

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^{1,a)} 塚本 雅生² 花田 良子² 山本 啓二¹

概要：並列計算機システムにおいて、平均ホスト間距離（h-ASPL：host-Average Shortest Path Length）が小さい間接網を導入することで、システム全体の性能を向上させることが期待できる。そのような間接網をグラフ理論上で議論するため、Order/Radix Problem (ORP) が提案されている。ORP とは、与えられたホスト数と Radix を満たす最小の h-ASPL を持つグラフを発見する問題である。ORP のグラフは間接網を表しており、頂点はホストとスイッチの 2 種類が存在し、スイッチはホストもしくは他のスイッチと隣接でき、ホストはスイッチとのみ隣接できる。本稿では、ORP のための最適化アルゴリズムの提案を行う。提案アルゴリズムの特徴として、グラフに対称性を持たせる点、各スイッチによって隣接するホストを偏らせる点が挙げられる。これらの特徴により、h-ASPL を求めるための計算時間を削減させ、かつ提案アルゴリズムの解探索性能を向上させることができる。ORP のための国際コンペティション Graph Golf が出題している問題を用いて提案アルゴリズムの性能評価を行った。その結果、提案アルゴリズムは既存アルゴリズムと比較して h-ASPL が小さいグラフを生成できることを示した。さらに、提案アルゴリズムが生成するグラフの性能を評価するため、並列シミュレーションフレームワーク SimGrid と並列ベンチマーク集 NAS Parallel Benchmarks による評価を行った。その結果、提案アルゴリズムが生成するグラフは、既存アルゴリズムが生成するグラフよりも高い性能を発揮できることを示した。

1. はじめに

並列計算機システムのネットワークポロジは、並列アプリケーションの性能に大きく影響を与えることが知られている。ここで、並列計算機システムの計算ノードを頂点、ネットワークケーブルをエッジとみなすことで、そのネットワークポロジは無向グラフとして表現できる。このことから、グラフ理論に基づくネットワークポロジの研究が活発に行われている [1–4]。例えば、ランダムトポロジの平均頂点間距離（ASPL：Average Shortest Path Length）は従来の k -ary n -cube などの規則的なトポロジの ASPL よりも小さくなるため、ランダムトポロジをネットワークポロジに採用することで、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 [4]。

ランダムトポロジの ASPL はその理論的な下界よりも大きな値をとるため [1]、より小さい ASPL を持つグラフを発見することが重要である。そのような背景から、与えられた頂点数と次数を満たす最小の ASPL を持つグラフを発見する問題である Order/Degree Problem (ODP) が注目さ

れている [5]。しかしながら、ODP におけるグラフの頂点は 1 種類のみであるため、ODP は直接網を表現するのに適しているが、間接網を表現することはできない。そこで、間接網を扱うために ODP を拡張した Order/Radix Problem (ORP) が提案されている [6]。ORP は与えられたホスト数と Radix を満たす最小の平均ホスト間距離（h-ASPL：host-ASPL）を持つグラフを発見する問題であり、ホストとスイッチという 2 種類の頂点が存在する。h-ASPL が小さいネットワークポロジは、既存の規則的なネットワークポロジと比較して、様々な並列アプリケーションの性能を向上させることが報告されている [6]。

ORP の定義は簡易であるが、その効率的な解法は次の理由により発見されていない。(1) h-ASPL を求めるための計算量が大きいこと、最適化に時間を要すること。(2) 与えられたホスト数と Radix を満たすグラフの数は膨大であり、かつ多数の局所解を持つような複雑な適応度景観を持つため、最適化が困難である。

そこで、本稿では Simulated Annealing (SA) [7, 8] をベースとした ORP のための最適化アルゴリズムを提案する。提案アルゴリズムの特徴は、グラフに対して対称性を与えることで、h-ASPL を求めるための計算量を削減し、かつ SA の解探索能力を高めることである。さらに、各ス

¹ 理化学研究所 計算科学研究センター 兵庫県神戸市中央区港島南町 7-1-26

² 関西大学 大阪府吹田市山手町 3-3-35

^{a)} masahiro.nakao@riken.jp

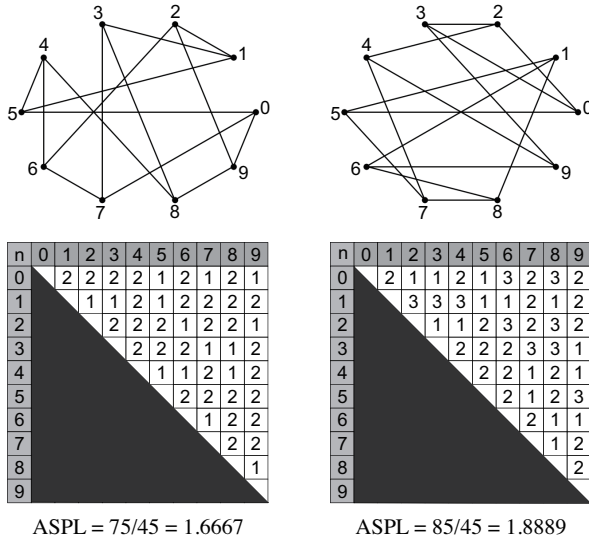


図 1: Examples of graph $(n, d) = (10, 3)$ [9]

スイッチにおいて隣接するホストが偏るように解探索を進めることにより、従来の方法と比較して解探索性能を向上させることができる。

本稿の構成は下記の通りである。2章では、ORP の関連研究について述べる。3章では、ORP のための最適化アルゴリズムを提案する。4章では、h-ASPL の計算に用いるアルゴリズムについて述べ、その速度について評価する。5章では、提案アルゴリズムの解探索性能を評価する。6章では、提案アルゴリズムが生成したグラフについて考察する。7章では、まとめと今後の課題を示す。

2. 関連研究

本章では、ODP とその関連研究について説明した後、その拡張である ORP とその関連研究について説明する。

2.1 Order/Degree Problem

ODP は与えられた頂点数 (n) と次数 (d) を満たす最小の ASPL を持つグラフを発見する問題である。本稿では、このグラフを“ODP グラフ”と呼ぶ。 $(n, d) = (10, 3)$ の ODP グラフの例とその距離行列を図 1 に示す。ASPL は距離行列の全要素の合計値を要素数 $(n^2 - n)/2$ で割った値である。この例では、より小さい ASPL を持つ左の方が良いグラフである。ASPL の理論的な下界は [10] で示されているが、下界を満たすグラフの存在は保証されない。

ODP の既存研究で用いられているアルゴリズムは、次の 2 つに分類できる。(1) 頂点数が小さい複数のグラフ同士を組合せる [11, 12]。(2) 最適化アルゴリズムを用いる [13, 14]。(1) は ASPL の計算回数が (2) と比べて少ないという利点を持つ反面、特定の頂点数を持つグラフしか作れないという問題点がある。[11] では、2 種類のグラフを掛け合わせて新しいグラフを構成する Multiple star product という手法を提案している。[12] では、複数の Petersen グ

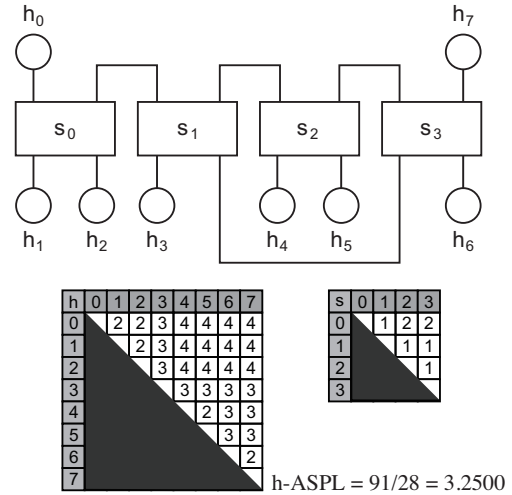


図 2: Example of graph $(h, s, r) = (8, 4, 4)$

ラフを組合せることで、直径 3 のグラフを構成する手法を提案している。(2) は (1) とは逆であり、任意の頂点数のグラフを作成できる利点を持つ反面、ASPL の計算を繰り返す必要があるため、その計算量が大きいという問題点がある。[13] は SA をベースとしており、直径 3 のグラフに限定しているが、ASPL を少ない計算量で近似する手法を開発している。我々が開発した最適化アルゴリズム [14] は、任意の直径のグラフを扱うことができる。また、グラフに対称性を与えることにより、ASPL を求める計算量の削減と SA の解探索性能の向上の両方を実現している。

2.2 Order/Radix Problem

ORP は与えられたホスト数 (h) と Radix (r) を満たす最小の h-ASPL を持つグラフを発見する問題である。本稿では、このグラフを“ホストスイッチグラフ”と呼ぶ [6]。ホストスイッチグラフは、ホスト数 $\cdot$ スイッチ数 $(s) \cdot$ Radix の 3 要素から構成される。ORP において、スイッチ数は任意の値を用いることができる。 $(h, s, r) = (8, 4, 4)$ のホストスイッチグラフの例を図 2 に示す。図では、 $\square$ と $\circ$ はそれぞれスイッチとホストを表している。ホストはスイッチのみと隣接できるのに対し、スイッチはホストもしくは他のスイッチと隣接できる。また、スイッチと隣接できるホスト数とスイッチ数の合計の最大数は r である。このように、ホストスイッチグラフは、1 ポートを持つホストと r ポートを持つスイッチングハブで構成された間接網のネットワークトポロジを表している。

図 2 では、ホスト間の距離行列を図左下、スイッチ間の距離行列を図右下に示している。h-ASPL はホスト間の距離行列の全要素の合計値を要素数 $(h^2 - h)/2$ で割った値である。ただし、h-ASPL はスイッチ間の距離行列と、各スイッチと隣接するホスト数からも求めることができる。スイッチ s_i と s_j 間の距離を $l(s_i, s_j)$ 、スイッチ s_i と隣接するホスト数を w_i とした場合、h-ASPL は

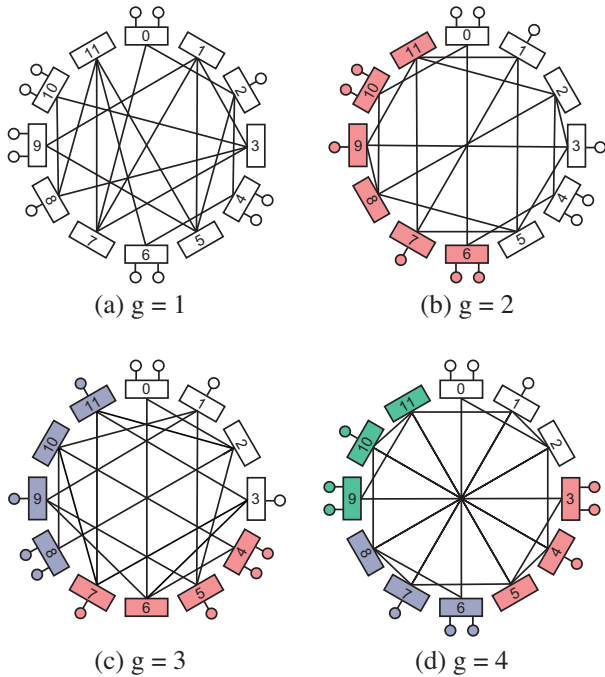


図 3: Examples of graph with symmetry $(h, s, r) = (12, 12, 4)$

$(\sum_{0 \leq i < j < s} (l(s_i, s_j) + 2) \cdot w_i w_j + \sum_{0 \leq i < s} w_i (w_i - 1)) / \binom{h}{2}$ である. h-ASPL の理論的な下界は [6] で示されているが, 下界を満たすホストスイッチグラフの存在は保証されない.

ORP の既存研究は [6] のみであり, SA をベースにした最適化アルゴリズムが提案されている. 5 章において, [6] と我々の提案する最適化アルゴリズムとの比較を行う.

2.3 Graph Golf

ODP と ORP を題材にした国際コンペティション Graph Golf が国立情報学研究所の主催で開催されている [15]. Graph Golf は年ごとに異なる問題 (ODP の場合は頂点数と次数の組合せ, ORP の場合はホスト数と Radix の組合せ) が出題される. Graph Golf が開始された 2015 年は ODP のみの出題だったが, 2021 年では新規に ORP も出題された. 2021 年の Graph Golf のスケジュールは次の通りであった.

4 月 27 日 ODP および ORP の問題の公開.

7 月 26 日～10 月 11 日 参加者は公式サイト Web フォームから自身が発見したグラフを投稿可能. この期間は一週間毎にランキングが更新される.

10 月 18 日 最終ランキングの発表.

3. 提案アルゴリズム

3.1 概要

2.1 節で述べた通り, 我々が開発した ODP のための最適化アルゴリズム [14] の特徴は, グラフに対称性を与えることにより, ASPL を求める計算量の削減と SA の解探索性

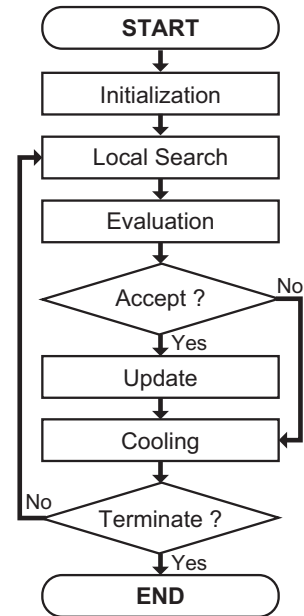


図 4: Basic flow of simulated annealing

能の向上を達成していることである. このことから, ORP においても同様の工夫を行うことで, 同じ効果を発揮できると考えられる. 本章では, [6] で提案されている SA に対してグラフに対称性を与える最適化アルゴリズムを提案する. また, さらなる工夫として, 各スイッチにおいて隣接するホストが偏るように解探索を行う手法についても提案する.

本稿では, ホストスイッチグラフにおいて任意のエッジ e を $f(e)$ にマップする自己同型 [16] が存在する場合に, そのグラフは対称性があると定義する. 対称性があるホストスイッチグラフの例を図 3 に示す. 図中の変数 g は対称性の数である. 例えば図 3-(d) の場合, エッジ 0-2 は, エッジ 3-5, 6-8, 9-11 にマップできる. なお, 図 3 において, 各ホストスイッチグラフを平面としてみた場合, $360/g$ 度回転させると頂点とエッジの隣接関係が同じグラフになる. このことから, g の取りうる値はホスト数とスイッチ数の公約数であることがわかる. $g = 1$ の場合は, 対称性を持たない通常のホストスイッチグラフである.

次に, 対称性を利用した h-ASPL の計算量の削減について述べる. 2.2 節で示した通り, h-ASPL の計算には全スイッチ間の距離が必要である. しかしながら, 例えば図 3-(d) の場合, スイッチ 0 から他のスイッチまでの距離は, スイッチ 3・6・9 から他のスイッチまでの距離と同じである. すなわち, スイッチ 0~2 から他のスイッチまでの距離を計算するのみで, 全スイッチ間の距離を計算することが可能である. このことから, 対称性を持つホストスイッチグラフの h-ASPL の計算量は, 対称性を持たない場合の計算量の $1/g$ になることがわかる. この特性を利用した h-ASPL の計算速度については, 4 章で評価を行う.

以降の節では, 図 4 に示す SA の基本的なフローを用

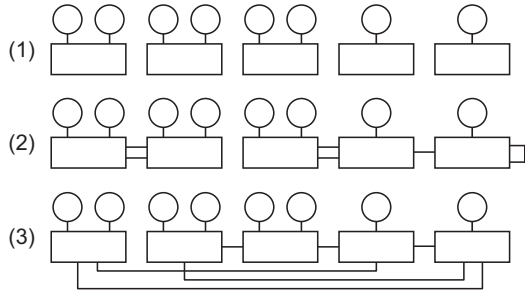


図 5: Initialization of graph $(h, s, r, g) = (8, 5, 4, 1)$

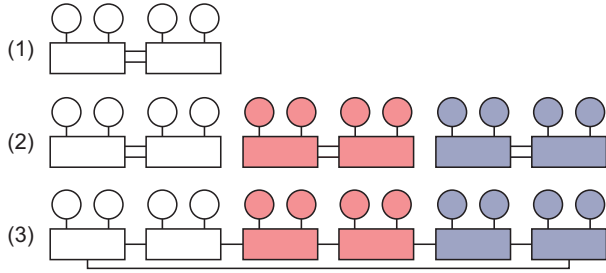


図 6: Initialization of graph $(h, s, r, g) = (12, 6, 4, 3)$

いて、我々が開発した最適化アルゴリズムの実装について説明する。各節のタイトルは、フローの各項目と対応している。

3.2 Initialization

ユーザから与えられるパラメータであるホスト数 (h)・スイッチ数 (s)・Radix (r)・対称性の数 (g)を満たすランダムなホストスイッチグラフを生成する。

まず対称性がない場合について、 $(h, s, r, g) = (8, 5, 4, 1)$ のホストスイッチグラフを用いて説明する (図 5)。(1) 各スイッチになるべく均等にホストを割り当てる。(2) 各スイッチの余ったポートを用いて隣のスイッチと接続する。接続可能なスイッチがない場合は自己ループを作成する。(3) 3.3 節で述べる Swap を適当な回数繰り返す。

次に対称性がある場合について、 $(h, s, r, g) = (12, 6, 4, 3)$ のホストスイッチグラフを用いて説明する (図 6)。(1) 対称性がない場合の方法を用いて、 $(h/g, s/g, r, 1) = (4, 2, 4, 1)$ のランダムなホストスイッチグラフを作成する。(2) (1) で作成したグラフを g 個複製する。(3) 3.3 節で述べる Swap を適当な回数繰り返す。

3.3 Local Search

ローカルサーチを行うため、Swing と Swap という操作を定義する。Swing はスイッチと隣接するホスト数を変化させる操作であり、Swap はスイッチ間接続を変化させる操作である。

まず対称性がない場合のローカルサーチについて説明する。この手法は [6] と同じである。Swing は、エッジ $\{s_a, s_b\}, \{s_c, h\}$ を $\{s_a, s_c\}, \{s_b, h\}$ に変える (図 7)。この操作

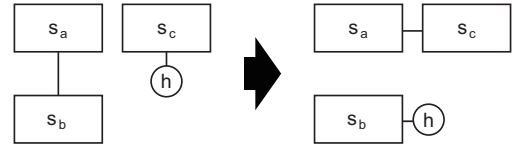


図 7: Swing operation [6]

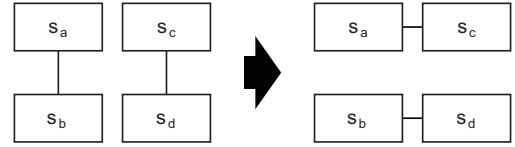


図 8: Swap operation [6]

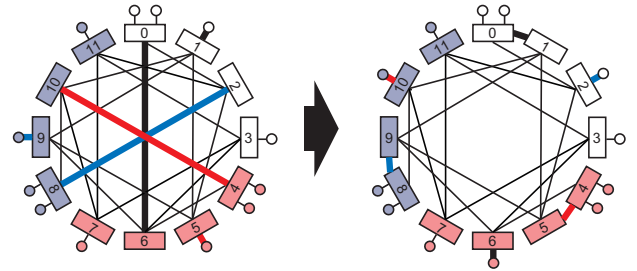


図 9: Swing operation with symmetry

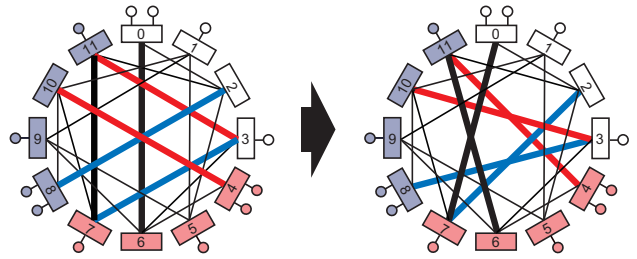


図 10: Swap operation with symmetry

を $SWING(s_a, s_b, s_c)$ とする。Swap は、エッジ $\{s_a, s_b\}, \{s_c, s_d\}$ を $\{s_a, s_c\}, \{s_b, s_d\}$ に変える (図 8)。この操作を $SWAP(s_a, s_b, s_c, s_d)$ とする。Swing と Swap を用いたローカルサーチは下記の手順で行う。(1) ランダムに 2 本のエッジ $\{s_a, s_b\}, \{s_c, s_d\}$ を選択し、 $SWING(s_a, s_b, s_c)$ を行う。(2) (1) で得た解を評価し、受理された場合は終了する。評価と受理については、3.4 節と 3.5 節で説明する。(3) (1) で選択したエッジに対し、 $SWAP(s_a, s_b, s_c, s_d)$ を行う。(4) (3) で得た解を評価し、受理された場合は終了する。

次に対称性がある場合のローカルサーチについて、 $(h, s, r, g) = (12, 12, 4, 3)$ のホストスイッチグラフを用いて説明する。まず、Swing について説明する (図 9)。(1) エッジ $\{0, 6\}, \{1, h\}$ を選択したとする。そして、そのエッジと対称の関係があるエッジ $\{4, 10\}, \{5, h\}$ と $\{8, 2\}, \{9, h\}$ を選択する。(2) それぞれの組について Swing を行う。次に、Swap について説明する (図 10)。(1) エッジ $\{0, 6\}, \{7, 11\}$ を選択したとする。そして、そのエッジと対称の関係があるエッジ $\{4, 10\}, \{11, 3\}$ と $\{8, 2\}, \{3, 7\}$ を選択

する。(2) それぞれの組について Swap を行う。このような手順でグラフを変化させることで、対称性を保ったままローカルサーチを行うことができる。

ここで、頂点の選択方法に着目する。一般的な最適化アルゴリズムでは、すべての変数（頂点）を均等な確率で変化させる。しかしながら、例えば Fat-Tree トポロジにおけるルートスイッチのように、ホストと隣接しないスイッチ（スイッチとのみ隣接するスイッチ）が存在する方がよいトポロジである可能性がある。そこで、各スイッチにおいて隣接するホストが偏るように頂点を選択する手法を提案する。本稿では、この偏りがある選択方法を“バイアス選択”と呼び、均等な確率で頂点を選択する従来の方法を“ランダム選択”と呼ぶ。

バイアス選択の手順は下記の通りである。(1) ランダム選択と同様に 2 本のエッジ $\{s_a, s_b\}, \{s_c, s_d\}$ を選択する。ここで、各スイッチの隣接するスイッチ数を $n_i (i = a, b, c, d)$ とする。(2) $n_b/(n_a + n_b)$ の確率で、 s_a と s_b とを置換する。(3) $n_d/(n_c + n_d)$ の確率で、 s_c と s_d とを置換する。(1)～(3) の手順を行うことで、Swing において、 s_b は隣接するホスト数が多いスイッチが選ばれる確率が高くなり、 s_c は隣接するホスト数が少ないスイッチが選ばれる確率が高くなる。Swing では、 s_b は隣接するホスト数が 1 つ増え、 s_c は隣接するホスト数が 1 つ減るので、バイアス選択を行うことで、各スイッチにおいて隣接するホストが偏る可能性を高めることができる。

3.4 Evaluation

SA は新しいグラフ (G_{new}) と現在のグラフ (G_{cur}) が持つ評価値の差分 ΔE を必要とする。評価値には h-ASPL を用いる。 ΔE を導く式を次に示す。 $E(G)$ はあるグラフ (G) の h-ASPL を表している。

$$\Delta E = E(G_{new}) - E(G_{cur}) \quad (1)$$

ここで、2.2 節で説明したホスト間の距離行列と h-ASPL の定義により、 ΔE の最小値は $2g/(h^2 - h)$ である。すなわち、 ΔE の値は対称性の数とホスト数に依存することがわかる。ここで、温度パラメータの設定を簡易にするため、 ΔE に重みとして $(h^2 - h)/g$ を掛けることを考えた。この方法は、ODP に対する最適化アルゴリズムと同じである [14]。しかしながら、予備実験を行ったところ、この重みは大きすぎることがわかった。その理由は、 ΔE が最小値を取る場合は、変化が起きた 2 つのスイッチと隣接するホスト数が 1 つだけの場合であり、その可能性は非常に小さいからである。ここで、各スイッチが均等にホストと隣接する場合の ΔE の最小値は $2g/(h^2 - h) \times h^2/s^2$ であるため、1 と s^2/h^2 の相乗平均である s/h を $(h^2 - h)/g$ に掛けた $s(h-1)/g$ を重みとして用いることにした。最終的な

評価値の差分 $\Delta E'$ は下記の通りである。

$$\Delta E' = \Delta E \times s(h-1)/g \quad (2)$$

3.5 Accept

式 2 の $\Delta E'$ および温度 T と次の Metropolis 基準を用いて、3.3 節で作成した新しいグラフを受理するか否かの判定を行う。

$$Probability = \begin{cases} 1 & \text{if } \Delta E' < 0 \\ \exp(-\frac{\Delta E'}{T}) & \text{otherwise} \end{cases} \quad (3)$$

Metropolis 基準では、新しいグラフの h-ASPL が現在のグラフの h-ASPL よりも小さい場合は、新しいグラフを必ず受理する。その逆の場合は、新しいグラフを確率的に受理する。温度が高いほど、その受理確率は高くなる。

3.6 Update

3.5 節で受理の場合は、現在のグラフを新しいグラフに置換する。

3.7 Cooling

次の指数型アニーリングを用いて、温度 T を下げる処理を行う。 α はクーリング率であり、最高温度 T_{max} 、最低温度 T_{min} 、イテレーション回数 N とすると、 $\alpha = (T_{min}/T_{max})^{1/N}$ である。

$$T \leftarrow \alpha \times T \quad (4)$$

3.8 Terminate

図 4 における Evaluation を行った回数が、イテレーション回数 N に達したら、プログラムを終了する。なお、図 4 では省略しているが、3.4 節で h-ASPL を計算する際、新しいグラフの h-ASPL が下界である場合もプログラムを終了する。

4. h-ASPL の計算速度の評価

4.1 概要

ORP において最適化アルゴリズムを用いる場合、グラフの h-ASPL を繰り返し計算する必要があるため、h-ASPL の高速化は非常に重要である。[6] では幅優先探索 (BFS: Breadth First Search) を用いたアルゴリズムが利用されている。具体的には、全スイッチを始点として BFS を s 回実行することにより、h-APSP を求めている。我々は過去の研究において、ODP の ASPL を求めるための隣接リストを用いたアルゴリズムを提案している [17]。ここで、2.2 節で述べた通り、ORP の h-ASPL はスイッチ間の距離を元に計算可能である。すなわち、ODP の頂点を ORP のス

```

1 level, distance ← 0
2 elements ← ⌈switches/E⌉
3 A, B ← INITIALIZE(switches, elements)
4 while(true)
5   for i=0 ... switches-1
6     for n ∈ nbr[i]
7       for j=0 ... elements-1
8         B[i][j] ← B[i][j] | A[n][j]
9
10  distance ← distance + DISTANCE(B, level, h_degree)
11  level++
12
13  num ← 0
14  for i=0 ... switches-1
15    for j=0 ... elements-1
16      num ← num + POPCNT(B[i][j])
17  if (num = switches * switches) return
18
19  SWAP(A, B)

```

図 11: h-ASPL Algorithm with adjacency list

B ₀	nbr	B ₁	B ₂
1000000000	23	1011000000	1011100000
0100000000	5	0100010000	1100010000
0010000000	34	0011100000	1011100010
0001000000	029	1011000001	1011101001
0000100000	279	0010100101	0011101111
0000010000	01	1100010000	1111010000
0000001000	9	0000001001	0001101001
0000000100	48	0000100110	0010100111
0000000010	7	0000000110	0000100110
0000000001	346	0001101001	1011101101

図 12: Calculation to obtain h-ASPL

イチとみなすことで、ASPL のアルゴリズムを h-ASPL のアルゴリズムとして利用することが可能である。

本章では、隣接リストを用いた h-ASPL のアルゴリズムを紹介し、BFS を用いたアルゴリズムとの比較を行う。また、対称性の数と計算時間の関係について評価を行う。なお、本章で用いたプログラムは <https://github.com/mnakao/ORP> で公開している。

4.2 隣接リストを用いたアルゴリズム

提案アルゴリズムの擬似コードを図 11 に示す。1 行目では、あるスイッチからの距離を表す変数 *level* とスイッチ間の総距離を表す変数 *distance* を初期化している。2～3 行目では、訪問の有無をビットで表す行列 *A* と *B* を用意し、単位行列として初期化している。変数 *switches* はスイッチ数、定数 *E* は 1 要素のビット数（本実装では行列の型に C 言語の `uint64_t` を用いているので $E = 64$ ）、変数 *elements* は実装上の行列の列数である。5～8 行目では、図 12 に示す通り、隣接リスト *nbr* と *A* を用いた演算を行い、その結果を *B* に代入している。ここで、図 12 の *nbr* の 1 行目は、スイッチ 0 番はスイッチ番号 2 と 3 と隣接していることを表している。10～11 行目では、各スイッチと隣接しているホスト数を格納した配列 *h_degree* を用いて、新しく訪問済となったスイッチ間の総距離を計算して

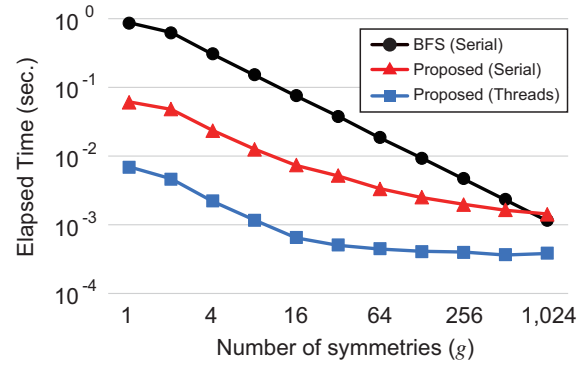


図 13: Speed performance result

いる。13～17 行目では、*B* の全要素が 1 であればプログラムを終了している。19 行目では、以上の動作を繰り返すため、*A* と *B* を置換している。また、図では省略しているが、高速化のために各 for 文と関数に対してスレッド並列化を行っている。

3.1 節で述べたグラフの対称性を利用することで、図 11 のアルゴリズムにおいても計算量の削減が可能である。具体的には、変数 *g* を対称性の数とした場合、図 11 の 2 行目の右辺を $\lceil (switches/g)/E \rceil$ に変更し、行列の初期化 (3 行目) を小さくなった行列 *B* に対応するなどを行う。この変更は ODP のアルゴリズム [17] と同様である。

4.3 評価

2021 年度の Graph Golf で出題されたホストスイッチグラフ $(h, r) = (65536, 64)$ を用いて、提案アルゴリズムの性能を評価する。スイッチ数は 4096 とした。評価機には、Intel Xeon Gold 6126 (2.6GHz, 12 コア) × 2 と DDR4 (192GiB, 256GB/s) を搭載した筑波大学計算科学研究センターの Cygnus システムのサーバを用いた。コンパイラは intel/19.0.5、オプションは `-O3 -std=gnu99 -Wno-unknown-pragmas -mavx2` である。逐次版の BFS、逐次版およびスレッド並列化版の提案アルゴリズムの結果を図 13 に示す。縦軸は 1 回の h-ASPL の計算に必要な時間、横軸は対称性の数 (*g*) である。スレッド並列化版に用いたスレッド数は 24 である。

対称性がない場合 ($g = 1$)、逐次版の提案アルゴリズムは BFS よりも 14.25 倍高速であった。しかしながら、*g* が増えるにつれ、それらの速度差は小さくなり、 $g = 1024$ の場合は BFS の方が高速であった。提案アルゴリズムは *E* 個のスイッチを出発点とするスイッチ間距離をまとめて計算している。しかしながら、図 11 の 2 行目の *elements* の値は、本実験で用いたホストスイッチグラフの場合 $\lceil (4096/g)/64 \rceil = \lceil 64/g \rceil$ であるので、*g* が 64 を越えると 5～8 行目などの計算量は変化しなくなる。言い換えると、利用しないビットを含めた余分な計算が生じてしまう。それに対して、BFS は 1 つのスイッチごとにスイッチ間距離を計算するた

め、 g にほぼ比例して計算時間を削減できる。これらのことから、 g が大きい場合は、BFS の方が性能が高くなったと考えられる。なお、提案アルゴリズムにおいて、 g が 64 を越えてもわずかに性能向上している理由は、行列の初期化（3 行目）などが高速化されたからである。

スレッド並列版の提案アルゴリズムの性能は逐次版の 3.56～11.34 倍であった。 g が増えるにつれ、性能向上率は減少傾向にあることがわかる。その理由は、上記と同じであり、スレッド並列化したイテレーションの総数が減るためと考えられる。

5. 解探索性能の評価

5.1 概要

2021 年度の GraphGolf で出題された問題 $(h, r) = (32, 4)$, $(80, 6)$, $(432, 12)$, $(1281, 21)$, $(3800, 30)$, $(1024, 5)$, $(1024, 10)$, $(4608, 36)$, $(8208, 48)$, $(10000, 10)$, $(10000, 100)$, $(65536, 64)$ を用いて、3 章で提案したアルゴリズムの性能評価を行う。また、性能比較を行うため、2.1 節で述べた ODP のための最適化アルゴリズム [14] を用いたホストスイッチグラフ構成法を 5.2 節で述べる。なお、2021 年度の GraphGolf では、 $(h, r) = (128, 24)$ も出題されているが、このグラフは“最適なグラフの直径が 3 である”という特徴がある。その場合は決定的な方法でグラフを構成することができるため、この問題については A.1 節で取り扱う。

5.2 ODP のための最適化アルゴリズムを用いた方法

5.2.1 概要

特殊なホストスイッチグラフとして“正則ホストスイッチグラフ”がある。正則ホストスイッチグラフとは、各スイッチと隣接するホスト数とスイッチ数が全スイッチで同じグラフのことである。正則ホストスイッチグラフにおいて、各スイッチと隣接するホスト数は h/s 、スイッチ数は $(r-h/s)$ である。ここで、正則ホストスイッチグラフからホストを取り除いた場合、そのグラフは頂点数 s 、次数 $(r-h/s)$ の ODP グラフになる。逆に言うと、ODP グラフの各頂点をスイッチとみなし、各スイッチに h/s 台のホストを割り当てることで、ODP グラフから正則ホストスイッチグラフを作成することができる。

ASPL が小さい ODP グラフから作成された正則ホストスイッチグラフは、その h-ASPL も小さいことが期待できる。そこで、本節では我々が過去に開発した ODP のための最適化アルゴリズム [14] を用いて正則ホストスイッチグラフを作成する。その ODP のための最適化アルゴリズムは、3 章で説明した ORP のための最適化アルゴリズムとほぼ同じであるが、スイッチと隣接するホスト数を変化させる Swing の操作はない。また、正則ホストスイッチグラフであるため、 s は h の約数であるという制約がある。

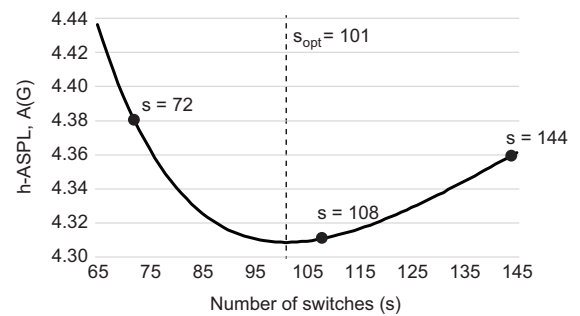


図 14: Continuous Moore bound in $(h, r) = (432, 12)$

5.2.2 スイッチ数の決定方法

スイッチ数が少なすぎる場合はホスト間の距離が増えるため、h-ASPL は大きくなる。逆に、スイッチ数が多すぎる場合は、ホスト間の経路に含まれないスイッチが増えるため、余計なポートがスイッチ間接続に利用されることになり、h-ASPL は大きくなる。以上のことから、ORP では適切なスイッチ数を設定することが極めて重要である。

適切なスイッチ数を求めるため、式 5 に示す連続 Moore Bound が提案されている [6]。

$$A(G) \geq \frac{M(s, r - h/s) \cdot (sh - h)}{sh - s} + 2 \quad (5)$$

$A(G)$ は正則ホストスイッチグラフの h-ASPL の下界を表す。 $M(n, d)$ は Moore bound [18] を計算する関数である。ORP においては h と r は定数であるため、 $A(G)$ が最小値になる s の値 (s_{opt}) が、正則ホストスイッチグラフにおいて有望なスイッチ数と考えることができる。なお、この式は正則でない場合（すなわち、 h/s が整数でない場合）も用いることができる。

例として、 $(h, r) = (432, 12)$ における連続 Moore Bound のグラフを図 14 に示す。このグラフから、 $s_{opt} = 101$ のときに $A(G)$ は最小値をとることがわかる。ただし、正則ホストスイッチグラフの場合、 s のとりうる値は h の約数である 72, 108, 144 などの値である。この中では、 $A(G)$ が最も低い値になる $s = 108$ が有望なスイッチ数になる。

5.2.3 性能評価

ODP のための最適化アルゴリズムのパラメータには、“最高温度”、“最低温度”、“評価計算回数”、“スイッチ数 (s)”、“対称性の数 (g)”がある。 s 以外は我々の過去の研究と同じように次の通り設定した [14]。最高温度は初期解から Swap を 100 回行った中の最悪値を 50% の確率で受理する値とした。最低温度は理論的な最小の改悪を 0.01% で確率で受理する値とした。評価計算回数は 10^6 とした。 s は 5.2.2 節で述べた方法と同様に、 h の約数の中から最小の $A(G)$ をとる値とした。 g は s の約数とした。

各問題において 20 試行を行った。その最良解を表 1 に示す。ASPL Gap と h-ASPL Gap は、ASPL および h-ASPL とそれぞれの下界との差であり、小さいほど良い値である。各問題における解探索の詳細については A.2 で説明する。

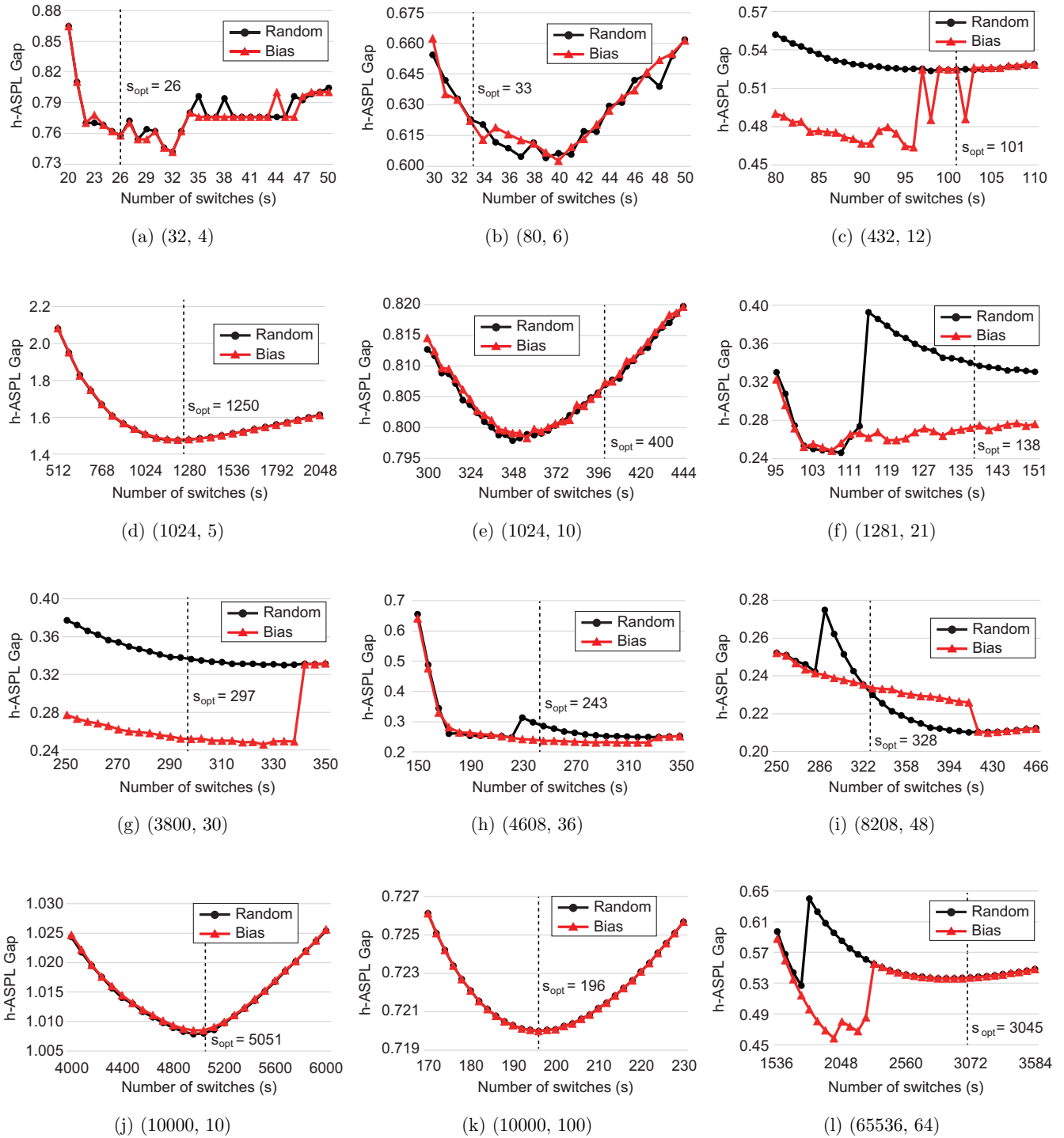


図 15: Solution search performance of proposed algorithm with no-symmetry

5.3 対称性がない場合の提案アルゴリズム

まず, $g = 1$ の場合の 3 章で提案したアルゴリズムの性能評価を行う. s の値については, s_{opt} 周辺の値を用いた. また, 3.3 節で説明したランダム選択とバイアス選択の両方を評価した. なお, $g = 1$ かつランダム選択の場合は, 既存研究のアルゴリズム [6] と同一である. それら以外のパラメータは 5.2.3 節で述べた方法を用いて決定した. 各問題において 20 試行を行った. 各 s における最良解を図 15 に示す. 図 15 より, s の値によって h-ASPL の値は大き

く異なることがわかる. また, ランダム選択とバイアス選択によっても最適な s の値が異なる場合があることがわかる. さらに, 各問題において, 最適な s の値と s_{opt} とが大きく異なる場合があることもわかる. その理由は, s_{opt} は正則ホストスイッチグラフを想定しているのに対し, 最良解の多くは正則ではないからである.

次に, ランダム選択とバイアス選択の比較を行う. 図 15 の各問題において, 最良解を得たときの s の値とその h-ASPL を表 2 に示す. 表中の “Ratio” は, ランダム選択と

表 1: Results using ODP optimization algorithm

$(h, r, s, r-h/s)$	s_{opt}	APSL Gap	h-APSL Gap
(32, 4, 32, 3)	26	0	0.7419
(80, 6, 40, 4)	33	0	0.5949
(432, 12, 108, 8)	101	0	0.5197
(1024, 5, 1024, 4)	1250	0.1453	1.4454
(1024, 10, 512, 8)	400	0.1705	0.8106
(1281, 21, 183, 14)	138	0.1172	0.3065
(3800, 30, 380, 20)	297	0.1214	0.2572
(4608, 36, 288, 20)	243	0	0.1682
(8208, 48, 342, 24)	328	0	0.1774
(10000, 10, 5000, 8)	5051	0.1202	0.9872
(10000, 100, 200, 50)	196	0	0.7201
(65536, 64, 4096, 48)	3045	0.0651	0.5356

表 2: Results using proposed algorithm with no-symmetry

(h, r)	Random		Bias		Ratio
	s	h-APSL Gap	s	h-APSL Gap	
(32, 4)	32	0.7419	32	0.7419	1.00
(80, 6)	39	0.6041	40	0.6025	1.00
(432, 12)	98	0.5235	96	0.4636	0.89
(1024, 5)	1184	1.4765	1248	1.4778	1.00
(1024, 10)	348	0.7979	354	0.7982	1.00
(1281, 21)	109	0.2459	104	0.2477	1.01
(3800, 30)	332	0.3293	326	0.2458	0.75
(4608, 36)	222	0.2485	310	0.2310	0.93
(8208, 48)	422	0.2101	426	0.2100	1.00
(10000, 10)	5000	1.0078	4920	1.0083	1.00
(10000, 100)	196	0.7200	196	0.7200	1.00
(65536, 64)	1728	0.5265	1984	0.4583	0.87

バイアス選択の h-ASPL の比であり、1.00 よりも小さい値の場合、バイアス選択で得られた解の方が良いことを示す。表中の数値は、0.05 よりも差がある場合に太字で表示している。表 2 より、(432, 12), (3800, 30), (4608, 36), (65536, 64) では、バイアス選択の方が良い解が得られた。他の問題においては、ランダム選択とバイアス選択の性能は同程度であった。

(432, 12), (3800, 30), (4608, 36), (65536, 64) の最良解のホストスイッチグラフにおいて、各スイッチと隣接するホスト数の分布を図 16 に示す。図 16(a) と図 16(b) では、バイアス選択の場合のみ、隣接するホスト数が 0 のスイッチが存在することがわかる。図 16(c) と図 16(d) においても、隣接するホスト数が 0 のスイッチはバイアス選択の方が多いことがわかる。この結果より、3.3 節で述べた通り、ホストと隣接しないスイッチが存在する方が良い場合があることを確かめることができた。なお、他の 8 個の問題に

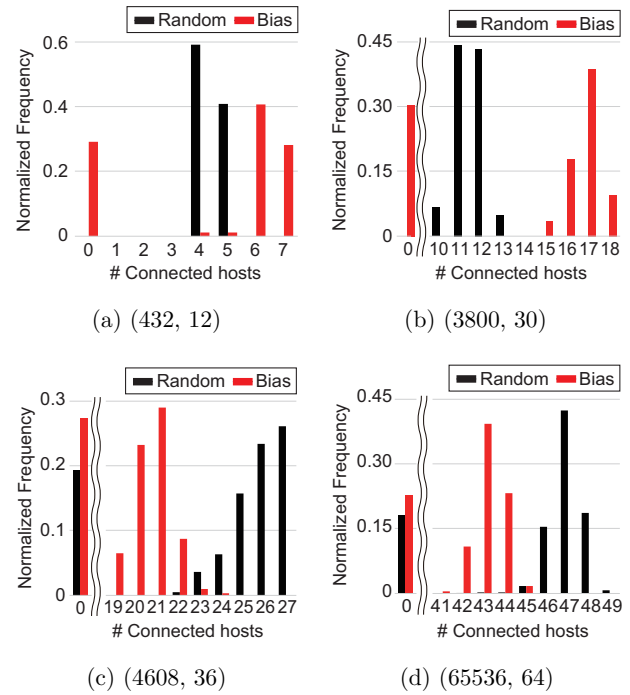


図 16: Host distribution when h-ASPL is the best solution

おけるホスト数の分布も調べたところ、ランダム選択とバイアス選択の分布はほぼ同じであったため、それらは同じような解が得られていると考えられる。以上のことから、バイアス選択は問題に応じてスイッチと隣接するホスト数を偏らせることにより、解探索性能を向上させると言える。

5.4 対称性がある場合の提案アルゴリズム

次に、 $g > 1$ の場合の 3 章で提案したアルゴリズムの性能評価を行う。3.1 節で述べた通り、 g の取りうる値は h と s の公約数であるため、いくつかの g と s の値の組合せを用いて実験を行った。各問題において 20 試行を行った中の最良解とその値を得たときの s と g の組合せを表 3 に示す。Random と Bias の列にある “Ratio to Table 2” は、表 2 に示した対称性がない提案アルゴリズムの解との比であり、1.00 よりも小さい値の場合は対称性のある提案アルゴリズムで得られた解の方が良いことを示す。表 3 より、対称性がない場合よりも対称性がある場合の方が良い解が得られることがわかる。また、表 2 と同様にバイアス選択はランダム選択と同等以上の性能を発揮していることがわかる。さらに、表 1 に示した ODP のための最適化アルゴリズムを用いた解と比較しても、対称性のある提案アルゴリズムの方が良い解が得られることがわかる。

図 16 と同様に、4 つの問題において、各スイッチと隣接するホスト数の分布を調べたところ、h-ASPL の差が大きかった (65536, 64) も含めてランダム選択とバイアス選択とではほぼ同じ分布であった。この理由は、最適化アルゴリズムに対称性を組み込んだ結果、問題が簡易化したため、良い解を発見しやすくなったからであると考えられる。な

表 3: Results using proposed algorithm with symmetry

(h, r)	Random				Bias				Ratio
	s	g	h-APSL Gap	Ratio to Table 2	s	g	h-APSL Gap	Ratio to Table 2	
(32, 4)	32	1	0.7419	1.00	32	1	0.7419	1.00	1.00
(80, 6)	40	2	0.5949	0.98	40	2	0.5949	0.99	1.00
(432, 12)	108	18	0.4432	0.85	99	9	0.4391	0.95	0.99
(1024, 5)	1280	128	1.4373	0.97	1152	128	1.4365	0.97	1.00
(1024, 10)	384	128	0.7659	0.96	384	128	0.7691	0.96	1.00
(1281, 21)	126	7	0.2394	0.97	126	7	0.2399	0.97	1.00
(3800, 30)	304	76	0.1707	0.52	304	76	0.1707	0.69	1.00
(4608, 36)	384	192	0.1775	0.71	288	144	0.1682	0.73	0.95
(8208, 48)	342	171	0.1774	0.84	342	171	0.1774	0.84	1.00
(10000, 10)	5000	500	0.9864	0.98	4500	500	0.9788	0.97	0.99
(10000, 100)	196	1	0.7200	1.00	196	1	0.7200	1.00	1.00
(65536, 64)	3072	1024	0.3306	0.63	3072	1024	0.2832	0.62	0.86

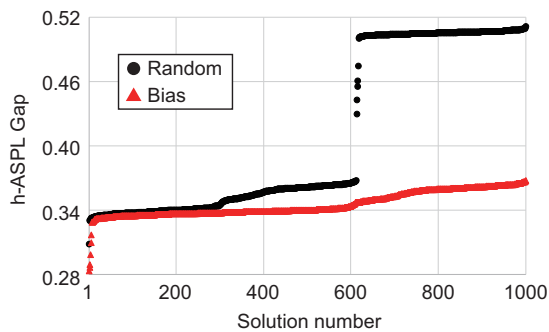


図 17: 1000 trials in $(h, r, s, g) = (65536, 64, 3072, 1024)$

お, $(65536, 64)$ のランダム選択とバイアス選択の最良解のホスト数の分布は, ホスト数が 0 のスイッチが 1024 個, ホスト数が 32 のスイッチが 2048 個の分布であった。

次に, ランダム選択とバイアス選択の間で最も差が大きかった $(65536, 64)$ の結果について考察するため, 表 3 のパラメータで 1000 試行を行った結果を図 17 に示す。図 17 では, 得られた結果を昇順でソートしている。この結果より, バイアス選択の方が安定して良い解を発見できることがわかる。また, ホスト分布について調べたところ, バイアス選択のすべての結果はホスト数が 0 のスイッチが 1024 個であったのに対し, ランダム選択の結果は 1000 試行中 616 試行のみがそのような分布であった。このことから, ホスト数を偏らせるというバイアス選択の効果は, $g > 1$ の場合の最適化アルゴリズムでも有効であることがわかる。

最後に, g の値と解探索性能の関係について $(h, r) = (10000, 10)$ と $(65536, 64)$ を用いて調べる。スイッチ数は表 3 の値を用いた。各問題において 20 試行を行った中の最良解を図 18 に示す。 $(10000, 10)$ については, 最適な s と g の値がランダム選択とバイアス選択で異なるため, 異なる図で結果を示している。図 18 より, g が大きくなる

ほど, h-ASPL の小さい解が発見できることがわかる。

6. 考察

5 章の結果より, ランダム選択よりもバイアス選択の方が, h-ASPL が小さいホストスイッチグラフを生成できることを示した。ここで, バイアス選択によって生成されたグラフはホストと隣接しないスイッチが多く現れるという特徴を持つため, 各選択方式によって得られるグラフのトポロジは大きく異なると考えられる。そのため, どちらのグラフの方が優れているかは h-ASPL のみで判断することはできない。そこで, 本章では, 提案アルゴリズムで得られるホストスイッチグラフに対する評価を行う。

評価を行うために, 並列シミュレーションフレームワーク SimGrid [19] と並列ベンチマーク集 NAS Parallel Benchmarks (NPB) [20] を用いることを考える。NPB のいくつかのベンチマークは 4 のべき乗のプロセス数のみサポートしている。4 章で用いたホストスイッチグラフでホスト数が 4 のべき乗なのは, $(h, r) = (1024, 5), (1024, 10), (65536, 64)$ のみである。しかしながら, $(1024, 5)$ と $(1024, 10)$ については, 図 15 の (d) と (e) より, 各選択方式から得られたグラフの h-ASPL の差は小さく, 各スイッチと隣接するホスト分布も似ていることを確かめた。 $(65536, 64)$ については, 図 15 の (l) より, h-ASPL については大きな差はあるものの, SimGrid において大きすぎるホスト数を用いることは難しい。

そこで, ホスト数を 1024 に固定し, r に様々な値を用いて事前評価を行った結果, $(h, r) = (1024, 16)$ が本実験に適していることがわかった。 $g = 1$ の場合の提案アルゴリズムを $(1024, 16)$ に適用した結果を図 19 に示す。ランダム選択とバイアス選択における最適なスイッチ数は共に 176 であり, h-ASPL Gap の最小値は 0.5205 と 0.4541

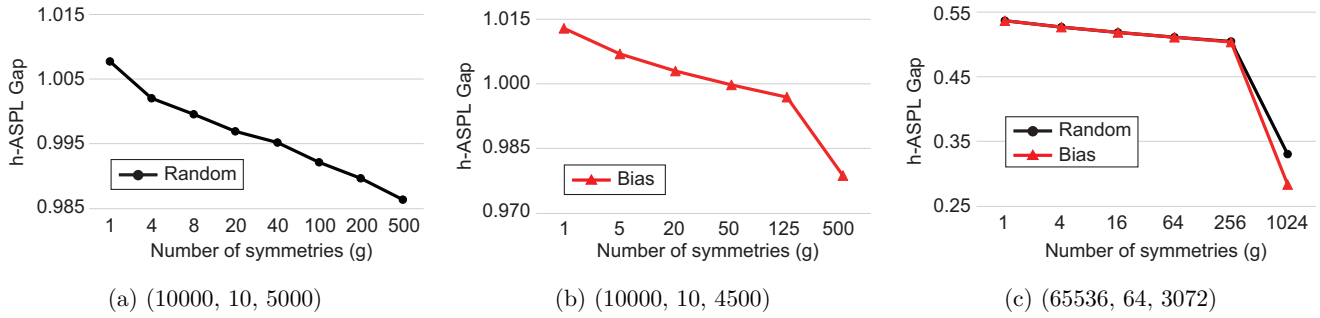


図 18: Solution search performance of proposed algorithm with symmetry (h, r, s)

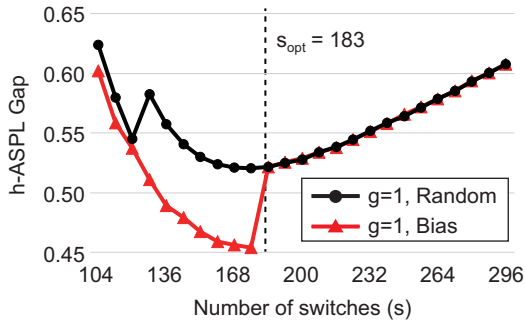


図 19: Result of graph (h, r) = (1024, 16)

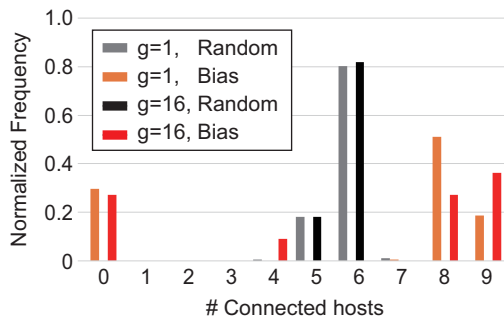


図 20: Host distribution (h, r, s) = (1024, 16, 176)

であった。次に、 $g > 1$ の場合を考える。性能比較を行うためにはスイッチ数は等しい方が望ましいため、スイッチ数を 176 に固定して提案アルゴリズムを適用した。その結果、ランダム選択とバイアス選択における h-ASPL Gap の最小値は 0.5069 と 0.4351 であった。なお、共に $g = 16$ の場合に最小値を得ることができた。それぞれのホストスイッチグラフにおけるスイッチと隣接するホスト数の分布を図 20 に示す。この結果より、バイアス選択の場合のみ、隣接するホスト数が 0 のスイッチが存在することがわかる。

次に、上記で生成した各ホストスイッチグラフを用いて SimGid-3.29 上に仮想的なネットワークを作成した。SimGrid で定義する並列計算機システムの計算ノードの性能は 500GFlops、ネットワークのバンド幅は 100Gbps とした。ネットワークレイテンシは、スイッチ間は 100ns、スイッチ-ホスト間は 600ns とした。MPI 集合通信の設定には、MVAPICH2 implementation を選択した。ホスト間のルーティングはワーシャル・フロイド法を用いた最短経

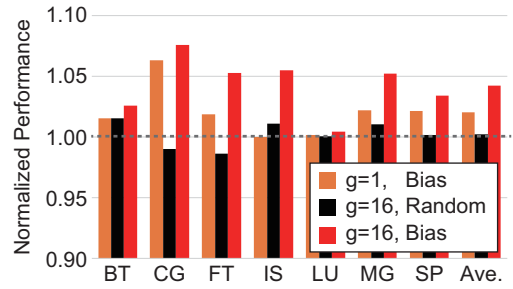


図 21: NPB results of graph (h, r, s) = (1024, 16, 176)

路を設定し、各スイッチにはランダムにホスト番号を割り当てた。並列ベンチマークとして、NPB-3.4-MPI の BT, CG, FT, IS, LU, MG, SP を用いた。問題サイズは、FT と IS は Class A、それ以外は Class B である。各アプリケーションにおける通信パターンは [4] が詳しい。

“ $g = 1$ かつランダム選択” の場合の性能を 1.0 として正規化した結果を図 21 に示す。1.0 より大きければ、“ $g = 1$ かつランダム選択” の場合よりも性能が高いことを表している。右端の「Ave.」は、すべてのベンチマーク性能の幾何平均である。この結果より、すべてのベンチマークにおいて、“ $g = 16$ かつバイアス選択” が最も性能が高いことがわかる。“ $g = 16$ かつバイアス選択” の幾何平均は 1.042 であった。LU については隣接通信が主であるため、すべてのアルゴリズムにおいて、大きな差は生じなかった。

7. まとめと今後の課題

本稿では、並列計算機システムにおいて低遅延な間接網を設計するため、グラフ理論の ORP に対する最適化アルゴリズムの提案を行った。提案アルゴリズムの特徴として、ホストスイッチグラフに対称性を持たせる点、スイッチによって隣接するホストを偏らせる点が挙げられる。これらの特徴により、h-ASPL を求めるための計算時間を削減でき、かつ h-ASPL がより小さいホストスイッチグラフを生成できる。さらに、SimGrid を用いて提案アルゴリズムによって得られたホストスイッチグラフから並列計算機システムのネットワークを仮想的に作成し、NPB による性能評価を行った。その結果、提案アルゴリズムが生成するネットワークは、既存アルゴリズムが生成するネットワー

クよりも高い性能を発揮できることを示した。

今後の課題として下記が挙げられる。(1) 図 15 の結果より、スイッチ数によって提案アルゴリズムの解探索性能は大きく変化することがわかる。提案アルゴリズムでは、スイッチ数は固定されたパラメータであるため、多くのスイッチ数との組合せを用いた実験が必要である。そこで、解探索中にスイッチ数を動的に変化させる仕組みを導入することにより、提案アルゴリズムの利便性を高めることができる。(2) 3.3 節で説明したバイアス選択において、ホストを偏らせる圧力を高めることにより、提案アルゴリズムの解探索性能を向上できる可能性がある。具体的には、3.3 節の最終段落で説明した確率式に重みとなるパラメータを導入することで実現可能である。

謝辞 This work was supported by JSPS KAKENHI Grant Number 21H03450, and used the Cygnus system provided by Interdisciplinary Computational Science Program in the Center for Computational Sciences, University of Tsukuba. We are grateful to the Graph Golf committee for giving us the interesting issues.

参考文献

- [1] M. Koibuchi, H. Matsutani, H. Amano, D. F. Hsu, H. Casanova. A case for random shortcut topologies for HPC interconnects. In *2012 39th Annual International Symposium on Computer Architecture (ISCA)*, pp. 177–188, June 2012.
- [2] A. Singla, C. Hong, L. Popa, P. B. Godfrey. Jellyfish: Networking Data Centers Randomly. *CoRR*, Vol. abs/1110.1687, , 2011.
- [3] J. Shin, B. Wong, E. G. Sirer. Small-world Datacenters. In *Proceedings of the 2Nd ACM Symposium on Cloud Computing, SOCC '11*, pp. 2:1–2:13, New York, NY, USA, 2011. ACM.
- [4] F. Chaix, I. Fujiwara, M. Koibuchi. Suitability of the random topology for hpc applications. In *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pp. 301–304, Feb 2016.
- [5] M. Koibuchi, I. Fujiwara, K. Ishii, S. Namiki, F. Chaix, H. Matsutani, H. Amano, T. Kudoh. Optical network technologies for HPC: computer-architects point of view. *IEICE Electronics Express*, Vol. 13, No. 6, pp. 1–14, 2016.
- [6] Y. Ryota, K. Michihiro, N. Koji, M. Hiroki, and A. Hideharu. Designing high-performance interconnection networks with host-switch graphs. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 30, No. 2, pp. 315–330, 2019.
- [7] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, Vol. 21, No. 6, pp. 1087–1092, 1953.
- [8] S. Kirkpatrick, C. D. Gelatt Jr., M. P. Vecchi. Optimization by simulated annealing. *Science*, Vol. 220, No. 4598, pp. 671–680, 1983.
- [9] M. Nakao, H. Murai, M. Sato. A method for order/degree problem based on graph symmetry and simulated annealing with mpi/openmp parallelization. In *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region, HPC Asia 2019*, pp. 128–137. ACM, 2019.
- [10] V. G. Cerf, D. D. Cowan, R. C. Mullin, R. G. Stanton. A lower bound on the average shortest path length in regular graphs. *Networks*, Vol. 4, No. 4, pp. 335–342, 1974.
- [11] R. Mizuno, Y. Ishida. Constructing large-scale low-latency network from small optimal networks. In *2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–5, Aug 2016.
- [12] T. Kitasuka, M. Iida. A heuristic method of generating diameter 3 graphs for order/degree problem (invited paper). In *2016 Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS)*, pp. 1–6, Aug 2016.
- [13] N. Shimizu, R. Mori. Average Shortest Path Length of Graphs of Diameter 3. *CoRR*, Vol. abs/1606.05119, , 2016.
- [14] M. Nakao, M. Sakai, Y. Hanada, H. Murai, and M. Sato. Graph optimization algorithm for low-latency interconnection networks. *Parallel Computing*, Vol. 106, p. 102805, 2021.
- [15] M. Koibuchi, I. Fujiwara. Graph Golf: The Order/degree Problem Competition, 2018. <http://research.nii.ac.jp/graphgolf>.
- [16] C. Godsil and G. Royle. *Algebraic Graph Theory*, Vol. 207 of *Graduate Texts in Mathematics*. volume 207 of Graduate Texts in Mathematics. Springer, 2001.
- [17] M. Nakao, H. Murai, M. Sato. Parallelization of all-pairs-shortest-path algorithms in unweighted graph. In *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region, HPCAsia2020*, pp. 63–72, New York, NY, USA, 2020. ACM.
- [18] M. Mirka, S. Jozef. Moore graphs and beyond: A survey of the degree/diameter problem. *Electronic Journal of Combinatorics, Dynamic survey*, Vol. 14, pp. 1–61, 12 2005.
- [19] C. Henri, G. Arnaud, L. Arnaud, Q. Martin, S. Frédéric. Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, Vol. 74, No. 10, pp. 2899–2917, June 2014.
- [20] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, S. Weeratunga. THE NAS PARALLEL BENCHMARKS. Technical Report NAS-94-007, Nasa Ames Research Center, 1994.

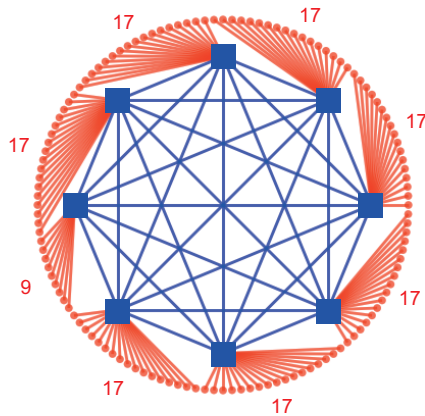


図 A-1: Optimal graph in $(h, r) = (128, 24)$

付 録

A.1 $(h, r) = (128, 24)$ について

直径が3であるホストスイッチグラフでは，“ r が偶数の場合， h の最大値は $r(r+2)/4$ である”という定理がある [6]。なお，直径が2であるホストスイッチグラフでは，1つのスイッチに全ホストが隣接するトポロジであるため， h の最大値は r である。このことから， $(h, r) = (128, 24)$ の直径は3であることがわかる。なお，5.1節で紹介した ORP の問題の内，直径が3になるのはこの問題だけである。

直径が3であるホストスイッチグラフの他の特徴として，全スイッチ間の距離は1であることが挙げられる。すなわち，スイッチは完全グラフを構成するため，各スイッチは $(s-1)$ 個のスイッチと隣接し，最大 $(r-s+1)$ 個のホストと隣接できる。このことから， h の最大値は $s(r-s+1)$ である。完全グラフの場合，1つのスイッチに多くのホストと隣接させる方が h-ASPL は小さくなるため， s は少ない方がよい。そのため， $s = 8$ が最適なスイッチ数となる。なお，5.2.2節で説明した方法を用いても，最適なスイッチ数は $s = 8$ になる。

次に，ホストとスイッチの接続について述べる。 $h = 128$ かつ $s = 8$ であるため，各スイッチに 16 個のホストを均等に割り当てることが，一般的なネットワーク設計としては妥当であると考えられる。しかしながら，1つのスイッチに可能な限り多くのホストを割り当てることが，h-ASPL がより小さいグラフになる。具体的には，8つ中7つのスイッチには 17 個のホストを割り当て，残りの1つのスイッチには 9 個のホストを割り当てたグラフが最適になる。なお，16 個ずつ均等に割り当てたグラフの h-ASPL Gap は 0.0222 であるのに対し，最適なグラフの h-ASPL Gap は 0.0202 である。図 A-1 に最適なグラフを示す。内側の□はスイッチ，外側の○はホスト，ホストの外側にある数字

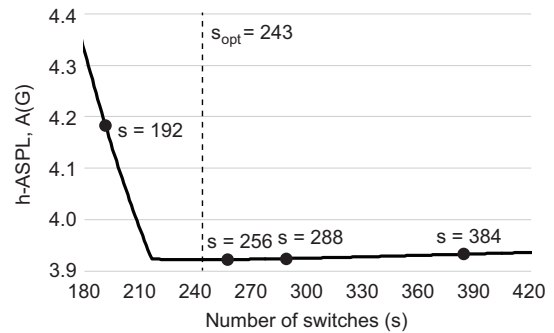


図 A-2: Continuous Moore bound in $(h, r) = (4608, 36)$

表 A-1: Results using ODP optimization algorithm

$(h, r, s, r-h/s)$	s_{opt}	APSL Gap	h-APSL Gap
$(4608, 36, 256, 18)$	243	0.0471	0.2134
$(4608, 36, 288, 20)$		0	0.1682
$(4608, 36, 384, 24)$		0.0013	0.1782

は，各スイッチと隣接するホスト数を示している。なお，図 A-1 は3章で提案したアルゴリズムを用いても得ることができるが，この最適なグラフは対称性を持たないため， $g = 1$ の場合に限ることに注意が必要である。

A.2 ODP における解探索性能

5.2.3節で行った ODP のためのアルゴリズムの性能評価について述べる。5.1節で紹介した ORP の各問題は，正則ホストスイッチグラフに解を限定する場合，頂点数 s ，次数 $(r-h/s)$ の ODP グラフを最適化する問題とみなすことができる。

まず， $(h, r) = (4608, 36)$ について，他の問題では見られなかった現象が発生したので紹介する。この問題における連続 Moore Bound のグラフを図 A-2 に示す。図から見て明らかのように，正則ホストスイッチグラフにおける有望な s の値は 256 である。しかし，実際は $s = 288$ の方が h-ASPL が小さいホストスイッチグラフを得ることができたため，表 1 には $s = 288$ の結果を示している。この理由について考察する。図 A-2 のグラフより， $s = 256$ 以降の $A(G)$ の値は大きく変化しないことがわかる。一般的に，ODP グラフでは頂点数 (s) が大きいほど解探索は難しくなり，次数 $(r-h/s)$ が大きいほど解探索は簡単になる。 $s = 256$ の次数は 18， $s = 288$ の次数は 20， $s = 384$ の次数は 24 である。どの組合せが最も良い解が見つかるかは，頂点数と次数の情報から判断することは難しいため， $A(G)$ の値の差が小さい s が複数ある場合は，すべてを試してみることが望ましいと言える。表 A-1 に，それぞれの場合において最適化を行った結果を示す。

次に，5.1節で紹介した ORP の各問題において，対称性の数 (g) を変化させながら最適化を行った。各 g における 20 試行中の最良解を図 A-3 に示す。この結果より， g を大

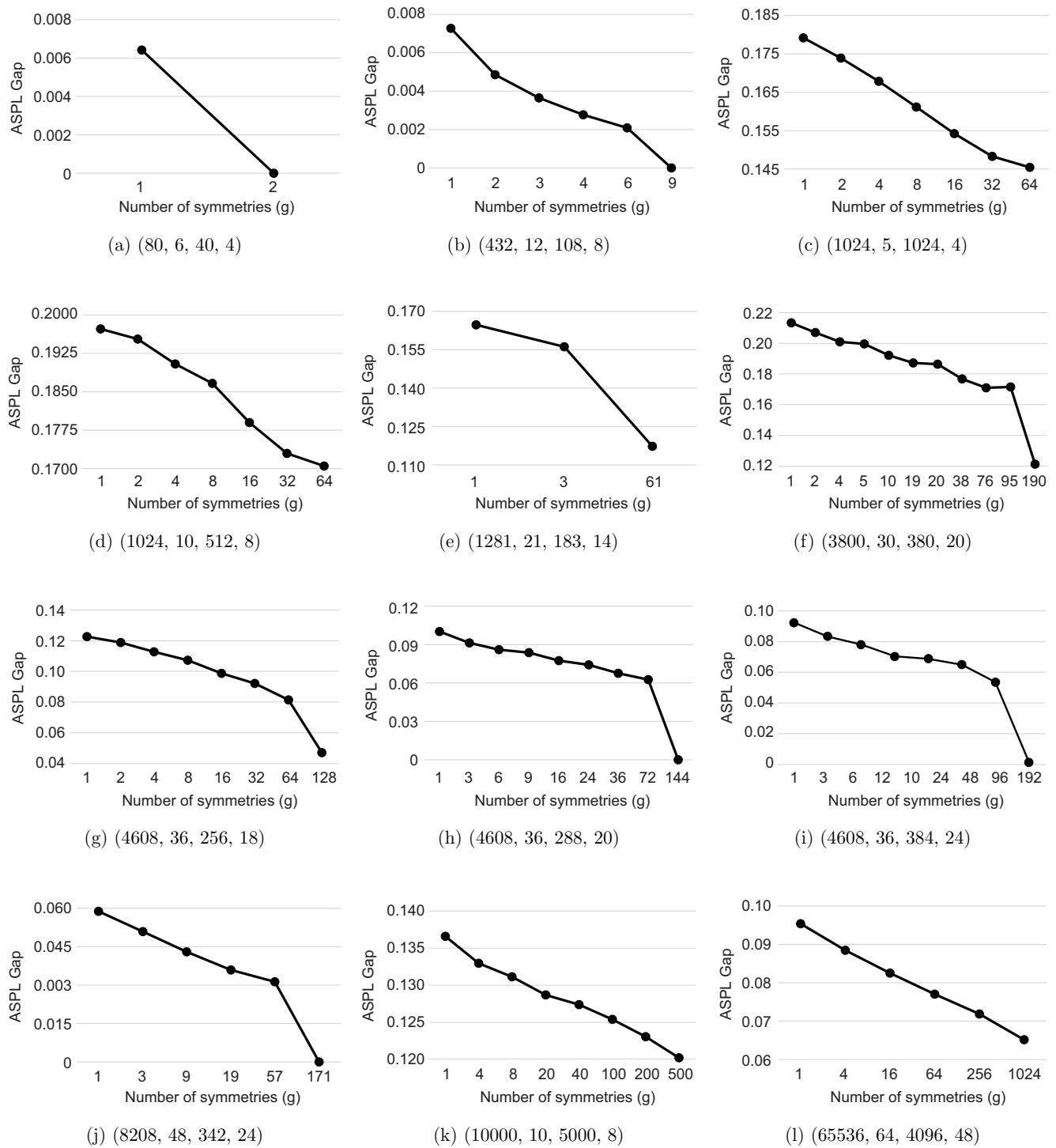


図 A-3: Relationship between solution search performance and number of symmetries ($h, r, s, r-h/s$)

きくするほど、良い解が発見できることがわかる。なお、 $(h, r, s, r-h/s) = (32, 4, 32, 3)$ と $(10000, 100, 200, 50)$ の問題は非常に簡単であり、 $g = 1$ でも理論的に下界のグラフ (ASPL Gap = 0) を容易に発見できたため、図 A-3 では省略している。また、図 A-3(g) と図 A-3(i) には、 $(h, r) = (4608, 36)$ の問題における $s = 256$ と $s = 384$ の結果を示している。

本著作物の著作権は情報処理学会に帰属します。本著作物は著作権者である情報処理学会の許可のもとに掲載するものです。ご利用に当たっては「著作権法」ならびに「情報処理学会倫理綱領」に従うことをお願いいたします。 <http://www.ipsj.or.jp/copyright/ronbun/copyright.html>