2-Diregular Multi Graphs

What they are, How we make them, and Why we care

Robert Dygert , Matthew Heavner, Devanshu Pandey, Michael Skvarch, Richard Gonsalves
{radygert, mheavner, dpandey, mskvarch, phygons}@buffalo.edu

University Honors College
University at Buffalo
The State University of New York

What is a 2-Diregular Multi Graph?

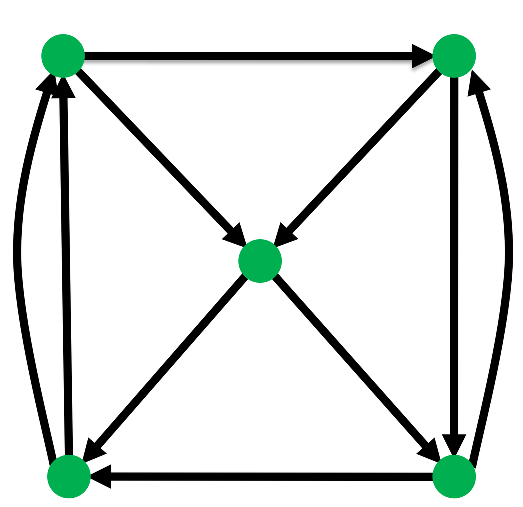
A directed multi graph is a pair *(V, E)* :

V being the set of vertices
- $|V| \equiv$ order of the graph

E being the set of edges
- Ordered pair *(u,v)* represents a directed edge from *u* to *v*
- May contain multiple instances of a given edge

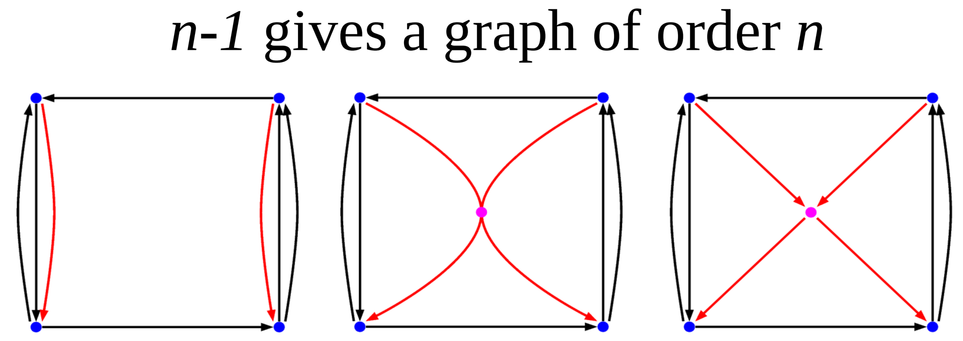
Our graphs have the additional properties:
- They are connected
- Have no loops (i.e. no edges *(v,v)*)
- Two edges coming into and going out of every vertex (called 2-diregular)

Example:

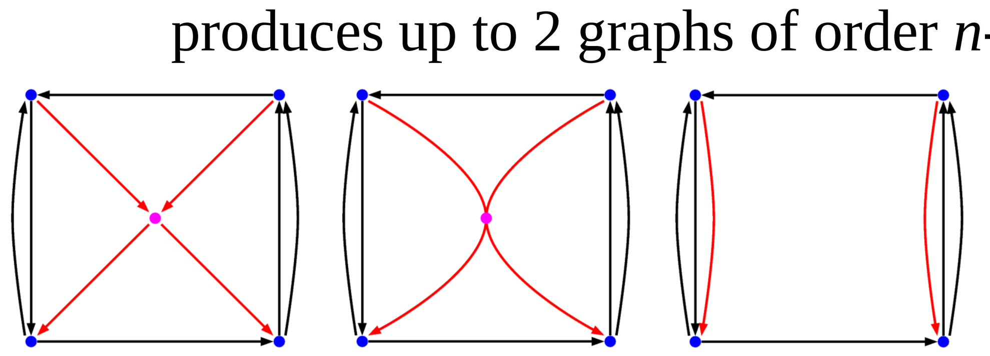


Generation process

Welding:
- "Pinching" two edges together to form a new vertex at the "pinching point"
- Welding two edges on a graph of order *n-1* gives a graph of order *n*

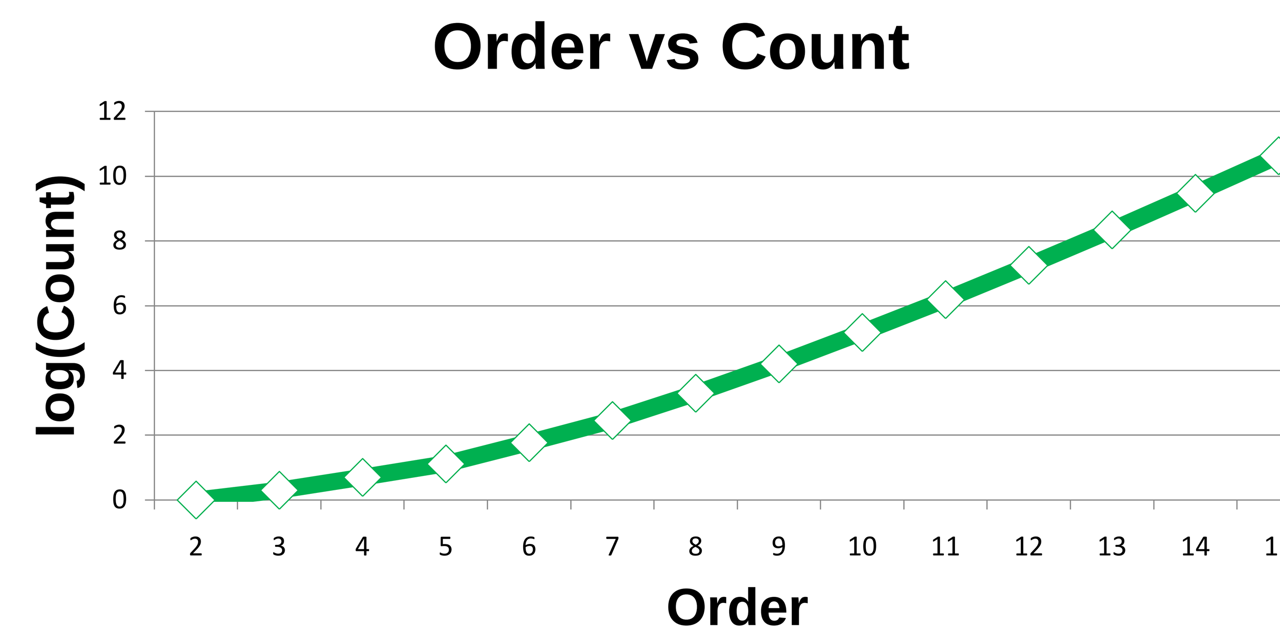


Unwelding:
- Inverse of the welding process
- Unwelding a vertex on a graph of order *n* produces up to 2 graphs of order *n-1*



Results

Order	Graph Count	CPU Years
14	2,975,850,329	5
15	43,060,406,372	84



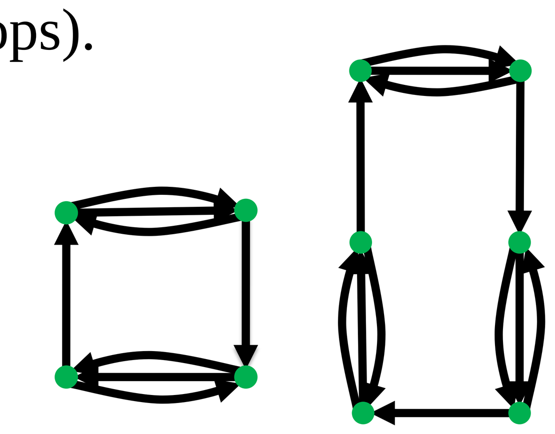
What makes this difficult?

We can generate all graphs of order n by welding graphs of order n-1. Except the primitives.
This is because they have illegal unweldings (i.e. they form loops).

This process produces many isomorphic child graphs.

We can intelligently produce children to restrict isomorph generation.
Only create children if their welding maximizes an invariant

Isomorphism checking is exponentially hard

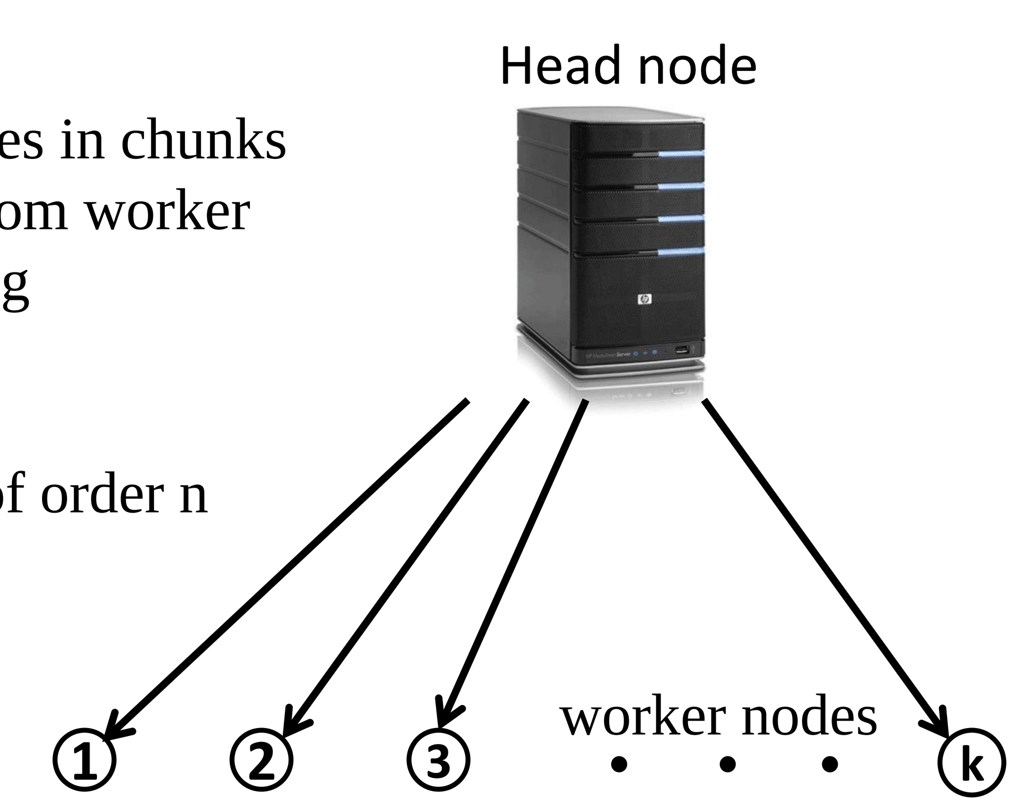


Implementation

We implement our solution on a compute cluster using Python and MPI at UB's Center for Computational Research. We used a client-server model for computation as follows:

Head node:
- Distributes graphs of order n-1 to worker nodes in chunks
- Manages load balancing by taking requests from worker nodes to send more graphs for processing

Worker nodes:
- Weld graphs of order n-1 to generate graphs of order n
- Use structural and canonical invariants to ensure isomorph-free graph generation
- Request more graphs from server as needed



The same configuration is used when analyzing the graphs of order n. by having the head node send chunks of graphs to the worker nodes which calculate the Euler path count and the size of the automorphism group for each graph.

Perturbation Series

$$f_{2D}(x) = \sum_n -(-4x)^n \sum_g \frac{1}{TG} \quad [1]$$

g = graphs of order n, T = number of Euler paths
G = size of the automorphism group

The Padé approximants of the perturbation series are used to extrapolate the Abrikosov Ratio

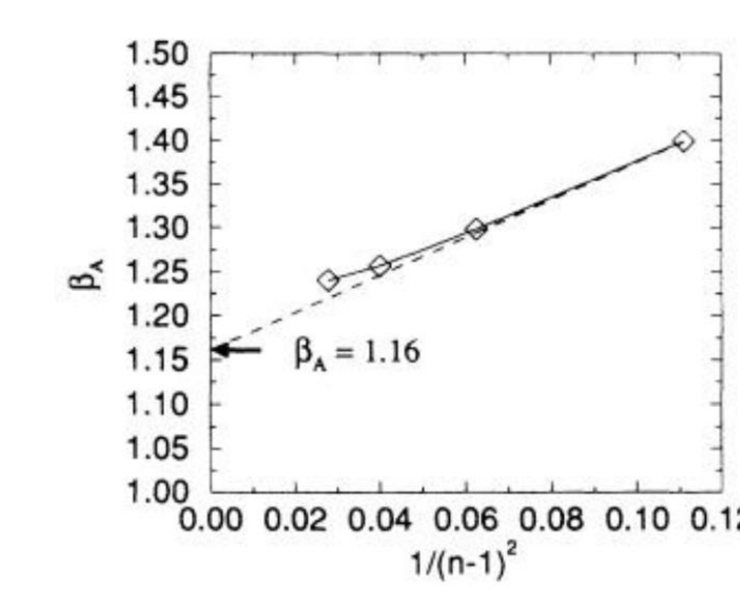
Padé Approximants

$$f_{2D}(x) = -2x - x^2 + \frac{38}{9}x^3 - \frac{1199}{30}x^4 + 471.3965945165944x^5 - 6471.5625749551446x^6 + 101279.32784597063x^7 - 1779798.7875947522x^8 + 34709019.614363678x^9 - 744093435.66822231x^{10} + 17399454123.559521x^{11} - 440863989257.28510x^{12} + 12035432945204.531x^{13}$$

Theory and physical description suggest that $f_{2D}(x)$ should behave linearly as $x \to \infty$.

We use Padé Approximants to obtain an asymptotically linear approximation to the high degree polynomial shown above.

The Padé approximation of the perturbation series at different orders:



[3]

FIG. 3. The extrapolation of the Abrikosov ratio β_A by $[n, n-1]$ Padé approximants (n = 4, 5, 6, and 7).

Summary

One can determine more accurately whether Padé approximants are a good mathematical representation of the perturbation series by enumerating and analyzing the 2-diregular multigraphs of higher orders

References

[1] G. J. Ruggeri and D. J. Thouless. *Perturbations series for the critical behavior of type II superconductors near H_{c2}.* J. Phys. F. 1976
[2] E. Brezin, A. Fujita and S. Hikami. *Large-Order behavior of the Perturbation series for Superconductors near H_{c2}. Phys Rev. Lett. 1990.*
[3] J. Hu, A. H. MacDonald and B. D. McKay. *Correlations in two dimensional vortex liquids. Phys. Rev. B.* 1994