A Variable Neighborhood Search approach for the S-labeling problem

The 10th International Conference On Variable Neighborhood Search

June, 05, 2024, Lorient

Marcos Robles – marcos.robles@urjc.es

Sergio Cavero – sergio.cavero@urjc.es

Eduardo G. Pardo – eduardo.pardo@urjc.es

INTERNATIONAL CONFERENCE ON VARIABLE NEIGHBORHOOD SEARCH
VNS 2024
MIC 2024
MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES
AGENCIA ESTATAL DE INVESTIGACIÓN
grafo RESEARCH
Universidad Rey Juan Carlos

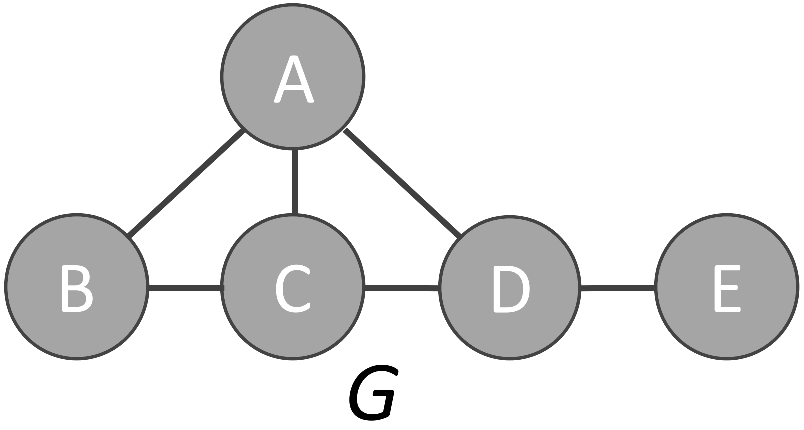
Index

Index

1. **Introduction**
2. The S-labeling problem
3. Previous works
4. Our proposal
5. Algorithmic results
6. Conclusions and future work

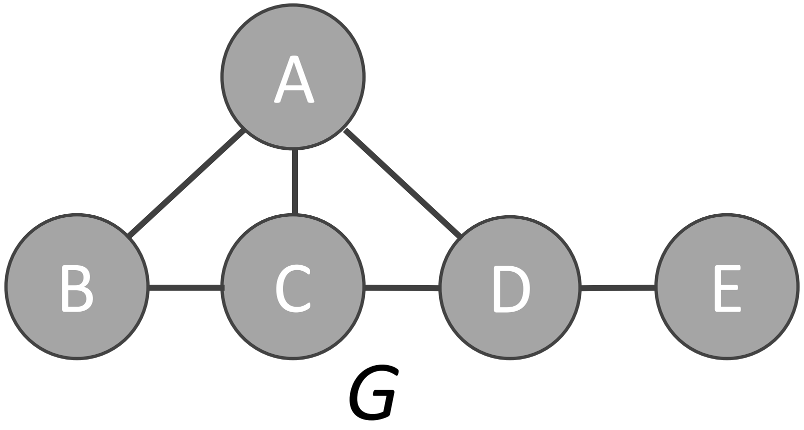
Introduction – Graph Labeling Problems

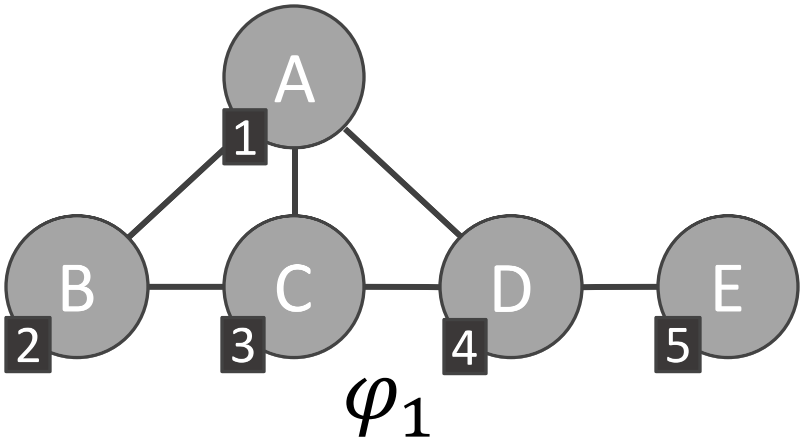
- Graph Labeling Problems are a kind of **combinatorial optimization problems**.
- A labeling of a graph consist of assigning **labels** to each vertex of an input graph to **optimize** a certain objective function.



Introduction – Graph Labeling Problems

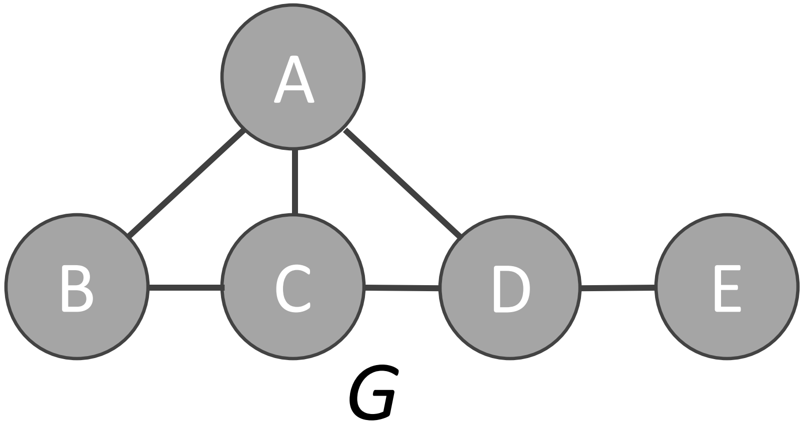
- Graph Labeling Problems are a kind of **combinatorial optimization problems**.
- A labeling of a graph consist of assigning **labels** to each vertex of an input graph to **optimize** a certain objective function.

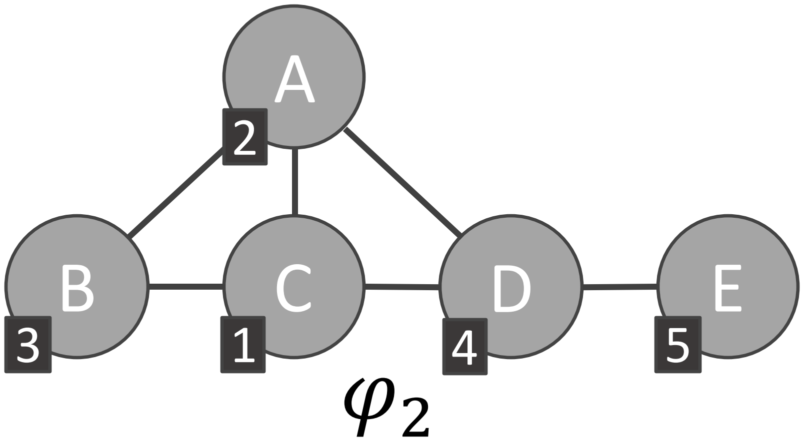




Introduction – Graph Labeling Problems

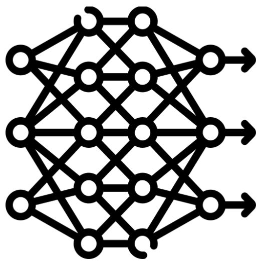
- Graph Labeling Problems are a kind of **combinatorial optimization problems**.
- A labeling of a graph consist of assigning **labels** to each vertex of an input graph to **optimize** a certain objective function.





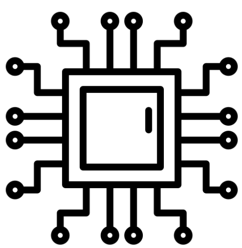
Introduction – Graph Labeling (2)

- **Graph Labeling Problems** has been found to have a lot of **real-world applications**:



Network optimization

Numerical analysis

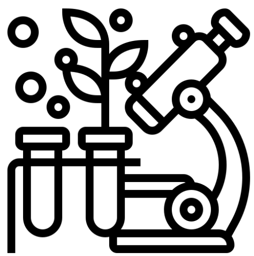


Circuit desing

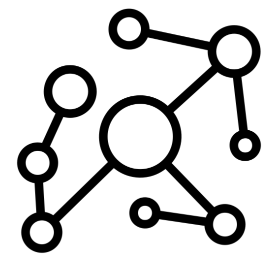
Information retrieval

Introduction – Graph Labeling (3)

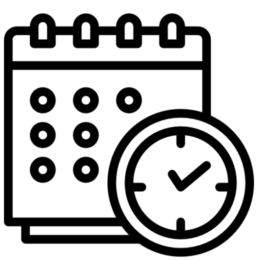
- **Graph Labeling Problems** has been found to have a lot of **real-world applications**:



Computational biology



Graph theory



Scheduling

Archaeology

Index

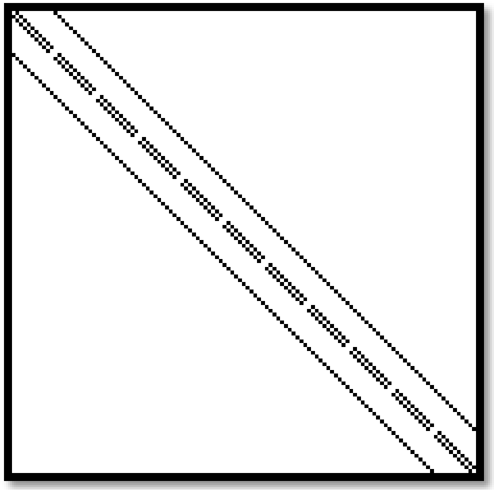
Practical applications for the S-labeling

- The S-labeling problem was originally proposed in in the context of the matrix packaging [7].
  - Matrix packaging consists of **permutating** the rows and columns of a sparse matrix to make **calculations or storage easier**.

- The S-labeling problem is a specific case of matrix packaging in which the matrix is **zero trace symmetric (0, 1)-matrix**, which represents an undirected graph.

Practical applications for the S-labeling

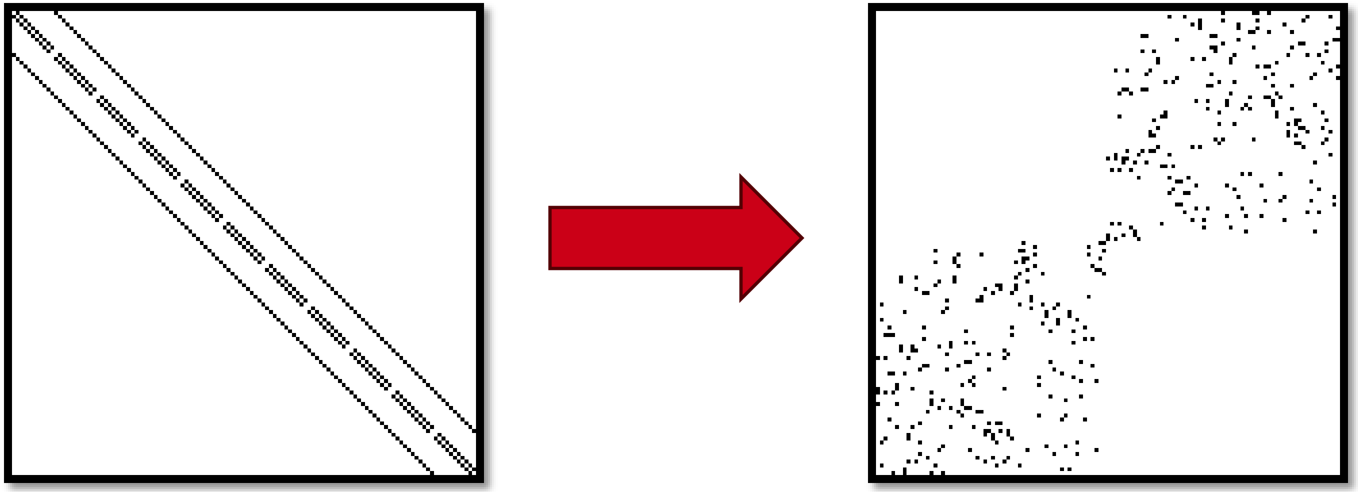
- By applying the S-labeling to a sparse matrix we get a new one that is **easier to compute**.
- **Example** of one of the instances used:



Before

Practical applications for the S-labeling

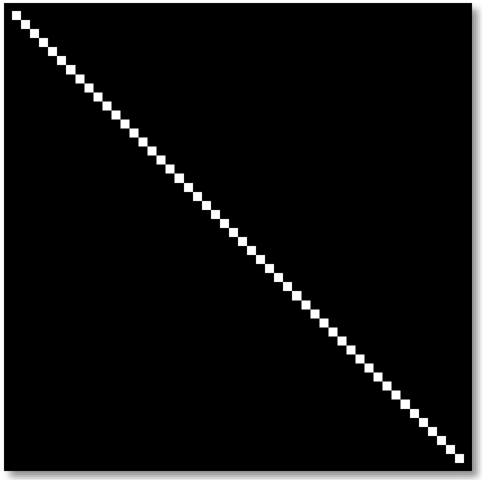
- By applying the S-labeling to a sparse matrix we get a new one that is **easier to compute**.
- **Example** of one of the instances used:



Before After

Practical applications for the S-labeling (2)

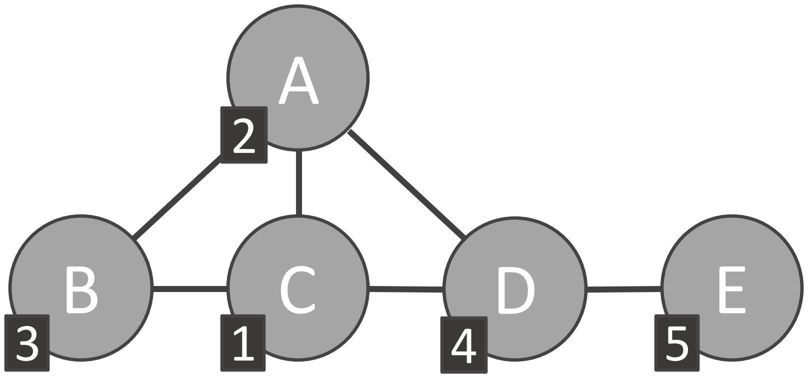
- The S-labeling is **only useful** when applied to **sparse** matrices.
- For example, the solution for **complete graphs** is **trivial**.



Complete graph

Problem description

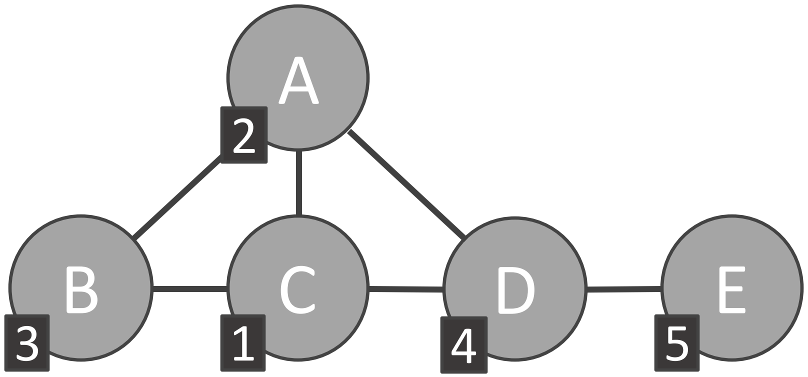
- Given a graph labeling φ, we define the **objective function value** as the sum of the evaluation of each edge.
- We evaluate an edge as the **minimum label** assigned to the vertices of that edge.



φ'

Problem description

- Given a graph labeling φ, we define the **objective function value** as the sum of the evaluation of each edge.
- We evaluate an edge as the **minimum label** assigned to the vertices of that edge.

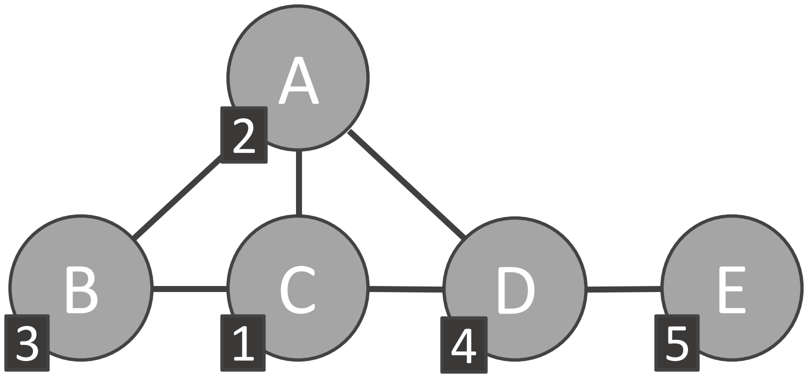


φ'

$$Eval(\varphi', (A, B)) = \min(2, 3) = 2$$

Problem description

- Given a graph labeling φ, we define the **objective function value** as the sum of the evaluation of each edge.
- We evaluate an edge as the **minimum label** assigned to the vertices of that edge.



φ'

$$Eval(\varphi', (A,B)) = \min(2,3) = 2$$
$$Eval(\varphi', (A,C)) = \min(2,1) = 1$$
$$Eval(\varphi', (A,D)) = \min(2,4) = 2$$
$$Eval(\varphi', (B,C)) = \min(3,1) = 1$$
$$Eval(\varphi', (C,D)) = \min(1,4) = 1$$
$$Eval(\varphi', (D,E)) = \min(4,5) = 4$$

Problem description (2)

- Given a graph labeling φ, we define the **objective function value** as the sum of the evaluation of each edge.
- We evaluate an edge as the **minimum label** assigned to the vertices of that edge.

$$\left.\begin{array}{l} Eval(\varphi', (A,B)) = \min(2,3) = 2 \\ Eval(\varphi', (A,C)) = \min(2,1) = 1 \\ Eval(\varphi', (A,D)) = \min(2,4) = 2 \\ Eval(\varphi', (B,C)) = \min(3,1) = 1 \\ Eval(\varphi', (C,D)) = \min(1,4) = 1 \\ Eval(\varphi', (D,E)) = \min(4,5) = 4 \end{array}\right\} \sum_{(u,v)\in E} Eval(\varphi', (u,v)) = \mathbf{11}$$

Problem description (3)

- Given a graph labeling φ, we define the **objective function value** as the sum of the minimum label of the vertices of each edge.

$$O.F.(\varphi') = \sum_{(u,v)\in E} \min(label(\varphi', u), label(\varphi', v))$$

- The objective in the S-labeling is to find the labeling φ^* among all the labelings ϕ that **minimizes the objective function**.

$$\varphi^* = \arg\min_{\varphi\in\Phi} O.F.(\varphi)$$

Index

1. Introduction
2. The S-labeling problem
3. **Previous works**
4. Our proposal
5. Algorithmic results
6. Conclusions and future work

Previous works

2006

- The problem is first proposed. [7]

2018

- The problem is studied **theoretically.** [2]

2019

- An **exact solution method** based on MIP is proposed. [6]

2023

- A heuristic approach based on a **Population-based Iterated Greedy** is presented. [5]

Index

Why VNS?

- Multiple **population-based methods** have already been studied.
  - We want to study other methods other than population-based metaheuristics.
- VNS have **multiple variants**, that fit different situations.
- We have **already used VNS** methods on other problems successfully.

Summary of our proposal

1 Random constructive method.

3 Different Shake methods

- Shuffle, random movement, and inverse.

2 Local Searches

- Swap First Improvement and Insert First-Best.

3 VNS variants

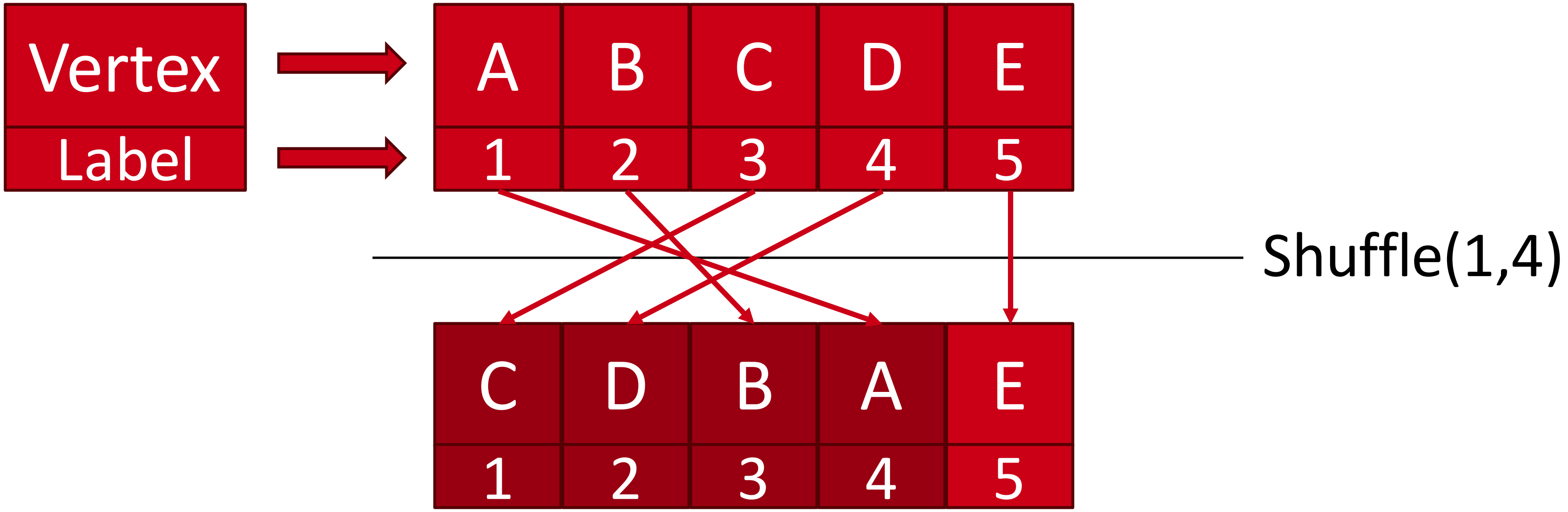
- BVNS, VND and GVNS.

VNS variants

- **Basic VNS (BVNS)**: applies the VNS schema without any modification.
- **Variable Neighborhood Descent (VND)**: removes the Shake step and adds another Local Search to scape from local optimums.
- **General VNS (GVNS)**: replaces the Local Search step in BVNS for a VND.

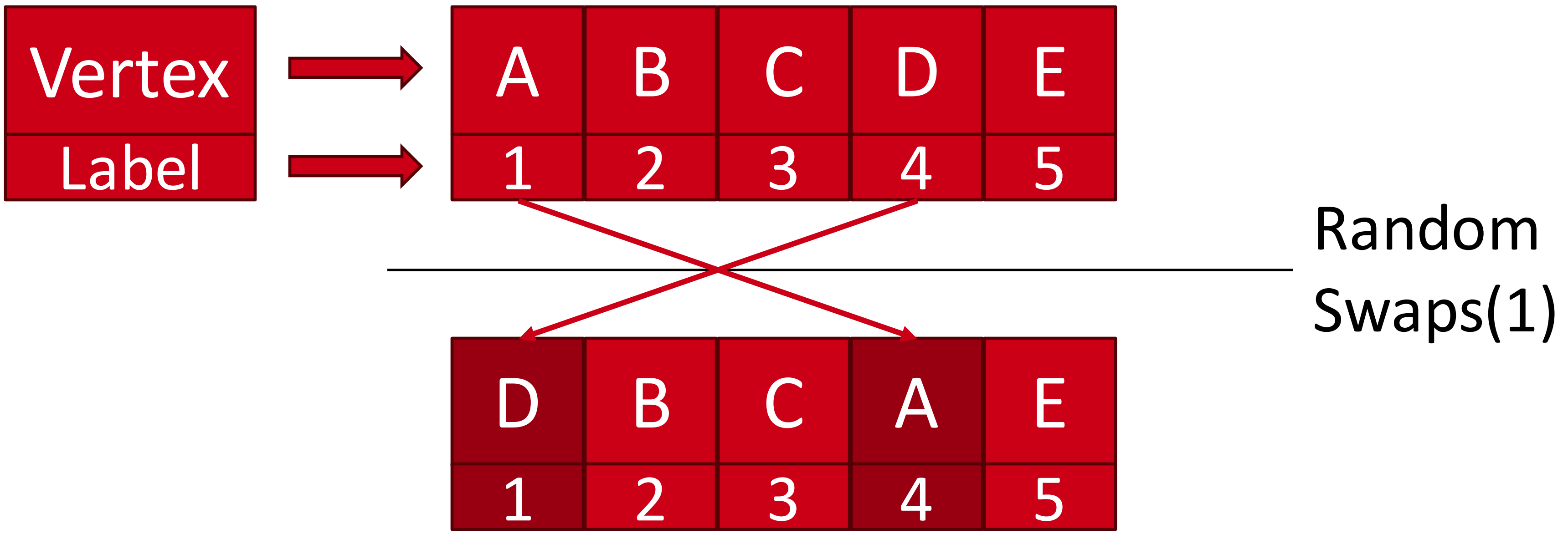
Shake methods (1)

- **ShuffleShake**: randomly shuffling the label of a subset of vertices with labels in the range *[1,k]*.



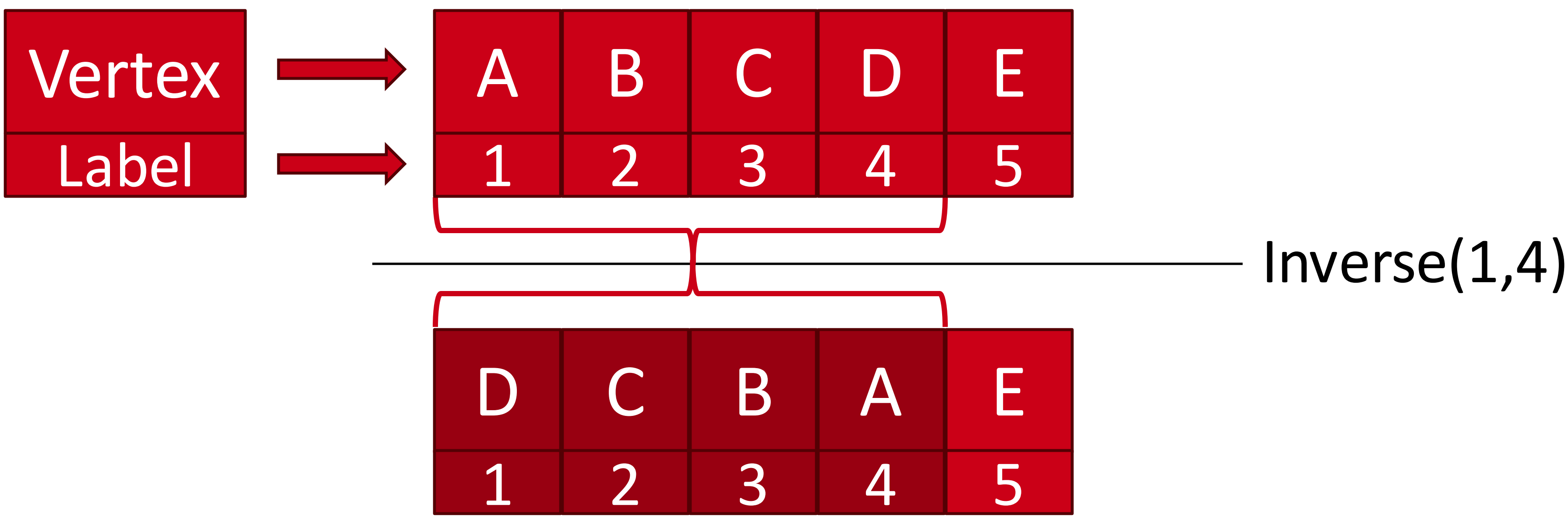
Shake methods (2)

- **NeighborhoodShake**: execute *k* random swap and insert movements.



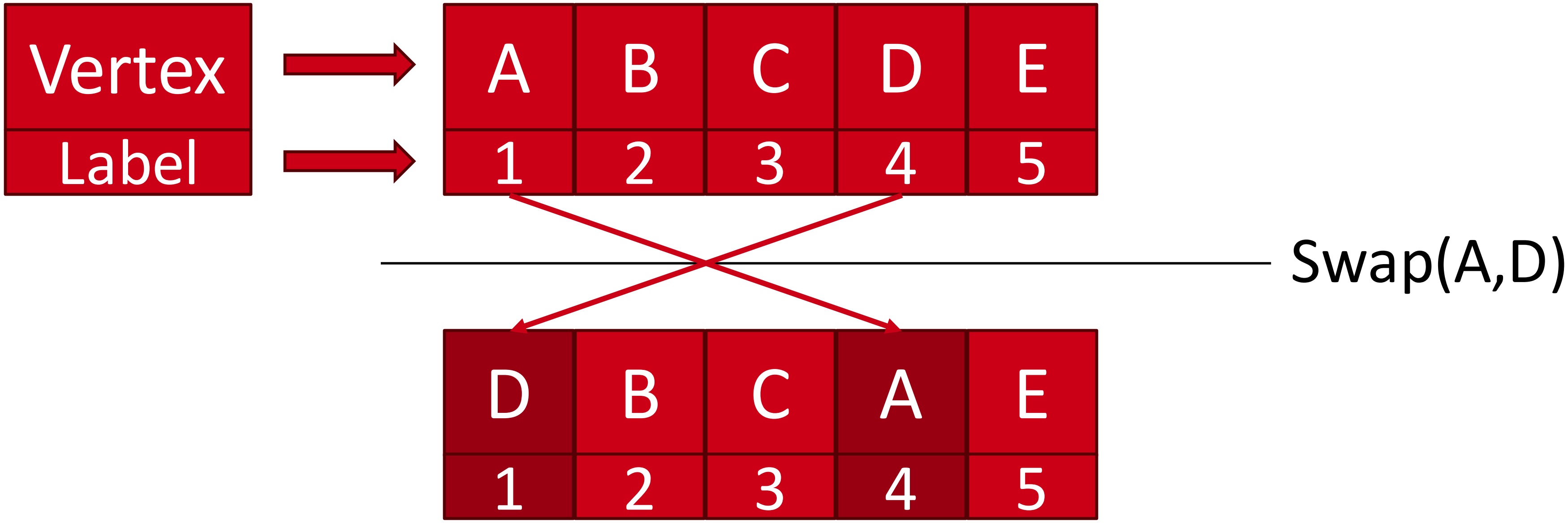
Shake methods (3)

- **InverseShake**: assign the highest labels to the vertices with the lowest initial labels, and vice versa.



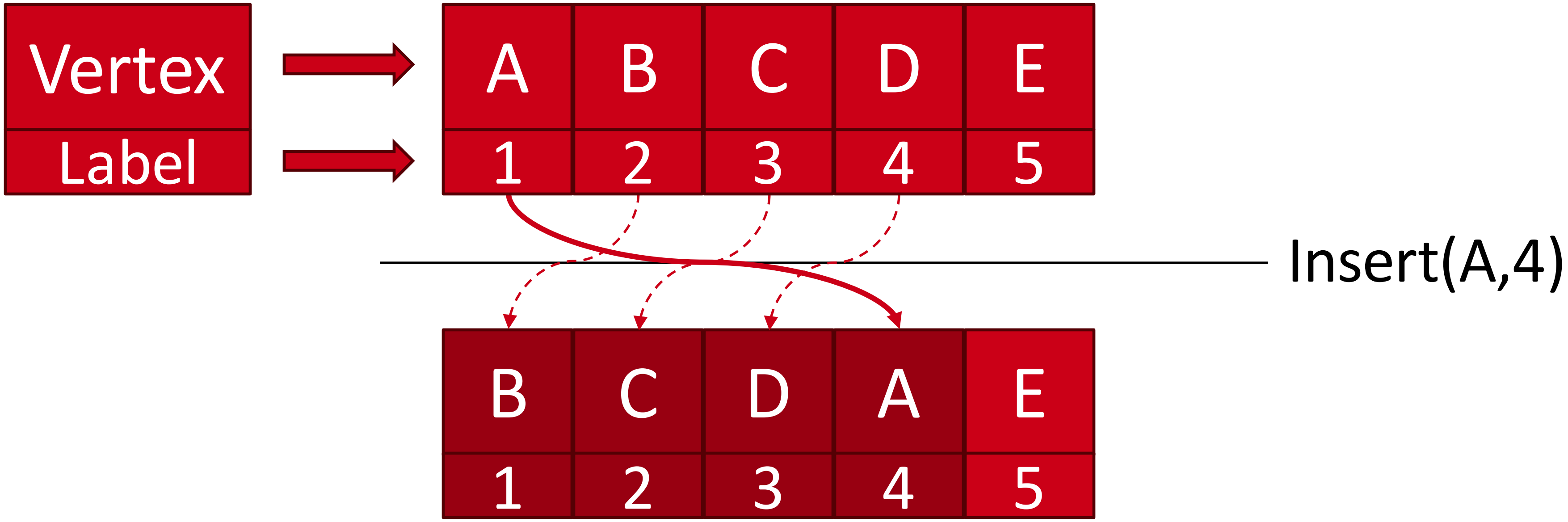
Local Search methods – Movements (1)

- **Swap movement**: exchange the assigned labels of two vertices.



Local Search methods – Movements (2)

- **Insert movement**: assign a certain label to a given vertex, displacing the rest of vertices.

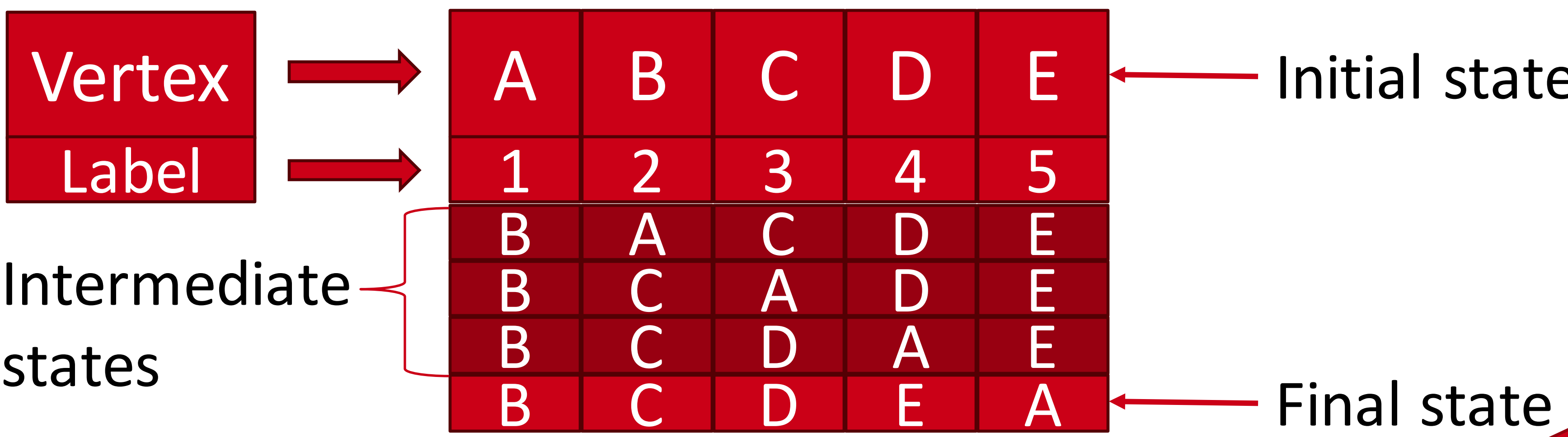


Local Search methods - Strategy

- The most common ones are
  - **First Improvement (FI)**: commit the first movement that produces an improvement.
  - **Best Improvement (BI)**: commit the best movement among all possibles.
- For both movements the **BI** strategy was tested and found **too costly**.
- For the **Swap** movement we chose the **FI strategy**, which produced good results and diversified the search.

Local Search methods – Insert strategy

- For the insert movement we also propose a **First-Best Strategy** for the **Insert movement**.
- This new strategy surged from the idea that the insert movement produces **multiple intermediate states** which **can be evaluated**.



Summary of our proposal

1 Random constructive method.

2 Local Searches

- Swap First Improvement and Insert First-Best.

3 Different Shake methods

- Shuffle, random movement, and inverse.

3 VNS variants

- BVNS, VND and GVNS.

- **A total of 14 different combinations.**

Index

BVNS – Tuning

Shake	Inverse	Movement	Shuffle	Inverse	Movement	Shuffle
LS	Insert			Swap		
Avg. O.F.	1539906	1687174	1548403	1489568	1576198	**1489481**
CPU T. (s)	300	300	300	300	300	300
% Dev.	1.77	10.58	1.95	**0.04**	4.48	**0.04**
% Best	0	0	0	45	0	**55**

- The most effective shake is the **Shuffle**.
- The most effective movement is the **Swap**.
- The best variant is **Shuffle + Swap**.

VND – Tuning

	VND – Insert&Swap	VND – Swap&Insert
Avg. O.F.	**1606570**	1629959
CPU T. (s)	303	301
% Dev.	**0**	1.15
% Best	**100**	0

- The most effective VND is the **Insert&Swap**.

GVNS – Tuning

Shake	Inverse	Movement	Shuffle	Inverse	Movement	Shuffle
VND	VND – Insert&Swap			VND – Swap&Insert		
Avg. O.F.	1505096	1524217	1496944	1508176	1531479	**1488297**
CPU T. (s)	300	300	300	300	300	300
% Dev.	0.67	1.57	0.25	0.74	1.93	**0.02**
% Best	10	0	15	0	0	**75**

- The most effective shake is the **Shuffle**.
- The most effective VND is the **Swap&Insert**.
- The best variant is **Shuffle + Swap&Insert**.

Comparison with the state-of-the-art

	State-of-the-art	BVNS	VND	GVNS
	Population-based Iterated Greedy (PIG)	Shuffle + Swap	Insert&Swap	Shuffle + Swap&Insert
Avg. O.F.	**1477107**	1489481	1606570	1488297
CPU T. (s)	200	300	303	300
% Dev.	**0.00**	0.72	6.17	0.60
% Best	**95**	5	0	0

- Among the proposed **VNS methods**, the best one is the GVNS and the worst one the VND.
- The VNS results are **close** to those of the PIG method, but they are still **outperformed by the state-of-the-art**.

Index

1. Introduction
2. The S-labeling problem
3. Previous works
4. Our proposal
5. Algorithmic results
6. **Conclusions and future work**

Conclusions

- We presented 14 different combinations of **VNS method** for the **S-labeling problem**. The **state-of-the-art** algorithm obtains **better results** than our proposal.

- The **GVNS** emerged as the **most effective**, with a **deviation lower than 1%**.

- The **VND** emerged as the **least effective**, showing that in this problem using **effective shake** methods is more important than using more local searches.

Future work

- Explore **new neighborhoods**, such as ejection chain.
- Explore alternative **constructive methods**.
- Implement methods which make use of the S-labeling **theoretical properties**.

A Variable Neighborhood Search approach for the S-labeling problem

The 10th International Conference On Variable Neighborhood Search

June, 05, 2024, Lorient

Marcos Robles – marcos.robles@urjc.es

Sergio Cavero – sergio.cavero@urjc.es

Eduardo G. Pardo – eduardo.pardo@urjc.es

INTERNATIONAL CONFERENCE ON VARIABLE NEIGHBORHOOD SEARCH VNS 2024

MIC 2024

MINISTERIO DE CIENCIA, INNOVACIÓN Y UNIVERSIDADES

AGENCIA ESTATAL DE INVESTIGACIÓN

grafo RESEARCH

Universidad Rey Juan Carlos

Bibliography

1. Díaz, J., Petit, J., Serna, M.: A survey of graph layout problems. ACM Computing Surveys (CSUR) 34(3), 313–356 (2002)
2. Fertin, G., Rusu, I., Vialette, S.: The S-labeling problem: An algorithmic tour. Discrete Applied Mathematics 246, 49–61 (2018)
3. Hansen, P., Mladenović, N., Brimberg, J., Pérez, J.A.M.: Variable neighborhood search. Springer (2019)
4. Hansen, P., Mladenović, N., Todosijević, R., Hanafi, S.: Variable neighborhood search: basics and variants. EURO Journal on Computational Optimization 5(3), 423–454 (2017)
5. Lozano, M., Rodriguez-Tello, E.: Population-based iterated greedy algorithm for th S-labeling problem. Computers & Operations Research 155, 106224 (2023)
6. Sinnl, M.: Algorithmic expedients for the S-labeling problem. Computers & Opera tions Research 108, 201–212 (2019)
7. Vialette, S.: Packing of (0, 1)-matrices. RAIRO-Theoretical Informatics and Appli- cations 40(4), 519–535 (2006)