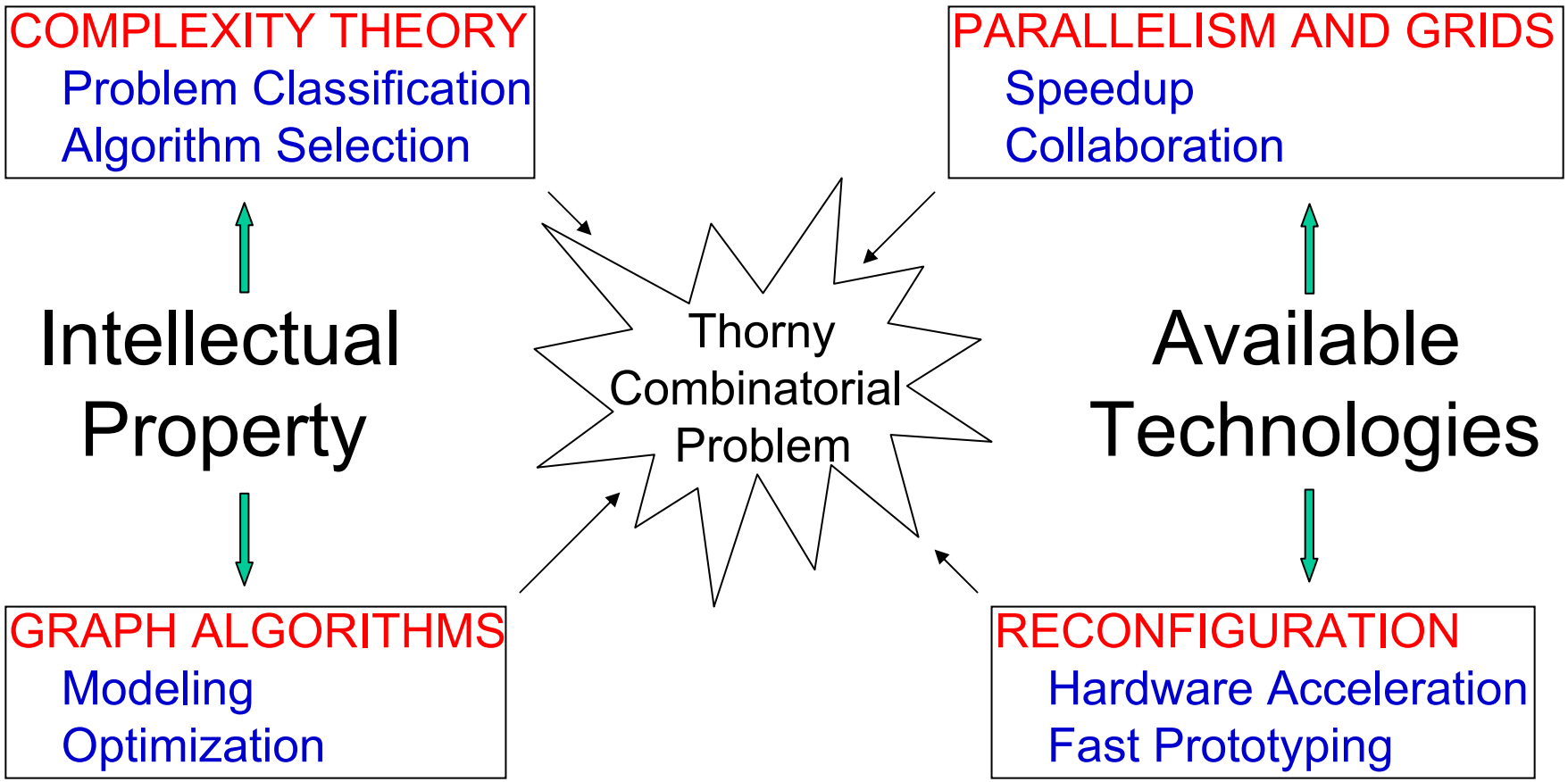
High-Performance Computational Tools for Biological Applications

Mike Langston

Department of Computer Science
University of Tennessee

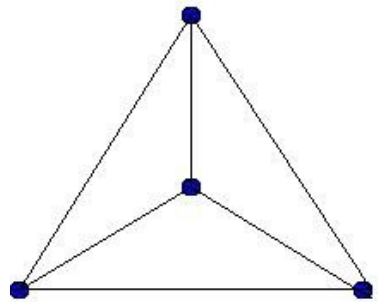
ORNL, CBI, 17 October 2003

Typical Application



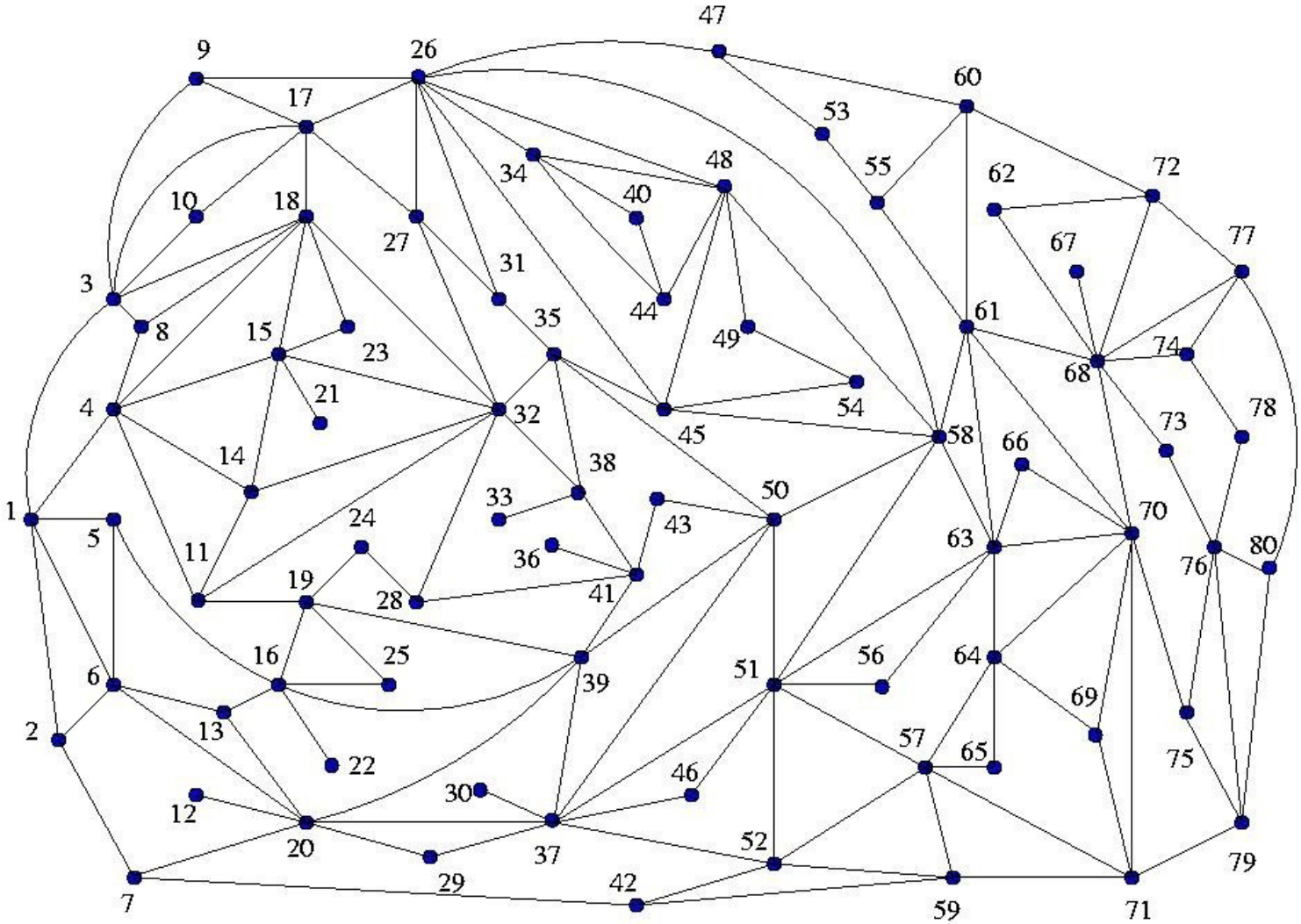
An Example: Clique

- A clique is a complete subgraph, for example, K_4

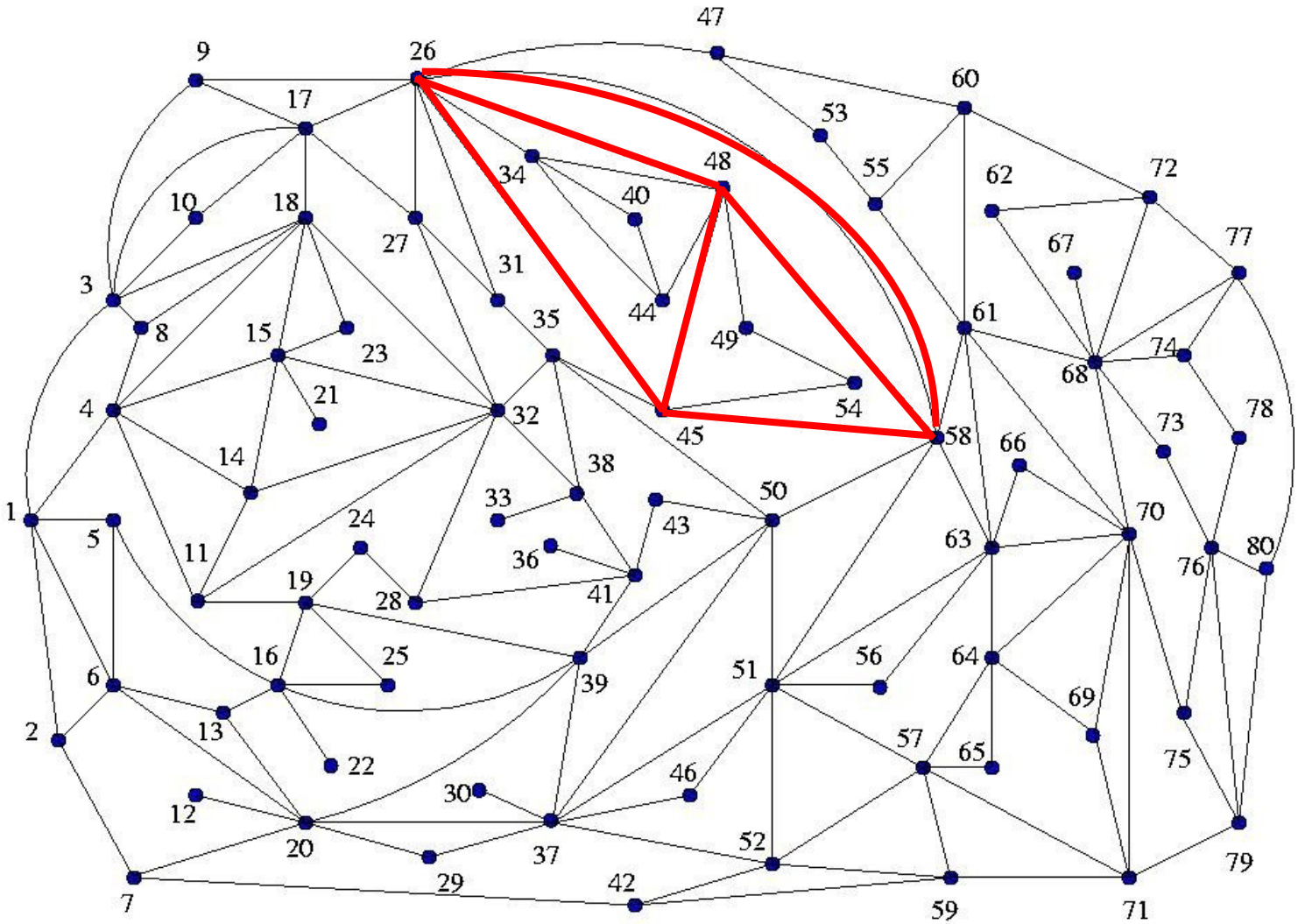


- Ubiquitous in computational biology
- *NP*-complete, difficult even for small cliques on planar graphs

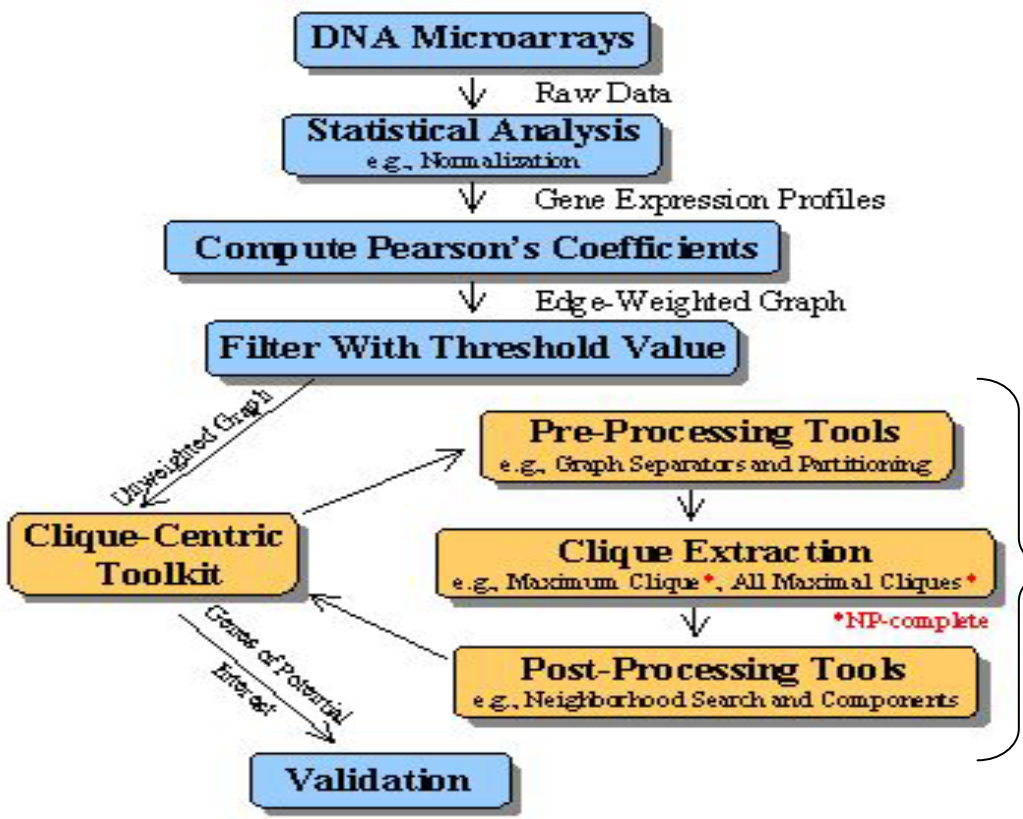
Does this graph contain K_4?



Indeed it does.



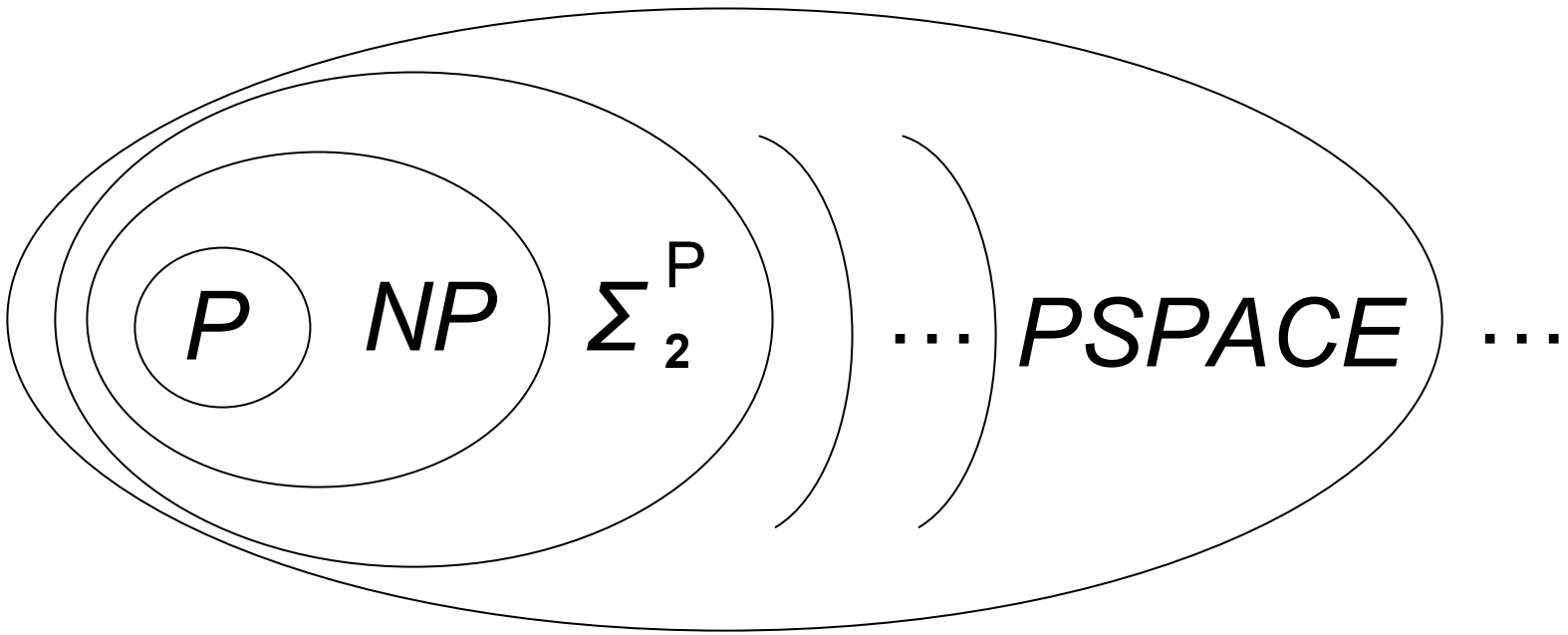
An Example of Clique's Utility: Microarray Data Analysis



Complexity Analysis
Algorithm Development
Extensive Parallelism
Hardware Acceleration

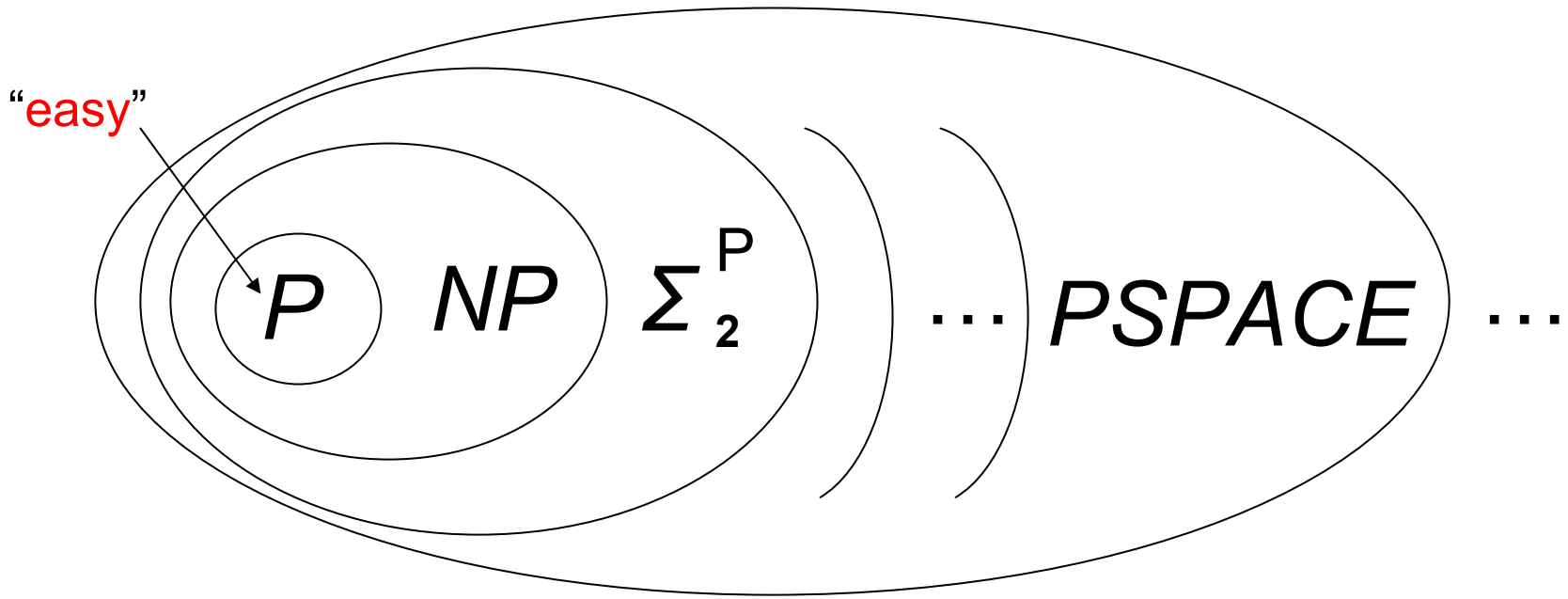
Complexity Theory

The Classic View:



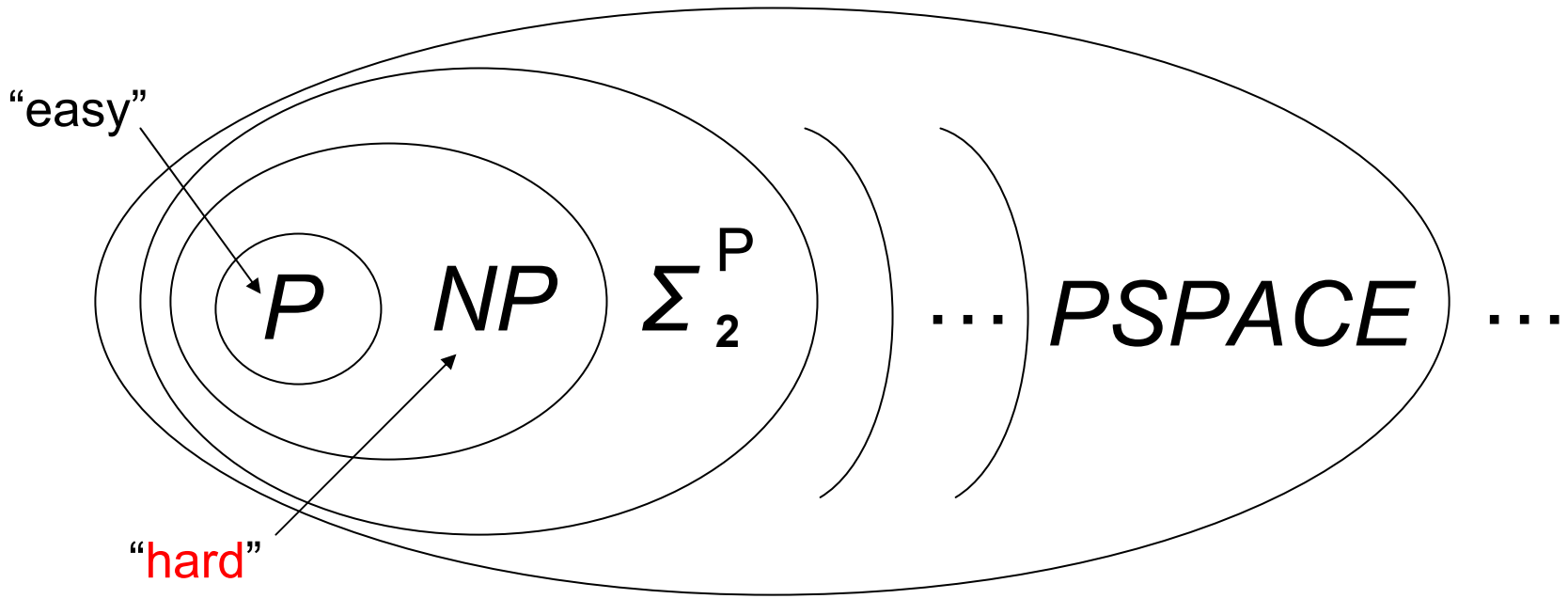
Complexity Theory

The Classic Interpretation:



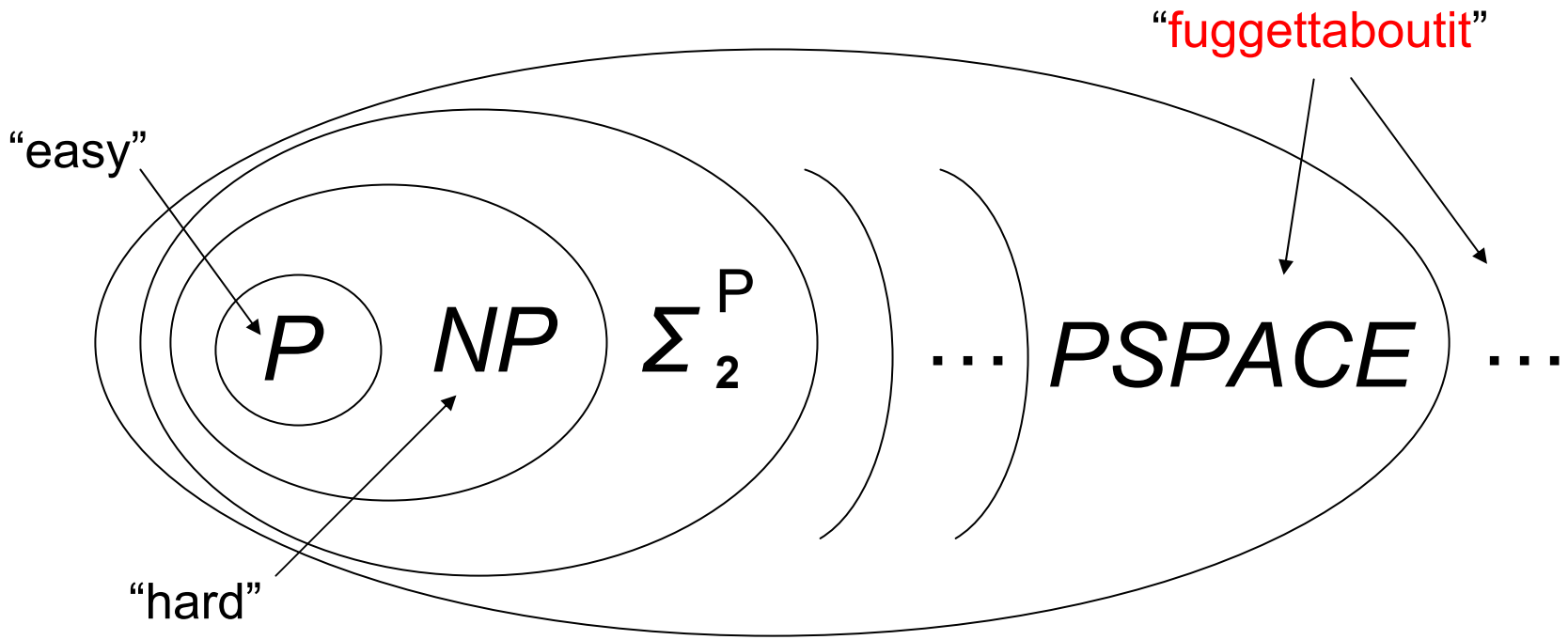
Complexity Theory

- *The Classic Interpretation:*



Complexity Theory

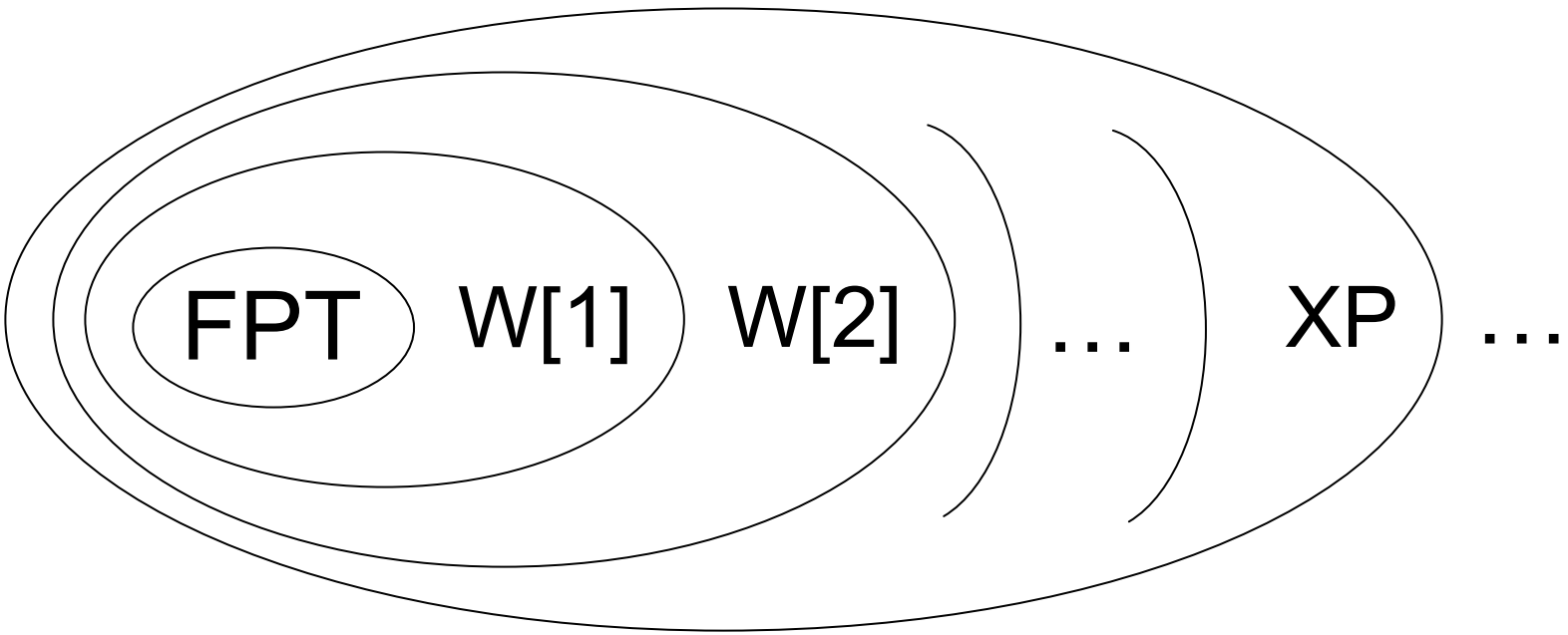
- *The Classic Interpretation:*



Complexity Theory

Can super-polynomial complexity be limited to problem parameter(s)? Hence,

A Parameterized View:



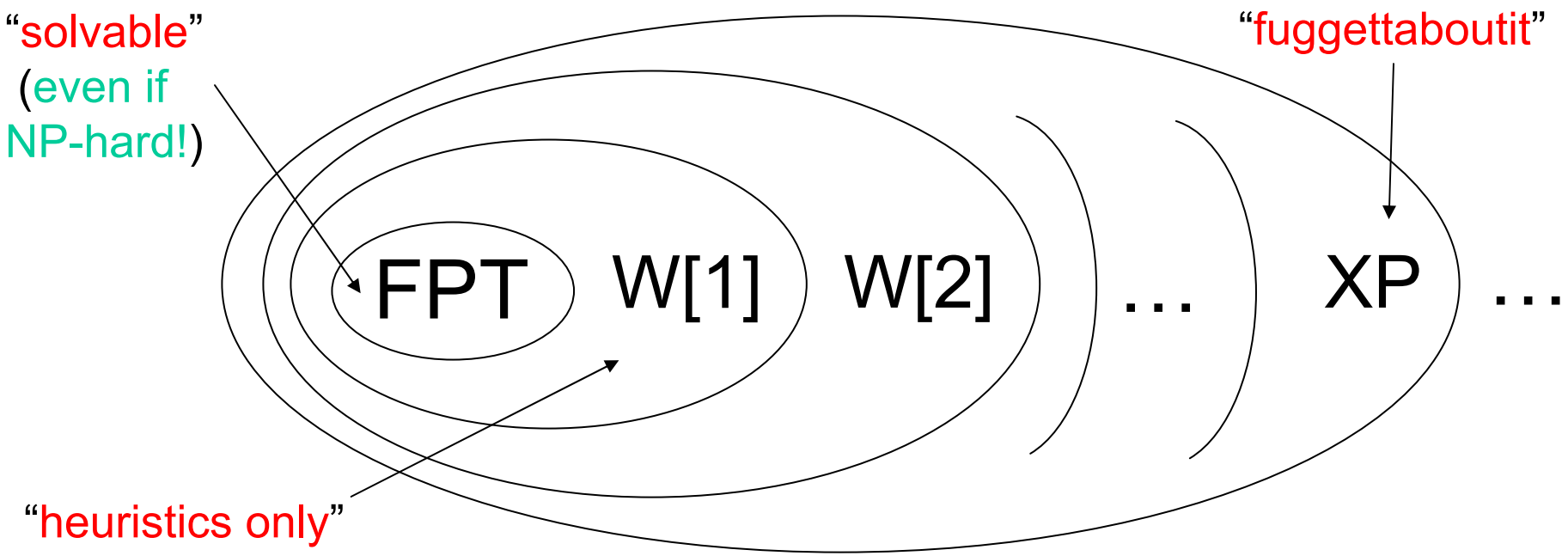
Complexity Theory

Consider time bounds of, say, $O(2^{kn})$ versus $O(2^k n)$, when k is fixed.

Complexity Theory

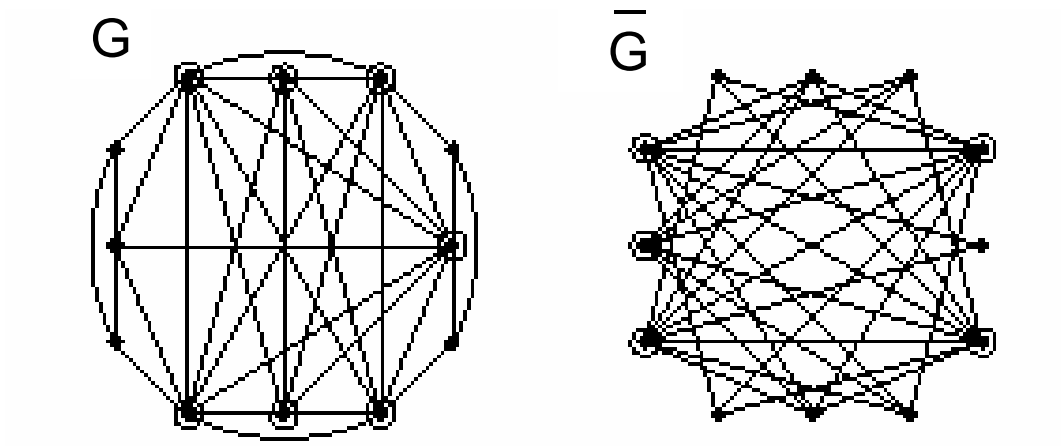
Consider time bounds of, say, $O(2^{kn})$ versus $O(2^k n)$, when k is fixed. Hence,

The Parameterized Interpretation:



Graph Algorithms

- Clique is a good example. But it is not FPT.
- Fortunately, Vertex Cover is FPT.
- And Vertex Cover is a dual to Clique:

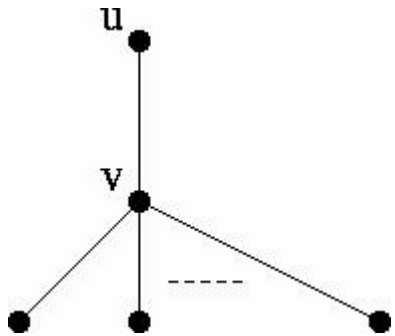


The Vertex Cover Project

- Preprocessing via degree structure
- Kernelize to computational core
- Branching explores core
- Interleave all three

Preprocessing

- Low degree rules
  (e.g., degree one)
- High degree rule
- Resultant graph has size $O(k^2)$
  [at most $k(1+k/3)$ vertices]

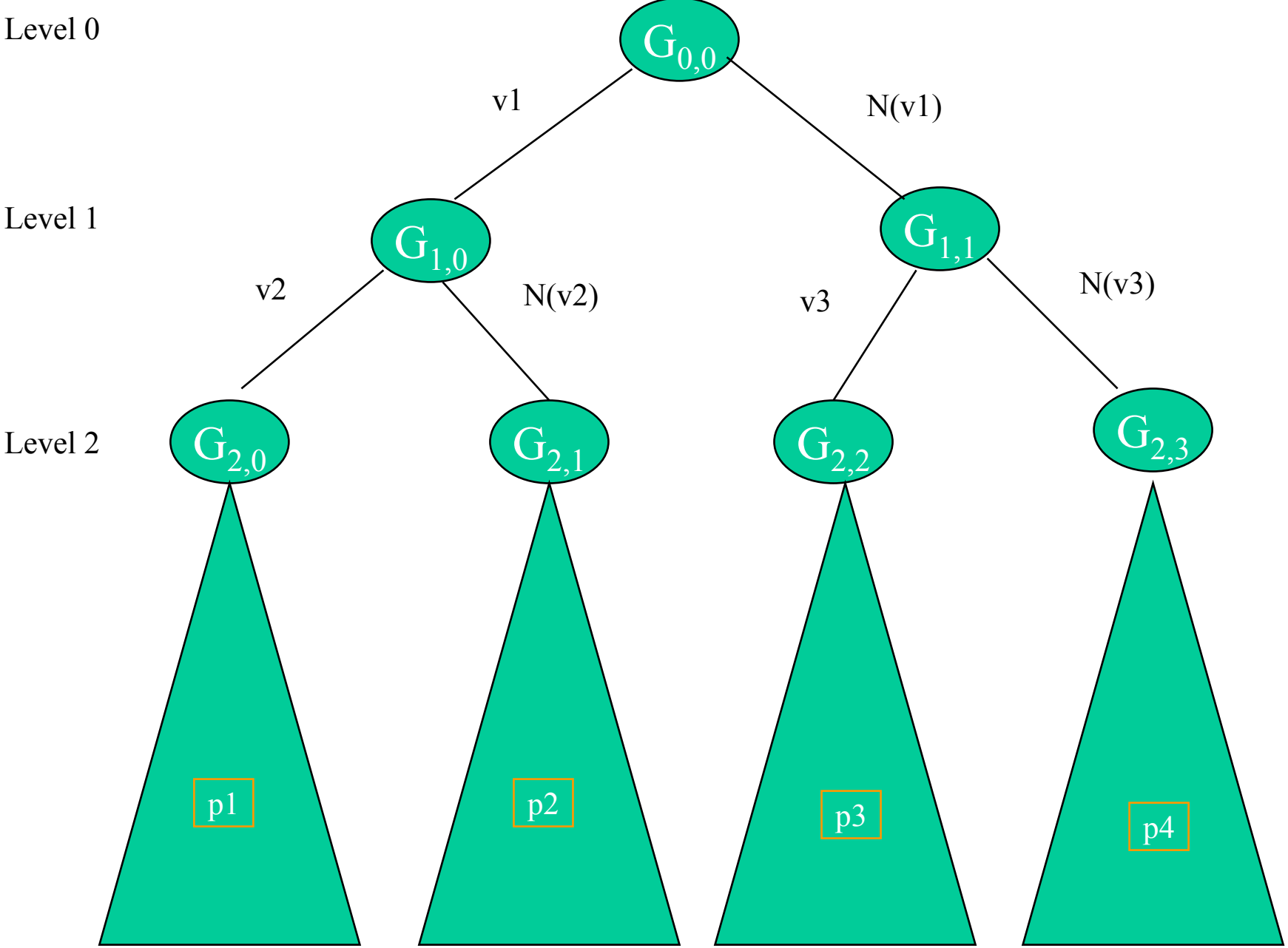


Kernelization

- Based on linear programming:
  - minimize: $\sum_{u \in V(G)} X_u$
  - subject to: $X_u + X_v \geq 1 \forall uv \in E(G)$
  - where: $X_u \geq 0 \forall u \in V(G)$
- Resultant graph has size $O(k)$ [at most $2k$ vertices]

Parallelism and Branching

- Decompose search tree
- Focus on vertices of highest degree
- Exploit all available processors
- Use SSH initially
- Minimal communication required


Level 0
Level 1
Level 2
$G_{0,0}$
v1
N(v1)
$G_{1,0}$
$G_{1,1}$
v2
N(v2)
v3
N(v3)
$G_{2,0}$
$G_{2,1}$
$G_{2,2}$
$G_{2,3}$
p1
p2
p3
p4

Intriguing Results on Synthetic Graphs of Size 600

Graph name	Sequential Reduction	Sequential branching	Parallel branching
RG30	1 second	Halted after two days	5 sec
RG31	1 second	Halted after two days	4 sec
RG32	1 second	Halted after two days	4 sec

What can explain such super-super-linear speedup?

- Caching effects? Lucky distributions?

What can explain such super-super-linear speedup?

- Caching effects? Lucky distributions?
- Neither. These are synthetic graphs (from Carleton). Rectangular grids. It turns out that the first processor gets a small subgraph with the solution.

What can explain such super-super-linear speedup?

- Caching effects? Lucky distributions?
- Neither. These are synthetic graphs (from Carleton). Rectangular grids. The first processor gets a small subgraph with the solution.
- Lesson: never trust synthetic data!

Implementation Problems

- Sensitivity to data and rule order

Implementation Problems

- Sensitivity to data and rule order
- An occasional observance of super-linear speedup on “yes” instances is more than offset by numerous super-linear slowdowns on “no” instances

Implementation Problems

- Sensitivity to data and rule order
- An occasional observance of super-linear speedup on “yes” instances is more than offset by numerous super-linear slowdowns on “no” instances
- Resultant lack of scalability

Implementation Problems

- Sensitivity to data and rule order
- An occasional observance of super-linear speedup on "yes" instances is more than offset by numerous super-linear slowdowns on "no" instances
- Resultant lack of scalability
- Solution: use dynamic decomposition to achieve a form of load balancing

Representative Results on Large Non-Synthetic Graphs

Graph Name	Graph Size	Cover Size	Instance Type	Sequential Kernelization	Sequential Branching	Parallel Branching	Dynamic Decomposition
SH2-5	839	399	Yes	34 seconds	7 seconds	Not needed	Not needed
SH2-5	839	398	No	34 seconds	141 minutes	82 minutes	20 minutes
SH3-10	2466	2044	Yes	203 minutes	~ 5 days	~ 5 days	140 minutes
SH3-10	2466	2043	No	203 minutes	6+ days	6+ days	620 minutes

The Case for Advanced Technologies

- SSH requires great care to manage memory, job queues, processor activation and loading

The Case for Advanced Technologies

- SSH requires great care to manage memory, job queues, processor activation and loading
- Now exploring Condor, NetSolve

The Case for Advanced Technologies

- SSH requires great care to manage memory, job queues, processor activation and loading
- Now exploring Condor, NetSolve
- Other possibilities are Globus, Harness

The Case for Advanced Technologies

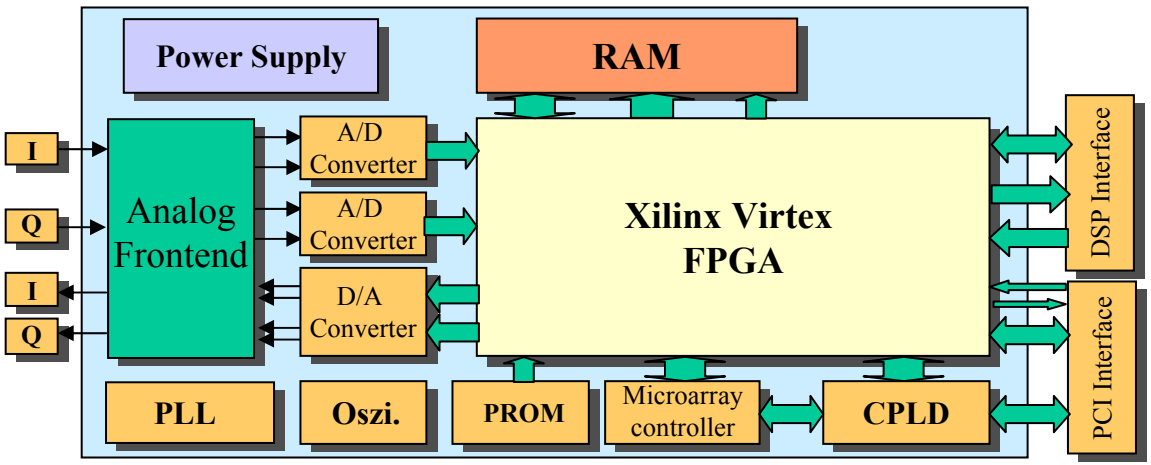
- SSH requires great care to manage memory, job queues, processor activation and loading
- Now exploring Condor, NetSolve
- Other possibilities are Globus, Harness
- Even supercomputers are a possibility, for example, the Cray X1 (Phoenix)

Reconfigurable Computing

- Hardware acceleration and prototyping
- Not important for kernelization
- Very useful for branching
- Inherently parallel
- Implemented on 8-node Pilchard system
- VHDL, circuit simulation, synthesis, etc.

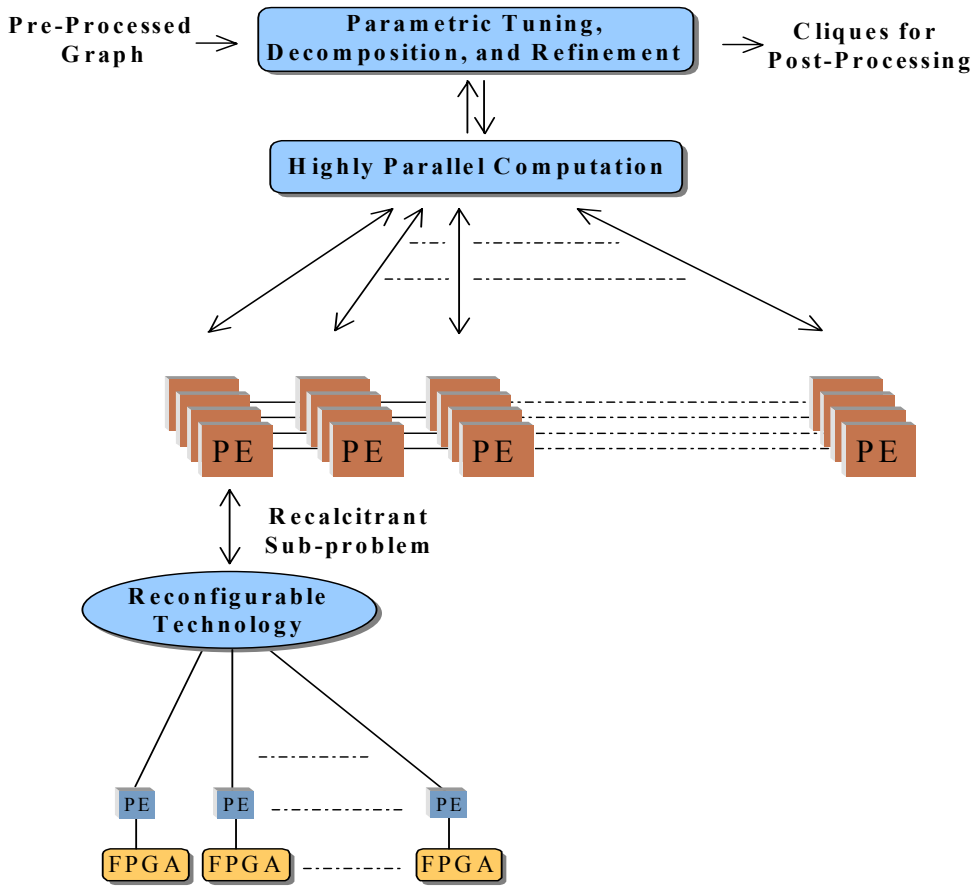
Reconfigurable Computing

Realized with the FPGA



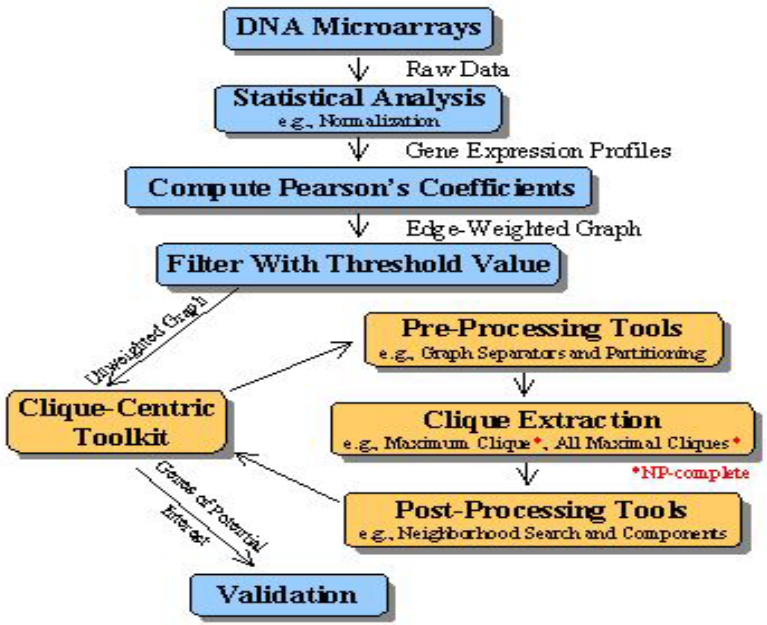
Source: http://www.ant.uni-bremen.de/whomes/haase

The Clique Compute Engine



Variety of Clique Applications

- Co-regulation (Rob Williams)



Variety of Clique Applications

- Co-regulation (Rob Williams)
- Data Mining (Jay Snoddy, Stefan Kirov, Bing Zhang)

Variety of Clique Applications

- Co-regulation (Rob Williams)
- Data Mining (Jay Snoddy, Stefan Kirov, Bing Zhang)
- Motif and Module Deduction (Brynn Voy, Mike Leuze)

Variety of Clique Applications

- Co-regulation (Rob Williams)
- Data Mining (Jay Snoddy, Stefan Kirov, Bing Zhang)
- Motif and Module Deduction (Brynn Voy, Mike Leuze)
- Phylogeny (Jim Cheetham, Frank Dehne)

Variety of Clique Applications

- Co-regulation (Rob Williams)
- Data Mining (Jay Snoddy, Stefan Kirov, Bing Zhang)
- Motif and Module Deduction (Brynn Voy, Mike Leuze)
- Phylogeny (Jim Cheetham, Frank Dehne)
- SELDI (Halima Bensmail, Ali Haoudi)

Variety of Clique Applications

- Co-regulation (Rob Williams)
- Data Mining (Jay Snoddy, Stefan Kirov , Bing Zhang)
- Motif and Module Deduction (Brynn Voy, Mike Leuze)
- Phylogeny (Jim Cheetham, Frank Dehne)
- SELDI (Halima Bensmail, Ali Haoudi)
- Protein Interaction Networks (Nagiza Samatova, Hoony Park)

Other Amenable Problems

- Hitting Set
- Dominating Set
- Cutwidth, Others…

Recent Codes Released

- CAMDA (Disease Screening)
- Clustal XP (High Performance, Parallel Clustal W)

Special Acknowledgment to my Great Team of Students

Faisal Abu-Khzam, Nicole Baldwin,
Rebecca Collins, Mahesh Dorai,
John Eblen, Lan Lin, Daniel Lucio,
Jon Scharff, Xinxia Peng,
Pushkar Shanbhag, Yongling Song,
Henry Suters, Chris Symons, Ian Watkins

PhD, CS

PhD, µBio

Ed

Jay

PhD, Math

Nagiza