SIAM Review
Vol. 54, No. 1, pp. 141–154

The Alcuin Number of a Graph and Its Connections to the Vertex Cover Number*

Péter Csorba†
Cor A. J. Hurkens†
Gerhard J. Woeginger†

Abstract. We consider a planning problem that generalizes Alcuin's river crossing problem to scenarios with arbitrary conflict graphs. This generalization leads to the so-called Alcuin number of the underlying conflict graph. We derive a variety of combinatorial, structural, algorithmical, and complexity theoretical results around the Alcuin number. Our technical main result is an NP-certificate for the Alcuin number. It turns out that the Alcuin number of a graph is closely related to the size of a minimum vertex cover in the graph, and we unravel several surprising connections between these two graph parameters. We provide hardness results and a fixed parameter tractability result for computing the Alcuin number. Furthermore we demonstrate that the Alcuin number of chordal graphs, bipartite graphs, and planar graphs is substantially easier to analyze than the Alcuin number of general graphs.

Key words. transportation problem, scheduling and planning, graph theory, vertex cover

AMS subject classifications. 90B06, 90B35, 90C27

DOI. 10.1137/110848840

1. Introduction.

Reachability Problems. Consider an abstract discrete system that consists of a set S of states and a transition relation $R \subseteq S \times S$. The transition relation determines the possible behavior of the system over time: if $(s, s') \in R$, then the system may move directly from state s to state s'. We say that a goal state t is *reachable* from an initial state s if there exists a finite sequence $s = s_1, s_2, \ldots, s_n = t$ of states with $(s_i, s_{i+1}) \in R$ for $1 \leq i \leq n-1$. Discrete applied mathematics is crowded with reachability questions. Here are some examples that fit into this framework:

- When a *computer program* starts running on a laptop, the memory of the laptop traverses a number of states. In the ideal case the program will terminate and thereby bring the laptop into the desired goal state.
- A *railroad shunting yard* consists of many tracks and switches. In the evening freight trains and passenger trains arrive at the yard in a certain order (which

*Published electronically February 8, 2012. This paper originally appeared in *SIAM Journal on Discrete Mathematics*, Volume 24, Number 3, 2010, pages 757–769. This research was supported by 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).
http://www.siam.org/journals/sirev/54-1/84884.html

†Department of Mathematics and Computer Science, TU Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands (p.csorba@tue.nl, wscor@win.tue.nl, gwoegi@win.tue.nl).

is schedule dependent). The trains are then decomposed and their cars are rearranged into other trains that leave the yard in the morning. The shunting process transforms the evening state of the yard into the morning state, and it should do this quickly and with a small number of moves while obeying all the restrictions arising from the local infrastructure.

- The famous *15-puzzle* of Sam Loyd is a sliding puzzle that consists of a 4×4 frame of square tiles numbered $1, \ldots, 15$ in random order with one tile missing. The objective is to place the tiles in order by making sliding moves that use the empty space.
- In manufacturing, an *assembly line* produces a final product by compiling parts in a sequential manner. The process starts with an empty product and runs through various states until the product is completed.

A fundamental problem in discrete systems is to decide whether a given goal state is reachable from a given initial state. Some of these reachability problems are relatively easy to solve, as they belong to the class P of polynomially solvable problems. Some of these reachability problems are difficult and belong to the class of NP-complete problems. And many of these reachability problems are extremely difficult and belong to the class of PSPACE-complete problems (Papadimitriou [11]).

In this paper we will study a fairly general reachability problem from the area of transportation planning. The considered problem looks computationally difficult at first sight, but a closer analysis of its combinatorial structure reveals that it actually allows short solution schedules of polynomially bounded length. Our results show that the problem falls into the complexity class NP.

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, developed the Carolingian minuscule (a script which has become the basis of the way the letters of the present Roman alphabet are written), and 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 himself and one of them. For that reason he sought a plan which would enable them all to get to the far side unhurt. Let he 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 medieval 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, and 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 E if they are *conflicting* and thus cannot be left 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 conditions (FS1)–(FS3) below. 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 redivide the set of items that currently are on that bank and in 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 $[g, c]$. Table 1.1 lists one possible feasible schedule for a boat of capacity $b = 1$. This implies $\textsc{Alcuin}(P_3) = 1$.

Table 1.1 *A solution for Alcuin's river crossing puzzle. The partitions L_k, B_k, R_k are listed as $L_k \,|\, B_k \,|\, R_k$; the arrows $\rightarrow$ and $\leftarrow$ indicate the current direction of 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.	$\emptyset \mid g \rightarrow \mid w, c$		

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 Table 1.1, 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 [12] 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 during the mathematics seminar at the University of North Carolina.

Bahls [2] (and later Lampis and Mitsou [8]) observed that it is NP-hard to compute the Alcuin number exactly; Lampis and Mitsou [8] 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 Observation 2.1. The papers [2, 8] provide a complete analysis of the Alcuin number of trees. Finally, Lampis and Mitsou [8] proved that the computation of the trip constrained Alcuin number $\textsc{Alcuin}_3(G)$ is NP-hard.

New Results. We derive a variety of combinatorial and algorithmical results around the Alcuin number of a graph. As a by-product, our results settle several open questions from [8] and also raise a number of new open problems.

Our main result is a structural characterization of the Alcuin number (as presented in section 3). This characterization yields an NP-certificate for the Alcuin number. We also derive that every feasible schedule (possibly of exponential length) can be transformed into a feasible schedule of (linear) length at most $2|V| + 1$, and that this bound $2|V| + 1$ is the strongest possible bound.

Computing the Alcuin number of a graph is NP-hard. Several proofs for this result are already in the literature [2, 8]. We provide a new proof for this (section 6), and we firmly believe that our three-line argument is considerably simpler than all previously published arguments. Section 6 also shows that computing the t-trip constrained Alcuin number $\textsc{Alcuin}_t(G)$ is NP-hard for every fixed value $t \geq 3$; in fact, this problem is NP-hard even for planar graphs. Furthermore we establish that approximating the Alcuin number is exactly as hard as approximating the vertex cover number. On the positive side we show that the Alcuin number of a *bipartite* graph can be determined in polynomial time (section 7.2). Standard techniques yield that computing the Alcuin number belongs to the class FPT of fixed-parameter tractable problems (section 5).

The close relationship between the Alcuin number and the vertex cover number of a graph (see Observation 2.1) naturally divides graphs into so-called *small-boat* and *large-boat* graphs: A graph is small-boat if its Alcuin number and vertex cover number coincide, and otherwise it is large-boat. We derive a number of combinatorial lemmas around the division line between these two classes (section 4); all of these lemmas fall out quite easily from our structural characterization theorem. Furthermore we establish the NP-hardness of distinguishing small-boat graphs from large-boat graphs (section 6). This general hardness result does not carry over to the more restricted classes of chordal graphs, bipartite graphs, and planar graphs (section 7) for which we give concise descriptions of the division line between small-boat and large-boat graphs. Although it is NP-hard to compute the Alcuin number and the vertex cover number of a planar graph, one can determine in polynomial time whether these two numbers are equal.

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.

Observation 2.1 (Prisner [12]; Bahls [2]; Lampis and Mitsou [8]). *Every graph G satisfies $\tau(G) \leq \textsc{Alcuin}(G) \leq \tau(G) + 1$.*

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).

OBSERVATION 2.2. *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 (R_s, B_s, L_s), $(R_{s-1}, B_{s-1}, L_{s-1}), \ldots, (R_1, B_1, L_1)$ is also 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* 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 3.1 (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 nonempty 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 3.1, we once again consider Alcuin's problem with $b = 1$; see Table 1.1. 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 3.1.

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 $1 \leq k \leq s - 1$. Observation 2.1 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 $X_3 = B_k \cap X$; note that Y_1 and Y_2 are disjoint. 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 $R_{k+1} \cap Y \neq \emptyset$. Furthermore, every element of X is contained either in $L_{k-1} \cup B_{k-1}$ or in $R_{k+1} \cup B_{k+1}$ (but not in both sets). 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

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

condition (iv) also holds. In subcase (b) we assume $R_s \cap Y = \emptyset$. We apply Observation 2.2 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.

For the *if* part, we construct a schedule that goes through several phases. We use the notation $L \mid B \mid R$ to denote a snapshot situation with item set L on the left bank, set B on the boat, and set R on the right bank.

(1) By condition (ii), the boat can carry set Y. We leave X on the left bank, put Y into the boat, drop off Y_1 on the right bank, and return to the left bank. This yields situation $X \mid Y - Y_1 \mid Y_1$.

(2) The boat now has at least $|Y_1| \geq 1$ empty places. We cut X_1 into packages of size at most $|Y_1|$, which we take to the right bank. Eventually this yields $X_2, X_3 \mid Y - Y_1 \mid X_1, Y_1$.

(3) Condition (iv) allows us to split X_3 into two disjoint subsets X_{31} and X_{32} with $|X_{31}| \leq |Y_1|$ and $|X_{32}| \leq |Y_2|$. Starting from the left bank, we make four trips:

$$X_2, X_{32} \mid Y - Y_1, X_{31} \mid X_1, Y_1, \qquad X_2, X_{32} \mid Y \mid X_1, X_{31},$$
$$X_2, Y_2 \mid Y - Y_2, X_{32} \mid X_1, X_{31}, \qquad X_2, Y_2 \mid Y - Y_2 \mid X_1, X_3.$$

(4) The boat now has at least $|Y_2| \geq 1$ empty places, which we use to transport X_2 to the right bank. Eventually this yields $Y_2 \mid Y - Y_2 \mid X$.

(5) In the last trip, we pick up Y_2 from the left bank and reach $\emptyset \mid Y \mid X$.

Conditions (i)–(iv) guarantee that the resulting schedule indeed is feasible.

What about the length of this schedule? In phase (1), (2), (3), (4), and (5) we, respectively, make 2, $2\lceil |X_1|/|Y_1| \rceil$, 4, $2\lceil |X_2|/|Y_2| \rceil$, and 1 boat trips. Since $|Y_1|, |Y_2| \geq 1$ and since $|V| \geq |X_1| + |X_2| + |X_3| + 1$, this yields a total number of at most $2|V| - 2|X_3| + 5$ trips.

- If $|X_3| \geq 2$, then this bound is less than or equal to $2|V| + 1$.
- If $|X_3| = 1$, then we change the last backward trip in phase (2) to $X_2, X_3 \mid Y \mid X_1$ and replace phase (3) by the following:

  $$X_2, Y_2 \mid Y - Y_2, X_3 \mid X_1 \quad \text{and} \quad X_2, Y_2 \mid Y - Y_2 \mid X_1, X_3.$$

  Since this saves us two trips, the schedule length is at most $2|V| + 1$.
- If $|X_3| = 0$, then we change the last backward trip in phase (2) to $X_2 \mid Y \mid X_1$, remove phase (3) altogether, and in the first forward trip of phase (4)

leave Y_2 on the left bank. Since this saves us four trips, the schedule length again is bounded by $2|V|+1$.

Summarizing, in all cases we have found a schedule of length at most $2|V|+1$. This bound $2|V|+1$ is the best possible, since it can be shown that for the following graph (V,E) and for a boat of capacity 1, all feasible schedules have length at least $2|V|+1$: The vertex set V consists of vertices $v_1,\ldots,v_n$, and the edge set E consists of two edges $[v_1,v_2]$ and $[v_2,v_3]$. (A closer analysis reveals that these are actually the only graphs for which all feasible schedules have length at least $2|V|+1$.) This completes the proof of the structure theorem, Theorem 3.1. □

4. Small Boats versus Large Boats. By Observation 2.1 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 *large-boat* graph. Note that for a small-boat graph G with $b=\tau(G)$, the stable set X in Theorem 3.1 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 4.1. *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 nontrivial connected components, then G is a small-boat graph.*

Proof. Let $V_1\subseteq V$ denote the vertex set of a nontrivial 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 $|X_3|\le 2$. Since V_1 and V_2 both induce edges, V_1-X and V_2-X are nonempty. 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, Theorem 3.1. □

LEMMA 4.2. *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.*

Proof. For $y\in Y$, we let $\Gamma_x(y)$ denote the set of neighbors of y in X. We apply Theorem 3.1. We let $X_1=X-\Gamma_x(u)$, $X_2=\Gamma_x(u)-\Gamma_x(v)$, and $X_3=\Gamma_x(u)\cap\Gamma_x(v)$, and we let $Y_1=\{u\}$ and $Y_2=\{v\}$. Then $|Y_1|+|Y_2|=2\ge|X_3|$, and also all other conditions in Theorem 3.1 are satisfied. □

LEMMA 4.3. *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.*

Proof. We apply Theorem 3.1. We set $X_1=S_1\cap S_2$, $X_2=\emptyset$, and $X_3=S_1-S_2$, which yields $X=S_1$, and we set $Y_1=Y_2=S_2-S_1$. Then by condition (iv) any boat of capacity $b\ge|Y|=\tau(G)$ allows a feasible schedule. □

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

LEMMA 4.4. *Let $G=(V,E)$ be a graph with $\alpha(G)=s$, let I be a stable set on $q\ge 1$ 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 $s/2\le q\le 2s$, and a large-boat graph if $q\ge 2s+1$.

Proof. First, consider the case $s/2\le q\le s-1$. If G (and hence G') contains two distinct maximum stable sets, then G' is a small-boat graph by Lemma 4.3. If G contains a unique maximum stable set S, then S is also the unique maximum stable set in G'. We choose $X_1=X_2=\emptyset$, $X_3=S$, and $Y_1=Y_2=I$. Then $|Y_1|+|Y_2|=2q\ge s=|S|$, and G' with $b=\tau(G')$ satisfies conditions (i)–(iv) in the structure theorem, Theorem 3.1.

In the second case, $q = s$, the graph G' contains two distinct maximum stable sets and is a small-boat graph by Lemma 4.3.

In the third case, $q \geq s+1$, the set I is the unique maximum stable set in G', and V is the unique minimum vertex cover in G'. Hence $\tau(G') = |V|$. Furthermore let S with $|S| = s$ denote a maximum stable set in G. We apply Theorem 3.1. If $s+1 \leq q \leq 2s$, we set $X_1 = X_2 = \emptyset$ and $X_3 = I$, and $Y_1 = Y_2 = S$. Then $Y = V$, and $|Y_1| + |Y_2| = 2s \geq q = |I|$. Since conditions (i)–(iv) are satisfied for G' with $b = \tau(G')$, the graph G' indeed is small-boat. If $q \geq 2s+1$ holds, we suppose for the sake of contradiction that G' with $b = \tau(G')$ satisfies conditions (i)–(iv). Then $X_1 \cup X_2 \cup X_3 = I$ by condition (i), Y_1 and Y_2 are nonempty by condition (ii), and $X_1 \cup Y_1$ and $X_2 \cup Y_2$ are stable sets by condition (iii). Since $X_1 \cup Y_1$ is stable, and since every vertex in $X_1 \subseteq I$ is connected to every vertex in $Y_1 \subseteq V$, and since Y_1 is nonempty, we conclude that $X_1 = \emptyset$. An analogous argument yields $X_2 = \emptyset$, and hence $X_3 = I$. Since $|Y_1| + |Y_2| \geq |X_3| \geq 2s+1$ by condition (iv), we get $|Y_1| \geq s+1$ or $|Y_2| \geq s+1$. Therefore G contains a stable set on at least $s+1$ vertices, which contradicts $\alpha(G) = s$. □

Corollary 4.5 below follows from Lemma 4.4. It also illustrates that the statement of Lemma 4.4 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 tripartite 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 tripartite graph $K_{q,q,s}$ is a large-boat graph.

COROLLARY 4.5. *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 large-boat graph otherwise.*

The following observation is a consequence of Lemma 4.1 (with $C = \emptyset$) and the structure theorem, Theorem 3.1. It allows us to concentrate our investigations on connected graphs.

OBSERVATION 4.1. *A disconnected graph G with $k \geq 2$ connected components is a large-boat graph if and only if $k-1$ components are isolated vertices, whereas the remaining component is a large-boat graph.*

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 5.1. *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* ALCUIN$(G) \leq A$.

Proof. Our main tool is the standard FPT search-tree algorithm for vertex cover, which yields an $O(2^B mn)$ solution to the question *"Given a graph G with n vertices and m edges and a bound B, is $\tau(G) \leq B$?"*; see, for instance, Niedermeier [10]. In fact, if the answer is positive, then the search-tree algorithm can be used to enumerate all minimum size vertex covers in $O(2^B mn)$ time.

We proceed as follows. We first check whether $\tau(G) \leq A-1$: If the answer is positive, then Observation 2.1 allows us to stop with the output YES and ALCUIN$(G) \leq A$. If the answer is negative, then $\tau(G) \geq A$ holds and we move on. We check whether $\tau(G) \leq A$: If the answer is negative, then Observation 2.1 allows us to stop with NO and ALCUIN$(G) \not\leq A$. If the answer is positive, then $\tau(G) = A$ holds and we move on. We check whether G possesses two distinct minimum size vertex covers. If it does, then Lemma 4.3 yields ALCUIN$(G) = \tau(G) = A$ and we stop with output YES.

In the only remaining case, the graph $G = (V, E)$ has a unique vertex cover Y of size $A = \tau(G)$, and the set $X = V - Y$ is the unique maximum size stable set

in G. This uniquely determines sets X and Y in conditions (i)–(iv) of the structure theorem, Theorem 3.1, and it remains to find appropriate sets X_1, X_2, X_3 and Y_1, Y_2. We distinguish $O(4^A)$ subcases by considering all possibilities for two nonempty, stable subsets $Y_1, Y_2 \subseteq Y$. It is not hard to see that X_1 can be chosen as the set of all vertices in X that are not adjacent to vertices in Y_1, that X_2 can be chosen as the set of all vertices in $X - X_1$ that are not adjacent to vertices in Y_2, and that $X_3 = X - (X_1 \cup X_2)$. We output YES if in any of these $O(4^A)$ subcases the sets Y_1, Y_2, X_3 satisfy condition (iv), and otherwise we output NO. □

6. Hardness Results. The reductions in this section are from the NP-hard Vertex Cover and NP-hard Stable Set problems; see Garey and Johnson [5]. Slightly weaker versions of the statements in Observations 6.1 and 6.2, and also the restriction of Theorem 6.2 to three boat trips, have been derived by Lampis and Mitsou [8].

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

Observation 6.1. *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 Observation 2.1 yields $2\,\tau(G) \leq \text{Alcuin}(G') \leq 2\,\tau(G) + 1$. Hence, we can deduce the vertex cover number $\tau(G)$ from $\text{Alcuin}(G')$. □

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 and Regev [7].

Observation 6.2. *The approximability threshold of the vertex cover problem coincides with the approximability threshold of the Alcuin number problem.*

Proof. We show that an approximation algorithm with worst-case ratio R for one of the two values implies an approximation algorithm with worst-case ratio $R + \varepsilon$ for the other value, where $\varepsilon > 0$ can be made arbitrarily close to 0.

First, consider an approximation algorithm with worst-case ratio R for Vertex Cover. For an input graph G we first check whether $\tau(G) \leq 1/\varepsilon$ holds. If it holds, then we compute the value $\text{Alcuin}(G)$ exactly in polynomial time; see section 5. If it does not hold, then we call the approximation algorithm for vertex cover to compute an approximation τ' of $\tau(G)$, and output $\tau' + 1$ as an approximation of $\text{Alcuin}(G)$. Then $\tau' + 1 \geq \text{Alcuin}(G)$, and Observation 2.1 yields $\tau' + 1 \leq (R + \varepsilon) \cdot \tau(G) \leq (R + \varepsilon) \cdot \text{Alcuin}(G)$.

Second, consider an approximation algorithm with worst-case ratio R for the Alcuin number. 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 A' of $\text{Alcuin}(G)$ as an approximation of $\tau(G)$. Then $A' \geq \tau(G)$, and Observation 2.1 yields $A' \leq R \cdot \text{Alcuin}(G) \leq R \cdot (\tau(G) + 1) \leq (R + \varepsilon)\tau(G)$. □

Theorem 6.1. *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 exists 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 4.4 yields that the stability number of G equals $q^*/2$. □

Since the structure theorem, Theorem 3.1, produces feasible schedules of length at most $2|V| + 1$, we have $\text{ALCUIN}_t(G) = \text{ALCUIN}(G)$ 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 $t \geq 3$, even in the case in which the input graph is planar.

THEOREM 6.2. *Let $r \geq 1$ be a fixed integer bound. Then it is NP-hard to decide for a given planar graph and a given boat capacity whether there exists a feasible schedule that uses only $2r + 1$ boat trips.*

Proof. Consider an instance of the PLANAR STABLE SET problem: Given a planar graph $G = (V, E)$ and an integer bound q, does G possess a stable set of size at least q? Let $n = |V|$ and $m = |E|$, and construct the following new graph $G' = (V', E')$:

- The vertices in G' are the n vertices in V together with a set U of m new vertices, and together with a set W of $(m + q)r$ new vertices. For every edge $e \in E$, there is a corresponding vertex $u(e) \in U$.
- The edge set E' contains every edge in E. Furthermore for every edge $e = [v_1, v_2] \in E$, the new vertex $u(e)$ is made adjacent to vertices v_1 and v_2. All vertices in W are of degree zero.

We first establish that graph G' satisfies the following technical statement: If $S_1, S_2 \subseteq V \cup U$ with $S_1 \cap S_2 = \emptyset$ are two stable sets in G', then $|S_1| + |S_2| \leq m + \alpha(G)$. Indeed, consider such a pair of disjoint stable sets S_1 and S_2 that maximizes the value $|S_1| + |S_2|$, and among all such pairs consider one that maximizes the cardinality of $(S_1 \cup S_2) \cap U$. Consider a vertex $u(e)$ corresponding to some edge $e = [v_1, v_2] \in E$, and note that $u(e), v_1, v_2$ form a triangle in G'. If $u(e)$ is not in $S_1 \cup S_2$, then we may assume $v_1 \in S_1$ and $v_2 \in S_2$. Then replacing v_1 in S_1 by $u(e)$ leaves the cardinalities of S_1 and S_2 unchanged, but increases the cardinality of $(S_1 \cup S_2) \cap U$, a contradiction. This implies $U \subseteq S_1 \cup S_2$. Now consider an edge $e = [v_1, v_2] \in E$. If $v_1 \in S_1$, then $u(e) \in S_2$, and $v_2 \notin S_1 \cup S_2$. Symmetrically, if $v_1 \in S_2$, then $v_2 \notin S_1 \cup S_2$. This implies that $(S_1 \cup S_2) \cap V$ forms a stable set in G. The resulting inequality $|(S_1 \cup S_2) \cap V| \leq \alpha(G)$, together with $|(S_1 \cup S_2) \cap U| = m$, completes the proof of the technical statement.

Furthermore, it is easily seen that graph G being planar implies that graph G' is also planar. We claim that the original graph G has a stable set of size at least q if and only if there exists a schedule with $2r + 1$ boat trips for G' and a boat of capacity $b = n + m$.

First, assume that G has a stable set $S \subseteq V$ of size at least q. Partition the set W into $r + 1$ disjoint sets $W_1, \ldots, W_{r+1}$, such that $|W_1| = q$, $|W_2| = |W_3| = \cdots = |W_r| = m + q$, and $|W_{r+1}| = m$. It is easily verified that the following constitutes a feasible schedule:

(1) In the first boat trip put $V - S$, U, and W_1 into the boat, while leaving the stable set $S \cup (W - W_1)$ behind. Drop $U \cup W_1$ on the right bank, and in the second trip return with $V - S$ to the left bank.
(2) Use the next $2(r - 1)$ boat trips to transport the sets $W_2, \ldots, W_r$ to the right bank. The set $V - S$ remains on board all the time and blocks $n - q$ spots. The remaining $m + q$ spots leave just enough room for one set W_i per trip.

(3) In the $(2r+1)$th trip finally pick up W_{r+1} and S, and take all remaining vertices to the right bank.

Next, assume that every stable set in G has size at most $q-1$. Consider an arbitrary feasible schedule (L_k, B_k, R_k) for graph G' and a boat of capacity $b = n+m$, and denote the length of this schedule by $2s+1$. Our technical statement implies for any k that the set $L_k \cup R_k$ contains at most $m+q-1$ vertices from $V \cup U$. Consequently every set B_k contains at least $n-q+1$ of the $n+m$ vertices in $V \cup U$, and at most $m+q-1$ vertices from W. Since the $s+1$ forward trips must bring all vertices from W and also the remaining $m+q-1$ vertices from $V \cup U$ to the right bank, we get $(s+1)(m+q-1) \geq |W| + (m+q-1)$, which implies $s \geq r+1$. Hence, there does not exist a schedule of length $2r+1$. □

We remark that the reduction in Theorem 6.2 can be rewritten into an L-reduction from the vertex cover problem to the t-trip constrained Alcuin number problem. Therefore, for every fixed $t \geq 3$ the t-trip constrained Alcuin number is APX-hard to approximate (even in planar graphs).

7. Special Graph Classes. We now discuss the Alcuin number for several classes of specially structured graphs, as well as how small-boat graphs can be distinguished from large-boat graphs in these classes.

7.1. Chordal Graphs and Trees. *Chordal graphs* are graphs in which every cycle of length exceeding three has a chord, that is, an edge joining two nonconsecutive vertices in the cycle; see, for instance, Golumbic [6]. An equivalent characterization states that a graph is chordal if and only if every minimal vertex separator induces a clique. A *split graph* is a graph $G = (V, E)$ whose vertex set can be partitioned into an induced clique and an induced 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. Note that split graphs and trees are special cases of chordal graphs.

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

LEMMA 7.1. *Let $G = (V, E)$ be a connected chordal graph. Then G is a small-boat graph if and only if one of the following two conditions 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.*

Proof. If G is a split graph, sufficiency of the stated condition follows essentially from Lemma 4.2. Necessity follows from the structure theorem, Theorem 3.1: Since Y is a clique, Y_1 and Y_2 both consist of a single element, and hence $|X_3| \leq 2$. Assume $Y_1 = \{u\}$ and $Y_2 = \{v\}$. If $u \neq v$, then all common neighbors of u and v in X are in X_3. If $u = v$, then all neighbors of u in X are in X_3.

If G is not a split graph, it must induce C_4 or C_5 or $2K_2$. Since G is also chordal, it hence must contain an induced subgraph $2K_2$ with two independent edges $[a, b]$ and $[c, d]$. Since the vertex set $V - \{a, b, c, d\}$ separates a and c, it contains a minimal separator C that induces a clique in G. Then $G - C$ has two nontrivial components, and Lemma 4.1 applies. □

As a special case, Lemma 7.1 contains the following classification of trees (which has already been derived in [2, 8]). Stars $K_{1,k}$ with $k \geq 3$ leaves are split graphs that do not satisfy condition (1) of Lemma 7.1; therefore they are large-boat graphs (note that this also follows from Lemma 4.4). 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 $i > 0$, edges $[b_0, b_j]$ for all $j > 0$, and the edge $[a_0, b_0]$. 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. 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 and Plummer [9]. In this section we show that also the Alcuin number of a bipartite graph can be computed in polynomial time. As an immediate consequence we get that bipartite small-boat graphs can be recognized in polynomial time.

THEOREM 7.2. *For a bipartite graph $G = (V, E)$, the Alcuin number can be computed in polynomial time.*

Proof. It is easy to decide whether the bipartite graph 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 4.3 yields $\text{ALCUIN}(G) = \tau(G)$. Hence, in the light of Theorem 3.1 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) with $b = \tau(G)$?

We consider two copies $G' = (V', E')$ and $G'' = (V'', E'')$ of G, and the corresponding maximum stable sets X' and X'' and minimum vertex covers Y' and Y''. We construct a new graph H that consists of the vertices and edges in G' and G'' and of a perfect matching between X' and X''; for every vertex $v \in X$, the perfect matching matches the two copies of v in X' and X'' to each other. It is easy to verify that H is bipartite, since G is bipartite.

Suppose that G contains sets X_1, X_2, X_3 and Y_1, Y_2 with the desired properties. Let X_1' and Y_1' denote the sets corresponding to X_1 and Y_1 in G', and let X_2'' and Y_2'' denote the sets corresponding to X_2 and Y_2 in G''. Since $X_1 \cup Y_1$ and $X_2 \cup Y_2$ are stable sets, and since X_1 and X_2 are disjoint, the set $X_1' \cup X_2'' \cup Y_1' \cup Y_2''$ is a stable set of size at least $\alpha(G)$ in H that has nonempty intersections with both Y' and Y''. On the contrary, if H contains a stable set Z of size at least $\alpha(G)$ that has nonempty intersections with Y' and Y'', then we may define $X_1 = X' \cap Z$, $X_2 = X'' \cap Z$, $Y_1 = Y' \cap Z$, and $Y_2 = Y'' \cap Z$. The matching between X' and X'' ensures that X_1 and X_2 are disjoint, and it can be verified that these sets satisfy conditions (i)–(iv) with $b = \tau(G)$.

So the entire problem boils down to finding a large stable set in H that has nonempty intersections with Y' and Y''. This can easily be done by computing maximum size stable sets in a sequence of graphs (for instance, check every possible pair of vertices in Y' and Y'' as potential members of the stable set Z, and update H appropriately). If we succeed in finding such a stable set of cardinality $\alpha(G)$, then $\text{ALCUIN}(G) = \tau(G)$; otherwise $\text{ALCUIN}(G) = \tau(G) + 1$. □

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 4.2 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 7.3. *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 4.2. 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 were to lie in different regions, they could not have three common neighbors, a contradiction. Thus y_3, y_4, y_5 must all lie in the same region, say in the region bounded by $y_1 - a - y_2 - b - y_1$. Hence there is a vertex $z_{1,2}$ (= vertex c) that is adjacent to both vertices y_1 and y_2, but not adjacent to any 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 $z_{i,j}$ that is not adjacent to the other three vertices in $\{y_1, y_2, y_3, y_4, y_5\}$. The 15 vertices y_i and $z_{i,j}$ form a subdivision of K_5 in G, and thus yield the desired contradiction to planarity. □

The condition $\tau(G) \geq 5$ in Lemma 7.3 cannot be dropped, since there exists a variety of planar graphs G with $\tau(G) \leq 4$ that are large-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 $\text{ALCUIN}(G) = 5$.

Lemma 7.3 implies that there is a polynomial time algorithm that decides whether a planar graph G is small-boat or large-boat: In case G has a vertex cover of size at most 4 we use Theorem 5.1 to decide whether $\text{ALCUIN}(G) = \tau(G)$, and in the case where 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 the following 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 8.1. *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].

QUESTION 8.2. *Is there a polynomial time algorithm for computing the Alcuin number of a perfect graph?*

Trees, split graphs, chordal graphs, and bipartite graphs are special cases of perfect graphs, and we have shown that for all of these classes the Alcuin number can be computed in polynomial time. Also for other well-known classes of perfect graphs like

cographs or permutation graphs we can compute the Alcuin number in polynomial time; this can be done by standard dynamic programming approaches that are similar to the dynamic programs for computing the stability number for these classes.

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

QUESTION 8.3. *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 [11]); the complexity class DP might be a reasonable guess.

REFERENCES

[1] M. ASCHER, *A river-crossing problem in cross-cultural perspective*, Math. Mag., 63 (1990), pp. 26–29.
[2] P. BAHLS, *The Wolf, the Goat, and the Cabbage: A Modern Twist on a Classical Problem*, unpublished manuscript, University of North Carolina at Asheville, Asheville, NC, 2005.
[3] N. L. BIGGS, *The roots of combinatorics*, Historia Math., 6 (1979), pp. 109–136.
[4] R. BORNDÖRFER, M. GRÖTSCHEL, AND A. LÖBEL, *Alcuin's transportation problems and integer programming*, in Charlemagne and His Heritage: 1200 Years of Civilization and Science in Europe, Vol. 2, Brepols, Turnhout, 1998, pp. 379–409.
[5] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.
[6] M. C. GOLUMBIC, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.
[7] S. KHOT AND O. REGEV, *Vertex cover might be hard to approximate to within* $2-\varepsilon$, J. Comput. System Sci., 74 (2008), pp. 335–349.
[8] M. LAMPIS AND V. MITSOU, *The ferry cover problem*, in Proceedings of the 4th International Conference on Fun with Algorithms (FUN 2007), Lecture Notes in Comput. Sci. 4475, Springer, Berlin, 2007, pp. 227–239.
[9] L. LOVÁSZ AND M. D. PLUMMER, *Matching Theory*, Ann. Discrete Math. 29, North-Holland, Amsterdam, 1986.
[10] R. NIEDERMEIER, *Invitation to Fixed-Parameter Algorithms*, Oxford University Press, New York, 2006.
[11] C. H. PAPADIMITRIOU, *Computational Complexity*, Addison-Wesley, Reading, MA, 1994.
[12] E. PRISNER, *Generalizing the wolf-goat-cabbage problem*, Electron. Notes Discrete Math., 27 (2006), p. 83.