Ferry Cover with Connectivity Constraints

Niranjan Balachandran
IIT Bombay, India

Ankita Dargad
IIT Bombay, India

Urban Larsson
IIT Bombay, India

Neeldhara Misra
IIT Gandhinagar, India

Umesh Shankar
IIT Bombay, India

Abstract

The classical Ferry Cover problem asks for the minimum boat capacity needed to transport all vertices of a graph across a river such that no edge remains on either bank at any time – a requirement that the banks induce stable (independent) sets. We study a natural generalization in which the banks must satisfy an arbitrary graph property. For hereditary properties such as acyclicity or planarity, we show that the structural characterization of small-boat and large-boat graphs established by Csorba, Hurkens, and Woeginger extends directly.

We then turn to the connected-bank variant, where the property of interest – connectedness – is not hereditary: both banks must induce connected subgraphs throughout the transfer. We provide a complete characterization of graphs that can be transferred with a boat of size one (boat-1 graphs): a connected graph is boat-1 if and only if its block-cut tree is a path. This characterization yields a linear-time recognition algorithm. As a consequence, we show that every biconnected graph is boat-1, since such graphs admit an st-numbering. We also develop an efficient algorithm for determining the boat number of trees. Our work opens new directions for river-crossing problems under non-hereditary bank constraints.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Mathematics of computing → Graph theory

Keywords and phrases ferry cover, river crossing, block-cut tree, st-numbering, hereditary graph property, connectivity

Digital Object Identifier 10.4230/LIPIcs.FUN.2026.6

Acknowledgements The authors thank Arjun Arul for very helpful discussions towards arguing that there exist optimal schedules that do not ferry people back. We also acknowledge the use of Claude in the generation of TikZ code and editorial inputs on the writeup, and Gemini Nano Banana for generating the image in the introduction.

1 Introduction

The classical *Alcuin's River Crossing Problem*, in which a man must transport a wolf, a goat, and a bundle of cabbages across a river without leaving incompatible pairs together, is among the earliest known combinatorial puzzles. The incompatible pairs are (wolf, goat) and (goat, cabbages), as the wolf would eat the goat and the goat would eat the cabbages if left together on any bank without human supervision. Formally, the puzzle can be modeled by a graph on three vertices (wolf, goat, cabbages), where the edges represent mutual incompatibilities. The goal is to move all vertices from one bank to the other using a boat of limited capacity, ensuring that two adjacent vertices are not left together on the same bank at any point.


© Niranjan Balachandran, Ankita Dargad, Urban Larsson, Neeldhara Misra, and Umesh Shankar;
licensed under Creative Commons License CC-BY 4.0

Figure 1 In Alcuin's classical river crossing puzzle, a farmer must transport a wolf, a goat, and a cabbage across a river using a boat that can carry only one item at a time, ensuring that incompatible pairs (wolf–goat, goat–cabbage) are never left alone together.

This problem was first generalized by Lampis and Mitsou [3], who asked for the smallest boat capacity b (excluding the sailor's seat) for a given graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ that allows all vertices of G to be transferred from one bank to the other without ever leaving an edge of G on a bank. They called it the Ferry Cover (FC) problem and established the following tight relation between the ferry cover number b and the vertex cover number $\mathrm{OPT}_{\mathrm{VC}}(\mathsf{G})$:

$$\mathrm{OPT}_{\mathrm{VC}}(\mathsf{G}) \leqslant \mathsf{b} \leqslant \mathrm{OPT}_{\mathrm{VC}}(\mathsf{G}) + 1.$$

A graph is called *small-boat* if the boat capacity required is exactly equal to its vertex cover number; otherwise, it is called *large-boat*. They proved that computing $\mathrm{OPT}_{\mathrm{FC}}(\mathsf{G})$ is NP-hard and APX-hard in general.

Following this foundational work, Csorba, Hurkens, and Woeginger [1] provided a structural characterization of small-boat graphs. Seify and Shahmohamad [5] extended these results by identifying additional graph classes where the Alcuin number coincides with the vertex cover number. Ito, Langerman, and Yoshida [2] proposed generalized river-crossing models that unify various well-known puzzles in this area and studied their algorithmic complexity. More recently, Shan and Kang [6] analyzed the Ferry Cover problem on regular and small-degree graphs.

Collectively, these works offer a strong characterization of small-boat and large-boat graphs for Ferry Cover problems where the required graph property on each bank is *stability* or *independence*. This naturally raises the question of how the problem behaves under alternative constraints on the banks, for instance, when the induced subgraphs on each bank must satisfy connectedness, acyclicity, or planarity. To motivate the need for considering other graph properties, let us introduce a new problem from a more intuitive perspective. Imagine a community living on one bank of a river that wishes to cross to the other side. Not everyone in the community knows each other directly, yet the community as a whole remains cohesive – there is always a chain of people linking any two individuals. However, if two groups of people have no such link between them, they may start seeing each other as strangers or even as threats to the community, which can lead to conflict.

The community wants to build the smallest possible boat, since each extra seat adds a lot to the cost, such that the sailor can ferry everyone across the river while making sure that at every step of the process, the people left on each bank still form a socially connected group. If at any point the people on one bank become split into mutually disconnected subgroups, the peace of the community may be disturbed. This story motivates studying a

new variant of the classical Ferry Cover problem, where the requirement on each bank is not stability but *connectedness*. In graph-theoretic terms, we may view each individual as a vertex, and an edge connects two vertices if the corresponding individuals know each other directly. The goal is to determine the smallest boat capacity that allows all vertices of a connected graph G to be transferred from one bank to the other, such that at every step, the subgraphs induced by the vertices on both banks remain connected. We refer to this model as the *connected-bank variant* of the Ferry Cover problem.

In this paper, we first focus on hereditary graph properties such as acyclicity and planarity. These are properties that, if satisfied by a graph, are also satisfied by all of its subgraphs. We prove that the structural theorem given by Csorba et al. [1] also holds when the required property on each bank is a hereditary property.

This naturally leads to the question of what happens when the property required on the banks is not hereditary. The story above already illustrates one such case, where the property of interest is connectedness. Here, we assume G is connected, and we say that G is *boat-*n if n is the minimum boat size sufficient to transfer all vertices of G across the river while preserving connectivity on both banks at all times.

Our main result provides a complete characterization of boat-1 graphs: a connected graph G is boat-1 if and only if its block-cut tree forms a path. This structural insight yields not only an exact combinatorial description but also algorithmic consequences, enabling linear-time recognition of boat-1 graphs. We also develop an efficient algorithm for determining the boat number of trees. We believe this work extends the foundational landscape of Ferry Cover and opens up new directions for variants in which the bank property is neither hereditary nor trivial.

Related Work

The Ferry Cover problem originates from the classical river crossing puzzle attributed to Alcuin of York (circa 8th century), which can be modeled as transferring vertices of a conflict graph where edges represent incompatible pairs that cannot be left together unsupervised. Lampis and Mitsou [3] formalized this as the Ferry Cover problem, establishing the fundamental bound $\tau(G) \leqslant c(G) \leqslant \tau(G) + 1$ relating the Alcuin number $c(G)$ to the vertex cover number $\tau(G)$, and proving NP-hardness and APX-hardness. This dichotomy partitions graphs into *class 1* (small-boat, where $c(G) = \tau(G)$) and *class 2* (large-boat, where $c(G) = \tau(G) + 1$).

Csorba, Hurkens, and Woeginger [1] provided the key structural characterization that precisely distinguishes these classes via a partition condition on stable sets. A necessary condition for class 2 is that G have a unique minimum vertex cover; graphs with multiple minimum covers can swap between them during transfer, avoiding the extra seat. Seify and Shahmohamad [5] gave a classification theorem: G is class 2 if and only if for every pair of independent sets $I, J \subseteq V$, the union $I \cup J$ does not cover all vertices. They also identified additional small-boat graph families. Lampis and Mitsou [3] provided a polynomial-time algorithm for trees, exploiting the tree structure via dynamic programming to determine the Alcuin number. They showed that stars $K_{1,m}$ are class 2 for $m \geqslant 3$ but class 1 for $m \leqslant 2$, and that graphs of maximum degree at most 2 (paths and cycles) are always class 1.

Shan and Kang [6] provided a comprehensive classification for regular graphs and graphs with maximum degree at most 5, confirming that all regular bipartite graphs are class 1 (by symmetry of covers) while certain regular non-bipartite graphs can be class 2. Ito, Langerman, and Yoshida [2] proposed a unifying framework for river-crossing puzzles, studying round-trip constrained variants where the number of crossings is limited, and proved these variants are also NP-hard.

Importantly, the Csorba–Hurkens–Woeginger structural theorem is remarkably versatile: as we show in this paper, it extends to any hereditary property Π by replacing the vertex cover with a minimum Π-deletion set. However, our main focus departs from all prior work by considering a non-hereditary constraint – connectedness – which introduces fundamentally different structural challenges. Unlike the classical setting where the boat number lies in $\{\tau(G), \tau(G)+1\}$, the "boat number" for connectivity is not bounded by any simple deletion-set measure, and the scheduling constraints become significantly more intricate.

2 Preliminaries

▶ **Definition 1** (Induced Subgraph). *Let $G = (V, E)$ be a graph and let $I \subseteq V$. The induced subgraph of G on I is a graph with vertex set I and edge set consisting of all edges of G with both endpoints in I. It is denoted by G_I, or $G[I]$.*

Throughout the paper, we do not distinguish between a vertex set $I \subseteq V(G)$ and the subgraph of G induced by I, and use I to denote both when the meaning is clear from the context.

▶ **Definition 2** (Hereditary Property). *A graph property Π is hereditary if every induced subgraph of a graph satisfying Π satisfies Π.*

For example, the property of not containing cycle is a hereditary property because if a graph G does not contain any cycle, then any induced subgraph of G also does not contain any cycle. The property of having maximum degree not more than a constant is also a hereditary property. Whereas the property of having a path of length larger than a constant is not a hereditary property. The property connectedness is also not hereditary as subgraph of a connected graph might not be connected.

We denote the set of hereditary properties by $\mathcal{P}_H$.

▶ **Definition 3** (biconnected). *A graph is biconnected if removal of any vertex does not disconnect the graph or equivalently, there are two vertex disjoint paths between any two vertices.*

▶ **Definition 4** (Block). *Blocks are the maximal biconnected components of a graph.*

▶ **Definition 5** (cut-vertex). *A vertex v of a graph G is called a cut-vertex if removal of v increases the number of connected components of G.*

▶ **Observation 6.** *Two blocks of a graph cannot have more than 1 vertex in common.*

▶ **Definition 7** (Small-Boat). *A graph G is called small-boat if the boat size required to transfer all vertices of G across the river is equal to the size of the minimum vertex cover.*

A graph, which is not small-boat, is called *large-boat*. Note that we do not need a boat of size greater than 1+ the size of the minimum vertex cover, because we can keep the minimum vertex cover on the boat and transfer the rest of the vertices one by one.

The next theorem characterizes the small and large boat graphs.

▶ **Theorem 8** ([1],Theorem 3.1). *A graph $G = (V, E)$ can be transferred across the river with boat of size $b \geqslant 1$ if and only if there exists five subsets X_1, X_2, X_3, Y_1 and Y_2 of V such that:*

1. *X_1, X_2 and X_3 is a partition of a stable set X in G,*
2. *Y_1 and Y_2 are non-empty subsets of the set $Y := V - X$, which satisfies $|Y| \leqslant b$,*
3. *$X_1 \cup Y_1$ and $X_2 \cup Y_2$ are stable sets in G, and*
4. *$|Y_1| + |Y_2| \geqslant |X_3|$.*

The original Alcuin's problem requires the banks to be stable sets but this property can be changed as per the requirement. For example, we might need the graphs on the banks to be connected or to be cycle free, etc. This gives the Alcuin's problem a completely new direction. We call this new set of Alcuin's problem as Generalized Alcuin problems. Thanks to [1], the structural theorem (Theorem 8) is so versatile that we can use the same theorem for many problems in Generalized Alcuin problems set. More precisely, whenever the required property P on the banks is hereditary, we can use the same theorem with Y being the subset to be removed so that $V \setminus Y$ satisfies P. We prove this in the next section.

Before moving to the proof of structural theorem for any hereditary property, we review a few notations of [1]. A *schedule* is a finite sequence of triplets (L_k, B_k, R_k), where L_k, B_k and R_k are the vertices on the left bank, boat and the right bank, respectively, during k^{th} transfer.

▶ **Definition 9** (Feasible Schedule with respect to P). *Given a graph $G = (V, E)$, boat size b and a hereditary graph property P, a feasible schedule is a schedule $(L_k, B_k, R_k)_{1 \leqslant k \leqslant t}$ such that*

1. *For any k, L_k, B_k and R_k is a partition of V such that both L_k and R_k satisfy P and $|B_k| \leqslant b$;*
2. *During the first transfer, we do not have anything on the right bank and in the last transfer we do not have anything on the left bank, therefore $L_1 \cup B_1 = V = B_t \cup R_t$ and $R_1 = \phi = L_t$;*
3. *Odd numbered transfer denotes transfer from left to right bank and even numbered transfer denotes transfer from right to left bank, therefore for even $k \geqslant 2$, $L_{k-1} = L_k$ and $B_{k-1} \cup R_{k-1} = B_k \cup R_k$, and for odd $k \geqslant 1$, $R_{k-1} = R_k$ and $B_{k-1} \cup L_{k-1} = B_k \cup L_k$.*

▶ **Example 10** (Transferring a 4-cycle with boat size 2). Consider the cycle C_4 on vertices $\{1, 2, 3, 4\}$ with edges $\{1, 2\}, \{2, 3\}, \{3, 4\}, \{4, 1\}$. The two diagonal pairs $\{1, 3\}$ and $\{2, 4\}$ are independent sets. With a boat of capacity $b = 2$, we can transfer all vertices while keeping both banks stable (independent) at all times.

3 Hereditary Properties

In this section, we show that the structural characterization of Csorba, Hurkens, and Woeginger [1] extends naturally to any hereditary graph property. We begin by establishing bounds on the boat size in terms of a minimum deletion set.

▶ **Observation 11.** *Let b_P denote the minimum boat size required to transfer $G = (V, E)$ when the required property on the banks is a hereditary property P. Let Y be a smallest subset of V such that the induced subgraph $G[V \setminus Y]$ satisfies P. Then*

$$|Y| \leqslant b_P \leqslant |Y| + 1.$$

Proof. For the lower bound, observe that during the first trip, the sailor must take at least $|Y|$ vertices so that the induced subgraph on the remaining vertices satisfies P. Thus $b_P \geqslant |Y|$.

For the upper bound, we exhibit a feasible schedule using a boat of size $|Y| + 1$. The sailor keeps Y on the boat permanently and transfers the remaining vertices one at a time using the one free seat. At any point during this process, both banks contain subsets of $V \setminus Y$. Since $G[V \setminus Y]$ satisfies P and P is hereditary, every induced subgraph of $G[V \setminus Y]$ also satisfies P. Hence $b_P \leqslant |Y| + 1$. ◀

▶ **Definition 12** (Small-Boat and Large-Boat with respect to P)**.** *Let* P *be a hereditary property and let* $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ *be a graph. Let* Y *be a smallest subset of* V *such that* $\mathsf{G}[\mathsf{V} \setminus \mathsf{Y}]$ *satisfies* P*. We say* G *is* small-boat with respect to P *if all vertices can be transferred with a boat of size* $|\mathsf{Y}|$*; otherwise,* G *is* large-boat with respect to P*.*

The following theorem generalizes Theorem 8 to arbitrary hereditary properties. The argument is very similar to the proof for the original setting, but we include it here for completeness.

▶ **Theorem 13** (Structure Theorem for Hereditary Properties)**.** *Let* $\mathsf{P} \in \mathcal{P}_{\mathrm{H}}$ *be a hereditary property. A graph* $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ *can be transferred across the river with a boat of size* $\mathsf{b} \geqslant 1$*, maintaining property* P *on both banks throughout, if and only if there exist subsets* $\mathsf{X}_1, \mathsf{X}_2, \mathsf{X}_3, \mathsf{Y}_1, \mathsf{Y}_2 \subseteq \mathsf{V}$ *such that:*

1. $\mathsf{X}_1, \mathsf{X}_2, \mathsf{X}_3$ *partition a set* $\mathsf{X} \subseteq \mathsf{V}$ *where* $\mathsf{G}[\mathsf{X}]$ *satisfies* P*;*
2. Y_1 *and* Y_2 *are non-empty subsets of* $\mathsf{Y} \coloneqq \mathsf{V} \setminus \mathsf{X}$ *with* $|\mathsf{Y}| \leqslant \mathsf{b}$*;*
3. $\mathsf{G}[\mathsf{X}_1 \cup \mathsf{Y}_1]$ *and* $\mathsf{G}[\mathsf{X}_2 \cup \mathsf{Y}_2]$ *both satisfy* P*; and*
4. $|\mathsf{Y}_1| + |\mathsf{Y}_2| \geqslant |\mathsf{X}_3|$*.*

Proof. We prove both directions separately.

(⇒) Sufficiency. Suppose subsets $\mathsf{X}_1, \mathsf{X}_2, \mathsf{X}_3, \mathsf{Y}_1, \mathsf{Y}_2$ satisfying conditions (1)–(4) exist. We construct a feasible schedule, illustrated in Figure 2. We use the notation $\mathsf{L} \mid \mathsf{B} \mid \mathsf{R}$ to denote a snapshot where L, B, and R are the vertices on the left bank, boat, and right bank, respectively.

1. **Initial transport of Y:** Load Y onto the boat and deposit Y_1 on the right bank, then return. This yields $\mathsf{X} \mid \mathsf{Y} \setminus \mathsf{Y}_1 \mid \mathsf{Y}_1$. This is valid since $|\mathsf{Y}| \leqslant \mathsf{b}$, and both X and Y_1 induce subgraphs satisfying P (by conditions (1) and (3), using heredity).
2. **Transfer $\mathbf{X_1}$:** Since $\mathsf{Y}_1 \neq \emptyset$, there is at least one free seat. Transfer elements of X_1 one by one from left to right, reaching $\mathsf{X}_2 \cup \mathsf{X}_3 \mid \mathsf{Y} \setminus \mathsf{Y}_1 \mid \mathsf{Y}_1 \cup \mathsf{X}_1$. Throughout, the left bank contains subsets of X and the right bank contains subsets of $\mathsf{X}_1 \cup \mathsf{Y}_1$, both satisfying P by heredity.
3. **Exchange via $\mathbf{X_3}$:** Partition X_3 into X_{31} and X_{32} with $|\mathsf{X}_{31}| \leqslant |\mathsf{Y}_1|$ and $|\mathsf{X}_{32}| \leqslant |\mathsf{Y}_2|$ (possible by condition (4)). Execute four crossings:

$$\begin{aligned}
&\mathsf{X}_2 \cup \mathsf{X}_{32} \mid (\mathsf{Y} \setminus \mathsf{Y}_1) \cup \mathsf{X}_{31} \mid \mathsf{X}_1 \cup \mathsf{Y}_1,\\
&\mathsf{X}_2 \cup \mathsf{X}_{32} \mid \mathsf{Y} \mid \mathsf{X}_1 \cup \mathsf{X}_{31},\\
&\mathsf{X}_2 \cup \mathsf{Y}_2 \mid (\mathsf{Y} \setminus \mathsf{Y}_2) \cup \mathsf{X}_{32} \mid \mathsf{X}_1 \cup \mathsf{X}_{31},\\
&\mathsf{X}_2 \cup \mathsf{Y}_2 \mid \mathsf{Y} \setminus \mathsf{Y}_2 \mid \mathsf{X}_1 \cup \mathsf{X}_3.
\end{aligned}$$

   At each step, both banks contain subsets of sets satisfying P, and the boat carries at most b vertices.
4. **Transfer $\mathbf{X_2}$:** Since $\mathsf{Y}_2 \neq \emptyset$, transfer elements of X_2 one by one, reaching $\mathsf{Y}_2 \mid \mathsf{Y} \setminus \mathsf{Y}_2 \mid \mathsf{X}$.
5. **Final transport:** Load Y_2 and deposit everyone on the right bank.

(⇐) Necessity. Given a feasible schedule $(\mathsf{L}_k, \mathsf{B}_k, \mathsf{R}_k)_{1 \leqslant k \leqslant t}$, we construct the required subsets. We may assume $\mathsf{B}_{k-1} \neq \mathsf{B}_k$ for all k, since consecutive identical boat contents indicate redundant trips.

Let Z be a smallest subset such that $\mathsf{G}[\mathsf{V} \setminus \mathsf{Z}]$ satisfies P. By Observation 11, $|\mathsf{Z}| \leqslant \mathsf{b}$. Let Y be any superset of Z with $|\mathsf{Y}| = \mathsf{b}$, and set $\mathsf{X} = \mathsf{V} \setminus \mathsf{Y}$. Then $\mathsf{G}[\mathsf{X}]$ satisfies P. We consider three cases.

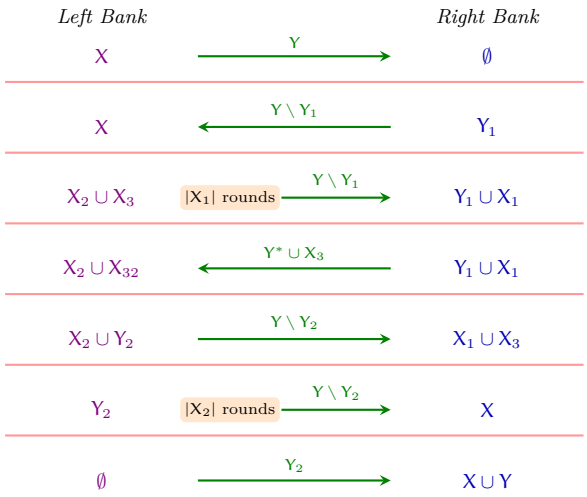


Figure 2 Schematic of the transfer schedule in the sufficiency proof of Theorem 13. Here $Y^* = (Y \setminus Y_1 \setminus Y_2)$. Each horizontal line represents a river crossing; arrows indicate the direction of travel, labeled with the boat's contents. The exchange phase (rows 3–4) swaps Y_1 and Y_2 between the banks while transferring X_3.

Case 1: There exists k with $L_k \cap Y \neq \emptyset$ and $R_k \cap Y \neq \emptyset$. Set $Y_1 = L_k \cap Y$, $Y_2 = R_k \cap Y$, $X_1 = L_k \cap X$, $X_2 = R_k \cap X$, and $X_3 = B_k \cap X$. Then X_1, X_2, X_3 partition X, satisfying (1). Both Y_1 and Y_2 are non-empty subsets of Y, satisfying (2). Since $L_k = X_1 \cup Y_1$ and $R_k = X_2 \cup Y_2$ must satisfy P (as they are bank contents), condition (3) holds. For (4):

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

giving $|X_3| \leqslant |Y_1| + |Y_2|$.

Case 2: There exists $1 < k < t$ with $B_k = Y$. Suppose k is even (the odd case is symmetric). Since $B_{k-1} \neq B_k = Y$, some vertices of Y were on the right bank during trip $k-1$; set $Y_1 = R_{k-1} \cap Y \neq \emptyset$. Similarly, $B_{k+1} \neq Y$ implies $Y_2 = L_{k+1} \cap Y \neq \emptyset$.

Every vertex of X lies in exactly one of L_k or R_k. Define $X_1 = R_{k-1} \cap X$, $X_2 = L_{k+1} \cap X$, $X_{31} = B_{k-1} \cap X$, $X_{32} = B_{k+1} \cap X$, and $X_3 = X_{31} \cup X_{32}$. Conditions (1)–(3) follow from the feasibility of the schedule. Since the sailor dropped X_{31} and picked up Y_1 (filling the boat), we have $|X_{31}| \leqslant |Y_1|$; similarly $|X_{32}| \leqslant |Y_2|$, yielding (4).

Case 3: Neither Case 1 nor Case 2 holds. Here, for all k, either $L_k \cap Y = \emptyset$ or $R_k \cap Y = \emptyset$, and $B_k \neq Y$ for $1 < k < t$. Since $L_1 \cap Y \neq \emptyset$ (initially all vertices are on the left) and $R_t \cap Y \neq \emptyset$ (finally all are on the right), there must be some transition point. A careful analysis shows this contradicts the assumption that neither Case 1 nor Case 2 applies. ◀

4 Connected Ferry Cover: Boat-1 Graphs

We now turn to the main focus of this paper: the connected-bank variant of Ferry Cover. Here, the property of interest, connectedness, is not hereditary, and therefore Theorem 13 does not apply. We restrict attention to connected graphs G, since disconnected graphs trivially fail the connectivity requirement.

▶ **Definition 14** (Boat-k Graph). *A connected graph* G *is* boat-k *if* k *is the minimum boat size sufficient to transfer all vertices of* G *across the river while maintaining connectivity on both banks throughout the process.*

Our goal in this section is to completely characterize boat-1 graphs. We begin with two simple but useful observations.

▶ **Observation 15.** *If* $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ *is boat-*$\mathfrak{b}$*, then any graph* $\mathsf{H} = (\mathsf{V}, \mathsf{E}')$ *with* $\mathsf{E} \subseteq \mathsf{E}'$ *is boat-*$\mathfrak{b}'$ *for some* $\mathfrak{b}' \leqslant \mathfrak{b}$*.*

Proof. Since H contains all edges of G, any subset of vertices inducing a connected subgraph in G also induces a connected subgraph in H. Thus, any valid transfer schedule for G remains valid for H. ◀

If the graph has a Hamiltonian path, them we can remove consecutive vertices along the path. This leads to the following observation.

▶ **Observation 16.** *If a graph* G *has a Hamiltonian path, then* G *is boat-1.*

This observation prompts the question of whether Hamiltonian graphs are the only boat-1 graphs. It turns out that the answer is NO!

▶ **Example 17** (A Boat-1 Graph without a Hamiltonian Path)**.** Consider the graph in Figure 3. This graph can be transferred with a boat of size 1 by following the vertex ordering $1, 2, 3, 4, 5, 6$.

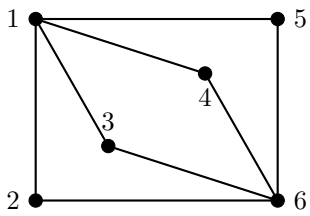


Figure 3 A boat-1 graph without a Hamiltonian path. The labeling $1, 2, \ldots, 6$ gives a valid transfer order.

The graph in Figure 3 is *biconnected* – removing any single vertex leaves it connected. This observation leads to a key structural insight.

Biconnected Graphs and st-Numberings

Recall that a graph is biconnected if the removal of any single vertex does not disconnect it. The crucial tool for understanding boat-1 graphs is the notion of an *st-numbering*, introduced by Lempel, Even, and Cederbaum [4].

▶ **Definition 18** (st-Numbering)**.** *Let* $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ *be a graph with* $|\mathsf{V}| = \mathfrak{n}$*. An* st-numbering *of* G *is a bijection* $\nu : \mathsf{V} \to \{1, 2, \ldots, \mathfrak{n}\}$ *such that for every vertex* v *with* $1 < \nu(v) < \mathfrak{n}$*, there exist neighbors* u *and* w *of* v *with* $\nu(u) < \nu(v) < \nu(w)$*.*

In other words, every vertex except the first and last has both a lower-numbered and a higher-numbered neighbor. The following classical result establishes the existence of st-numberings for biconnected graphs.

▶ **Theorem 19** (Lempel–Even–Cederbaum [4]). *Let* $G = (V, E)$ *be a biconnected graph and let* $\{s, t\} \in E$ *be any edge. Then* G *admits an st-numbering in which* s *receives number* 1 *and* t *receives number* $|V|$.

The connection to our problem is immediate: an st-numbering provides a transfer order that maintains connectivity.

▶ **Lemma 20.** *If* $G = (V, E)$ *admits an st-numbering* ν*, then for all* $i \in \{1, \ldots, |V|\}$*:*
1. *the subgraph induced by* $\{v : \nu(v) \leqslant i\}$ *is connected; and*
2. *the subgraph induced by* $\{v : \nu(v) \geqslant i\}$ *is connected.*

Proof. We prove (1) by induction on i. For $i = 1$, the subgraph contains a single vertex and is trivially connected. Assume the subgraph induced by vertices numbered 1 through $i - 1$ is connected. The vertex v with $\nu(v) = i$ has a neighbor u with $\nu(u) < i$ (unless $i = |V|$, in which case v is the last vertex and has a neighbor with smaller number by the st-numbering property). Thus, v is adjacent to the connected subgraph on vertices 1 through $i - 1$, so the subgraph on 1 through i is connected.

The proof of (2) is symmetric, proceeding by downward induction from $|V|$. ◀

▶ **Lemma 21.** *If a graph admits an st-numbering, then it is boat-1.*

Proof. Let ν be an st-numbering of G. Transfer vertices in the order $\nu^{-1}(1), \nu^{-1}(2), \ldots, \nu^{-1}(n)$. After transferring the vertex numbered i, the left bank contains vertices numbered $i + 1$ through n, and the right bank contains vertices numbered 1 through i. By Lemma 20, both subgraphs are connected. ◀

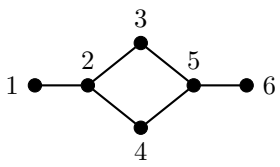
Combining Theorem 19 and Lemma 21 yields:

▶ **Corollary 22.** *Every biconnected graph is boat-1.*

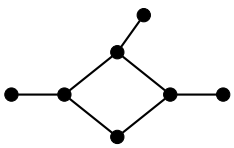
Beyond Biconnected Graphs

Corollary 22 characterizes a large class of boat-1 graphs, but it is not complete. There exist boat-1 graphs that are neither biconnected nor Hamiltonian.

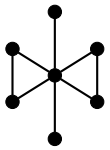
▶ **Example 23** (A Non-Biconnected Boat-1 Graph). The graph in Figure 4a(a) is not biconnected (vertex 2 is a cut-vertex) and has no Hamiltonian path, yet it is boat-1: transfer vertices in the order $1, 2, 3, 4, 5, 6$.



(a) Boat-1, not biconnected.



(b) Not boat-1: block has 3 cut-vertices.



(c) Not boat-1: cut-vertex in 4 blocks.

Figure 4 (a) A boat-1 graph that is neither biconnected nor Hamiltonian. (b) A graph where a block contains three cut-vertices. (c) A graph where a cut-vertex belongs to four blocks.

The key to understanding which non-biconnected graphs are boat-1 lies in their *block-cut tree* structure. Recall that the blocks of a graph are its maximal biconnected subgraphs (including bridges, viewed as biconnected graphs on two vertices).

▶ **Definition 24** (Block-Cut Tree). *The* block-cut tree *of a connected graph* G *is the bipartite graph* T *whose vertices are the blocks and cut-vertices of* G*, with an edge between a block* B *and a cut-vertex* c *if and only if* $c \in B$*.*

The block-cut tree is indeed a tree (hence the name) and captures how the blocks of G are connected through cut-vertices.

Obstructions to being Boat-1

We now identify structural conditions that prevent a graph from being boat-1.

▶ **Lemma 25.** *A connected graph* G *is not boat-1 if either:*

1. *some block of* G *contains three or more cut-vertices of* G*; or*
2. *some cut-vertex of* G *belongs to three or more blocks of* G*.*

Proof. We prove each case by showing that any transfer schedule requires a boat of size at least 2.

Case 1: A block B contains cut-vertices c_1, c_2, c_3 (and possibly more). Denote the set of cut-vertices of G by C. Consider any transfer schedule and let c be a cut-vertex in B that is neither the first nor the last among $\{c_1, c_2, c_3\}$ to be transferred. Such a vertex exists since $|\{c_1, c_2, c_3\}| \geqslant 3$. At the moment c is transferred, some cut-vertices from B are on the left bank and some are on the right bank.

Since c is a cut-vertex, removal of c divides G into 2 components, say G_1 and G_2. The block B is contained either in G_1 or in G_2 but not both, because B is a block. So, without loss of generality, assume that B is contained in G_1. This implies C is a subset of vertices of G_1 and therefore, during the transfer of u, some vertices of G_1 are on the left bank and some on the right bank. Consider the following cases depending upon the time when u is transferred:

1. u is transferred before transferring G_2: Just after the transfer of u, G_2 and a part of G_1 are on the left bank and the two cannot be connected in the absence of u because it contradicts u being the cut-vertex.
2. u is transferred after transferring G_2 partially or fully: just before the transfer of u, a part of G_1 and G_2 are on the right bank and the two cannot be connected in the absence of u because it contradicts u being the cut-vertex.

This contradicts the existence of a feasible schedule with a boat of size 1. This completes the proof of (1).

Case 2: A cut-vertex v belongs to blocks B_1, B_2, B_3 (and possibly more). In any transfer schedule, consider when v is transferred. By Observation 6, the blocks B_1, B_2, B_3 are pairwise vertex-disjoint except at v. Thus, at the moment v is transferred:

- If v is transferred before completely transferring at least two blocks, then after v leaves the left bank, at least two blocks (minus v) remain, and they are disconnected.
- If v is transferred after parts of at least two blocks have reached the right bank, then before v arrives, those partial blocks are disconnected on the right bank.

In either case, connectivity fails without transferring additional vertices alongside v. ◀

▶ **Theorem 26.** *If the block-cut tree of a connected graph* G *is not a path, then* G *is not boat-1.*

Proof. The block-cut tree T is bipartite with blocks on one side and cut-vertices on the other. If T is not a path, then some vertex of T has degree at least 3. If this vertex is a block, then that block contains at least three cut-vertices; if it is a cut-vertex, then that cut-vertex belongs to at least three blocks. By Lemma 25, G is not boat-1. ◀

The Complete Characterization

We now prove that the obstruction identified in Theorem 26 is the only obstruction.

▶ **Theorem 27** (Characterization of Boat-1 Graphs). *A connected graph G is boat-1 if and only if its block-cut tree is a path.*

Proof. The "only if" direction is Theorem 26. We prove the "if" direction by showing that G admits an st-numbering, from which boat-1 follows by Lemma 21. Let T be the block-cut tree of G. If G is biconnected, then T consists of a single block, which is trivially a path, and G is boat-1 by Corollary 22. Assume G has at least two blocks. We first establish:

▷ Claim 28. The endpoints of T are blocks, not cut-vertices.

Proof of Claim. Suppose an endpoint of T is a cut-vertex c. Then c has degree 1 in T, meaning c belongs to exactly one block B. But then removing c from G cannot disconnect G (since all of $G \setminus \{c\}$ remains within the components attached to B), contradicting that c is a cut-vertex. ◁

Thus T has the structure shown in Figure 5: blocks $B_1, B_2, \ldots, B_k$ connected by cut-vertices $c_1, c_2, \ldots, c_{k-1}$, where c_i is the unique vertex shared by B_i and B_{i+1}.

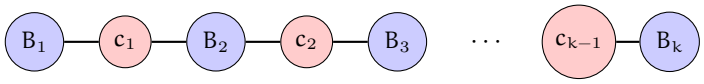


Figure 5 The block-cut tree of a boat-1 graph is a path alternating between blocks (blue) and cut-vertices (red).

We construct an st-numbering of G by combining st-numberings of the individual blocks.

▷ Claim 29. G admits an st-numbering.

Proof of Claim. For each block B_i, we construct an st-numbering with specified endpoints:

- For B_1: Choose any neighbor u of c_1 in B_1. Since B_1 is biconnected, by Theorem 19 there exists an st-numbering of B_1 with u numbered 1 and c_1 numbered $|B_1|$.
- For B_i with $1 < i < k$: We need c_{i-1} numbered first and c_i numbered last. If $\{c_{i-1}, c_i\} \in E(B_i)$, apply Theorem 19 directly. Otherwise, temporarily add the edge $\{c_{i-1}, c_i\}$ to B_i, obtain an st-numbering, then remove the edge. The st-numbering property is preserved since we only required c_{i-1} and c_i to be endpoints.
- For B_k: Choose any neighbor v of c_{k-1} in B_k. Obtain an st-numbering with c_{k-1} numbered 1 and v numbered $|B_k|$.

Now combine these numberings. Let $n_i = |B_i|$ for each i. For vertices in B_1, keep their numbers. For vertices in B_i with $i > 1$, add $(n_1-1)+(n_2-1)+\cdots+(n_{i-1}-1) = \sum_{j=1}^{i-1}(n_j-1)$ to each number. This offset accounts for the fact that each cut-vertex c_j is numbered last in B_j and first in B_{j+1}, receiving number $\sum_{j'=1}^{j} n_{j'} - (j-1) = \sum_{j'=1}^{j} n_{j'} - j + 1$ in the combined numbering.

The resulting numbering is an st-numbering of G: every vertex except the first and last has a smaller-numbered and larger-numbered neighbor (within its block, and the block numberings are compatible at cut-vertices). ◁

By Lemma 21, G is boat-1. ◀

▶ **Corollary 30.** *There is a linear-time algorithm to determine whether a connected graph is boat-1.*

5 Connected Ferry Cover on Trees

In this section, we determine the minimum boat size required to transfer any tree while maintaining connectivity on both banks. The key observation is that trees have a natural "peeling" structure: we can transfer vertices one at a time, starting from a leaf and working inward. However, when removing a vertex would disconnect the remaining graph, we must transfer the orphaned components along with it.

▶ **Definition 31** (Demand and Cost). *Let $S = (L_k, B_k, R_k)$ be a schedule for a graph G. The demand, denoted by $D_G(S)$ is defined as $\max_k | B_k |$. The cost, denoted by $C_G(S)$, is defined as $\sum_k | B_k |$.*

We will use $D(S)$ (resp. $C(S)$) instead of $D_G(S)$ (resp. $C_G(S)$) whenever G is clear from the context. Demand is nothing but the minimum boat size required to execute the given schedule and cost is the total amount paid to the sailor if each person pays 1 unit of currency every time (s)he crosses the river.

If each person crosses the river only once in a schedule S of G, the cost $C(S)$ is the same as $| V(G) |$. Note that this is minimal possible cost as each person has to cross the river at least once.

▶ **Definition 32** (Adjacent). *Two mutually disjoint induced connected subgraphs C_1 and C_2 of a graph G are said to be adjacent if the graph induced by $V(C_1) \cup V(C_2)$ is connected.*

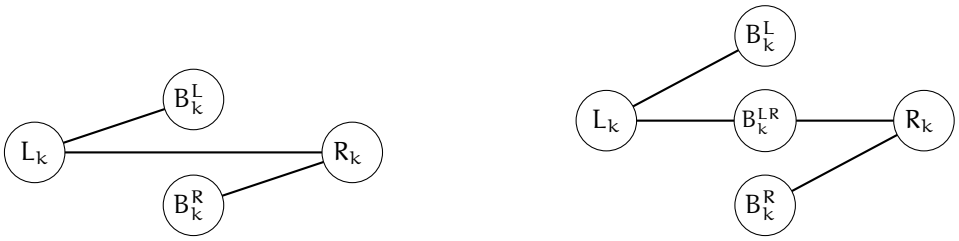
Equivalently, C_1 and C_2 are adjacent if there exists an edge of G between a vertex of C_1 and a vertex of C_2. By abuse of language, we sometimes say C_1 is adjacent to C_2 and vice-versa. By convention, we declare that the empty graph is adjacent to any connected graph, and vice versa.

Let $S = (L_k, B_k, R_k)_{1 \leqslant k \leqslant t}$ be a schedule of a tree $T = (V, E)$. For any index k, each CC of $T[B_k]$ is adjacent to $T[L_k]$, or to $T[R_k]$, or to both. When no confusion arises, we use the same symbol to denote a vertex subset and the subgraph induced by it.

Let B_k^L denote the union of vertices of all CCs of B_k that are adjacent to L_k, and let B_k^R denote the union of vertices of all CCs of B_k that are adjacent to R_k. Since T is acyclic, there can be at most one CC of B_k that is adjacent to both L_k and R_k; denote this component by B_k^{LR}. Moreover, if $B_k^{LR} = \varnothing$, then L_k and R_k are adjacent, as T is connected. This can be visualized in Figure 6.

▶ **Definition 33** (Uniformly adjacent). *Given a schedule $S = (L_k, B_k, R_k)$ of a graph, the connected components of B_k are said to be uniformly adjacent: if either all are adjacent to L_k, or all are adjacent to R_k, or all are adjacent to both L_k and R_k.*

Given a schedule, if the CCs of B_k are not uniformly adjacent, by abuse of language, we say that the k^{th} trip violates the uniform adjacency property. For brevity, we may also say that B_k is not uniformly adjacent, instead of explicitly referring to its connected components.



(a) When no component of B_k is adjacent to both banks.

(b) When B_k contains a component adjacent to both banks.

Figure 6 The tree T expressed in terms of the partition (L_k, B_k, R_k), together with the adjacencies between these parts; edges indicate adjacency in T.

The following lemma says that any schedule for a tree can be modified such that the contents of the boat are uniformly adjacent. The proof of the lemma is a case wise analysis of the first time the boat is not uniformly adjacent. We see that some vertices carried in that trip are brought back in a future trip, meaning that part of the movement was unnecessary. So, we modify the trips to avoid carrying extra vertices in such a way as to not increase the demand or cost of the schedule. By repeating this process whenever a bad trip appears, we eventually obtain a schedule where the contents of the boat of every trip are uniformly adjacent. We omit the proof here due to page constraints and direct the interested reader to the full version of the paper for a complete proof.

▶ **Lemma 34.** *Given a schedule* $S = (L_k, B_k, R_k)_{1 \leqslant k \leqslant t}$ *of a tree* T*, there exists another schedule* $\hat{S} = (\hat{L}_i, \hat{B}_i, \hat{R}_i)_{1 \leqslant i \leqslant \hat{t}}$ *of* T *with demand and cost at most* $D(S)$ *and* $C(S)$*, respectively, such that for each* i*, the connected components of* $\hat{B}_i$ *are uniformly adjacent.*

In Lemma 34, we showed that for any schedule S, there exists a schedule $\hat{S}$ with cost and demand no greater than those of S, such that in every trip of $\hat{S}$, the boat set is uniformly adjacent. The next lemma strengthens this result. It shows that, starting from $\hat{S}$, we can construct another schedule $\bar{S}$, again without increasing the cost or demand, such that in each trip of $\bar{S}$, the boat contains at most one connected component that is adjacent to both banks. Moreover, the boat set of every even-indexed trip in $\bar{S}$ is empty.

Before moving to the next lemma, we define another property of the boat set, called *bi-adjacency*.

▶ **Definition 35** (Bi-adjacent). *Given a schedule* $S = (L_k, B_k, R_k)$*, the boat set* B_k *is said to be* bi-adjacent *(or to satisfy* bi-adjacency*) if* B_k *is adjacent to both* L_k *and* R_k*.*

By abuse of terminology, we say that the k^{th} trip of S satisfies bi-adjacency if B_k is bi-adjacent.

▶ **Lemma 36.** *Given a schedule* $S = (L_k, B_k, R_k)_{1 \leqslant k \leqslant t}$ *of a tree* T*, there exists another schedule* $\bar{S} = (\bar{L}_i, \bar{B}_i, \bar{R}_i)_{1 \leqslant i \leqslant \bar{t}}$ *of* T *with cost and demand at most* $C(S)$ *and* $D(S)$*, respectively, such that for each* i*,* $\bar{B}_i$ *is bi-adjacent.*

Proof. Using Lemma 34, we get $\hat{S}$ of length $\hat{t}$ from the given schedule S. The cost and demand of $\hat{S}$ is not more than those of S, moreover, the boat set in each trip of $\hat{S}$ is uniformly adjacent.

Let ℓ be the first trip of $\hat{S}$ whose boat set $(\hat{B}_\ell)$ is not bi-adjacent. The trips of $\hat{S}$ after index ℓ may also violate bi-adjacency. Hence, the total number of trips of $\hat{S}$ that violate this property is at most $\hat{t} - \ell + 1$.

We construct a new schedule $\bar{S}$ with the following properties:

1. $\bar{S}$ is a legal schedule;
2. The cost and demand of $\bar{S}$ is at most those of $\hat{S}$; and
3. The number of trips of $\bar{S}$ that violates the bi-adjacency is at most $\hat{t} - \ell$.

We then apply the same procedure to the new schedule. Since at each step the upper bound on the number of trips violating bi-adjacency strictly decreases, and the schedule has finite length, the process terminates after finitely many steps.

We now construct the new schedule. Let ℓ be the first trip of $\hat{S}$ that violates bi-adjacency. The construction of $\bar{S}$ is divided into multiple cases, depending on whether ℓ is even or odd and boat set is adjacent to the left bank or the right bank.

Case 1: ℓ is odd and all CCs of $\hat{B}_\ell$ are not adjacent to $\hat{R}_\ell$. Let $\beta^1, \beta^2, \ldots, \beta^\alpha$ denote the connected components of $\hat{B}_\ell$, where $\alpha \geqslant 1$. Note that α cannot take the value 0, as otherwise ℓ^{th} trip satisfy bi-adjacency. For pictorial representation of this case, refer to Figure 7. We further subdivide this case depending on what is dropped from the boat at the end of the ℓ^{th} trip.

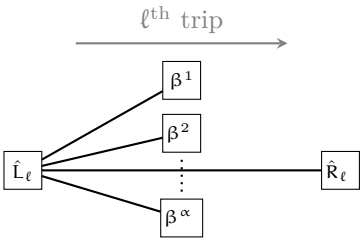


Figure 7 The ℓ^{th} trip of $\hat{S}$ in Case 1.

Subcase 1a: $\hat{R}_{\ell+1} \cap \hat{R}_\ell \neq \varnothing$. In this case, since $\hat{R}_{\ell+1}$ has some elements of $\hat{R}_\ell$, it cannot have any element of $\hat{B}_\ell$, otherwise $\hat{R}_{\ell+1}$ would be disconnected. This means $\hat{R}_{\ell+1} \subseteq \hat{R}_\ell$. Define a new schedule $\bar{S}$ as:

$$\begin{aligned}
(\bar{L}_i, \bar{B}_i, \bar{R}_i) &= (\hat{L}_i, \hat{B}_i, \hat{R}_i) \quad \text{for all } i \leqslant \ell - 1 \text{ and } i \geqslant \ell + 2,\\
(\bar{L}_\ell, \bar{B}_\ell, \bar{R}_\ell) &= (\hat{L}_\ell \cup \hat{B}_\ell,\ \varnothing,\ \hat{R}_\ell),\\
(\bar{L}_{\ell+1}, \bar{B}_{\ell+1}, \bar{R}_{\ell+1}) &= (\bar{L}_\ell,\ \hat{R}_\ell \backslash \hat{R}_{\ell+1},\ \hat{R}_{\ell+1}).
\end{aligned}$$

The schedule $\bar{S}$ is vacuously legal and

$$\begin{aligned}
&\bar{B}_i \subseteq \hat{B}_i \quad \text{for all } i \leqslant \ell - 1 \text{ and } i \geqslant \ell + 2, \qquad \bar{B}_\ell \subseteq \hat{B}_\ell,\\
&\bar{B}_{\ell+1} \subseteq \hat{B}_\ell \cup (\hat{R}_\ell \backslash \hat{R}_{\ell+1})\\
&\qquad = (\hat{B}_\ell \cup \hat{R}_\ell) \backslash \hat{R}_{\ell+1} \qquad \text{(as } \hat{R}_{\ell+1} \subseteq \hat{R}_\ell\text{)}\\
&\qquad = \hat{B}_{\ell+1}.
\end{aligned}$$

Hence, the cost and demand of $\bar{S}$ is at most those of $\hat{S}$. Additionally, ℓ^{th} trip of $\bar{S}$ satisfy bi-adjacency and therefore, the number of trips of $\bar{S}$ that violates bi-adjacency is at most $\hat{t} - \ell$.

Subcase 1b: $\hat{R}_{\ell+1} \cap \hat{R}_\ell = \varnothing$. In this case, either $\hat{R}_{\ell+1} = \varnothing$ or $\hat{R}_{\ell+1} \cap \beta^i \neq \varnothing$ for exactly one $i = 1, 2, \ldots, \alpha$ as otherwise $\hat{R}_{\ell+1}$ would be disconnected. In the former case, $\hat{R}_{\ell+1} = \varnothing$, define the new schedule as old schedule starting from $\ell + 2^{\text{th}}$ trip and it is easy to verify

that the new schedule is legal, less costly and demanding than $\hat{S}$ and the number of trips in new schedule is $\hat{t} - \ell$. In the later case, $\hat{R}_{\ell+1} \cap \beta^i \neq \varnothing$ for $i = m$ (say), everything that was sent to the right bank before ℓ^{th} trip is brought back in $\ell + 1^{th}$ trip, so to reduce the cost occurred in sending $\hat{R}_\ell$ is wasted. Hence, it would be better to start the schedule by sending just β^m and then continue as per the further steps of the old schedule. So define the new schedule $\bar{S}$ as:

$$(\bar{L}_1, \bar{B}_1, \bar{R}_1) = (V \backslash \beta^m,\ \beta^m,\ \varnothing),$$
$$(\bar{L}_2, \bar{B}_2, \bar{R}_2) = (\bar{L}_1,\ \beta^m \backslash \hat{R}_{\ell+1},\ \hat{R}_{\ell+1}),$$
$$(\bar{L}_i, \bar{B}_i, \bar{R}_i) = (\hat{L}_{i+\ell-1}, \hat{B}_{i+\ell-1}, \hat{R}_{i+\ell-1}) \quad \text{for all } i \geqslant 3.$$

The schedule $\bar{S}$ is vacuously legal and the cost and demand of $\bar{S}$ as at most those of $\hat{S}$ as

$$\bar{B}_i \subseteq \hat{B}_{i+\ell-1} \quad \text{for all } i \geqslant 3, \qquad \bar{B}_1 \subseteq \hat{B}_\ell,$$
$$\bar{B}_2 \subseteq (\hat{B}_\ell \backslash \hat{R}_{\ell+1}) \cup \hat{R}_\ell = (\hat{B}_\ell \cup \hat{R}_\ell) \backslash \hat{R}_{\ell+1} = \hat{B}_{\ell+1}. \qquad (\text{as } \hat{R}_{\ell+1} \subseteq \hat{B}_\ell)$$

The 1^{st} trip of the $\bar{S}$ satisfy bi-adjacency, hence the total number of trips of $\bar{S}$ that violates bi-adjacency is at most $\hat{t} - \ell$.

Case 2: ℓ is even and all CCs of B_ℓ are not adjacent to $\hat{R}_\ell$. Since ℓ^{th} trip is the first trip to violate bi-adjacency, $(\ell - 1)^{th}$ trip satisfy bi-adjacency. This implies $\hat{B}_{\ell-1} \cup \hat{R}_{\ell-1}$ is connected as $\hat{R}_{\ell-1}$ is connected and $\hat{B}_{\ell-1}$ is adjacent to $\hat{R}_{\ell-1}$. Since, the schedule $\hat{S}$ is legal, and ℓ is even, $\hat{B}_\ell \cup \hat{R}_\ell = \hat{B}_{\ell-1} \cup \hat{R}_{\ell-1}$. This implies, all CCs of $\hat{B}_\ell$ must be adjacent to $\hat{R}_\ell$, which contradicts the assumption.

The omit the details of the two cases one, when ℓ is even and all CCs of B_ℓ are not adjacent to $\hat{L}_\ell$ and other one, when ℓ is odd and all CCs of B_ℓ are not adjacent to $\hat{L}_\ell$, as they are similar to Cases 1 and 2.

This completes the proof. ◀

▶ Remark 37. Lemma 36 also implies that there is at most one connected component on the boat in each trip of $\bar{S}$.

In the next lemma, we show that given any optimal schedule, there exists another optimal schedule in which none of the vertices are ever transferred from the right bank to the left bank. To prove it, we use the following: In a schedule S of T, if all the vertices are transferred only once, $C(S) =| V(T) |$ and vice-versa. This means, if $C(S) >| V(T) |$, there is at least one trip from the right bank to the left bank, in which the boat is non-empty. We rectify the last such trip and reduce the cost by not increasing the demand. Doing this repetitively gives an optimal schedule with $C(S) =| V(T) |$.

▶ **Lemma 38** (Empty Return Trips). *For any tree* T*, there exists an optimal schedule in which no vertices are transferred from the right bank to the left bank. That is, all return trips* B_{2i} *are empty.*

Proof. Among the set of optimal schedules (the ones minimizes the boat size) of T, let $S = (L_k, B_k, R_k)_{1 \leqslant k \leqslant t}$ be the one that minimizes the cost. Then by Lemma 36, there exists $\bar{S} = (\bar{S}_k, \bar{B}_k, \bar{R}_k)_{1 \leqslant k \leqslant \bar{t}}$ such that $\bar{S}$ is optimal and $C(\bar{S}) \leqslant C(S)$. If the boat in all the return trips of $\bar{S}$ is empty, i.e., $C(\bar{S}) = |V(T)|$, then we are done. Otherwise, assume i is the first return trip of $\bar{S}$ in which the boat is non-empty, i.e., $\bar{B}_k = \varnothing$ for all even $k < i$ and $\bar{B}_i \neq \varnothing$.

By Lemma 36, $\bar{B}_{i-1}$ is adjacent to both $\bar{L}_{i-1}$ and $\bar{R}_{i-1}$ and $\bar{B}_i$ is adjacent to both $\bar{L}_i$ and $\bar{R}_i$.

If $\bar{B}_{i-1} \cap \bar{B}_i = \varnothing$, all the vertices sent to right bank in $i-1^{th}$ trip stayed on the right bank and a subset of vertices from $\bar{R}_{i-1}$ were taken on the boat in the next trip, i.e., $\bar{B}_i \subseteq \bar{R}_{i-1}$. Since $\bar{R}_{i-1}$ is not adjacent to $\bar{L}_{i-1} = \bar{L}_i$ (as otherwise it contradicts T being a tree), $\bar{B}_i$ cannot be adjacent to $\bar{L}_i$, which is a contradiction. Hence, $\bar{B}_{i-1} \cap \bar{B}_i \neq \varnothing$. Let $P = \bar{B}_{i-1} \cap \bar{B}_i$, $X = \bar{B}_{i-1} \setminus P$ and $Y = \bar{B}_i \setminus P$. Then,

1. $X \cap Y = \varnothing$;
2. $X \subseteq \bar{R}_i$ and $Y \subseteq \bar{R}_{i-1}$ as $X \cup \bar{R}_{i-1} = Y \cup \bar{R}_i$.

We know $\bar{B}_i = P \cup Y$ is adjacent to $\bar{L}_i$. If Y is adjacent to $\bar{L}_i = \bar{L}_{i-1}$, then that implies $\bar{R}_{i-1}$ is adjacent to $\bar{L}_{i-1}$, which contradicts T being a tree. Hence P is adjacent to $\bar{L}_i = \bar{L}_{i-1}$. Now consider the following schedule $S' = (L'_k, B'_k, R'_k)_{1 \leqslant k \leqslant \bar{t}}$:

1. All the trips of S' up to $i-2$ are same as those of $\bar{S}$,
2. $L'_{i-1} = \bar{L}_{i-1} \cup P, \quad B'_{i-1} = X, \quad R'_{i-1} = R_{i-1}$,
3. $L'_i = L'_{i-1}, \quad B'_i = Y, \quad R'_i = R_i$, and
4. All trips of S' from $i+1$ to $\bar{t}$ is same as those of $\bar{S}$.

▷ Claim 39. S' is legal and $C(S') < C(\bar{S})$ and $D(S') \leqslant D(S)$.

Proof. To show that S' is legal, we must show the following:

1. $L'_{i-1} = L'_i$ is connected,
2. $L'_{i-1} \cup B'_{i-1} = \bar{L}_{i-1} \cup \bar{B}_{i-1}$,
3. $B'_{i-1} \cup R'_{i-1} = B'_i \cup R'_i$, and
4. $L'_i \cup B'_i = \bar{L}_i \cup \bar{B}_i$.

Starting with item (1), since P is adjacent to $\bar{L}_{i-1}$, L'_{i-1} and L'_i are connected each.

For item (2), we know $L'_{i-1} \cup B'_{i-1} = \bar{L}_{i-1} \cup P \cup X = \bar{L}_{i-1} \cup \bar{B}_{i-1}$.

For item (3), we know $B'_{i-1} \cup R'_{i-1} = X \cup R_{i-1} = Y \cup R_i = B'_i \cup R'_i$.

For item (4), we know $L'_i \cup B'_i = \bar{L}_{i-1} \cup P \cup Y = \bar{L}_i \cup \bar{B}_i$.

The cost of S' is strictly less than $C(\bar{S})$ and $D(S') \leqslant D(S)$, vacuously. This completes the proof of the claim. ◁

This implies S' is an optimal schedule with lesser cost than S, which is a contradiction. Hence, the assumption about existence of i such that $\bar{B}_i$ is non-empty is not true. This completes the proof. ◀

▶ **Definition 40.** *Let $S = (L_k, B_k, R_k)_{1 \leqslant k \leqslant t}$ be a schedule. The* weight sequence *of the schedule, denoted by $\omega(S)$, is defined as the sequence $\omega(S) := (|B_i|)_{i=1}^{\infty}$ (with $|B_j| = 0$ for $j > t$).*

▶ **Lemma 41.** *There exists an optimal schedule $S = (L_k, B_k, R_k)_{1 \leqslant k \leqslant t}$ of a tree T that satisfies the following.*

1. *B_i is connected and there exists a vertex $u_i \in B_i$ adjacent to both L_i, R_i for all $i \in [t]$.*
2. *There are no empty forward trips i.e., $B_i \neq \varnothing$ for all odd $i \in [t]$.*
3. *There are no non-empty backward trips i.e., $B_i = \varnothing$ for all even $i \in [t]$.*

Proof. Let $\mathcal{S}$ be the set of optimal schedules that have no non-empty backward trips and no empty forward trips such that it satisfies bi-adjacency. By Lemmas 36 and 38, we know that the set $\mathcal{S}$ is non-empty. We define a ordering on the schedules of the set $\mathcal{S}$ as follows: Define $S_1 \leqslant S_2$ if $\omega(S_1) \leqslant_{lex} \omega(S_2)$ where $\leqslant_{lex}$ is the lexicographic ordering of positive integer sequences. We observe that the exclusion of non-empty backward trips and empty forward trips forces the sequences $\omega(S)$ for $S \in \mathcal{S}$ to be in $\mathbb{N}_{\geqslant 0}^{2n}$ where $n = |V(T)|$. Since the lexicographic ordering on $\mathbb{N}_{\geqslant 0}^{2n}$ is a well-ordering, we can find a lexicographically minimal $\omega(S_0)$ for some $S_0 \in \mathcal{S}$.

We claim that $S_0 = (L_k, B_k, R_k)_{1 \leqslant k \leqslant 2n}$ is the required schedule. By contradiction, assume that S_0 does not satisfy item (1) or S_0 is not the required schedule, then there exists a smallest i such that B_i has u_i adjacent to R_i and $v_i (\neq u_i)$ adjacent to L_i. Let P be unique path on the tree T from u_i to v_i. Let $C_1, \ldots, C_r$ be the connected components formed when u_i is deleted from B_i and adjust notation to let $v_i \in C_1$. We could break the forward trip B_i into two smaller forward trips $B'_i := u_i \cup_{j=2}^{r} C_j$ and $B''_i := C_1$ while keeping the other trips the same. It is a legal schedule as v_i and hence C_1 is adjacent to L_i and $C_2, \ldots, C_r$ are all adjacent to R_i in presence of u_i. This new schedule lies in $\mathcal{S}$ because it's demand is no more than $D(S_0)$. Moreover, it's weight sequence is lexicographically smaller than S_0, which is a contradiction. This proves our claim. ◀

▶ Remark 42. Note that the optimal schedule with the lexicographically minimal weight sequence has to start its trip by ferrying a leaf vertex.

The following lemma follows directly from the definition of demand of a schedule.

▶ **Lemma 43.** *Given a schedule* $S = (L_k, B_k, R_k)_{1 \leqslant k \leqslant t}$ *of a tree* T *such that* $B_i = \varnothing$ *for even* $i \in [t]$*, then*

$$D(S) = \max\{D(S(L_i)), |B_i|, D(S(R_i))\},$$

where $S(R_i) = (L_k \cap R_{i-1}, B_k, R_k)_{1 \leqslant k < i}$ *and* $S(L_i) = (L_k, B_k, R_k \cap L_i)_{i < k \leqslant t}$ *and all* $i \in [t]$*.*

We describe a tree transfer algorithm that works greedily.

Algorithm 1 Greedy Tree Transfer Algorithm.

Given a tree T on n vertices and a designated starting leaf ℓ:

1. **Initialize:** Transfer the leaf ℓ to the right bank. Set $b \leftarrow 1$.
2. **Repeat** until all vertices have been transferred:
   a. Let v be the unique vertex on the left bank that has a neighbor already on the right bank.
   b. Let $c_1, c_2, \ldots, c_k$ be the connected components of the left bank after removing v, ordered so that $|c_1| \leqslant |c_2| \leqslant \cdots \leqslant |c_k|$.
   c. If $k \leqslant 1$: Transfer v alone (boat size 1 suffices for this trip).
   d. If $k \geqslant 2$: Transfer v together with all vertices in $c_1, c_2, \ldots, c_{k-1}$. Update $b \leftarrow \max\big(b,\ 1 + \sum_{i=1}^{k-1} |c_i|\big)$.
3. **Return** b as the boat size required for this choice of starting leaf ℓ.

▶ **Theorem 44** (Optimal Tree Transfer)**.** *Let* T *be a tree. For each leaf* ℓ *of* T*, let* b_ℓ *denote the boat size computed by the Tree Transfer Algorithm starting from* ℓ*. Then* $\min_\ell b_\ell$ *is the minimum boat size required to transfer* T *while maintaining connectivity on both banks, and the corresponding schedule is optimal.*

Proof. We prove that the optimal schedule S_0 that satisfies the conditions of Lemma 41 has more demand than the greedy tree transfer algorithm. Let

$$S_0 = (L_k, B_k, R_k)_{1 \leqslant k \leqslant t}$$

where $t \leqslant 2n$ as before and let S_g be the schedule that is obtained by applying the greedy algorithm starting from the leaf in B_1 (see Remark 42). Let $i \in [2n]$ such that B_i was not chosen greedily. Let u_i be the vertex on B_i that is adjacent to both banks. Let us look at

L_{i-1}. Since there are no backward trips, it contains the vertex u_i. Let $C_1, \ldots, C_r$ be the components obtained when u_i is deleted from L_{i-1}. Let C_1 be the component with largest number of vertices and so, S_g would take $u_i \cup_{j=2}^{r} C_j$ instead of $u_i \cup_{j=1}^{r-1} C_j$ (WLOG, we let C_r be the one not taken by S_0). We will show that $D(S_0) \geqslant D(S_g)$ using Lemma 43.

To this end, we have to show that

$$\max\{D(S_0(L_i)), |u_i \cup_{j=1}^{r-1} C_j|, D(S_0(R_i))\} \geqslant \max\{D(S_g(L_i)), |u_i \cup_{j=2}^{r} C_j|, D(S_g(R_i))\}.$$

We know that $S_0(R_i) = S_g(R_i)$ and therefore, $D(S_0(R_i)) = D(S_g(R_i))$ as they do not differ before trip i. Furthermore, we have that $D(S_0(L_i)) \leqslant |C_r| \leqslant |C_1|$. Similarly, $D(S_g(L_i)) \leqslant |C_1|$ and finally, $|C_1| \leqslant |u_i \cup_{j=1}^{r-1} C_j|$. Therefore, the LHS of the inequality is either $D(S_0(R_i))$ or $|u_i \cup_{j=1}^{r-1} C_j|$. If the LHS were equal to $D(S_0(R_i)) = D(S_g(R_i))$, then the inequality follows as $D(S_0(R_i)) \geqslant |u_i \cup_{j=1}^{r-1} C_j|$ and so, the RHS is $D(S_g(R_i))$. Similarly, if the LHS is $|u_i \cup_{j=1}^{r-1} C_j|$, then inequality holds as this term is larger than every term on the RHS.

Therefore, we have shown that if we apply the greedy algorithm starting from the same vertex as the schedule S_0, we get a smaller demand than that of S_0. Since S_0 is optimal, it follows that the greedy algorithm is also optimal when starting from the appropriate vertex. If we were to apply the greedy algorithm from all possible leaves of the tree and find the minimum, we would an obtain an optimal schedule to ferry the tree under the given constraints. ◀

6 Concluding Remarks

We have initiated the study of Ferry Cover problems under connectivity constraints, departing from the classical requirement that banks remain independent (stable) sets. Our main contributions are:

1. **Hereditary properties.** We showed that the structural characterization of Csorba, Hurkens, and Woeginger [1] extends to any hereditary graph property Π: the boat size lies in $\{|Y|, |Y|+1\}$ where Y is a minimum Π-deletion set.
2. **Boat-1 characterization.** We completely characterized boat-1 graphs for the connected-bank variant: a connected graph is boat-1 if and only if its block-cut tree is a path. This yields a linear-time recognition algorithm.
3. **Trees.** We established that optimal schedules for trees need not have non-empty return trips, leading to an efficient algorithm for computing the minimum boat size.

We believe the connected-bank variant of Ferry Cover opens a rich landscape of combinatorial and algorithmic questions at the intersection of classical river-crossing puzzles and algorithmic and stuructural graph theory.

References

1 Peter Csorba, Cor A. J. Hurkens, and Gerhard J. Woeginger. The alcuin number of a graph and its connections to the vertex cover number. *SIAM Journal on Discrete Mathematics*, 24(3):757–769, 2010. doi:10.1137/090759987.

2 Hiro Ito, Stefan Langerman, and Yuichi Yoshida. Generalized river crossing problems. *Theory of Computing Systems*, 56(2):418–435, 2015. doi:10.1007/s00224-014-9557-1.

3 Michael Lampis and Valia Mitsou. The ferry cover problem. In *Fun with Algorithms, 4th International Conference, FUN 2007*, volume 4475 of *Lecture Notes in Computer Science*, pages 227–239. Springer, 2007. doi:10.1007/978-3-540-72914-3_21.

4 Abraham Lempel, Shimon Even, and Isaac Cederbaum. An algorithm for planarity testing of graphs. In *Theory of Graphs: International Symposium*, pages 215–232. Gordon and Breach, 1967.

5 Abbas Seify and Hossein Shahmohamad. Some new results in the alcuin number of graphs. *arXiv preprint arXiv:1409.6949*, 2014.

6 Erfang Shan and Liying Kang. The ferry cover problem on regular graphs and small-degree graphs. *Chinese Annals of Mathematics, Series B*, 39(6):933–946, 2018. doi:10.1007/s11401-018-0106-0.