Order/Radix Problemにおける対称性とホストの偏りを利用した最適化アルゴリズムの提案

中尾 昌広[†] 塚本 雅生[‡] 花田 良子[‡] 山本 啓二[†]

† 理化学研究所 計算科学研究センター

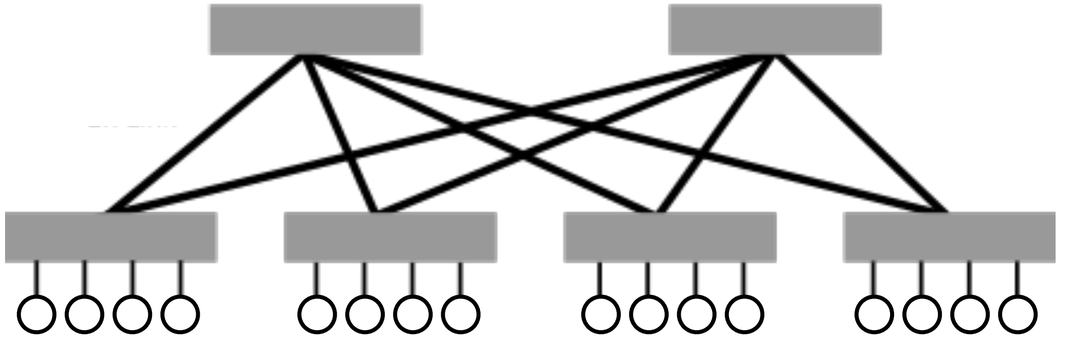
‡関西大学 システム理工学部

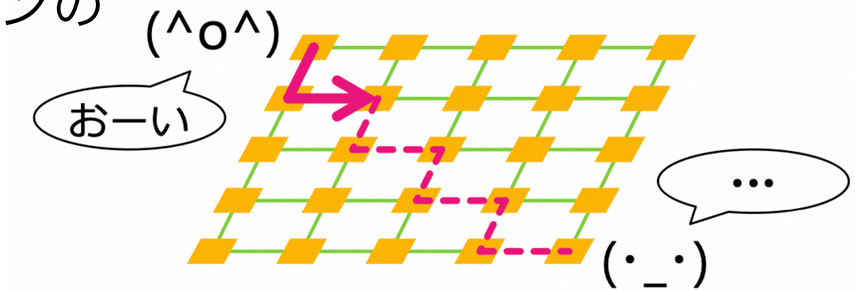
もくじ

- 背景：Order/Radix Problemとは
  - 既存研究[R. Yasudo, 2019]
  - 国際コンペティションGraphGolf
- 目的：Order/Radix Problemに対する最適化アルゴリズムの開発
- 提案：既存のアルゴリズムに対して2つの工夫を追加する
  - ホストの偏り
  - 対称性
- 評価：既存研究との比較
- 考察：NAS Parallel Benchmarkを用いた性能評価
- まとめ

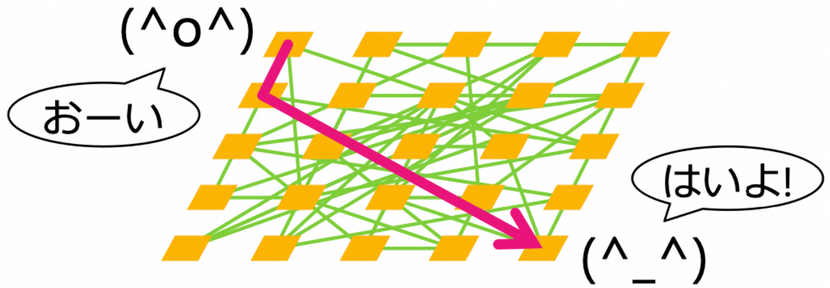
背景

- 並列計算機システムのネットワークトポロジは、
  並列アプリケーションの性能に強く影響を与える
- 規則的なトポロジ：Fat-tree, k-ary n-cube,
  Dragonfly, Slim Fly, etc.
- ランダムトポロジ： [M. Koibuchi, 2012]
  - スモールワールド効果
  - 多くの並列アプリケーションの
    性能が向上することが
    報告されている





▲ 規則的なトポロジ：
メッセージが届くのに時間がかかる

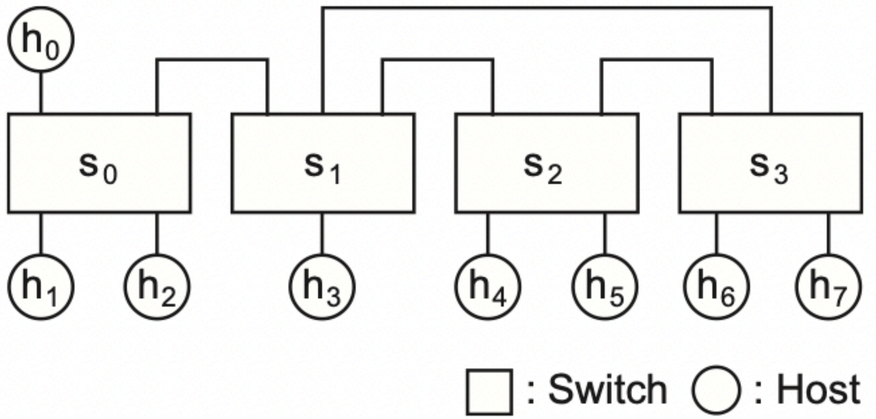


▲ ランダムトポロジ：
近道があるのでメッセージがすぐ届く

https://www.nii.ac.jp/userdata/shimin/documents/H27/150820_2ndlec.pdf

Order/Radix Problem（ORP）

- グラフ理論に基づいた並列計算機システムの間接網に関する研究が注目を集めている
- グラフは頂点とエッジから構成されている
- ホストとスイッチを頂点，ケーブルをエッジとみなすことで、間接網をグラフと表現
- **ORPは、与えられた「ホスト数」と「スイッチのradix」を満たす最小の平均ホスト間距離（h-ASPL： host-Average Shortest Path Length）を持つグラフを発見する問題**
- スイッチ数は任意。ホストはスイッチとのみ隣接可能であり、スイッチはホスト or スイッチと隣接可能
- 各スイッチの最大隣接数はradixである



ホスト数：8、radix：4, スイッチ数：4

h	0	1	2	3	4	5	6	7
0		2	2	3	4	4	4	4
1			2	3	4	4	4	4
2				3	4	4	4	4
3					3	3	3	3
4						2	3	3
5							3	3
6								2
7								

ホスト間の距離行列

h-ASPL =
91/28 = 3.25

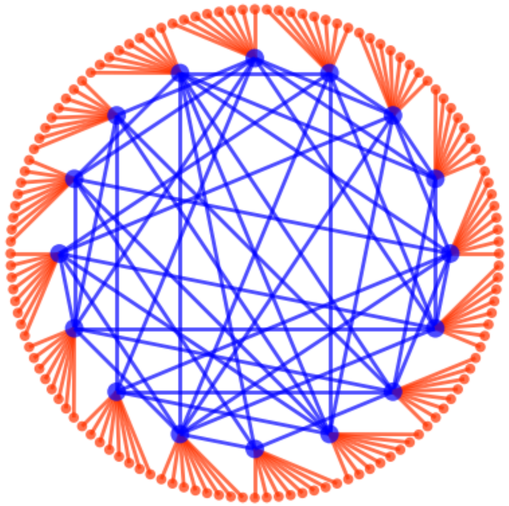
既存研究

- R. Yasudo et al. ‘’Designing High-Performance Interconnection Networks with Host-Switch Graphs’’, IEEE Transactions on Parallel and Distributed Systems, 2019
  - ORPの定義と数学的な性質の説明（h-ASPLの理論的な下界など）
  - Simulated Annealing（SA）を用いて、小さいh-ASPLを持つグラフを探索
  - 規則的なトポロジとの比較（シミュレータ上の並列ベンチマークの性能、電力、コスト）
  - SAによって最適化されたトポロジが他の規則的なトポロジと比較して優れていることを示した

本発表の目的

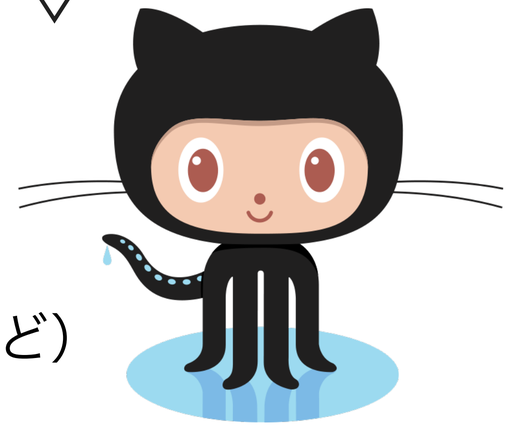
- ORPにおける解探索能力がより優れた最適化アルゴリズムの提案
  - 既存研究のSA[R. Yasudo, 2019]に、2つの工夫点を追加する
    - スイッチにホストを偏らせる
    - グラフに対称性を持たせる
- ORPのためのライブラリを作成

```
#include "orp.h"
…
ORP_Init_aspl(…);
for(int i=0;i<ITERATIONS;i++){
  /* Optimization */
  ORP_Set_aspl(...);
}
ORP_Finalize_aspl();
```



https://github.com/mnakao/ORP

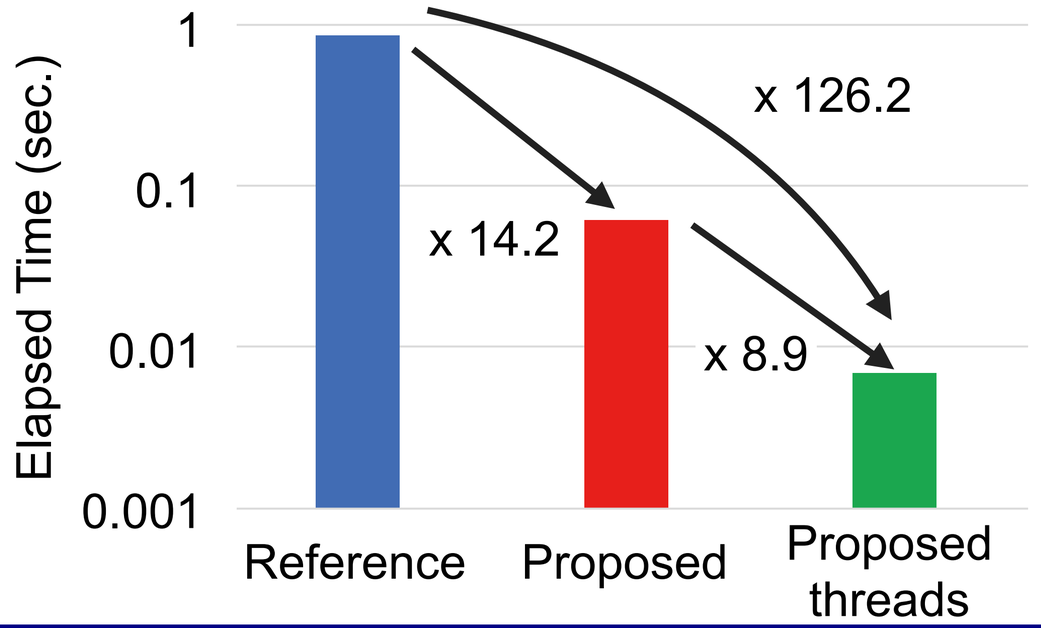
- h-ASPLの高速計算
  - C言語
  - スレッド並列
- ツール群（可視化など）



本発表の目的

- ORPにおける解探索能力がより優れた最適化アルゴリズムの提案
  - 既存研究のSA[R. Yasudo, 2019]に、2つの工夫点を追加する
    - スイッチにホストを偏らせる
    - グラフに対称性を持たせる
- ORPのためのライブラリを作成

```
#include "orp.h"
…
ORP_Init_aspl(…);
for(int i=0;i<ITERATIONS;i++){
  /* Optimization */
  ORP_Set_aspl(...);
}
ORP_Finalize_aspl();
```



hosts = 65536
radix = 64
switches = 4096

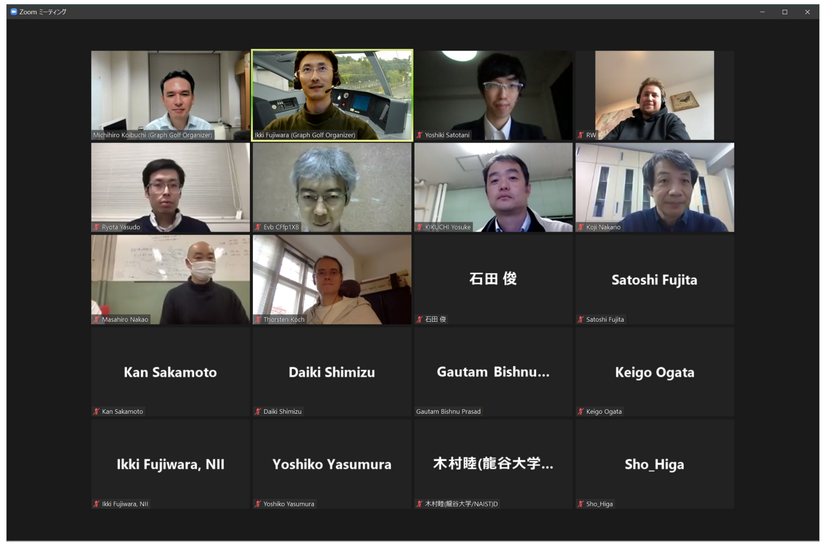
Intel Xeon Gold 6126
(2.6GHz, 12 Cores) x 2

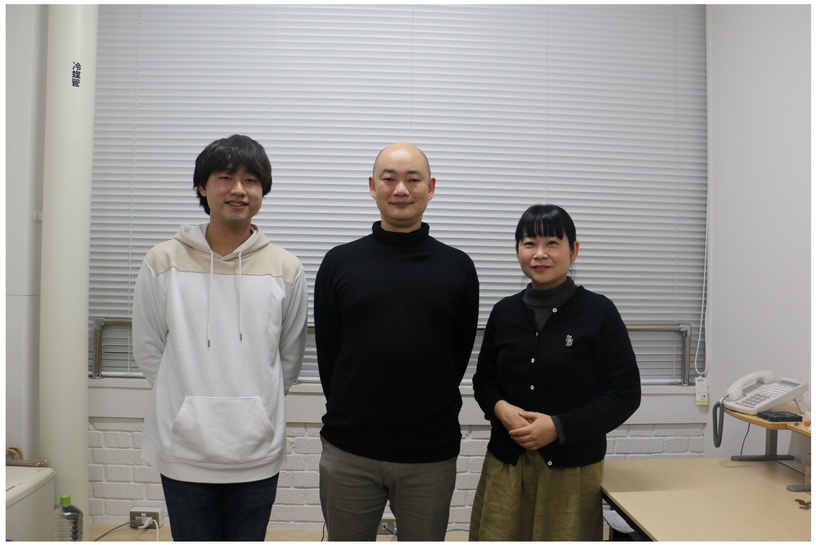
Cygnus system@筑波大

GraphGolf

http://research.nii.ac.jp/graphgolf/

- NIIが主催している国際コンペティション
- **間接網部門（Order/Radix Problem）**と直接網部門（Order/Degree Problem）
- あるホスト数とRadixの組合せが出題され、期間内（4/27〜10/11）にグラフを作成する
- (hosts, radix) = (32, 4), (80, 6), (128, 24), (432, 12), (1281, 21), (3800, 30), (1024, 5), (1024, 10), (4608, 36), (8208, 48), (10000, 10), (10000, 100), (65536, 64)



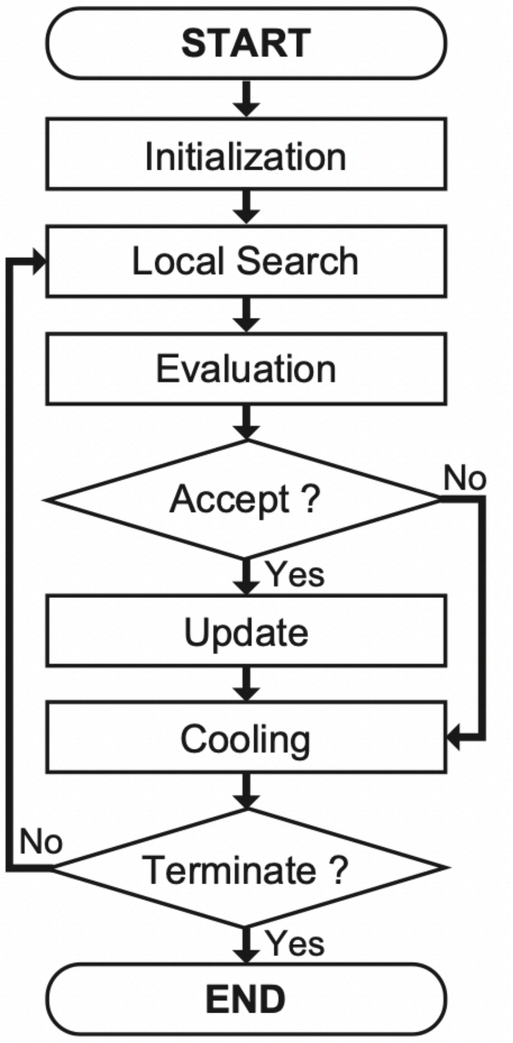


http://research.nii.ac.jp/graphgolf/2021/candar21/

もくじ

- 背景：Order/Radix Problemとは
  - 既存研究[R. Yasudo, 2019]
  - 国際コンペティションGraphGolf
- 目的：Order/Radix Problemに対する最適化アルゴリズムの開発
- 提案：既存のアルゴリズムに対して2つの工夫を追加する
  - ホストの偏り
  - 対称性
- 評価：既存研究との比較
- 考察：NAS Parallel Benchmarkを用いた性能評価
- まとめ

既存研究（SAをORPに適用）



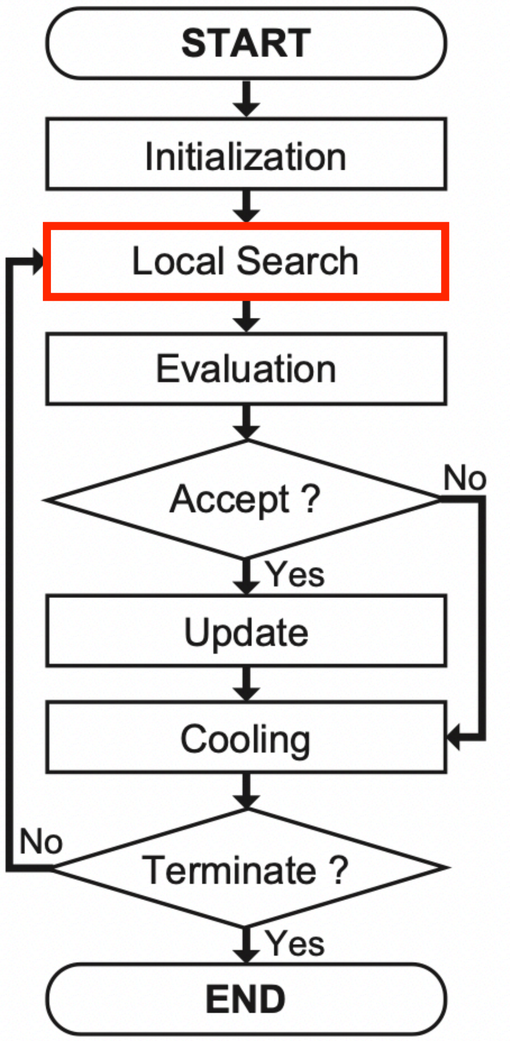
- Initialization：ランダムグラフを生成。初期温度を設定
- Local Search：グラフを少し変える
- Evaluation：h-ASPLを計算
- Accept：メトロポリス基準に従った受理判定
  - h-ASPLが小さくなった（改善）・・・必ず受理する
  - h-ASPLが大きくなった（改悪）・・・確率的に受理する

$$Probability = \begin{cases} 1 & \text{if } \Delta E' < 0 \\ \exp(-\frac{\Delta E'}{T}) & otherwise \end{cases}$$

温度が高いときほど、
改悪時の受理確率は高くなる

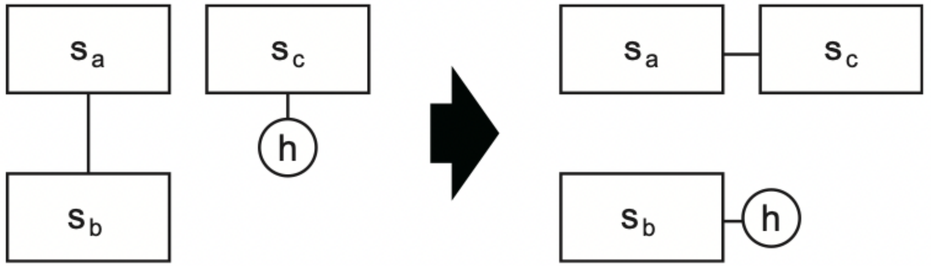
- Update：（受理されたとき）グラフを更新する
- Cooling：温度を下げる
- Terminate：規定回数を繰り返したら終了

既存研究（SAをORPに適用）

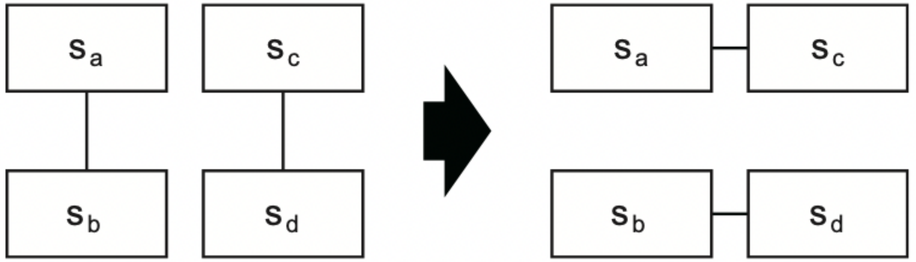


- **SWING**
  - ホストを移動させる
  - 1本のエッジを付け替える

s : Switch, h: Host



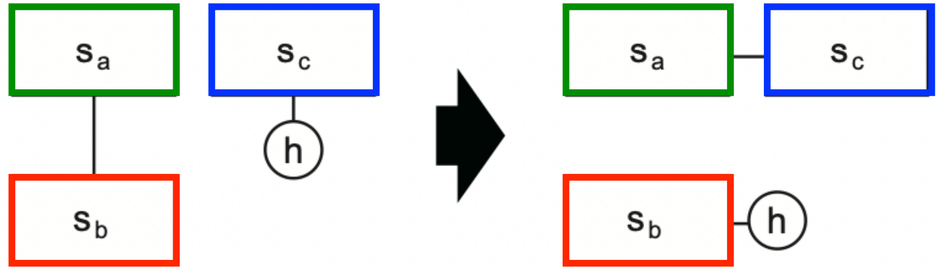
- **SWAP**
  - 2本のエッジを付け替える

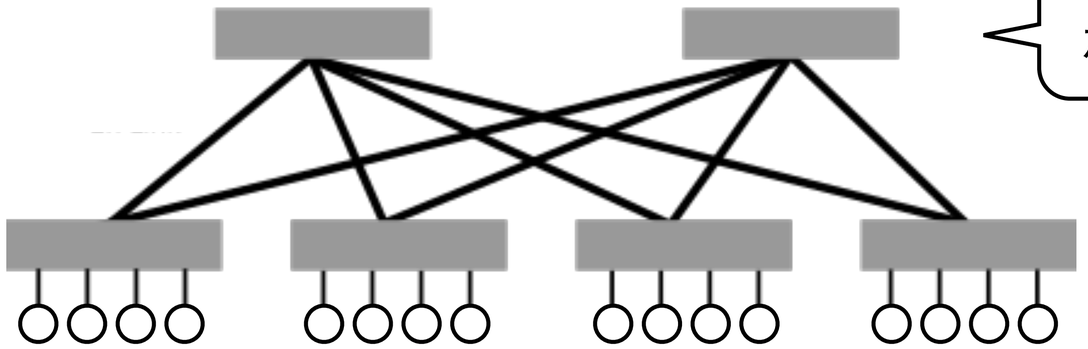


スイッチと隣接している頂点数（ホスト+スイッチの数）は変化しない

SWING操作における新しい工夫

- 既存のSWING操作の詳細
  1. グラフの中からランダムにエッジs_a - s_bと
     ホストを持つスイッチs_cを選択する
  2. スイッチs_bと隣接するホストを1つ増やし、
     スイッチs_cと隣接するホストを1つ減らす
  3. エッジs_a - s_bをエッジs_a - s_cに付け替える

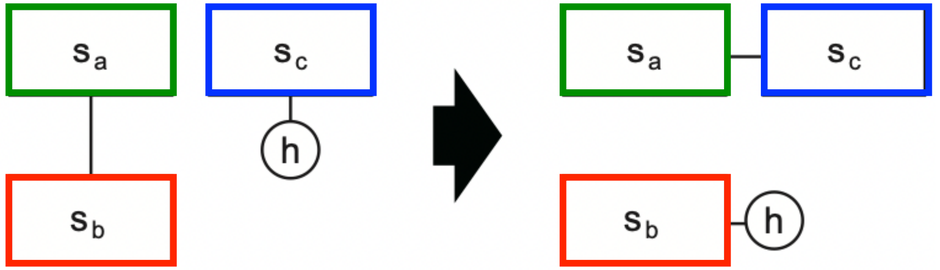




Fat-Treeのルートスイッチのように
ホストがないスイッチがある方が良い？

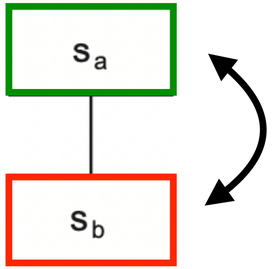
SWING操作における新しい工夫

- 既存のSWING操作の詳細
  1. グラフの中からランダムにエッジSa - Sbと
     ホストを持つスイッチScを選択する
  2. スイッチSbと隣接するホストを1つ増やし、
     スイッチScと隣接するホストを1つ減らす
  3. エッジSa - SbをエッジSa - Scに付け替える



ランダム選択

- 新しいSWING
  - 上の1.と2.の間で、SaとSbが持つホスト数を調べ、
    Sbが多くのホストを持つように確率的に置換する
  - この操作の目的は、多くのホストを持つスイッチは
    さらに多くのホストを持つようにすること



saとsbのスイッチ数をAとBとすると、
B/(A+B)の確率で置換を発生させる

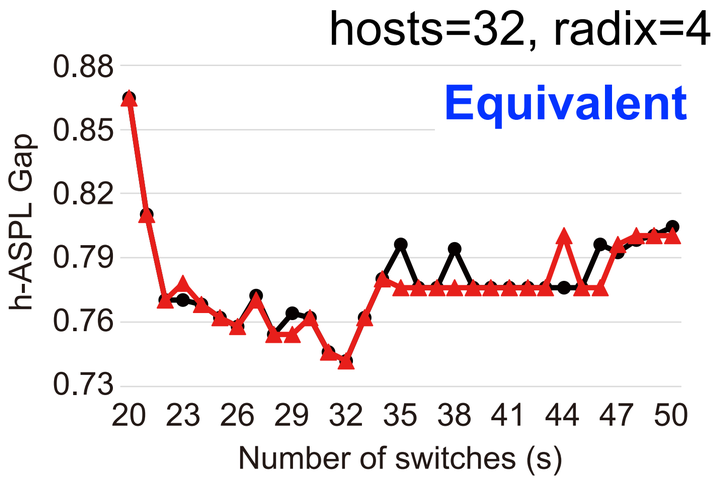
バイアス選択

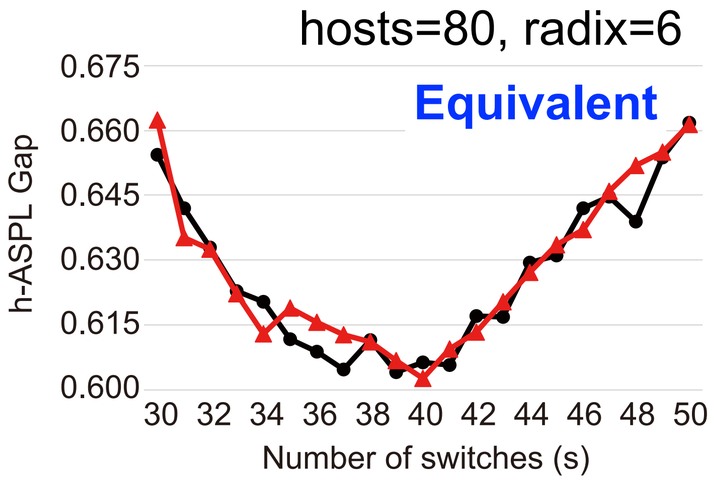
性能評価

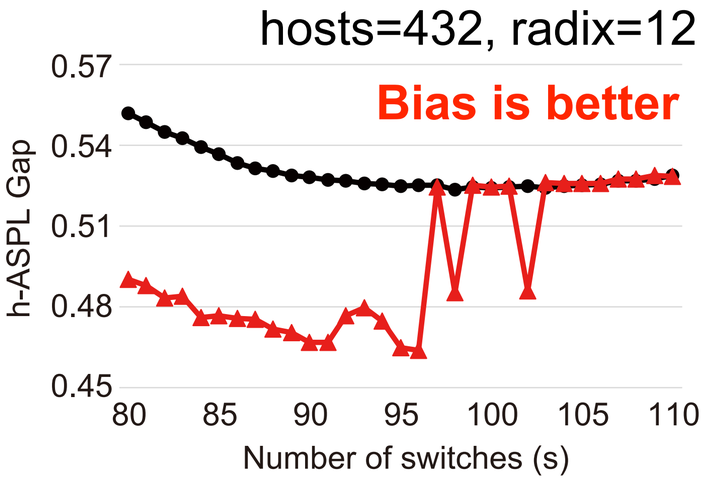
- ランダム選択とバイアス選択との比較
- ランダム選択は既存アルゴリズム[R. Yasudo, 2019]と同じ
- GraphGolf2021の問題を用いる
- 各問題において、スイッチ数を変化させて、それぞれ100万回の評価計算を行う
- 20試行の最良値
- h-ASPLと理論的下界との差である「h-ASPL Gap」を評価指標として用いる

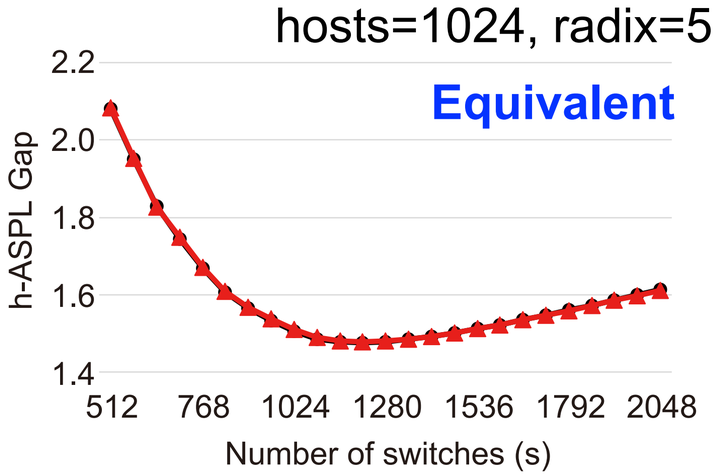
結果（1/2）

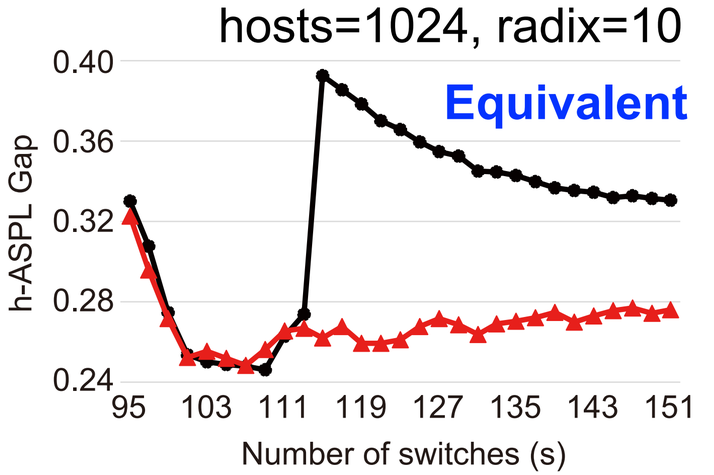

ランダム
バイアス

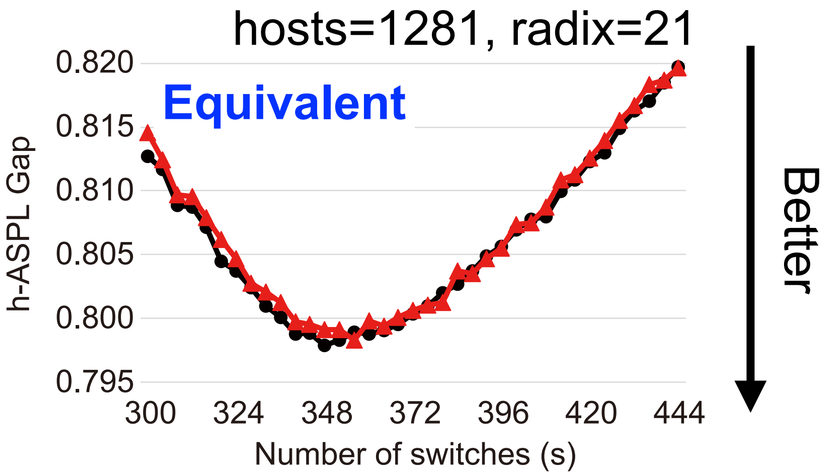

hosts=32, radix=4
Equivalent
h-ASPL Gap
0.88
0.85
0.82
0.79
0.76
0.73
20 23 26 29 32 35 38 41 44 47 50
Number of switches (s)


hosts=80, radix=6
Equivalent
h-ASPL Gap
0.675
0.660
0.645
0.630
0.615
0.600
30 32 34 36 38 40 42 44 46 48 50
Number of switches (s)


hosts=432, radix=12
Bias is better
h-ASPL Gap
0.57
0.54
0.51
0.48
0.45
80 85 90 95 100 105 110
Number of switches (s)

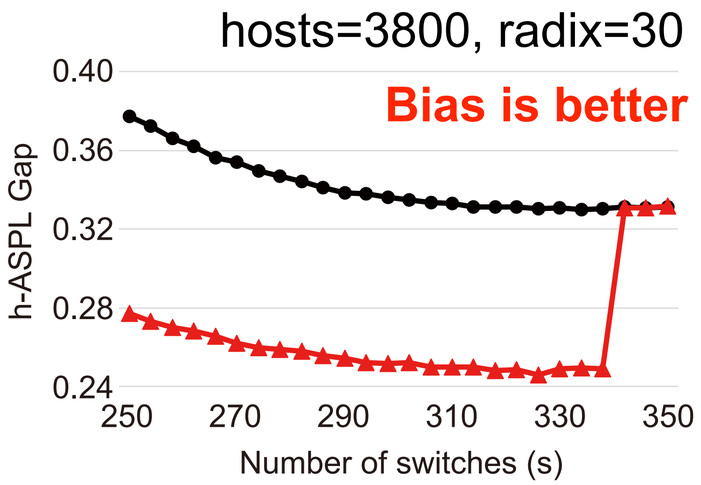

hosts=1024, radix=5
Equivalent
h-ASPL Gap
2.2
2.0
1.8
1.6
1.4
512 768 1024 1280 1536 1792 2048
Number of switches (s)

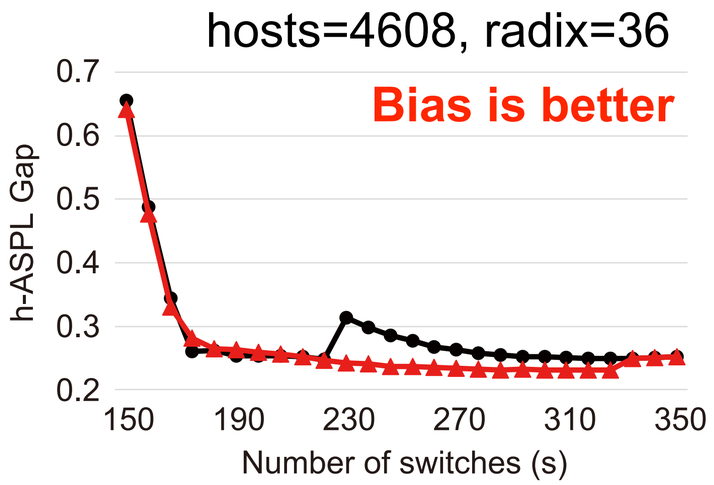

hosts=1024, radix=10
Equivalent
h-ASPL Gap
0.40
0.36
0.32
0.28
0.24
95 103 111 119 127 135 143 151
Number of switches (s)

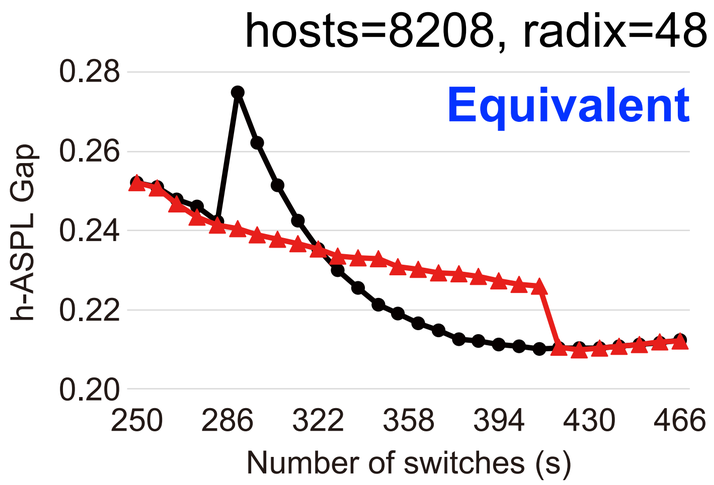

hosts=1281, radix=21
Equivalent
h-ASPL Gap
0.820
0.815
0.810
0.805
0.800
0.795
300 324 348 372 396 420 444
Number of switches (s)
Better

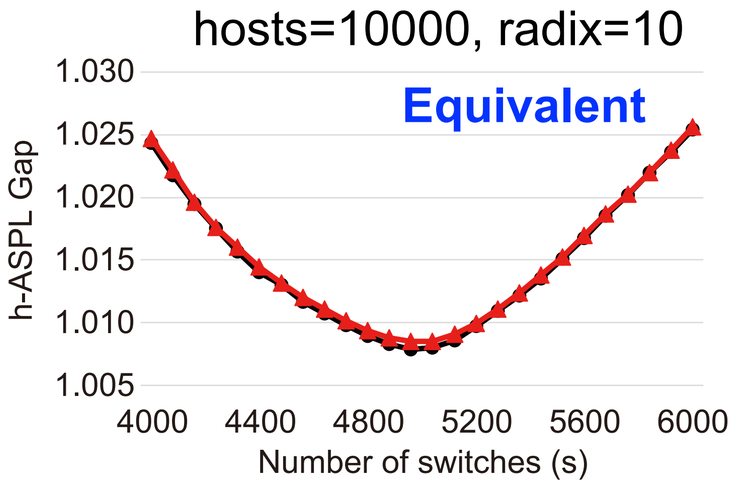
結果（2/2）

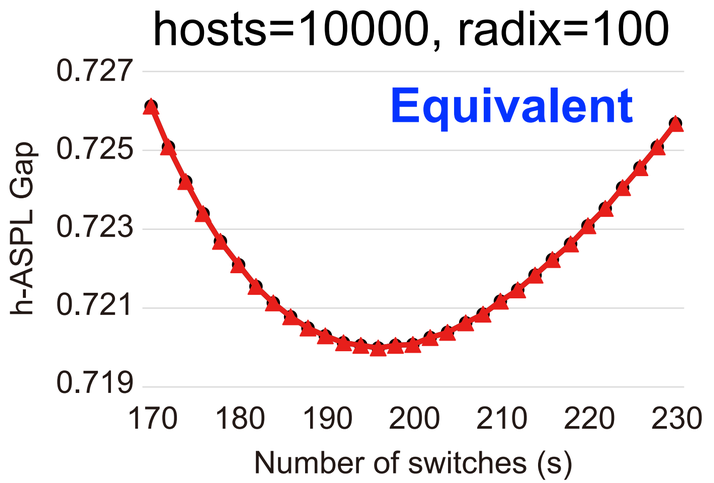

ランダム
バイアス

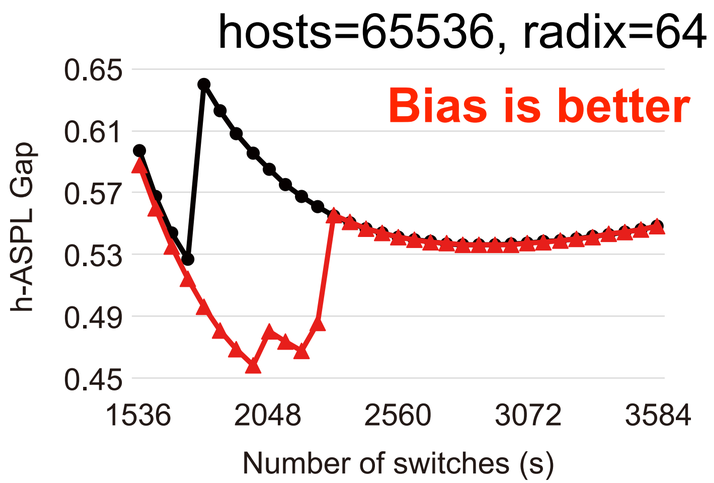

hosts=3800, radix=30
Bias is better
h-ASPL Gap
0.40
0.36
0.32
0.28
0.24
250
270
290
310
330
350
Number of switches (s)


hosts=4608, radix=36
Bias is better
h-ASPL Gap
0.7
0.6
0.5
0.4
0.3
0.2
150
190
230
270
310
350
Number of switches (s)


hosts=8208, radix=48
Equivalent
h-ASPL Gap
0.28
0.26
0.24
0.22
0.20
250
286
322
358
394
430
466
Number of switches (s)


hosts=10000, radix=10
Equivalent
h-ASPL Gap
1.030
1.025
1.020
1.015
1.010
1.005
4000
4400
4800
5200
5600
6000
Number of switches (s)

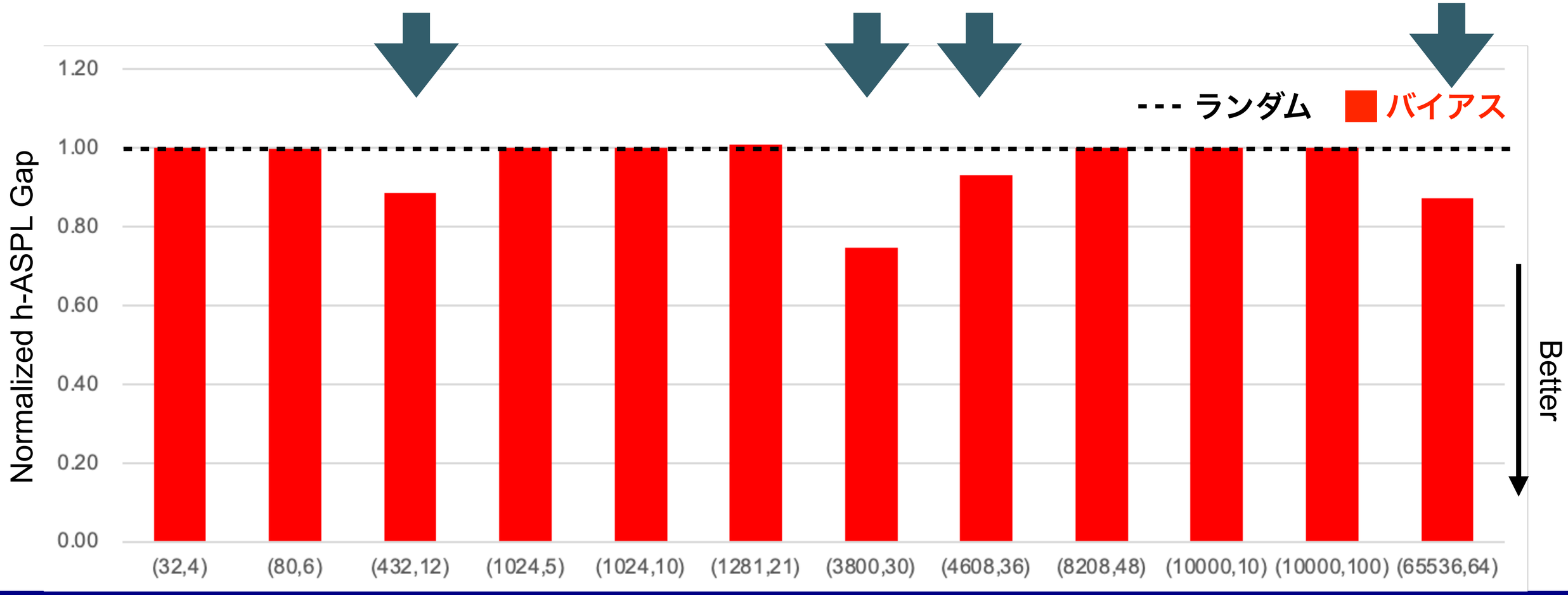

hosts=10000, radix=100
Equivalent
h-ASPL Gap
0.727
0.725
0.723
0.721
0.719
170
180
190
200
210
220
230
Number of switches (s)


hosts=65536, radix=64
Bias is better
h-ASPL Gap
0.65
0.61
0.57
0.53
0.49
0.45
1536
2048
2560
3072
3584
Number of switches (s)

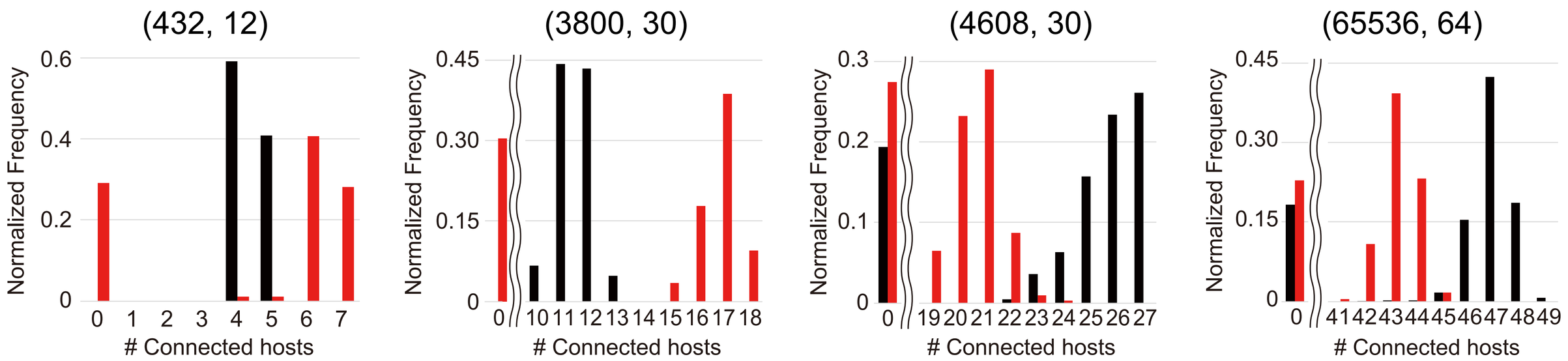

Better

結果まとめ

- **ランダム選択**のh-ASPL Gapsを1.0で正規化
- **バイアス選択**はランダム選択と同等以上の結果



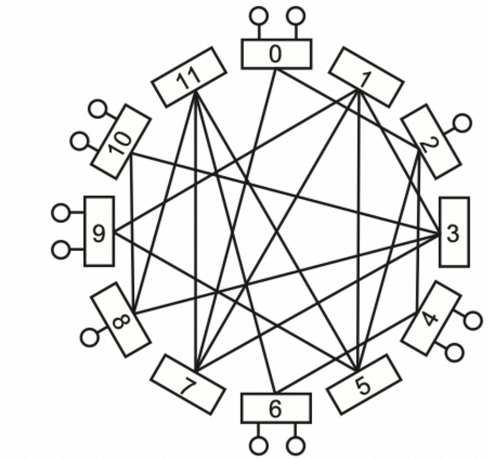
各スイッチにおける隣接ホスト数の分布



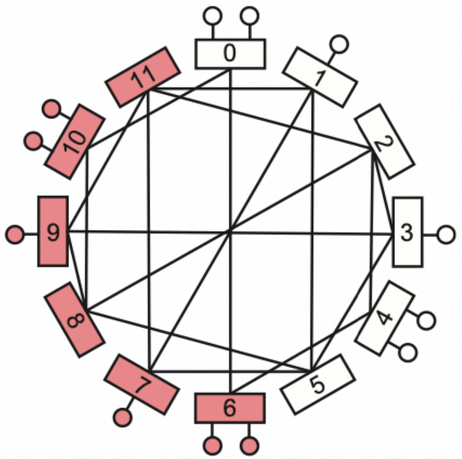
- 隣接ホスト数が0ということは、スイッチとしか隣接しないことを意味する
- (432, 12)と(3800, 30)では、バイアス選択の場合のみ，隣接ホスト数が0のスイッチが存在
- (4608, 30)と(65536, 64)でも、隣接ホスト数が0のスイッチ数はバイアス選択の方が多い
- **この結果より，ホストと隣接しないスイッチが存在する方が、h-ASPLが小さくなる場合があることを確かめた**

対称性

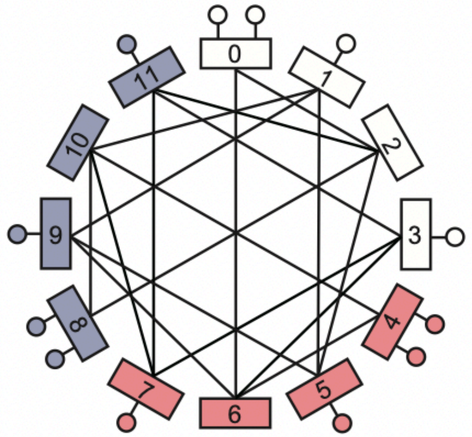
- 任意のエッジeをf(e)にマップする自己同型が存在する場合に、そのグラフは対称性を持つと定義
- 対称性を持つグラフの例 (hosts=12, radix=4, switches=12)



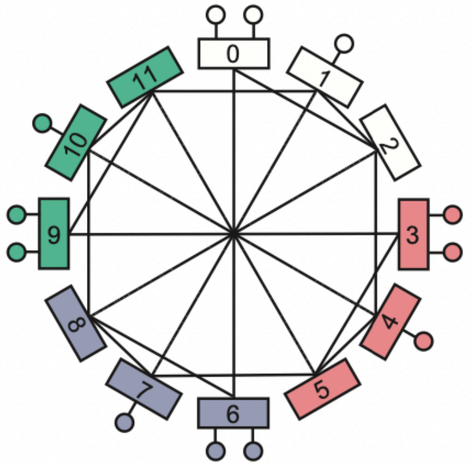
g=1



g=2



g=3



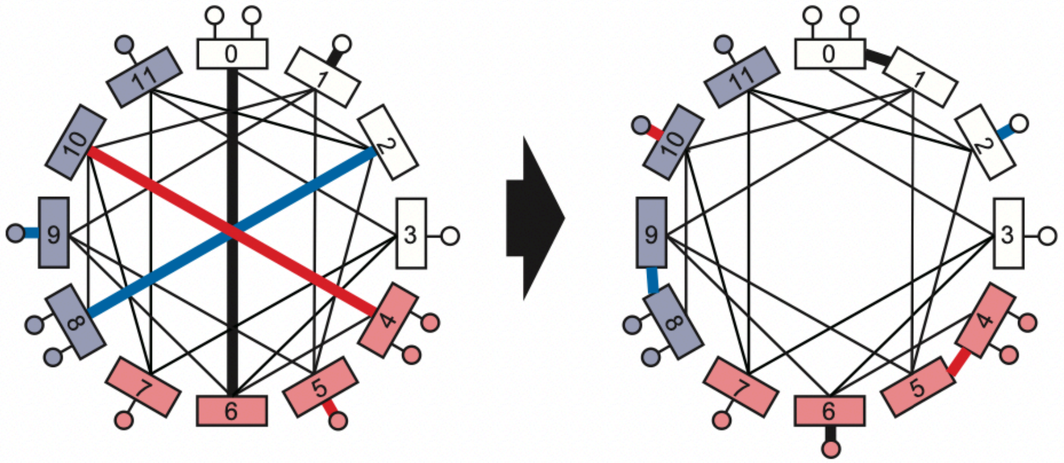
g=4

- 360/g 度回転させると、頂点（ホストとスイッチ）とエッジの関係が同じになる
- 変数 g はホスト数とスイッチ数の公約数
- g=1の場合は対称性を持たない

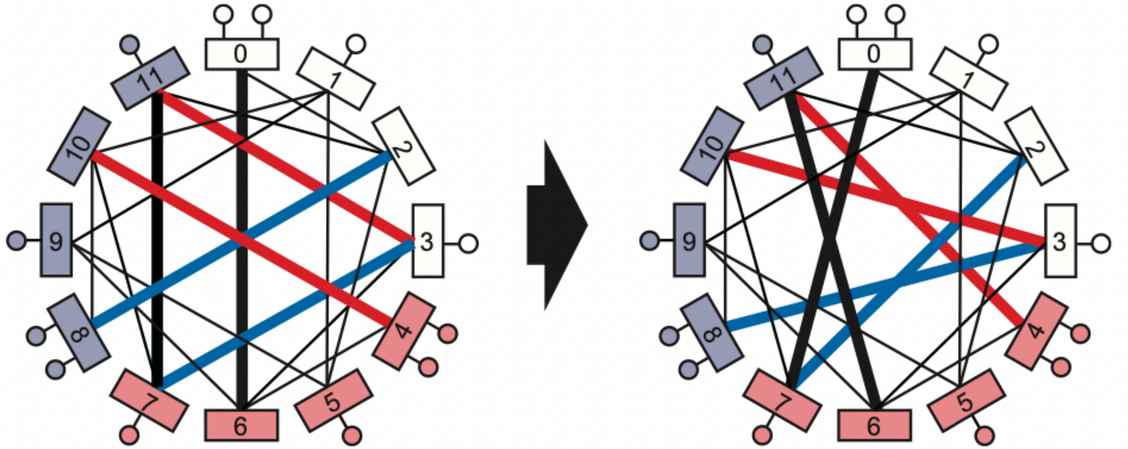
対称性を維持しながらグラフを最適化する

- 下記のグラフはhosts=12, radix=4, switches=12, **g=3**
- 初期解は対称性を持つグラフを生成する
- 対称的な関係を持つすべての頂点とエッジに対して、**SWING**と**SWAP**を実行する

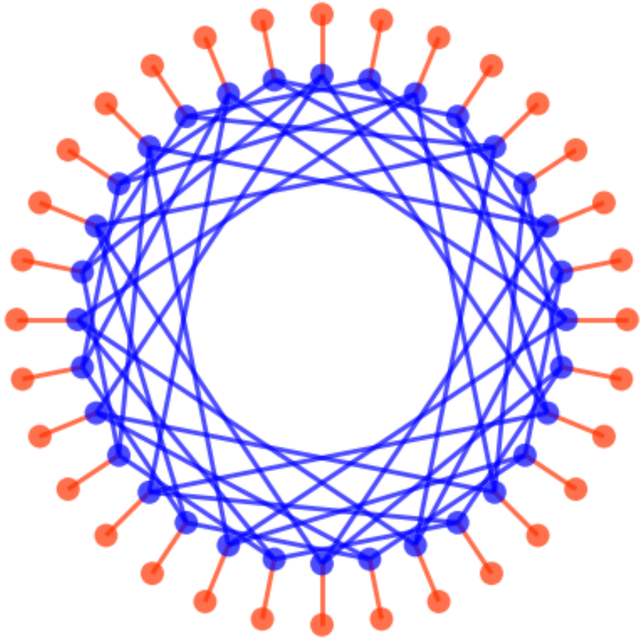
- **SWING**



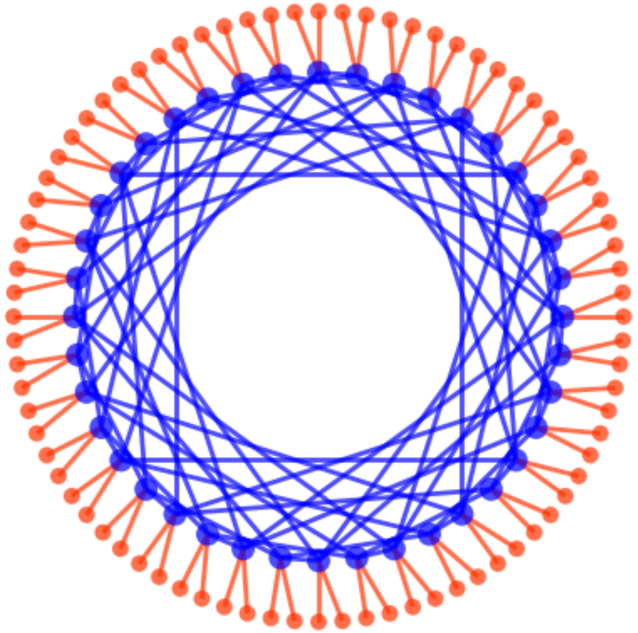
- **SWAP**



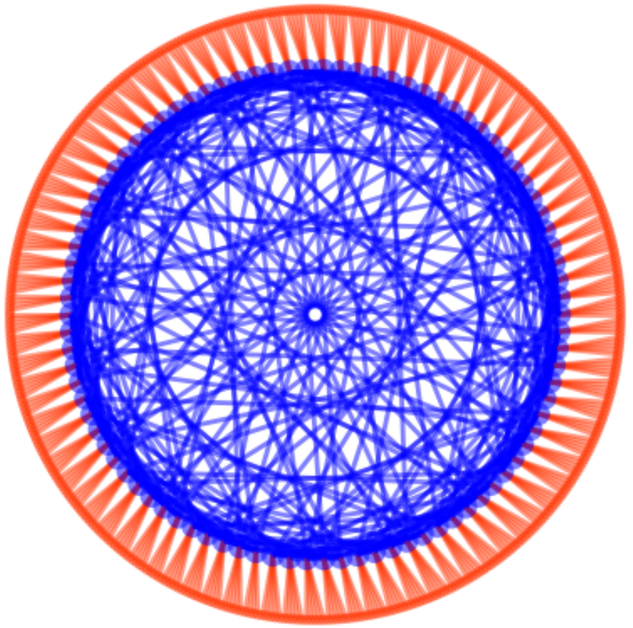
結果の例



hosts = 32
radix = 4
switches = 32
g = 16



hosts = 80
radix = 6
switches = 40
g = 20

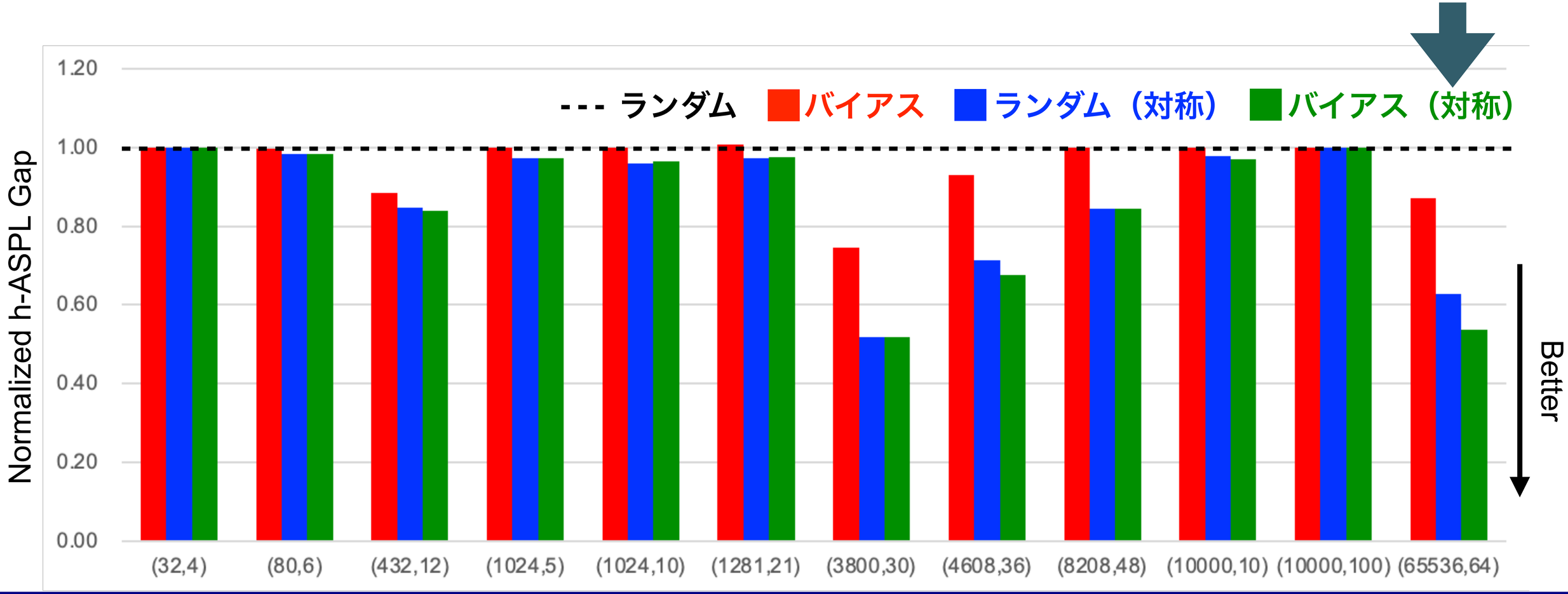


hosts = 432
radix = 12
switches = 90
g = 18

結果まとめ

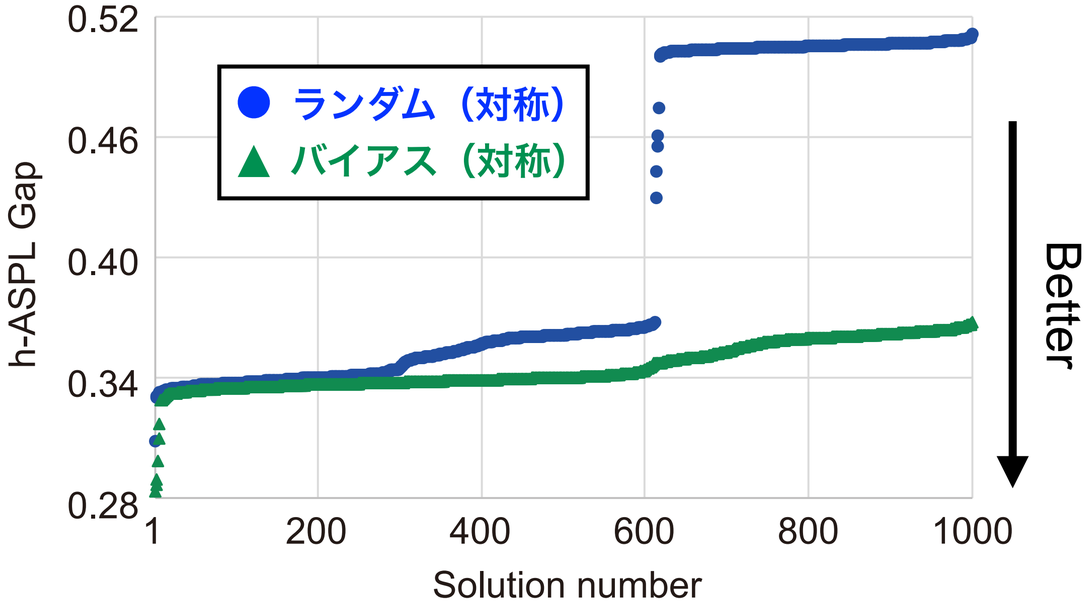
- **g=1の場合のランダム選択**のh-ASPL Gapsを1.0で正規化
- **バイアス選択（対称）**が最も良い結果

大きな差があったのは
(65536, 64)のみ



(65536, 64)について

- 1000試行の結果を昇順でソート
- **バイアス選択（対称）**の方が安定



hosts = 65536
radix = 64
switches = 3072
g = 1024

ホスト分布について調べたところ、**バイアス選択（対称）**のすべての結果はホスト数が0のスイッチを持つが、
ランダム選択（対称）のh-ASPL Gapが悪い結果はホスト数が0のスイッチは持たなかった。

以上の結果より、バイアス選択と対称性の工夫を組合せると、良い結果が得られることがわかった

もくじ

- 背景：Order/Radix Problemとは
  - 既存研究[R. Yasudo, 2019]
  - 国際コンペティションGraphGolf
- 目的：Order/Radix Problemに対する最適化アルゴリズムの開発
- 提案：既存のアルゴリズムに対して2つの工夫を追加する
  - ホストの偏り
  - 対称性
- 評価：既存研究との比較
- 考察：NAS Parallel Benchmarkを用いた性能評価
- まとめ

SimGridによる並列アプリケーションのシミュレーション

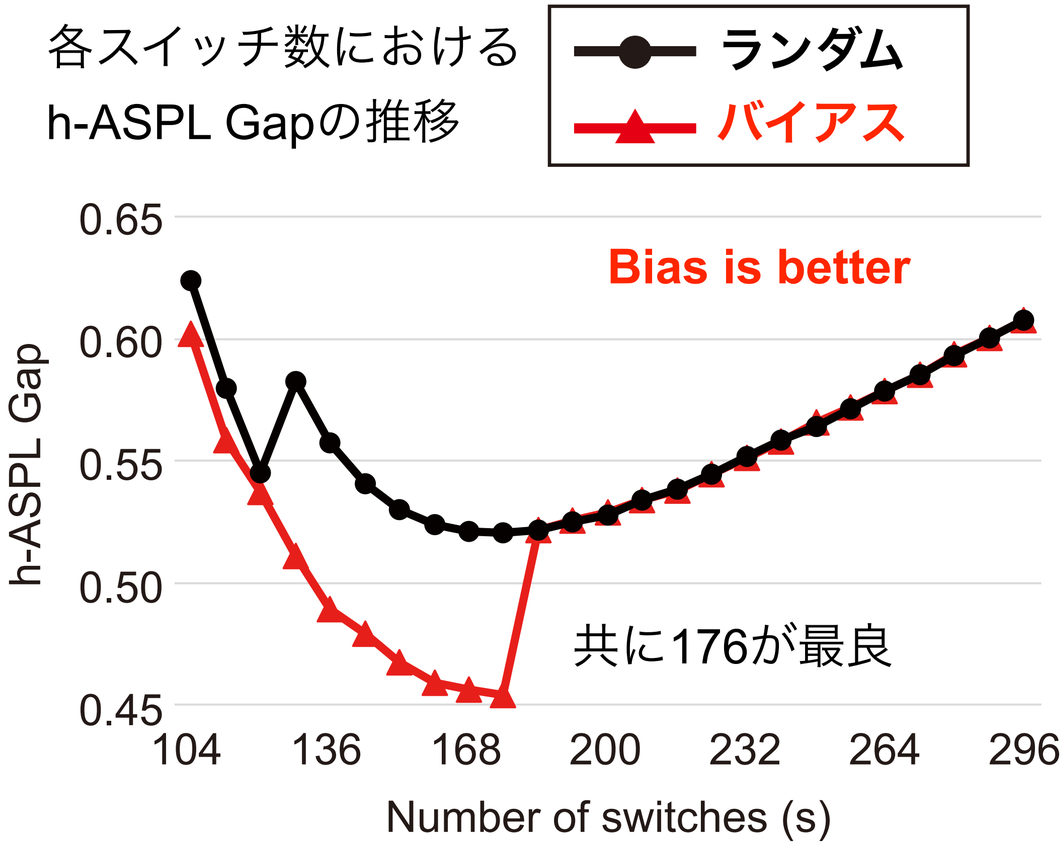
- 過去の研究により、h-ASPLの小さいトポロジは様々な並列アプリケーションの性能が向上する
- バイアス選択によって生成されるトポロジはランダム選択よりもh-ASPLは小さい
- しかし、バイアス選択によって生成されたトポロジは、ホストを持たないスイッチを多く持つ
  - そのようなトポロジは、ランダム選択よりも良いトポロジなのか？

- SimGridを用いた評価を行う
  - 仮想的なプラットフォーム上で並列アプリケーションを実行できる
  - 提案した最適化アルゴリズムの出力グラフを元に仮想プラットフォームを作成
  - 並列ベンチマーク集としてNAS Parallel Benchmark（NPB）を用いる

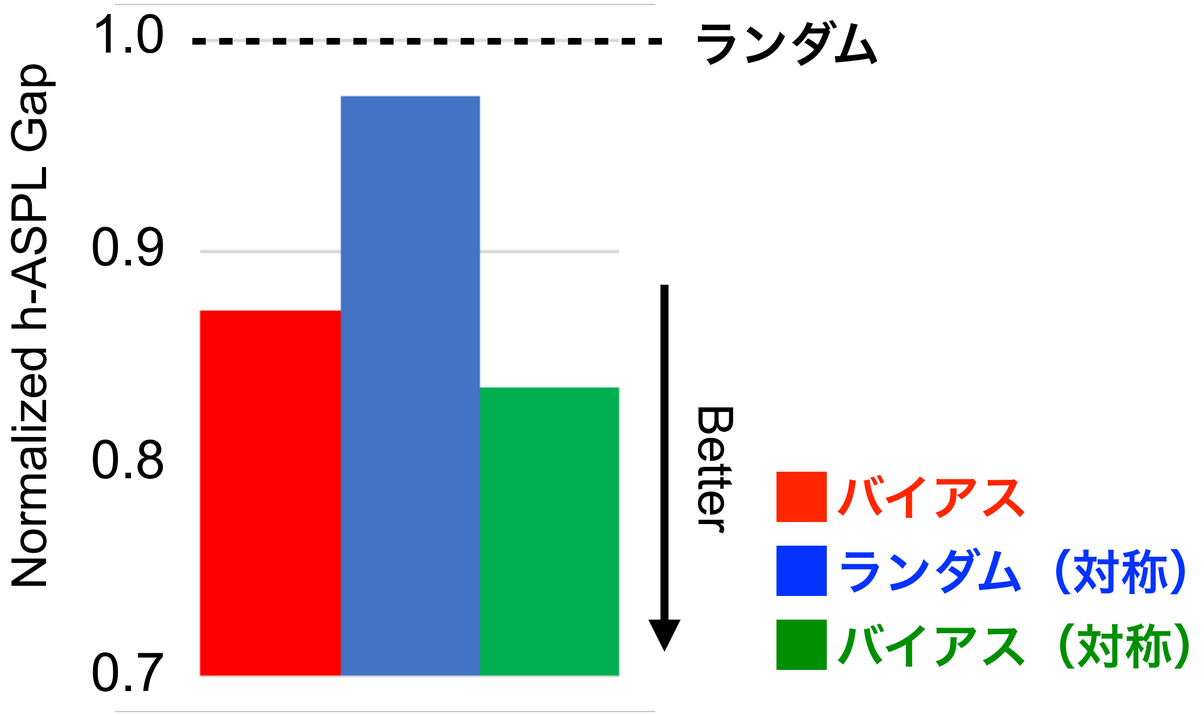
SimGridによる並列アプリケーションのシミュレーション

- (hosts, radix) = (1024, 16)：NPBの制限のため、ホスト数は4べきの値

各スイッチ数における
h-ASPL Gapの推移

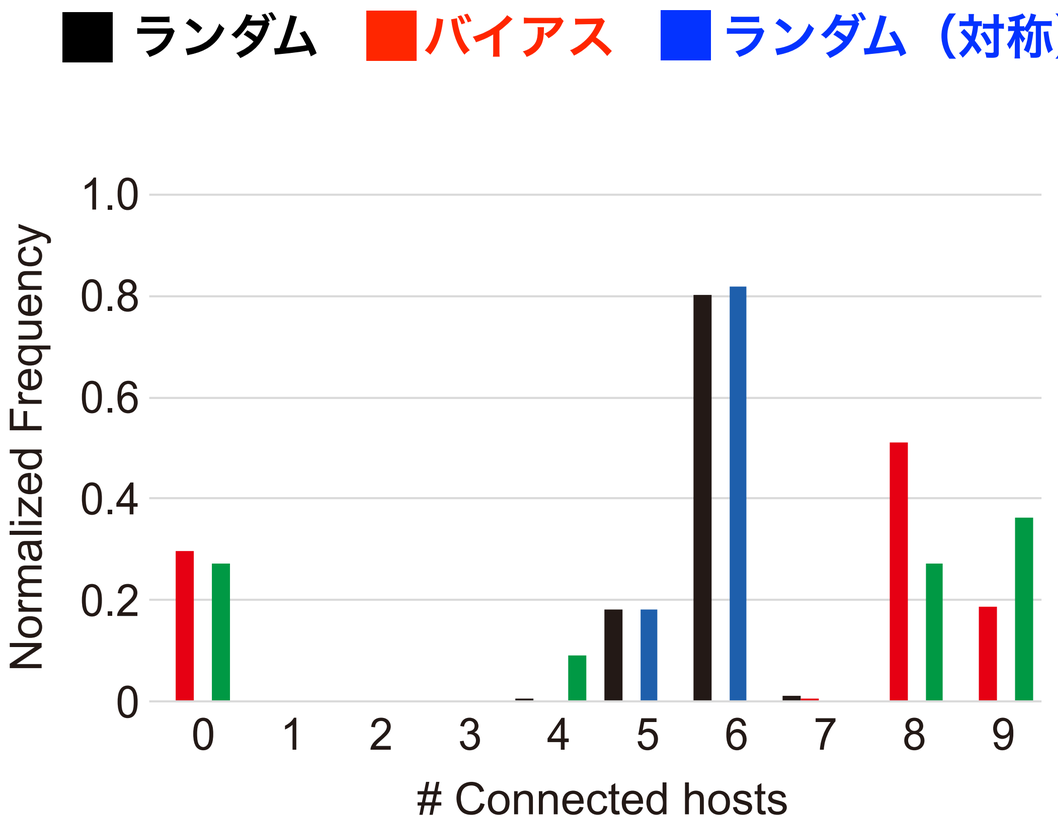


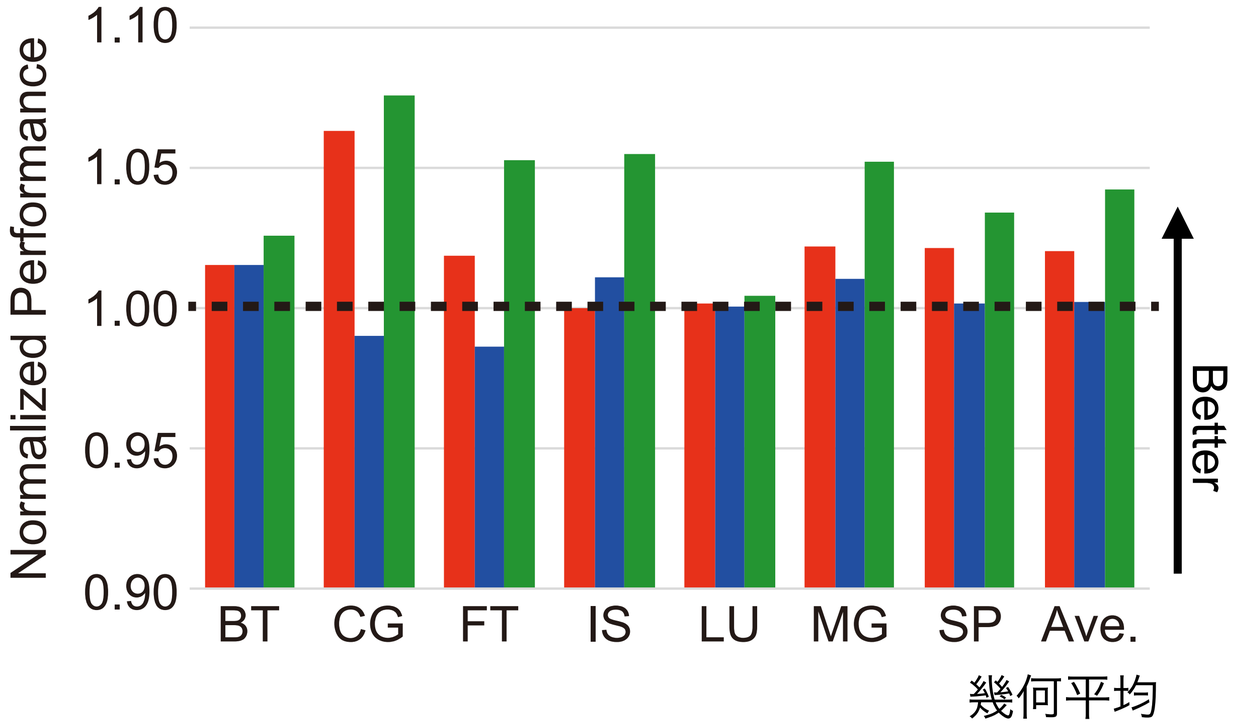
スイッチ数が176、g=16のときのh-ASPL Gapの比較



SimGridによる並列アプリケーションのシミュレーション

- ホスト数の分布：バイアスの場合のみ
  ホスト数が0のスイッチがある
- NPBの結果：**バイアス（対称）**が最も良い





まとめ

- ORPのための最適化アルゴリズムの提案
- h-ASPLを高速に計算するためのライブラリの作成
- 既存のアルゴリズムに2つの工夫を追加
  - ホストを偏らせる
  - 対称性を与える
- 既存アルゴリズムよりもh-ASPLが小さいグラフを生成できることがわかった
- SimGridとNPBを用いてネットワークトポロジの評価を行った結果、提案アルゴリズムから生成されたネットワークトポロジは、既存アルゴリズムよりも良い結果を得ることができた

- 今後の課題
  - スイッチ数の自動調整機能の追加