Towards a Broader View of the Theory of Computing

Narendra Karmarkar

Part - 1, (A6), 12/11/14, 6:30pm

now available in arXiv

Part - 2, (A4), 12/13/14, 5:00pm

Part - 3, (B2), 12/16/14, 4:00pm

Part - 4, (B3), 12/17/14, 3:00pm

Models of Computation

- A model of computation provides a **mathematical abstraction** of the basic **data objects** and **operations** on them provided as building blocks by the physical hardware.

- This effectively decouples
  design of algorithms for complex tasks
  from
  implementation details of the basic primitives in the underlying hardware.

- Examples:
  - **Turing Model**
    uses strings of zeroes and ones and finite state machines as primitives.
  - **BSS model** (Blum, Shub, Smale)
    model uses real numbers or complex numbers as data objects and algebraic operations (including comparisons in real case) as basic operations.

Various types of applications - discrete and continuous

- Many business and IT applications of computing use discrete models.
- There are many applications based on continuous models
  - Numerical simulation of natural phenomena or engineered systems based on differential equations
  - Estimation of probabilities, assessing strength of association between various entities in social networks
  - Interior point algorithms in optimization which are based on embedding discrete configurations in a multidimensional continuous space and constructing trajectories converging to the solution

A model of computation is used to map these on physical machines

Examples of physical computers

- Standard digital computers
- Analog computers explored in the past e.g. Shannon's differential analyzer
- Quantum computers
- Biological systems that appear to do information processing / computing
- Computers partially mimicking some aspects of those systems e.g. neural networks, neuromorphic computing, etc.

Early Pioneers views on the need for broader approach to computation

Alan Turing

was interested in understanding the origin of intelligence in biological systems

He believed it is due to some form of computing happening in these systems.

There doesn't seem to be anything similar to Turing machines in these systems.

Many years after developing the discrete model of computation,

Turing started exploration of PDE's modelling biological functions

These are clearly continuum based models of biological systems

Von Neumann

Defined architecture of a digital computer using many ideas from TM

However, he was particularly critical of the limitations imposed on the theory of automata by its foundations in formal logic, combinatorics.

He articulated the need for a detailed, highly mathematical, and more specifically, analytical theory of computation based on mathematical analysis

Shannon

worked on analog computer - differential analyzer

However, direct implementation of differentiation in analog system is highly error prone as it involves subtraction of two nearly equal continuous quantities.

constructing analog computers

- **Role of Integral Transform**
  - It's well known that for non-linear DE's which are difficult to solve numerically, it helps to convert them into equivalent integral equations.
  - Biological systems also seem to use integral transform. e.g.

    Human ear computes the magnitude of Fourier Transform of speech signal.

    The transform of the derivative is then obtained by scaling,
    a simpler algebraic operation than differentiation in analog setting
  - Some optical computers are implementing Fourier transform directly for computing differential operators required in computational fluid dynamics.
- **Robustness**
  - In digial systems, the mapping
    Physical states / signals $\longrightarrow$ information states/signals
    is many to one, In fact, each information state has infinite pre images.
  - This is to gain robustness in the face of small variations in physical states, due to noise, thermal effects, variations in manufacturing etc.
  - However, this does not necessarily require information states to be discrete.
    e.g. an ideal low pass filter also provides infinity to one mapping
    but the output signal still belongs to the continuum.

it is important to include these two mechanisms when constructing machines that support continuum computing

Intertwining of discrete and continuous models

Logical and physical process underlying the computation involves several levels of abstractions, both continuous and discrete.

Continuous models	**Discrete models**
Solution algorithm for a non-linear DE	
	Floating point arithmetic gives discrete approximation of real numbers for mapping onto Turing machine model.
The computer implementing TM is a network of transistors. As a circuit, it is modeled by non-linear DE. Restricted interpretation is applied to continuous quantities. e.g. $V > V_{Thr}$ represents logical 1. Continuous time is discretized into clock cycles bigger than settling time of the transient solution to DE	
	This model is approximating underlying discrete phenomenon – Fluid like model for the statistical mechanics of large number of discrete particles (electrons and holes) -- Continuous energy bands representing large number of closely spaced quantized energy levels
Mathematical model for quantum mechanical phenomenon uses Hilbert space which is based on the continuum concept, and so is more recent string theory .	

Towards a broader view of theory of computing

- Given this intertwining of discrete and continuous models from top to bottom, it would be more illuminating to take a broader view
- Exploration of what a continuum view of computing might suggest for algorithms, relative difficulty of various computing tasks etc.

Abstract Continuum Computing Model AC^2M or just CM

- **Basic data objects**
  algebraic closure of meromorphic functions (over suitable domain)
  the machine can store, evaluate and compose such function objects.
- **Basic unit operations**
  - **field operations**: $+, -, \times, \div$
    and comparison ($=$, (and also $<$ for real quantities))
  - **analytic operations**

    integration: $\int_C f(z)dz$ this is a binary operation, function f and specification contour C are the two operands

    differentiation: $\frac{\partial f}{\partial z_i}$ is also a binary operation with inputs f and z_i.

    (a word of caution when comparing with TM - there is no such thing as "conservation of difficulty" across the models.)

Extension of $P \neq NP$ conjecture from TM theory

- Computing models can be organized in the same way as Cantor organized infinite sets i.e. according to the cardinal number
  of set of all possible data objects and machines in the model.
- At each cardinal number in the sequence $\beth_0 < \beth_1 < \beth_2 < \ldots$, you have models of computation, and corresponding $P \neq NP$ question

$$\begin{array}{lcl} TM & \longrightarrow & \beth_0, cardinalnumber(\mathbb{Q}) \\ CM & \longrightarrow & \beth_1, cardinalnumber(\mathbb{R}) \end{array}$$

- It appears that $P \neq NP$ problem for TM is just the first member in a sequence of strict inclusions

$$P(TM) \subsetneq NP(TM) \subsetneq P(CM) \subsetneq NP(CM) \subsetneq P(\beth_2) \ldots$$

- An interesting question across adjacent level:
  $NP(TM) \subsetneq P(CM)$

  i.e., is non-determisitic computing at any one level is no more powerful than deterministic computing at the next level ?

Our Research Program

Is aimed at understanding the following:

- how to construct physical machines supporting Continuum Computing ?
- Can non-deterministic computing at the TM level be simulated by deterministic computing at next level i.e. by Continuum Computing ?

  and a harder question:

- to what extent can one approximate deterministic Continuum Computing by deterministic computing in TM?
- initially, approximate cross-simulation was meant to be "stop-gap" measure, but now it appears that building continuum machine may take many years, hence simulation becomes more important.
- Floating Point numbers allows approximation of reals by rationals
- similarly one can approximate functions by other simpler functions
- in this investigation, length of "binary encoding" of data object does not have the same fundamental significance as in TM theory. Instead, other properties of the problem space seem more important

Computationally generated proofs for optimality and non-satisfiability

Converting results of continuum based algorithms so that optimality or non-existence of solutions can be transferred to standard model.

Both involve proofs of non-negativity of functions.

- **Optimality**:

  Proving that $\mathbf{x}_{min}$ is a global minimum of $f(x_1, x_2, \dots, x_n)$ is equivalent to showing that for $f_{min} = f(\mathbf{x}_{min})$, we have

$$f(\mathbf{x}) - f_{min} \geq 0 \qquad \forall \mathbf{x} \in \mathbb{R}^n$$

- **Non-satisfiability**: Each variable $x_i = \pm 1$
  let V denote the variety defined by $x_i^2 = 1 \forall i$
  For each clause associate a polynomial. e.g.
  $C = x_i \vee \bar{x_j} \vee x_k$, associated $p_c = [1 - x_i] \cdot [1 + x_j] \cdot [1 - x_k]$
  For a ± 1 vector $\mathbf{x}$ $p_c = \begin{cases} 0 & \text{if } \mathbf{x} \text{ is a satisfying assignment} \\ 8 & \text{otherwise} \end{cases}$
  Define $f = \sum_{C \in \text{Clauses}} p_c$.
  Then $f(\mathbf{x}) > 0 \quad \forall \mathbf{x} \in V$ gives proof of non-satisfiability.

Relation of Positivity Proofs to Hilbert's 17th problem

- One approach to proving positivity of a polynomial is to express it as sums of squares of other functions
- **Using polynomials**:
  Hilbert (1888) realized that it is **not** always possible , but gave a **non-constructive** proof of the existence of a counter example.
- **Using rational functions**:
  Always possible – **Artin's** (1926) solution of Hilbert's 17th problem.
  But exponentially many terms of high degree required (**Pfister**).
- **Concrete examples** of Hilbert's result took long time to construct.
  First counter example - Motzkin

  $$f(x, y, z) = z^6 + x^4y^2 + x^2y^4 - 3x^2y^2z^2$$

  Further examples - Lam, Robinson and others.
- **Computational results**:
  for these "counterexamples", we have computed expressions as sums of squares of polynomials, with a modified interpretation.
  Algorithm underlying the solver was described in previous lecture [Karmarkar, MIT91, IPCO92]

Computational approach to positivity proofs

- Consider a homogeneous polynomial $f(x_1, x_2, \ldots, x_n)$ of degree d in n real variables $x_1, x_2, \ldots, x_n$, which is non-negative everywhere.
- To show that a function is non-negative on a compact set, it is enough to show that the function is **non-negative** at all its **critical points**. (note we have homogeneous polynomials over projective space).
- This can be achieved if we
  - (1) construct a **variety containing all critical points** of the function and
  - (2) construct an expression of the function as sums of squares of polynomials **in the coordinate ring of that variety**, instead of the polynomial ring $\Re(x_1, \ldots, x_n)$.
- Additionally, without loss of generality, we impose a spherical constraint
$$x_1^2 + x_2^2 + \ldots + x_n^2 = 1 \tag{1}$$
- Each point in the real projective space is covered twice (by a pair of antipodal points) in this representation.
- A critical point of the function satisfies
$$\frac{\partial f}{\partial x_i} = \lambda x_i, \quad i = 1, ..., n \tag{2}$$

Computational approach to positivity proofs (Contd.)

- We now work in $\Re^{n+1}$, and points in this expanded space will be denoted by $(x_1, x_2, ..., x_n, \lambda)$
- The equations (1) and (2) define an algebraic variety U (i.e an algebraic set – in our terminology a variety does not have to be irreducible).
- Our approach is to construct a variety V such that $U \subseteq V \subseteq \Re^{n+1}$ and in the co-ordinate ring of V, construct an expression for f as sums of squares

$$\sum_i S_i^2(x_1, x_2, \ldots, x_n, \lambda)$$

- We also produce an explicit expression for

$$f \ - \ \sum_i S_i^2 \ (x_1, x_2, \ldots, x_n, \lambda)$$

as $\quad \sum_j \ a_j \ (x, \lambda) \ g_j \ (x_1, x_2, \ldots, x_n, \lambda)$

where $\quad g_j \ (x_1, x_2, \ldots, x_n, \lambda) \ = \ 0$

These are the defining equations for V.

Our computer generated proofs

Motzkin's first counter example

$$\begin{aligned}
f(x,y,z) &= z^6 + x^4y^2 + x^2y^4 - 3x^2y^2z^2 \\
f &\equiv \left(\frac{1}{4}S_1^2 + S_2^2 + S_3^2 + S_4^2 + \frac{3}{4}S_5^2\right) \text{ mod } V \\
S_1 &= xy(x^2 - y^2) \\
S_2 &= x^4 + y^4 - 2x^2 - 2y^2 + x^2y^2 + 1 \\
S_3 &= xz(x^2 + 2y^2 - 1) \\
S_4 &= yz(2x^2 + y^2 - 1) \\
S_5 &= xy(3x^2 + 3y^2 - 2)
\end{aligned} \tag{3}$$

Robinson's counter example

$$\begin{aligned}
f &= x^6 + y^6 + z^6 - (x^4y^2 + x^2y^4 + y^4z^2 + y^2z^4 + z^4x^2 + z^2x^4) + 3x^2y^2z^2 \\
f &\equiv \left(S_1^2 + \frac{3}{4}S_2^2 + \frac{1}{4}S_3^2 + S_4^2 + S_5^2\right) \text{ mod } V \\
S_1 &= -x^3y + xy^3 \\
S_2 &= -1 + 3x^2 - 2x^4 - 4x^2y^2 + 2y^2 \\
S_3 &= 1 - x^2 - 2x^4 + 4x^2y^2 - 4y^2 + 4y^4 \\
S_4 &= -2x^3z - xy^2z + xz \\
S_5 &= -x^2yz - 2y^3z + yz
\end{aligned}$$

Extensions of previous proof systems

Rules for positivity proofs

1. **Starting primitives**:
    1. **Constant functions**
       Let $\alpha \in \mathbb{R}$ be a positive scalar.
       If $f(x) = \alpha$, then $f > 0$.
    2. **Square functions**
       If $f(x) = g^2(x)$, then $f \geq 0$.
2. **Algebraic operations preserving positivity**:
   $$\text{If } f \geq 0 \text{ and } g \geq 0, \text{ then } f + g \geq 0$$
   $$\text{If } f \geq 0 \text{ and } g \geq 0, \text{ then } f \cdot g \geq 0$$
3. **Positivity restricted to variety defining constraints**:
   Suppose there are integrality constraints, $x_i^2 = 1$, or other constraints like $A\mathbf{x} = \mathbf{b}$ etc. They define an algebraic set or subvariety V of $\mathbb{R}^n$. Algebraic operations are defined upto equivalence classes of functions on V.

Lovaśz, Schrijver, Grigoriev, Worah, and references therein, have explored rules of this kind along with rules for cuts based on integrality constraints.

New rules

We strengthen the previous proof systems by adding the following new rules.

1. **Substitution or Composition rule**:

   Let $f(x_1, \ldots, x_n)$ and $g_i(y_1, \ldots, y_m)$ for $1 \leq i \leq n$ be given.
   Let $h(y_1, \ldots, y_m) = f(g_1(y_1, \ldots, y_m), \ldots, g_n(y_1, \ldots, y_m))$.

   1. If $f(x_1, \ldots, x_n) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n$, then $h(y_1, \ldots, y_m) \geq 0$.
   2. If $f(x_1, \ldots, x_n) \geq 0$ for all $\mathbf{x} \in \mathbb{R}^n_+$ and $g_i(y_1, \ldots, y_m) \geq 0$ for all $\mathbf{y} \in \mathbb{R}^m$, then $h(y_1, \ldots, y_m) \geq 0$.

2. **Division rule**: While number of rational terms in Artin's approach can be exponential, one can apply the following rule selectively ::
   Suppose $g(x) > 0, f(x) \geq 0$ and $g(x) \mid f(x)$.

   $$\text{If } f(x) = g(x) \cdot h(x), \text{ then } h(x) \geq 0.$$

   This rule reduces the degree.

3. **Odd Radical rule**:
   Suppose $f(\mathbf{x})$ is a perfect odd ($(2k+1)^{\text{th}}$) power. Let $g(\mathbf{x})$ be the $(2k+1)^{th}$ root of $f(\mathbf{x})$, i.e. $f(\mathbf{x}) = g^{2k+1}(\mathbf{x})$.

   $$\text{If } f(\mathbf{x}) \geq 0, \text{ then } g(\mathbf{x}) \geq 0.$$

   This is one of the rare rules that reduces the degree.
   Note :Lower bounds based on proof system w/o these rules don't show "intrinsic" difficulty of any problem, but only show "intrinsic" limitations of the proof system used

Maximum Independent Set

- Let $G = (V, E)$ be a graph with vertex set V and edge set E.
- A subset $U \subseteq V$ of nodes is called *independent* if for every edge $(i, j) \in E$, either $i \notin U$ or $j \notin U$
- Problem: Find the largest independent set.
- ± 1 integer programming formulation
  Constraints:
  $$\begin{array}{rcll} w_i & = & \pm 1; & i = 1, \ldots, n \\ w_i + w_j & \leq & 0; & \forall (i, j) \in E \end{array}$$
  Objective: Maximize $\sum w_i$
- Relaxation of the feasible set allows $-1 \leq w_i \leq 1$
- Each extreme point of an LP has co-ordinates that are ± 1 or 0.
- For a ± 1 solution, $\sum_i w_i^2 = n$.
- Non-convex objective: $\sum w_i + \beta \sum w_i^2$

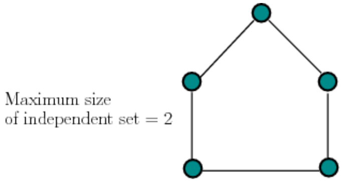
Defining a sharper polytope

There are additional inequalities

- that hold for the solution points, i.e. feasible points with ± 1 solution
- but cannot be expressed as non-negative linear combinations of basic inequalities
- number of such inequalities grows exponentially with n
- significant effort in polyhedral combinatorics is aimed at identifying such constraints (e.g. see Schrijver).

Example: odd cycle inequalities

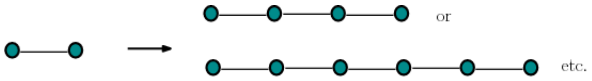
- Consider an odd cycle in the graph, having $2k + 1$ vertices. At most k of the $2k + 1$ vertices can be in an independent set.



- Let $i_1, i_2, \ldots, i_{2k+1}$ be the vertices in the cycle; hence $w_{i_1} + w_{i_2} + \ldots + w_{i_{2k+1}} \leq -1$.
- This inequality is sharper, since the previous inequalities only imply : $w_{i_1} + w_{i_2} + \ldots + w_{i_{2k+1}} \leq 0$.
- Number of odd cycles, and hence the corresponding number of inequalities, can grow exponentially with n.

Inequalities for other subgraphs

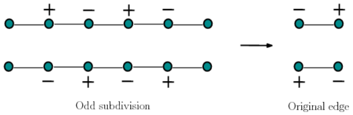
- Similar to odd cycles, there are constraints based on other types of subgraphs.
- Let
$$\sum_{i \in G_0} a_i \cdot w_i \leq b$$
be a constraint based on a fixed graph G_0.
- Let $\widetilde{G}$ be a graph obtained by an odd sub-division of the edges of G_0, i.e. replace an edge by an odd path.



- Then sum of newly added vertices is always non-positive, i.e. $\sum_{i \in \text{New vertex}} w_i \leq 0$.

Inequalities for other subgraphs (Contd.)

- Furthermore, to achieve the equality, $\sum_{i \in \text{New vertex}} w_i = 0$,there are only two ways to assign values to new vertices and each corresponds to a particular feasible assignment to the end points of the original edge.



- Using this observation, it is easy to show that the inequality for G_0 implies a similar inequality for $\widetilde{G}$, i.e.

$$\sum_{i \in G_0} a_i \cdot w_i \leq b \quad \Rightarrow \quad \sum_{i \in \widetilde{G}} \widetilde{a}_i \cdot w_i \leq b$$

- In this context, we are using a more restricted notion of homeomorphism.
- Specifically, in order for two paths to be homeomorphic both must have odd length or both must have even length.
- We call this "parity-respecting" homeomorphisms, and denote it by

$$\widetilde{G} \cong G_0 (\text{mod} 2)$$

(different authors use different terminology/notation, e.g. see [])

Inequalities for other subgraphs (Contd.)

- Given a fixed graph G_0 and an input graphs G,
  an inequality for $G_0 \Rightarrow$ an inequality for each subgraph $\widetilde{G}$ of G that is homeomorphic to G_0.
- Number of such subgraphs can grow exponentially with n.
- As an example, let G_0 be a triangle and G be a given graph.
  Then the subgraphs of G that are homeomorphic to G_0 respecting parity, are exactly the odd cycles of G.
  All odd cycle inequalities for G are obtained from inequalities for G_0.
- We will use this example to show how the combined effect of all odd cycle inequalities can be computed in polynomial time in the continuum model.
- Observe that all the inequalities in this example are linear, which simplifies our exposition.
- However, these techniques are not limited to the linear case. Later, we will show an example of non-linear, non-convex inequalities as well.

Combining Effect of Exponential Number of Inequalities

Projectively Invariant Metric

- Consider the projectively invariant metric we use in linear programming algorithm. Let
$$\triangle = \left\{ x_i \mid x_i \geq 0, \sum x_i = 1 \right\}$$
be the simplex used in the algorithm.
- Let **x**, **y** $\epsilon \; int\,(\triangle)$ be two interior points in the simplex. Projectively invariant distance between **x** and **y**, based on p-norms :
$$d(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \cdot \left(\sum_i \left(\frac{x_i}{y_i} - \frac{y_i}{x_i} \right)^p \right)^{\frac{1}{p}}$$
- The infinitesimal version of d gives the Riemannian metric $g(\mathbf{x})$ for $p = 2$ and Riemann-Finsler metric for $p > 2$.
$$g_{ij}^p(x)\, dx_i\, dx_j = \sum_i \left(\frac{dx_i}{x_i} \right)^p$$
The performance of the algorithm depends on curvature in this metric. (Karmarkar, N., AMS, Contemporary Mathematics 114, pp. 51-75)

Projectively Invariant Metric (Contd.)

- For inequalities in the form $A\mathbf{x} \leq b$, let $\mathbf{x}, \mathbf{y}$ be two interior points, and let $\mathbf{s}, \mathbf{t}$ be the corresponding slack variables i.e.

$$A\mathbf{x} + \mathbf{s} = \mathbf{b}, \qquad A\mathbf{y} + \mathbf{t} = \mathbf{b}$$

  Then projectively invariant distance between $\mathbf{x}$ and $\mathbf{y}$ is given by

$$d^p(\mathbf{x}, \mathbf{y}) = \sum_i \left[\frac{1}{2} \cdot \left(\frac{s_i}{t_i} - \frac{t_i}{s_i} \right) \right]^p$$

- Let $\mathbf{s}_0 =$ slack variable for the current interior point $\mathbf{x}_0$ (constant) and $\mathbf{s} =$ slack variable for the next (unknown) interior point $\mathbf{x}$.
- Observe the particular **"distributive"** form of the function $d^p(\mathbf{x}, \mathbf{x}_0)$. If S is the set of all slack variables, then d^p distributes linearly over S.

$$d^p(\mathbf{x}, \mathbf{x}_0) = \sum_{s \epsilon S} \psi(s)$$

  where the $\psi(s)$ for the **individual slack variable** is

$$\psi(s) \quad = \quad \left[\frac{1}{2} \cdot \left(\frac{s}{s_0} - \frac{s_0}{s} \right) \right]^p$$

Computing exponential sums efficiently

- Note that the potential function used in the interior point methods also has the same **distributive** form,
  where the corresponding ψ could be a non-linear rational or transcendental function of individual slack variable.
- We are interested in the case when S is exponentially large.
- Such sums can be evaluated efficiently for
  - Large class of problems involving exponential number of inequalities,
  - when $\psi(s)$ belongs to certain special parametric family of functions such as exponential function, e.g $\psi(s) = e^{-zs}$.
- While the actual function $\psi(s)$ of interest is not of this form,
  it can be expressed as a linear superposition of functions of this form.
- E.g. techniques such as
  Laplace transform or Fourier transform and their inverse transforms enable expressing ψ as (infinite)superposition of (real or complex) exponentials.
- For approximate cross-simulation on the standard model,
  we use a suitable **finite superposition.**

Combined Effect of Inequalities for all Odd Cycles in a Graph

Contribution of an individual slack variable s: $\psi(s) = e^{-zs}$ where $z \epsilon \mathbb{C}$.

Goal: We want to find closed form meromorphic function for total contribution of all slack variables.

Edge Matrix

- For a single edge, we have $\dfrac{w_i + w_j}{2} \leq 0$
  Note: since each node in a cycle has degree 2, we are dividing by 2
  For more general graph minors, the weighting factors are different.
- Introducing slack variable s_{ij}, we get $\dfrac{w_i + w_j}{2} + s_{ij} = 0$. Then,

$$\begin{aligned} s_{ij} &= -\frac{w_i + w_j}{2} \\ \psi &= e^{-z \cdot s_{ij}} = e^{z\left\{\frac{w_i + w_j}{2}\right\}} \end{aligned}$$

- Define edge matrix $A(z, \mathbf{w})$ over the field of meromorphic functions as

$$A_{ij}(z, \mathbf{w}) = \begin{cases} e^{\frac{z}{2}\{w_i + w_j\}} & \text{if } (i,j) \text{ is an edge} \\ 0 & \text{otherwise.} \end{cases}$$

Combined Effect of Inequalities for Odd Cycles (Contd.)

- For a single slack variable s, there is a relation between the derivative operators $\dfrac{\partial}{\partial s}$ and $\dfrac{\partial}{\partial z}$

$$s\frac{\partial\psi}{\partial s} = z\frac{\partial\psi}{\partial z}$$

- For the edge matrix $A(z, \mathbf{w})$, derivative with respect to w_i, the co-ordinates of the interior point, are expressed in terms of the "J-products".

$$\frac{\partial A}{\partial w_p} = \frac{z}{2}\left\{ I \underset{\text{p}}{\textcircled{J}} A + A \underset{\text{p}}{\textcircled{J}} I \right\}$$

- This gives an autonomous differential equation for A, which allows us to create recurrence relation connecting
  - higher powers of A to lower powers and
  - higher derivatives of A to lower order derivatives.
- These recurrence relations lead to closed form expressions for higher derivatives and higher powers.

Permitting additional inequalities

We are interested in all odd cycles in the graph.

A odd cycle is a special case of a closed odd walk.

Since a closed odd walk contains an odd cycle as a subgraph, a sharper version of inequality is also valid for closed odd walk.

However this inequality is implied by the odd cycle inequalities.

Therefore, from the point of view of formulation, these additional inequalities are superfluous:

useless $\rightarrow$ since they are implied by other inequalities but
harmless $\rightarrow$ they are still valid.

However from the point of view of obtaining a closed form expression that can be evaluated in polynomial time, they are essential.

Getting the closed form expression for the combined effect of inequalities

Steps:

(1) Get expressions for the effect of the **implied** inequalities for the following:

(a) For a **given walk** W of **length** l : ψ_W

(b) For a **given pair** of nodes i and j , **all** walks of length l : ψ_{ij}

(c) In the **entire** graph, all **closed** walks of length l : ψ_l

(d) In the entire graph, all closed **odd** walks of **length at most** n : ψ

(2) Transitioning from expressions for implied inequalities to expressions for **sharper** inequalities for the following:

(a) All closed odd walks of length at most n in the entire graph : $\widetilde{\psi}$

Closed form expression: Step (1a)

Single walk W of length l:

Consider a walk W with l edges $i_1, i_2, \ldots, i_{l+1}$. Summing the edge inequalities, we have

$$\frac{1}{2}w_1 + w_2 + \ldots + w_l + \frac{1}{2}w_{l+1} \leq 0$$

Let s_w be a slack variable for the "implied" inequality for walk W.

$$\begin{aligned} s_w &= \sum_{e \in W} s_e \\ \psi_W(s_w) &= e^{-z \cdot s_w} = e^{-z \sum_{e \in W} s_e} = \prod_{e \in W} e^{-z \cdot s_e} \\ \psi_W &= A_{i_1 i_2} A_{i_2 i_3} \cdots A_{i_l i_{l+1}} \end{aligned}$$

Closed form expression: Steps (1b) & (1c)

Between given pair of nodes i and j effect of all walks of length l :

Let $i_1 = i$ and $i_{l+1} = j$.

$$\begin{aligned}\psi_{ij} &= \sum_{i_2, i_3, \ldots, i_l} A_{ii_2} A_{i_2 i_3} \cdots A_{i_l j} \\ \psi_{ij} &= A^l_{ij}\end{aligned}$$

All closed walks of length l in the entire graph:

$$\psi_l = \sum_i A^l_{ii} = tr\{A^l(z, \mathbf{w})\}$$

However, a closed walk of length l is counted $2l$ times in the expression above,

- Due to l vertices of the walk and
- The two senses of traversal along the walk.

Compensating for this repetition, we have

$$\psi_l = \frac{tr\{A^l(z, \mathbf{w})\}}{2l}$$

Closed form expression: Step (1d)

All closed odd walks of length at most n in the entire graph:

Let $k_{max} = \lfloor \frac{n-1}{2} \rfloor$.

$$\begin{aligned}
\psi &= \sum_{k=1}^{k_{max}} \psi_{2k+1} \\
&= \sum_{k=1}^{k_{max}} \frac{1}{2 \cdot (2k+1)} tr\{A^{2k+1}(z, \mathbf{w})\} \\
&= tr\{B(z, \mathbf{w})\}
\end{aligned}$$

where

$$B(z, \mathbf{w}) = \sum_{k=1}^{k_{max}} \frac{A^{2k+1}}{2 \cdot (2k+1)}$$

Transitioning from effect of implied inequalities to sharper ones

Let $i_1, i_2, \ldots, i_{l+1}$ be a walk of length l; its implied inequality is given by

$$\frac{w_{i_1}}{2} + w_{i_2} + \ldots + w_{i_l} + \frac{w_{i_{l+1}}}{2} \leq 0$$

If $i_{l+1} = i_1$, we have a closed walk with the implied inequality given by

$$w_{i_1} + w_{i_2} + \ldots + w_{i_l} \leq 0$$

For closed odd walks ($l = 2k + 1$), we get the following **sharper** inequality.

$$w_{i_1} + w_{i_2} + \ldots + w_{i_l} \leq -1$$

since at most k nodes can be in an independent set.
If s is the slack variable for the implied inequality, let $\widetilde{s}$ denote the slack variable for the corresponding sharper inequality.

$$\begin{aligned} \widetilde{s} &= s - 1 \\ e^{-z \cdot \widetilde{s}} &= e^{z} \cdot e^{-s} \\ \widetilde{\psi}_l &= e^{z} \psi_l \end{aligned}$$

Closed form expression: Step (2a)

Transitioning from the expression for implied inequality for a closed odd walk to the sharper inequality for it involves multiplication by e^z which is independent of the length of the walk.
Therefore, for l odd, $\widetilde{\psi}_l$ is given by

$$\widetilde{\psi}_l = e^z \cdot \frac{tr\{A^l(z, \mathbf{w})\}}{2l}$$

And $\widetilde{\psi}$ is given by

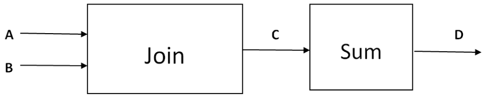
$$\widetilde{\psi} \quad = \quad e^z \cdot B(z, \mathbf{w})$$

It is trivial to see that this can be evaluated in polynomial time in the continuum model.

- Straightforward as written, evaluation of $\widetilde{\psi}$ will take $O(n^4)$ operations but
- With some rearrangement, it is possible to do with $O(n^3)$ operations.

Join Product

We split the tensor product involving contraction into two steps The first step is similar to join operation on relations in a relational data base



Simple examples of J-products –

1. **Matrix Multiplication :**

$$C = A.B, \quad C_{pr} = \sum_q A_{pq} B_{qr}$$

corresponding "join" product (definition):

$$C_{pqr} = A_{pq} B_{qr}$$

without the summation (contraction) over q
corresponding "join" product (notation):

$$C = A \underset{\text{q}}{\textcircled{\text{J}}} B$$

1. **Dot Product of vectors :**

$$\mathbf{c} = \mathbf{a}.\mathbf{b} = \sum_i a_i b^i$$

   Corresponding join product (definition) :

$$c_i = a_i b^i$$

   Corresponding join product (notation) :

$$\mathbf{c} = \mathbf{a} \underset{\text{i}}{\text{Ⓙ}} \mathbf{b}$$

2. **Hadamard Product of matrices :**
   is an example of J-product with two repeated indices

$$A_{pq} = B_{pq} C_{pq}$$

$$A = B \underset{\text{p, q}}{\text{Ⓙ}} C$$

Tensor contraction

input tensors : A, B, with repeated covariant and contravarient indices , and **output tensor :** C with summation over repeated indices $r_1, r_2, ...r_m$.

$$C^{p_1\,p_2\ldots p_k\,s_1\,s_2\ldots s_n}_{q_1\,q_2\ldots q_l\,t_1\,t_2\ldots t_u} = \sum_{r_1\,r_2\ldots r_m} A^{p_1\,p_2\ldots p_k}_{q_1\,q_2\ldots q_l\,r_1\,r_2\ldots r_m} B^{r_1\,r_2\ldots r_m\,s_1\,s_2\ldots s_n}_{t_1\,t_2\ldots t_u}$$

If there are m repeated indices, rank(C) = rank(A) + rank(B) - 2.m
Corresponding **join operator :**

- There is no summation over repeated indices
- One copy of such indices is present in the output, enclosed in round ()
- They don't have significance as tensor indices but simply denote an indexed family of tensors of the same rank as above

$$[C^{p_1\,p_2\ldots p_k\,s_1\,s_2\ldots s_n}_{q_1\,q_2\ldots q_l\,t_1\,t_2\ldots t_u}]_{(}r_1 r_2 ... r_m) = A^{p_1\,p_2\ldots p_k}_{q_1\,q_2\ldots q_l\,r_1\,r_2\ldots r_m} \text{Ⓙ}\; B^{r_1\,r_2\ldots r_m\,s_1\,s_2\ldots s_n}_{t_1\,t_2\ldots t_u}$$

Notation :

$$C = A \textcircled{J} B$$

Sometimes, the repeated indices are noted below $\textcircled{J}$ as shown :

$$C = A \underset{\text{r1r2...rm}}{\textcircled{J}} B$$

Tensor Product as composition of join and summation operation :

$$A.B = \sum_{r_1 r_2 \ldots r_m} A \underset{\text{r1r2...rm}}{\textcircled{J}} B$$

Properties of join product

- **Linear in both arguments :**
  If α and β are scalars

$$\{\alpha A + \beta B\} \textcircled{J}\ C = \alpha(A \textcircled{J}\ C) + \beta(B \textcircled{J}\ C)$$

  If A, B, X are matrices of compatible dimensions,

$$X(A \textcircled{J}\ B) = \{XA\} \textcircled{J}\ B$$

$$(A \textcircled{J}\ B)X = A \textcircled{J}\ \{BX\}$$

  Similar rule applies for general compatible tensor multiplication.
- **Associative :**

$$A \textcircled{J}\ [B \textcircled{J}\ C] = [A \textcircled{J}\ B] \textcircled{J}\ C$$

- **Derivative rule :**

$$\frac{\partial}{\partial x_i}\{A(x_1 x_2 \ldots x_n) \textcircled{J}\ B(x_1 x_2 \ldots x_n)\} = \frac{\partial A}{\partial x_i} \textcircled{J}\ B + A \textcircled{J}\ \frac{\partial}{\partial x_i} B$$

- **Transpose :**

$$C^T_{ijk} = C_{kji}$$

$$[A \textcircled{\scriptsize J}\ B]^T = [B^T \textcircled{\scriptsize J}\ A^T]$$

For symmetric A,B

$$[A \textcircled{\scriptsize J}\ B] = [B \textcircled{\scriptsize J}\ A]$$

- **Transpose of triple product :**

$$[A \underset{\text{p}}{\textcircled{\scriptsize J}} B \underset{\text{q}}{\textcircled{\scriptsize J}} C]^T = C^T \underset{\text{q}}{\textcircled{\scriptsize J}} B^T \underset{\text{p}}{\textcircled{\scriptsize J}} A^T$$

Note the reversal of p, q

Application of J-product in the present context

Derivative of Edge Matrix w.r.t. co-ordinates of interior point can be expressed in terms of J-product , which reduces derivative to an algebraic operation :

$$\frac{\partial A}{\partial w_q} = \frac{z}{2}\left\{ I \underset{q}{\textcircled{J}} A + A \underset{q}{\textcircled{J}} I \right\}$$

Recurrence relation for derivative of kth power of the Edge Matrix in terms of J-products of lower powers :

$$\frac{\partial A^k}{\partial w_q} = \frac{z}{2} sym \left\{ \sum_{i=1}^{k} A^i \underset{q}{\textcircled{J}} A^{k-i} \right\}$$

Recurrence relation for kth derivative of the Edge Matrix in terms of derivatives of lower order e.g. second derivative in terms of first derivative :

$$\frac{\partial^2 A}{\partial w_p \partial w_q} = \frac{z}{2}\left\{ \frac{\partial A}{\partial w_p} \underset{q}{\textcircled{J}} I + I \underset{q}{\textcircled{J}} \frac{\partial A}{\partial w_p} \right\}$$

Solution of recurrence relations for J-Products

$$\frac{\partial A^k}{\partial w_p} = \sum_{\boldsymbol{\alpha}} \frac{z}{2} \left\{ W^{k,1}_{rank(\boldsymbol{\alpha})} \cdot A^{\alpha_1} \underset{p}{\textcircled{J}} A^{\alpha_2} \right\} \qquad (5)$$

$$\frac{\partial^2 A^k}{\partial w_p \partial w_q} = \sum_{\boldsymbol{\alpha}} \frac{z^2}{4} \cdot \frac{1}{2!} \left\{ W^{k,2}_{rank(\boldsymbol{\alpha})} \left[A^{\alpha_1} \underset{p}{\textcircled{J}} A^{\alpha_2} \underset{q}{\textcircled{J}} A^{\alpha_3} + A^{\alpha_1} \underset{q}{\textcircled{J}} A^{\alpha_2} \underset{p}{\textcircled{J}} A^{\alpha_3} \right] \right\} \qquad (6)$$

where $\boldsymbol{\alpha}$ is a (weak) composition of k given by

$$\begin{aligned}
\boldsymbol{\alpha} &= (\alpha_1, \alpha_2) \qquad \alpha_1 + \alpha_2 = k, \alpha_i \geq 0 && \text{for (1)} \\
\boldsymbol{\alpha} &= (\alpha_1, \alpha_2, \alpha_3) \qquad \alpha_1 + \alpha_2 + \alpha_3 = k, \alpha_i \geq 0 && \text{for (2)} \\
rank(\boldsymbol{\alpha}) &= \text{number of non-zero } \alpha_i' s \\
W^{k,m}_{rank(\boldsymbol{\alpha})} &= \text{fixed weights depending on rank}
\end{aligned}$$

Similar formulas hold for higher derivatives. For efficient computation, instead of these "flat" expressions, new recurrence relations involving "forward" and "backward" sweep are created.

Approximate Evaluation on the Standard Model

- We want to illustrate how a function of distributive type over the set of all slack variable can be approximated by superposition of exponentials, for various $\psi(s)$.
- For simplicity of exposition consider

$$\psi(s) = \frac{1}{s}, \;\; s > 0$$
$$\frac{1}{s} = \int_0^\infty e^{-sx}\, dx$$

- In the problem of interest,
  all slack variable lie in a bounded interval of real axis.
- Upper bound : trivial since $-1 \le w_i \le 1$. Hence $|a^T w| < |a|_1 \le n$ for closed walks of length at most n.
- Lower bound: at each iteration we round the interior point to nearest valid ± 1 solution by a simple method, until optimal solution is identified.
- Hence the minimum value of the slack variable remains bounded away from zero.

Approximate Evaluation on the Standard Model (Contd.)

Consider the following nested regions

$$R_i = \{x \in \mathbb{R} \mid e^{-i} \leq x \leq e^{j_{max}}\} \quad \text{for } i = 1, \ldots, L$$

so that

$$R_1 \subset R_2 \subset R_3 \cdots \subset R_L$$

Note: L is not known in advance; we increment it dynamically based on the number of iterations so far.
Projective invariance of the algorithm implies invariance w.r.t. simple uniform scaling transformations $T_k : s \rightarrow e^k s, \quad k \in \mathbb{Z}$.

To get an efficient approximation exploiting the scale invariance, we make further substitution

$$x = e^{-at}$$

in the integral for $\frac{1}{s}$.

$$\frac{1}{s} = \int_{-\infty}^{\infty} a \cdot e^{-at} \cdot e^{-\left[e^{-at}.s\right]} dt$$

Approximate Evaluation on the Standard Model (Contd.)

Approximate the integral by sum, with integer nodes in a bounded range $[-m, M]$, $m, M \in \mathbb{N}$.

$$\frac{1}{s} \sim \sum_{i=-m}^{i=M} a \cdot e^{-ia} \cdot e^{-\left[e^{-ia}.s\right]}$$

Let $\lambda_i = e^{-ia}$

$$\frac{1}{s} \sim \sum_{i=-m}^{i=M} a \cdot \lambda_i \cdot e^{-\lambda_i s} \qquad (*)$$

As an example, suppose the region of interest is

$$e^{-30} < s < e^{1}$$

Taking $a = \frac{1}{2}$, $m = \frac{1}{a} = 2$, $M = \frac{30}{a} = 60$, we have 63 terms and this gives the l.h.s. and r.h.s. of $(*)$ which are identical when evaluated in standard double precision (64-bit) arithmetic.

Superposition of Exponentials

- By similar techniques approximation to the function $\psi(s)$ of interest is expressed as sum of exponential

$$\psi(s) = \sum_i c_i e^{-\lambda_i s}$$

- The main function ϕ (either metric or potential) is expressed in terms of $\psi(s)$ in distributive form

$$\begin{aligned}
\phi &= \sum_{s \epsilon S} \psi(s), \\
&= \sum_{s \epsilon S} \sum_i c_i e^{-\lambda_i s} \\
&= \sum_i c_i \left\{ \sum_{s \epsilon S} e^{-\lambda_i s} \right\} \\
&= \sum_i c_i \cdot e^z \cdot tr\{B(z)\}|_{z=\lambda_i}
\end{aligned}$$

- Note that the number of terms is $O(L)$, and evaluation of each term is polynomial in n.

An example of application to non-convex, non-linear problem

- Consider set of inequalities of the form

$$(a_i^T x)^2 - (b_i^T x)^2 \leq c, \;\; c > 0$$

  The set defined by these inequalities is $(2,2)$-connected [Kar, 10] slack variable $s_i = c - (a_i^T x)^2 + (b_i^T x)^2$

- With the function $\psi(s) : \mathbb{R} \to \mathbb{R}$ of interest, we associate another function $\tilde{\psi} : \mathbb{R}^2 \to \mathbb{R}$ as follows

$$\tilde{\psi}(u,v) = \psi(c - u^2 + v^2)$$

  Let R be the region of interest in the (u,v) plane.
  Note: due to the symmetries based on flipping signs of u and v, we can consider only one quarter of the plane.

[Kar, 10] = Beyond Convexity, LNCS 6457, Dec. 2010

An example of application to non-convex, non-linear problem

- Let χ_R : Characteristic function of R. By considering 2-D inverse Laplace transform of $\tilde{\chi}_R \tilde{\psi}(u, v)$, (or by other means) we can obtain finite approximation based on exponentials functions in the plane

$$\chi_R \tilde{\psi} \sim \sum_{(\mu_k, \lambda_k)} c_k e^{-[\mu_k u + \lambda_k v]}$$

- Since $u = a_i^T x$ and $v = b_i^T x$ are linear in x, the above expression is of the same type as considered before.

Extension to the satisfiability problem

Recall the definition of SAT...

- We have boolean variables $x_1, \dots, x_n$.
- A literal l is a variable or its complement: $l = x$ or $l = \bar{x}$
- A clause is an OR of literals. Eg. $C = x \vee y \vee \bar{z}$. In 3-SAT, each clause has 3 literals.
- A satisfiability formula $f = C_1 \wedge C_2 \wedge \dots \wedge C_m$
- Problem (SAT): Find truth assignments to the variables $x_1, \dots, x_n$ such that f is true, or prove that no such assignment exists.

Generation of implied clauses

- Consider the following pair of clauses

$$\begin{aligned} C_1 &= a \vee b \vee \bar{c} \\ C_2 &= c \vee d \vee e \end{aligned}$$

- Together they imply the clause $C_3 = a \vee b \vee d \vee e$.
- We refer to this operation as the "join" operation on clauses.
- This operation is fundamental in resolution or elimination methods for solving the SAT problem.
- It is known that the number of such new clauses generated during the resolution process grows exponentially with n for almost all instances of the problem with parameters in a certain range (Chvátal and Szemerédi).

Continuum based approach

- Associate real variables taking ± 1 values with the boolean variables.
- Embedding the problem in $\mathbb{R}^n$, allow the variables to take on values in the interval $[-1, 1]$, i.e. $-1 \leq x_i \leq 1$.
- For a literal l,

$$v(l) = \begin{cases} x & \text{if } l = x \\ -x & \text{if } l = \bar{x} \end{cases}$$

- Each clause corresponds to an inequality

$$C \quad = \quad l_1 \vee l_2 \vee l_3 \quad \longrightarrow \quad v(l_1) + v(l_2) + v(l_3) \quad \geq \quad \text{-1}$$

  Explanation: Sum of three ± 1 variables can be -3, -1, 1 or 3 of which the value -3 is forbidden since it corresponds to all literals being false.

Interpreting the join operation in the continuum model

	Clauses			Inequalities			
	$a \vee b \vee \bar{c}$	(C_1)	$\longrightarrow$	$a + b - c$	$\geq$	-1	(1)
	$c \vee d \vee e$	(C_2)	$\longrightarrow$	$c + d + e$	$\geq$	-1	(2)
Join:	$a \vee b \vee d \vee e$	(C_3)	$\longrightarrow$	$a + b + d + e$	$\geq$	-2	(3)

- Observe that the inequality (3) corresponding to the newly generated clause is just the sum of (1) and (2).
- Imposition of constraint (3) does not change the feasible set, hence it is superfluous.
- This is the first reason why the continuum approach is more economical than resolution.

A thought experiment

- Run the resolution algorithm.
- For each new clause generated, ask if the corresponding inequality is superfluous or essential (i.e. not derivable as a non-negative combination of the previous inequalities).
- Unless the previous inequalities have only vectors with all ± 1 co-ordinates as extreme points, an essential constraint must get generated during the course of the algorithm since resolution is a complete method.
- This leads us to the following question: Which sequence of clauses when joined, yields an essential constraint, and is such that no subsequence has the same property?
- To understand this, we introduce the concepts of paths, walks, cycles, etc. formed by subformulas in a SAT problem.

Paths and cycles

<u>(Open) Path</u>:

- Consider a sequence of clauses of the following type:
  $l_1 \vee l_2 \vee \overline{l_3}, \quad l_3 \vee l_4 \vee \overline{l_5}, \quad l_5 \vee l_6 \vee \overline{l_7}, \ldots, l_{2k-1} \vee l_{2k} \vee \overline{l_{2k+1}}$
  where each l_i is a literal based on a distinct variable.
- For any two consecutive clauses in the sequence, the last literal of the earlier clause and the first literal of the latter clause are complementary.
- The above sequence yields the following joined clause:
  $$l_1 \vee l_2 \vee l_4 \vee l_6 \vee \ldots \vee l_{2k} \vee \overline{l_{2k+1}}$$
  and the corresponding inequality is superfluous.

<u>(Ordinary) Cycle</u>:

- If $l_{2k+1} = l_1$ above, then we can join the two ends by normal rules of joining, but the joined clause of the sequence is a tautology (always true) due to the presence of l_1 and $\overline{l_1}$.

Mobius cycles

- Consider the above sequence of clauses again:
  $l_1 \vee l_2 \vee \overline{l_3},\ l_3 \vee l_4 \vee \overline{l_5},\ l_5 \vee l_6 \vee \overline{l_7}, \ldots, l_{2k-1} \vee l_{2k} \vee \overline{l_{2k+1}}$
- If $l_{2k+1} = \overline{l_1}$, then the joined clause of the sequence contains two copies of l_1, but we need only one. Hence, the joined clause is equivalent to $l_1 \vee l_2 \vee l_4 \vee l_6 \vee \ldots \vee l_{2k}$.
- Corresponding inequality
  $$v(l_1) + v(l_2) + v(l_4) + \ldots + v(l_{2k}) \geq -k + 1$$
  where
  $$v(l) = \begin{cases} x & \text{if } l = x \\ -x & \text{if } l = \bar{x} \end{cases}$$
- This is sharper than the inequality implied by the sum of the constituent inequalities, namely
  $$2 \cdot v(l_1) + v(l_2) + v(l_4) + \ldots + v(l_{2k}) \geq -k$$

Mobius cycles (Contd.)

- We call the cycles of the kind referred to above as “mobius cycles”.
- The mobius cycles are the only mechanism giving rise to new sharper inequalities.
- If there were no mobius cycles, then solving the L.P. corresponding to the SAT problem is sufficient to solve the latter.
- In other words, the special classes of SAT formulas in which mobius cycles are forbidden as subformulas, can be solved in polynomial time.

Computing the effect of mobius cycles in the continuum model

- The total number of mobius cycles in the original formula can be exponentially large, but we can compute their net effect in polynomial time.
- The method is similar to the one used for odd cycles in the maximum independent set problem, except that we construct a directed graph as explained below.
- $2n$ nodes corresponding to the literals $x_1, \overline{x_1}, x_2, \overline{x_2}, \ldots, x_n, \overline{x_n}$.
- Let $C = l_1 \vee l_2 \vee l_3$ be a clause.
- Inequality corresponding to this clause C

$$v(l_1) + v(l_2) + v(l_3) \geq -1$$

- Slack variable $s = 1 + v(l_1) + v(l_2) + v(l_3)$

$$\psi_C(s) = e^{-z \cdot s} = e^{-z} \cdot e^{z[v(l_1)+v(l_2)+v(l_3)]}$$

Computing the effect of mobius cycles in the continuum model

- Corresponding to C, put the following 6 edges in the graph.

$$(l_1, \overline{l_2}), (l_1, \overline{l_3}), (l_2, \overline{l_3})$$
$$(\overline{l_1}, l_2), (\overline{l_1}, l_3), (\overline{l_2}, l_3)$$

- Each of the 6 edges corresponding to clause C is labelled with $\psi_C(s)$.
- Construct a "clause matrix" $A(z, \mathbf{x})$

$$A_{ij} = \begin{cases} \text{sum of all } \psi_C(s) \text{ of all parallel edges} & \\ \text{between } i \text{ and } j & \text{if any} \\ 0 & \text{otherwise} \end{cases}$$

- Note that the clause matrix is not symmetric

$$A(i, j) \neq A(j, i)$$

but it has another kind of symmetry

$$A(i, j) = A(\bar{j}, \bar{i})$$

Closed form expression for combined effect of all mobius cycles

- As before, we permit self-intersecting, directed, closed walks and thereby include the effect of "useless but harmless" inequalities. This leads to closed form expression for the combined effect as before.
- Consider a mobius cycle $l_1 \vee l_2 \vee \overline{l_3},\ l_3 \vee l_4 \vee \overline{l_5},\ l_5 \vee l_6 \vee \overline{l_7}, \ldots, l_{2k-1} \vee l_{2k} \vee l_1$
- Let s and $\widetilde{s}$ be the slack variables for the implied and sharper inequalities respectively.

$$\begin{aligned}
s &= k + 2 \cdot v(l_1) + v(l_2) + v(l_4) + \ldots + v(l_{2k}) \\
\widetilde{s} &= k - 1 + v(l_1) + v(l_2) + v(l_4) + \ldots + v(l_{2k}) \\
s - \widetilde{s} &= 1 + v(l_1) \\
e^{z \cdot (s - \widetilde{s})} &= e^{z[1+v(l_1)]} \\
\psi(\widetilde{s}) &= e^{-z \cdot \widetilde{s}} = e^{-z \cdot s} \cdot e^{z[1+v(l_1)]} = e^{z[1+v(l_1)]} \cdot \psi(s)
\end{aligned}$$

- The factor $e^{z[1+v(l_1)]}$ is associated with a special edge from $\overline{l_1}$ to l_1. Traversing this edge differentiates mobius cycles from ordinary cycles.

Closed form expression for combined effect of all mobius cycles (Contd.)

- Construct "mobius completion matrix" M_c from special edges as follows:

$$\begin{aligned} M_C(z,\mathbf{x})_{i,\bar{i}} &= e^{z[1-x_i]} \\ M_C(z,\mathbf{x})_{\bar{i},i} &= e^{z[1+x_i]} \\ M_C(z,\mathbf{x})_{i,j} &= 0 \text{ if } j \neq \bar{i} \quad \{\text{Note:} \bar{\bar{i}} = i\} \end{aligned}$$

- Define walk matrix $B(z,\mathbf{x})$ as before, except that we include all walks (odd and even) upto length k_{max}, where k_{max} = number of clauses.

$$B(z,\mathbf{x}) = \sum_{k=2}^{k=k_{max}} \frac{A^k(z,\mathbf{x})}{2k}$$

- For each walk, there is a "mirror-image" walk obtained by complementing all the nodes in t he walk, hence factor of 2 in the denominator
- Multiplication by M_c also achieves the transition from implied to sharper inequalities. The combined effect is given by

$$\phi(z,\mathbf{x}) = tr\{M_c(z,\mathbf{x}) \cdot B(z,\mathbf{x})\}$$

- Lengths of mobius cycles have expontential effect on resolution. In contrast, we are able to include the effect of mobius cycles of all lengths in polynomial number of operations.