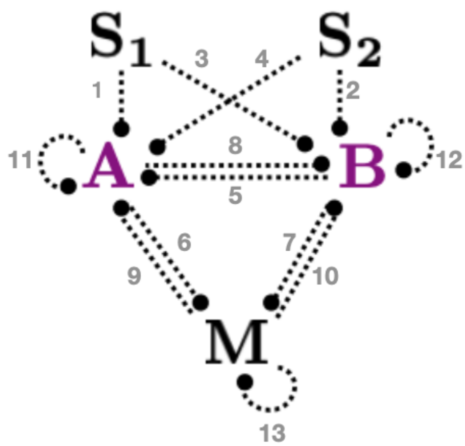
Minimal regulation network for value antagonism

In this code we extract minimal networks with nedges that shows value antagonism using regulation. The most general network looks like -



Now each network is perfectly characterized by two lists of P_i's and A_i's (i=1 to 13 for 13 possible edges). P_i=1(0) represents i^{th} edge being present(absent) and A_i=1(0) represents i^{th} edge being activation(repression) given P_i=1. The code below selects successful five node nedge(possible number of edges) network in terms of P_i's and A_i's that can satisfy value antagonism.

```
(*Extract all possible networks with nedges*)
nedges = 6; (*numbr of edges present*)
HA = 1; (*Power of auto activation da/dt=a^HA,
in our calculations we use linear auto-activation*)
maxedges = 13;
npts = 100;
smax = 100;
AAR = 1; (*Auto-activation rate/degradation rate, default is 1*)
nedgenetwork = {};
For[i = 0, i < 2^maxedges, i++,
 ls = PadLeft[IntegerDigits[i, 2], maxedges]; (*Getting list of P's*)
```

```
 (*Getting possible lists*)
 T = Total[ls]; (*Total number of edges present*)
 If[T == nedges, AppendTo[nedgenetwork, ls], 0]]
(*Only choosing network with nedges present*)

(*Extract list of Aᵢs, if Aᵢ is 1(0) and edge i is present Pᵢ=
 1 the interaction is activation(repression)*)
L = Length[nedgenetwork]; (*Length of three edge network*)
listofA = {}; (*possible values of 3 Aᵢ's present*)
For[i = 0, i < 2^nedges, i++, ls = PadLeft[IntegerDigits[i, 2], nedges];
 (*Getting possible list of A's*)
 AppendTo[listofA, ls]
 (*Append possible lists*)]
Print[listofA]
LA = Length[listofA]; (*Length of list of subsets in our case it is 8*)
listofallA = {}; (*All possible list of A's*)
For[i = 1, i ≤ L, i++, ls = nedgenetwork[[i]];
 (*i^{th} list in 3 edge network*)
 lP = Position[ls, 1]; (*List of position of 1's present*)
 lA = {};
 (*Initialising list of networks
  with A for a given network with 3 edges present*)
 For[j = 1, j ≤ LA, j++, ls = ConstantArray[0, maxedges];
  For[k = 1, k ≤ nedges, k++, pos = lP[[k]][[1]];
   (*position in A list which needs to be updated*)
   elem = listofA[[j]][[k]];
   (*corresponding elements in list of A's microlist*)
   ls[[pos]] += elem];
  AppendTo[lA, ls](*Append microlist to lA*)];
 AppendTo[listofallA, lA](*Append list lA to superlist*)]

NP = Length[nedgenetwork];  (*number of 3 edge netoworks with unique Pᵢ*)
NA = 2^nedges;  (*number of 3 edge netoworks with unique Aᵢ per NP network*)
For[i = NP, i ≥ 1, i--,
 Print[i];
 Plist = nedgenetwork[[i]]; (*Extracting three edge network*)
 (*extracting individual Pᵢ values*)
 P1 = Plist[[1]];
 P2 = Plist[[2]];
 P3 = Plist[[3]];
 P4 = Plist[[4]];
 P5 = Plist[[5]];
 P6 = Plist[[6]];
 P7 = Plist[[7]];
```

```
P8 = Plist[[8]];
P9 = Plist[[9]];
P10 = Plist[[10]];
P11 = Plist[[11]];
P12 = Plist[[12]];
P13 = Plist[[13]];
For[k = 1, k ≤ NA, k++,
 Alist = listofallA[[i]][[k]]; (*Corresponding list of A's*)
 (*extracting individual Ai values*)
 A1 = Alist[[1]];
 A2 = Alist[[2]];
 A3 = Alist[[3]];
 A4 = Alist[[4]];
 A5 = Alist[[5]];
 A6 = Alist[[6]];
 A7 = Alist[[7]];
 A8 = Alist[[8]];
 A9 = Alist[[9]];
 A10 = Alist[[10]];
 A11 = Alist[[11]];
 A12 = Alist[[12]];
 A13 = Alist[[13]];

 (*Both signals present*)
 (*Dynamical equation of A*)
 dadt = 1 - a + P1 * A1 * s1 + P4 * A4 * s2 +
   P5 * A5 * b + P9 * A9 * m + AAR * P11 * A11 * a ^ HA - a * (P1 * (1 - A1) * s1 +
      P4 * (1 - A4) * s2 + P5 * (1 - A5) * b + P9 * (1 - A9) * m + P11 * (1 - A11) * a);
 (*Dynamical equation of B*)
 dbdt = 1 - b + P2 * A2 * s2 + P3 * A3 * s1 +
   P8 * A8 * a + P10 * A10 * m + AAR * P12 * A12 * b ^ HA - b * (P2 * (1 - A2) * s2 +
      P3 * (1 - A3) * s1 + P8 * (1 - A8) * a + P10 * (1 - A10) * m + P12 * (1 - A12) * b);
 (*Dynamical equation of M*)
 dmdt = 1 - m + P6 * A6 * a + P7 * A7 * b + AAR * P13 * A13 * m ^ HA -
   m * (P6 * (1 - A6) * a + P7 * (1 - A7) * b + P13 * (1 - A13) * m);



 eqn = Solve[{dadt == 0, dbdt == 0,
    dmdt == 0 && s1 ≥ 0 && s2 ≥ 0 && a ≥ 0 && b ≥ 0 && m ≥ 0}, {a, b, m}];
 (*Successful solutions with positivity conditions*)
 aS = a /. eqn;
 bS = b /. eqn;
 mS = m /. eqn;
```

```
(*Write the equations with no signals present*)
dadt00 = dadt /. s1 → 0 /. s2 → 0;
dbdt00 = dbdt /. s1 → 0 /. s2 → 0;
dmdt00 = dmdt /. s1 → 0 /. s2 → 0;

eqn00 = Solve[{dadt00 == 0, dbdt00 == 0,
   dmdt00 == 0 && s1 ≥ 0 && s2 ≥ 0 && a ≥ 0 && b ≥ 0 && m ≥ 0}, {a, b, m}];
(*Successful solutions with positivity conditions*)
aS00 = a /. eqn00;
bS00 = b /. eqn00;
mS00 = m /. eqn00;


(*Write the equations with only signal 1 present*)
dadt1 = dadt /. s2 → 0;
dbdt1 = dbdt /. s2 → 0;
dmdt1 = dmdt /. s2 → 0;

eqn1 = Solve[{dadt1 == 0, dbdt1 == 0,
   dmdt1 == 0 && s1 ≥ 0 && s2 ≥ 0 && a ≥ 0 && b ≥ 0 && m ≥ 0}, {a, b, m}];
(*Successful solutions with positivity conditions*)
aS1 = a /. eqn1;
bS1 = b /. eqn1;
mS1 = m /. eqn1;


(*Write the equations with only signal 2 present*)
dadt2 = dadt /. s1 → 0;
dbdt2 = dbdt /. s1 → 0;
dmdt2 = dmdt /. s1 → 0;

eqn2 = Solve[{dadt2 == 0, dbdt2 == 0,
   dmdt2 == 0 && s1 ≥ 0 && s2 ≥ 0 && a ≥ 0 && b ≥ 0 && m ≥ 0}, {a, b, m}];
(*Successful solutions with positivity conditions*)
aS2 = a /. eqn2;
bS2 = b /. eqn2;
mS2 = m /. eqn2;



If[Length[mS] == 1, (*If a unique solution exist*)
 (*For a given network let us run a loop for *)
 For[j = 0, j < npts, j++,
```

```
(*Let us randomly choose s₁,s₂ values*)
inp1 = smax * RandomReal[]; (*s₁*)
inp2 = smax * RandomReal[]; (*s₂*)

(*Let us evaluate m for fixed value of s₁ and s₂*)
mP = mS /. s₁ → inp1 /. s₂ → inp2; (*m value for both signal present*)

mP1 = mS1 /. s₁ → inp1 /. s₂ → inp2;
(*m value for only signal 1 signal present*)

mP2 = mS2 /. s₁ → inp1 /. s₂ → inp2; (*m value for only signal 2 present*)

m00 = mS00 /. s₁ → inp1 /. s₂ → inp2;
(*m value for no signals present*)
(*Print and break if value antagonism is present*)
If[mP1〚1〛 > m00〚1〛 && mP2〚1〛 > m00〚1〛 && mP〚1〛 < 0.99 * Min[mP1〚1〛, mP2〚1〛],
 Print["list of Ps=", Plist, ", list of As=", Alist,
   ", Total edges=", Total[Plist]] && Break[], 0]], 0]]]
```