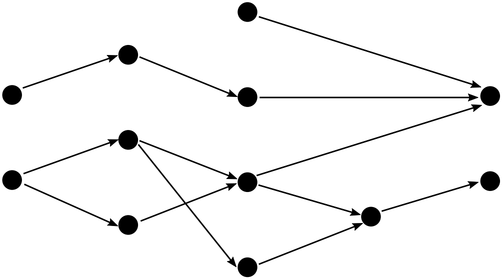
Sparsifying, Shrinking and Splicing for Minimum Path Cover in Parameterized Linear Time

Manuel Cáceres, Massimo Cairo, **Brendan Mumey**,
Romeo Rizzi and Alexandru I. Tomescu

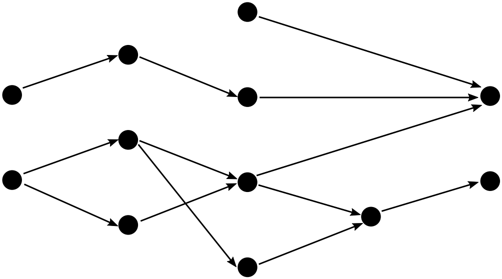
January 2022, SODA

Problem & Motivation

MPC of a DAG

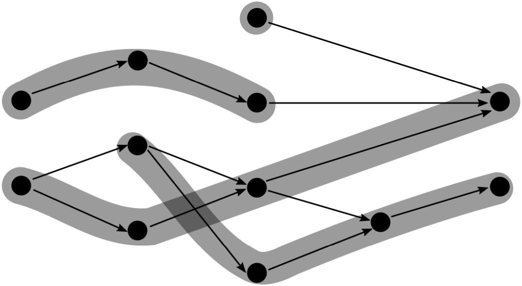


MPC of a DAG



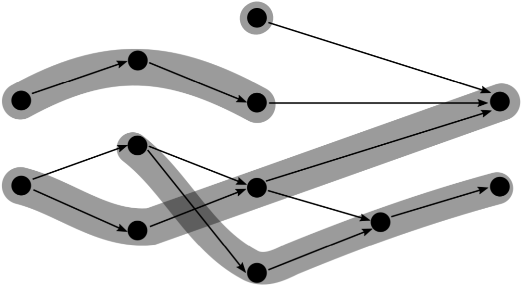
A minimum-sized set of paths such that every vertex appears in at least one path of the set

MPC of a DAG



A minimum-sized set of paths such that every vertex appears in at least one path of the set

MPC of a DAG



A minimum-sized set of paths such that every vertex appears in at least one path of the set

→ Solvable in polynomial time [9, 11]

Motivation

Applications in various fields

- Bioinformatics
  - Multi-assembly [10, 27, 25, 4, 19]
  - Perfect phylogeny haplotyping [1, 13]
  - Pan-genome alignment [22, 20]

Motivation

Applications in various fields

- Bioinformatics
  - Multi-assembly [10, 27, 25, 4, 19]
  - Perfect phylogeny haplotyping [1, 13]
  - Pan-genome alignment [22, 20]

- Scheduling [7, 8, 3, 28, 29, 23], computational logic [2, 12], distributed computing [26, 15], databases [16], evolutionary computation [17], program testing [24], cryptography [21], programming languages [18]

Motivation

Applications in various fields

- Bioinformatics
  - Multi-assembly [10, 27, 25, 4, 19]
  - Perfect phylogeny haplotyping [1, 13]
  - Pan-genome alignment [22, 20]

- Scheduling [7, 8, 3, 28, 29, 23], computational logic [2, 12], distributed computing [26, 15], databases [16], evolutionary computation [17], program testing [24], cryptography [21], programming languages [18]

$\longrightarrow$Since the size k of an MPC (*width*) is small, research has focused in solutions parameterized by k

Previous work

MPC algorithms

Reduction to Maximum Matching [11]

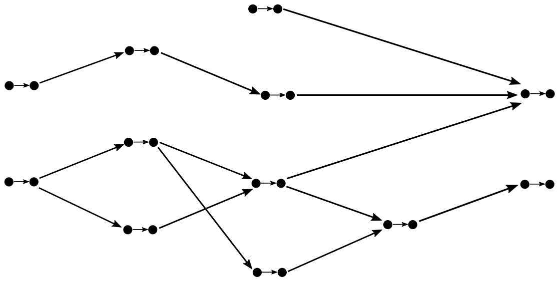
- $O(\sqrt{|V|}|E|)$ [14] $\rightarrow$ transitive DAGs
- $O(|V|^2 + k\sqrt{k}|V|)$ and $O(\sqrt{|V|}|E| + k\sqrt{k}|V|)$ [5, 6]

MPC algorithms

Reduction to Maximum Matching [11]

- $O(\sqrt{|V|}|E|)$ [14] $\rightarrow$ transitive DAGs
- $O(|V|^2 + k\sqrt{k}|V|)$ and $O(\sqrt{|V|}|E| + k\sqrt{k}|V|)$ [5, 6]

Reduction to Minimum Flow [24]

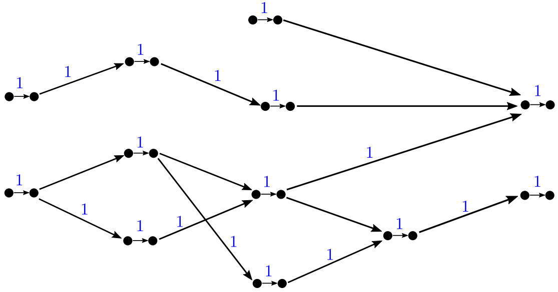


$$f(v^{in}, v^{out}) \geq d(v^{in}, v^{out}) = 1$$

Reduction to Maximum Matching [11]

- $O(\sqrt{|V|}|E|)$ [14] → transitive DAGs
- $O(|V|^2 + k\sqrt{k}|V|)$ and $O(\sqrt{|V|}|E| + k\sqrt{k}|V|)$ [5, 6]

Reduction to Minimum Flow [24]



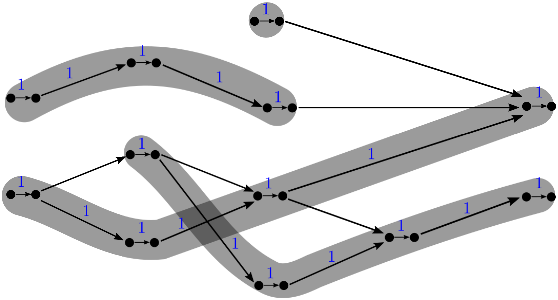
$f(v^{in}, v^{out}) \geq d(v^{in}, v^{out}) = 1$

MPC algorithms

Reduction to Maximum Matching [11]

- $O(\sqrt{|V|}|E|)$ [14] → transitive DAGs
- $O(|V|^2 + k\sqrt{k}|V|)$ and $O(\sqrt{|V|}|E| + k\sqrt{k}|V|)$ [5, 6]

Reduction to Minimum Flow [24]



$f(v^{in}, v^{out}) \geq d(v^{in}, v^{out}) = 1$

MPC algorithms

Reduction to Maximum Matching [11]

- $O(\sqrt{|V|}|E|)$ [14] $\rightarrow$ transitive DAGs
- $O(|V|^2 + k\sqrt{k}|V|)$ and $O(\sqrt{|V|}|E| + k\sqrt{k}|V|)$ [5, 6]

Reduction to Minimum Flow [24]

- $O(f(G, |V|))$: maximum flow on G, capacities $\leq |V|$
- $O(k(|V| + |E|) \log |V|)$ [22]

MPC algorithms

Reduction to Maximum Matching [11]

- $O(\sqrt{|V|}|E|)$ [14] $\rightarrow$ transitive DAGs
- $O(|V|^2 + k\sqrt{k}|V|)$ and $O(\sqrt{|V|}|E| + k\sqrt{k}|V|)$ [5, 6]

Reduction to Minimum Flow [24]

- $O(f(G, |V|))$: maximum flow on G, capacities $\leq |V|$
- $O(k(|V| + |E|) \log |V|)$ [22]

None of them reaches parameterized linear time

Our results

MPC in parameterized linear time

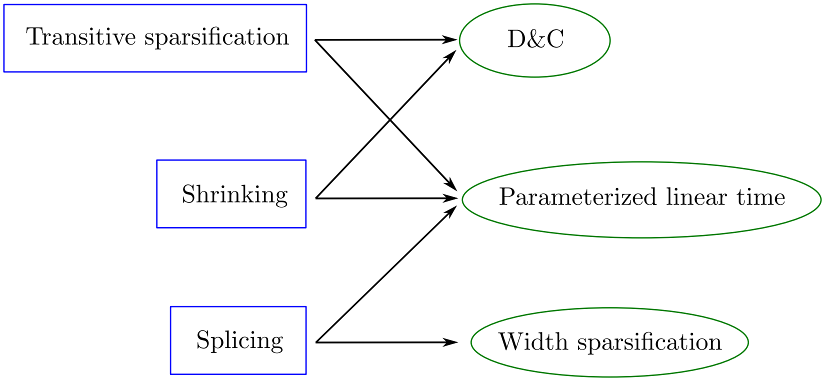
- A simple D&C algorithm
  - $O(k^2|V|\log|V| + |E|)$
  - $O(k^2|V| + |E|)$ in PRAM
- The first parameterized linear time algorithm
  - $O(k^3|V| + |E|)$
- Width sparsification of edges to $< 2|V|$
  - $O(k^3|V| + |E|)$

MPC in parameterized linear time

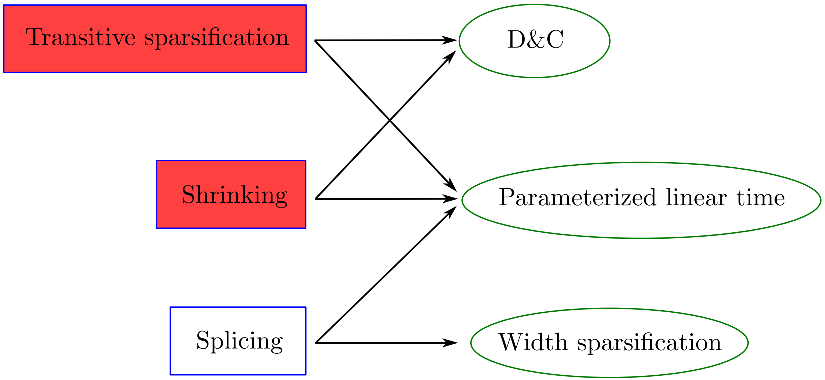
- A simple D&C algorithm
  - $O(k^2|V|\log|V| + |E|)$
  - $O(k^2|V| + |E|)$ in PRAM
- The first parameterized linear time algorithm
  - $O(k^3|V| + |E|)$
- Width sparsification of edges to $< 2|V|$
  - $O(k^3|V| + |E|)$

$\longrightarrow$At the core of our solutions we use:
Transitive sparsification, shrinking and splicing

Roadmap

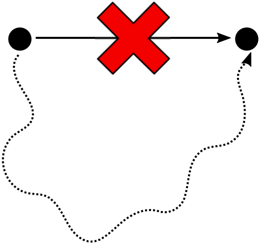

Transitive sparsification
Shrinking
Splicing
D&C
Parameterized linear time
Width sparsification

Roadmap



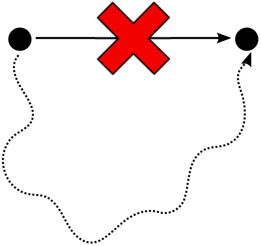
Transitive sparsification
Shrinking
Splicing
D&C
Parameterized linear time
Width sparsification

Transitive sparsification



A spanning subgraph S preserving the reachability relation between its vertices

Transitive sparsification



A spanning subgraph S preserving the reachability relation between its vertices
$\Rightarrow$

- $\mathsf{width}(S) = \mathsf{width}(G)$
- Every path cover of S is path cover of G

Transitive sparsification

Inspired by Jagadish's work [16], we propose a transitive sparsification algorithm based on a path cover $\mathcal{P}$ of size t

Observation 2.1

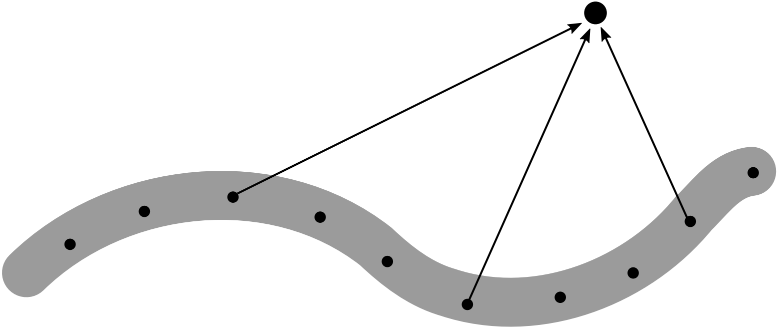
We sparsify the incoming edges of v to $\leq t$ in $O(t + |N^{-}(v)|)$

Transitive sparsification

Inspired by Jagadish's work [16], we propose a transitive sparsification algorithm based on a path cover $\mathcal{P}$ of size t

Observation 2.1

We sparsify the incoming edges of v to $\leq t$ in $O(t + |N^{-}(v)|)$

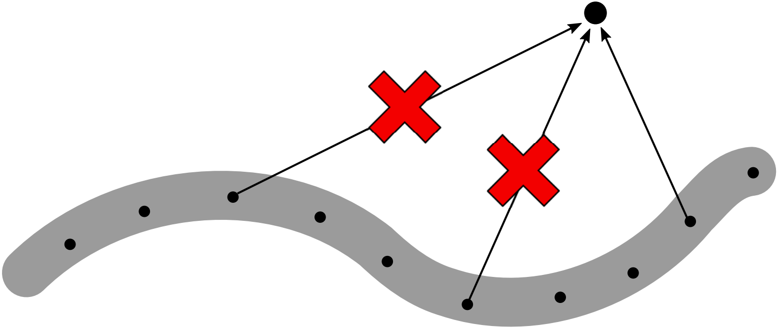


Transitive sparsification

Inspired by Jagadish's work [16], we propose a transitive sparsification algorithm based on a path cover $\mathcal{P}$ of size t

Observation 2.1

We sparsify the incoming edges of v to $\leq t$ in $O(t + |N^-(v)|)$



Shrinking

Shrink a path cover $\mathcal{P}$ of size t in an MPC

Can be solved by using the flow reduction:

- Interpret $\mathcal{P}$ as a flow
- Find $\leq t - k$ decrementing paths
- Extract the MPC

Shrinking

Shrink a path cover $\mathcal{P}$ of size t in an MPC

Can be solved by using the flow reduction:

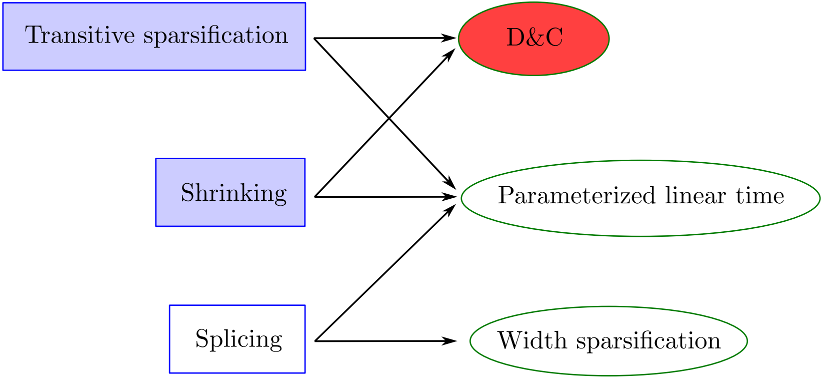
- Interpret $\mathcal{P}$ as a flow
- Find $\leq t - k$ decrementing paths
- Extract the MPC

Lemma 2.5

We can obtain an MPC of G in time $O(t(|V| + |E|))$

$\longrightarrow$ Generalization of approach used by Mäkinen et.al [22]

Roadmap



Transitive sparsification
D&C
Shrinking
Parameterized linear time
Splicing
Width sparsification

D&C Algorithm

- Compute a **topological order** of the vertices $v_1, \ldots, v_{|V|}$

D&C Algorithm

- Compute a **topological order** of the vertices $v_1, \ldots, v_{|V|}$
- Run the following **recursive** algorithm. Starting from G, on the subgraph $G_{i,j}[\{v_i, \ldots, v_j\}]$:

D&C Algorithm

- Compute a **topological order** of the vertices $v_1, \ldots, v_{|V|}$
- Run the following **recursive** algorithm. Starting from G, on the subgraph $G_{i,j}[\{v_i, \ldots, v_j\}]$:
  - Solve recursively on $G_\ell = (V_\ell, E_\ell)$, induced by $v_i, \ldots, v_{(j-i+1)/2}$. Obtaining a MPC $P_1^\ell, \ldots, P_{k_\ell}^\ell$
  - Solve recursively on $G_r = (V_r, E_r)$, induced by $v_{(j-i+1)/2}, \ldots, v_j$. Obtaining a MPC $P_1^r, \ldots, P_{k_r}^r$

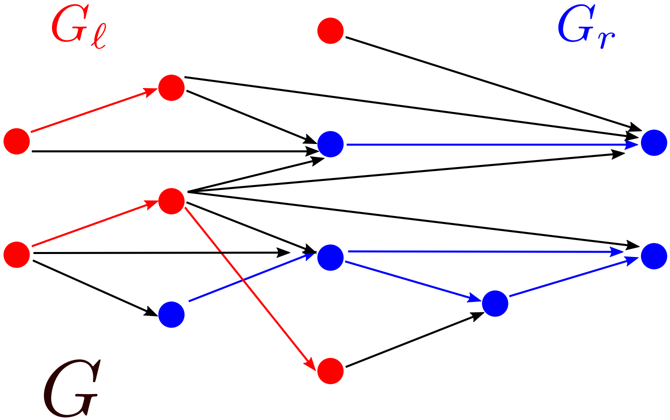
D&C Algorithm

- Compute a **topological order** of the vertices $v_1, \ldots, v_{|V|}$
- Run the following **recursive** algorithm. Starting from G, on the subgraph $G_{i,j}[\{v_i, \ldots, v_j\}]$:
  - Solve recursively on $G_\ell = (V_\ell, E_\ell)$, induced by $v_i, \ldots, v_{(j-i+1)/2}$. Obtaining a MPC $P_1^\ell, \ldots, P_{k_\ell}^\ell$
  - Solve recursively on $G_r = (V_r, E_r)$, induced by $v_{(j-i+1)/2}, \ldots, v_j$. Obtaining a MPC $P_1^r, \ldots, P_{k_r}^r$
  - Obtain **sparsification** of $G_{i,j}$ using the path cover $P_1^\ell, \ldots, P_{k_\ell}^\ell, P_1^r, \ldots, P_{k_r}^r$

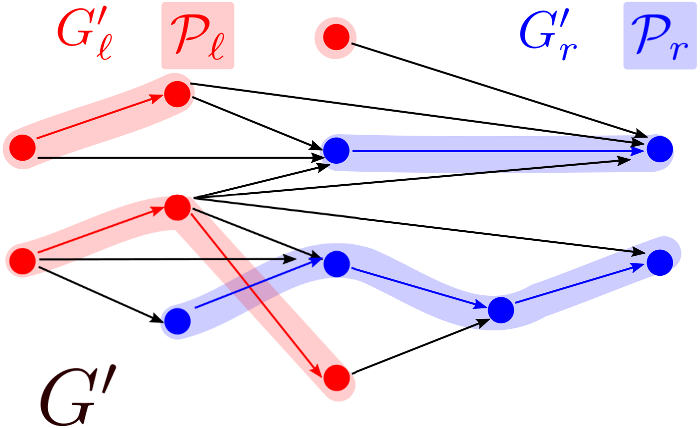
D&C Algorithm

- Compute a **topological order** of the vertices $v_1, \ldots, v_{|V|}$
- Run the following **recursive** algorithm. Starting from G, on the subgraph $G_{i,j}[\{v_i, \ldots, v_j\}]$:
  - Solve recursively on $G_\ell = (V_\ell, E_\ell)$, induced by $v_i, \ldots, v_{(j-i+1)/2}$. Obtaining a MPC $P^\ell_1, \ldots, P^\ell_{k_\ell}$
  - Solve recursively on $G_r = (V_r, E_r)$, induced by $v_{(j-i+1)/2}, \ldots, v_j$. Obtaining a MPC $P^r_1, \ldots, P^r_{k_r}$
  - Obtain **sparsification** of $G_{i,j}$ using the path cover $P^\ell_1, \ldots, P^\ell_{k_\ell}, P^r_1, \ldots, P^r_{k_r}$
  - **Shrink** the path cover solution to $P_1, \ldots, P_k$

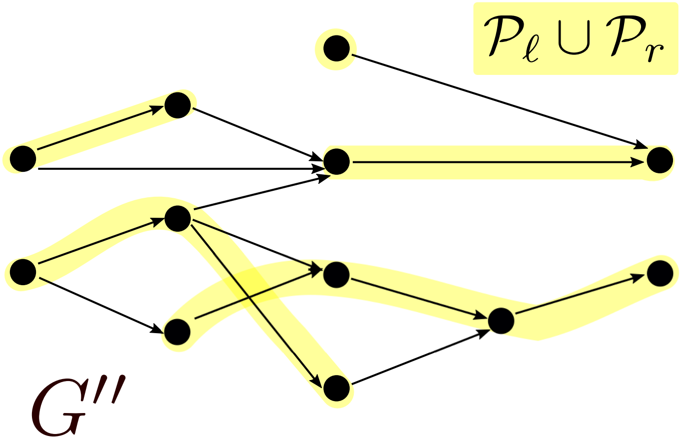
An example - Division

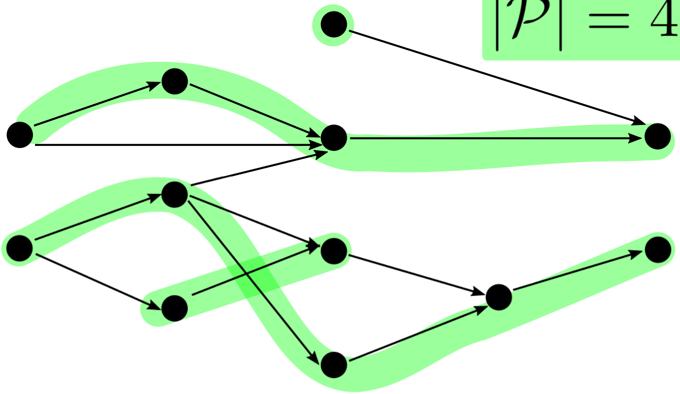


An example - Recursion



An example - Sparsification





$|\mathcal{P}| = 4$

D&C algorithm

Theorem 1.1

We compute an MPC in time $O(k^2|V|\log|V| + |E|)$

D&C algorithm

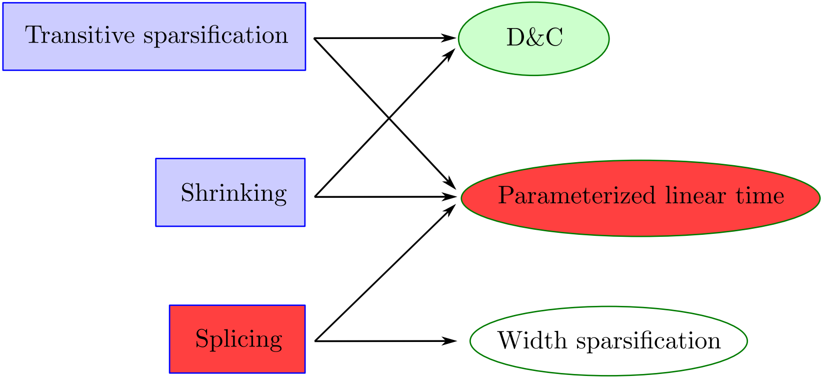
Theorem 1.1

We compute an MPC in time $O(k^2|V|\log|V| + |E|)$

Theorem 1.2

We compute an MPC in $O(k^2|V| + |E|)$ *parallel steps using* $O(\log|V|)$ *single processors in the PRAM model*

Roadmap


Transitive sparsification
Shrinking
Splicing
D&C
Parameterized linear time
Width sparsification

Parameterized linear time

High-level approach:

- Process vertices in topological order $v_1, \ldots, v_{|V|}$
- At each step compute an MPC $\mathcal{P}_i$ of G_i
- In step $i+1$ consider the path cover $\mathcal{T}_{i+1} = \mathcal{P}_i \cup \{v_{i+1}\}$, and **shrink** it to $\mathcal{P}_{i+1}$

Parameterized linear time

High-level approach:

- Process vertices in topological order $v_1, \ldots, v_{|V|}$
- At each step compute an MPC $\mathcal{P}_i$ of G_i
- In step $i+1$ consider the path cover $\mathcal{T}_{i+1} = \mathcal{P}_i \cup \{v_{i+1}\}$, and **shrink** it to $\mathcal{P}_{i+1}$

$\longrightarrow$ We work directly with the *flow reduction* $\mathcal{G}_i$ of G_i, and at each step we look for a *decrementing path* in the residual network $\mathcal{R}(\mathcal{G}_{i+1}, \mathcal{T}_{i+1})$

Parameterized linear time

At step $i+1$:

1. **Sparsify** the edges incoming to v_{i+1}^{in} using $\mathcal{P}_i$
   - Ensures $O(k)$ out-neighbors in $\mathcal{R}(\mathcal{G}_{i+1}, \mathcal{T}_{i+1})$
2. Layered traversal of $\mathcal{R}(\mathcal{G}_{i+1}, \mathcal{T}_{i+1})$
3. If a decrementing path D is found, **splice** $\mathcal{T}_{i+1}$ along D to get $\mathcal{P}_{i+1}$. Otherwise, $\mathcal{P}_{i+1} \leftarrow \mathcal{T}_{i+1}$
4. Update level of vertices

Layered Traversal

We maintain:

- Level assignment ℓ of the vertices of $\mathcal{G}_i$ to $\{0, 1, \ldots, \mathsf{width}(G_i)\}$. Partition of vertices into *layers*
- **Invariant A**: For each (u, v) in $\mathcal{R}(\mathcal{G}_i, \mathcal{P}_i)$, $\ell(u) \geq \ell(v)$
- **Invariants B and C**

Layered Traversal

We maintain:

- Level assignment ℓ of the vertices of $\mathcal{G}_i$ to $\{0, 1, \ldots, \mathsf{width}(G_i)\}$. Partition of vertices into *layers*
- **Invariant A**: For each (u, v) in $\mathcal{R}(\mathcal{G}_i, \mathcal{P}_i)$, $\ell(u) \geq \ell(v)$
- **Invariants B and C**

Main idea:

BFS in each reachable layer from highest to lowest

- Stop when reaching t
- Only continues to the next highest reachable layer once all reachable vertices from the current layer have been visited

Splicing

Reconnecting paths in a path cover $\mathcal{P}$ so that one follows a certain path D

Splicing

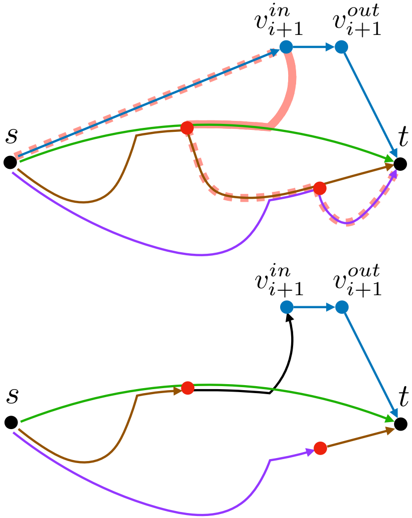
Reconnecting paths in a path cover $\mathcal{P}$ so that one follows a certain path D

- Requires edges in D covered by $\mathcal{P}$
- Preserves covering of vertices, size of path cover, and multiplicity of edges

Lemma 2.6

We can obtain, in $O(|D|)$, a path cover $\mathcal{P}'$ of the same size such that $\mu_{\mathcal{P}}(e) = \mu_{\mathcal{P}'}(e)$, and there exists $P \in \mathcal{P}'$ containing D

Splice $\mathcal{T}_{i+1}$ along D



Transform $\mathcal{T}_{i+1}$ into $\mathcal{P}_{i+1}$ in $O(|D|)$

Level update

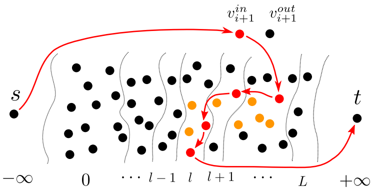
If l is the smallest layer visited by the traversal, we set:

- $\ell(v_{i+1}^{in}) = l$, $\ell(v_{i+1}^{out}) = l + 1$
- For each u visited, $\ell(u) = l$

Level update

If l is the smallest layer visited by the traversal, we set:

- $\ell(v_{i+1}^{in}) = l$, $\ell(v_{i+1}^{out}) = l + 1$
- For each u visited, $\ell(u) = l$



Parameterized linear time

We show that:

- Invariants are maintained
- A step takes $O(|N^-(v_{i+1})|)$ and $O(k)$ per vertex of level at least l
- Each vertex is charged $O(k^2)$ times during the algorithm

Parameterized linear time

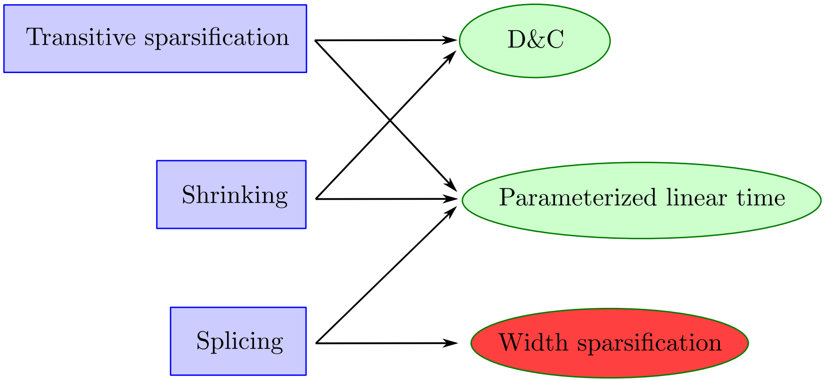
We show that:

- Invariants are maintained
- A step takes $O(|N^-(v_{i+1})|)$ and $O(k)$ per vertex of level at least l
- Each vertex is charged $O(k^2)$ times during the algorithm

Theorem 1.3

We compute an MPC in time $O(k^3|V| + |E|)$

Roadmap


Transitive sparsification
Shrinking
Splicing
D&C
Parameterized linear time
Width sparsification

Width sparsification

We show the following result for a path cover of size t

Theorem 1.4

We compute, in $O(t^2|V|)$ time, a path cover $\mathcal{P}', |\mathcal{P}'| = t$, whose number of distinct edges is less than $2|V|$

Width sparsification

We show the following result for a path cover of size t

Theorem 1.4

We compute, in $O(t^2|V|)$ time, a path cover $\mathcal{P}', |\mathcal{P}'| = t$, whose number of distinct edges is less than $2|V|$

Thus we obtain

Corollary 1.1

We compute a spanning subgraph $G' = (V, E')$ of G with $|E'| < 2|V|$ and width k in time $O(k^3|V| + |E|)$.

Width sparsification

We show the following result for a path cover of size t

Theorem 1.4

We compute, in $O(t^2|V|)$ time, a path cover $\mathcal{P}', |\mathcal{P}'| = t$, whose number of distinct edges is less than $2|V|$

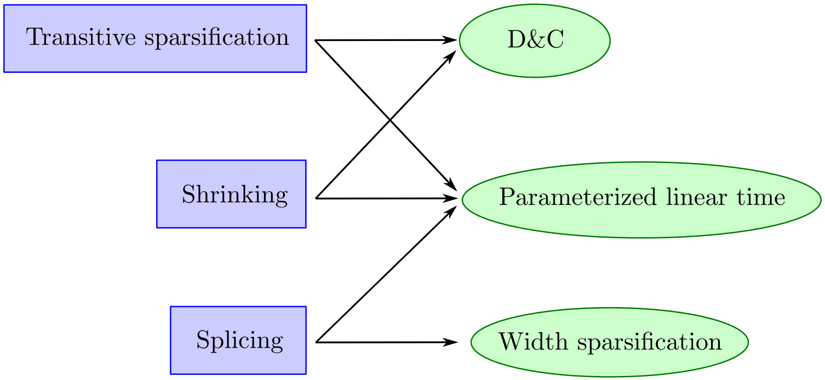
Thus we obtain

Corollary 1.1

We compute a spanning subgraph $G' = (V, E')$ of G with $|E'| < 2|V|$ and width k in time $O(k^3|V| + |E|)$.

$\longrightarrow$ We also show that the bound $2|V|$ is asymptotically tight

Roadmap



Transitive sparsification
Shrinking
Splicing
D&C
Parameterized linear time
Width sparsification

Funding

European Research Council

Established by the European Commission

[1] Bonizzoni, P.
A linear-time algorithm for the perfect phylogeny haplotype problem.
Algorithmica 48, 3 (2007), 267–285.

[2] Bova, S., Ganian, R., and Szeider, S.
Model checking existential logic on partially ordered sets.
ACM Transactions on Computational Logic (TOCL) 17, 2 (2015), 1–35.

[3] Bunte, S., and Kliewer, N.
An overview on vehicle scheduling models.
Public Transport 1, 4 (2009), 299–317.

[4] Chang, Z., Li, G., Liu, J., Zhang, Y., Ashby, C., Liu, D., Cramer, C. L., and Huang, X.
Bridger: a new framework for de novo transcriptome assembly using RNA-seq data.
Genome Biology 16, 1 (2015), 1–10.

[5] Chen, Y., and Chen, Y.
An efficient algorithm for answering graph reachability queries.
In *2008 IEEE 24th International Conference on Data Engineering* (2008), IEEE, pp. 893–902.

[6] Chen, Y., and Chen, Y.
On the graph decomposition.
In *2014 IEEE Fourth International Conference on Big Data and Cloud Computing* (2014), IEEE, pp. 777–784.

References III

[7] Colbourn, C. J., and Pulleyblank, W. R.
Minimizing setups in ordered sets of fixed width.
Order 1, 3 (1985), 225–229.

[8] Desrosiers, J., Dumas, Y., Solomon, M. M., and Soumis, F.
Time constrained routing and scheduling.
Handbooks in Operations Research and Management Science 8 (1995), 35–139.

[9] Dilworth, R. P.
A decomposition theorem for partially ordered sets.
Annals of Mathematics 51, 1 (1950), 161–166.

References IV

[10] Eriksson, N., Pachter, L., Mitsuya, Y., Rhee, S.-Y., Wang, C., Gharizadeh, B., Ronaghi, M., Shafer, R. W., and Beerenwinkel, N.
Viral population estimation using pyrosequencing.
PLoS Computational Biology 4, 5 (2008), e1000074.

[11] Fulkerson, D. R.
Note on Dilworth's decomposition theorem for partially ordered sets.
Proceedings of the American Mathematical Society 7, 4 (1956), 701–702.

[12] GAJARSKỲ, J., HLINENỲ, P., LOKSHTANOV, D., OBDRALEK, J., ORDYNIAK, S., RAMANUJAN, M., AND SAURABH, S.
FO model checking on posets of bounded width.
In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science* (2015), IEEE, pp. 963–974.

[13] GRAMM, J., NIERHOFF, T., SHARAN, R., AND TANTAU, T.
Haplotyping with missing data via perfect path phylogenies.
Discrete Applied Mathematics 155, 6-7 (2007), 788–805.

[14] HOPCROFT, J. E., AND KARP, R. M.
An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs.
SIAM Journal on Computing 2, 4 (1973), 225–231.

[15] Ikiz, S., and Garg, V. K.
Efficient incremental optimal chain partition of distributed program traces.
In *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)* (2006), IEEE, pp. 18–18.

[16] Jagadish, H. V.
A compression technique to materialize transitive closure.
ACM Transactions on Database Systems (TODS) 15, 4 (1990), 558–598.

[17] Jaśkowski, W., and Krawiec, K.
Formal analysis, hardness, and algorithms for extracting internal structure of test-based problems.
Evolutionary Computation 19, 4 (2011), 639–671.

[18] Kowaluk, M., Lingas, A., and Nowak, J.
A path cover technique for LCAs in DAGs.
In *Scandinavian Workshop on Algorithm Theory* (2008), Springer, pp. 222–233.

[19] Liu, R., and Dickerson, J.
Strawberry: Fast and accurate genome-guided transcript reconstruction and quantification from RNA-Seq.
PLoS Computational Biology 13, 11 (2017), e1005851.

[20] Ma, J.
Co-linear Chaining on Graphs With Cycles.
Master's thesis, University of Helsinki, Faculty of Science, 2021.

[21] MacKinnon, S. J., Taylor, P. D., Meijer, H., and Akl, S. G.
An optimal algorithm for assigning cryptographic keys to control access in a hierarchy.
IEEE Transactions on Computers 34, 09 (1985), 797–802.

[22] Mäkinen, V., Tomescu, A. I., Kuosmanen, A., Paavilainen, T., Gagie, T., and Chikhi, R.
Sparse Dynamic Programming on DAGs with Small Width.
ACM Transactions on Algorithms (TALG) 15, 2 (2019), 1–21.

[23] Marchal, L., Nagy, H., Simon, B., and Vivien, F.
Parallel scheduling of dags under memory constraints.
In *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (2018), IEEE, pp. 204–213.

[24] Ntafos, S. C., and Hakimi, S. L.
On path cover problems in digraphs and applications to program testing.
IEEE Transactions on Software Engineering 5, 5 (1979), 520–529.

[25] Rizzi, R., Tomescu, A. I., and Mäkinen, V.
On the complexity of minimum path cover with subpath constraints for multi-assembly.
BMC Bioinformatics 15, S-9 (2014), S5.

[26] Tomlinson, A. I., and Garg, V. K.
Monitoring functions on global states of distributed programs.
Journal of Parallel and Distributed Computing 41, 2 (1997), 173–189.

[27] Trapnell, C., Williams, B. A., Pertea, G., Mortazavi, A., Kwan, G., Van Baren, M. J., Salzberg, S. L., Wold, B. J., and Pachter, L.
Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation.
Nature Biotechnology 28, 5 (2010), 511.

[28] Van Bevern, R., Bredereck, R., Bulteau, L., Komusiewicz, C., Talmon, N., and Woeginger, G. J.
Precedence-constrained scheduling problems parameterized by partial order width.
In *International Conference on Discrete Optimization and Operations Research* (2016), Springer, pp. 105–120.

[29] Zhan, X., Qian, X., and Ukkusuri, S. V.
A graph-based approach to measuring the efficiency of an urban taxi service system.
IEEE Transactions on Intelligent Transportation Systems 17, 9 (2016), 2479–2489.