Comparative Analysis of Discovered Models for the TCP Protocol

Experimental Report

December 22, 2025

1 Introduction

This report presents the Petri net models discovered for the TCP Protocol log. The behavior includes dual initialization (Active/Passive open), multiple phases, and resets returning to the "Closed" state.

The following sections present the model discovered by each algorithm individually, followed by a specific analysis of its structural and behavioral characteristics.

2 Discovered Models

2.1 Alpha Miner Family

Alpha Miner

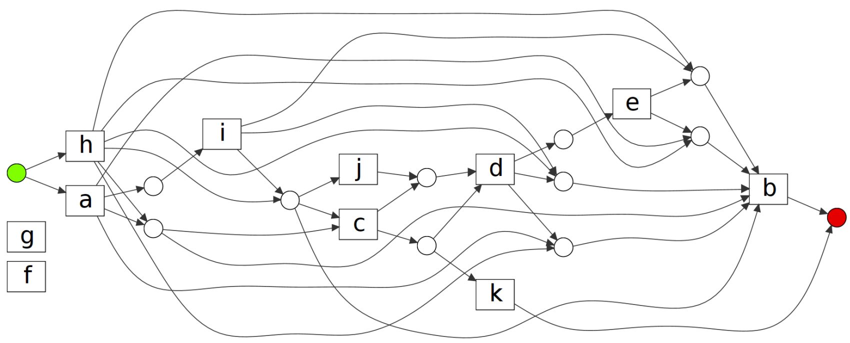

Figure 1: Model discovered by Alpha Miner.

The Alpha Miner model correctly identifies the start events (a and h) and connects the return events (b and k) to an end place. However, it presents events g and f disconnected (always enabled), representing a structural over-generalization. Furthermore, due to the multiple input places connected to transition b, the model imposes severe restrictions, making it impossible to execute valid sequences found in the log, such as ab or hb.

Alpha+ Miner

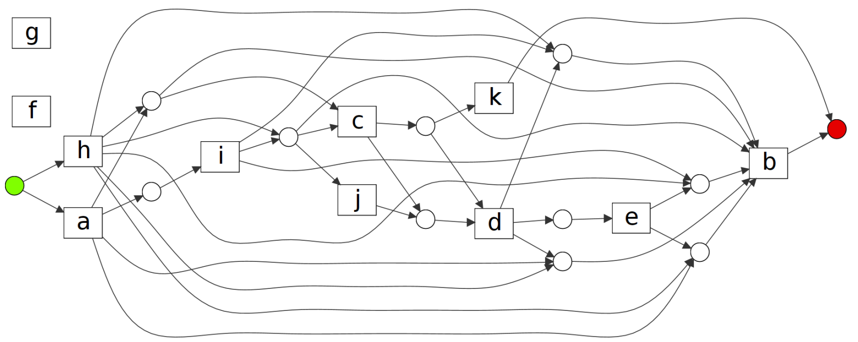

Figure 2: Model discovered by Alpha+ Miner.

Similar to the standard Alpha, the Alpha+ model identifies the start/end events but leaves g and f disconnected (over-generalization). The complexity of the connections creates significant behavioral blocks. For instance, due to the multiple input places connected to b, the sequence ab, as well as other sequences present in the log (e.g., ack, aib), cannot be executed.

Alpha++ Miner

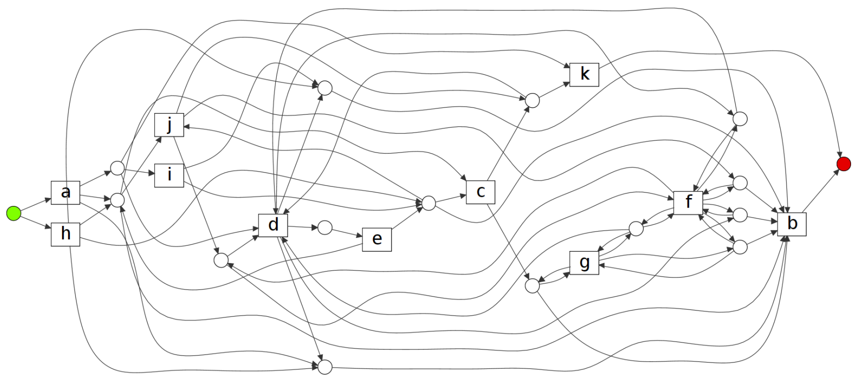

Figure 3: Model discovered by Alpha++ Miner.

The Alpha++ model properly identifies the start events (a, h) and connects the return events (b, k) to an end place. However, the complexity of the inferred dependencies imposes a series of behavioral restrictions. Specifically, the multiple input places connected to b prevent the execution of valid sequences found in the log, such as ab and aib.

Alpha$ Miner

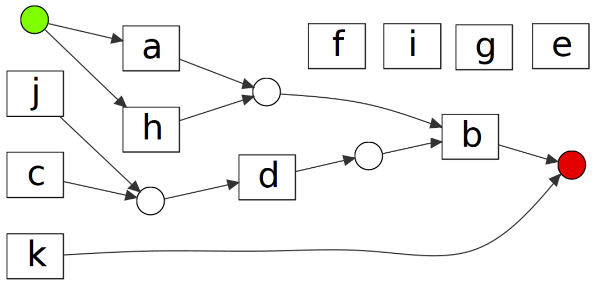

Figure 4: Model discovered by Alpha$ Miner.

The Alpha$ model correctly handles the start/end connections for a, h, b, and k. However, it presents a critical failure in internal structure: events c, e, f, g, i, j, and k appear without any input connections. This represents a total absence of execution constraints for these events (they are always enabled), implying an extreme over-generalization of the process behavior.

2.2 Heuristic and Inductive Approaches

Inductive Miner

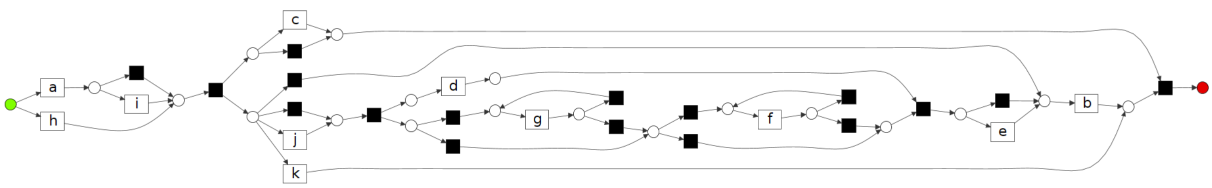

Figure 5: Model discovered by Inductive Miner.

The model correctly identifies a and h as start events, with b and k connecting to a place preceding a silent transition to 'end'. However, the silent transitions used to ensure fitness severely penalize **precision**. For example, a silent transition produces tokens enabling c, j, and k immediately after sequences like h, a, or ai. This erroneously allows sequences such as ak and hk (non-existent in the log), as well as af, ad, ae, among others. Thus, the silent transitions represent problematic generalizations allowing arbitrary event sequences.

Heuristics Miner

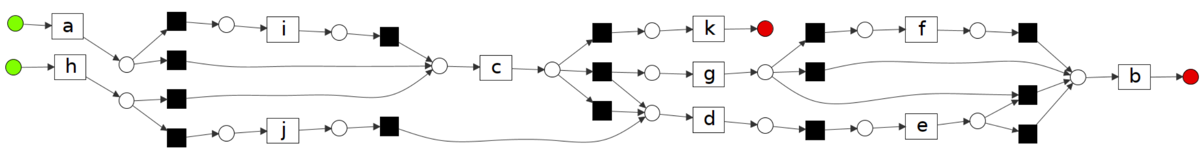

Figure 6: Model discovered by Heuristics Miner.

The model identifies a and h as start events but creates separate places for each, theoretically allowing duplicities (e.g., ah). Events b and k connect to distinct end places.

Despite using silent transitions, the model fails to replay valid sequences like ab or hb, and fails to capture contiguous repetitions of g and f existing in the log. Conversely, it allows invalid sequences such as $acgb, ahi$, and haj.

Fodina

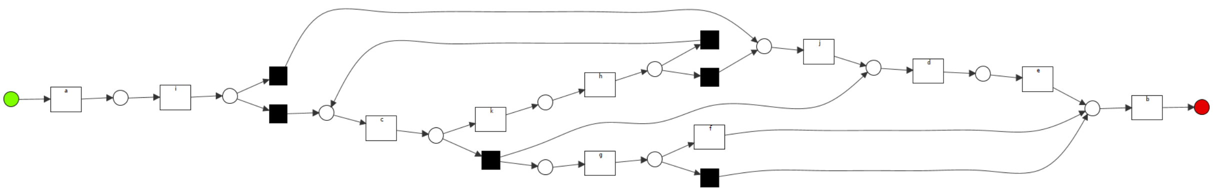

Figure 7: Model discovered by Fodina.

The model presents only a enabled at the start. Event b connects to an 'end' place, while k connects to a place enabling h. This implies an incorrect rigid logic where all connections must open with a and, if closed by k, the next instance **must** start with h (a restriction not present in the log). Furthermore, it restricts a to be followed only by i (excluding valid traces starting with ac) and allows only h after k.

2.3 Optimization-based Approaches

ILP Miner

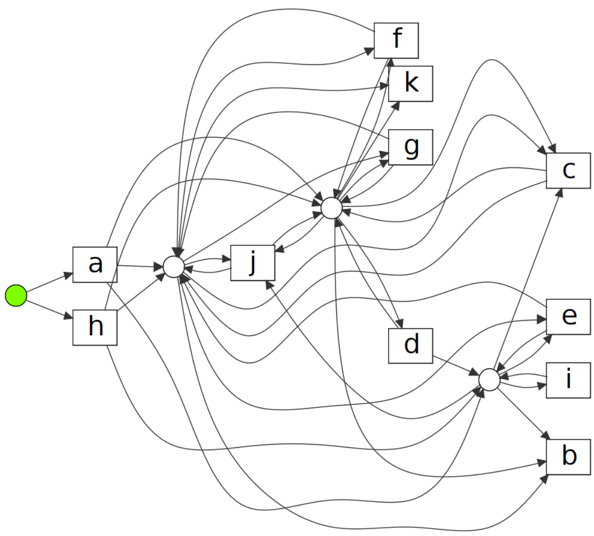

Figure 8: Model discovered by ILP Miner.

The ILP method identifies a and h as start events and represents b and k as consuming transitions (sinks). The initial execution enables c, d, e, f, g, i, j, k simultaneously. The model uses self-loops for most events (e, f, g, i) to keep them enabled. Event c acts as a pivot, but the event d creates a network imbalance: it is self-looped but also produces tokens for a place connected to c, e, i, b. Consequently, the network is highly generalist, permitting non-existent sequences such as repeated strings of d, i, or g $(\ldots dddd \ldots)$.

Evolutionary Tree Miner (ETMd)

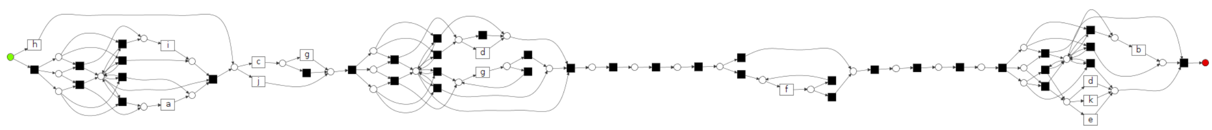

Figure 9: Model discovered by ETMd.

The network identifies h as a start transition but, via silent transitions, allows a and incorrectly i to execute initially. Visually, 5 blocks of events are separated by sync places. The first block mixes a, h, i. Subsequent blocks contain loops (e.g., g loop) allowing indefinite execution. A critical flaw is the complex structure of silent transitions that allows the process to "bypass" blocks (e.g., terminating immediately after c or j), and the incorrect assignment of d and e as termination events alongside b and k.