An Efficient Local Search Algorithm for Correlation Clustering on Large Graphs

Nathan Cordner[1][0000−0001−6075−1621] and George Kollios[2]

[1] Utah Valley University, Orem, UT
ncordner@uvu.edu
[2] Boston University, Boston, MA

Abstract. Correlation clustering (CC) is a widely-used clustering paradigm, with many applications to problems such as classification, database deduplication, and community detection. CC instances represent objects as graph nodes, and clustering is performed based on relationships between objects (positive or negative edges between pairs of nodes). The CC objective is to obtain a graph clustering that minimizes the number of incorrectly assigned edges (negative edges within clusters, and positive edges between clusters).

For large CC instances, lightweight algorithms like the Pivot method have been preferred due to their scalability. Because these algorithms do not have state-of-the-art approximation guarantees, LocalSearch (LS) methods have often then been applied to refine their clustering results. Unfortunately, LS does not enjoy the same ability to scale since it is inherently sequential and has the potential to converge slowly.

We propose a lightweight, parallelizable LS method called InnerLocalSearch (ILS) to use in conjunction with the Pivot algorithm. We show that ILS still provides a significant improvement to clustering quality while dramatically reducing the additional running time costs incurred by LS. We demonstrate our algorithm's effectiveness against several LS benchmarks and other popular CC methods on real and synthetic data sets.

Keywords: data mining · correlation clustering · local search[3]

1 Introduction

The "min disagreement" correlation clustering (CC) problem, as originally defined by Bansal et al. [6], inputs a complete graph $G = (V, E)$ where every pair of nodes is assigned a positive (+) or negative (-) relationship. The objective is to cluster together positively related nodes and separate negatively related ones, minimizing the total number of clustering "mistakes" (negatively related pairs within clusters, and positively related pairs separated between clusters). This clustering paradigm has been used in many applications, such has its original motivation of classification [6], database deduplication [20], and community detection in social networks [30, 33]. This formulation of graph clustering has been

[3] This preprint has not undergone peer review or any post-submission improvements or corrections. The Version of Record of this contribution is published in Combinatorial Optimization and Applications, and is available online at doi.org/10.1007/978-3-031-49611-0_1

especially useful, since a specific number of clusters does not need to be specified beforehand and the only information needed as input concerns the relationship between objects—not about the objects themselves.

One of the most popular CC algorithms is Pivot, presented by Ailon et al. [3], which gives an expected 3-approximation result for correlation clustering. It runs by choosing a "pivot" node at random, adding it and all other unclustered nodes with an edge to it into a cluster, and repeating until all nodes are clustered (think of choosing a random person in a social media network and clustering together all "friends" of the pivot person). Currently the best known approximation factor is $1.994 + \epsilon$, from a linear program rounding method due to Cohen et al. [13]. Unfortunately, linear programs for correlation clustering are quite large (the number of constraints is at least cubic in the number of graph nodes). Some work has been done to reduce the size of these linear programs [21], but even algorithms based on these methods become intractable once input graphs have millions of nodes. Instead, the Pivot algorithm has been revisited many times [1, 2, 4, 12, 11, 16, 19, 22–24, 26–28, 32] because of its ease of implementation and scalability for large graphs.

A popular method of improving Pivot and other CC results is LocalSearch (LS)[4] [1, 10, 14, 15, 17, 18, 25, 26, 29, 31] which moves nodes one at a time to improve clustering costs until no more improvements can be made or a self-imposed limit is reached. Each LS pass through the node set V examines all nodes and all positive edges E^+ (yielding a time complexity of $\Theta(|V| + |E^+|)$), and LS has the potential to make multiple passes while improvements slowly accrue. A LS pass that yields only a small number of improvements can trigger another full pass through the node and positive edge sets. For larger graphs, even running a small number of LS rounds can be less practical. The purpose of this paper is to develop a new LS technique that still yields significant improvement to clusterings produced by the Pivot algorithm, without the exorbitant time cost imposed by running a full LS algorithm.

One weakness in the design of the Pivot algorithm is that it only considers immediate connections of chosen pivot nodes. In many real-world settings, it is reasonable to assume that not all "friends-of-friends" edges of pivot nodes are present. We propose a new LS method called InnerLocalSearch (ILS), which runs LS *inside* clusters only. The ILS algorithm still starts out with an $O(|V| + |E^+|)$ pass through the node set, but now has the ability to ignore positive graph edges that go between clusters. Convergence within clusters tends to be much quicker since smaller sets of nodes are being compared against each other, and once individual clusters are converged the algorithm does not need to consider the nodes inside them in future iterations. ILS is also easily parallelizable, making it possible to run LS within multiple clusters at the same time. And though it necessarily will not yield the same level of clustering improvement as the full LS algorithm, we show experimentally that ILS still lowers objective values significantly while drastically reducing the running time needed for convergence.

[4] Also called "Best One Element Move" (BOEM).

We compare Pivot with ILS against Pivot, several versions of Pivot with LS, and another popular CC algorithm called Vote [15].

1.1 Related Work

The NP-hard correlation clustering problem was introduced by Bansal et al. [6], who also provided its first constant approximation algorithm in the min disagreement setting. The best known approximation factor is $1.994 + \epsilon$, from a linear program rounding method due to Cohen et al. [13]. Correlation clustering remains an active area of research, and many variations of the problem have arisen over time; a general introduction to the correlation clustering problem and some of its early variants is given by Bonchi et al. [8]. Recent research has gone into developing sublinear time [5] and better parallel [7] CC algorithms.

The Pivot algorithm was first introduced by Ailon et al. [3]. Its efficient run time and ease of implementation have made it very popular, and it has been applied to many variants of correlation clustering that have arisen since. Recently, it has been used for uncertain graphs [26], query-constrained CC [16], online CC [24], chromatic CC [22], and fair CC [1]. It has also been shown how to run the Pivot algorithm in parallel in various settings [11, 27]. Zuylen and Williamson [32] developed a deterministic version of Pivot that picks a best pivot at each round, though at the cost of an increased running time complexity. The most efficient non-parallel implementation of Pivot uses a neighborhood oracle, where a hash table stores lists of neighbors for each node [4].

Various authors have employed LocalSearch as post-processing to improve clustering results. LS refinements and LS-based algorithms have been used in many CC variants and applications [1, 10, 14, 15, 17, 18, 25, 26, 29, 31]. Bonchi et al. [9] experimented with a heuristic method for running LocalSearch in parallel, sacrificing monotone decreasing objective values for potentially faster runtimes. Levinkov et al. [25] studied and compared other LocalSearch-based algorithms for correlation clustering.

2 Previous Algorithms

Let G be a complete graph on node set $V = \{1, \ldots, n\}$. A *clustering* of the graph G is a partition $\mathcal{C}$ of the node set V. For a given clustering $\mathcal{C} = \{C_1, ..., C_k\}$, define *intra-cluster edges* to be edges between nodes within the same cluster; define *inter-cluster edges* to be edges between nodes in distinct clusters.

For correlation clustering we assume the edge set E is partitioned into a set of *positive edges* E^+ and *negative edges* E^-. The objective of min disagreement correlation clustering is to find a clustering $\mathcal{C}$ of V that minimizes the number of negative intra-cluster edges and positive inter-cluster edges. Let similarity function $s(u, v) = 1$ if $(u, v) \in E^+$, and 0 otherwise. We write the *cost* (or *objective value*) of clustering $\mathcal{C}$ as

$$\mathrm{Cost}(\mathcal{C}, V) = \sum_{\substack{u,v \in V,\ u \neq v \\ (u,v) \text{ is intra-cluster}}} (1 - s(u, v)) + \sum_{\substack{u,v \in V,\ u \neq v \\ (u,v) \text{ is inter-cluster}}} s(u, v).$$

Algorithm 1 Pivot Clustering

```
1: function PIVOT(G = (V, E, s))
2:     Initialize empty clustering 𝒞
3:     while V ≠ ∅ do
4:         Choose random u ∈ V
5:         Let C = {u} ∪ {v ∈ V | (u, v) ∈ E+}      ▷ u and its unclustered neighbors
6:         𝒞 = 𝒞 ∪ {C}                              ▷ Add cluster C to clustering 𝒞
7:         V = V \ C                                ▷ Remove clustered nodes from V
8:     return the finished clustering 𝒞
```

For a given clustering $\mathcal{C}$, we define the *precision* of $\mathcal{C}$ to be the average number of positive edges inside clusters of $\mathcal{C}$. Let $\text{Intra}(\mathcal{C}) = \{(u, v) \mid (u, v)$ is intra-cluster$\}$. We write

$$\text{Precision}(\mathcal{C}) = \frac{1}{|\text{Intra}(\mathcal{C})|} \cdot \sum_{(u,v)\in\text{Intra}(\mathcal{C})} s(u, v).$$

We also define the *recall* of $\mathcal{C}$ to be the ratio of the number of positive edges within clusters of $\mathcal{C}$ to the total number of positive edges $|E^+|$. We write

$$\text{Recall}(\mathcal{C}) = \frac{1}{|E^+|} \cdot \sum_{(u,v)\in\text{Intra}(\mathcal{C})} s(u, v).$$

2.1 Pivot

Ailon et al. [3] proposed the randomized Pivot algorithm (Algorithm 1) for unweighted correlation clustering. A cluster C is formed by picking a *pivot node* u at random from V, then adding u and all other nodes v in V to C that are connected by a positive edge to u (that is, $(u, v) \in E^+$). If $V \setminus C$ is not empty, the algorithm continues on the subgraph induced by $V \setminus C$.

The Pivot algorithm yields a 3-approximation clustering result. Following a common implementation method [4], we assume that every node $u \in V$ has access to a *neighborhood oracle* $N(u)$ that contains all nodes v with a positive relationship to u. We write

$$N(u) = \{v \in V \mid (u, v) \in E^+\}.$$

The time complexity of Pivot is thus $O(|V| + |E^+|)$. However, Pivot often runs much quicker than its worst-case time bound since many nodes can be removed from V with each choice of pivot (see lines 5 - 7 of Algorithm 1).

2.2 LocalSearch

LocalSearch (Algorithm 2) has been a popular technique for improving the clusterings output by CC algorithms. LS takes a current CC instance G and a current

Algorithm 2 Local Search Improvements

```
1: function LocalSearch(G = (V, E, s), C, π)   ▷ π is a permutation of V (line 10)
2:     for i ∈ {1, ..., |V|} do
3:         Let u = π(i), with current cluster C
4:         Let C' = arg min_{C''∈C∪{∅}} {Cost(C'', u)}
5:         if C' ≠ C then
6:             C = (C \ {C}) ∪ {C \ {u}}                    ▷ Remove u from C
7:             C = (C \ {C'}) ∪ {C' ∪ {u}}                  ▷ Add u to C'
8:     return the augmented clustering C
9: function LocalSearchLoop(G = (V, E, s), C)
10:    Choose a random permutation π of V
11:    C' = LocalSearch(G, C, π)
12:    while Cost(C', V) < Cost(C, V) do
13:        C = C'
14:        C' = LocalSearch(G, C, π)
15:    return C
```

clustering $\mathcal{C}$ of the node set V. LS chooses a random permutation of nodes, and iteratively makes improvements to the given clustering. A current node u considers all current clusters $C \in \mathcal{C}$, and the possibility of forming a new singleton cluster, and chooses to move to whatever cluster minimizes its own contribution to the overall clustering cost. When some pass through the node set yields no new improvements, the LS algorithm halts and returns the modified clustering $\mathcal{C}'$. Limits can be imposed on LS, such as the max number of allowable iterations through the node set or a time limit on how long LS can run before returning its improved clustering.

For a given cluster $C \subseteq V$, and current node $u \in V$, we define the cost of node u to be

$$\text{Cost}(C, u) = \sum_{v \in C \setminus \{u\}} (1 - s(u, v)) + \sum_{v \in V \setminus (C \cup \{u\})} s(u, v).$$

The cost of opening a new cluster is $\text{Cost}(\emptyset, u) = \sum_{v \in V \setminus \{u\}} s(u, v)$. The algorithm greedily minimizes the cost of each node relative to the current clustering.

With a neighborhood oracle and a hash table containing the current node-to-cluster assignment, line 4 of Algorithm 2 can be computed in $O(|N(u)|)$ time. We do this by iterating over $N(u)$, tracking how many neighbors of u lie in each cluster. We then consider only the clusters C that contain neighbors of u (as well as the possibility of opening a new cluster), and compute $\text{Cost}(C, u) = |C| - |\{\text{neighbors of } u \text{ within } C\}| + (|N(u)| - |\{\text{neighbors of } u \text{ within } C\}|)$; $\text{Cost}(\emptyset, u) = |N(u)|$, so we can safely ignore clusters that do not contain neighbors of u since in that case $\text{Cost}(C, u) = |C| + |N(u)| > \text{Cost}(\emptyset, u)$. At most $|N(u)| + 1$ clusters are considered in this process, so finding the minimum is still $O(|N(u)|)$. Looping over every node in V, the time complexity of a single LS iteration is thus $\Theta(|V| + |E^+|)$. The overall running time of LocalSearch is $\Theta((|V| + |E^+|)I)$, where I is the total number of iterations made through the node set V.

Algorithm 3 InnerLocalSearch Improvements

1: **function** InnerLocalSearch($G = (V, E, s)$, $\mathcal{C} = \{C_1, \ldots, C_k\}$)
2: **for** each cluster $C_i \in \mathcal{C}$ **do** ▷ Form subgraph induced by C_i
3: Let $E_i^+ = \{(u, v) \mid u, v \in C_i, u \neq v, s(u, v) = 1\}$
4: **for** $v \in C_i$ **do**
5: Let $N_i(v) = \{u \in C_i \mid (u, v) \in E_i^+\}$ ▷ new neighborhood oracle
6: Let $G_i = (C_i, E_i^+, s)$ be the subgraph of G induced by C_i
7: Let $\mathcal{C}_i$ = LocalSearchLoop(G_i, $\{C_i\}$)
8: **return** $\mathcal{C}_1 \cup \cdots \cup \mathcal{C}_k$

3 InnerLocalSearch

Though LocalSearch has been used effectively in several applications, its has a few drawbacks that make it less practical to run on larger instances. LS is inherently sequential, since the decision of where to place a current node depends on the the decision of the previous nodes in the ordering, making it difficult to run in parallel (e.g. [9]). LS decisions are also slow, since comparisons are made across the entire vertex set (or at least all the neighbors of any given node).

To make up for these shortcomings, we propose an InnerLocalSearch algorithm (Algorithm 3). On a given clustering $\mathcal{C}$, we run LocalSearch to convergence inside each cluster $C \in \mathcal{C}$ and return the updated clustering.

We note that ILS runs LS inside each cluster, with a runtime of $O((|C_i| + |E_i^+|)I_i)$ per cluster C_i (line 7 of Algorithm 3, where I_i is the number of LS iterations needed for cluster C_i to converge). Forming the subgraph (lines 3 to 6) across all clusters can be done in $O(|V| + |E^+|)$ time. Thus the overall time complexity of ILS is $O(|V| + |E^+| + \sum_{i=1}^{k}(|C_i| + |E_i^+|)I_i)$, where k is the size of the input clustering $\mathcal{C}$. We note that ILS tends to converge much more quickly than LS since it greatly reduces the number of comparisons needed between nodes across the entire node set V. ILS cluster improvement can also be done in parallel by running the For loop in line 2 of Algorithm 3 on multiple threads.

3.1 InnerLocalSearch and Pivot

Though InnerLocalSearch (and LocalSearch too) can be run on any clustering result, we will focus on how ILS improves the Pivot algorithm. On any given clustering round, the Pivot algorithm only checks to see if nodes have positive edges to the chosen pivot node before deciding whether to cluster them together. As such it is possible for many negative edges to exist within clusters formed by the Pivot algorithm, making for a low precision clustering.

Consider Figure 1, which shows a sample cluster from the Pivot algorithm (only positive edges are drawn between nodes). Here node A was chosen as pivot and all 10 nodes from A to J are put into one cluster (drawn in purple). The cost of this cluster alone is 27, whereas the optimal partition into 3 clusters (drawn in red) reduces the cost to just 6. By putting all nodes into one cluster, the Pivot

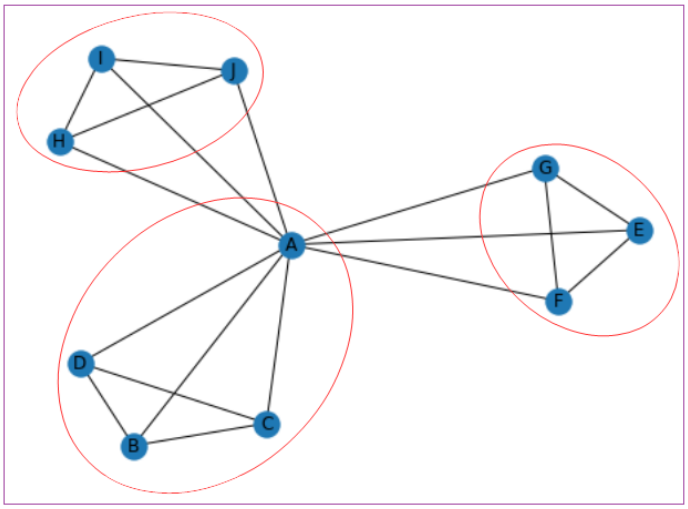

Fig. 1: Pivot Cluster Example

algorithm incurred a higher cost and ignored smaller tight-knit communities. In the worst case a Pivot cluster C might only contain $|C| - 1$ positive edges and $(|C|^2 - 3|C| + 2)/2$ negative edges, with a ratio of $(|C| - 2)/2$ negative edges for every positive.

The goal of InnerLocalSearch is thus to split larger Pivot clusters containing many negative edges into smaller clusters that contain a high number of positive edges. In other words, InnerLocalSearch seeks to quickly boost the precision of a Pivot clustering without greatly reducing its recall.

4 Experiments

We test the following algorithms: Pivot, Pivot with InnerLocalSearch (ILS), Pivot with Timed LocalSearch (Timed), Pivot with full LocalSearch (Full). For comparison, we include the Vote algorithm [15] which chooses a random node to start a cluster and adds new nodes by greedily minimizing increase of clustering cost. We present two implementations of ILS—sequential (clusters are improved one at a time), and parallel (multiple clusters being improved at once). The parallel implementation is done via Java parallel streams. For objective values, we include one benchmark (Outer) that lists the inter-cluster cost from the Pivot clustering; this is the maximum level of improvement that ILS can obtain if every misclassified edge within Pivot clusters is resolved. The Timed LocalSearch method allows full LocalSearch to run for the same amount of time used by InnerLocalSearch. For running times we include another benchmark (Match) that records the time full LocalSearch takes to match the same level of clustering improvement obtained by InnerLocalSearch.

All algorithms were implemented in Java[5] and tested on a Linux server running Rocky Linux 8.7 with a 2.9 GHz processor and 16.2 GB of RAM. Mean results are taken over 10 runs of each algorithm.

We test our algorithms on five real data sets[6] (Amazon, DBLP, Youtube, Livejournal, and Orkut), and one synthetic data set. Brief descriptions are provided in Table 1, including the value of U (the mean largest cluster size generated by the Pivot algorithm). Edges present in these data sets are interpreted as positive edges; all other node pairs are interpreted as negative edges. The synthetic data was generated by randomly assigning up to 5000 positive edges per node.

Objective values, relative improvements against the Pivot algorithm, and running times are reported in Tables 2 and 3. In Table 3 running times are reported for both sequential ILS (SeqILS) and parallel ILS (ParILS). The average number of clusters produced by each algorithm is provided in Table 4, and average precision and recall percentages for each clustering is provided in Table 5. Figure 2 contains scatter plots that show the distribution of disagreements for Pivot, ILS, Full, and Vote across the 10 runs of each algorithm.

We first note that InnerLocalSearch improves the Pivot clustering results significantly across all data sets. The Amazon data set has the smallest objective value decrease at just over 20% lower than Pivot, with all others decreasing over 25%. Some (Orkut and Synthetic) even decrease by at least 30% from the Pivot baseline.

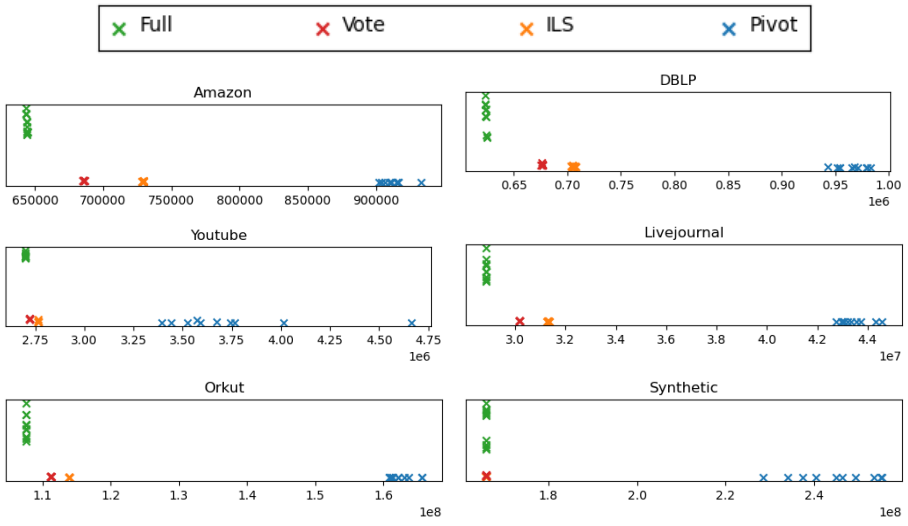

Fig. 2: Disagreement Plots for Pivot, ILS, Full, and Vote

[5] Code available at `github.com/cc-conf-sub/ils-improvement`

[6] Available at `snap.stanford.edu/data/#communities`

Table 1: Data Set Descriptions

Data Set	$\|V\|$	$\|E^+\|$	U	**Description**
Amazon	334863	925872	113	joint-purchase network
DBLP	317080	1049866	149	co-author network
YouTube	1134890	2987624	500	friend network
LiveJournal	3997962	34681189	926	friend network
Orkut	3072441	117185083	1762	friend network
Synthetic	100000	165987514	4852	random edges

Table 2: Mean Objective Values

Data Set	**Pivot**	**Outer**	**ILS**	**Timed**	**Full**	**Vote**
Amazon	912658	0.667	0.799	0.966	0.706	0.751
DBLP	964983	0.664	0.731	0.973	0.647	0.701
YouTube	3738286	0.7	0.739	1.0	0.722	0.728
LiveJournal	43455676	0.666	0.721	0.984	0.664	0.694
Orkut	162654949	0.67	0.7	0.995	0.661	0.684
Synthetic	244469396	0.665	0.679	0.888	0.678	0.679

Table 3: Mean Running Times (s)

Data Set	**Pivot**	**SeqILS**	**ParILS**	**Match**	**Full**	**Vote**
Amazon	0.12	0.24	0.13	1.2	14.73	0.52
DBLP	0.11	0.22	0.12	1.12	9.65	0.62
Youtube	0.73	0.3	0.32	4.92	24.8	1.92
Livejournal	2.01	6.2	3.5	47.8	1076	20.5
Orkut	1.73	14.6	7.93	134	4380	61.5
Synthetic	0.13	13.6	13.6	63.7	1370	53.4

Table 4: Mean Number of Clusters

Data Set	**Pivot**	**ILS**	**Full**	**Vote**
Amazon	143675	193028	155423	139842
DBLP	133627	158448	145191	134345
Youtube	801928	894597	827679	7851778
Livejournal	1817576	2556263	1995222	1837134
Orkut	651394	1771069	874592	818587
Synthetic	4327	37709	18669	14581

Table 5: Mean Precision / Recall (%)

Data Set	**Pivot**	**ILS**	**Full**	**Vote**
Amazon	51.4 / 34.2	88.5 / 24.4	85.4 / 36.7	76.8 / 37.2
DBLP	55.8 / 39.0	97.0 / 33.8	95.7 / 42.5	87.2 / 41.7
Youtube	25.9 / 12.4	87.3 / 8.7	84.0 / 12.0	76.2 / 13.0
Livejournal	28.4 / 16.6	88.7 / 11.1	85.4 / 20.3	78.0 / 18.1
Orkut	13.3 / 7.0	83.3 / 3.5	81.1 / 10.7	72.7 / 8.1
Synthetic	3.9 / 2.0	80.3 / 0.06	82.2 / 0.12	61.0 / 0.12

As expected, we see that in all cases Pivot with the full LS yields the lowest objective values. On data sets where the Pivot clusterings have higher precision (like Amazon and DBLP) the benefit of the full LocalSearch is greater. For example, LS decreases by nearly 30% on Amazon compared with the just-over 20% decrease yielded by ILS. However, on data sets where Pivot precision is quite low (like Orkut and Synthetic), the benefit of the full LS is negligible compared to ILS. We also note that ILS improvements approach their theoretical maximum (measured by Outer) on these low-precision data sets.

We see that the running times of full LocalSearch becomes a significant obstacle on the largest data sets. For Livejournal, full LocalSearch takes an average of 20 minutes to complete, and on Orkut the average is over an hour and 15 minutes. By contrast, ILS takes just under 15 seconds to finish on Orkut and provides nearly the same level of improvement as the full LocalSearch.

The Timed LocalSearch is unable to keep up with ILS in most cases; it is unable to improve even 5% over Pivot within the time limit across all of the real data sets, and on Youtube it is unable to make any gains whatsoever. The Match benchmark also slows down considerably on the larger examples. On Livejournal sequential ILS takes about 6 seconds to finish, whereas LocalSearch takes 47.8 seconds on average to match the same level of improvement. Again on Orkut sequential ILS takes under 15 seconds to finish, while LocalSearch takes over 2 minutes to match. Using parallel threads we see that ILS is able to halve its sequential running time on four out of the six data sets, and produce about the same running time as sequential ILS on the other two (Youtube and Synthetic).

As we have already noted, the Pivot algorithm has the potential to yield low-precision clusterings. On the other hand, for all the real data sets ILS reports the highest precision (with a close second on the synthetic data set). ILS necessarily reduces the recall from the Pivot algorithm, but it is not significantly lower than the recall of other clustering results across all data sets. In the worst case, ILS recall is still under 10 points lower than the Pivot recall.

5 Conclusion

In this paper we presented the InnerLocalSearch method, a viable alternative to running a full LocalSearch process to improve clustering results from the Pivot algorithm. We showed experimentally that ILS requires only a fraction of the amount of time spent by LS, while still yielding significant decreases in objective value. In many cases ILS yields nearly the same level of improvement as LS, especially when Pivot produces low-precision clusterings. We also showed that ILS greatly boosts Pivot precision without too much sacrifice to its clustering recall, and that it has the advantage of easily being run in parallel.

References

1. Ahmadian, S., Epasto, A., Kumar, R., Mahdian, M.: Fair correlation clustering. In: International Conference on Artificial Intelligence and Statistics. pp. 4195–4205. PMLR (2020)

2. Ahn, K., Cormode, G., Guha, S., McGregor, A., Wirth, A.: Correlation clustering in data streams. In: International Conference on Machine Learning. pp. 2237–2246. PMLR (2015)
3. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: ranking and clustering. Journal of the ACM (JACM) **55**(5), 1–27 (2008)
4. Ailon, N., Liberty, E.: Correlation clustering revisited: The "true" cost of error minimization problems. In: International Colloquium on Automata, Languages, and Programming. pp. 24–36. Springer (2009)
5. Assadi, S., Wang, C.: Sublinear time and space algorithms for correlation clustering via sparse-dense decompositions. In: 13th Innovations in Theoretical Computer Science Conference (ITCS 2022) (2022)
6. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Machine Learning **56**(1-3), 89–113 (2004)
7. Behnezhad, S., Charikar, M., Ma, W., Tan, L.Y.: Almost 3-approximate correlation clustering in constant rounds. In: 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS). pp. 720–731. IEEE (2022)
8. Bonchi, F., Garcia-Soriano, D., Liberty, E.: Correlation clustering: from theory to practice. In: KDD. p. 1972 (2014)
9. Bonchi, F., Gionis, A., Ukkonen, A.: Overlapping correlation clustering. Knowledge and Information Systems **35**, 1–32 (2013)
10. Chehreghani, M.H.: Clustering by shift. In: 2017 IEEE International Conference on Data Mining (ICDM). pp. 793–798. IEEE (2017)
11. Chierichetti, F., Dalvi, N., Kumar, R.: Correlation clustering in mapreduce. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 641–650 (2014)
12. Christiansen, L., Mobasher, B., Burke, R.: Using uncertain graphs to automatically generate event flows from news stories. In: Proceedings of Workshop on Social Media World Sensors at ACM Hypertext 2017 (SIDEWAYS, HT'17) (2017)
13. Cohen-Addad, V., Lee, E., Newman, A.: Correlation clustering with sherali-adams. In: 2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS). pp. 651–661. IEEE (2022)
14. Coleman, T., Saunderson, J., Wirth, A.: A local-search 2-approximation for 2-correlation-clustering. In: European Symposium on Algorithms. pp. 308–319. Springer (2008)
15. Elsner, M., Schudy, W.: Bounding and comparing methods for correlation clustering beyond ilp. In: Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing. pp. 19–27 (2009)
16. García-Soriano, D., Kutzkov, K., Bonchi, F., Tsourakakis, C.: Query-efficient correlation clustering. In: Proceedings of The Web Conference 2020. pp. 1468–1478 (2020)
17. Gionis, A., Mannila, H., Tsaparas, P.: Clustering aggregation. Acm Transactions on Knowledge Discovery from Data (TKDD) **1**(1), 4–es (2007)
18. Goder, A., Filkov, V.: Consensus clustering algorithms: Comparison and refinement. In: 2008 Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments (ALENEX). pp. 109–117. SIAM (2008)
19. Halim, Z., Waqas, M., Hussain, S.F.: Clustering large probabilistic graphs using multi-population evolutionary algorithm. Information Sciences **317**, 78–95 (2015)
20. Haruna, C.R., Hou, M., Eghan, M.J., Kpiebaareh, M.Y., Tandoh, L.: A hybrid data deduplication approach in entity resolution using chromatic correlation clustering. In: International Conference on Frontiers in Cyber Security. pp. 153–167. Springer (2018)

21. Hua, J., Yu, J., Yang, M.S.: Star-based learning correlation clustering. Pattern Recognition **116**, 107966 (2021)
22. Klodt, N., Seifert, L., Zahn, A., Casel, K., Issac, D., Friedrich, T.: A color-blind 3-approximation for chromatic correlation clustering and improved heuristics. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 882–891 (2021)
23. Kollios, G., Potamias, M., Terzi, E.: Clustering large probabilistic graphs. IEEE Transactions on Knowledge and Data Engineering **25**(2), 325–336 (2011)
24. Lattanzi, S., Moseley, B., Vassilvitskii, S., Wang, Y., Zhou, R.: Robust online correlation clustering. Advances in Neural Information Processing Systems **34** (2021)
25. Levinkov, E., Kirillov, A., Andres, B.: A comparative study of local search algorithms for correlation clustering. In: Pattern Recognition: 39th German Conference, GCPR 2017, Basel, Switzerland, September 12–15, 2017, Proceedings 39. pp. 103–114. Springer (2017)
26. Mandaglio, D., Tagarelli, A., Gullo, F.: In and out: Optimizing overall interaction in probabilistic graphs under clustering constraints. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1371–1381 (2020)
27. Pan, X., Papailiopoulos, D., Oymak, S., Recht, B., Ramchandran, K., Jordan, M.I.: Parallel correlation clustering on big graphs. In: Advances in Neural Information Processing Systems. pp. 82–90 (2015)
28. Puleo, G.J., Milenkovic, O.: Correlation clustering with constrained cluster sizes and extended weights bounds. SIAM Journal on Optimization **25**(3), 1857–1872 (2015)
29. Queiroga, E., Subramanian, A., Figueiredo, R., Frota, Y.: Integer programming formulations and efficient local search for relaxed correlation clustering. Journal of Global Optimization **81**, 919–966 (2021)
30. Shi, J., Dhulipala, L., Eisenstat, D., Lacki, J., Mirrokni, V.: Scalable community detection via parallel correlation clustering. Proceedings of the VLDB Endowment **14**(11), 2305–2313 (2021)
31. Thiel, E., Chehreghani, M.H., Dubhashi, D.: A non–convex optimization approach to correlation clustering. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 5159–5166 (2019)
32. Van Zuylen, A., Williamson, D.P.: Deterministic pivoting algorithms for constrained ranking and clustering problems. Mathematics of Operations Research **34**(3), 594–620 (2009)
33. Veldt, N., Gleich, D.F., Wirth, A.: A correlation clustering framework for community detection. In: Proceedings of the 2018 World Wide Web Conference. pp. 439–448 (2018)