Extended Affinity Propagation Clustering for Multi-source Entity Resolution

Stefan Lerm[1], Alieh Saeedi[2], Erhard Rahm[3]

Abstract: Entity resolution is the data integration task of identifying matching entities (e.g. products, customers) in one or several data sources. Previous approaches for matching and clustering entities between multiple (>2) sources either treated the different sources as a single source or assumed that the individual sources are duplicate-free, so that only matches between sources have to be found. In this work we propose and evaluate a general Multi-Source Clean Dirty (MSCD) scheme with an arbitrary combination of clean (duplicate-free) and dirty sources. For this purpose, we extend a constraint-based clustering algorithm called Affinity Propagation (AP) for entity clustering with clean and dirty sources (MSCD-AP). We also consider a hierarchical version of it for improved scalability. Our evaluation considers a full range of datasets containing 0% to 100% of clean sources. We compare our proposed algorithms with other clustering schemes in terms of both match quality and runtime. The proposed algorithms outperform previous methods and achieve an excellent precision in MSCD scenarios.

Keywords: Entity Resolution; Clustering; Affinity Propagation; MSCD-AP

1 Introduction

Entity Resolution (ER), also referred to as record linkage or deduplication, is a main data integration task. It is used to identify entities, such as specific costumer or product descriptions, in one or several data sources that refer to the same real-world entity. Most previous ER approaches focus on finding such matches in either a single source or between two sources. Multi-source ER aims at finding matching entities in an arbitrary number of sources which is more challenging than dealing with 1-2 sources since not only the degree of heterogeneity but also the variance in data quality generally increases with the number of sources.

There are two main phases for multi-source ER [Ra16, Sa18, Ch19]. First, similar pairs of entities are determined over all sources as match candidates. These can be recorded in a similarity graph where each vertex represents an entity and each edge a match relationship between two entities. Edges may have a similarity score reflecting the match probability. In the second phase, the matches are determined by a clustering algorithm on the similarity graph. All matching entities from any source referring to the same real-world entity are

[1] University of Leipzig & ScaDS.AI Dresden/Leipzig, s.lerm@studserv.uni-leipzig.de
[2] University of Leipzig & ScaDS.AI Dresden/Leipzig, saeedi@informatik.uni-leipzig.de
[3] University of Leipzig & ScaDS.AI Dresden/Leipzig, rahm@informatik.uni-leipzig.de

grouped in one cluster. There are many possible approaches for this entity clustering, especially the ones that have been proposed for clustering matches in a single source [Ha09, SPR17]. In the special case of duplicate-free (clean) sources each cluster contains at most one entity per source so that the cluster size is limited by the number of sources. Cluster algorithms that utilize this restriction have been shown to achieve better match quality than the more general approaches [NGR16, SPR18].

In this paper, we investigate a Multi-Source Clean Dirty (MSCD) entity clustering approach that can utilize clean sources but can also deal with dirty sources so that only a fraction (possibly 0%) of the sources have to be clean. The goal is to achieve better match quality than with a general clustering scheme when there are clean sources while avoiding the limitation of requiring that all sources have to be clean. While one could first deduplicate dirty sources and then apply a clustering for clean sources, the effort to determine these source-specific deduplication approaches is immense and perhaps not completely successful. We experimented with such an approach for a data integration challenge [OSR19] but it performed worse than matching dirty sources. Consequently, it is more flexible to support a mix of both dirty and clean sources. For this purpose, we propose extensions to the Affinity Propagation (AP) clustering approach [FD07] that converts the problem of clustering into a constraint optimization problem. Our extension MSCD-AP adds a new constraint to AP to deal with clean sources. We also consider a hierarchical variation of MSCD-AP for improved scalability, provide parallel implementations based on Apache Flink and comparatively evaluate the new approaches.

We make the following contributions:

- We are the first to consider a mix of clean and dirty sources for multi-source entity resolution and propose an extended version of affinity propagation clustering, MSCD-AP, for this purpose.
- For improved scalability, we propose a hierarchical variation, MSCD-HAP, and provide parallel implementations for the clustering schemes based on Apache Flink.
- We perform a comprehensive evaluation of the match quality, runtimes and scalability of the new approaches for different datasets and compare them with previous clustering schemes.

After a discussion of related work, we give a brief summary of the standard AP algorithm in Section 3. Section 4 presents the new clustering method MSCD-AP in detail while Section 5 describes the scalable approach MSCD-HAP. In Section 6 we present our evaluation.

2 Related Work

Entity resolution has been the subject of a large amount of research as can be seen from many surveys and books such as [Ch12, Ch19, KR10, GM12, Pa19]. For larger datasets it is imperative to apply blocking techniques to reduce the number of comparisons of entity pairs.

There are also many ways to determine match candidates, e.g. match rules requiring that a combined similarity of selected attributes exceed some threshold or supervised approaches using training data with both matching and non-matching pairs of entities to determine a match classifier.

This paper focuses on the final step of the ER pipeline, entity clustering on a similarity graph, to group together all matches of a real-world object. Clustering can improve match quality over the binary links in the similarity graph as it is possible to transitively infer additional links or to eliminate links that are unlikely to be correct. There are numerous approaches for clustering and also for entity clustering. Most previous entity clustering approaches focus on finding matches in a single (dirty) source. Example approaches include Connected Components, Center and Merge-Center clustering [HM09], Affinity Propagation [FD07], Ricochet clustering [WB09], Markov clustering [VD00] and Correlation clustering [BBC04]. [Ha09] comparatively evaluated many of these algorithms for a single source.

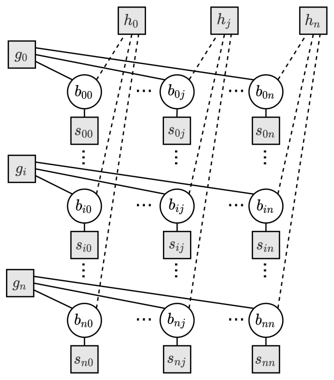
In [SPR17] we have shown that these approaches can be adapted for multi-source entity clustering and we comparatively evaluated several approaches for such a setting. We further developed new multi-source entity clustering approaches such as CLIP [SPR18], that work for clean (duplicate-free) data sources and can outperform the more general approaches for dirty sources. In our evaluation we will compare the new MSCD entity clustering approaches based on affinity propagation with these previous methods for dirty and clean sources. The previous clustering approaches including CLIP have been integrated into the FAMER[4] framework [Sa18] for multi-source entity resolution, that is used for our comparative evaluation. All match and clustering approaches in FAMER are implemented on top of Apache Flink to achieve a parallel entity resolution on a cluster of machines in order to reduce runtimes and improve scalability to larger datasets.

3 Affinity Propagation Clustering

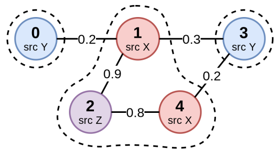
The Affinity Propagation clustering algorithm [FD07] groups entities by identifying exemplars. An exemplar is the entity that best represents all the entities of a cluster. The non-exemplar entities are assigned to the most appropriate exemplar. The goal of AP is to find exemplars and cluster assignments in a way that the sum of similarities inside clusters are maximized.

In [GF09], AP is solved by the iterative max-sum algorithm on a factor graph. The factor graph is a bipartite graph between the exemplar assignments (variable nodes) and factor nodes representing two constraints, called the g- and h-constraints. Figure 1a illustrates such a factor graph for AP. Variable nodes and factor nodes are represented as circles and rectangles respectively. For clustering n entities, the factor graph is represented by a n^2 binary matrix $\mathbb{B}$. The variable b_{ij} has the value 1 if the datapoint (entity) j is the exemplar

[4] https://dbs.uni-leipzig.de/research/projects/object_matching/famer



(a)



(b)

	0	1	2	3	4
0	1	0	0	0	0
1	0	0	1	0	0
2	0	0	1	0	0
3	0	0	0	1	0
4	0	0	1	0	0

(c)

s_{ij}	**0**	**1**	**2**
0	0,8	**0,9**	0,1
1	**0,9**	0,8	0,1
2	0,1	0,1	0,8

1	0	0
1	0	0
0	0	1

0	1	0
0	1	0
0	0	1

(d)

Fig. 1: a) Factor graph of AP [AKK19] b) AP clustering example c) Binary matrix d) Oscillation

of i. The factor nodes g_i and h_j assure a valid clustering by applying the constraints. The g-constraint enforces that a datapoint has to have exactly one exemplar. It means in each row of the binary matrix there must be exactly one variable with value 1. The h-constraint assures that a datapoint selects itself as its exemplar, if it is already chosen as exemplar by at least one other datapoint. It means, if there exists at least one 1 in a column of the binary matrix, then the diagonal element b_{jj} of that column must be set to 1 too. The cluster assignments are based on the similarities between entities so that similarity values are also represented as factor nodes (factor node s_{ij} provides the similarity information between the entities i and j).

Figure 1b illustrates an example clustering of AP where five entities 0-4 from three (differently colored) sources X, Y and Z are grouped in three clusters. The corresponding output binary matrix in Figure 1c shows that entities 0, 2 and 3 are the exemplars of the three clusters. As described above, the rows of the binary matrix illustrate the exemplar (cluster) assignment while the columns depict the clusters. The group of 1 values in column j represents the entities of the cluster with exemplar j.

AP aims at finding a cluster assignment maximizing the sum of similarities within clusters. This optimization problem can be formulated with the energy function [AKK19] shown in Equation (1). Maximizing the function requires to find an optimal configuration of the variables in $\mathbb{B}$ so that the sum of the similarities between entities and their exemplars is maximized and the two constraints are met. An exact maximization of the energy function is computationally intractable because a special case of this maximization problem is the NP-hard k-median problem [FD07].

$$E(\mathbb{B}) = \sum_{ij} s_{ij} b_{ij} + \sum_{i} g_i(\mathbb{B}(i,:)) + \sum_{j} h_j(\mathbb{B}(:,j)) \tag{1}$$

with

$$g_i(\mathbb{B}(i,:)) = \begin{cases} 0 & \text{if } \sum_j b_{ij} = 1 \\ -\infty & \text{otherwise} \end{cases} \qquad h_j(\mathbb{B}(:,j)) = \begin{cases} 0 & \text{if } b_{jj} = \max_i b_{ij} \\ -\infty & \text{otherwise} \end{cases}$$

The proposed iterative max-sum algorithm uses several parameters that affect the clustering result and that deal with the problem of non-convergence. The most important parameter is called *preference*. It defines the self-similarity s_{ii} of an entity i. The higher the preference value is chosen the more likely the entity becomes an exemplar. Parameters to deal with non-convergence are the noise level and the damping factor λ. AP suffers from oscillation between solutions that are similarly well suited for optimizing the energy function. For the similarity matrix in the top portion of Figure 1d, the symmetrical similarity values between entities 0 and 1 make both equally well suited as an exemplar. In such a situation, AP does not converge and oscillates between the two solutions with either entity 0 or 1 as the exemplar as shown in the bottom part of Figure 1d. Oscillation is avoided by adding a tiny amount of noise to the similarity values. The damping factor has a similar goal and is related to the used message passing implementation for the iterative computation. It leads to an adaptation of values exchanged between iterations. If oscillations nevertheless occur, the preference or the damping factor must be adapted (see next section).

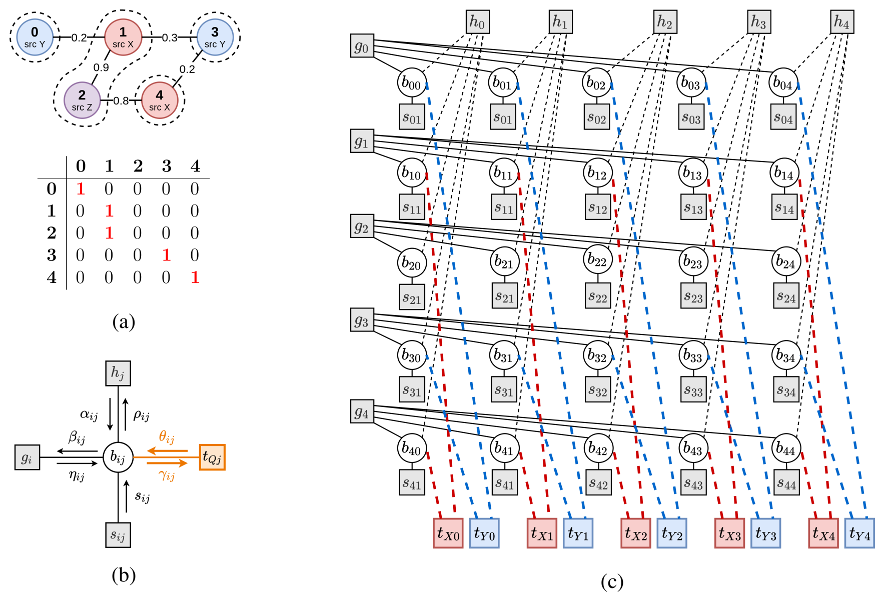
4 MSCD Affinity Propagation

To cluster mixed datasets of clean and dirty sources, we propose an extension to AP called MSCD-AP. Since clean sources have no duplicates, every cluster should have at most one entity of a clean source. This is now controlled by an additional *clean-source consistency* constraint. It means that in each column of the binary assignment matrix $\mathbb{B}$, value 1 is allowed for at most one (row) entity of a clean source.

Figure 2a shows a possible clustering of MSCD-AP for the running example when sources X and Y are clean. There are four *source-consistent* clusters with at most one entity per clean source. In the corresponding binary matrix, each column has at most one entity with value 1 per clean source. For example, the column (cluster) for exemplar entity 1 has two associated entities (1 and 2) from different sources.

Our proposed clean-source consistency constraint is expressed in Equation (2). It uses function t to add a large penalty to the extended energy function in Equation (3) when the constraint is violated. The constraint requires that for a column j the value 1 is allowed for at most one datapoint from a clean source Q.

$$t_{Qj}(\mathbb{B}(i \in Q, j) = \begin{cases} 0 & \text{if } \sum_{i \in Q} b_{ij} \leq 1 \\ -\infty & \text{otherwise} \end{cases} \tag{2}$$



	0	1	2	3	4
0	1	0	0	0	0
1	0	1	0	0	0
2	0	1	0	0	0
3	0	0	0	1	0
4	0	0	0	0	1

Fig. 2: a) MSCD-AP clustering example b) Messages of the MSCD-AP factor graph
c) The factor graph for MSCD-AP for the running example. The sources X and Y are clean.

$$E(\mathbb{B}) = \sum_{ij} s_{ij} b_{ij} + \sum_{i} g_i(\mathbb{B}(i,:)) + \sum_{j} h_j(\mathbb{B}(:,j)) + \sum_{Q} \sum_{i \in Q, j} t_{Qj}(\mathbb{B}(i,j)) \tag{3}$$

Figure 2c illustrates the extension of the AP factor graph to cluster our running example data. For clean sources X and Y, additional factor nodes t_x and t_y (marked in red and blue) are added to each column of the binary matrix. The factor node t_{xj} assures the clean-source consistency constraint for source X and column j. It is connected to the variable node b_{ij} only if entity i is from data source X. The clean-source constraint may get in conflict with the h-constraint of AP. The h-constraint enforces a datapoint to choose itself as its own exemplar, if it is selected by at least one other datapoint. So the diagonal element b_{jj} of column j is enforced to be 1, if there is any other 1 in that column. On the other hand, the clean-source constraint enforces b_{jj} to be 0, if another datapoint of the same clean source selected it as its exemplar. So the two constraints enforce different values for b_{jj} and thus the algorithm may struggle to converge. This situation is simply avoided in our implementation by not having links between entities of the same clean source which is a default feature of the linking component of FAMER.

For the traditional AP clustering, the max-sum optimization has been implemented by a message passing algorithm [GF09]. The messages are exchanged between factor and variable nodes of the factor graph to reflect the mutual dependencies within an iterative process. The messages are computed differently depending on whether the recipient node is a variable node or a factor node. Figure 2b shows the messages exchanged between the nodes of the new factor graph of MSCD-AP. The grey-colored factor nodes enforce the g and h constraints while the new factor node t (marked in orange) applies the clean-source consistency constraint via the θ and γ messages.

We build on the formulae from [Gi12] to update messages for the original constraints and specify the new message formulas for our MSCD extension. In the max-sum algorithm, outgoing messages of a variable node summarize all incoming messages to that node, except of the node to which the new message will be sent. Due to the new constraint, all outgoing messages from variable nodes to factor nodes are now modified because the new factor nodes t_{Qj} are additional neighbours of b_{ij}. As sum of the incoming messages from the neighbouring nodes, except of the recipient, the modified messages β and ρ as well as the new message γ are easily deduced as listed in Equation (4) - (6).

The message formulas from factor nodes to variable nodes do not change in AP when a new factor node is added. Therefore the incoming messages of α (eq. (7)) and η (eq. (8)) remain unchanged compared to AP. The new incoming message θ from the new factor node t_{Qj} is expressed in Equation (9). The more complex derivation of message θ from the max-sum algorithm is given in the appendix. The variable assignments that maximize the energy function are calculated by Equation (10).

$$\beta_{ij} = s_{ij} + \alpha_{ij} + \theta_{ij} \quad (4) \qquad \rho_{ij} = s_{ij} + \eta_{ij} + \theta_{ij} \quad (5) \qquad \gamma_{ij} = s_{ij} + \alpha_{ij} + \eta_{ij} \quad (6)$$

$$\alpha_{ij} = \begin{cases} \sum_{k \neq j} \max(0, \rho_{kj}) & i = j \\ \min[0, \rho_{jj} + \sum_{k \neq \{i,j\}} \max(0, \rho_{kj})] & i \neq j \end{cases} \quad (7)$$

$$\eta_{ij} = -\max_{k \neq j} \beta_{ik} \quad (8) \qquad \theta_{ij} = min(0, -\max_{k \neq i}[\gamma_{kj}]) \quad (9)$$

$$b_{ij} = \begin{cases} 1 & \alpha_{ij} + \rho_{ij} > 0 \\ 0 & \alpha_{ij} + \rho_{ij} \leq 0 \end{cases} \quad (10)$$

Algorithm 1 lists the pseudo code of MSCD-AP with focus on the parameter adaptation. There are several inputs for the algorithm. The clustering problem is defined by the similarity matrix S and the specification of the clean sources ($srcInfo$). λ denotes the damping factor. The preference can be set separately for dirty (p_{dirty}) and clean (p_{clean}) sources. Random gaussian noise is added to the similarity values at a decimal position specified by the $noiseLevel$. Parameter adaption for the preference values and the damping factor is done stepwise by $step_{pref}$ and $step_{dmp}$. The adaptation steps are real values in (0,1] that are used to increase the original values towards the maximum 1 or decrease them towards 0. As algorithm output the binary matrix $\mathbb{B}$ describes the exemplar assignment of every entity.

Algorithm 1: MSCD-AP

```
Input: $S$, $srcInfo$, $\lambda$, $p_{dirty}$, $p_{clean}$, $noiseLevel$, $step_{pref}$, $step_{dmp}$
Result: $\mathbb{B}$ with exemplar assignments
1  repeat
2      initializeMessages();
3      initializeB();
4      modifyS($p_{dirty}$, $p_{clean}$, $noiseLevel$, $srcInfo$);
5      for $iteration = 0 : max$ do
6          updateMessages($\lambda$);
7          updateB();
8          if isConverged() then break;
9      $solutionFound \leftarrow$ isSolutionFound($\mathbb{B}$);
10     if $\neg solutionFound$ then adaptParameters($step_{pref}$, $step_{dmp}$);
11 until $solutionFound$;
```

After the initialization of the messages and output matrix (line 2 and 3) the diagonal elements s_{jj} of the similarity matrix are set to the defined preference values and noise is added to all similarity values in line 4. The iterative message passing starts in line 5. In each iteration, the messages are updated in line 6 according to Equation (4) - (8). Additionally, α and ρ messages are damped in order to prevent oscillations. Finally in line 7, the binary matrix values are updated according to Equation (10). If no changes are observed in the binary matrix after a specific number of iterations, the algorithm converges and is ended (line 8). Otherwise it ends after a maximal number of iterations. If the algorithm stops but the solution is not found yet (line 9 and 10), then it has to be restarted with adapted parameters. For this purpose, function `adaptParameters` initially decreases the preference values by *preference adaption step* ($step_{pref}$) until the minimum value 0. If convergence is still not reached, the preference values are then increased step by step until the maximum 1 is reached. In case of no success, the preference values are reset to their original values and the damping factor λ is now increased by *damping adaption step* ($step_{dmp}$). This process continues until the algorithm finds a valid solution.

5 Scalable MSCD Affinity Propagation

Clustering large datasets is a challenge for AP since its time and memory complexity grows quadratically with the number of entities and thus the data volume[5]. Liu et al. [Li13] proposed Hierarchical Affinity Propagation (HAP) to make AP suitable for clustering large-scale datasets. Following a divide and conquer strategy, HAP clusters the dataset by executing AP several times on different hierarchy levels.

[5] In the case of a sparse similarity matrix, the time complexity reduces to $Nklog(N)$ with k being the average connectivity of the similarity matrix [Zh10].

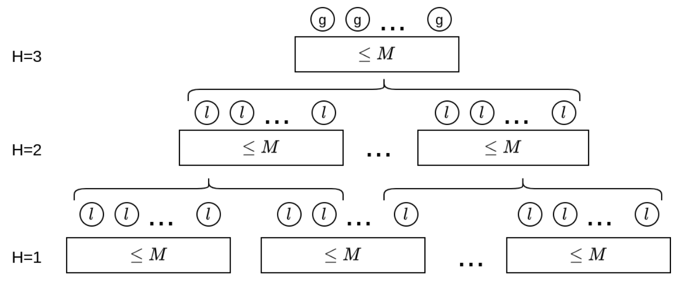


Fig. 3: HAP for three hierarchy levels H. Circles illustrate local (l) and global (g) exemplars, rectangles represent partitions.

Figure 3 illustrates the hierarchical clustering for three levels. In the first (lowest) hierarchy level, the dataset is randomly divided into equal-sized partitions of maximal size M. Then AP is executed on each partition, resulting into a set of so called *local exemplars* for each partition. In the next hierarchy level, the exemplars of the previous level are merged and again partitioned. This process is repeated until the input size of a hierarchy level is lower or equal to M. The execution of AP on the top hierarchy level determines the *global exemplars* for the dataset. All non-exemplar entities are assigned to the global exemplar with the highest similarity. Thus AP is executed once for each partition of each hierarchy level with a complexity of $O(M^2)$.

Unfortunately, applying the hierarchical algorithm for MSCD-AP does not guarantee the clean-source consistency. This is because, the clustering of local exemplars by MSCD-AP on intermediate hierarchy levels violates the clean-source consistency when two local exemplars from a previous level are clustered together although they have associated entities from the same clean source. A naive solution is to extend each local exemplar with the source information of the entities assigned to it in the previous hierarchy level. This could be used in subsequent cluster decisions to avoid that more than one entity of a clean source is assigned to an exemplar. This approach, however, can lead to poor clustering results. A bad decision in a lower level of the hierarchy, where an entity of a clean source with a low similarity is assigned to a local exemplar, can prevent that a much more similar entity from the respective source is merged at a higher level resulting in poor cluster decisions.

A more promising solution is to assign entities to global exemplars separately for clean and dirty sources. Initially, HAP is executed using MSCD-AP to determine local and global exemplars on the partitions. As in HAP, dirty source entities are then assigned to the exemplars with the highest similarity. By contrast, clean source entities are assigned using the Hungarian algorithm [Ku55, Mu57]. Given the similarities between these entities and exemplars, the Hungarian algorithm finds a 1:1 assignment between entities of a clean source and exemplars (i.e., each exemplar is assigned to at most one entity of a clean source) so that the overall similarity of all assignments is maximized. If the number of entities from a clean source exceeds the number of exemplars, the excess points form singleton

Tab. 1: Overview of evaluation datasets

	General information					**Perfect result**	
	domain	entity properties	#entity	#src	type	#clusters	#links
DS-G	geography	label, longitude, latitude	3,054	4	MSC	820	4,391
DS-M	music	artist, title, album, year, length	19,375	5	MSC	10,000	16,250
DS-P1	persons	name, surname, suburb, postcode	5,000,000	5	MSC	3,500,840	3,331,384
DS-P2			10,000,000	10	MSC	6,625,848	14,995,973
DS-C	camera	heterogenous key-value pairs	21,023	23	MSCD	3,910	368,546

clusters. When a global exemplar is from a clean source, the clean-source consistency is also enforced since there is no similarity link between entities of the same clean source.

The Hungarian algorithm has a computational complexity of $O(mk^2)$ for a $m \times k$ cost matrix [Cu16] with k global exemplars and m entities from one clean source. The complexity is higher compared to AP, but the bipartite matching is executed on small subsets of the n-sized dataset ($m, k \ll n$). Thus the combination of HAP with MSCD-AP and the Hungarian algorithm is still more suitable for large datasets than MSCD-AP. We call this combination MSCD-HAP and comparatively evaluate it in the next section.

6 Evaluation

We now evaluate the cluster effectiveness and efficiency of the proposed MSCD extensions of AP in comparison to standard AP and previous clustering schemes. We first describe the used datasets from four domains. We then analyze comparatively the effectiveness of the proposed algorithm. Finally, we evaluate runtime performance and scalability.

6.1 Datasets and Configuration Setup

We evaluate the new approaches with four multi-source datasets of clean sources (MSC) that have also been used in previous studies [SPR17, SPR18, Sa18]. Table 1 gives an overview of the datasets from three domains (geography, music, persons) including available properties and number of entities. For the evaluation of mixed datasets of clean and dirty sources, we use the dataset of the ACM SIGMOD 2020 Programming Contest[6]. It contains approximately 30k product specifications from 24 dirty sources. For our purposes, we determine a subset called DS-C focussing on camera products (Table 1). We excluded the source *www.alibaba.com* because it contains just a few cameras but many non-cameras. Table 2 lists the 23 remaining sources and their number of entities with and without duplicates. The matching result of the SIGMOD contest winner [Bl20] is considered as the ground truth. It achieved f-measure of 99% by extensive domain-specific preprocessing and

[6] `http://www.inf.uniroma3.it/db/sigmod2020contest/index.html`

Tab. 2: Overview of camera dataset (DS-C)

Source name	**ID**	**#entity**	**#entity dedup.**
buy.net	1	358	244
cammarkt.com	2	198	94
www.buzzillions.com	3	832	630
www.cambuy.com.au	4	118	56
www.camerafarm.com.au	5	120	59
www.canon-europe.com	6	164	163
www.ebay.com	7	14,009	3,255
www.eglobalcentral.co.uk	8	190	75
www.flipkart.com	9	118	47
www.garricks.com.au	10	130	69
www.gosale.com	11	895	578
www.henrys.com	12	181	137
www.ilgs.net	13	102	64
www.mypriceindia.com	14	347	279
www.pcconnection.com	15	211	126
www.price-hunt.com	16	327	282
www.pricedekho.com	17	366	325
www.priceme.co.nz	18	740	475
www.shopbot.com.au	19	516	334
www.shopmania.in	20	630	556
www.ukdigitalcameras.co.uk	21	129	73
www.walmart.com	22	195	115
www.wexphotographic.com	23	147	87
	sum	21,023	8,123

Tab. 3: MSCD datasets

Name	**%cln**[1]	**cln**[2]	**#cln**[3]	**#dirt**[4]
DS-C0	0		0	21,023
DS-C26	26	1-6, 8-23	4,868	14,009
DS-C32	32	7	3,255	7,014
DS-C50	50	7, 18, 19,20, 22, 23	4,822	4,786
DS-C62A	62	1, 4, 6 7, 9, 11, 13, 15, 17, 19, 20	5,748	3,536
DS-C62B	62	2, 3, 5, 7, 8, 10, 12, 14, 16, 18, 21-23	5,630	3,478
DS-C80	80	1-12 14-18	6,894	1,719
DS-C100	100	1-23	8,123	0

[1] Percentage of entities from clean sources
[2] Clean source IDs
[3] Number of entities from clean sources
[4] Number of entities from dirty sources

matching camera entities against a prepared list of nearly all available cameras in the market. Our matching and clustering approaches are generic and applicable to different datasets. Our goal is not to achieve the best possible result but to enable a fair comparison of the clustering schemes based on reasonably good input similarity graphs for different datasets.

Using DS-C, we create eight datasets with different combinations of clean and dirty sources and thus different degrees of dirtiness. As shown in Table 3, we name the datasets according to the percentage of entities from clean sources, where DS-C0 and DS-C100 means that all entities are from dirty and clean sources, respectively. For the mixed cases, an important distinction is whether a clean or dirty version of source 7 (*www.ebay.com*) is considered because it is the largest source and contains many duplicates. In DS-C62A and DS-C62B, the clean form of source 7 is included, while all other sources that are clean in 62A are dirty in 62B and vice versa.

Tab. 4: Linking configurations of clean multi-source datasets

	Blocking Key	**Similarity Function**
DS-G	prefixLength1 (label)	Jaro-Winkler (label) & geographical distance
DS-M	prefixLength1 (album)	Trigram (title)
DS-P1/P2	prefixLength3 (surname) + prefixLength3 (name)	avg (Trigram (name) + Trigram (surname) + Trigram (postcode) + Trigram (suburb))

The blocking and matching configurations for the clean datasets are listed in Table 4 and correspond to the ones in previous studies [SPR18, Sa18]. For the camera dataset, we extracted the manufacturer name, a list of model names, manufacturer part number (mpn), european article number (ean), digital and optical zoom, camera dimensions, weight, product code, sensor type, price and resolution from the heterogeneous product specifications. In order to reduce the number of comparisons, standard blocking with a combined key of manufacturer name and model number is applied. Within these blocks, all pairs with exactly the same model name, mpn or ean are classified as matches. We assign a similarity value to the matched pairs determined from a weighted average of the character-3Gram Dice similarity of string values and a numerical similarity of numerical values (within a maximal distance of 30%).

6.2 ER Quality of Clustering Algorithms

To evaluate the quality of the clustering results, we use the standard metrics precision, recall and their harmonic mean, f-measure w.r.t. the links of the perfect cluster results (last column of table 1). We compare the quality of AP and the proposed MSCD-AP approaches with seven previous clustering schemes comprised in FAMER [Sa18]. The CLIP approach is tailored to clean sources. The other six algorithms are general approaches for dirty sources (connected components, correlation clustering CCPivot, two variants of star clustering and two variants of center clustering). We also provide the quality of the input similarity graph (without clustering) in our figures. For AP and MSCD-AP we manually determined suitable parameter configurations. We use the interval [0.01, 0.7] for preference values and set a higher preference value for clean sources than for dirty sources to choose exemplars preferably from clean sources. The damping factor is set to 0.5 and noise is added to the similarity values from the third decimal place. For the smaller datasets DS-G, DS-M and DS-C, we used a partition size of 1000 while for the person datasets we apply MSCD-HAP with partition size 100 to reduce runtimes. When the size of a connected component is smaller than the partition size, MSCD-AP is executed. Higher similarity thresholds result in fewer links and smaller components that are mostly executed without partitioning. Because DS-G and DS-C consist of small components, partitioning is not used. Thus AP and MSCD-AP are executed. In DS-P the hierarchical algorithms are used. For reasons of space the results of DS-P1 are omitted, as they are very similar to those of DS-P2.

We first analyze cluster quality for the datasets with only clean sources. Figure 4 shows the results for the three MSC datasets for different similarity thresholds to generate the input similarity graph. As expected, the f-measure results are best for the CLIP approach tailored to ER for clean sources. However, the proposed MSCD-AP approach achieves about the same quality for two datasets (DS-G, DS-P2) and performs better than the six general clustering schemes for DS-M. It also outperforms AP in all cases. These surprisingly good results are mainly due to an excellent precision of MSCD-AP which can outweigh its comparatively low recall. The recall is limited since AP and MSCD-AP strongly depend

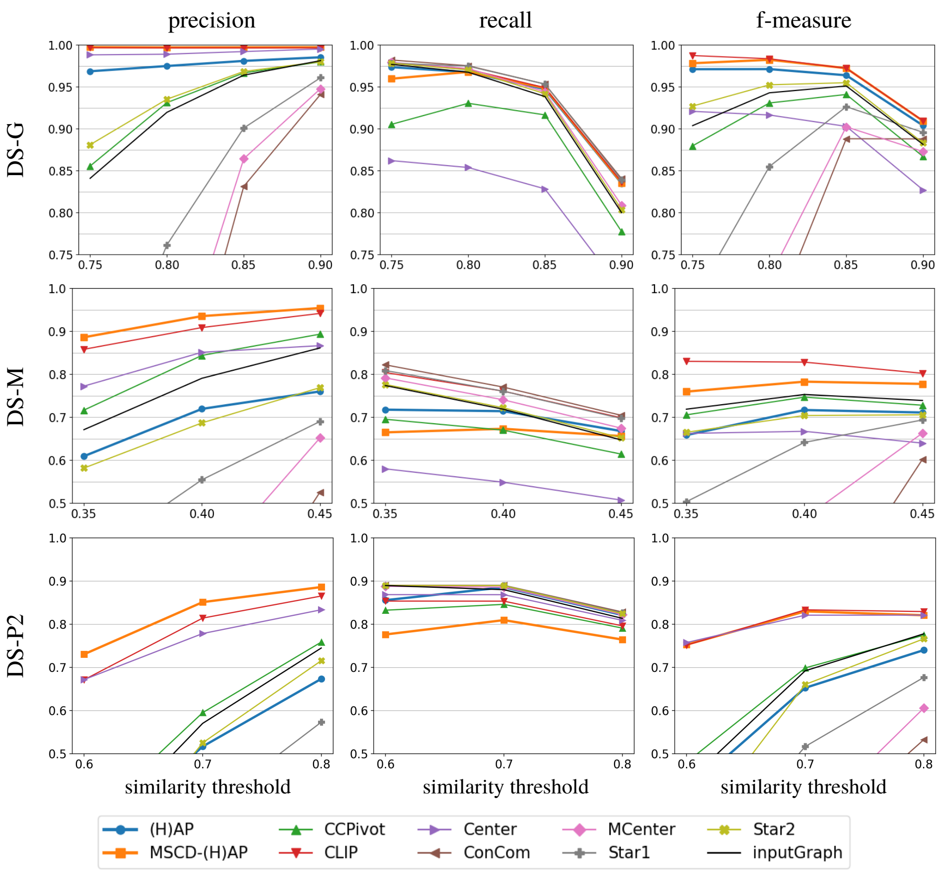


Fig. 4: Clustering quality for the multi-source clean ER datasets

on the relative similarity values and can even consider a high similarity value such as 0.8 as low if it is below the average of the considered value range, e. g. [0.8, 1.0]. This leads to more small clusters and thus a lower recall compared to other algorithms. Due to the clean-source constraint, MSCD-AP creates more exemplars than AP and therefore obtains a lower recall compared to AP but a much better precision.

Figure 5 shows the quality of the clustering results for the camera datasets with different degrees of dirtiness. Due to space constraints we show results for 5 of the 8 cases but the results for the remaining datasets confirm the overall outcome. We observe that MSCD-AP achieves the best f-measure for all cases with a mix of dirty and clean sources. For the case of only clean sources (DS-C100) it is only outperformed by CLIP. For dirty sources only (DS-C0) MSCD-AP is identical to AP which is among the best approaches. As a result, MSCD-AP is the best or among the best approaches over all configurations while other

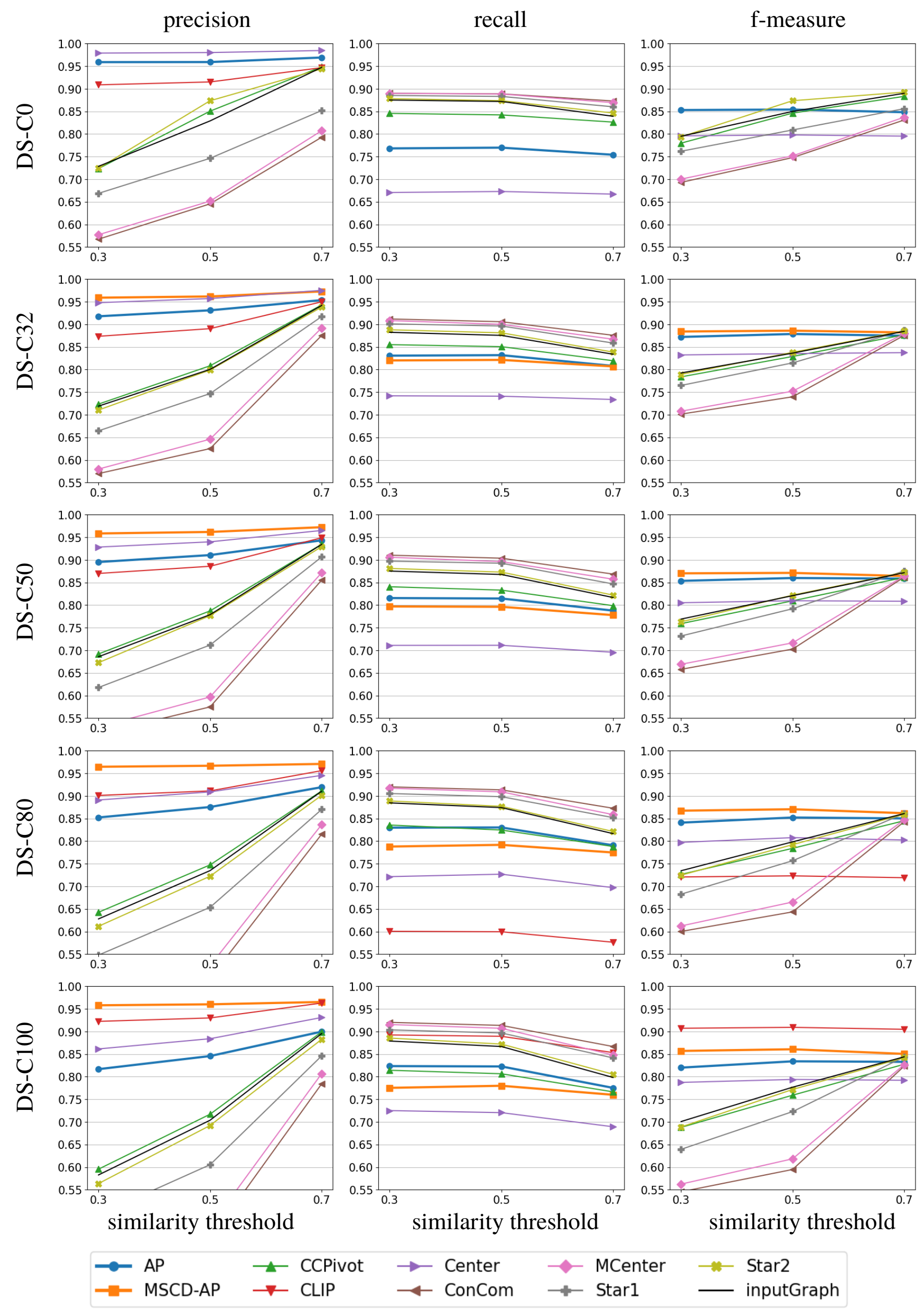


Fig. 5: Clustering quality for the multi-source clean/dirty ER datasets

schemes like CLIP are good in only one configuration. Another strong point of MSCD-AP is that its f-measure is nearly stable over all threshold values used to determine the input similarity graph while the general clustering schemes depend on finding a suitable threshold. As for the MSC datasets, the good results of MSCD-AP are mainly due to its excellent precision values in all cases that outweigh its lower recall results.

6.3 Runtimes and Speedups

We evaluate runtimes and speedup behavior for the larger datasets from the person domain. The speedup of MSCD-HAP is determined for the parallel execution with different numbers of workers. We also analyze the effect of different MSCD-HAP partition sizes on runtime as well as on clustering quality. The experiments are performed on a shared nothing cluster with 16 worker nodes. Each worker consists of an E5-2430 6(12) 2.5 Ghz CPU, 48 GB RAM, two 4 TB SATA disks and runs openSUSE 13.2. The nodes are connected via 1 Gigabit Ethernet. Our evaluation is based on Hadoop 2.6.0 and Flink 1.9.0. We run Apache Flink standalone with 6 threads and 40 GB memory per worker.

Figure 6 shows the runtime of each clustering approach for a parallel execution on 16 workers. As expected, the larger dataset DS-P2 leads to higher runtimes than for DS-P1 while higher similarity thresholds reduce runtimes due to the lower number of edges in the similarity graph. MSCD-HAP is slower than HAP because the calculations for the clean-source constraint and the exemplar assignment by the Hungarian algorithm need additional runtime. The clean-source constraint of MSCD-AP also leads to more exemplars and potentially more entities of the same source that are equally well suited to be an exemplar. Thus, oscillations occur more frequently for MSCD-AP compared to AP leading to more parameter adaptions to find a converging solution.

MSCD-HAP along with CCPivot and MergeCenter are among the slowest algorithms for the lowest threshold. Yet with higher similarity thresholds the runtime of MSCD-HAP

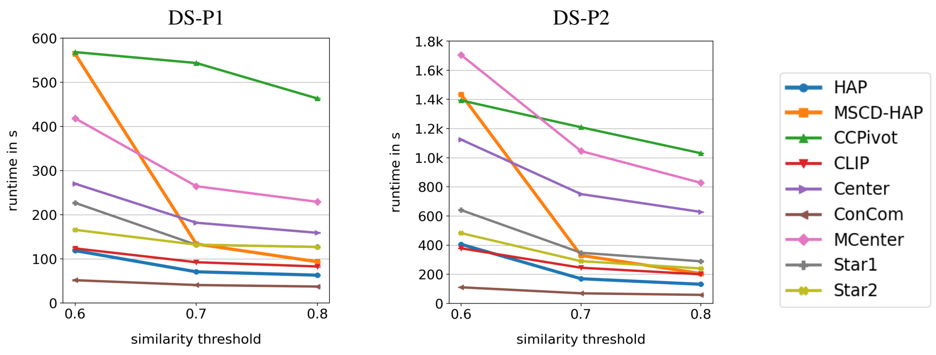


Fig. 6: Runtimes for clustering schemes (with partition size 100 for HAP and MSCD-HAP)

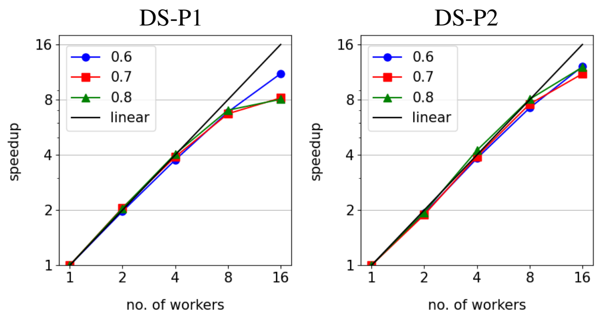


Fig. 7: Speedup of MSCD-HAP for different similarity thresholds

improves significantly making it one of the fastest algorithms. This is because a high minimum threshold avoids that a large number of entities are connected in the similarity graphs resulting in mostly small clusters and reduced work for the Hungarian algorithm. Moreover, oscillations occur less in such cases.

Figure 7 depicts the speedup of MSCD-HAP with partition size 100 for different similarity thresholds and for 1 to 16 worker machines. We observe that close to perfect speedup is achieved for the larger dataset DS-P2 and for a lower similarity threshold (bigger similarity graph) for the smaller DS-P1 dataset. For the higher thresholds the needed computations for DS-P1 cannot utilize 16 machines so that a good speedup is only achieved until 8 workers.

Figure 8 investigates the effect of partition size on both runtime and clustering quality.

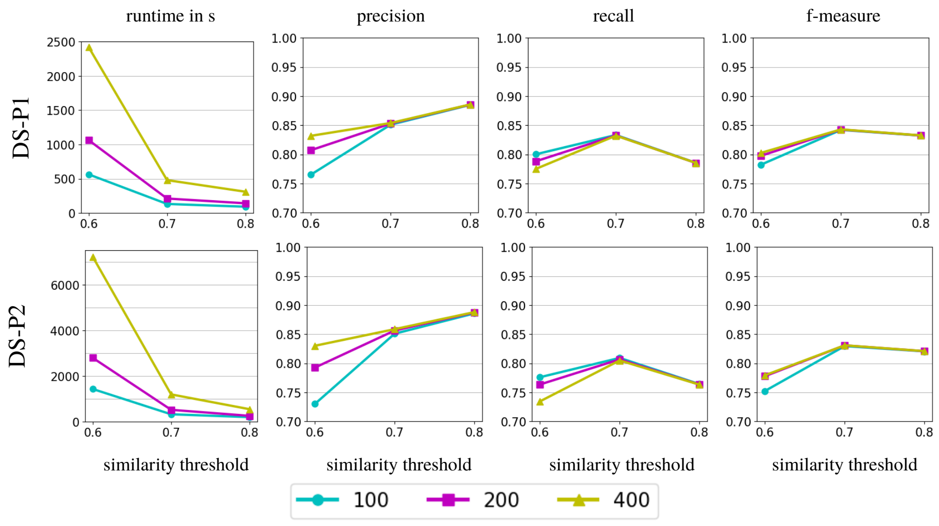


Fig. 8: Clustering quality and runtime for different partitions sizes of MSCD-HAP

We observe that larger partition sizes lead to much higher runtimes but also to improved clustering quality. These effects are most pronounced for smaller similarity threshold such as 0.6 that lead to bigger similarity graphs and thus to more computations. With larger partition sizes there are more entities and more similarity values in each partition. Therefore, the probability of finding good local and global exemplars rises and consequently the precision is improved. Yet recall drops slightly, because on bigger partitions more exemplars can be found and AP generally tends to form many small clusters. While the runtime is up to seven times higher for partition size 400 compared to 100 (for DS-P2) for threshold 0.6, these differences largely go away for higher thresholds and much smaller similarity graphs. This is also the case for clustering quality, where similarity value 0.7 or higher leads to about the same f-measure for all partition sizes.

7 Conclusions

We studied how to support multi-source entity clustering for a mix of clean (duplicate-free) and dirty data sources. The proposed extension of Affinity Propagation clustering, MSCD-AP, showed to be highly effective and perform better than previous methods for mixed configuration where a subset of the sources is duplicate-free. To improve runtimes we proposed the use of a hierarchical version MSCD-HAP and provide parallel implementations of the algorithms. The parallel implementations achieve good speedup values thereby supporting scalability to larger datasets. In future work, we will investigate how to extend additional clustering schemes for multi-source ER for mixed configurations with both clean and dirty sources.

8 Acknowledgements

This work is partially funded by the German Federal Ministry of Education and Research under grant BMBF 01IS18026B in project ScaDS.AI Dresden/Leipzig.

Bibliography

[AKK19] Amjad, R.; Khan, R.; Kleinsteuber, M.: Extended Affinity Propagation: Global Discovery and Local Insights. IEEE Transactions on Knowledge & Data Engineering, 2019.

[BBC04] Bansal, N.; Blum, A.; Chawla, S.: Correlation clustering. Machine learning, 56(1-3):89–113, 2004.

[Bl20] Blacher, M.; Klaus, J.; Mitterreiter, M.; Giesen, J.; Laue, S.: Fast Entity Resolution With Mock Labels and Sorted Integer Sets. CEUR Workshop Proceedings, 2726, 2020.

[Ch12] Christen, P.: Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection. Springer Science & Business Media, 2012.

[Ch19] Christophides, V.; Efthymiou, V.; Palpanas, T.; Papadakis, G.; Stefanidis, K.: End-to-End Entity Resolution for Big Data: A Survey. arXiv preprint arXiv:1905.06397, 2019.

[Cu16] Cui, H.; Zhang, J.; Cui, C.; Chen, Q.: Solving large-scale assignment problems by Kuhn-Munkres algorithm. 2nd International Conference on Advances in Mechanical Engineering and Industrial Informatics (AMEII), 2016.

[FD07] Frey, B. J.; Dueck, D.: Clustering by passing messages between data points. science, 315(5814):972–976, 2007.

[GF09] Givoni, I. E.; Frey, B. J.: A binary variable model for affinity propagation. Neural computation, 21(6):1589–1600, 2009.

[Gi12] Givoni, I. E.: Beyond affinity propagation: Message passing algorithms for clustering. Citeseer, 2012.

[GM12] Getoor, L.; Machanavajjhala, A.: Entity resolution: theory, practice & open challenges. Proceedings of the VLDB Endowment, 5(12):2018–2019, 2012.

[Ha09] Hassanzadeh, O.; Chiang, F.; Lee, H. C.; Miller, R. J.: Framework for evaluating clustering algorithms in duplicate detection. PVLDB, 2(1):1282–1293, 2009.

[HM09] Hassanzadeh, O.; Miller, R. J.: Creating probabilistic databases from duplicated data. The VLDB Journal, 18(5):1141, 2009.

[KFL01] Kschischang, F. R.; Frey, B. J.; Loeliger, H-A: Factor graphs and the sum-product algorithm. IEEE Transactions on information theory, 47(2):498–519, 2001.

[KR10] Köpcke, H.; Rahm, E.: Frameworks for entity matching: A comparison. Data & Knowledge Engineering, 69(2):197–210, 2010.

[Ku55] Kuhn, H. W.: The Hungarian method for the assignment problem. Naval research logistics quarterly, 2(1-2):83–97, 1955.

[Li13] Liu, X.; Yin, M.; Luo, J.; Chen, W.: An improved affinity propagation clustering algorithm for large-scale data sets. In: 2013 Ninth International Conference on Natural Computation (ICNC). IEEE, pp. 894–899, 2013.

[Mu57] Munkres, J.: Algorithms for the assignment and transportation problems. Journal of the society for industrial and applied mathematics, 5(1):32–38, 1957.

[NGR16] Nentwig, M.; Groß, A.; Rahm, E.: Holistic entity clustering for linked data. In: 2016 IEEE 16th Int. Conf. on Data Mining Workshops (ICDMW). IEEE, pp. 194–201, 2016.

[OSR19] Obraczka, D.; Saeedi, A.; Rahm, E.: Knowledge Graph Completion with FAMER. In: Proc. KDD workshop Data Integration for Knowledge Graphs (DI2KG). 2019.

[Pa19] Papadakis, G.; Skoutas, D.; Thanos, E.; Palpanas, T.: A survey of blocking and filtering techniques for entity resolution. CoRR, abs/1905.06167, 2019.

[Ra16] Rahm, E.: The case for holistic data integration. In: East European Conference on Advances in Databases and Information Systems. Springer, pp. 11–27, 2016.

[Sa18] Saeedi, A.; Nentwig, M.; Peukert, E.; Rahm, E.: Scalable matching and clustering of entities with FAMER. Complex Systems Informatics and Modeling Quarterly, pp. 61–83, 2018.

[SPR17] Saeedi, A.; Peukert, E.; Rahm, E.: Comparative evaluation of distributed clustering schemes for multi-source entity resolution. In: European Conference on Advances in Databases and Information Systems. Springer, pp. 278–293, 2017.

[SPR18] Saeedi, A.; Peukert, E.; Rahm, E.: Using link features for entity clustering in knowledge graphs. In: European Semantic Web Conference. Springer, pp. 576–592, 2018.

[VD00] Van Dongen, S. M.: Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht, 2000.

[WB09] Wijaya, D. T.; Bressan, S.: Ricochet: A family of unconstrained algorithms for graph clustering. In: International Conference on Database Systems for Advanced Applications. Springer, pp. 153–167, 2009.

[Zh10] Zhang, X.: Contributions to Large Scale Data Clustering and Streaming with Affinity Propagation. Application to Autonomic Grids. PARIS: University PARIS-SUD, 2010.

Appendix: Derivation of Equation (9)

In MSCD-AP, θ is a message from a factor node to a variable node and therefore is derived from Equation (11) of the max-sum algorithm, following [KFL01] and [Gi12].

$$\mu_{f \to x}(x) = \max_{n(f)\backslash\{x\}} \left(lnf(x, y_1, ..., y_m) + \sum_{y_i \in n(f)\backslash\{x\}} \mu_{y_i \to f}(y_i) \right) \tag{11}$$

The binary variable b_{ij} either obtains value 1 or 0. Firstly, we investigate both cases by considering *all possible configurations* of all neighboring variable nodes $b_{kj}(k \neq i)$ of t_{Qj} and then according to Equation (12) [Gi12], we combine them in order to to get a scalar value for the θ message.

$$\mu_{ij} = \mu_{ij}(1) - \mu_{ij}(0) \tag{12}$$

For $b_{ij} = 1$: Equation (13) shows θ for the case that i chooses j as its exemplar. All neighbors of t_{Qj} are from the same clean source Q. Let q be the number of entities in Q. All incoming messages $\mu_{b_{kj} \to t_{Qj}}(b_{kj})$ of t_{Qj} are defined as $\gamma_{kj}(b_{kj})$. For not hurting the *clean source constraint*, no other datapoint in Q is allowed to choose j as its exemplar. Therefore all other neighboring variable nodes $b_{kj}(k \neq i)$ of t_{Qj} are set to 0. This is the only configuration that satisfies the clean source constraint and thus the optimal one. Acording to Equation (2), the t_{Qj} function evaluates its maximum value of 0.

$$\theta_{ij}(1) = \max_{b_{kj},k\neq i}[ln\, t_{Qj}(b_{1j} = 0, ..., b_{ij} = 1, ..., b_{qj} = 0) + \sum_{b_{kj},k\neq i} \gamma_{kj}(b_{kj} = 0)] \\ = \sum_{k\neq i} \gamma_{kj}(0) \tag{13}$$

For $b_{ij} = 0$: There is more flexibility for finding the optimal solution if datapoint i does not choose j as its exemplar. In order to guarantee the clean source consistency, utmost one of the b_{kj} variables is allowed to be set to 1. There are q possible solutions that satisfy the clean source constraint: $q - 1$ for each b_{kj} being set to 1 and one for all b_{kj} variables being set to 0. Let the case when all b_{kj} are set to 0 be x (eq. (14)) and the case when exactly one of the b_{kj} is set to 1 be y (eq. (15)). The message for $b_{ij} = 0$ in Equation (16) is the maximum of the two cases x and y.

$$x = 0 + \sum_{k\neq i} \gamma_{kj}(0) \tag{14}$$

$$y = \max_{k\neq i}[0 + \gamma_{kj}(1) + \sum_{p\notin\{k,i\}} \gamma_{pj}(0)] \tag{15}$$

$$\theta_{ij}(0) = \max_{b_{kj},k\neq i}[ln\, t_{Qj}(b_{1j}, ..., b_{ij} = 0, ..., b_{qj}) + \sum_{b_{kj},k\neq i} \gamma_{kj}(b_{kj})] \\ = \max(x, y) \tag{16}$$

$\theta_{ij}(1)$ and $\theta_{ij}(0)$ combined: In Equation (17) - 23, we bring both formulas for the cases $b_{ij} = 0$ and $b_{ij} = 1$ together. According to Equation (12), the scalar message is the difference of the message values for the two settings of the binary variable.

Equation (20) is transformed to Equation (21) by the transformation $a - max(b_0, b_1, ..., b_n) = -max(b_0 - a, b_1 - a, ..., b_n - a)$. Subtracting the two sums in Equation (20), only $-\gamma_{kj}(0)$ is left (eq. (22)) and then Equation (22) is transformed to Equation (23), according to Equation (12).

$$\theta_{ij} = \theta_{ij}(1) - \theta_{ij}(0) \tag{17}$$

$$= x - max(x, y) \tag{18}$$

$$= min(0, x - y) \tag{19}$$

$$= min(0, \sum_{k\neq i} \gamma_{kj}(0) - max_{k\neq i}[\gamma_{kj}(1) + \sum_{p\notin\{k,i\}} \gamma_{pj}(0)]) \tag{20}$$

$$= min(0, -max_{k\neq i}[\gamma_{kj}(1) + \sum_{p\notin\{k,i\}} \gamma_{pj}(0) - \sum_{k\neq i} \gamma_{kj}(0)]) \tag{21}$$

$$= min(0, -max_{k\neq i}[\gamma_{kj}(1) - \gamma_{kj}(0)]) \tag{22}$$

$$= min(0, -max_{k\neq i}[\gamma_{kj}]) \tag{23}$$