Computing the frustration index in signed graphs using binary programming

Samin Aref   Andrew J. Mason*   Mark C. Wilson
Department of Computer Science   *Department of Engineering Science
University of Auckland, Auckland, Private Bag 92019 New Zealand
sare618@aucklanduni.ac.nz

Abstract

Computing the frustration index of a signed graph is a key to solving problems in many fields including social networks, physics, material science, and biology. In social networks the frustration index determines the distance of a network from a state of structural balance. Although the definition of the frustration index goes back to 1960, there is no method for computing the frustration index of large scale networks. The main reason seems to be the complexity of computing the frustration index which is closely related to well-known NP-hard problems such as MAXCUT.

New quadratic and linear binary programming models are developed to compute the frustration index exactly. We introduce several speed-up techniques involving prioritised branching and valid inequalities inferred from graph structural properties. The speed-up techniques make our models capable of processing graphs with thousands of nodes and edges in seconds on inexpensive hardware. The solve time and solution quality comparison against the literature shows the superiority of our models in both random and real signed networks.

Keywords: 0-1 programming, Optimisation, Frustration index, Mixed integer programming, Signed graphs, Structural balance

Mathematics Subject Classification: 90C09 90C11 90C90 90C35 05C22

1 Introduction

Local ties between entities lead to global structures in networks. Ties can be formed as a result of interactions and individual preferences of the entities in the network. The dual nature of interactions in various contexts means the ties may form in two opposite types, namely positive ties and negative ties. In a social context, this is interpreted as friendship vs. enmity or trust vs. distrust between people. The term *signed network* embodies a multitude of concepts involving relationships characterisable by ties with plus and minus signs. *Signed graphs* are used to model such networks where edges have positive and negative signs. *Structural balance* in signed graphs is a macro-scale structural property that has become a focus in network science.

Structural balance theory was the first attempt to understand the sources of tensions and conflicts in groups of people with signed ties [17]. Cartwright and Harary identified cycles of the graph (closed-walks with distinct nodes) as the origins of tension, in particular cycles containing an odd number of negative edges [7]. Signed graphs in which no such cycles are present satisfy the property of structural balance. The vertex set of *balanced* signed networks can be partitioned into $k \leq 2$ subsets such that each negative edge joins vertices belonging to different subsets [7]. For graphs that are not totally balanced, a distance from total balance (a measure of partial balance) can be computed. Among various measures is the *frustration index* that indicates the minimum number of edges whose removal (or equivalently, negation) results in balance [1, 15, 24].

2 Literature review

The frustration index is a key to frequently stated problems in many different fields of research [11, 12, 16, 19, 20]. In biological networks, optimal decomposition of a network into monotone subsystems is made possible by calculating the signed graph frustration index [19]. In finance, risk of a portfolio is related to the balance of its underlying signed graphs [16]. In Physics, the frustration index provides the minimum energy state in models of atomic magnets [20]. In international relations, signed clustering of countries in a region can be investigated using the frustration index [11]. In Chemistry, bipartite edge frustration is an indicator of the chemical stability of carbon structures known as fullerenes [12].

Calculating the frustration index is an NP-hard problem equivalent to the ground state calculation of spin glass model on unstructured graphs [4, 23]. Computation of the frustration index can also be reduced from the graph maximum cut (MAXCUT) problem, in a special case of all negative edges, which is known to be NP-hard [18].

For planar graphs MAXCUT can be solved in polynomial time [14]. The frustration index can also be computed in polynomial time for planar graphs [18, 21], which is equivalent to the ground state calculation of a two-dimensional spin glass model with no periodic boundary conditions and no magnetic field that is also solvable in polynomial time [10, 13]. In general graphs however, the frustration is even NP-hard to approximate within any constant factor [18]. The computational complexity of the problem might have played a role in the lack of systematic investigation of computing the frustration index while there are many studies on approximating it.

The frustration index can be approximated to a factor of $O(\sqrt{\log n})$ [2] or $O(k \log k)$ [3] where n is the number of vertices and k is the frustration index. Coleman et al. provides a review on the performance of several approximation algorithms of the frustration index [8]. Using a parametrised algorithmics approach, Hüffner, Betzler, and Niedermeier show that the frustration index (under a different name) is *fixed parameter tractable* and can be computed in $O(2^k m^2)$ [18] where m is the number of edges and k is the fixed parameter (the frustration index). The values of k we have observed in signed graphs inferred from the literature makes this approach impractical. There is no method suggested for computing the frustration index of large scale networks that guarantees the solution quality.

The principal focus of this research study is to provide insight into computing the frustration index. Besides multiple applications in various fields of research, another motivation for computing this measure is to systematically investigate signed networks transition to balance using basic graph operations on frustrated edges. Thus we develop an efficient algorithmic method for exact computation of the frustration index.

3 Preliminaries

3.1 Basic notation

We consider undirected signed networks $G = (V, E, \sigma)$. The set of nodes is denoted by V, with $|V| = n$. The set of edges is represented by E that is partitioned into the set of positive edges E^+ and the set of negative edges E^- with $|E| = m$, $|E^-| = m^-$, and $|E^+| = m^+$ where $m = m^- + m^+$. The sign function is denoted by $\sigma : E \to \{-1, +1\}^m$.

We represent the m undirected edges in G as ordered pairs of vertices $E = \{e_1, e_2, ..., e_m\} \subseteq \{(i, j) \mid i, j \in V, i < j\}$, where a single edge e_k between nodes i and j, $i < j$, is denoted by $e_k = (i, j), i < j$. The entries of the symmetric adjacency matrix $\mathbf{A} = (a_{ij})$ are defined in (1).

$$a_{ij} = \begin{cases} \sigma_{(i,j)} & \text{if } (i,j) \in E \text{ or } (j,i) \in E \\ 0 & \text{if } (i,j) \notin E \end{cases} \tag{1}$$

The number of positive (negative) edges connected to the node $i \in V$ represents positive (negative) degree of the node and is denoted by $d^+(i)$ $(d^-(i))$. The degree of node i is defined by $d(i) = d^+(i) + d^-(i)$.

A *walk* of length k in G is a sequence of nodes $v_0, v_1, ..., v_{k-1}, v_k$ such that for each $i = 1, 2, ..., k$ there is an edge from v_{i-1} to v_i. If $v_0 = v_k$, the sequence is a *closed walk* of length k. If the nodes in a closed walk are distinct except for the endpoints, it is a directed cycle (for simplicity *cycle*) of length k. The *sign* of a cycle is the product of the signs of its edges. Cycles with negative signs are unbalanced. A balanced cycle is one with positive sign. A balanced graph is one with no negative cycles.

3.2 Node colouring and frustration count

Satisfied and *frustrated* edges are defined based on colourings of the nodes. Colouring the nodes with black and white, a frustrated (satisfied) edge (i, j) is either a positive (negative) edge with different colours on the endpoints i, j or a negative (positive) edge with the same colours on the endpoints i, j.

To be more specific, for any signed graph $G = (V, E, \sigma)$, we can partition V into two sets, denoted $X \subseteq V$ and $\overline{X} = V \backslash X$. We call $X = (x_1, x_2, \ldots, x_n)$ the colouring set and we think of this partitioning as specifying a colouring of the nodes, where each node $i \in X$ is coloured black, and $i \in \overline{X}$ is coloured white. We let x_i denote the colour of node $i \in V$ under X, where $x_i = 1$ if $i \in X$ and $x_i = 0$ otherwise.

We define the *frustration count* $f_G(X)$ as the number of frustrated edges of G under X:

$$f_G(X) = \sum_{(i,j) \in E} f_{ij}(X)$$

where $f_{ij}(X)$ is the frustration state of edge (i, j) taking value 1 for a frustrated and values 0 for a satisfied edge.

The frustration index $L(G)$ of a graph G can be found by finding a subset $X^* \subseteq V$ of G that minimises the frustration count $f_G(X)$, i.e., solving Eq. (2). Note that $f_G(X)$ gives an upper bound on $L(G)$ for any $X \subseteq V$.

$$L(G) = \min_{X \subseteq V} f_G(X) \tag{2}$$

4 Mathematical programming models

In this section, we formulate four mathematical programming models in (3) – (6) to minimise the frustration count as the objective function.

4.1 An unconstrained binary quadratic programming model

We start with an objective function to minimise the frustration count. Note that the frustration of a positive edge (i,j) can be represented by $f_{ij} = x_i + x_j - 2x_ix_j \ \forall (i,j) \in E^+$ using the two binary variables x_i, x_j for the endpoint colours. For a negative edge, we have $f_{ij} = 1 - (x_i + x_j - 2x_ix_j) \ \forall (i,j) \in E^-$. This gives the UBQP model in (3) that calculates the frustration index in the minimisation objective function. The optimal solution represents a subset $X^* \subseteq V$ of G that minimises the frustration count. Note that the binary quadratic model in Eq. (3) has n decision variables and no constraints.

$$\begin{aligned} \min_{x_i : i \in V} Z = & \sum_{(i,j) \in E^+} x_i + x_j - 2x_ix_j + \sum_{(i,j) \in E^-} 1 - (x_i + x_j - 2x_ix_j) \\ \text{s.t.} \quad & x_i \in \{0,1\} \quad \forall i \in V \end{aligned} \tag{3}$$

The optimal value of the objective function in Eq. (3) is denoted by Z^* which represents the frustration index. The next three subsections address three binary linear programming models in (4), (5), and (6).

4.2 The AND model

Our first linear model is the AND model. As formulated in Eq. (4), the non-linear term x_ix_j in the objective function of Eq. (3) is replaced by additional binary variables $x_{ij} = x_i \text{AND} x_j$ for each edge (i,j) that take value 1 whenever $x_i = x_j = 1$ (both endpoints are coloured black) and 0 otherwise.

$$\begin{aligned} \min_{x_i : i \in V, x_{ij} : (i,j) \in E} Z = & \sum_{(i,j) \in E^+} x_i + x_j - 2x_{ij} + \sum_{(i,j) \in E^-} 1 - (x_i + x_j - 2x_{ij}) \\ \text{s.t.} \quad & x_{ij} \leq x_i \quad \forall (i,j) \in E^+ \\ & x_{ij} \leq x_j \quad \forall (i,j) \in E^+ \\ & x_{ij} \geq x_i + x_j - 1 \quad \forall (i,j) \in E^- \\ & x_i \in \{0,1\} \quad \forall i \in V \\ & x_{ij} \in \{0,1\} \quad \forall (i,j) \in E \end{aligned} \tag{4}$$

The dependencies between the x_{ij} and x_i, x_j values are taken into account using standard AND constraints. The AND model has $n + m$ variables and $2m^+ + m^-$ constraints. Note that x_{ij} variables are dependent variables because of the constraints. Therefore, we may drop the integrality constraint of the x_{ij} variables and consider them as continuous variables in the unit interval, $x_{ij} \in [0,1]$. The next subsection discusses an alternative binary linear model for calculating the frustration index.

4.3 The XOR model

Minimising frustration count can be directly formulated as a binary linear model. The XOR model is designed to directly count the frustrated edges using binary variables $f_{ij}\ \forall (i,j) \in E$. As before, we use $x_i\ \forall i \in V$ to denote the colour of a node. Our model is formulated by observing that the frustration state of a positive edge $(i,j) \in E^+$ is given by $f_{ij}(X) = x_i \text{XOR} x_j$. Similarly for $(i,j) \in E^-$, we have $f_{ij}(X) = 1 - x_i \text{XOR} x_j$.

Therefore, the minimum frustration count under all node colourings $(x_1, x_2, \ldots, x_n)$ is obtained by solving (5):

$$\begin{aligned}
\min_{x_i : i \in V, f_{ij} : (i,j) \in E} \quad & Z = \sum_{(i,j) \in E} f_{ij} \\
\text{s.t.} \quad & f_{ij} \geq x_i - x_j \quad \forall (i,j) \in E^+ \\
& f_{ij} \geq x_j - x_i \quad \forall (i,j) \in E^+ \\
& f_{ij} \geq x_i + x_j - 1 \quad \forall (i,j) \in E^- \\
& f_{ij} \geq 1 - x_i - x_j \quad \forall (i,j) \in E^- \\
& x_i \in \{0,1\} \quad \forall i \in V \\
& f_{ij} \in \{0,1\} \quad \forall (i,j) \in E
\end{aligned} \tag{5}$$

The dependencies between the f_{ij} and x_i, x_j values are taken into account using two standard XOR constraints per edge. Therefore, the XOR model has $n + m$ variables and $2m$ constraints. Note that f_{ij} variables are dependent variables because of the constraints and the objective function pressure. Therefore, we may specify f_{ij} variables as continuous variables in the unit interval, $f_{ij} \in [0,1]$. A third linear formulation of the problem is provided in the next subsection.

4.4 The ABS model

In this section, we propose the ABS model, a binary linear model in which two edge variables are used to represent the frustration state of an edge.

We start by observing that for a node colouring $(x_1, x_2, \ldots, x_n)$, $|x_i - x_j| = 1$ for a positive frustrated edge and $|x_i - x_j| = 0$ for a positive satisfied edge $(i,j) \in E^+$. Similarly, $1 - |x_i - x_j| = |x_i + x_j - 1|$ gives the frustration state of a negative edge $(i,j) \in E^-$.

To model the absolute value function, we introduce additional binary variables $e_{ij}, h_{ij} \in \{0,1\}$. We observe that for a positive edge if $x_i - x_j = e_{ij} - h_{ij}$ then $|x_i - x_j| = e_{ij} + h_{ij}$. Similarly, for a negative edge $(i,j) \in E^-$ if $x_i + x_j - 1 = e_{ij} - h_{ij}$ then $|x_i + x_j - 1| = e_{ij} + h_{ij}$. This allows us to formulate the linear model in (6).

The objective function, being the total number of frustrated edges, sums the above-mentioned absolute value terms to compute the frustration count in (6). The conditions observed for positive and negative edges are expressed as linear constraints in (6). Therefore, the ABS model has $n + 2m$ variables and m constraints.

$$\begin{aligned} \min_{x_i:i\in V, e_{ij}, h_{ij}:(i,j)\in E} Z = \sum_{(i,j)\in E} e_{ij} + h_{ij} & \\ \text{s.t.} \quad x_i - x_j = e_{ij} - h_{ij} \quad \forall (i,j) \in E^+ & \\ x_i + x_j - 1 = e_{ij} - h_{ij} \quad \forall (i,j) \in E^- & \\ x_i \in \{0,1\} \quad \forall i \in V & \\ e_{ij} \in \{0,1\} \quad \forall (i,j) \in E & \\ h_{ij} \in \{0,1\} \quad \forall (i,j) \in E & \end{aligned} \tag{6}$$

4.5 Comparison of the four models

In this subsection we compare the four models introduced above, based on the number and type of constraints. Table 1 summarises the comparison results.

Table 1: Comparison of the variables and constraints in the four models

	UBQP (3)	AND (4)	XOR (5)	ABS (6)
Variables	n	$n+m$	$n+m$	$n+2m$
Constraints	0	$2m^+ + m^-$	$2m$	m
Constraint type	-	linear	linear	linear
Objective	quadratic	linear	linear	linear

As Table 1 shows, the number of variables and constraints differ in the four models. Eq. (7) shows that the three linear models are mathematically equivalent.

$$f_{ij} = e_{ij} + h_{ij} = (1 - a_{ij})/2 + a_{ij}(x_i + x_j - 2x_{ij}) \tag{7}$$

However, the three linear models perform differently in terms of solve time and the number of branch and bound (B&B) nodes required to solve a given problem. We continue to techniques of improving the performance of linear models in the next section.

5 Speed-up techniques

In this section we discuss techniques to speed up the branch and bound algorithm for solving the binary linear models. To improve the branch and bound, one may use certain conditions based on the structure of the binary programming problem [5]. Two techniques often deployed in solving Interger Programming (IP) models are valid inequalities and branching priority.

We implement some valid inequalities as additional non-core constraints that are kept aside from the core constraints of the model. Such implementation of additional restrictions is referred to as lazy constraints [22]. The branch and bound algorithm can be provided with a list of prioritised variables for branching which may speed up the solver if the prioritised list is more effective in making integer values. The solve time improvement evaluation is based on 100 random graphs with $n = 30, m = 300$, and $m^- = 150$.

5.1 Branching priority and fixing a colour

In binary programming models, the root node solution is integral if the constraint matrix is unimodular and the right hand side vector is integral. However, most practical optimisation problems, including the problem under investigation, do not have such matrices [5] resulting in a fractional root node solution.

In order to speed up the algorithms, we consider adding a valid inequality to increase the root node objective function. In the XOR and ABS models, we observe that there always exists a fractional solution of $x_i = 0.5 \quad \forall i \in V$ which gives an optimal root node objective function value of 0.

We can increase the root node objective by fixing one node variable which breaks the symmetry that exists and allows changing all node colours to give an equivalent solution. While in the AND model the fractional solution is different, the root node objective function can still be increased following the same process. We conjecture that the best node variable is the one associated with the highest degree. This constraint is formulated in (8).

$$x_k = 1 \quad k = \arg\max_{i \in V} d_i \tag{8}$$

In our experiments, we always observed an improvement in the root node objective value when (8) was added, which shows it is useful.

Based on the same idea, we may modify the branch and bound algorithm so that it branches first on the node with the highest degree. This modification is implemented by specifying a branching priority for the node variables in which variable x_i has a priority given by its degree d_i.

Our experiments on random graphs show that fixing a colour and using prioritised branching lead to 65%, 70%, and 43% reduction in the average solve time of AND, XOR, and ABS models respectively.

5.2 Unbalanced triangle valid inequalities

The structural properties of the problem allow us to restrict the model by adding valid inequalities as additional constraints [5]. Structural properties of signed graphs can be used to determine valid inequalities.

In this subsection, we consider adding one inequality for each unbalanced cycle of length 3 (triangle) in the graph. Every unbalanced cycle of the graph contains an odd number of frustrated edges. This means that any colouring of the nodes in an unbalanced triangle must produce at least one frustrated edge. Recalling that under a colouring the variable f_{ij} is 1 if edge (i, j) is frustrated (and 0 otherwise), then for any node triple (i, j, k) defining an unbalanced triangle in G, we have the inequality (9) which is valid for all feasible solutions of the problem.

$$f_{ij} + f_{ik} + f_{jk} \geq 1 \quad \forall (i, j, k) \in T^- \tag{9}$$

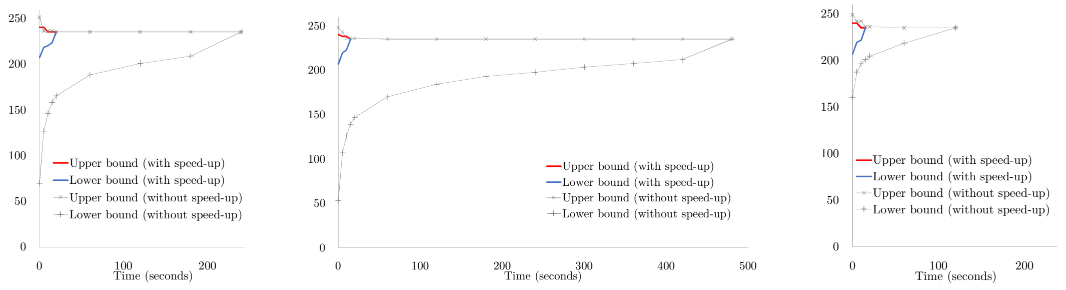
In (9), $T^- = \{(i, j, k) \in V^3 \mid \sigma_{(i,j)}\sigma_{(i,k)}\sigma_{(j,k)} = -1\}$ denotes the set of node triples that define an unbalanced triangle. The expression in inequality (9) denotes the sum of frustration states for the three edges $(i, j), (i, k), (j, k)$ making an unbalanced triangle. Note that in order to implement the unbalanced triangle valid inequality (9), f_{ij} must be represented using the decision variables in the particular model. Eq. (7) shows how f_{ij} can be defined in the AND and ABS models.

We observe an improvement in the root node objective when the unbalanced triangle valid inequalities are added to the models which shows they are useful. From a solve time perspective, our experiments on random graphs show that implementing this speed-up technique leads to 65% and 87% reduction in the average solve time of AND and XOR models respectively. For the ABS model the solve time improvement is very small ($< 5\%$).

5.3 Overall improvement made by the speed-up techniques

In this section, various random signed networks are solved by our optimisation models using Gurobi version 7.0.2 on a desktop computer with an Intel Corei5 4670 @ 3.40 GHz and 8.00 GB of RAM running 64-bit Microsoft Windows 7. The models were created using the Gurobi Python environment in Anaconda 3-4.2.0 Jupyter.

We discussed the solve time improvement made by the individual implementation of the speed-up techniques on the binary linear models in the Section 5. According to the same analysis, the total solve time reduction observed when all speed-up techniques are implemented is 68% for the AND model, 90% for the XOR model, and 45% for the ABS model. Figure 1 shows the upper and lower bounds in solving the three linear models with and without the speed-up techniques for a random graph with $n = 40, m = 620, m^- = 434$.



(a) The AND model (b) The XOR model (c) The ABS model

Figure 1: The impact of speed-up techniques on the solve time and the upper and lower bounds for a random graph with $n = 40, m = 620, m^- = 434$ (colour version online)

6 Evaluating performance against the literature

In this section, we use both random and real networks to evaluate not only the solve time, but also the solution quality of our models against other methods in the literature.

6.1 Solve time in random graph

In this subsection, we compare the solve time of our algorithm against all other exact algorithms suggested for computing the frustration index. Our review of the literature finds only two exact methods capable of computing the frustration index [6, 18].

Brusco and Steinley have reported running times for very small graphs with only up to $n = 21$ vertices. While, their exact algorithm fails to solve graphs as large as $n = 30$ in a reasonable time [6], our binary linear models solve such instances in split seconds.

Hüffner, Betzler, and Niedermeier have generated random graphs by specifying n, degree distribution, clustering coefficient, and the percentage of negative edges [18]. The largest of such random graphs solvable by their algorithm in 20 hours has $n = 500$ nodes. They also reported that only 3 out of 5 random graphs with $n \in \{100, 200, 300, 400, 500\}$ can be solved by their method in 20 hours. Our XOR model solves all such instances in less than 100 seconds.

6.2 Solve time and solution quality in real networks

In this section we use signed network datasets from biology and international relations. The frustration index of biological networks has been a subject of interest to measure the network distance to *monotonicity* [9, 19]. In this section, the frustration index is computed in real biological networks by solving the XOR model (6) coupled with the speed-up techniques.

There are four signed biological networks analysed by [9] and [19]. The epidermal growth factor receptor (EGFR) pathway is a signed network with 779 edges. The molecular interaction map of a white blood cell (macrophage) is another well-studied signed network containing 1425 edges. We also investigate two gene regulatory networks, related to two organisms: a eukaryote, the yeast Saccharomyces cerevisiae (yeast), and a bacterium: Escherichia coli (E.coli). The yeast and E.coli networks have 1080 and 3215 edges respectively. Figure 2 shows the four biological signed networks. The colour of edges correspond to the signs on the edges (green for +1 and red for −1). For more details on the four biological datasets, one may refer to [19].

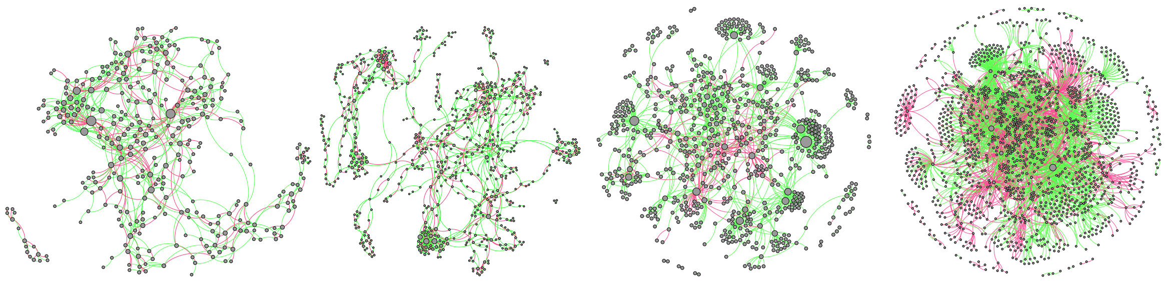

(a) EGFR (b) Macrophage (c) Yeast (d) E.coli

Figure 2: Four biological signed networks visualised using Gephi (colour version online)

As the signed graph frustration problem has been previously investigated mainly using heuristic and approximation algorithms, we compare the solution quality and solve time of our exact algorithm with such techniques as well. Various performance measures for the XOR model (6) solving real signed networks are illustrated in Table 2.

DasGupta et al. have suggested approximation algorithms [9] that are later tested on the four biological networks by [18]. Their approximation method provides $196 \leq L(G)_{\text{EGFR}} \leq 219$ which our exact model proves to be incorrect. The bounds obtained by implementing DasGupta et al. approximation are not incorrect for the other three networks, but they have very large gaps between lower and upper bounds.

Table 2: Performance of the XOR model (6) tested on large biological signed networks

Graph	$L(G)$	Root node objective	Number of B&B nodes	Solve time (s)
EGFR	193	15.5	3	0.28
macrophage	332	15.0	64	0.56
yeast	41	11.5	3	0.13
E.coli	371	127.5	30	2.21

Hüffner, Betzler, and Niedermeier have previously investigated frustration in the four biological networks suggesting a data reduction scheme and an exact algorithm [18]. Their suggested data reduction scheme can take more than 5 hours for yeast, more than 15 hours for EGFR, and more than 1 day for macrophage if the parameters are not perfectly tuned. Besides the solve time issue, their exact algorithm provides $L(G)_{\text{EGFR}} = 210, L(G)_{\text{macrophage}} = 374$, both of which are incorrect. They report their algorithm failed to terminate for E.coli [18].

Iacono et al. have also investigated frustration in the four networks [19]. Their heuristic algorithm provides upper and lower bounds for EGFR, macrophage, yeast, and E.coli with 96.37%, 90.96%, 100%, and 98.38% ratio of lower to upper bound respectively. Regarding solve time, they have only mentioned that their heuristic requires a fairly limited amount of time (a few minutes on an ordinary PC).

Table 3 sums up the solution quality and solve time comparison of our suggested model against the literature. We compare our solve times against the best times reported in the previous works.

Table 3: Comparison of the solution quality and solve time against the literature

	Graph	DasGupta et al. [9]	Hüffner et al. [18]	Iacono et al. [19]	XOR
Quality	EGFR	[196, 219]	210	[186, 193]	193
	macrophage	[218,383]	374	[302, 332]	332
	yeast	[0, 43]	41	41	41
	E.coli	[0, 385]	Not converged	[365, 371]	371
Time	EGFR	420 s	6480 s	>60 s	0.28 s
	macrophage	2640 s	60 s	>60 s	0.56 s
	yeast	4620 s	60 s	>60 s	0.13 s
	E.coli	Not reported	Not converged	>60 s	2.21 s

While data reduction schemes [18] can take up to 1 day for these datasets and heuristic algorithms [19] only provide bounds with up to 9% gap from optimality, our XOR model equipped with the speed-up techniques solves the 4 datasets to optimality in a few seconds. Note that the AND and ABS models also solve the datasets to optimality, but their executions take longer (still less than a minute when coupled with the speed-up techniques).

7 Conclusion

In this study, we provided a novel method for computing a standard measure in signed graphs which has many application in different disciplines. The present study suggested efficient mathematical programming models and speed-up techniques for solving an NP-hard graph optimisation problem. We develop four binary optimisation models which contrary to the current methods in the literature come with a guarantee of solution quality. Then we suggest prioritised branching and valid inequalities which provide up to 90% solve time improvement for our linear optimisation models.

The speed-up techniques make the models capable of processing relatively large graphs on an inexpensive computer. Solve time and solution quality comparison to the literature proves the superiority of our models tested on both random and real signed graphs. Considering the recent developments in the area of quantum computing and the ever-increasing size of practical instances of the problem, a possibility for future investigation is applying more advanced computing technology for solving the UBQP formulation of the problem or analysing the boolean quadric polytope using a polyhedral approach.

References

[1] R. P. Abelson and M. J. Rosenberg. Symbolic psycho-logic: A model of attitudinal cognition. *Behavioral Science*, 3(1):1–13, Jan. 1958.

[2] A. Agarwal, M. Charikar, K. Makarychev, and Y. Makarychev. O(log n) approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 573–581, New York, NY, USA, 2005. ACM.

[3] A. Avidor and M. Langberg. The multi-multiway cut problem. *Theoretical Computer Science*, 377(1):35 – 42, 2007.

[4] F. Barahona. On the computational complexity of Ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241, 1982.

[5] A. Bilitzky and A. Sadeh. Efficient solutions for special zero-one programming problems. *Journal of Combinatorial Optimization*, 10(3):227–238, Nov 2005.

[6] M. Brusco and D. Steinley. K-balance partitioning: An exact method with applications to generalized structural balance and other psychological contexts. *Psychological Methods*, 15(2):145–157, 2010.

[7] D. Cartwright and F. Harary. Structural balance: a generalization of Heider's theory. *Psychological Review*, 63(5):277–293, 1956.

[8] T. Coleman, J. Saunderson, and A. Wirth. A local-search 2-approximation for 2-correlation-clustering. In *European Symposium on Algorithms*, pages 308–319. Springer, 2008.

[9] B. DasGupta, G. A. Enciso, E. Sontag, and Y. Zhang. Algorithmic and complexity results for decompositions of biological networks into monotone subsystems. *Biosystems*, 90(1):161–178, 2007.

[10] C. De Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt, and G. Rinaldi. Exact ground states of Ising spin glasses: New experimental results with a branch-and-cut algorithm. *Journal of Statistical Physics*, 80(1):487–496, Jul 1995.

[11] P. Doreian and A. Mrvar. Structural Balance and Signed International Relations. *Journal of Social Structure*, 16:1–49, 2015.

[12] T. Doslic and D. Vukicevic. Computing the bipartite edge frustration of fullerene graphs. *Discrete Applied Mathematics*, 155(10):1294–1301, May 2007.

[13] N. G. Fytas, P. E. Theodorakis, and A. K. Hartmann. Revisiting the scaling of the specific heat of the three-dimensional random-field Ising model. *The European Physical Journal B*, 89(9):200, 2016.

[14] F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing*, 4(3):221–225, 1975.

[15] F. Harary. On the measurement of structural balance. *Behavioral Science*, 4(4):316–323, Oct. 1959.

[16] F. Harary, M.-H. Lim, and D. C. Wunsch. Signed graphs for portfolio analysis in risk management. *IMA Journal of Management Mathematics*, 13(3):201–210, Jan. 2002.

[17] F. Heider. Social perception and phenomenal causality. *Psychological Review*, 51(6):358–378, 1944.

[18] F. Hüffner, N. Betzler, and R. Niedermeier. Separator-based data reduction for signed graph balancing. *Journal of Combinatorial Optimization*, 20(4):335–360, 2010.

[19] G. Iacono, F. Ramezani, N. Soranzo, and C. Altafini. Determining the distance to monotonicity of a biological network: a graph-theoretical approach. *Systems Biology, IET*, 4(3):223–235, 2010.

[20] P. W. Kasteleyn. Dimer Statistics and Phase Transitions. *Journal of Mathematical Physics*, 4(2):287–293, Feb. 1963.

[21] O. Katai and S. Iwai. Studies on the balancing, the minimal balancing, and the minimum balancing processes for social groups with planar and nonplanar graph structures. *Journal of Mathematical Psychology*, 18(2):140–176, 1978.

[22] E. Klotz and A. M. Newman. Practical guidelines for solving difficult mixed integer linear programs. *Surveys in Operations Research and Management Science*, 18(1):18–32, 2013.

[23] D. Sherrington and S. Kirkpatrick. Solvable model of a spin-glass. *Physical Review Letters*, 35:1792–1796, Dec 1975.

[24] T. Zaslavsky. Balanced decompositions of a signed graph. *Journal of Combinatorial Theory, Series B*, 43(1):1–13, 1987.