraph problem seeks a locally optimal solution. The algorithms for finding a maximal planar subgraphs can be used to approximate MPS. These methods are studied in Chapter 4.

Next we introduce graph classes for which MPS is known. But first, we give a useful corollary of Theorem 2.1. The corollary shows a connection between the number of boundary edges of regions and the total number of edges in a graph.

**Corollary 3.3 ([60]).** Let $G$ be a connected planar graph with $n$ vertices, $m$ edges and $f$ faces in which every face is a cycle of length $k$, where $n,k \geq 3$. Then for the plane embedding of $G$ it holds that

$$m = \frac{k(n - 2)}{k - 2}.$$

**Proof.** Since every face of $G$ is $C_k$, each face has $k$ edges and in addition, each edge of $G$ belongs to two faces. Thus $kf = 2m$, which when substituted into the formula of Theorem 2.1, gives the result. $\square$

The following theorems are direct applications of Corollary 3.3. The next theorem gives the maximum number of edges in a planar subgraph. The equality holds for graphs which contain a maximal planar graph with $|V|$ vertices as a subgraph.

**Theorem 3.4 ([60]).** Let $G' = (V, E')$ be a maximum planar subgraph of a graph $G = (V, E)$. Then $|E'| \leq 3|V| - 6$.

If the graph does not contain triangles, a sharper bound can be derived. The equality holds for graphs having a maximal bipartite planar graph with $|V|$ vertices as a subgraph.
Theorem 3.5 ([60]). Let \( G' = (V, E') \) be a maximum planar subgraph of a graph \( G = (V, E) \) which does not contain any triangles. Then \(|E'| \leq 2|V| - 4\).

The last nontrivial graph class whose MPS is known is hypercubes. The result is given by Kotzig.

Theorem 3.6 ([78] (See also [100], pp. 21–22 and [29])). The maximum planar subgraph of \( Q_n \) contains \( 2^{n+1} - 4 \) edges.

3.2 Maximum outerplanar subgraph

Outerplanar graphs are a widely studied graph class with applications in graph drawing [73, 17] and with interesting theoretical properties [46, 87, 89].

The problem of finding a maximum outerplanar subgraph (MOPS) can be defined as follows.

Definition 3.7. MOPS: given a graph \( G = (V, E) \), how many edges are there in a maximum outerplanar subgraph \( G' \) of \( G \)?

The \( NP \)-completeness of MOPS was shown by Liu and Geldmacher [85], Yannakakis [135], and Watanabe et al. [129].

Theorem 3.8 ([85, 135, 129]). MOPS is \( NP \)-complete.

MOPS is known for the same graph classes as MPS. We refer to Harary [60] for these results that are direct applications of Corollary 3.3. The next theorem gives the maximum number of edges in an outerplanar subgraph. The equality holds for graphs containing a maximal outerplanar graph with \(|V|\) vertices as a subgraph.

Theorem 3.9. Let \( G' = (V, E') \) be a maximum outerplanar subgraph of a graph \( G = (V, E) \). Then \(|E'| \leq 2|V| - 3\).

If the graph does not contain triangles, again a sharper bound can be found.

Theorem 3.10. Let \( G' = (V, E') \) be a maximum outerplanar subgraph of a graph \( G = (V, E) \) which does not contain any triangles. Then \(|E'| \leq 3|V|/2 - 2\).

The equality in the above theorem holds for graphs containing a maximal bipartite outerplanar graph with with \(|V|\) vertices, such that \(|V|\) is even, as a subgraph.

Guy and Nowakowski constructed the maximum outerplanar subgraph for hypercubes.

Theorem 3.11 ([56]). The maximum outerplanar subgraph of \( Q_n \) contains \( 3 \cdot 2^{n-1} - 2 \) edges.
3.3 Thickness

The following conjecture was given by Harary [59]:

Prove or disprove the following conjecture: For any graph $G$ with 9 points, $G$ or its complementary graph $\overline{G}$ is nonplanar.

The problem is the same as determining whether $K_9$ is biplanar or not, that is, a union of two planar graphs. The problem was solved independently by Battle et al. [9] and Tutte [125] by constructing all subgraphs for $K_9$. They showed that $K_9$ is not biplanar. Tutte [126] generalized the problem as the thickness of a graph.

**Definition 3.12.** The graph-theoretical thickness (thickness, for short) of a graph, denoted by $\Theta(G)$, is the minimum number of planar subgraphs into which the graph can be decomposed.

The thickness of a planar graph is 1 and the thickness of a nonplanar graph is at least 2. Thickness has applications in VLSI (Very Large Scale Integration) design [3]. The computational complexity of thickness was solved by Mansfield.

**Theorem 3.13 ([90]).** Determining the thickness of a graph is NP-complete.

The only non-trivial graph classes with known thicknesses are the complete graphs, complete bipartite graphs, and hypercubes. The optimal solution for the thickness of complete graphs $K_n$ was given for almost all values of $n$ by Beineke and Harary [11]. A decade later Alekseev and Gonchakov [4], and independently Vasak [128], solved the remaining cases.

**Theorem 3.14 ([4, 11, 128]).** For complete graphs, $\Theta(K_n) = \left\lceil \frac{n+7}{6} \right\rceil$, except that $\Theta(K_9) = \Theta(K_{10}) = 3$.

See Figure 3.1 for a decomposition of $K_9$ into three planar subgraphs.

For complete bipartite graphs $K_{m,n}$, thickness is solved for almost all values of $m$ and $n$.

**Theorem 3.15 ([12]).** For complete bipartite graphs, $\Theta(K_{m,n}) = \left\lceil \frac{mn}{2(m+n-2)} \right\rceil$, except possibly when $m$ and $n$ are odd, and there exists an integer $k$ satisfying

$$n = \left\lceil \frac{2k(n-2)}{n-2k} \right\rceil.$$ 

We show in Chapter 8, by applying a simulated annealing algorithm, that Theorem 3.15 holds for all $m < 30$. For example, it was unknown if $\Theta(K_{17,21})$ is equal to 5 or 6 (the thickness of $K_{13,17}$ is at least 5 due to Euler’s polyhedron
3.3. THICKNESS

Figure 3.1: A minimum planar decomposition of $K_9$.

The thickness of hypercubes was determined by Kleinert [76].

**Theorem 3.17 ([76]).** $\Theta(Q_n) = \left\lceil \frac{n+1}{4} \right\rceil$.

Next we give a new upper bound for complete tripartite graphs whose vertex sets are of equal size [108]. Theorem 3.18 shows that the thickness of the complete tripartite graph $K_{n,n,n}$ is less than or equal to $n$ divided by two and rounded up to the nearest integer.

**Theorem 3.18 ([108]).** For a tripartite graph $K_{n,n,n}$, it holds that $\Theta(K_{n,n,n}) \leq \left\lceil \frac{n}{2} \right\rceil$.

*Proof.* Let $K_{n,n,n}$ be a complete tripartite graph, and suppose that $n = 2l$, where $l$ is a positive integer. Denote the vertex sets of size $n$ of $V_1$, $V_2$ and $V_3$. We give a constructive method to achieve a planar decomposition with the claimed thickness:

1. Arrange all vertices from $V_1$ and $V_2$ to a line in alternating order, that is, vertices from $V_1$ are placed on locations $2i$ and vertices from $V_2$ on locations $2i + 1$, for all $0 \leq i < n$.

2. Join each vertex on location $i$ with the vertices on locations $i - 1$ and $i + 1$ modulo $2n$ to obtain a cycle $C$. 


Choose a vertex $v$ from $V_3$ and place it inside $C$ and join $v$ with all the vertices of $C$.

4. Choose a vertex $u$ from $V_3$ and place it outside $C$ and join $u$ with all the vertices of $C$.

5. The edges added in Steps 2, 3, and 4 form a planar subgraph of $K_{n,n,n}$ with $6n$ edges. Remove these edges and delete vertices $v$ and $u$ from $V_3$.

6. If $V_3 \neq \emptyset$, move the vertices from $V_1$ at locations $2i$ to $2i + 4$ modulo $2n$ for all $0 \leq i < n$ and goto Step 2.

7. A planar decomposition of $K_{n,n,n}$ is achieved.

Since in each iteration we remove two vertices from $V_3$, the process extracts $n/2$ subgraphs from the original graph. Steps 2, 3, and 4 guarantee that we always extract a planar graph with $6n$ edges from the original graph. When two vertices from $V_3$ are placed in Steps 3 and 4 inside and outside $C$, all their incident edges are extracted from the graph. Therefore, every edge between $V_3$ and $V_1$ ($V_3$ and $V_2$) is placed in one of the extracted planar subgraphs. To see that edges between sets $V_1$ and $V_2$ are also handled, it is enough to notice that when moving vertices of $V_1$ to new locations in Step 6, each vertex becomes adjacent with two new vertices from $V_2$. Thus, each edge between $V_1$ and $V_2$ is placed in one of the planar subgraphs. Since a subgraph of a graph must have a thickness not larger than the graph itself, the theorem holds also for odd values of $n$.

The upper bound of Theorem 3.18 for the complete tripartite graphs $K_{n,n,n}$ is one less than or equal to the upper bound taken from Theorem 3.14 for the complete graphs $K_{3n}$. Unfortunately, the equality in Theorem 3.18 does not hold for all graphs. For example, if $n = 20$, there is a gap between the upper bound and the results achieved in our experiments (see Chapter 8).

Next we give two lower bounds for thickness, see Beineke et al. [12] for references concerning their origin. The first lower bound is a direct application of Theorem 3.4.

**Theorem 3.19.** Let $G = (V, E)$ be a graph with $|V| = n$ and $|E| = m$. Then $\Theta(G) \geq \left\lceil \frac{m}{3n-6} \right\rceil$.

If a graph does not contain any triangles, as it is for bipartite graphs, a tighter lower bound can be derived by applying Theorem 3.5.
Theorem 3.20. Let \( G = (V, E) \) be a graph with \( |V| = n \), \( |E| = m \) and with no triangles. Then \( \Theta(G) \geq \left\lceil \frac{m}{2n-4} \right\rceil \).

The lower bounds of Theorems 3.19 and 3.20 are also the exact values for the thickness of almost all complete and complete bipartite graphs.

Wessel [130] gave lower and upper bounds for the thickness of a graph as a function of the minimum and maximum degree. The upper bound was independently given also by Halton [58].

Theorem 3.21 ([58, 130]). Let \( G \) be a graph with minimum degree \( \delta \) and maximum degree \( \Delta \). Then it holds that \( \left\lceil \frac{\delta + 1}{4} \right\rceil \leq \Theta(G) \leq \left\lceil \frac{\Delta + 2}{4} \right\rceil \).

Halton conjectured a stronger upper bound \( \Theta(G) \leq \left\lceil \frac{\Delta + 2}{4} \right\rceil \). Sýkora et al. [119] gave a counterexample by constructing a class of regular graphs of degree \( d \) with thickness \( \lceil d/2 \rceil \). The construction shows that the upper bound of Theorem 3.21 is tight.

Dean et al. [37] gave an upper bound as a function of the number of edges.

Theorem 3.22 ([37]). Let \( G \) be a graph with \( m \) edges, then it holds that \( \Theta(G) \leq \lceil \sqrt{m/3} + 3/2 \rceil \).

They also conjectured a tighter upper bound.

Conjecture 3.23 ([37]). \( \Theta(G) \leq \sqrt{m/16} + O(1) \) for an arbitrary graph \( G \) with \( m \) edges.

The thickness of degree-constrained graphs is studied by Bose and Prabh [21], and results for the thickness of random graphs are given by Cooper [35]. Mutzel et al. [71] have shown that the thickness of the class of graphs without \( K_5 \)-minors is at most two. The genus of a graph is the minimum number of handles that must be added to the plane to embed the graph without any crossings. Asano has studied the thickness of graphs with genus at most 2 [6, 7]. Thickness results for other surfaces are reported by White and Beineke [131] and Ringel [113].

### 3.4 Outerthickness

Instead of decomposing the graph into planar subgraphs, outerthickness seeks a decomposition into outerplanar subgraphs.

**Definition 3.24.** The outerthickness of a graph, denoted by \( \Theta_o(G) \), is the minimum number of outerplanar subgraphs into which the graph can be decomposed.
Outerthickness seems to be studied first in Geller’s unpublished manuscript (see [60], pp. 108 and 244), where it was shown that \( \Theta_o(K_7) \) is 3 by similar exhaustive search as in the case of the thickness of \( K_9 \). See Figure 3.2 for a decomposition of \( K_7 \) into three outerplanar subgraphs.

![Figure 3.2: A minimum outerplanar decomposition of \( K_7 \).](image)

The complexity status of outerthickness is open, but since thickness and MOPS are \( NP \)-complete, we conjecture that determining the outerthickness of a graph is also \( NP \)-complete.

**Conjecture 3.25.** Determining the outerthickness of a graph is \( NP \)-complete.

The outerthickness of complete graphs was solved by Guy and Nowakowski.

**Theorem 3.26 ([56]).** For complete graphs, \( \Theta_o(K_n) = \lceil \frac{n+1}{4} \rceil \), except that \( \Theta_o(K_7) = 3 \).

The same authors also gave optimal solutions for the outerthickness of complete bipartite graphs and hypercubes.

**Theorem 3.27 ([57]).** For complete bipartite graphs with \( m \leq n \), \( \Theta_o(K_{m,n}) = \lceil \frac{mn}{2m+n-2} \rceil \).

**Theorem 3.28 ([56]).** \( \Theta_o(Q_n) = \lceil \frac{n+1}{3} \rceil \).

Next we give for outerthickness two lower bounds that can be derived from Theorems 3.4 and 3.5.

**Theorem 3.29.** Let \( G = (V, E) \) be a graph with \( |V| = n \) and \( |E| = m \). Then \( \Theta_o(G) \geq \lceil \frac{m}{2n-3} \rceil \).

**Theorem 3.30.** Let \( G = (V, E) \) be a graph with \( |V| = n \), \( |E| = m \) and with no triangles. Then \( \Theta_o(G) \geq \lceil \frac{m}{3n/2-2} \rceil \).
The lower bounds of Theorems 3.29 and 3.30 are also the exact values for the outerthickness of complete graphs, complete bipartite graphs, and hypercubes.

To give similar lower and upper bounds based on the minimum and maximum degree of the graph as in Theorem 3.21 for thickness, we recall some preliminary results from Halton’s article [58].

Lemma 3.31 describes the possibility to augment a given graph to a regular graph containing the original graph as a subgraph.

**Lemma 3.31 ([58]).** Given a graph $G$ of degree $\Delta$, there exists a regular graph of degree $\Delta$, containing $G$ as a subgraph.

A graph is 2-factorable, if it is a union of edge-disjoint cycles. The well known Petersen’s theorem states that regular graphs of even degree have this property.

**Lemma 3.32 ([104]).** If a graph is regular and of even degree, then it is 2-factorable.

Next we give lower and upper bounds for outerthickness in the terms of minimum and maximum degree. The proof of the following theorem is similar to that given in [58, 130], but we use outerplanar subgraphs instead of planar subgraphs.

**Theorem 3.33.** For a graph with minimum degree $\delta$ and maximum degree $\Delta$, it holds that $\lceil \delta/4 \rceil \leq \Theta_o(G) \leq \lceil \Delta/2 \rceil$.

**Proof.** To prove the first inequality, let $\delta$ be the minimum degree of a graph $G$ with $n$ vertices. $G$ has at least $\frac{\delta n}{2}$ edges. Since the maximal outerplanar graph has at most $2n - 3$ edges by Theorem 3.9, we have

$$\Theta_o(G) \geq \left\lceil \frac{\delta n}{2(2n-3)} \right\rceil \geq \left\lceil \frac{\delta n}{4n} \right\rceil = \left\lceil \frac{\delta}{4} \right\rceil .$$

To prove the second inequality, let $\Delta$ be the maximum degree of $G$. By Lemma 3.31, there exists a regular graph $G_r$ of degree $\Delta$, if $\Delta$ is even, or of degree $\Delta + 1$, if $\Delta$ is odd, which contains $G$ as a subgraph. Applying Lemma 3.32 to extract cycles (which are obviously outerplanar) from the regular graph of even degree and using induction over the degree of the regular graph, the claimed upper bound can be obtained. The theorem follows.

For more details concerning the proof of the second inequality, see Halton’s article [58]. Since $\Theta_o(G) \geq \Theta(G)$ and the upper bound is tight for thickness [119], it follows that the upper bound is tight also for outerthickness.

33
Dean et al. [37] gave an upper bound for thickness as a function of the number of edges (see, also Theorem 3.22). If their proof technique is applied straightforward to outerplanar graphs, bound \(\lfloor \sqrt{m/2} + 1/2 \rfloor\) is obtained. The upper bound is of the right order, since the outerthickness of the complete graph with \(n\) vertices is \(O(n)\). On the other hand, since \(\Theta_o(K_n)\) is approximately \(\sqrt{m/8}\) and \(\Theta_o(K_{n,n})\) is approximately \(\sqrt{m/9}\), it seems that the constant is not the best possible. We conjecture the following upper bound for outerthickness.

**Conjecture 3.34.** \(\Theta_o(G) \leq \sqrt{m/8} + O(1)\) for an arbitrary graph \(G\) with \(m\) edges.

Heath [62] has shown that a planar graph can be divided into two outerplanar graphs. Therefore, \(\Theta_o(G) \leq 2\Theta(G)\).

### 3.5 Other invariants

Next we give a brief introduction to other topological invariants that are closely related to MPS, MOPS, thickness and outerthickness.

#### 3.5.1 Arboricity

The *arboricity* of a graph, denoted by \(\Upsilon(G)\), is the minimum number of trees whose union is \(G\). Nash-Williams [103] gave the exact solution for arboricity

\[
\Upsilon(G) = \max \left\lceil \frac{m_H}{n_H - 1} \right\rceil,
\]

where the maximum is taken over all nontrivial subgraphs \(H\) of \(G\). The number of vertices and edges in \(H\) are denoted by \(n_H\) and \(m_H\), respectively. Applying Nash-Williams’ result, Dean et al. [37] showed that \(\Upsilon(G) \leq \lfloor \sqrt{m/2} \rfloor\). This gives also a lower bound for outerthickness.

#### 3.5.2 Book thickness

In the *book thickness* of a graph [15], which is sometimes called the *pagenumber* or *stacknumber*, vertices are placed on a line (the spine) and edges are routed without intersections via half-planes (pages) having common boundary with the spine. Book thickness indicates the minimum number of needed pages, and it is denoted by \(\Theta_b(G)\). Every page of the book contains an outerplanar subgraph with a fixed permutation for the vertices. Therefore book thickness
gives an upper bound for the outerthickness of a graph. Also, it is shown that \( \Theta(G) \leq [\Theta_b(G)/2] \) \cite{18}. See, for example, Bilski \cite{19}, Yannakakis \cite{136}, and Wood \cite{133} for more results and references concerning book thickness.

### 3.5.3 Geometric thickness

*Geometric thickness* is the smallest number of layers such that the graph can be drawn in the plane with straight line edges and each edge assigned to a layer such that no two edges cross. Eppstein showed \cite{43} that graph-theoretic thickness and geometric thickness are not asymptotically equivalent, that is, for every positive \( t \), there exists a graph with thickness three and geometric thickness \( \geq t \). See articles by Dillencourt et al. \cite{40} and Wood \cite{134} for recent results and references for geometric thickness.
Chapter 4

Approximating invariants

In this chapter we introduce approximation algorithms from the literature for MPS, MOPS, thickness and outerthickness. In the next chapter we will represent more algorithms based on triangular cactus heuristic. In what follows, we assume that graphs are connected.

Throughout this work, all algorithms for MPS and MOPS returns a subgraph of the input graph as a result. The solution for the problem in question is the number of edges in the returned approximation. Thickness and outerthickness algorithms returns a partition of the edges of the input graph as a result, the solution for the problem is the number of subsets in the partition.

4.1 Algorithms for MPS

Since MPS is $NP$-complete, we mainly concentrate on the approximation results and approximation algorithms. The simplest way to find an approximation for MPS is to produce a spanning tree for the input graph. Since a spanning tree contains $n - 1$ edges, and a maximum planar subgraph could contain at most $3n - 6$ edges, the performance ratio of this method is $1/3$. This ratio was first improved by Călinescu et al. [26] to $4/9$. The algorithm and its variations are studied in the next chapter.

Another approach is to search for a maximal planar subgraph. A maximal planar graph can be recognized by performing a planarity test for the graph and counting the number of edges. Recently Nagamochi et al. [102] gave a simple linear time algorithm to recognize maximal planar graphs without planarity tests. A greedy algorithm to search a maximal planar subgraph is to apply a planarity testing algorithm and to add as many edges as possible to a planar subgraph. See Algorithm 4.1 (GRE) for a detailed description of this edge adding method.
CHAPTER 4. APPROXIMATING INVARIANTS

\( \text{GRE}(G = (V, E), G' = (V, E')) \)
1. \( E' = E \setminus E' \);
2. \textbf{while} there is an edge \((u, v)\) in \(E''\)
3. \quad \textbf{do} \( E' \leftarrow E' \cup \{(u, v)\}, \ E'' \leftarrow E'' \setminus \{(u, v)\} \);
4. \quad \textbf{if} \ (V, E') \text{ is not planar}
5. \quad \textbf{then} \( E' \leftarrow E' \setminus \{(u, v)\} ; \)
6. \textbf{return} \((V, E')\);

Algorithm 4.1: GRE for MPS.

GRE takes as input a graph \( G = (V, E) \) and its planar subgraph \( G' = (V, E') \). The algorithm returns a maximal planar subgraph. The construction starts often with no edges \( (G' = (V, \emptyset)) \), but using our notation, we can apply the algorithm to augment every planar subgraph to a maximal planar subgraph. If a linear time planarity testing algorithm is applied, the running time of GRE is \( O(nm) \) for a graph with \( n \) vertices and \( m \) edges.

It is sometimes possible to decrease the number of planarity test calls in GRE by handling the edges in other than random order. By performing a dfs-traversal for the graph and scanning the edges in the same order as they were found during dfs, there is no need to apply planarity test for the tree-edges. This decreases the number of planarity test calls by \( n - 1 \) for a graph with \( n \) vertices.

There are also faster algorithms to search a maximal planar subgraph. Di Battista and Tamassia [39] gave a semi-dynamic planarity testing algorithm supporting only edge insertion. The algorithm can be used to find the maximal planar subgraph in time \( O(m \log n) \). Cai et al. [25] gave an algorithm based on Hopcroft-Tarjan planarity testing algorithm with the same running time. Then La Poutré [80] gave another incremental planarity testing algorithm that also produced maximal planar subgraphs. The running time of this method is \( O(n + m \alpha(m, n)) \), where \( \alpha \) is a very slowly growing function whose value for all practical purposes is at most five (see [36], pp. 450-453).

Djidjev [41] improved the time complexity of determining the maximal planar subgraph to be linear. Liebers [84] claims that the algorithm of Djidjev is so involved, that a linear time implementation is hard to achieve. Further, the planarity testing algorithm of Shih and Hsu [115] can be extended to produce a maximal planar subgraph in linear time [64].

Cimikowski and Coppersmith [34] showed that there exist graphs for which
the worst case ratio of the number of edges in maximal and maximum planar subgraph is 1/3. Cimikowski [31] proved that the ratio holds for several algorithms based on path-addition and vertex addition approaches. The same ratio also holds for GRE.

Cimikowski [30] proposed a branch-and-bound algorithm for MPS. It takes $O(n^{2m})$ time to test exhaustively the planarity of all subgraphs for a graph with $n$ vertices and $m$ edges. Therefore several conditions to decrease the number of subgraphs were studied, including methods to make the subgraph generation in a better order. The algorithm is applicable only for graphs with no more than 300 edges due to the high running time.

Another powerful approximation method, called J-M, based on polyhedral combinatorics and a branch-and-cut approach was given by Jäger and Mutzel [70]. Experimental test showed that the method improves remarkably the earlier results, although its running time can be very high. Therefore a time limit was used in test runs for large graphs in the experiments. The Hopcroft-Tarjan’s planarity testing algorithm plays an important role also in this algorithm.

Goldschmidt and Takvorian [53] introduced 2Phase heuristics based on ideas given by Takefuji and Lee [122]. First a linear order of vertices is constructed and then edges are divided into three sets $A$, $B$ and $C$ in such a way that $|A| + |B|$ is maximum so that no two edges both in $A$ and $B$ intersect with respect to the linear ordering of the vertices. The edges in $A$ and $B$ induce a planar subgraph and $C$ contains the remaining edges. The solution quality depends highly on the permutation of the vertices and the method used when edges are divided into different sets.

Later, Resende and Ribeiro [112] improved the algorithm of Goldschmidt and Takvorian by applying a greedy randomized adaptive search procedure (GRASP) [106] to obtain a permutation of vertices and then modifying the current permutation by swapping the positions of the vertices to get locally optimal solutions. GRASP is also an iterative method, since obtained solutions are taken into account when constructing new permutations. One good property of GRASP is that it does not need planarity testing algorithms, but on the other hand, there is a need to search a maximum induced bipartite subgraph of an overlap graph. This construction uses a greedy method since the problem is NP-complete [114]. Moreover, each heuristic that does not perform a planarity test during its execution may fail to notice that a maximum planar subgraph of a planar graph is the graph itself. This is also a shortcoming of 2Phase heuristic. Resende and Ribeiro showed that GRASP gives on average better results for Hamiltonian graphs but for many non-Hamiltonian
graphs J-M performs better [112].

More algorithms and results are known for the weighted MPS problem, where a non-negative weight is associated with each edge and one tries to maximize the sum of the edge weights of a planar subgraph. Algorithms for the weighted maximum planar subgraph problem include a simulate annealing algorithm by Hasan and Osman [61] and heuristics based on the so-called deltahedron heuristic [48, 42, 83, 47]. These algorithms do not perform planarity tests. See also the references given by Liebers [84, pp. 14-23]. The first algorithm with nontrivial performance ratio for the weighted MPS is provided by Calinescu et al. [27].

Another approach to construct large planar subgraphs is to try to optimize existing solutions by some sequence of edge removal and insertion operations. Cimikowski [32] introduced a method for increasing the size of a maximal planar subgraph by removing first an edge and then adding as many new edges as possible. This edge manipulating method increased slightly the size of the planar subgraph. The method was originally given for the thickness problem.

Next we introduce a simulated annealing algorithm (SA) given by Poranen [108] that is a generalization of the edge-change method by Cimikowski [32]. See Algorithm 4.2 for a detailed description of SA. The algorithm gets as input a graph \( G = (V, E) \) and a planar subgraph \( G' = (V, E') \) of \( G \). SA maintains two sets of edges. The first set, \( E_1 \), is initialized as \( E' \). The second set, \( E_2 \), is initialized as \( E \setminus E' \). The first set maintains the edges of a maximum planar subgraph and the second set always contains the remaining edges of the input graph. In what follows, we say shortly that a set of edges is planar, if the graph induced by this edge set is planar.

After initialization, local optimization guided by the SA scheme is applied to increase the size of \( E_1 \) in the following way. An edge \( e_2 \) from \( E_2 \) is chosen randomly. First SA tries to move \( e_2 \) to \( E_1 \) without violating the planarity of \( E_1 \) (this increases the size of \( E_1 \)). If this is not possible, SA randomly chooses an edge \( e_1 \) from \( E_1 \) and tries two swap this edge with \( e_2 \) (the size of \( E_1 \) does not change). If the planarity of \( E_1 \) is again violated, SA checks if it is acceptable (according to the rules of SA) to move \( e_1 \) to \( E_2 \) (this decreases the size of \( E_1 \)).

To check that is it allowed to move an edge from \( E_1 \) to \( E_2 \), we make a random test. A random real \( i, 0 \leq i \leq 1 \), is generated. If \( i \leq e^{-1/t} \), where \( t \) is the current temperature, holds then a move that decreases the size of \( E_1 \) is accepted.

Since an \( n \)-vertex planar graph has at most \( 3n - 6 \) edges, we have added a test to recognize optimal solutions in the while-loops of SA. The exact
4.1. ALGORITHMS FOR MPS

\[ SA(G = (V, E), G' = (V, E')) \]

1. select a cooling ratio \( \alpha \) and an initial temperature \( t_0 \);
   select a frozen temperature \( t_l \) and an equilibrium detection rate \( r \);
   set \( E_1 \leftarrow E', E_2 \leftarrow E \setminus E', t \leftarrow t_0 \) and \( e \leftarrow 0 \);
2. while \( t \geq t_l \) and \((|E_1| < 3|V| - 6)\)
3.   do while \( e \leq r \) and \((|E_1| < 3|V| - 6)\)
4.      do \( e \leftarrow e + 1 \);
5.         randomly select an edge \( e_2 \) from \( E_2 \); \n6.          if \( E_1 \cup \{e_2\} \) is planar
7.              then \( E_1 \leftarrow E_1 \cup \{e_2\} \) and \( E_2 \leftarrow E_2 \setminus \{e_2\} \); \n8.          else randomly select an edge \( e_1 \) from \( E_1 \); \n9.              if \( E_1 \setminus \{e_1\} \cup \{e_2\} \) is planar
10.             then \( E_1 \leftarrow E_1 \setminus \{e_1\} \cup \{e_2\} \); \n11.                \( E_2 \leftarrow E_2 \setminus \{e_2\} \cup \{e_1\} \); \n12.          else generate a random real \( i \), \( 0 \leq i \leq 1 \);
13.             if \( i \leq e^{-1/t} \)
14.                then \( E_1 \leftarrow E_1 \setminus \{e_1\} \); \n15.                  set \( E_2 \leftarrow E_2 \cup \{e_1\} \);
16. return \((V, E_1)\);

Algorithm 4.2: \( SA \) for MPS.
running time of $SA$ depends only on the chosen parameters, namely the initial and frozen temperatures, the cooling ratio and the equilibrium detection rate and the implementation of the planarity test algorithm. In our experiments, given in Chapter 6, the equilibrium rate was set to be equal with the number of edges in the graph, and the other parameters were constants. This leads to an $O(ne)$ running time, although the constant factor is high.

In what follows, we write the name of the algorithm that produces the initial solution for $SA$ to the subscript. For example, if GRE is applied, the corresponding new algorithm is denoted by $SA_{GRE}$.

It is rather easy to modify $SA$ to approximate also the weighted maximum planar subgraph problem by counting the sum of the edge weights in set $E_1$ and to accept always directly such insertions and swaps that increases the sum of the weights. For a move that decreases the sum of the edge weights in $E_1$, let $\delta$ be the decrement. Now to test whether this worsening move is allowed in $SA$, can be tested by the rule

$$
\text{if } i \leq e^{-\delta/i},
$$

where $0 \leq i \leq 1$ is a randomly generated real.

## 4.2 Algorithms for MOPS

Since a tree is outerplanar, MOPS can also be approximated by constructing a spanning tree. This yields an algorithm with performance ratio $1/2$, since the maximum number of edges in an outerplanar graph is at most $2n - 3$ by Theorem 3.9. The same worst case ratio holds also for the ratio of edges in maximal and maximum outerplanar subgraphs [34].

For determining a maximum outerplanar subgraph, a similar algorithm as GRE for MPS can be applied. Only Step 4 needs a change: an outerplanarity test should be used instead of a planarity test. See Algorithm 4.3 for the change. Also, we require that the subgraph $G_0$ given as input is outerplanar.

In what follows, when GRE is applied for MOPS, we assume that outerplanarity test is used instead of planarity test. Whether the planarity or outerplanarity test is applied, should be clear from the context. The same holds also for other algorithms in this work.

Like GRE, also $SA$ can be modified to approximate MOPS. To recognize an outerplanar subgraph of maximum size, the condition statements of the while loops of $SA$ should be modified and planarity tests should be changed.
4.2. ALGORITHMS FOR MOPS

Algorithm 4.3: GRE for MOPS: changes for Algorithm 4.1 to approximate MOPS.

: 
4  \textbf{if} \ (V, E') \text{ is not outerplanar} 
: 

Algorithm 4.4: SA for MOPS: changes for Algorithm 4.2 to approximate MOPS.

: 
2  \textbf{while} \ t \geq t_i \text{ and } (|E_1| < 2|V| - 3) 
3  \quad \textbf{do while} \ e \leq r \text{ and } (|E_1| < 2|V| - 3) 
: 
6  \quad \textbf{if} \ E_1 \cup \{e_2\} \text{ is outerplanar} 
: 
9  \quad \textbf{if} \ E_1 \setminus \{e_1\} \cup \{e_2\} \text{ is outerplanar} 
:
to outerplanarity tests, as shown in Algorithm 4.4. Also, the subgraph $G'$
given as input, should be outerplanar.

In the literature there are no more algorithms that are designed to search
maximal outerplanar subgraphs, but using the result of Theorem 2.6, we can
apply some of the maximal planar subgraph algorithms for this problem. The
general method, described in Algorithm 4.5, was given by Djidjev [41].

\[
\text{MAX-PLAN}(G = (V, E))
\]
1. $G \leftarrow G + ([v], \emptyset)$;
2. find a maximal planar subgraph $G'$ of $G$ with the
   constraint that $v$ and all its adjacent edges belong to $G'$;
3. return $G' - v$;

Algorithm 4.5: Basic structure of the maximal outerplanar subgraph algo-
rithm.

Algorithm MAX-PLAN adds a new vertex $v$ and joins it with all the other
vertices. After this, the algorithm searches for a maximal planar subgraph $G'$
with the constraint that $v$ and all its adjacent edges are added to $G'$. Finally,
algorithm returns $G' - v$, a maximal outerplanar subgraph of $G$. The hardest
part is to satisfy the constraint. It is not clear, if it is possible to modify a
given maximal planar subgraph algorithm to also search maximal outerplanar
subgraphs. However, it is possible to modify the Djidjev’s [41] algorithm.

Kant [73] gives a dynamic programming algorithm to augment an embed-
ded outerplanar graph to the maximal outerplanar graph with the property
that the maximum degree is minimized. The algorithm assumes that an edge
connecting a pair of vertices can be always added in the maximalization phase,
so this approach does not work when constructing a maximal outerplanar sub-
graph for an arbitrary graph.

4.3 Algorithms for thickness

Only one approach to obtain approximations for the thickness problem is
known in the literature: extract maximal planar subgraphs from the origi-
nal graph until the remaining graph is planar. The algorithms differ by the
method used in planar subgraph search from the original graph and by the
optimization of the extracted subgraphs.
Next we describe the basic approach to get approximations for thickness. The extraction method was first studied by Cimikowski [32] and Mutzel et al. [101]. For a detailed description of the extracting method see Algorithm 4.6 (Thick). Step 4 of the algorithm is usually given as “find a maximal/maximum planar subgraph” instead of finding just a planar subgraph.

\[
\text{Thick}(G = (V, E))
\]

1. \(P \leftarrow \emptyset\) and \(t \leftarrow 1\).
2. \(\textbf{while } E \neq \emptyset\)
   3. \(\textbf{do }\) find a planar subgraph \(G' = (V, E_t)\) of \(G\);
   4. \(E \leftarrow E \setminus E_t\);
   5. \(P \leftarrow P \cup \{E_t\}\);
   6. \(t \leftarrow t + 1\);
3. \(\textbf{return } P;\)

Algorithm 4.6: Basic structure of the extraction algorithm for the thickness problem.

The running time of Thick depends highly on the method used in Step 3. It is clear that any algorithm that approximates thickness has at least linear running time, since all edges have to be considered.

Cimikowski [32] applied the maximal planar subgraph algorithm by Cai et al. [25] to extract subgraphs. We will denote this algorithm by HT in Chapter 8. Further, Cimikowski experimented a maximal planar subgraph algorithm by Kant [72] based on vertex addition approach and \(PQ\)-trees. These two methods are also used by Mutzel et al. [101].

Basically, the larger subgraphs we can extract from the input graph, the better approximation for thickness we can obtain. Of course, this method does not necessarily lead to optimal solution, as shown by Mutzel et al. [101] by giving a counterexample. Recently, Kawano and Yamazaki [74] showed that it makes no difference from the theoretical point of view whether we extract maximal planar subgraphs or maximum planar subgraphs from the input graph. Both approaches guarantee only solutions that are \(\Omega(\log n)\) times the optimal for a graph with \(n\) vertices. The graph used in this construction was quite different from the graphs used in any practical applications.

After obtaining an initial solution by Thick, it is possible to try to optimize this solution by edge removal and insertion operations. Poranen [108] gave a simulated annealing algorithm that can be used to decrease the number of
A planar partition of the edges of a nonplanar graph $G = (V, E)$ is a partition of $E$, where each subset induces a planar subgraph. The simulated annealing algorithm for determining the thickness of a graph ($SA$), gets as input a planar partition of the edges. $SA$ makes a local optimization guided by the simulated annealing scheme to decrease the number of planar subgraphs. The number of planar subgraphs can only decrease, worsening the result is not allowed.

The neighborhood structure for a nonplanar graph $G = (V, E)$ is defined as follows. Let $P = \{E_1, E_2, \ldots, E_k\}$ and $P' = \{E'_1, E'_2, \ldots, E'_{k'}\}$ be planar partitions of the edges of $G$ and let $e_i$ be an edge of $E_i$ and $e_j$ an edge of $E_j$, where $i \neq j$. The planar partitions $P$ and $P'$ are neighbors provided that

- $P'$ can be obtained from $P$ by swapping the edges $e_i$ and $e_j$, or
- $P'$ can be obtained from $P$ by moving $e_i$ to $E_j$.

Notice that since we assume partitions to be planar, swapping and moving the edges are not allowed to violate the planarity of the subsets. See Algorithm 4.7 ($SA$) for a detailed description of the method. Algorithm 4.6 can be used to construct the initial solution for $SA$.

The general ideology behind $SA$ is that when the sizes of the subsets of an edge partition have as high a deviation as possible, we have good chances to move edges from the smaller subsets to the larger ones. And finally, when the last edge of a subset is successfully moved to any other subset of the edge partition, we have decreased the number of planar subgraphs and obtained a better approximation for thickness. The empty subset is removed from the partition.

Increasing the deviation of the sizes of the subsets is done by choosing two edges randomly from distinct subsets. $SA$ tries to move the edge from the smaller subset to the larger one without violating the planarity of the subset. If this is not possible, $SA$ tries to swap the edges. If the swap is not possible, $SA$ checks if it is possible to move the edge from the larger set to the smaller one.

To check if this move is possible, we calculate the standard deviations of the sizes of the subsets of the original partition and the partition after the move. For an edge partition $P = \{E_1, E_2, \ldots, E_k\}$, the deviation is calculated from the formula

$$deviation(\{E_1, E_2, \ldots, E_k\}) = \sqrt{\frac{\sum_{i=1}^{k}(|E_i| - \text{AVE})^2}{k}},$$
4.3. ALGORITHMS FOR THICKNESS

\[ SA(G = (V, E), P = \{E_1, E_2, \ldots, E_k\}) \]

1. select a cooling ratio \( \alpha \) and an initial temperature \( t_0 \);
2. select a frozen temperature \( t_f \) and an equilibrium detection rate \( r \);
   \( t \leftarrow t_0 \) and \( e \leftarrow 0 \);
3. while \( t \geq t_f \) and \( |P| > \) lower bound
   do while \( e \leq r \) and \( |P| > \) lower bound
   e \leftarrow e + 1;
4. randomly select two edges \( e_i \) and \( e_j \) from distinct subsets \( E_i \) and \( E_j \) of \( P \) with \( |E_i| \leq |E_j| \);
5. if \( E_j \cup \{e_i\} \) is planar
   then \( E_j \leftarrow E_j \cup \{e_i\} \) and \( E_i \leftarrow E_i \setminus \{e_i\} \);
6. if \( E_i \) is empty
   then remove \( E_i \) from \( P \);
7. else if \( E_j \setminus \{e_j\} \cup \{e_i\} \) and \( E_i \setminus \{e_i\} \cup \{e_j\} \) are planar
   then \( E_j \leftarrow E_j \setminus \{e_j\} \cup \{e_i\} \) and \( E_i \leftarrow E_i \setminus \{e_i\} \cup \{e_j\} \);
8. else let \( \delta \leftarrow \text{deviation}(P) - \text{deviation}(P') \) where \( P' \)
   is the edge partition where \( e_j \) is moved to \( E_i \);
9. if \( \delta \leq 0 \) and \( E_i \cup \{e_j\} \) is planar
   then \( P \leftarrow P' \);
10. else generate a random real \( i, 0 \leq i \leq 1 \);
11. if \( i \leq e^{-\delta/t} \) and \( E_i \cup \{e_j\} \) is planar
    then \( P \leftarrow P' \);
12. \( t \leftarrow \alpha t \);
13. \( e \leftarrow 0 \);
14. return \( P \);

Algorithm 4.7: \( SA \) for determining the thickness of a graph.
where $P_{\text{AVE}}$ is the average size of the subsets.

The deviation can increase, decrease or keep unchanged when we move an edge from the larger set to the smaller one. The deviation is used to test if the move is acceptable as follows. Let $P$ and $P'$ be the partitions before and after a move, respectively, and let $\delta = \text{deviation}(P) - \text{deviation}(P')$. The following simple rule is now applied:

- If $\delta \leq 0$, the move is accepted.
- Otherwise, a random real $i$, $0 \leq i \leq 1$, is generated. If $i \leq e^{-\delta/t}$, where $t$ is the current temperature, the move is accepted.

The running time of $SA$ depends only on the annealing parameters and the implementation of the planarity testing algorithm. If the equilibrium detection rate is set to be equal with the number of edges in the input graph, $SA$ runs in $O(nm)$ time for a graph with $n$ vertices and $m$ edges.

In general, $SA$ can improve the solutions of the other heuristics almost always, as demonstrated in Chapter 8.

Furthermore, a genetic algorithm is implemented [88] for approximating the thickness of a graph. The algorithm is based on dividing the input graph into planar subgraphs and one nonplanar subgraph. After initialization it uses local optimization guided by the genetic operations to planarize the last subset. The number of planar subgraphs does not change dynamically during the execution of the algorithm; if a predefined limit exceeds, the number of planar subgraphs is increased by one and again the last nonplanar subset is planarized by the genetic operations.

4.4 Algorithms for outerthickness

To the best of our knowledge, no algorithms for the outerthickness problem are presented in the literature. However, many thickness algorithms can be modified to approximate outerthickness by performing outerplanarity tests instead of planarity tests. For example, it is enough to search for an outerplanar subgraph in Step 3 of Thick to get an algorithm for approximating the outerthickness of a graph ($O_{\text{thick}}$). The change is given in Algorithm 4.8.

Similarly, it is easy to modify $SA$ to approximate outerthickness by applying lower bounds for outerthickness instead of thickness lower bounds in Steps 2 and 3, and by changing the planarity tests to outerplanarity tests in Steps 6, 10, 13, and 16. Also, an outerplanar partition of the edges should be given as input instead of a planar partition. We omit the algorithm.
4.4. ALGORITHMS FOR OUTERTHICKNESS

Algorithm 4.8: OTHICK: changes for Algorithm 4.6 to approximate the outerthickness of a graph.

; 3 find an outerplanar subgraph $G' = (V, E_t)$ of $G$; ;
Chapter 5

Cactus-tree heuristics

Călinescu et al. [26] gave the first approximation algorithms for MPS and MOPS with nontrivial performance ratios. Two algorithms were given for MPS, a simple linear time algorithm with performance ratio 7/18 and a more complicated algorithm with the performance ratio 4/9. In this chapter we introduce these two algorithms. Then we improve the simpler algorithm. The new algorithm still runs in linear time for bounded degree graphs, produces outerplanar subgraphs, and has at least the same performance ratio as the original algorithm. We conjecture that the new algorithm has at least the same performance ratio as the more complicated algorithm. Then we introduce another new algorithm that produces only planar subgraphs, runs in linear time for bounded degree graphs, and has at least the same performance ratio as the original simple algorithm for MPS. Finally the greedy versions for these algorithms are given. The experimental comparisons of these algorithms are reported in Chapter 6 for MPS, in Chapter 7 for MOPS, and in Chapter 8 for the thickness and outerthickness problems.

5.1 Original heuristic

A triangular structure is a graph whose cycles, if any, are triangles. A triangular cactus is a graph whose cycles, if any, are triangles and such that all edges belong to some cycle. A triangular structure is outerplanar and therefore also planar.

Given a connected graph $G = (V, E)$, the triangular cactus heuristic (cactus-tree, for short) is based on finding first a subgraph $G' = (V, E')$ whose components are triangular cacti. The subgraph is constructed in the following way: $E'$ is initialized to be empty. Triangles having all vertices in different components in $G'$ are searched from $G$ and added to $E'$. After all suitable triangles
are added to $G'$, the subgraph is made connected by adding edges until the resulting graph is a connected triangular structure. See Algorithm 5.1 (CA) for a detailed description of the cactus-tree heuristic. Steps 2 and 3 are called Phase 1 (the construction phase of a triangular cactus) and Steps 4 and 5 are called Phase 2 (the connection phase) of the algorithm.

Algorithm 5.1: CA for MPS and MOPS.

Algorithm CA can be implemented to run in linear time as shown by Călinescu et al. [26], provided that the maximum degree of the graph is bounded by a constant.

**Theorem 5.1** ([26]). CA runs in linear time for bounded-degree graphs.

*Proof.* Suppose that there are $m$ edges in a graph $G$ whose degree is bounded by a constant $d$. The adjacency lists for all vertices contain at most $d$ vertices. All triangles which share a vertex $v_1$ of $G$ can be found by checking the adjacency list of $v_1$ and the adjacency lists of adjacent vertices of $v_1$ in the following way: use an auxiliary array $A$ to mark all neighbors of $v_1$ by 1. If there is a vertex $v_2$ in the adjacency list of $v_1$ and in the adjacency list of $v_2$ there is some vertex $v_3$ having 1 in $A$, we have found a triangle. Therefore, to find out all triangles for a vertex takes $d^2$ time.

We can check that are $v_1$, $v_2$ and $v_3$ in different components in constant time, if the vertices are searched in a fixed order. Divide the vertices of the graph into three sets called new, active and used. In the beginning place all the vertices to the set new. Continue the following operations until there is no new vertices left anymore: if there is no active vertices, choose a vertex from set new and make it active. Choose an active vertex $v_1$ and search all triangles with vertices $v_1$, $v_2$ and $v_3$ such that $v_2$ and $v_3$ are new. Place $v_2$ and $v_3$ to
the active set and add the triangle to the subgraph. When there is no more triangles with two new vertices, place \( v_1 \) to the used set.

The observations above lead to \( O(d^2n) = O(n) \) time algorithm to find a subgraph \( G' \) whose components are triangular cacti.

The edges that connect different components of \( G' \) can be found in \( O(m) \) time by performing one dfs-traversal for \( G' \) and one for \( G \). First, search the connected components of \( G' \) by dfs. To maintain the information on which component vertices belong, an auxiliary array \( C \) can be used. For a vertex \( v \) there is an integer \( k \) in \( C \) to show that \( v \) belong to \( k \)'th component. Also, we use an array \( B \) having an entry for each connected component of \( G_0 \) to keep up the information about the components that are found in the second dfs-traversal. All entries of \( B \) are marked first by 0. Second, we perform a dfs-traversal for \( G \). If we find a tree-edge \( (u, v) \) with the property that \( u \) and \( v \) belong to different components in \( G' \) and the component of \( v \) is marked by 0 (that is, the component of \( v \) is new), then this edge can be added to \( G' \) without violating planarity. When a component is found for the first time, we update the array \( B \) by changing the corresponding entry to 1. Since the connection phase takes \( O(m) \) time (which is \( O(n) \) for bounded degree graphs), the total running time of CA is \( O(n) \) for bounded degree graphs.

To prove that the performance ratio of CA for MPS is \( 7/18 \), it is enough to show first that the performance ratio is at least \( 7/18 \), and then to give a sample graph which verifies that the performance ratio is at most \( 7/18 \).

**Lemma 5.2 ([26]).** The performance ratio of CA for MPS is at least \( 7/18 \).

**Proof.** Assume that \( G = (V, E) \) is a connected graph with at least three vertices. Let \( S_1 \) be the subgraph of \( G \) constructed by CA after Phase 1 and let \( S_2 \) be the same subgraph after Phase 2. The number of edges in \( S_1 \) is the number of triangles in \( S_1 \) multiplied by three and the number of edges in \( S_2 \) is \( |V| - 1 \) added by the number of triangles in \( S_1 \).

Let \( H \) be a plane embedding of a maximum planar subgraph of \( G \) and \( n = |V| \). plane embedding of \( H \) has \( 3n - 6 - t \) edges (by Theorem 3.4, where \( t \geq 0 \) stands for the missing edges of \( H \) to be a triangulated planar graph. Since a plane embedding of a maximal planar graph has \( 2n - 4 \) triangular faces by Theorem 2.4, and each missing edge of \( H \) destroys at most two of these triangular faces, \( H \) has \( 2n - 4 - 2t \) triangular faces.

Let \( k \) be the number of components containing at least one triangle of \( S_1 \), and let \( p_1, p_2, \ldots, p_k \) be the number of triangles in these components. Let \( p = \sum p_i \) be the total number of triangles of \( S_1 \).
If a triangle was not added to $S_1$, it was because two of its vertices were in the same component of $S_1$ (Step 2 of CA). Therefore, at least one edge of this rejected triangle has its vertices in the same component of $S_1$. In general, every triangle in $G$ must have two vertices in the same component of $S_1$. In particular, every triangular face in $H$ must have some two vertices in the same component of $S_1$, and therefore one of its three edges must be in the subgraph of $H$ induced by the vertices in a component of $S_1$. Thus, each triangular face $F$ in $H$ can be associated with an edge $e$ in $F$ whose endpoints are in the same component of $S_1$. But any edge $e$ in $H$ belongs to at most two triangular faces of $H$, so $e$ could have been chosen by at most two triangular faces of $H$. It follows that the number of triangular faces in $H$ is at most twice the number of edges in $H$ whose endpoints are in the same component of $S_1$.

Let $H'$ be the subgraph of $H$ induced by the edges of $H$ whose endpoints are in the same component of $S_1$. Since there is at least one triangle in each component, the number of vertices in the $i$th component of $S_1$ is $2p_i + 1 \geq 3$. Also, since $H'$ is planar, it has at most $\sum_{i=1}^{k}(3(2p_i + 1) - 6) = 6p - 3k$ edges. Combining the previous result, and the fact that the number of triangular faces in $H$ is at most twice the number of edges in $H'$ whose endpoints are in the same component of $S_1$, we have $2(6p - 3k) \geq 2|E(H')| \geq \text{(number of triangular faces in } H) \geq 2n - 4 - 2t$. From this, we can derive a lower bound for the number of triangles $p$ in $S_1$:

$$p \geq \frac{n - 2 - t + 3k}{6} \geq \frac{n - 2 - t}{6}.$$

Since $t \geq 0$, the ratio of the number of edges in $S_2$ and the number of edges in $H$ is at least

$$\frac{n - 1 + (n - 2 - t)/6}{3n - 6 - t} = \frac{7n - 8 - t}{18n - 36 - 6t} \geq 7/18.$$

Next we prove the upper bound for the performance ratio of CA.

**Lemma 5.3.** [26] The performance ratio of CA for MPS is at most $7/18$.

**Proof.** To show that the performance ratio is at most $7/18$, it is enough to construct such a planar graph that CA finds for it a subgraph with the claimed ratio. Let $S_1$ be any connected triangular cactus with $p > 0$ triangles. The number of vertices in $S_1$ is $2p + 1 \geq 3$. Let $S'$ be a triangulation of $S_1$. Since $S'$ is triangulated, it has $2(2p + 1) - 4 = 4p - 2$ triangular faces. For each face of $S'$, add a new vertex in the face and make the vertex adjacent to all vertices
on the boundary of that face. Let $G$ be the new graph so constructed. $G$ is a
triangulated planar graph and it has $(2p + 1) + (4p - 2) = 6p - 1$ vertices and
$3(6p - 1) - 6 = 18p - 9$ edges. For $G$, CA can construct $S_1$ after Phase 1, and
then in Phase 2, CA adds $4p - 2$ new edges to $S_1$ to make the graph connected.
Denote the graph $S_1$ after Phase 2 by $S_2$. $S_2$ has $3p + (4p - 2) = 7p - 2$ edges,
while $G$ has $18p - 9$ edges. The ratio between the numbers of edges is
\[
\frac{7p - 2}{18p - 9}.
\]
The limit of the ratio is $7/18$ as $p$ tends to infinity.

The performance ratio of CA for MPS follows directly by Lemmas 5.2 and 5.3.

**Theorem 5.4 ([26]).** The performance ratio of CA for MPS is $7/18$.

See Figure 5.1 for an illustration of the worst case performance ratio of
CA. The graph is a maximal planar graph with 27 edges, and the algorithm
can find a planar subgraph having 12 edges. The two triangles found in Phase
1 are drawn as bold lines, and the edges added in Phase 2 are drawn as solid
curves, the missing edges are drawn as dashed curves.

The performance ratio of CA for MOPS was not given in the original paper,
but CA is actually an unweighted version of the Berman-Ramayer algorithm
[27] for the weighted version of MOPS. From this result, we get the performance
ratio for CA in the case of outerplanar subgraphs.

**Theorem 5.5 ([27]).** The performance ratio of CA for MOPS is $7/12$. 
If a maximum triangular cactus is searched in Phase 1 instead of the triangular cactus of CA, the performance ratio increases up to $4/9$ [26] for MPS and $2/3$ for MOPS. See Algorithm 5.2 for a description of the algorithm $CA_M$ that searches a maximum triangular cactus in Phase 1.

\[CA_M(G = (V, E))\]

1. Find a maximum triangular cactus $G' = (V, E')$ of $G$.
2. while there is an edge $(v_1, v_2) \in E$ such that $v_1$ and $v_2$ belong to different components in $G'$
   do $E' \leftarrow E' \cup \{(v_1, v_2)\}$;
3. return $(V, E')$;

Algorithm 5.2: $CA_M$ for MPS and MOPS.

All known algorithms for finding the maximum triangular structure are very complicated. The method given by Călinescu et al. [26] was based on reducing the problem of finding maximum triangular structure to so called graphic matroid parity problem [86] and then solving it with an algorithm by Gabow and Stallman [49]. This method leads to running time $O(m^{3/2} \log^6 n)$. The following theorem formulates the properties of $CA_M$.

**Theorem 5.6 ([26]).** The performance ratio of $CA_M$ for MPS is $4/9$ and for MOPS $2/3$. $CA_M$ runs in $O(m^{3/2} \log^6 n)$ for a graph with $m$ edges and $n$ vertices.

### 5.2 Modified heuristics

When a triangle is found in CA, it always connects three vertices from different components of the subgraph. It is easy to see that all the vertices of a triangle need not belong to different components. It is enough to have two vertices $v_1$ and $v_2$ joined by an edge $(v_1, v_2)$ in one component and the third vertex $v_3$ in an other component forming a triangle $(v_1, v_2, v_3)$. When triangles are added using this principle whenever it is possible, and otherwise demanding that the vertices of the triangle belong to different components, the planarity is not violated. If any triangle is added with this new principle, the resulting graph is not any more a triangular structure. To guarantee that the constructed subgraph is also outerplanar, it is necessary and sufficient to demand that $(v_1, v_2)$ belongs to at most two triangles at the same time. The algorithm
applying this restriction and producing outerplanar subgraphs, is denoted by CA1, and the algorithm without the restriction is denoted by CA2. Properties of CA1 are studied first. The exact description of CA1 is given in Algorithm 5.3.

Algorithm 5.3: CA1 for MPS and MOPS.

The observations that led to CA1 were inspired by the recognition algorithm for the maximal outerplanar graphs given by Mitchell [96]. The algorithm (described in Section 2.3) was based on extracting degree 2 vertices from the graph. In CA1, vertices of degree 2 are added to an outerplanar subgraph.

In Figure 5.2 there is an illustration of the behavior of CA and CA1 for graph cimi-g4 [33] (see also Table 6.1 in Chapter 6 for more information of the graph). The found triangles are enumerated in the order they are found. This order depends on the implementation of the algorithm and the representation of the graph. CA finds first four triangles and then it connects one remaining vertex with the rest of the subgraph. The planar subgraph contains 13 edges. CA1 finds first one triangle, then it adds 5 triangles that increase the number of edges by 2 and finally a triangle with three new edges is added. The size of the planar subgraph is now 16.

Next we show that CA1 can be implemented to run in linear time, if the maximum degree of the input graph is bounded by a constant.

**Lemma 5.7.** CA1 runs in linear time for bounded-degree graphs.
Since it is enough to consider (linear for bounded degree graphs, as shown in Theorem 5.1. the degree of the graph is bounded. The total time for all other operations is steps where a triangle connecting two vertices from the same component and one vertex from another component takes in total linear time provided that

Suppose that there are \( \text{CA1} \) for MPS is at least 7

To see that

Each time when an edge \((v_1, v_2)\) is considered in Step 2, it takes at most \( d^2 \) time to check adjacency lists of \( v_1 \) and \( v_2 \) to recognize a triangle. Since it is enough to consider \((v_1, v_2)\) only once in Step 2, CA1 runs in time \( O(n) \) for bounded degree graphs.

To show that the performance ratio of CA1 for MPS is at least 7/18 and
for MOPS at least $7/12$, the proof of Lemma 5.2 can be applied directly. We only outline the proof.

**Lemma 5.8.** The performance ratio of CA1 for MPS is at least $7/18$ and for MOPS at least $7/12$.

*Proof.* It is enough to notice that the proof of Lemma 5.2 for CA can be applied directly for CA1. Let $G_{CA}$ and $G_{CA1}$ be the planar subgraphs produced by CA and CA1 after Phase 1, respectively. A triangle was not added to $G_{CA}$ if two of its vertices were in the same component. The same also holds for $G_{CA1}$: there is no triangle in the input graph with its vertices in different components in $G_{CA1}$. The original proof was based on this observation, and therefore it follows that CA1 has at least the same performance ratio as CA. The lower bound of the performance ratio for MOPS follows directly from Theorem 5.5.

The upper bound given in Lemma 5.3 for the performance ratio of CA cannot be applied for CA1 (for graph in Figure 5.1, CA1 can find a subgraph with 17 or 19 edges) but it is clear that the ratio cannot exceed $1/2$, as shown by the following constructive proof.

**Lemma 5.9.** The performance ratio of CA1 for MPS is at most $1/2$.

*Proof.* Let $G$ be a $n \times n$ grid graph with $n \geq 2$. Now each side of the grid contains $n$ vertices and $G$ has in total $n^2$ vertices and $2n^2 - 2n$ edges. Since $G$ is planar, the maximum planar subgraph is the graph itself. CA1 finds a planar subgraph with $n^2 - 1$ edges by constructing a spanning tree of $G$. The ratio between the number of edges found by CA1 and the number of edges in $G$ is

$$\frac{n^2 - 1}{2n^2 - 2n}.$$ 

The limit of the ratio is $1/2$ as $n$ tends to infinity.

We give next a sample graph which shows that the performance ratio of CA1 for MOPS is at most $2/3$.

**Lemma 5.10.** The performance ratio of CA1 for MOPS is at most $2/3$.

*Proof.* Let $G$ be a $2 \times n$ grid graph. Now $G$ has in total $2n$ vertices and $3n - 2$ edges. Since $G$ is outerplanar, the maximum outerplanar subgraph is the graph itself. CA1 finds a outerplanar subgraph with $2n - 1$ edges by constructing
a spanning tree of $G$. The ratio between the number of edges found by CA1 and the number of edges in $G$ is

$$\frac{2n - 1}{3n - 2}.$$

The limit of the ratio is $2/3$ as $n$ tends to infinity.

Now we can conclude the properties of CA1 for MPS and MOPS.

**Theorem 5.11.** The performance ratio of CA1 for MPS is at least $7/18$ and at most $1/2$. The performance ratio of CA1 for MOPS is at least $7/12$ and at most $2/3$. The algorithm runs in linear time for bounded-degree graphs.

See Figure 5.3 for an illustration of the worst case performance ratio of CA1 for MPS and MOPS. The left graph is a planar $4 \times 4$ grid graph and the right graph is an outerplanar $2 \times 7$ grid graph. For the left graph CA1 finds an approximation with 15 edges and for the right graph with 13 edges.

There is a gap between the lower and upper bounds of the performance ratios of CA1 for MPS and MOPS, and the exact performance ratio is left open. One way to confirm or refute that the performance ratio is at least $4/9$ for MPS is to show that a subgraph produced by CA1 has always at least the same number of edges as a maximum triangular structure of a graph. We give a conjecture for the performance ratio of CA1 for MPS and MOPS. The computational experiments reported in Section 6.5 support the conjecture.

**Conjecture 5.12.** The performance ratio of CA1 for MPS is at least $4/9$ and for MOPS exactly $2/3$. 

60
Next we study CA2. From the condition in the inner while loop of CA1 it follows that in the end of the algorithm an edge of $G'$ belongs to at most two triangles. It is not necessary to demand that one edge belongs to at most two triangles at the same time in the case of planar subgraphs. The restriction “$(v_1, v_2)$ belongs to one triangle in $E'$” of the while loop of CA1 can be changed to “edge $(v_1, v_2)$ belongs to $E'$“. This observation leads to Algorithm CA2. Now outerplanarity is violated if in the end of the algorithm any edge belongs to more than two triangles (a forbidden subgraph $K_{3,2}$ is created). The subgraph remains planar. In Figure 5.2, there is an illustration of the behavior of CA2 for graph cimi-g4 [33]. Notice that the edge $(f, i)$ belongs to three triangles and hence the outerplanarity is violated. The planar subgraph found by CA2 contains 16 edges.

\begin{verbatim}
CA2(G = (V, E))
1  E' ← ∅;
2  repeat while there is a triangle (v1, v2, v3) in G such that (v1, v2) belongs to E' and v3 to a different component of (V, E')
   3     do E' ← E' ∪ {(v2, v3), (v3, v1)};
   4     if there is a triangle (v1, v2, v3) in G such that
   5        v1, v2 and v3 belong to different components in (V, E')
   6        then E' ← E' ∪ {(v1, v2), (v2, v3), (v3, v1)};
   7     until the number of edges in E' increases during the loop
7  while there is an edge (v1, v2) ∈ E such that v1 and v2 belong to
different components in (V, E')
8     do E' ← E' ∪ {(v1, v2)};
9  return (V, E');
\end{verbatim}

Algorithm 5.4: CA2 for MPS.

The linear running time of CA2 for bounded degree graphs follows directly from the observations given in Theorem 5.1 for CA and Lemma 5.7 for CA1. The bounds for the performance ratio of CA2 are the same as they are for CA1. The following theorem concludes the properties of CA2.

**Theorem 5.13.** The performance ratio of CA2 for MPS is at least $\frac{7}{18}$ and at most $\frac{1}{2}$, and the algorithm runs in linear time for bounded-degree graphs.

Next we give two simple corollaries that describe the differences of CA, $CA_M$, CA1 and CA2. Corollary 5.14 is illustrated in Figure 5.4 and Corollary 5.15 is illustrated in Figure 5.5.
Corollary 5.14. There exist graphs for which the limit of the ratio of the solutions of CA1 (CA2) and CA (CA_M) is 4/3.

Corollary 5.15. There exist graphs for which the limit of the ratio of the solutions of CA2 and CA1 (CA, CA_M) is 2.

The final difference between CA1, CA2, and CA is that CA1 and CA2 recognize maximal outerplanar graphs. This follows directly from Definition 2.5, which gave a recursive method to construct a maximal outerplanar graph.

Corollary 5.16. CA1 and CA2 recognize maximal outerplanar graphs.

CA, CA1, and CA2 can be made greedy by giving the subgraph constructed in Phase 1 as input to GRE. These greedy versions are denoted by GCA, GCA1, and GCA2. Since GRE connects the subgraph, at least the same number of edges is added as in Phase 2 of CA, CA1, and CA2. Therefore, GCA, GCA1, and GCA2 have the same performance ratios as CA, CA1, and CA2, respectively.
Chapter 6

MPS experiments

In this chapter, different algorithms for MPS are compared. The first section describes how the new algorithms are implemented and how the solutions for algorithms from the literature are obtained. The second section introduces the test graph suite used in the experiments. In Section 6.3 we describe the used comparison methods. The fourth section covers the parameter detection for the simulated annealing algorithm. Sections 6.5, 6.6, and 6.7 cover the experiments for MPS. The last section summarizes the comparison results. The comparison of MOPS algorithms is given in Chapter 7, but most of the material given in Sections 6.1 – 6.5 are common for both problems. The comparison methods described in Section 6.3 are also applied for the thickness and outerthickness problems in Chapter 8.

6.1 The experimented algorithms and their implementation

We implemented the following algorithms for MPS: CA, CA1, CA2, and their greedy versions GCA, GCA1, GCA2 (described in Chapter 5) with the pure greedy algorithm GRE (described in Chapter 4).

The simulated annealing algorithm was tested with the empty set initialization ($SA_E$) and CA1 initialization ($SA_{CA1}$). CA1 was chosen to construct the initial solution instead of CA or CA2, because it produces better solutions than CA in the same computation time (see Section 6.5), and the solutions are also outerplanar. $SA_E$ solutions for MPS were taken from our earlier experiments [109].

Algorithms CA, CA1, and CA2 were randomized by choosing the edges and start vertices always randomly. The greedy heuristics were randomized by
handling the edges in a random order.

All algorithms were written in C++ programming language and their source codes are available in apptopinv program [107]. For the planarity test apptopinv uses LEDA 4.3 [81]. All test runs were executed on a computer (1992 BogoMips) which has one AMD Athlon 1GHz processor with 256 Megabytes memory running under Linux Mandrake 8.1.

We also compared the solution quality and running time of MPS algorithms against the branch-and-cut heuristic by Junger and Mutzel (J-M) [70] as well as GRASP and 2Phase heuristics implemented by Resende and Ribeiro [112]. For GRASP we used the original source code, which was compiled with the f77 compiler. For 2Phase and J-M, the solutions are taken from the reported results.

CA, CA1, CA2, GRE, GCA, GCA1, and GCA2 were repeated 100 times for graphs with no more than 100 edges and 25 times for the other graphs. Greedy heuristics were not performed for graphs having over 15000 edges. \( SA_E \) and \( SA_{CA1} \) were repeated 20 times for graphs with no more than 373 edges (except that for g13 only 10 were performed), and 10 times for graphs with no more than 814 edges and 5 times for the remaining graphs. \( SA_E \) and \( SA_{CA1} \) were not performed for graphs with more than 1507 edges.

Algorithms CA, CA1 and CA2 are compared in Section 6.5, and GCA, GCA1, GCA2, and GRE in Section 6.6. The comparison of \( SA_E \), \( SA_{CA1} \), GRASP, J-M, and 2Phase is given in Section 6.7.

More detailed solution statistics (number of repeats (rep.), average running times (ave t.), worst (worst), average solutions (ave), and best (best)) for CA, CA1, CA2, GCA, GCA1, GCA2, \( SA_E \), and \( SA_{CA1} \), can be found in Appendix A. Also some solution quality and running time statistics for GRASP, J-M, and 2Phase are reported in the appendix.

### 6.2 Test graph suite

Since MPS is a well studied optimization problem, there already exists a large collection of suitable test graphs. We mainly used the same test graph suite as Resende and Ribeiro [112].

The test graph set used in this work contains 50 graphs. Statistics for the graphs are given in Tables 6.1, 6.2, and 6.3. For graphs in Table 6.1 and 6.2, we have listed the name of the graph (name), the number of vertices (\(|V|\)), and

---

### Table 6.1: Statistics for test graph set I.

| Graph  | $|V|$ | $|E|$ | MPS ub | MPS sol’n | MPS ref. | MOPS ub | MOPS sol’n | MOPS ref. |
|--------|-----|-----|--------|----------|----------|---------|------------|----------|
| cimi-g1 | 10  | 21  | 19     | 19       | [31, 112, 100, 109] | 17      | 15         | †        |
| cimi-g2 | 60  | 166 | 165*   | 165      | [112, 100, 109]     | 117*    | 117        | †        |
| cimi-g3 | 28  | 75  | 73*    | 73       | [31, 112, 100, 109] | 53      | 51         | †        |
| cimi-g4 | 10  | 22  | 20*    | 20       | [112, 100, 109]     | 17      | 16         | †        |
| cimi-g5 | 45  | 85  | 82*    | 82       | [112, 100, 109]     | 82      | 79         | †        |
| cimi-g6 | 43  | 63  | 59*    | 59       | [112, 100, 109]     | 59      | 52         | †        |
| g10    | 25  | 71  | 69*    | 69       | [112, 70, 109]      | 47*     | 47         | †        |
| g11    | 25  | 72  | 69*    | 69       | [112, 70, 109]      | 47*     | 47         | †        |
| g12    | 25  | 90  | 69*    | 69       | [109]               | 47*     | 47         | †        |
| g13    | 50  | 367 | 144    | 137      | [109]               | 97*     | 97         | †        |
| g14    | 50  | 491 | 144    | 144      | †                   | 97*     | 97         | †        |
| g15    | 50  | 582 | 144    | 144      | [112, 109]          | 97*     | 97         | †        |
| g16    | 100 | 451 | 294    | 208      | †                   | 197     | 171        | †        |
| g17    | 100 | 742 | 294    | 247      | †                   | 197     | 193        | †        |
| g18    | 100 | 922 | 294    | 257      | [109]               | 197     | 196        | †        |
| g19    | 150 | 1064| 444    | 339      | †                   | 297     | 275        | †        |
| r0     | 1000| 14985| 2994  | 2005     | †                   | 1997    | 1791       | †        |
| r1     | 1000| 49950| 2994  | 1997     | †                   | 1997    | 1997       | †        |
| r9     | 1000| 449550| 2994 | 1997     | †                   | 1997    | 1997       | †        |
| crack  | 10240| 30380| 30380*| 30380    | †                   | 20437   | 20067      | †        |

* Upper bound is known to be optimal.
† Solution is obtained in this work.
### Table 6.2: Statistics for test graph set II.

| graph | $|V|$ | $|E|$ | ub sol'n | ref. | ub sol'n | ref. |
|-------|------|------|---------|------|---------|------|
| rg100.1 | 100 | 261 | 260 | 170 | [109] | 197 | 145 |
| rg100.2 | 100 | 271 | 270 | 174 | [109] | 197 | 148 |
| rg100.3 | 100 | 297 | 294 | 179 | [109] | 197 | 149 |
| rg100.4 | 100 | 334 | 294 | 186 | [109] | 197 | 156 |
| rg100.5 | 100 | 373 | 294 | 198 | y | 197 | 164 |
| rg150.1 | 150 | 387 | 386 | 243 | [109] | 297 | 208 |
| rg150.2 | 150 | 402 | 401 | 241 | [109] | 297 | 210 |
| rg150.3 | 150 | 453 | 444 | 254 | [109] | 297 | 213 |
| rg150.4 | 150 | 473 | 444 | 259 | y | 297 | 219 |
| rg150.5 | 150 | 481 | 444 | 262 | y | 297 | 219 |
| rg200.1 | 200 | 514 | 513 | 308 | [109] | 397 | 268 |
| rg200.2 | 200 | 519 | 518 | 307 | [109] | 397 | 263 |
| rg200.3 | 200 | 644 | 594 | 334 | [109] | 397 | 285 |
| rg200.4 | 200 | 684 | 594 | 341 | y | 397 | 290 |
| rg200.5 | 200 | 701 | 594 | 344 | y | 397 | 289 |
| rg300.1 | 300 | 814 | 813 | 453 | [109] | 597 | 394 |
| rg300.2 | 300 | 1159 | 894 | 506 | y | 597 | 423 |
| rg300.3 | 300 | 1176 | 894 | 499 | y | 597 | 417 |
| rg300.4 | 300 | 1474 | 894 | 540 | y | 597 | 437 |
| rg300.5 | 300 | 1507 | 894 | 541 | y | 597 | 443 |

* Upper bound is known to be optimal.
† Solution is obtained in this work.

### Table 6.3: More graphs for test graph set II.

<table>
<thead>
<tr>
<th>graph data</th>
<th>MPS</th>
<th>MOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>graph</td>
<td>$</td>
<td>V</td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>tg100.1</td>
<td>100</td>
<td>304</td>
</tr>
<tr>
<td>tg100.3</td>
<td>100</td>
<td>324</td>
</tr>
<tr>
<td>tg100.5</td>
<td>100</td>
<td>344</td>
</tr>
<tr>
<td>tg100.7</td>
<td>100</td>
<td>364</td>
</tr>
<tr>
<td>tg100.9</td>
<td>100</td>
<td>384</td>
</tr>
<tr>
<td>tg200.1</td>
<td>200</td>
<td>604</td>
</tr>
<tr>
<td>tg200.3</td>
<td>200</td>
<td>624</td>
</tr>
<tr>
<td>tg200.5</td>
<td>200</td>
<td>644</td>
</tr>
<tr>
<td>tg200.7</td>
<td>200</td>
<td>664</td>
</tr>
<tr>
<td>tg200.9</td>
<td>200</td>
<td>684</td>
</tr>
</tbody>
</table>

* Upper bound is known to be optimal.
† Solution is obtained in this work.
6.2. TEST GRAPH SUITE

edges (|E|). Then we give the upper bound for MPS (ub), and the best known solution from the literature or from this work (sol’n). If the upper bound is known to be optimal, it is marked with a star (*). Also a reference for the best known solution is given (ref.) or if the solution is obtained in this work, it is marked with a dag (†). Similar data is given also for MOPS. Table 6.3 introduces more test graphs for MPS and MOPS, but the graphs are different for the two problems.

For graphs with unknown optima, the upper bounds for MPS and MOPS were obtained by applying Euler’s polyhedron formula (see Theorems 3.4 and 3.9). If the number of the edges was less than the bound obtained from the formula, the upper bound is the number of edges decreased by one for nonplanar and non-outerplanar graphs.

The first six graphs in Table 6.1 (cimi-g1 – cimi-g6) were taken from the experiments of Cimikowski [31]. These graphs have relevance to applications or have their origin in other research papers. Graphs cimi-g4, cimi-g5, and cimi-g6 were introduced originally in [67], [72], and [123], respectively. Graph cimi-g6 does not contain any triangles.

Graphs g10 – g19 in Table 6.1 are Hamiltonian graphs constructed by Goldsmith and Takvorian [53]. The original report also contains graphs g1 – g9, but those graphs contain at most 70 edges, and almost every algorithm given in this work always found an optimal solution for them in our preliminary tests. Therefore, we did not include those graphs any more in the test graph suite. Computational experiments for graphs g1 – g9 can be found in [53, 112].

Graphs r0, r1, and r9 in Table 6.1 are random graphs and they were generated using LEDA’s [81] random graph generator. The last graph in Table 6.1, crack, is a real world graph taken from the experiments of Petit [105] for the optimal linear arrangement problem. Crack is planar but not outerplanar.

The graphs in Table 6.2 are random graphs (rg100.1 – rg300.5) with the number of vertices varying between 100 and 300 and the number of edges varying between 261 and 1507. These graphs were generated by Cimikowski [31], who also used smaller random graphs with 50 or 75 vertices. However, they are not included in our test graph suite. Computational results for these graphs can be found in [53, 112].

Table 6.3 contains graphs with a planar subgraph of maximum size (tg100.1 – tg200.9) and graphs with an outerplanar subgraph of maximum size (o100.1

---

3The graph can be downloaded from http://www.lsi.upc.es/~jpetit/MinLA/Experiments/ (Feb. 23, 2004).
Graphs \( \text{tg100.1} - \text{tg200.10} \) were generated by Cimikowski [31] by first constructing a random maximal planar graph with 100 vertices, and then adding \( k \cdot 10 \), where \( 1 \leq k \leq 10 \), new edges randomly. Graphs \( \text{tg200.1} - \text{tg200.10} \) were constructed in the same way. In our experiments we only used every second of these graphs. The original numbering for these graphs is still used since it helps the comparison with the earlier works.

Graphs \( \text{o100.1} - \text{o200.5} \) were generated for this work in the following way. First, a cycle with 100 (200) vertices was constructed. Then, two distinct vertices \( u \) and \( v \) from the cycle were chosen randomly. If there was not yet edge \((u, v)\) in the graph, and the edge did not violate the outerplanarity, the \((u, v)\) was added to the graph. The second step was repeated until the graph was a maximal outerplanar graph. Then \( k \cdot 100, 1 \leq k \leq 5 \), new edges were added randomly to obtain graph \( \text{o100.k} (\text{o200.k}) \). We tested the increments of 10 and 50 edges in our preliminary tests, but for those graphs, the algorithms CA1, \( SA_E \) and \( SA_{CA1} \) found always the optimal solutions. For the experiments of MOPS, graphs \( \text{tg100.1} - \text{tg200.9} \) were replaced with the graphs \( \text{o100.1} - \text{o200.5} \).

### 6.3 Comparison measures

To compare the algorithms, we have concentrated on studying the running time and solution quality differences of the algorithms. Methods and measures for the experimental analysis of the heuristics used in this work are mainly given by Golden and Stewart [52].

Running times for the algorithms were obtained by running all test runs as background processes and performing the linux `time` command to get the total running time. Finally, this total running time was divided by the number of repeats to get the average running time of one run.

For each algorithm, it is easy to select the best solution found from all repeats for a test instance. Then we can count the total number of the best solutions found for each algorithm, that is, an algorithm is awarded 1 point, if it obtained among all algorithms the best or tied best solution for a test instance.

Another measure is the total number of points for an algorithm: a heuristic is awarded \( p \) points, if it obtained the \( p \)th best solution for an instance. The average rank of an algorithm is the total number of points divided by

---

the number of test instances. Since all the algorithms are randomized, it is possible that the solution quality varies highly between distinct runs. Therefore, we have also calculated the range of the solutions, that is, the maximum difference between the worst and the best solutions over all test instances for an algorithm.

Another feature that describes the solution quality is the distance from the optimum. We have calculated the worst case ratio of the worst found solution and the optimal solution (or the best known solution, if the optima is unknown) for each algorithm.

In general, if the differences between the solution quality of experimented heuristics is not clear, it is possible to apply statistical methods in comparison. In this work it was almost always very easy to recognize the best algorithm, and hence, no statistical methods are applied.

6.4 Parameter detection for SA

To find good cooling parameters for \( S_A^E \) and \( S_A^{CA_1} \), we tested how the initial temperature affects the solution quality. The first preliminary test was performed for graph \( rg_{100.1} \) with 261 edges. We set the cooling ratio to 0.999, and the equilibrium detection rate to be equal with the number of edges in the graph. The initial temperature varied between 0.95 and 0.10 with steps of 0.05. The frozen temperature was set to be so low that the geometric decrement was performed as many times as it is performed with the initial temperature 0.25 and frozen temperature 0.20. The same test was also made for \( S_A^{CA} \) (simulated annealing with the initial solution constructed by CA), although we did not make any further experiments with this heuristic.

We performed only one run for each initial temperature. Solutions for MPS varied between 147 and 165 for \( S_A^E \), between 146 and 167 for \( S_A^{CA} \), and between 147 and 167 for \( S_A^{CA_1} \). The best solutions for all these heuristics were obtained between temperatures 0.40 and 0.20. The same also holds for MOPS: the best solutions were found within the same temperature interval, and the solutions varied between 118 and 142 for \( S_A^E \), between 120 and 143 for \( S_A^{CA} \), and between 124 and 144 for \( S_A^{CA_1} \). The solutions of this preliminary test are illustrated in Figure 6.1; the solutions for MPS are on the left and the solutions for MOPS are on the right.

To recognize a good temperature interval more accurately, we traced for the second test \( S_A^E \) and \( S_A^{CA_1} \) within temperature intervals 0.35 – 0.30, 0.30 – 0.25 and 0.25 – 0.20 for graphs \( rg_{100.1} \) and \( rg_{300.5} \) with 261 and 1507 edges, respectively. In Figure 6.2 there are traces of \( S_A^E \) and \( S_A^{CA_1} \) for \( rg_{100.1} \) and
in Figure 6.3 for rg300.5 for MPS. Figures 6.4 and 6.5 have similar traces for MOPS. For rg100.1, there were no significant differences between the solution quality and the initial temperature. Solutions with the initial temperature 0.3 and 0.25 were only slightly better than with the initial temperature 0.35. For rg300.5, the solution quality correlate highly with the initial solution and with the initial temperature. $SA_E$ converged to better approximations with the initial temperatures 0.30 and 0.25 for both problems, but $SA_{CA1}$ found the best approximation only with the initial temperature 0.25. For MOPS, there was also a remarkable worsening in the solution quality after the start of the annealing process for rg300.5, as shown in Figure 6.5. The explanation is that the initial solution produced by CA1 is rather good, and with the higher temperatures the inferior solutions are accepted too often causing the overall worsening.

As a result of our preliminary tests, the following parameters were chosen for $SA_E$ and $SA_{CA}$: $\alpha = 0.999$, $t_0 = 0.25$ and $t_t = 0.20$. The equilibrium detection rate was set to be the number of edges in the input graph. The only exception was that the $SA_E$ solutions were computed with the initial temperature $t_0 = 0.3$ in our earlier experiments [109]. The increment of the initial temperature for $SA_E$ should slightly improve the solution quality at the cost of a longer running time.

Of course, we cannot be sure if the chosen parameters for the cooling process are optimal. Moreover, the good parameters for different graphs could vary heavily.
### Table 6.4: The best found solutions for the test graph set I.

<table>
<thead>
<tr>
<th>graph</th>
<th>CA</th>
<th>CA1</th>
<th>CA2</th>
<th>GRE</th>
<th>GCA</th>
<th>GCA1</th>
<th>GCA2</th>
<th>$SA_E$</th>
<th>$SA_{CA1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>cimi-g1</td>
<td>11</td>
<td>13</td>
<td>13</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>cimi-g2</td>
<td>88</td>
<td>117</td>
<td>117</td>
<td>165</td>
<td>165</td>
<td>165</td>
<td>165</td>
<td>165</td>
<td>165</td>
</tr>
<tr>
<td>cimi-g3</td>
<td>38</td>
<td>49</td>
<td>49</td>
<td>73</td>
<td>73</td>
<td>73</td>
<td>73</td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>cimi-g4</td>
<td>13</td>
<td>16</td>
<td>16</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>cimi-g5</td>
<td>59</td>
<td>73</td>
<td>73</td>
<td>82</td>
<td>82</td>
<td>82</td>
<td>82</td>
<td>82</td>
<td>82</td>
</tr>
<tr>
<td>cimi-g6</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>59</td>
<td>59</td>
<td>59</td>
<td>59</td>
<td>59</td>
<td>59</td>
</tr>
<tr>
<td>g10</td>
<td>36</td>
<td>47</td>
<td>47</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
</tr>
<tr>
<td>g11</td>
<td>36</td>
<td>47</td>
<td>47</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
</tr>
<tr>
<td>g12</td>
<td>36</td>
<td>47</td>
<td>47</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>69</td>
</tr>
<tr>
<td>g13</td>
<td>73</td>
<td>97</td>
<td>97</td>
<td>119</td>
<td>120</td>
<td>128</td>
<td>125</td>
<td>137</td>
<td>137</td>
</tr>
<tr>
<td>g14</td>
<td>73</td>
<td>97</td>
<td>97</td>
<td>127</td>
<td>132</td>
<td>134</td>
<td>133</td>
<td>143</td>
<td>144</td>
</tr>
<tr>
<td>g15</td>
<td>73</td>
<td>97</td>
<td>97</td>
<td>133</td>
<td>136</td>
<td>138</td>
<td>137</td>
<td>144</td>
<td>144</td>
</tr>
<tr>
<td>g16</td>
<td>137</td>
<td>162</td>
<td>167</td>
<td>175</td>
<td>193</td>
<td>200</td>
<td>196</td>
<td>206</td>
<td>208</td>
</tr>
<tr>
<td>g17</td>
<td>147</td>
<td>194</td>
<td>196</td>
<td>196</td>
<td>224</td>
<td>237</td>
<td>229</td>
<td>244</td>
<td>247</td>
</tr>
<tr>
<td>g18</td>
<td>147</td>
<td>197</td>
<td>197</td>
<td>210</td>
<td>230</td>
<td>244</td>
<td>239</td>
<td>257</td>
<td>253</td>
</tr>
<tr>
<td>g19</td>
<td>218</td>
<td>274</td>
<td>283</td>
<td>266</td>
<td>305</td>
<td>326</td>
<td>323</td>
<td>331</td>
<td>339</td>
</tr>
<tr>
<td>r0</td>
<td>1475</td>
<td>1766</td>
<td>1842</td>
<td>1400</td>
<td>1827</td>
<td>1975</td>
<td>2005</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>r1</td>
<td>1497</td>
<td>1997</td>
<td>1997</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>r9</td>
<td>1498</td>
<td>1997</td>
<td>1997</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>crack</td>
<td>15107</td>
<td>20067</td>
<td>20086</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

* The algorithm is not applied for this graph.
Table 6.5: The best found solutions for the test graph set II.

<table>
<thead>
<tr>
<th>graph</th>
<th>CA</th>
<th>CA1</th>
<th>CA2</th>
<th>GCA</th>
<th>GCA1</th>
<th>GCA2</th>
<th>$SA_E$</th>
<th>$SA_{CA1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>rg100.1</td>
<td>119</td>
<td>124</td>
<td>125</td>
<td>150</td>
<td>157</td>
<td>157</td>
<td>170</td>
<td>169</td>
</tr>
<tr>
<td>rg100.2</td>
<td>118</td>
<td>125</td>
<td>127</td>
<td>151</td>
<td>160</td>
<td>162</td>
<td>174</td>
<td>173</td>
</tr>
<tr>
<td>rg100.3</td>
<td>120</td>
<td>128</td>
<td>128</td>
<td>153</td>
<td>163</td>
<td>163</td>
<td>179</td>
<td>179</td>
</tr>
<tr>
<td>rg100.4</td>
<td>126</td>
<td>136</td>
<td>140</td>
<td>155</td>
<td>172</td>
<td>175</td>
<td>186</td>
<td>186</td>
</tr>
<tr>
<td>rg100.5</td>
<td>137</td>
<td>153</td>
<td>153</td>
<td>162</td>
<td>186</td>
<td>186</td>
<td>197</td>
<td>198</td>
</tr>
<tr>
<td>rg150.1</td>
<td>171</td>
<td>174</td>
<td>175</td>
<td>214</td>
<td>222</td>
<td>223</td>
<td>243</td>
<td>241</td>
</tr>
<tr>
<td>rg150.2</td>
<td>176</td>
<td>182</td>
<td>182</td>
<td>213</td>
<td>224</td>
<td>225</td>
<td>241</td>
<td>241</td>
</tr>
<tr>
<td>rg150.3</td>
<td>171</td>
<td>179</td>
<td>179</td>
<td>221</td>
<td>237</td>
<td>230</td>
<td>254</td>
<td>252</td>
</tr>
<tr>
<td>rg150.4</td>
<td>180</td>
<td>190</td>
<td>190</td>
<td>217</td>
<td>236</td>
<td>241</td>
<td>258</td>
<td>259</td>
</tr>
<tr>
<td>rg150.5</td>
<td>178</td>
<td>185</td>
<td>185</td>
<td>221</td>
<td>237</td>
<td>236</td>
<td>261</td>
<td>262</td>
</tr>
<tr>
<td>rg200.1</td>
<td>222</td>
<td>227</td>
<td>227</td>
<td>270</td>
<td>278</td>
<td>283</td>
<td>308</td>
<td>306</td>
</tr>
<tr>
<td>rg200.2</td>
<td>216</td>
<td>219</td>
<td>219</td>
<td>268</td>
<td>274</td>
<td>277</td>
<td>307</td>
<td>301</td>
</tr>
<tr>
<td>rg200.3</td>
<td>235</td>
<td>243</td>
<td>244</td>
<td>280</td>
<td>303</td>
<td>306</td>
<td>334</td>
<td>332</td>
</tr>
<tr>
<td>rg200.4</td>
<td>237</td>
<td>254</td>
<td>261</td>
<td>282</td>
<td>308</td>
<td>317</td>
<td>336</td>
<td>341</td>
</tr>
<tr>
<td>rg200.5</td>
<td>235</td>
<td>251</td>
<td>253</td>
<td>285</td>
<td>311</td>
<td>314</td>
<td>342</td>
<td>344</td>
</tr>
<tr>
<td>rg300.1</td>
<td>324</td>
<td>330</td>
<td>330</td>
<td>390</td>
<td>402</td>
<td>406</td>
<td>453</td>
<td>446</td>
</tr>
<tr>
<td>rg300.2</td>
<td>355</td>
<td>376</td>
<td>377</td>
<td>412</td>
<td>455</td>
<td>461</td>
<td>503</td>
<td>506</td>
</tr>
<tr>
<td>rg300.3</td>
<td>360</td>
<td>376</td>
<td>378</td>
<td>411</td>
<td>457</td>
<td>461</td>
<td>490</td>
<td>499</td>
</tr>
<tr>
<td>rg300.4</td>
<td>389</td>
<td>422</td>
<td>426</td>
<td>432</td>
<td>497</td>
<td>508</td>
<td>530</td>
<td>540</td>
</tr>
<tr>
<td>rg300.5</td>
<td>400</td>
<td>428</td>
<td>430</td>
<td>438</td>
<td>504</td>
<td>515</td>
<td>525</td>
<td>541</td>
</tr>
<tr>
<td>tg100.1</td>
<td>138</td>
<td>188</td>
<td>197</td>
<td>292</td>
<td>292</td>
<td>294</td>
<td>290</td>
<td>294</td>
</tr>
<tr>
<td>tg100.3</td>
<td>142</td>
<td>191</td>
<td>197</td>
<td>264</td>
<td>290</td>
<td>284</td>
<td>283</td>
<td>294</td>
</tr>
<tr>
<td>tg100.5</td>
<td>138</td>
<td>187</td>
<td>197</td>
<td>251</td>
<td>262</td>
<td>268</td>
<td>272</td>
<td>294</td>
</tr>
<tr>
<td>tg100.7</td>
<td>138</td>
<td>191</td>
<td>197</td>
<td>236</td>
<td>255</td>
<td>262</td>
<td>276</td>
<td>294</td>
</tr>
<tr>
<td>tg100.9</td>
<td>140</td>
<td>189</td>
<td>196</td>
<td>226</td>
<td>260</td>
<td>263</td>
<td>272</td>
<td>290</td>
</tr>
<tr>
<td>tg200.1</td>
<td>275</td>
<td>375</td>
<td>397</td>
<td>582</td>
<td>594</td>
<td>592</td>
<td>594</td>
<td>594</td>
</tr>
<tr>
<td>tg200.3</td>
<td>279</td>
<td>382</td>
<td>397</td>
<td>558</td>
<td>579</td>
<td>592</td>
<td>586</td>
<td>594</td>
</tr>
<tr>
<td>tg200.5</td>
<td>275</td>
<td>373</td>
<td>397</td>
<td>515</td>
<td>551</td>
<td>569</td>
<td>578</td>
<td>594</td>
</tr>
<tr>
<td>tg200.7</td>
<td>275</td>
<td>372</td>
<td>397</td>
<td>492</td>
<td>552</td>
<td>578</td>
<td>589</td>
<td>594</td>
</tr>
<tr>
<td>tg200.9</td>
<td>279</td>
<td>377</td>
<td>397</td>
<td>487</td>
<td>543</td>
<td>558</td>
<td>566</td>
<td>594</td>
</tr>
</tbody>
</table>
6.4. PARAMETER DETECTION FOR SA

Figure 6.2: MPS: Traces of $SA_E$ and $SA_{CA1}$ for rg100.1 with different initial temperatures.

Figure 6.3: MPS: Traces of $SA_E$ and $SA_{CA1}$ for rg300.5 with different initial temperatures.
CHAPTER 6. MPS EXPERIMENTS

Figure 6.4: MOPS: Traces of $SA_E$ and $SA_{CA1}$ for $rg100.1$ with different initial temperatures.

Figure 6.5: MOPS: Traces of $SA_E$ and $SA_{CA1}$ for $rg300.5$ with different initial temperatures.
6.5 Comparison of CA, CA1, and CA2

The best found results for the heuristics are reported in Tables 6.4 and 6.5.

<table>
<thead>
<tr>
<th>Number of times heuristic is best or tied for best</th>
<th>CA</th>
<th>CA1</th>
<th>CA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average rank</td>
<td>2.96</td>
<td>1.54</td>
<td>1</td>
</tr>
<tr>
<td>Minimum ratio of worst found solution and optimal or best known</td>
<td>0.45</td>
<td>0.61</td>
<td>0.65</td>
</tr>
<tr>
<td>Maximum range</td>
<td>6</td>
<td>12</td>
<td>16</td>
</tr>
</tbody>
</table>

The solution quality difference of the fast algorithms is clear. CA1, and CA2 find quite similar solutions, and they outperform CA with a clear margin. For all 50 test instances, algorithm CA2 finds the best solution, and CA1 finds the same solution as CA2 for 23 graphs. The solutions of CA are always worse than the those of CA1 and CA2 for graphs that contain triangles. The only graph for which all the algorithms found the same solution was cimi-g6. The average rank of the heuristics is 2.96 for CA, 1.54 for CA1, and 1 for CA2. The comparison statistics are collected in Table 6.6.

Figure 6.6 shows the average running times of one run for CA, CA1, and CA2 as a function of the number of edges. Further, the figure contains the average running time of GRE to illustrate the running time difference of the greedy heuristics and heuristics without planarity test. The running times of CA and CA1 are less than one second for all graphs up to 50000 edges and for graph r1 having 49950 edges CA2 took on average 1.3 seconds. For the largest graph, r9 with 449550 edges, the average running times were 4.2, 5.8, and 6.5 seconds, respectively.

The running time differences between CA, CA1, and CA2 is in general very small. Only with the graphs having more than 10000 edges it can be seen that CA is slightly faster than the other two algorithms. CA2 is the slowest method. The sharp turns in the curve are the influence of the number of vertices. All three heuristics run faster for a sparse graph than for a dense graph with the same number of edges.

The variation of the solutions of all three algorithms is quite small. Figure 6.7 illustrates differences of the best found solution and the worst found solution of the heuristics for graphs with less than 1600 edges. The variation of the solutions of CA was very small, the worst case difference of the worst and best
Figure 6.6: Average running times of CA, CA1, CA2, and GRE. Notice that the axes are logarithmic.

solutions found solutions were at most 6 edges (tg200.9). The ranges of the solutions of CA1 and CA2 were at most 12 (g19), and 16 (g19), respectively. See Table 6.6 for these results. For the largest graphs, r0, r1, r9, and crack, the ranges were higher, up to 138 for crack by CA2. See Appendix A for the best and worst solutions of CA, CA1, and CA2 for graphs having over 1600 edges.

To further compare the solutions of CA2 and CA, we studied their solution quality difference. See Figure 6.8 for the ratios of the best found solutions of CA2 and CA and Figure 6.9 for the ratios of the worst found solutions of CA2 and the best found solutions of CA. The figures are very similar, since the solution qualities of CA2 and CA do not vary much. The highest
improvement was for tg200.1 with a 1.44 times better solution. In general, the highest improvements were obtained for graphs containing a planar subgraph of maximum size (tg100.1 – tg200.9). The solutions of CA2 were approximately 20 percentages better than those of CA.

CA was the first approximation algorithm for MPS with a nontrivial performance ratio 7/18, as discussed in Chapter 5. Our experiments verify this theoretical worst case solution ratio. The worst case ratio of CA solutions and the optimal or the best known solution was always more than 7/18 (the best known solutions are mainly taken from the experiments for $SA_{CA1}$ described in Section 6.7 and in [109]). This is illustrated in Figure 6.10.

Theorems 5.11 and 5.13 show that the performance ratios of CA1 and CA2 are at least the same as the performance ratio of CA. Our experiments also verify this, and in addition, the experiments also give evidence on the conjectured performance ratio 4/9: the solutions by CA1 and CA2 were never more than 4/9 away from the optima. For the ratios of the worst found solutions and the optimal or lower bound for CA2, see Figure 6.11.

Also, the worst solutions by CA1 and CA2 were always as good as the best solution by CA, as illustrated in Figure 6.9 for CA2. The solutions of CA1
and CA2 were never less than 0.61 and 0.65 times the optima, respectively.

Table 6.6 lists the minimum of the worst case ratios of the worst found solutions and the optimal or the best known solutions for the heuristics. CA achieved the worst solutions for graph tg200.1, CA1 for graph tg200.5, and CA2 for graph tg100.9.

### 6.6 Comparison of GRE, GCA, GCA1, and GCA2

It is clear that when a greedy method to add edges is applied instead of just connecting the subgraphs in Phase 2 of CA, CA1, and CA2, the solutions quality remains at least the same. The main questions are, thus, how much the greedy approach improves the solution quality, how much longer running time is needed, and does there exist graphs for which CA, CA1, or CA2 outperform GRE.

As shown in the previous section, CA2 outperformed CA1, but GCA1 produced slightly better approximations than GCA2. GCA1 found the best
Table 6.7: Comparison of the performance of GRE, GCA, GCA1, and GCA2.

<table>
<thead>
<tr>
<th></th>
<th>GRE</th>
<th>GCA</th>
<th>GCA1</th>
<th>GCA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of times heuristic is</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>best or tied for best</td>
<td>9</td>
<td>15</td>
<td>31</td>
<td>30</td>
</tr>
<tr>
<td>Average rank</td>
<td>3.38</td>
<td>2.23</td>
<td>1.36</td>
<td>1.47</td>
</tr>
<tr>
<td>Minimum ratio of worst found</td>
<td>0.67</td>
<td>0.75</td>
<td>0.78</td>
<td>0.82</td>
</tr>
<tr>
<td>solution and optimal or best known</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum range</td>
<td>94</td>
<td>70</td>
<td>74</td>
<td>79</td>
</tr>
</tbody>
</table>

or the tied best solution for 31 and GCA2 for 30 from 46 test instances. The average ranks for these two heuristics were 1.36 and 1.47, respectively. One explanation for the success of GCA1 is that the method of constructing a solution in Phase 1 of CA2 is greedier than that in CA1. The solution of CA2 could contain more edges than that of CA1, but it is more difficult to insert additional edges to the solution. GCA and GRE found the best or tied best solutions for 15 and 9 graphs and the average ranks were 2.23 and 3.38, respectively. These results are listed in Table 6.7.

![Figure 6.12: Average running times of CA, GRE, GCA1, GCA2, and SA\textsubscript{CA1}. Notice that the axes are logarithmic.](image)

All greedy algorithms have a similar running time. This is illustrated in
Figure 6.12, where the average running running times of GRE, GCA, and GCA2 are shown. The figure also contains the average running times of CA and $SA_{CA_1}$ to demonstrate the running time difference between the algorithms. The running time curve for GCA1 is not drawn, since it coincides with that of GCA and GCA2.

![Graph showing running times of different algorithms](image)

Figure 6.13: Ranges of the solutions of GRE, GCA, GCA1, and GCA2.

The variation of the solutions of the greedy algorithms is much higher than with their non-greedy versions. Figure 6.13 shows the difference of the best found solution and the worst found solution for GRE, GCA, GCA1, and GCA2 for graphs with less than 1600 edges. The range of the solutions is mostly less than 25, but for graphs containing a planar subgraph of maximum size, the ranges were up to 94 (tg200.3) for GRE. Also the solutions of the other three heuristics varied highly with graphs containing a planar subgraph of maximum size: the ranges of GCA, GCA1, and GCA2 were 70 (tg200.7), 74 (tg200.3) and 79 (tg200.5), respectively.

Since the running times of the greedy algorithms are much higher than those of CA, CA1, and CA2, it is an interesting question how much the greedy approach improves the solution quality, and does any of the heuristics CA, CA1 and CA2 outperform any greedy heuristic in the terms of the solution quality.

CA found a better solution than GRE for graph r0, CA1 for graphs r0 and g19, and CA2 for graphs r0, g19 and g17. This shows that CA, CA1, and CA2 can find better solutions for large and sparse graphs than GRE in significantly shorter computation time. This also coincides with the theoretical properties of CA, CA1, CA2, and GRE.

Applying a greedy method for the solutions of CA2, the solution quality improves on average 30 percentages. The same also holds for GCA and GCA1. See Figure 6.14 for the ratios of the best found solutions of GCA2 and CA2.

The ratio of the worst found solution of GCA1 and the optimal or best known solution was always at least 0.78 (tg100.5). GRE and GCA performed
6.7 Comparison of \( SA_E, SA_{CA1}, \) GRASP, J-M, and 2PHASE

Next we compare \( SA_E, SA_{CA1}, \) and other MPS heuristics from the literature. In Table 6.8, the number best solutions and the average ranks of \( SA_E, SA_{CA1}, \) GRASP, J-M, and 2PHASE heuristics are reported. For all graphs, both \( SA_E \) and \( SA_{CA1} \) found at least the same solution as the other algorithms.

The solution quality difference of \( SA_E \) and \( SA_{CA1} \) is not high (10 best found solutions for \( SA_E \), 15 for \( SA_{CA1} \), and 22 tied), but due to our parameter decision, the running time of \( SA_{CA1} \) was on average 40 – 50 percentage shorter than that of \( SA_E \). The average rank of \( SA_{CA1} \) was slightly better (1.20 against
The relative order of the other heuristics is GRASP (average rank 3.11), 2Phase (average rank 3.48), and J-M (average rank 3.61).

Table 6.8: Comparison of the performance of $SA_E$, $SA_{CA1}$, GRASP, J-M, and 2Phase.

<table>
<thead>
<tr>
<th></th>
<th>$SA_E$</th>
<th>$SA_{CA1}$</th>
<th>GRASP</th>
<th>J-M</th>
<th>2Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of times heuristic is best or tied for best</td>
<td>32</td>
<td>37</td>
<td>9</td>
<td>5</td>
<td>13</td>
</tr>
<tr>
<td>Average rank</td>
<td>1.30</td>
<td>1.20</td>
<td>3.11</td>
<td>3.61</td>
<td>3.48</td>
</tr>
<tr>
<td>Minimum ratio of worst found solution and optimal or best known</td>
<td>0.90</td>
<td>0.91</td>
<td>0.78</td>
<td>0.79</td>
<td>0.83</td>
</tr>
<tr>
<td>Maximum range</td>
<td>26</td>
<td>23</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 6.16: Average running times of $SA_E$, $SA_{CA1}$, and GRASP. Notice that the time axis is logarithmic.

The average running times of GRASP, $SA_E$, and $SA_{CA1}$ are shown in Figure 6.16. The sharp downturns in the curves of $SA_E$ and $SA_{CA1}$ indicate that the optimal solution is found before reaching the stopping criterion. For example, for graphs g10 and g11 with 71 and 72 edges, respectively, $SA_{CA1}$ finds an optimal solution in less than 0.1 seconds while GRASP uses over 50 seconds, since it does not recognize the possible optimality of a result. The
The greatest running time difference is for graph rg300.5, for which \(SA_{CA1}\) finds solution 541 in 7710 seconds, but for GRASP it takes over 200000 seconds to find solution 487.

Since the solutions of GRASP are one run results, it is possible that using different seeds for the random number generator, better results can be achieved. Nevertheless, the running time of GRASP is much higher than that of \(SA_{E}\) or \(SA_{CA1}\).

The variation of the solutions of \(SA_{E}\) and \(SA_{CA1}\) is much lower than that of the greedy algorithms, but slightly higher than in CA, CA1, and CA2. Figure 6.17 shows the difference of the best found solution and the worst found solution of \(SA_{E}\) and \(SA_{CA1}\) for graphs with less than 1600 edges. The difference was always less than or equal to 26 (tg100.9) for \(SA_{E}\) and 23 for \(SA_{CA1}\) (tg200.9).

Since the efficiency of the computers has increased rapidly during the past ten years, it is very probable that the solution quality of J-M is better with the modern computers because of the possibility to allow longer computation time. J-M can still be a competitive method for sparse graphs and very dense graphs, for which it produces good approximations [70, 112].

Since the solutions for J-M, GRASP, and 2Phase are obtained from one run studies, it is not known how much their solutions vary between distinct runs.

The overall solution quality of \(SA_{CA1}\) is very high. The worst case ratios of the worst found solutions and the optimal or the best known solutions were always at least 0.91 (g12), see Figure 6.18 where the worst case ratios are shown. For \(SA_{E}\), the ratio was 0.90 (tg100.9). GRASP, J-M, and 2Phase performed weaker, their ratios were 0.78 (tg200.7), 0.79 (tg200.7) and 0.83 (rg300.2), respectively. The ratios are listed in Table 6.8. The solutions of CA1 can be improved up to 60 percentages by applying \(SA_{CA1}\), as illustrated in Figure 6.19, where the ratios of the best found solutions of \(SA_{CA1}\) and CA1...
are shown.

Figure 6.18: Ratios of the worst found solutions of $SA_{CA1}$ and the optimal or the best known solution.

Figure 6.19: Ratios of the best found solutions of $SA_{CA1}$ and $CA1$.

6.8 Summary

$CA2$ gives good approximations for MPS in very short computation time. Even if the input graph contains up to 500000 edges, the running time is under ten seconds. For graphs with less than 1500 edges, GCA1 and GCA2 are preferable methods since the running time stays still in a few seconds and the number of edges in the constructed planar subgraph may increase up to 50 percentage from that of $CA1$ and $CA2$. If there is no need to limit the running time, the best approximation for MPS can be obtained using $SA_{CA1}$.
Chapter 7

MOPS experiments

For MOPS, we experimented the same heuristics as for MPS, except that CA2 and GCA2 were not applied. The test graph set was the same with one exception: graphs tg100.1 – tg200.9 were replaced by graphs o100.1 – o200.5. The number of repeats for each heuristic was the same as in MPS experiments. The outerplanarity test was implemented by using planarity test, as described in Theorem 2.6.

Since the relative performances of CA and CA1 were studied in Chapter 6, we compare CA and CA1 together with GRE, GCA, and GCA1 for MOPS in Section 7.1. SA_{CA1}, and SA_E are compared in Section 7.2. The comparison of MOPS algorithms is made in the same way as MPS comparisons as described in the previous chapter.

More detailed solution statistics for CA, CA1, GCA, GCA1, SA_E and SA_{CA1} can be found in Appendix B. CA and CA1 statistics are partially given in Appendix A.

7.1 Comparison of CA, CA1, GRE, GCA, and GCA1

The best found results for each algorithm are reported in Tables 7.1 and 7.3. GCA1 outperformed CA, CA1, GRE, and GCA with a high margin in the terms of the solution quality. For 44 of 47 test instances, GCA1 found the best or tied best solution. The ranks for the heuristics GCA1, GCA, CA1, GRE, and CA were 1.06, 2.19, 2.45, 3.60, and 4.68, respectively. The ranks and the number of best found solutions for the heuristics are listed in Table 7.2.

An interesting feature in the solution quality of CA1 was that it found
Table 7.1: The best found solutions of MOPS heuristics for the test graph set I.

<table>
<thead>
<tr>
<th>graph</th>
<th>CA</th>
<th>CA1</th>
<th>GRE</th>
<th>GCA</th>
<th>GCA1</th>
<th>$SA_E$</th>
<th>$SA_{CA1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>cimi-g1</td>
<td>11</td>
<td>13</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>cimi-g2</td>
<td>88</td>
<td>117</td>
<td>112</td>
<td>116</td>
<td>117</td>
<td>117</td>
<td>117</td>
</tr>
<tr>
<td>cimi-g3</td>
<td>38</td>
<td>49</td>
<td>51</td>
<td>51</td>
<td>51</td>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td>cimi-g4</td>
<td>13</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>cimi-g5</td>
<td>59</td>
<td>73</td>
<td>78</td>
<td>78</td>
<td>79</td>
<td>79</td>
<td>79</td>
</tr>
<tr>
<td>cimi-g6</td>
<td>42</td>
<td>42</td>
<td>51</td>
<td>51</td>
<td>51</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>g10</td>
<td>36</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>g11</td>
<td>36</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>g12</td>
<td>36</td>
<td>47</td>
<td>46</td>
<td>47</td>
<td>47</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>g13</td>
<td>73</td>
<td>97</td>
<td>83</td>
<td>92</td>
<td>97</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td>g14</td>
<td>73</td>
<td>97</td>
<td>90</td>
<td>94</td>
<td>97</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td>g15</td>
<td>73</td>
<td>97</td>
<td>93</td>
<td>97</td>
<td>97</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td>g16</td>
<td>137</td>
<td>162</td>
<td>136</td>
<td>161</td>
<td>168</td>
<td>167</td>
<td>171</td>
</tr>
<tr>
<td>g17</td>
<td>147</td>
<td>194</td>
<td>149</td>
<td>171</td>
<td>193</td>
<td>181</td>
<td>193</td>
</tr>
<tr>
<td>g18</td>
<td>147</td>
<td>197</td>
<td>154</td>
<td>189</td>
<td>197</td>
<td>188</td>
<td>196</td>
</tr>
<tr>
<td>g19</td>
<td>218</td>
<td>274</td>
<td>204</td>
<td>252</td>
<td>275</td>
<td>257</td>
<td>275</td>
</tr>
<tr>
<td>r0</td>
<td>1475</td>
<td>1766</td>
<td>1184</td>
<td>1601</td>
<td>1791</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>r1</td>
<td>1497</td>
<td>1997</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>r9</td>
<td>1498</td>
<td>1997</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>crack</td>
<td>15107</td>
<td>20067</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

* The algorithm is not applied for this graph.

Table 7.2: Comparison of the performance of the fast and greedy heuristics.

<table>
<thead>
<tr>
<th></th>
<th>CA</th>
<th>CA1</th>
<th>GRE</th>
<th>GCA</th>
<th>GCA1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of times</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>heuristic is best</td>
<td>0</td>
<td>18</td>
<td>6</td>
<td>8</td>
<td>44</td>
</tr>
<tr>
<td>Average rank</td>
<td>4.68</td>
<td>2.45</td>
<td>3.60</td>
<td>2.19</td>
<td>1.06</td>
</tr>
<tr>
<td>Minimum ratio</td>
<td>0.71</td>
<td>0.81</td>
<td>0.62</td>
<td>0.82</td>
<td>0.87</td>
</tr>
<tr>
<td>Maximum range</td>
<td>6</td>
<td>22</td>
<td>40</td>
<td>44</td>
<td>15</td>
</tr>
</tbody>
</table>
Table 7.3: The best found solutions of MOPS heuristics for the test graph set II.

<table>
<thead>
<tr>
<th>graph</th>
<th>CA</th>
<th>CA1</th>
<th>GRE</th>
<th>GCA</th>
<th>GCA1</th>
<th>$SA_E$</th>
<th>$SA_{CA1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>rg100.1</td>
<td>119</td>
<td>124</td>
<td>126</td>
<td>134</td>
<td>135</td>
<td>144</td>
<td>145</td>
</tr>
<tr>
<td>rg100.2</td>
<td>118</td>
<td>125</td>
<td>126</td>
<td>135</td>
<td>136</td>
<td>145</td>
<td>148</td>
</tr>
<tr>
<td>rg100.3</td>
<td>120</td>
<td>128</td>
<td>125</td>
<td>137</td>
<td>141</td>
<td>145</td>
<td>149</td>
</tr>
<tr>
<td>rg100.4</td>
<td>126</td>
<td>136</td>
<td>128</td>
<td>144</td>
<td>147</td>
<td>151</td>
<td>153</td>
</tr>
<tr>
<td>rg100.5</td>
<td>137</td>
<td>153</td>
<td>135</td>
<td>156</td>
<td>158</td>
<td>163</td>
<td>164</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rg150.1</td>
<td>171</td>
<td>174</td>
<td>180</td>
<td>191</td>
<td>195</td>
<td>207</td>
<td>208</td>
</tr>
<tr>
<td>rg150.2</td>
<td>176</td>
<td>182</td>
<td>179</td>
<td>194</td>
<td>199</td>
<td>210</td>
<td>207</td>
</tr>
<tr>
<td>rg150.3</td>
<td>171</td>
<td>179</td>
<td>181</td>
<td>195</td>
<td>198</td>
<td>210</td>
<td>213</td>
</tr>
<tr>
<td>rg150.4</td>
<td>180</td>
<td>190</td>
<td>183</td>
<td>201</td>
<td>207</td>
<td>217</td>
<td>219</td>
</tr>
<tr>
<td>rg150.5</td>
<td>178</td>
<td>185</td>
<td>186</td>
<td>200</td>
<td>205</td>
<td>218</td>
<td>217</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rg200.1</td>
<td>222</td>
<td>227</td>
<td>231</td>
<td>245</td>
<td>246</td>
<td>264</td>
<td>265</td>
</tr>
<tr>
<td>rg200.2</td>
<td>216</td>
<td>219</td>
<td>232</td>
<td>238</td>
<td>242</td>
<td>261</td>
<td>263</td>
</tr>
<tr>
<td>rg200.3</td>
<td>235</td>
<td>243</td>
<td>235</td>
<td>261</td>
<td>265</td>
<td>280</td>
<td>285</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>rg300.1</td>
<td>324</td>
<td>330</td>
<td>342</td>
<td>357</td>
<td>360</td>
<td>390</td>
<td>394</td>
</tr>
<tr>
<td>rg300.2</td>
<td>355</td>
<td>376</td>
<td>352</td>
<td>392</td>
<td>405</td>
<td>415</td>
<td>422</td>
</tr>
<tr>
<td>rg300.3</td>
<td>360</td>
<td>376</td>
<td>353</td>
<td>394</td>
<td>405</td>
<td>414</td>
<td>417</td>
</tr>
<tr>
<td>rg300.4</td>
<td>389</td>
<td>422</td>
<td>361</td>
<td>428</td>
<td>444</td>
<td>432</td>
<td>437</td>
</tr>
<tr>
<td>rg300.5</td>
<td>400</td>
<td>428</td>
<td>365</td>
<td>436</td>
<td>446</td>
<td>441</td>
<td>443</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>o100.1</td>
<td>144</td>
<td>196</td>
<td>171</td>
<td>190</td>
<td>197</td>
<td>197</td>
<td>196</td>
</tr>
<tr>
<td>o100.2</td>
<td>144</td>
<td>194</td>
<td>146</td>
<td>177</td>
<td>194</td>
<td>190</td>
<td>197</td>
</tr>
<tr>
<td>o100.3</td>
<td>145</td>
<td>194</td>
<td>145</td>
<td>177</td>
<td>196</td>
<td>187</td>
<td>194</td>
</tr>
<tr>
<td>o100.4</td>
<td>145</td>
<td>195</td>
<td>147</td>
<td>172</td>
<td>195</td>
<td>187</td>
<td>195</td>
</tr>
<tr>
<td>o100.5</td>
<td>147</td>
<td>194</td>
<td>152</td>
<td>175</td>
<td>194</td>
<td>189</td>
<td>193</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>o200.1</td>
<td>292</td>
<td>397</td>
<td>349</td>
<td>394</td>
<td>397</td>
<td>397</td>
<td>397</td>
</tr>
<tr>
<td>o200.2</td>
<td>292</td>
<td>397</td>
<td>307</td>
<td>366</td>
<td>397</td>
<td>397</td>
<td>397</td>
</tr>
<tr>
<td>o200.3</td>
<td>281</td>
<td>396</td>
<td>284</td>
<td>370</td>
<td>395</td>
<td>389</td>
<td>397</td>
</tr>
<tr>
<td>o200.4</td>
<td>289</td>
<td>391</td>
<td>282</td>
<td>351</td>
<td>391</td>
<td>379</td>
<td>394</td>
</tr>
<tr>
<td>o200.5</td>
<td>291</td>
<td>394</td>
<td>268</td>
<td>353</td>
<td>389</td>
<td>374</td>
<td>389</td>
</tr>
</tbody>
</table>
Figure 7.1: Average running times of CA1, GRE, GCA1, and $SA_{CA1}$. Notice that the axes are logarithmic.

Figure 7.2: Ranges of the solutions of GRE, GCA, and GCA1.

quite often better solutions than GRE. Especially, CA1 found good solutions for Hamiltonian graphs (g10 – g19) and for graphs containing an outerplanar subgraph of maximum size (o100.1 – o200.5). GRE achieved better solution for some real world graphs (cimi-g1, cimi-g3 – g6) and for sparse random graphs (rg100.1, rg100.2, rg150.1, rg200.1, for example). CA1 found better solutions than GCA1 for g17, o200.3 and o200.5, but this was due to the randomness of the algorithms. The performance ratio of CA is better than that of GRE, and in our experiments also CA found a better approximation than GRE for nine test graphs (rg300.2 – 5, o200.4 – 5, for example). In general, GRE finds better approximations for small and sparse graphs, CA for large and dense graphs.
7.1. **Comparison of CA, CA1, GRE, GCA, and GCA1**

![Image of Figure 7.3: Ratios of the worst found solutions of CA and the optimal or the best known solution.](image1)

![Image of Figure 7.4: Ratios of the worst found solutions of CA1 and the optimal or the best known solution.](image2)

The running time difference of the algorithms is clear. From fastest to the slowest the algorithms are CA, CA1, GRE, GCA, and GCA1. The relative order of the greedy heuristics also depends on the input graph. Algorithms GCA and GCA1 do not perform as many outerplanarity tests as GRE due to good initial solution, but for the same reason, to make an outerplanarity test in GCA or GCA1, it takes in general a longer time than in GRE. The average running times of heuristics CA1, GRE, and GCA1 are shown in Figure 7.1. The sharp turns in the curves of GCA1 and SA_{CA1} indicate that the optimal solution is recognized. The running time curve for GCA is not drawn since it coincides with that of GCA1.

The variation of the solutions of the greedy algorithms is higher than the variations of CA and CA1 (see Figure 6.7 in Section 6.5). Figure 7.2 illustrates the difference of the best found solution and the worst found solution of heuristics GRE, GCA, and GCA1. The ranges of the solutions of GCA and GRE are highest, up to 44 and 40 (o200.1), respectively. The range of the solutions of GCA1 was at most 15 (rg300.4).

In our experiments, the worst solutions of CA were never less than 2/3 away from the optimum, as illustrated in Figure 7.3. The theoretical performance
CHAPTER 7. MOPS EXPERIMENTS

Figure 7.5: Ratios of the worst found solutions of GCA1 and the optimal or the best known solution.

The ratio of CA is 7/12, so the algorithm seems to work slightly better in practice. The minimum ratio of the worst found solutions of CA and the optimal or the best known solution was 0.71 for graph o100.1.

The solutions of CA1 and GCA1 were never less than 0.81 and 0.87 (cimig6), respectively, away from the optima. The worst case ratio of GCA was 0.82 (o100.2). GRE performed the weakest, its ratio was only 0.62 (o200.5). The worst case ratios are collected in Table 7.2 and the ratios of CA1 and GCA1 are illustrated in Figures 7.4 and 7.5.

7.2 Comparison of $SA_E$, $SA_{CA1}$, and the other heuristics

In Section 6.7 it was shown that simulated annealing improves significantly the solutions of CA1 and CA2 when finding planar subgraphs with a large number of edges. Simulated annealing also works quite efficiently for MOPS, but now $SA_E$ is not always able to find as good solutions as CA1. For example, for graph g19, CA1 finds an outerplanar subgraph with 274 edges, but $SA_E$ solution is only 257.

The effect of the initial solution on the final solution is remarkable for MOPS. $SA_{CA1}$ found the best solutions for 29 graphs and $SA_E$ only for 3 with 14 tied solutions. If the solution of CA1 was already close to the optimal solution, there was no improvement at all. For example, for graphs g17–g19 and o100.1–o200.5, the solutions of $SA_{CA1}$ are of the same quality as that of CA1. $SA_{CA1}$ outperforms GCA1 clearly with 23 against 6 best found solutions (17 tied), but now for graphs g17–g19 and o100.1–o200.5, GCA1 or CA1 solutions were sometimes similar or even better due to the randomness of the algorithms.
Table 7.4: Comparison of the performances of the fast and greedy heuristics.

<table>
<thead>
<tr>
<th></th>
<th>$SA_E$</th>
<th>$SA_{CA1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of times heuristic is</td>
<td></td>
<td></td>
</tr>
<tr>
<td>best or with tied for best</td>
<td>17</td>
<td>43</td>
</tr>
<tr>
<td>Average rank</td>
<td>1.63</td>
<td>1.07</td>
</tr>
<tr>
<td>Minimum ratio of worst found</td>
<td>0.89</td>
<td>0.93</td>
</tr>
<tr>
<td>solution and optimal or best known</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum range</td>
<td>20</td>
<td>11</td>
</tr>
</tbody>
</table>

The running times of $SA_E$ and $SA_{CA1}$ are similar, although $SA_{CA1}$ runs slightly faster for graphs o100.1 – o200.5, since the algorithm finds the optimal solution faster. The average running time of $SA_{CA1}$ is illustrated in Figure 7.1.

![Figure 7.6: Ranges of the solutions of $SA_E$ and $SA_{CA1}$.](image)

In general, the differences between the worst and the best found solutions for $SA_E$ and $SA_{CA1}$ were smaller than with the greedy heuristics and slightly higher than with CA and CA1. The highest range for $SA_E$ was 20 (o200.4) and 11 (rg200.5) for $SA_{CA1}$. The ranges of the solutions of $SA_E$ and $SA_{CA1}$ are shown in Figure 7.6.

The worst case ratio of the solution obtained from $SA_E$ and the optimal or the best known solution was 0.89, and for $SA_{CA1}$, only 0.93. This shows that simulated annealing with the initial solution taken from CA1 is a very efficient method to approximate MOPS. See Figure 7.7 for an illustration of the worst case solution quality of $SA_{CA1}$.

### 7.3 Summary

For MOPS, CA1 produces rather good approximations for all tested graphs in a very short computation time. If there is a need to improve the solution
Figure 7.7: Performance ratio of the worst found solutions of SA_{CA1} and the optimal or best known solution.

quality at the cost of the running time, GCA1 and SA_{CA1} produce slightly better approximations. In our experiments, the solutions of SA_{CA1} were never less than 0.93 away from the optimum.
Chapter 8

Thickness and outerthickness experiments

In this chapter, we compare algorithms for approximating the thickness and outerthickness of a graph. The first section describes the experimented algorithms and their implementation. Section 8.2 introduces graphs used in the experiments and Section 8.3 describes the parameter selection for SA. Sections 8.4 and 8.5 cover the comparison of the fast heuristics and greedy heuristics for the thickness. Section 8.6 compares the heuristics HT, J-M, GA, and SA. In Section 8.7, we give theoretically interesting results of SA for some complete bipartite and tripartite graphs. Finally, the outerthickness algorithms are compared. The last section summarizes the results.

8.1 The experimented algorithms and their implementation

For the thickness problem, we tested the extraction algorithm Thick by applying in Step 4 algorithms CA, CA1, CA2, GCA, GCA1, GCA2 (see Chapter 5), and GRE (see Chapter 4). Also, we implemented such an ST heuristic which in each iteration extracts the set of tree-edges found by DFS (see Chapter 2). In what follows, these algorithms are denoted simply by the name of the extraction method. All these algorithms approximate thickness, but algorithms ST, CA, and CA1 directly produce approximations also for outerthickness. The greedy heuristics, except GCA2, are used to approximate outerthickness problem by applying an outerplanarity test instead of a planarity test in Phase 2 of GRE.

We also compared the above methods against algorithms J-M given by
Junger and Mutzel [101], HT experimented by Cimikowski [32], and GA reported by Mäkinen et al. [88]. The solutions are taken directly from the articles since the source codes are not available. The SA solutions come mainly from our earlier work [108], but in this work we also apply the same algorithm for some larger graphs. To generate the initial solutions for the simulated annealing algorithm, GRE and GCA1 were applied. The SA solutions for outerthickness are new.

The implementation and comparison details given in Chapter 6 also hold for the thickness and outerthickness experiments.

ST, CA, CA1, and CA2 were repeated 25 times for graphs with less than 2000 edges, 10 times for graphs having more than 200 edges but no more than 250000 edges and 5 times for the graphs larger than that. Greedy heuristics for the thickness were applied only for graphs with less than or equal to 79800 edges and for outerthickness only for graphs with less than or equal to 4950 edges. The number of repeats were 25 for graphs with less than 2000 edges and 10 times for graphs with less than 5000 edges. For larger graphs only one run was performed.

SA was applied for complete graphs up to 100 vertices, complete bipartite graphs up to 100 vertices, and random graphs with less than 1600 edges. SA was repeated 20 times for graphs with at most 30 vertices, and for larger graphs 5 or 10 times. For complete graphs with more than 50 vertices, only one run was performed. SA was applied only for graphs with less than or equal to 100 vertices.

More detailed solution statistics for ST, CA, CA1, CA2, GCA, GCA1, and GCA2, for thickness can be found in Appendix C. For outerthickness, GCA, GCA1, SA, and GRE statistics for several bipartite graphs are given in Appendix D.

8.2 Test graph suite

Algorithms for the thickness problem are compared in the literature using complete graphs, complete bipartite graphs and some random graphs [101, 32, 88, 108]. We use mainly the same graphs as in the earlier experiments, but we have included also larger complete and random graphs to the test graph set. Only ST, CA, CA1, and CA2 are run for the largest graphs. In this work the highest number of edges have been at most 5000, in this work the highest number of edges is 499500 in \(K_{1000}\).

Information on test graphs are collected in Tables 8.1 and 8.2. For all graphs, we have listed the name of the graph (name) and the number of vertices
8.2. TEST GRAPH SUITE

Table 8.1: Test graph statistics for thickness and outerthickness.

<table>
<thead>
<tr>
<th>graph data</th>
<th></th>
<th></th>
<th>graph data</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td></td>
<td></td>
<td>lb</td>
<td>sol’n</td>
<td>ref.</td>
<td>lb</td>
</tr>
<tr>
<td>$K_{10}$</td>
<td>10</td>
<td>45</td>
<td>3*</td>
<td>3</td>
<td>[101, 32, 88, 108]</td>
<td>3*</td>
</tr>
<tr>
<td>$K_{15}$</td>
<td>15</td>
<td>105</td>
<td>3*</td>
<td>3</td>
<td>[88, 108]</td>
<td>4*</td>
</tr>
<tr>
<td>$K_{20}$</td>
<td>20</td>
<td>190</td>
<td>4*</td>
<td>4</td>
<td>[88, 108]</td>
<td>6*</td>
</tr>
<tr>
<td>$K_{30}$</td>
<td>30</td>
<td>435</td>
<td>6*</td>
<td>6</td>
<td>[88, 108]</td>
<td>8*</td>
</tr>
<tr>
<td>$K_{40}$</td>
<td>40</td>
<td>780</td>
<td>7*</td>
<td>7</td>
<td>[108]</td>
<td>11*</td>
</tr>
<tr>
<td>$K_{50}$</td>
<td>50</td>
<td>1225</td>
<td>9*</td>
<td>9</td>
<td>[108]</td>
<td>13*</td>
</tr>
<tr>
<td>$K_{60}$</td>
<td>60</td>
<td>1770</td>
<td>11*</td>
<td>11</td>
<td>†</td>
<td>16*</td>
</tr>
<tr>
<td>$K_{70}$</td>
<td>70</td>
<td>2415</td>
<td>12*</td>
<td>13</td>
<td>†</td>
<td>18*</td>
</tr>
<tr>
<td>$K_{80}$</td>
<td>80</td>
<td>3160</td>
<td>14*</td>
<td>15</td>
<td>†</td>
<td>21*</td>
</tr>
<tr>
<td>$K_{90}$</td>
<td>90</td>
<td>4005</td>
<td>16*</td>
<td>16</td>
<td>†</td>
<td>23*</td>
</tr>
<tr>
<td>$K_{100}$</td>
<td>100</td>
<td>4950</td>
<td>17*</td>
<td>18</td>
<td>†</td>
<td>26*</td>
</tr>
<tr>
<td>$K_{10,10}$</td>
<td>20</td>
<td>100</td>
<td>3*</td>
<td>3</td>
<td>[88, 108]</td>
<td>4*</td>
</tr>
<tr>
<td>$K_{15,15}$</td>
<td>30</td>
<td>225</td>
<td>5*</td>
<td>5</td>
<td>[101, 88, 108]</td>
<td>6*</td>
</tr>
<tr>
<td>$K_{20,20}$</td>
<td>40</td>
<td>400</td>
<td>6*</td>
<td>6</td>
<td>[88, 108]</td>
<td>7*</td>
</tr>
<tr>
<td>$K_{30,30}$</td>
<td>60</td>
<td>900</td>
<td>8*</td>
<td>8</td>
<td>[108]</td>
<td>11*</td>
</tr>
<tr>
<td>$K_{40,40}$</td>
<td>80</td>
<td>1200</td>
<td>11*</td>
<td>11</td>
<td>[108]</td>
<td>14*</td>
</tr>
<tr>
<td>$K_{50,50}$</td>
<td>100</td>
<td>2500</td>
<td>13*</td>
<td>13</td>
<td>[108]</td>
<td>17*</td>
</tr>
<tr>
<td>$hc_4$</td>
<td>16</td>
<td>32</td>
<td>2*</td>
<td>2</td>
<td>[108]</td>
<td>2*</td>
</tr>
<tr>
<td>$hc_5$</td>
<td>32</td>
<td>80</td>
<td>2*</td>
<td>2</td>
<td>[108]</td>
<td>2*</td>
</tr>
<tr>
<td>$hc_6$</td>
<td>64</td>
<td>192</td>
<td>2*</td>
<td>2</td>
<td>[108]</td>
<td>3*</td>
</tr>
<tr>
<td>$hc_7$</td>
<td>128</td>
<td>448</td>
<td>2*</td>
<td>2</td>
<td>[108]</td>
<td>3*</td>
</tr>
<tr>
<td>$hc_8$</td>
<td>256</td>
<td>1024</td>
<td>3*</td>
<td>3</td>
<td>[108]</td>
<td>3*</td>
</tr>
<tr>
<td>$r_{10,25}$</td>
<td>10</td>
<td>25</td>
<td>2*</td>
<td>2</td>
<td>[108]</td>
<td>2*</td>
</tr>
<tr>
<td>$r_{20,92}$</td>
<td>20</td>
<td>92</td>
<td>2*</td>
<td>2</td>
<td>[108]</td>
<td>3*</td>
</tr>
<tr>
<td>$r_{30,230}$</td>
<td>30</td>
<td>230</td>
<td>3</td>
<td>4</td>
<td>[108]</td>
<td>5*</td>
</tr>
<tr>
<td>$r_{40,311}$</td>
<td>40</td>
<td>311</td>
<td>3</td>
<td>4</td>
<td>[108]</td>
<td>5*</td>
</tr>
<tr>
<td>$r_{50,495}$</td>
<td>50</td>
<td>495</td>
<td>4</td>
<td>5</td>
<td>[108]</td>
<td>6*</td>
</tr>
<tr>
<td>$r_{60,556}$</td>
<td>60</td>
<td>556</td>
<td>4</td>
<td>5</td>
<td>[108]</td>
<td>5</td>
</tr>
<tr>
<td>$r_{70,766}$</td>
<td>70</td>
<td>766</td>
<td>4</td>
<td>6</td>
<td>[108]</td>
<td>6</td>
</tr>
<tr>
<td>$r_{80,939}$</td>
<td>80</td>
<td>939</td>
<td>5</td>
<td>6</td>
<td>[108]</td>
<td>6</td>
</tr>
<tr>
<td>$r_{90,1263}$</td>
<td>90</td>
<td>1263</td>
<td>5</td>
<td>7</td>
<td>[108]</td>
<td>8</td>
</tr>
<tr>
<td>$r_{100,1508}$</td>
<td>100</td>
<td>1508</td>
<td>5</td>
<td>8</td>
<td>[108]</td>
<td>8</td>
</tr>
</tbody>
</table>

* Lower bound is known to be optimal.
† Solution is obtained in this work.
Table 8.2: More test graph statistics for thickness and outertickness.

<table>
<thead>
<tr>
<th>name</th>
<th>graph data</th>
<th>THICKNESS</th>
<th>OUTERTHICKNESS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>lb</td>
</tr>
<tr>
<td>$K_{150}$</td>
<td></td>
<td></td>
<td>26*</td>
</tr>
<tr>
<td>$K_{200}$</td>
<td></td>
<td></td>
<td>34*</td>
</tr>
<tr>
<td>$K_{300}$</td>
<td></td>
<td></td>
<td>51*</td>
</tr>
<tr>
<td>$K_{400}$</td>
<td></td>
<td></td>
<td>67*</td>
</tr>
<tr>
<td>$K_{500}$</td>
<td></td>
<td></td>
<td>84*</td>
</tr>
<tr>
<td>$K_{600}$</td>
<td></td>
<td></td>
<td>101*</td>
</tr>
<tr>
<td>$K_{700}$</td>
<td></td>
<td></td>
<td>117*</td>
</tr>
<tr>
<td>$K_{800}$</td>
<td></td>
<td></td>
<td>134*</td>
</tr>
<tr>
<td>$K_{900}$</td>
<td></td>
<td></td>
<td>151*</td>
</tr>
<tr>
<td>$K_{1000}$</td>
<td></td>
<td></td>
<td>167*</td>
</tr>
<tr>
<td>r0</td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>r1</td>
<td></td>
<td></td>
<td>17</td>
</tr>
<tr>
<td>r2</td>
<td></td>
<td></td>
<td>34</td>
</tr>
<tr>
<td>r3</td>
<td></td>
<td></td>
<td>51</td>
</tr>
<tr>
<td>r4</td>
<td></td>
<td></td>
<td>67</td>
</tr>
<tr>
<td>r5</td>
<td></td>
<td></td>
<td>84</td>
</tr>
<tr>
<td>r6</td>
<td></td>
<td></td>
<td>101</td>
</tr>
<tr>
<td>r7</td>
<td></td>
<td></td>
<td>117</td>
</tr>
<tr>
<td>r8</td>
<td></td>
<td></td>
<td>134</td>
</tr>
<tr>
<td>r9</td>
<td></td>
<td></td>
<td>151</td>
</tr>
<tr>
<td>rr1</td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>rr2</td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>rr3</td>
<td></td>
<td></td>
<td>17</td>
</tr>
<tr>
<td>rr4</td>
<td></td>
<td></td>
<td>26</td>
</tr>
<tr>
<td>rr5</td>
<td></td>
<td></td>
<td>34</td>
</tr>
<tr>
<td>rr6</td>
<td></td>
<td></td>
<td>42</td>
</tr>
<tr>
<td>rr7</td>
<td></td>
<td></td>
<td>51</td>
</tr>
<tr>
<td>rr8</td>
<td></td>
<td></td>
<td>59</td>
</tr>
<tr>
<td>rr9</td>
<td></td>
<td></td>
<td>67</td>
</tr>
<tr>
<td>rr10</td>
<td></td>
<td></td>
<td>76</td>
</tr>
<tr>
<td>rr11</td>
<td></td>
<td></td>
<td>84</td>
</tr>
</tbody>
</table>

* Lower bound is known to be optimal.
† Solution is obtained in this work.
8.3 Parameter detection for SA

To find good parameters for SA, we used \( K_{15}, K_{30}, \) and \( K_{20,20} \) as test graphs in the parameter detection. We ran SA 20 times with different initial temperatures for graphs \( K_{15} \) and \( K_{20,20} \), and 10 times for \( K_{30} \). The initial solutions were generated by GRE. The initial temperature varied between 0.20625 and 0.001. The cooling ratio was set to 0.995 and the equilibrium detection rate was set to be the number of edges in the input graph. In all preliminary test runs, if the algorithm converged to any solution, it found the optimal. See Figure 8.1 for the running times with the different initial temperatures. The running times of \( K_{15} \) and \( K_{20,20} \) are the total running time of 20 distinct runs of SA. The running time of \( K_{30} \) is the average running time of one run.

\(^1\)The random graphs can be downloaded from http://http://www.cs.uta.fi/~tp/apptopinv (Feb. 26, 2004). The other graphs can be constructed by giving command line parameters for apptopinv [107].

1 The random graphs can be downloaded from http://http://www.cs.uta.fi/~tp/apptopinv (Feb. 26, 2004). The other graphs can be constructed by giving command line parameters for apptopinv [107].
Figure 8.1: Test runs for detecting good initial temperature.

If the initial temperature was more than 0.20625 for $K_{15}$ and $K_{20,20}$, and more than 0.15 for $K_{30}$, SA wandered aimlessly in the search space. When the initial temperature was lower than 0.15, the optimal solution was found in all runs. The speed of the convergence to the optimal solution was fastest when the initial temperature was between 0.0955 and 0.038. When the temperature was decreased below 0.038, the convergence got slightly slower. This was due to the fact that the current temperature had no effect to the computation, almost all “bad” moves were rejected anyway. The probability of getting stuck at a local optima for a long time increases, if the temperature gets too cold. The rectangle in Figure 8.1 denotes a temperature interval where the convergence to the optimal solution was fast.

This preliminary test gave a direction to detect the good initial temperature. The initial temperature should be at most 0.0955. Of course, we cannot be sure whether this temperature interval is suitable for very large graphs. By our preliminary tests, we chose the following parameters for the graphs with less than 300 edges: we set the initial temperature to 0.0955 and the frozen temperature to 0.038. Cooling ratio was set to 0.98 to ensure that the algorithm stops in a reasonable amount of time. For graphs with at least 300 edges, we increased the cooling ratio to 0.99 and decreased the initial temperature to 0.075. This allows SA to run slightly longer with a better temperature interval for large graphs. The equilibrium detection rate was set to be the number of edges in the graph. We applied exactly the same parameters for
8.4. COMPARISON OF ST, CA, CA1, AND CA2

Table 8.3: The best found solutions for thickness.

<table>
<thead>
<tr>
<th>graph</th>
<th>ST</th>
<th>CA</th>
<th>CA1</th>
<th>CA2</th>
<th>GRE</th>
<th>GCA</th>
<th>GCA1</th>
<th>GCA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{10}$</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$K_{15}$</td>
<td>8</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$K_{20}$</td>
<td>11</td>
<td>9</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$K_{25}$</td>
<td>16</td>
<td>13</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>$K_{40}$</td>
<td>21</td>
<td>18</td>
<td>12</td>
<td>12</td>
<td>11</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>$K_{50}$</td>
<td>27</td>
<td>22</td>
<td>15</td>
<td>15</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>$K_{60}$</td>
<td>32</td>
<td>27</td>
<td>18</td>
<td>18</td>
<td>16</td>
<td>14</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>$K_{70}$</td>
<td>38</td>
<td>32</td>
<td>21</td>
<td>21</td>
<td>19</td>
<td>16</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>$K_{80}$</td>
<td>43</td>
<td>36</td>
<td>24</td>
<td>24</td>
<td>21</td>
<td>19</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>$K_{90}$</td>
<td>48</td>
<td>41</td>
<td>27</td>
<td>27</td>
<td>24</td>
<td>21</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>$K_{100}$</td>
<td>54</td>
<td>45</td>
<td>30</td>
<td>31</td>
<td>27</td>
<td>23</td>
<td>21</td>
<td>22</td>
</tr>
<tr>
<td>$r_{10,25}$</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$r_{20,92}$</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$r_{30,230}$</td>
<td>9</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$r_{40,311}$</td>
<td>9</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$r_{50,495}$</td>
<td>11</td>
<td>9</td>
<td>7</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$r_{60,556}$</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$r_{70,766}$</td>
<td>13</td>
<td>9</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$r_{80,939}$</td>
<td>13</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$r_{90,1263}$</td>
<td>16</td>
<td>12</td>
<td>9</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>$r_{100,1508}$</td>
<td>17</td>
<td>13</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

8.4 Comparison of ST, CA, CA1, and CA2

Since CA, CA1, and CA2 behave similarly as ST for graphs without triangles, bipartite graphs are omitted from the comparison. The total number of test graphs for the fast algorithms was 52. The best found solutions for the fast heuristics and the greedy heuristics are listed in Tables 8.3 and 8.4.

CA1 and CA2 outperformed ST and CA with a clear margin. CA1 and CA2 found 42 and 32 the best found solutions, respectively, and their average ranks were 1.19 and 1.38, respectively. CA and ST found only 1 and 2 tied best solutions and their average ranks were 2.88 and 3.90, respectively. These comparison results are collected in Table 8.5. The reason for the relative performance of CA2 and CA1 is that CA2 is too greedy. It might extract larger planar graphs than CA1 (as shown in Chapter 6), but this affects the
Table 8.4: More best found solutions for thickness.

<table>
<thead>
<tr>
<th>graph</th>
<th>ST</th>
<th>CA</th>
<th>CA1</th>
<th>CA2</th>
<th>GRE</th>
<th>GCA</th>
<th>GCA1</th>
<th>GCA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{150}$</td>
<td>81</td>
<td>69</td>
<td>45</td>
<td>45</td>
<td>42</td>
<td>39</td>
<td>35</td>
<td>34</td>
</tr>
<tr>
<td>$K_{200}$</td>
<td>108</td>
<td>91</td>
<td>60</td>
<td>60</td>
<td>56</td>
<td>52</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>$K_{300}$</td>
<td>164</td>
<td>137</td>
<td>92</td>
<td>90</td>
<td>84</td>
<td>79</td>
<td>71</td>
<td>72</td>
</tr>
<tr>
<td>$K_{400}$</td>
<td>217</td>
<td>183</td>
<td>121</td>
<td>120</td>
<td>112</td>
<td>105</td>
<td>96</td>
<td>98</td>
</tr>
<tr>
<td>$K_{500}$</td>
<td>271</td>
<td>230</td>
<td>152</td>
<td>149</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$K_{600}$</td>
<td>334</td>
<td>277</td>
<td>185</td>
<td>186</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$K_{700}$</td>
<td>379</td>
<td>320</td>
<td>210</td>
<td>218</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$K_{800}$</td>
<td>437</td>
<td>366</td>
<td>247</td>
<td>250</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$K_{900}$</td>
<td>490</td>
<td>412</td>
<td>276</td>
<td>281</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$K_{1000}$</td>
<td>551</td>
<td>456</td>
<td>303</td>
<td>315</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>r0</td>
<td>17</td>
<td>15</td>
<td>14</td>
<td>14</td>
<td>14</td>
<td>12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>r1</td>
<td>53</td>
<td>43</td>
<td>32</td>
<td>31</td>
<td>36</td>
<td>30</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>r2</td>
<td>103</td>
<td>89</td>
<td>57</td>
<td>57</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>r3</td>
<td>160</td>
<td>130</td>
<td>82</td>
<td>83</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>r4</td>
<td>213</td>
<td>177</td>
<td>108</td>
<td>110</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>r5</td>
<td>264</td>
<td>224</td>
<td>133</td>
<td>138</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>r6</td>
<td>312</td>
<td>270</td>
<td>158</td>
<td>170</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>r7</td>
<td>363</td>
<td>316</td>
<td>184</td>
<td>198</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>r8</td>
<td>413</td>
<td>361</td>
<td>209</td>
<td>230</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>r9</td>
<td>465</td>
<td>411</td>
<td>235</td>
<td>261</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>rr1</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>rr2</td>
<td>26</td>
<td>22</td>
<td>19</td>
<td>19</td>
<td>20</td>
<td>17</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>rr3</td>
<td>51</td>
<td>43</td>
<td>32</td>
<td>31</td>
<td>36</td>
<td>30</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>rr4</td>
<td>76</td>
<td>65</td>
<td>44</td>
<td>44</td>
<td>50</td>
<td>43</td>
<td>37</td>
<td>37</td>
</tr>
<tr>
<td>rr5</td>
<td>101</td>
<td>88</td>
<td>57</td>
<td>57</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>rr6</td>
<td>126</td>
<td>112</td>
<td>69</td>
<td>70</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>rr7</td>
<td>151</td>
<td>136</td>
<td>82</td>
<td>83</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>rr8</td>
<td>176</td>
<td>160</td>
<td>94</td>
<td>96</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>rr9</td>
<td>201</td>
<td>184</td>
<td>107</td>
<td>108</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>rr10</td>
<td>226</td>
<td>209</td>
<td>119</td>
<td>121</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>rr11</td>
<td>251</td>
<td>233</td>
<td>132</td>
<td>134</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
solution quality in the long run for large graphs \((K_{600} - K_{1000})\) and dense random graphs \((r3 - r9 \text{ and } rr6 - rr11)\) by decreasing the sizes of the extracted subgraphs more rapidly than CA1. For complete graphs with less than 600 vertices and sparse random graphs \((r1 - r2 \text{ and } rr0 - rr5)\), CA2 obtained approximations of least the same quality than CA1.

Table 8.5: Comparison of the performance of ST, CA, CA1, and CA2.

<table>
<thead>
<tr>
<th></th>
<th>ST</th>
<th>CA</th>
<th>CA1</th>
<th>CA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of times heuristic is</td>
<td>1</td>
<td>2</td>
<td>42</td>
<td>32</td>
</tr>
<tr>
<td>best or tied for best</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average rank</td>
<td>3.90</td>
<td>2.88</td>
<td>1.19</td>
<td>1.38</td>
</tr>
<tr>
<td>Maximum ratio of worst found</td>
<td>3.6 (3.33)</td>
<td>3.01 (3.01)</td>
<td>3 (2.02)</td>
<td>3 (2.12)</td>
</tr>
<tr>
<td>solution and lower bound</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(graphs at least 5000 edges)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum range</td>
<td>10</td>
<td>37</td>
<td>24</td>
<td>45</td>
</tr>
</tbody>
</table>

Figure 8.2: Average running times of GRE, CA, and ST. Notice that the axes are logarithmic.

The running time difference of the heuristics is clear. The relative order of the algorithms from the slowest to the fastest is CA, CA1, CA2, and ST. The explanation for the relative order of CA and CA1 (CA2) is that CA1 (CA2) extracts larger planar subgraphs and, therefore, it also finds the
approximation faster. See Figure 8.2 for the average running times of CA and ST as the function of the number of the edges. The curves for CA1 and CA2 are not drawn since the running times of CA1 and CA2 are between that of CA and ST. The sharp turns in the curves are due to the influence of the random test graphs. The running time is higher for a random graph containing the same number of edges as a complete graph. Also, the average running time of GRE (see Section 8.5) is drawn to illustrate the running time differences of the heuristics. The average running times of the heuristics for $K_{400}$ are given in Table 8.6.

Table 8.6: Average running times of heuristics for $K_{400}$ (79800 edges).

<table>
<thead>
<tr>
<th>Heuristic</th>
<th>ST</th>
<th>CA2</th>
<th>CA1</th>
<th>CA</th>
<th>GRE</th>
<th>GCA</th>
<th>GCA1</th>
<th>GCA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>28</td>
<td>42</td>
<td>149</td>
<td>232</td>
<td>83260</td>
<td>93581</td>
<td>90421</td>
<td>82197</td>
</tr>
</tbody>
</table>

The maximum range of the solutions of ST is only 10 ($K_{700}$), the ranges of CA, CA1, and CA2 are 37 ($K_{800}$), 24 ($K_{900}$), and 45 (r9), respectively. Figure 8.3 illustrates the difference of the best found solutions and the worst found solutions of the heuristics. Ranges of the heuristics are collected in Table 8.5. The larger the graph is, the higher is the variation of CA, CA1, and CA2. Sharp upward turns in the curves are due the influence of the complete graphs, for a complete graph the ranges are higher than for random graphs with the same number of edges.

![Figure 8.3: Ranges of the solutions of ST, CA, CA1, and CA2.](image)

CA1 and CA2 approximate thickness of large graphs in a reasonable amount of time. Since their solutions seem to be at least twice the optimal solution, we have compared the ratios of the worst found solutions and the lower bounds, instead of the optimal or the best known solution as in Chapters 6 and 7.

The ratio of the worst found solution and the lower bound was at most 3.6 ($r_{100,1508}$) for ST, 3.01 ($K_{800}$) for CA, 3 (rr1) for CA1, and 3 (rr1) for CA2.
The worst cases happened for graphs with less than or equal to 5000 edges for ST, CA1, and CA2. When only graphs with at least 5000 edges were considered, their worst case ratios improved to 3.33, 2.02, and 2.12, respectively. The worst case instance for all heuristics was $K_{1000}$. See Figure 8.4 for the ratios of the worst found solutions and the lower bounds for graphs with less than or equal to 5000 edges and Figure 8.5 for the larger graphs. The ratios are listed in Table 8.5, the worst case ratios for graphs over 5000 edges are given in the parenthesis.

Figure 8.4: Ratios of the worst found solutions of ST, CA, CA1, and CA2 and the lower bound for graphs with no more than 5000 edges.

Figure 8.5: Ratios of the worst found solutions of ST, CA, CA1, and CA2 and the lower bound for graphs with more than 5000 edges.

8.5 Comparison of GRE, GCA, GCA1, and GCA2

The greedy heuristics were applied only for 31 graphs, since in the case of the bipartite graphs, GCA, GCA1, and GCA2 perform similar as GRE. The greedy algorithms achieved significantly better approximations than their non-greedy variants. For example, GCA1 decreased the solutions of
CA1 30 percentage: CA1 got solution 27 for $K_{90}$, but the GCA1 solution was only 19. The average improvements were about $15 \rightarrow 20$ percentage.

The best algorithm was GCA1 with 29 best solutions found against 25 of GCA2, 13 of GCA, and 10 of GRE. The average ranks for the heuristics were 2.93 for GRE, 2.13 for GCA, 1.06 for GCA1, and 1.19 for GCA2. The ranks are collected in Table 8.7.

Table 8.7: Comparison of the performance of GRE, GCA, GCA1, and GCA2.

<table>
<thead>
<tr>
<th></th>
<th>GRE</th>
<th>GCA</th>
<th>GCA1</th>
<th>GCA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of times heuristic is best or tied for best</td>
<td>10</td>
<td>13</td>
<td>29</td>
<td>25</td>
</tr>
<tr>
<td>Average rank</td>
<td>2.94</td>
<td>2.13</td>
<td>1.06</td>
<td>1.19</td>
</tr>
<tr>
<td>Maximum ratio of worst found solution and lower bound (graphs with at least 10000 edges)</td>
<td>2.5 (2.33)</td>
<td>2.5 (2)</td>
<td>2.5 (2)</td>
<td>2.5 (2)</td>
</tr>
<tr>
<td>Maximum range</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

The running times of the heuristics were similar to that of GRE, although GCA and GCA1 take somewhat longer running time in the case of the largest graphs, as shown in Table 8.6 for $K_{400}$. The average running time of GCA1 is illustrated in Figure 8.6. Since the average running times of GRE, GCA, and GCA2 coincide with that of GCA1, they are not drawn. Figure 8.6 also depicts the average running times for CA1 and SA. The running time differences between the heuristics are remarkable. GCA1 runs 10 to 100 times slower than CA1, and SA is at least 10 times slower than GCA1.

The variation of the solution quality of the greedy algorithms was in general very small. The maximum range of the solutions of GRE was only 4 ($K_{100}$), and the ranges of GCA, GCA1, and GCA2 were 3, 2, and 1, respectively. One reason for the small ranges might be that the number of repeats was quite small for large graphs. The ratios of the worst solutions found and the lower bound were 2.5 for all greedy heuristics. If only graphs with at least 10000 edges were considered, the ratios decreased to 2.33 for GRE and to 2 for the other three heuristics. The ranges and the ratios of the worst found solutions and the lower bound are reported in Table 8.7. The worst case ratios for graphs with at least 10000 edges are given in parenthesis. See also Figure 8.7 for an illustration of the worst case ratios of the greedy heuristics for graphs having over 5000 edges.
8.5. **COMPARISON OF GRE, GCA, GCA1, AND GCA2**

![Graph showing the average running time of SA, GCA1, and CA1.](image)

**Figure 8.6:** Average running time of SA, GCA1, and CA1. Notice that the axes are logarithmic.

![Graph showing the ratios of the worst found solutions of GRE, GCA, GCA1, and GCA2 and the optimal or lower bound for graphs over 5000 edges.](image)

**Figure 8.7:** Ratios of the worst found solutions of GRE, GCA, GCA1, and GCA2 and the optimal or lower bound for graphs over 5000 edges.

There were also graphs for which CA1 and CA2 outperformed GRE. CA1 and CA2 found better approximations for graphs r1, r2, r3, and r4. These results are not very reliable, since for these graphs GRE was run only once. One explanation for the better solution quality of CA1 and CA2 against GRE for large random graphs is that the sizes of the extracted planar subgraphs are larger than they are with GRE. This is illustrated in Figure 8.8 where there are sample traces of ST, CA, CA1, CA2, GRE, and GCA1 for a complete graph with 350 vertices (not included in the test graph set) and for r1. For $K_{350}$, GCA1 found the best solution and GRE found the second best solution. The sizes of the extracted planar subgraphs are of maximum size in the beginning of the runs for both heuristics, but the number of edges in the
extracted subgraphs of GRE decreases more rapidly than that of GCA1. The sizes of the extracted subgraphs of CA, CA1, and CA2 do not vary as much as with GRE and GCA1. The solutions of CA1 and CA2 are of the same quality, and the solutions of ST are slightly worse. For the random graph r1 in the right figure, CA1 and CA2 extract in the beginning much larger planar subgraphs than GRE. Now CA1 and CA2 obtained better approximations. The solution of CA is worse than the solution of GRE, but clearly better than the solution of ST. The sizes of the extracted planar subgraphs of GCA1 were much larger in the beginning than those of CA1 and CA2, and the final solution of GCA1 was the best.

8.6 Comparison of HT, GA, J-M, and SA

There were 17 common test graphs for HT, J-M, and SA, and for 12 of them we have GA solutions. Moreover, almost all their solutions are obtained from
Table 8.8: The best found solutions of the powerful heuristics for thickness.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{10}$</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$K_{15}$</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$K_{20}$</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$K_{30}$</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$K_{40}$</td>
<td>9</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>7 $^\dagger$</td>
</tr>
<tr>
<td>$K_{50}$</td>
<td>11</td>
<td>13</td>
<td>11</td>
<td>11</td>
<td>9 $^\dagger$</td>
</tr>
<tr>
<td>$K_{50}$</td>
<td>13</td>
<td>15</td>
<td>13</td>
<td>-</td>
<td>11 $^\dagger$</td>
</tr>
<tr>
<td>$K_{70}$</td>
<td>15</td>
<td>18</td>
<td>15</td>
<td>-</td>
<td>13 $^\dagger$</td>
</tr>
<tr>
<td>$K_{80}$</td>
<td>17</td>
<td>21</td>
<td>17</td>
<td>-</td>
<td>15 $^\dagger$</td>
</tr>
<tr>
<td>$K_{90}$</td>
<td>19</td>
<td>23</td>
<td>19</td>
<td>-</td>
<td>16 $^\dagger$</td>
</tr>
<tr>
<td>$K_{100}$</td>
<td>21</td>
<td>26</td>
<td>21</td>
<td>-</td>
<td>18 $^\dagger$</td>
</tr>
<tr>
<td>$K_{10,10}$</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$K_{15,15}$</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$K_{20,20}$</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$K_{30,30}$</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>$K_{40,40}$</td>
<td>15</td>
<td>13</td>
<td>12</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>$K_{50,50}$</td>
<td>18</td>
<td>16</td>
<td>14</td>
<td>15</td>
<td>13</td>
</tr>
<tr>
<td>$hc_4$</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>$hc_5$</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>$hc_6$</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>$hc_7$</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>$hc_8$</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>3</td>
</tr>
<tr>
<td>$r_{10,25}$</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>$r_{20,92}$</td>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>$r_{30,230}$</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>$r_{40,311}$</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>$r_{50,495}$</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>$r_{60,556}$</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5</td>
</tr>
<tr>
<td>$r_{70,766}$</td>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>$r_{80,939}$</td>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>$r_{90,1263}$</td>
<td>7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>7</td>
</tr>
<tr>
<td>$r_{100,1508}$</td>
<td>8</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>8</td>
</tr>
</tbody>
</table>

$^\dagger$ Solution is obtained using GCA1 initial solution with parameters $t_l = 0.038$, $t_0 = 0.05$, and $\alpha = 0.999$. 

107
Table 8.9: Comparison of the performance of GCA1, HT, J-M, GA, and SA.

<table>
<thead>
<tr>
<th></th>
<th>GCA1</th>
<th>HT</th>
<th>J-M</th>
<th>GA</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of times heuristic is best or tied for best</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>Average rank</td>
<td>3.25</td>
<td>3.67</td>
<td>2.25</td>
<td>1.5</td>
<td>1</td>
</tr>
<tr>
<td>Maximum ratio of worst found solution and lower bound</td>
<td>1.38</td>
<td>1.43</td>
<td>1.33</td>
<td>1.29</td>
<td>1</td>
</tr>
</tbody>
</table>

a one run study. Therefore, our comparison of these powerful heuristics is not very reliable. It was already shown in our earlier experiments [108] that SA outperforms HT, J-M, and GA. In Table 8.8 there is a list of the best solutions found using these four heuristics, including the best found solutions of GCA1, which was shown to be the best greedy heuristics (GCA1 solutions for complete bipartite graphs are actually same as GRE solutions). Table 8.8 also contains five new SA approximations for complete graphs with the number of vertices varying between 60 and 100. The initial solution was constructed by GCA1. SA was able to decrease the initial solution by at least 2 for each graph. For other graphs, GRE was applied to get the initial solutions.

The number of the best found solutions and the average ranks for common test instances are listed in Table 8.9. Since the number of repeats for the heuristics was mainly one, the ranges are not calculated.

SA is clearly the best heuristic, since for all common test instances, SA found an optimal solution. In terms of the solution quality, the second best heuristic was GA. J-M approximations were better than those of GCA1. Especially J-M produces clearly better solutions for bipartite graphs than GCA1. For complete graphs, solutions were exactly the same. GCA1 approximations for the small random graphs seem to be good, since there was only one graph \(r_{20,92}\), for which SA found a better solution.

The worst case ratios of the approximations and the optimal solutions for heuristics GCA1, HT, J-M, and GA were 1.38, 1.43, 1.33, and 1.29, respectively. The ratios are collected in Table 8.9.

In Figure 8.9, there is a graphical representation of sample traces of SA for graphs \(K_{70,40}\), \(K_{30,30,30}\) in the left and \(K_{40}, K_{15,15,15}\), and for random graphs with 100 (\(r_{100,1508}\)) and 85 (\(r_{85,1101}\)) vertices in the right. Graphs \(K_{70,40}, K_{30,30,30} K_{15,15,15},\) and \(r_{85,1101}\) are not included in the test graph set, more detailed approximations for these graphs are given our earlier experiments [108]. The \(x\)-axis shows the running time in seconds and the \(y\)-axis shows the cur-
8.7 **SA solutions for some** $K_{m,n}$ **and** $K_{n,n,n}$

By Theorem 3.15, thickness is known for almost all complete bipartite graphs. Beineke [10] writes that

“To the best of our knowledge, no progress has been made on this problem in the past thirty years”.

Next we give a constructive proof that Theorem 3.15, which solves the thickness of complete bipartite graphs $K_{m,n}$ for almost all values of $m$ and $n$, where $m \leq n$, also holds for all previously unknown values $m < 30$. 

![Figure 8.9: Traces of the annealing phase for the complete graphs $K_{70,40}$, $K_{30,30,30}$, $K_{40}$, $r_{100,1508}$, $K_{15,15,15}$, and for a random graphs with 85 vertices and 1101 edges ($r_{85,1101}$).](image)
Table 8.10: SA results for some complete bipartite and tripartite graphs.

<table>
<thead>
<tr>
<th>Graph</th>
<th>Edges</th>
<th>$\Theta(K_{m,n})</th>
<th>SA</th>
<th>Graph</th>
<th>Edges</th>
<th>LB</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{13,17}$</td>
<td>221</td>
<td>4</td>
<td>4</td>
<td>$K_{3,3,3}$</td>
<td>27</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$K_{17,21}$</td>
<td>357</td>
<td>5</td>
<td>5</td>
<td>$K_{4,4,4}$</td>
<td>48</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$K_{19,29}$</td>
<td>551</td>
<td>6</td>
<td>6</td>
<td>$K_{5,5,5}$</td>
<td>75</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>$K_{19,47}$</td>
<td>893</td>
<td>7</td>
<td>7</td>
<td>$K_{6,6,6}$</td>
<td>108</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$K_{21,25}$</td>
<td>525</td>
<td>6</td>
<td>6</td>
<td>$K_{7,7,7}$</td>
<td>147</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$K_{23,75}$</td>
<td>1725</td>
<td>9</td>
<td>9</td>
<td>$K_{8,8,8}$</td>
<td>192</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>$K_{25,29}$</td>
<td>725</td>
<td>7</td>
<td>7</td>
<td>$K_{9,9,9}$</td>
<td>243</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$K_{25,59}$</td>
<td>1475</td>
<td>9</td>
<td>9</td>
<td>$K_{10,10,10}$</td>
<td>300</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>$K_{27,71}$</td>
<td>1917</td>
<td>10</td>
<td>10</td>
<td>$K_{15,15,15}$</td>
<td>675</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>$K_{29,33}$</td>
<td>957</td>
<td>8</td>
<td>8</td>
<td>$K_{20,20,20}$</td>
<td>1200</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>$K_{29,129}$</td>
<td>3741</td>
<td>12</td>
<td>12</td>
<td>$K_{25,25,25}$</td>
<td>1875</td>
<td>9</td>
<td>12</td>
</tr>
</tbody>
</table>

In Table 8.10 we have listed the first 11 pairs of $m$ and $n$ with unknown thickness ($K_{m,n}$), the number of edges (edges) and the thickness derived from Theorem 3.15 ($\Theta(K_{m,n})$), and the solution obtained by using $SA^2$ with the same parameters setting as described in Section 8.3. Actually the thickness of $K_{13,17}$ was already solved by Beineke et al. [12] as a special case.

Our results verify that the general formula for the thickness of complete bipartite graphs $K_{m,n}$, $m \leq n$, holds for all $m < 30$. The planar decomposition of $K_{19,29}$ into six subgraphs is given in Appendix E.

Table 8.10 also contains $SA$ approximations for some complete tripartite graphs $K_{n,n,n}$, where $n \leq 25$. For these graphs we have listed the number of edges in the graph (edges) and the lower bound (LB) for thickness. Finally $SA$ approximation is given (SA). Theorem 3.18 shows that $\Theta(K_{20}) \leq 10$ and Theorem 3.19 shows that $\Theta(K_{20}) \geq 7$. Since $SA$ approximation is 9, the upper bound taken from Theorem 3.18 is not tight.

### 8.8 Outerhickness experiments

The best found approximations of ST, CA, CA1, GRE, GCA, GCA1, and $SA$ for the outerthickness problem are listed in Table 8.11. The solutions of ST, CA, and CA1 are the same as in Table 8.3. It was shown in Section 8.4 that CA1 produces the best approximations among fast heuristics for thickness and the same holds for outerthickness.

---

### 8.8. OUTERHICKNESS EXPERIMENTS

Table 8.11: The best found solutions for outerthickness.

<table>
<thead>
<tr>
<th>graph</th>
<th>ST</th>
<th>CA</th>
<th>CA1</th>
<th>GRE</th>
<th>GCA</th>
<th>GCA1</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{10}$</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$K_{15}$</td>
<td>8</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>$K_{20}$</td>
<td>11</td>
<td>9</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$K_{30}$</td>
<td>16</td>
<td>13</td>
<td>10</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>$K_{40}$</td>
<td>21</td>
<td>18</td>
<td>12</td>
<td>15</td>
<td>13</td>
<td>12</td>
<td>11</td>
</tr>
<tr>
<td>$K_{50}$</td>
<td>27</td>
<td>22</td>
<td>15</td>
<td>18</td>
<td>16</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>$K_{60}$</td>
<td>32</td>
<td>27</td>
<td>18</td>
<td>22</td>
<td>20</td>
<td>17</td>
<td>16†</td>
</tr>
<tr>
<td>$K_{70}$</td>
<td>38</td>
<td>32</td>
<td>21</td>
<td>26</td>
<td>23</td>
<td>20</td>
<td>18†</td>
</tr>
<tr>
<td>$K_{80}$</td>
<td>43</td>
<td>36</td>
<td>24</td>
<td>28</td>
<td>26</td>
<td>23</td>
<td>21†</td>
</tr>
<tr>
<td>$K_{90}$</td>
<td>48</td>
<td>41</td>
<td>27</td>
<td>33</td>
<td>29</td>
<td>26</td>
<td>24†</td>
</tr>
<tr>
<td>$K_{100}$</td>
<td>54</td>
<td>45</td>
<td>30</td>
<td>37</td>
<td>32</td>
<td>29</td>
<td>26†</td>
</tr>
</tbody>
</table>

| $K_{10,10}$ | 6  | -  | -   | 4   | -   | -    | 4  |
| $K_{15,15}$ | 8  | -  | -   | 6   | -   | -    | 6  |
| $K_{20,20}$ | 11 | -  | -   | 7   | -   | -    | 7  |
| $K_{30,30}$ | 16 | -  | -   | 11  | -   | -    | 11 |
| $K_{40,40}$ | 22 | -  | -   | 15  | -   | -    | 15 |
| $K_{50,50}$ | 27 | -  | -   | 19  | -   | -    | 18 |

| $hc_4$   | 3  | -  | -   | 2   | -   | -    | 2  |
| $hc_5$   | 3  | -  | -   | 2   | -   | -    | 2  |
| $hc_6$   | 4  | -  | -   | 3   | -   | -    | 3  |
| $hc_7$   | 4  | -  | -   | 3   | -   | -    | 3  |
| $hc_8$   | 5  | -  | -   | 3   | -   | -    | 3  |

| $r_{10,25}$ | 3  | 2  | 2   | 2   | 2   | 2    | 2  |
| $r_{20,92}$ | 6  | 3  | 3   | 4   | 3   | 3    | 3  |
| $r_{30,230}$| 9  | 5  | 5   | 6   | 5   | 5    | 5  |
| $r_{40,311}$| 9  | 6  | 5   | 6   | 6   | 5    | 5  |
| $r_{50,495}$| 11 | 7  | 7   | 7   | 7   | 7    | 6  |
| $r_{60,556}$| 10 | 7  | 6   | 7   | 7   | 6    | 6  |
| $r_{70,766}$| 13 | 8  | 7   | 8   | 8   | 7    | 7  |
| $r_{80,939}$| 13 | 9  | 8   | 9   | 9   | 8    | 7  |
| $r_{90,1263}$| 16 | 10 | 9   | 10  | 10  | 9    | 9  |
| $r_{100,1508}$| 17 | 11 | 10  | 11  | 11  | 10   | 9  |

† Solution obtained using initial solution constructed by GCA1 with parameters $t_l = 0.038$, $t_0 = 0.05$ and $\alpha = 0.999$. 

---

111
Table 8.12: Comparison of the performance of outerthickness heuristics.

<table>
<thead>
<tr>
<th></th>
<th>ST</th>
<th>CA</th>
<th>CA1</th>
<th>GRE</th>
<th>GCA</th>
<th>GCA1</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of times heuristic is best or tied for best</td>
<td>0</td>
<td>3</td>
<td>8</td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>21</td>
</tr>
<tr>
<td>Average rank</td>
<td>7</td>
<td>4.43</td>
<td>1.95</td>
<td>3.90</td>
<td>3.29</td>
<td>1.57</td>
<td>1</td>
</tr>
<tr>
<td>Maximum ratio of worst found solution and lower bound</td>
<td>2.3</td>
<td>2</td>
<td>1.67</td>
<td>1.7</td>
<td>1.5</td>
<td>1.5</td>
<td>1.14</td>
</tr>
<tr>
<td>Maximum range</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 8.12 collects the comparison data of the heuristics for their 21 common test graphs (complete and random graphs with no more than 100 vertices). SA was able to improve the solutions of GRE and GCA1 almost always. For every test instance (including bipartite graphs), SA found the best solution.

The second best algorithm was GCA1, which found tied best solution for nine graphs. The third algorithm was, surprisingly, CA1 with eight tied best solution. The average ranks for these heuristics were 1.57 and 1.95, respectively. GCA, GRE, CA, and ST produced worse approximations, their average ranks were 3.29, 3.90, 4.43, and 7, respectively.

Since the running times of the outerthickness heuristics are quite similar to the corresponding thickness versions, Figure 8.6 also illustrates well the differences of the outerthickness algorithms. In practice, the running times of GCA1 and SA are slightly higher for outerthickness than for thickness. This is due to the fact that the extracted outerplanar subgraphs do not contain as many edges as the corresponding planar subgraphs.

![Figure 8.10: Ratios of the worst solutions found of ST, GRE, CA1, GCA1, and SA and the lower bound for graphs with at most 5000 edges.](image)

In general, the solution quality of SA was high, the algorithm found almost always the optimal solution. The worst case ratio of the best found solution and the lower bound never exceeded 1.14, as illustrated in Figure 8.10. The worst case ratios of GCA1 and GCA were the same, 1.5. For CA1 and
GRE, the ratios were approximately 1.7. For CA and ST ratios were 2 and 2.3, respectively. Figure 8.10 shows the worst case ratios of ST, GRE, CA1, and GCA1 for graphs with no more than 5000 edges.

![Figure 8.10: Ratios of the worst solutions found of ST, GRE, CA1, and GCA1 for graphs with no more than 5000 edges for outerthickness.]

Figure 8.11: Ratios of the worst solutions found of ST, CA, and CA1 and the lower bound for graphs with more than 5000 edges for outerthickness.

Figure 8.11 shows the worst case ratios of the worst found solutions and the lower bound of ST, CA, and CA1 for large complete and random graphs given in Table 8.2. The solutions were at most 1.5, 2, and 2.3 times the lower bound, respectively. The worst case ratios for all heuristics are given in Table 8.12.

The highest range of the solutions was 6 ($K_{90}$) for GRE heuristic. The ranges for the other heuristics were close to 3, except that for SA, which was only 1. This is due to the small number of repeats. The exact ranges for the heuristics are listed in Table 8.12.

8.9 Summary

In this chapter we showed that CA1 produces the best approximations among fast heuristics for the thickness problem. GCA1 is an efficient method for non-bipartite graphs, and SA produces the best approximations in general, since it can improve the solutions of the other heuristics almost always. Only GRE and GCA1 initial solutions for SA were tested, but for bipartite graphs, J-M seems to be a possible choice to generate initial solutions.

SA found the best approximations for the outerthickness problem. Quite often SA solutions were optimal or close to optimum. CA1 found usually similar approximations as its greedy variant GCA1. Other tested heuristics found much worse approximations for the outerthickness.
Chapter 9

Conclusions

In this work new algorithms were developed and experimented for MPS, MOPS, thickness, and outer thickness of a graph. Furthermore, two new theoretical results for the thickness and outer thickness were given.

The main theoretical contributions of this work were two new algorithms, CA1 and CA2. It was shown that the performance ratio of both algorithms is at least same as that of CA [26], and that the algorithms run in linear time for graphs of a bounded degree. CA1 approximates MPS and MOPS, but CA2 only MPS. It was shown that the performance ratios of CA1 and CA2 are at least 7/18 and at most 1/2 for MPS. The performance ratio of CA1 for MOPS is at least 7/12 and at most 1/2. It was conjectured that the performance ratio of CA1 for MPS is at least 4/9 and for MOPS exactly 2/3. A clear goal for future research is to solve the performance ratios of CA1 and CA2.

Although CA2 produced slightly better approximations for MPS than CA1, it is possible that the performance ratio of CA2 is lower than the ratio of CA1 due to the possibility of adding triangles too greedy in Phase 1.

Moreover, the status of the relative performance of CA_M and the new algorithms is open. The performance ratio of CA_M is 4/9, but the best known way to implement the algorithm requires $O(m^{3/2} \log^6 n)$ time [26]. Algorithms CA1 and CA2 are much simpler to implement.

The approximations of CA1 and CA2 were in our experiments still quite distant from the optima for MPS, and it was always possible to improve their solution by SA. The simulated annealing was demonstrated to be the best known approximation method for MPS.

Since SA performs numerous planarity tests during its execution, SA is not a good method for large graphs due to its high running time. One way to make SA to run faster is to implement a dynamic planarity testing algorithm. The implementation of such an algorithm might be a hard task. Another way
to improve the solution quality of SA is to find better heuristics than pure randomness to choose edges. Perhaps some kind of degree based heuristics to choose edges could make the algorithm converge faster. It was shown that \textit{SA}_{CA1} usually finds better approximations than \textit{SA}_{E}, so a good initial solution shortens the running time and improves the solution quality. But so far, \textit{CA1} and \textit{CA2} are the fastest methods to construct reasonable quality initial solutions. It was also described shortly how it is possible to apply SA to weighted planar and outerplanar subgraph problems.

\textit{CA1} and \textit{SA}_{CA1} produced the best approximations for MOPS. There were test instances for which \textit{SA}_{CA1} was not able to improve the solutions anymore. Experiments showed that \textit{CA1} gives very good approximations in practice, the worst case ratio of the worst found solution of \textit{CA1} and the optimal or lower bound was always at least 0.81. The solutions of \textit{SA}_{CA1} were very close to the optima.

The greedy versions of the new algorithms, \textit{GCA1} and \textit{GCA2}, found good approximations for MPS. These greedy algorithms are good alternatives for SA, if the computation time is limited.

\textit{CA1} and \textit{CA2} can be used to approximate the thickness and outerthickness of a graph. Kawano and Yamazaki [74] showed that it makes no difference in the theoretical point of view whether we extract maximal planar subgraphs or maximum planar subgraphs from the input graph in the case of thickness. Both approaches guarantee only solutions that are \(\Omega(\log n)\) times the optimal for a graph with \(n\) vertices [74]. This poses interesting questions: Does this also hold for outerthickness? What is the performance ratio of \textit{CA1} and \textit{CA2} for thickness and outerthickness?

Also a new simulated annealing algorithm was given for approximating thickness and outerthickness. For thickness, SA found better solutions than any earlier heuristic. On the other hand, the algorithm was not able to find the optimal solution for every graph. It seems to be possible that the solution quality can still be improved by allowing more time for the annealing process. The efficiency of SA was shown by solving several previously unknown cases of the thickness of complete bipartite graphs. Unfortunately, it seems to be very hard to recognize any combinatorial patterns from these solutions to find a mathematical proof for the unknown cases.

To improve the solution quality of SA for thickness and outerthickness, the same methods as in the case of MPS and MOPS may help: an implementation of a dynamic planarity test, better initial solutions, and wiser ways to choose the edges from the subsets.

Further, two theoretical results were given for thickness and outerthick-
ness. The first result gave a new upper bound for the thickness of complete tripartite graphs. It was shown by experiments that the upper bound is not tight by obtaining better approximations by SA. Thickness and outerthickness are known well only for complete graphs, complete bipartite graphs and hypercubes, so the exact thickness of many graph classes is open (for example, $k$-partite graphs, with $3 \leq k \leq n - 1$, where $n$ is the number of vertices in the graph).

As the second theoretical result, we introduced lower and upper bounds for outerthickness in the terms of minimum and maximum degree, similarly as Wessel [130] and Halton [58] for thickness.

Dean et al. [37] have conjectured that for an arbitrary graph $G$ with $m$ edges it holds that $\Theta(G) \leq \sqrt{m/16} + O(1)$. We conjectured that $\Theta_o(G) \leq \sqrt{m/8} + O(1)$. The best known upper bounds are $\Theta(G) \leq \lceil \sqrt{m/3} + 3/2 \rceil$ and $\Theta_o(G) \leq \lceil \sqrt{m/2} \rceil$ [37]. The gap between the best proved and the conjectured upper bound is considerable.

The complexity status of the outerthickness is open, but since thickness and MOPS are $NP$-complete, it was conjectured that determining the outerthickness of a graph is also $NP$-complete.
Appendix A

Statistics for MPS

A.1 CA and CA1 statistics

Table A.1: CA and CA1 solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
</tr>
</thead>
<tbody>
<tr>
<td>cimi-g1</td>
<td>100</td>
<td>0.001</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>0.001</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>cimi-g2</td>
<td>50</td>
<td>0.001</td>
<td>86</td>
<td>86.82</td>
<td>88</td>
<td>0.002</td>
<td>114</td>
<td>116.18</td>
<td>117</td>
</tr>
<tr>
<td>cimi-g3</td>
<td>100</td>
<td>0.002</td>
<td>37</td>
<td>37.79</td>
<td>38</td>
<td>0.002</td>
<td>49</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td>cimi-g4</td>
<td>100</td>
<td>0.001</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>0.001</td>
<td>14</td>
<td>15.75</td>
<td>16</td>
</tr>
<tr>
<td>cimi-g5</td>
<td>100</td>
<td>0.002</td>
<td>58</td>
<td>58.77</td>
<td>59</td>
<td>0.002</td>
<td>73</td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>g10</td>
<td>100</td>
<td>0.002</td>
<td>42</td>
<td>42</td>
<td>42</td>
<td>0.001</td>
<td>42</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>g11</td>
<td>100</td>
<td>0.001</td>
<td>34</td>
<td>35.21</td>
<td>36</td>
<td>0.001</td>
<td>46</td>
<td>46.95</td>
<td>47</td>
</tr>
<tr>
<td>g12</td>
<td>100</td>
<td>0.002</td>
<td>34</td>
<td>35.07</td>
<td>36</td>
<td>0.002</td>
<td>46</td>
<td>46.98</td>
<td>47</td>
</tr>
<tr>
<td>g13</td>
<td>25</td>
<td>0.004</td>
<td>71</td>
<td>72.4</td>
<td>73</td>
<td>0.006</td>
<td>95</td>
<td>96.76</td>
<td>97</td>
</tr>
<tr>
<td>g14</td>
<td>25</td>
<td>0.006</td>
<td>72</td>
<td>72.64</td>
<td>73</td>
<td>0.006</td>
<td>97</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td>g15</td>
<td>25</td>
<td>0.008</td>
<td>72</td>
<td>72.72</td>
<td>73</td>
<td>0.007</td>
<td>97</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td>g16</td>
<td>25</td>
<td>0.009</td>
<td>134</td>
<td>135.72</td>
<td>137</td>
<td>0.009</td>
<td>155</td>
<td>158.52</td>
<td>162</td>
</tr>
<tr>
<td>g17</td>
<td>25</td>
<td>0.012</td>
<td>143</td>
<td>144.56</td>
<td>147</td>
<td>0.012</td>
<td>187</td>
<td>189.88</td>
<td>194</td>
</tr>
<tr>
<td>g18</td>
<td>25</td>
<td>0.011</td>
<td>144</td>
<td>146</td>
<td>147</td>
<td>0.013</td>
<td>192</td>
<td>194.24</td>
<td>197</td>
</tr>
<tr>
<td>g19</td>
<td>25</td>
<td>0.018</td>
<td>214</td>
<td>216.04</td>
<td>218</td>
<td>0.019</td>
<td>262</td>
<td>268.44</td>
<td>274</td>
</tr>
<tr>
<td>r0</td>
<td>25</td>
<td>0.395</td>
<td>1461</td>
<td>1468.72</td>
<td>1475</td>
<td>0.408</td>
<td>1723</td>
<td>1748.12</td>
<td>1766</td>
</tr>
<tr>
<td>r1</td>
<td>25</td>
<td>0.845</td>
<td>1491</td>
<td>1493.8</td>
<td>1497</td>
<td>0.864</td>
<td>1997</td>
<td>1997</td>
<td>1997</td>
</tr>
<tr>
<td>crack</td>
<td>25</td>
<td>0.755</td>
<td>15074</td>
<td>15092.5</td>
<td>15107</td>
<td>0.814</td>
<td>19948</td>
<td>19978.6</td>
<td>20067</td>
</tr>
</tbody>
</table>
### Table A.2: More CA and CA1 solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>CA</th>
<th>CA1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ave t.</td>
<td>worst</td>
</tr>
<tr>
<td>tg100.1</td>
<td>25</td>
<td>0.005</td>
<td>118</td>
</tr>
<tr>
<td>tg100.2</td>
<td>25</td>
<td>0.007</td>
<td>118</td>
</tr>
<tr>
<td>tg100.3</td>
<td>25</td>
<td>0.008</td>
<td>119</td>
</tr>
<tr>
<td>tg100.4</td>
<td>25</td>
<td>0.006</td>
<td>125</td>
</tr>
<tr>
<td>tg100.5</td>
<td>25</td>
<td>0.009</td>
<td>134</td>
</tr>
<tr>
<td>tg150.1</td>
<td>25</td>
<td>0.009</td>
<td>171</td>
</tr>
<tr>
<td>tg150.2</td>
<td>25</td>
<td>0.009</td>
<td>175</td>
</tr>
<tr>
<td>tg150.3</td>
<td>25</td>
<td>0.010</td>
<td>171</td>
</tr>
<tr>
<td>tg150.4</td>
<td>25</td>
<td>0.011</td>
<td>178</td>
</tr>
<tr>
<td>tg150.5</td>
<td>25</td>
<td>0.011</td>
<td>177</td>
</tr>
<tr>
<td>tg200.1</td>
<td>25</td>
<td>0.011</td>
<td>222</td>
</tr>
<tr>
<td>tg200.2</td>
<td>25</td>
<td>0.013</td>
<td>215</td>
</tr>
<tr>
<td>tg200.3</td>
<td>25</td>
<td>0.013</td>
<td>235</td>
</tr>
<tr>
<td>tg200.4</td>
<td>25</td>
<td>0.014</td>
<td>236</td>
</tr>
<tr>
<td>tg200.5</td>
<td>25</td>
<td>0.014</td>
<td>235</td>
</tr>
<tr>
<td>tg300.1</td>
<td>25</td>
<td>0.017</td>
<td>324</td>
</tr>
<tr>
<td>tg300.2</td>
<td>25</td>
<td>0.028</td>
<td>355</td>
</tr>
<tr>
<td>tg300.3</td>
<td>25</td>
<td>0.026</td>
<td>359</td>
</tr>
<tr>
<td>tg300.4</td>
<td>25</td>
<td>0.031</td>
<td>386</td>
</tr>
<tr>
<td>tg300.5</td>
<td>25</td>
<td>0.036</td>
<td>395</td>
</tr>
<tr>
<td>tg100.1</td>
<td>25</td>
<td>0.007</td>
<td>135</td>
</tr>
<tr>
<td>tg100.3</td>
<td>25</td>
<td>0.007</td>
<td>138</td>
</tr>
<tr>
<td>tg100.5</td>
<td>25</td>
<td>0.007</td>
<td>135</td>
</tr>
<tr>
<td>tg100.7</td>
<td>25</td>
<td>0.006</td>
<td>134</td>
</tr>
<tr>
<td>tg100.9</td>
<td>25</td>
<td>0.006</td>
<td>136</td>
</tr>
<tr>
<td>tg200.1</td>
<td>25</td>
<td>0.012</td>
<td>270</td>
</tr>
<tr>
<td>tg200.3</td>
<td>25</td>
<td>0.011</td>
<td>276</td>
</tr>
<tr>
<td>tg200.5</td>
<td>25</td>
<td>0.012</td>
<td>273</td>
</tr>
<tr>
<td>tg200.7</td>
<td>25</td>
<td>0.012</td>
<td>271</td>
</tr>
<tr>
<td>tg200.9</td>
<td>25</td>
<td>0.012</td>
<td>273</td>
</tr>
</tbody>
</table>
A.2 CA2 and GCA2 statistics

Table A.3: CA2 and GCA2 solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>CA2</th>
<th></th>
<th>GCA2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ave t.</td>
<td>worst</td>
<td>ave</td>
<td>best</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cimi-g1</td>
<td>100</td>
<td>0.001</td>
<td>13</td>
<td>13</td>
<td>13</td>
</tr>
<tr>
<td>cimi-g2</td>
<td>50</td>
<td>0.002</td>
<td>112</td>
<td>115.96</td>
<td>117</td>
</tr>
<tr>
<td>cimi-g3</td>
<td>100</td>
<td>0.002</td>
<td>49</td>
<td>49</td>
<td>49</td>
</tr>
<tr>
<td>cimi-g4</td>
<td>100</td>
<td>0.001</td>
<td>14</td>
<td>15.7</td>
<td>16</td>
</tr>
<tr>
<td>cimi-g5</td>
<td>100</td>
<td>0.002</td>
<td>73</td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>cimi-g6</td>
<td>100</td>
<td>0.002</td>
<td>42</td>
<td>42</td>
<td>42</td>
</tr>
<tr>
<td>g10</td>
<td>100</td>
<td>0.002</td>
<td>46</td>
<td>46.94</td>
<td>47</td>
</tr>
<tr>
<td>g11</td>
<td>100</td>
<td>0.002</td>
<td>46</td>
<td>46.89</td>
<td>47</td>
</tr>
<tr>
<td>g12</td>
<td>100</td>
<td>0.001</td>
<td>46</td>
<td>46.84</td>
<td>47</td>
</tr>
<tr>
<td>g13</td>
<td>25</td>
<td>0.005</td>
<td>96</td>
<td>96.8</td>
<td>97</td>
</tr>
<tr>
<td>g14</td>
<td>25</td>
<td>0.006</td>
<td>96</td>
<td>96.96</td>
<td>97</td>
</tr>
<tr>
<td>g15</td>
<td>25</td>
<td>0.007</td>
<td>96</td>
<td>96.96</td>
<td>97</td>
</tr>
<tr>
<td>g16</td>
<td>25</td>
<td>0.008</td>
<td>158</td>
<td>163.32</td>
<td>167</td>
</tr>
<tr>
<td>g17</td>
<td>25</td>
<td>0.013</td>
<td>191</td>
<td>192.56</td>
<td>196</td>
</tr>
<tr>
<td>g18</td>
<td>25</td>
<td>0.015</td>
<td>193</td>
<td>194.68</td>
<td>197</td>
</tr>
<tr>
<td>g19</td>
<td>25</td>
<td>0.019</td>
<td>269</td>
<td>277.08</td>
<td>285</td>
</tr>
<tr>
<td>r0</td>
<td>25 / 1</td>
<td>0.440</td>
<td>1812</td>
<td>1825.48</td>
<td>1842</td>
</tr>
<tr>
<td>r1</td>
<td>25</td>
<td>1.279</td>
<td>1997</td>
<td>1997</td>
<td>1997</td>
</tr>
<tr>
<td>crack</td>
<td>25</td>
<td>0.903</td>
<td>19948</td>
<td>19979</td>
<td>20086</td>
</tr>
</tbody>
</table>

* The algorithm is not applied for this graph.
### Table A.4: More CA2 and GCA2 solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
</tr>
</thead>
<tbody>
<tr>
<td>rg100.1</td>
<td>25</td>
<td>0.007</td>
<td>124</td>
<td>124.04</td>
<td>125</td>
<td>0.406</td>
<td>148</td>
<td>152.56</td>
<td>157</td>
</tr>
<tr>
<td>rg100.2</td>
<td>25</td>
<td>0.007</td>
<td>127</td>
<td>127</td>
<td>127</td>
<td>0.425</td>
<td>151</td>
<td>156.8</td>
<td>162</td>
</tr>
<tr>
<td>rg100.3</td>
<td>25</td>
<td>0.006</td>
<td>128</td>
<td>128</td>
<td>128</td>
<td>0.499</td>
<td>153</td>
<td>158.64</td>
<td>164</td>
</tr>
<tr>
<td>rg100.4</td>
<td>25</td>
<td>0.008</td>
<td>136</td>
<td>138.16</td>
<td>140</td>
<td>0.547</td>
<td>163</td>
<td>168.04</td>
<td>174</td>
</tr>
<tr>
<td>rg100.5</td>
<td>25</td>
<td>0.010</td>
<td>149</td>
<td>150.68</td>
<td>154</td>
<td>0.599</td>
<td>180</td>
<td>182.48</td>
<td>186</td>
</tr>
<tr>
<td>rg150.1</td>
<td>25</td>
<td>0.010</td>
<td>175</td>
<td>175</td>
<td>175</td>
<td>1.290</td>
<td>208</td>
<td>216.28</td>
<td>223</td>
</tr>
<tr>
<td>rg150.2</td>
<td>25</td>
<td>0.012</td>
<td>182</td>
<td>182</td>
<td>182</td>
<td>1.321</td>
<td>212</td>
<td>219.88</td>
<td>226</td>
</tr>
<tr>
<td>rg150.3</td>
<td>25</td>
<td>0.011</td>
<td>179</td>
<td>179</td>
<td>179</td>
<td>1.551</td>
<td>219</td>
<td>225.72</td>
<td>232</td>
</tr>
<tr>
<td>rg150.4</td>
<td>25</td>
<td>0.011</td>
<td>190</td>
<td>190</td>
<td>190</td>
<td>1.700</td>
<td>228</td>
<td>233.32</td>
<td>238</td>
</tr>
<tr>
<td>rg150.5</td>
<td>25</td>
<td>0.011</td>
<td>184</td>
<td>184.04</td>
<td>185</td>
<td>1.622</td>
<td>220</td>
<td>229.2</td>
<td>236</td>
</tr>
<tr>
<td>rg200.1</td>
<td>25</td>
<td>0.010</td>
<td>227</td>
<td>227</td>
<td>227</td>
<td>2.303</td>
<td>266</td>
<td>274.12</td>
<td>280</td>
</tr>
<tr>
<td>rg200.2</td>
<td>25</td>
<td>0.012</td>
<td>219</td>
<td>219</td>
<td>219</td>
<td>2.417</td>
<td>261</td>
<td>269</td>
<td>277</td>
</tr>
<tr>
<td>rg200.3</td>
<td>25</td>
<td>0.015</td>
<td>244</td>
<td>244</td>
<td>244</td>
<td>2.869</td>
<td>291</td>
<td>299.68</td>
<td>309</td>
</tr>
<tr>
<td>rg200.4</td>
<td>25</td>
<td>0.017</td>
<td>257</td>
<td>258.92</td>
<td>261</td>
<td>3.117</td>
<td>301</td>
<td>307.64</td>
<td>317</td>
</tr>
<tr>
<td>rg200.5</td>
<td>25</td>
<td>0.017</td>
<td>252</td>
<td>252.96</td>
<td>253</td>
<td>3.344</td>
<td>302</td>
<td>307.28</td>
<td>314</td>
</tr>
<tr>
<td>rg300.1</td>
<td>25</td>
<td>0.017</td>
<td>330</td>
<td>330</td>
<td>330</td>
<td>8.764</td>
<td>388</td>
<td>395.92</td>
<td>407</td>
</tr>
<tr>
<td>rg300.2</td>
<td>25</td>
<td>0.031</td>
<td>377</td>
<td>377</td>
<td>377</td>
<td>12.46</td>
<td>441</td>
<td>451.92</td>
<td>461</td>
</tr>
<tr>
<td>rg300.3</td>
<td>25</td>
<td>0.029</td>
<td>377.72</td>
<td>377</td>
<td>12.49</td>
<td>445</td>
<td>452.44</td>
<td>464</td>
<td></td>
</tr>
<tr>
<td>rg300.4</td>
<td>25</td>
<td>0.038</td>
<td>419</td>
<td>421.8</td>
<td>425</td>
<td>16.90</td>
<td>486</td>
<td>497.36</td>
<td>509</td>
</tr>
<tr>
<td>rg300.5</td>
<td>25</td>
<td>0.040</td>
<td>422</td>
<td>426.76</td>
<td>430</td>
<td>16.42</td>
<td>491</td>
<td>503</td>
<td>512</td>
</tr>
<tr>
<td>tg100.1</td>
<td>25</td>
<td>0.006</td>
<td>194</td>
<td>195.88</td>
<td>197</td>
<td>0.368</td>
<td>257</td>
<td>276.56</td>
<td>290</td>
</tr>
<tr>
<td>tg100.3</td>
<td>25</td>
<td>0.009</td>
<td>193</td>
<td>195.68</td>
<td>197</td>
<td>0.426</td>
<td>249</td>
<td>269.52</td>
<td>283</td>
</tr>
<tr>
<td>tg100.5</td>
<td>25</td>
<td>0.008</td>
<td>190</td>
<td>194.72</td>
<td>197</td>
<td>0.480</td>
<td>240</td>
<td>256.72</td>
<td>272</td>
</tr>
<tr>
<td>tg100.7</td>
<td>25</td>
<td>0.008</td>
<td>190</td>
<td>193.72</td>
<td>196</td>
<td>0.539</td>
<td>244</td>
<td>258.12</td>
<td>276</td>
</tr>
<tr>
<td>tg100.9</td>
<td>25</td>
<td>0.010</td>
<td>189</td>
<td>193.76</td>
<td>197</td>
<td>0.599</td>
<td>240</td>
<td>253.16</td>
<td>272</td>
</tr>
<tr>
<td>tg200.1</td>
<td>25</td>
<td>0.013</td>
<td>394</td>
<td>395.96</td>
<td>397</td>
<td>2.016</td>
<td>533</td>
<td>575.72</td>
<td>594</td>
</tr>
<tr>
<td>tg200.3</td>
<td>25</td>
<td>0.014</td>
<td>392</td>
<td>394.96</td>
<td>397</td>
<td>2.147</td>
<td>522</td>
<td>557.6</td>
<td>586</td>
</tr>
<tr>
<td>tg200.5</td>
<td>25</td>
<td>0.017</td>
<td>386</td>
<td>393.84</td>
<td>397</td>
<td>2.294</td>
<td>499</td>
<td>545</td>
<td>578</td>
</tr>
<tr>
<td>tg200.7</td>
<td>25</td>
<td>0.015</td>
<td>392</td>
<td>395.32</td>
<td>397</td>
<td>2.482</td>
<td>523</td>
<td>549.56</td>
<td>589</td>
</tr>
<tr>
<td>tg200.9</td>
<td>25</td>
<td>0.016</td>
<td>389</td>
<td>394.12</td>
<td>397</td>
<td>2.646</td>
<td>522</td>
<td>540.28</td>
<td>566</td>
</tr>
</tbody>
</table>
### A.3 GCA and GCA1 statistics

Table A.5: GCA and GCA1 solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>GCA ave t.</th>
<th>GCA worst</th>
<th>GCA ave</th>
<th>GCA best</th>
<th>GCA ave t.</th>
<th>GCA worst</th>
<th>GCA ave</th>
<th>GCA best</th>
</tr>
</thead>
<tbody>
<tr>
<td>cimi-g1</td>
<td>100</td>
<td>0.004</td>
<td>17</td>
<td>18.04</td>
<td>19</td>
<td>0.003</td>
<td>16</td>
<td>18.01</td>
<td>19</td>
</tr>
<tr>
<td>cimi-g2</td>
<td>50</td>
<td>0.063</td>
<td>142</td>
<td>161.38</td>
<td>165</td>
<td>0.043</td>
<td>153</td>
<td>164.36</td>
<td>165</td>
</tr>
<tr>
<td>cimi-g3</td>
<td>100</td>
<td>0.028</td>
<td>55</td>
<td>72.48</td>
<td>73</td>
<td>0.022</td>
<td>68</td>
<td>72.69</td>
<td>73</td>
</tr>
<tr>
<td>cimi-g4</td>
<td>100</td>
<td>0.003</td>
<td>17</td>
<td>19.21</td>
<td>20</td>
<td>0.002</td>
<td>18</td>
<td>19.27</td>
<td>20</td>
</tr>
<tr>
<td>cimi-g5</td>
<td>100</td>
<td>0.038</td>
<td>79</td>
<td>81.05</td>
<td>82</td>
<td>0.026</td>
<td>79</td>
<td>81.25</td>
<td>82</td>
</tr>
<tr>
<td>cimi-g6</td>
<td>100</td>
<td>0.040</td>
<td>54</td>
<td>57.28</td>
<td>59</td>
<td>0.041</td>
<td>55</td>
<td>56.95</td>
<td>59</td>
</tr>
<tr>
<td>g10</td>
<td>100</td>
<td>0.024</td>
<td>61</td>
<td>66.82</td>
<td>69</td>
<td>0.017</td>
<td>60</td>
<td>67.61</td>
<td>69</td>
</tr>
<tr>
<td>g11</td>
<td>100</td>
<td>0.025</td>
<td>58</td>
<td>66.07</td>
<td>69</td>
<td>0.018</td>
<td>61</td>
<td>67.3</td>
<td>69</td>
</tr>
<tr>
<td>g12</td>
<td>100</td>
<td>0.035</td>
<td>56</td>
<td>60.62</td>
<td>66</td>
<td>0.028</td>
<td>57</td>
<td>63.01</td>
<td>67</td>
</tr>
<tr>
<td>g13</td>
<td>25</td>
<td>0.380</td>
<td>111</td>
<td>116.68</td>
<td>120</td>
<td>0.374</td>
<td>117</td>
<td>123.08</td>
<td>128</td>
</tr>
<tr>
<td>g14</td>
<td>25</td>
<td>0.513</td>
<td>116</td>
<td>123.68</td>
<td>132</td>
<td>0.524</td>
<td>125</td>
<td>128.52</td>
<td>134</td>
</tr>
<tr>
<td>g15</td>
<td>25</td>
<td>0.656</td>
<td>127</td>
<td>131.08</td>
<td>136</td>
<td>0.668</td>
<td>129</td>
<td>133.4</td>
<td>138</td>
</tr>
<tr>
<td>g16</td>
<td>25</td>
<td>0.796</td>
<td>178</td>
<td>184.88</td>
<td>193</td>
<td>0.800</td>
<td>183</td>
<td>191.84</td>
<td>200</td>
</tr>
<tr>
<td>g17</td>
<td>25</td>
<td>1.525</td>
<td>202</td>
<td>211.88</td>
<td>224</td>
<td>1.627</td>
<td>223</td>
<td>229.2</td>
<td>237</td>
</tr>
<tr>
<td>g18</td>
<td>25</td>
<td>2.073</td>
<td>206</td>
<td>219.96</td>
<td>230</td>
<td>2.201</td>
<td>234</td>
<td>237.96</td>
<td>244</td>
</tr>
<tr>
<td>g19</td>
<td>25</td>
<td>4.194</td>
<td>287</td>
<td>297.92</td>
<td>305</td>
<td>4.532</td>
<td>306</td>
<td>318.36</td>
<td>326</td>
</tr>
<tr>
<td>r0</td>
<td>1</td>
<td>1082</td>
<td>1827</td>
<td>1827</td>
<td>1827</td>
<td>1194</td>
<td>1975</td>
<td>1975</td>
<td>1975</td>
</tr>
<tr>
<td>r1</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>r9</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>crack</td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

* The algorithm is not applied for this graph.
Table A.6: More GCA and GCA1 solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>GCA</th>
<th></th>
<th>GCA1</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ave t.</td>
<td>worst</td>
<td>ave</td>
<td>best</td>
</tr>
<tr>
<td>rg100.1</td>
<td>25</td>
<td>0.411</td>
<td>147</td>
<td>151.84</td>
<td>157</td>
</tr>
<tr>
<td>rg100.2</td>
<td>25</td>
<td>0.447</td>
<td>151</td>
<td>154.44</td>
<td>160</td>
</tr>
<tr>
<td>rg100.3</td>
<td>25</td>
<td>0.517</td>
<td>151</td>
<td>156.32</td>
<td>163</td>
</tr>
<tr>
<td>rg100.4</td>
<td>25</td>
<td>0.558</td>
<td>158</td>
<td>166.04</td>
<td>172</td>
</tr>
<tr>
<td>rg100.5</td>
<td>25</td>
<td>0.606</td>
<td>174</td>
<td>180.64</td>
<td>186</td>
</tr>
<tr>
<td>rg150.1</td>
<td>25</td>
<td>1.311</td>
<td>213</td>
<td>216.84</td>
<td>222</td>
</tr>
<tr>
<td>rg150.2</td>
<td>25</td>
<td>1.349</td>
<td>212</td>
<td>218.12</td>
<td>224</td>
</tr>
<tr>
<td>rg150.3</td>
<td>25</td>
<td>1.582</td>
<td>216</td>
<td>221.28</td>
<td>227</td>
</tr>
<tr>
<td>rg150.4</td>
<td>25</td>
<td>1.598</td>
<td>227</td>
<td>229.72</td>
<td>236</td>
</tr>
<tr>
<td>rg150.5</td>
<td>25</td>
<td>1.666</td>
<td>220</td>
<td>229.36</td>
<td>237</td>
</tr>
<tr>
<td>rg200.1</td>
<td>25</td>
<td>2.381</td>
<td>260</td>
<td>272.16</td>
<td>278</td>
</tr>
<tr>
<td>rg200.2</td>
<td>25</td>
<td>2.427</td>
<td>256</td>
<td>266.56</td>
<td>274</td>
</tr>
<tr>
<td>rg200.3</td>
<td>25</td>
<td>2.898</td>
<td>284</td>
<td>294.76</td>
<td>303</td>
</tr>
<tr>
<td>rg200.4</td>
<td>25</td>
<td>3.191</td>
<td>294</td>
<td>302.2</td>
<td>308</td>
</tr>
<tr>
<td>rg200.5</td>
<td>25</td>
<td>3.264</td>
<td>291</td>
<td>301.76</td>
<td>311</td>
</tr>
<tr>
<td>rg300.1</td>
<td>25</td>
<td>8.680</td>
<td>388</td>
<td>394.44</td>
<td>402</td>
</tr>
<tr>
<td>rg300.2</td>
<td>25</td>
<td>12.38</td>
<td>437</td>
<td>444.56</td>
<td>455</td>
</tr>
<tr>
<td>rg300.3</td>
<td>25</td>
<td>12.57</td>
<td>436</td>
<td>446.48</td>
<td>457</td>
</tr>
<tr>
<td>rg300.4</td>
<td>25</td>
<td>15.81</td>
<td>478</td>
<td>486.6</td>
<td>497</td>
</tr>
<tr>
<td>rg300.5</td>
<td>25</td>
<td>16.46</td>
<td>480</td>
<td>492.76</td>
<td>504</td>
</tr>
<tr>
<td>tg100.1</td>
<td>25</td>
<td>0.575</td>
<td>259</td>
<td>279.16</td>
<td>292</td>
</tr>
<tr>
<td>tg100.3</td>
<td>25</td>
<td>0.587</td>
<td>238</td>
<td>260.56</td>
<td>290</td>
</tr>
<tr>
<td>tg100.5</td>
<td>25</td>
<td>0.672</td>
<td>238</td>
<td>248.56</td>
<td>262</td>
</tr>
<tr>
<td>tg100.7</td>
<td>25</td>
<td>0.712</td>
<td>223</td>
<td>237.92</td>
<td>255</td>
</tr>
<tr>
<td>tg100.9</td>
<td>25</td>
<td>0.744</td>
<td>224</td>
<td>239.2</td>
<td>260</td>
</tr>
<tr>
<td>tg200.1</td>
<td>25</td>
<td>3.199</td>
<td>556</td>
<td>574.8</td>
<td>594</td>
</tr>
<tr>
<td>tg200.2</td>
<td>25</td>
<td>3.226</td>
<td>514</td>
<td>546.4</td>
<td>579</td>
</tr>
<tr>
<td>tg200.5</td>
<td>25</td>
<td>3.378</td>
<td>496</td>
<td>526.72</td>
<td>551</td>
</tr>
<tr>
<td>tg200.7</td>
<td>25</td>
<td>3.519</td>
<td>482</td>
<td>515.48</td>
<td>552</td>
</tr>
<tr>
<td>tg200.9</td>
<td>25</td>
<td>3.639</td>
<td>484</td>
<td>508.44</td>
<td>543</td>
</tr>
</tbody>
</table>
### A.4 $SA_E$ and $SA_{CA1}$ statistics

Table A.7: $SA_E$ and $SA_{CA1}$ solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
</tr>
</thead>
<tbody>
<tr>
<td>cimi-g1</td>
<td>20</td>
<td>3</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>1</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>cimi-g2</td>
<td>20</td>
<td>147</td>
<td>165</td>
<td>165</td>
<td>165</td>
<td>92</td>
<td>165</td>
<td>165</td>
<td>165</td>
</tr>
<tr>
<td>cimi-g3</td>
<td>20</td>
<td>28</td>
<td>73</td>
<td>73</td>
<td>73</td>
<td>18</td>
<td>73</td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>cimi-g4</td>
<td>20</td>
<td>3</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>2</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>cimi-g5</td>
<td>20</td>
<td>42</td>
<td>82</td>
<td>82</td>
<td>82</td>
<td>25</td>
<td>82</td>
<td>82</td>
<td>82</td>
</tr>
<tr>
<td>cimi-g6</td>
<td>20</td>
<td>26</td>
<td>59</td>
<td>59</td>
<td>59</td>
<td>15</td>
<td>59</td>
<td>59</td>
<td>59</td>
</tr>
<tr>
<td>g10</td>
<td>20</td>
<td>0</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>0</td>
<td>69</td>
<td>69</td>
<td>69</td>
</tr>
<tr>
<td>g11</td>
<td>20</td>
<td>0</td>
<td>69</td>
<td>69</td>
<td>69</td>
<td>0</td>
<td>69</td>
<td>69</td>
<td>69</td>
</tr>
<tr>
<td>g12</td>
<td>20</td>
<td>12</td>
<td>64</td>
<td>68.35</td>
<td>69</td>
<td>9</td>
<td>63</td>
<td>67.85</td>
<td>69</td>
</tr>
<tr>
<td>g13</td>
<td>10</td>
<td>319</td>
<td>130</td>
<td>133.6</td>
<td>137</td>
<td>166</td>
<td>127</td>
<td>133.3</td>
<td>137</td>
</tr>
<tr>
<td>g14</td>
<td>10</td>
<td>481</td>
<td>139</td>
<td>141.2</td>
<td>143</td>
<td>231</td>
<td>140</td>
<td>141.2</td>
<td>144</td>
</tr>
<tr>
<td>g15</td>
<td>10</td>
<td>580</td>
<td>142</td>
<td>143.3</td>
<td>144</td>
<td>268</td>
<td>141</td>
<td>142.4</td>
<td>144</td>
</tr>
<tr>
<td>g16</td>
<td>10</td>
<td>720</td>
<td>199</td>
<td>203.4</td>
<td>206</td>
<td>404</td>
<td>201</td>
<td>204.9</td>
<td>208</td>
</tr>
<tr>
<td>g17</td>
<td>10</td>
<td>1471</td>
<td>235</td>
<td>239.7</td>
<td>244</td>
<td>770</td>
<td>239</td>
<td>241.4</td>
<td>247</td>
</tr>
<tr>
<td>g18</td>
<td>10</td>
<td>1985</td>
<td>245</td>
<td>250.8</td>
<td>257</td>
<td>969</td>
<td>245</td>
<td>249.1</td>
<td>253</td>
</tr>
<tr>
<td>g19</td>
<td>10</td>
<td>3705</td>
<td>319</td>
<td>326.1</td>
<td>331</td>
<td>2043</td>
<td>322</td>
<td>333.3</td>
<td>339</td>
</tr>
</tbody>
</table>

$\dagger$ Parameters: $t_0 = 0.30$, $t_l = 0.20$ and $\alpha = 0.999$.

$\ddagger$ Parameters: $t_0 = 0.25$, $t_l = 0.20$ and $\alpha = 0.999$. 

125
Table A.8: More $S_A E$ and $S_A C A I$ solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>$S_A E$</th>
<th>$S_A C A I$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ave t.</td>
<td>worst</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_0 = 0.30, t_l = 0.20$ and $\alpha = 0.999$.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>$S_A E$</th>
<th>$S_A C A I$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ave t.</td>
<td>worst</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_0 = 0.25, t_l = 0.20$ and $\alpha = 0.999$.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A.5 Comparison of $SA_{CA1}$ and other heuristics

Table A.9: Comparison statistics of $SA_{CA1}$ and other heuristics.

<table>
<thead>
<tr>
<th>graph</th>
<th>$SA_E$ [^{[109]}] ave t.</th>
<th>$SA_{CA1}$ [^{[\dagger]}] ave t.</th>
<th>GRASP [^{[112]}] time*</th>
<th>2Phase [^{[112]}] sol'n</th>
<th>J-M [^{[112]}] sol'n</th>
</tr>
</thead>
<tbody>
<tr>
<td>cimi-g1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td>cimi-g2</td>
<td>147</td>
<td>165</td>
<td>92</td>
<td>165</td>
<td>165</td>
</tr>
<tr>
<td>cimi-g3</td>
<td>28</td>
<td>73</td>
<td>18</td>
<td>73</td>
<td>63</td>
</tr>
<tr>
<td>cimi-g4</td>
<td>3</td>
<td>20</td>
<td>2</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>cimi-g5</td>
<td>42</td>
<td>82</td>
<td>25</td>
<td>82</td>
<td>82</td>
</tr>
<tr>
<td>cimi-g6</td>
<td>26</td>
<td>59</td>
<td>15</td>
<td>59</td>
<td>58</td>
</tr>
<tr>
<td>g10</td>
<td>0</td>
<td>69</td>
<td>0</td>
<td>69</td>
<td>69</td>
</tr>
<tr>
<td>g11</td>
<td>0</td>
<td>69</td>
<td>0</td>
<td>69</td>
<td>69</td>
</tr>
<tr>
<td>g12</td>
<td>12</td>
<td>69</td>
<td>9</td>
<td>67</td>
<td>65</td>
</tr>
<tr>
<td>g13</td>
<td>319</td>
<td>137</td>
<td>166</td>
<td>135</td>
<td>134</td>
</tr>
<tr>
<td>g14</td>
<td>481</td>
<td>143</td>
<td>231</td>
<td>143</td>
<td>143</td>
</tr>
<tr>
<td>g15</td>
<td>580</td>
<td>144</td>
<td>268</td>
<td>144</td>
<td>144</td>
</tr>
<tr>
<td>g16</td>
<td>720</td>
<td>206</td>
<td>404</td>
<td>196</td>
<td>192</td>
</tr>
<tr>
<td>g17</td>
<td>1471</td>
<td>244</td>
<td>770</td>
<td>236</td>
<td>234</td>
</tr>
<tr>
<td>g18</td>
<td>1985</td>
<td>257</td>
<td>969</td>
<td>236</td>
<td>234</td>
</tr>
<tr>
<td>g19</td>
<td>3705</td>
<td>331</td>
<td>2043</td>
<td>331</td>
<td>313</td>
</tr>
</tbody>
</table>

\[^{\dagger}]\text{Parameters: } t_0 = 0.30, t_l = 0.20 \text{ and } \alpha = 0.999.

\[^{\dagger}]\text{Parameters: } t_0 = 0.25, t_l = 0.20 \text{ and } \alpha = 0.999.

* With the same platform as $SA_E$ and $SA_{CA1}$ approximations.
### Table A.10: More comparison statistics of $SA_{CA_1}$ and other heuristics.

<table>
<thead>
<tr>
<th>graph</th>
<th>$SA_E$*</th>
<th>$SA_{CA_1}$†</th>
<th>GRASP [112]</th>
<th>2Phase [112]</th>
<th>J-M [70]</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ave t.</td>
<td>best</td>
<td>ave t.</td>
<td>best</td>
<td>time</td>
</tr>
<tr>
<td>rg100.1</td>
<td>338</td>
<td>170</td>
<td>187</td>
<td>169</td>
<td>1239</td>
</tr>
<tr>
<td>rg100.2</td>
<td>333</td>
<td>174</td>
<td>195</td>
<td>173</td>
<td>1458</td>
</tr>
<tr>
<td>rg100.3</td>
<td>405</td>
<td>179</td>
<td>225</td>
<td>179</td>
<td>1928</td>
</tr>
<tr>
<td>rg100.4</td>
<td>435</td>
<td>186</td>
<td>265</td>
<td>186</td>
<td>2545</td>
</tr>
<tr>
<td>rg100.5</td>
<td>561</td>
<td>197</td>
<td>315</td>
<td>198</td>
<td>3348</td>
</tr>
<tr>
<td>rg150.1</td>
<td>1000</td>
<td>243</td>
<td>541</td>
<td>241</td>
<td>3856</td>
</tr>
<tr>
<td>rg150.2</td>
<td>1157</td>
<td>241</td>
<td>587</td>
<td>241</td>
<td>4181</td>
</tr>
<tr>
<td>rg150.3</td>
<td>1216</td>
<td>254</td>
<td>648</td>
<td>252</td>
<td>5740</td>
</tr>
<tr>
<td>rg150.4</td>
<td>1291</td>
<td>258</td>
<td>707</td>
<td>259</td>
<td>6633</td>
</tr>
<tr>
<td>rg150.5</td>
<td>1304</td>
<td>261</td>
<td>714</td>
<td>262</td>
<td>6831</td>
</tr>
<tr>
<td>rg200.1</td>
<td>1620</td>
<td>308</td>
<td>871</td>
<td>306</td>
<td>8082</td>
</tr>
<tr>
<td>rg200.2</td>
<td>1638</td>
<td>307</td>
<td>851</td>
<td>301</td>
<td>8239</td>
</tr>
<tr>
<td>rg200.3</td>
<td>2133</td>
<td>334</td>
<td>1240</td>
<td>332</td>
<td>15912</td>
</tr>
<tr>
<td>rg200.4</td>
<td>2440</td>
<td>336</td>
<td>1371</td>
<td>341</td>
<td>19097</td>
</tr>
<tr>
<td>rg200.5</td>
<td>2550</td>
<td>342</td>
<td>1386</td>
<td>344</td>
<td>19728</td>
</tr>
<tr>
<td>rg300.1</td>
<td>6131</td>
<td>453</td>
<td>3350</td>
<td>446</td>
<td>31009</td>
</tr>
<tr>
<td>rg300.2</td>
<td>9600</td>
<td>503</td>
<td>5462</td>
<td>506</td>
<td>88688</td>
</tr>
<tr>
<td>rg300.3</td>
<td>9226</td>
<td>490</td>
<td>5629</td>
<td>499</td>
<td>93412</td>
</tr>
<tr>
<td>rg300.4</td>
<td>13117</td>
<td>530</td>
<td>7659</td>
<td>540</td>
<td>186425</td>
</tr>
<tr>
<td>rg300.5</td>
<td>13781</td>
<td>525</td>
<td>7710</td>
<td>541</td>
<td>203948</td>
</tr>
<tr>
<td>tg100.1</td>
<td>88</td>
<td>294</td>
<td>11</td>
<td>294</td>
<td>1944</td>
</tr>
<tr>
<td>tg100.3</td>
<td>132</td>
<td>294</td>
<td>79</td>
<td>294</td>
<td>2339</td>
</tr>
<tr>
<td>tg100.5</td>
<td>410</td>
<td>294</td>
<td>188</td>
<td>294</td>
<td>2825</td>
</tr>
<tr>
<td>tg100.7</td>
<td>721</td>
<td>294</td>
<td>407</td>
<td>294</td>
<td>3323</td>
</tr>
<tr>
<td>tg100.9</td>
<td>837</td>
<td>290</td>
<td>525</td>
<td>291</td>
<td>3729</td>
</tr>
<tr>
<td>tg200.1</td>
<td>79</td>
<td>594</td>
<td>38</td>
<td>594</td>
<td>11865</td>
</tr>
<tr>
<td>tg200.3</td>
<td>458</td>
<td>594</td>
<td>165</td>
<td>594</td>
<td>12433</td>
</tr>
<tr>
<td>tg200.5</td>
<td>1560</td>
<td>594</td>
<td>472</td>
<td>594</td>
<td>15006</td>
</tr>
<tr>
<td>tg200.7</td>
<td>2542</td>
<td>594</td>
<td>1020</td>
<td>594</td>
<td>17294</td>
</tr>
<tr>
<td>tg200.9</td>
<td>2606</td>
<td>594</td>
<td>1354</td>
<td>594</td>
<td>18883</td>
</tr>
</tbody>
</table>

† Parameters: $t_0 = 0.30$, $t_l = 0.20$ and $\alpha = 0.999$.

‡ Parameters: $t_0 = 0.25$, $t_l = 0.20$ and $\alpha = 0.999$.

* With the same platform as $SA_E$ and $SA_{CA_1}$ approximations.
## Appendix B

### Statistics for MOPS

#### B.1 CA and CA1 statistics

Table B.1: CA and CA1 solution statistics for graphs containing an outerplanar subgraph of maximum size.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
</tr>
</thead>
<tbody>
<tr>
<td>o100.1</td>
<td>25</td>
<td>0.009</td>
<td>140</td>
<td>142.92</td>
<td>144</td>
<td>0.010</td>
<td>191</td>
<td>193.36</td>
<td>196</td>
</tr>
<tr>
<td>o100.2</td>
<td>25</td>
<td>0.012</td>
<td>140</td>
<td>141.96</td>
<td>144</td>
<td>0.012</td>
<td>186</td>
<td>190.4</td>
<td>194</td>
</tr>
<tr>
<td>o100.3</td>
<td>25</td>
<td>0.024</td>
<td>140</td>
<td>143.04</td>
<td>145</td>
<td>0.012</td>
<td>179</td>
<td>187.08</td>
<td>194</td>
</tr>
<tr>
<td>o100.4</td>
<td>25</td>
<td>0.016</td>
<td>142</td>
<td>143.36</td>
<td>145</td>
<td>0.016</td>
<td>184</td>
<td>189.04</td>
<td>195</td>
</tr>
<tr>
<td>o100.5</td>
<td>25</td>
<td>0.023</td>
<td>142</td>
<td>144.8</td>
<td>147</td>
<td>0.020</td>
<td>188</td>
<td>191.24</td>
<td>194</td>
</tr>
<tr>
<td>o200.1</td>
<td>25</td>
<td>0.016</td>
<td>288</td>
<td>290.2</td>
<td>292</td>
<td>0.014</td>
<td>396</td>
<td>396.52</td>
<td>397</td>
</tr>
<tr>
<td>o200.2</td>
<td>25</td>
<td>0.022</td>
<td>289</td>
<td>290.52</td>
<td>292</td>
<td>0.022</td>
<td>390</td>
<td>393.56</td>
<td>397</td>
</tr>
<tr>
<td>o200.3</td>
<td>25</td>
<td>0.025</td>
<td>285</td>
<td>288.84</td>
<td>291</td>
<td>0.024</td>
<td>379</td>
<td>387.36</td>
<td>396</td>
</tr>
<tr>
<td>o200.4</td>
<td>25</td>
<td>0.028</td>
<td>284</td>
<td>286.56</td>
<td>289</td>
<td>0.026</td>
<td>372</td>
<td>383.36</td>
<td>391</td>
</tr>
<tr>
<td>o200.5</td>
<td>25</td>
<td>0.032</td>
<td>286</td>
<td>288.56</td>
<td>291</td>
<td>0.030</td>
<td>372</td>
<td>380.6</td>
<td>394</td>
</tr>
</tbody>
</table>
## B.2 GCA and GCA1 statistics

Table B.2: GCA and GCA1 solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>GCA</th>
<th>GCA1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ave t.</td>
<td>worst</td>
</tr>
<tr>
<td>cimi-g1</td>
<td>100</td>
<td>0.004</td>
<td>13</td>
</tr>
<tr>
<td>cimi-g2</td>
<td>50</td>
<td>0.067</td>
<td>105</td>
</tr>
<tr>
<td>cimi-g3</td>
<td>100</td>
<td>0.028</td>
<td>45</td>
</tr>
<tr>
<td>cimi-g4</td>
<td>100</td>
<td>0.004</td>
<td>14</td>
</tr>
<tr>
<td>cimi-g5</td>
<td>100</td>
<td>0.047</td>
<td>74</td>
</tr>
<tr>
<td>cimi-g6</td>
<td>100</td>
<td>0.045</td>
<td>44</td>
</tr>
<tr>
<td>g10</td>
<td>100</td>
<td>0.023</td>
<td>42</td>
</tr>
<tr>
<td>g11</td>
<td>100</td>
<td>0.023</td>
<td>43</td>
</tr>
<tr>
<td>g12</td>
<td>100</td>
<td>0.034</td>
<td>42</td>
</tr>
<tr>
<td>g13</td>
<td>25</td>
<td>0.391</td>
<td>85</td>
</tr>
<tr>
<td>g14</td>
<td>25</td>
<td>0.563</td>
<td>87</td>
</tr>
<tr>
<td>g15</td>
<td>25</td>
<td>0.690</td>
<td>90</td>
</tr>
<tr>
<td>g16</td>
<td>25</td>
<td>1.001</td>
<td>149</td>
</tr>
<tr>
<td>g17</td>
<td>25</td>
<td>1.951</td>
<td>163</td>
</tr>
<tr>
<td>g18</td>
<td>25</td>
<td>2.575</td>
<td>169</td>
</tr>
<tr>
<td>g19</td>
<td>25</td>
<td>5.843</td>
<td>238</td>
</tr>
<tr>
<td>r0</td>
<td>1</td>
<td>1766</td>
<td>1601</td>
</tr>
<tr>
<td>r1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>r9</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>crack</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

* The algorithm is not applied for this graph.
### Table B.3: More GCA and GCA1 solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
<th>GCA1</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
</tr>
</thead>
<tbody>
<tr>
<td>rg100.1</td>
<td>25</td>
<td>0.517</td>
<td>126</td>
<td>130.36</td>
<td>134</td>
<td>0.521</td>
<td>129</td>
<td>131.96</td>
<td>135</td>
<td></td>
</tr>
<tr>
<td>rg100.2</td>
<td>25</td>
<td>0.554</td>
<td>128</td>
<td>131.44</td>
<td>135</td>
<td>0.558</td>
<td>131</td>
<td>133.16</td>
<td>136</td>
<td></td>
</tr>
<tr>
<td>rg100.3</td>
<td>25</td>
<td>0.610</td>
<td>129</td>
<td>132.76</td>
<td>137</td>
<td>0.604</td>
<td>135</td>
<td>137.72</td>
<td>141</td>
<td></td>
</tr>
<tr>
<td>rg100.4</td>
<td>25</td>
<td>0.708</td>
<td>137</td>
<td>140.04</td>
<td>144</td>
<td>0.673</td>
<td>141</td>
<td>143.88</td>
<td>147</td>
<td></td>
</tr>
<tr>
<td>rg100.5</td>
<td>25</td>
<td>0.796</td>
<td>147</td>
<td>151.28</td>
<td>156</td>
<td>0.774</td>
<td>153</td>
<td>156.32</td>
<td>158</td>
<td></td>
</tr>
<tr>
<td>rg150.1</td>
<td>25</td>
<td>1.698</td>
<td>182</td>
<td>187.24</td>
<td>191</td>
<td>1.706</td>
<td>182</td>
<td>188.16</td>
<td>195</td>
<td></td>
</tr>
<tr>
<td>rg150.2</td>
<td>25</td>
<td>1.738</td>
<td>188</td>
<td>190.4</td>
<td>194</td>
<td>1.722</td>
<td>190</td>
<td>194.4</td>
<td>199</td>
<td></td>
</tr>
<tr>
<td>rg150.3</td>
<td>25</td>
<td>2.082</td>
<td>184</td>
<td>188.52</td>
<td>195</td>
<td>2.047</td>
<td>191</td>
<td>194.72</td>
<td>198</td>
<td></td>
</tr>
<tr>
<td>rg150.4</td>
<td>25</td>
<td>2.146</td>
<td>194</td>
<td>197.48</td>
<td>201</td>
<td>2.120</td>
<td>198</td>
<td>203.16</td>
<td>207</td>
<td></td>
</tr>
<tr>
<td>rg150.5</td>
<td>25</td>
<td>2.211</td>
<td>190</td>
<td>195.6</td>
<td>200</td>
<td>2.208</td>
<td>194</td>
<td>198.88</td>
<td>205</td>
<td></td>
</tr>
<tr>
<td>rg200.1</td>
<td>25</td>
<td>3.094</td>
<td>235</td>
<td>240.4</td>
<td>245</td>
<td>3.062</td>
<td>240</td>
<td>242.56</td>
<td>246</td>
<td></td>
</tr>
<tr>
<td>rg200.2</td>
<td>25</td>
<td>3.181</td>
<td>228</td>
<td>233.32</td>
<td>238</td>
<td>3.139</td>
<td>233</td>
<td>237.12</td>
<td>242</td>
<td></td>
</tr>
<tr>
<td>rg200.3</td>
<td>25</td>
<td>3.955</td>
<td>250</td>
<td>255.84</td>
<td>261</td>
<td>3.937</td>
<td>256</td>
<td>260.56</td>
<td>265</td>
<td></td>
</tr>
<tr>
<td>rg200.4</td>
<td>25</td>
<td>4.255</td>
<td>254</td>
<td>259.68</td>
<td>270</td>
<td>4.229</td>
<td>264</td>
<td>267.28</td>
<td>271</td>
<td></td>
</tr>
<tr>
<td>rg200.5</td>
<td>25</td>
<td>4.435</td>
<td>252</td>
<td>260.48</td>
<td>266</td>
<td>4.391</td>
<td>264</td>
<td>268.8</td>
<td>274</td>
<td></td>
</tr>
<tr>
<td>rg300.1</td>
<td>25</td>
<td>11.92</td>
<td>340</td>
<td>349.64</td>
<td>357</td>
<td>11.85</td>
<td>347</td>
<td>352.68</td>
<td>360</td>
<td></td>
</tr>
<tr>
<td>rg300.2</td>
<td>25</td>
<td>17.63</td>
<td>378</td>
<td>385.52</td>
<td>392</td>
<td>17.61</td>
<td>392</td>
<td>399.4</td>
<td>405</td>
<td></td>
</tr>
<tr>
<td>rg300.3</td>
<td>25</td>
<td>17.93</td>
<td>385</td>
<td>389.04</td>
<td>394</td>
<td>17.86</td>
<td>394</td>
<td>398.96</td>
<td>405</td>
<td></td>
</tr>
<tr>
<td>rg300.4</td>
<td>25</td>
<td>23.40</td>
<td>413</td>
<td>419.28</td>
<td>428</td>
<td>23.69</td>
<td>429</td>
<td>437.92</td>
<td>444</td>
<td></td>
</tr>
<tr>
<td>rg300.5</td>
<td>25</td>
<td>23.96</td>
<td>421</td>
<td>428.76</td>
<td>436</td>
<td>24.36</td>
<td>436</td>
<td>441.6</td>
<td>446</td>
<td></td>
</tr>
<tr>
<td>o100.1</td>
<td>25</td>
<td>0.390</td>
<td>173</td>
<td>181.92</td>
<td>190</td>
<td>0.200</td>
<td>192</td>
<td>194.12</td>
<td>197</td>
<td></td>
</tr>
<tr>
<td>o100.2</td>
<td>25</td>
<td>0.708</td>
<td>161</td>
<td>170.08</td>
<td>177</td>
<td>0.600</td>
<td>187</td>
<td>190.8</td>
<td>194</td>
<td></td>
</tr>
<tr>
<td>o100.3</td>
<td>25</td>
<td>1.020</td>
<td>163</td>
<td>168.52</td>
<td>177</td>
<td>0.966</td>
<td>183</td>
<td>190.24</td>
<td>196</td>
<td></td>
</tr>
<tr>
<td>o100.4</td>
<td>25</td>
<td>1.348</td>
<td>164</td>
<td>167.44</td>
<td>172</td>
<td>1.300</td>
<td>186</td>
<td>188.96</td>
<td>195</td>
<td></td>
</tr>
<tr>
<td>o100.5</td>
<td>25</td>
<td>1.678</td>
<td>166</td>
<td>169.48</td>
<td>175</td>
<td>1.731</td>
<td>186</td>
<td>191.4</td>
<td>194</td>
<td></td>
</tr>
<tr>
<td>o200.1</td>
<td>25</td>
<td>1.721</td>
<td>350</td>
<td>378.96</td>
<td>394</td>
<td>0.256</td>
<td>396</td>
<td>396.48</td>
<td>397</td>
<td></td>
</tr>
<tr>
<td>o200.2</td>
<td>25</td>
<td>2.631</td>
<td>350</td>
<td>360.6</td>
<td>366</td>
<td>1.614</td>
<td>389</td>
<td>393.48</td>
<td>397</td>
<td></td>
</tr>
<tr>
<td>o200.3</td>
<td>25</td>
<td>3.577</td>
<td>336</td>
<td>350.96</td>
<td>370</td>
<td>2.892</td>
<td>386</td>
<td>389.84</td>
<td>395</td>
<td></td>
</tr>
<tr>
<td>o200.4</td>
<td>25</td>
<td>4.541</td>
<td>329</td>
<td>340.08</td>
<td>351</td>
<td>4.007</td>
<td>380</td>
<td>387.92</td>
<td>391</td>
<td></td>
</tr>
<tr>
<td>o200.5</td>
<td>25</td>
<td>5.381</td>
<td>325</td>
<td>342</td>
<td>353</td>
<td>5.144</td>
<td>377</td>
<td>382.52</td>
<td>389</td>
<td></td>
</tr>
</tbody>
</table>
### B.3 $SA_E$ and $SA_{CA1}$ statistics

Table B.4: $SA_E$ and $SA_{CA1}$ solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>$SA_E$ ave t.</th>
<th>$SA_E$ worst ave</th>
<th>$SA_E$ best ave</th>
<th>$SA_{CA1}$ ave t.</th>
<th>$SA_{CA1}$ worst ave</th>
<th>$SA_{CA1}$ best ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>cimi-g1</td>
<td>20</td>
<td>2</td>
<td>15</td>
<td>15</td>
<td>2</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>cimi-g2</td>
<td>20</td>
<td>84</td>
<td>114</td>
<td>115.8</td>
<td>12</td>
<td>116</td>
<td>117</td>
</tr>
<tr>
<td>cimi-g3</td>
<td>20</td>
<td>19</td>
<td>51</td>
<td>51</td>
<td>19</td>
<td>51</td>
<td>51</td>
</tr>
<tr>
<td>cimi-g4</td>
<td>20</td>
<td>2</td>
<td>16</td>
<td>16</td>
<td>2</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>cimi-g5</td>
<td>20</td>
<td>37</td>
<td>79</td>
<td>79</td>
<td>40</td>
<td>79</td>
<td>79</td>
</tr>
<tr>
<td>cimi-g6</td>
<td>20</td>
<td>21</td>
<td>52</td>
<td>52</td>
<td>22</td>
<td>52</td>
<td>52</td>
</tr>
<tr>
<td>g10</td>
<td>20</td>
<td>1</td>
<td>47</td>
<td>47</td>
<td>0</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>g11</td>
<td>20</td>
<td>1</td>
<td>47</td>
<td>47</td>
<td>0</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>g12</td>
<td>20</td>
<td>5</td>
<td>46</td>
<td>46.85</td>
<td>0</td>
<td>47</td>
<td>47</td>
</tr>
<tr>
<td>g13</td>
<td>10</td>
<td>180</td>
<td>94</td>
<td>95.7</td>
<td>32</td>
<td>96</td>
<td>97</td>
</tr>
<tr>
<td>g14</td>
<td>10</td>
<td>117</td>
<td>96</td>
<td>96.9</td>
<td>0</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td>g15</td>
<td>10</td>
<td>153</td>
<td>97</td>
<td>97</td>
<td>0</td>
<td>97</td>
<td>97</td>
</tr>
<tr>
<td>g16</td>
<td>10</td>
<td>528</td>
<td>158</td>
<td>161.5</td>
<td>553</td>
<td>162</td>
<td>166.1</td>
</tr>
<tr>
<td>g17</td>
<td>10</td>
<td>898</td>
<td>179</td>
<td>179.8</td>
<td>956</td>
<td>187</td>
<td>190.5</td>
</tr>
<tr>
<td>g18</td>
<td>10</td>
<td>1160</td>
<td>183</td>
<td>185.8</td>
<td>1180</td>
<td>193</td>
<td>194.5</td>
</tr>
<tr>
<td>g19</td>
<td>10</td>
<td>2682</td>
<td>252</td>
<td>254.4</td>
<td>2903</td>
<td>267</td>
<td>269.5</td>
</tr>
</tbody>
</table>

132
### Table B.5: More $SA_E$ and $SA_{CA1}$ solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>$SA_E$</th>
<th></th>
<th></th>
<th></th>
<th>$SA_{CA1}$</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ave t.</td>
<td>worst</td>
<td>ave</td>
<td>best</td>
<td>ave t.</td>
<td>worst</td>
<td>ave</td>
</tr>
<tr>
<td>rg100.1</td>
<td>20</td>
<td>268</td>
<td>139</td>
<td>140.9</td>
<td>144</td>
<td>271</td>
<td>138</td>
<td>141.25</td>
</tr>
<tr>
<td>rg100.2</td>
<td>20</td>
<td>276</td>
<td>138</td>
<td>143</td>
<td>145</td>
<td>279</td>
<td>141</td>
<td>144.1</td>
</tr>
<tr>
<td>rg100.3</td>
<td>20</td>
<td>307</td>
<td>140</td>
<td>142.1</td>
<td>145</td>
<td>305</td>
<td>143</td>
<td>145.3</td>
</tr>
<tr>
<td>rg100.4</td>
<td>20</td>
<td>353</td>
<td>146</td>
<td>148.9</td>
<td>151</td>
<td>353</td>
<td>148</td>
<td>151.05</td>
</tr>
<tr>
<td>rg100.5</td>
<td>20</td>
<td>413</td>
<td>156</td>
<td>158.65</td>
<td>163</td>
<td>428</td>
<td>159</td>
<td>161.15</td>
</tr>
<tr>
<td>rg150.1</td>
<td>10</td>
<td>828</td>
<td>202</td>
<td>204.6</td>
<td>207</td>
<td>824</td>
<td>202</td>
<td>205.4</td>
</tr>
<tr>
<td>rg150.2</td>
<td>10</td>
<td>880</td>
<td>201</td>
<td>205.2</td>
<td>210</td>
<td>855</td>
<td>202</td>
<td>205.6</td>
</tr>
<tr>
<td>rg150.3</td>
<td>10</td>
<td>1003</td>
<td>205</td>
<td>207.2</td>
<td>210</td>
<td>985</td>
<td>206</td>
<td>208.6</td>
</tr>
<tr>
<td>rg150.4</td>
<td>10</td>
<td>1102</td>
<td>211</td>
<td>213.5</td>
<td>217</td>
<td>1088</td>
<td>211</td>
<td>214.9</td>
</tr>
<tr>
<td>rg150.5</td>
<td>10</td>
<td>1190</td>
<td>212</td>
<td>214.2</td>
<td>218</td>
<td>1113</td>
<td>211</td>
<td>214</td>
</tr>
<tr>
<td>rg200.1</td>
<td>10</td>
<td>1503</td>
<td>258</td>
<td>262.4</td>
<td>264</td>
<td>1451</td>
<td>260</td>
<td>262.3</td>
</tr>
<tr>
<td>rg200.2</td>
<td>10</td>
<td>1496</td>
<td>254</td>
<td>258.3</td>
<td>261</td>
<td>1455</td>
<td>256</td>
<td>259.6</td>
</tr>
<tr>
<td>rg200.3</td>
<td>10</td>
<td>1942</td>
<td>271</td>
<td>274.6</td>
<td>280</td>
<td>1990</td>
<td>275</td>
<td>280.4</td>
</tr>
<tr>
<td>rg200.4</td>
<td>10</td>
<td>2081</td>
<td>273</td>
<td>277.3</td>
<td>281</td>
<td>2161</td>
<td>277</td>
<td>282.7</td>
</tr>
<tr>
<td>rg200.5</td>
<td>10</td>
<td>2183</td>
<td>277</td>
<td>280.9</td>
<td>289</td>
<td>2125</td>
<td>279</td>
<td>284.6</td>
</tr>
<tr>
<td>rg300.1</td>
<td>5</td>
<td>5860</td>
<td>388</td>
<td>389.2</td>
<td>390</td>
<td>5960</td>
<td>385</td>
<td>389.6</td>
</tr>
<tr>
<td>rg300.2</td>
<td>5</td>
<td>8707</td>
<td>405</td>
<td>411</td>
<td>415</td>
<td>9197</td>
<td>413</td>
<td>416.4</td>
</tr>
<tr>
<td>rg300.3</td>
<td>5</td>
<td>8787</td>
<td>410</td>
<td>411.6</td>
<td>414</td>
<td>9314</td>
<td>409</td>
<td>413.6</td>
</tr>
<tr>
<td>rg300.4</td>
<td>5</td>
<td>11296</td>
<td>424</td>
<td>427.6</td>
<td>432</td>
<td>11840</td>
<td>433</td>
<td>435.4</td>
</tr>
<tr>
<td>rg300.5</td>
<td>5</td>
<td>11729</td>
<td>422</td>
<td>429.6</td>
<td>441</td>
<td>11981</td>
<td>435</td>
<td>439</td>
</tr>
<tr>
<td>o100.1</td>
<td>10</td>
<td>336</td>
<td>189</td>
<td>193.6</td>
<td>197</td>
<td>344</td>
<td>193</td>
<td>194.8</td>
</tr>
<tr>
<td>o100.2</td>
<td>10</td>
<td>501</td>
<td>175</td>
<td>182.3</td>
<td>190</td>
<td>483</td>
<td>184</td>
<td>191.9</td>
</tr>
<tr>
<td>o100.3</td>
<td>10</td>
<td>600</td>
<td>177</td>
<td>182.5</td>
<td>187</td>
<td>600</td>
<td>188</td>
<td>191.3</td>
</tr>
<tr>
<td>o100.4</td>
<td>10</td>
<td>698</td>
<td>178</td>
<td>181.8</td>
<td>187</td>
<td>720</td>
<td>187</td>
<td>189.5</td>
</tr>
<tr>
<td>o100.5</td>
<td>10</td>
<td>829</td>
<td>179</td>
<td>184</td>
<td>189</td>
<td>812</td>
<td>188</td>
<td>190.7</td>
</tr>
<tr>
<td>o200.1</td>
<td>10</td>
<td>529</td>
<td>397</td>
<td>397</td>
<td>397</td>
<td>137</td>
<td>397</td>
<td>397</td>
</tr>
<tr>
<td>o200.2</td>
<td>10</td>
<td>2189</td>
<td>385</td>
<td>394.2</td>
<td>397</td>
<td>2027</td>
<td>393</td>
<td>396</td>
</tr>
<tr>
<td>o200.3</td>
<td>10</td>
<td>2561</td>
<td>373</td>
<td>382.9</td>
<td>389</td>
<td>2548</td>
<td>390</td>
<td>394.2</td>
</tr>
<tr>
<td>o200.4</td>
<td>10</td>
<td>2880</td>
<td>359</td>
<td>370.4</td>
<td>379</td>
<td>3133</td>
<td>385</td>
<td>388.6</td>
</tr>
<tr>
<td>o200.5</td>
<td>10</td>
<td>3130</td>
<td>355</td>
<td>366.2</td>
<td>374</td>
<td>3471</td>
<td>381</td>
<td>383.6</td>
</tr>
</tbody>
</table>
Appendix C

Statistics for thickness

C.1 ST and CA statistics

Table C.1: ST and CA solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>ST ave</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
<th>CA ave</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{10}$</td>
<td>25</td>
<td>0.003</td>
<td>6</td>
<td>5.88</td>
<td>5</td>
<td>0.005</td>
<td>5</td>
<td>4.96</td>
<td>4</td>
</tr>
<tr>
<td>$K_{15}$</td>
<td>25</td>
<td>0.006</td>
<td>9</td>
<td>8.28</td>
<td>8</td>
<td>0.008</td>
<td>8</td>
<td>7.04</td>
<td>7</td>
</tr>
<tr>
<td>$K_{20}$</td>
<td>25</td>
<td>0.009</td>
<td>12</td>
<td>11.04</td>
<td>11</td>
<td>0.012</td>
<td>10</td>
<td>9.12</td>
<td>9</td>
</tr>
<tr>
<td>$K_{30}$</td>
<td>25</td>
<td>0.016</td>
<td>17</td>
<td>16.32</td>
<td>16</td>
<td>0.028</td>
<td>15</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>$K_{40}$</td>
<td>25</td>
<td>0.035</td>
<td>23</td>
<td>22</td>
<td>21</td>
<td>0.062</td>
<td>19</td>
<td>18.52</td>
<td>18</td>
</tr>
<tr>
<td>$K_{50}$</td>
<td>25</td>
<td>0.060</td>
<td>29</td>
<td>27.6</td>
<td>27</td>
<td>0.118</td>
<td>25</td>
<td>23.24</td>
<td>22</td>
</tr>
<tr>
<td>$K_{60}$</td>
<td>25</td>
<td>0.095</td>
<td>34</td>
<td>32.6</td>
<td>32</td>
<td>0.204</td>
<td>29</td>
<td>27.88</td>
<td>27</td>
</tr>
<tr>
<td>$K_{70}$</td>
<td>25</td>
<td>0.146</td>
<td>39</td>
<td>38.12</td>
<td>37</td>
<td>0.356</td>
<td>33</td>
<td>32.48</td>
<td>32</td>
</tr>
<tr>
<td>$K_{80}$</td>
<td>25</td>
<td>0.208</td>
<td>45</td>
<td>43.92</td>
<td>43</td>
<td>0.560</td>
<td>39</td>
<td>37.4</td>
<td>36</td>
</tr>
<tr>
<td>$K_{90}$</td>
<td>25</td>
<td>0.278</td>
<td>50</td>
<td>49.4</td>
<td>49</td>
<td>0.854</td>
<td>45</td>
<td>42.28</td>
<td>41</td>
</tr>
<tr>
<td>$K_{100}$</td>
<td>25</td>
<td>0.386</td>
<td>56</td>
<td>54.72</td>
<td>54</td>
<td>1.259</td>
<td>49</td>
<td>47</td>
<td>45</td>
</tr>
<tr>
<td>$r_{10,25}$</td>
<td>25</td>
<td>0.005</td>
<td>4</td>
<td>3.92</td>
<td>3</td>
<td>0.006</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$r_{20,92}$</td>
<td>25</td>
<td>0.013</td>
<td>7</td>
<td>6.12</td>
<td>6</td>
<td>0.011</td>
<td>5</td>
<td>4.84</td>
<td>4</td>
</tr>
<tr>
<td>$r_{30,230}$</td>
<td>25</td>
<td>0.018</td>
<td>10</td>
<td>9.12</td>
<td>9</td>
<td>0.022</td>
<td>8</td>
<td>7.04</td>
<td>7</td>
</tr>
<tr>
<td>$r_{40,311}$</td>
<td>25</td>
<td>0.029</td>
<td>10</td>
<td>9.12</td>
<td>9</td>
<td>0.032</td>
<td>8</td>
<td>7.2</td>
<td>7</td>
</tr>
<tr>
<td>$r_{50,495}$</td>
<td>25</td>
<td>0.040</td>
<td>12</td>
<td>11.64</td>
<td>11</td>
<td>0.056</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>$r_{60,556}$</td>
<td>25</td>
<td>0.047</td>
<td>11</td>
<td>10.96</td>
<td>10</td>
<td>0.065</td>
<td>9</td>
<td>8.72</td>
<td>8</td>
</tr>
<tr>
<td>$r_{70,766}$</td>
<td>25</td>
<td>0.068</td>
<td>13</td>
<td>13</td>
<td>13</td>
<td>0.091</td>
<td>9</td>
<td>8.72</td>
<td>8</td>
</tr>
<tr>
<td>$r_{80,939}$</td>
<td>25</td>
<td>0.086</td>
<td>14</td>
<td>13.72</td>
<td>13</td>
<td>0.117</td>
<td>11</td>
<td>10.68</td>
<td>10</td>
</tr>
<tr>
<td>$r_{90,1263}$</td>
<td>25</td>
<td>0.121</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>0.169</td>
<td>13</td>
<td>12.72</td>
<td>12</td>
</tr>
<tr>
<td>$r_{100,1508}$</td>
<td>25</td>
<td>0.148</td>
<td>18</td>
<td>17.04</td>
<td>17</td>
<td>0.203</td>
<td>14</td>
<td>13.2</td>
<td>13</td>
</tr>
</tbody>
</table>
## Table C.2: More ST and CA solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{150}$</td>
<td>10</td>
<td>1.380</td>
<td>83</td>
<td>82.3</td>
<td>81</td>
<td>5.122</td>
<td>72</td>
<td>70.3</td>
<td>69</td>
</tr>
<tr>
<td>$K_{200}$</td>
<td>10</td>
<td>3.382</td>
<td>111</td>
<td>109.4</td>
<td>108</td>
<td>15.41</td>
<td>96</td>
<td>93.8</td>
<td>91</td>
</tr>
<tr>
<td>$K_{300}$</td>
<td>10</td>
<td>11.74</td>
<td>167</td>
<td>165.3</td>
<td>164</td>
<td>73.4</td>
<td>145</td>
<td>141.1</td>
<td>137</td>
</tr>
<tr>
<td>$K_{400}$</td>
<td>10</td>
<td>28.83</td>
<td>221</td>
<td>219.2</td>
<td>217</td>
<td>231.7</td>
<td>194</td>
<td>188.4</td>
<td>183</td>
</tr>
<tr>
<td>$K_{500}$</td>
<td>10</td>
<td>58.77</td>
<td>278</td>
<td>275.5</td>
<td>271</td>
<td>565.1</td>
<td>241</td>
<td>235.6</td>
<td>230</td>
</tr>
<tr>
<td>$K_{600}$</td>
<td>5</td>
<td>100.3</td>
<td>335</td>
<td>334.2</td>
<td>334</td>
<td>1108</td>
<td>288</td>
<td>282.6</td>
<td>277</td>
</tr>
<tr>
<td>$K_{700}$</td>
<td>5</td>
<td>161.6</td>
<td>389</td>
<td>386</td>
<td>379</td>
<td>2371</td>
<td>349</td>
<td>332.2</td>
<td>320</td>
</tr>
<tr>
<td>$K_{800}$</td>
<td>5</td>
<td>245.6</td>
<td>441</td>
<td>438.8</td>
<td>437</td>
<td>4561</td>
<td>403</td>
<td>381.6</td>
<td>366</td>
</tr>
<tr>
<td>$K_{900}$</td>
<td>5</td>
<td>357.2</td>
<td>498</td>
<td>493.8</td>
<td>490</td>
<td>6607</td>
<td>438</td>
<td>422.2</td>
<td>412</td>
</tr>
<tr>
<td>$K_{1000}$</td>
<td>5</td>
<td>502.1</td>
<td>556</td>
<td>554</td>
<td>551</td>
<td>10680</td>
<td>489</td>
<td>470.4</td>
<td>456</td>
</tr>
<tr>
<td>r0</td>
<td>10</td>
<td>0.978</td>
<td>18</td>
<td>17.3</td>
<td>17</td>
<td>5.215</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>r1</td>
<td>10</td>
<td>6.234</td>
<td>53</td>
<td>53</td>
<td>53</td>
<td>21.03</td>
<td>43</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>r2</td>
<td>10</td>
<td>22.20</td>
<td>104</td>
<td>103.2</td>
<td>103</td>
<td>61.63</td>
<td>90</td>
<td>89.9</td>
<td>89</td>
</tr>
<tr>
<td>r3</td>
<td>10</td>
<td>49.34</td>
<td>161</td>
<td>160.2</td>
<td>160</td>
<td>156.8</td>
<td>136</td>
<td>133.2</td>
<td>130</td>
</tr>
<tr>
<td>r4</td>
<td>10</td>
<td>87.03</td>
<td>214</td>
<td>213.6</td>
<td>213</td>
<td>313.7</td>
<td>182</td>
<td>179.9</td>
<td>177</td>
</tr>
<tr>
<td>r5</td>
<td>10</td>
<td>135.3</td>
<td>264</td>
<td>264</td>
<td>264</td>
<td>576.6</td>
<td>229</td>
<td>226.7</td>
<td>224</td>
</tr>
<tr>
<td>r6</td>
<td>5</td>
<td>191.5</td>
<td>313</td>
<td>312.8</td>
<td>312</td>
<td>891.2</td>
<td>277</td>
<td>274.2</td>
<td>270</td>
</tr>
<tr>
<td>r7</td>
<td>5</td>
<td>260.2</td>
<td>363</td>
<td>363</td>
<td>363</td>
<td>1373</td>
<td>325</td>
<td>321.8</td>
<td>316</td>
</tr>
<tr>
<td>r8</td>
<td>5</td>
<td>341.4</td>
<td>414</td>
<td>413.4</td>
<td>413</td>
<td>2129</td>
<td>374</td>
<td>369</td>
<td>361</td>
</tr>
<tr>
<td>r9</td>
<td>5</td>
<td>434.6</td>
<td>465</td>
<td>465</td>
<td>465</td>
<td>3238</td>
<td>424</td>
<td>418.6</td>
<td>411</td>
</tr>
<tr>
<td>rr1</td>
<td>10</td>
<td>0.401</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>0.702</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>rr2</td>
<td>10</td>
<td>1.892</td>
<td>26</td>
<td>26</td>
<td>26</td>
<td>6.819</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>rr3</td>
<td>10</td>
<td>6.084</td>
<td>51</td>
<td>51</td>
<td>51</td>
<td>15.63</td>
<td>43</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>rr4</td>
<td>10</td>
<td>12.96</td>
<td>76</td>
<td>76</td>
<td>76</td>
<td>30.52</td>
<td>66</td>
<td>65.4</td>
<td>65</td>
</tr>
<tr>
<td>rr5</td>
<td>10</td>
<td>22.32</td>
<td>101</td>
<td>101</td>
<td>101</td>
<td>52.55</td>
<td>89</td>
<td>88.6</td>
<td>88</td>
</tr>
<tr>
<td>rr6</td>
<td>10</td>
<td>34.25</td>
<td>126</td>
<td>126</td>
<td>126</td>
<td>81.74</td>
<td>113</td>
<td>112.4</td>
<td>112</td>
</tr>
<tr>
<td>rr7</td>
<td>10</td>
<td>48.68</td>
<td>151</td>
<td>151</td>
<td>151</td>
<td>118.3</td>
<td>137</td>
<td>136.3</td>
<td>136</td>
</tr>
<tr>
<td>rr8</td>
<td>10</td>
<td>65.71</td>
<td>177</td>
<td>176.2</td>
<td>176</td>
<td>165.6</td>
<td>161</td>
<td>160.3</td>
<td>160</td>
</tr>
<tr>
<td>rr9</td>
<td>10</td>
<td>85.11</td>
<td>202</td>
<td>201.1</td>
<td>201</td>
<td>219.0</td>
<td>185</td>
<td>184.7</td>
<td>184</td>
</tr>
<tr>
<td>rr10</td>
<td>10</td>
<td>107.2</td>
<td>227</td>
<td>226.2</td>
<td>226</td>
<td>277.8</td>
<td>210</td>
<td>209.4</td>
<td>209</td>
</tr>
<tr>
<td>rr11</td>
<td>10</td>
<td>131.8</td>
<td>252</td>
<td>251.2</td>
<td>251</td>
<td>352.4</td>
<td>234</td>
<td>233.5</td>
<td>233</td>
</tr>
</tbody>
</table>
C.2 CA1 and CA2 statistics

Table C.3: CA1 and CA2 solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{10}$</td>
<td>25</td>
<td>0.005</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0.006</td>
<td>4</td>
<td>3.96</td>
<td>3</td>
</tr>
<tr>
<td>$K_{15}$</td>
<td>25</td>
<td>0.010</td>
<td>6</td>
<td>5.12</td>
<td>5</td>
<td>0.008</td>
<td>6</td>
<td>5.56</td>
<td>5</td>
</tr>
<tr>
<td>$K_{20}$</td>
<td>25</td>
<td>0.012</td>
<td>7</td>
<td>6.84</td>
<td>6</td>
<td>0.014</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>$K_{30}$</td>
<td>25</td>
<td>0.024</td>
<td>10</td>
<td>9.44</td>
<td>9</td>
<td>0.034</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>$K_{40}$</td>
<td>25</td>
<td>0.048</td>
<td>13</td>
<td>12.28</td>
<td>12</td>
<td>0.060</td>
<td>14</td>
<td>12.96</td>
<td>12</td>
</tr>
<tr>
<td>$K_{50}$</td>
<td>25</td>
<td>0.077</td>
<td>16</td>
<td>15.2</td>
<td>15</td>
<td>0.102</td>
<td>17</td>
<td>16</td>
<td>15</td>
</tr>
<tr>
<td>$K_{60}$</td>
<td>25</td>
<td>0.122</td>
<td>19</td>
<td>18.16</td>
<td>18</td>
<td>0.159</td>
<td>20</td>
<td>19</td>
<td>18</td>
</tr>
<tr>
<td>$K_{70}$</td>
<td>25</td>
<td>0.178</td>
<td>23</td>
<td>21.48</td>
<td>21</td>
<td>0.233</td>
<td>23</td>
<td>22.12</td>
<td>21</td>
</tr>
<tr>
<td>$K_{80}$</td>
<td>25</td>
<td>0.259</td>
<td>26</td>
<td>24.68</td>
<td>24</td>
<td>0.332</td>
<td>27</td>
<td>24.8</td>
<td>24</td>
</tr>
<tr>
<td>$K_{90}$</td>
<td>25</td>
<td>0.355</td>
<td>30</td>
<td>28.2</td>
<td>27</td>
<td>0.462</td>
<td>30</td>
<td>27.76</td>
<td>27</td>
</tr>
<tr>
<td>$K_{100}$</td>
<td>25</td>
<td>0.515</td>
<td>33</td>
<td>31.56</td>
<td>31</td>
<td>0.627</td>
<td>33</td>
<td>30.76</td>
<td>30</td>
</tr>
<tr>
<td>$r_{10,25}$</td>
<td>25</td>
<td>0.007</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0.007</td>
<td>3</td>
<td>2.72</td>
<td>2</td>
</tr>
<tr>
<td>$r_{20,92}$</td>
<td>25</td>
<td>0.014</td>
<td>5</td>
<td>4.04</td>
<td>4</td>
<td>0.014</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$r_{30,230}$</td>
<td>25</td>
<td>0.027</td>
<td>6</td>
<td>5.88</td>
<td>5</td>
<td>0.030</td>
<td>6</td>
<td>5.88</td>
<td>5</td>
</tr>
<tr>
<td>$r_{40,311}$</td>
<td>25</td>
<td>0.036</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>0.040</td>
<td>6</td>
<td>5.96</td>
<td>5</td>
</tr>
<tr>
<td>$r_{50,495}$</td>
<td>25</td>
<td>0.058</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>0.064</td>
<td>8</td>
<td>7.04</td>
<td>7</td>
</tr>
<tr>
<td>$r_{60,556}$</td>
<td>25</td>
<td>0.068</td>
<td>8</td>
<td>7.04</td>
<td>7</td>
<td>0.074</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>$r_{70,766}$</td>
<td>25</td>
<td>0.094</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>0.104</td>
<td>8</td>
<td>7.96</td>
<td>7</td>
</tr>
<tr>
<td>$r_{80,939}$</td>
<td>25</td>
<td>0.118</td>
<td>9</td>
<td>8.28</td>
<td>8</td>
<td>0.125</td>
<td>9</td>
<td>8.2</td>
<td>8</td>
</tr>
<tr>
<td>$r_{90,1263}$</td>
<td>25</td>
<td>0.162</td>
<td>10</td>
<td>9.64</td>
<td>9</td>
<td>0.177</td>
<td>10</td>
<td>9.72</td>
<td>9</td>
</tr>
<tr>
<td>$r_{100,1508}$</td>
<td>25</td>
<td>0.202</td>
<td>11</td>
<td>10.16</td>
<td>10</td>
<td>0.224</td>
<td>11</td>
<td>10.16</td>
<td>10</td>
</tr>
</tbody>
</table>
# APPENDIX C. STATISTICS FOR THICKNESS

<table>
<thead>
<tr>
<th>Graph</th>
<th>Rep.</th>
<th>CA1</th>
<th>CA2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Ave t.</td>
<td>Worst</td>
</tr>
<tr>
<td>$K_{150}$</td>
<td>10</td>
<td>2.236</td>
<td>50</td>
</tr>
<tr>
<td>$K_{200}$</td>
<td>10</td>
<td>7.006</td>
<td>67</td>
</tr>
<tr>
<td>$K_{300}$</td>
<td>10</td>
<td>37.78</td>
<td>102</td>
</tr>
<tr>
<td>$K_{400}$</td>
<td>10</td>
<td>149.2</td>
<td>134</td>
</tr>
<tr>
<td>$K_{500}$</td>
<td>10</td>
<td>395.0</td>
<td>166</td>
</tr>
<tr>
<td>$K_{600}$</td>
<td>5</td>
<td>728.2</td>
<td>199</td>
</tr>
<tr>
<td>$K_{700}$</td>
<td>5</td>
<td>1344</td>
<td>233</td>
</tr>
<tr>
<td>$K_{800}$</td>
<td>5</td>
<td>2773</td>
<td>267</td>
</tr>
<tr>
<td>$K_{900}$</td>
<td>5</td>
<td>4875</td>
<td>300</td>
</tr>
<tr>
<td>$K_{1000}$</td>
<td>5</td>
<td>8551</td>
<td>337</td>
</tr>
<tr>
<td>r0</td>
<td>10</td>
<td>3.011</td>
<td>14</td>
</tr>
<tr>
<td>r1</td>
<td>10</td>
<td>11.53</td>
<td>32</td>
</tr>
<tr>
<td>r2</td>
<td>10</td>
<td>29.90</td>
<td>57</td>
</tr>
<tr>
<td>r3</td>
<td>10</td>
<td>58.18</td>
<td>87</td>
</tr>
<tr>
<td>r4</td>
<td>10</td>
<td>96.61</td>
<td>115</td>
</tr>
<tr>
<td>r5</td>
<td>10</td>
<td>144.8</td>
<td>143</td>
</tr>
<tr>
<td>r6</td>
<td>5</td>
<td>210.4</td>
<td>172</td>
</tr>
<tr>
<td>r7</td>
<td>5</td>
<td>283.5</td>
<td>200</td>
</tr>
<tr>
<td>r8</td>
<td>5</td>
<td>368.5</td>
<td>228</td>
</tr>
<tr>
<td>r9</td>
<td>5</td>
<td>472.4</td>
<td>256</td>
</tr>
<tr>
<td>rr1</td>
<td>10</td>
<td>0.566</td>
<td>6</td>
</tr>
<tr>
<td>rr2</td>
<td>10</td>
<td>5.318</td>
<td>19</td>
</tr>
<tr>
<td>rr3</td>
<td>10</td>
<td>11.38</td>
<td>32</td>
</tr>
<tr>
<td>rr4</td>
<td>10</td>
<td>19.88</td>
<td>44</td>
</tr>
<tr>
<td>rr5</td>
<td>10</td>
<td>30.10</td>
<td>57</td>
</tr>
<tr>
<td>rr6</td>
<td>10</td>
<td>42.55</td>
<td>70</td>
</tr>
<tr>
<td>rr7</td>
<td>10</td>
<td>57.60</td>
<td>82</td>
</tr>
<tr>
<td>rr8</td>
<td>10</td>
<td>74.68</td>
<td>95</td>
</tr>
<tr>
<td>rr9</td>
<td>10</td>
<td>93.41</td>
<td>107</td>
</tr>
<tr>
<td>rr10</td>
<td>10</td>
<td>115.4</td>
<td>120</td>
</tr>
<tr>
<td>rr11</td>
<td>10</td>
<td>138.8</td>
<td>132</td>
</tr>
</tbody>
</table>
## C.3 GCA statistics

Table C.5: GCA solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{10}$</td>
<td>25</td>
<td>0.013</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$K_{15}$</td>
<td>25</td>
<td>0.051</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$K_{20}$</td>
<td>25</td>
<td>0.174</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$K_{30}$</td>
<td>25</td>
<td>0.846</td>
<td>8</td>
<td>7.2</td>
<td>7</td>
</tr>
<tr>
<td>$K_{40}$</td>
<td>25</td>
<td>3.117</td>
<td>11</td>
<td>9.96</td>
<td>9</td>
</tr>
<tr>
<td>$K_{50}$</td>
<td>25</td>
<td>7.637</td>
<td>13</td>
<td>12.04</td>
<td>12</td>
</tr>
<tr>
<td>$K_{60}$</td>
<td>25</td>
<td>15.89</td>
<td>16</td>
<td>14.36</td>
<td>14</td>
</tr>
<tr>
<td>$K_{70}$</td>
<td>10</td>
<td>37.22</td>
<td>19</td>
<td>17.1</td>
<td>16</td>
</tr>
<tr>
<td>$K_{80}$</td>
<td>10</td>
<td>64.02</td>
<td>21</td>
<td>19.2</td>
<td>19</td>
</tr>
<tr>
<td>$K_{90}$</td>
<td>10</td>
<td>103.4</td>
<td>23</td>
<td>21.4</td>
<td>21</td>
</tr>
<tr>
<td>$K_{100}$</td>
<td>10</td>
<td>161.8</td>
<td>26</td>
<td>24</td>
<td>23</td>
</tr>
<tr>
<td>$r_{10,25}$</td>
<td>25</td>
<td>0.010</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$r_{20,92}$</td>
<td>25</td>
<td>0.056</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$r_{30,230}$</td>
<td>25</td>
<td>0.271</td>
<td>5</td>
<td>4.36</td>
<td>4</td>
</tr>
<tr>
<td>$r_{40,311}$</td>
<td>25</td>
<td>0.546</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$r_{50,495}$</td>
<td>25</td>
<td>1.326</td>
<td>6</td>
<td>5.96</td>
<td>5</td>
</tr>
<tr>
<td>$r_{60,556}$</td>
<td>25</td>
<td>1.669</td>
<td>6</td>
<td>5.84</td>
<td>5</td>
</tr>
<tr>
<td>$r_{70,766}$</td>
<td>25</td>
<td>3.853</td>
<td>7</td>
<td>6.2</td>
<td>6</td>
</tr>
<tr>
<td>$r_{80,939}$</td>
<td>25</td>
<td>5.734</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>$r_{90,1263}$</td>
<td>25</td>
<td>10.29</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>$r_{100,1508}$</td>
<td>25</td>
<td>14.61</td>
<td>9</td>
<td>8.96</td>
<td>8</td>
</tr>
</tbody>
</table>

K150 1 ? 39 39 39
K200 1 ? 52 52 52
K300 1 ? 79 79 79
K400 1 93581 105 105 105
r0 1 5365 12 12 12
r1 1 ? 30 30 30
rr1 1 ? 5 5 5
rr2 1 ? 17 17 17
rr3 1 ? 30 30 30
rr4 1 ? 43 43 43

The exact running time was lost by a script error.
C.4 GCA1 and GCA2 statistics

Table C.6: GCA1 and GCA2 solution statistics.

<table>
<thead>
<tr>
<th></th>
<th>GCA1</th>
<th>GCA2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ave t.</td>
<td>worst</td>
</tr>
<tr>
<td>graph</td>
<td>rep.</td>
<td></td>
</tr>
<tr>
<td>$K_{10}$</td>
<td>25</td>
<td>0.011</td>
</tr>
<tr>
<td>$K_{15}$</td>
<td>25</td>
<td>0.043</td>
</tr>
<tr>
<td>$K_{20}$</td>
<td>25</td>
<td>0.156</td>
</tr>
<tr>
<td>$K_{30}$</td>
<td>25</td>
<td>0.751</td>
</tr>
<tr>
<td>$K_{40}$</td>
<td>25</td>
<td>2.859</td>
</tr>
<tr>
<td>$K_{50}$</td>
<td>25</td>
<td>7.158</td>
</tr>
<tr>
<td>$K_{60}$</td>
<td>25</td>
<td>15.27</td>
</tr>
<tr>
<td>$K_{70}$</td>
<td>10</td>
<td>30.00</td>
</tr>
<tr>
<td>$K_{80}$</td>
<td>10</td>
<td>61.60</td>
</tr>
<tr>
<td>$K_{90}$</td>
<td>10</td>
<td>100.6</td>
</tr>
<tr>
<td>$K_{100}$</td>
<td>10</td>
<td>154.5</td>
</tr>
<tr>
<td>$r_{10,25}$</td>
<td>25</td>
<td>0.009</td>
</tr>
<tr>
<td>$r_{20,92}$</td>
<td>25</td>
<td>0.048</td>
</tr>
<tr>
<td>$r_{30,230}$</td>
<td>25</td>
<td>0.251</td>
</tr>
<tr>
<td>$r_{40,311}$</td>
<td>25</td>
<td>0.510</td>
</tr>
<tr>
<td>$r_{50,495}$</td>
<td>25</td>
<td>1.248</td>
</tr>
<tr>
<td>$r_{60,556}$</td>
<td>25</td>
<td>1.578</td>
</tr>
<tr>
<td>$r_{70,766}$</td>
<td>25</td>
<td>3.696</td>
</tr>
<tr>
<td>$r_{80,939}$</td>
<td>25</td>
<td>5.486</td>
</tr>
<tr>
<td>$r_{90,1263}$</td>
<td>25</td>
<td>9.998</td>
</tr>
<tr>
<td>$r_{100,1508}$</td>
<td>25</td>
<td>14.25</td>
</tr>
<tr>
<td>$K_{150}$</td>
<td>1</td>
<td>1071</td>
</tr>
<tr>
<td>$K_{200}$</td>
<td>1</td>
<td>3514</td>
</tr>
<tr>
<td>$K_{300}$</td>
<td>1</td>
<td>27836</td>
</tr>
<tr>
<td>$K_{400}$</td>
<td>1</td>
<td>90421</td>
</tr>
<tr>
<td>$r_{0}$</td>
<td>1</td>
<td>5345</td>
</tr>
<tr>
<td>$r_{1}$</td>
<td>1</td>
<td>55482</td>
</tr>
<tr>
<td>$rr_{1}$</td>
<td>1</td>
<td>81.60</td>
</tr>
<tr>
<td>$rr_{2}$</td>
<td>1</td>
<td>14222</td>
</tr>
<tr>
<td>$rr_{3}$</td>
<td>1</td>
<td>56529</td>
</tr>
<tr>
<td>$rr_{4}$</td>
<td>1</td>
<td>126283</td>
</tr>
</tbody>
</table>
Appendix D

Statistics for outerthickness

D.1 GRE, GCA, GCA1, and SA statistics

Table D.1: GCA1 and SA solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
<th>ave t.</th>
<th>worst</th>
<th>ave</th>
<th>best</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{10}$</td>
<td>25/20</td>
<td>0.009</td>
<td>4</td>
<td>3.8</td>
<td>3</td>
<td>0.051</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$K_{15}$</td>
<td>25/20</td>
<td>0.020</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>2.988</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$K_{20}$</td>
<td>25/20</td>
<td>0.050</td>
<td>7</td>
<td>6.32</td>
<td>6</td>
<td>0.992</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$K_{30}$</td>
<td>25/20</td>
<td>0.243</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>62.65</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>$K_{40}$</td>
<td>25/10</td>
<td>0.907</td>
<td>13</td>
<td>12.04</td>
<td>12</td>
<td>136.0</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>$K_{50}$</td>
<td>25/5</td>
<td>2.132</td>
<td>16</td>
<td>15.04</td>
<td>15</td>
<td>841.5</td>
<td>14</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>$K_{60}$</td>
<td>25/1</td>
<td>4.205</td>
<td>19</td>
<td>17.68</td>
<td>17</td>
<td>1101</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>$K_{70}$</td>
<td>10/1</td>
<td>11.36</td>
<td>22</td>
<td>20.3</td>
<td>20</td>
<td>11672</td>
<td>18</td>
<td>18</td>
<td>18†</td>
</tr>
<tr>
<td>$K_{80}$</td>
<td>10/1</td>
<td>17.80</td>
<td>25</td>
<td>23.3</td>
<td>23</td>
<td>11595</td>
<td>21</td>
<td>21</td>
<td>21†</td>
</tr>
<tr>
<td>$K_{90}$</td>
<td>10/1</td>
<td>30.52</td>
<td>29</td>
<td>26.3</td>
<td>26</td>
<td>42529</td>
<td>24</td>
<td>24</td>
<td>24†</td>
</tr>
<tr>
<td>$K_{100}$</td>
<td>10/1</td>
<td>47.30</td>
<td>32</td>
<td>29.4</td>
<td>29</td>
<td>38526</td>
<td>26</td>
<td>26</td>
<td>26†</td>
</tr>
<tr>
<td>$r_{10.25}$</td>
<td>25/10</td>
<td>0.010</td>
<td>3</td>
<td>2.2</td>
<td>2</td>
<td>0.032</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$r_{20.92}$</td>
<td>25/10</td>
<td>0.038</td>
<td>4</td>
<td>3.6</td>
<td>3</td>
<td>1.505</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>$r_{30.230}$</td>
<td>25/10</td>
<td>0.120</td>
<td>6</td>
<td>5.12</td>
<td>5</td>
<td>4.149</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$r_{40.311}$</td>
<td>25/10</td>
<td>0.343</td>
<td>6</td>
<td>5.6</td>
<td>5</td>
<td>17.17</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>$r_{50.495}$</td>
<td>25/10</td>
<td>0.710</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>112.0</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$r_{60.556}$</td>
<td>25/10</td>
<td>1.327</td>
<td>7</td>
<td>6.4</td>
<td>6</td>
<td>507.0</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$r_{70.766}$</td>
<td>25/10</td>
<td>2.800</td>
<td>8</td>
<td>7.24</td>
<td>7</td>
<td>1092</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>$r_{80.939}$</td>
<td>25/5</td>
<td>4.076</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>1409</td>
<td>8</td>
<td>7.8</td>
<td>7</td>
</tr>
<tr>
<td>$r_{90.1263}$</td>
<td>25/5</td>
<td>7.343</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>2075</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>$r_{100.1508}$</td>
<td>25/5</td>
<td>9.598</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>2617</td>
<td>10</td>
<td>9.8</td>
<td>9</td>
</tr>
</tbody>
</table>

† Solution obtained using initial solution constructed by GCA1.
### APPENDIX D. STATISTICS FOR OUTERTHICKNESS

#### Table D.2: More GRE and SA solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>GRE ave t.</th>
<th>worst ave</th>
<th>best ave</th>
<th>SA ave t.</th>
<th>worst ave</th>
<th>best ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{10,10}$</td>
<td>25/20</td>
<td>0.064</td>
<td>5</td>
<td>4.04</td>
<td>4</td>
<td>0.285</td>
<td>4</td>
</tr>
<tr>
<td>$K_{15,15}$</td>
<td>25/20</td>
<td>0.283</td>
<td>7</td>
<td>6.04</td>
<td>6</td>
<td>1.252</td>
<td>6</td>
</tr>
<tr>
<td>$K_{20,20}$</td>
<td>25/20</td>
<td>0.990</td>
<td>10</td>
<td>7.84</td>
<td>7</td>
<td>183.0</td>
<td>8</td>
</tr>
<tr>
<td>$K_{30,30}$</td>
<td>25/10</td>
<td>4.691</td>
<td>15</td>
<td>11.84</td>
<td>11</td>
<td>121.1</td>
<td>11</td>
</tr>
<tr>
<td>$K_{40,40}$</td>
<td>25/5</td>
<td>18.16</td>
<td>20</td>
<td>15.68</td>
<td>15</td>
<td>1957</td>
<td>15</td>
</tr>
<tr>
<td>$K_{50,50}$</td>
<td>10/5</td>
<td>43.41</td>
<td>25</td>
<td>19.9</td>
<td>19</td>
<td>3613</td>
<td>18</td>
</tr>
<tr>
<td>$hc_4$</td>
<td>25/10</td>
<td>0.011</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.019</td>
<td>2</td>
</tr>
<tr>
<td>$hc_5$</td>
<td>25/10</td>
<td>0.048</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0.701</td>
<td>2</td>
</tr>
<tr>
<td>$hc_6$</td>
<td>25/10</td>
<td>0.280</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0.711</td>
<td>3</td>
</tr>
<tr>
<td>$hc_7$</td>
<td>25/10</td>
<td>1.709</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>14.74</td>
<td>3</td>
</tr>
<tr>
<td>$hc_8$</td>
<td>25/10</td>
<td>12.40</td>
<td>4</td>
<td>3.88</td>
<td>3</td>
<td>69993</td>
<td>4</td>
</tr>
</tbody>
</table>

#### Table D.3: GCA solution statistics.

<table>
<thead>
<tr>
<th>graph</th>
<th>rep.</th>
<th>GCA ave t.</th>
<th>worst ave</th>
<th>best ave</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{10}$</td>
<td>25</td>
<td>0.017</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>$K_{15}$</td>
<td>25</td>
<td>0.064</td>
<td>6</td>
<td>5.12</td>
</tr>
<tr>
<td>$K_{20}$</td>
<td>25</td>
<td>0.222</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>$K_{30}$</td>
<td>25</td>
<td>1.086</td>
<td>11</td>
<td>10.08</td>
</tr>
<tr>
<td>$K_{40}$</td>
<td>25</td>
<td>4.100</td>
<td>14</td>
<td>13.28</td>
</tr>
<tr>
<td>$K_{50}$</td>
<td>25</td>
<td>10.03</td>
<td>18</td>
<td>16.72</td>
</tr>
<tr>
<td>$K_{60}$</td>
<td>25</td>
<td>21.15</td>
<td>22</td>
<td>20.08</td>
</tr>
<tr>
<td>$K_{70}$</td>
<td>10</td>
<td>51.95</td>
<td>24</td>
<td>23.2</td>
</tr>
<tr>
<td>$K_{80}$</td>
<td>10</td>
<td>88.65</td>
<td>27</td>
<td>26.1</td>
</tr>
<tr>
<td>$K_{90}$</td>
<td>10</td>
<td>142.9</td>
<td>31</td>
<td>29.3</td>
</tr>
<tr>
<td>$K_{100}$</td>
<td>10</td>
<td>218.6</td>
<td>35</td>
<td>33.1</td>
</tr>
<tr>
<td>$r_{10,25}$</td>
<td>25</td>
<td>0.011</td>
<td>3</td>
<td>2.2</td>
</tr>
<tr>
<td>$r_{20,92}$</td>
<td>25</td>
<td>0.060</td>
<td>4</td>
<td>3.96</td>
</tr>
<tr>
<td>$r_{30,230}$</td>
<td>25</td>
<td>0.324</td>
<td>6</td>
<td>5.96</td>
</tr>
<tr>
<td>$r_{40,311}$</td>
<td>25</td>
<td>0.692</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>$r_{50,495}$</td>
<td>25</td>
<td>1.721</td>
<td>8</td>
<td>7.04</td>
</tr>
<tr>
<td>$r_{60,556}$</td>
<td>25</td>
<td>2.179</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>$r_{70,766}$</td>
<td>25</td>
<td>5.309</td>
<td>9</td>
<td>8.04</td>
</tr>
<tr>
<td>$r_{80,939}$</td>
<td>25</td>
<td>7.960</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>$r_{90,1263}$</td>
<td>25</td>
<td>14.31</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>$r_{100,1508}$</td>
<td>25</td>
<td>20.68</td>
<td>11</td>
<td>11</td>
</tr>
</tbody>
</table>
Appendix E

Decomposition of $K_{19,29}$ into six planar subgraphs

In this appendix a planar decomposition of $K_{19,29}$ into six subgraphs is given. Vertices 0...18 belong to the first vertex set and vertices 19...47 to the second set. Subgraphs were constructed by SA and drawn using AGD [2].

Figure E.1: Planar subgraph I of $K_{19,29}$. 
Figure E.2: Planar subgraph II of $K_{19,29}$. 
Figure E.3: Planar subgraph III of $K_{19,29}$. 
APPENDIX E. DECOMPOSITION OF $K_{19,29}$ INTO SIX PLANAR SUBGRAPHS

Figure E.4: Planar subgraph IV of $K_{19,29}$. 
Figure E.5: Planar subgraph $V$ of $K_{19,29}$. 
Figure E.6: Planar subgraph VI of $K_{19,29}$. 
Bibliography


