Semantic Word Cloud Representations: Hardness and Approximation Algorithms

Lukas Barth[1], Sara Irina Fabrikant[2], Stephen Kobourov[3⋆], Anna Lubiw[4], Martin Nöllenburg[1], Yoshio Okamoto[5⋆⋆], Sergey Pupyrev[3,9⋆], Claudio Squarcella[6], Torsten Ueckerdt[7], and Alexander Wolff[8⋆⋆⋆]

[1] Institute of Theoretical Informatics, Karlsruhe Institute of Technology
[2] Department of Geography, University of Zurich
[3] Department of Computer Science, University of Arizona
[4] David R. Cheriton School of Computer Science, University of Waterloo
[5] Dept. Comm. Engineering and Informatics, University of Electro-Communications
[6] Dipartimento di Ingegneria, Roma Tre University
[7] Department of Mathematics, Karlsruhe Institute of Technology
[8] Lehrstuhl für Informatik I, Universität Würzburg
[9] Institute of Mathematics and Computer Science, Ural Federal University

Abstract. We study a geometric representation problem, where we are given a set $\mathcal{B}$ of axis-aligned rectangles (boxes) with fixed dimensions and a graph with vertex set $\mathcal{B}$. The task is to place the rectangles without overlap such that two rectangles touch if the graph contains an edge between them. We call this problem CONTACT REPRESENTATION OF WORD NETWORKS (CROWN). It formalizes the geometric problem behind drawing word clouds in which semantically related words are close to each other. Here, we represent words by rectangles and semantic relationships by edges.
We show that CROWN is strongly NP-hard even if restricted to trees and weakly NP-hard if restricted to stars. We also consider the optimization problem MAX-CROWN where each adjacency induces a certain profit and the task is to maximize the sum of the profits. For this problem, we present constant-factor approximations for several graph classes, namely stars, trees, planar graphs, and graphs of bounded degree. Finally, we evaluate the algorithms experimentally and show that our best method improves upon the best existing heuristic by 45%.

1 Introduction

Word clouds and tag clouds are popular ways to visualize text. They provide an appealing way to summarize the content of a webpage, a research paper, or a political speech. Often such visualizations are used to contrast two documents; for example, word cloud visualizations of the speeches given by the candidates in the 2008 US Presidential elections were used to draw sharp contrast between them in the popular media.

⋆ Supported in part by NSF grants CCF-1115971 and DEB 1053573
⋆⋆ Supported by Grant-in-Aid for Scientific Research from Ministry of Education, Science and Culture, Japan, and Japan Society for the Promotion of Science (JSPS)
⋆⋆⋆ Supported by the ESF EuroGIGA project GraDR (DFG grant Wo 758/5-1).

A practical tool, Wordle [23], which is available on-line, offers high-quality design, graphics, style and functionality, but ignores relationships between words in the input. While some of the more recent word cloud visualization tools aim to incorporate semantics in the layout, none provides any guarantees about the quality of the layout in terms of semantics. We propose a mathematical model of the problem via a simple edge-weighted graph. The vertices in the graph are the words in the document. The edges in the graph correspond to semantic relatedness, with weights corresponding to the strength of the relation. Each vertex must be drawn as an axis-aligned rectangle (*box*, for short) with fixed dimensions. Usually, the dimensions will be determined by the size of the word in a certain font, and the font size will be related to the importance of the word. The goal is to "realize" as many edges as possible by contacts between their corresponding rectangles; see Fig. 1.

Related Work. Hierarchically clustered document collections are often visualized with self-organizing maps [15] and Voronoi treemaps [18]. The early word-cloud approaches did not explicitly use semantic information, such as word relatedness, when placing the words in the cloud. More recent approaches attempt to do so, as in ManiWordle [14] and in parallel tag clouds [4]. The most relevant approaches rely on force-directed graph visualization methods [5] and a seam-carving image processing method together with a force-directed heuristic [24]. The semantics-preserving word cloud problem is related to classic graph layout problems, where the goal is to draw graphs so that vertex labels are readable and Euclidean distances between pairs of vertices are proportional to the underlying graph distance between them. Typically, however, vertices are treated as points and label overlap removal is a post-processing step [7,11].

In *rectangle representations* of graphs, vertices are axis-aligned rectangles with non-intersecting interiors and edges correspond to rectangles with non-zero length common boundary. Every graph that can be represented this way is planar and every triangle in such a graph is a facial triangle; these two conditions are also sufficient to guarantee a rectangle representation [22]. In a recent survey, Felsner [9] reviews many rectangulation variants, including squarings. Algorithms for area-preserving rectangular cartograms are also related [21]. Area-universal rectangular representations where vertex weights are represented by area have been characterized [8] and edge-universal representations, where edge weights are represented by length of contacts have been studied [19]. Unlike cartograms, in our setting there is no inherent geography, and hence, words can be positioned anywhere. Moreover, each word has fixed dimensions enforced by its frequency in the input text, rather than just fixed area.

Our Contribution. The input to the problem variants we consider is a (multi)set $\mathcal{B}$ of axis-aligned boxes $B_1, \ldots, B_n$ with fixed positive dimensions and an edge-weighted undirected graph $G = (\mathcal{B}, E)$, called the *profit graph*. Box B_i has an associated *size* (w_i, h_i), where w_i and h_i are its width and height. For some of our results, boxes may be rotated by $90°$, which means exchanging w_i and h_i. By scaling appropriately, we may always assume that all heights and widths are positive integers. The vertex set of G is $\mathcal{B}$. Every edge $(B_i, B_j) \in E$ has a positive weight p_{ij}, called its *profit*, representing the gain for making boxes B_i and B_j touch. A *representation* of $\mathcal{B}$ is a map that associates with each box a position in the plane so that no two boxes overlap. A *contact* between

Fig. 1: A hierarchical word cloud for complexity classes. A class is above another class when the former contains the latter. The font size is the square root of millions of Google hits for the corresponding word. This is an instance of the problem variant HIER-CROWN.

two boxes is a maximal line segment *of positive length* in the boundary of both. If two boxes are in contact, we say that they *touch*. If two boxes touch and one lies above the other, we call this a *vertical contact*. We define *horizontal contact* symmetrically. We say that a representation *realizes* an edge $(B_i, B_j) \in E$ if B_i and B_j touch. Finally, we define the *total profit* of a representation to be the sum of profits over all edges of G that the representation realizes. Our problems and results are as follows.

Contact Representation of Word Networks (CROWN) is to decide whether there exists a representation of the given boxes that realizes all edges of the profit graph. This is equivalent to deciding whether there is a representation whose contact graph contains the profit graph as a subgraph. If such a representation exists, we say that it *realizes the profit graph* and that the given instance of CROWN is *realizable*.

We show that CROWN is strongly NP-hard even if restricted to trees and weakly NP-hard if restricted to stars; see Theorem 1. We also consider two variants of the problem that can be solved efficiently. First we present a linear-time algorithm for CROWN on so-called irreducible triangulations; see Section 2.1. Then we turn to the problem variant **HIER-CROWN**, where the profit graph is a single-source directed acyclic graph with fixed plane embedding, and the task is to find a representation in which each edge corresponds to a vertical contact directed upwards; see Fig. 1. We solve this variant efficiently; see Section 2.2.

MAX-CROWN is the optimization version of CROWN where the task is to find a box representation maximizing the total profit. We present constant-factor approximation algorithms for stars, trees, and planar graphs, and a $2/\lfloor \Delta + 1 \rfloor$-approximation for graphs of maximum degree Δ; see Section 3. We have implemented two approximation algorithms and evaluated them experimentally in comparison to three existing algorithms (two of which are semantics-aware). Based on a dataset of 120 Wikipedia documents our best method outperforms the best previous method by more than 45%; see Section 5. We also consider an extremal version of the MAX-CROWN problem and show that the complete graph K_n ($n \geq 7$) with unit profits can always be realized with total profit $2n - 2$, which is sometimes the best possible; see Section 3.2.

AREA-CROWN is as follows: Given a realizable instance of CROWN, find a representation that realizes the profit graph and minimizes the area of the representation's bounding box. We show that the problem is NP-hard even if restricted to paths; see Section 4.

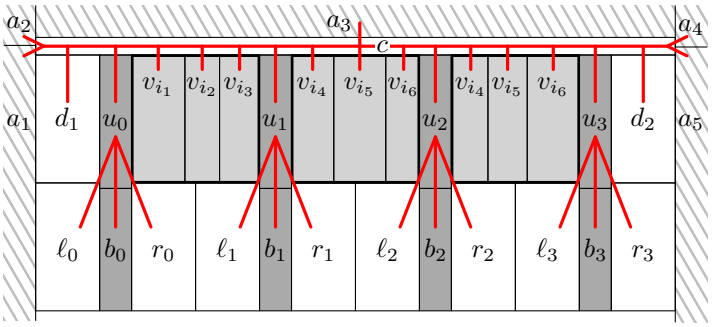

Fig. 2: Given an instance S of 3-PARTITION, we construct a tree T_S (thick red line segments) and define boxes such that T_S has a realization if and only if S is feasible.

2 The CROWN Problem

In this section, we investigate the complexity of CROWN for several graph classes.

Theorem 1. CROWN *is (strongly) NP-hard. The problem remains strongly NP-hard even if restricted to trees and is weakly NP-hard if restricted to stars.*

Proof. To show that CROWN on stars is weakly NP-hard, we reduce from the weakly NP-hard problem PARTITION, which asks whether a given multiset of n positive integers $a_1, \ldots, a_n$ that sum to B can be partitioned into two subsets, each of sum $B/2$. We construct a star graph whose central vertex corresponds to a $(B/2, \delta)$-box (for some $0 < \delta < \min_i a_i$). We add four leaves corresponding to (B, B)-squares and, for $i = 1, \ldots, n$, a leaf corresponding to an (a_i, a_i)-square. It is easy to verify that there is a realization for this instance of CROWN if and only if the set can be partitioned.

To show that CROWN is (strongly) NP-hard, we reduce from 3-PARTITION: Given a multiset S of $n = 3m$ integers with $\sum S = mB$, is there a partition of S into m subsets $S_1, \ldots, S_m$ such that $\sum S_i = B$ for each i? It is known that 3-PARTITION is NP-hard even if, for every $s \in S$, we have $B/4 < s < B/2$, which implies that each of the subsets $S_1, \ldots, S_m$ must contain exactly three elements [12].

Given an instance $S = \{s_1, s_2, \ldots, s_n\}$ of 3-PARTITION as described above, we define a tree T_S on $n + 4(m-1) + 7$ vertices as in Fig. 2 (for $n = 9$ and $m = 3$). Let $K = (m+1)B + m + 1$. We make a vertex c of size $(K, 1/2)$. For each $i = 1, \ldots, n$, we make a vertex v_i of size (s_i, B). Let $\varepsilon \in (0, B/2)$. For each $j = 0, \ldots, m$, we make a vertex u_j of size $(1, B + \varepsilon)$, a vertex b_j of size $(1, B - \varepsilon)$, and vertices ℓ_j and r_j of size $(B/2, B)$. Finally, we make vertices $a_1, \ldots, a_5$ of size (K, K), and vertices d_1 and d_2 of size $(B/2, B)$. The tree T_S is as shown by the thick lines in Fig. 2: vertex c is adjacent to all the v_i's, u_j's, a's, and d's; and each vertex u_j is adjacent to b_j, ℓ_j, and r_j.

We claim that an instance S of 3-PARTITION is feasible if and only if T_S can be realized with the given box sizes. It is easy to see that T_S can be realized if S is feasible: we simply partition vertices $v_1, \ldots, v_n$ into groups of three (by vertices $u_0, \ldots, u_m$) in the same way as their widths $s_1, \ldots, s_n$ are partitioned in groups of three; see Fig. 2.

For the other direction, consider any realization of T_S. Let us refer to the box of some vertex v also as v. Since c touches the five large squares $a_1, \ldots, a_5$, at least three sides of c are partially covered by some a_k and at least one horizontal side of c is completely covered by some a_k. Since c has height 1/2 only, but touches all the v_i's and u_j's and d_1 and d_2 (each of height $B > 1$), all these boxes must touch c on its free horizontal side, say, the bottom side. Furthermore, the sum of the widths of the boxes exactly matches the width of c; so they must pack side by side in some order.

This means that the only free boundary of u_j is at the bottom, and u_j must make contact there with b_j, ℓ_j, and r_j. This is only possible if b_j is placed directly beneath u_j, and ℓ_j and r_j make contact with the bottom corners of u_j. (They need not appear to the left and right as shown in Fig. 2.) Because the sum of the widths of the b_j's, ℓ_j's, and r_j's exactly matches the width of c, they must pack side by side, and therefore the u_j's are spaced distance B apart. There is a gap of width $B/2$ before the first u_j and after the last u_j. These gaps are too wide for one box in $v_1, \ldots, v_n$ and too small for two of them since their widths are contained in the *open* interval $(B/4, B/2)$. Therefore, the boxes d_1 and d_2 must occupy these gaps, and the boxes $v_1, \ldots, v_n$ are packed into m groups each of width B, as required. □

In case rectangles may be rotated, both proofs still hold: the weak NP-hardness proof for stars still works because all boxes are squares–except the central one. The strong NP-hardness for trees also still holds, basically because the boxes $a_1, \ldots, a_5$ are squares and because all boxes (except c) are at least as high as wide (and at least as wide as c is high), so there is no advantage to rotating any box.

Although CROWN is NP-hard in general, on some graph classes the problem can be solved efficiently. In the remainder of this section, we investigate such a class: irreducible triangulations. We also consider a restricted variant of CROWN: HIER-CROWN.

2.1 The CROWN Problem on Irreducible Triangulations

A box representation is called a *rectangular dual* if the union of all rectangles is again a rectangle whose boundary is formed by exactly four rectangles. A graph G admits a rectangular dual if and only if G is planar, internally triangulated, has a quadrangular outer face and does not contain separating triangles [2]. Such graphs are known as *irreducible triangulations*. The four outer vertices of an irreducible triangulation are denoted by v_N, v_E, v_S, v_W in clockwise order around the outer quadrangle. An irreducible triangulation G may have exponentially many rectangular duals. Any rectangular dual of G, however, can be built up by placing one rectangle at a time, always keeping the union of the placed rectangles in staircase shape.

Theorem 2. CROWN *on irreducible triangulations can be solved in linear time.*

Proof (sketch). The algorithm greedily builds up the box representation, similarly to an algorithm for edge-proportional rectangular duals [19]. We define a *concavity* as a point on the boundary of the so-far constructed representation, which is a bottom-right or top-left corner of some rectangle. Start with a vertical and a horizontal ray emerging from the same point p, as placeholders for the right side of v_W and the top side of v_S,

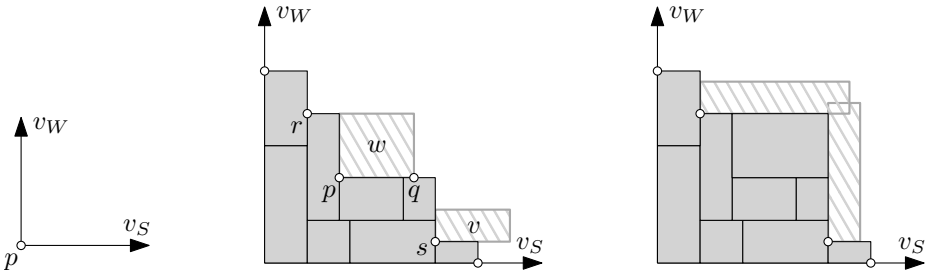

Fig. 3: Left: starting configuration with rays v_S and v_W. Center: representation at an intermediate step: vertex w fits into concavity p and results in a staircase, vertex v fits into concavity s but does not result in a staircase. Adding box w to the representation introduces a new concavity q and allows wider boxes to be placed at r. Right: no box can be placed, so the algorithm terminates.

respectively. Then at each step consider a concavity, with p as the initial one. Since each concavity p is contained in exactly two rectangles, there exists a unique rectangle R_p that is yet to be placed and has to touch both these rectangles. If by adding R_p we still have a staircase shape representation, then we do so. If no such rectangle can be added, we conclude that G is not realizable; see Fig. 3. The complete proof is in the full version [1]. □

2.2 The HIER-CROWN Problem

The HIER-CROWN problem is a restricted variant of the CROWN problem that can be used to create word clouds with a hierarchical structure; see Fig. 1. The input is a directed acyclic graph G with only one sink and with a plane embedding. The task is to find a representation that *hierarchically realizes* G, meaning that for each directed edge (v, u) in G the top of the box v is in contact with the bottom of the box u.

If the embedding of G is not fixed, the problem is NP-hard even for a tree, by an easy adaptation of the proof of Theorem 1. (Remove the vertices a_2, a_3, a_4, and orient the remaining edges of T_S upward according to the representation shown in Fig. 2.) However, if we fix the embedding of the profit graph G, then HIER-CROWN can be solved efficiently.

Theorem 3. HIER-CROWN *can be solved in polynomial time.*

Proof. Let G be the given profit graph, with vertex set $\mathcal{B} = \{B_1, \ldots, B_n\}$, where B_i has height h_i and width w_i, and B_1 is the unique sink. We first check that the orientation and embedding of G are compatible, that is, that incoming edges and outgoing edges are consecutive in the cyclic order around each vertex.

The main idea is to set up a system of linear equations for the x- and y-coordinates of the sides of the boxes. Let variables t_i and b_i represent the y-coordinates of the top and bottom of B_i respectively, and variables ℓ_i and r_i represent the x-coordinates of the left and right of B_i, respectively. For each $i = 1, \ldots, n$, impose the linear constraints

$t_i = b_i + h_i$ and $r_i = \ell_i + w_i$. For each directed edge (B_i, B_j), impose the constraints $t_i = b_j, r_i \geq \ell_j + 1/2$, and $r_j \geq \ell_i + 1/2$. The last two constraints force B_i and B_j to share some x-range of positive length in which they touch. Initialize $t_1 = 0$.

With these equations, variables t_i and b_i are completely determined since every box B_i has a directed path to B_1. Furthermore, the values for t_i and b_i can be found using a depth-first-search of G starting from B_1.

The x-coordinates are not yet determined and depend on the horizontal order of the boxes, which can be established as follows. We scan the boxes from top to bottom, keeping track of the left-to-right order of boxes intersected by a horizontal line that sweeps from $y = 0$ downwards. Initially the line is at $y = 0$ and intersects only B_1. When the line reaches the bottom of a box B, we replace B in the left-to-right order by all its predecessors in G, using the order given by the plane embedding. In case multiple boxes end at the same y-coordinate, we make the update for all of them. Whenever boxes B_a and B_b appear consecutively in the left-to-right order, we impose the constraint $r_a \leq \ell_b$.

The scan can be performed in $O(n \log n)$ time using a priority queue to determine which boxes in the current left-to-right order have maximum b_i value. The resulting system of equations has size $O(n)$ (because the constraints correspond to edges of a planar graph). It is straightforward to verify that the system of equations has a solution if and only if there is a representation of the boxes that hierarchically realizes G. The constraints define a linear program (LP) that can be solved efficiently. (A feasible solution can be found faster than with an LP, but we omit the details in this paper.) □

We can show that HIER-CROWN becomes weakly NP-complete if rectangles may be rotated, by a simple reduction from SUBSET SUM (for details, see the full version [1]).

3 The MAX-CROWN Problem

In this section, we study approximation algorithms for MAX-CROWN and consider an extremal variant of the problem.

3.1 Approximation Algorithms

We present approximation algorithms for MAX-CROWN restricted to certain graph classes. Our basic building blocks are an approximation algorithm for stars and an exact algorithm for cycles. Our general technique is to find a collection of disjoint stars or cycles in a graph. We begin with stars, using a reduction to the MAXIMUM GENERALIZED ASSIGNMENT PROBLEM (GAP) defined as follows: Given a set of bins with capacity constraints and a set of items that may have different sizes and values in each bin, pack a maximum-value subset of items into the bins. It is known that the problem is NP-hard (KNAPSACK and BIN PACKING are special cases of GAP), and there exists a $(1 - 1/e)$-approximation algorithm [10]. In the remainder, we assume that there is an α-approximation algorithm for GAP, setting $\alpha = 1 - 1/e > 0.632$.

Theorem 4. *There exists an α-approximation algorithm for* MAX-CROWN *on stars.*

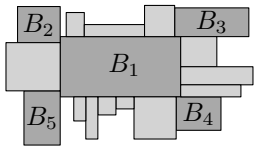

Fig. 4: A solution to an instance of MAX-CROWN whose profit graph is a star with center B_1.

Proof. We can solve instances with $n < 5$ by brute force exactly, so let's assume that $n \geq 5$. Let B_1 denote the box corresponding to the center of the star. Given an optimal solution to MAX-CROWN, we can modify it (by sliding boxes along the sides of B_1) such that there are four boxes B_2, B_3, B_4, B_5 whose contact with B_1 has length exactly $1/2$. In particular, each of these boxes touches exactly one corner of B_1. The problem reduces to choosing four corner boxes and the way they touch B_1, and assigning the remaining boxes to one of the sides of B_1; see Fig. 4.

Each corner of B_1 can be touched in two different ways, via its incident horizontal or vertical sides. Depending on the way the corners of B_1 are touched, we create the following instance of GAP. We introduce eight bins, one bin for each side of B_1 with appropriately adjusted sizes and one bin of size 1 for each corner. For $i = 2, \ldots, n$, the value of item B_i is the profit of the edge (B_1, B_i). The size of B_i is w_i for each horizontal bin, h_i for each vertical bin, and 1 for each corner bin. For each of the 16 ways the corners of B_1 can be touched, we apply an α-approximation algorithm for GAP [10]. Hence, we obtain an α-approximation for MAX-CROWN. □

In the case where rectangles may be rotated by $90°$, the MAX-CROWN problem on a star reduces to an easier problem, the MULTIPLE KNAPSACK PROBLEM, where every item has the same size and value no matter which bin it is placed in. This is because, for non-corner bins, we will always attach a rectangle B to the central rectangle of the star using the smaller dimension of B. For the corner bins, we can try all possible choices of which box to put into which bin. There is a PTAS for MULTIPLE KNAPSACK [3]. Therefore, there is a PTAS for MAX-CROWN on stars if we may rotate rectangles.

A *star forest* is a disjoint union of stars. Theorem 4 applies to a star forest since we can combine the solutions for the disjoint stars.

Theorem 5. MAX-CROWN *on the class of graphs that can be partitioned in polynomial time into k star forests admits an α/k-approximation algorithm.*

Proof. We partition the edges of the profit graph into k star forests, apply the approximation algorithm of Theorem 4 to each of them, and take the best of the k solutions. We claim that this (polynomial-time) method yields the desired approximation factor.

Consider an optimum solution, and let W_{opt} be its profit. By the pigeon-hole principle, our partition of the profit graph contains a star forest F that realizes a profit of at least W_{opt}/k in the optimum solution. Hence, on F, the approximation algorithm of Theorem 4 achieves a profit of at least $\alpha W_{\mathrm{opt}}/k$. □

Corollary 1. MAX-CROWN *admits an $\alpha/2$-approximation algorithm on trees and an $\alpha/5$-approximation algorithm on planar graphs.*

Proof. It is easy to partition any tree into two star forests in linear time. Moreover, it is known that every planar graph has star arboricity at most 5, that is, it can be partitioned into at most five star forests, and such a partition can be found in polynomial time [13]. The results now follow directly from Theorem 5. □

Our algorithms involve approximating a number of GAP instances, using the LP-based algorithm of Fleischer et al. [10]. Because of this, the runtime of our approximation algorithms is dominated by the runtime of solving linear programs.

Our star forest partition method is possibly not optimal. Nguyen et al. [17] show how to find a star forest of an arbitrary weighted graph carrying at least half of the profits of an optimal star forest in polynomial time. We cannot, however, guarantee that the approximation of the optimal star forest carries a positive fraction of the total profit in an optimal solution to MAX-CROWN. Hence, approximating MAX-CROWN for general graphs remains an open problem. As a first step into this direction, we present a constant-factor approximation for profit graphs with bounded maximum degree. First we need the following lemma.

Lemma 1. *Given $n \geq 3$ boxes and an n-cycle defined on them, we can find a representation realizing the n-cycle in linear time.*

Proof. Let $C = (B_1, \ldots, B_n)$ be the given cycle, let W be the sum of all the widths, that is, $W = \sum_i w_i$, and let t be the maximum index such that $\sum_{i \leq t} w_i < W/2$. We place $B_1, \ldots, B_t$ in this order side by side from left to right with their bottom sides on a horizontal line h; see Fig. 5. We call this the *top channel*. Starting at the same point on h, we place $B_n, B_{n-1}, \ldots, B_{t+2}$ in this order side by side from left to right with their top sides on h. We call this the *bottom channel*. Note that B_1 and B_n are in contact. It remains to place B_{t+1} in contact with B_t and B_{t+2}. It is easy to see that the following works: add B_{t+1} to the channel of minimum width or, in case of a tie, place B_t straddling the line h; see Fig. 5. □

Following the idea of Theorem 5, we can approximate MAX-CROWN by applying Lemma 1 to a partition of the profit graph into sets of disjoint cycles.

Theorem 6. MAX-CROWN *on the class of graphs that can be partitioned into k sets of disjoint cycles (in polynomial time) admits a (polynomial-time) algorithm that achieves total profit at least $\frac{1}{k} \sum_{i \neq j} p_{ij}$. In particular, there is a $1/k$-approximation algorithm for* MAX-CROWN *on this graph class.*

Corollary 2. MAX-CROWN *on graphs of maximum degree Δ admits a $2/\lfloor \Delta + 1 \rfloor$-approximation.*

Proof. As Peterson [20] shows, the edges of any graph of maximum degree Δ can be covered by $\lceil \Delta/2 \rceil$ sets of cycles and paths, and such sets can be found in polynomial time. The result now follows from Theorem 6. □

3.2 An Extremal MAX-CROWN Problem

In the following, we bound the maximum number of contacts that can be made when placing n boxes. It is easy to see that for $n = 2, 3, 4$ any set of boxes allows $2n - 3$ contacts. For larger n we have:

Theorem 7. *For $n \geq 7$ and any set of n boxes, the boxes can be placed in the plane to realize $2n-2$ contacts. For some sets of boxes this is the best possible.*

Proof. Let $B_1, \ldots, B_n$ be any set of boxes. First we place $k \in \{5, 6, 7\}$ boxes to make $2(k-1)$ contacts, and then place the remaining boxes to make 2 contacts each for a total of $2(k-1) + 2(n-k) = 2n-2$ contacts. Let B_1 and B_2 be the boxes with largest height, and B_3 and B_4 be the boxes with largest width. Let B_5 be any further box. Place the five boxes as in Fig. 6. This realizes 8 contacts, unless one or two of B_1, B_2 has the same height as B_5, in which case we consider one or two further boxes B_6, B_7 and represent in total 10 or 12 contacts as in Fig. 6.

Place the remaining boxes one by one as in the proof of Lemma 1 along the horizontal line between B_2 and B_3. Then each remaining box makes two new contacts.

Next we describe a set of n boxes for which the maximum number of contacts is $2n-2$. Let B_i be a square box of side length 2^i. Consider any placement of the boxes and partition the contacts into the set of horizontal contacts and the set of vertical contacts. From the side lengths of the boxes, it follows that neither set of contacts contains a cycle. Thus each set of contacts has size at most $n-1$ for a total of $2n-2$. □

4 The AREA-CROWN Problem

The same profit graph can often be realized by different box representations, not all of which are equally useful or visually appealing when viewed as word clouds. In this section we consider the AREA-CROWN problem and show that finding a "compact" representation that fits into a small bounding box is another NP-hard problem.

The reduction is from the (strongly) NP-hard problem 2D BOX PACKING: The input is a set $\mathcal{R}$ of n rectangles with width and height functions $w\colon \mathcal{R} \to \mathbb{N}$ and $h\colon \mathcal{R} \to \mathbb{N}$, and a box of width W and height H. All the input numbers are bounded by some polynomial in n. The task is to pack the given rectangles into the box. The problem is known to be NP-complete even if the box is a square, that is, if $W = H$ [16].
The BOX PACKING problem is equivalent to AREA-CROWN when the profit graph has no edges. Edges in the profit graph, however, impose additional constraints on the representation, which may make AREA-CROWN easier for certain (simple) profit graph classes. In the full version [1], we show that this is not the case.

Theorem 8. AREA-CROWN *is (strongly) NP-hard even on paths.*

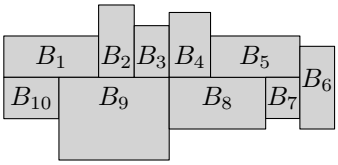

Fig. 5: Example for Lemma 1: Realizing the 10-cycle $(B_1, \ldots, B_{10})$.

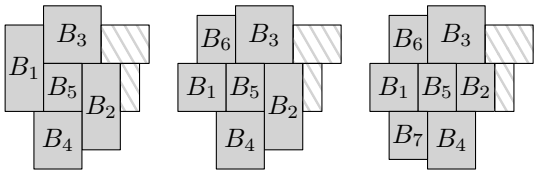

Fig. 6: Examples for Theorem 7: 8, 9, or 10 adjacencies with 5, 6, or 7 boxes, respectively.

5 Experimental Results

We implemented two new methods for constructing word clouds: the STAR FOREST algorithm based on extracting star forests (Corollary 1) and the CYCLE COVER algorithm based on decomposing edges of a graph into cycle covers (Theorem 6). We compared the two algorithms to the following existing methods: WORDLE [23], CPDWCV [5], and SEAM CARVING [24]. Our dataset consists of 120 Wikipedia documents, each with 400 words or more. Frome these, we removed stop words (e.g., "the") and constructed profit graphs G_{50} and G_{100} for the 50 and 100 most frequent words, respectively. We set profits using the so-called *Latent Semantic Analysis* [6] based on the co-occurrence of these words within the same sentence. For details, see the full version [1].

We compare the percentage of realized profit in the box representations. Since STAR FOREST handles planar profit graphs, we first extracted maximal planar subgraphs of the profit graphs and then applied the algorithm the the planar subgraphs. The percentage of realized profit is presented in the table below. Our results indicate that, in terms of the realized profit, CYCLE COVER outperforms existing approaches, realizing more than 17% (13%) of the total profit of graphs with 50 (100) vertices, that is, 45% (55%) more than the second best known heuristic, CPDWCV. On the other hand, existing algorithms may perform better in terms of compactness, aspect ratio, and other aesthetic criteria; we leave a deeper comparison of word cloud algorithms to the future.

Algorithm	Realized Profit of G_{50}	Realized Profit of G_{100}
WORDLE [23]	3.4%	2.2%
CPDWCV [5]	12.2%	8.9%
SEAM CARVING [24]	7.4%	5.2%
STAR FOREST	11.4%	8.2%
CYCLE COVER	17.8%	13.8%

6 Conclusions and Future Work

We formulated the Contact Representation of Word Networks (CROWN) problem, motivated by the desire to provide theoretical guarantees for semantics-preserving word cloud visualization. We showed that some variants of CROWN are NP-hard, gave efficient algorithms for others, and presented approximation algorithms. A natural open problem is to find an approximation algorithm for general graphs with arbitrary profits.

Acknowledgments. This work began at Dagstuhl Seminar 12261. We thank organizers and participants (in particular Therese Biedl), as well as Steve Chaplick and Günter Rote.

References

1. Barth, L., Fabrikant, S.I., Kobourov, S., Lubiw, A., Nöllenburg, M., Okamoto, Y., Pupyrev, S., Squarcella, C., Ueckerdt, T., Wolff, A.: Semantic word cloud representations: Hardness and approximation algorithms. Arxiv report arxiv.org/abs/1311.4778 (2013)

2. Buchsbaum, A.L., Gansner, E.R., Procopiuc, C.M., Venkatasubramanian, S.: Rectangular layouts and contact graphs. ACM Trans. Algorithms 4(1) (2008)
3. Chekuri, C., Khanna, S.: A polynomial time approximation scheme for the multiple knapsack problem. SIAM J. Comput. 35(3), 713–728 (2005)
4. Collins, C., Viégas, F.B., Wattenberg, M.: Parallel tag clouds to explore and analyze faceted text corpora. In: Proc. IEEE Symp. Vis. Analytics Sci. Tech. pp. 91–98 (2009)
5. Cui, W., Wu, Y., Liu, S., Wei, F., Zhou, M., Qu, H.: Context-preserving dynamic word cloud visualization. IEEE Comput. Graphics Appl. 30(6), 42–53 (2010)
6. Dumais, S.T.: Latent semantic analysis. Annu. Rev. Inform. Sci. Tech. 38(1), 188–230 (2004)
7. Dwyer, T., Marriott, K., Stuckey, P.J.: Fast node overlap removal. In: Healy, P., Nikolov, N.S. (eds.) GD 2005. LNCS 3843, pp. 153–164. Springer, Heidelberg (2006)
8. Eppstein, D., Mumford, E., Speckmann, B., Verbeek, K.: Area-universal and constrained rectangular layouts. SIAM J. Comput. 41(3), 537–564 (2012)
9. Felsner, S.: Rectangle and square representations of planar graphs. In: Pach, J. (ed.) Thirty Essays on Geometric Graph Theory, pp. 213–248. Springer, Heidelberg (2013)
10. Fleischer, L., Goemans, M.X., Mirrokni, V.S., Sviridenko, M.: Tight approximation algorithms for maximum separable assignment problems. Math. Oper. Res. 36(3), 416–431 (2011)
11. Gansner, E.R., Hu, Y.: Efficient, proximity-preserving node overlap removal. J. Graph Algortihms Appl. 14(1), 53–74 (2010)
12. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York, NY, USA (1979)
13. Hakimi, S.L., Mitchem, J., Schmeichel, E.F.: Star arboricity of graphs. Discrete Math. 149(1–3), 93–98 (1996)
14. Koh, K., Lee, B., Kim, B.H., Seo, J.: Maniwordle: Providing flexible control over Wordle. IEEE Trans. Vis. Comput. Graph. 16(6), 1190–1197 (2010)
15. Lagus, K., Honkela, T., Kaski, S., Kohonen, T.: Self-organizing maps of document collections: A new approach to interactive exploration. In: Simoudis, E., Han, J., Fayyad, U.M. (eds.) KDD 1996. pp. 238–243. AAAI Press (1996)
16. Leung, J.Y.T., Tam, T.W., Wong, C., Young, G.H., Chin, F.Y.: Packing squares into a square. J. Parallel Distrib. Comput. 10(3), 271–275 (1990)
17. Nguyen, C.T., Shen, J., Hou, M., Sheng, L., Miller, W., Zhang, L.: Approximating the spanning star forest problem and its application to genomic sequence alignment. SIAM J. Comput. 38(3), 946–962 (2008)
18. Nocaj, A., Brandes, U.: Organizing search results with a reference map. IEEE Trans. Vis. Comput. Graphics 18(12), 2546–2555 (2012)
19. Nöllenburg, M., Prutkin, R., Rutter, I.: Edge-weighted contact representations of planar graphs. In: Didimo, W., Patrignani, M. (eds.) GD 2012. LNCS 7704, pp. 224–235. Springer, Heidelberg (2013)
20. Petersen, J.: Die Theorie der regulären Graphen. Acta Mathematica 15(1), 193–220 (1891)
21. Raisz, E.: The rectangular statistical cartogram. Geogr. Review 24(3), 292–296 (1934)
22. Thomassen, C.: Interval representations of planar graphs. J. Combin. Theory, Ser. B 40(1), 9–20 (1986)
23. Viégas, F.B., Wattenberg, M., Feinberg, J.: Participatory visualization with Wordle. IEEE Trans. Vis. Comput. Graphics 15(6), 1137–1144 (2009)
24. Wu, Y., Provan, T., Wei, F., Liu, S., Ma, K.L.: Semantic-preserving word clouds by seam carving. Comput. Graphics Forum 30(3), 741–750 (2011)