Hitting Families of Schedules

Dmitry Chistikov\textsuperscript{1,2}, Rupak Majumdar\textsuperscript{1}, Filip Niksic\textsuperscript{1}

\textsuperscript{1}Max Planck Institute for Software Systems (MPI-SWS), Germany
\textsuperscript{2}University of Oxford, UK
Ninjas at a conference banquet

A banquet is **complete** if for every pair of ninjas \((i, j)\), there’s a course served to ninja \(i\) before ninja \(j\).

**How many courses make a banquet complete?**
Two courses suffice:
Ninjas at a conference banquet

What if ninjas form a hierarchy?
A *master* is always served *before* their *student*.
Ninjas at a conference banquet

Again, two courses suffice:
Ninjas at a conference banquet

What if instead of pairs we consider **triplets** of ninjas?

A banquet is **3-complete** if for every triplet of ninjas \((i, j, k)\), there’s a course served to ninja \(i\) before \(j\), and \(j\) before \(k\).
Ninjas at a conference banquet

What if instead of pairs we consider *triplets* of ninjas?

A banquet is 3-*complete* if for every triplet of ninjas \((i, j, k)\), there’s a course served to ninja \(i\) before \(j\), and \(j\) before \(k\).
Ninjas at a conference banquet

What if instead of pairs we consider **triplets** of ninjas?

A banquet is **3-complete** if for every triplet of ninjas \((i, j, k)\), there’s a course served to ninja \(i\) before \(j\), and \(j\) before \(k\).

Naive approach with **2n** courses:

for each \(i \in \{1, \ldots, n\}\):
  serve ancestry line to \(i\); **ldfs** the rest
  serve ancestry line to \(i\); **rdfs** the rest
Ninjas at a conference banquet

What if instead of pairs we consider **triplets** of ninjas?

A banquet is **3-complete** if for every triplet of ninjas \((i, j, k)\), there’s a course served to ninja \(i\) before \(j\), and \(j\) before \(k\).

Naive approach with **2n** courses:

\[
\text{for each } i \in \{1, \ldots, n\}: \quad \text{serve ancestry line to } i; \quad \text{ldfs the rest} \\
\text{serve ancestry line to } i; \quad \text{rdfs the rest}
\]

Can be done with **\(O(\log n)\)** courses!
From ninjas to concurrent systems

- ninjas
- hierarchy
- courses
- d-complete banquet
- events
- partial order
- schedules
  - d-hitting family of schedules

Icons made by Freepik at www.flaticon.com
d-hitting families of schedules

Given a poset of events, a schedule hits a d-tuple of events \((e_1, \ldots, e_d)\) if it executes the events in the order \(e_1 < \ldots < e_d\).

Given a poset of events, a family of schedules \(F\) is d-hitting if for every admissible d-tuple of events there is a schedule in \(F\) that hits it.
Empirically: Many bugs involve small number of events—bug depth $d$
[Lu et al. ASPLOS ’08] [Burckhardt et al. ASPLOS ’10] [Jensen et al. OOPSLA ’15] [Qadeer et al. TACAS ’05]

• $d = 2$: order violation
• $d = 3$: atomicity violation

A $d$-hitting family of schedules provides a notion of coverage: it hits any bug of depth $d$.

Moreover, for certain kinds of partial orders we can explicitly construct small $d$-hitting families.
Contributions

- The notion of d-hitting families of schedules

- For antichains with n elements, existence of hitting families of size $O(\exp(d) \cdot \log n)$

- For trees of height h:
  - $d = 3$: explicit construction of hitting families of size $4h$ (optimal)
  - $d > 3$: explicit construction of hitting families of size $O(\exp(d) \cdot h^{d-1})$
Contributions

- The notion of d-hitting families of schedules

For antichains with n elements, existence of hitting families of size $O(\exp(d) \cdot \log n)$

For trees of height h:
- $d = 3$: explicit construction of hitting families of size $4h$ (optimal)
- $d > 3$: explicit construction of hitting families of size $O(\exp(d) \cdot h^{d-1})$
Why trees?

Trees arise from a simple fire-and-forget model of asynchronous programs.
3-hitting families for trees

admissible \((a, b, c)\)

height \(h\)
3-hitting families for trees

admissible \((a,b,c)\)

height \(h\)
3-hitting families for trees

admissible \((a,b,c)\)

d \(= \text{lca}(a,c)\)  
(could be \(a\) itself)
3-hitting families for trees

admissible \((a,b,c)\)

\(d = \text{lca}(a,c)\)  \(\text{(could be a itself)}\)
3-hitting families for trees

admissible \((a,b,c)\)

d = lca(a,c) \hspace{1em} \text{(could be a itself)}
3-hitting families for trees

Admissible \((a,b,c)\)

\(d = \text{lca}(a,c)\) (could be \(a\) itself)

dfs blocking right@\(i\); dfs the rest
3-hitting families for trees

admissible \((a,b,c)\)

d = \text{lca}(a,c) \quad \text{(could be} \ a \ \text{itself)}

dfs blocking right@\(i\); dfs the rest

dfs blocking left@\(i\); dfs the rest
3-hitting families for trees

admissible \((a,b,c)\)
\[d = \text{lca}(a,c)\]  (could be \(a\) itself)

dfs blocking right@\(i\); dfs the rest
dfs blocking left@\(i\); dfs the rest
3-hitting families for trees

admissible \((a,b,c)\)

d = lca(a,c)  \(\text{(could be } a\text{ itself)}\)

ldfs blocking right@i; ldfs the rest
ldfs blocking left@i; ldfs the rest
rdfs blocking right@i; rdfs the rest
rdfs blocking left@i; rdfs the rest
3-hitting families for trees

admissible \((a, b, c)\)

\(d = \text{lca}(a, c)\)  \((\text{could be } a \text{ itself})\)

for each \(i \in \{0, \ldots, h-1\}\):
  - \(\text{ldfs blocking right}@i; \text{ldfs the rest}\)
  - \(\text{ldfs blocking left}@i; \text{ldfs the rest}\)
  - \(\text{rdfs blocking right}@i; \text{rdfs the rest}\)
  - \(\text{rdfs blocking left}@i; \text{rdfs the rest}\)
3-hitting families for trees

admissible \((a,b,c)\)

\(d = \text{lca}(a,c)\)  (could be \(a\) itself)

for each \(i \in \{0, \ldots, h-1\}\):
- \(\text{ldfs blocking right}@i; \text{ldfs the rest}\)
- \(\text{ldfs blocking left}@i; \text{ldfs the rest}\)
- \(\text{rdfs blocking right}@i; \text{rdfs the rest}\)
- \(\text{rdfs blocking left}@i; \text{rdfs the rest}\)

Total: \(4h\) schedules

\((4 \cdot \log n\) for a balanced tree)
d-hitting families for $d \geq 4$

admissible $(x_1, \ldots, x_d)$

height $h$
d-hitting families for $d \geq 4$

admissible $(x_1, \ldots, x_d)$

$D = \text{lca-closure}(x_1, \ldots, x_d)$ (an ordered tree)
d-hitting families for $d \geq 4$

admissible $(x_1, \ldots, x_d)$

$D = \text{lca-closure}(x_1, \ldots, x_d)$ (an ordered tree)

$i_1, \ldots, i_k$ — levels of $D$'s internal nodes
d-hitting families for $d \geq 4$

admissible $(x_1,\ldots,x_d)$

$D = \text{lca-closure}(x_1,\ldots,x_d)$  \hspace{1cm} (an ordered tree)

$i_1,\ldots,i_k$ — levels of $D$'s internal nodes
d-hitting families for $d \geq 4$

admissible $(x_1,\ldots,x_d)$

$D = \text{lca-closure}(x_1,\ldots,x_d)$ (an ordered tree)

$i_1,\ldots,i_k$ — levels of $D$’s internal nodes

$\pi$ — schedule of $D$ that hits $(x_1,\ldots,x_d)$

$(D, i_1,\ldots,i_k, \pi)$ is a **pattern**:

- determines a partition of the tree
- by scheduling parts according to $\pi$, determines a schedule that hits $(x_1,\ldots,x_d)$
d-hitting families for $d \geq 4$

admissible $(x_1, \ldots, x_d)$

$D = \text{lca-closure}(x_1, \ldots, x_d)$ (an ordered tree)

$i_1, \ldots, i_k$ — levels of $D$'s internal nodes

$\pi$ — schedule of $D$ that hits $(x_1, \ldots, x_d)$

$(D, i_1, \ldots, i_k, \pi)$ is a pattern:

• determines a partition of the tree
• by scheduling parts according to $\pi$,
  determines a schedule that hits $(x_1, \ldots, x_d)$

for each pattern:
  schedule according to pattern
d-hitting families for $d \geq 4$

**Claim.** For any nodes $x_1, \ldots, x_d$, $|D| \leq 2d-1$. Moreover, $D$ has at most $d-1$ internal nodes.

**Accounting:**
- at most $\exp(d)$ ordered trees with $2d-1$ nodes
- at most $h^{d-1}$ choices for levels $i_1, \ldots, i_{d-1}$
- at most $d!$ schedules $\pi$

Total: at most $\exp(d) \cdot d! \cdot h^{d-1}$ patterns
**d-hitting families for \( d \geq 4 \)**

**Claim.** For any nodes \( x_1, \ldots, x_d \), \(|D| \leq 2d-1\). Moreover, \( D \) has at most \( d-1 \) internal nodes.

**Accounting:**
- at most \( \exp(d) \) ordered trees with \( 2d-1 \) nodes
- at most \( h^{d-1} \) choices for levels \( i_1, \ldots, i_{d-1} \)
- at most \( d! \) schedules \( \pi \)

Total: at most \( \exp(d) \cdot d! \cdot h^{d-1} \) patterns

**Note:** For \( d=3 \), this is \( O(h^2) \) instead of \( O(h) \) schedules
From hitting families to systematic testing

**Posets of events need not be static**
- Use on-the-fly constructions as a heuristic

**Beyond trees**
- Our results extend to series-parallel graphs
- In general, even the case of $d=2$ is difficult (order dimension [Dushnik & Miller, ’41])

**Unbalanced trees**
- Height $h$ can be close to number of nodes $n$
- Use domain-specific properties to first reduce the poset
Summary

- The notion of \(d\)-hitting families of schedules

- For antichains with \(n\) elements, existence of hitting families of size \(O(\exp(d) \cdot \log n)\)

- For trees of height \(h\):
  - \(d = 3\): explicit construction of hitting families of size \(4h\) (optimal)
  - \(d > 3\): explicit construction of hitting families of size \(O(\exp(d) \cdot h^{d-1})\)

http://www.mpi-sws.org/~fniksic/

Icons made by Freepik at www.flaticon.com