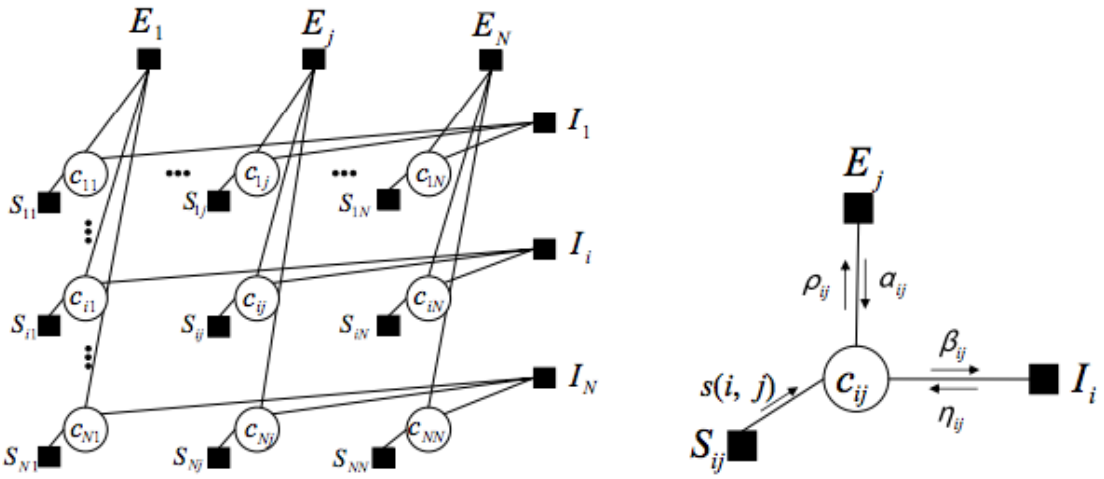
Binary Affinity Propagation in Flink
Design doc

1 Introduction

Aim of this document is explaining the design of the Binary Affinity Propagation [2] algorithm for Gelly (Flink). The Binary model of Affinity Propagation was published after the original Affinity Propagation [1] and tries to make it more understandable and easily extensible when adding new constraints to the model.

2 Binary Model for Affinity Propagation

The graphical model that needs to be implemented is shown in figure 1 and a brief explanation of it can be found below.



Let {cij} N j=1 be N binary variables associated with data point i(i ∈ {1, ..., N}), such that cij=1 iff the exemplar for point i is point j. In this notation cjj = 1 indicates that j is an exemplar. All assignments to exemplars and all exemplar choices can be described by the set of N2 binary variables {cij}i, j ∈ {1, ..., N}.

Each data point in affinity propagation clustering is assigned to a single exemplar. Therefore, the first constraint that must be accounted for in the binary variable formulation is that sumN j=1cij = 1. We refer to this as the 1−of−N constraint. An additional constraint from the original AP formulation is the exemplar consistency constraint stating that node i may choose j as its exemplar only if j chose itself as an exemplar. Figure 1a shows a factor graph for affinity propagation that uses the above variables and constraints. The 1 − of − N constraints are introduced via the I function nodes; in every row i of the grid, exactly one cij , j ∈ {1, ..., N}, variable must be set to 1. The E function nodes enforce the exemplar consistency

constraints; in every column j, the set of cij , i ∈ 1, ..., N, variables set to 1 indicates all the points that have chosen point j as their exemplar. For these points to be able to choose point j as an exemplar, j must also choose itself as an exemplar (cjj must also equal 1).

This model is based in the max-sum (log-domain max-product) formulation, where local functions are added to form the global objective function to maximize. After defining the I and E functions and taking the max-sum messages calculations the messages sent in this model can be derived, this messages are shown in figure 1b. Details on how to derive the messages are not explained here but can be found in the paper. The final messages sent and being calculated in this implementation are:

$$
\begin{aligned}
\beta_{ij} &= s(i,j) + \alpha_{ij} \\
\eta_{ij} &= -\max_{i \neq j} \beta_{ik} \\
\rho_{ij} &= s(i,j) + \eta_{ij} \\
\alpha_{ij} &= \begin{cases} \sum_{k \neq j} \max[\rho_{kj}, 0] & i = j \\ \min\left[0, \rho_{jj} + \sum_{k \notin \{i,j\}} \max[\rho_{kj}, 0]\right] & i \neq j \end{cases}
\end{aligned}
$$

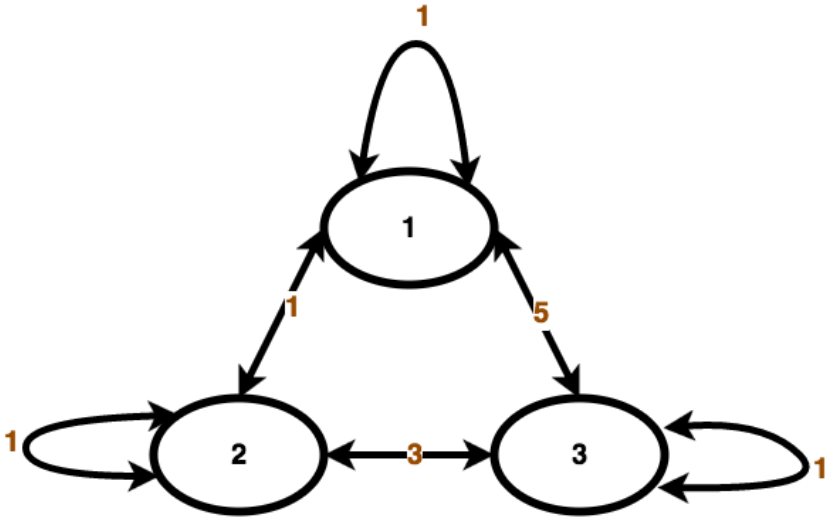
The αij messages are identical to the AP availability messages a(i, j), and the ρij messages are identical to the AP responsibility messages r(i, j).

3 Implementation

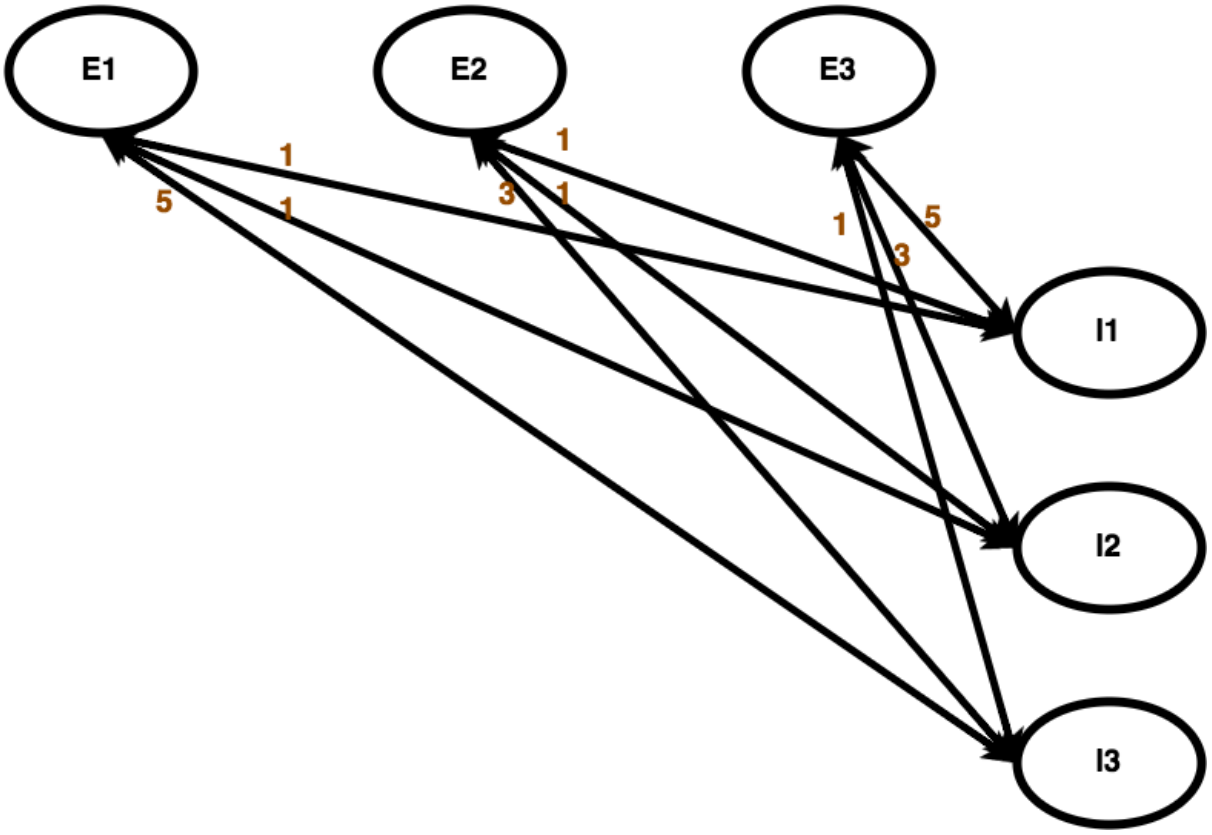
Implementation of this algorithm will be done with the vertex centric model using an aggregator to check the convergence of all vertices.

3.1 Input

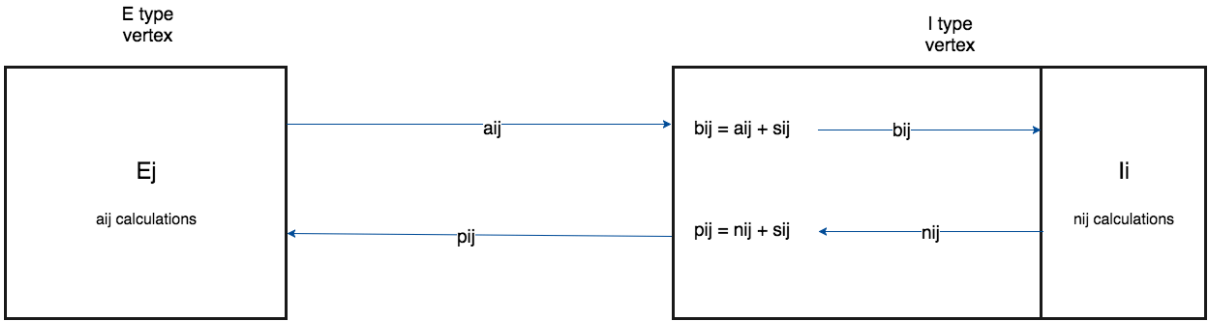
The input for the algorithm will be a graph where edges are similarities between points



This graph will be transformed to the Binary Affinity formated graph



3.2 Vertices



Implementation of this algorithm needs three types of vertices E, I and S. S vertices functionality is containing and sending similarities and this can be done in one of the two vertices, this implementation will use I vertex to do it. This way I nodes will calculate ρij (s(i, j)+ηij) and βij (s(i, j)+αij) messages simulating S vertices and avoiding the need of having them in the implementation.
I vertices need to contain similarities with all the points.

~~**3.2 Scatter-Gather**~~

~~**Gather**~~:

- ~~E nodes compute: a_ij~~
- ~~I nodes compute: p_ij = n_ij + s(i,j) (and it also calculates b_ij = a_ij +s(i,j) to be used as incoming messages)~~

~~Doing calculations this way S nodes are not needed. This is the implementation I did with giraph, maybe the implementation will be simpler if E nodes calculate b_ij = a_ij + s(i,j).~~

~~**Scatter**~~:

- ~~I and E nodes send their messages~~

3.3 Inputs

Inputs for the algorithm will be:

- Points similarity matrix
- Max iterations: maximum number of iterations if algorithm does not converge
- Damping factor applied to each message for the convergence

3.3 Convergence

From the original paper: *The message-passing procedure may be terminated after a fixed number of iterations, after changes in the messages fall below a threshold, or after the local decisions stay constant for some number of iterations*. We could say that once any of the messages do not change the algorithm has converged. One of the issues we have to take into account is that even

3.3 Example

Example spreadsheet

References

[1] Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. Science, 315:972–976, 2007.
[2] Inmar E Givoni and Brendan J Frey. A binary variable model for affinity propagation. Neural computation, 21(6):1589–1600, 2009.