SPISEK UKAZOV V PAJKU

S KRATKO RAZLAGO

Pajek

Pajek XXL

Pajek 3XL

ANDREJ MRVAR

Fakulteta za družbene vede
Univerza v Ljubljani

Pajek 5.06

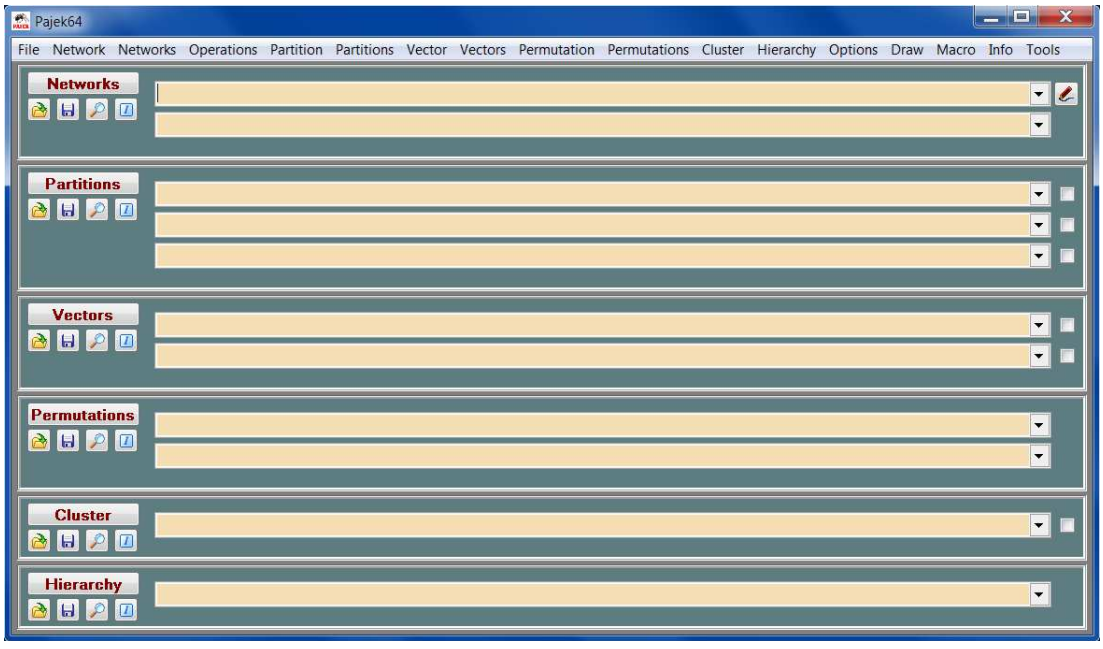
1. oktober 2018

Program Pajek

Program Pajek je namenjen analizi in prikazu velikih omrežij. Analizo izvajamo s pomočjo šestih podatkovnih struktur: omrežje, razbitje, permutacija, skupina, hierarhija in vektor. Vsaka struktura, ki jo preberemo ali dobimo kot rezultat neke analize ostane shranjena v pomnilniku in na voljo za nadaljnje delo. V program je vgrajenih veliko število učinkovitih algoritmov, prava moč programa pa se pokaže šele ob kombiniranju večih algoritmov za dosego željenih rezultatov in uporabi jezika makro.

Splošno o programu

Ko poženemo program, se nam odpre glavno okno, ki je narisano na sliki 1. V zgornjem delu okna se nahaja glavna izbira, ki je sestavljena iz večih podizbir, ki so opisane v nadaljevanju. Glavni del okna pa zavzema šest podatkovnih struktur. Za vsako od teh struktur se zgradi poseben seznam v katerega se dodajajo rezultati analiz in iz katerega izbiramo strukture, nad katerimi bi radi opravljali analize.



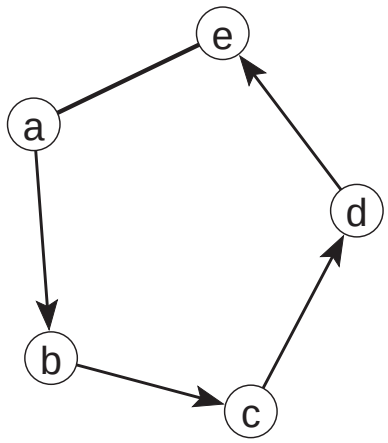
Slika 1: Glavno okno programa Pajek.

Pregled in kratek opis ukazov

Kot smo že omenili, je program Pajek v stalnem razvoju. V nadaljevanju so izpisani in na kratko opisani algoritmi, ki so (bili) implementirani v zadnji izdaji. Ker je v program vključenih veliko število algoritmov, sledi struktura navodil strukturi gnezdenih izbir v programu (zaradi lažje orientacije).

- **File** – Delo z datotekami.
  - **Network – N** (Omrežje, privzeti podaljšek je .NET)
    - **Read** – Omrežje preberemo z vhodne (Ascii/Unicode UTF8) datoteke. Omrežje lahko opišemo na več načinov. Najpreprosteje ga opišemo tako, da najprej podamo število točk omrežja (*Vertices) ter naštejemo vse številke in oznake točk. Nato navedemo še urejene pare, ki določajo usmerjene (*Arcs) ali neusmerjene (*Edges) povezave. Povezavi priredimo vrednost tako, da urejenemu paru pripišemo še realno število.
      poleg običajnih omrežij pozna Pajek tudi *dvodelna* (*2-mode* ali *affiliation*) omrežja. Opis dvodelnih omrežij je predstavljen na strani 24.
      Leva stran slike 2 prikazuje opis petkotnika s štirimi usmerjenimi in eno neusmerjeno povezavo na vhodni datoteki. Dve usmerjeni povezavi imata vrednost 2, ostale povezave pa imajo vrednost 1. Če vrednosti povezav na vhodni datoteki izpustimo, se vse vrednosti postavijo na vrednost 1. Sredina slike prikazuje sliko opisanega omrežja. Na vhodni datoteki koordinate točk še niso zapisane, saj so bile dobljene kasneje na avtomatičen način z uporabo lastnih vektorjev. Če potem, ko določimo sliko omrežja, omrežje še enkrat shranimo, se na izhodno datoteko pripišejo tudi koordinate točk (desna stran slike).
      Poleg naštevanja urejenih parov (*Arcs ali *Edges), lahko naštejemo tudi vse povezave, ki pripadajo dani točki naenkrat (z uporabo ukazov *ArcsList oziroma *EdgesList) ali pa podamo usmerjene povezave v obliki matrike (*Matrix).
      Na vhodni datoteki lahko navedemo tudi opise kako naj se narišejo točke in povezave (oblike, barve, debeline, vzorci...). Podroben opis je na voljo na predstavitveni strani programa Pajek. Večino teh lastnosti pa lahko nastavimo tudi v oknu Draw v izbiri Export/Options. Poleg svojih *vhodnih* oblik podpira Pajek še več drugih oblik: UCINET DL; Vega; gedcom in nekaj kemijskih oblik: BS (Ball and Stick), MAC (Mac Molecule) in MOL (MDL MOLfile).

```
*Vertices 5
1 ”a”
2 ”b”
3 ”c”
4 ”d”
5 ”e”
*Arcs
1 2 1
2 3 2
3 4 2
4 5 1
*Edges
5 1 1
```




```
*Vertices 5
1 ”a” 0.17 0.33 0.5
2 ”b” 0.20 0.74 0.5
3 ”c” 0.61 0.83 0.5
4 ”d” 0.83 0.48 0.5
5 ”e” 0.56 0.17 0.5
*Arcs
1 2 1
2 3 2
3 4 2
4 5 1
*Edges
5 1 1
```

Slika 2: Opis petkotnika na vhodni datoteki (levo), ustrezna slika (sredina) in omrežje na izhodni datoteki s pripisanimi koordinatami točk (desno).

* **Read Time Events** – Preberemo opis časovnega omrežja zapisanega v ***razvojni*** obliki (.TIM): spremembe omrežja skozi čas so opisane z zaporedjem osnovnih dogodkov:

ukaz	pomen
`TI` t	initial events – sledijo dogodki, ki se zgodijo ob nastopu trenutka t
`TE` t	end events – sledijo dogodki, ki se zgodijo ob preteku trenutka t
`AV` vns	add vertex – dodaj točko v z imenom n in lastnostmi s
`HV` v	hide vertex – skrij točko v
`SV` v	show vertex – kaži točko v
`DV` v	delete vertex – odstrani točko v
`AA` uvs	add arc – dodaj usmerjeno povezavo (u, v) z lastnostmi s
`HA` uv	hide arc – skrij usmerjeno povezavo (u, v)
`SA` uv	show arc – kaži usmerjeno povezavo (u, v)
`DA` uv	delete arc – odstrani usmerjeno povezavo (u, v)
`AE` uvs	add edge – dodaj neusmerjeno povezavo $(u\colon v)$ z lastnostmi s
`HE` uv	hide edge – skrij neusmerjeno povezavo $(u\colon v)$
`SE` uv	show edge – kaži neusmerjeno povezavo $(u\colon v)$
`DE` uv	delete edge – odstrani neusmerjeno povezavo $(u\colon v)$
`CV` vs	change vertex property – spremeni lastnosti točke v s s
`CA` uvs	change arc property – spremeni lastnosti usmerjene povezave (u, v) v s
`CE` uvs	change edge property – spremeni lastnosti neusmerjene povezave $(u\colon v)$ v s
`CT` uv	change type – spremeni (ne)usmerjenost povezave (u, v)
`CD` uv	change direction – spremeni smer usmerjene povezave (u, v)
`PE` uvs	pair of arcs to edge – dve dvosmerni usmerjeni povezavi (u, v) in (v, u) zamenjaj z eno neusmerjeno povezavo $(u : v)$ z lastnostjo s
`AP` uvs	add pair of arcs – dodaj dvosmerni usmerjeni povezavi (u, v) in (v, u) z lastnostjo s
`DP` uv	delete pair of arcs – odstrani dvosmerni usmerjeni povezavi (u, v) in (v, u)
`EP` uvs	edge to pair of arcs – neusmerjeno povezavo $(u : v)$ zamenjaj z dvosmernima usmerjenima povezavama (u, v) in (v, u) z lastnostjo s

Seznam lastnosti s je lahko tudi prazen. Če lahko par točk povezuje več istovrstnih povezav, je potrebno dodati ustreznim ukazom še podatek na katero od teh povezav se ukaz nanaša – na primer z dodatkom ukazu oblike `:`k (k-ta povezava). Tako ukaz `HE:3 14 37` zahteva, da se skrije tretja neusmerjena povezava točk 14 in 37. Primer omrežja opisanega s časovnimi dogodki:

```
*Vertices 3
*Events
TI 1
AV 2 "b"
TE 3
HV 2
TI 4
```

```
AV 3 "e"
TI 5
AV 1 "a"
TI 6
AE 1 3 1
TI 7
SV 2
AE 1 2 1
TE 7
DE 1 2
DV 2
TE 8
DE 1 3
TE 10
HV 1
TI 12
SV 1
TE 14
DV 1
```

Poleg opisa časovnega omrežja v razvojni obliki prepozna Pajek tudi opis s pomočjo časovnih intervalov (glej **Network/ temporal networks**).

* **Save** – Omrežje shranimo na izhodno (Ascii/Unicode UTF8) datoteko. Omrežje lahko shranimo v eni od internih oblik (slika 2) ali pa v eni do prej omenjenih standardnih oblik.
  Če omrežje predstavlja rodovnik v navadni obliki z naslednjimi relacijami: 1. Wi→Hu, 2. Mo→Da, 3. Mo→So, 4. Fa→Da, 5. Fa→So;
  ga lahko shranimo tudi kot datoteko GEDCOM.
  Druga možnost je Pajkov navadni rodovnik: 1.Fa→Ch, 2.Mo→Ch, 3.Hu-Wi (edge), ali 1.Pa→Ch, 3.Hu-Wi (edge).
* **Save as Time Events** – Časovno omrežje shranimo v ***razvojni*** obliki.
* **View/Edit** – Urejanja omrežja: Najprej izberemo dano točko, nato pa pregledujemo in urejamo njene povezave: točki dodajamo nove ali odstranjujemo obstoječe povezave (z dvojnim pritiskom na levo tipko na miški); spreminjamo vrednosti na povezavah (s pritiskom na desno tipko na miški); zamenjamo izbrano povezavo z dvema pravokotnima povezavama s skupnim krajiščem v novi, nevidni točki.

* **Change Label** – preimenujemo trenutno izbrano omrežje.
* **Dispose** – Omrežje odstranimo iz pomnilnika (sprostimo prostor).
* **Export as Matrix to [EPS, SVG + PDF]** – Omrežje izpišemo kot matriko v obliki EPS (SVG in PDF). Rezultat je slika, ki jo lahko vključimo v urejevalnike besedila ali prikažemo v spletnih brskalnikih.
  - **Original** – Vrstni red vrstic oziroma stolpcev določa oštevilčenje, ki je definirano v opisu omrežja. To izbiro lahko uporabimo tudi za dvovrstna omrežja.
  - **Using Partition** – Izpiše se matrika, ki prikazuje gostote povezav med skupinami in ne povezav med posameznimi točkami omrežja. Gostote povezav so prikazane z različnimi sivinami. Poleg tega lahko izberemo dve skupini, ki ju prikažemo podrobno (v matriki prikažemo vse predstavnike teh dveh skupin). Obstajata dva različna načina izračuna gostot:
    Structural – Gostote se normalizirajo glede na največje možno število povezav med skupinama (primerno za gosta omrežja).
    Delta – Gostote se normalizirajo glede na točke, ki imajo največje število sosedov v posameznih skupinah (primerno za redka omrežja).
  - **Using Permutation** – Vrstni red vrstic oziroma stolpcev določa oštevilčenje, ki je določeno s permutacijo. To izbiro lahko uporabimo tudi za dvovrstna omrežja.
  - **Using Permutation + Partition** – Vrstni red vrstic oziroma stolpcev določa oštevilčenje, ki je določeno s permutacijo. Skupine v razbitju uporabimo za označitev mej med skupinami s črtami.
    To izbiro lahko uporabimo tudi za dvovrstna omrežja.
  - **Options** – Izbira dodatnih nastavitev.
    1. **Diamonds for Negative Values, Circles for 0** – Če izberemo to možnost, bodo pozitivne vrednosti predstavljene s kvadratki, negativne z rombi, vrednosti 0 pa s krožci. S tem lahko pozitivne in negativne vrednosti razlikujemo tudi v primeru črnobelega tiska.
    2. **Diamonds, Circles and Lines in GreyScale** – Rombi bodo narisani črno (sivo) in ne v rdeči barvi, krožci bodo beli, črte med skupinami pa črtkano črne.
    3. **Labels on Top/Right** – Oznake se izpisujejo nad matriko in desno od matrike (primerno za daljše oznake).

4. **Only Black Borders** – Če izberemo to možnost, bodo imeli vsi kvadratki v matriki črno obrobo, v nasprotnem primeru pa bodo imeli temnejši kvadratki belo, svetlejši pa črno obrobo.
5. **Thick Boundary Line** – Pri risanju ločevalnih črt med razredi naj se uporabijo debelejše črte.
6. **Large Squares/Diamonds/Circles** – Kvadratki, rombi in krožci bodo narisani večje ali manjše.
7. **Use Partition Colors for Vertex Labels** – Oznake točk bodo napisane z barvami, ki so določene z razbitjem.

– ****XXL** VertexID** - ID (Identifikatorji, privzeti podaljšek je .VID)

* **Read** – identifikatorje preberemo z vhodne datoteke.
* **Save** – identifikatorje shranimo na izhodno datoteko.
* **Save to NET File as Vertex Labels** – Identifikatorje shranimo v izhodno NET datoteko, iz katere jih lahko kasneje preberemo kot oznake točk z ukazom Network/Transform/Add/Vertex Labels/from File(s).
* **View/Edit** – Urejanje / spreminjanje identifikatorjev.
* **Change Label** – preimenujemo trenutno izbrane identifikatorje.
* **Dispose** – Identifikatorje odstranimo iz pomnilnika.

– **Partition – C** (Razbitje, privzeti podaljšek je .CLU)

* **Read** – Razbitje preberemo z vhodne (Ascii) datoteke. Če želimo v omrežju na sliki 2 postaviti točki a in e v skupino 1, ostale točke pa v skupino 2, opišemo ustrezno razbitje na vhodni datoteki takole:
  *Vertices 5
  1
  2
  2
  2
  1
* **Save** – Razbitje shranimo na izhodno (Ascii) datoteko.
* **View/Edit** – Urejanje razbitja: postavljanje točk v izbrane skupine.
* **Change Label** – preimenujemo trenutno izbrano razbitje.
* **Dispose** – Razbitje odstranimo iz pomnilnika.

– **Vector – V** (Vektor, privzeti podaljšek je .VEC)

* **Read** – Vektor preberemo z vhodne (Ascii) datoteke. Primer opisa vektorja na vhodni datoteki: Predpostavimo, da poznamo verjetnosti

nastopa posameznih črk v besedilu. V našem primeru s petimi črkami bi omenjene verjetnosti podali kot vektor:
*Vertices 5
0.36
0.07
0.10
0.03
0.44

  - **Save** – Vektor shranimo na izhodno (Ascii) datoteko. Na isto datoteko lahko shranimo tudi več vektorjev. V tem primeru podamo vektorje, ki jih želimo shraniti na isto datoteko s skupino - Cluster. To storimo tako, da najprej zgeneriramo prazno skupino, nato pa s pritiskom na tipko V pri izbranem vektorju, številko vektorja prenesemo v skupino. Vsi vektorji, ki jih želimo shraniti na isto datoteko, morajo imeti enako dimenzijo.
  - **View/Edit** – Urejanje vektorja: postavljanje komponent na izbrane vrednosti.
  - **Change Label** – preimenujemo trenutno izbran vektor.
  - **Dispose** – Vektor odstranimo iz pomnilnika.

– **Permutation – P** (Permutacija, privzeti podaljšek je .PER)

  - **Read** – Permutacijo preberemo z vhodne (Ascii) datoteke. Če želimo, da postane točka e prva točka omrežja, točka c druga, točka a tretja, točka b četrta in točka d peta, opišemo to na naslednji način:
*Vertices 5
5
3
1
2
4
  - **Save** – Permutacijo shranimo na izhodno (Ascii) datoteko.
  - **View/Edit** – Urejanje permutacije: zamenjave vrstnega reda točk. Z dvakratnim pritiskom na levo tipko na miški izberemo točko, ki jo želimo prestaviti na drugo mesto.
  - **Change Label** – preimenujemo trenutno izbrano permutacijo.
  - **Dispose** – Permutacijo odstranimo iz pomnilnika.

– **Cluster – S** (Skupina – seznam izbranih točk, privzeti podaljšek je .CLS)

* **Read** – Skupino preberemo z vhodne (Ascii) datoteke. Na vhodni datoteki navedemo seznam izbranih točk. Če nas zanimajo samo točke b, d in e, opišemo ustrezno skupino na vhodni datoteki takole:
  ```
  2
  4
  5
  ```
* **Save** – Skupino shranimo na izhodno (Ascii) datoteko.
* **View/Edit** – Urejanje skupine: dodajanje novih in odstranjevanje obstoječih točk.
* **Change Label** – preimenujemo trenutno izbrano skupino.
* **Dispose** – Skupino odstranimo iz pomnilnika.

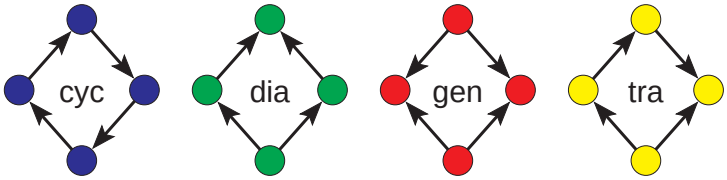
– **Hierarchy – H** (Hierarhija, privzeti podaljšek je .HIE)

* **Read** – Hierarhijo preberemo z vhodne (Ascii) datoteke. Primer hierarhije nad našim omrežjem: Točke najprej razdelimo v dve skupini (samoglasniki in soglasniki). V okviru soglasnikov pa točki b in c še dodatno združimo v novo skupino z imenom bc.
  ```
  +Root
    +soglasniki
      +bc
          * 2
          * 3
      * 4
    +samoglasniki
      * 1
      * 5
  ```
  Kot vidimo so skupine označene z znakom $+$, točke pa z znakom $*$. Gnezdenje je nakazano z zamiki (po dva presledka na nivo).
* **Save** – Hierarhijo shranimo na izhodno (Ascii) datoteko.
* **View/Edit** – Urejanje hierarhije: spreminjanje tipa vozlišč (npr. tip Close pomeni, da se točke, ki se nahajajo v izbranem vozlišču ali gnezdenih podvozliščih, pri stiskanju stisnejo v skupno točko) in spreminjanje imen vozlišč. Z izbiro Show Subtree dosežemo, da se ob zahtevi za izpis točk, ki pripadajo danemu vozlišču, izpišejo tudi vse točke, ki se nahajajo v ustreznem poddrevesu. Vozlišča hierarhije lahko premikamo navzgor in navzdol v okviru nivoja (Push Up in Push Down).
* **Change Label** – preimenujemo trenutno izbrano hierarhijo.
* **Dispose** – Hierarhijo odstranimo iz pomnilnika.

    - **Export as Dendrogram to [EPS, SVG + PDF]** – Narišemo dendrogram za dano dvojiško hierarhijo. Različnosti morajo biti shranjene v imenih vozlišč hierarhije med [in] (dobljeno avtomatično kot rezultat hierarhičnega razvrščanja ali razvrščanja z omejitvami).
  - **Pajek Project File –** Sestavljena datoteka, privzeti podaljšek je .PAJ
    - **Read** – Sestavljeno datoteko (datoteko, ki lahko vsebuje vse objekte: omrežja, razbitja, permutacije, skupine, hierarhije in vektorje) preberemo z vhodne (Ascii) datoteke.
    - **Save** – Vse trenutno naložene objekte shranimo kot sestavljeno datoteko.
  - **Ini File** – S pomočjo te izbire lahko shranimo trenutne nastavitve v Pajku (npr. velikosti oken, točk,...) in jih kdaj kasneje spet uporabimo.
    - **Load** – Preberi nastavitve za Pajka iz vhodne datoteke (*.ini).
    - **Save** – Shrani trenutne nastavitve v datoteko (*.ini).
  - **Exit** – Izhod iz programa.

- **Network** – operacije, ki kot vhod potrebujejo le omrežje (ali niti tega ne v primeru generiranja slučajnih omrežij). Uporabi se tisto omrežje, ki je trenutno izbrano v seznamu omrežij.
  - **Create Random Network** – Zgradimo slučajno omrežje z danim številom točk.
    - **Total No. of Arcs** – Določimo skupno število usmerjenih povezav.
    - **Vertices Output Degree** – Izhodne stopnje točk omejimo s spodnjo in zgornjo mejo.
    - **Bernoulli/Poisson** – Zgradimo slučajno omrežje po Bernoulli/Poisson-ovem modelu. Omrežje je lahko usmerjeno, neusmerjeno, dvodelno, aciklično ali dvovrstno.
    - **Scale Free** – Zgradimo 'Scale Free' slučajno omrežje (Barabasi, Albert). Omrežje je lahko usmerjeno, neusmerjeno ali aciklično.
    - **Small World** – Zgradimo 'Small World' slučajno omrežje (Batagelj, Brandes).
    - **Extended model** – Dodajanje točk, povezav in preklaplanje povezav so določeni z razširjenim modelom: R. Albert in A.-L. Barabasi: Topology of evolving networks: local events and universality. http://xxx.lanl.gov/abs/cond-mat/0005085
  - **Create New Network** – Iz danega omrežja zgradimo novo omrežje.

* **Empty Network -** Zgeneriramo omrežje z danim številom točk in brez povezav.
* **Complete Network -** Zgeneriramo neusmerjeno ali usmerjeno omrežje z danim številom točk in vsemi možnimi povezavami.
* **with Bi-Connected Components stored as Relation Numbers** – Dvopovezane komponente. Rezultati so:
  1. večrelacijsko omrežje kjer številka relacija pove številko dvopovezane komponentame;
  2. hierarhija z dvopovezanimi komponentami;
  3. razbitje s točkami, ki pripadajo natanko eni dvopovezani komponenti, točkami izven dvopovezanih komponent in presečišči (komponente so zaporedoma oštevilčene, v razredu 999999998 so presečišča, v razredu 0 pa so točke, ki ne pripadajo nobeni komponenti glede na zahtevano velikost komponente).
  4. vektor s presečišči (artikulacijskimi točkami): za vsako točko omrežja podamo v koliko dvopovezanih komponentah se nahaja;
* **with Ring Counts stored as Line Values** – Za vsako povezavo preštejemo koliko verigam (trikotnikom, štirikotnikom) pripada.
  - **3-Rings** – Za vsako povezavo preštejemo koliko trikotnikom pripada.
    1. **Undirected** – preštejemo število trikotnikov v neusmerjenem omrežju.
    2. **Directed** – preštejemo ciklične, tranzitivne ali vse trikotnike v usmerjenem omrežju, ali pa kolikokrat neka povezava nastopa kot bližnjica (shortcut).
  - **4-Rings** – Za vsako povezavo preštejemo koliko štirikotnikom pripada.
    1. **Undirected** – preštejemo število štirikotnikov v neusmerjenem omrežju.
    2. **Directed** – preštejemo ciklične, 'karo', genealoške, tranzitivne, ali vse štirikotnike v usmerjenem omrežju ali pa kolikokrat neka povezava nastopa kot bližnjica (shortcut). (slika 3).
* **SubNetwork with Paths** – Določitev poti med dvema točkama.
  - **One Shortest Path between Two Vertices** – Poiščemo najkrajšo pot med dvema točkama in jo izločimo kot novo omrežje. Pri tem se vrednosti na povezavah lahko upoštevajo ali pa se zanemarijo.



Slika 3: Ciklični (cyc), karo (dia), genealoški (gen) in tranzitivni (tra) štirikotniki.

- **All Shortest Paths between Two Vertices** – Poiščemo vse najkrajše poti med dvema točkama in jih izločimo kot novo omrežje. Vrednosti na povezavah se lahko upoštevajo ali pa se zanemarijo.
- **Walks with Limited Length between Two Vertices** – Poiščemo vse sprehode med dvema točkama, katerih dolžina je omejena z izbrano vrednostjo.
- **Geodesics Matrices*** – Izračuna matriko dolžin najkrajših poti med vsemi pari točk in matriko s številom najkrajših poti med vsemi pari točk (*samo za manjša omrežja!*)
- **Info on Diameter** – Poiščemo najdaljšo najkrajšo pot v omrežju.

* **SubNetwork with Flows** – Določitev pretokov med dvema točkama.
  - **Maximum Flow between Two Vertices** – Največji pretok med izbranima točkama. Rezultat je podomrežje, ki realizira ta pretok.
  - **Maximum Flow between Pairs in Cluster*** – Največji pretok med vsemi pari točk v skupini. Rezultat je omrežje, kjer vrednost na povezavi predstavlja največji pretok med odgovarjajočima točkama. Postopek je počasen, zato ga uporabljajmo samo na manjših omrežjih ali manjših podmnožicah točk.
* **Transform** – Pretvorbe danega omrežja.
  - **Transpose 1-Mode** – Transponirano omrežje danega enovrstnega omrežja – omrežje z obrnjenimi smermi puščic.
  - **Remove**
    1. **Selected Vertices** – Iz omrežja odstrani izbrane točke.
    2. **all Edges** – Iz omrežja odstranimo vse neusmerjene povezave.
    3. **all Arcs** – Iz omrežja odstranimo vse usmerjene povezave.
    4. **Multiple Lines** – Iz omrežja odstranimo vse večkratne povezave. Vrednost preostale povezave med dvema točkama postane lahko vsota, število povezav med točkama, najmanjša ali največja vred-

nost vseh vrednosti povezav, ki so obstajale med tema točkama, ali enostavno vrednost 1.

5. **Loops** – Iz omrežja odstranimo vse zanke.
6. **Lines with Value** – Iz omrežja odstranimo vse povezave z vrednostjo nižjo ali višjo od izbrane vrednosti ali vrednostjo v izbranem intervalu.
7. **all Arcs from each Vertex except k with Lowest (Highest) Line Values** – V vsaki točki uredimo povezave, ki imajo eno krajišče v tej točki v padajočem/naraščajočem vrstnem redu glede na vrednost povezave. Ohranimo samo izbrano število povezav z največjo/najmanjšo vrednostjo. Z izbiro **Keep Arcs with Value equal to the k-th Value** določimo, kaj storiti s povezavami, ki imajo vrednost natanko enako k-ti vrednosti po velikosti (te povezave zbrišemo ali ne).
8. **Triangle** – Iz omrežja odstranimo usmerjene povezave, ki pripadajo spodnjemu ali zgornjemu trikotniku.

· **Add**

1. **Vertices** – Naredimo kopijo izbranega omrežja, ki ji lahko dodamo tudi dodatne točke s stopnjo 0.
2. **Vertex Labels**
   (a) **Default** – Zamenjaj oznake točk s privzetimi (v1, v2...).
   (b) **from File(s)** – Privzete oznake točk ($v1$, $v2$,...) zamenjaj s pravimi oznakami z vhodne datoteke. V primeru dvovrstnih omrežij lahko izberemo dve datoteki z oznakami (za vsako množico drugo datoteko). To je še posebej pomembno pri dvovrstnih omrežjih, ki jih dobimo z množenji omrežij. V takih omrežjih so oznake za vsako množico shranjene v različnih datotekah.
   (c) **and Descriptions from File(s)** – Privzete oznake točk ($v1$, $v2$,...) zamenjaj s pravimi oznakami z vhodne datoteke. Z datoteke se preberejo tudi ostale lastnosti točk (oblike, barve...). Tudi tu lahko v primeru dvovrstnih omrežij izberemo dve datoteki z oznakami.
3. **Line Labels as Line Values** - Oznake na povezavah zamenjamo z vrednostmi na povezavah. Število decimalnih mest je določeno s prikazom v oknu Draw.
4. **Sibling edges** – Dodamo povezave tipa 'brat-sestra' med točkami s skupnim prednikom (**Input**) oziroma naslednikom (**Output**).

- **Edges –> Arcs** – Poljubno omrežje pretvorimo v usmerjeno omrežje: vse neusmerjene povezave pretvorimo v dvosmerne usmerjene povezave.
- **Arcs –>Edges**
  1. **All** – Poljubno omrežje pretvorimo v neusmerjeno omrežje: vse usmerjene povezave pretvorimo v neusmerjene.
  2. **Bidirected only** – V neusmerjene pretvorimo samo dvosmerne usmerjene povezave. Usmerjene povezave, ki gredo samo od ene točke do druge (in ne tudi nazaj) pustimo pri miru. Vrednost neusmerjene povezave postane lahko vsota, manjša ali večja vrednost vrednosti dvosmernih usmerjenih povezav.
- **Bidirected Arcs $\rightarrow$ Arc**
  1. **Select Min Value** – Če med točkama obstajata povezavi v obe smeri, ohranimo samo tisto z manjšo vrednostjo. Če sta obe vrednosti enaki, obe usmerjeni povezavi zamenjamo z neusmerjeno.
  2. **Select Max Value** – Če med točkama obstajata povezavi v obe smeri, ohranimo samo tisto z večjo vrednostjo. Če sta obe vrednosti enaki, obe usmerjeni povezavi zamenjamo z neusmerjeno.
- **Line Values** – Pretvorbe vrednosti na povezavah:
  1. **Set All Line Values to 1** – Postavi vrednosti vseh povezav na 1.
  2. **Recode** – Rekodiranje vrednosti na povezavah glede na izbrane intervale.
  3. **Multiply by** – Množenje vrednosti povezav s konstanto.
  4. **Add Constant** – Prištevanje konstante vsem vrednostim povezav.
  5. **Constant** – Vsaka povezava dobi vrednost, ki je enaka manjši (ali večji) vrednosti vrednosti povezave in izbrane konstante.
  6. **Absolute** – Absolutne vrednosti povezav.
  7. **Absolute + Sqrt** – Kvadratni koreni absolutnih vrednosti povezav.
  8. **Truncate** – Navzdol zaokrožene vrednosti povezav.
  9. **Exp** – Eksponent vrednosti povezav (e^x).
  10. **Ln** – Naravni logaritem vrednosti povezav ($\ln x$).
  11. **Power** – Izbrana potenca vrednosti povezav (x^r).
  12. **Normalize** – Normalizacije vrednosti povezav:
      Sum – Normalizacija vrednosti povezav, tako da je vsota vrednosti povezav 1.

Max – Normalizacija vrednosti povezav, tako da je največja vrednost 1.

13. **Replace Arc Values with Ranks** – V usmerjenem omrežju sortira povezave okoli točke po vrednosti povezave (naraščajoče ali padajoče) in vrednosti zamenja z rangi.

· **Reduction** – Redukcije danega omrežja.

1. **Degree** – Iz omrežja rekurzivno odstranimo vse točke s stopnjo nižjo od predpisane vrednosti. Operacijo lahko omejimo na izbrano skupino točk.
2. **Pathfinder*** – iz uteženega omrežja, kjer vrednosti na povezavah predstavljajo različnosti, odstranimo vse povezave, ki ne zadoščajo trikotniški neenekosti. Potrebno je vnesti še parameter r (realno število večje od 0), ki določa način izračunavanja razdalj (Minkowski distance):

$$a \boxed{\mathrm{r}} b = \sqrt[r]{a^r + b^r}$$

Ponavadi vzamemo $r = 1$, $r = 2$ ali $r = \infty$:

$$r = 1 \Rightarrow a \boxed{\mathrm{r}} b = a + b$$

$$r = 2 \Rightarrow a \boxed{\mathrm{r}} b = \sqrt{a^2 + b^2}$$

$$r = \infty \Rightarrow a \boxed{\mathrm{r}} b = \max(a, b).$$

Algoritem je, glede na časovno zahtevnost, smiselno uporabljati za omrežja kjer je število točk manjše od 10000. Podrobnosti: http://en.wikipedia.org/wiki/Pathfinder_network

3. **Hierarchy** – Iz omrežja rekurzivno odstranimo vse točke, ki imajo samo 0 ali 1 soseda. Rezultat te operacije je enostavnejše omrežje brez drevesnih točk in hierarhija, ki vsebuje odstranjene točke.
4. **Subdivisions** – Iz omrežja rekurzivno odstranimo vse točke, ki imajo natančno 2 soseda (odstranimo tudi odgovarjajoči povezavi). Namesto obeh odstranjenih povezav, dodamo novo povezavo, ki omenjena soseda poveže neposredno.
5. **Design (flow graph)** – Poenostavitev strukturiranih gradnikov v diagramu poteka (McCabe).

· **1-Mode to 2-Mode** – Omrežje pretvorimo v dvovrstno omrežje.

        - **Incidence Network** – Pretvori Pajkovo omrežje (*'adjacency matrix'*) v *'incidence matrix'*, ki se lahko nadalje uporabi za generiranje *'linegraph'*.
        - **Sort Lines** – preuredimo povezave, struktura omrežja se sicer ne spremeni.
            1. **Neighbors around Vertices** – V vsaki točki uredimo povezave, ki imajo eno krajišče v tej točki v naraščajočem vrstnem redu glede na drugo krajišče povezave.
            2. **Line Values** – Povezave uredimo po vrednosti naraščajoče ali padajoče.
            3. **Line Values around Vertices** – V vsaki točki uredimo povezave, ki imajo eno krajišče v tej točki, v naraščajočem ali padajočem vrstnem redu glede na vrednosti povezav.

– **Create Partition** – Določanje razbitij točk danega omrežja.

    * **Degree** – Razbitje po stopnjah točk.
        - **Input** – Vhodne stopnje točk.
        - **Output** – Izhodne stopnje točk.
        - **All** – Skupne (vhodne in izhodne) stopnje točk.
    * **Components** – Določitev komponent v omrežju.
        - **Weak** – Razbitje po šibko povezanih komponentah omrežja.
        - **Strong** – Razbitje po krepko povezanih komponentah omrežja.
        - **Strong-Periodic** – Podrobnejše razbitje krepko povezanih komponent omrežja – po periodah.
    * k**-Neighbours** – Določitev oddaljenosti vseh točk od dane točke.
        - **Input** – oddaljenosti v smeri nasprotni smeri povezav.
        - **Output** – oddaljenosti v smeri povezav.
        - **All** – oddaljenosti ne glede na smer povezav (vse povezave obravnavamo kot neusmerjene).

      Točke, ki so iz dane točke nedosegljive, uvrstimo v skupino s številko 999999998.
    * k**-Core** – k-jedra v omrežju glede na:
        - **Input** – povezave, ki prihajajo v točke.
        - **Output** – povezave, ki zapuščajo točke.
        - **All** – vse povezave.

* **Valued Core** – Posplošena k-jedra: namesto štetja povezav (sosedov) uporabimo vrednosti na povezavah. Pri tem se kot kriterij za pripadnost danemu jedru lahko upošteva vsota vrednosti povezav ali največja vrednost na povezavah znotraj jedra. Za razliko od običajnih jeder, kjer so meje zaporedna naravna števila, je potrebno mejne vrednosti pri tej vrsti jeder podati vnaprej. To lahko naredimo na dva načina:
  - **First Threshold and Step** – izberemo prvo mejno vrednost in korak v katerem se le ta povečuje.
  - **Selected Thresholds** – Vse mejne vrednosti podamo s pomočjo vektorja.

  Tudi pri posplošenih jedrih se lahko omejimo na povezave, ki prihajajo v točke, povezave, ki vodijo iz točk ali pa upoštevamo vse povezave.
* **Communities** – poišči skupnosti v podanem omrežju.
  - **Louvain Method** – Razbitje točk omrežja po algoritmu 'Louvain Communities Detection'. Pri tem se upoštevajo tudi vrednosti povezav. Glede na to ali je omrežje običajno ali označeno (vsaj ena povezava ima negativno vrednost) se izbere ustrezna verzija algoritma. Glede na standardno verzijo algoritma ima Pajkova implementacija še dodaten parameter - resolution. Vrednost 1 pomeni standarden algoritem, večje vrednosti vrnejo kot rezultat večje število skupin, manjše vrednosti pa manjše število skupin. Obstajata dve različici algoritma:
    1. **Multi-Level Coarsening + Single Refinement** – kjer se izvede 'refinement' samo za zadnje (najbolj grobo) dobljeno razbitje.
    2. **Multi-Level Coarsening + Multi-Level Refinement** – tu se iterativno ponavljata fazi 'coarsening' in 'refinement' na vseh dobljenih nivojih. 'Modularities', ki jih dobimo s to metodo, so praviloma višji. Podrobnosti o tem algoritmu so na voljo na strani: http://dl.acm.org/citation.cfm?id=1970376
  - **VOS Clustering** – Podobna metoda kot Louvain: obe metodi imata 'resolution parameter', po metodi Louvain se optimizira *modularity* medtem ko se po metodi VOS Clustering optimizira *VOS quality function*. Podrobnosti o tej metodi se lahko najdejo na spletni strani programa VOSviewer: http://www.vosviewer.com/. Algoritem obravnava vse povezave kot pozitivne, zato ni primeren za označena omrežja. Tudi tu obstajata dve različici algoritma:
    1. **Multi-Level Coarsening + Single Refinement** – kjer se izvede 'refinement' samo za zadnje (najbolj grobo) dobljeno razbitje.

2. **Multi-Level Coarsening + Multi-Level Refinement** – tu se iterativno ponavljata fazi 'coarsening' in 'refinement' na vseh dobljenih nivojih. 'VOS Qualities', ki jih dobimo s to metodo, so praviloma višji.

* **Islands - Line Weights** – Razbitje točk na kohezivne skupine glede na vrednosti povezav – povezave znotraj skupin morajo biti močnejše kot povezave navzven. Postopek vrne tudi vektor, ki določa višine točk – največjo vrednost povezave sosednih povezav. Če je označeno **Generate Network with Islands**, se zgenerira tudi novo omrežje, ki vsebuje le povezave, ki so nad 'gladino' otoka.
* **Blockmodeling*** – Posplošeno bločno modeliranje običajnih in dvovrstnih omrežij.
  - **Random Start** – Začemo s slučajnimi začetnimi razbitji in jih optimiziramo.
  - **Optimize Partition** – Izračunamo napako podanega razbitja in ga optimiziramo.
  - **Restricted Options** – Prikažemo del ali vse možne nastavitve.
* **Vertex Labels** – Razbitje točk omrežja po oznakah točk – točke z enako oznako uvrstimo v isto skupino (npr. označitev atomov v molekuli).
* **Vertex Labels matching Regular Expression** – Razbitje točk omrežja glede na vnešen regularni izraz. Nekaj primerov regularnih izrazov:
  - ^D - oznake točk, ki se začnejo z 'D';
  - a$ - oznake točk, ki se končajo z 'a';
  - edu - oznake točk, ki vsebujejo 'edu'.

  Točke, pri katerih oznaka ustreza regularnemu izrazu gredo v razred 1, ostale v razred 0. Več o regularnih izrazih:
  `http://docwiki.embarcadero.com/RADStudio/Seattle/en/Regular_Expressions`
* **Vertex Shapes** – Razbitje točk po obliki točk – točke z enako obliko uvrstimo v isto skupino (npr. označitev spola v rodovniku: moški – trikotnik, ženske – krog).
* **Bow-Tie** – Razbitje točk usmerjenega omrežja (struktura WWW) v naslednje razrede: 1 – LSCC, 2 – IN, 3 – OUT, 4 – TUBES, 5 – TENDRILS, 0 – OTHERS.
* **Default Labels Partition** – Za to operacijo potrebujemo omrežje s standardnimi oznakami npr., $v3$, $v9$,... Rezultat je razbitje izbrane velikosti, kjer gredo točke določene s številkami, ki so shranjene v oz-

nakah (npr., 3, 9,...) v razred 1, ostale točke pa v razred 0.
Operacija je pride prav, če na omrežju naredimo več operacij (npr. izrezov) in želimo zgenerirati še primerljive ostale objekte (razbitja, vektorje...).

* **p-Cliques** – Razbitje točk acikličnega omrežja glede na p-klike (določitev takih skupin, da imajo vse točke v skupini vsaj delež p ($0 \leq p \leq 1$) sosedov znotraj skupine. Obstajate dve verziji algoritma:
  - **Strong** – za usmerjena omrežja.
  - **Weak** – za neusmerjena omrežja.

– **Create Vector** – Določitev vektorja iz omrežja.

* **Centrality** – Izračun nekaterih mer središčnosti točk in mer usredinjenosti omrežja.
  - **Degree** – Vektor stopenj točk.
    1. **Input** – Upoštevajo naj se samo vhodne stopnje točk.
    2. **Output** – Upoštevajo naj se samo izhodne stopnje točk.
    3. **All** – Upoštevajo naj se skupne (vhodne in izhodne) stopnje točk.
  - **Weighted Degree** – Stopnje z upoštevanjem vrednosti na povezavah: Sešteje vrednosti vseh povezav ki prihajajo v točko, gredo iz točke, ali vrednosti vseh povezav povezanih s točko.
  - **Closeness** – mera središčnosti glede na dostopnost (Sabidussi). Lahko jo definiramo glede na oddaljenost dane točke od vseh drugih (**Input**) ali oddaljenosti vseh drugih točk od dane točke (**Output**). Pravtako pa lahko mero definiramo ne glede na smer povezav – smeri povezav zanemarimo (**All**).
  - **Betweenness** – mera središčnosti glede na vmesnost (Freeman).
  - **Hubs-Authorities** – V usmerjenih omrežjih si lahko predstavljamo, da vsaka točka igra dvojno vlogo. Ponavadi točka nekaj opisuje (je opis in se druge nanjo sklicujejo), lahko pa tudi kaže na eno ali več drugih točk v omrežju (je kazalo). Točka je dobro *kazalo (hub)*, če kaže nanjo veliko dobrih opisov in je dober *opis (authority)*, če nanjo kaže veliko dobrih kazal. V dobljenem razbitju vrednost 1 pomeni, da je točko dober opis, vrednost 2 pomeni, da je točka dober opis in dobro kazalo, vrednost 3 pa pomeni, da je točka dobro kazalo.
  - **Proximity Prestige** – za vsako točko določimo njeno območje

(vpliva ali podpore) glede na vhodne, izhodne ali vse sosede. Rezultati so:

1. Razbitje, ki vsebuje število točk v območju vpliva / podpore (število točk, ki so dosegljive oziroma iz katerih dosežemo dano točko).
2. Vektor, ki vsebuje normalizirano število točk v območju vpliva / podpore – normalizacija je s številom vseh točk - 1.
3. Vektor, ki vsebuje povprečno oddaljenost do/od vseh točk v območju vpliva / podpore.
4. Vektor, ki predstavlja Proximity prestige (bližino).

· **Laplace** – Laplacova središčnost. Podrobnosti: http://www.scirp.org/journal/PaperDownload.aspx?paperID=27402

· **Line Values**

1. **Min** – Poišče najmanjšo vrednost od vrednosti povezav ki prihajajo v točko, gredo iz točke, ali vseh povezav povezanih s točko.
2. **Max** – Poišče največjo vrednost od vrednosti povezav ki prihajajo v točko, gredo iz točke, ali vseh povezav povezanih s točko.

· **Centers** – Določitev centrov omrežja: točk, ki so glede na stopnje močnejše od svojih sosedov.

∗ **Clustering Coefficients** – Mere nakopičenosti v točkah omrežja: Naj bo deg_v stopnja točke v, $|E(G^1(v))|$ število povezav med točkmi v 1-okolici točke v, Δ največja stopnja točke v omrežju in $|E(G^2(v))|$ število povezav med točkmi v 2-okolici točke v.

· **CC1** – mere, ki upoštevajo le 1-okolico točke:

$$CC_1(v) = \frac{2|E(G^1(v))|}{deg_v(deg_v - 1)}$$

$$CC_1\prime(v) = \frac{deg_v}{\Delta} CC_1(v)$$

$CC_1(v)$ je *Clustering Coefficient*, kot sta ga definirala Watts in Strogatz, 1998.

· **CC2** – mere, ki upoštevajo 2-okolico točke:

$$CC_2(v) = \frac{|E(G^1(v))|}{|E(G^2(v))|}$$

$$CC_2\prime(v) = \frac{deg_v}{\Delta} CC_2(v)$$

Če je $deg_v \leq 1$, zavzamejo vsi koeficienti za točko v vrednost 999999998 (manjkajoča vrednost). V oknu Report se izpišeta tudi Watts-Strogatz Clustering Coefficient (Transitivity) in Network Clustering Coefficient.

* **Structural Holes** – Burtova mera omejitev. Rezultati so:
  - omrežje p_{ij} z deleži vrednosti povezav točke i s točko j glede na skupno vrednost povezav točke i.

    $$p_{ij} = \frac{a_{ij} + a_{ji}}{\sum_k a_{ik} + a_{ki}}$$

    kjer je a_{ij} vrednost povezave od i do j.
  - omrežje, ki vsebuje *dyadic constraint* – c_{ij}:

    $$c_{ij} = (p_{ij} + \sum_k p_{ik} p_{kj})^2$$

  - vektor, ki vsebuje *aggregate constraint* – C_i:

    $$C_i = \sum_j c_{ij}$$

* **Generalized Core** – posplošena jedra.
  - **Degree** – običajna jedra.
  - **Sum** – upoštevamo vrednosti na povezavah (vsota vrednosti na povezavah znotraj jedra).
  - **Max** – upoštevamo vrednosti na povezavah (največja vrednost na povezavah znotraj jedra).
* **Distribution of Distances*** – Vrne porazdelitev dolžin najkrajših poti med vsemi (dosegljivimi) pari točk. Za začetne točke iz katerih računa najkrajše poti vzamemo vse točke omrežja.
* **Get Loops** – Shrani vrednosti zank v vektor.
* **Get Coordinate** – Izloči x, y, z ali pa vse koordinate točke v vektor(je).

– **Create Permutation** – Določitev oštevilčenja točk omrežja.

* **Depth First** – Oštevilčenje omrežja glede na obhod v globino:
  - **Strong** – usmerjen obhod.
  - **Weak** – neusmerjen obhod.
* **Breadth First** – Oštevilčenje omrežja glede na obhod v širino:
  - **Strong** – usmerjen obhod.

  - **Weak** – neusmerjen obhod.
* **Reverse Cuthill-McKee** – RCM oštevilčenje.
  http://www.osl.iu.edu/~chemuell/projects/bioinf/sparse-matrix-clustering-chris-mueller.pdf
* **Core + Degree** – Oštevilčenje v padajočem vrstnem redu glede na jedra. Znotraj jeder so točke urejene v padajočem vrstnem redu glede na število vseh sosedov, ki se nahajajo v istem ali višjem jedru.

– **Create Hierarchy** – Hierarhična dekompozicija omrežja.

* **Clustering*** – Hierarhično združevanje v skupine. Vhod je matrika (omrežje) različnosti, ki jo lahko dobimo z uporabo Operations/ Cluster/ Dissimilarity ali pa preberemo iz vhodne datoteke.
  - **Options** – Izbereemo metodo hierarhičnega združevanja (general, minimum, maximum, average, ward ali ward2).
  - **Run** – Poženemo postopek hierarhičnega združevanja. Rezultat je hierarhija gnezdenih skupin in slika dendrograma v EPS.
* **Symmetric-Acyclic** – Simetrično aciklična dekompozicija omrežja. Rezultat je hierarhija gnezdenih skupin.
* **Clustering with Relational Constraint** – Hierarhično združevanje v skupine z relacijsko omejitvijo. Podrobnosti:
  Ferligoj A., Batagelj V. (1983): Some types of clustering with relational constraints. *Psychometrika*, **48(4)**, 541-552.
  V izračunu se upoštevajo samo različnosti med točkami, ki so neposredno povezane, kar omogoča, da poiščemo razvrstitve tudi za zelo velika omrežja. Vhod v proceduro je omrežje z različnostmi, ki jih dobimo z *Operations/Cluster/Dissimilarity/Network* ali *Vector based* ali pa preberemo z datoteke.
  - **Run** – Rezultati so: razbitje, ki predstavlja drevo razvrščanja: očete vozlišč hierarhije; dva vektorja z višinami točk in številom točk v poddrevesu. Dobljeni objekti ne morejo biti direktno uporabljeni, npr pri risanju omrežja, ampak moramo predhodno uporabiti *Make Partition*.
    Options – Izbira metode združevanja (*minimum*, *maximum* ali *average*) in strategije (*strict*, *leader* ali *tolerant*).
  - **Make Partition** – Iz zgoraj dobljenega razbitja, ki predstavlja drevo in enega od obeh vektorjev zgeneriramo razbitje, primerljivo z originalnim omrežjem:
    using Threshold determined by Vector – Iz dobljenega razbitja, ki predstavlja drevo in enega od obeh dobljenih vektorjev (vsi

imajo razsežnost 2*n-1) zgradimo razbitje, ki je združljivo z originalnim omrežjem, s tem da podamo vrednost za prag.
with selected Size of Clusters - Iz dobljenega razbitja, ki predstavlja drevo in podane velikosti skupin zgradimo razbitje, ki je združljivo z originalnim omrežjem.

· **Extract Subtree as Hierarchy** – Izloči poddrevo, ki pripada določenemu vozlišču kot Pajkovo hierarhijo.

– **2-Mode Network** – Operacije, ki jih lahko izvajamo samo nad dvovrstnimi omrežji.

  * **Partition into 2 Modes** – Razbitje točk dvovrstnega omrežja na prvo in drugo množico.
  * **Transpose 2-Mode** – Zamenjamo vrstice in stolpce dvovrstnega omrežja.
  * **2-Mode to 1-Mode** – Dvodelno omrežje (npr. omrežje vključenosti oseb v dogodke) pretvorimo v običajno omrežje. Primer dvodelnega omrežja, ki predstavlja vključenost treh oseb v pet dogodkov:

```
*Vertices  8 3
1 "Actor 1"
2 "Actor 2"
3 "Actor 3"
4 "Event 1"
5 "Event 2"
6 "Event 3"
7 "Event 4"
8 "Event 5"
*Edges
1 4
1 5
2 4
2 5
2 6
2 8
3 4
3 7
3 8
```

Prva številka za besedo *Vertices (v našem primeru 8) pomeni število vseh točk v omrežju, druga (3) pa moč prve množice (število oseb). Pri branju dvodelnega omrežja si zapomnimo, da je omrežje dvovrstno.

Pajek prepozna tudi dvodelna omrežja shranjena v obliki zapisa Ucinet (npr. omrežje Davis.dat).
Pri pretvorbi v enodelno omrežje dobimo za rezultat omrežje z vrednostmi na povezavah. Predpostavimo da prva množica predstavlja osebe, druga pa dogodke. Iz dvodelnega omrežja lahko dobimo dve običajni omrežji:
Rows – Točke novodobljenega omrežja predstavljajo osebe. Vrednost povezave med dvema točkama predstavlja število dogodkov, katerih sta se udeležili obe osebi hkrati. Razsežnost dobljenega omrežja je enaka številu oseb.
Columns – Točke novodobljenega omrežja predstavljajo dogodke. Vrednost povezave med dvema točkama predstavlja število oseb, ki so se udeležile obeh dogodkov. Razsežnost dobljenega omrežja je enaka številu dogodkov.
Include Loops – Dodamo tudi zanke, kjer vrednost na povezavi pomeni število dogodkov, ki se jih je neka oseba udeležila (Rows), oziroma število oseb, ki so se udeležile nekega dogodka (Cols).
Multiple Lines – Namesto omrežja z vrednostmi na povezavah zahtevamo, da se zgenerira običajno omrežje z (morebitnimi) večkratnimi povezavami. Zgenerirana povezava dobi oznako, ki je enaka oznaki dogodka/osebe, ki je povzročila njen nastanek. Če je izbrano tudi razbitje, ki po velikosti ustreza omrežju, se lahko zgenerira večrelacijsko omrežje.
Normalize 1-Mode – Normaliziramo omrežje, ki ga dobimo iz dvodelnega omrežja. Pri pretvorbi dvodelnega omrežja v običajno omrežje moramo imeti vključeno opcijo **include loops** in izključeno **multiple lines**. Obstajajo naslednje možnosti za normalizacijo:

$$Geo[i,j] = \frac{A[i,j]}{\sqrt{A[i,i] * A[j,j]}}$$

$$Input[i,j] = \frac{A[i,j]}{A[j,j]} \qquad Output[i,j] = \frac{A[i,j]}{A[i,i]}$$

$$Min[i,j] = \frac{A[i,j]}{\min(A[i,i], A[j,j])}$$

$$Max[i,j] = \frac{A[i,j]}{\max(A[i,i], A[j,j])}$$

$$MinDir[i,j] = \begin{cases} \frac{A[i,j]}{A[i,i]} & A[i,i] \leq A[j,j] \\ 0 & \text{otherwise} \end{cases}$$

$$MaxDir[i,j] = \begin{cases} \frac{A[i,j]}{A[j,j]} & A[i,i] \leq A[j,j] \\ 0 & \text{otherwise} \end{cases}$$

Rows=Cols – To je tretja možna pretvorba v običajno omrežje, kjer morata biti obe množici začetnega dvovrstnega omrežja isti.
Cols=0 – Pretvorba dvovrstnega omrežja v enovrstmo, kjer postavimo število točk v drugi množici na 0. Rezultat je enak, kot če bi npr. na vhodni datoteki zamenjali `*Vertices 32 18` z `*Vertices 32`.

- **Core** – Razbitje dvovrstnega omrežja glede na jedra.
  - **2-Mode** – jedro dvovrstnega omrežja. Za podano najmanjšo stopnjo točk prve (k_1) in najmanjšo stopnjo točk druge podmnožice (k_2) zgeneriramo razbitje kjer vrednost 0 pomeni, da točka ne pripada jedru, vrednost 1 pa pomeni, da točka pripada jedru.
  - **2-Mode Review** – Za podani začetni vrednosti k_1 in k_2 se izpišejo naslednji podatki:
    k_1 k_2 Rows Cols Comp
    kjer je k_1 najmanjša stopnja v prvi, k_2 najmanjša stopnja v drugi podmnožici, $Rows$ in $Cols$ sta števili točk v prvi in drugi podmnožici, $Comp$ pa je število povezanih komponent v omrežju določenim s k_1 in k_2. k_1 in k_2 povečujemo za 1 dokler rezultat ni prazno omrežje.
  - **2-Mode Border** – Izračunamo samo robne vrednosti $k1$ in $k2$ za dano dvovrstno omrežje.
  - **Valued 2-Mode Core** – Jedra dvovrstnega omrežja glede na vrednosti na povezavah. Za podano najmanjšo uteženo stopnjo (k_1) v prvi in najmanjšo uteženo stopnjo (k_2) v drugi množici se zgenerira novo razbitje kjer vrednost 0 pomeni, da točka ne pripada temu jedru, vrednost 1 pa pomeni, da točka pripada temu jedru.
- **Important Vertices** – Posplošitev kazal in opisov na dvovrstna omrežja – poiščemo pomembne točke iz prve in druge podmnožice.
- **Info** – Osnovne informacije o dvovrstnem omrežju - npr. gostota dvovrstnega omrežja.

– **Multiple Relations Network** – Operacije, ki jih lahko izvajamo samo nad večrelacijskimi omrežji.

- **Select Relation(s) into One Network** – V omrežju ohrani samo izbrano relacijo (izbrane relacije). Rezultat je le eno omrežje z izbranimi relacijami.
- **Extract Relation(s) into Separate Network(s)** – Izloči eno ali več

relacij iz večrelacijskega omrežja. Za vsako relacijo se zgenerira svoje omrežje.

* **Canonical Numbering** – Oštevilči relacije z zaporednimi številkami 1, 2,...
* **Generate 3-Mode Network** – zgradi trovrstno omrežje iz običajnega ali dvovrstnega večrelacijskega omrežja. Za vsako povezavo v večrelacijskem omrežju `r: i j v` (povezava od `i` do `j` z vrednostjo `v`, in številko relacije `r`) zgeneriraj naslednje tri povezave (trikotnik):
  za običajna omrežja:
  ```
  i N+j v
  i 2N+r v
  N+j 2N+r v
  ```
  za dvovrstna omrežja:
  ```
  i j v
  i N+M+r v
  j N+M+r v
  ```
  kjer je `N` moč prve množice in `M` moč druge množice.
* **Line Values** $->$ **Relation Numbers** – Shrani vrednosti povezav v številke relacij (zaokrožene pozitivne vrednosti).
* **Relation Numbers** $->$ **Line Values** – Shrani številke relacij v vrednosti povezav.
* **Change Relation Number - Label** – Spremeni staro številko relacije v novo in ji dodeli novo oznako.
* **Info** – Osnovne informacije o večrelacijskem omrežju: število relacij, število usmerjenih, neusmerjenih in vseh povezav.

– **Acyclic Network** – Operacije, ki jih lahko izvajamo samo nad acikličnimi omrežji.

* **Create Partition** – Izračun razbitij acikličnega omrežja.
  · **Depth Partition** – Določanje globin v acikličnih omrežjih.
    1. **Acyclic** – Razbitje točk acikličnega omrežja glede na običajne globine točk, to je oddaljenosti od začetnih točk, ki jim pripišemo globino 1.
    2. **Genealogical** – Razbitje točk acikličnega omrežja (rodovnika) po generacijah.
    3. **Generational** – Razbitje točk acikličnega omrežja (rodovnika) po generacijah (druga možnost).

- **Select First Vertices** – V razbitju označimo začetne točke omrežja z zaporednimi števili od 1 naprej. Dobljeno razbitje lahko kasneje uporabimo kot vhod za določanje genetske strukture omrežja glede na dano razbitje **Genetic Structure**.
- **Select Last Vertices** – V razbitju označimo končne točke omrežja z zaporednimi števili od 1 naprej.

* **Create Weighted Network + Vector** – Izračuna uteži (pomembnosti) točk (člankov) in povezav (citiranj) v acikličnem omrežju.
  1. **Traversal Weights** – Določitev vplivnosti (uteži) člankov in citiranj v acikličnem omrežju citiranj glede na število vseh poti, ki vodi skozi dano točko, oziroma uporablja dano povezavo. Rezultati so:
     - Omrežje z vrednostmi na povezavah, ki ustrezajo vplivnosti citiranj.
     - Vektor, ki predstavlja število vseh različnih poti iz začetne točke v dano točko.
     - Vektor, ki predstavlja število vseh različnih poti iz dane točke do končne točke.
     - Vektor, ki ustreza vplivnosti člankov.

     Vrednosti so lahko normirane (z vrednostjo pretoka ali največjo vrednostjo) ali pa logaritmirane. Obstaja več različic algoritma:
     - **Search Path Count (SPC)** – povezave štejemo le od začetne do končne točke (Source-Sink).
     - **Search Path Link Count (SPLC)** – vsako točko obravnavamo kot začetek (Vertex-Sink).
     - **Search Path Node Pair (SPNP)** – število povezanih parov točk.
  2. **Probabilistic Flow** – Rezultat je vektor in novo omrežje z vrednostmi na povezavah. Vrednosti v vektorju povejo, kakšna je verjetnost, da gre slučajen sprehod, ki se začne v začetnih točkah omrežja, skozi izbrano točko. Vrednosti na povezavah novo zgeneriranega omrežja pa povejo, kakšna je verjetnost, da gre slučajen sprehod, ki se začne v začetnih točkah omrežja, skozi izbrano povezavo.
* **Create (Sub)Network** – Poiščemo pomembne poti v uteženem acikličnem omrežju.
  - **Main paths** – V omrežju citiranj z izračunanimi utežmi (ali kakem drugem acikličnem omrežju z vrednostmi na povezavah) izračunamo

glavne poti. Podrobnosti so na voljo v članku:
Liu, J.S. and L.Y.Y. Lu (2012): An Integrated Approach for Main Path Analysis: Development of the Hirsch Index as an Example. *Journal of the American Society for Information Science and Technology*, 63, 528-542
http://onlinelibrary.wiley.com/doi/10.1002/asi.21692/abstract

1. **Local Search** – v vsakem koraku iskanja izberemo povezave z največjo utežjo, ki so sosednje trenutni povezavi. Lahko uporabimo tudi toleranco (število med 0 in 1). Toleranca 0 pomeni, da v vsakem koraku upoštevamo samo povezave z največjo utežjo, medtem ko večje tolerance pomenijo, da upoštevamo tudi povezave z malenkost manjšimi utežmi.
   (a) **Forward** – poiščemo lokalne glavne poti od začetnih do končnih točk.
   (b) **Backward** – poiščemo lokalne glavne poti v nasprotni smeri (od končnih do začetnih točk).
   (c) **Key-Route** – izberemo nekaj (najpomembnejših) povezav (key-routes) in poiščemo glavne poti iz obeh koncev teh povezav.
   (d) **Through Vertices in Cluster** – poiščemo lokalne glavne poti, ki vodijo skozi točke, ki se nahajajo v izbrani skupini (Cluster).
2. **Global Search** – poiščemo glavne poti z največjo skupno vsoto povezav na poti.
   (a) **Standard** – poiščemo glavne poti od začetnih do končnih točk z največjo skupno vsoto povezav na poti. Ta postopek se imenuje tudi Metoda kritične poti.
   (b) **Key-Route** – poiščemo vse glavne poti, ki vsebujejo izbrane povezave (key-routes) z največjo skupno vsoto povezav na poti.
   (c) **Through Vertices in Cluster** – poiščemo glavne poti z največjo skupno vsoto povezav, ki vodijo skozi točke, ki se nahajajo v izbrani skupini (Cluster).
3. **Mark Main Paths as Multirelational Network** – Začetno omrežje (omrežje z utežmi na povezavah) se pretvori v večrelacijsko omrežje: povezave, ki ležijo na glavnih poteh, dobijo številko relacije 2, ostale pa številko relacije 1.

Rezultat iskanja je tudi dvojiško razbitje, kjer so točke, ki ležijo na glavnih poteh, v skupini 1, vse ostale točke pa v skupini 0.

- **Critical Path Method (CPM)** – Metoda kritične poti: V acikličnem omrežju poiščemo kritično pot. Poleg tega izračunamo za vsako povezavo (aktivnost) celotno in prosto dovoljeno zamujanje - dve večrelacijski omrežji (povezave, ki ležijo na kritični poti dobijo številko relacije 2). Za vsako točko (stanje) dobimo še najzgodnejši možni in najkasnejši dovoljeni čas prihoda v dano točko.

* **Create Permutation** – Iz (skoraj) acikličnega omrežja zgeneriramo permutacijo.
  - **Topological Ordering** – Topološka urejenost točk acikličnega omrežja.
  - **Becker's Heuristic** – Beckerjeva heuristika za približek linearne urejenosti. Podobna je topološki urejenosti, a se lahko uporablja tudi za 'rahlo' ciklična omrežja. Namenjena je generiranju permutacij za usmerjena omrežja z velikim številom majhnih krepko povezanih komponent.
* **Transform** – pretvorbe (skoraj) acikličnih omrežij.
  - **Add Source and Sink** – Acikličnemu omrežju dodamo enolično začetno in enolično končno točko.
  - **Preprint Transformation** – Pretvori skoraj aciklično omrežje v pravo aciklično omrežje: Za vsak članek u iz krepke komponente se naredi duplikat - preprint u'. Članki znotraj krepke komponente citirajo preprinte, preprinti citirajo izhodne članke. Vsaka krepka komponenta se nadomesti z ustreznim popolnim usmerjenim dvodelnim podomrežjem članki $\times$ preprints.
* **+ Partition** – Za izvedbo teh operacij potrebujemo poleg acikličnega omrežja še razbitje.
  1. **Create Vectors with Genetic Structure** – Genetska struktura točk acikličnega omrežja glede na dano razbitje začetnih točk (točk brez predhodnikov). To razbitje lahko dobimo z ukazom **Network / Acyclic Network / Create Partition / Select First Vertices**. Za rezultat dobimo toliko vektorjev, kolikor je razredov v razbitju. Zato previdno: če je število različnih razredov v razbitju veliko, lahko postopek traja precej dolgo. Poleg omenjenih vektorjev dobimo za dodaten rezultat še dominantno razbitje in dominanten vektor.
* **Info** – Izpiše število prvih in zadnjih točk, najmanjšo in največjo stopnjo, število komponent, velikost največje komponente in v primeru parnih rodovnikov tudi indeks prepletenosti.

– **Temporal Network** – Operacije, ki jih lahko izvajamo samo nad časovnimi omrežji.

  * **Generate in Time** – Iz časovnega omrežja zgradimo omrežja v danih časovnih trenutkih ali omrežje v danem intervalu. Vnesti moramo prvi čas, zadnji čas in korak (cela števila). V časovnem omrežju morajo biti podani časi kdaj so določene točke in povezave aktivne. Te čase podamo med znakoma [in] takole:
    · znak - pomeni interval;
    · znak , je ločilo med intervali;
    · * pomeni neskončno (od danega trenutka dalje).

    Primer:

```
*Vertices 3
1 "a" [5-10,12-14]
2 "b" [1-3,7]
3 "e" [4-*]
*Edges
1 2 1 [7]
1 3 1 [6-8]
```

    Točka a je aktivna od časa 5 do 10 in od 12 do 14, točka b od časa 1 do 3 in v času 7, točka e pa od časa 4 naprej. Povezava od 1 do 2 je aktivna samo v času 7, povezava 1 do 3 pa od časa 6 do 8.
    Dodatna omejitev: ko gradimo trenutna omrežja, se določena povezava ne ustvari (tudi če je aktivna), če nista aktivni tudi krajišči povezave. Časovne oznake se morajo vedno nahajati na koncu vrstice, kjer so definirane točke oziroma povezave.
    Omrežja v izbranih trenutkih (intervalu) lahko gradimo na več načinov:
    · **All** – Zgradijo se vsa omrežja v izbranih trenutkih.
    · **Only Different** – V danem trenutku se zgradi omrežje le, če se bo novo zgrajeno omrežje razlikovalo od zadnjega zgrajenega vsaj v eni točki ali povezavi.
    · **Interval** – Zgradimo omrežje v danem časovnem intervalu.
  * **Info** – Izpis števila točk in povezav v vsaki časovni točki časovnega omrežja.

– **Signed Network** – Operacije, ki jih lahko izvajamo samo nad označenimi omrežji (omrežji s pozitivnimi in negativnimi povezavami).

    - **Set All Positive Line Values to 1, Negative to -1** – Postavi vrednosti vseh povezav, ki imajo pozitivno vrednost, na 1 in vrednosti vseh povezav, ki imajo negativno vrednost, na -1.
    - **Create Partition** – Iz označenega omrežja (in začetnega slučajnega razbitja) zgeneriramo razbitje.
        - **Doreian-Mrvar method*** – Razvrstitev točk označenega grafa v dano število skupin – iščemo čimboljšo razvrstitev točk grafa v dano število skupin, tako da bo čimveč (če že ne vse) povezav znotraj skupin pozitivnih, čimveč (če že ne vse) povezav med skupinami pa negativnih. Lahko uporabimo tudi drugi algoritem, ki ne razločuje med diagonalnimi in izvendiagonalnimi bloki: vsak blok je lahko pozitiven, negativen ali prazen (null). Podrobnejši opis algoritma je na voljo v: Doreian P., Mrvar A. (1996): A Partitioning Approach to Structural Balance. *Social Networks* (**18**). 149-168.

          Ukaz lahko uporabimo tudi za dvovrstna omrežja: vhod mora biti dvovrstno razbitje. V tem primeru skuša algoritem poiskati čimbolj čiste pozitivne, negativne in prazne bloke.

          Če postavimo kazen za majhne prazne bloke na neko pozitivno vrednost, potem skušamo poiskati čimvečje prazne bloke.
        - **Louvain method** – Metoda Louvain posplošena na označena omrežja.
    - **Info** – Izpiše nekonsistentnost glede na strukturno uravnoteženost, nekonsistentnost glede označeno modularnost danega razbitja ter oba označena indeksa E-I (pozitivnega in negativnega).

– **Info** – Izpis osnovnih podatkov o omrežju.

    - **General** – Izpis splošnih podatkov o omrežju: število točk, število usmerjenih povezav, število neusmerjenih povezav, število zank, gostota omrežja in povprečna stopnja, nekaj največjih oziroma najmanjših vrednosti na povezavah.
    - **Line Values** – Izpis frekvenčne porazdelitve vrednosti na povezavah.
    - **Triadic Census** – Izpis števila vseh različnih triad (omrežij s tremi točkami) v omrežju (Faust & Wasserman). Izkaže se, da je število vseh različnih triad 16. Algoritem prešteje pojavitve vseh triad v omrežju in jih oštevilči na način, ki je prikazan na sliki 4 (stran 55).
    - **Degree Assortativity** – indeksi 'asortativnosti' omrežja.
    - **Vertex Label -> Vertex Number** – Poišči številko točke, ki ustreza točki z dano oznako, ali poišči oznako točke, ki pripada podani številki točke.

    - **Line -> Rank of its Value** – Za podano povezavo izpiše njeno zaporedno številko glede na vrednost povezave.
    - **Indices** – Izpis različnih kemijskih indeksov omrežja.

- **Networks** – Operacije, ki za vhod potrebujejo dve omrežji. Pred izvedbo katerekoli operacije iz te skupine moramo najprej izbrati obe omrežji.

  – **Union of Lines** – Predpostavimo, da sta izbrani omrežji definirani na isti množici točk. Rezultat operacije je novo omrežje, kjer so povezave sestavljene iz povezav prvega in drugega omrežja (povezave drugega omrežja se dodajo prvemu). Rezultat je predstavljen kot večrelacijsko omrežje (povezave se razlikujejo).

  – **Union of Vertices** – Celotno drugo omrežje (točke in povezave) dodamo ('prilepimo') prvemu omrežju. Rezultat je omrežje, kjer je število točk enako vsoti števila točk prvega in drugega omrežja. Število šibko povezanih komponent dobljenega omrežja je enako vsoti števila šibko povezanih komponent izbranih omrežij.

  – **Cross-Intersection** – Predpostavimo, da sta izbrani omrežji definirani na isti množici točk. Rezultat operacije je novo omrežje, ki vsebuje tiste povezave iz prvega omrežja, ki obstajajo tudi v drugem omrežju (presek povezav). Vrednost na povezavah je lahko vsota, razlika, produkt, kvocient, minimum ali maksimum vrednosti na prvotnih povezavah. Lahko vzememo tudi vrednost iz prvega ali vrednost iz drugega omrežja.

  – **Intersection of Multiple Relations Networks** – Presek povezav, ki upošteva tudi številke relacij.

  – **Cross-Difference** – Tudi tu predpostavimo, da sta izbrani omrežji definirani na isti množici točk. Rezultat operacije je novo omrežje, ki vsebuje tiste povezave iz prvega omrežja, ki ne obstajajo v drugem omrežju (razlika množic povezav).

  – **Difference of Multiple Relations Networks** – Razlika povezav, ki upošteva tudi številke relacij.

  – **Multiply Networks** – zmnoži prvo in drugo eno ali dvovrstno omrežje (kjer razsežnosti zadostujejo kriterijem za množenje). Če je eno omrežje dvovrstno drugo pa enovrstno, moramo enovrstno omrežje pred množenjem spremeniti v dvovrstno (**Network/Create New Network/1-Mode to 2-Mode**).

– **Multiply Networks - Get Loops only** - vrne samo diagonalo matrike, ki jo dobimo kot rezultat množenja dveh enovrstnih omrežij (ali dveh dvovrstnih omrežij, za kateri je rezultat enovrstno omrežje).

– **Fragment (First in Second)** – V drugem omrežju poiščemo vse vzorce, ki so določeni s prvim omrežjem,.

  * **Induced** – Iščemo samo inducirana podomrežja.
  * **Negative lines inside fragment not allowed** – V osnovnem omrežju med točkami v vzorcu ne sme biti negativnih povezav. Negativna povezava pomeni, da dve točki ne smeta nastopati skupaj v istem vzorcu. Ta izbira je vidna samo v primeru, da *Induced* ni izbrano.
  * **Labeled** – Zahtevamo tudi ujemanje po oznakah točk (npr. atomi v molekuli). Oznake točk so določene z razredi (barvami) v razbitju, zato moramo pred izvedbo iskanja vzorcev tega tipa, izbrati še prvo in drugo razbitje. Prvo razbitje določa oznake točk prvega omrežja (vzorca), drugo pa oznake točk drugega (originalnega) omrežja.
  * **Check values of lines** – Zahtevamo ujemanje po vrednostih na povezavah (npr. pri parnih rodovnikih pomeni vrednost 1 moške, vrednost 2 pa ženske).
  * **Check relation number** – Zahtevamo ujemanje po številki relacije.
  * **Check only cluster** – V drugem omrežju poiščemo samo tiste pojavitve vzorcev, kjer je prva točka v vzorcu ena od izbranih točk (en od točk v izbrani skupini).
  * **Extract subnetwork** – Točke in povezave, ki sestavljajo posamezne vzorce, izločimo kot samostojno omrežje.
    1. **Retain all vertices after extracting** – v omrežju, ki ga dobimo kot dodaten rezultat, so prisotne vse točke iz originalnega omrežja, odstranjene so le povezave, ki ne pripadajo nobenemu vzorcu.
  * **Same vertices determine one fragment at most** – kako obravnavamo najdene vzorce z isto množico točk.
    1. **Create Hierarchy with fragments** – rezultat iskanja vzorcev je tudi hierarhija s točkami v vzorcih (na voljo le, če **Same vertices determine one fragment at most**, ni izbrano.
  * **Repeating vertices in fragment allowed** – ali se lahko ista točka v vzorcu pojavi večkrat. V tem primeru ima lahko najdeni vzorec manj točk kot originalni vzorec.

– **Match Vertex Labels** - Za dani dve omrežji določimo dve razbitji: položaje oznak točk prvega omrežja v drugem omrežju in položaje točk drugega

omrežja v prvem omrežju (0 pomeni, da oznaka v drugem omrežju ne obstaja).
Operacijo uporabimo za iskanje skupnega dela dveh omrežij, ki imata nekaj (a ne vseh) oznak točk enakih.
Dobljeno razbitje lahko uporabimo za usklajevanje razbitij in vektorjev z izločenimi podomrežji.

  - **Shrink coordinates (First to Second)** – Razmnoži koordinate točk v stisnjenem (prvem) omrežju na koordinate točk v originalnem (drugem) omrežju. Stiskanje je lahko določeno z razbitjem ali hierarhijo, zato mora biti pred izvedbo te operacije izbrano tudi ustrezno razbitje ali hierarhija.

- ****XXL** VertexID** Za izvedbo operacij iz te skupine potrebujemo samo identifikatorje točk (VertexID).

  - **Default VertexIDs** – Zgeneriramo identifikatorje z danim začetkom in izbrane dimenzije.
  - **Copy VertexIDs to Partition** – Identifikatorje prenesemo v razbitje.
  - **Expand VertexIDs to Partition** – Rezultat je novo razbitje izbane dimenzije, kjer se točke, ki jih določajo identifikatorji, nahajajo v razredu 1, ostale pa v razredu 0.
  - **Info** – Splošni podatki o identifikatorjih: število vrednosti, največja in najmanjša vrednost.

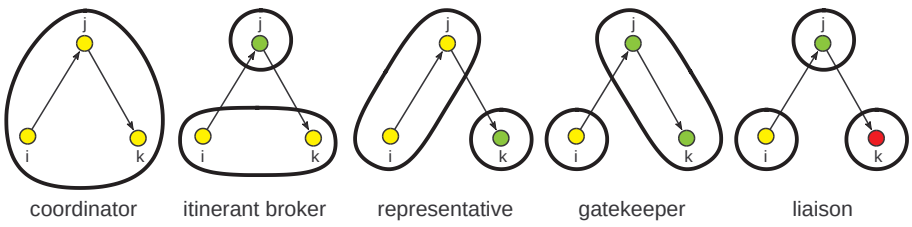
- **Operations** – Za izvedbo operacij iz te skupine potrebujemo poleg omrežja vsaj še eno dodatno strukturo.

  - ****XXL** Network + VertexID** – Operacije nad omrežjem in identifikatorji točk.
    - **Shrink Network** – Stiskanje omrežja glede na podane identifikatorje.
  - **Network + Partition** – Operacije nad omrežjem in razbitjem (in skupino v primeru Count Neighbour Clusters; in permutacijo v primeru Coloring).
    - **Extract** – Izločanje podomrežij glede na razbitje:
      - **SubNetwork Induced by Union of Selected Clusters** – Iz omrežja izločimo točke (in povezave med njimi), ki pripadajo izbranim skupinam v razbitju. Rezultat je samo eno podomrežje.
      - **SubNetworks Induced by Each Selected Cluster Separately** – Za vsako izbrano skupino iz razbitja izločimo iz omrežja inducirano podomrežje (točke, ki so v dani skupini in povezave med

njimi). Za rezultat dobimo toliko omrežij, kolikor skupin izberemo.

- **2-Mode Network** – Iz običajnega omrežja izločimo dvovrstno omrežje: prva in druga množica sta določeni z izbranimi skupinami iz razbitja.
- **to GEDCOM** – Izločanje podrodovnika glede na izbrano razbitje (npr. izločanje izbrane šibko povezane komponente) v novo datoteko GEDCOM. Pred tem moramo rodovnik prebrati v navadni obliki.

* **Shrink Network** – Stiskanje omrežja glede na izbrano razbitje. Točke v eni od skupin (privzeta je skupina 0) ostanejo nestisnjene, ostale skupine se predstavijo s skupno točko. Rezultat je novo omrežje, kjer so določene točke iz danega omrežja predstavljene s skupno točko. Pred izvedbo te operacije je potrebno izbrati ustrezni bločni model v izbiri Options. Privzeta vrednost je, da se povezava med skupinama pojavi, če obstaja vsaj ena povezava med točkami iz teh dveh skupin.
* **Brokerage Roles** – Preštejemo kolikokrat vsaka točka v omrežju (j) nastopa v vlogi določene vrste posrednika glede na dano razbitje, to je kot notranji posrednik (coordinator), zunanji posrednik (itinerant broker), predstavnik (representative), vratar (gatekeeper) in zveza (liaison).



* **Internal/External Cluster** – Za dano omrežje in razbitje izračunamo dva vektorja (*Internal* in *External*). Dimenzija vektorjev je enaka največji vrednosti iz razbitja. Za vsako skupino i iz razbitja dobljena vektorja podajata:
  - **Min** - $Internal[i]$ najmanjšo vrednost povezave v omrežju kjer sta obe krajišči v skupini i. $External[i]$ najmanjšo vrednost povezave v omrežju kjer je samo eno krajišče v skupini i.
  - **Max** - $Internal[i]$ največjo vrednost povezave v omrežju kjer sta obe krajišči v skupini i. $External[i]$ največjo vrednost povezave

v omrežju kjer je samo eno krajišče v skupini i.

* **Expand Partition** – Razširi podano delno razbitje (nekatere točke so še v razredu 0) na preostale točke v omrežju.
  - **Greedy Partition** – Točke, ki jih lahko dosežemo v k korakih iz izbranih točk, postavimo v isto skupino kot so izbrane točke. Dosegljivost lahko obravnavamo glede na usmerjeno ali neusmerjeno omrežje. V primeru, da bi neka točka hkrati pripadala večim skupinam, se ustrezne skupine zlijejo v eno.
  - **Influence Partition** – Vsako točko, ki ima še nedoločeno vrednost v razbitju (0), postavimo v skupino, v kateri je najbližja točka. Če je več točk na enaki oddaljenosti, postavimo točko v skupino, ki je največja od skupin točk, ki so na enaki oddaljenosti.
  - **Make Multiple Relations Network** – Pretvori omrežje v večrelacijsko omrežje s pomočjo razbitja: če obe krajišči povezave pripadata istemu razredu v razbitju, bo povezava pripadala relaciji s to številko, sicer bo pripadala relaciji s številko 0, razredu začetne ali razredu končne točke.
* **Identify** – Identificiraj (preuredi in/ali stisni) izbrane točke glede na dano razbitje. Če je neka točka edina točka v danem razredu, se samo na novo oštevilči, če pa je v tem razredu več točk, se ustrezne točke tudi stisnejo v skupno točko. Rezultat je novo omrežje, ki je preurejeno in ima morda tudi manj točk kot začetno omrežje.
* **Refine Partition** – Iz izbranega razbitja zgradimo podrobnejše razbitje, glede na dosegljivost v danem omrežju:
  - **Strong** – za usmerjena omrežja (točke v skupini morajo biti krepko povezane).
  - **Weak** – za neusmerjena omrežja (točke v skupini morajo biti šibko povezane).
* **Leader Partition** – Poiščemo skupine točk omrežja znotraj nivojev po metodi voditeljev.
* **Petri** – Petrijevo mrežo izvedemo glede na začetno označitev določeno z razbitjem. Glede na istočasen možen vžig večih prehodov Petrijeve mreže, lahko Petrijevo mrežo izvajamo na dva načina:
  - **Random** – Prehod, ki vžge, izberemo slučajno.
  - **Complete** – Zgradimo drevo (hierarhijo) vseh možnih izvajanj Petrijeve mreže.
* **Transform** – Pretvorbe omrežja glede na razbitje.

- **Remove Lines** – Iz omrežja odstranimo določene povezave glede na razbitje.
  1. **Inside Clusters** – Iz omrežja odstranimo vse povezave, ki imajo obe krajišči v izbranih skupinah.
  2. **Between Clusters** – Iz omrežja odstranimo vse povezave, ki imajo krajišči v različnih skupinah.
  3. **Between Two Clusters** – Iz omrežja odstranimo vse usmerjene ali neusmerjene povezave, ki imajo krajišči v izbranih dveh skupinah.
  4. **Inside Clusters with value** – Iz omrežja odstranimo vse povezave, ki imajo obe krajišči v isti skupini in pri katerih je vrednost na povezavi večja/manjša od tiste določene v vektorju. Razsežnost vektorja mora biti enaka največji številki skupine v razbitju.
- **Add Time Intervals determined by Partitions** – omrežje spremenimo v dvovrstno omrežje z uporabo dveh razbitij: prvo razbitje določa začetno drugo pa končno časovno točko.
- **Direction** – Vsako neusmerjeno povezavo v omrežju spremenimo v usmerjeno, ki kaže
  1. **Lower –> Higher** – od točke z nižjo proti točki z višjo vrednostjo v razbitju.
  2. **Higher –> Lower** – od točke z višjo proti točki z nižjo vrednostjo v razbitju.

  Usmerjene povezave v obratni smeri se odstranijo. Povezave znotraj razredov lahko odstranimo ali pustimo.

* **+ Cluster** – Operacije, ki potrebujejo za vhod omrežje, razbitje in skupino.
  - **Count Neighbour Clusters** – Za izbrano omrežje in razbitje (ki določa skupine) izračunamo nov vektor, kjer za vsako točko omrežja preštejemo, koliko je točk iz izbrane skupine v njeni okolici. Za vsako skupino (ki je navedena v objektu Cluster) dobimo za rezultat en vektor. Vrednosti v skupini torej v tej operaciji predstavljajo številke skupin in ne številke točk (kot je to običajno).
* **+ Permutation** – Operacije, ki potrebujejo za vhod omrežje, razbitje in permutacijo.
  1. **Coloring** – Barvanja točk omrežja.
     - **Create New** – Zaporedno barvanje točk omrežja. Točke se barvajo v vrstnem redu določenim s permutacijo. Rezultat je v

veliki meri odvisen od permutacije, zato je izbira permutacije zelo pomembna. Dobro permutacijo dobimo npr. z **Network/Permutation/Core+Degree**.

- **Complete Old** – Dokončamo delno barvanje točk omrežja v vrstnem redu določenim s permutacijo. Nekaj točk npr. pobarvamo ročno, preostanek dela pa prepustimo programu.

* **Info** – Informacije o razbitju danega omrežja.
  - **Modularity** – Modularnost omrežja glede na dano razbitje.
  - **VOS Quality** – 'VOS Quality' omrežja glede na dano razbitje.
  - **E-I Index** – E-I Index omrežja glede na dano razbitje (z upoštevanjem vrednosti na povezavah ali brez).

– **Network + Vector** – Operacije nad omrežjem in vektorjem (in skupino v primeru Diffusion Partition, ter razbitje in permutacijo v primeru Harmonic Function).

* **Network * Vector** – Običajno množenje omrežja N (matrike) z vektorjem x. Rezultat je vektor $y = Nx$
* **Vector # Network** – Omrežje N 'pomnožimo' z vektorjem x, tako da i-ti stolpec (vrstico) pripadajoče matrike pomnožimo z i-to komponento vektorja x.
* **Neighbours**
  1. **Sum** – Za vsako točko izračunamo vsoto vrednosti v vektorju vseh sosedov, pri čemer za sosede upoštevamo
     - **Input** – vhodne sosede;
     - **Output** – izhodne sosede;
     - **All** – vse sosede;
  2. **Min** – Za vsako točko izračunamo najmanjšo vrednost v vektorju vseh sosedov, pri čemer za sosede upoštevamo
     - **Input** – vhodne sosede;
     - **Output** – izhodne sosede;
     - **All** – vse sosede.
  3. **Max** – Za vsako točko izračunamo največjo vrednost v vektorju vseh sosedov, pri čemer za sosede upoštevamo
     - **Input** – vhodne sosede;
     - **Output** – izhodne sosede;
     - **All** – vse sosede.

* **Islands** – Razbitje točk na kohezivne skupine glede na vrednosti točk določene v vektorju.
  - **Vertex Weights** – V vrednostnem omrežju je otok skupina točk, kjer so vrednosti točk znotraj otoka večje kot v okolici. Uteži imenujemo tudi višine točk.
  - **Vertex Weights [Simple]** – Enostavni otok je otok z enim samim vrhom.
* **Transform** – Pretvorbe omrežja glede na vektor.
  - **Put Loops** – Komponente trenutnega vektorja naj postanejo zanke (usmerjene ali neusmerjene) v omrežju.
  - **Put Coordinate** – Komponente trenutnega vektorja naj postanejo koordinate x, y ali z točk v omrežju ali pa naj postanejo polarni polmeri ali polarni koti točk. Omrežje in vektor morata imeti enako razsežnost.
  - **Vector(s) –> Line Values** – Vrednosti na povezavah zamenjamo z rezultatom izbrane operacije (vsota, razlika, produkt, deljenje, minimum, maximum) v vektorju (ali dveh vektorjih) v krajiščih povezav. Vrednost povezave je lahko tudi vrednost v vektorju za začetno ali končno točko povezave.
* **+ Cluster** – Operacije, ki potrebujejo za vhod omrežje, vektor in skupino.
  - **Diffusion Partition** – Izračunamo čase (razbitje), ko se okužijo (sprejmejo inovacijo) posamezne točke v grafu. Deleži okuženih sosedov, ki povzročijo okužbo posameznih oseb, so dani v vektorju. Predpostavimo, da točke v trenutni skupini sprejmejo inovacijo v času 1.
* **+ Partition + Permutation** – Operacije, ki potrebujejo za vhod omrežje, vektor, razbitje in permutacijo.
  1. **Harmonic Function** – Naj bo (G, a) povezano vrednostno omrežje z vrednostno funkcijo $a(x, y)$, pri čemer sta x in y točki omrežja. S naj bo podmnožica točk $(S \subseteq V(G))$. Funkcija $f : V(G) \to \mathbb{R}$ je *harmonična* na (G, a), z omejitvijo S, natanko takrat, ko

$$f(x) = \frac{1}{A(x)} \sum_y a(x, y) * f(y)$$

za vsak $x \in V(G) \setminus S$, in

$$A(x) = \sum_y a(x, y)$$

Podrobnosti so opisane v: B. Bollobas: *Random Walks on Graphs*, stran 328. Harmonične funkcije so v programu Pajek izvedene na naslednji način:

· funkcija f je določena z vektorjem;
· vrednostna funkcija $a(x, y)$ je podana z omrežjem z vrednostmi na povezavah;
· podmnožica S je določena z razbitjem – točke v razredu 1 se nahajajo v podmnožici S (nepremične točke), ostale točke so v $V(G) \setminus S$;
· dodatno določa izbrana permutacija še vrstni red v katerem obravnavamo točke pri izračunih.

Harmonično funkcijo lahko izračunamo enkrat ali ponavljamo postopek izračunavanja toliko časa, da postanejo razlike med sosednjimi izračuni zanemarljive. Komponente vektorja, ki predstavljajo funkcijo f lahko popravljamo sproti ali pa samo na koncu vsake ponovitve (ko so poznane vse komponente). Postopek lahko poženemo glede na:

· **Input** – vhodne sosede;
· **Output** – izhodne sosede;
· **All** – vse sosede.

∗ **Info** – indeksi izračunani iz omrežja s pomočjo enega ali dveh vektorjev.

1. **Assortativity** – indeks 'asortativnosti' omrežja glede na vrednosti v enem ali dveh vektorjih.

– **Network + Scalar** – Operacije nad omrežjem in skalarjem (vektorjem dolžine 1).

∗ **Line Values** – Vrednosti na povezavah 'normaliziramo' glede na skalar (operacije Add, Subtract, Multiply, Divide, Min, Max).

– **Network + Permutation** – Operacije nad omrežjem in permutacijo.

∗ **Reorder Network** – Točke v omrežju preuredimo glede na izbrano permutacijo.
∗ **Numbering*** – Izboljšaj dano permutacijo glede na omrežje in dodaten kriterij:

· **Travelling Salesman** – Določi obhod v omrežju po metodi potujočega trgovca. Vhodno omrežje je lahko matrika različnosti, ali pa tudi običajno omrežje, kjer pa moramo zapolniti diagonalo in vrednosti 0 zamenjati z ustreznimi velikimi števili.

· **Seriation** – *Murtaghovo oštevilčenje* – za običajna in dvovrstna omrežja. (*Compstat Lectures*, 11-16, 1985).

· **Clumping** – Drugo *Murtaghovo oštevilčenje* – za običajna in dvovrstna omrežja. (*Compstat Lectures*, 11-16, 1985).

· **R-Enumeration** – *Richardsovo oštevilčenje* – oštevilčenje točk omrežja, kjer imajo sosednje točke, čimbolj zaporedne številke.

– **Network + Cluster** – Operacije nad omrežjem in skupino.

∗ **Extract SubNetwork** – Iz omrežja izločimo točke (in povezave med njimi), ki se nahajajo v izbrani skupini.

∗ **Dissimilarity*** – Izračun različnosti med točkami omrežja.

· **Network based**
Izračun izbrane matrike različnosti (d_1, d_2, d_3 ali d_4) za točke v skupini glede na skupne sosede. Poleg omenjenih lahko izračunamo še popravljeno evklidsko (d5) ali manhattansko (d6) različnost. Označimo *soseščino* točke $v \in V$ z $N(v)$:

$$N(v) = \{u \in V : (v : u) \in L\}$$

kjer je V množica vseh točk, L pa množica povezav. $maxdegree1$ je največja stopnja točke v omrežju, $maxdegree2$ pa druga največja stopnja. Pri izbiri soseščine se lahko omejimo samo na vhodne ali izhodne sosede, lahko pa upoštevamo vse sosede. V določenih primerih je smiselno med sosede dane točke šteti tudi točko samo. V tem primeru označimo soseščino točke v z $N^+(v)$:

$$N^+(v) = N(v) \cup \{v\}$$

Matriko različnosti lahko v primeru manjšega števila točk izpišemo v izhodnem oknu.
Omenjene matrike različnosti so definirane takole:

$$d_1(u,v) = \begin{cases} 1 & N(u) = N(v) = \emptyset \\ \dfrac{|N(u) \oplus N(v)|}{maxdegree1 + maxdegree2} & \text{sicer} \end{cases}$$

$$d_2(u,v) = \begin{cases} 1 & N(u) = N(v) = \emptyset \\ \dfrac{|N(u) \oplus N(v)|}{|N(u) \cup N(v)|} & \text{sicer} \end{cases}$$

$$d_3(u,v) = \begin{cases} 1 & N(u) = N(v) = \emptyset \\ \dfrac{|N(u) \oplus N(v)|}{|N(u)| + |N(v)|} & \text{sicer} \end{cases}$$

$$d_4(u,v) = \begin{cases} 1 & N(u) = N(v) = \emptyset \\ \dfrac{\max(|N(u) \setminus N(v)|, |N(v) \setminus N(u)|)}{\max(|N(u)|, |N(v)|)} & \text{sicer} \end{cases}$$

$$d_5(u,v) = \sqrt{\sum_{s=1, s\neq u,v}^{n} ((r_{us} - r_{vs})^2 + (r_{su} - r_{sv})^2) + p((r_{uu} - r_{vv})^2 + (r_{uv} - r_{vu})^2)}$$

$$d_6(u,v) = \sqrt{\sum_{s=1, s\neq u,v}^{n} (|r_{us} - r_{vs}| + |r_{su} - r_{sv}|) + p(|r_{uu} - r_{vv}| + |r_{uv} - r_{vu}|)}$$

$$p \in [0, 1, 2]$$

Dobljeno matriko različnosti lahko uporabimo za hierarhično razvrščanje točk omrežja v skupine.
Če je označeno tudi **Among all linked Vertices only**, se različnosti izračunajo kot vrednosti na povezavah za dano omrežje.

- **Vector based** – Različnosti: **Euclidean, Manhattan, Canberra, ali (1-Cosine)/2** se izračunajo glede na vrednosti v vektorjih določenimi s skupino in se dodajo kot vrednosti na povezave za dano omrežje.

* k**-Neighbours from Vertices in Cluster** – Določitev oddaljenosti vseh točk od točk določenih s skupino. Rezultat je torej toliko vektorjev kolikor je točk v skupini.
  - **Input** – oddaljenosti v smeri nasprotni smeri povezav.
  - **Output** – oddaljenosti v smeri povezav.
  - **All** – oddaljenosti ne glede na smer povezav (vse povezave obravnavamo kot neusmerjene).

  Točkam, ki so iz dane točke nedosegljive, priredimo oddaljenost 999999998.
* **Distribution of Distances from Vertices in Cluster*** – Vrne porazdelitev dolžin najkrajših poti med vsemi (dosegljivimi) pari točk. Pregleda le poti, ki se začnejo v točkah, ki so izbrane s skupino.
* **Transform** – Pretvorbe omrežja glede na skupino.
  - **Add Arcs from Vertex to Cluster** – dodamo povezave od izbrane točke do vseh točk iz skupine.
  - **Add Arcs from Cluster to Vertex** – dodamo povezave od vseh točk iz skupine do izbrane točke.

– **Network + Hierarchy** – Operacije nad omrežjem in hierarhijo.

    - **Shrink Network** – Stiskanje omrežja glede na izbrano hierarhijo. Točke omrežja, ki v hierarhiji pripadajo vozlišču, označenim s Close, se stisnjejo v novo točko, ostale točke ostanejo nestisnjene.
    - **Expand Reduction** – Iz okleščenega omrežja (dobljenega s hierarhično redukcijo) in ustrezne hierarhije vzpostavimo začetno omrežje – okleščenemu omrežju dodamo drevesne točke in povezave. Rezultat je neusmerjeno omrežje.
- ****XXL** VertexID + Partition** – Operacije nad identifikatorji in razbitjem.
    - **Extract SubVertexIDs** – Izberemo samo identifikatorje, ki izpolnjujejo določen pogoj (so npr. v izbranem intervalu vrednosti), ki ga določa razbitje.
- **Partition + Permutation** – Operacije nad razbitjem in permutacijo.
    - **Reorder Partition** – razbitje preuredimo glede na permutacijo. Enak rezultat dobimo s funkcijsko kompozicijo **Permutation*Partition**: Naj bo f permutacija in g razbitje. Rezultat je novo razbitje r, ki ga dobimo takole $r[v] = (f * g)[v] = g[f[v]]$.
    - **Functional Composition Partition*Permutation** – Naj bo f razbitje in g permutacija. Rezultat je novo razbitje r, ki ga dobimo takole $r[v] = (f * g)[v] = g[f[v]]$.
- **Partition + Cluster** – Operacije nad razbitjem in skupino.
    - **Binarize Partition** – Razbitje pretvorimo v dvojiško razbitje glede na izbrano skupino. Točke, ki imajo v razbitju vrednosti, ki se nahajajo tudi v skupini, gredo v razred 1, ostale v razred 0. Vrednosti v skupini torej v tej operaciji predstavljajo številke skupin in ne številke točk (kot je to običajno).
- **Vector + Partition** – Operacije nad vektorjem in razbitjem.
    - **Extract Subvector** – Iz izbranega vektorja izločimo tiste komponente, ki se nahajajo v določenih skupinah v razbitju.
    - **Shrink Vector** – Vrednosti v vektorju stisnemo glede na izbrano razbitje in s tem prilagodimo vektor stisnjenemu omrežju. Pri stiskanju večih vrednosti v eno je lahko nova vrednost vsota vrednosti, povprečje, najmanjša ali največja vrednost ali pa mediana.
    - **Functional Composition Partition*Vector** – Naj bo f razbitje in g vektor. Rezultat je nov vektor r, ki ga dobimo takole $r[v] = (f * g)[v] = g[f[v]]$.

  - **Vector + Permutation** – Operacije nad vektorjem in permutacijo.
    - **Reorder Vector** – vektor preuredimo glede na permutacijo. Enak rezultat dobimo s funkcijsko kompozicijo **Permutation*Vector**: Naj bo f permutacija in g vektor. Rezultat je nov vektor r, ki ga dobimo takole $r[v] = (f * g)[v] = g[f[v]]$.

- **Partition** – Za izvedbo operacij iz te skupine potrebujemo samo razbitje.
  - **Create Constant Partition** – Zgradimo razbitje izbrane razsežnosti, kjer so vse vrednosti enake izbrani konstanti.
  - **Create Identity Partition** – Zgradimo razbitje z vrednostmi $C[i] := i$.
  - **Create Random Partition** – Zgradimo slučajno eno ali dvovrstno razbitje.
  - **Binarize Partition** – Dano razbitje pretvorimo v dvojiško, tako da izberemo skupine, ki gredo v novo skupino 1, ostale skupine gredo v novo skupino 0.
  - **Fuse Clusters** – Zlij izbrane razrede iz razbitja v nov razred.
  - **Canonical Partition**
    - **with Vertex 1 in Cluster 1** – Dano razbitje pretvorimo v kanonsko obliko (oblika kjer je točka 1 (in ostale točke, ki so v isti skupini kot točka 1) vedno v skupini 1, če točka 2 ni v isti skupini kot točka 1, gre točka 2 v skupino 2...).
    - **with Decreasing Frequencies** – Dano razbitje pretvorimo v kanonsko obliko (v novem razredu 1 bo stari razred z največjo frekvenco...).
  - **Make Network** – Iz razbitja naredimo omrežje.
    - **Random Network** – Zgradimo naključno omrežje, kjer so stopnje (undirected), vhodne stopnje (input) ali izhodne stopnje (output) točk določene z razbitjem.
    - **2-Mode Network** – Zgradimo dvovrstno omrežje: prva množica so točke ($v_1 \dots v_n$), druga pa razredi iz razbitja ($c_0 \dots c_m$). Če se točka i nahaja v razredu j, potem v ustreznem omrežju obstaja povezava od v_i do c_j. Če izberemo *Existing Clusters only*, se edino obstoječi razredi (razredi, ki vsebujejo vsaj eno točko) proglasijo za točke druge množice.
  - ****XXL** Copy to VertexID** – Razrede iz razbitja proglasi za identifikatorje.

- **Make Permutation** – Iz razbitja naredimo permutacijo, tako da pridejo na vrsto najprej vse točke iz prve skupine, nato vse točke iz druge skupine...
- **Make Cluster** – Iz razbitja izberemo določene točke in jih shranimo v skupino.
  - **Vertices from Selected Clusters** – Izberemo točke, ki se v razbitju nahajajo v določenih skupinah.
  - **Random Representatives of each Cluster** – Slučajno izberemo po eno točko (predstavnika) iz vsake skupine v razbitju. Predstavnike shranimo v Cluster. Kasneje jih lahko uporabimo kot 'pivote' pri risanju **Pivot MDS**.
- **Make Hierarchy** – Razbitje pretvorimo v hierarhijo. Razbitje lahko pretvorimo v enonivojsko hierarhijo ali v gnezdeno hierarhijo – skupine z višjo številko so gnezdene v skupinah z nižjo ali obratno.
- **Copy to Vector** – Razbitje prenesemo v vektor ($V[i] := C[i]$).
- **Count, Min-Max Vector** – Za rezultat dobimo razbitje s številom točk v posameznih razredih podanega razbitja, in v primeru, da je vhod tudi vektor enake razsežnosti tudi največjo in najmanjšo vrednost v vektorju po posameznih razredih.
- **Info** – Splošni podatki o razbitju: nekaj največjih / najmanjših vrednosti v razbitju, frekvenčna porazdelitev po skupinah, povprečna številka skupine, mediana in standardni odklon.

- **Partitions** – Za izvedbo operacij iz te skupine potrebujemo dve razbitji.
  - **Extract SubPartition (Second from First)** – Iz prvega razbitja izločimo točke, ki ustrezajo določenemu pogoju (ležijo na nekem intervalu v drugem razbitju). Če podano razbitje predstavlja neko lastnost točk omrežja in iz omrežja izločimo podomrežje glede na dobljeno razbitje (npr. izločimo šibko povezano komponento), potem moramo uporabiti tudi operacijo izločanja podrazbitja glede na razbitje, ki določa željeno komponento, da dobimo razbitje, ki ustreza novemu omrežju (opisuje točke novega omrežja) .
  - **Add (First+Second)** – Seštejemo izbrani razbitji (po komponentah).
  - **Min (First, Second)** – Za vsako točko izbere manjšo vrednost od obeh vrednosti v razbitju.
  - **Max (First, Second)** – Za vsako točko izbere večjo vrednost od obeh vrednosti v razbitju.

- **Fuse Partitions** – Drugo razbitje pripišemo na konec prvega (uporabno npr. za dvovrstna omrežja).
- **Expand Partition** – Vračanje podrazbitja na originalno velikost.
  - **First According to Second (Shrink)** – Razmnožimo prvo razbitje (C_1) glede na stiskanje določeno z drugim razbitjem (C_2). To operacijo uporabimo, če želimo razbitje, ki ga izračunamo nad točkami stisnjenega omrežja prenesti na celotno omrežje. Rezultat (razbitje C) je torej določen takole: $C[i] = C_1[C_2[i]]$
  - **Insert First into Second according to Third (Extract)** – Vstavimo trenutno izbrano podrazbitje v prvo (originalno) razbitje (C_1) glede na opravljen izrez določen z razbitjem C_2. To operacijo uporabimo, če iz razbitja izrežemo izbrano podrazbitje. To podrazbitje v toku dela lahko spremenjamo, na koncu pa ga želimo vstaviti nazaj na prava mesta v razbitju (C_1).
- **Intersection of Partitions** – Presek izbranih razbitij.
- **Cover with** – Naj bosta dana razbitje p in dvojiško razbitje b ter številka skupine c. Rezultat operacije je novo razbitje q določeno s predpisom $q(v) = p(v)$, če $b(v) = 0$ in $q(v) = c$, sicer.
- **Merge Partitions** – Naj bosta dani razbitji p in q ter dvojiško razbitje b. Rezultat operacije je novo razbitje s določeno s predpisom $s(v) = p(v)$, če $b(v) = 0$ in $s(v) = q(v)$, sicer.
- **Make Random Network** – Zgradimo naključno omrežje, kjer so vhodne stopnje določene s prvim, izhodne pa z drugim razbitjem.
- **Functional Composition First*Second** – Naj bosta f in g dve razbitji. Rezultat je novo razbitje r, ki ga dobimo takole $r[v] = (f * g)[v] = g[f[v]]$.
- **Info** – Povezanost dveh razbitij:
  - **Cramer's V, Rajski, Adjusted Rand Index** – Cramerjev koeficient, koeficienti Rajskega in Adjusted Rand Index med izbranima razbitjema (predpostavimo, da med razredi v razbitju ni urejenosti).
  - **Spearman Rank** – Spearmanov koeficient korelacije rangov med izbranima razbitjema (predpostavimo, da obstaja urejenost med razredi v razbitju).

- **Vector** – Za izvedbo operacij iz te skupine potrebujemo samo vektor (in morda razbitje).

- **Create Constant Vector** – Zgradimo vektor izbrane razsežnosti, kjer so vse vrednosti enake konstanti, ki jo navedemo.
- **Create Scalar** – Zgenerira skalar (vektor dolžine 1) iz vektorja.
- **Make Partition** – Vektor pretvorimo v razbitje:
  - **by Intervals** – Skupine so določene z mejami, ki jih vnesemo:
    - **First Threshold and Step** – Izberemo prvo mejo in korak v katerem se le ta povečuje.
    - **Selected Thresholds** – Naštejemo vse mejne vrednosti ali podamo število razredov (#).
  - **Copy to Partition by Truncating (Abs)** – Skupine so absolutne in zaokrožene vrednosti v vektorju.
- **Make Permutation** – Vektor pretvorimo v permutacijo (oštevilčenje po naraščajoči vrednosti komponent vektorja).
- **Make Cluster** – Vektor pretvorimo v skupino – izberemo samo tiste točke, ki imajo v vektorju vrednost manjšo/večjo od izbrane vrednosti.
- **Make 2-Mode Network** – Vektor pretvorimo v dvovrstno omrežje (vrstico ali stolpec).
- **Transform** – Pretvorbe izbranega vektorja:
  - **Multiply by** – Množenje vektorja s konstanto.
  - **Add Constant** – Vsem komponentam vektorja prištejemo isto konstanto.
  - **Absolute** – Absolutne vrednosti komponent.
  - **Absolute + Sqrt** – Kvadratni koreni absolutnih vrednosti komponent.
  - **Truncate** – Navzdol zaokrožene vrednosti komponent.
  - **Exp** – Eksponent komponent (e^x).
  - **Ln** – Naravni logaritem komponent ($\ln x$).
  - **Power** – Izbrana potenca komponent (x^r).
  - **Normalize** – Normalizacije komponent:
    - **Sum** – Normalizacija komponent, tako da je vsota komponent 1.
    - **Max** – Normalizacija komponent, tako da je največji element 1.
    - **Standardize** – Normalizacija komponent, tako da je povprečje 0 in standardni odklon 1.
  - **Invert** – Inverzne vrednosti vektorja.
  - **Cumulatives** – Izračuna nov vektor u kjer so shranjene kumulative (delne vsote) v vektorju v: $u_1 = v_1, u_i = u_{i-1} + v_i, i > 1$.

  - **Missing Values** – Izberemo kako naj se vrednosti večje od 999.999.997 obravnavajo v operacijah nad vektorji: kot običajne vrednosti ali kot manjkajoče vrednosti?
  - **Info** – Izpis splošnih podatkov o vektorju: vrednosti urejene po velikosti, povprečna vrednost, standardni odklon ter frekvenčna porazdelitev vrednosti v izbrano število razredov (podamo lahko meje razredov ali število razredov – #).

- **Vectors** – Za izvedbo operacij iz te skupine potrebujemo dva vektorja. Vektorja morata biti enake dimenzije, lahko pa je eden tudi skalar.
  - **Add (First+Second)** – Vsota vektorjev.
  - **Subtract (First-Second)** – Razlika vektorjev.
  - **Multiply (First*Second)** – Produkt vektorjev (po komponentah).
  - **Divide (First/Second)** – Kvocient vektorjev (po komponentah).
  - **Min (First, Second)** – Izbira manjše komponente v vektorjih.
  - **Max (First, Second)** – Izbira večje komponente v vektorjih.
  - **Fuse Vectors** – Zlitje dveh vektorjev v večji vektor (vektorja se zapišeta eden za drugim).
  - **Linear Regression** – linearna regresijska ocena drugega vektorja z uporabo linearne regresije. Rezultati so: regresijska premica, regresijske ocene drugega vektorja in pripadajoče napake ocen.
  - **Transform** – Preoblikovanja dveh vektorjev v nova dva vektorja.
    * **Cartesian - Polar** – Pretvorba kartezijskih koordinat v polarne. Vhod sta dva vektorja: prvi vsebuje x drugi pa y koordinate točk. Rezultat sta spet dva vektorja: prvi vsebuje polarne radije drugi pa kote v stopinjah.
    * **Polar - Cartesian** – Pretvorba polarnih koordinat v kartezijske. Vhod sta dva vektorja: prvi vsebuje polarne radije drugi pa kote v stopinjah. Rezultat sta spet dva vektorja: prvi vsebuje x drugi pa y koordinate.

    Če bomo izračunane vektorje uporabljali za koordinate točk v oknu Draw, izberemo tudi opcijo za (de)normalizacijo vektorjev.
  - **Missing Values** – Izberemo kako naj se vrednosti večje od 999.999.997 obravnavajo v operacijah nad vektorji: kot običajne vrednosti ali kot manjkajoče vrednosti?
  - **Info** – Pearsonov korelacijski koeficient med izbranima vektorjema.

- **Permutation** – Za izvedbo operacij iz te skupine potrebujemo samo permutacijo.
  - **Create Identity Permutation** – Zgradimo identično permutacijo izbrane razsežnosti.
  - **Create Random Permutation** – Zgradimo naključno eno ali dvo-vrstno permutacijo izbrane razsežnosti.
  - **Inverse Permutation** – Zgradimo inverzno permutacijo izbrani permutaciji.
  - **Mirror Permutation** – Zgradimo zrcalno permutacijo izbrani permutaciji (ureditev v obratnem vrstnem redu).
  - **Make Partition** – Iz izbrane permutacije zgradimo razbitje.
    - **Into Given Number of Clusters** – Zgradimo razbitje z danim številom razredov (glede na naraščajoče vrednosti v permutaciji).
    - **Orbits** – Zgradimo razbitje po orbitah permutacije.
  - **Copy to Vector** – Izbrano permutacijo prenesemo v vektor.
  - **Info** – Preverimo pravilnost permutacije in vrnemo število orbit.
- **Permutations** – Operacije nad dvema permutacijama.
  - **Fuse Permuations** – Zlitje permutacij – drugo permutacijo dodamo zamakjneno na konec prve (za dvovrstna omrežja).
  - **Functional Composition First*Second** – Naj bosta f in g dve permutaciji. Rezultat je nova permutacija r, ki jo dobimo takole $r[v] = (f * g)[v] = g[f[v]]$. Če sta permutaciji različnih dimenzij, je rezultat razbitje.
- **Cluster** – Za izvedbo operacij iz te skupine potrebujemo samo podatkovno strukturo skupina.
  - **Create Empty Cluster** – Zgradimo prazno skupino.
  - **Create Complete Cluster** – Zgradimo skupino s točkami $1 \ldots n$.
  - **Create Random Cluster** – Zgradimo skupino z izbrano največjo vrednostjo in številom enot v skupini.
  - **Make Partition** – Skupino pretvorimo v dvojiško razbitje izbrane razsežnosti.
  - **Info** – Splošni podatki o skupini: število vrednosti, največja in najmanjša vrednost.
- **Hierarchy** – Za izvedbo operacij iz te skupine potrebujemo samo hierarhijo.

- **Extract Cluster** – Vse točke, ki se nahajajo v izbranem poddrevesu v hierarhiji, izločimo kot samostojno skupino.
- **Make Network** – Hierarhijo pretvorimo v omrežje – drevo. Pri tem se upoštevajo tudi zaprta vozlišča v hierarhiji (Close).
- **Make Partition** – Hierarhijo pretvorimo v razbitje. Skupine so določene z zaprtimi vozlišči hierarhije.
- **Make Permutation** – Hierarhijo pretvorimo v permutacijo.
- **Info** – Izpis splošnih podatkov o hierarhiji. Izbira je možna le, če so številke skupin cela števila. Dobimo število točk v vozliščih na prvem nivoju hierarhije.

• **Options** – Izbira splošnih nastavitev – program prilagodimo svojim potrebam.

- **Read - Write**
  - **Threshold** – Meja, ki jo morajo doseči vrednosti na povezavah, da se pri branju omrežja generira povezava.
  - **Large Network (Vertices)** – Določimo katera omrežja se smatrajo za zelo velika. Pri teh omrežjih pri nekaterih operacijah Pajek postavi vprašanje ali naj se izvedejo na mestu, brez generiranja novega omrežja – staro omrežje se povozi. To je tudi mejna vrednost od katere naprej se računajo kvantili v **Vector/Info** (za velike vektorje, kjer je veliko vrednosti enakih, je ta operacija lahko počasna).
  - **x / 0 =** – Določimo rezultat deljenja neničelne vrednosti z vrednostjo 0.
  - **0 / 0 =** – Določimo rezultat deljenja 0 z 0.
  - **Ignore Missing Values in menu Vector and Vectors** – Ko izvajamo operacije v menuju Vector in Vectors ali računamo opisne statistike za dani vektor, obravnavamo manjkajoče vrednosti (vrednosti večje od 999999997) kot običajna števila.
  - **Use Scientific format for small/large real numbers** – če so vrednosti premajhne ali prevelike, naj se izpišejo v znanstvenem načinu (npr. 2.4E+015 ali 2.4E-015).
  - **Save Files as Unicode UTF8** – Namesto na datoteko ASCII se Pajkovi objekti in datoteke z ločili (*delimited files*) shranjujejo na datoteko Unicode UTF8.
    - **With BOM** – Izbira shranjevanja z ali brez oznake BOM.

    Nekaj pravil pri uporabi formata UTF8 v Pajku:

- Pajek prepozna datoteke UTF8 s pomočjo oznake BOM. Zato v primeru, da omrežja vsebujejo oznake z neangleškimi znaki, priporočamo shranjevanje v formatu UTF8 z oznako BOM.
- Izvozi v SVG in X3D so vedno v formatu UTF8 brez BOM (ker temeljijo na standardu **xml**).
- Izpisi (reports) se vedno shranijo v formatu UTF8 z ali brez BOM (odvisno od izbire).
- Za datoteke z ločili (*delimited files*) je uporaba ali neuporaba oznake BOM določena v **Tools / Delimited File Properties**.

* **Auto Report** – Vsi rezultati naj se sproti pišejo na tekstovno datoteko rep1.rep.
* **Read-Save vertices labels** – Pri branju/shranjevanju omrežja naj se preberejo/shranijo tudi oznake točk in ostala navodila za risanje omrežja. Če oznak točk ne preberemo, jih lahko dodamo kasneje z Network / Transform / Add / Vertex Labels from File (priporočljivo za zelo velika omrežja z dolgimi oznakami točk)
* **Save coordinates of vertices** – Pri shranjevanju omrežja na izhodno datoteko shranimo tudi koordinate točk.
* **Save complete vertex description** – Pri shranjevanju omrežja na izhodno datoteko zapišemo na datoteko za vsako točko tudi popoln opis njenih lastnosti (oblika točke, časovni intervali...) ne glede na to, če ima točka popolnoma enak opis kot prejšnja (če tega ne izberemo, je izpis lahko precej krajši).
* **Check equality of vertex descriptions by reading** – Uporabnik lahko sam izbira med varčevanjem s pomnilnikom in hitrejšim branjem: Opcija naj bo izbrana, da se prihrani prostor v pomnilniku, če se v datoteki nahaja le nekaj različnih opisov točk (npr. oblike točk). Opcija naj ne bo izbrana, da se prihrani pri času branja, kadar je v datoteki veliko različnih opisov točk (npr. intervali prisotnosti točk v časovnih omrežjih).
* **Check equality of line descriptions by reading** – Uporabnik lahko sam izbira med varčevanjem s pomnilnikom in hitrejšim branjem: Opcija naj bo izbrana, da se prihrani prostor v pomnilniku, če se v datoteki nahaja le nekaj različnih opisov povezav (npr. vzorci izrisa povezav – Dots/Solid). Opcija naj ne bo izbrana, da se prihrani pri času branja, kadar je v datoteki veliko različnih opisov povezav (npr. intervali prisotnosti povezav v časovnih omrežjih).

  - **Max. vertices to draw** – Omejitev slikovnih prikazov na omrežja z največ določenim številom točk.
  - **Ore: Different relations for male and female links** – Pri branju datotek GEDCOM v navadni obliki naj se generirajo povezave s številko relacije 1 (tudi vrednostjo 1) za relacijo oče-otrok (boter-otrok) in povezave (s številko relacije 2) (tudi vrednostjo 2) za relacijo mati-otrok (botrica-otrok).
  - **Ore: Generate Godparent relation** – Pri branju datotek GEDCOM v navadni obliki naj se zgenerira tudi relacija *je boter od* (številka relacije 4) ali boter (številka relacije 4) in botra (številka relacije 5).
  - **GEDCOM – Pgraph** – Pri branju datotek GEDCOM naj se rodovnik prebere v parni obliki.
  - **Bipartite Pgraph** – Pri branju datotek GEDCOM naj se rodovnik prebere v dvodelni parni obliki (kvadratne točke predstavljajo poroke, trikotniki predstavljajo moške, krožci pa ženske).
  - **Pgraph+labels** – Pri branju datotek GEDCOM v parni obliki naj se povezavam dodajo tudi oznake, ki ustrezajo osebam.

– **Background Colors** - Izbira barv za različna ozadja in barve za panele.

– **Font (Typeface, Style, Size, Color)**

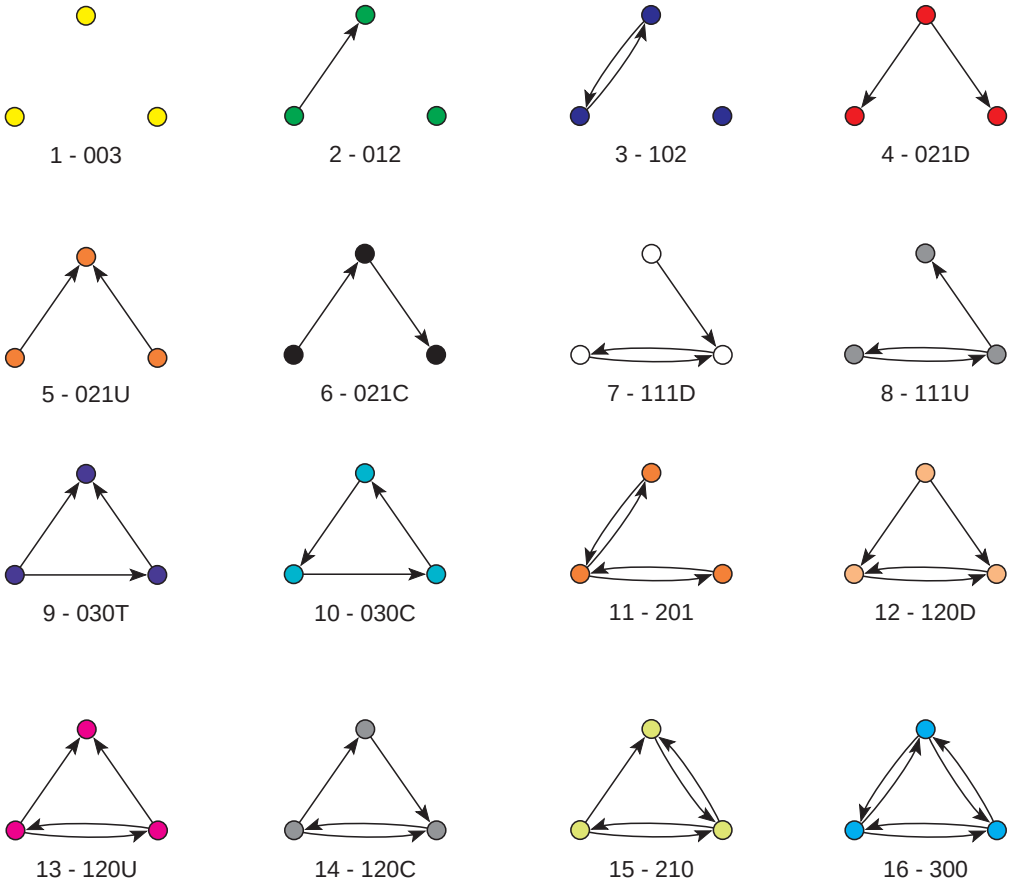
  - **Proportional Font** – Izbira fonta za Unicode, ki se uporablja v oknu Draw in pri izpisih, ki ne zahtevajo poravnavanja.
  - **Monospaced Font** – Izbira neproporcionalnega fonta za Unicode, ki se uporablja v oknu Report in drugih oknih, kjer se izpisi poravnavajo po stolpcih.
  - **Default Fonts** – Za porporcionalni font uporabi *Arial Unicode MS (Bold, Maroon)* za neproporcionalni pa *Courier New (Bold, Maroon)*.

– **Blockmodel - Shrink** – Izbira vrste bločnega modela za stiskanje:

  - 0 . . . Min Number of Links (najmanjše število povezav);
  - 1 . . . Null (prazna povezava);
  - 2 . . . Complete (polna povezava);
  - 3 . . . Row-Dominant (vrstično dominantna povezava);
  - 4 . . . Col-Dominant (stolpčno dominantna povezava);
  - 5 . . . Row-Regular (vrstično regularna povezava);
  - 6 . . . Col-Regular (stolpčno regularna povezava);
  - 7 . . . Regular (regularna povezava);

    - 8 . . . Row-Functional (vrstično funkcijska povezava);
    - 9 . . . Col-Functional (stolpčno funkcijska povezava);
    - 10 . . . Degree Density (povezava glede na gostoto stopenj).
  - **Hide Drawing Icons** – Odstranimo ikone za odpiranje okna Draw.
  - **Use Old Style Dialogs** – Izberemo, če imajo Windows 7 težave z odpiranjem ali shranjevanjem datotek.
  - **Refresh Objects** – Osvežimo sezname vseh objektov, ki so trenutno naloženi v Pajku.

- **Macro**
  - **Record** – Izberemo datoteko na katero se bodo v nadaljevanju shranjevali (snemali) vsi koraki, ki jih bomo naredili v programu Pajek. Shranjevanje zaključimo s ponovnim pritiskom na izbiro **Record**.
  - **Add message** – V datoteko, v katero se zapisujejo ukazi izvajanja, dodamo poljubno besedilo. To besedilo se ob izvajanju *makra* izpisuje v posebnem oknu ter nam služi za orientacijo, do kje je izvajanje že prišlo.
  - **Play** – Izvajanje izbranega *makra*.
  - **Repeat Session** – Med izvajanjem programa se vse operacije shranjujejo na datoteko logxy.log, kjer je xy zaporedna številka izvajanja programa. Z uporabo tega ukaza lahko izvedemo vse operacije, ki smo jih opravili v eni od predhodnih uporab programa in na ta način vzpostavimo stanje, do katerega smo prišli takrat. Če na direktoriju kjer se nahaja program Pajek.exe, obstaja tudi datoteka Pajek.log, se ta datoteka izvede vsakič ko poženemo program Pajek.
  - **Run Command** – najprej prikažemo vse izvedene ukaze, izberemo enega od ukazov in spremenimo vhodne parametre, nato pa ukaz z novimi parametri izvršimo.
  - **Repeat Last Command** – poženemo zadnji izvedeni ukaz na naslednjih objektih, ki so naloženi v Pajku (omrežjih, razbitjih,. . .).
  - **Store constant(s) reported by last command to Scalar(s)** – Med izvajanjem nekaterih ukazov se v okno report izpisujejo konstante (npr. vsi ukazi Info, komponente (število, velikost največje), usredinjenost omrežja, premer,...). Konstante, ki so bile izpisane ob *zadnjem* ukazu, lahko shranimo v skalar (vektor dimenzije 1) in uporabimo kasneje (npr. za normalizacijo omrežja ali vektorja).



Slika 4: Vse različne triade.

- **Info**
  - **Show Report Window** - Pokažemo okno z izpisi.
  - **Memory** – Informacija o porabljenem in razpoložljivem prostoru v pom-

nilniku.

  - **About** – Splošni podatki o programu (avtor, verzija,...).

- **Tools**

  - **Pajek** – Klicanje 'običajnega' programa Pajek, potem ko smo s programom PajekXXL izločili manjše omrežje. Pri tem lahko točkam omrežja, ki je bilo poslano v program Pajek, pripišemo še 'prave' oznake in ostale lastnosti točk (npr. oblike, koordinate...). Poleg omrežja lahko v program Pajek direktno pošljemo še odgovarjajoče razbitje in/ali vektor.
  - **R**
    - **Send to R** – Statistični program R pokličemo s trenutnim vektorjem / omrežjem, vektorji / omrežji določenimi s skupino (Cluster) ali vsemi trenutnimi vektorji in / ali omrežji.
    - **Locate R** – Poiščemo program R (Rgui.exe ali Rterm.exe) na disku.
  - **SPSS**
    - **Send to SPSS** – Statistični program SPSS pokličemo s trenutnim razbitjem, vektorjem / omrežjem, razbitji / vektorji določenimi s skupino (Cluster) ali vsemi trenutnimi razbitji in / ali vektorji.
    - **Locate SPSS** – Poiščemo program SPSS (runsyntx.exe) na disku.
  - **Excel**
    - **Send to Excel** – Program Excel pokličemo s trenutnim omrežjem / razbitjem / vektorjem, razbitji določenimi s skupino, vsemi razbitji, vektorji določenimi s skupino, vsemi vektorji, ali vsemi razbitji in vektorji.
    - **Locate Excel** – Poiščemo program Excel (Excel.exe) na disku.
  - **Export to Delimited File** – Omrežja, razbitja in/ali vektorje zapišemo na datoteko kjer so podatki ločeni z izbranimi ločili. To datoteko lahko potem uvozimo v druge programe, npr Excel.
  - **Delimited File Properties** – Določimo format zgenerirane datoteke ločene z izbranimi ločili.
    - **Delimiter** – Izberemo ločilo (*delimiter*), ki naj se uporablja pri izvozih v datoteke ločene z izbranimi ločili ali Excel. Ločilo je lahko *tabulator*, *podpičje*, *vejica* ali *presledek*. Glede na to ali izberemo format *ASCII* ali *UTF8* lahko izberemo tudi dve različni ločili:

    - **ASCII** - Izberemo ločilo, ki naj se uporabi, če izbira **Save Files as Unicode UTF8 with BOM** ni izbrana (v **Options / Read - Write**). Privzeto ločilo za datoteke te vrste je *tabulator*.
    - **UTF8** - Izberemo ločilo, ki naj se uporabi, če je izbira **Save Files as Unicode UTF8 with BOM** izbrana (v **Options / Read - Write**). Privzeto ločilo za datoteke te vrste je *podpičje*.
  * **Add BOM to UTF8 files** – Če shranjujemo na datoteko UTF8, naj se na začetek take datoteke doda še BOM (Byte Order Mark).

– **Web Browser** – Kateri spletni brskalnik naj se odpre, če v oknu *Draw* klikemo na izbrano točko s *Shift* in desno tipko na miški.

– **Inkscape** – Poiščemo kje je nameščen program **Inkscape** (pri izvozih matrik v SVG in PDF si deloma pomagamo s programom Inkscape).

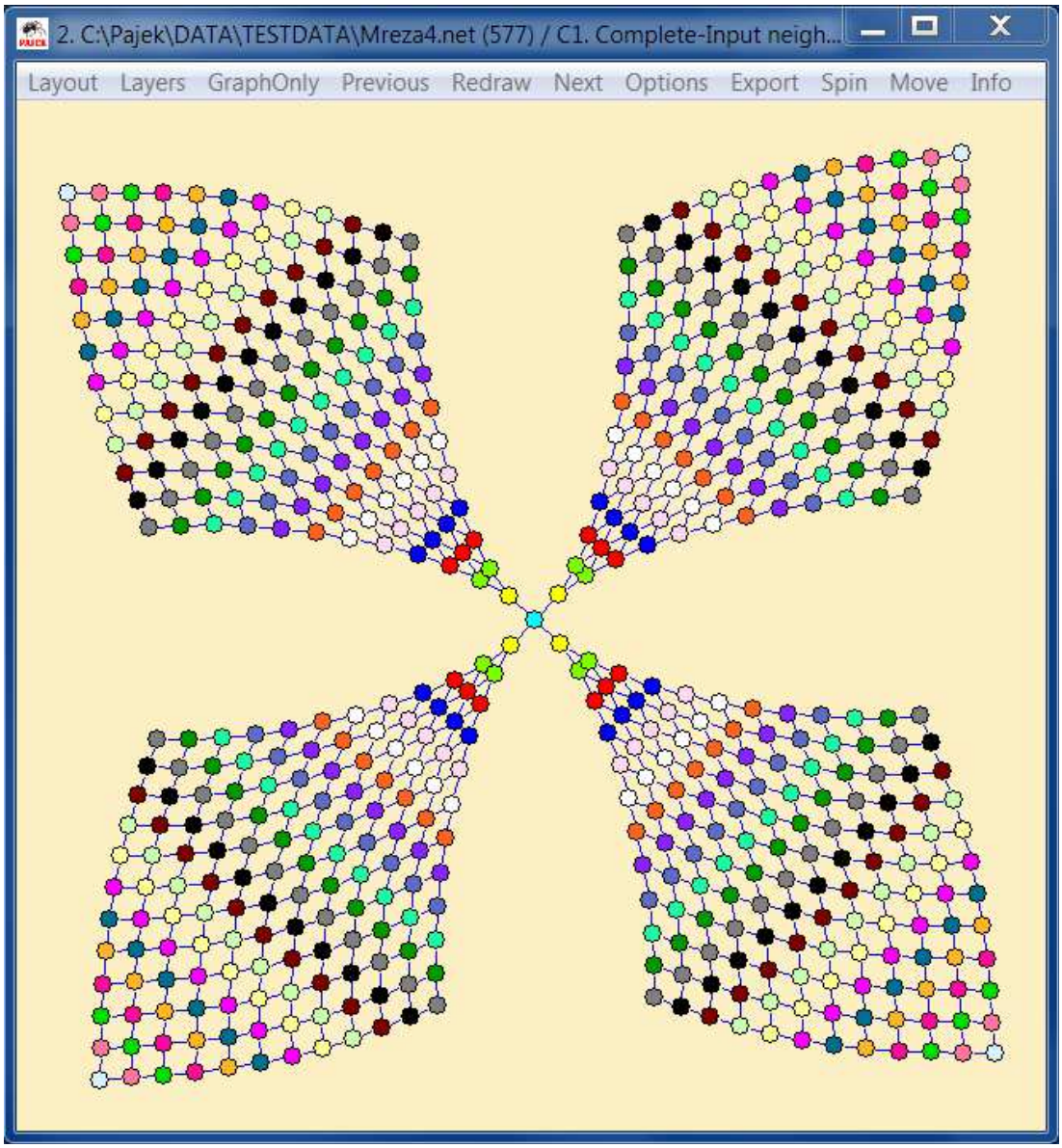
– **Add Program** – v menu tools dodamo klic novega zunanjega programa z ustreznimi parametri.

– **Edit Parameters** – spremenimo vhodne parametre za izbrani zunanji program.

– **Remove Program** – iz menuja tools izločimo izbrani program.

Risanje omrežij

Za risanje omrežij se odpre posebno (grafično) okno z novim naborom ukazov (Draw). Okno z izbrano sliko omrežja je prikazano na sliki 5. V tem oknu lahko točke pre-



Slika 5: Okno za risanje omrežij v programu Pajek.

mikamo ročno (z uporabo leve tipke na miški), izbiramo del slike (z desno tipko), dodajamo in odstranjujemo povezave, ki pripadajo določeni točki (točko izberemo z desno tipko). S pritiskom na tipke X, Y, Z, S, x, y, z, s zavrtimo sliko v smeri osi določeni s črko (S ali s pomeni vrtenje okoli osi, ki jo izberemo s Spin/Normal).

Če pa zahtevamo risanje omrežja z izbranim razbitjem (Draw / Network + First Partition), se skupine točk označijo z različnimi barvami. Slika 6 prikazuje katere barve predstavljajo določene številke skupin. Te barve lahko zamenjamo v izbiri **Options/Colors/Partition Colors**.

Dodatno lahko točkam povečujemo številko skupine za 1 z izbiranjem točke s srednjo tipko na miški. Če pa je tenutno izbran le del omrežja, s pritiskom na desno tipko povečamo številke skupin vseh točk, ki so trenutno izbrane (vidne). Podobno lahko z levo tipko na miški premikamo vse točke, ki pripadajo dani skupini, s pritiskom na mesto v bližini točke, ki pripada tej skupini. V primeru risanja omrežja z označenimi skupinami se v izbiri pojavijo nekatere nove možnosti: risanje po nivojih, ki so določeni s skupinami, optimizacija energije pri kateri so točke v izbrani skupini nepremične. . . .

Če pa zahtevamo risanje omrežja z izbranim vektorjem (Draw / Network + First Vector), določa izbrani vektor velikosti točk. Izbira Draw / Network + First Vector + Second Vector omogoča prikaz dveh vektorjev (prvi vektor določa širino, drugi pa višino točke).

V nadaljevanju so našteti in kratko opisani še ukazi, ki so na voljo v oknu za risanje.

Pregled in kratek opis ukazov za prikaz omrežij

- **Layout** – Določanje prikazov omrežij
  - **Circular** – Postavljanje točk na krožnico
    - **Original** – v vrstnem redu določenim z omrežjem
    - **using Permutation** – v vrstnem redu določenim s trenutno permutacijo
    - **using Partition** – zgenerira toliko krožnic kot je skupin v razbitju in nanje razvrsti točke, ki pripadajo posameznim skupinam. Središče posamezne krožnice je v težišču skupine.
    - **Random** – v slučajnem vrstnem redu
  - **Energy** – Avtomatično določanje prikazov omrežij z uporabo energijskih risanj.
    - **Kamada-Kawai** – Algoritem za določanje prikaza omrežja z minimalno skupno energijo v ravnini.

Slika 6: Številke skupin in ustrezne barve pri uporabi Draw / Network + First Partition.

- **Free** – v toku minimizacije skupne energije se točke prosto premikajo po ravnini.
- **Separate Components** – Optimiziraj vsako šibko povezano kom-

ponento posebej. Na koncu razporedi komponente v ravnini.

- **Optimize inside Clusters only** – Optimiziraj samo podomrežja določena z razbitjem - optimizira se točke znotraj skupin ne med skupinami.
- **Fix first and last** – Položaja prve in zadnje točke omrežja sta na nasprotnih straneh slike in se med optimizacijo ne spreminjata.
- **Fix One Vertex in the Middle** – Izbrano točko omrežja postavimo na sredino slike in je med optimizacijo ne premikamo.
- **Selected group only** – Optimiziramo samo trenutno izbrani del omrežja.
- **Fix selected vertices** – Z razbitjem izberemo nekaj nepremičnih točk, tako da točke postavimo v skupino 1. Te točke ostanejo do konca optimizacije na začetnih mestih (se med optimizacijo ne premikajo).

* **Fruchterman-Reingold** – Drugi algoritem za določanje prikaza omrežja z minimalno skupno energijo (hitrejši kot Kamada-Kawai).
  - **2D** – Optimizacija v dveh razsežnostih.
  - **3D** – Optimizacija v treh razsežnostih.
  - **Factor** – Optimalna razdalja med točkami, ki jo želimo doseči z optimizacijo.
* **Starting positions** – Začetna razmestitev točk za energijska risanja. Možne so naslednje razmestitve:
  - slučajna razmestitev točk v ravnini ali prostoru;
  - razmestitev točk na krožnici;
  - podana razmestitev točk v ravnini;
  - podane koordinate z, ki ostanejo nespremenjene do konca optimizacije.

– **Pivot MDS** – Avtomatično določanje prikazov omrežij z uporabo algoritma Pivot MDS (Brandes and Pich). Rezultat so slike v 2D ali 3D. Predstavnike lahko izberemo slučajno ali jih pripravimo v skupini. Algoritem je hitrejši kot energijska risanja in ga lahko uporabimo za večja omrežja (omrežja z nekaj 100 tisoč točkami).

– **VOS Mapping** – Določanje slike omrežja z algoritmom Visualization of Similarities Mapping (Van Eck in Waltman). Ponavadi dobimo lepe slike za gosta omrežja, kjer vrednosti na povezavah predstavljajo podobnosti. V algoritmu so uporabljene intenzivne operacije na matrikah, zato je časovno

in prostorsko precej zahteven in postane pri omrežjih z več kot 20.000 precej počasen. Optimizacijo nadzorujemo z dvema parametroma: *Number of Restarts* (ponavadi zadošča samo eno poganjanje) in *Maximum Number of Iterations in each Restart* (ponavadi je 100 ponovitev dovolj).

  - **EigenValues** – Določitev prikaza na osnovi lastnih vektorjev po Lanczosevem algoritmu. Možno je upoštevati tudi vrednosti na povezavah.
    - **1 1 1** – Izbira dveh ali treh lastnih vrednosti, za katere naj se izračunajo lastni vektorji. Lastne vrednosti so lahko tudi večkratne. Primeri:
      - **1 1 1** – izračunaj 3 lastne vektorje, ki pripadajo prvi (največji) lastni vrednosti;
      - **1 1 2** – izračunaj 2 lastna vektorja, ki pripadata prvi (največji) lastni vrednosti in lastni vektor, ki pripada drugi lastni vrednosti;
      - **1 2 2** – izračunaj lastni vektor, ki pripada prvi (največji) lastni vrednosti in dva lastna vektorja, ki pripadata drugi lastni vrednosti;
      - **1 2 3** – izračunaj po en lastni vektor, ki pripada prvi, drugi in tretji lastni vrednosti;
      - **1 1** – izračunaj 2 lastna vektorja, ki pripadata prvi (največji) lastni vrednosti (slika v ravnini).
  - **Tile Components** – Razporedi ločene šibko povezane komponente v ravnini.

- **Layers** – Ta izbira je na voljo le, če pri risanju omrežja upoštevamo tudi razbitje.
  - **Type of Layout** – Izbira vrste prikaza: (2D – prikaz v ravnini, 3D – prikaz v prostoru). Glede na to se pojavi ustrezna izbira.
  - **In y direction** – Omrežje narišemo po nivojih – koordinate točk y so določene z razbitjem. V okviru nivojev se točke razmestijo enakomerno po naraščajoči številki točke.
  - **In y direction+random in x** – Razlika v primerjavi s predhodno možnostjo je le v tem, da se točke v okviru nivojev razmestijo enakomerno a v slučajnem vrstnem redu.
  - **In z direction** – Omrežje narišemo po nivojih – koordinate z so določene z razbitjem. Koordinate x in y ostanejo nespremenjene.
  - **In z direction + random in xy** – Omrežje narišemo po nivojih – koordinate z so določene z razbitjem. V okviru nivojev se koordinate x in y določijo slučajno.

  - **Averaging x coordinate** – Potem, ko smo točke razmestili po nivojih v ravnini, postavljamo koordinato x vsake točke na povprečno vrednost koordinat x njenih sosedov. Postopek ponavljamo toliko časa, da se koordinate ustalijo.
  - **Averaging x and y coordinates** – Potem, ko smo točke razmestili po nivojih v prostoru, postavljamo koordinati x in y vsake točke na povprečno vrednost koordinat x in y njenih sosedov. Postopek ponavljamo toliko časa, da se koordinate ustalijo.
  - **Tile in x direction** – Točke, ki so razmeščene po nivojih v ravnini, razmaknemo na izbrano najmanjšo oddaljenost v smeri x.
  - **Tile in xy plane** – Točke, ki so razmeščene po nivojih v prostoru, razmaknemo na izbrano najmanjšo oddaljenost v smereh x in y.
  - **Optimize layers in x direction** – Optimizacija slike po nivojih v ravnini z minimizacijo skupne dolžine povezav.
  - **Optimize layers in xy plane** – Optimizacija slike po nivojih v prostoru z minimizacijo skupne dolžine povezav.
  - **Resolution** – Izbira najmanjše dopustne razdalje pri razmikanju točk.

- **GraphOnly** – Omrežje ponovno narišemo brez dodatnih oznak točk in povezav.
- **Previous** – Nariše se slika prejšnjega omrežja (in/ali razbitja), ki je shranjena pomnilniku.
- **Redraw** – Slika omrežja se ponovno nariše.
- **Next** – Nariše se slika naslednjega omrežja (in/ali razbitja), ki je shranjena v pomnilniku.
- **ZoomOut** – Povečaj na prejšnji pogled (dostopno le kadar gledamo samo izbrani del slike).
- **Options** – Dodatne nastavitve in preoblikovanja slik.
  - **Transform** – Preoblikovanja slik.
    * **Fit area** – Celoten prostor, ki je na voljo, zapolnimo s sliko omrežja. Pri zapolnjevanju imamo na voljo dve možnosti:
      · **max(x), max(y), max(z)** – Sliko raztegnemo, da zapolni okvir. Razteg se zgodi v vseh treh smereh neodvisno.

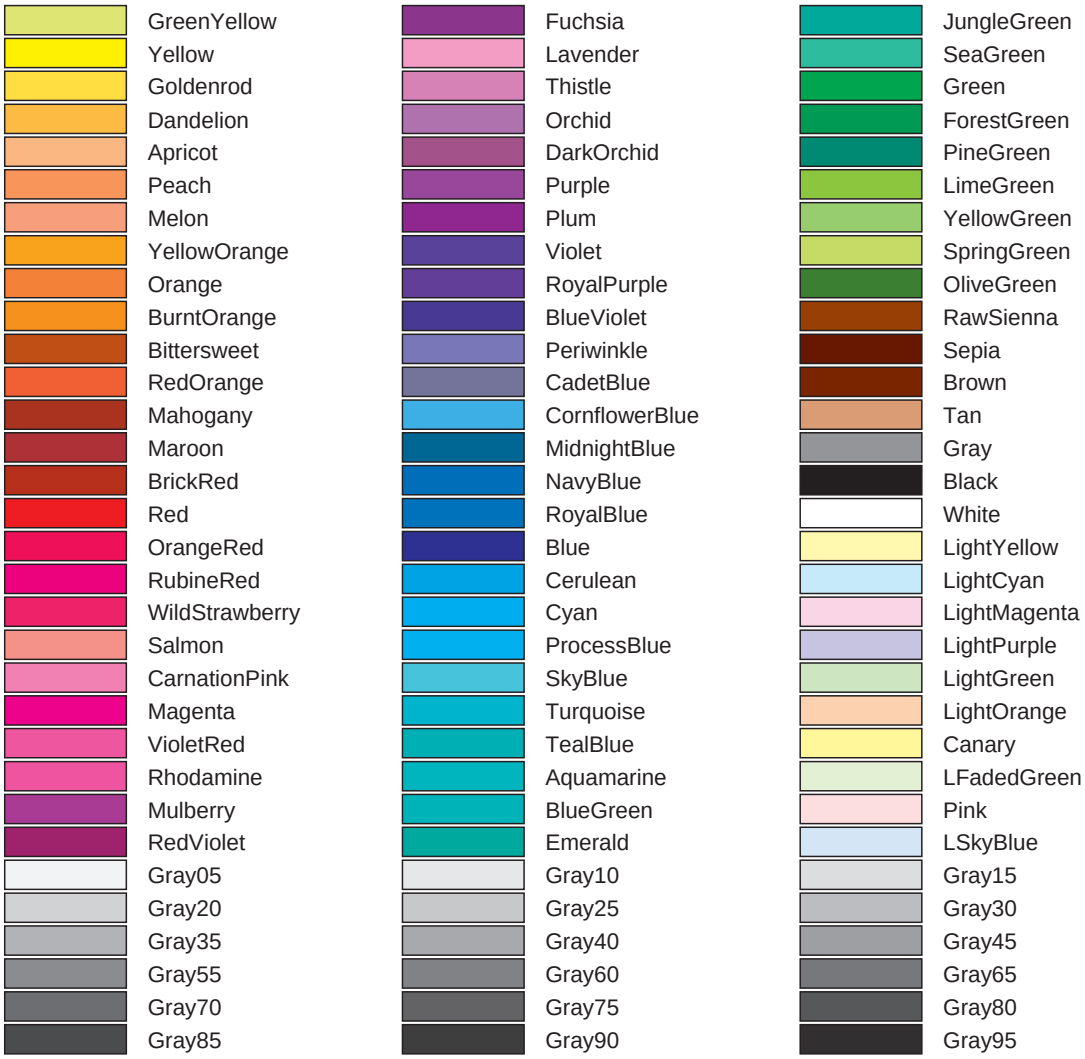
        - **max(x,y,z)** – Sliko raztegnemo kolikor je mogoče, a pri raztegovanju ohranimo razmerja med koordinatami (razteg glede na največjo razdaljo v vseh treh smereh).
    - **Resize** – Sliko povečamo / pomanjšamo za izbrani faktor v določenih smereh.
    - **Translate** – Sliko premaknemo v določeni smeri.
    - **Reflect y axis** – Sliko prezrcalimo okoli osi y.
    - **Set z-coordinate to 0.5** – sliko omrežja projeciraj v ravnino.
    - **Rotate 2D** – Sliko zavrtimo v ravnini xy za določen kot.
    - **FishEye** – Transformacija FishEye (kartezijska ali polarna). Če trenutno ni izbrana nobena točka, se za žarišče slike izbere sredina slike, sicer se za žarišče izbere dana točka.
    - **Resize Clusters Area** – Povečaj (ali pomanjšaj) območje, ki ga zaseda dana skupine za izbrani faktor.
- **Values of lines** – Vrednostim na povezavah določimo pomen:
    - **Forget** – Obravnavaj vse povezave enako ne glede na vrednosti na povezavah.
    - **Similarities** – Vrednosti na povezavah predstavljajo podobnosti.
    - **Dissimilarities** – Vrednosti na povezavah predstavljajo različnosti (razdalje).

  Pomen, ki ga izberemo, se upošteva pri energijskih risanjih in risanjih z uporabo lastnih vektorjev.
- **Mark vertices using** – Na sliki omrežja
    - **Labels** – označimo točke z oznakami;
    - **Numbers** – označimo točke s številkami točk;
    - **No Labels** – ne označujemo točk;
    - **No Labels no Arrows** – ne označujemo točk in vse povezave rišemo brez puščic;
    - **Mark Cluster Only** – označujemo samo tiste točke, ki se nahajajo v dani skupnini;
    - **Vector Values** – dodatno označimo točke z vrednostmi iz vektorja (istočasno mora biti izbran tudi vektor (ali dva vektorja) enake dimenzije);
    - **Clusters of Second Partition** – dodatno označimo točke z razredi iz drugega razbitja (istočasno mora biti izbrano tudi drugo razbitje enake dimenzije);

    - **Cluster Symbols of Second Partition** – dodatno označimo točke s simboli, ki pripadajo posameznim razredom iz drugega razbitja (istočasno mora biti izbrano tudi drugo razbitje enake dimenzije). Ko izvozimo sliko v SVG ali EPS, se simboli izpišejo v sredine točk (v primeru izvoza v EPS so simboli lahko samo ASCII znaki).
    - **Labels as Tooltips** – oznake (in lastnosti) točk se izpišejo kot namig ('tooltip'), ko se kazalec miške dotakne točke (in ne kot besedilo zraven točke).
    - **Labels Centered** – oznake točk naj se izpisujejo v sredini točk.
- **Lines** – Izbira načina prikaza in označevanja povezav.
    - **Draw Lines** – Prikaz povezav.
        - **Edges** – Prikazujemo neusmerjene povezave.
        - **Arcs** – Prikazujemo usmerjene povezave.
        - **Relations** – Prikazujemo vse povezave (pustimo prazen niz) ali samo povezave, ki pripadajo izbranim relacijam, npr. 1-3,6,10-15.
    - **Mark Lines** – Izpis dodatnih informacij ob povezavah.
        - **No** – Povezave samo narišemo in jih dodatno ne opisujemo.
        - **with Labels** – Prikazujemo oznake povezav.
        - **with Values** – Prikazujemo vrednosti na povezavah.
    - **Different Widths** – Debelina povezava naj bo določena z vrednostjo na povezavi.
    - **GreyScale** – Sivina s katero je povezava narisana naj bo določena z vrednostjo na povezavi.
- **Size** – Določimo velikost točk, debelino roba točk, debeline povezav in puščic ter velikost simbolov in pisave. Pri velikosti točk lahko izberemo *AutoSize* – povprečna velikost točke bo vedno enaka. Velikosti točk lahko preberemo z vhodnih datotek (x_fact in y_fact) ali pa določimo z enim ali dvema trenuto izbranima vektorjema. Velikost pisave lahko določimo s tretjim razbitjem.
- **Colors** – Izberemo barvo ozadja, točk, roba točk, neusmerjenih in usmerjenih povezav, pisave ter simbolov. Spreminjamo lahko tudi barve, ki naj predstavljajo dane razrede iz razbitja (s kliki na možne barve). Barvo pisave lahko določimo z drugim razbitjem. Barvo simbolov določimo s tretjim razbitjem. Barva povezave lahko predstavlja številko relacije. Tabelo barv in ustreznih številk relacij izberemo v **Relation Colors**.

  Pri risanju točk, roba točk ter usmerjenih in neusmerjenih povezav lahko zahtevamo uporabo barv, ki so navedene na vhodni datoteki (*As Defined*

on Input File). Barvo točke določimo s parametrom ic (*interior color*), barvo roba točke s parametrom bc (*border color*), barvo povezave pa s parametrom c (*color*). Barve, ki jih razpozna Pajek, so prikazane na sliki 7. Poleg barv, ki jih desežemo z imeni, lahko opišemo barve tudi s formatom RGB (npr. RGBFF0000 ali RGB(1,0,0)) in CMYK (npr. CMYK00FF0000 ali CMYK(0,1,0,0)). V nizu, ki določa barvo, ne smejo nastopati presledki.



Slika 7: Barve v programu Pajek.

– **Symbols for Partition Clusters** – razredom iz razbitja pripišemo simbole. Simboli so lahko poljubni znaki Unicode.

– **Layout** – Dodatne nastavitve:

  * **Redraw** – Omrežje naj se na novo izriše
    · **during Vertex Moving** - med premikanjem točk;
    · **after Vertex Moving** - po premikanju točk;
    · **on Layout Paint** - če je bilo okno Draw prekrito;
    · **on Layout Resized** - če je bilo okno Draw povečano ali pomanjšano;
    · **Rendering Refresh Period** - perioda v kateri se risanje omrežja osvežuje. V času risanja omrežja se izpiše še obvestilo 'Wait'. Če za periodo izberemo 0, se slika ne osvežuje, niti se ne izpiše obvestilo. Nekatere možnosti risanja so omogočene samo, če osveževanje ni vključeno (npr. vrtenje slike z uporabo tipk 'x', 'y', 'z', ali 's', uporaba drsnikov - 'ScrollBars').
  * **Real xy proportions** – Pri spreminjanju velikosti okna za risanje naj se ohranjajo prava razmerja med velikostmi v smereh x in y.
  * **Arrows in the Middle** – Puščice naj se rišejo na sredini povezav in ne pri končnih točkah.
  * **Vertices of size 0** – Kako obravnavati točke z velikostjo 0 (velikost točke je določena na vhodni datoteki ali s pomočjo vektorja)? Točke z velikostjo 0 lahko prikazujemo ali ne, pravtako lahko prikazujemo ali ne tudi povezave, ki imajo vsaj eno od krajišč v točki z velikostjo 0.
  * **Decimal Places** – Koliko decimalnih mest naj se uporabi pri označevanju točk z vrednostmi iz vektorja.
  * **Show SubLabel** – Izberemo podniz oznake točke, ki naj se prikazuje na slikah omrežij.
  * **Labels with Transparent Background** – Izberemo izpis oznak s prosojnim ozadjem.

– **ScrollBar On/Off** – Vključitev pomičnih letev, ki jih uporabljamo v dva namena:

  * V primeru prikaza celotne slike jih uporabljamo za vrtenje slike.
  * Če je izbran le del slike, jih uporabljamo za premikanje vidnega dela slike.

– **Interrupt** – Izbira prekinitvenega intervala med optimizacijo. Ko poteče izbrani čas, lahko zahtevamo potrditev nadaljevanja ali pa z optimizacijo

zaključimo. S tem uvedemo možnost prekinitve ob predolgih risanjih velikih omrežij.

– **Previous/Next** – Izbira načina prikaza zaporedja omrežij:
  * **Max. number** – Prikaz željenega števila zaporednih omrežij. Če je to število višje od števila obstoječih omrežij, se prikaz predčasno zaključi.
  * **Seconds to wait** – Prekinitev med zaporednimi prikazi.
  * **Optimize Layouts** – Optimizacija trenutnega prikaza. Če je zaporedje omrežij dobljeno iz istega omrežja, je smiselno izbrati še nastavitev Energy/ Starting Positions/ Given xy, da se optimizacija začne z obstoječimi koordinatami.
    · **Kamada-Kawai** – Trenutni prikaz se optimizira s Kamada-Kawaijevim algoritmom.
    · **2D Frucht. Rein.** – Trenutni prikaz se optimizira s Fruchterman-Reingolodovim algoritmom v ravnini.
    · **3D Frucht. Rein.** – Trenutni prikaz se optimizira s Fruchterman-Reingolodovim algoritmom v prostoru.
    · **No** – Prikazov ne optimiziramo, ampak samo prikazujemo zaporedje.
  * **Aplly to** – Kaj naj se zgodi, ko izberemo Previous ali Next:
    · **Network** – Nariše se slika predhodnega / naslednjega omrežja. Če smo v načinu risanja Draw / Network + First Partition in ima tudi naslednje omrežje enako število točk, bo isto razbitje določalo barve točk novega omrežja.
    To izbiro uporabljamo za prikaz različnih omrežij (ki se ujemajo po številu točk) z istim razbitjem.
    · **Partition** – Če smo v načinu risanja Draw / Network + First Partition: Isto omrežje se nariše z uporabo predhodnega ali naslednjega razbitja.
    To izbiro uporabljamo za prikaz različnih razbitij na istem omrežju.
    · **Vector** – Če smo v načinu risanja Draw / Network + First Vector: Isto omrežje se nariše z uporabo predhodnega ali naslednjega vektorja, ki določa velikosti točk.
    To izbiro uporabljamo za prikaz različnih vektorjev na istem omrežju.

Opisana navodila veljajo, če je izbran samo en objekt (omrežje, razbitje ali vektor). Možne pa so tudi vse kombinacije po dveh ali treh objektov naenkrat. Na ta način lahko istočasno preklapljamo omrežja, razbitja in vektorje, če se le ta ujemajo v dimenzijah.

- **Export** – Izpisi v izhodne oblike, ki so namenjene vključevanju v urejevalnike besedila ali pregledovanju s posebnimi pregledovalniki za dvo ali trirazsežne formate.

  2D – dvorazsežni formati:

  - **EPS/PS** – Zapis v obliki Encapsulated PostScript EPS (samostojna slika) ali PS (brez začetne glave, zato moramo v urejevalniku, v katerega vključimo sliko, glavo posebej definirati). Za pregledovanje slik v obliki EPS lahko uporabljamo program GSView, ki je prosto dostopen na naslovu: ftp://ftp.cs.wisc.edu/pub/ghost/rjl/
  - **SVG** – Izpis v obliko SVG (Scalable Vector Graphics). Sliko v SVG lahko nadalje urejamo s programom Inkscape.

    Sliki lahko dodamo zvezno prehajanje barve ozadja iz ene v drugo (linear ali radial gradients). Tri barve ozadja med katerimi se prehaja izberemo v oknu Export/Options.

    * **General** – Izpis brez možnosti izbir posameznih delov slike.
    * **Labels/Arcs/Edges** – Izpis z možnostjo vključevanja in izključevanja oznak, usmerjenih in/ali neusmerjenih povezav.
    * **Partition** – Izpis z upoštevanjem trenutnega razbitja ali dveh izbranih razbitij. V primeru, da dveh razbitij ne izberemo, določa trenutno razbitje tako razrede kot tudi barve točk. Če pa izberemo dve razbitji, določa prvo barve točk, drugo pa razrede. Lahko pa drugo in tretje razbitje porabimo tudi v druge namene: Drugo razbitje lahko uporabimo za prikaz simbolov ali za izbiro barve oznake točke. Tretje razbitje lahko uporabimo za barvo simbolov ali velikost izpisa oznake točke.
      - **Classes** – Možnost vključevanja in izključevanja posameznih razredov iz razbitja in povezav med razredi.
      - **Classes with semi-lines** – Možnost vključevanja in izključevanja posameznih razredov iz razbitja. Povezave med razredi se rišejo kot polpovezave.
      - **Nested Classes** – Višji razredi so gnezdeni v nižjih – kadarkoli vključimo izbrani razred, se vključijo tudi vsi višji razredi in

izključijo vsi nižji razredi od izbranega (primerno za prikazovanje jeder)

  - **Line Values** – Izpis z upoštevanjem vrednosti na povezavah. Vnesemo lahko meje razredov ali število razredov. Če vnesemo $\#n$, se zgenerira n enako širokih razredov. Glede na dobljene meje dobimo podmnožice povezav (in točk, ki so njihova krajišča). Dobljene podmnožice lahko prikazujemo ali skrivamo v okviru spletnega pregledovalnika.
    - **Classes** – Možnost prikazovanja in skrivanja povezav z izbranimi vrednostmi in pripadajočih točk.
    - **Nested classes** – Možnost prikazovanja in skrivanja povezav z izbranimi ali višjimi vrednostmi in pripadajočih točk.
    - **Options** – Dodatne možnosti za vizualno predstavitev vrednosti povezav: **Different Colors** – podmnožice povezav se narišejo z različnimi barvami. **Using GreyScale** – podmnožice povezav se narišejo z različnimi sivinami. **Different Widths** – podmnožice povezav se narišejo z različnimi debelinami.
  - **Multiple Relations Network** – Izvoz večrelacijskega omrežja v SVG. Uporabnik lahko vključuje in izključuje vidnost posameznih relacij.
  - **Current and all Subsequent** – Če je opcija izbrana, se v obliko SVG izpišejo trenutno omrežje in vsa sledeča omrežja. Za vsako omrežje se zgenerira svoja datoteka v html. Datoteke dobijo naslednja imena: file0001.htm, file0002.htm,..., file9999.htm in so med sabo povezane z dodatnimi sidri (Previous/Next). Če se tudi razsežnosti razbitij/vektorjev ujemajo z ustreznimi omrežji, se razbitja uporabijo za določitev barv, vektorji pa za določitev velikosti točk. Pri tem se za naslednji objekt lahko vzame katerakoli kombinacija objektov: omrežje, razbitje in vektor (samo eden od treh, poljubna dva, ali vsi trije) glede na nastavitev v Options/Previous/Next/Aplly to v oknu Draw.

- **JPEG** – Izpis v obliko JPEG/JPG. Izberemo lahko še kvaliteto stiskanja (0-100) in črnobeli izpis.
- **Bitmap** – Izpis v obliko Windows bitmap (bmp).
- **VOSviewer** – Pokliči VOSviewer (1.5.2 ali novejši) s Pajkovim omrežjem, razbitjem in/ali vektorjem.

3D - trirazsežni formati:

- **X3D** – Izpis v obliko X3D (vektorska grafika, ki temelji na standardu XML

in je naslednica VRML). X3D format lahko uvozimo v program za tridimenzionalno tiskanje shapeways: https://www.shapeways.com/

– **Kinemages** – Izpis v obliko Kinemages. Za pregledovanje slik v tej obliki potrebujemo program Mage. Program je prosto dostopen na naslovu: http://www.prosci.org/Kinemage/MageSoftware.html
  * **Current Network Only** – V obliko Kinemages se izpiše samo trenutno omrežje. Pri tem se lahko uporabita dve razbitji, ki ju izbereno s Partitions: eno za generacije, drugo za barve točk. Generacije nato predstavimo s preprosto animacijo.
  * **Current and all Subsequent** – V obliko Kinemages se izpišejo trenutno omrežje in vsa sledeča omrežja. Med dobljenimi prikazi omrežij nato v programu Mage prehajamo z uporabo ukazov KINEMAGE/Next ali Ctrl N. Če se tudi razsežnosti razbitij/vektorjev ujemajo z ustreznimi omrežji, se razbitja uporabijo za določitev barv, vektorji pa za določitev velikosti točk. Pri tem se za naslednji objekt lahko vzame katerakoli kombinacija objektov: omrežje, razbitje in vektor (samo eden od treh, poljubna dva, ali vsi trije) glede na nastavitev v Options/Previous/Next/Aplly to v oknu Draw.
  * **Multiple Relations Network** – Izvoz z možnostjo izbire vidnosti izbranih relacij.
– **VRML** – Izpis v obliko VRML (Virtual Reality Modeling Language). Za pregledovanje slik v obliki VRML lahko uporabljamo program Cortona Viewer, ki je prosto dostopen na naslovu: http://www.parallelgraphics.com/products/cortona/
– **MDL MOL file** – Izpis v kemijsko obliko MDL Molfile. Za pregledovanje slik v tej obliki potrebujemo program Chime, ki se ga vključi v Netscape. Program je prosto dostopen na naslovu: http://www.mdli.com/

Options – Izbira dodatnih nastavitev pri izpisih v obliki EPS/PS/SVG (dodatni robovi, barve, velikosti, oblike...) in X3D/VRML.

Append to Pajek project file – Trenutno omrežje izpiše na konec projektne datoteke (ki jo potem lahko uporabi program **PajekToSvgAnim** za izdelavo anmacije v SVG).

– **Select file** – Izbira projektne datoteke.
– **Append** – Dodajanje omrežja na konec izbrane projektne datoteke.

Nasvet: Potem ko dodamo eno omrežje na projektno datoteko, lahko več omrežij, ki sledijo trenutnemu omrežju, dodamo na isto datoteko z uporabo ukaza **Macro / Repeat Last Command** (namesto, da omrežja dodajamo 'ročno' eno za drugim).

- **Spin** – Vrtenje slike.
  - **Spin around** – Začnemo z vrtenjem slike okrog izbrane osi.
  - **Perspective** – Pri vrtenju naj se upošteva perspektiva (točke, ki so bolj oddaljene, naj bodo manjše).
  - **Normal** – Izberemo os vrtenja.
  - **Step in degrees** – Izberemo korak pri vrtenju v stopinjah.
- **Move** – Pri ročnem urejanju slike uvedemo dodatne omejitve:
  - **Fix** – Onemogočeno premikanje v x (ali y) smeri, ali onemogočeno spreminjanje razdalje od središča (kroženje).
  - **Grid** – Dopustna mesta za točke omrežja so samo tista na mreži izbranih razsežnosti.
  - **Circle** – Dopustna mesta za točke omrežja so samo tista na izbranih koncentričnih krožnicah.
  - **Grasp** – Določimo način premikanja skupin. Pri premikanju skupine naj se premaknejo:
    - **Closest Class Only** – samo točke, ki so v dani skupini;
    - **Closest Class and Higher** – točke, ki so v dani skupini in točke, ki so v višjih skupinah;
    - **Closest Class and Lower** – točke, ki so v dani skupini in točke, ki so v nižjih skupinah.
- **Info** – Izberemo lastnost lepe slike, ki jo želimo preveriti:
  - **Closest Vertices** – Poiščemo najbližji točki na sliki;
  - **Smallest Angle** – Poiščemo najmanjši kot med povezavama, ki imata eno krajišče v skupni točki;
  - **Shortest/Longest Line** – Poiščemo najkrajšo in najdaljšo povezavo;
  - **Number of crossings if lines** – Preštejemo število vseh presečišč povezav;
  - **Vertex Closest to Line** – Poiščemo točko, ki je najbližja povezavi;
  - **All Properties** – Preverimo vse naštete lastnosti lepe slike.

  - **Correlation (Layout, Geodesics)*** – Izračunamo korelacijo med razdaljami med točkami na sliki in razdaljami med točkami v omrežju.

- **FishEye** – Prikaz FishEye: Ko se z miško premikamo po sliki, se področje v okolici povečuje, kot bi ga pregledovali z lupo. S klikom nekje v oknu lahko ohranimo trenutne koordinate točk.

  - **Cartesian** – uporablja se kartezijska transformacija FishEye.
  - **Polar** – uporablja se polarna transformacija FishEye.
  - **Factor** – izberemo povečavo.
  - **Exit** – izključimo prikaz FishEye in povrnemo stare koordinate točk.