The Alcuin Number of a Graph

Péter Csorba, Cor A.J. Hurkens, and Gerhard J. Woeginger

Department of Mathematics and Computer Science
TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

Abstract. We consider a planning problem that generalizes Alcuin's river crossing problem (also known as: The wolf, goat, and cabbage puzzle) to scenarios with arbitrary conflict graphs. We derive a variety of combinatorial, structural, algorithmical, and complexity theoretical results around this problem.

Keywords: Transportation problem; scheduling; graph theory.

1 Introduction

Alcuin's river crossing problem. The Anglo-Saxon monk Alcuin (735–804 A.D.) was one of the leading scholars of his time. He served as head of Charlemagne's Palace School at Aachen, he developed the Carolingian minuscule (a script which has become the basis of the way the letters of the present Roman alphabet are written), and he wrote a number of elementary texts on arithmetic, geometry, and astronomy. His book *"Propositiones ad acuendos iuvenes"* (Problems to sharpen the young) is perhaps the oldest collection of mathematical problems written in Latin. It contains the following well-known problem.

> A man had to transport to the far side of a river a wolf, a goat, and a bundle of cabbages. The only boat he could find was one which would carry only two of them. For that reason he sought a plan which would enable them all to get to the far side unhurt. Let him, who is able, say how it could be possible to transport them safely?

In a safe transportation plan, neither wolf and goat nor goat and cabbage can be left alone together. Alcuin's river crossing problem differs significantly from other mediaeval puzzles, since it is neither geometrical nor arithmetical but purely combinatorial. Biggs [3] mentions it as one of the oldest combinatorial puzzles in the history of mathematics. Ascher [1] states that the problem also shows up in Gaelic, Danish, Russian, Ethiopian, Suaheli, and Zambian folklore. Borndörfer, Grötschel & Löbel [4] use Alcuin's problem to provide the reader with a leisurely introduction into integer programming.

Graph-theoretic model. We consider the following generalization of Alcuin's problem to arbitrary graphs $G = (V, E)$. Now the man has to transport a set V of items/vertices across the river. Two items are connected by an edge in

D. Halperin and K. Mehlhorn (Eds.): ESA 2008, LNCS 5193, pp. 320–331, 2008.

E, if they are *conflicting* and thus cannot be left alone together without human supervision. The available boat has capacity $b \geq 1$, and thus can carry the man together with any subset of at most b items. A *feasible schedule* is a finite sequence of triples (L_1, B_1, R_1), (L_2, B_2, R_2), $\ldots$, (L_s, B_s, R_s) of subsets of the item set V that satisfies the following conditions (FS1)–(FS3). The odd integer s is called the *length* of the schedule.

(FS1) For every k, the sets L_k, B_k, R_k form a partition of V. The sets L_k and R_k form stable sets in G. The set B_k contains at most b elements.
(FS2) The sequence starts with $L_1 \cup B_1 = V$ and $R_1 = \emptyset$, and the sequence ends with $L_s = \emptyset$ and $B_s \cup R_s = V$.
(FS3) For even $k \geq 2$, we have $B_k \cup R_k = B_{k-1} \cup R_{k-1}$ and $L_k = L_{k-1}$. For odd $k \geq 3$, we have $L_k \cup B_k = L_{k-1} \cup B_{k-1}$ and $R_k = R_{k-1}$.

Intuitively speaking, the kth triple encodes the kth boat trip: L_k contains the items on the left bank, B_k the items in the boat, and R_k the items on the right bank. Odd indices correspond to forward boat trips from left to right, and even indices correspond to backward trips from right to left. Condition (FS1) states that the sets L_k and R_k must not contain conflicting item pairs, and that set B_k must fit into the boat. Condition (FS2) concerns the first boat trip (where the man has put the first items into the boat) and the final trip (where the man transports the last items to the right bank). Condition (FS3) says that whenever the man reaches a bank, he may arbitrarily re-divide the set of items that currently are on that bank and in the boat.

1. $w, c \mid g \rightarrow \mid \emptyset$	2. $w, c \mid \leftarrow \emptyset \mid g$
3. $w \mid c \rightarrow \mid g$	4. $w \mid \leftarrow g \mid c$
5. $g \mid w \rightarrow \mid c$	6. $g \mid \leftarrow \emptyset \mid w, c$
7. $\emptyset \mid g \rightarrow \mid w, c$	

Fig. 1. A solution for Alcuin's river crossing puzzle. The partitions L_k, B_k, R_k are listed as $L_k \mid B_k \mid R_k$; the arrows $\rightarrow$ and $\leftarrow$ indicate the current direction of the boat.

We are interested in the smallest possible capacity of a boat for which a graph $G = (V, E)$ possesses a feasible schedule; this capacity is called the *Alcuin number* $\textsc{Alcuin}(G)$ of the graph. In our graph-theoretic model Alcuin's river crossing problem corresponds to the path P_3 with three vertices w(olf), g(oat), c(abbage) and two edges $[w, g]$ and $[g, c]$. Figure 1 lists one possible feasible schedule for a boat of capacity $b = 1$. This implies $\textsc{Alcuin}(P_3) = 1$.

A natural problem variant puts a hard constraint on the length of the schedule: Let $t \geq 1$ be an odd integer. The smallest possible capacity of a boat for which G possesses a feasible schedule with at most t boat trips is called the *t-trip constrained* Alcuin number $\textsc{Alcuin}_t(G)$. Of course, $\textsc{Alcuin}_1(G) = |V|$ holds for

any graph G. For our example in Figure 1, it can be seen that $\textsc{Alcuin}_1(P_3) = 3$, that $\textsc{Alcuin}_t(P_3) = 2$ for $t \in \{3, 5\}$, and that $\textsc{Alcuin}_t(P_3) = 1$ for $t \geq 7$.

Known results. The idea of generalizing Alcuin's problem to arbitrary conflict graphs goes back (at least) to Prisner [13] and Bahls [2]: Prisner introduced it in 2002 in his course on Discrete Mathematics at the University of Maryland, and Bahls discussed it in 2005 in a talk in the Mathematics Seminar at the University of North Carolina.

Bahls [2] (and later Lampis & Mitsou [9]) observed that it is NP-hard to compute the Alcuin number exactly; Lampis & Mitsou [9] also showed that the Alcuin number is hard to approximate. These negative results follow quite easily from the close relationship between the Alcuin number and the vertex cover number; see Lemma 1. The papers [2,9] provide a complete analysis of the Alcuin number of trees. Finally, Lampis & Mitsou [9] proved that the computation of the trip constrained Alcuin number $\textsc{Alcuin}_3(G)$ is NP-hard.

Our results. We derive a variety of combinatorial and algorithmical results around the Alcuin number. As a by-product, our results also settle several open questions from [9].

Our main result is the structural characterization of the Alcuin number in Section 3. This characterization yields an NP-certificate for the Alcuin number. It also yields that every feasible schedule (possibly of exponential length) can be transformed into a feasible schedule of linear length.

The close relationship between the Alcuin number and the vertex cover number of a graph (see Lemma 1) naturally divides graphs into so-called *small-boat* and *big-boat* graphs. In Section 4 we derive a number of combinatorial lemmas around the division line between these two classes. All these lemmas fall out quite easily from our structural characterization. Standard techniques yield that computing the Alcuin number belongs to the class FPT of fixed-parameter tractable problems; see Section 5.

In Section 6 we discuss the computational hardness of the Alcuin number. First, we provide a new NP-hardness proof for this problem. Other proofs of this result are already in the literature [2,9], but we think that our three-line argument is considerably simpler than all previously published arguments. Secondly, we establish the NP-hardness of distinguishing small-boat graphs from big-boat graphs. Thirdly, we prove NP-hardness of computing the t-trip constrained Alcuin number $\textsc{Alcuin}_t(G)$ for every fixed value $t \geq 3$.

In Section 7 we finally apply our machinery to chordal graphs, trees, and planar graphs, for which we get concise descriptions of the division line between small-boat and big-boat graphs. We also show that the Alcuin number of a bipartite graph can be determined in polynomial time.

2 Definitions and Preliminaries

We first recall some basic definitions. A set $S \subseteq V$ is a *stable* set for a graph $G = (V, E)$, if S does not induce any edges. The *stability number* $\alpha(G)$ of G

is the size of a largest stable set in G. A set $W \subseteq V$ is a *vertex cover* for G if $V - W$ is stable. The *vertex cover number* $\tau(G)$ of G is the size of a smallest vertex cover for G. We denote the set of neighbors of a vertex set $V' \subseteq V$ by $\Gamma(V')$.

The Alcuin number of a graph is closely related to its vertex cover number.

Lemma 1. *(Prisner [13]; Bahls [2]; Lampis & Mitsou [9])*
Every graph G satisfies $\tau(G) \leq \text{ALCUIN}(G) \leq \tau(G) + 1$.

Indeed during the first boat trip of any feasible schedule, the man leaves a stable set L_1 on the left bank and transports a vertex cover B_1 with the boat. This implies $b \geq \tau(G)$. And it is straightforward to find a schedule for a boat of capacity $\tau(G) + 1$: The man permanently keeps a smallest vertex cover $W \subseteq V$ in the boat, and uses the remaining empty spot to transport the items in $V - W$ one by one to the other bank.

The following observation follows from the inherent symmetry in conditions (FS1)–(FS3).

Lemma 2. *If $(L_1, B_1, R_1), \ldots, (L_s, B_s, R_s)$ is a feasible schedule for a graph G and a boat of capacity b, then also (R_s, B_s, L_s), $(R_{s-1}, B_{s-1}, L_{s-1})$, $\ldots$, (R_1, B_1, L_1) is a feasible schedule.*

3 A Concise Characterization

The definition of a feasible schedule does not a priori imply that the decision problem *"Given a graph G and a bound A, is $\text{ALCUIN}(G) \leq A$?"* is contained in the class NP: Since the length s of the schedule need not be polynomially bounded in the size of the graph G, this definition does not give us any obvious NP-certificate. The following theorem yields such an NP-certificate.

Theorem 1. *(Structure theorem)*
A graph $G = (V, E)$ possesses a feasible schedule for a boat of capacity $b \geq 1$, if and only if there exist five subsets X_1, X_2, X_3, Y_1, Y_2 of V that satisfy the following four conditions.

(i) *The three sets X_1, X_2, X_3 are pairwise disjoint. Their union $X := X_1 \cup X_2 \cup X_3$ forms a stable set in G.*
(ii) *The (not necessarily disjoint) sets Y_1, Y_2 are non-empty subsets of the set $Y := V - X$, which satisfies $|Y| \leq b$.*
(iii) *$X_1 \cup Y_1$ and $X_2 \cup Y_2$ are stable sets in G.*
(iv) *$|Y_1| + |Y_2| \geq |X_3|$.*

If these four conditions are satisfied, then there exists a feasible schedule of length at most $2|V| + 1$. This bound $2|V| + 1$ is the best possible (for $|V| \geq 3$).

As an illustration for Theorem 1, we once again consider Alcuin's problem with $b = 1$; see Figure 1. The corresponding sets in conditions (i)–(iv) then are

$X_1 = X_2 = \emptyset$, $X_3 = \{w, c\}$, and $Y_1 = Y_2 = \{g\}$. The rest of this section is dedicated to the proof of Theorem 1.

For the (only if)-part, we consider a feasible schedule (L_k, B_k, R_k) with $1 \leq k \leq s$. Without loss of generality we assume that $B_{k+1} \neq B_k$ for $1 \leq k \leq s-1$. Lemma 1 yields that there exists a vertex cover $Y \subseteq V$ with $|Y| = b$ (which is not necessarily a vertex cover of minimum size). Then the set $X = V - Y$ is stable. We branch into three cases.

In the first case, there exists an index k for which $L_k \cap Y \neq \emptyset$ and $R_k \cap Y \neq \emptyset$. We set $Y_1 = L_k \cap Y$, $X_1 = L_k \cap X$, and $Y_2 = R_k \cap Y$, $X_2 = R_k \cap X$, and $X_3 = B_k \cap X$. This construction yields $X = X_1 \cup X_2 \cup X_3$, and obviously satisfies conditions (i), (ii), (iii). Since

$$|Y| \;=\; b \;\geq\; |B_k \cap X| + |B_k \cap Y| \;=\; |X_3| + (|Y| - |Y_1| - |Y_2|),$$

we also derive the inequality $|Y_1| + |Y_2| \geq |X_3|$ for condition (iv).

In the second case, there exists an index k with $1 < k < s$ such that $B_k = Y$. If index k is odd (and the boat is moving forward), our assumption $B_{k-1} \neq B_k \neq B_{k+1}$ implies that $L_{k-1} \cap Y \neq \emptyset$ and $R_{k+1} \cap Y \neq \emptyset$. We set $Y_1 = L_{k-1} \cap Y$, $X_1 = L_{k-1} \cap X$, and $Y_2 = R_{k+1} \cap Y$, $X_2 = R_{k+1} \cap X$, and $X_3 = (B_{k-1} \cup B_{k+1}) \cap X$. Then X_1, X_2, X_3 are pairwise disjoint, and conditions (i), (ii), (iii) are satisfied. Furthermore,

$$|Y| \;=\; b \;\geq\; |B_{k-1} \cap X| + |B_{k-1} \cap Y| \;=\; |B_{k-1} \cap X| + (|Y| - |Y_1|)$$

implies $|B_{k-1} \cap X| \leq |Y_1|$, and a symmetric argument yields $|B_{k+1} \cap X| \leq |Y_2|$. These two inequalities together imply $|Y_1| + |Y_2| \geq |X_3|$ for condition (iv). If the index k is even (and the boat is moving back), we proceed in a similar way with the roles of $k-1$ and $k+1$ exchanged.

The third case covers all remaining situations: All k satisfy $L_k \cap Y = \emptyset$ or $R_k \cap Y = \emptyset$, and all k with $1 < k < s$ satisfy $B_k \neq Y$. We consider two subcases. In subcase (a) we assume $R_s \cap Y \neq \emptyset$. We define $Y_1 = R_s \cap Y$ and $X_1 = R_s \cap X$, and we set $Y_2 = Y_1$, $X_2 = \emptyset$, and $X_3 = B_s \cap X$. Then conditions (i), (ii), (iii) are satisfied. Since

$$|Y| \;=\; b \;\geq\; |B_s \cap X| + |B_s \cap Y| \;=\; |X_3| + (|Y| - |Y_1|),$$

also condition (iv) holds. In subcase (b) we assume $R_s \cap Y = \emptyset$. We apply Lemma 2 to get a symmetric feasible schedule with $L_1 \cap Y = \emptyset$. We prove by induction that this new schedule satisfies $R_k \cap Y \neq \emptyset$ for all $k \geq 2$. First, $L_1 \cap Y = \emptyset$ implies $Y \subseteq B_1$, and then $B_2 \neq B_1$ implies $R_2 \cap Y \neq \emptyset$. In the induction step for $k \geq 3$ we have $R_{k-1} \cap Y \neq \emptyset$, and hence $L_{k-1} \cap Y = \emptyset$. If k is odd, then $R_k = R_{k-1}$ and we are done. If k is even, then $R_k \cap Y = \emptyset$ would imply $B_k = Y$, a contradiction. This completes the inductive argument. Since the new schedule has $R_s \cap Y \neq \emptyset$, we may proceed as in subcase (a). This completes the proof of the (only if)-part.

The proof of the (if)-part can be found in the full version of this paper.

4 Small Boats Versus Big Boats

By Lemma 1 every graph G has either $\text{ALCUIN}(G) = \tau(G)$ or $\text{ALCUIN}(G) = \tau(G)+1$. In the former case we call G a *small-boat* graph, and in the latter case we call G a *big-boat* graph. Note that for a small-boat graph G with $b = \tau(G)$, the stable set X in Theorem 1 is a maximum size stable set and set Y is a minimum size vertex cover.

The following three lemmas provide tools for recognizing small-boat graphs.

Lemma 3. *Let $G = (V, E)$ be a graph, and let set $C \subseteq V$ induce a subgraph of G with stability number at most 2. If the graph $G - C$ has at least two non-trivial connected components, then G is a small-boat graph.*

PROOF. Let $V_1 \subseteq V$ denote the vertex set of a non-trivial connected component of $G - C$, and let $V_2 = V - (V_1 \cup C)$ be the vertex set of all other components. Let X be a stable set of maximum size in G.

We set $X_1 = V_1 \cap X$, $X_2 = V_2 \cap X$, and $X_3 = C \cap X$; note that $X_1 \cup X_2 \cup X_3 = X$ and $|X_3| \leq 2$. Since V_1 and V_2 both induce edges, $V_1 - X$ and $V_2 - X$ are non-empty. We put a single vertex from $V_2 - X$ into Y_1, and a single vertex from $V_1 - X$ into Y_2. This satisfies all conditions of the Structure Theorem 1. □

Lemma 4. *Let $G = (V, E)$ be a graph with a minimum vertex cover Y and a maximum stable set $X = V - Y$. If Y contains two (not necessarily distinct) vertices u and v that have at most two common neighbors in X, then G is a small-boat graph.* □

Lemma 5. *Let $G = (V, E)$ be a graph that has two distinct stable sets $S_1, S_2 \subseteq V$ of maximum size (or equivalently: two distinct vertex covers of minimum size). Then G is a small-boat graph.* □

The following lemma allows us to generate a plethora of small-boat and big-boat graphs.

Lemma 6. *Let $G = (V, E)$ be a graph with $\alpha(G) = s$, let I be a stable set on $q \geq 1$ vertices that is disjoint from V, and let G' be the graph that results from G and I by connecting every vertex in V to every vertex in I.*

Then G' is a small-boat graph if $s/2 \leq q \leq 2s$, and a big-boat graph if $q \geq 2s + 1$. □

The following Corollary 1 follows from Lemma 6. It also illustrates that the statement of Lemma 6 cannot be extended in any meaningful way to the cases with $1 \leq q < s/2$: If we join the graph $G = K_{s,s}$ with stability number s to a stable set I on q vertices, then the resulting tri-partite graph $K_{q,s,s}$ is a small-boat graph. On the other hand, if we join the graph $G = K_{q,s}$ with stability number s to a stable set I on q vertices, then the resulting tri-partite graph $K_{q,q,s}$ is a big-boat graph.

Corollary 1. *Let $k \geq 2$ and $1 \leq n_1 \leq n_2 \leq \cdots \leq n_k$ be positive integers. Then the complete k-partite graph $K_{n_1,\ldots,n_k}$ is a small-boat graph if $n_k \leq 2n_{k-1}$, and it is a big-boat graph otherwise.*

The following observation is a consequence of Lemma 3 (with $C = \emptyset$) and the Structure Theorem 1. It allows us to concentrate our investigations on connected graphs.

Lemma 7. *A disconnected graph G with $k \geq 2$ connected components is a big-boat graph, if and only if $k-1$ components are isolated vertices, whereas the remaining component is a big-boat graph (which might be another isolated vertex).*

5 An Algorithmic Result

The following theorem demonstrates that determining the Alcuin number of a graph belongs to the class FPT of fixed-parameter tractable problems.

Theorem 2. *For a given graph G with n vertices and m edges and a given bound A, we can decide in $O(4^A mn)$ time whether* $\textsc{Alcuin}(G) \leq A$.

Proof. The proof can be found in the full version of this paper. □

6 Hardness Results

The reductions in this section are from the NP-hard Vertex Cover and from the NP-hard Stable Set problem; see Garey & Johnson [5]. Slightly weaker versions of the statements in Lemma 8 and 9, and also the restriction of Theorem 4 to three boat trips have been derived by Lampis & Mitsou [9].

The following observation implies that finding the Alcuin number is NP-hard for planar graphs and for graphs of bounded degree.

Lemma 8. *Let $\mathcal{G}$ be a graph class that is closed under taking disjoint unions. If the vertex cover problem is NP-hard for graphs in $\mathcal{G}$, then it is NP-hard to compute the Alcuin number for graphs in $\mathcal{G}$.*

Proof. For a graph $G \in \mathcal{G}$, we consider the disjoint union G' of two independent copies of G. Then $\tau(G') = 2\,\tau(G)$, and Lemma 1 yields $2\,\tau(G) \leq \textsc{Alcuin}(G') \leq 2\,\tau(G) + 1$. Hence, we can deduce the vertex cover number $\tau(G)$ from $\textsc{Alcuin}(G')$. □

The *approximability threshold* of a minimization problem $\mathcal{P}$ is the infimum of all real numbers $R \geq 1$ for which problem $\mathcal{P}$ possesses a polynomial time approximation algorithm with worst case ratio R. The approximability threshold of the vertex cover problem is known to lie somewhere between 1.36 and 2, and it is widely conjectured to be exactly 2; see for instance Khot & Regev [8].

Lemma 9. *The approximability threshold of the vertex cover problem coincides with the approximability threshold of the Alcuin number problem.*

Proof. First, we show that an approximation algorithm with worst case ratio R for Vertex Cover implies an approximation algorithm with worst case ratio R for the Alcuin number problem. For an input graph G, we call

the approximation algorithm for vertex cover and simply output its approximation of $\tau(G)$ as approximation A' of $\textsc{Alcuin}(G)$. Then Lemma 1 yields $A' \leq R \cdot \tau(G) \leq R \cdot \textsc{Alcuin}(G)$.

Secondly, we show that an approximation algorithm with worst case ratio R for the Alcuin number problem implies an approximation algorithm with worst case ratio $R+\varepsilon$ for the Vertex Cover, where $\varepsilon > 0$ can be brought arbitrarily close to 0. For an input graph G we first check whether $\tau(G) \leq R/\varepsilon$ holds. If it holds, then we compute the value $\tau(G)$ exactly in polynomial time; see Section 5. If it does not hold, then we call the approximation algorithm for the Alcuin number, and output its approximation of $\textsc{Alcuin}(G)$ as approximation τ' of $\tau(G)$. Then Lemma 1 yields $\tau' \leq R \cdot \textsc{Alcuin}(G) \leq R \cdot (\tau(G)+1) \leq (R+\varepsilon)\ \tau(G)$. □

Theorem 3. *It is NP-hard to decide whether a given graph is a small-boat graph.*

Proof. We show that if small-boat graphs can be recognized in polynomial time, then there existis a polynomial time algorithm for computing the stability number of a graph.

Indeed, consider a graph $G = (V, E)$ on $n = |V|$ vertices. For $q = 1, \ldots, 2n+1$, let I_q be a stable set on q vertices that is disjoint from V, and let G_q be the graph that results from G and I_q by connecting every vertex in V to every vertex in I_q. We check for every q whether G_q is small-boat, and we let q^* denote the largest index q for which G_q is small-boat. Lemma 6 yields that the stability number of G equals $q^*/2$. □

Since the Structure Theorem 1 produces feasible schedules of length at most $2|V|+1$, we have $\textsc{Alcuin}_t(G) = \textsc{Alcuin}(G)$ for all $t \geq 2|V|+1$. Consequently, computing the t-trip constrained Alcuin number is NP-hard, if t is part of the input. The following theorem shows that this problem is NP-hard for every fixed $t \geq 3$.

Theorem 4. *Let $r \geq 1$ be a fixed integer bound. Then it is NP-hard to decide for a given graph and a given boat capacity, whether there exists a feasible schedule that only uses $2r+1$ boat trips.*

Proof. The proof can be found in the full version of this paper. □

7 Special Graph Classes

In this section we discuss Alcuin number, small-boat graphs, and big-boat graphs in several classes of specially structured graphs.

7.1 Chordal Graphs and Trees

A *split graph* is a graph $G = (V, E)$ whose vertex set can be partitioned into a clique and a stable set; see Golumbic [6]. An equivalent characterization states

that a graph is a split graph, if and only if it does not contain C_4, C_5, and $2K_2$ (= two independent edges) as induced subgraphs. *Chordal graphs* are the graphs in which every cycle of length exceeding three has a chord, that is, an edge joining two non-consecutive vertices in the cycle; see Golumbic [6]. An equivalent characterization states that a graph is chordal, if and only if every minimal vertex separator induces a clique. Note that split graphs and trees are special cases of chordal graphs.

The following lemma provides a complete charaterization of chordal small-boat graphs.

Lemma 10. *Let $G = (V, E)$ be a connected chordal graph. Then G is a small-boat graph, if and only if one of the following holds:*

(1) G is a split graph with a maximum stable set X and a clique $Y = V - X$, such that there exist two (not necessarily distinct) vertices u, v in Y that have at most two common neighbors in X.
(2) G is not a split graph. □

As a special case, Lemma 10 contains the following classification of trees (which has already been derived in [2,9]). Stars $K_{1,k}$ with $k \geq 3$ leaves are split graphs that do not satisfy condition (1) of Lemma 10; therefore they are big-boat graphs (note that this also follows from Lemma 6). All remaining trees T are small-boat graphs: Either such a tree T has two independent edges (and thus is small-boat), or it is of the following form: There are vertices $a_0, \ldots, a_k$ and $b_0, \ldots, b_\ell$ with $k, \ell \geq 0$, and edges $[a_0, a_i]$ for all $i > 0$, and edges $[b_0, b_j]$ for all $j > 0$. Then T is a split graph with clique $\{a_0, b_0\}$ that satisfies condition (1); hence T is small-boat.

7.2 Bipartite Graphs

The proof of the following theorem is centered around submodular functions. We recall that a function $f : 2^X \to \mathbb{R}$ over a set X is *submodular*, if $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$ holds for all $A, B \subseteq X$; see for instance Grötschel, Lovász & Schrijver [7] or Schrijver [14]. Standard examples of submodular functions are $f(A) = c\,|A|$ for any real parameter c, and the function $f(A) = |\Gamma(A)|$ that assigns to a subset $A \subseteq V$ of vertices the number of neighbors in an underlying graph. If $f(A)$ is submodular, then also $f_{\min}(A) = \min\{f(B) \,|\, B \subseteq A\}$ and $f'(A) = f(X - A)$ are submodular. Also the sum of two submodular functions is submodular. The minimum of a submodular function f can be determined in polynomial time [7,14].

Theorem 5. *For a bipartite graph $G = (V, E)$, the Alcuin number can be computed in polynomial time.*

Proof. It is well-known that the stability number and the vertex cover number of a bipartite graph G can be computed in polynomial time; see for instance Lovász & Plummer [10]. Hence it is also easy to decide whether G has a unique

maximum size stable set (for instance, by finding some maximum size stable set X, and by checking for every $x \in X$ whether $G - x$ has a stable set of cardinality $|X|$). If G possesses two distinct maximum size stable sets, then Lemma 5 yields $\textsc{Alcuin}(G) = \tau(G)$. Hence, in the light of Theorem 1 the only interesting situation is the following: The graph G has a unique maximum size stable set X and a unique minimum size vertex cover $Y = V - X$. Do there exist sets X_1, X_2, X_3 and Y_1, Y_2 that satisfy conditions (i)–(iv)?

Let $V = V_1 \cup V_2$ denote a bipartition of V with $E \subseteq V_1 \times V_2$. If $Y \cap V_1 \neq \emptyset$ and $Y \cap V_2 \neq \emptyset$, then we may choose $X_1 = X \cap V_1$, $X_2 = X \cap V_2$, $X_3 = \emptyset$, and $Y_1 = Y \cap V_1$, $Y_2 = Y \cap V_2$. Otherwise $Y \subseteq V_1$ or $Y \subseteq V_2$ holds, and Y is also stable. Hence, we may concentrate on the case where $X = V_1$ and $Y = V_2$ form the bipartition. Our problem boils down to identifying the two disjoint sets X_1 and X_2: Then $X_3 = X - (X_1 \cup X_2)$ is fixed. By condition (iv), Y_1 should be chosen as large as possible and hence should be equal to $Y - \Gamma(X_1)$; symmetrically we set $Y_2 = Y - \Gamma(X_2)$. Condition (iv) can now be rewritten into

$$|\Gamma(X_1)| - |X_1| \; + \; |\Gamma(X_2)| - |X_2| \; \leq \; 2|Y| - |X|.$$

We define a function $f : 2^X \to \mathbb{R}$ by $f(X_1) = |\Gamma(X_1)| - |X_1|$, and a function $g : 2^X \to \mathbb{R}$ by $g(X_1) = \min\{f(X_2) \,|\, X_2 \subseteq X - X_1\}$. Since functions f, g, and their sum $f + g$ are submodular, the minimum of $f + g$ can be determined in polynomial time. If the corresponding minimum value is at most $2|Y| - |X|$, then $\textsc{Alcuin}(G) = \tau(G)$. Otherwise $\textsc{Alcuin}(G) = \tau(G) + 1$. □

7.3 Planar Graphs

Next, let us turn to planar and outer-planar graphs. Outer-planar graphs are easy to classify: Any outer-planar graph G with $\tau(G) = 1$ is a star, and hence a small-boat if and only if it has at most two leaves; see Section 7.1. Any outer-planar graph G with $\tau(G) \geq 2$ satisfies the conditions of Lemma 4 and thus is small-boat: Two arbitrary vertices u and v in a minimum vertex cover cannot have more than two common neighbors, since otherwise $K_{2,3}$ would occur as a subgraph. The behavior of general planar graphs is more interesting.

Lemma 11. *Every planar graph $G = (V, E)$ with $\tau(G) \geq 5$ is a small-boat graph.*

Proof. Let $Y = \{y_1, \ldots, y_t\}$ with $t \geq 5$ be a vertex cover of minimum size, and let $X = V - Y$ denote the corresponding stable set. For $y \in Y$ we denote by $\Gamma_x(y)$ the set of neighbors of y in X. If there exist two indices i, j with $1 \leq i < j \leq 5$ such that $\Gamma_x(y_i) \cap \Gamma_x(y_j)$ contains at most two vertices, then G is small-boat by Lemma 4. We will show that no other case can arise.

Suppose for the sake of contradiction that for every two indices i, j with $1 \leq i < j \leq 5$, the set $\Gamma_x(y_i) \cap \Gamma_x(y_j)$ contains at least three vertices. Then let a, b, c be three vertices in $\Gamma_x(y_1) \cap \Gamma_x(y_2)$. In any planar embedding of G the three paths $y_1 - a - y_2$, $y_1 - b - y_2$, $y_1 - c - y_2$ divide the plane into three regions. If two of y_3, y_4, y_5 would lie in different regions, they could not have three common

neighbors; a contradiction. Hence y_3, y_4, y_5 all lie in the same region, say in the region bounded by $y_1 - a - y_2 - b - y_1$, and hence vertex c is not a neighbor of y_3, y_4, y_5. An analogous argument yields that for any $1 \leq i < j \leq 5$, the two vertices y_i and y_j have a common neighbor that is not adjacent to the other three vertices in $\{y_1, y_2, y_3, y_4, y_5\}$. This yields that G contains a subdivision of K_5, the desired contradiction. □

The condition $\tau(G) \geq 5$ in Lemma 11 cannot be dropped, since there exists a variety of planar graphs G with $\tau(G) \leq 4$ that are big-boat. Consider for instance the following planar graph G: The vertex set contains four vertices y_1, y_2, y_3, y_4, and for every i, j with $1 \leq i < j \leq 4$ a set V_{ij} of $t \geq 3$ vertices. The edge set connects every vertex in V_{ij} to y_i and to y_j. It can be verified that G is planar, that $\tau(G) = 4$, and that $\textsc{Alcuin}(G) = 5$.

Lemma 11 implies that there is a polynomial time algorithm that decides whether a planar graph G is small-boat or big-boat: In case G has a vertex cover of size at most 4 we use Theorem 2 to decide whether $\textsc{Alcuin}(G) = \tau(G)$, and in case G has vertex cover number at least 5 we simply answer YES.

Summarizing, this yields the following (perhaps unexpected) situation: Although it is NP-hard to compute the Alcuin number and the vertex cover number of a planar graph, we can determine in polynomial time whether these two values coincide.

8 Conclusions

In this paper we have derived a variety of combinatorial, structural, algorithmical, and complexity theoretical results around a graph-theoretic generalization of Alcuin's river crossing problem.

Our investigations essentially revolved around three algorithmic problems: (1) Computation of the stability number; (2) Computation of the Alcuin number; (3) Recognition of small-boat graphs. All three problems are polynomially solvable, if the input graph has bounded treewidth (the Alcuin number can be computed along the lines of the standard dynamic programming approach).

Question 1. Does there exist a graph class $\mathcal{G}$, for which computing the stability number is easy, whereas computing the Alcuin number is hard?

In particular, the case of perfect graphs remains open. A graph is *perfect*, if for every induced subgraph the clique number coincides with the chromatic number; see for instance Golumbic [6]. Trees, split graphs, and chordal graphs are special cases of perfect graphs.

Question 2. Is there a polynomial time algorithm for computing the Alcuin number of a perfect graph?

Also the computational complexity of recognizing small-boat graphs remains unclear.

Question 3. Is the problem of recognizing small-boat graphs contained in NP?

We have proved that this problem is NP-hard, but there is no reason to assume that it lies in NP: To demonstrate that a graph is small-boat in a straightforward way, we have to show that its Alcuin number is small (NP-certificate) and that its vertex cover number is large (coNP-certificate). This mixture of NP- and coNP-certificates suggests that the problem might be located in one of the complexity classes above NP (see for instance Chapter 17 in Papadimitriou's book [12]); the complexity class DP might be a reasonable guess.

Acknowledgement. This research has been supported by the Netherlands Organisation for Scientific Research (NWO), grant 639.033.403; by DIAMANT (an NWO mathematics cluster); and by BSIK grant 03018 (BRICKS: Basic Research in Informatics for Creating the Knowledge Society).

References

1. Ascher, M.: A river-crossing problem in cross-cultural perspective. Mathematics Magazine 63, 26–29 (1990)
2. Bahls, P.: The wolf, the goat, and the cabbage: A modern twist on a classical problem. University of North Carolina Asheville (unpublished manuscript, 2005)
3. Biggs, N.L.: The roots of combinatorics. Historia Mathematica 6, 109–136 (1979)
4. Borndörfer, R., Grötschel, M., Löbel, A.: Alcuin's transportation problems and integer programming. In: Charlemagne and his heritage. 1200 years of civilization and science in Europe, Brepols, Turnhout, vol. 2, pp. 379–409 (1998)
5. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, San Francisco (1979)
6. Golumbic, M.C.: Algorithmic Graph Theory and Perfect Graphs. Academic Press, New York (1980)
7. Grötschel, M., Lovász, L., Schrijver, A.: Geometric algorithms and combinatorial optimization. Springer, New York (1988)
8. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within $2-\varepsilon$. Journal of Computer and System Sciences 74, 335–349 (2008)
9. Lampis, M., Mitsou, V.: The ferry cover problem. In: Crescenzi, P., Prencipe, G., Pucci, G. (eds.) FUN 2007. LNCS, vol. 4475, pp. 227–239. Springer, Heidelberg (2007)
10. Lovász, L., Plummer, M.D.: Matching Theory. Annals of Discrete Mathematics, vol. 29. North-Holland, Amsterdam (1986)
11. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford University Press, New York (2006)
12. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley, Reading (1994)
13. Prisner, E.: Generalizing the wolf-goat-cabbage problem. Electronic Notes in Discrete Mathematics 27, 83 (2006)
14. Schrijver, A.: A combinatorial algorithm minimizing submodular functions in strongly polynomial time. Journal of Combinatorial Theory, Series B 80, 346–355 (2000)