Research Points – Identification & Design

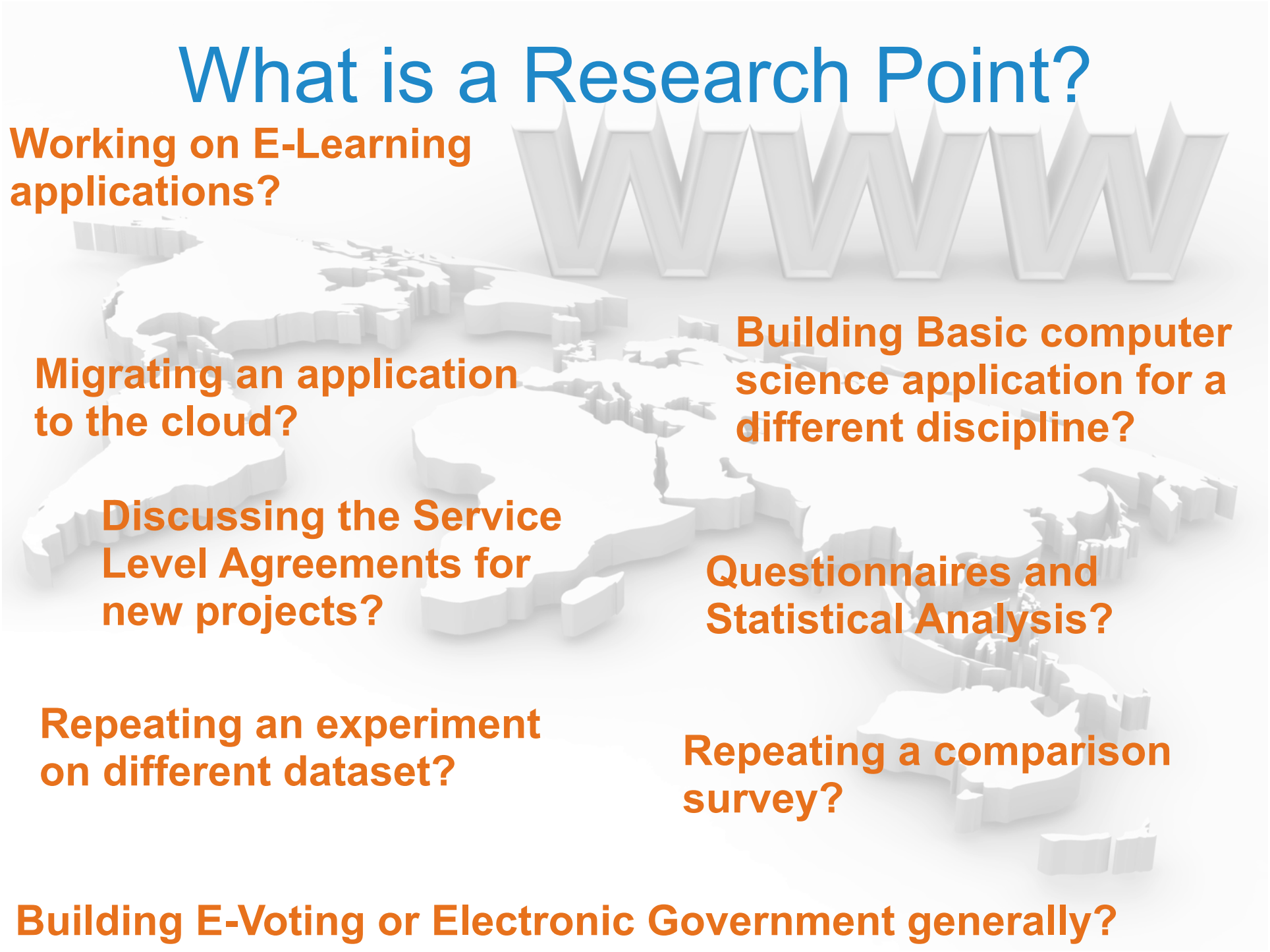
Dr. Manal Helal
CCIT – AASTMT
19/1/2013

Agenda

1. Identifying a research question
2. Planning a research Project
3. ✓ My Research Interests
   - a. Computational Science
   - b. Parallel Processing
   - c. Optimization
   - d. AI Interests
     - i. Data mining
     - ii. Knowledge Bases
     - iii. Ontology Building
     - iv. Machine Learning & Reasoning
   - e. Bioinformatics

Research Question & Plan

What is a Research Point?



What is a Research Question?

Where to find, and how to Develop?

- Personal Interests:
  - Identify Journals and Conferences in the research topic and their impact factors and keep up with their call for papers.
  - Read in the latest publications in the interesting topic
- National Problems:
  - Problems that has solutions internationally are not research problems, unless the adaptation to Egyptian requirements is challenging.
- Worldwide Grand Challenges:
  - A grand challenge is a fundamental problem in science or engineering, with broad applications, whose solution would be enabled by the application of high performance computing resources that could become available in the near future.

What is a Research Question?

Where to find, ***and how to Develop?***

- Research GAPS:
  - Through reading books, publications, internet, and colleagues discussions, identify gaps in the literature to formulate questions that need further investigation. This usually requires a few iterations.
- Fill the GAP:
  - Identifying a solution hypothesis to fill in the gap or optimize an existing solution.
- Design Experiments:
  - Design the experiment to test your hypothesis or suggested methods.
  - Collect Results and validate
- Draw conclusions & Identify Future Work Ideas.

Grand Computational Challenges Examples

1 Symbolic computations Topics

•speech recognition,
•computer vision,
•natural language understanding,
•automated reasoning, and
•tools for design, manufacturing, and simulation of complex systems."

2 The Human Genome Project

The word genome refers to all the DNA in an organism, including its genes. The Human Genome Project aims to discover all the 100,000 human genes, to determine the complete sequence of the 3 billion DNA subunits that make up the genome, to develop data-analysis and sequencing tools, and to make this information accessible for further biological study

Grand Computational Challenges Examples

3 Ground Water Contamination

In order to study ground water contamination, complex chemical and physical interactions must be modeled. Approximations must be made because the exact properties of a contaminated site are unknown. The model will be divided into more than 100,000 grid blocks, each describing a small geographic area. Within each grid block, equations will describe the behavior of gases and liquids as they interact and move from location to location. Some blocks may represent locations in a body of water, while others will model liquid moving through dirt or seeping through cracks in rock. Research work should design remediation methods that work, and must recommend methods that can be carried out at the least cost.

4 Simulation of X-Ray Clusters

By comparing numerical simulations and the real universe, scientists hope to learn more about the composition and distribution of the mysterious dark which pervades the universe. X-ray clusters are clusters of galaxies immersed in halos of million-degree gas which emit energy in the form of X-rays. Astronomers study X-ray clusters because they map out the large-scale structure of the universe. Scientists at the Laboratory for Computational Astrophysics, National Center for Supercomputing Applications, at the University of Illinois at Urbana Champaign studied the formation of X-ray clusters using numerical simulations running on massively parallel computers. Their model represented a cube 500 million light years on each side. The cube was divided into a network of 134 million smaller cubes, each approximately one million light years on a side. In each cell, they solved the equations of hydrodynamics (these deal with the motions of the gas) to predict the behavior of gas density, pressure, temperature, and volume.

Funding

Sources & Applications

Egyptian National S&T Information Network

1

- •GERSS-German-Egyptian Research Short term Scholarships
- •EGYPT-SOUTH AFRICA JOINT SCIENCE & TECHNOLOGY RESEARCH PROGRAMME
- •EU-Egypt Science and Innovation
- •Science and Technology Development Fund (STDF)

World-Wide Universities

2

- •Funded Research Projects
- •Fulbright Egypt
- •EURAXESS
- •FP7
- •Marie Curie

Industrial Research & Development

3

Companies such as Google, Microsoft, Oracle, and other specialised companies announce research funding and sponsorships

- Planning A Masters topic that can lead to a PhD topic is better.
- Develop a research plan for 5 or 7 years.
- Developing research plan tailored to funding requirements, otherwise work on your own.
- Focus on one or a few related domains of applications and methods. Don't work on many unrelated problems, you won't build an experience this way.
- Collaborate with other experts in other disciplines.

Computer Science Research Methods

- Formal
  - Prove facts about algorithms and system formal specification in order to allow the automatic verification of an implementation of that component.
  - Alternatively, researchers may be interested on the time or space complexity of an algorithm, or on the correctness or the quality of the solutions generated by the algorithm.

Computer Science Research Methods

- Experimental
  - Used in CS to evaluate new solutions for problems.
  - Experimental evaluation is often divided into two phases.
    - In an exploratory phase the researcher is taking measurements that will help identify what are the questions that should be asked about the system under evaluation.
    - Then an evaluation phase will attempt to answer these questions. A well-designed experiment will start with a list of the questions that the experiment is expected to answer.

Computer Science Research Methods

- Build
  - A “build” research methodology consists of building an artifact:
    - either a physical artifact or a software system to demonstrate that it is possible.
    - To be considered research, the construction of the artifact must be new or it must include new features that have not been demonstrated before in other artifacts.

Computer Science Research Methods

- Process
  - A process methodology is used to understand the processes used to accomplish tasks in Computing Science.
  - This methodology is mostly used in the areas of Software Engineering and Man-Machine Interface which deal with the way humans build and use computer systems.
  - The study of processes may also be used to understand cognition in the field of Artificial Intelligence.

Computer Science Research Methods

- Model
  - The model methodology is centered on defining an abstract model for a real system. This model will be much less complex than the system that it models, and therefore will allow the researcher to better understand the system and to use the model to perform experiments that could not be performed in the system itself because of cost or accessibility.
  - The model methodology is often used in combination with the other four methodologies. Experiments based on a model are called simulations. When a formal description of the model is created to verify the functionality or correctness of a system, the task is called model checking.

Research Plan

From Start to Finish

Finding a GAP

1

Find a gap in literature of your interest areas, for a problem with no solutions, or drawbacks in solutions that you can address.

You can also look for a grand challenge.

Proposing an Original Method

2

Research existing solutions, and critically compare and analyse their performance. Sometimes merging with new emerging technologies is the method, and sometimes simple try and error can formulate a new method.

Validation and Conclusion

3

Design your experiment, execute it and gather results. Analyze the results using known validation and verification methods used in the problem domain, compare with previous solutions, identify limitations, and draw conclusion.

My Research Interests

My Research Interests

1. Computational Science
2. Parallel Processing & Algorithms
3. Operation Research & Optimisation
4. Artificial Intelligence
   - a. Data Mining & Knowledge Bases
   - b. Building Ontology, Machine Learning & Reasoning
5. Bioinformatics

Computational Science

What is Computational Science?

- With computers, scientists and engineers have made numerous discoveries that they would not have made otherwise. In fact, computers have revolutionized the way that many scientists do their work.
- Solve $2x + 5 = 7$. Sure, that's easy enough. Or solve a system of two equations and two unknowns like

$$2x + 5y = 17$$

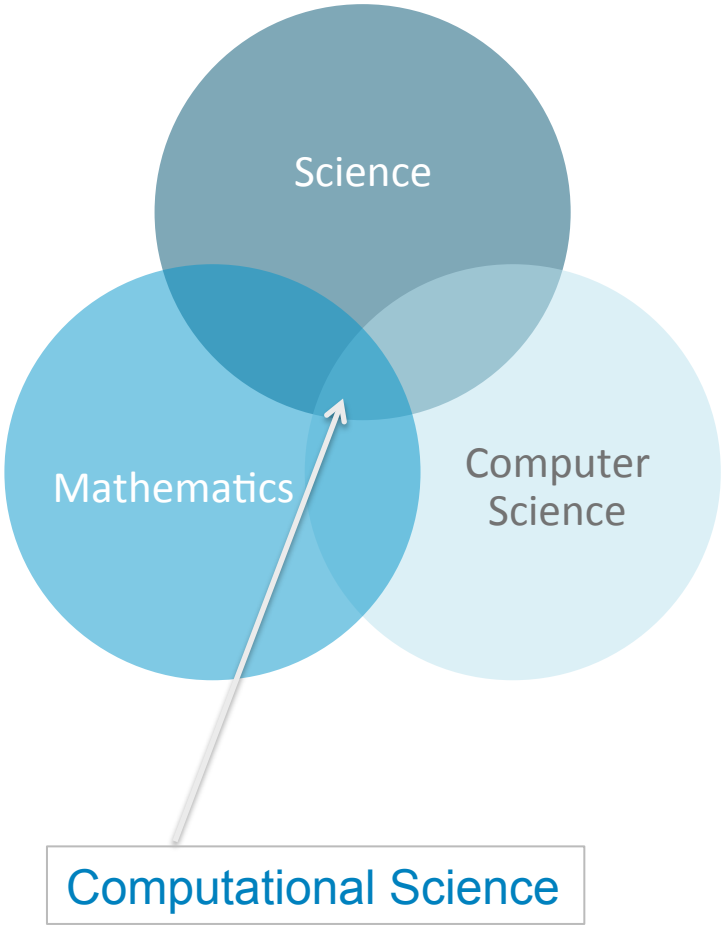
$$3x - 2y = 4$$

- You don't need a computer for that. But imagine solving a problem by hand with 3 million variables. That's how many are required in the Spectral Element Ocean Model1, a vast computer simulation, that tests the wind's effects on the Earth's oceans. Problems of this magnitude, common in today's science, wouldn't be possible without computers.

What is Computational Science?

- Traditionally, science was done in a laboratory as a combination of theory and physical experimentation (which included hand calculations), but computers have made possible a new and powerful way of doing science -- numerical simulation -- that augments the old.
- Numerical simulation is the process of modeling mathematically a physical phenomenon, and then running an experiment with the mathematical model. Computational mathematicians or computational scientists play a major role in this new way of doing science, creating, evaluating, and refining the mathematical models used to simulate the physical phenomena.
- Simulation can be used when physical experiments are too costly, time consuming, dangerous, or even impossible.

Computational Science?



1. Computational science is an interdisciplinary field at the intersection of three domains: mathematics, computer science, and the biological and physical sciences. The computational scientist uses tools from computer science and mathematics to study problems from physical science, social science, engineering, etc.
2. Most of the problems that computational scientists work on involve vast amounts of data and a large number of variables. Through the advances in computer technology and numerical methods, mathematicians and scientists are able to work together modeling and solving problems that were impossible to address ten years ago.
3. Computational scientists do more than use a computer to find solutions to mathematical models developed from scientific problems however. They also develop new mathematical tools and theory and develop new numerical methods and improve the accuracy and speed of existing methods.

Parallel Processing & Algorithms

HPC – Computational Science

- Most large scale scientific computing performed is some form of simulation
  - Simulation can always use
    - A few more objects to simulate
    - Smaller timesteps
    - More timesteps
    - More precision
    - Real time behavior
      - Computational steering

Grand challenge: modeling the Sun

- Why?
  - Solar flare prediction, improve general physics, etc
  - There exist a number of theoretic models for the Sun
  - There are numerous observations (X-ray/visual/magnetic)
- Which model is (most) correct?
  - Processes not well understood
  - Simulation is the only way to tell…
    - 3D and O(1024^3), 60TByte memory,4000FP's per grid point
    - Multipeta/teraflop range computing.

Scientific Simulations

- Fixed Time Step
  - Integration over time
  - Each time step, all individuals in the simulation are updated by advancing ‘simulated time’ by a constant delta
- Variable time step:
  - Integration over time
  - Whenever the simulation becomes ‘interesting’ take smaller time steps

Monte Carlo Simulations

- Start with “random” or “reasonable” initial conditions
  - For example, place simulated individuals somewhere in a grid
- Pick a random individual
  - Move in random way
  - Check if movement is allowed
    - If allowed, update the whole system to take movement into account.
    - If not allowed, take back movement as if it didn’t happen
- http://sic.epfl.ch/SA/publications/SCR95/7-95-21a.html

Finite Element Methods

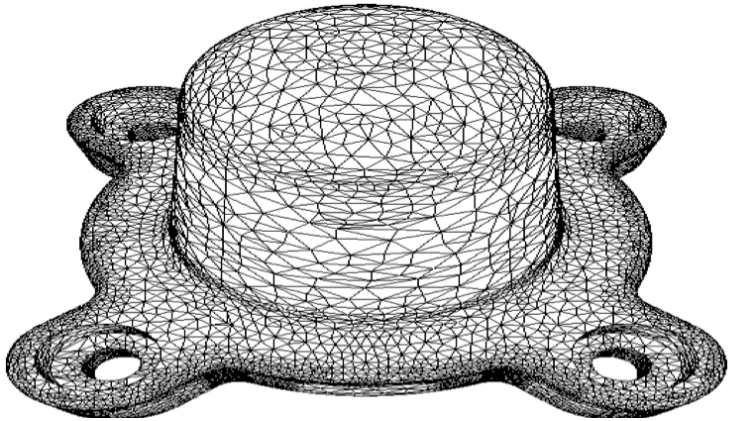
- The finite element method (FEM) (its practical application often known as finite element analysis (FEA)) is a numerical technique for finding approximate solutions to partial differential equations (PDE) and their systems, as well as (less often) integral equations. In simple terms, FEM is a method for dividing up a very complicated problem into small elements that can be solved in relation to each other. FEM is a special case of the more general Galerkin method with polynomial approximation functions. The solution approach is based on eliminating the spatial derivatives from the PDE. This approximates the PDE with
  - a system of algebraic equations for steady state problems,
  - a system of ordinary differential equations for transient problems.
- These equation systems are linear if the underlying PDE is linear, and vice versa. Algebraic equation systems are solved using numerical linear algebra methods. Ordinary differential equations that arise in transient problems are then numerically integrated using standard techniques such as Euler's method or the Runge-Kutta method.

Finite Element Methods

- What is a finite element?
  - Take a continuum model
    - Discretize.
    - Limit size of continuum
  - Each element of discretized continuum is a Finite element
  - Useful if
    - Global continuum system is too complex
      - Break it down into ‘primitive elements’
      - Simulate the primitive elements separately (divide & conquer style)
      - Sum the effects of the individual parts somehow to approximate the continuum

Geographic partitioning

- Partition data geographically:
  - Give each processor the same amount of 'space' to work on
    - Not fair because some parts of the data can be more computationally intensive than others!
    - Does not take communication patterns into account



Geographic partitioning

```
List< list<Node> > partitions_per_cpu;
void partition(int num_cpus, list<Node> g) {
    If (num_cpus == 1) {
            partitions_per_cpu += g;
            return;
    } If (x_distance_is_largest(g)) {
            list<Node> list = g.nodes_sorted_over_x();
            list<Node> first_half = list[0 .. list.length/2];
            list<Node> second_half = list[list.length/2 .. list.length];
            partition(num_cpus/2, first_half);
            partition(num_cpus/2,  second_half);
    } else if (y_distance_is_largest(g) {
            // symmetric to 'x' ...
    } else {
            // symmetric to 'x' ....
            assert(z_distance_is_largest(g));
    }
}
```

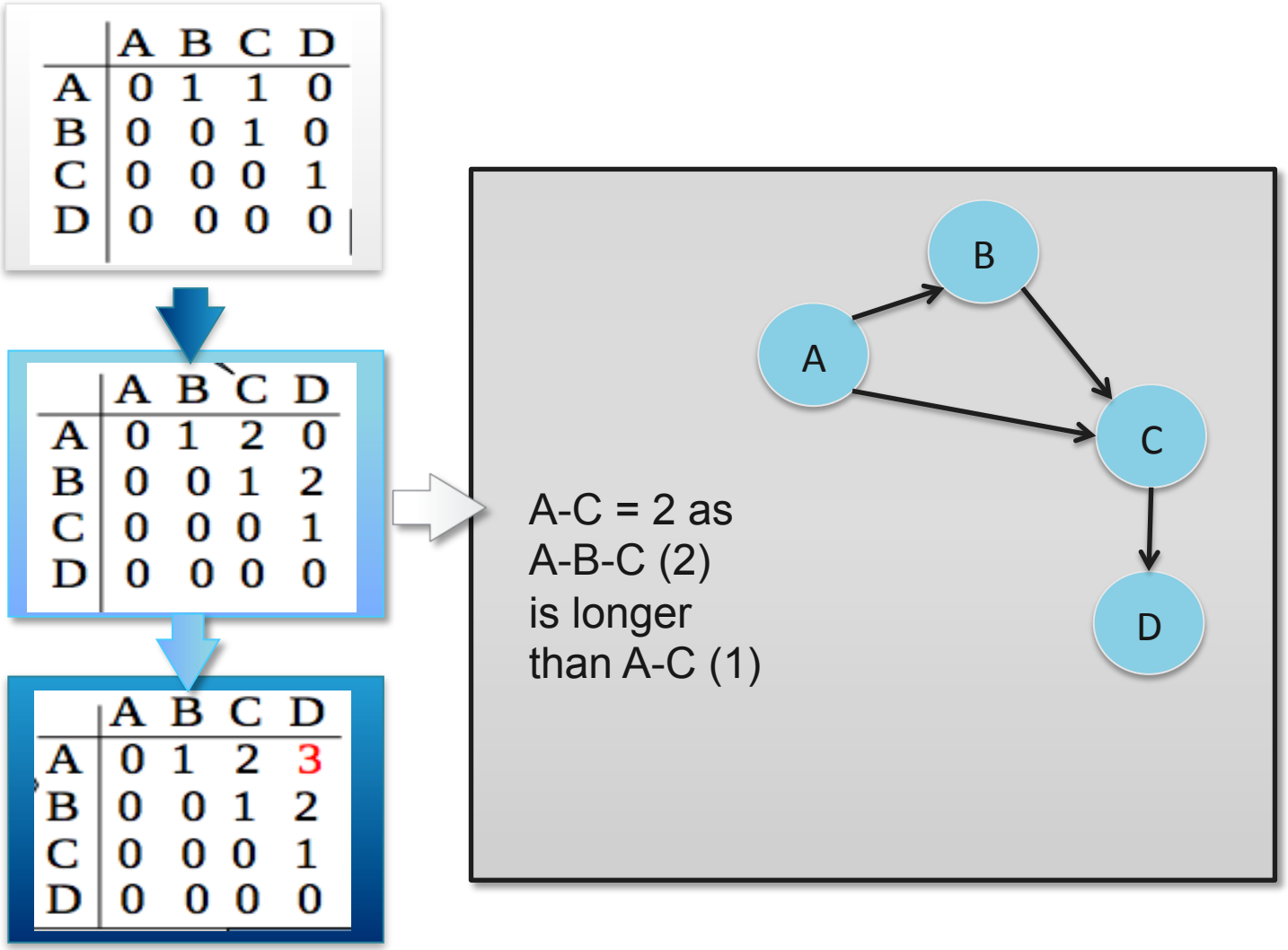
Graph partitioning

- Find the longest path in the graph, cut in half, and recursively apply to partitions:

```
List< list<Node> > partitions_per_cpu;
void partition(int num_cpus, list<Node> g) {
    If (num_cpus == 1) {
        partitions_per_cpu += g;
        return;
    }
    list<path> p = find_longest_path(g);
    list<Node> first_half = list[0 .. list.length/2];
    list<Node> second_half = list[list.length/2 .. list.length];
    partition(num_cpus/2, first_half);
    partition(num_cpus/2, second_half);
}
```

Graph Partitioning

Finding the longest path in a graph



Finding the longest path in a graph

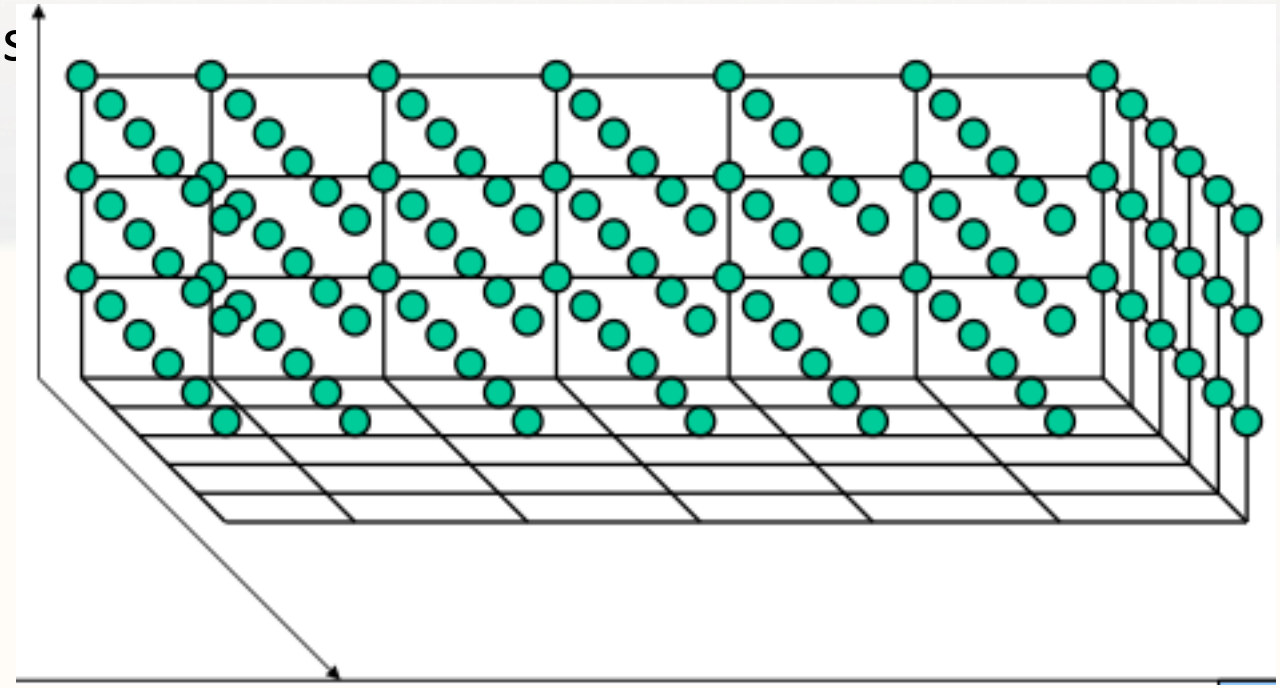
```
int distances[N,N];     // initialized with '0'
Path paths[N,N];        // a path is a list of nodes, each path has a
                        // length (#nodes)
Path best;              // best path found thus far
Path find_it(DirectedEdge[N] edges) {
    for each edge E in edges:
        distances[E.from, E.to] = 1;
        path[E.from, E.to] = path(E.from, E.to);
    for x = 0 to N:
        for y = 0 to N:
            for z = 0 to N:
                If (distances[x,y] > 0 && distances[y,z] > 0) {
                    If (distances[x,y] + distances[y,z] > distances[x,z]) {
                        distances[x,z] = distances[x,y] + distances[y,z]
                        paths[x,z] = paths[x,y].append(path[y,z]);
                        If best.length < paths[x,z].length:
                            best = paths[x,z];
                    }
                }
}
```

Atmosphere Modeling (1)

- Simulate wind, clouds, precipitation, etc that influence wind & weather
- Uses basic physics (mechanics, fluid property formulas)
  - Conservation of mass, energy and momentum
  - Hydrostatic approximation
  - Gas state equations
    - pressure = density * temperature * height

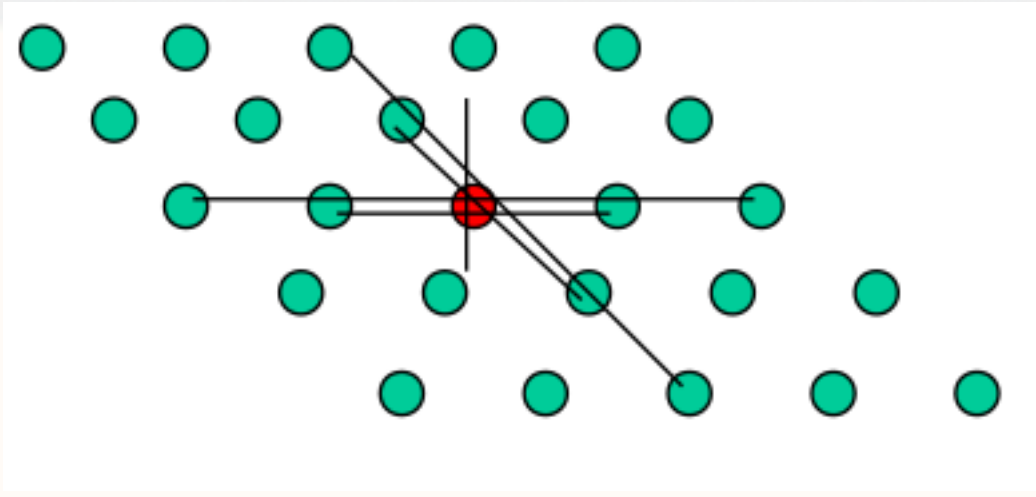
Atmosphere Modeling (2)

- First try, put every thing on a 3D grid
  - Each grid point = 1 task
    - Note: points



Atmosphere Modeling (3)

- Every grid point
  - Communicates with 11 others
  - Most communication is horizontal



Atmosphere Modeling (4)

- Agglomeration
  - Each grid point = 1 task
    - Nx*Ny*Nz tasks
      - Too many
  - Most communication is horizontal, thus agglomerate mostly horizontally
- Load imbalances
  - At night no radiation in physics model
  - Clouds only at threshold humidity
- Question: is this a finite element simulation ?

Other Parallel Computational Problems

- Particle Simulation
- Rendering
  - Parallel raytracing
    - Simulate individual ‘rays’ of light
  - Radiosity rendering
    - Simulate light as an amount of energy that is emitted by each surface
- Fluid-Dynamics
- Fourier methods
- Wavelets

Cloud Computing Parallel Processing

- The cloud is mostly used for Distributed Processing.
- We are investigating the cloud reliability for parallel processing:
  - Check-pointing / restarting mechanisms
  - Static & Dynamic Data Partitioning
  - Scheduling & load balance

GPU Programming

- Parallelizing interesting algorithms to benefit from GPU technologies

Operation Research & Optimization

Roads Design Optimization

- A Macroscopic traffic flow model is a mathematical model that formulates the relationships among traffic flow characteristics like density, flow, mean speed of a traffic stream, etc.
- Studying major Egyptian cities traffic flow and optimizing major road design decisions such as: U-Turns, round-about, roads intersections, high-way exits distances, … etc.

Artificial Intelligence

Arabic Lexicon

- Building a complete Arabic lexicon for use in building knowledge bases, Arabic Ontology, text and data mining.

Legal Ontology

- Building a legal ontology for the Egyptian constitution and laws.

AI Reasoning and Learning

- Legal automatic reasoners to model the change of a constitution subject and its cascades to the relevant laws.

Bioinformatics

Bioinformatics

- Bioinformatics applies algorithms and statistical techniques to the interpretation, classification and understanding of biological datasets.
- These datasets typically consist of large numbers of DNA, RNA, or protein sequences. Sequence alignment is used to assemble the datasets for analysis.
- Multiple Sequence Alignment (MSA) is one of the most important computational biology problems, whose optimal methods are still an active area of research.

Bio Computing (1)

- Has large computational requirements
  - DNA sequence alignment
  - Protein database search
  - Molecule matching (see if molecule X can be attached to molecule Y)

Bio Computing (2)

- DNA sequence alignment
  - DNA scanning machines deliver chunks of dna strings
    - We want the large complete string, not the fragments
  - Dna scans deliver large amounts of DNA fragments
  - DNA encoded as string of base pairs (A, C, T, G)
  - Human has 48 chromosomes, *$3*10^9$ bases

Bio Computing (3)

- DNA sequence alignment example:

- Have string

  ACTGAGCTTCAC

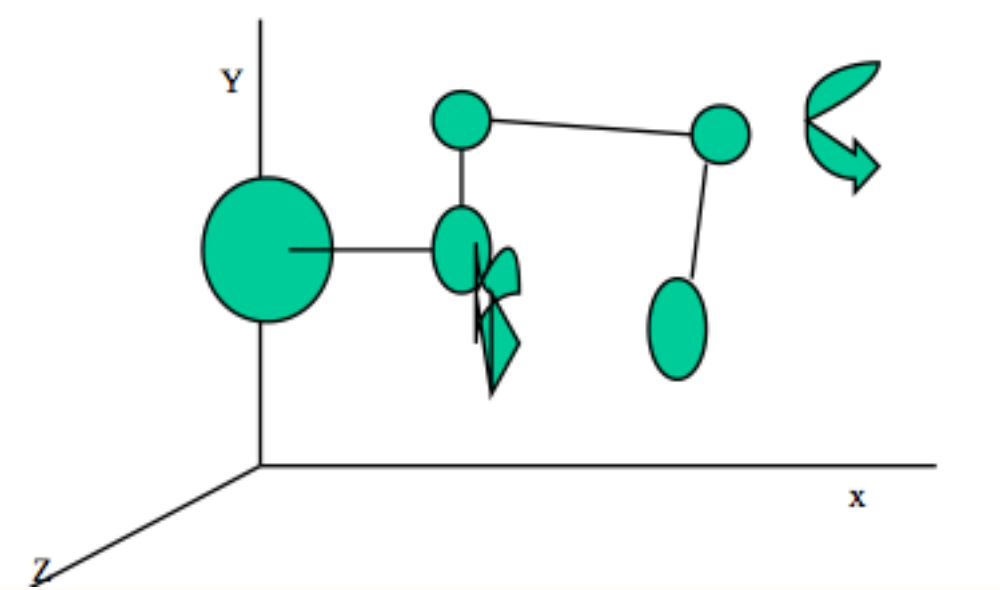
- And string

  CACAGAGTATC

- Head-tail match, thus make a larger string.
  - use probability that it's the correct match before making the decision to merge
  - potentially large numbers of possible matchesto consider
  - 3 Gbytes of input * N times for maintaining probable matches....

Bio Computing (4)

- Protein Folding problem
  - Given a sequence of amino-acid molecules, find the least energy 3D configuration
    - C3OH3CHCOGCS3.....



Bio Computing (4a)

- When able to predict the correct stable folding of an arbitrary protein
  - Can see if it 'fits' inside another molecule
    - If fit then possible medicine (protein blocker for other protein)
  - See if surface properties equal to other molecule in 3D
  - Etc.

Bio Computing (5)

- Protein Folding
  - Partially embarrassingly parallel
    - All possible foldings can be tried in parallel
      - Misses cut-offs
      - Testing if a state is possible is non trivial...

Bio Computing (6)

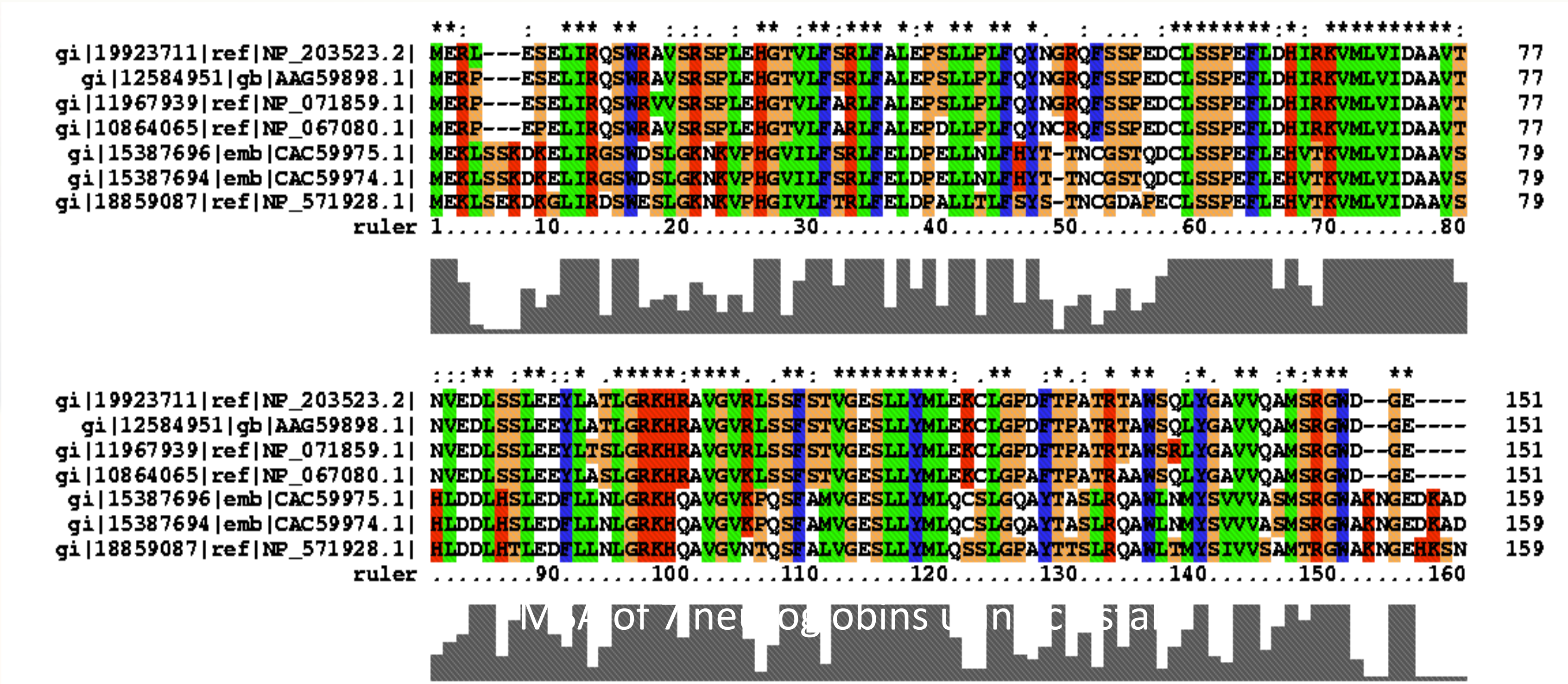
```
SequentialFindMinimumEnergyConfiguration(mol) {
    Queue = empty
    Min_energy=energy(mol)
    Min_config=mol
    Put mol in queue
    while not queue is empty
        m = queue.get();
        for I=0 to #joints in m
            m'=twist joint I in m
            if(m' is valid configuration)
                put m' in queue
                if energy(m') < min_energy
                    min_energy = energy(m')
                    min_config = m'
```

Bio Computing (7)

```
ParallelFindMinimumEnergyConfiguration(molecule mol)
    Queue = empty
    Min_energy=energy(mol)
    Min_config=mol
    Put mol in queue
    Parallel while not queue is empty
        m = queue.get();
        for I=0 to #joints in m
            m' = twist joint I in m
            if (m' is valid configuration)
                put m' in queue
                if energy(m') < min_energy
                    min_energy = energy(m')
                    min_config = m'
```

My Previous Work

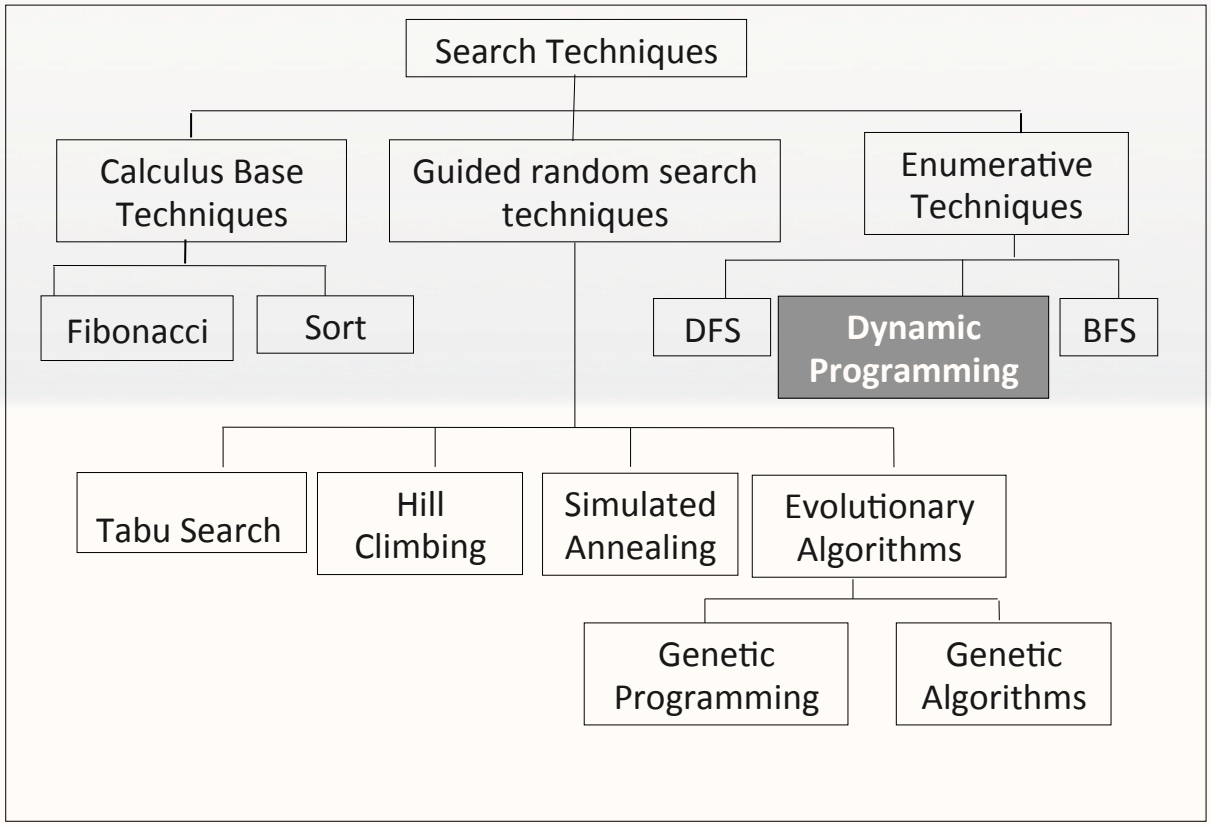
Multiple Sequence Alignment (MSA)



Why MSA?

- Compare a new sequence with the sequences in a protein family.
- Phylogenetic analysis: Gain insight into evolutionary relationships.
- Identify conserved domains/elements in sequences.
- Compare regions of similarity among multiple organisms.
- Identify probes for similar sequences in other organisms.
- Develop PCR primers.

MSA Methods


Search Techniques
Calculus Base Techniques
Guided random search techniques
Enumerative Techniques
Fibonacci
Sort
DFS
Dynamic Programming
BFS
Tabu Search
Hill Climbing
Simulated Annealing
Evolutionary Algorithms
Genetic Programming
Genetic Algorithms

Problems with Existing MSA Methods

- Assume Min. percent identity of ~40% for proteins and ~70% for DNA, otherwise, much higher likelihood of errors.
- Sensitive to sequences input order.
- Depend on pair-wise alignments, which is less sensitive and cause bias in the positioning of gaps.
- Statistical Uncertainty.
- Assume conserved order of aligned residues. ABA, ProDA, TBA, MAUVE don't assume this.
- Care must be made in choosing scoring matrices and penalties.

2D - Dynamic Programming MSA

Seq1: ATCGCGTATGC

Seq2: ATTCGGCTATCGGC

			-	A	T	C	G	C	G	T	A	T	G	C	
			0	1	2	3	4	5	6	7	8	9	10	11	
-	0	< <	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22	-24	>
A	1	<	-4	-4	-6	-8	-10	-12	-14	-16	-18	-20	-22	-24	>
T	2	<	-6	-6	-2	-4	-6	-8	-10	-12	-14	-16	-18	-20	>
T	3	<	-8	-8	-4	0	-2	-4	-6	-8	-10	-12	-14	-16	>
C	4	<	-10	-10	-6	-2	0	-2	-4	-6	-6	-8	-10	-12	>
G	5	<	-12	-12	-8	-4	0	0	0	-2	-4	-6	-8	-10	>
G	6	<	-14	-14	-10	-6	-2	2	0	2	0	-2	-4	-6	>
C	7	<	-16	-16	-12	-8	-4	0	2	2	2	0	-2	-2	>
T	8	<	-18	-18	-14	-10	-6	-2	2	2	2	2	0	-2	>
A	9	<	-20	-20	-16	-12	-8	-4	0	2	4	2	4	2	>
T	10	<	-22	-22	-18	-14	-10	-6	-2	0	2	6	4	4	>
C	11	<	-24	-24	-20	-16	-12	-8	-4	-2	2	4	8	6	>
G	12	<	-26	-26	-22	-18	-14	-10	-6	-4	0	2	6	8	>
G	13	<	-28	-28	-24	-20	-16	-12	-8	-4	-2	0	4	8	>
C	14	<	-30	-30	-26	-22	-18	-14	-10	-6	-4	-2	2	6	> >

$$S_{ij} = MAX\begin{cases} S_{i-1,j-1} + sub(a_i, b_j) \\ S_{i-1,j} + g \\ S_{i,j-1} + g \end{cases}$$

Aligned Sequences :

>Seq2	A	T	-	C	G	C	G	T	A	T	G	-	-	C
>Seq1	A	T	T	C	G	G	C	T	A	T	C	G	G	C
	1	1	-1	1	1	0	0	1	1	1	0	-1	-1	1

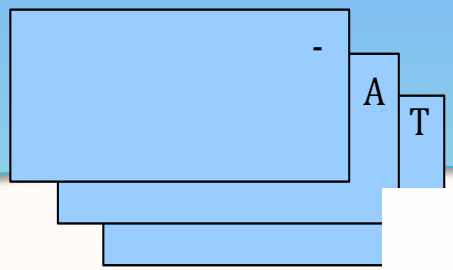
The Alignment Score = 5

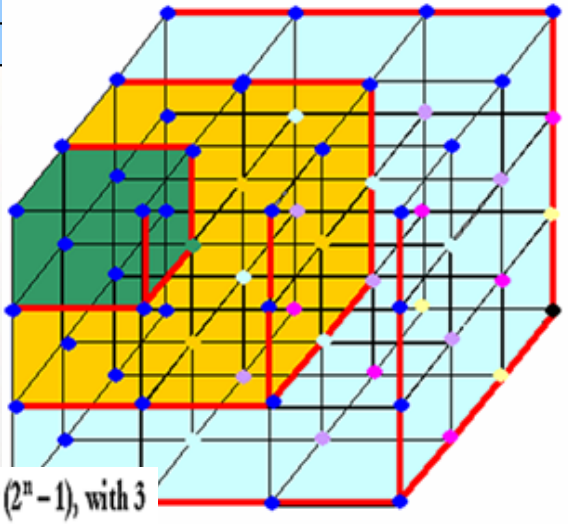
3D DP MSA

			-	A	C	G	T		
			0	1	2	3	4		0 -
-	0	< < <	-2	-4	-6	-8	-10	>	
A	1	<	-4	-8	-12	-16	-20	>	
C	2	<	-6	-12	-18	-24	-30	>	
T	3	<	-8	-16	-24	-32	-40	> >	

			-	A	C	G	T		1 A
-	0	< <	-4	-8	-12	-16	-20	>	
A	1	<	-8	-8	-10	-12	-14	>	
C	2	<	-12	-10	-8	-10	-12	>	
T	3	<	-16	-12	-10	-8	-10	> >	

			-	A	C	G	T		2 T
-	0	< <	-6	-12	-18	-24	-30	>	
A	1	<	-12	-10	-8	-10	-12	>	
C	2	<	-18	-8	-2	-4	-6	>	
T	3	<	-24	-10	-4	-2	-4	> > >	





> seq4	A	C	G	T
> seq3	A	C	T	-
> seq	A	T	-	-
	3	1	-2	-2

The Alignment Score = 0

Lower neighbors to check is growing by $(2^n - 1)$, with 3 sequences, we have 7 neighbors to check as follows:

$A = S_{i-1j-1k-1} + sub(a_i, b_j) + sub(a_i, c_k) + sub(b_j, c_k);$

$B = S_{i-1j-1k} + sub(a_i, b_j) + (2 * g);$

$C = S_{i-1jk-1} + sub(a_i, c_k) + (2 * g);$

$D = S_{ij-1k-1} + sub(b_j, c_k) + (2 * g);$

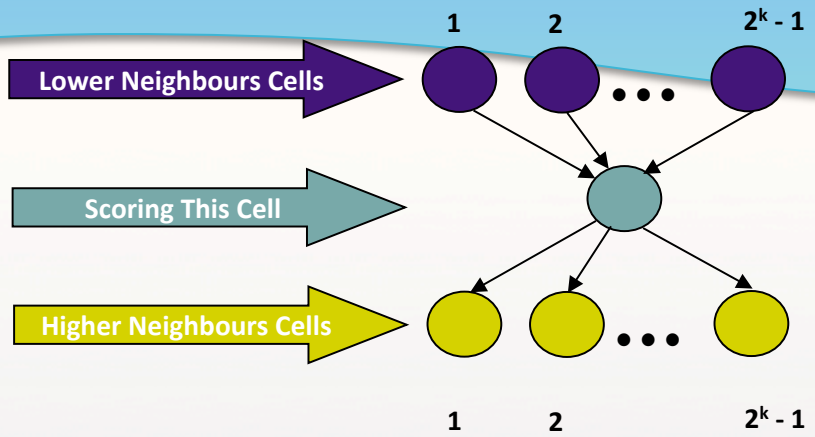
$E = S_{i-1jk} + (2 * g);$

$F = S_{ij-1k} + (2 * g);$

$G = S_{ijk-1} + (2 * g);$

$S_{ijk} = max(A, B, C, D, E, F, G);$

K-D DP MSA



$$S(i_0\, i_1\, i_2\, i_3 \ldots i_k) = max \begin{pmatrix} G_1 + TS\,(G_1) \\ G_2 + TS\,(G_2) \\ : \\ G_{2^k-1} + TS\,(G_{2^k-1}) \end{pmatrix}$$

Where:

$TS\,(G_i)$ = ($sub(d_j, d_k)$ for each pair j, k in G) +($gS * (K-D)$)

G_i: Neighbour i of current cell, up to 2^k-1 neighbours

D: No of decremented indices to get this particular neighbour

TS: Temporary Score function assigned to each neighbour based on how many multidimensional indices were decremented to get to this neighbour

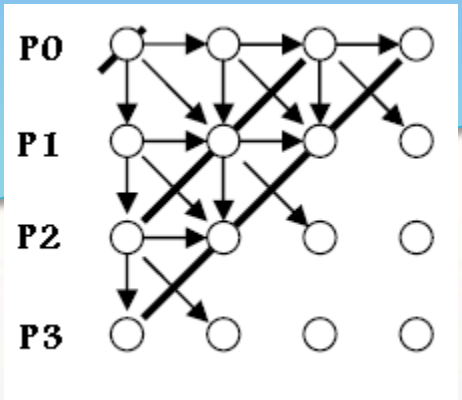
gS: gap Score Value * (K-D): multiply the gap Score Value with number of indices that remained the same (were not decremented to get this neighbour), retrieved by Total Dimensions K (Sequences) – D.

Parallelization Technique

- Distributed MSA based on MOA is designed by retrieving diagonals of partitions that can be scored simultaneously in one wave of computation.

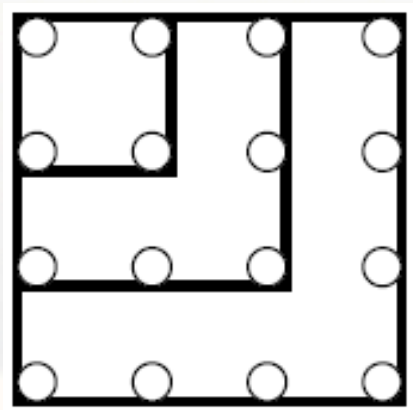
- Their dependencies are computed in an earlier wave of computation, and sent to the waiting processors.

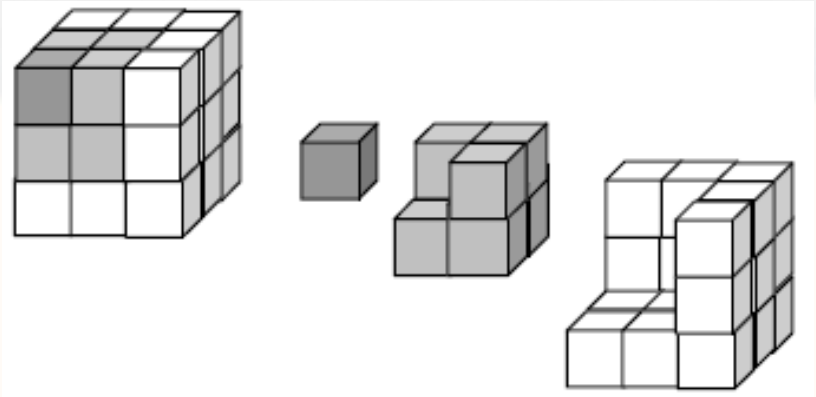
Master / Slave Dependency Analysis



MSA Pair Wise wave-front Dependency: top, left, and left-up diagonal. So, each processor can process a row.

2D MoA MSA Waves Partitions

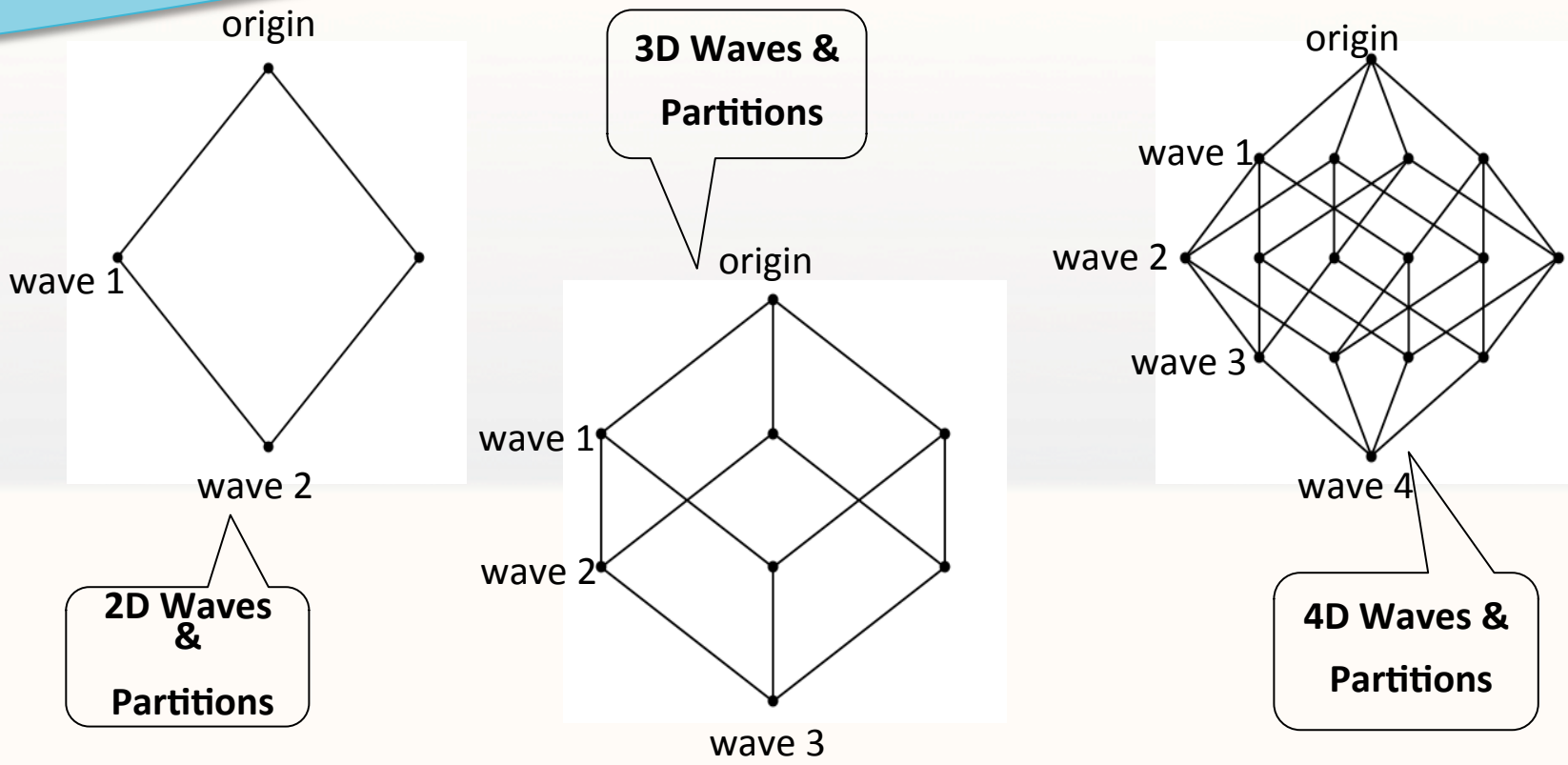




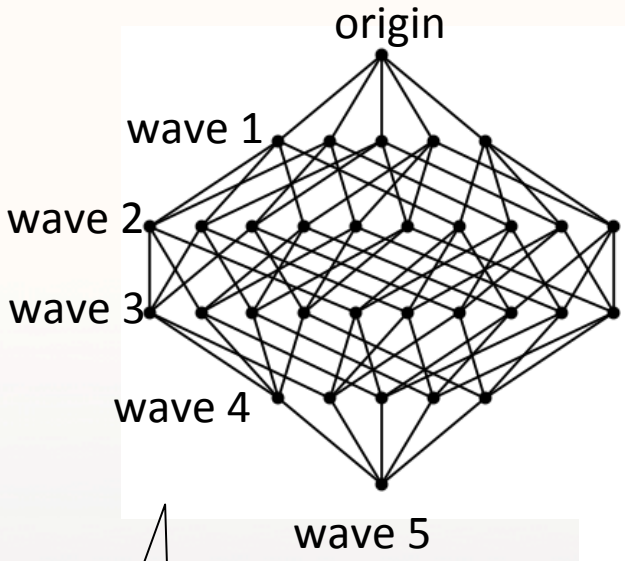
3D MoA MSA Waves Partitions for shape <3 3 3>, and the partitions in each wave are shown independently.

Peer-to-Peer Partitioning

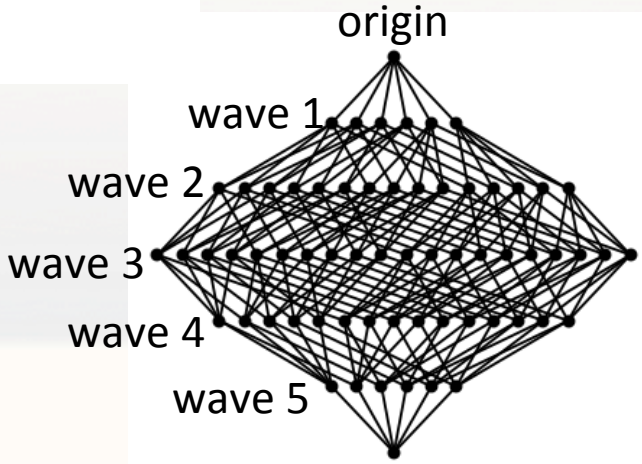
Waves of computations based on clustering partitions on equal distances from the origin as independent (can be computed simultaneously on parallel processors) calculated as :



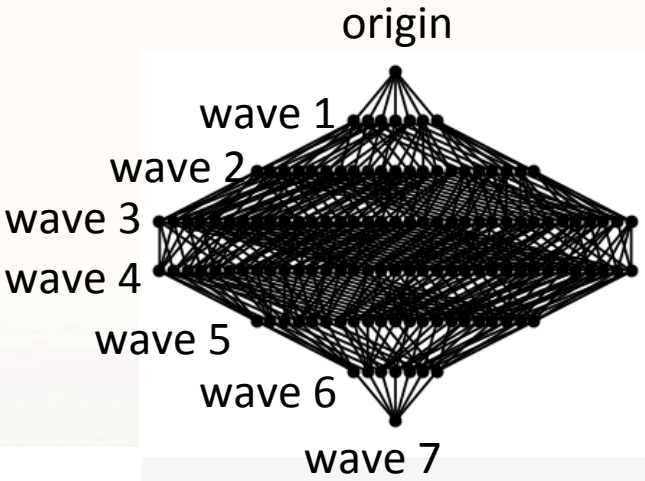
HELAL, M, El-Gindy, H, Mullin, LM, Gaeta, B. 2008. Parallelizing Optimal Multiple Sequence Alignment by Dynamic Programming. *In*: *Proceedings of the International Symposium on Advances in Parallel and Distributed Computing Techniques (APDCT-08) held in conjunction with 2008 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA-08)*. Sydney, Australia: IEEE Computer Society, pp.669-674.



5D Waves & Partitions

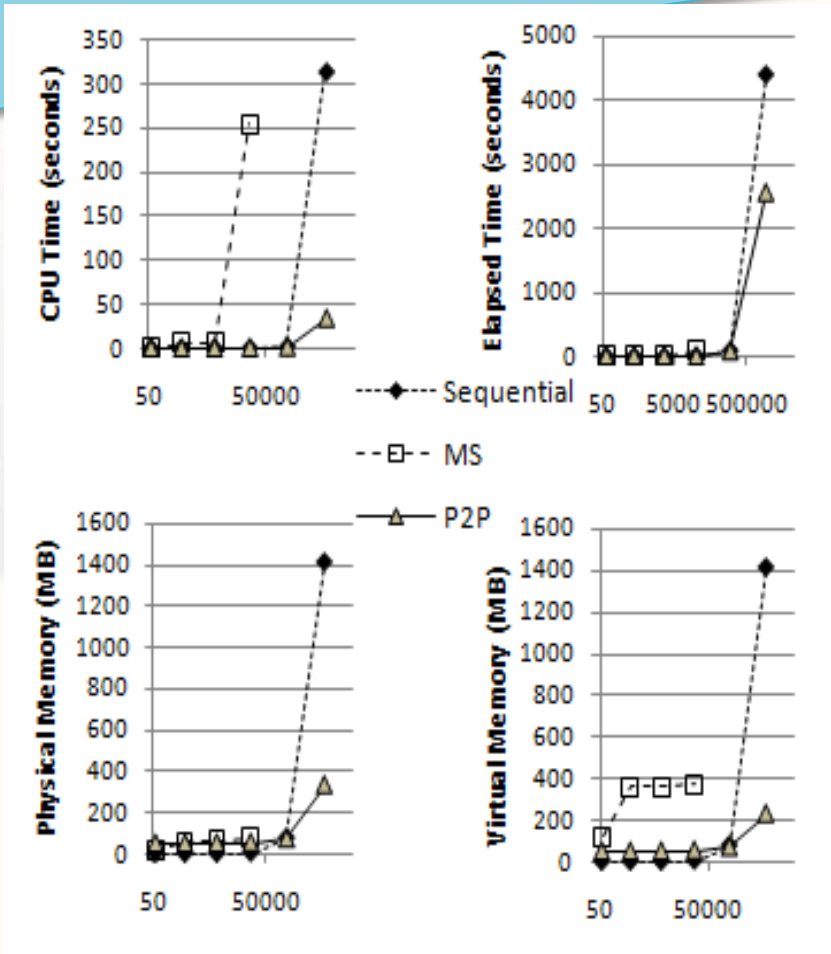


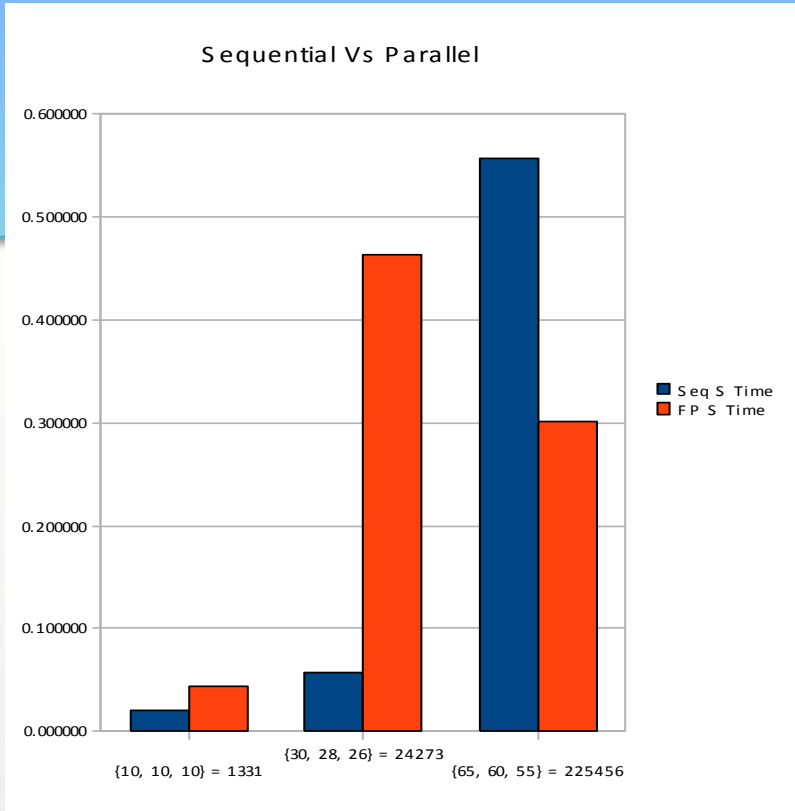
6D Waves & Partitions

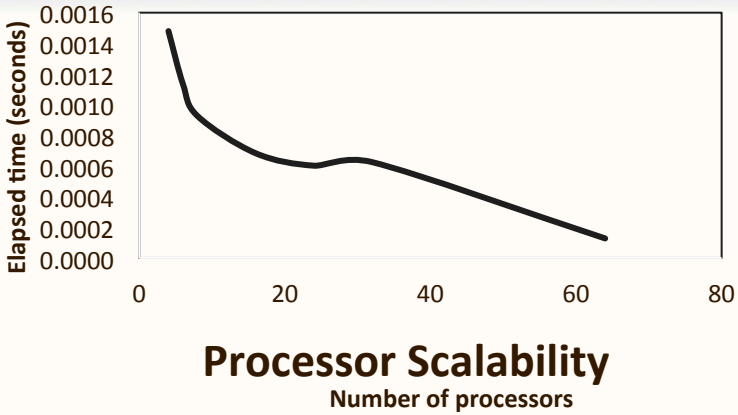


7D Waves & Partitions

Parallelization Results







Optimization

- Using Matlab® optimization tool box to produce the optimal S, V, e values that minimize the total distributed time equation:

$$r * \max(p_m) + (c/2) * (P^2 \Sigma p^2_m)$$

k = dimension; S = Partition size; I_i = shape at dimension i; V = Cluster Size with processors 0 <= m < V

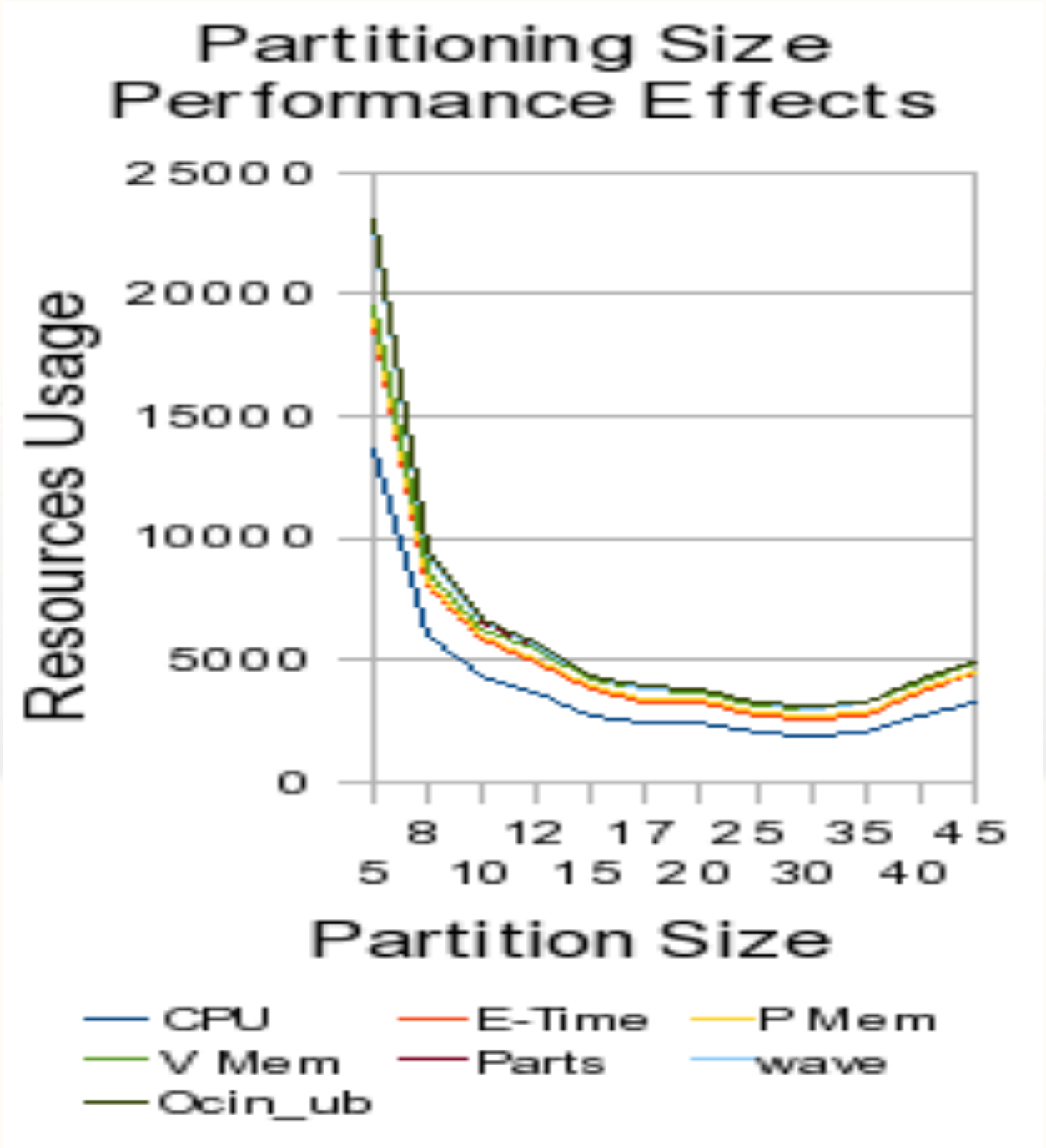
r = $(2^k-1) * s^k$ = a partition's scoring cost

c = $s^k-(s-1)^k$ = a partition's communication cost

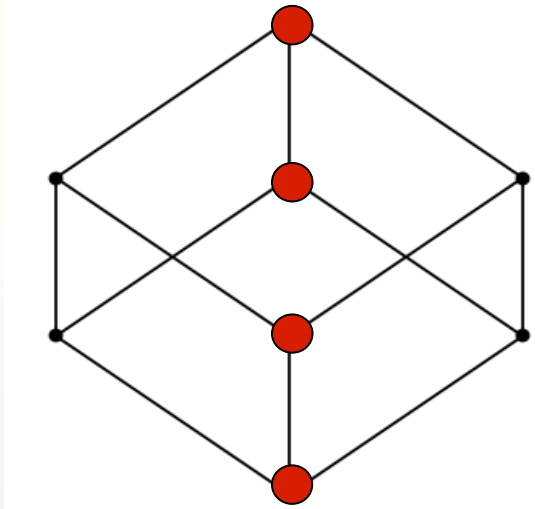
P = $\pi I_i-1 / s-1$ = Total partitions in all waves, all processors

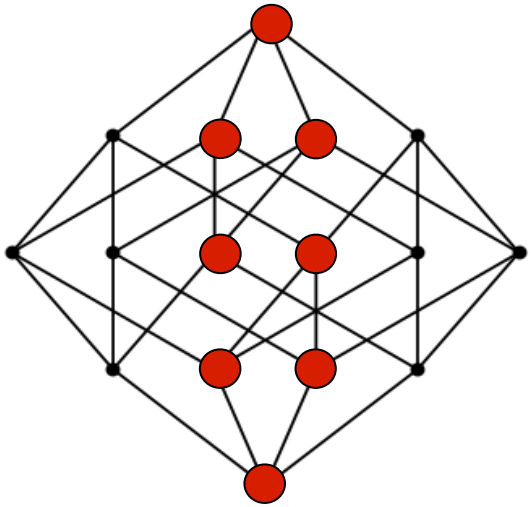
Pe = (e * (t-2)) + 2 for e > k otherwise, (e * (t-4))+k+2

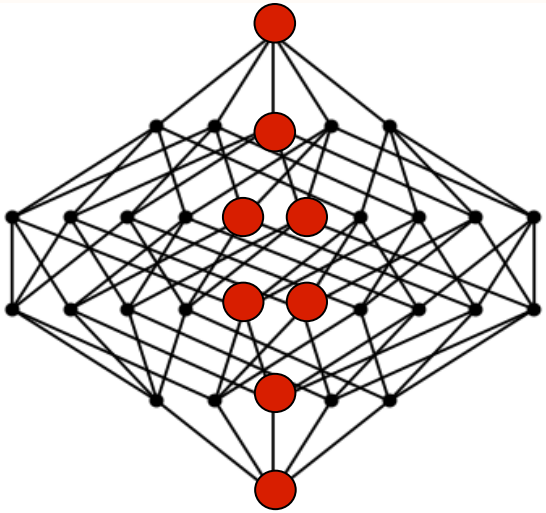
max (p_m) = $\lceil P/V \rceil + 2$

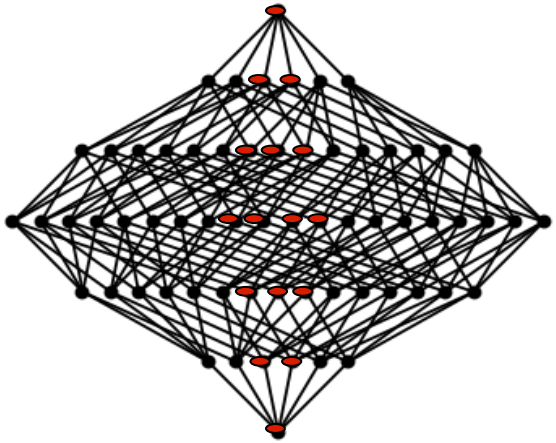

Partitioning Size
Performance Effects
Resources Usage
25000
20000
15000
10000
5000
0
5
8
10
12
15
17
20
25
30
35
40
45
Partition Size
CPU
E-Time
P Mem
V Mem
Parts
wave
Ocin_ub

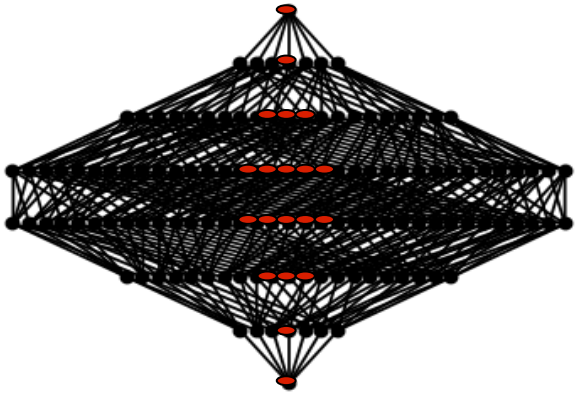
Search Space Reduction



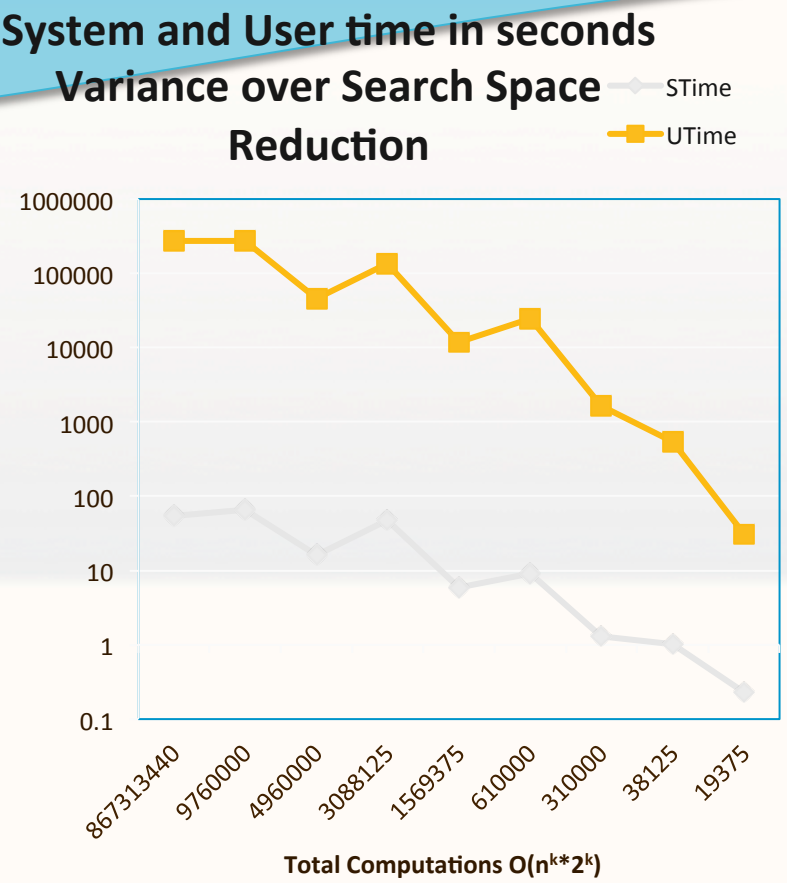


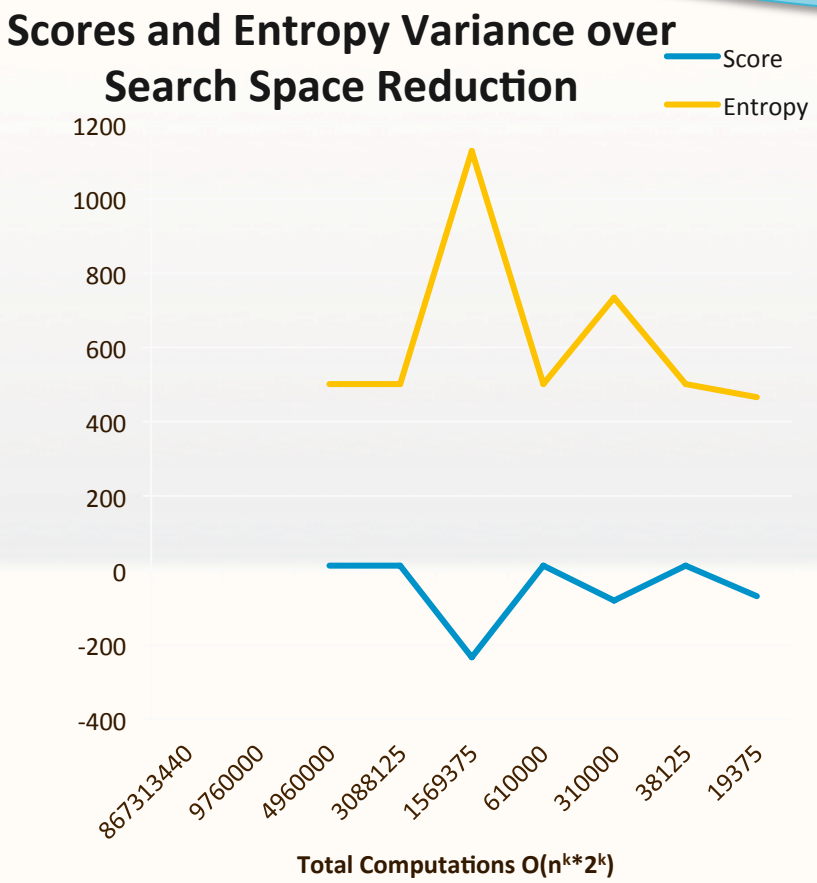






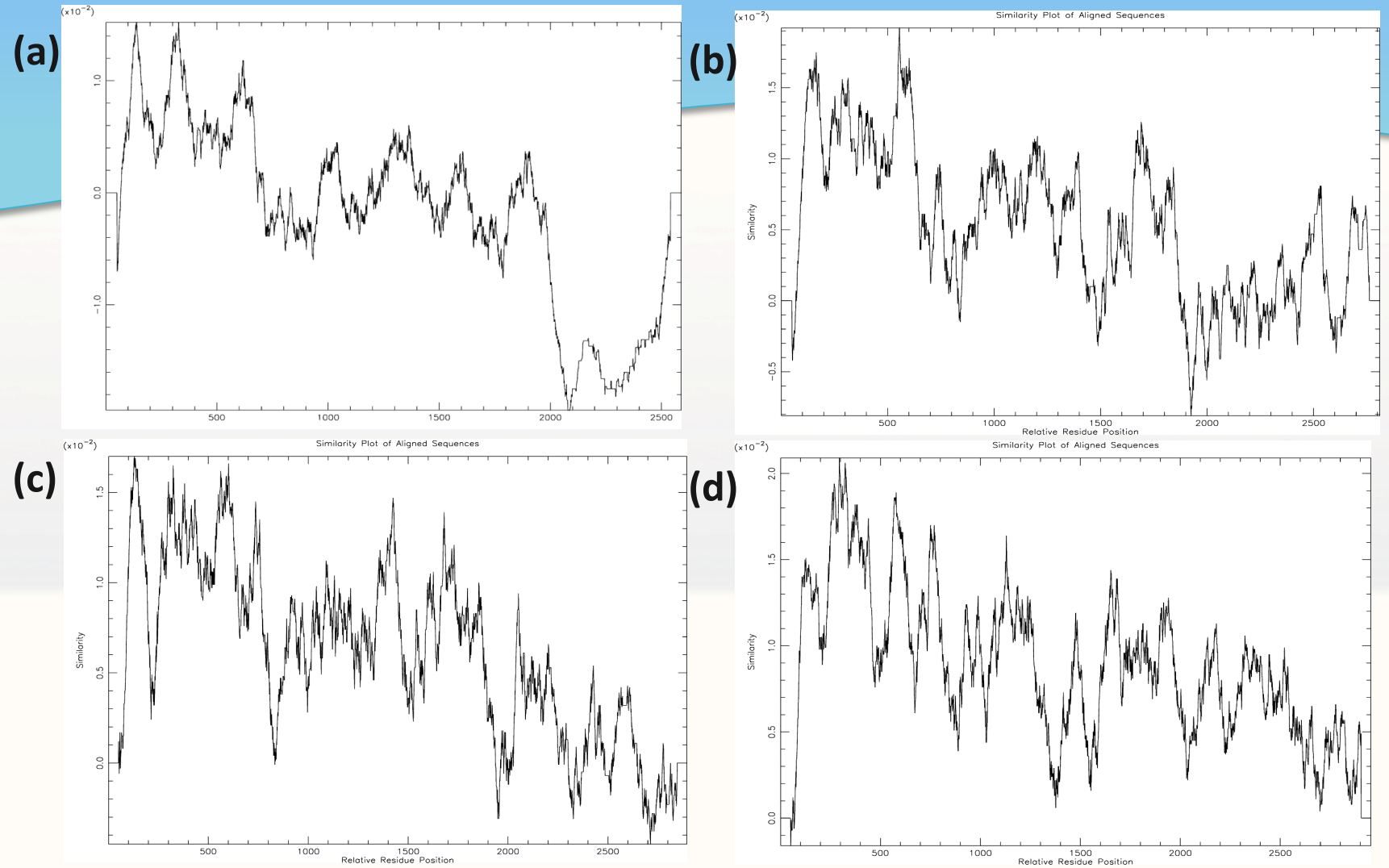
Reduced Search Space Performance





HELAL, M, Mullin, L, Potter, J, Sintchenko, V. 2009. Search Space Reduction Technique for Distributed Multiple Sequence Alignment. *In*: *Sixth IFIP International Conference on Network and Parallel Computing (NPC 2009)*. Gold Coast, Queensland, Australia.

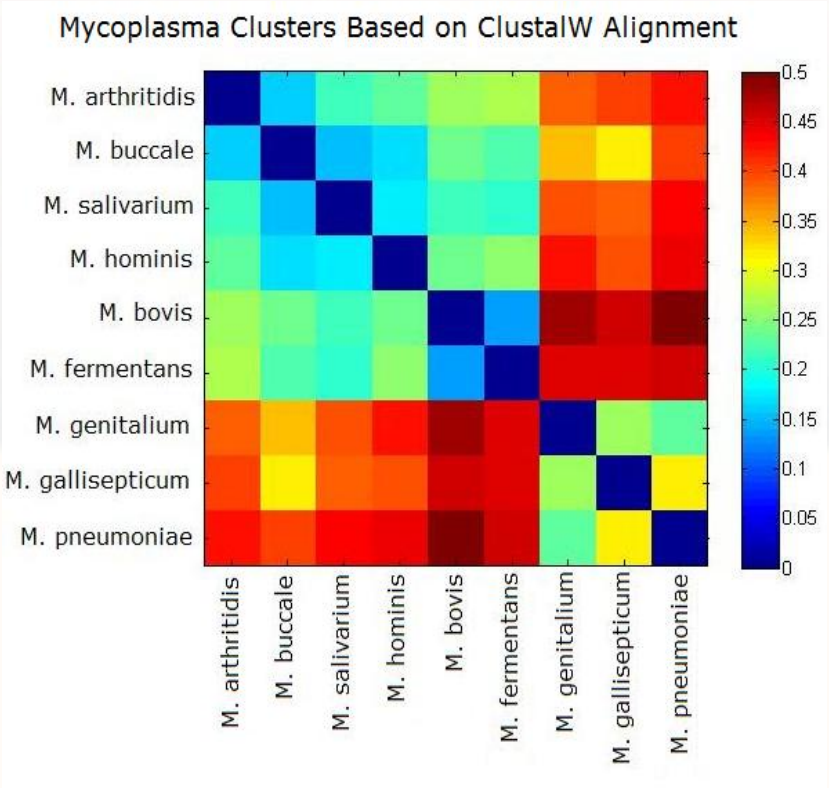
Quinolone-Resistance Determining Regions (QRDRs) Experiment

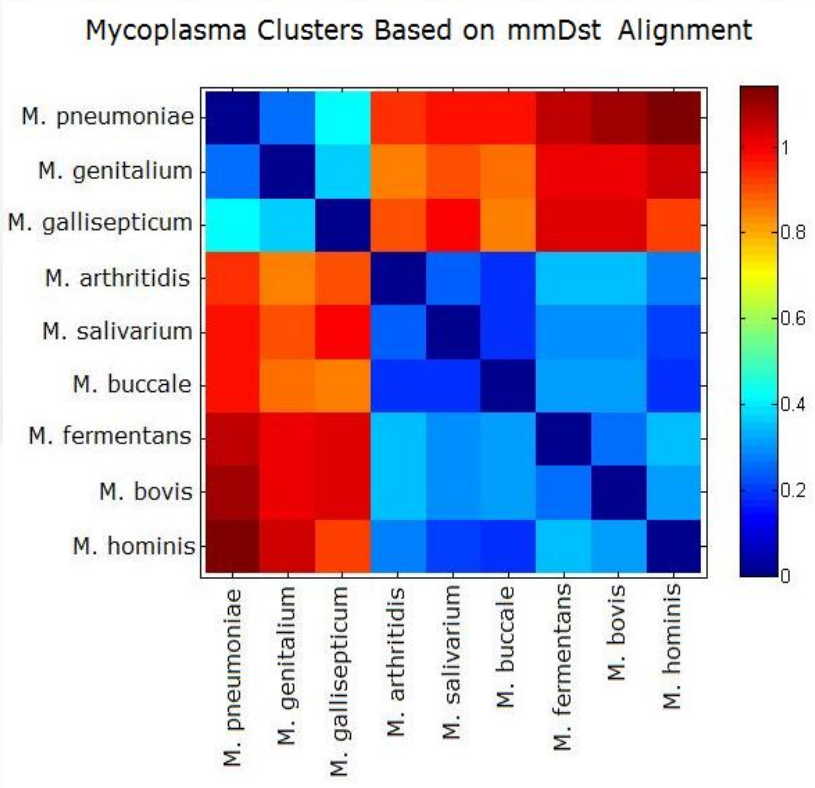


Similarity regions' plots: mmDst (a), Muscle (b), Tcoffee (c), clustalW(d).

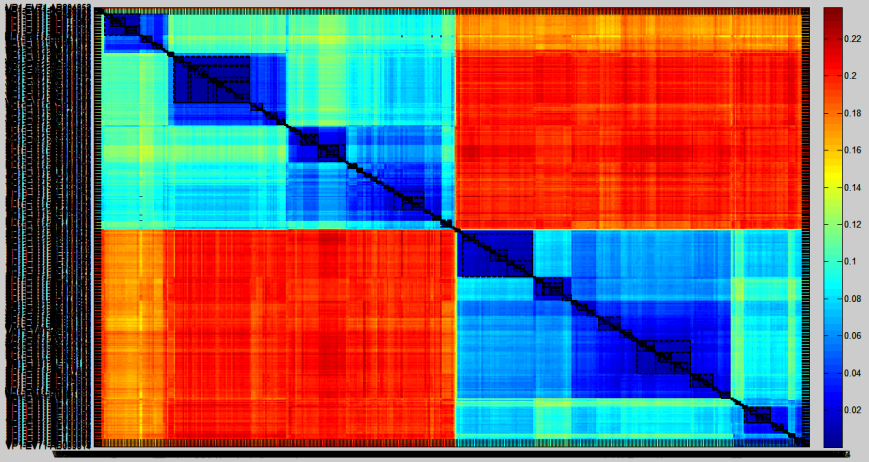
HELAL, M, Sintchenko, V. 2009. Dynamic Programming Algorithms for Discovery of Antibiotic Resistance in Microbial Genomes. *In*: *Health Informatics Conference (HIC-09)*. Canberra, Australia.

Mycoplasma Clusters Visualized

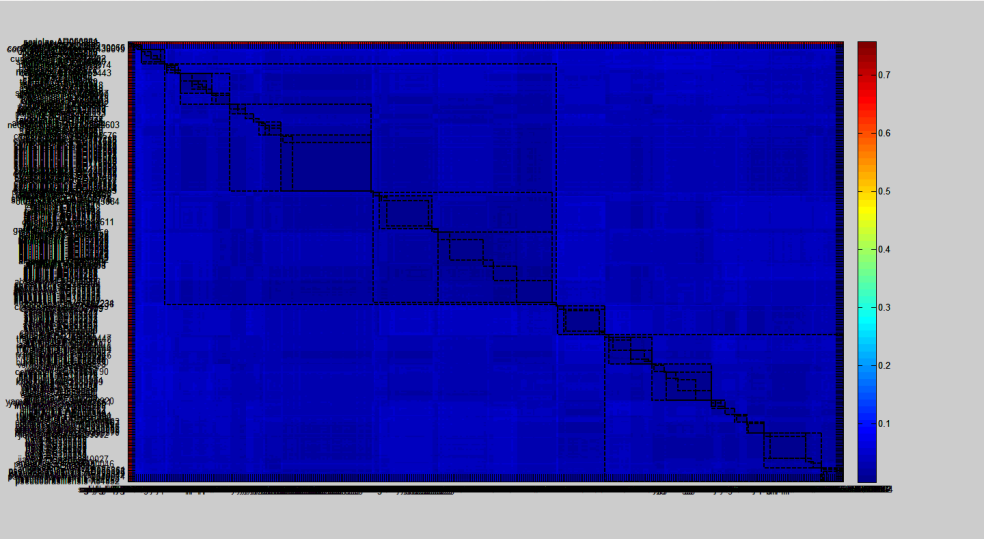




Clustering & Classification

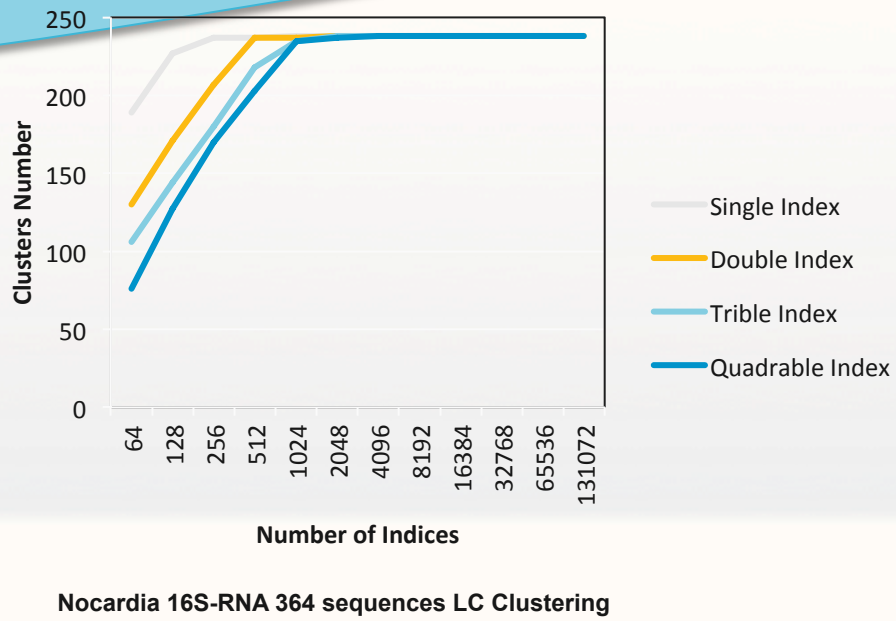


VP1-EV71 500 sequences Heatmap

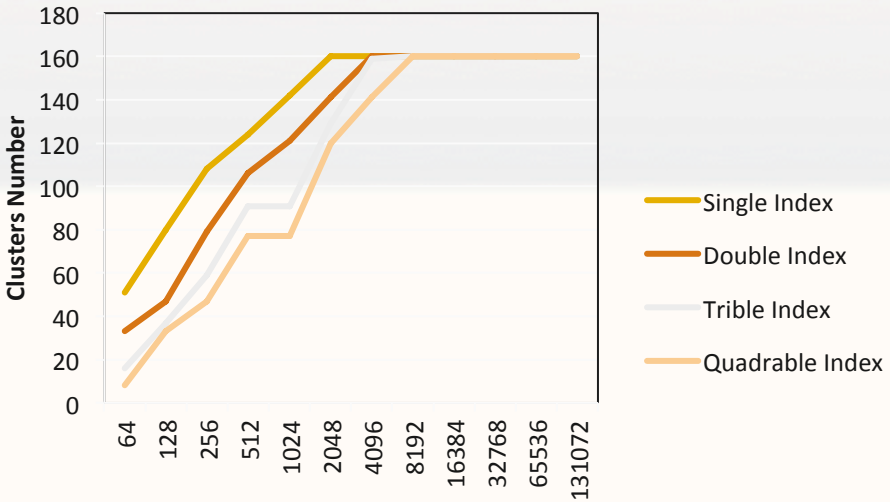


Nocardia 16S-RNA 364 sequences Heatmap

Distance Matrix Linear Mapping Clustering

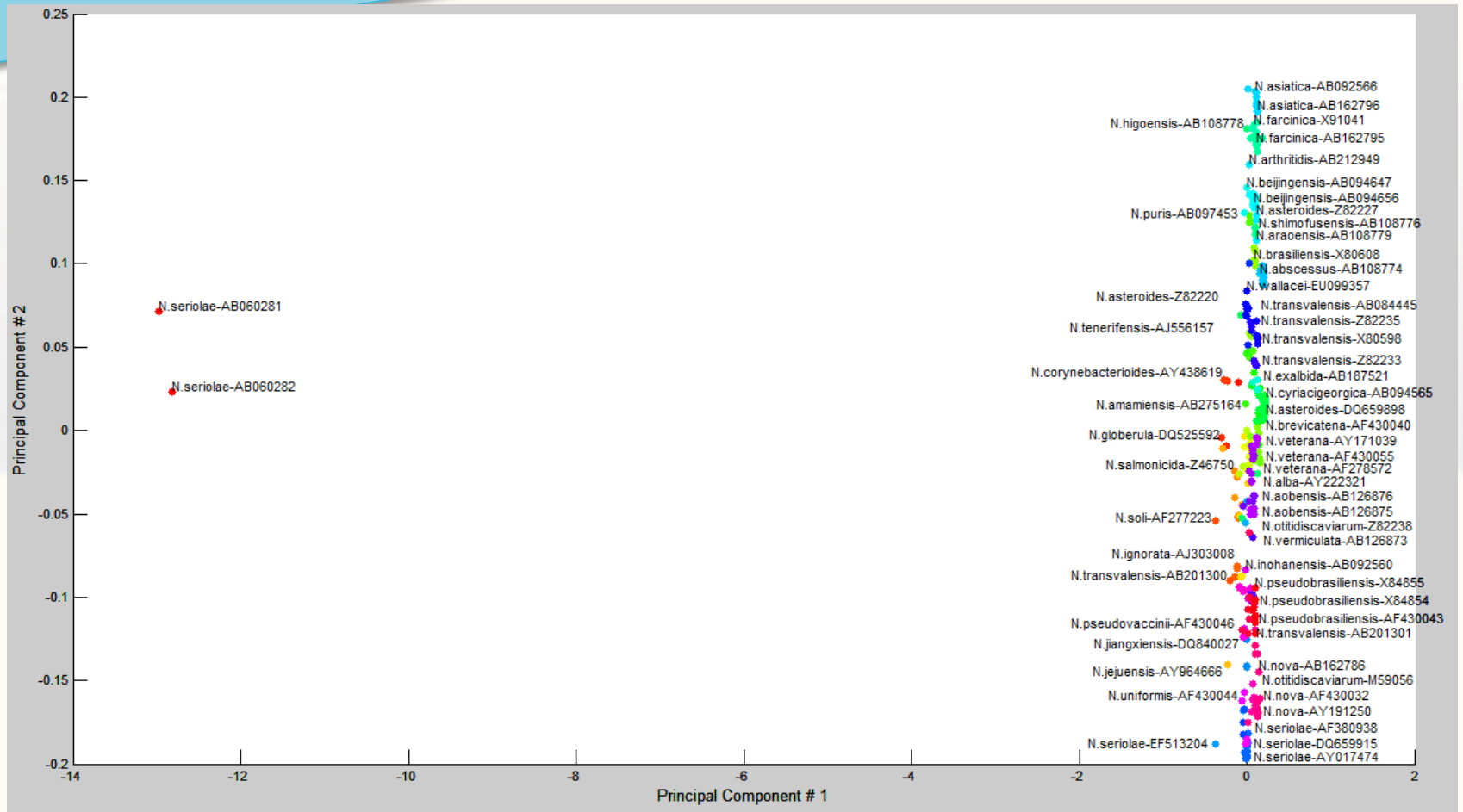


Nocardia 16S-RNA 364 sequences LC Clustering



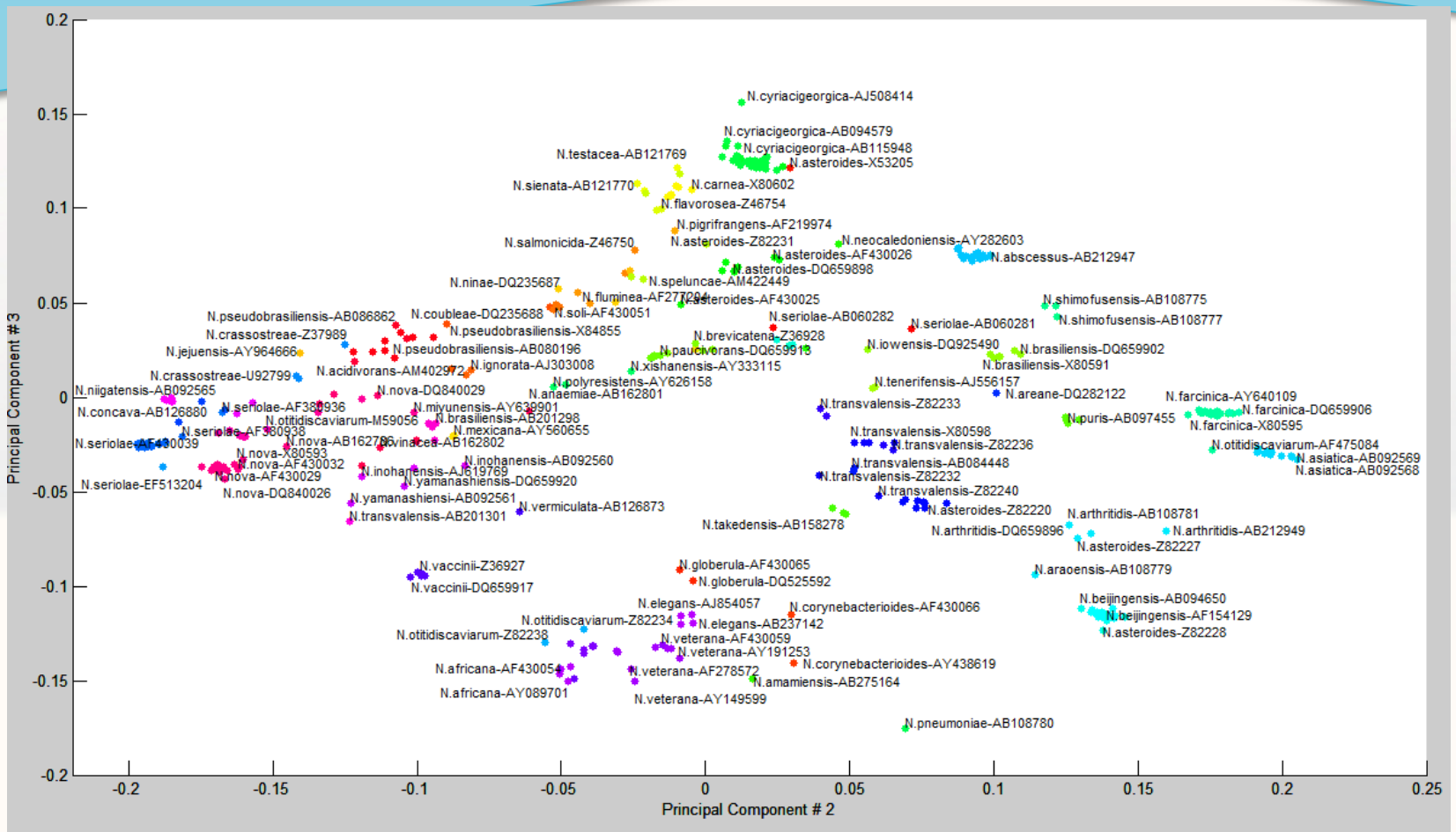
VP1-EV71 500 sequences LC Clustering

PCA Clustering



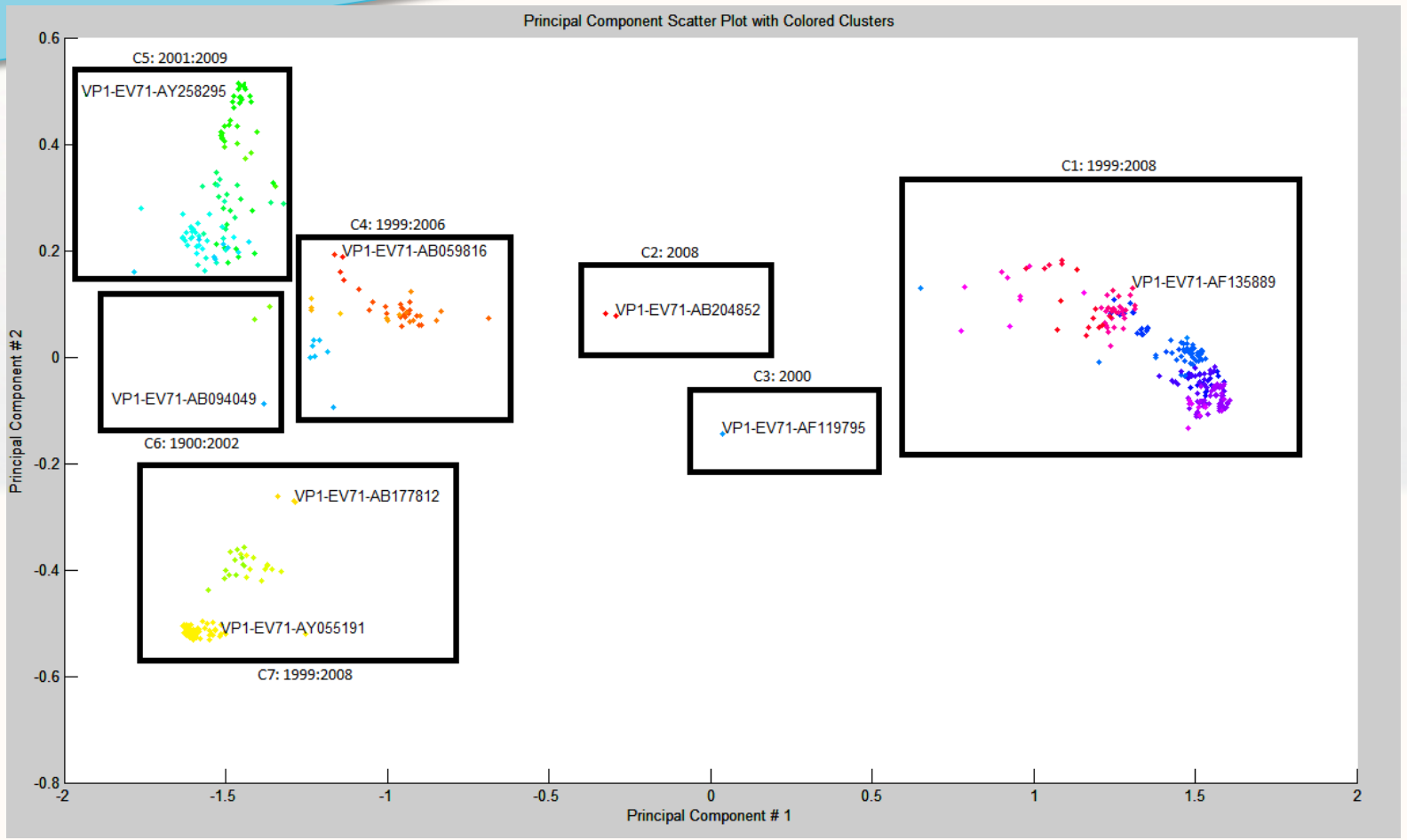
Nocardia 16S-RNA 364 sequences PCA (1 and 2) Clustering

PCA Clustering



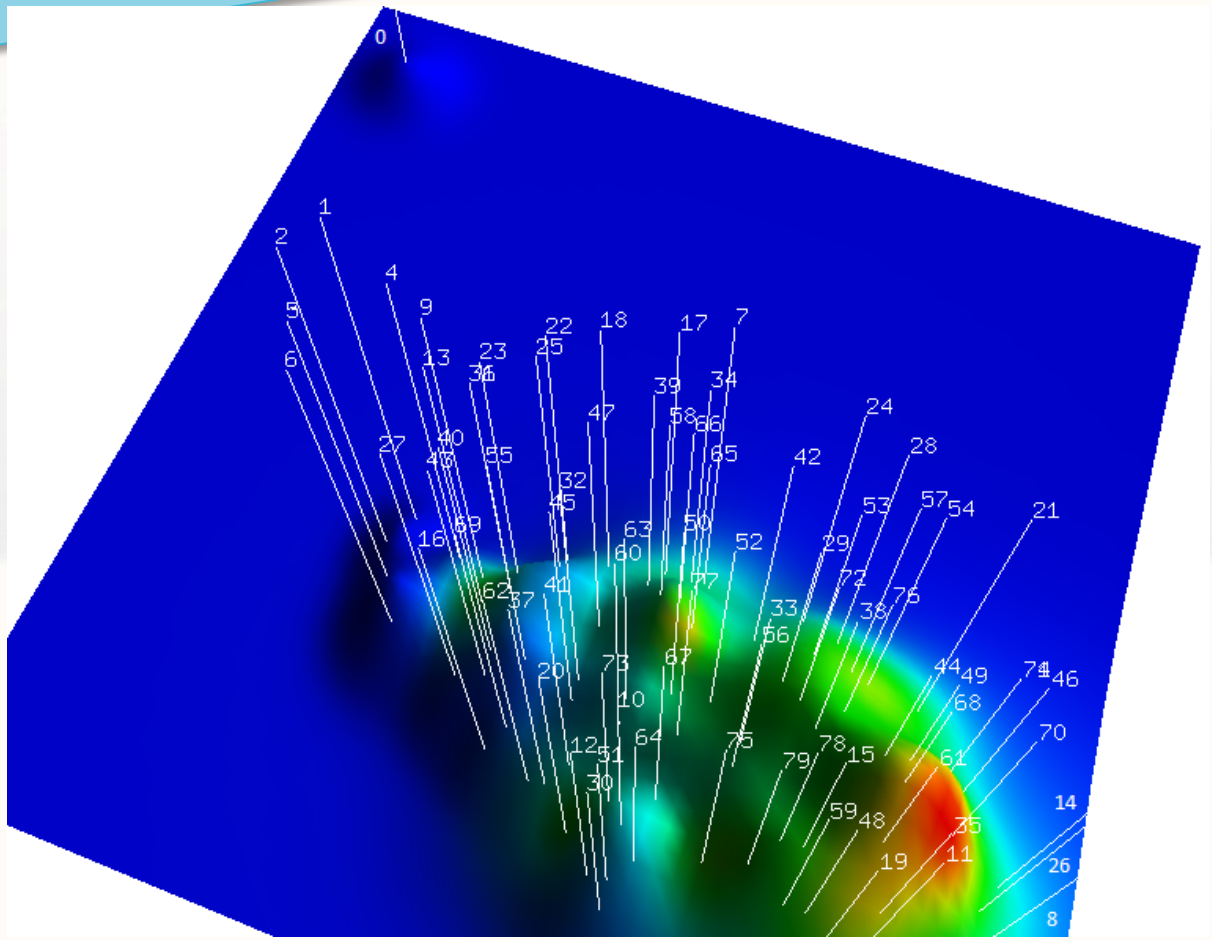
Nocardia 16S-RNA 364 sequences PCA (2 and 3) Clustering

PCA Clustering



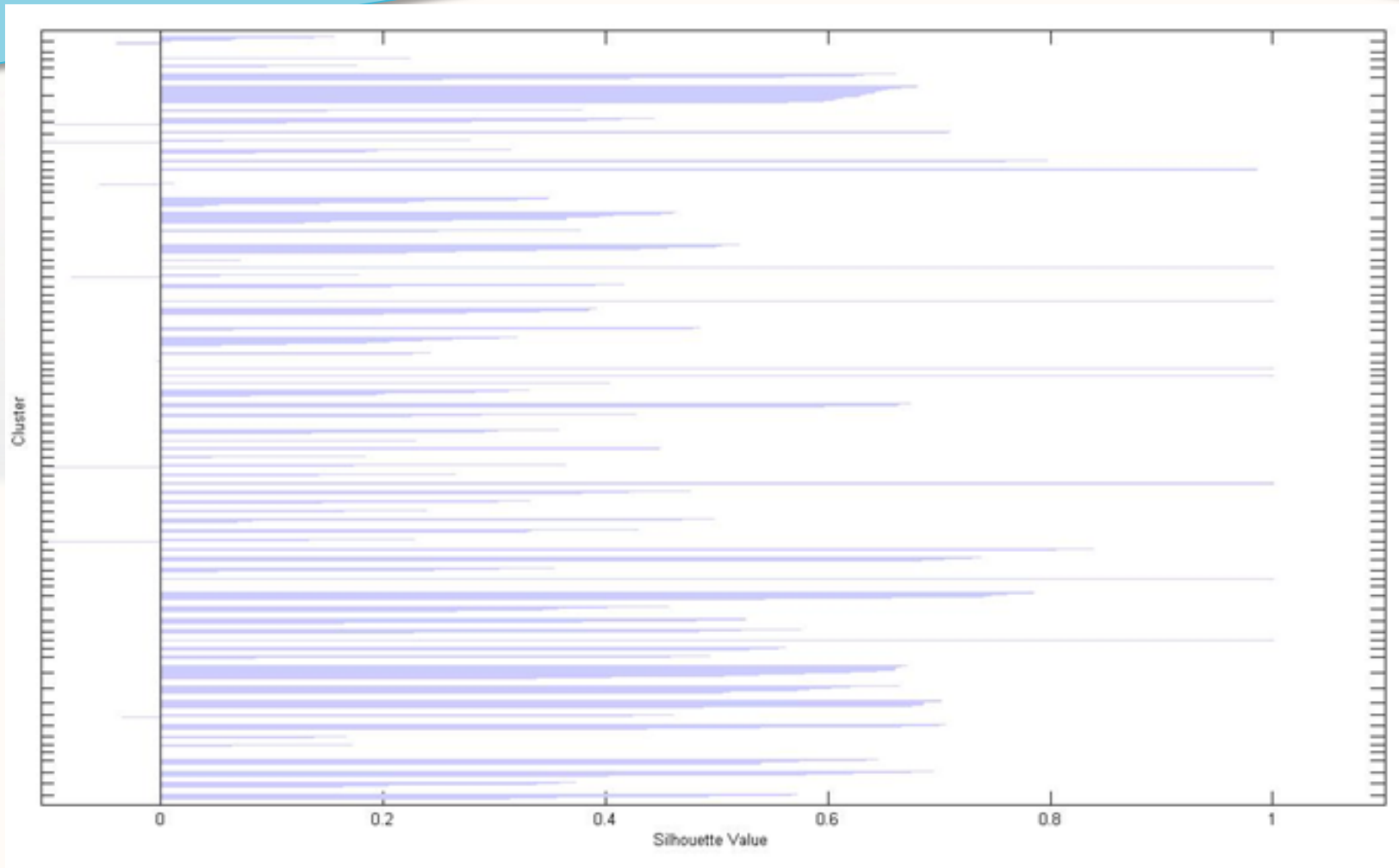
VP1-EV71 500 sequences PCA (1 and 2) Clustering

Cluto Optimization Clustering



Nocardia 16S-RNA 364 Cluto Clustering

K-Means



Clustering Comparison

Algorithm	Exact	%	Partial	%
LM –m=128, c=1	304	83.52	339	93.13
Cluto	304	83.52	332	91.21
HC - 77	294	80.77	320	87.91
Manual PCA	291	79.95	326	89.56
LM Over PCA	277	76.10	305	83.79
Kmeans - 77	258	70.88	309	84.89

Future Work

- Developing the Parallel Optimal MSA Tool for public access, and different datasets and functions.
- Further investigating the clustering and classification techniques for different data sets.
- Applying the design on GPU and Clouds architectures.
- Optimise the scoring function.
- Design high dimensional generic database schema to enable further data mining techniques.

References

1) Richard A. Tapia, Cynthia Lanius, “Computational Science: Tools for a Changing World - A High School Curriculum”, Rice university publications, 2003.
2) Ronald Veldema, “Parallel Algorithms Lecture Slides”, Department Informatik, Friedrich-Alexander-Universität Erlangen-Nürnberg

THANK YOU!