SAND2019-5812C

17th Annual Workshop on
Charm++ and Its Applications
May 1&2, 2019

RECENT RESULTS IN DISTRIBUTED LOAD BALANCING

Philippe P. Pébaÿ[Δ] & Jonathan Lifflander[◊]
[Δ]NexGen Analytics, [◊]Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

NexGen Analytics
Sandia National Laboratories

Motivation

- Currently developed DARMA AMT runtime system has a need for efficient distributed load-balancing (LB) strategies for target applications.
  - ➞ Cf. J. Lifflander's presentation on DARMA
- DARMA is in the process of being reimplemented to reap LB benefits. End-use case still in development.
- EMPIRE-PIC (ES/EM) application will rely on LB adjustments between iterations, with moderate persistence.
- Communication-awareness is sought but the ability to optimally balance weights is a requirement.

Premises

- Centralized LB strategies don't scale; hierarchical LB ones have limited scalability and tend to be costly to run.

- For the scale of the problems we're considering, this only leaves out fully distributed LB approaches. Being inexpensive (ideally), these can be run more frequently. Historically those however yielded poor LB results.

- Distributed LB schemed described in Menon & Kalé [SC'2013] appears to be inexpensive to run, and to yield good processor/load distributions.

Premises

- Centralized LB strategies don't scale; hierarchical LB ones have limited scalability and tend to be costly to run.

- For the scale of the problems we're considering, this only leaves out fully distributed LB approaches. Being inexpensive (ideally), these can be run more frequently. Historically those however yielded poor LB results.

- Distributed LB schemed described in Menon & Kalé [SC'2013] appears to be inexpensive to run, and to yield good processor/load distributions.

- Using the same notations for continuity

starting point of this work

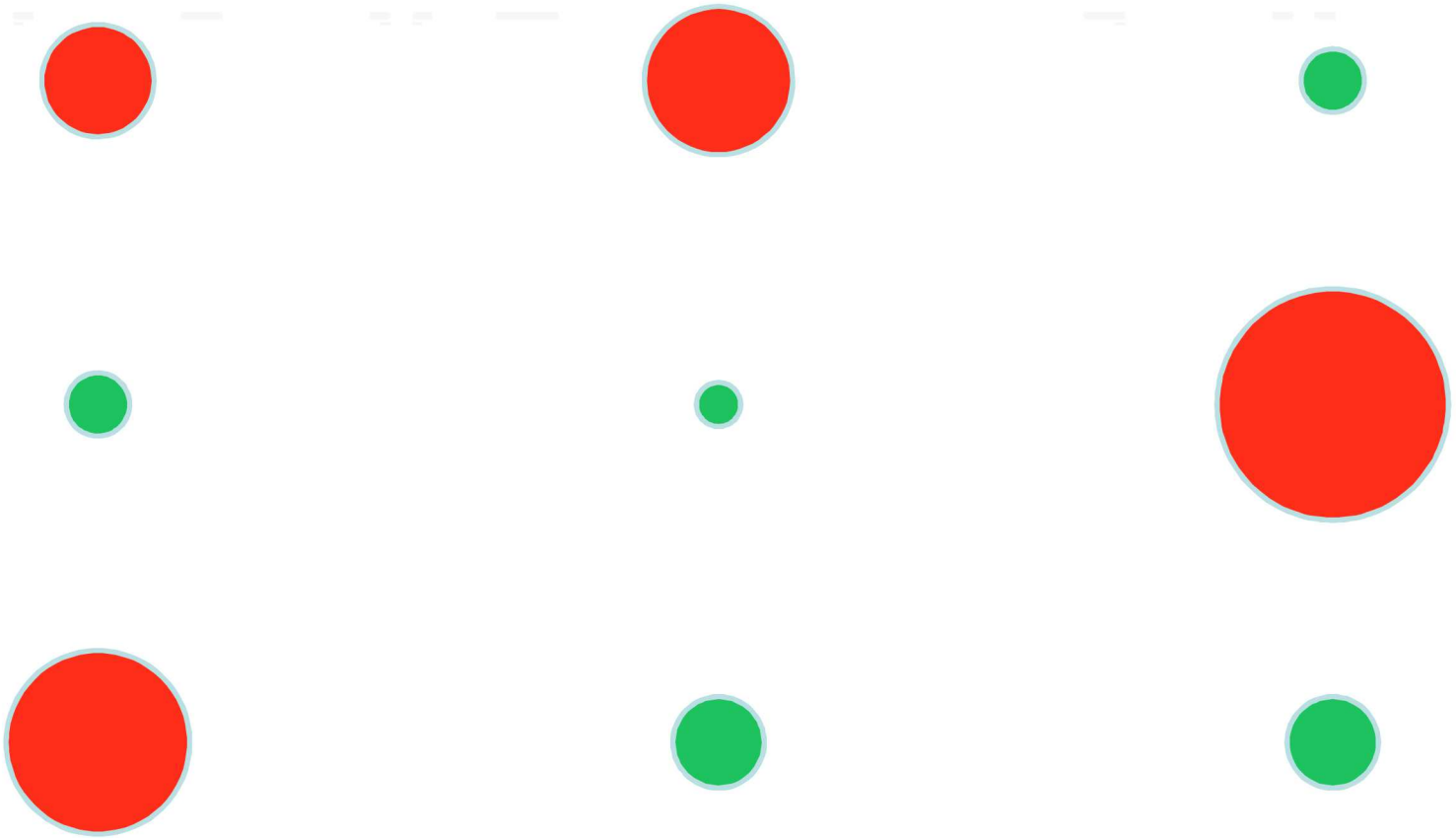
- Retaining the split in two main phases (*gossiping* and

The *GrapevineLB* Algorithm

- Distributed LB algorithm originally presented by Menon & Kalé at SC'13.
- Includes:
  1. an information propagation stage (*gossip*) to obtain partial processor/load distributions across system;
  2. a probabilistic transfer stage to offload work units from overloaded onto underloaded processors
- Original paper demonstrated performance gains as compared to centralized, distributed, and hierarchical LB for 2 applications:
  1. matches centralized strategies in terms of time/iteration, while avoiding associated bottlenecks.
  2. reduces total application time (molecular dynamics and AMR) with acceptable load distribution while incurring less overhead.

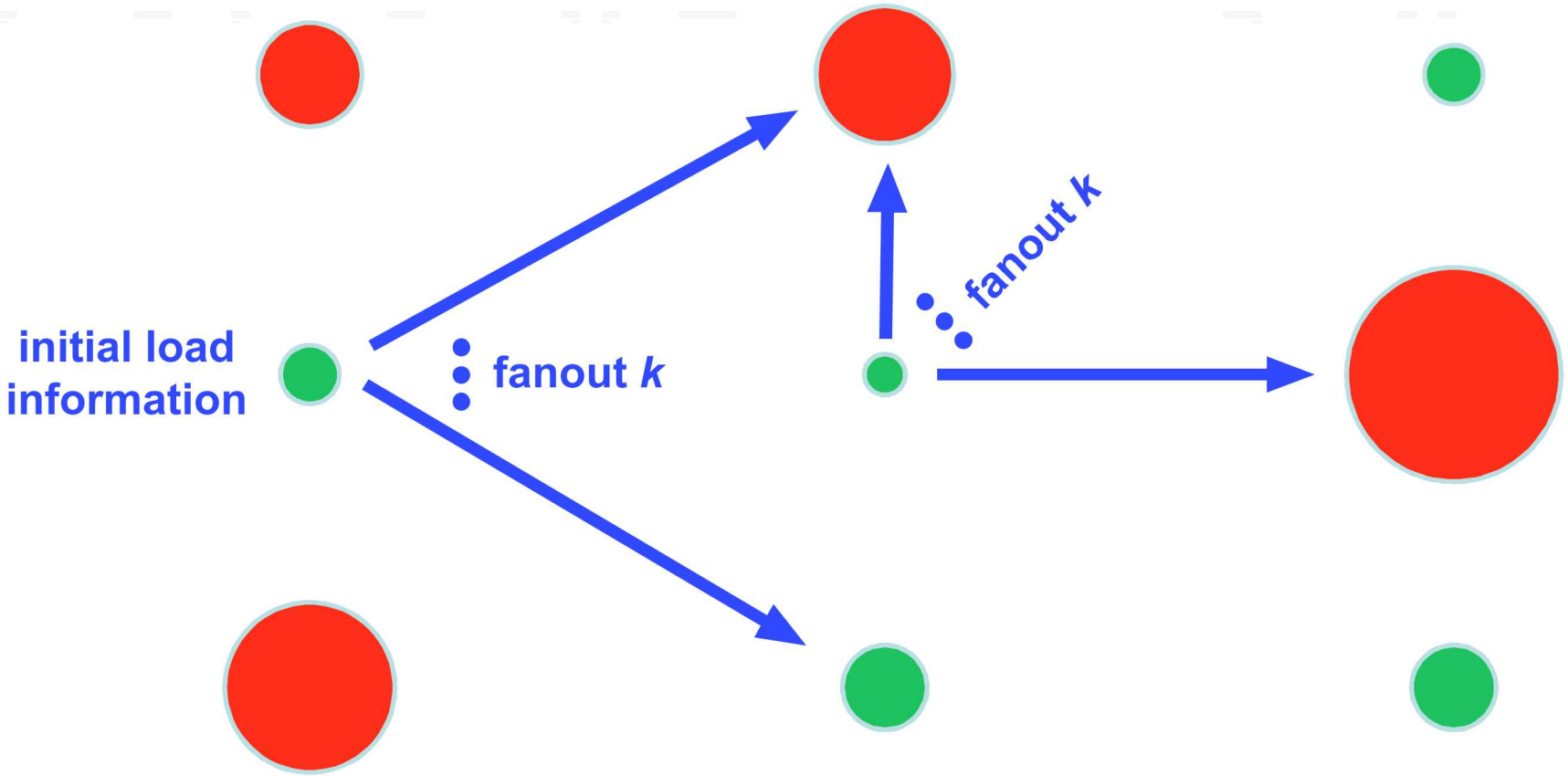
Initialization

$L_{underloaded} < L_{average} < L_{overloaded}$



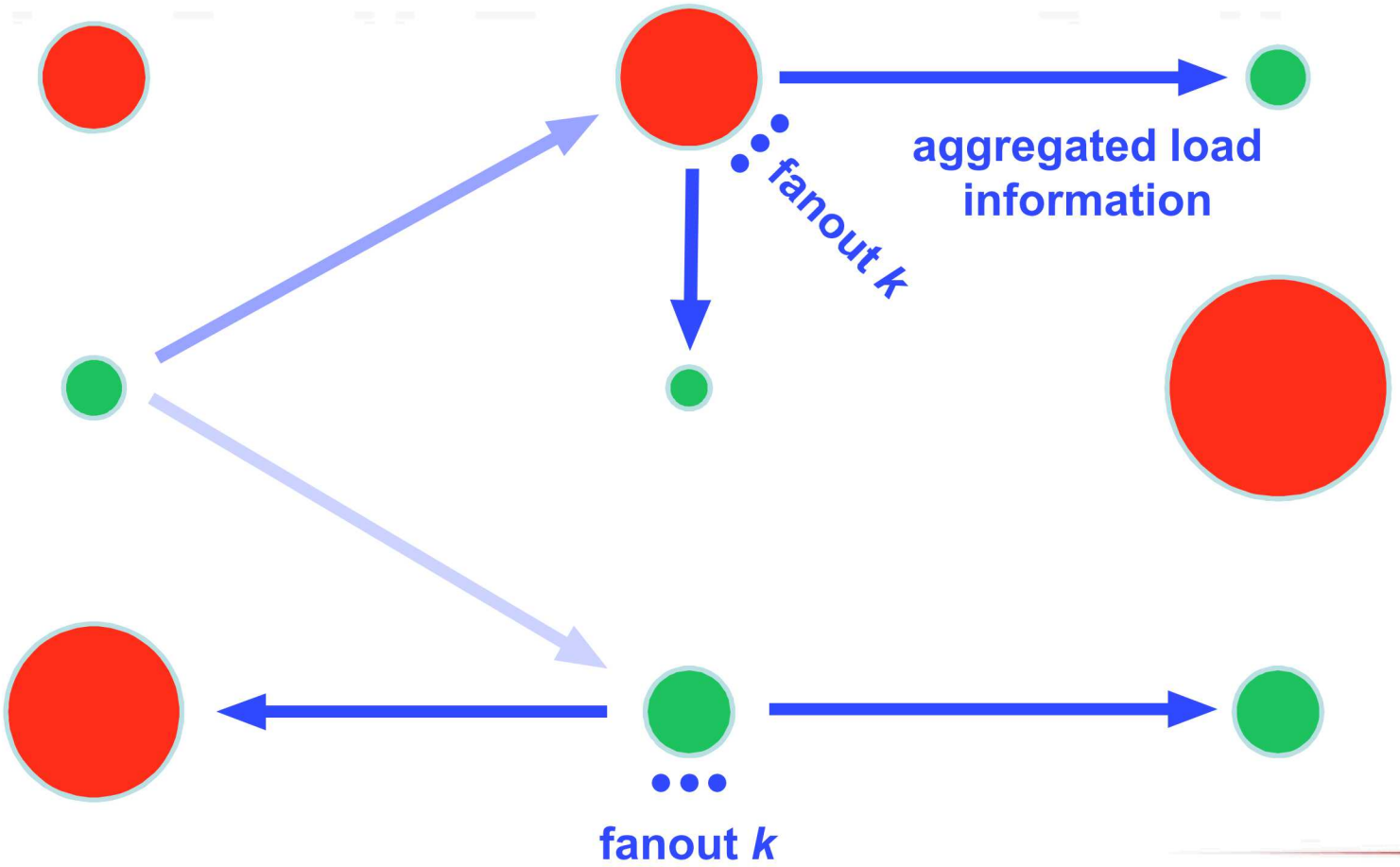
Gossiping Phase — round 1

For all underloaded



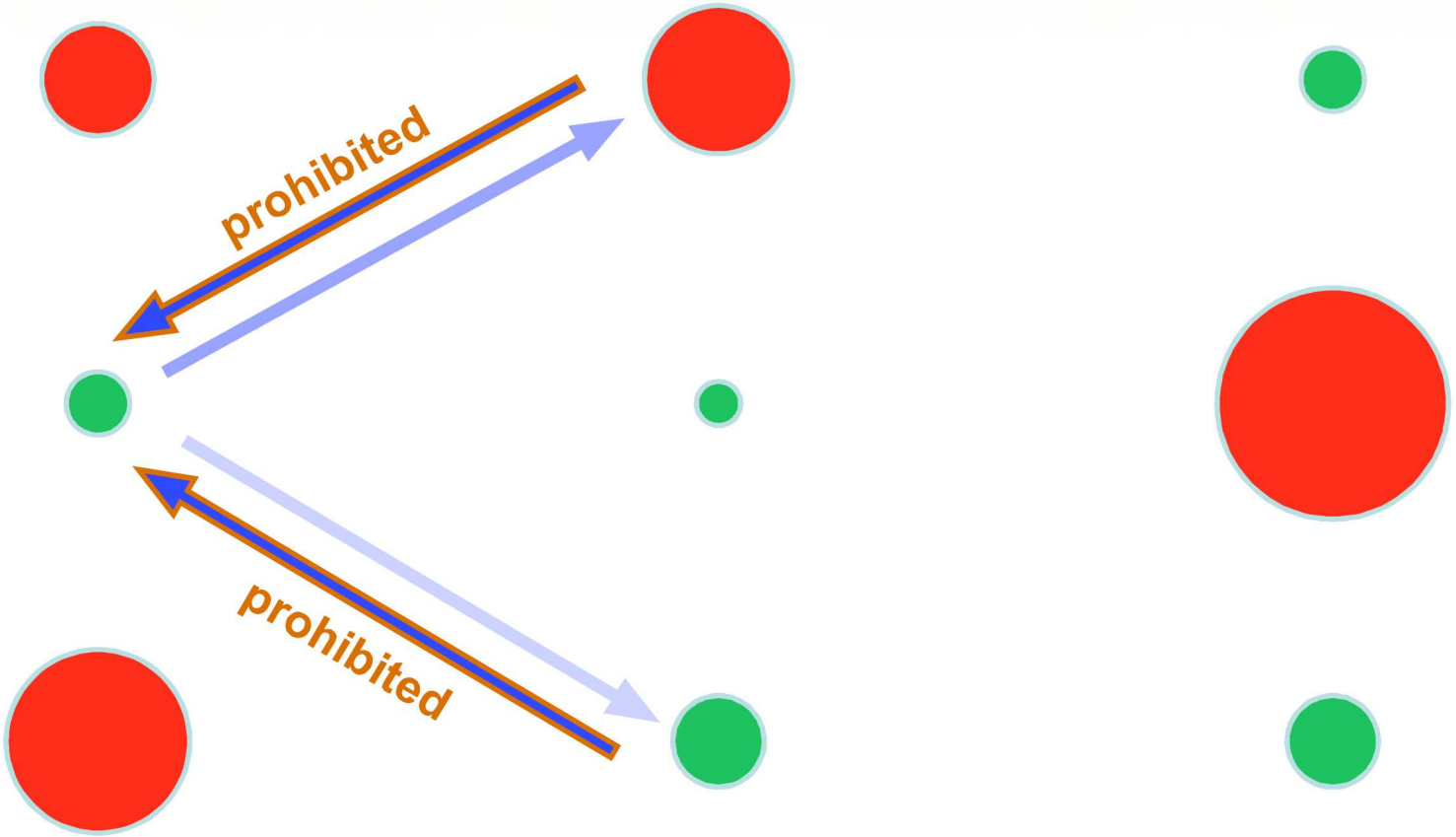
Gossiping Phase — rounds 2,…,*k*

For all recipients



Gossiping Phase — Informed Selection

Underload knowledge accumulates over rounds

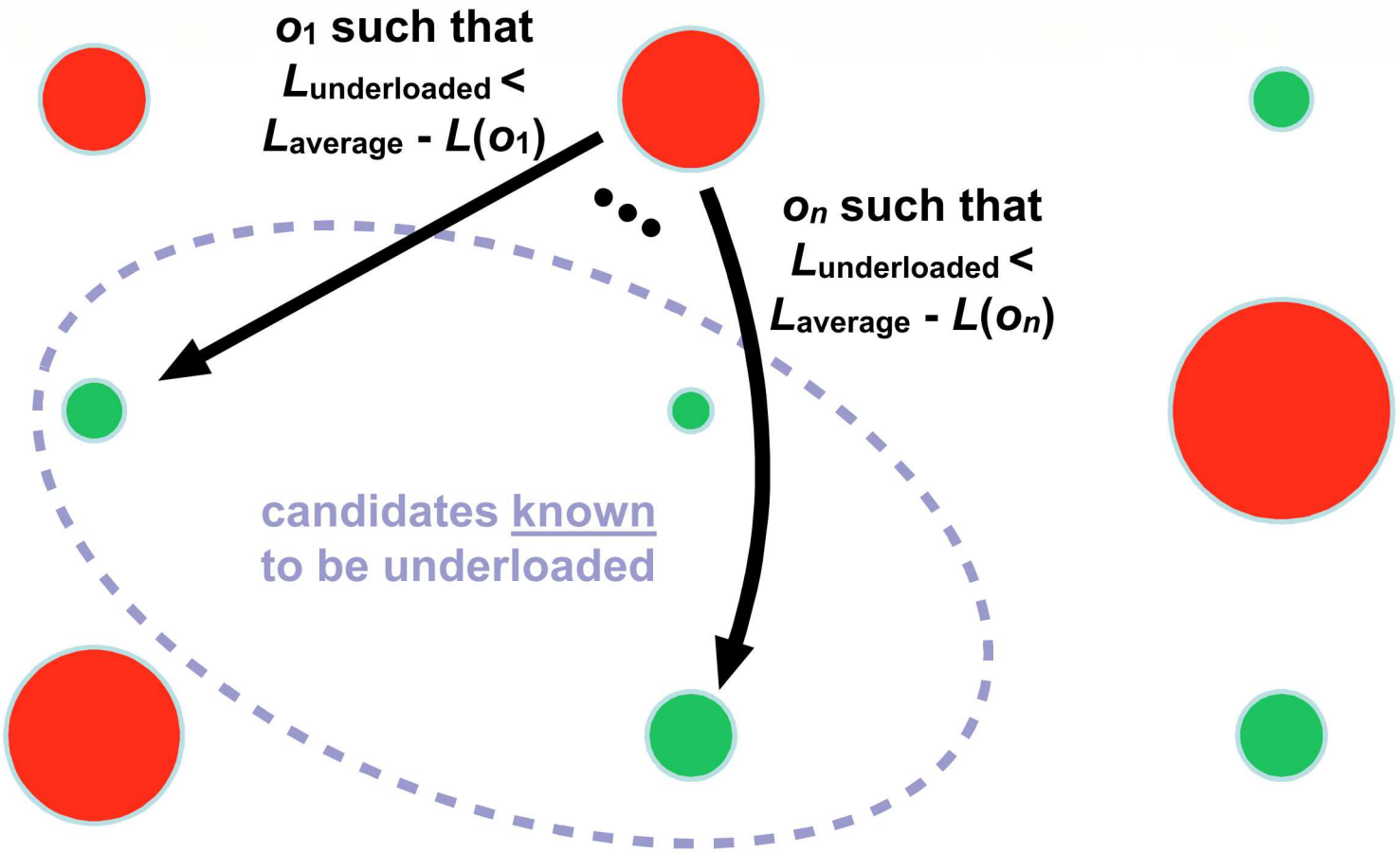


NexGen Analytics

Sandia National Laboratories

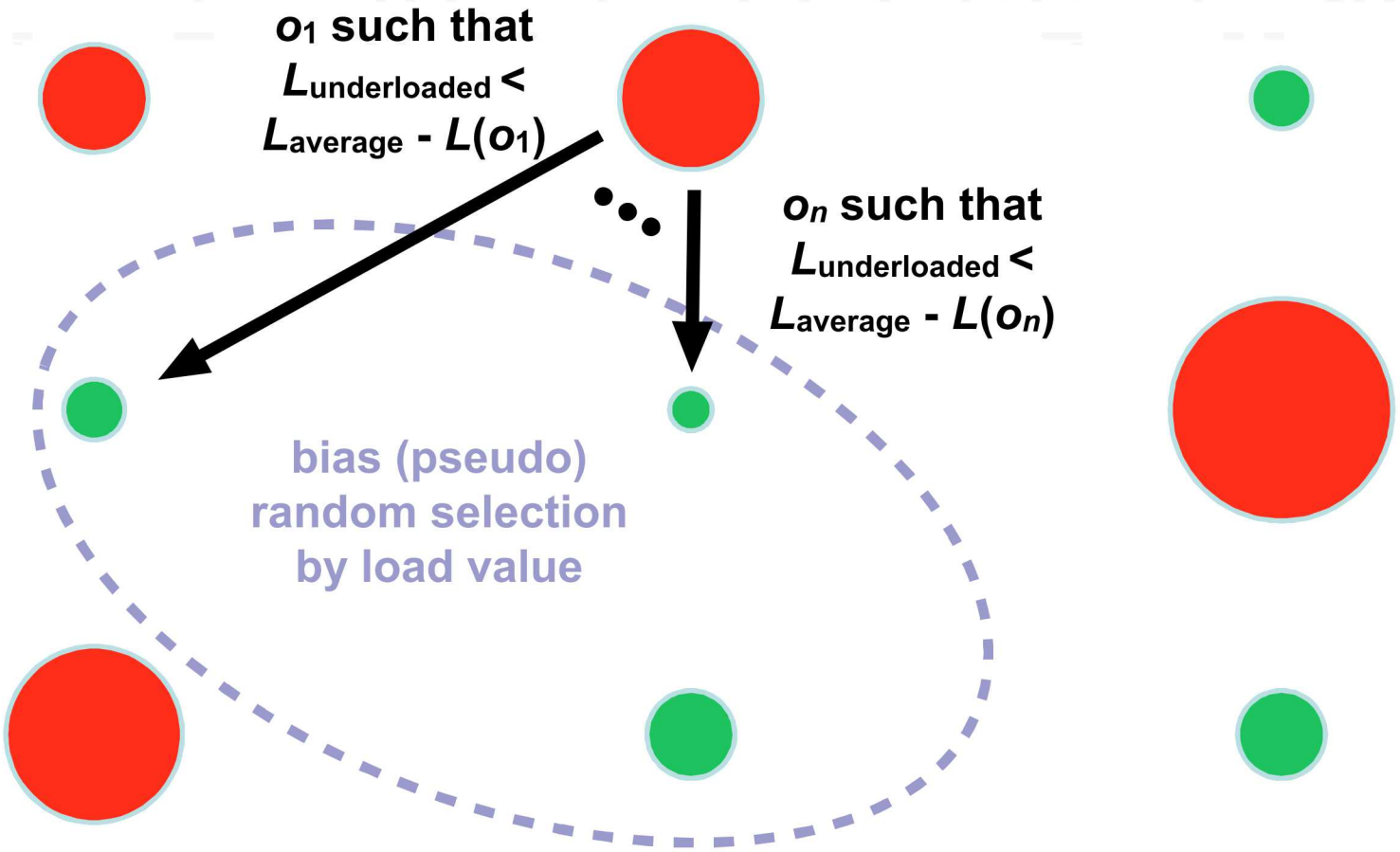
Transfer Phase

For all overloaded as long as $L > t_{overload} \times L_{average}$



Transfer Phase — Informed Selection

For all overloaded as long as $L > t_{overload} \times L_{average}$



Transfer Phase — Decision Criterion

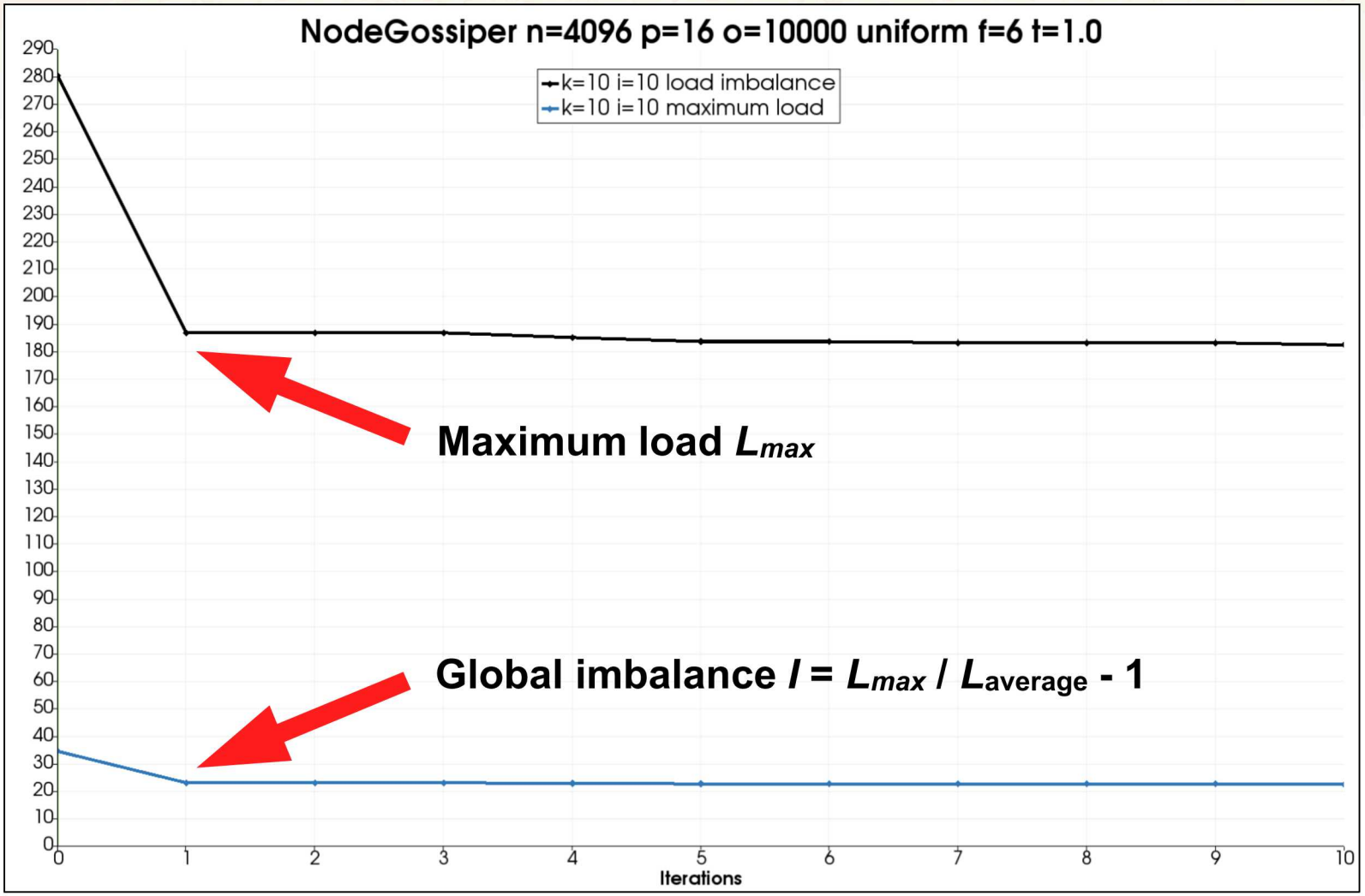
Input:
O - Set of objects in this processor
S - Set of underloaded processors
T - Threshold to transfer
L_i - Load of this processor
L_{avg} - Average load of the system

1: Compute $p_j \quad \forall\ P_j \in S$
2: Compute $F_j = \sum_{k<j} p_k$
3: **while** $(L_i > (T \times L_{avg}))$ **do**
4: Select object $O_i \in O$
5: Randomly sample $X \in S$ using F
6: **if** $(L_X + load(O_i) < L_{avg})$ **then**
7: $L_X = L_X + load(O_i)$
8: $L_i = L_i - load(O_i)$
9: $O \longleftarrow O \setminus O_i$
10: **end if**
11: **end while**

notice here

and there

Experimental Findings



Experimental Findings

iteration (index)	transfers (number)	rejected (number)	rejection rate (%)	imbalance ($\mathcal{I}$)
0				280
1	9084	154931	94.46	187
2	4	1654	99.76	187
3	1	1130	99.91	187
4	7	2682	99.74	185
5	6	2396	99.75	183
6	2	1143	99.83	183
7	1	1041	99.90	183
8	0	882	100.0	183
9	0	882	100.0	182
10	3	1405	99.79	182

Experimental Findings

- Confirming original findings that increasing fanout rapidly had diminishing returns.

- In particular increasing fanout/number of round does not resolve sub-optimal convergence even with perfect information.

- Changing *o*-picking strategy (which may require sorting and be costly) does not help either.

→ **high rejection rates hint at too tight criterion**

Criterion Analysis

- Confirming original findings that increasing fanout rapidly had diminishing returns.

- In particular increasing fanout/number of round does not resolve sub-optimal convergence even with perfect information.

- Changing *o*-picking strategy (which may require sorting and be costly) does not help either.

→ **high rejection rates hint at too tight criterion**

Transfer Phase — Monotonicity

Input:
O - Set of objects in this processor
S - Set of underloaded processors
T - Threshold to transfer
L_i - Load of this processor
L_{avg} - Average load of the system

1: Compute $p_j \quad \forall\ P_j \in S$
2: Compute $F_j = \sum_{k<j} p_k$
3: **while** $(L_i > (T \times L_{avg}))$ **do**
4: Select object $O_i \in O$
5: Randomly sample $X \in S$ using F
6: **if** $(L_X + load(O_i) < L_{avg})$ **then**
7: $L_X = L_X + load(O_i)$
8: $L_i = L_i - load(O_i)$
9: $O \longleftarrow O \setminus O_i$
10: **end if**
11: **end while**

try to decrease all overloads ($||.||_\infty$)…

Transfer Phase — Monotonicity

Input:
O - Set of objects in this processor
S - Set of underloaded processors
T - Threshold to transfer
L_i - Load of this processor
L_{avg} - Average load of the system

1: Compute $p_j \quad \forall\, P_j \in S$
2: Compute $F_j = \sum_{k<j} p_k$
3: **while** $(L_i > (T \times L_{avg}))$ **do**
4: Select object $O_i \in O$
5: Randomly sample $X \in S$ using F
6: **if** $(L_X + load(O_i) < L_{avg})$ **then**
7: $L_X = L_X + load(O_i)$
8: $L_i = L_i - load(O_i)$
9: $O \longleftarrow O \setminus O_i$
10: **end if**
11: **end while**

try to decrease all overloads ($||.||_\infty$)…
… but potentially reject global improvements if locally worse ($||.||_1$).

Transfer Phase — Modified Criterion

- **Proposal**: replace **6**:

$$\textbf{if } L(o) < L_{\text{average}} - L_{\text{underloaded}} \textbf{ then}$$

with **6'**:

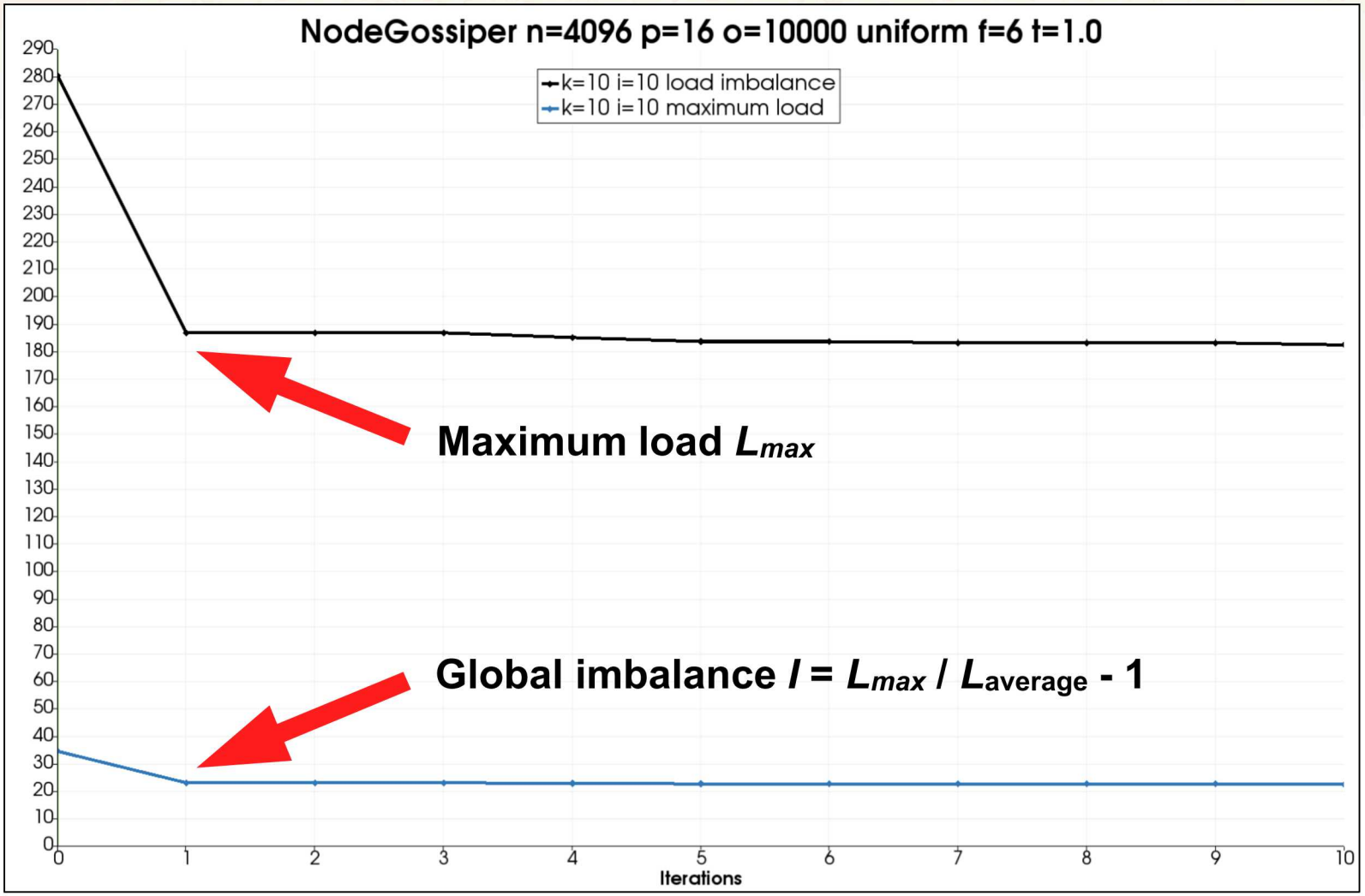
$$\textbf{if } L(o) < L_{\text{overloaded}} - L_{\text{underloaded}} \textbf{ then}$$

- **Lemma**: Alternate criterion **6'** is necessary and sufficient for monotonic decrease of Algorithm 2's objective function.

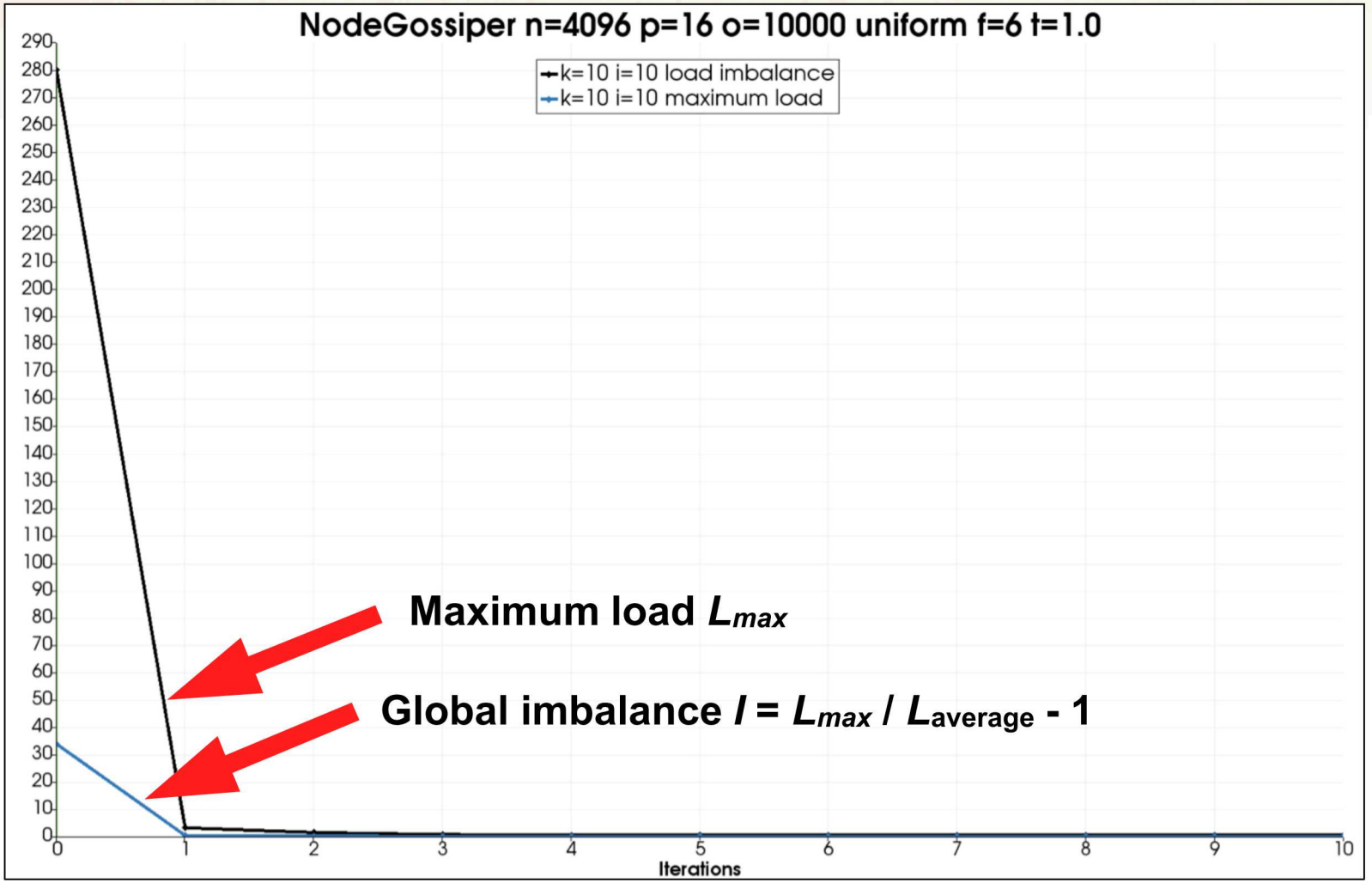
- **Remark**: Alternate criterion **6'** is weaker than **6**.

- **Proof**: cf. to-be-released SAND Report (Pébaÿ & Lifflander).

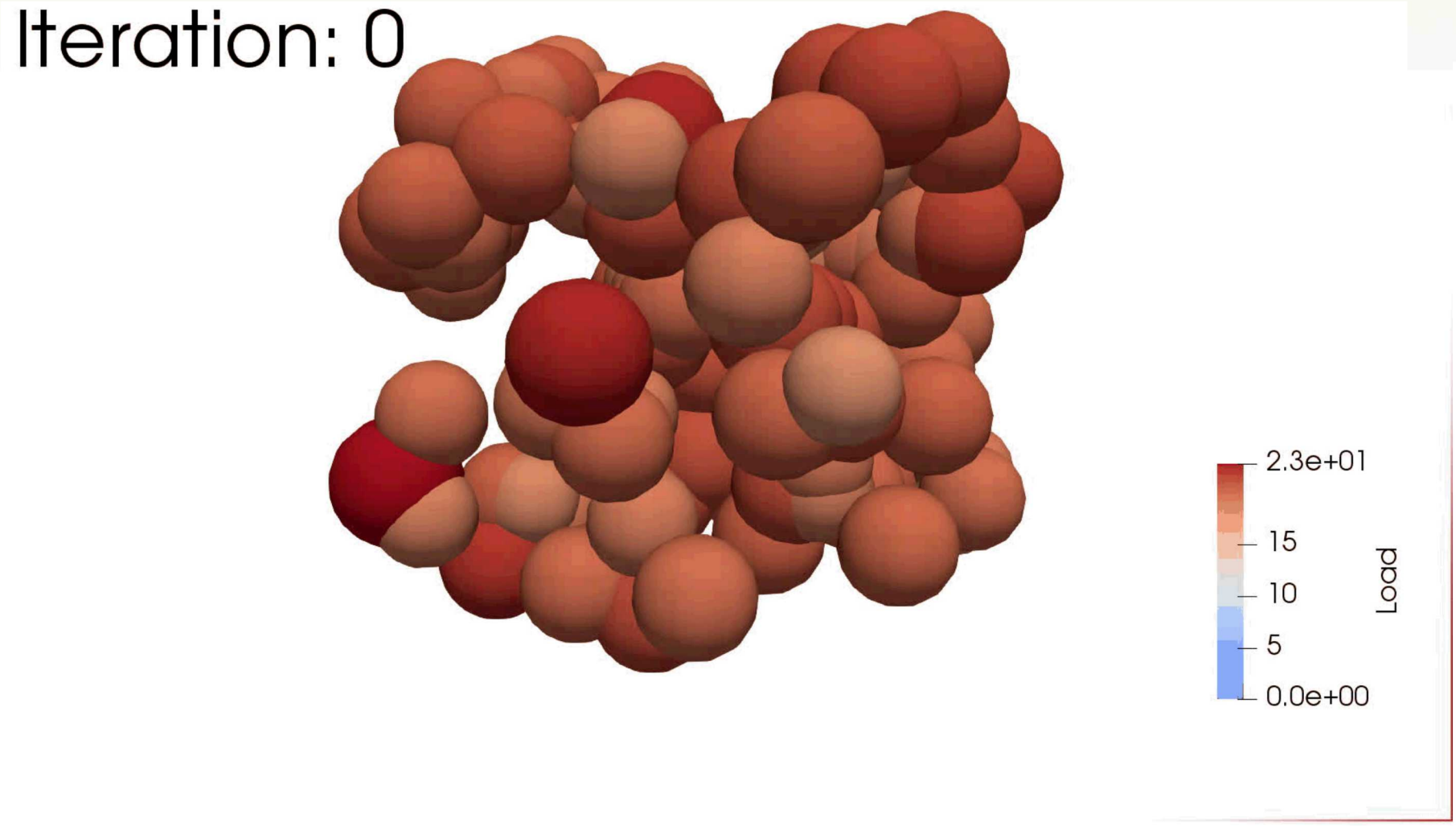
Experimental Findings with 6



Experimental Findings with 6’



Experimental Findings with 6’



Experimental Findings with 6

iteration (index)	transfers (number)	rejected (number)	rejection rate (%)	imbalance ($\mathcal{I}$)
0				280
1	9084	154931	94.46	187
2	4	1654	99.76	187
3	1	1130	99.91	187
4	7	2682	99.74	185
5	6	2396	99.75	183
6	2	1143	99.83	183
7	1	1041	99.90	183
8	0	882	100.0	183
9	0	882	100.0	182
10	3	1405	99.79	182

Experimental Findings with 6’

iteration (index)	transfers (number)	rejected (number)	rejection rate (%)	imbalance ($\mathcal{I}$)
0				280
1	11292	648	5.427	3.34
2	4044	3603	47.12	1.60
3	2201	3412	60.79	0.873
4	1324	3586	73.03	0.632
5	765	3171	80.56	0.632
6	410	2969	87.87	0.626
7	247	2794	91.88	0.626
8	159	2749	94.53	0.626
9	120	2682	95.72	0.626
10	72	2643	97.35	0.623

Experimental Findings 6 *vs.* 6'

iteration (index)	criterion 6 ($\mathcal{I}$)	criterion 6' ($\mathcal{I}$)
0	280	280
1	187	3.34
2	187	1.60
3	187	0.873
4	185	0.632
5	183	0.632
6	183	0.626
7	183	0.626
8	183	0.626
9	182	0.626
10	182	0.623

no longer locked in local optimum

2^{15} instead of 10^4 objects with 6'

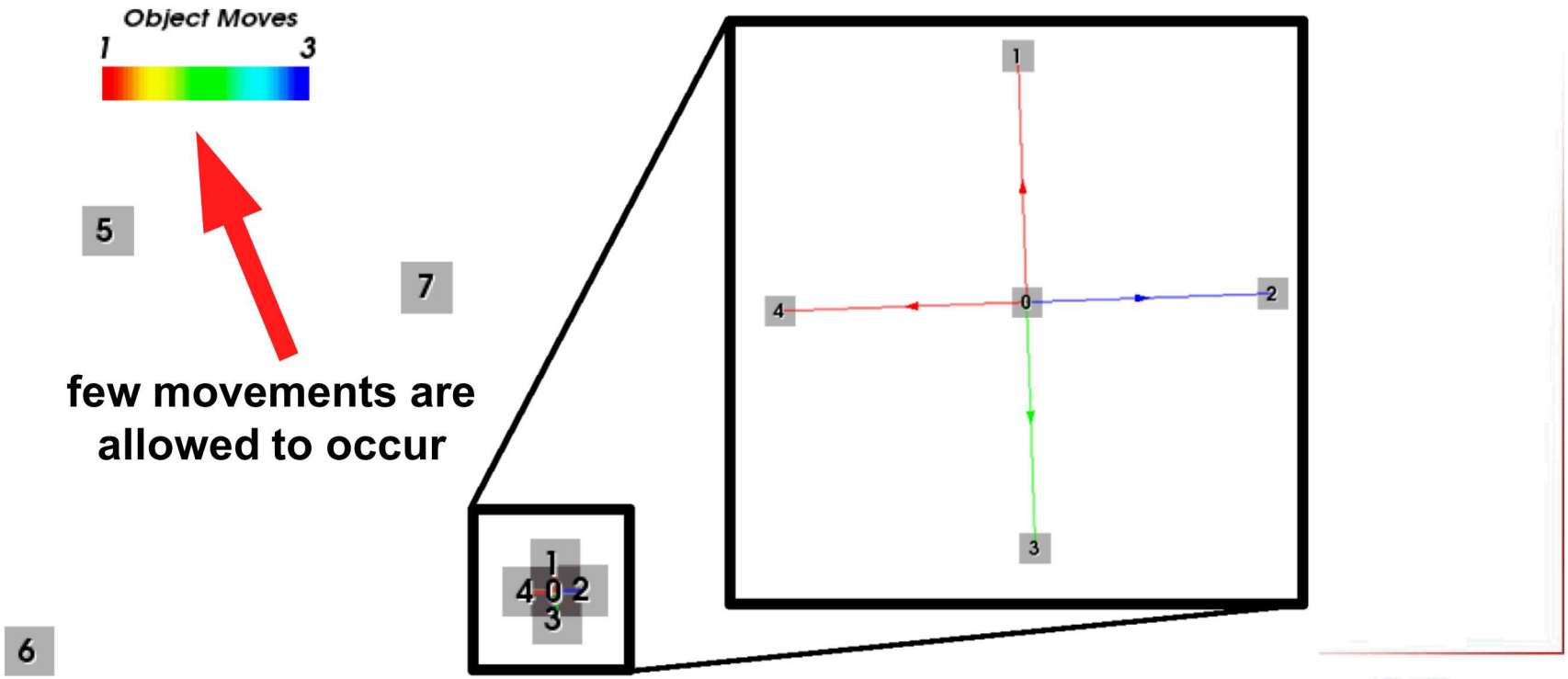
iteration (index)	transfers (number)	rejected (number)	rejection rate (%)	imbalance ($\mathcal{I}$)
0				270
1	33761	269	0.790	1.52
2	7179	3642	33.66	0.714
3	4227	5946	58.45	0.399
4	3304	9023	73.20	0.317
5	2496	9570	79.31	0.273
6	1978	10138	83.67	0.187
7	1419	9895	87.46	0.139
8	1026	9312	90.08	0.139
9	692	8938	92.81	0.139
10	463	8379	94.76	0.139

***I* improves as object-to-processor ratio increases (8 > 2.5)**

Prescribed VT Object Moves with 6

Application to real VT case with 8 processors and 100 objects, *k*=*f*=4, *i*=5

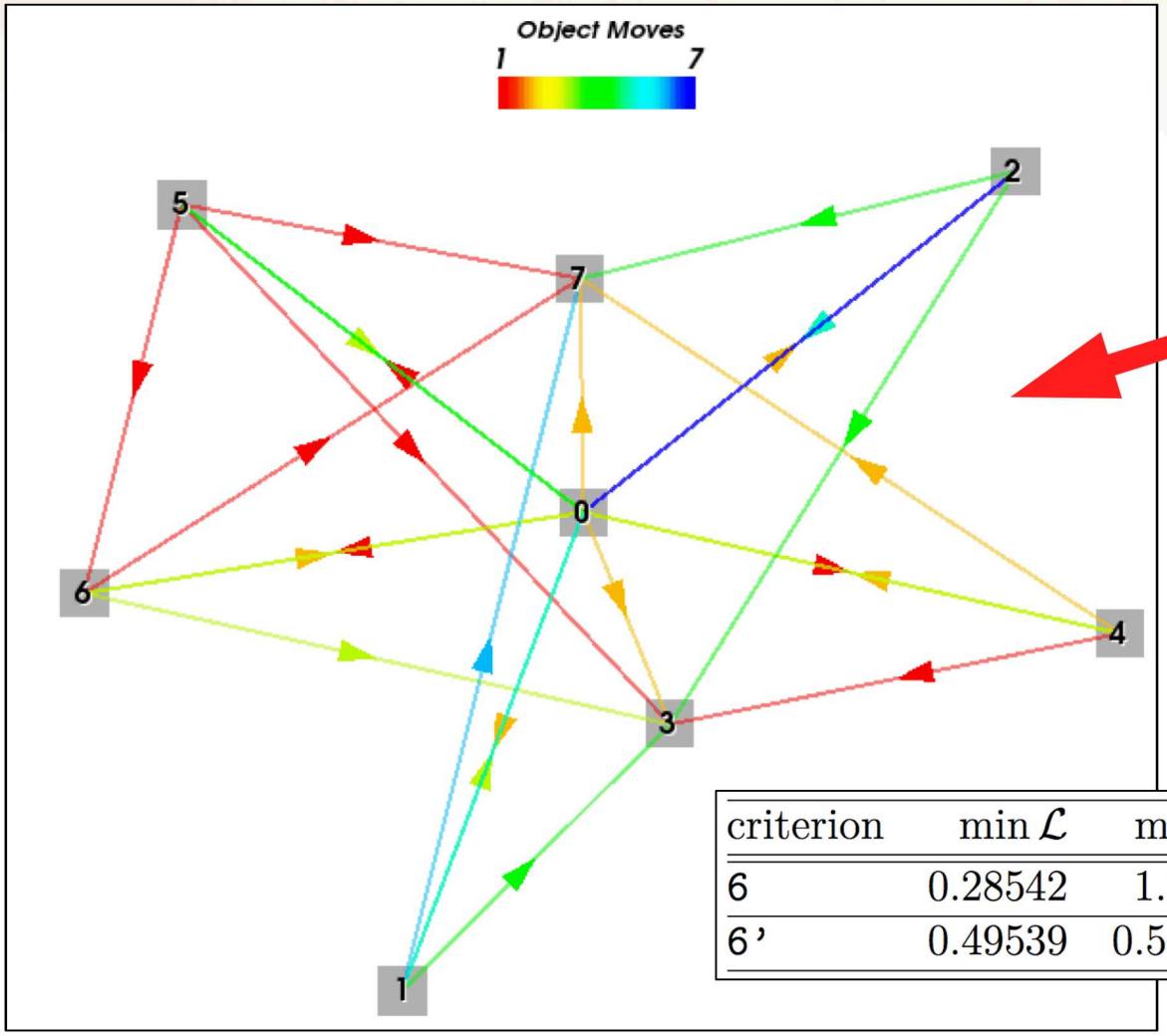
$\mathcal{D}$	$\lvert\mathcal{D}\rvert$	$\min \mathcal{D}$	$\overline{\mathcal{D}}$	$\max \mathcal{D}$	$R_{\mathcal{D}}$	$\sigma_{\mathcal{D}}$	$\gamma_{1,\mathcal{D}}$	$\gamma_{2,\mathcal{D}}$	$\mathcal{I}_{\mathcal{D}}$
$\mathbf{O}$	100	0.00026703	0.042076	0.30404	0.30377	0.074587	2.8855	9.8574	N/A
$\mathcal{L}$	8	0.004673	0.52595	2.3209	2.3162	0.69632	2.030	5.5987	3.4128



NexGen Analytics

Sandia National Laboratories

Prescribed VT Object Moves with 6’



many movements are allowed to occur

load range ~ object standard deviation; optimality achieved?

criterion	$\min \mathcal{L}$	$\max \mathcal{L}$	$R_{\mathcal{L}}$	$\sigma_{\mathcal{L}}$
6	0.28542	1.7508	1.4654	0.46454
6’	0.49539	0.57315	0.077759	0.024726

Summary & Current Work

- Surprising performance findings made us deep-dive into algorithm and propose improvement that appears substantial.
- Integrating modified algorithm into DARMA.
- Developing a parallel performance model (time and size complexity).
  - ➞ Time complexity $O[f \times \min(k, \log_f(n_p)) + n_o + 2\ \Omega(2 \log(n_p))]$
- Developing an objective function integrating communication costs.

Thank You

philippe.pebay@ng-analytics.com

gliffla@sandia.gov

NexGen Analytics

Sandia National Laboratories