Partitioning signed networks using constraints

Andrej Mrvar and Patrick Doreian

Partitioning signed networks using additional constraints is available in Pajek version 4.03 or later. The usage of constraints is demonstrated using the well-known Sampson_T4 data. The constraints that were used in this example have no substantive meaning, they are only used for demonstration purposes. In general, specifying constraints requires considerable thought.

The implementation of constraints for partitioning signed networks is much more efficient than the one used for constraints in blockmodeling – it almost does not cost any additional time. Also, so called *penalties* are not needed anymore – partitions that do not fulfill constraints are simply ignored.

Existing Pajek objects are used for specifying constraints: Two Partitions are used for specifying constraints on vertices, and two Vectors are used for specifying constraints on clusters.

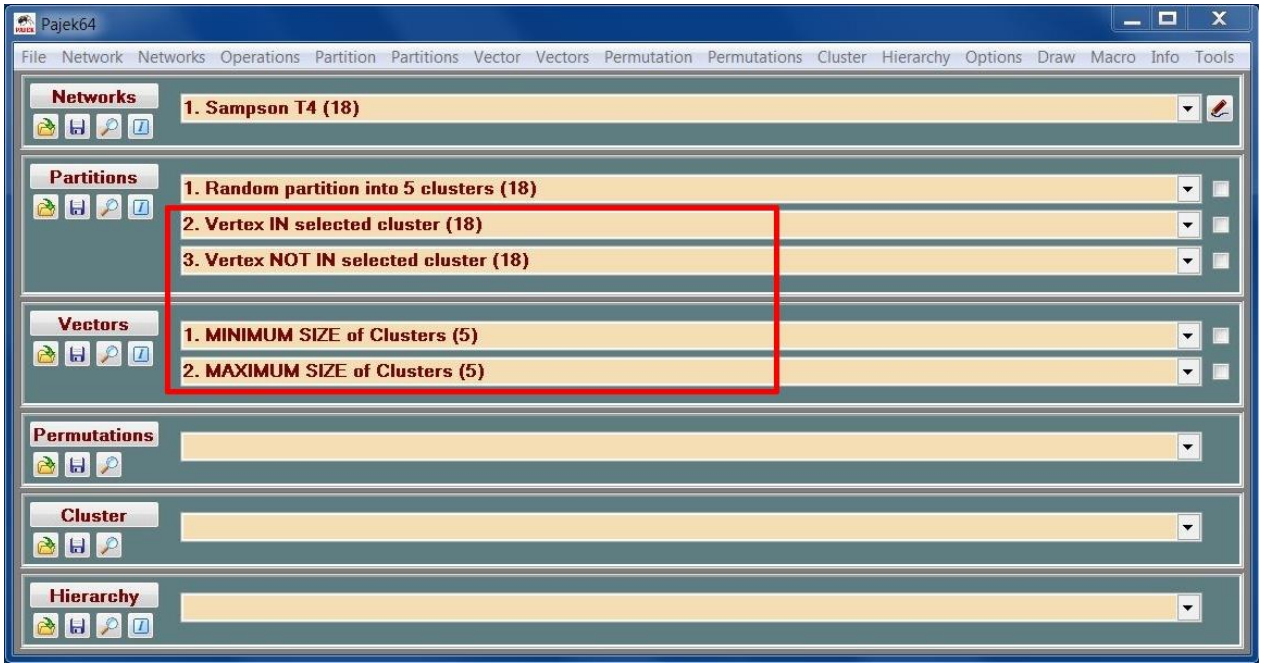

Figure 1: Pajek main window with partitions and vectors selected to be used as constraints[1].

[1] The names of the constraint partitions and the constraint vectors can have any name.

Selecting objects and setting up constraints

1. Select Second and Third Partition and First and Second Vector in the main Pajek window as shown in Figure 1.
2. Check **»Relaxed Balance«** and **»Prespecification«** in the window for partitioing signed networks. Then check the constraints you want to use (as shown in Figure 2). You do not need to specify all four constraints as shown in the figure, you can use any combination of only one, two, three, or all four constraints.

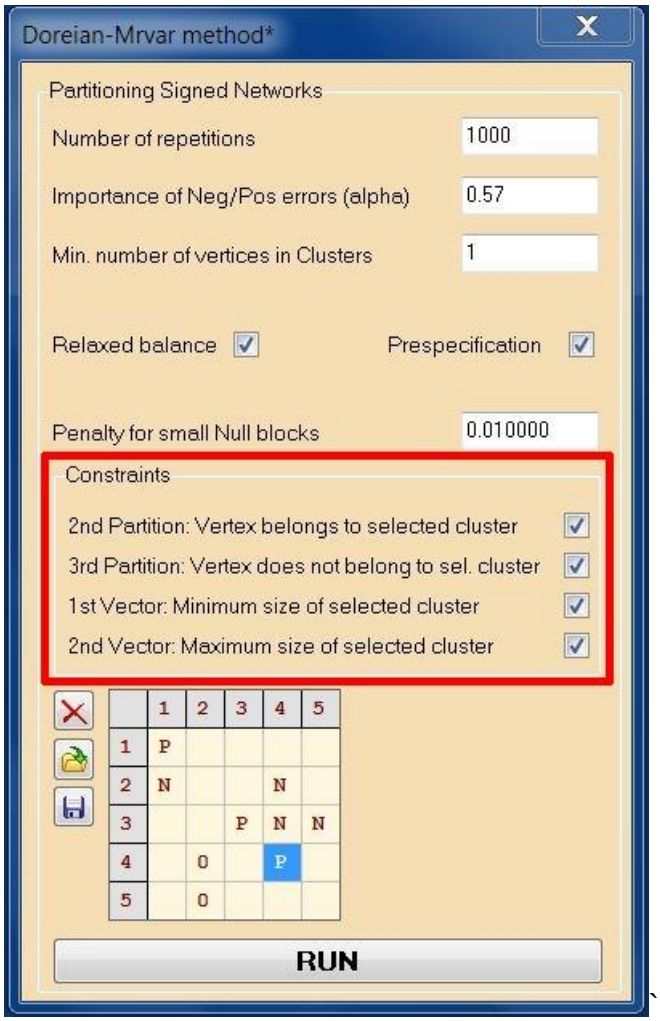

Figure 2: The four checkboxes for including constraints.

Constraints (and the model matrix where you can pre-specify locations for the types of blocks by typing **P**, **N**, or **0**) are visible only if **»Relaxed Balance«** and **»Prespecification«** are checked.

a) Constraints on vertices

The second partition is used for the constraint: ***Vertex belongs to selected cluster***. The dimension of the partition must be equal to number of vertices in the network.

A value of 0 for a vertex in a partition stands for no constraint, values larger than 1 are the constraints specifying the cluster numbers in which vertices **should** belong (e.g. value 1 means that selected vertex must be in cluster 1). In the example Partition from SampsonT4.paj only one constraint on one vertex is selected:

Vertex 18 (Simplicius) belongs to cluster 1.

The third partition is used for the constraint: ***Vertex does not belong to selected cluster***. Also the dimension of this partition must be equal to number of vertices in the network.

A value 0 in partition means no constraint, values larger than 1 are the constraints specifying the cluster numbers in which vertices **should not** belong (e.g. value 1 means that the selected vertex should not belong to cluster 1). In the example Partition constraint, only one vertex was selected:

Vertex 17 (Elias) does not belong to cluster 1.

Note that: to create a partition for the two constraints 'by hand' you can use:

Partition/Create Constant Partition

with dimension equal to number of vertices in a network and constant 0 – no constraint for all vertices. Then click the ***Edit-Partition*** *icon to manually put some vertices in clusters different from 0.*

b) Constraints on clusters sizes

The first vector is used for a constraint: ***Minimum size of selected cluster***. The dimension of the vector must be equal to the number of clusters.

A value 0 in vector means no constraint, values larger than 0 are cluster sizes (e.g. 12 means that selected cluster must contain at least 12 vertices). In the example Vector, only one constraint on size of one cluster is selected:

Cluster 2 contains at least 12 vertices.

The second vector is used for a constraint: ***Maximum size of selected cluster***. The dimension of vector must be equal to number of clusters.

A value 0 in vector means no constraint, values larger than 0 are cluster sizes (e.g. value 1 means that selected cluster must contain at most 1 vertex). In the example Vector, three constraints were selected:

Cluster 1 contains at most 1 vertex.
Cluster 3 contains at most 1 vertex.
Cluster 5 contains at most 1 vertex.

Note that: to create vector for these two constrains you can use

Vector/Create Constant Vector

with dimension equal to number of clusters (five in the example) and constant 0. Then click the ***Edit-Vector*** *icon to manually put some vertices in clusters different from 0.*

Execution in Pajek

When you **Run** the optimization all additional constraints that are used are reported at the beginning of the Report window as shown below:

Constraints:
Vertex 18 (Simplicius) belongs to cluster 1.
Vertex 17 (Elias) does not belong to cluster 1.
Cluster 2 contains at least 12 vertices.
Cluster 1 contains at most 1 vertices.
Cluster 3 contains at most 1 vertices.
Cluster 5 contains at most 1 vertices.

Be careful: Before starting the optimization, the initial partition is modified (if needed) to fulfill all of the specified constraints. In this case, a warning is added to the report window:

The initial partition was modified to fulfill constraints.

Later (comparing to unconstraint optimization), you will not notice any changes in optimization and reporting results, but only partitions that fulfill the constraints are taken into account as feasible partitions.