A Local-Search 2-Approximation for 2-Correlation-Clustering[⋆]

Tom Coleman, James Saunderson, and Anthony Wirth

The University of Melbourne

Abstract. CorrelationClustering is now an established problem in the algorithms and constrained clustering communities. With the requirement that at most two clusters be formed, the minimisation problem is related to the study of *signed graphs* in the social psychology community, and has applications in statistical mechanics and biological networks.

Although a PTAS exists for this problem, its running time is impractical. We therefore introduce a number of new algorithms for 2CC, including two that incorporate some notion of local search. In particular, we show that the algorithm we call PASTA-toss is a 2-approximation on complete graphs.

Experiments confirm the strong performance of the local search approaches, even on non-complete graphs, with running time significantly lower than rival approaches.

1 Introduction

The Two-Correlation-Clustering (2CC) problem asks us to partition a dataset into two clusters given only *advice* about *pairs* of points in the dataset. This *advice* comes in the form of soft *must-link* and *cannot-link* constraints. The aim is to minimise the number of such constraints violated in forming the clusters.

1.1 The 2CC Problem

The CorrelationClustering problem [1] asks us to form a clustering of a signed graph that minimises the number of edges that are not respected. In the 2CC variant, the number of clusters is restricted to two. This bears some similarity to the MaxCut problem. Formally, the input is a graph $G = (V, E)$ and a labelling on edges $l : E \to \{-1, +1\}$. The output is a clustering of the vertices $c : V \to \{-1, +1\}$. The aim is to choose a clustering that minimises the number of edges that disagree with the labelling, viz.

$$|\{e = (v, w) \in E \text{ s.t. } l(e) \neq c(v) \cdot c(w)\}| .$$

[⋆] This work was supported by the Australian Research Council through Discovery Grant DP0663979.

We refer to this as the *cost of the clustering c under labelling l* or just the *cost* if the clustering and labelling are clear from the context. Note that *labelling* refers to edges, whereas *clustering* refers to vertices, and that n refers to $|V|$.

If the graph is not two-connected, then each two-connected component can be considered independently. Without loss of generality, we will therefore assume that the input graph is two-connected.

1.2 Related Work

Initial work on signed graphs [2, 3] focused on graph theory, rather than optimisation. Early results [4] demonstrated that 2CC is an NP-complete problem, both on complete graphs, and in general.

Bansal *et al.* [1] put forward the first approximation algorithm for MIN-2CC on *complete* graphs, with factor 3. Giotis and Guruswami [5] completed the picture, from a *theoretical* viewpoint, for 2CC on complete graphs by developing a PTAS (polynomial time approximation scheme) for both the maximisation and minimisation versions of the problem. For CorrelationClustering on complete graphs, a PTAS exists for maximisation, but minimisation is APX-hard [6]. The best known upper bound for min-CorrelationClustering on complete graphs is a 5/2-approximation developed by Ailon *et al.* [7].

On *general* graphs, the problem is more difficult to solve. There is a direct relationship between 2CC and the classic MaxCut problem: replace all + edges on the signed graph with a pair of − edges meeting at a new vertex. The classic SDP-based approximation algorithm, by Goemans and Williamson [8], achieves a 0.878-approximation for MaxCut. Dasgupta *et al.* [9] extend this result to the maximization version of the 2CC problem, achieving the same approximation factor. Note that CorrelationClustering on general graphs and Minimum Multicut reduce to one another, leading to $O(\log n)$ approximations [6]. Finally, Huffner *et al.* [10] use a fixed parameter algorithm, and some data reduction rules, to solve 2CC exactly in greatly reduced time compared to a brute force algorithm. However, such algorithms are still exponential in running time.

1.3 History of the 2CC Problem

The 2CC problem has been repeatedly rediscovered, and renamed, since it was first defined by Harary [2] in 1950. Harary introduced the *signed graph*: an undirected graph with +1 or −1 labels on the edges (corresponding to must-link and cannot-link advice). He also introduced the notion of *imbalance* in a signed graph, which corresponds to the 2CC cost of the graph, the number of violated constraints. Harary considered a psychological interpretation of the problem: positive edges correspond to pairs of people who like one another, and negative edges to pairs who dislike one another. His aim was to find two highly *cliquey* groups.

Apart from social psychology, the study of signed graphs has many other applications, notably in statistical mechanics, where it relates to energy configurations of the Ising model with no external field. Solé and Zaslavsky [3] show a connection to coding theory: between signings of a graph and the cutset code defined by that graph. Also, Dasgupta *et al.* [9] apply the problem to the decomposition of large-scale biological networks into monotonic subsystems.

1.4 Layout of the Paper

In Section 2, we outline the majority of the algorithms used in this paper. In Section 3, we show that the PASTA-TOSS algorithm is a 2-approximation. Section 4 explains a more involved algorithm, PASTA-FLIP, which is similar in structure to PASTA-TOSS. Finally, Section 5 outlines the experiments we conducted to validate the practical performance of these algorithms.

2 Algorithms to Solve 2CC

In this section, we provide details about most of the algorithms for 2CC that we will run experiments on. This includes both existing work and two new algorithms: PAST and SPECTRAL.

2.1 Pick-a-Vertex Type Algorithms

The Pick-a-Vertex Algorithm Bansal *et al.* [1] outline a simple approximation algorithm for 2CC, which we call PICK-A-VERTEX. It provides the inspiration for a number of algorithms that we introduce in this paper. First some notation: let $N^+(v)$ be the set of vertices that share a positive-labelled edge with v, and $N^-(v)$ those that share a negative edge. So PICK-A-VERTEX is

> For each vertex v, there is an associated partitioning: one cluster being $\{v\} \cup N^+(v)$, the other $N^-(v)$. Of these n partitionings, return the one that minimises the number of disagreements with the labels.

Bansal *et al.* demonstrate that this simple algorithm is a 3-approximation for the 2CC problem on complete graphs. It turns out that the 3-approximation is tight, a fact not mentioned in the original paper. Consider a *complete* graph, consisting solely of positive edges, apart from a Hamiltonian cycle of negative edges. The optimal solution (placing all vertices together) has cost n, whilst any PICK-A-VERTEX solution will have cost $3n - 10$.

Notice that the PICK-A-VERTEX clustering described above is *not* a local optimum. This fact is the inspiration for some of the algorithms we introduce.

The PAST Algorithm The PICK-A-VERTEX algorithm was designed for complete graphs. There is no obvious extension to incomplete graphs, as there may not be candidate vertices v that are adjacent to every other vertex.

In the complete case, the edges incident to v form a *spanning tree* of the underlying graph G. Now, any spanning tree of G induces a unique clustering that is consistent with the tree. From this perspective, PICK-A-VERTEX is considering spanning tree-based clusterings. We therefore propose the PAST (Pick-a-Spanning-Tree) algorithm:

> For each vertex v, perform a breadth first search from v to find a spanning tree, and use that tree to induce a clustering. Return the *best* of the n clusterings.

By using breadth-first-search trees, PAST chooses the same spanning trees as Pick-a-Vertex on complete graphs, and is thus a generalisation.

2.2 Local Search

Local-search algorithms have been successful in practice for many years, and more recently as approximation algorithms [11], for various combinatorial problems. For the 2CC problem, the obvious local improvement to make to is to move (*toss*) a vertex from one cluster to the other if, by doing so, the cost of the clustering is lowered.

Given a clustering c, the clustering c_v represents the same clustering as c, except with $v \in V$ in the opposite cluster. We then define $\lambda_v = \text{cost}(c) - \text{cost}(c_v)$, the improvement caused by the change (non-negative, if there is some improvement). With this in mind, we define LocalSearch as follows:

> Given a clustering c, let w be the vertex with maximum λ_w. If $\lambda_w \leq 0$, stop, otherwise let $c \leftarrow c_w$ and repeat.

Counter Example for Local Search Approximation The LocalSearch algorithm, used naively, has no good approximation guarantee. Consider a complete graph with $n/2$ disjoint edges labelled $-$ (all other edges are labelled $+$). The global minimum here has cost $n/2$, however, there is a local minimum—which cuts across each minus-edge—that has cost $n(n-2)/4$.

2.3 The PASTA-toss Algorithm

PASTA-toss is defined in the following way:

> Generate n breadth-first-search trees, one emanating from each vertex. For each such tree T, after finding the 2-clustering ${}^{T}c$ consistent with T, run LocalSearch on ${}^{T}c$. Return the best locally-optimal solution.

Clearly the PASTA-toss algorithm will return a solution no worse than the PAST algorithm. In Section 3 we show that PASTA-toss is a two-approximation on complete graphs.

2.4 A Spectral Algorithm

In an earlier paper [12], we formulated 2CC as an eigenvalue problem, similar to the spectral clustering approach. We refer the reader to that paper for a full exposition of that algorithm, which we will refer to as Spectral.

2.5 The PTAS

Giotis and Guruswami [5] discovered a PTAS (polynomial time approximation scheme) for the k-CorrelationClustering problem, with arbitrary k. They first take a random sample of the vertices and then use each possible clustering of the sample as a basis for a clustering of the entire data set.

Giotis and Guruswami's scheme provides a $(1+\varepsilon)$-factor approximation algorithm that runs in time $2^{O(1/\varepsilon^3)}$. However, the constants involved are large enough that the smallest possible sample size is greater than 4000. In practice, checking every sample clustering is infeasible. We investigated using the same techniques with smaller sample sizes. Consequently, there are no approximation guarantees, but we anticipated similar behaviour to the full-blown PTAS.

3 PASTA-toss is a 2-approximation

In this section we develop some theoretical results, leading to a proof that PASTA-toss is a 2-approximation. To begin, we need the concept of a switching class.

3.1 Switching

The notion of *switching* in signed graphs is well established [13]. Given a labelling l, we generate another labelling l_v by selecting a vertex v and flipping the labels on the edges incident to v. We may repeat this switching operation at other vertices, generating further labellings. The family of all possible labellings can be partitioned into equivalence classes under this (multiple) switching operation: we refer to labellings in the same class as *switching equivalent*.

In this paper we also introduce the notion of switching on 2-clusterings: we *switch* a clustering c to c_v by tossing v to the other cluster. In this way, every clustering can be obtained by a series of switching steps from c.

Lemma 1 *The cost of c under l is the same as the cost of c_v under l_v.*

Proof. The only edges affected by these operations are edges incident to v. For such an edge $e=(v,u)$, $l(e)=-l_v(e)$, and $c(v)=-c_v(v)$, whilst $c(u)=c_v(u)$. Thus the cost of such an edge is unchanged. □

Lemma 1 tells us that if l has a solution of cost k, l_v also has a solution of cost k. Also as $(l_v)_v=l$, the converse is true. Consequently, we see the following useful corollaries.

Corollary 1 *The optimal costs of all labellings in a switching class are the same. In particular, if c^* is an optimal clustering for l, then c^*_v is an optimal clustering for l_v.*

Corollary 2 *For a given labelling l, there exists a labelling l', switching equivalent to l, for which placing all vertices together in one cluster is optimal.*

Note that the optimal cost for l' in Corollary 2 equals the number of negative edges in l'.

3.2 Switching-Invariant Algorithms

Imagine we knew that an algorithm behaved in essentially the same way on all switching-equivalent labellings. Then Corollary 2 tells us that we can focus on labellings in which the optimum has all elements in one cluster.

Definition 1 *An algorithm is* switching invariant *if, whenever it produces c on input l, it produces c_v on input l_v.*

We now investigate the behaviour of two key algorithms under switching.

Lemma 2 PAST *is switching invariant.*

Proof. Let T be any spanning tree of G inducing a clustering ${}^{T}\!c$ under PAST. Consider two vertices u and x in V. Whether they are clustered together depends only on the parity of the number of negative edges on the path between u and x in tree T. If v is not on this path, clearly the parity is unchanged. If v is on the path, the parity is changed only if u or x is v.

So under l_v, the clustering based on T is switching invariant. Lemma 1 tells us that the spanning tree that induces the best clustering on l also induces the best clustering on l_v. □

We can now infer an interesting fact about spanning trees.

Lemma 3 *For a given labelling l on a graph G, there exists a spanning tree T that induces an optimal clustering.*

Proof. Consider labelling l' as defined in Corollary 2. The positively-labelled edges in l' form a subgraph that is connected and spans G: if they did not, then there would be a non-trivial cut of G with only l'-negative edges. This would imply that the optimum clustering must use two clusters, contradicting the definition of l'. Hence, we can find a spanning tree T of positively-labelled edges in l': this induces a solution with all vertices in one cluster.

The proof of Lemma 2, combined with Corollary 1 shows that T will induce the optimum solution on l. □

LocalSearch is not exactly switching invariant, but we can prove a similar result.

Lemma 4 *If* LocalSearch *is given l as input and uses c as a starting point, resulting in solution $\bar{c}$, then given input l_v and starting point c_v,* LocalSearch *produces solution $(\bar{c})_v$.*

Proof. Consider running two simultaneous instances of LocalSearch, one starting from c and the other starting from c_v. To begin with, the only edges that could possibly be different are the edges incident to v. The proof of Lemma 1 shows that the edges that incur a cost are the same in (l_v, c_v) as they are in (l, c). So in both cases, λ_u is the same, for all $u \in V$. Therefore the same sequence of vertices will be chosen to be tossed. □

The following lemma is an immediate consequence of Lemmas 2 and 4.

Lemma 5 PASTA-toss *is switching invariant.*

3.3 Proof that PASTA-toss is a 2-approximation

Since PASTA-TOSS is switching invariant, we can analyse its behaviour on input labellings in which the optimum places all vertices in one cluster (refer to Corollaries 1 and 2). For such a labelling, no vertex has minus-degree more than $n/2$, and the optimum cost is simply the total number of minus-edges in the graph. If we let β be the minimum of the minus-degree of all the vertices, then $\mathrm{cost}^* \geq \beta n/2$.

To analyse the performance of PASTA-TOSS consider the iteration where PASTA-TOSS uses the spanning tree from v, a node of minus-degree β. Initially, the algorithm splits the vertices into two sets, $X_0 = \{v\} \cup N^+(v)$ and $Y_0 = N^-(v)$. As the local search progresses, vertices will be tossed from one set to the other (call them X and Y). Consider the point at which the first vertex is tossed from X to Y. Note that this means $|Y| \leq \beta$.

We can compare the cost of the clustering (X, Y) to cost^* in a fashion similar to Bansal *et al.*. We can form an estimate of the difference by counting the number of $+$ edges between X and Y, discounting the $-$ edges. For a vertex $v \in V$, and a set A, define A_v^+ to be the number of $+$ edges from v to A. A_v^- is defined in the analogous way. Then

$$\mathrm{cost}(X, Y) - \mathrm{cost}^* = \sum_y X_y^+ - \sum_y X_y^- = \sum_y \mathrm{pull}_y \leq \beta \max_{y \in Y} \mathrm{pull}_y \qquad (1)$$

Where $\mathrm{pull}_y = X_y^+ - X_y^-$ is the "pull" that X exerts on y.

If we let $\mathrm{push}_y = Y_y^- - Y_y^+$ (the "push" that Y exerts on y), the local improvement of swapping any node $y \in Y$ is given by

$$\mathrm{imp}_y = \mathrm{pull}_y + \mathrm{push}_y \qquad (2)$$

So we can use a bound on the local improvement of swapping a node (from X) to get a contradictory bound on pull_y for any y.

Theorem 1 PASTA-TOSS *is a 2-approximation on complete graphs.*

Proof. We claim at this point, when the first node to be swapped is from X, that $\mathrm{cost}(X, Y) \leq 2\mathrm{cost}^*$.

Arguing by contradiction, suppose that $\mathrm{cost}(X, Y)$ is not within a factor 2 of cost^*. Then there must be some $y_0 \in Y$ such that $\mathrm{pull}_{y_0} > n/2$.

Let x_0 be the vertex from X that is about to be swapped. By definition,

$$Y_{x_0}^+ + Y_{x_0}^- = |Y| \text{ and } X_{x_0}^+ + X_{x_0}^- = n - |Y| - 1$$

Since we have assumed that the optimum solution places all vertices together, x_0 is incident to at most $n/2$ negative edges, and so $X_{x_0}^- \leq n/2$. So we have

$$\mathrm{imp}_{x_0} = X_{x_0}^- + Y_{x_0}^+ - X_{x_0}^+ - Y_{x_0}^- \leq 2X_{x_0}^- - (X_{x_0}^- + X_{x_0}^+) + (Y_{x_0}^+ + Y_{x_0}^-),$$

which is at most $2|Y| + 1$. Given x_0 is the vertex which is about to be swapped, $\mathrm{imp}_{y_0} \leq \mathrm{imp}_{x_0} \leq 2|Y| + 1$.

Alternatively, if the algorithm ends without ever swapping an $x \in X$, at the conclusion, $\text{imp}_{y_0} \leq 0 < 2|Y| + 1$.

So, using (2) and our assumption, we have

$$\text{push}_{y_0} = \text{imp}_{y_0} - \text{pull}_{y_0} < 2|Y| - 1 - n/2 \tag{3}$$

Now we use the fact that y_0 has to have at least β minus-edges to show a contradictory lower-bound on push_{y_0}. We have

$$\begin{aligned}
\text{push}_y &= Y^-_{y_0} - Y^+_{y_0} \\
&= 2Y^-_{y_0} - |Y| + 1 \\
&\geq |Y| - 2X^-_{y_0} + 1 \\
&= |Y| + X^+_{y_0} - X^-_{y_0} - (X^+_{y_0} + X^-_{y_0}) + 1 \\
&> 2|Y| + 1 - n/2
\end{aligned}$$

The first equality follows as $Y^-_{y_0} + Y^+_{y_0} = |Y| - 1$, the first inequality as the minus-degree of y_0, $Y^-_{y_0} + X^-_{y_0}$ is at least $\beta \geq |Y|$, and the second as $X^+_{y_0} + X^-_{y_0} = n - |Y|$ and $X^+_{y_0} - X^-_{y_0} = \text{pull}_{y_0} > n/2$. □

4 The PASTA-flip Algorithm

The PASTA-FLIP algorithm is another local-search approach, but rather more involved than tossing vertices between clusters.

4.1 Removing Bad Cycles

By removing bad cycles, defined below, we will produce a graph that is trivial to cluster.

Definition 2 *A* bad cycle *is a cycle C in G in which there is an odd number of negative-labelled edges.*

These cycles are called *bad* as there is no clustering of the vertices in C that satisfies all the labels of C's edges. On the other hand, if there are no bad cycles in a graph, solving the problem is easy.

Lemma 6 *Suppose l is a labelling that causes G to have no bad cycles. Then there is a clustering of the vertices with cost zero.*

Proof. Choose some vertex v and assign every other vertex u to a cluster based on the parity of the number of negative edges on the paths between u and v. Note that the parity is uniquely defined: if not, there would be a cycle with an odd number of negative edges. This proves the lemma, as all paths (and edges, as length one paths) are respected. □

The basic principle of PASTA-flip is that it might be a good idea to flip the label on an edge that is involved in many bad cycles: the cycles would then become good. If this process could be repeated in an organised way so that no bad cycles remained, then the clustering problem would be trivial. This approach has two drawbacks. Firstly, it is conceivable that there is a scenario where there exists a bad cycle, yet there is no edge to flip that will reduce the number of bad cycles. More importantly, there are too many cycles to consider (possibly an exponential number).

Cycle Bases Let us represent a set of edges by an $|E|$-dimensional vector with entries in $\mathbb{Z}_2$. The set of all cycles is a subspace of this vector space. Standard results show that the cycle space has dimension $|E|-|V|+1$ and can be generated by a spanning tree T of G. We obtain a *fundamental basis* of the cycles by forming a cycle C_e for each edge $e = (v, w) \notin T$: C_e is e plus the path in T from v to w.

4.2 The PASTA-flip Algorithm

Consider such a fundamental cycle basis. Each edge $e \notin T$ will only be involved in a single cycle in the basis, C_e. So there is one straightforward technique to ensure that each cycle in the basis is good: if C_e is bad, simply flip e. At the end of this process, the labellings on the T-edges will be respected, which is exactly the PAST algorithm.

However this is wasteful—edges inside T are involved in many basis cycles. Flipping one of these edges could potentially fix a number of bad cycles (in the basis), and thus mean fewer flips. Each flip represents a disagreement between the output clustering and the (original) edge labelling. With this in mind, we define the PASTA-flip (Pick-a-Spanning Tree and flip) algorithm as follows:

> For each vertex v create a breadth-first-search tree T_v. While there is an edge $e \in T_v$ which is involved in more bad cycles than good, flip e. When there are no more such edges, flip edges outside of T_v. Return the solution found of lowest cost.

The action of "flipping" never worsens the 2CC cost, giving the following lemma.

Lemma 7 *The cost of the solution returned by* PASTA-flip *is no greater than the cost of the solution returned by* PAST.

5 Experimental Work

5.1 Algorithms Tested

In our experimental work, we tested all algorithms mentioned in Section 2, along with PASTA-flip. As mentioned, the PTAS was not feasible to implement, so we tested a PTAS-like algorithm, called PTAS-k, where k is the sample size. Also, the algorithm we refer to below as LocalSearch takes n randomised starting clusterings, and produces the best clustering found after toss-based search from each. This is to compare it to PASTA-TOSS, which uses n PAST-style starting clusterings. Also we experimented with PASTA-FLIP+TOSS, which performs PAST, and for each tree flips edges (until no more flips are possible) and then tosses vertices. Additionally we used the code provide by Dasgupta *et al.* [9] to test the Goemans-Williamson style SDP algorithm that they developed.

5.2 Datasets

For our experimental work, we used three datasets: the regulatory network of human epidermal growth factor (EGFR), as used by both Dasgupta *et al.* [9] and Huffner *et al.* [10] in their investigations, and two synthetic datasets.

Each synthetic dataset was generated randomly subject to two parameters, which were independent over each edge. The first, p_e, is the probability that an edge exists (with either sign), and given the edge exists. The second, p, is the probability that the edge agrees to a randomly generated initial clustering.

The first data set, called SPARSE, had problems of size 200, a p_e value of 0.05, and a p value of 0.3, which was an attempt to approximate the EGFR dataset. The second data set, called COMPLETE, had problems of size 100, $p_e = 1$—thus all graphs are complete—and $p = 0.45$. We found empirically that lower values of p result in 2CC problems that are easy to solve.

All experiments were run on a 2 GHz Intel Core 2 Duo machine with 2GB of RAM, running MAC OS X 10.5.2. All algorithms were implemented in C, apart from SPECTRAL and GW-SDP, which were run in Matlab 7.4.0. Note that this means the times recorded for the Matlab algorithms perhaps were not entirely appropriate for comparison.

5.3 Results

Figures 1 and 2 show the relative performances of the algorithms discussed in this paper on the EGFR and COMPLETE datasets. These plots compare the algorithmic performance (number of errors) to the time taken to achieve that performance. As we can see, the PTAS algorithms and PAST perform poorly as a rule; DASGUPTA can achieve good results, but is very slow in comparison to our algorithms.Although the SPECTRAL technique can be quite fast on sparse graphs, its performance is not great. Table 5.3 summarises the results on all three datasets.

6 Conclusions

In this paper, we have introduced some new algorithms for solving the 2CC problem: PASTA-TOSS, PASTA-FLIP, and the spectral method. The PASTA-TOSS algorithm is a 2-approximation on complete graphs. In general, performances of the local-search enhanced algorithms are impressive, with comparatively low running times. Certainly, they form a more practical approach than the existing PTAS, whilst retaining some proved performance bounds.

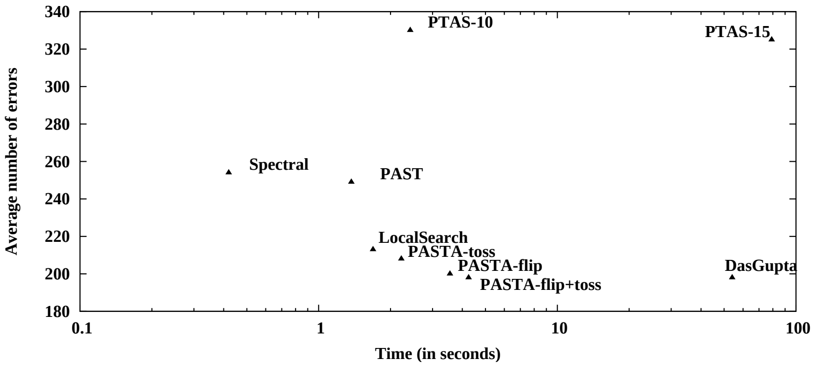

Fig. 1. The time/effectiveness profile on the EGFR Dataset.

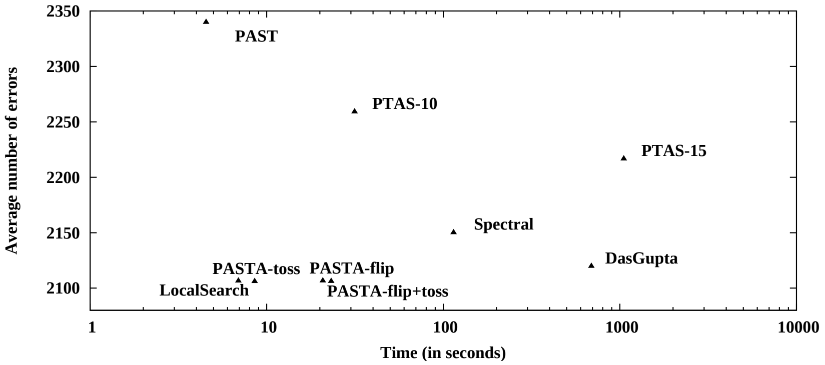

Fig. 2. The time/effectiveness profile on 100 instances of *complete* signed graphs, $n = 100$ and $p = 0.45$.

Table 1. The results of running all algorithms on all datasets. We report the average of the percentage (%) relative difference between the number of errors compared to LocalSearch, over all problem instances in the dataset. The running time is measured in seconds.

	EGFR		Sparse		Complete	
Algorithm	Cost	Time	Cost	Time	Cost	Time
Dasgupta	**-0.070**	54.01	-0.011	26.28	0.006	689.01
Spectral	0.192	0.42	0.224	2847.70	0.021	114.35
PAST	0.169	1.37	0.298	29.16	0.111	4.54
PASTA-flip	-0.061	3.55	-0.033	80.60	0.000	20.77
PASTA-toss	-0.023	2.22	-0.020	54.82	**-0.000**	8.54
PASTA-flip+toss	**-0.070**	4.25	**-0.042**	99.62	-0.000	23.18

6.1 Further Work

Is it possible to apply these methods to solve CorrelationClustering problems in which the required number of clusters is fixed at a number larger than two? Although the local search step generalises easily, it is not at all clear how to generalise the spanning tree approach.

We have no example showing that the 2-approximation result for PASTA-toss is tight. We suspect that the proof technique for the approximation performance of PASTA-toss extends to PASTA-flip, but have not yet investigated this in detail. The approximability of 2CC on general graphs is not well understood: the results for MaxCut also apply, but they tell us little about the minimisation problem. The good performance of our algorithms on general graphs suggest that we may obtain something better than the reduction to Minimum Multicut of the generic CorrelationClustering problem.

References

1. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Machine Learning **56**(1) (2004) 89–113
2. Harary, F.: On the notion of balance of a signed graph. Michigan Mathematical Journal **2** (1953) 143–46
3. Solé, P., Zaslavsky, T.: A coding approach to signed graphs. SIAM Journal on Discrete Mathematics **7** (1994) 544–53
4. Shamir, R., Sharan, R., Tsur, D.: Cluster graph modification problems. Discrete Applied Mathematics **144**(1-2) (2004) 173–82
5. Giotis, I., Guruswami, V.: Correlation clustering with a fixed number of clusters. Proceedings of the 17th Annual ACM-SIAM Symposium on Discrete Algorithms (2006) 1167–76
6. Charikar, M., Guruswami, V., Wirth, A.: Clustering with qualitative information. Journal of Computer and System Sciences **71**(3) (2005) 360–83
7. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: Ranking and clustering. Proceedings of the 37th annual ACM Symposium on Theory of Computing (2005) 684–93
8. Goemans, M., Williamson, D.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM **42**(6) (1995) 1115–45
9. DasGupta, B., Enciso, G., Sontag, E., Zhang, Y.: Algorithmic and complexity results for decompositions of biological networks into monotone subsystems. BioSystems **90**(1) (2007) 161–78
10. Huffner, F., Betzler, N., Niedermeier, R.: Optimal edge deletions for signed graph balancing. Proceedings of the 6th Workshop on Experimental Algorithms (2007) 297–310
11. Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., Pandit, V.: Local search heuristics for k-median and facility location problems. SIAM Journal on Computing **33**(3) 544–62
12. Coleman, T., Saunderson, J., Wirth, A.: Spectral clustering with inconsistent advice. Proceedings of the 25th Annual International Conference on Machine Learning (2008) 152–159
13. Zaslavsky, T.: Signed Graphs. Discrete Applied Mathematics **4** (1982) 47–74