Vol.2018-HPC-167 No.23
2018/12/18

MPI/OpenMP 並列によるグラフ対称性と Simulated Annealing を用いた Order/Degree 問題の一解法

中尾 昌広[1,a)]　村井 均[1]　佐藤 三久[1]

概要：大規模システム内における各要素間を接続するネットワークのトポロジは，システム全体の通信遅延に強く関係していることが知られている．通信遅延が小さいネットワークトポロジを設計することは，そのネットワークトポロジを無向グラフとしてモデル化することで，グラフ理論上の Order/Degree 問題として定義できる．本稿では，Simulated Annealing（SA）をベースにした Order/Degree 問題を効率良く解くことができる手法を提案する．本手法は，ネットワークのトポロジに対称性を持たせることにより SA の解探索性能を高め，また計算時間を大幅に削減している．Order/Degree 問題のための国際コンペティションである Graph Golf で出題されているいくつかの問題に対して本手法を適用した結果，通信遅延の十分小さいネットワークトポロジを発見した．また，その対称性を利用した計算時間の削減手法を用いることで，Graph Golf が出題している $(n, d) = (72, 4)$，$(256, 5)$，$(256, 10)$ の問題において，それぞれ 8.11 倍，31.76 倍，15.67 倍の高速化を達成した．さらに，MPI と OpenMP を用いたハイブリッド並列化を行うことにより，さらなる計算時間の削減を行った．その結果，Graph Golf が出題している $(n, d) = (400000, 32)$ の問題において，最大 209.80 倍の性能向上を達成した．また，トポロジの対称性とハイブリッド並列化を組合せることにより，その問題においては最大 2,098,000 倍の高速化を達成できたと考えられる．

Masahiro Nakao[1,a)]　Hitoshi Murai[1]　Mitsuhisa Sato[1]

1. はじめに

大規模データセンタ内の計算機をつなぐネットワークケーブルやチップ内の配線などにおいて，そのネットワークトポロジがシステム全体の通信遅延に強く影響を与えることが知られている．その通信遅延を削減する方法の 1 つに，小さい直径および頂点間の平均距離（ASPL：Average Shortest Path Length）を持つネットワークトポロジを採用することが挙げられる [1, 2]．そのネットワークトポロジは無向グラフとしてモデル化でき，グラフ理論上の Order/Degree 問題として定義できる．Order/Degree 問題とは，与えられた頂点数と次数を満たす直径と ASPL が最小の無向グラフを求める問題のことである．以降，本稿では無向グラフを単に “グラフ” と呼ぶ．

国立情報学研究所が主催している Order/Degree 問題の国際コンペティションに “Graph Golf” がある [3]．このコンペティションは 2015 年から毎年開催されており，年ごとにいくつかの頂点数と次数の組合せを出題している．出題は，頂点を自由に配置できる “General Graph Category” と頂点を格子上に配置する “Grid Graph Category” があり，本稿では General Graph Category について扱う．Graph Golf の参加者は限られた期間内（2018 年は 5 月 14 日から 10 月 14 日）に，Graph Golf 公式サイトの Web フォームから，自身が発見したグラフを投稿することができる．

Order/Degree 問題の定義はシンプルであるが，その効率的な解法は発見されていない．Order/Degree 問題が難しい理由は，下記の 2 点であると考えられる．(1) 与えられた頂点数と次数を満たすグラフの数は膨大であり，また Order/Degree 問題を離散最適化問題と見なした場合，多数の局所解を持つような問題であるため，最適解を発見することが難しい点．(2) 直径と ASPL を求めるための計算時間が膨大である点．直径と ASPL を求めるためには，すべての頂点から，その頂点以外の最短経路を求める必要がある．いわゆる全点対最短経路問題（APSP：All Pair Shortest Paths）である．APSP を行うためには全頂点に対して幅優先探索を行えば良いので，ある頂点数 n と次数

[1] 理化学研究所 計算科学研究センター
RIKEN Center for Computational Science
[a)] masahiro.nakao@riken.jp

d の場合の APSP の計算オーダは $O(n^2 d)$ である[*1]．このため，頂点数が増えると APSP のための計算時間は非常に大きくなる．

本稿では Order/Degree 問題に対する新しい解法を提案する．本解法の特徴は下記の通りである．(1) 汎用的な最適化手法である Simulated Annealing（SA）[7,8] を用いることで，局所解に容易に陥らない解探索を行う．(2) グラフに対称性を持たせることにより，探索空間を削減し，SA の解探索を効率的に行う．さらに，その対称性を利用することにより，APSP の計算時間を大幅に削減する．(3) APSP の計算に対して MPI および OpenMP を用いたハイブリッド並列化を行い，クラスタシステム上で動作させることにより，さらなる計算時間の削減を行う．

本稿の構成は下記の通りである．2 章では Order/Degree 問題の概要とその関連研究について述べる．3 章では提案手法について述べる．4 章では提案手法の解探索性能について評価し，5 章ではその高速化について評価する．6 章では本稿のまとめと今後の課題について述べる．

2. Order/Degree 問題

2.1 概要

Order/Degree 問題とは，与えられた頂点数と次数を満たすグラフの直径と ASPL を最小化する問題のことである．その頂点数 n と次数 d から，理論的な直径の下界 $K_{n,d}$ と ASPL の下界 $L_{n,d}$ は，下記のように計算できる [1,9]．

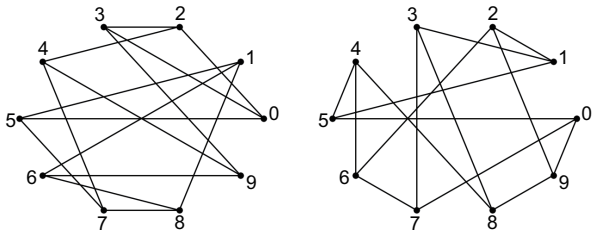
$$K_{n,d} = \begin{cases} \left\lceil \frac{n-1}{2} \right\rceil & \text{if } d = 2 \\ \left\lceil \log_{d-1}\left(\frac{(n-1)(d-2)}{d}\right) + 1 \right\rceil & \text{if } d > 2 \end{cases} \quad (1)$$

$$L_{n,d} = \begin{cases} 1 & \text{if } K_{n,d} = 1 \\ \frac{S_{n,d} + K_{n,d} R_{n,d}}{n-1} & \text{if } K_{n,d} \geq 2 \end{cases} \quad (2)$$

$$S_{n,d} = \sum_{i=1}^{K_{n,d}-1} i d (d-1)^{i-1} \quad (3)$$

$$R_{n,d} = n - 1 - \sum_{i=1}^{K_{n,d}-1} d (d-1)^{i-1} \quad (4)$$

$n = 10$, $d = 3$ のグラフの例とそれぞれの距離行列を**図 1** に示す．**図 1a** はランダムにエッジを配置したグラフであり，**図 1b** は最適化した結果，その直径と ASPL は共に下界であるグラフである．距離行列はグラフの各点から他の頂点までの距離を表している．また，距離行列は対称行列である．距離行列が持つ要素の最大値が直径であり，距離行列の値の総和をその対角成分を含まない要素数（$n^2 - n$）で割った値が ASPL である．



	0	1	2	3	4	5	6	7	8	9
0		2	1	1	2	1	3	2	3	2
1	2		3	3	3	1	1	2	1	2
2	1	3		1	1	2	3	2	3	2
3	1	3	1		2	2	2	3	3	1
4	2	3	1	2		2	2	1	2	1
5	1	1	2	2	2		2	1	2	3
6	3	1	3	2	2	2		2	1	1
7	2	2	2	3	1	1	2		1	2
8	3	1	3	3	2	2	1	1		2
9	2	2	2	1	1	3	1	2	2	

(a) 直径=3，ASPL=1.89

	0	1	2	3	4	5	6	7	8	9
0		2	2	2	2	1	2	1	2	1
1	2		1	1	2	1	2	2	2	2
2	2	1		2	2	2	1	2	2	1
3	2	1	2		2	2	2	1	1	2
4	2	2	2	2		1	1	2	1	2
5	1	1	2	2	1		2	2	2	2
6	2	2	1	2	1	2		1	2	2
7	1	2	2	1	2	2	1		2	2
8	2	2	2	1	1	2	2	2		1
9	1	2	1	2	2	2	2	2	1	

(b) 直径=2，ASPL=1.67

図 1: $n = 10$，$d = 3$ のグラフの例

2.2 関連研究と提案手法との比較

各頂点とエッジとをランダムに繋いだグラフ（ランダムグラフ）の直径と ASPL は，それらを規則的に繋いだグラフよりも小さいことが経験的に知られている．ただし，ほとんどの場合，ランダムグラフの直径と ASPL はそれらの下界よりも大きな値をとるため，直径と ASPL がより小さいグラフを発見することが重要である．また，ランダムグラフが持つランダム性を様々なネットワークに利用する研究が行われている [10–13]．

Graph Golf の公式サイト [3] において，過去の参加者が開発した解法のスライドと論文が公開されている [14–16]．これまでの手法は，下記の 2 つに区分することができる．(1) 頂点数が小さい複数のグラフを組合せることで，Graph Golf が出題するグラフを構成する．この手法は APSP のための計算をほとんど行わなくても良いという利点はあるが，特定の頂点数と次数のグラフしか作れない問題点がある．(2) ランダムグラフをベースに SA などの最適化手法を用いて最小の直径と ASPL を持つグラフを探索する．この手法は任意の頂点数と次数のグラフを作成できるという利点はあるが，APSP のための膨大な繰り返し計算が必要であるため，その計算には近似を用いる場合が多い．しかし，近似を用いると誤差が発生するため，最適化手法の解探索性能に悪影響を与える可能性がある．また，最適解発見時における解探索の停止も行えない．

本稿で提案する手法では，頂点数の少ないランダムなグラフを対称的になるように組合せ，その対称性を保ったまま SA を用いた最適化を行う．また，SA による解探索を

[*1] 密なネットワークにおける APSP のためのアルゴリズムとしては Floyd-Warshal Algorithm [4,5] や Seidel Algorithm [6] があり，それらの計算オーダはそれぞれ $O(n^3)$ と $O(n^{2.376} \log n)$ である．しかしながら，Graph Golf では比較的疎なネットワークが用いられているため，幅優先探索を利用した手法の方が高速であると考えられる．

精度良く行うために，近似は用いずに正確な直径と ASPL を求めている．すなわち，本手法は上記の (1) と (2) の両方の特徴を持っていると言える．ただし，(1) の問題点である頂点数と次数の制約については，比較的緩い制約しか持っていない．本手法の制約については 3.3.6 節で述べる．また，(2) の問題点である APSP のための計算時間は，グラフの対称性とハイブリッド並列を利用することにより，大幅に削減している．

3. グラフの対称性を利用した SA

本章では，まず 3.1 節で通常の SA の概要を述べる．次に，3.2 節で通常の SA を Order/Degree 問題に適応する方法について述べる．最後に，3.3 節で通常の SA を拡張した本提案手法について述べる．

3.1 SA の概要

SA とは，焼きなましと呼ばれる金属工学における金属材料の冷却過程をシミュレーションした汎用近似解法の 1 つである．SA の特徴は，山登り法のように解が改善される方向のみに探索を行うのではなく，解が悪化する方向にも確率的に探索を行うことである．この特徴により，SA は局所解から脱け出すことができ，効率的な解探索を行うことができる．また，Genetic Algorithm（GA）[17,18] などの他の進化的計算手法と比べて，SA のアルゴリズムは非常に簡易かつ汎用的であるため，広範囲な実問題に適用しやすいという特徴を持つ．

SA は温度というパラメータを持ち，解探索が進むにつれ，温度は高い状態から低い状態に遷移する．**図 2** と下記に SA の基本アルゴリズムを示す．(1) $k = 1$ とし，初期解 x_k を用意する．そして，その初期解 x_k に対応するエネルギー $E(x_k)$ を計算する．なお，低いエネルギーほど良い値である．また，初期温度 T_k を設定する．(2) x_k を元に新しい解 x' を生成する．(3) x' のエネルギー $E(x')$ を計算し，元の解とのエネルギー差 $\Delta E = E(x') - E(x_k)$ を計算する．なお，図 2 では省略しているが，$E(x')$ が最適解の場合は，解探索を終了する．(4) エネルギー差 ΔE と温度 T_k に応じて，その新しい解 x' を受理するかを決める．(5) 受理の場合は x' を x_{k+1} に，$E(x')$ を $E(x_{k+1})$ に変更する．(6) (2) から (5) の処理をある定めた回数繰り返す．(7) ある定めた回数に達したら，温度 T_k から新しい温度 T_{k+1} に変更する．(8) 温度が十分に下がったとき，もしくはある定めた回数に達したとき終了する．

図 2 の受理判定 (4) では，下記の Metropolis 基準がよく用いられる．

$$Probability = \begin{cases} 1 & \text{if } \Delta E < 0 \\ \exp(-\frac{\Delta E}{T}) & otherwise \end{cases} \quad (5)$$

Metropolis 基準では，新しい解のエネルギーが現在の解のエネルギーよりも良い場合，新しい解は 100%の確率で受理される．その逆の場合，新しい解は確率的に受理される．温度が高いほど，その受理確率は高くなる．すなわち，解探索初期は改悪方向への解探索も積極的に行う大域的探索が行われ，解探索末期は改善方向への解探索を主に行う局所的探索が行われる．

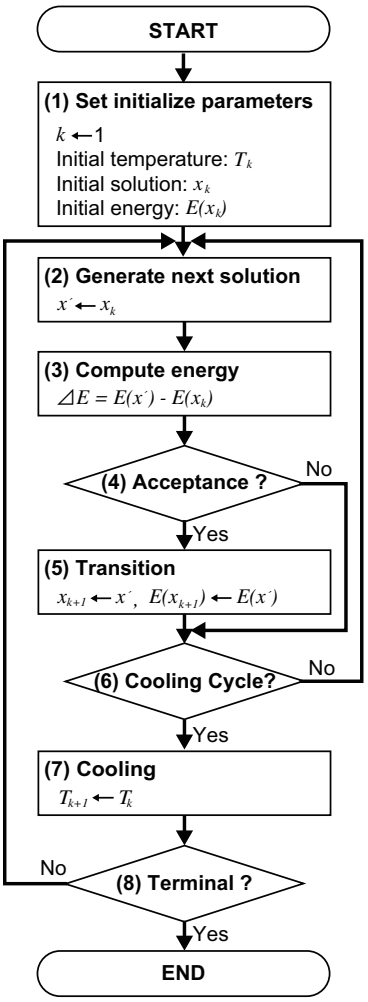


図 2: SA のアルゴリズム

図 2 の徐冷処理 (7) では，下記の指数型アニーリングがよく用いられる．

$$T_{k+1} = \alpha T_k \quad (6)$$

指数的アニーリングでは，ある温度 T_k にクーリング率 α を掛けることで，新しい温度 T_{k+1} を計算する．クーリング率 α の値が 1.0 に近いほど徐冷スピードは遅くなるため，良い解が発見できる確率は高くなるが，温度が低くなるまでの繰り返し回数が増大する．

3.2 SA の Order/Degree 問題への適用

3.2.1 初期解の生成方法

初期解の生成には，グラフ理論のための Python パッケージである networkx [19] の random_regular_graph メソッド

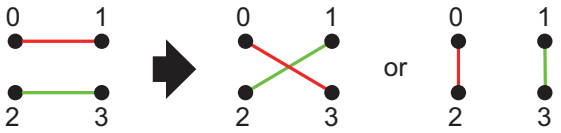


図 3: 2-opt 法の概要

を利用する[*2]．このメソッドは，指定した頂点数と次数を持つ正則グラフを出力する．なお，出力される正則グラフは，ループと重複エッジを含まない単純グラフである．

3.2.2 エネルギーの計算方法

Order/Degree 問題の目的は最小の直径と ASPL を求めることであるため，エネルギーの計算式には直径と ASPL を用いることが自然である．そこで，ある解 x における直径を $D(x)$，ASPL を $ASPL(x)$，重みを w とした場合のエネルギー $E(x)$ の値を下記の式で表す．重み w は適当な正の実数である．

$$E(x) = w \times D(x) + ASPL(x) \qquad (7)$$

しかしながら，この式では重み w を問題毎に設定する必要があるため，パラメータチューニングの手間が増えるという問題点がある．また予備実験を行った結果，重み w に大きな値を設定すると，改悪方向への探索が行われにくくなるため，探索性能が悪化することがわかった．最小の ASPL を持つネットワークは最小の直径を持つと考えられるため，本手法では式 (7) において $w = 0$ とし，直径についてはエネルギーの計算式に用いないこととした．

3.2.3 新しい解の生成方法

SA では，現在の解に微小な摂動を加えることで次の解を生成する．その摂動の範囲を近傍と呼ぶ．Order/Degree 問題のような組合せ最適化問題では，解の変更に必要な最小の集合を近傍と定義することができる．

そこで，巡回セールスマン問題などでもよく用いられる 2-opt 法を用いて，Order/Degree 問題における次の解の生成を行う．**図 3** に 2-opt 法の概要を示す．2-opt 法では，グラフの中から 2 つのエッジをランダムに選択し，その 2 つのエッジを入れ替えることにより，次の解を生成する．なお，入れ替え方は 2 通りあるため，そのどちらかをランダムに選択する．

本手法における 2-opt 法の疑似コードを**図 4** に示す．変数 *lines* は総エッジ数を示しており，配列 *edge* はエッジの情報が格納されている配列である．なお，配列 *edge* の 2 次元目にはエッジの両端の頂点の番号が格納されている．5–6 行目では，全エッジの中からランダムに 2 つのエッジを選んでいる．下記の場合は，エッジの選択をやり直すフローになっている．(1) 選択された 2 つのエッジが同じであった場合．(2) 選択された 2 つのエッジの頂点の内の 1

[*2] このメソッドを利用した Python プログラムは Graph Golf 公式サイトで公開されている．`http://research.nii.ac.jp/graphgolf/py/create-random.py`

```
1  function edge_exchange(lines, edge[lines][2])
2    do while
3      do while
4        do while
5          line[0] = Random(lines)
6          line[1] = Random(lines)
7        end do while line[0] == line[1] // (1)
8      end do while check_duplicated_vertex(line, edge) // (2)
9      edge_exchange_2opt(line, edge)
10   end do while check_multigraph(line, edge) // (3)
11 end function
```

図 4: 2-opt 法の疑似コード

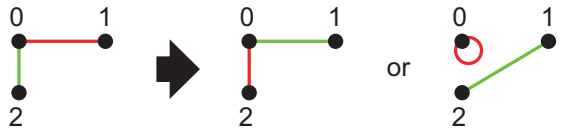


図 5: 2 つのエッジの頂点の内の 1 つが同一の場合

つが同一であった場合．**図 5** に示すように，それらの線を入れ替えても同一のグラフになるかループができてしまうため，エネルギー E の改善は見込めないからである．(3) 9 行目で 2-opt 法を適用した後に多重辺を持つグラフ（多重グラフ）になった場合．多重グラフのエネルギーは単純グラフのエネルギーと比較して悪いと考えられるためである．また，図 4 では省略しているが，2-opt 法の適用後にグラフが連結でなくなった場合も，2-opt 法をやり直すフローになっている．グラフの連結性については幅優先探索を行うことにより調べることができるため，関数 *edge_exchange()* の後に行う ASPL を求める計算と同時にグラフの連結性についても調べている．

3.2.4 受理判定で用いるエネルギー差

2.1 節で説明した距離行列と ASPL の定義により，式 (5) におけるエネルギー差 ΔE の最小値は $2/(n^2 - n)$ である．すなわち，頂点数が多いほどエネルギー差 ΔE の最小値は小さくなる．その最小値が問題毎に異なると温度パラメータの設定が煩雑になるため，下記の式の通り，エネルギー差 ΔE に重みとして $(n^2 - n)$ を掛けることにする．

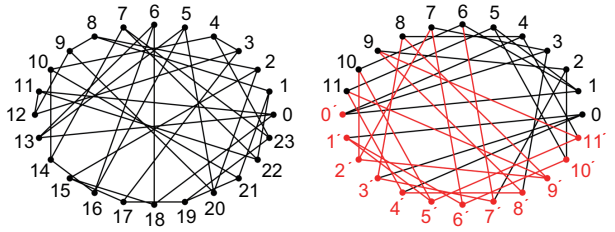
$$\Delta E = (E(x') - E(x_k)) \times (n^2 - n) \qquad (8)$$

3.3 提案手法

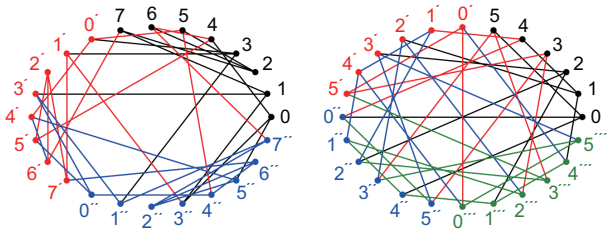
3.3.1 概要

1 章で述べた通り，Order/Degree 問題の解探索が難しい理由として，その探索空間が非常に広いこと，ASPL の計算に時間を要することが挙げられる．そこで，グラフに対して対称性を持たせることにより，探索空間と ASPL の計算時間を共に削減する方法について述べる．

本手法で用いる対称性を持たせたグラフの例を**図 6** に示す．各グラフは $n = 24$，$d = 3$ であり，対称的なトポロジを持っている．図中の変数 g はグループ数であり，各グラ



(a) g=1　(b) g=2



(c) g=3　(d) g=4

図 6: $n = 24$，$d = 3$ の対称的なグラフの例

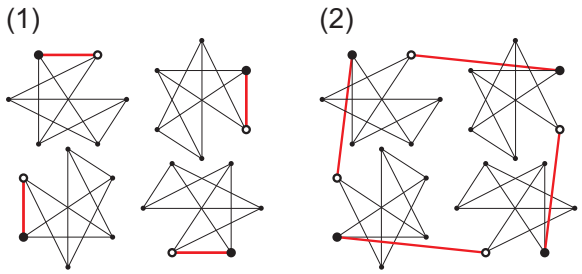


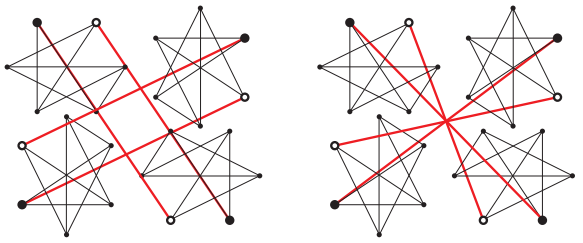
図 7: 対称的なグラフの初期解の生成方法（$r = 1$）

フを平面としてみた場合，$360/g$ 度回転させるとエッジと頂点の接続関係が同じグラフになる．$g = 1$ の場合は，通常のグラフである．

3.3.2 1g-opt 法による初期解の生成方法

解探索を始めるために，対称的なトポロジを持つ初期解を生成する．対称的なトポロジを持つ初期解を作成する方法はいくつかあるが，本節では 3.2.1 節で述べた random_regular_graph メソッドを利用する方法を説明する．

例として，**図 6d** に示した $n = 24$，$d = 3$，$g = 4$ のグラフのための初期解を作成する．その概要を**図 7** と下記に示す．(1) 対象となるグラフの頂点数 n をグループ数 g で割った値 (6 = n/g) を頂点数として持つグラフを random_regular_graph メソッドを用いて作成する．次数は対象となるグラフと同じである．このグラフをベースグラフと呼ぶ．ベースグラフを g 個複製した後，対称的になるように各ベースグラフからエッジを 1 つずつ選択する．便宜上，図 7 では，4 つのベースグラフは 90 度ずつ回転させており，また選択したエッジの始点 ◦ と終点 • を設定している．(2) エッジの始点 ◦ と右回りに r 個離れたベースグラフのエッジの終点 • とを接続する．r は 1 から $g - 1$ までのランダムな整数である．図 7 では，$r = 1$ の場合を示



(a) $r = 2$ の場合　(b) g が偶数の場合の特殊例

図 8: 対称的なグラフの初期解の生成の失敗例

```
1  function edge_exchange(lines, edge[lines][2], groups)
2    do while
3      do while
4        do while
5          line[0] = Random(lines)
6          line[1] = Random(lines)
7        end do while line[0] == line[1]
8      end do while check_duplicated_vertex(line, edge)
9      if check_symmetric_edge(line, edge, groups) then
10       edge_exchange_1g_opt(line[Random(2)], edge, groups)
11     else
12       edge_exchange_2g_opt(line, edge, groups)
13     end if
14   end do while check_multigraph(line, edge, groups)
15 end function
```

図 9: 対称的である新しい解の生成方法の疑似コード

している．

次に，上記のエッジの交換が失敗する例を示す．$r = 2$ の場合の例を**図 8a** に示す．このグラフは，接続されていない部分構造がある非連結グラフである（例えば，左上と右上のベースグラフは接続されていない）．Order/Degree 問題では，非連結グラフは解として扱うことはできない．また，g の値が偶数の場合は，180 度回転した方向にいるベースグラフの始点同士，終点同士をつなげることで，対称グラフを作成することができる．その例を**図 8b** に示す．しかしながら，ベースグラフに対してこの操作を行うと，必ず非連結グラフになる．

以上のようにエッジの交換後に非連結グラフになった場合は，連結グラフになるまでエッジと r の値を選び直してエッジ交換を繰り返すこととする．このエッジの交換方法を本稿では 1g-opt 法と呼ぶ．1g-opt 法は，3.3.3 節で述べる新しい解の生成においても利用する．

3.3.3 1g-opt 法と 2g-opt 法による新しい解の生成方法

本手法では，初期解が持つグラフの対称性を保ったまま，現在の解の近傍にある新しい解を生成する．この操作には，3.3.2 節で述べた 1g-opt 法と，3.2.3 節で述べた 2-opt 法を拡張した 2g-opt 法を用いる．新しい解を生成するフローの疑似コードを**図 9** に示す．このコードは図 4 を拡張したものであり，主要な変更点は 9–13 行目のみである．

まず，図 9 の 12 行目で実行する 2g-opt 法の概要を**図 10** と下記に示す．図 10 では，$n = 12$，$d = 3$，$g = 2$ のグラ

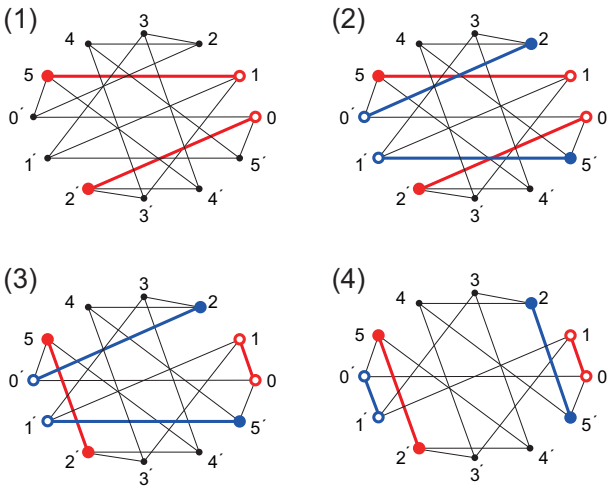
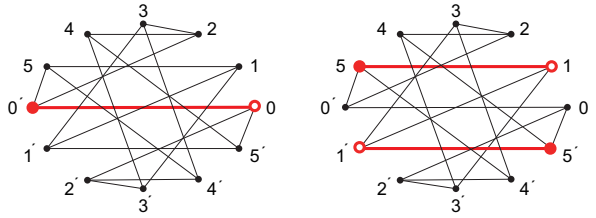


図 10: 2g-opt 法の概要（$n = 12$，$d = 3$，$g = 2$）



(a) 選択したエッジの両端が対称的な場合　(b) 選択した 2 つのエッジが対称的な場合

図 11: 2g-opt 法の失敗例

フを用いている．(1) すべてのエッジの中からランダムに 2 つのエッジを選択する．(2) 選択した 2 つのエッジと対称的な関係にあるエッジを選択する．(3) (1) で選択したエッジ間で 2-opt 法を行う．(4) (3) と同様の 2-opt 法を (2) で選択したエッジ間でも行う．

次に，上記の 2g-opt 法が失敗する例を**図 11** と下記に示す．図 10 の (1) で選択した 1 つ（もしくは 2 つとも）のエッジの両端が対称的な関係である場合（**図 11a**），もしくは図 10 の (1) で選択した 2 つのエッジが対称的な関係である場合（**図 11b**），図 10 の (2) の操作を行うことができない．このように 2g-opt 法が適用できない場合は，図 9 の 9–10 行目にあるように，2g-opt 法の代わりに 1g-opt 法による解生成を行う．具体的には，選択済みの 2 つのエッジからランダムに 1 つを選択し，そのエッジに対して 1g-opt 法を適用する．

3.3.4 ASPL の計算量削減方法

グラフの対称性を利用することにより，ASPL を求めるために必要な計算量を削減することが可能である．一般に，ASPL を求めるためには，すべての頂点から他の頂点までの距離を幅優先探索を用いて計算する必要がある．しかし，本手法で用いている対称的なグラフでは，対称的な関係のある頂点同士において，その頂点から他の頂点までの距離はすべて同じであるという性質がある．例えば，図 6d では，頂点 0，0′，0″，0‴ から，他の頂点までの距離の集合はすべて同じである．すなわち，本手法において ASPL を求めるには，(n/g) 個の頂点から幅優先探索を行うだけでよい．図 6d では，頂点 0〜5 から幅優先探索を行うだけで ASPL を求めることができる．

1 章で述べた通り，通常の ASPL を求めるための計算オーダは $O(n^2 d)$ であるが，この計算量削減方法を適用することで，その計算オーダは $O(n^2 d/g)$ になる．

3.3.5 受理判定で用いるエネルギー差

3.2.3 節で述べた通常の 2-opt 法と比較して 3.3.3 節で述べた対称的にエッジを交換する方法は，より多くのエッジの交換を行う．そのため，式 (8) におけるエネルギー差 ΔE も大きくなる．予備実験を行った結果，その変化量はグループ数 g に比例して大きくなることがわかった．

ある 1 つの問題において，グループ数 g の値を変える毎に，温度パラメータも調整することは煩雑な作業である．そこで，下記のように式 (8) に重みとして $1/g$ を掛けることにより，同じ問題であればグループ数 g の値に依らずに同じ温度で解探索を行えるようにした．

$$\Delta E = (E(x') - E(x_k)) \times (n^2 - n)/g \qquad (9)$$

3.3.6 本手法の制約

本手法の制約として下記が挙げられる．(1) グラフは対称的なトポロジを持つため，グループ数 g は頂点数 n の約数である必要がある．(2) 正則グラフにおいて頂点数は必ず次数よりも大きいため，ベースグラフは $n/g > d$ である必要がある．ただし，(2) については，適当な d の値を用いて 1g-opt 法でグラフを作成した後，各頂点に対して手動で対称的にエッジを足す（すなわち d を増やす）という処理を追加することにより，その制約を回避することができる．

4. パラメータと解探索性能の評価

4.1 パラメータの設定指針

3.1 節に示した通常の SA のパラメータは，初期温度（最高温度）T，終了温度（最低温度）C，徐冷処理のための繰り返し数 I，総計算回数 N の 4 つである．式 (6) のクーリング率 α は上の 4 つの値から下記の式で導くことができる．

$$\alpha = (C/T)^{I/N} \qquad (10)$$

最高温度 T の設定については，解探索初期における解の改悪方向への受理確率は高い方が望ましいと考えられる．そこで，各問題に対してランダムグラフから新しい解の生成を 100 回行い，得られた最大のエネルギー差 ΔE を 50%の確率で受理する温度を最高温度と設定した．最低温度 C の設定については，解探索末期における解の改悪方向への受理確率は低い方が望ましいと考えられる．3.2.4 節で

述べた通り，どのような問題でも最小のエネルギー差 ΔE は 2 である．そこで，最小のエネルギー差を 0.01%で確率で受理する温度を最低温度とした．徐冷処理のための繰り返し数 I の設定については，予備実験を行った結果，どのような値であっても解探索性能に変化はないことがわかった．そのため $I = 1$ とした．総計算回数 N は 10^7 とした．

提案手法では，上記に挙げた通常の SA のパラメータにグループ数 g というパラメータがさらに追加されている．グループ数 g については，4.2 節においてグループ数 g と解探索性能の関係について調査を行う．

4.2 解探索性能の評価

2018 年度の Graph Golf が出題した問題 [3] の中から $(n, d) = (72, 4), (256, 5), (256, 10)$ の 3 つの問題を用いて，提案手法の解探索性能を評価する．3 つの問題に設定した最高温度は，それぞれ 238.91，452.43，79.63 である．最低温度は，すべての問題で 0.22 である．

各問題において，グループ数 g を変えて提案手法を 100 回ずつ実行した．その結果を**図 12** に示す．縦軸の ASPL Gap とは，提案手法が最終的に得た値と ASPL の下界 $L_{n,d}$ との差であり，最良値は 0 である．図 12 では，各グループ数 g における ASPL Gap の最悪値，平均値，最良値の 3 つを示している．図 12 の結果より，グループ数 g の値が大きくなるほど，解探索性能が高くなる傾向があることがわかった．ただし，**図 12a** では $g = 12$ よりも $g = 9$ の方が解探索性能は高いため，グループ数 g の値が大きすぎると解探索性能が劣化する場合があることもわかった．各問題と ASPL Gap の平均値が最も良かったグループ数 g との組は，$(n, d, g) = (72, 4, 9)$，$(256, 5, 32)$，$(256, 10, 16)$ であった．

図 12a において，グループ数 g が大きい場合に提案手法の解探索性能が劣化した理由について考察する．提案手法のパラメータであるグループ数 g は，そのグラフの規則性を強さを示しており，グループ数 g が大きくなるほど規則性は強くなる．2.2 節で述べたように，規則的なネットワークの ASPL よりもランダムグラフの ASPL の方が短い傾向にあるため，図 12a ではグループ数 g が大きい場合に性能が劣化したと考えられる．

4.3 一定温度 SA

前節までに示したように，通常の SA では温度を高温から低温まで変化させる温度スケジュールが用いられる．これに対し，解探索開始から終了までを一定の温度とする温度スケジュールを用いることにより，解探索性能が上がる場合があることが報告されている [20]．一定の温度スケジュールを用いた SA を一定温度 SA と呼ぶ．一定温度 SA では最高温度 T と最低温度 C は同じ値であるため，式 (10) のクーリング率 α の値は 1 になる．そこで，本節では前節で用いた問題を用いて，一定温度 SA と通常の SA との解探索性能の比較を行う．

表 1: 一定温度 SA と通常の SA との比較

(n, d, g)	Optimum solutions		Calculations	
	Fixed Temp.	Normal	Fixed Temp.	Normal
(72, 4, 9)	100/100	70/100	92,564	621,272
(256, 10, 16)	77/100	17/100	442,372	921,633

表 2: COMA システム

CPU	Intel Xeon-E5 2670v2, 10 Cores × 2 Sockets
Memory	DDR3-SDRAM, 1866MHz, 59.7GB/s, 64GB
Network	InfiniBand FDR, 7GB/s
Software	intel/18.0.1, intelmpi/2018.1

一定温度 SA で用いる温度の候補として，前節で設定した各最高温度と最低温度を等比的に 100 個に分割し，それぞれの温度で 100 回ずつ実行した．その結果を**図 13** に示す．上のグラフの横軸の $n = 0$ は最高温度であり，$n = 99$ は最低温度である．また，上のグラフから平均解が最も良い温度の周りの温度を拡大したものを下のグラフに示している．下のグラフにおいて，点線は図 12 における各問題の最良のグループ数 g の場合の最悪値，平均値，最良値を示している．図 13 より，温度パラメータ n が最適およびその周辺であれば，一定温度 SA の方が高い解探索性能を示すことを確認した．

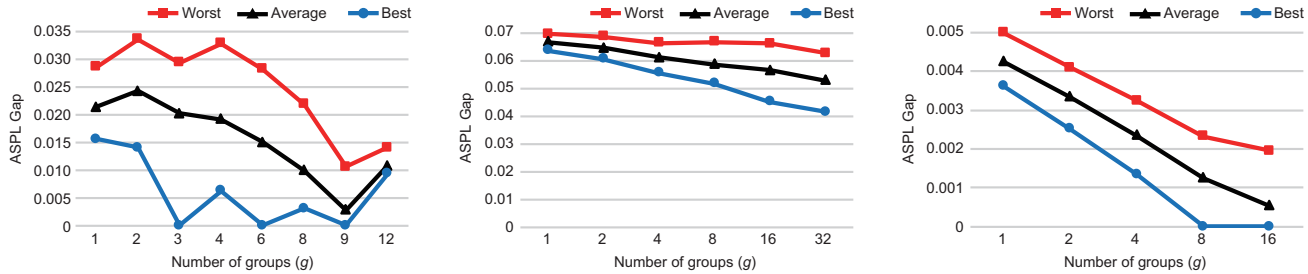
また，解探索性能を示す別の指標として，通常の SA と温度一定 SA の両方で最適解を発見できた $(n, d, g) = (72, 4, 9)$ と $(256, 10, 16)$ において，それぞれの最適解発見回数と，その最適解発見に必要だったエネルギー計算回数の平均値を**表 1** に示す．温度一定 SA の値は，最も平均値が良かった温度の結果を示している．表 1 より，一定温度 SA の方が最適解発見回数は多く，エネルギー計算回数は少ないことを確認した．

一定温度 SA において解探索性能が強くなる温度の範囲は，4.1 節で説明した最高温度 T と最低温度 C の間にあると考えられる．図 13 から，一定温度 SA の解探索性能と温度の関係は比較的単純な単峰性問題とみなすことができる．このことから，簡易な最適化手法を用いることで，一定温度 SA の解探索性能が強くなる温度を特定できると考えられる．その特定方法については今後の課題である．

5. 高速性の評価

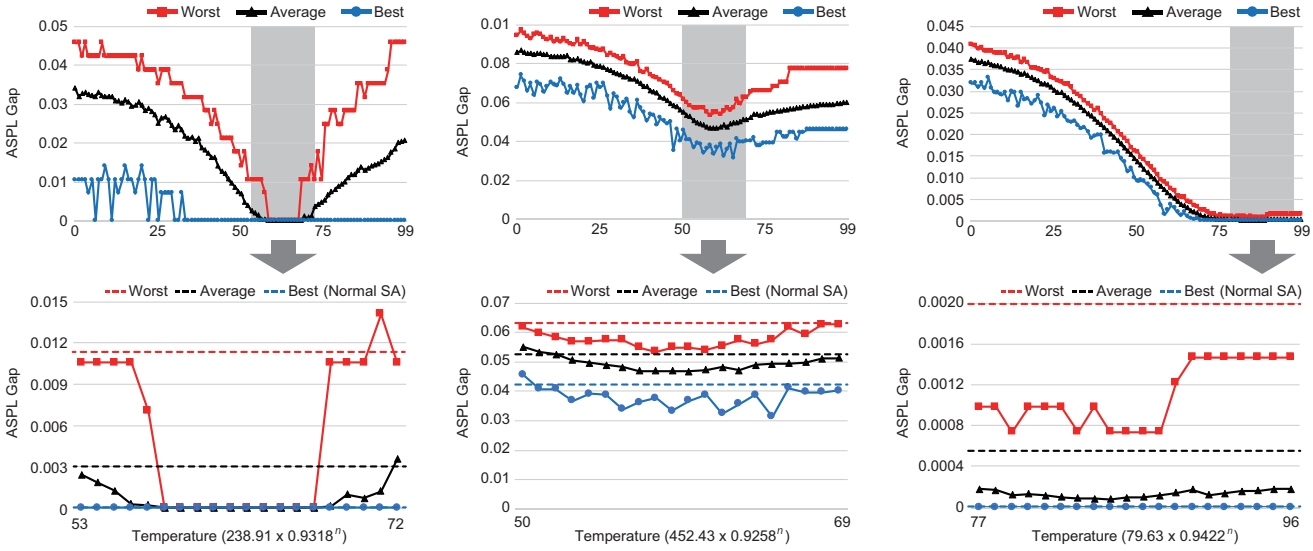
5.1 グラフの対称性の利用

3.3.4 節で述べたように，グラフの対称性を利用することにより，ASPL の計算量を削減することができる．本節では，$(n, d) = (72, 4)$，$(256, 5)$，$(256, 10)$ の問題を用いて，グループ数 g と ASPL の計算時間の関係について調べる．



(a) $n = 72$, $d = 5$　(b) $n = 256$, $d = 5$　(c) $n = 256$, $d = 10$

図 12: グループ数 g を変えた場合の解探索性能の結果



(a) $n = 72$, $d = 4$, $g = 9$　(b) $n = 256$, $d = 5$, $g = 32$　(c) $n = 256$, $d = 10$, $g = 16$

図 13: 一定温度 SA の解探索性能の結果

ASPL を求めるために用いた幅優先探索のアルゴリズムは 5.2 節で説明を行う．評価には筑波大学の COMA システムを用いた．その仕様を**表 2** に示す．コンパイラに用いた最適化オプションは “-O2” であり，1 プロセス 1 スレッドで実行を行った．

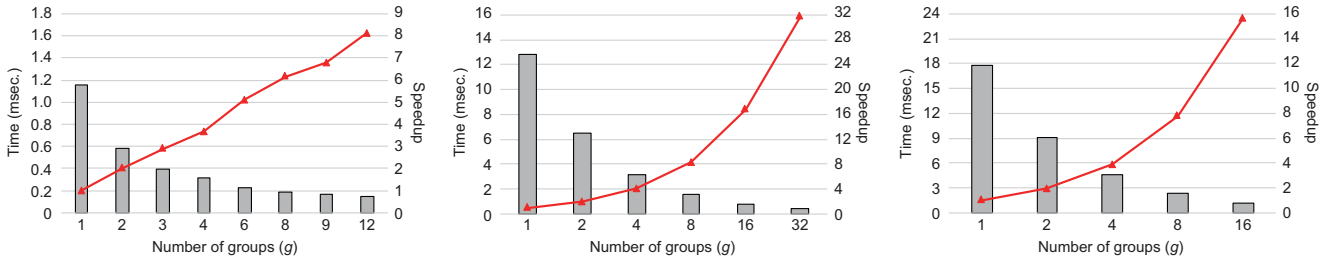
各問題における 100 回の ASPL の計算時間を**図 14** に示す．棒グラフは左縦軸で時間を，線グラフは右縦軸で $g = 1$ とのスピードアップ比を示している．図 14 の結果より，$(n, d, g) = (72, 4, 12)$，$(256, 5, 32)$，$(256, 10, 16)$ において，それぞれ 8.11 倍，31.76 倍，15.67 倍の高速化を達成した．$(72, 4)$ でややスケールしていない理由は，1 回あたりの幅優先探索の時間が他の問題と比較して短いため，グループ数 g に関係のない初期化など処理に要する時間が影響したからであると考えられる．

5.2 MPI/OpenMP ハイブリッド並列化

本節では，本手法の APSL の計算に対して MPI と OpenMP を用いたハイブリッド並列化について説明し，その高速性の評価を行う．ハイブリッド並列化の概要としては，MPI を用いて複数の幅優先探索を同時に実行し，さらにその各幅優先探索を複数のスレッドを用いて並列実行する．

3.3.4 節で述べたように，ASPL を求めるためには (n/g) 個の頂点から幅優先探索を行う必要がある．その幅優先探索は並列に実行することが可能であるため，その出発点となる頂点を各 MPI のランクに割り当てることにより並列化を行う．各ランクでは，幅優先探索を用いて割り当てられた頂点と他の頂点間の距離を計算し，その合計値を *MPI_Allreduce()* を用いて集約することにより，ASPL を計算することができる．そのため，MPI のランク数 P の上限は (n/g) である．また，幅優先探索時に対象となるグラフの連結性についてチェックを行い，その結果も *MPI_Allreduce()* を用いて集約している．

ASPL の計算のために行う幅優先探索には，広く利用されている Level-synchronized BFS の Top-down アプローチ [21] を用いた．その擬似コードを**図 15** に示す．さらに，図 15 をスレッド並列化した擬似コードを**図 16** に示す．Level-synchronized BFS では，ある頂点集合 *frontier* からエッジを 1 つ辿ることで新たに訪問可能となる頂点の集合 *next* を求めて次の処理の入力とする，という処理を繰り



(a) $n = 72$, $d = 4$　(b) $n = 256$, $d = 5$　(c) $n = 256$, $d = 10$

図 14: グループ数 g と ASPL の計算時間の関係

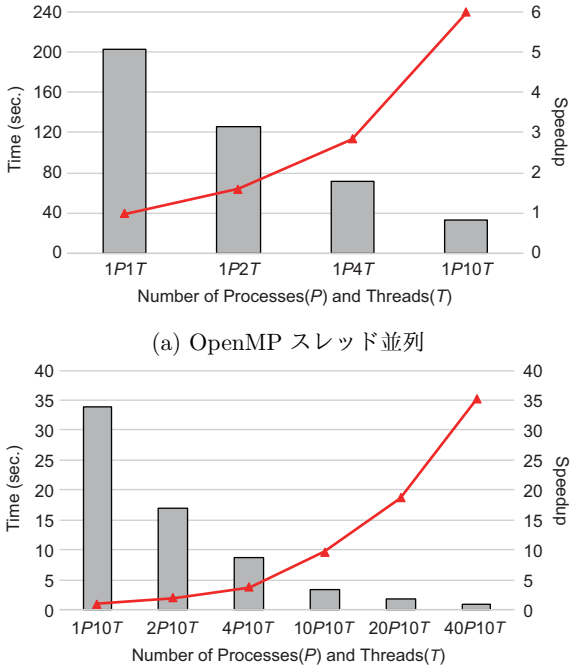
```
1  function BreadthFirstSearch(vertices, source)
2    frontier ← {source}
3    next ← {}
4    parents ← {−1, −1, ..., −1}
5    while frontier ≠ {} do
6      TopDownStep(vertices, frontier, next, parents)
7      frontier ← next
8      next ← {}
9    end while
10 end function
11
12 function TopDownStep(vertices, frontier, next, parents)
13   for v ∈ frontier do
14     for n ∈ neighbors(v, vertices) do
15       if parents[n] == −1 then
16         parents[n] ← v
17         next ← next ∪ {n}
18       end if
19     end for
20   end for
21 end function
```

図 15: 幅優先探索の疑似コード

```
1  function TopDownStep(vertices, frontier, next, parents)
2    count = 0
3    for v ∈ frontier omp parallel do
4      local_next ← {}
5      for n ∈ neighbors(v, vertices) do
6        if compare_and_swap(parents[n] == −1, parents[v]+1)
             then
7          local_next ← local_next ∪ {n}
8        end if
9      end for
10     omp critical
11       next[count] ← local_next
12       count += get_count(local_next)
13     end omp critical
14   end for omp parallel
15 end function
```

図 16: スレッド並列化した幅優先探索の疑似コード

返すことで幅優先探索を行う．図 16 の 3 行目では，始点となる頂点集合 *frontier* の探索を各スレッドが分割して行っている．図 16 の 6 行目では，探索候補点が既探索かどうかのチェックを行っている．この操作はバリア同期が必要であり，具体的には関数*__sync_bool_compare_and_swap()*



(a) OpenMP スレッド並列

(b) MPI/OpenMP ハイブリッド並列

図 17: 高速性の評価（$n = 400000$，$d = 32$，$g = 10000$）

を用いて実装している．図 16 の 10–13 行目では，各スレッドが発見した新たに訪問可能となる頂点の集合 *local_next* を *next* に格納している．

MPI/OpenMP ハイブリッド並列化を行った本手法を用いて，ASPL に要する時間を計測する．計測環境は前節と同様である．このハイブリッド並列化は問題サイズが大きい場合に高い並列化効率が望めるため，2018 年度の Graph Golf が出題した最大の問題である $(n, d) = (400000, 32)$ を用いた．グループ数 g は 10000 と設定した．そのため，最大の MPI ランク数は 40 である．1 つの MPI ランクには 1 つの CPU を割り当てる．COMA システムの各計算ノードは 2 つの CPU を持つので，最大 20 計算ノードを用いた計測を行う．また，スレッドについては，1 つの CPU につき最大 10 スレッドを割り当てた．

100 回の ASPL の計算時間を**図 17** に示す．**図 17a** は CPU 1 ソケット内における OpenMP スレッド並列化のみの結果であり，**図 17b** は MPI/OpenMP ハイブリッド並

列の結果である．図中の P はプロセス数であり，T はスレッド数である．図 17a において，$1P1T$ と $1P10T$ との性能を比較すると，5.97 倍の性能向上を達成した．バリア同期を行っているため，スレッド数に対して性能はややスケールしないことがわかる．図 17b において，$1P10T$ と $40P10T$ との性能を比較すると，35.12 倍の性能向上を達成した．本手法で発生する通信は 1 要素の *MPI_Allreduce()* を 2 回行っているのみであるため，全計算時間に対する通信時間の割合は小さい．そのため，MPI による並列化効率は非常に高くなったと考えられる．最終的に，$1P1T$ と $40P10T$ との性能を比較すると，209.80 倍（$= 5.9744 \times 35.1161$）の性能向上を達成した．なお，グラフの対称性を利用した高速化も適用しているため，グループ数 g が 1 の場合と比較すると，約 2,098,000 倍の性能向上を達成していると考えられる．

6. まとめと今後の課題

本稿では，様々なネットワークの通信遅延を削減するため，SA をベースとする Order/Degree 問題に対する新しい解法を開発した．本手法は，グラフのトポロジに対称性を持たせることにより，ASPL の小さいトポロジを効率よく発見できる．2018 年度の Graph Golf で出題されている $(n, d) = (72, 4)$，$(256, 5)$，$(256, 10)$ の問題に対して本手法を適用した結果，$(72, 4)$ と $(256, 10)$ の問題では最適解を発見し，$(256, 5)$ の問題でも ASPL の十分小さいネットワークトポロジを発見した．

また，グラフのトポロジの対称性を利用することにより，計算時間の削減を行った．その結果，$(n, d, g) = (72, 4, 12)$，$(256, 5, 32)$，$(256, 10, 16)$ において，それぞれ 8.11 倍，31.76 倍，15.67 倍の高速化を達成した．さらに，MPI と OpenMP を用いたハイブリッド並列化を行うことにより，さらなる計算時間の削減を行った．その結果，$(n, d, g) = (400000, 32, 10000)$ において，209.80 倍の性能向上を達成した．この 2 つの高速化手法は組合せることができるため，最終的には約 2,098,000 倍の性能向上を達成していると考えられる．

今後の課題として下記が挙げられる．(1) 本稿で解探索を行った以外の Graph Golf が提出している問題についても，本手法の解探索性能について調査する．(2) 3.3.6 節から，頂点数が素数の場合は本手法の特徴であるグラフの対称性が利用できないことがわかる．その問題を克服するため，グラフの中心に頂点を追加することにより，頂点数が素数であっても対称性を得ることができる手法を開発している．(3) 4.3 節で述べたように，一定温度 SA のための最適な温度を自動的に発見する手法について開発する．(4) GA などの SA とは解探索のメカニズムが異なる進化的計算手法を用いて，解探索性能の比較を行う．特に，GA の解探索の本質は交叉であるため，Order/Degree 問題のように部分解を持っている問題については有効であると考えられる．(5) さらなる高速化のために，GPU などのアクセラレータを用いた実装を行う．現在，著者らが開発しているマルチ GPU を扱えるように拡張した OpenMP 構文 [22] を用いた幅優先探索のコードを開発している．また，本手法では 1 つの幅優先探索の対する並列化は OpenMP スレッド並列化のみであったが，その幅優先探索にも MPI/OpenMP ハイブリッド並列化を行うことにより，さらなる高速化が可能になると考えられる．(6) プログラミングの生産性を上げるため，最新の並列言語である XcalableMP を用いた実装も行っている．その予備調査の結果は文献 [23] が詳しい．

Acknowledgements

This research used the coma system provided by Interdisciplinary Computational Science Program in the Center for Computational Sciences, University of Tsukuba. This work was supported by JSPS KAKENHI Grant Number 18K11331.

参考文献

[1] 藤原一毅, 藤田聡, 中野浩嗣, 井上武, 鯉渕道紘. みんなで order/degree 問題を解いて究極の低遅延相互結合網をつくろう．電子情報通信学会技術研究報告 信学技報, Vol. 115, No. 174, pp. 223–228, Aug 2015.

[2] Michihiro Koibuchi and Ikki Fujiwara and Kiyo Ishii and Shu Namiki and Fabien Chaix and Hiroki Matsutani and Hideharu Amano and Tomohiro Kudoh. Optical network technologies for HPC: computer-architects point of view. *IEICE Electronics Express*, Vol. 13, No. 6, pp. 20152007–20152007, 2016.

[3] Graph Golf: The Order/degree Problem Competition. `http://research.nii.ac.jp/graphgolf`.

[4] Floyd, Robert W. Algorithm 97: Shortest Path. *Commun. ACM*, Vol. 5, No. 6, pp. 345–, June 1962.

[5] Warshall, Stephen. A Theorem on Boolean Matrices. *J. ACM*, Vol. 9, No. 1, pp. 11–12, January 1962.

[6] R. Seidel. On the All-Pairs-Shortest-Path Problem in Unweighted Undirected Graphs. *Journal of Computer and System Sciences*, Vol. 51, No. 3, pp. 400–403, 1995.

[7] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward 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, and M. P. Vecchi. Optimization by simulated annealing. *Science*, Vol. 220, No. 4598, pp. 671–680, 1983.

[9] V. G. Cerf, D. D. Cowan, R. C. Mullin, and R. G. Stanton. A lower bound on the average shortest path length in regular graphs. *Networks*, Vol. 4, No. 4, pp. 335–342, 1974.

[10] Shin, Ji-Yong and Wong, Bernard and Sirer, Emin Gün. 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.

[11] M. Koibuchi and H. Matsutani and H. Amano and D.

F. Hsu and 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.

[12] H. Matsutani and M. Koibuchi and I. Fujiwara and T. Kagami and Y. Take and T. Kuroda and P. Bogdan and R. Marculescu and H. Amano. Low-latency wireless 3D NoCs via randomized shortcut chips. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, March 2014.

[13] Ankit Singla and Chi-Yao Hong and Lucian Popa and Philip Brighten Godfrey. Jellyfish: Networking Data Centers Randomly. *CoRR*, Vol. abs/1110.1687, , 2011.

[14] Teruaki Kitasuka and Masahiro Iida. A Heuristic Method of Generating Diameter 3 Graphs for Order/Degree Problem. *CoRR*, Vol. abs/1609.03136, , 2016.

[15] R. Mizuno and 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.

[16] Nobutaka Shimizu and Ryuhei Mori. Average Shortest Path Length of Graphs of Diameter 3. *CoRR*, Vol. abs/1606.05119, , 2016.

[17] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1992.

[18] David E. Goldberg. Genetic Algorithms in Search Optimization and Machine Learning. *AI Magazine*, Vol. 12, pp. 102–103, 1989.

[19] NetworkX. `https://networkx.github.io`.

[20] Fielding, M. Simulated Annealing With An Optimal Fixed Temperature. *SIAM Journal on Optimization*, Vol. 11, No. 2, pp. 289–307, 2000.

[21] Scott Beamer, Krste Asanović, and David Patterson. Direction-optimizing breadth-first search. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '12, pp. 12:1–12:10, Los Alamitos, CA, USA, 2012. IEEE Computer Society Press.

[22] 中尾昌広, 村井均, 佐藤三久. PGAS モデルによるマルチ GPU 対応 OpenMP コンパイラ. 情報処理学会研究報告, No. 40, pp. 1–7, Jul 2018-HPC-165.

[23] Nakao Masahiro and Murai Hitoshi and Boku Taisuke and Sato Mitsuhisa. Linkage of XcalableMP and Python Languages for High Productivity on HPC Cluster System: Application to Graph Order/Degree Problem. In *Proceedings of Workshops of HPC Asia*, HPC Asia '18, pp. 39–47. ACM, 2018.