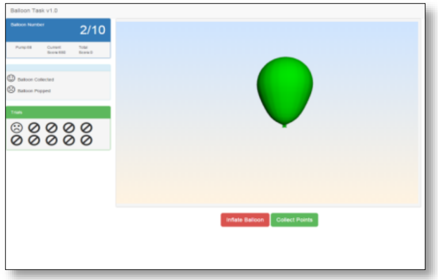
INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING
ICPP/2021/CHICAGO/USA
acm In-Cooperation
sighpc
AUGUST 9-12, 2021

Towards Faster Execution of Ensemble ML Bootstrap Based Techniques

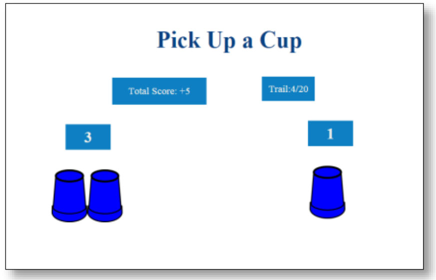
Vinay Gavirangaswamy, Ajay Gupta, Hisham Saleh, Dept. Computer Science, Western Michigan University; Vasilije Perovic, Dept. of Mathematics, University of Rhode Island

Decision Making, A Fundamental Aspect of life…

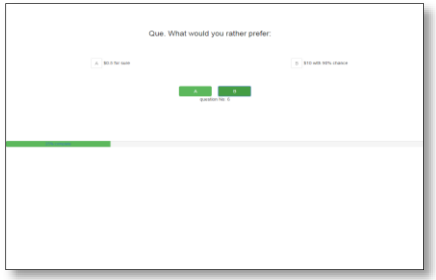
- Studied using tools for a specific competency e.g., surveys, computer tasks / games, etc.
- Finds application in life science (Stanford University HAI COVID19 and AI), Marketing, and Finance
- Analysis broadly classified for group / individual, aggregate / differential behaviors
- We applied Ensemble Clustering to tasks data from BART, CUPS, and IGT



BART



CUPS



Delayed Discount

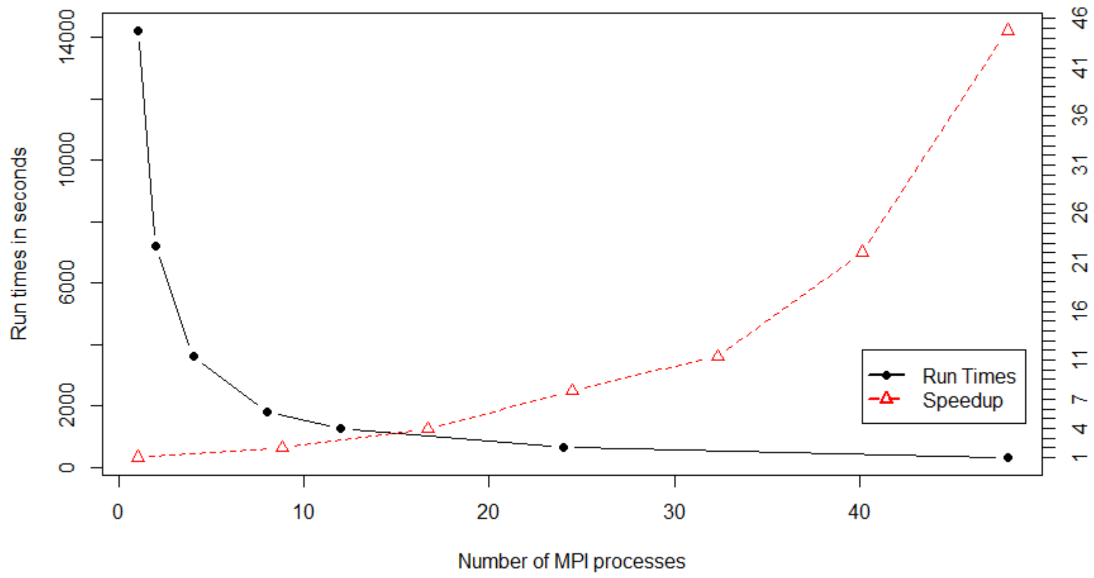
IGT

Parallel Implementation on Distributed Memory System

Run Times, Speedup and Efficiency of Ensemble Clustering after Parallelization 1000 Participants' Data

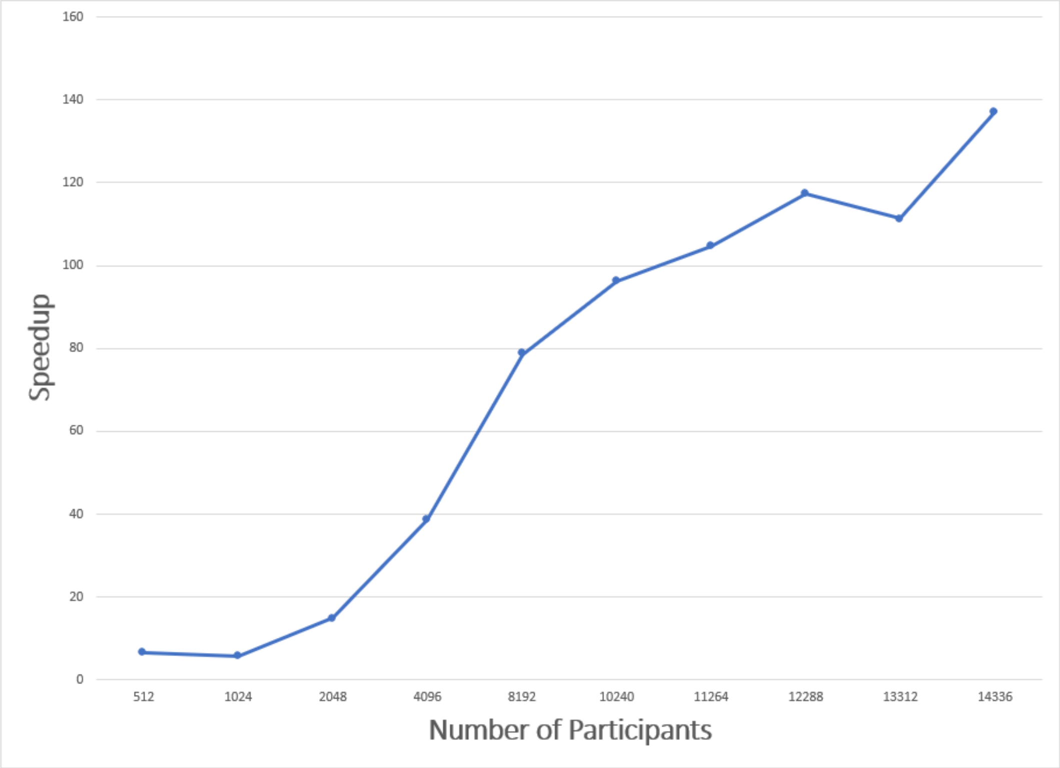
#MPI threads	1	2	4	8	12	24	48
Exec Time (in sec)	14237	7198	3603	1808	1251	645	317
Speedup	1	1.97	3.95	7.87	11.37	22.04	44.79
Efficiency	1	0.98	0.98	0.98	0.94	0.91	0.93

Execution Times And Speedup for Parallelized RDM Reinforcement Learning Algorithm

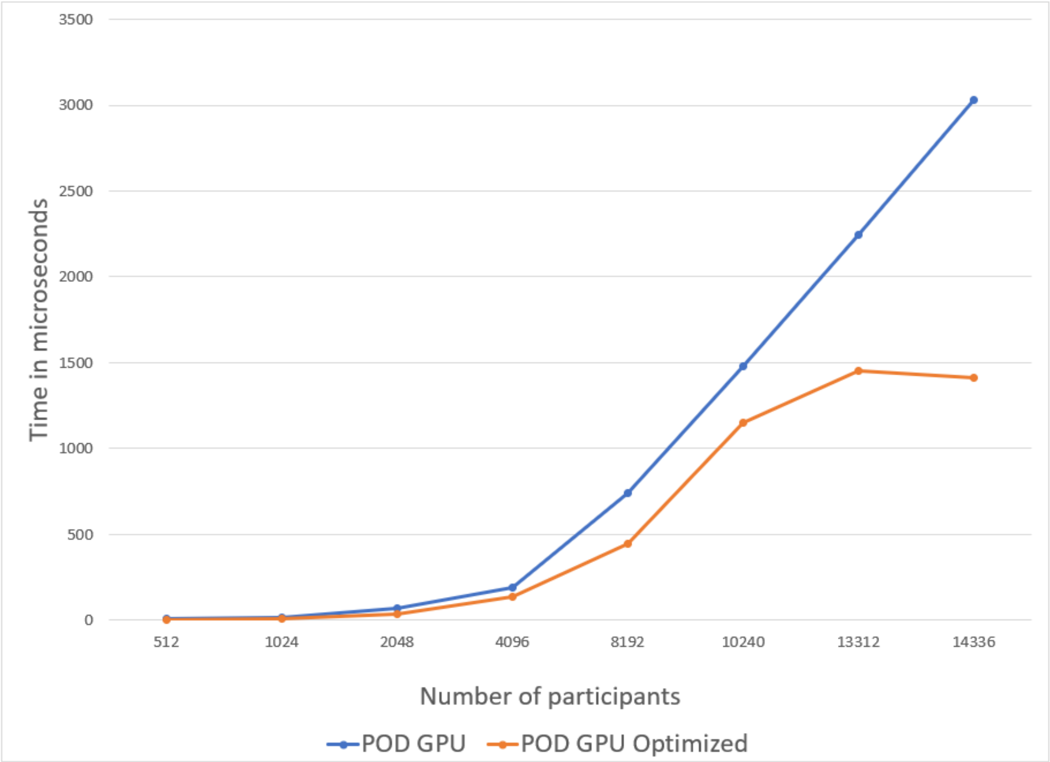


Parallel Implementation on Shared Memory System

Speedup achieved for running Ensemble Clustering on CPU vs. CPU-GPU



Run times for unordered vs. batch reconfigured model sequence execution



Parallel Implementation on Shared Memory System (Contd.)

Speedup achieved for running Ensemble Clustering on **CUBLAS-ARPACK**

	Parallel CUBLAS-ARPACK			
#Participants	Preprocessing	Ensemble Creation	Ensemble Extraction	Total
512	0.000002	0.027031	0.200994	0.228027
1024	0.000004	0.065028	0.829694	0.894726
2048	0.000008	0.179218	1.74629185	1.9255178
4096	0.000025	0.353409	4.55996992	4.9134039
8192	0.000097	0.918421	16.4327205	17.351238
10240	0.000197	1.4468	25.6589273	27.105924
11264	0.000228	1.762412	31.1415162	32.904156
12288	0.000216	1.773678	36.1456167	37.919510
13312	0.000254	8.05293	42.3747132	50.427897
14336	0.000309	2.353111	48.4853123	50.838732

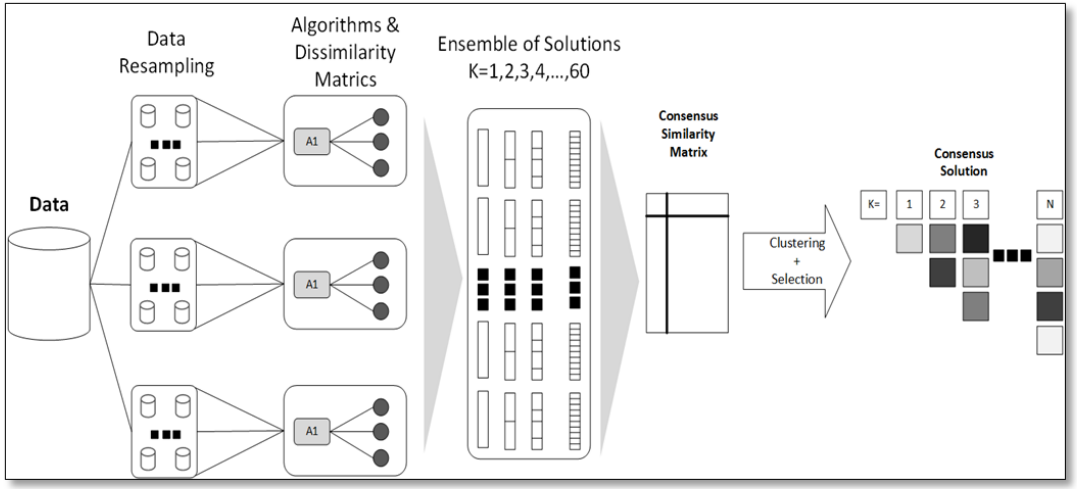
Speedup achieved for running Ensemble Clustering on **BLAS-ARPACK**

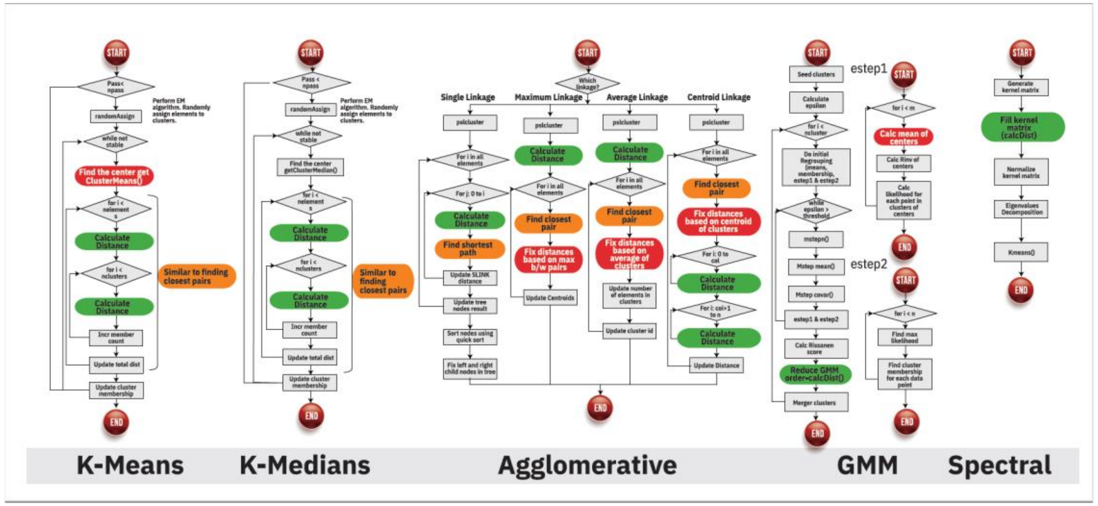
	Sequential BLAS-ARPACK			
#Participants	Preprocessing	Ensemble Creation	Ensemble Extraction	Total
512	0.008813	0.920119	0.596379	1.52531
1024	0.034078	3.890402	1.2575	5.18198
2048	0.133376	24.678988	3.91411	28.7264
4096	0.5259915	178.714399	10.220642	189.460
8192	2.0748	1325.326938	36.832031	1364.23
10240	3.240139	2550.207621	57.5115	2610.95
11264	3.89515	3374.181781	69.8000851	3447.87
12288	4.637481	4361.379918	81.016194	4447.03
13312	5.440463	5515.57373	94.977989	5615.99
14336	6.315224	6858.803793	108.674186	6973.79

Ensemble Clustering, optimizing for redundant computations (RC)

Lee I Newman. Decision Making under Uncertainty: Revealing, Characterizing and Modeling Individual Differences in the Iowa Gambling Task. PhD thesis, The University of Michigan, 2009

Highlighted boxes denote an example of steps in ensemble clustering that lead to redundant computations





Example: Dist-Program

1: **Input:** $a, b, c \in \mathbb{R}^2$.

2: **Output:** dac - squared Euclidean dist. b/w a and c,
$davgc$ - squared Euclidean dist. b/w $(a+b)$ and c.

3: $dac = (a_1 - c_1) \cdot (a_1 - c_1) + (a_2 - c_2) \cdot (a_2 - c_2)$

4: $davgc = (a_1 - c_1 + b_1) \cdot (a_1 - c_1 + b_1) + (a_2 - c_2 + b_2) \cdot (a_2 - c_2 + b_2)$

Redundancies in code due to numerical operations applied over same data

Original Machine Assembly Code with Color Coding for Redundancies

```
. . .
Omitted code
. . .

mov     DWORD PTR [rbp-52], 4
mov     DWORD PTR [rbp-56], 0
mov     DWORD PTR [rbp-60], 0
mov     edx, DWORD PTR [rbp-52]
mov     eax, DWORD PTR [rbp-24]
mov     esi, edx
mov     edi, eax
call    substract(int, int)
mov     ebx, eax
mov     edx, DWORD PTR [rbp-52]
mov     eax, DWORD PTR [rbp-24]
mov     esi, edx
mov     edi, eax
call    substract(int, int)
mov     esi, ebx
mov     edi, eax
call    multiply(int, int)
mov     ebx, eax
mov     edx, DWORD PTR [rbp-48]
mov     eax, DWORD PTR [rbp-20]
mov     esi, edx
mov     edi, eax
call    substract(int, int)
mov     r12d, eax
mov     edx, DWORD PTR [rbp-48]
mov     eax, DWORD PTR [rbp-20]
mov     esi, edx
mov     edi, eax
call    substract(int, int)
mov     esi, r12d
mov     edi, eax
call    multiply(int, int)
mov     esi, ebx
mov     edi, eax
call    add(int, int)
mov     DWORD PTR [rbp-56], eax
mov     edx, DWORD PTR [rbp-52]
mov     eax, DWORD PTR [rbp-24]
mov     esi, edx
mov     edi, eax
call    substract(int, int)
mov     edx, eax
mov     eax, DWORD PTR [rbp-32]
mov     esi, eax
mov     edi, edx
call    add(int, int)
mov     ebx, eax
```

```
. . .
Continues from Column 1
. . .

mov     ebx, eax
mov     edx, DWORD PTR [rbp-52]
mov     eax, DWORD PTR [rbp-24]
mov     esi, edx
mov     edi, eax
call    substract(int, int)
mov     edx, eax
mov     eax, DWORD PTR [rbp-32]
mov     esi, eax
mov     edi, edx
call    add(int, int)
mov     esi, ebx
mov     edi, eax
call    multiply(int, int)
mov     ebx, eax
mov     edx, DWORD PTR [rbp-48]
mov     eax, DWORD PTR [rbp-20]
mov     esi, edx
mov     edi, eax
call    substract(int, int)
mov     edx, eax
mov     eax, DWORD PTR [rbp-28]
mov     esi, eax
mov     edi, edx
call    add(int, int)
mov     r12d, eax
mov     edx, DWORD PTR [rbp-48]
mov     eax, DWORD PTR [rbp-20]
mov     esi, edx
mov     edi, eax
call    substract(int, int)
mov     edx, eax
mov     eax, DWORD PTR [rbp-28]
mov     esi, eax
mov     edi, edx
call    add(int, int)
mov     esi, r12d
mov     edi, eax
call    multiply(int, int)
mov     esi, ebx
mov     edi, eax
call    add(int, int)
mov     DWORD PTR [rbp-60], eax
mov     eax, 0
add     rsp, 48
pop     rbx
pop     r12
pop     rbp
ret
```

An Example Optimized Assembly Code

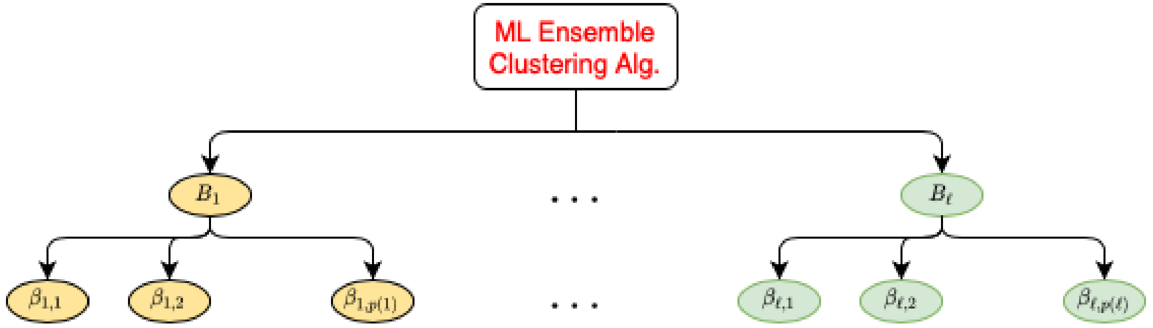
```
. . .
Omitted code
. . .

mov     DWORD PTR [rbp-52], 4
mov     DWORD PTR [rbp-56], 0
mov     DWORD PTR [rbp-60], 0
<stored result D, Computational Unit D>
mov     ebx, eax
<stored result D, Computational Unit C>
<stored result F, Computational Unit F>
mov     ebx, eax
<stored result B, Computational Unit B>
mov     r12d, eax
<stored result B, Computational Unit A>
<stored result F, Computational Unit E>
<stored result G, Computational Unit G>
mov     DWORD PTR [rbp-56], eax
<stored result D, Computational Unit K>
mov     edx, eax
<stored result O, Computational Unit O>
mov     ebx, eax
<stored result D, Computational Unit K>
```

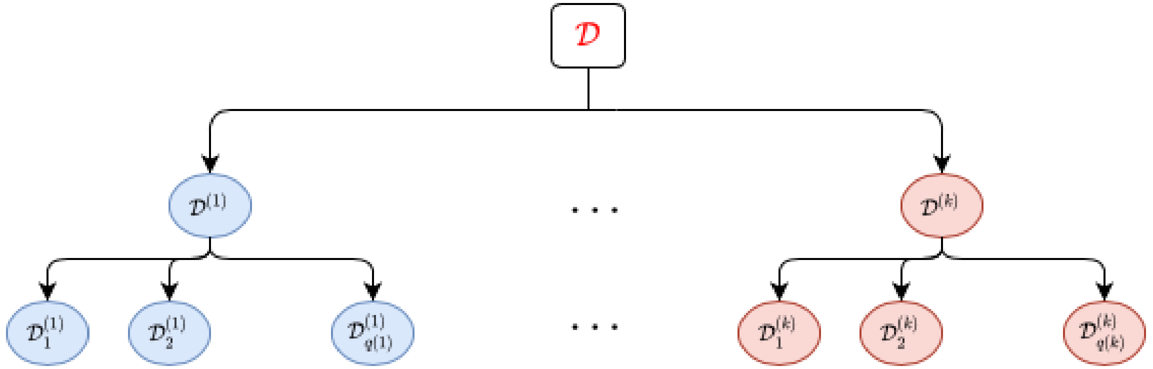
```
. . .
Continues from Column 3
. . .

mov     ebx, eax
<stored result D, Computational Unit K>
mov     edx, eax
<stored result O, Computational Unit N>
<stored result Q, Computational Unit Q>
mov     ebx, eax
<stored result B, Computational Unit I>
mov     edx, eax
<stored result M, Computational Unit M>
mov     r12d, eax
<stored result B, Computational Unit H>
mov     edx, eax
<stored result M, Computational Unit L>
<stored result P, Computational Unit P>
<stored result R, Computational Unit R>
mov     DWORD PTR [rbp-60], eax
mov     eax, 0
add     rsp, 48
pop     rbx
pop     r12
pop     rbp
ret
```

Problem Formulation



Distinct clustering algorithms



Bootstrapped data sets from the main data set *D* *(t=1,2,…,k)*

Computational unit (cu) – an atomic unit consisting of both data *and* mathematical computations operating on the data set.

Example: Dist-Program

Input: $a, b, c \in \mathbb{R}^2$.

Output: dac - squared Euclidean dist. b/w a and c,
$davgc$ - squared Euclidean dist. b/w $(a+b)$ and c.

$dac = (a_1 - c_1)\cdot(a_1 - c_1) + (a_2 - c_2)\cdot(a_2 - c_2)$

$davgc = (a_1 - c_1 + b_1)\cdot(a_1 - c_1 + b_1) + (a_2 - c_2 + b_2)\cdot(a_2 - c_2 + b_2)$

# elt's.	subsets of $\mathcal{D}^{(t)}$
0	$\mathcal{D}_1^{(t)} = \varnothing$
1	$\mathcal{D}_2^{(t)} = \{a\}$, $\mathcal{D}_3^{(t)} = \{b\}$, $\mathcal{D}_4^{(t)} = \{c\}$,
2	$\mathcal{D}_5^{(t)} = \{a,b\}$, $\mathcal{D}_6^{(t)} = \{a,c\}$, $\mathcal{D}_7^{(t)} = \{b,c\}$,
3	$\mathcal{D}_8^{(t)} = \{a,b,c\}$

Nontrivial cu's with repetitions

$$\left.\begin{array}{l}\left.\begin{array}{l} m_1^{6,t} = a_1 - c_1 \\ m_2^{6,t} = a_1 - c_1 \end{array}\right\} \longrightarrow m_3^{6,t} = m_1^{6,t} \cdot m_2^{6,t} \\ \left.\begin{array}{l} m_4^{6,t} = a_2 - c_2 \\ m_5^{6,t} = a_2 - c_2 \end{array}\right\} \longrightarrow m_6^{6,t} = m_4^{6,t} \cdot m_5^{6,t} \end{array}\right\} \longrightarrow m_7^{6,t} = m_3^{6,t} + m_6^{6,t}$$

$$\left.\begin{array}{l} m_7^{6,t} = a_1 - c_1 \} \longrightarrow m_1^{8,t} = m_7^{6,t} + b_1 \\ m_8^{6,t} = a_1 - c_1 \} \longrightarrow m_2^{8,t} = m_8^{6,t} + b_1 \end{array}\right\} \longrightarrow m_5^{8,t} = m_1^{8,t} \cdot m_2^{8,t}$$

$$\left.\begin{array}{l} m_9^{6,t} = a_2 - c_2 \} \longrightarrow m_3^{8,t} = m_9^{6,t} + b_2 \\ m_{10}^{6,t} = a_2 - c_2 \} \longrightarrow m_4^{8,t} = m_{10}^{6,t} + b_2 \end{array}\right\} \longrightarrow m_6^{8,t} = m_3^{8,t} \cdot m_4^{8,t}$$

$$\longrightarrow m_7^{8,t} = m_5^{8,t} + m_6^{8,t}$$

Note: $m_j^{i,t}$ = j^{th} computational unit acting on subset $D_i^{(t)}$

Example: Dist-Program

Nontrivial cu's with repetition

$$\left.\begin{array}{l} m_1^{6,t} = a_1 - c_1 \\ m_2^{6,t} = a_1 - c_1 \end{array}\right\} \longrightarrow m_3^{6,t} = m_1^{6,t} \cdot m_2^{6,t}$$

$$\left.\begin{array}{l} m_4^{6,t} = a_2 - c_2 \\ m_5^{6,t} = a_2 - c_2 \end{array}\right\} \longrightarrow m_6^{6,t} = m_4^{6,t} \cdot m_5^{6,t}$$

$$\left.\begin{array}{l} m_3^{6,t} = m_1^{6,t} \cdot m_2^{6,t} \\ m_6^{6,t} = m_4^{6,t} \cdot m_5^{6,t} \end{array}\right\} \longrightarrow m_7^{6,t} = m_3^{6,t} + m_6^{6,t}$$

$$\begin{array}{l} m_7^{6,t} = a_1 - c_1 \} \longrightarrow m_1^{8,t} = m_7^{6,t} + b_1 \\ m_8^{6,t} = a_1 - c_1 \} \longrightarrow m_2^{8,t} = m_8^{6,t} + b_1 \end{array} \Bigg\} \longrightarrow m_5^{8,t} = m_1^{8,t} \cdot m_2^{8,t}$$

$$\begin{array}{l} m_9^{6,t} = a_2 - c_2 \} \longrightarrow m_3^{8,t} = m_9^{6,t} + b_2 \\ m_{10}^{6,t} = a_2 - c_2 \} \longrightarrow m_4^{8,t} = m_{10}^{6,t} + b_2 \end{array} \Bigg\} \longrightarrow m_6^{8,t} = m_3^{8,t} \cdot m_4^{8,t}$$

$$\longrightarrow m_7^{8,t} = m_5^{8,t} + m_6^{8,t}$$

M = collection of all nontrivial computational units along with their dependencies

Nontrivial cu's without repetitions

		Original cu	Relabeled cu	Dependency
$m_1^{6,t}$	=	$a_1 - c_1$	$\widetilde{m}_1$	none
$m_3^{6,t}$	=	$m_1^{6,t} \cdot m_2^{6,t} = m_1^{6,t} \cdot m_1^{6,t}$	$\widetilde{m}_2$	$\widetilde{m}_1$
$m_4^{6,t}$	=	$a_2 - c_2$	$\widetilde{m}_3$	none
$m_6^{6,t}$	=	$m_4^{6,t} \cdot m_5^{6,t} = m_4^{6,t} \cdot m_4^{6,t}$	$\widetilde{m}_4$	$\widetilde{m}_3$
$m_7^{6,t}$	=	$m_3^{6,t} + m_6^{6,t}$	$\widetilde{m}_5$	$\widetilde{m}_2$, $\widetilde{m}_4$
$m_1^{8,t}$	=	$m_7^{6,t} + b_1 = m_1^{6,t} + b_1$	$\widetilde{m}_6$	$\widetilde{m}_1$
$m_5^{8,t}$	=	$m_1^{8,t} \cdot m_2^{8,t} = m_1^{8,t} \cdot m_1^{8,t}$	$\widetilde{m}_7$	$\widetilde{m}_6$
$m_3^{8,t}$	=	$m_9^{6,t} + b_2 = m_4^{6,t} + b_2$	$\widetilde{m}_8$	$\widetilde{m}_3$
$m_6^{8,t}$	=	$m_3^{8,t} \cdot m_4^{8,t} = m_3^{8,t} \cdot m_3^{8,t}$	$\widetilde{m}_9$	$\widetilde{m}_8$
$m_7^{8,t}$	=	$m_5^{8,t} + m_6^{8,t}$	$\widetilde{m}_{10}$	$\widetilde{m}_7$, $\widetilde{m}_9$

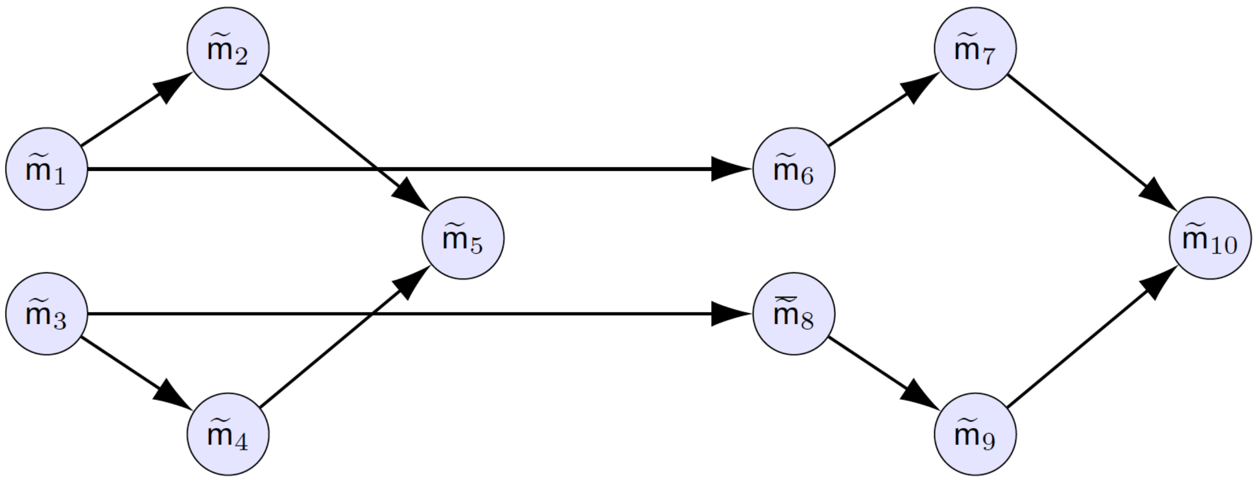
Nontrivial cu's with repetition (Dist– Program)

		Original cu	Relabeled cu	Dependency
$m_1^{6,t}$	$=$	$a_1 - c_1$	$\widetilde{m}_1$	none
$m_3^{6,t}$	$=$	$m_1^{6,t} \cdot m_2^{6,t} = m_1^{6,t} \cdot m_1^{6,t}$	$\widetilde{m}_2$	$\widetilde{m}_1$
$m_4^{6,t}$	$=$	$a_2 - c_2$	$\widetilde{m}_3$	none
$m_6^{6,t}$	$=$	$m_4^{6,t} \cdot m_5^{6,t} = m_4^{6,t} \cdot m_4^{6,t}$	$\widetilde{m}_4$	$\widetilde{m}_3$
$m_7^{6,t}$	$=$	$m_3^{6,t} + m_6^{6,t}$	$\widetilde{m}_5$	$\widetilde{m}_2$, $\widetilde{m}_4$
$m_1^{8,t}$	$=$	$m_7^{6,t} + b_1 = m_1^{6,t} + b_1$	$\widetilde{m}_6$	$\widetilde{m}_1$
$m_5^{8,t}$	$=$	$m_1^{8,t} \cdot m_2^{8,t} = m_1^{8,t} \cdot m_1^{8,t}$	$\widetilde{m}_7$	$\widetilde{m}_6$
$m_3^{8,t}$	$=$	$m_9^{6,t} + b_2 = m_4^{6,t} + b_2$	$\widetilde{m}_8$	$\widetilde{m}_3$
$m_6^{8,t}$	$=$	$m_3^{8,t} \cdot m_4^{8,t} = m_3^{8,t} \cdot m_3^{8,t}$	$\widetilde{m}_9$	$\widetilde{m}_8$
$m_7^{8,t}$	$=$	$m_5^{8,t} + m_6^{8,t}$	$\widetilde{m}_{10}$	$\widetilde{m}_7$, $\widetilde{m}_9$

Capturing Interdependencies between cu's

$$\widetilde{n}_4 = (0,0,1,0,0,0,0,0,0,0), \qquad \widetilde{n}_5 = (0,1,0,1,0,0,0,0,0,0),$$

$$\widetilde{n}_6 = (1,0,0,0,0,0,0,0,0,0), \qquad \widetilde{n}_{10} = (0,0,0,0,0,0,1,0,1,0).$$



Interaction graph capturing interdependencies between $\widetilde{m}$'s (directed acyclic graph – DAG)

Algorithm: Reduction in Redundant Computations (RRC)

Step 1: Decompose Ensemble Clustering ML algorithm into collection of computational units M.

Step 2: Identify and collect all distinct computational units from M into $\widetilde{M}$ along with interdependencies.

Step 3: Find an aggregation of cu's , $\mathfrak{C}$, with execution cost lower than that of M.

Step 4: FTiP, Floor Tile Planning - Find an efficient execution plan for $\mathfrak{C}$ on the compute resource.

Step 3 (RRC): Find an aggregation of cu's , $\mathfrak{C}$, with execution cost lower than that of M

The choice of aggregation of cu's $\mathfrak{C}$ is limited by computer resources, some of which are as follow:

- Machine with *restrictive compute* and *infinite memory*
- Machine with *infinite compute* and *restrictive memory*
- Machine with *restrictive compute* and *restrictive memory*

Definition: *r-s* Set Cover

Let $\widetilde{\mathcal{M}} = \{\widetilde{\mathrm{m}}_1, \widetilde{\mathrm{m}}_2, \ldots, \widetilde{\mathrm{m}}_{\widetilde{\omega}}\}$ be a set of distinct cu's and $\widetilde{\mathcal{G}}$ a directed acyclic (interaction) graph corresponding to $\widetilde{\mathcal{M}}$. An *ordered list*

$$\mathfrak{C} = (\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_\delta), \quad \mathcal{A}_i \subseteq \widetilde{\mathcal{M}}, \quad i = 1, 2, \ldots, \delta,$$

is said to be an *(r, s) set cover of $\widetilde{\mathcal{M}}$*, where r and s are positive integers, if for all $i = 1, 2, \ldots, \delta$ sets $\mathcal{A}_i$ are nonempty and the following conditions are satisfied:

(i) $\bigcup_{i=1}^{\delta} \mathcal{A}_i = \widetilde{\mathcal{M}}$,

(ii) cardinality of $\mathcal{A}_i$ is at most r, i.e., $|\mathcal{A}_i| \leq r$, and

(iii) an arbitrary $\mathrm{m}_\alpha \in \mathcal{A}_i$ can be executed given a subset, possibly empty, of tasks from $\mathcal{A}_i, \mathcal{A}_{i-1}, \ldots, \mathcal{A}_{i-s}$.

Connection with the Directed Bandwidth Problem

Question:

Given $r \geq 1$, what is the smallest *s* for which (r,s) set cover of $\widetilde{M}$ exists?

Def: Given a directed acyclic graph $\widetilde{\mathcal{G}}$ associated with a collection of cu's in $\widetilde{\mathcal{M}}$, the *directed bandwidth of* $\widetilde{\mathcal{G}}$, $\overrightarrow{Bw}(\widetilde{\mathcal{G}})$, is given by

$$\overrightarrow{Bw}(\widetilde{\mathcal{G}}) := \min \Big\{ Bw(\widetilde{\mathcal{G}}, \sigma) \;:\; \sigma \text{ is a layout of } \widetilde{\mathcal{G}} \text{ and } \sigma(\mathrm{m}_\alpha) < \sigma(\mathrm{m}_\beta) \text{ for all directed arcs from } \mathrm{m}_\alpha \text{ to } \mathrm{m}_\beta \Big\}.$$

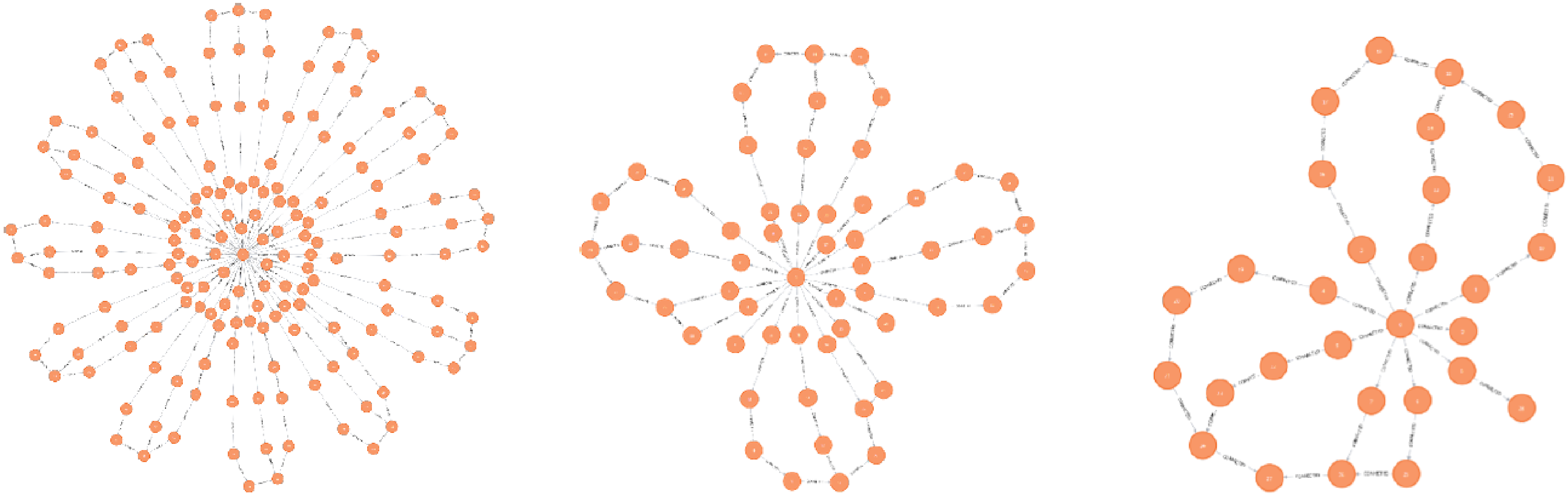
INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING

acm In-Cooperation sighpc

Simulation Study – Experimental Setup

- Based on ensemble clustering with only *k-means* with variation for data and number of clusters
- Variation in $\widehat{M}$ studied for occurrence of RR under different random distributions
- Built a generic FTiP – Simulator from scratch for, studying RR computation across different ML models
- Simulator written using python, neo4j graph database

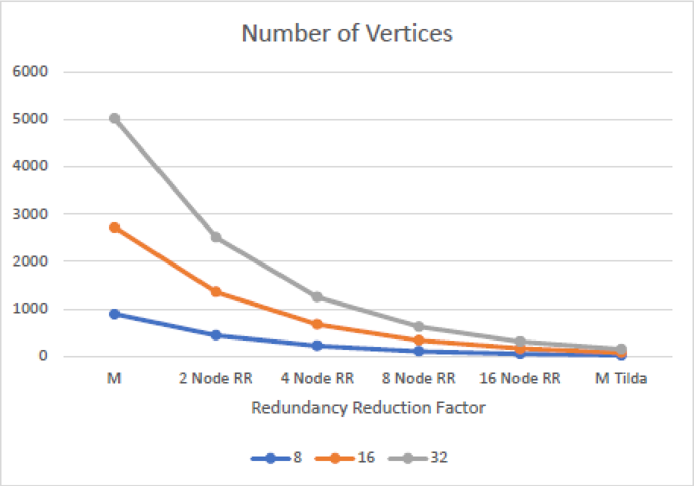
Simulation Study – Results

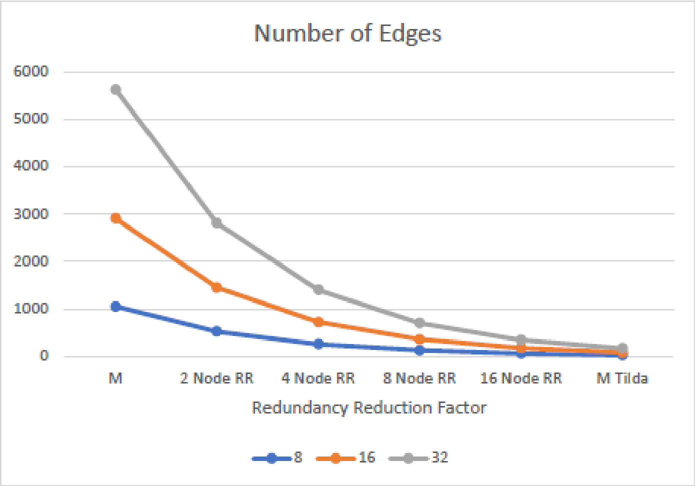


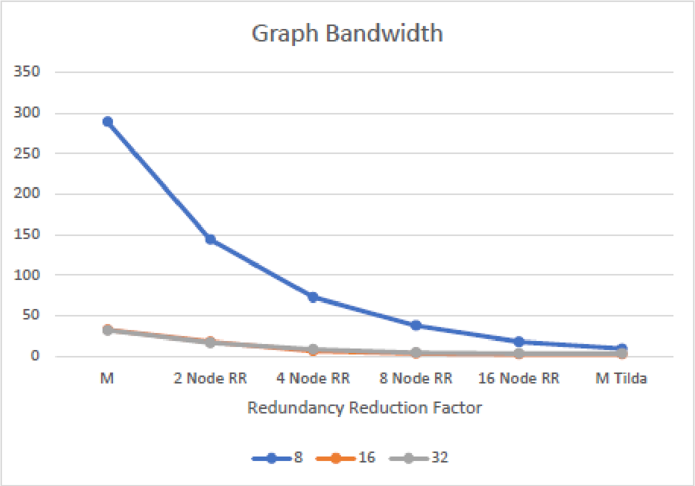
(a) $(\mathcal{M}, \mathcal{G})$ (b) $(\mathcal{M}_a, \mathcal{G}_a)$ (c) $(\widetilde{\mathcal{M}}, \widetilde{\mathcal{G}})$

- DAGs resulting from cu's in (a) M, (b) M_a, and (c) $\widehat{M}$, with $RR(M) \leq RR(M_a) \leq RR(\widehat{M})$
- Variation in $\widehat{M}$ i.e. M_a generated for RR under normally distributed randomization

Simulation Study – Results (Contd.)







- Metrics on DAGs resulting from cu's in M, M_a, and $\widehat{M}$ for $\left|D^{(t)}\right| = 8, 16, 32$, corresponds to a variation in RR
- Results clearly indicate for presence of lower and upper bounds
- Clear evidence for existence of opportunities to optimize

CONCLUSIONS & FUTURE WORK

- Explored avenues for further improvements in computational performance in EM algorithms
- We work at the intersection of high-performance computation, compiler, and reinforcement learning algorithms
- Proposed a novel theoretical framework which can help us identify previously undetected redundancies
- Solve the RRC problem for a restricted case
- Extend simulation experiments for various randomization of RR
- Study a wider ML models for RRC

Thanks to NIH, NSF, WMU and URI for partial support. Thanks to reviewers for valuable feedback that resulted in improved paper.

Q&A