# On Improving the Cohesiveness of Graphs by Merging Nodes:

## Formulation, Analysis, and Algorithms


Fanchen Bu *and* Kijung Shin

KAIST EE & AI

Email: {boqvezen97,kijungs}@kaist.ac.kr

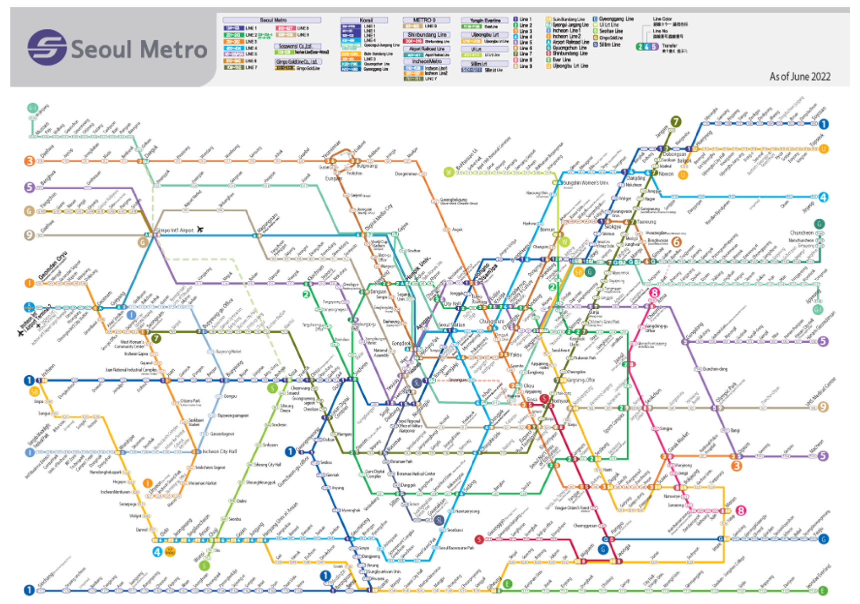
Code: bit.ly/truss_merge_code


### Summary


- **Goal:** To study the problem of improving the cohesiveness of real-world systems using graph analysis and graph algorithms
- **Formulation:** How can we mathematically formulate such a problem?
- **Analysis:** To analyze the problem from both theoretical and empirical aspects
- **Experiments:** To show the empirical performance of the proposed algorithm based on our analysis
  - **Fast:** The proposed algorithm time-efficiently finds good node pairs by pruning unpromising node pairs and identifying a small number of promising candidate node pairs
  - **Effective:** Merging the node pairs selected by the proposed algorithm effectively improves the optimization objective, and also effectively improves the cohesiveness of graphs


### Motivation

- Many real-world systems can be abstracted and represented as graphs
- **Cohesiveness:** We love cohesive systems in the real world




Cohesive public traffic networks are convenient and robust to unexpected problems


Cohesive social networks imply good communication and tight relations

- Therefore, improving the cohesiveness of graphs is a meaningful problem
- How can we mathematically formulate such a problem?
  - How (what *metric*) can we measure the cohesiveness of graphs?
  - How (what *operation*) can we improve the cohesiveness of graphs?

### Existing Research

- How does existing research formulate such a problem? What metrics and operations have researchers considered?
  - **Metrics:** Maximizing the size of a cohesive subgraph model
  - **Operations:** Anchoring nodes (forcing nodes to stay in the considered cohesive subgraph model) or adding edges

| | | Operation: Anchoring nodes | Operation: Adding edges |
|---|---|---|---|
| Metric | Maximizing the size of a $k$-core | Bhawalkar et al. 2015<br>Zhang et al. 2022 | Zhou et al. 2022 |
| Metric | Maximizing the size of a $k$-truss | Zhang et al. 2018 | Sun et al. 2021<br>Chen et al. 2022 |

- However, a realistic and interesting operation has been overlooked!

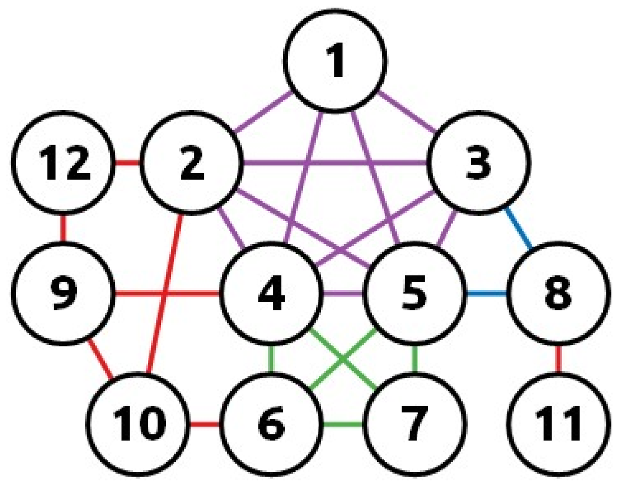
### Problem Formulation

- **Merging nodes:** An overlooked operation, which is realistic and interesting


Merging stations can make public traffic network more compact and reduce maintenance expenses

In social networks, merging people can represent forming teams, which can create a more collaborative and synergic environment

- **Metric:** We use the size of a $k$-truss as the cohesiveness metric
- **Definition:** Given a graph $G$ and $k$, the $k$-truss of $G$ is the maximal subgraph such that each edge in the $k$-truss is in at least $k-2$ triangles
  - Node engagements (neighbors) and edge interrelatedness (triangles)




- 5-truss: only purple edges
- 4-truss: + green edges
- 3-truss: + blue edges
- 2-truss: + red edges (i.e., all the edges)
- The *trussness* of a node (or an edge) is the maximum $k$ such that the node (or the edge) is in the $k$-truss

- **Final problem statement:**
  - **Given:** graph $G=(V,E)$, $k\in\mathbb{N}$, and budget $b\in\mathbb{N}$
  - **Find:** $b$ pairs of nodes to be merged
  - **To maximize:** # edges in the $k$-truss after merging those $b$ node pairs

### Hardness Analysis & A Naive Algorithm

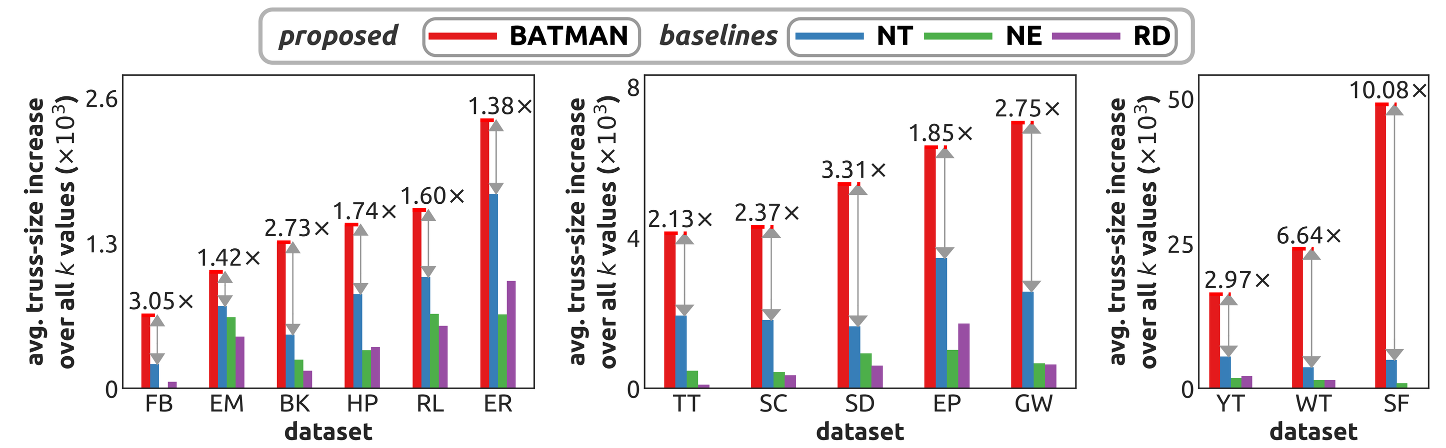
- **Theorems:** The considered problem is NP-hard and not submodular
  - Therefore, instead of aiming to solve the problem optimally, we aim to find a practically well-performing algorithm
- **A naive algorithm:** Until the budget is exhausted, we repeatedly compute # edges in the $k$-truss after all possible mergers and operate the best one
- But it takes prohibitive $O(b|V|^2|E|^{1.5})$ times!
  - We need to repeat $b$ times to use all the budget
  - There are $O(|V|^2)$ node pairs in total
  - Computing the number of edges in a $k$-truss takes $O(|E|^{1.5})$
- We aim to find good node pairs more time-efficiently
  - We do not need to check all the possible node pairs
  - Among all the nodes, how can we quickly find promising candidate nodes that may constitute good node pairs?
  - Among all the node pairs between the chosen candidate nodes, how can we quickly identify promising node pairs?

### Proposed Method: BATMAN (Best-merger seArcher for Truss MAximizatioN)

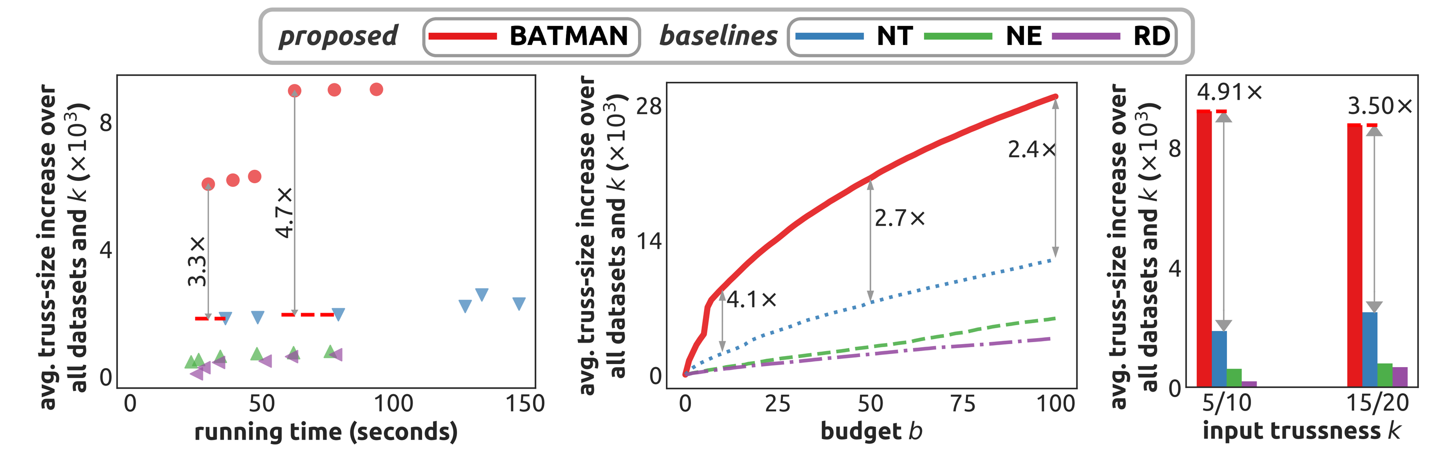
- **Check fewer edges:** We prove that, after merging any two nodes, the trussness of all the edges (except for those incident to a merged node) changes by at most one, and thus most edges currently with trussness lower than $k-1$ can be safely ignored when evaluating each node pair
- **Categorize nodes and node pairs:** We theoretically show the importance of nodes and node pairs with specific trussness (especially those with trussness at least $k-1$), and divide them into different groups based on their trussness
- **Heuristics for each category:** We develop efficient and effective heuristics for each category, where we first find promising nodes and then identify promising node pairs between the chosen nodes
- **Distribute the budget to different categories:** We exclude one category of node pairs that is theoretically unlikely to be helpful, and propose an adaptive way to distribute the budget based on the actual performance of the evaluated node pairs in each category
- **Ranking by score functions:** We finally rank candidate node pairs by score functions considering both positive and negative impacts
- With all the proposed techniques, we reduce the time complexity to $\hat{O}(b|E|^{1.5})$, where some parameters (e.g., the numbers of promising nodes and node pairs to be chosen and evaluated) set by the user are hidden in $\hat{O}$

### Experimental Results

- Extensive experiments on fourteen real-world graphs show the superiority of the proposed algorithm (**BATMAN**) w.r.t both speed and effectiveness
- Since we are the first to study this problem, we use several other heuristic-based methods as our baselines
  - **NT:** To form most new triangles consisting of the nodes in the $(k-1)$-truss
  - **NE:** To form most new edges between the nodes in the $(k-1)$-truss
  - **RD:** Randomly sampling node pairs
- We also conduct an ablation study to provide justification for each component of the proposed method.




The proposed method shows consistent superiority on different datasets




The proposed method shows consistent superiority when using different parameters