When Should You Wait Before Updating?

Toward a Robustness Refinement

Swan Dubois, Laurent Feuilloley, Franck Petit, Mikaël Rabie

Sorbonne University, University of Lyon, Paris Cité University

SAND 2023 · Pisa
Dynamic algorithms and our approach
Dynamic algorithms and our approach

**Dynamic Algorithm Approach**

- Computation
- Network Changes

**Our Approach**

- Find the Good Spot
Dynamic algorithms and our approach

![Diagram showing comparison between Dynamic Algorithm Approach and Our Approach in terms of Solution Quality and Number of Changes.](image-url)
General goal

Identify instances and solutions that are more robust than others
Necessary restrictions

In some situations there is no hope to get any sort of robustness.

→ Very sensitive problems, eg maintain the number of edges

→ Arbitrary dynamic.
Restriction of the dynamic

Assumption on the dynamic:

- **Edge removals** only (static vertex set, no edge additions)
- Keeping **connectivity**.
Restriction of the dynamic

Assumption on the dynamic:

- Edge removals only (static vertex set, no edge additions)
- Keeping connectivity.

Motivations:

- Decaying networks
- More promising
- Eventual footprint.

Granularity of edge-removals

New parameter: $k =$ number of edges removed.

A solution is $k$-robust if it is still correct after $k$ edge removals.
Focus on local problems

Locally checkable problems (LCL): the solution is correct if and only if it is locally correct everywhere.

Our three problems:
Two basic questions

For a problem $P$ and a parameter $k$:

**Universal question**: What is the set of graphs such that all solutions are $k$-robust, $\mathcal{U}_k^P$?

**Existential question**: What is the set of graphs such that there exists a solution that is $k$-robust, $\mathcal{E}_k^P$?

→ Examples with $P =$ maximal matching and $k = 1$. 
Basic building blocks

Trees

Sputnik graphs
Minimal dominating set results

**Theorem:** $\forall k, \mathcal{U}_k^{MDS} = \{\text{Sputnik graphs}\}$.

Proof idea:
Minimal dominating set results

**Theorem**: \( \forall k, \mathcal{U}_k^{MDS} = \{ \text{Sputnik graphs} \}. \)

**Proof idea**: 

![Diagram of network and set relationship]
Maximal matchings results

Theorem:

$U_{1}^{MM} =$ balanced bicliques $(K_{t,t})$, even cliques $(K_{2n})$, and trees.

$U_{k>1}^{MM} =$ cycle $C_{4}$, and trees.
Maximal matchings results

Theorem:

\[ \mathcal{U}_{\text{MM}}^1 = \text{balanced bicliques } (K_{t,t}), \text{ even cliques } (K_{2n}), \text{ and trees.} \]

\[ \mathcal{U}_{k>1}^\text{MM} = \text{cycle } C_4, \text{ and trees.} \]

Core of the proof: proving that “\( G \in \mathcal{U}_{1}^\text{MM} \)” is equivalent to “all maximal matchings in \( G \) are perfect matchings” (except for trees).

Theorem [Summer79]: Randomly matchable graphs = \( \{K_{tt}, K_{2n}\} \).
Maximal independent set results

→ No exact characterization of $\mathcal{U}_{k}^{\text{MIS}}$. :-(

**Theorem:** A strict hierarchy: $\mathcal{U}_{k+1}^{\text{MIS}} \subsetneq \mathcal{U}_{k}^{\text{MIS}}$.

**Proof:** $G_k \in \mathcal{U}_{k}^{\text{MIS}} \setminus \mathcal{U}_{k+1}^{\text{MIS}}$: 

![Diagram](image)
Maximal independent set results

→ No exact characterization of $\mathcal{U}_k^{MIS}$. :-(

**Theorem:** A strict hierarchy: $\mathcal{U}_{k+1}^{MIS} \subsetneq \mathcal{U}_k^{MIS}$.

Proof: $G_k \in \mathcal{U}_k^{MIS} \setminus \mathcal{U}_{k+1}^{MIS}$:
Maximal independent set results

→ No exact characterization of $U^\text{MIS}_k$. :-(

**Theorem:** A strict hierarchy: $U^\text{MIS}_{k+1} \subsetneq U^\text{MIS}_k$.

**Proof:** $G_k \in U^\text{MIS}_k \setminus U^\text{MIS}_{k+1}$.
About the existential question

**Theorem [CDPR20]**: There exists a polynomial time algorithm to decide if a graph belongs to $E_\infty^{MIS}$.

**Theorem**: Deciding $E_1^{MIS}$ is NP-hard.

Proof: Focus on the 2-connected case where it is equivalent to “perfect stable” (every node not in the set is dominated at least twice).

**Theorem** [Croitoru-Suditu 83] Deciding if there exists a perfect stable is NP-hard.
Conclusion

- **Universal question**: Good understanding, but small classes.
- **Existential question**: Large classes but difficult to recognize and use.
- Maybe the right question is in between: graphs such that there exist robust solutions that are easy to find/maintain.
- Even better: new criteria to quickly measure the robustness of subgraphs, in order to guide algorithms.