This cannot be the end!

Ekkehard Köhler

joint work with
Jesse Beisegel\textsuperscript{1}, Carolin Denkert\textsuperscript{1},
Matjaž Krnc\textsuperscript{2}, Nevena Mitrović\textsuperscript{2},
Robert Scheffler\textsuperscript{1}, Martin Strehler\textsuperscript{1}

\textsuperscript{1}BTU in Cottbus
\textsuperscript{2}University of Primorska in Koper

Paris, October 2018
How to start to find an end?
BFS — This cannot be the end(-vertex)!

Idea of BFS:
initialize queue \( Q = \{s\} \)
iternatively take top vertex \( v \) from \( Q \)
add all non-queue neighbors of \( v \) to end of \( Q \)

\[
Q = \{s\}; i=1;
\textbf{foreach} \ v \in Q \textbf{ do}
\begin{align*}
\sigma(i) & \leftarrow v; i++;
\textbf{foreach unvisited neighbor} \ w \text{ of } v \text{ with } w \notin Q \textbf{ do}
\text{append } w \text{ to } Q
\end{align*}
\textbf{end}
\textbf{end}

Example

• Can \( d \) be endvertex of a BFS?
  Yes! Any BFS starting at \( e \) ends at \( d \)

• Can \( c \) be endvertex of a BFS?
  No! all non-neighbors of \( c \)
have distance 2 from \( c \) but eccentricity 3
What is so great about ends?

- end-vertex of BFS: helpful for fast diameter computation
- end-vertex of LexBFS: is simplicial in chordal graphs
  key property for recognition and many opt. alg. in chordal graphs
- end-vertex of LexBFS in cocomparability graphs: always a source/sink in some transitive orientation
- end-vertex of LDFS in cocomparability graphs: start-vertex of hamiltonian path (if it exists)
- end-vertex of LexBFS in AT-free graphs: dominating pair vertex
- end-vertices of graph searches in general: their properties are the key for many multi-sweep algorithms on graphs
On the search for the right search
LexBFS — This cannot be the end(-vertex)!

Idea of LexBFS:
iteratively select vertex with lexicogr. largest label;
selected vertex appends number \((n - i)\) to label of neighbors

\[
\text{foreach } v \in V \text{ do } \text{label}(v) = 0; \\
\text{label}(s) = \{0\}; \quad n = |V| \\
\text{for } i \leftarrow 1 \text{ to } n \text{ do } \\
\quad v \leftarrow \text{unnumbered vertex with lexic. largest label } l(v); \\
\quad \sigma(i) \leftarrow v; \\
\quad \text{foreach unnumb. neighbor } w \text{ of } v \text{ do } \\
\quad \quad \text{append } (n - i) \text{ to } l(w) \\
\text{end}
\]

Theorem (Corneil, K., Lanlignel, 2010)
It is NP-hard to decide whether a given vertex is end-vertex of some LexBFS.

Example

- \(g\) is end-vertex of a BFS: acbdefg
- Can \(g\) be end-vertex of a LexBFS?
  No!
  The only non-neighbor of \(g\) is \(a\), thus have to start LexBFS at \(a\)
  However, by LeXBFS selection rule \(g\) will be visited before \(e\) and \(f\)
What makes this problem hard?
BFS — This cannot be the end(-vertex)!

Idea of BFS:
initialize queue \( Q = \{ s \} \)
iteratively take top vertex \( v \) from \( Q \)
add all non-queue neighbors of \( v \) to end of \( Q \)

---

Example

```
Q = \{s\}; i=1;
foreach v ∈ Q do
    σ(i) ← v; i++;
    foreach unvisited neighbor w of v with w ∉ Q do
        append w to Q
    end
end
```

Theorem (Charbit, Habib, Mamcarz '14)

*It is NP-hard to decide whether a given vertex is end-vertex of some BFS.*
## Known results for the end-vertex problem

<table>
<thead>
<tr>
<th></th>
<th>BFS</th>
<th>LBFS</th>
<th>DFS</th>
<th>LDFS</th>
<th>MCS</th>
<th>MNS</th>
<th>GenS</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Graphs</td>
<td>NPC</td>
<td>NPC</td>
<td>NPC</td>
<td>NPC</td>
<td>?</td>
<td>?</td>
<td>P</td>
</tr>
<tr>
<td>Weakly Chordal</td>
<td>NPC</td>
<td>NPC</td>
<td>NPC</td>
<td>NPC</td>
<td>?</td>
<td>?</td>
<td>P</td>
</tr>
<tr>
<td>Split</td>
<td>P</td>
<td>P</td>
<td>NPC</td>
<td>P</td>
<td>?</td>
<td>P</td>
<td>P</td>
</tr>
</tbody>
</table>

- Corneil, Köhler, Lanlignel, 2010
- Berry, Blair, Bordat, Simonet, 2010
- Charbit, Habib, Mamcarz, 2014
- Kratsch, Liedloff, Meister 2015: Exact algorithms for BFS and DFS \(O^*(2^n)\); strong connection between end-vertex for DFS and Hamiltonian path problem

Our work:

- maximal neighborhood search (MNS)
- maximum cardinality search (MCS)
On the search for the right search

Generic Search

- BFS
- LexBFS

- MNS
- MCS

- DFS
- LexDFS
MNS — Maximal Neighborhood Search

Idea of MNS:
At each step pick: a vertex whose set of neighbors already explored is maximal with respect to set inclusion.

\[
\begin{align*}
\text{foreach } v \in V & \text{ do label}(v) = \emptyset; \\
\text{label}(s) & = \{0\}; \\
\text{for } i \leftarrow 1 \text{ to } n & \text{ do } \\
& v \leftarrow \text{unnumbered vertex with inclusion maximal label } l(v); \\
& \sigma(i) \leftarrow v; \\
& \text{foreach unnumb. neighbor } w \text{ of } v \text{ do } \\
& \quad \text{append } i \text{ to } l(w) \\
\end{align*}
\]

→ every LexBFS and every LexDFS is an MNS
→ if $G$ chordal then MNS perfect elim. ordering

How Hard is end-vertex for MNS?
Charbit, Habib, Mamcarz: end-vertex problem for MNS open
(in their complexity table: ?(P) for this problem)
MNS — How Hard is End-Vertex Problem?

Theorem

The end-vertex problem for MNS is \textit{NP-complete}.

Proof: (reduction from 3-SAT).

Let $\mathcal{I}$ be instance of 3-SAT:
- variables of $\mathcal{I}$: $x_1, \ldots, x_n$
- clauses of $\mathcal{I}$: $C_1, \ldots, C_k$

construct graph $G(\mathcal{I}) = (V, E)$ as follows:
- literal vertices $x_i, \overline{x_i}$: compl. of perf. matching
- clause vertices $c_j$: independent set
- additional vertices $b, s, t$
- $c_j$: adjacent to all literal vertices except “its own”
- $b$: adjacent to all literal vertices
- $s$: adjacent to all vertices but $b$ and $t$
- $t$: adjacent to all but $s$
**MNS — NP-Completeness Proof**

**Question**
Can $t$ be end-vertex of an MNS?

**Observations**
- If MNS chooses a clause vertex before vertex $b$ or $b$ before $s$, then $t$ cannot be the end-vertex.
- For $t$ being the end-vertex of MNS, the algorithm has to choose $s$ and an assignment right at the beginning.
- For each instance $\mathcal{I}$ of 3-SAT, the graph $G(\mathcal{I})$ is weakly chordal.

**Theorem**

*3-SAT instance has satisfying assignment iff $t$ is end-vertex of an MNS in $G$*
On the search for the right search

→ every MNS for a chordal graph is a perfect elimination ordering
MCS — Maximum Cardinality Search

Idea of MCS:
At each step: pick a vertex whose set of neighbors already explored is maximal with respect to cardinality.

```
foreach v ∈ V do label(v) = 0;
label(s) = \{0\};
for i ← 1 to n do
    v ← unnumbered vertex with label l(v) that has maximum cardinality;
    σ(i) ← v;
    foreach unnumb. neighbor w of v do
        append i to l(w)
    end
end
```

How hard is end-vertex problem of MCS?
Nothing known up to now.

Example
resulting MCS ordering: adcgfb
MCS — How Hard is End-Vertex Problem?

Theorem
The end-vertex problem for MCS is \textit{NP-complete}.

Idea of proof: (reduction from 3-SAT).

- If 3-SAT instance has satisfying assignment then exists MCS ordering with $t$ being end-vertex.

- 3-SAT satisfying assignment iff $t$ is end-vertex of MCS
MCS — End-Vertex in Split Graphs

Theorem

\( G = (C \cup I, E) \) split graph (\( C \) maximal clique, \( I \) indep. set).

\( t \in V \) is end-vertex of some MCS ordering \( \sigma \) iff (I) \( t \) is simplicial and (II) the neighborhoods of the vertices with a smaller degree than \( t \) are totally ordered by inclusion.

Idea of proof. \( \Rightarrow \).

Assume \( t \) is the last vertex in \( \sigma \).

(I) \( G \) is split, thus it is chordal. Hence, end-vertex of MCS is simplicial [Tarjan, Yannakakis, 1984]

(II) Assume there exist \( u, v \) with smaller degree than \( t \) such that \( N(u) \not\subseteq N(v) \) and \( N(v) \not\subseteq N(u) \).

- can show that \( u, v \in I \)
- wlog \( v \) taken before \( u \) in MCS
- \( \Rightarrow \) all neighbors of \( v \) visited before \( u \)
- as \( N(u), N(v) \) incomparable: \( \exists w \in N(v) \setminus N(u) \)
- \( \Rightarrow \) all vertices of \( C \) visited before \( u \)
- now label of \( t \) larger than label of \( u \), thus \( t \) chosen before \( u \)
- Contradiction!
MCS — End-Vertex in Split Graphs

Theorem

$G = (C \cup I, E)$ split graph ($C$ maximal clique, $I$ indep. set).

$t \in V$ is end-vertex of some MCS ordering $\sigma$ iff (I) $t$ is simplicial and (II) the neighborhoods of the vertices with a smaller degree than $t$ are totally ordered by inclusion.

Idea of proof. $\Leftarrow$.

- $t$ be simplicial and neighborhoods of vertices with smaller degree than $t$ totally ordered by set inclusion.
- Define $U := \{ u \in V \mid d(u) < d(t) \}$.
- $t$ simplicial $\Rightarrow U \subseteq I$.
- take the neighborhoods of all vertices $u \in U$ in the order of the inclusion ordering.
- Every time the complete neighborhood of a vertex $u$ is taken: take $u$. $\Rightarrow$ all these vertices are taken before $t$.
- If $t \in C$: then $U = I$ and we can take the remaining vertices of $C$ in an arbitrary ordering, with $t$ being last.
- If $t \in I$: we first take the remaining vertices of $C$. Since the neighborhood of $t$ is not greater than the neighborhood of all remaining vertices, we can take $t$ as last vertex.
MCS — End-Vertex in (Unit) Interval graphs

Theorem (Gilmore and Hoffman, 1964)
A graph $G$ is an interval graph if and only if the maximal cliques of $G$ can be linearly ordered such that, for every vertex $v \in G$, the maximal cliques containing $v$ occur consecutively.

Lemma
Let $G = (V, E)$ be an interval graph and let $C_1, C_2, \ldots, C_k$ be linear order of the maximal cliques of $G$. Then $t \in V$ is the last vertex of some MCS-ordering $\sigma$ of $G$ if:

1. $t$ is simplicial, and
2. If $C_i$ is the unique clique containing $t$, then $i = 1$ or $i = k$ or $C_{i-1} \cap C_i \subseteq C_i \cap C_{i+1}$ and $|C_i \cap C_{i+1}| \leq |C_j \cap C_{j+1}|$ for all $j > i$, or the same holds for the reverse order $C_k, C_{k-1}, \ldots, C_1$.

Theorem
Let $G = (V, E)$ be a unit interval graph and let $C_1, C_2, \ldots, C_k$ be linear order of the maximal cliques of $G$. Then $t \in V$ is the last vertex of some MCS-ordering $\sigma$ of $G$ iff:

1. $t$ is simplicial, and
2. If $C_i$ is the unique clique containing $t$, then $i = 1$ or $i = k$.  

Conclusions

- Finding an end is surprisingly difficult
- MCS is a quite challenging search for end-vertex problem
- These simple graph searches are not well understood.
- Big step: repeated application of (LBFS) searches (Dusart and Habib)
- Need more and better tools for analysing search algorithms
- A lot of work has to be done here

... and there is no time for retirement!

Thank you Michel and lets keep working!