一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
CPSY2018-110,DC2018-92（2019-03）

Approximate 相互結合網を用いた巡回セールスマン問題の並列蟻コロニー最適化による解法の高速化

丹羽　直也†　平澤　将一††　鯉渕　道紘††　天野　英晴†

†† 国立情報学研究所
† 慶應義塾大学
E-mail: †{naoya,hunga}@am.ics.keio.ac.jp, ††{hirasawa,koibuchi}@nii.ac.jp

あらまし　我々は，通信路上で一定のビット誤りを許容することで，相互結合網の広帯域化と低遅延化を両立し，その結果，データセンターやスーパーコンピュータにおいて並列計算アプリケーション高速化を達成する Approximate ネットワークを提案している．誤りを許容する相互結合網を活用した上で正しい (許容誤差範囲内の) 計算結果を得られるかどうかは，アプリケーションの性質に依存する．本報告では，ケーススタディとして，NP 困難である巡回セールスマン問題の近似解を並列蟻コロニー最適化で求める手法を用いて Approximate ネットワークの高速化を効果を評価した．その結果，解の精度にほとんど影響を与えずに，問題サイズが 100 以上の場合は 5% 前後の高速化ができることがわかった．

キーワード　Approximate Computing, NP 困難, 相互結合網, 光通信, シミュレーション

Accelerating solution method by parallel ant colony optimization of traveling salesman problem using Approximate interconnection network

Naoya NIWA†, Shoichi HIRASAWA††, Michihiro KOIBUCHI††, and Hideharu AMANO†

†† National Institute of Informatics
† Keio University
E-mail: †{naoya,hunga}@am.ics.keio.ac.jp, ††{hirasawa,koibuchi}@nii.ac.jp

1. はじめに

長距離通信のみならず，データセンターやスーパーコンピュータのような室内の短距離通信においても，今後多値変調による通信の広帯域化が見込まれている．例えば，従来は一度に 1 ビット (2 値) ずつ変調する on/off keying (OOK) が用いられていたが，同レンジのイーサネット規格である 400GBASE-DR4 と 200GBASE-DR4 では，Pulse Amplitude Modulation (PAM) により一度に 2 ビット (4 値) 変調する．今後は，多値数が益々大きくなることが想定される．多値変調をした場合，そのままではビット誤り率 (BER) が悪化するため，強力なエラー検出訂正により BER を 10^{-12} 以下という，およそ “エラーフリー” にする必要がなる．そのために，従来は Cyclic Redundancy Check (CRC) を用いた再送で十分な信頼性を確保できた．しかし，今後は CRC では足りず，FEC (Forward Error Correction) が変調するごと，すなわち hop-by-hop で必要となる．

スーパーコンピュータの設計では，超並列処理をサポートするために通信遅延を小さくすること (例えば，10 万ノードにおいて最長通信遅延 1 μ 秒) が重要である．しかし，典型的な FEC を実装する場合，大きい通信データのバッファリングが必要となる．例えば，switch-to-switch BASE-R FEC では 2112 ビットを格納するために 84 ナノ秒 [1]，InfiniBand では 120 ナノ秒 [2] と見込まれることが報告されている．この見積より，1 本のリンク経由毎に行う FEC 処理遅延がスイッチ処理遅延 (例：Anton-2 や BlueGene/Q ではスイッチ遅延が約 40 ナノ秒) と比べて大きくなることになる．これはデータセンター，スパコンにおいて，通信の信頼性，高バンド幅，低遅延性すべての面を満たすように改良することが難しくなってきていることを意味している．しかし，計算ノードの処理能力の劇的な向上により広バンド幅通信の実現は常に要求され，かつ，1 台のスパコンを構成するプロセッサコア数の激増 (現状，数百万コア数/台) により，通信粒度が小さくなり，その結果，通信遅延を小さくすることが益々重要となっている．

そこで，我々は低精度の通信を許容することにより広バンド

幅低遅延を達成する Approximate ネットワークを提案してきた [3]．Approximate ネットワークでは，エラーフリー(注1)伝送を前提としない．そのため，多値変調による光リンクのバンド幅を 2〜10 倍向上させることが期待できる一方，FEC によるエラー検出訂正を行わないことで 1 ホップ遅延を現状維持する．

本報告では，ケーススタディとして，NP 困難である巡回セールスマン問題の近似解を並列化された蟻コロニー最適化で求める手法を選び，Approximate ネットワークの高速化の効果を評価する．我々は，精度に関して並列化された蟻コロニー最適化を，エラーが生じないハードウェアを用いたときの近似解と，Approximate ネットワークを用いて実行したときの解との乖離に基づいて議論する．また，実行時間に関しては，Approximate ネットワークによる高速化をシミュレーションして得られた結果により議論する．そして，最終的に精度と実行時間のトレードオフについて，Approximate ネットワークがほど良いバランスであることを示す．

本報告は，2.節において Approximate ネットワーク要素技術などの関連研究について述べ，3. 節において巡回セールスマン問題を対象とした理由を示し，4. 節において並列蟻コロニー最適化を Approximate ネットワーク上で実行する手法を述べ，5. 節において評価の諸条件とその結果を示し，6. 節にて結論と今後の課題を述べる．

2. 技術的背景

2.1 ベースラインの相互結合網

本報告では，電気スイッチ間をアクティブオプティカルケーブル (AOC) で接続したネットワークをベースラインとする．現状実現可能なネットワークにおいて，リンク帯域は 100Gbps (25Gbps×4) とし，OOK により変調する．転送データのビット列に対して CRC を用いてエラー検出，再送し，信号からビット列への変換には Gray コードを用いる．そして，浮動小数のデータ転送の場合，IEEE754 浮動小数点フォーマットに沿ってビット列を順に格納することで数値化する．

2.2 Approximate ネットワーク

光リンクと電気スイッチで構成される現状のスパコン，データセンターを対象とした，Approximate ネットワークについて述べる．本研究で用いる Approximate ネットワークは文献 [3] によるものである．また，報告 [4] と同じ仮定である．実装手法のさらなる詳細に関してはこれらを参照いただきたい．

2.2.1 物理/リンク層

400GbE といった広帯域通信規格では，多値変調が用いられる見込みである．多値変調のトレードオフとして，2 値変調と比べて同一信号パワーでは SNR（Signal to Noise Ratio）が劣化するため，受信側でのビットエラー率悪化につながる点がある．Approximate ネットワークではこの点を利用する．

我々が想定する Approximate ネットワークでは最大 1,024-QAM (Quadrature Amplitude Modulation)，あるいは PAM により 1 シンボルあたり最大 10 ビット転送することで 1Tbps(250Gbps×4) リンクをベースラインとする．そして，エラー検出訂正に関して，FEC を用いない．ただし，この場合，我々が想定する変調速度 250Gbps の場合ではリンク上の伝送においてエラーフリー ($< 10^{-12}$) を達成できなくなる．なお，信号からビット列への変換については，現状通り，信号の隣接符号間のハミング距離を 1 とする Gray コードを想定する．

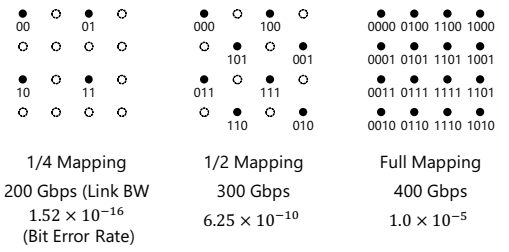


図 1: 16-QAM における 4 値 (2bit)，8 値 (3bit)，16 値 (4bit) 転送の例．16-QAM の一部のシンボルのみをデータ転送に用いることで，隣接符号間の距離が大きくなるため，ノイズ耐性が高くなる (ビットエラー率の解析は文献 [3] による．)

2.3 アプリケーション層

Approximate ネットワークでは伝送路上においてビットエラーが生じることを前提にデータ転送する．しかし，ビットエラーが生じた場合でも原則再送せず，データを受信したプロセス (プログラム) はそのまま計算処理を続行する．

そこで，ビット列から数値の符号化を工夫することで，ビットエラーがもたらす転送データ値の誤差を最小化することが望ましい．しかし，本報告で対象とする並列アプリケーションにおいてプロセス間通信の転送データのフォーマットである IEEE754 浮動小数点数表現における符号部，指数部，仮数部の各ビットエラーがもたらす数値誤差は均一ではない．例えば，符号部にエラーが生じた場合，値が反転するため，その数値の絶対値の 2 倍の誤差が生じることになる．一方，仮数部の下端ビットが反転した場合，極めて小さな値の誤差に留まる．そこで，我々はビットエラーがもたらすアプリケーションの転送データの数値への影響を最小化するように符号部，指数部の上位ビットに対して M-QAM の一部の符号のみを用いている．つまり，保護したいビット列の転送時にはビットエラー率を改善させる．図 1の例では，文献 [3] において解析したビットエラー率とバンド幅の関係を示している．

3. 巡回セールスマン問題と近似

本研究では巡回セールスマン問題 (TSP) を Approximate ネットワークで高速化する対象とする．

TSP は，工作機械のドリルでより効率的に穴を開ける順序を求める，物資の効率的な運搬経路を求めるなど実用的な応用が多くある．これらの応用において TSP を解くというのは重要ではあるが，一連の動作の一部でしかなく多くの時間を割くことができない．そのため，より短時間で解を出すことが要求される．一方で，必ずしも厳密解を必要としないことも多い．例

(注1)：本研究では，イーサネット規格に準じて，ビットエラーレート 10^{-12} 環境でのエラーフリーを指すこととする．

えば工作機械の制御の例の場合，厳密解を得ようと機械を止めて計算をするより，それなりの解をすぐに出して，作業を始めたほうが全体として高速に動作させられることになる．この様に TSP は近似解でも良いのでなるべく速く解を出すことが要求される応用例が多い．

一方，Approximate ネットワークを用いる対象を考えた場合，精度をトレードオフの対象とするため厳密解が要求される並列計算は適さない．そのため，計算対象は，NP 困難に代表される「厳密に解くことは難しいが，解の検証，近似解同士の比較が容易である」問題と，ビッグデータ，画像処理，機械学習に代表される「全体の結果の傾向」が重要な問題の 2 種類となる．この中で，本研究で対象とする TSP は前者のケーススタディとなる．

これらより，我々は Approximate ネットワークにより TSP を高速化することは，その応用例において有効な場面が多くあること考える．本研究では TSP の中でも特に応用例が多い，辺の重みが平面上の頂点間のユークリッド距離であるユークリッド TSP を対象とする．

この TSP は計算困難問題の代表例であり，現時点で多項式時間で解くアルゴリズムは発見されていない．計算困難問題一般に対して個別の問題の特徴を活かしてより速く厳密解を求めるアプローチと求めるものを近似解とすることで大規模な問題に対しても現実的な時間内に答えを出すアプローチがある．本研究では後者の近似的なアプローチに注目する．計算困難問題を近似アルゴリズムで解く際に，厳密解との誤差と計算量の関係によって問題が分類される．TSP 自体は多項式時間で定数以下の誤差の近似解を求めることができない，すなわち定数近似不可能であるとわかっている．ただし，TSP には問題の特徴によって細分化することができ，問題によって分類が変わる．

この TSP を近似的に解く方法として焼きなまし法などに代表されるメタヒューリスティクスが挙げられる．メタヒューリスティクスは広い探索空間において局所最適解に陥るのを避けながら候補解の生成と候補解の評価を繰り返しながら最適解を探す手法である．メタヒューリスティクスは解の精度保証や計算時間の保証を行うことはできないが，汎用性が高く幅広い計算困難問題に適応できる．

TSP に対しても様々なメタヒューリスティクスを適用する研究がされており，本研究ではその中でも蟻コロニー最適化を対称とした．

4. 蟻コロニー最適化と Approximate ネットワークの利用

4.1 並列蟻コロニー最適化

蟻コロニー最適化 (Ant Colony Optimization： ACO) は蟻が餌を探す仕組みを模倣したメタヒューリスティクスである [5]．蟻のコロニーでは食物を見つけた蟻がフェロモンを分泌し，そのフェロモンに惹かれて蟻が食物を探す．この時，フェロモンは時間が経つと徐々に蒸発するため，食物までの距離が遠いとフェロモンが消えていく．このフェロモンを介在した情報伝達により，最終的に巣に近い食物を効率よく得ることができる．

蟻コロニー最適化はこの蟻の行動に着想を得ているが，対象とする問題と蟻の食物探しとは別物であり，問題によってどのようにフェロモンの分泌や蒸発を再現するかが問題となる．TSP への蟻コロニー最適化の適用でも様々な手法が研究されている．

本研究では Ant System(AS) と呼ばれる Dorigo らの手法 [6] を基本とする．この手法ではそれぞれのエージェント (Ant) が探索して得た経路長に応じて式 (1,2) に従いフェロモン τ_{ij} を更新していく．なお，ij はそれぞれ問題の都市に振られた番号であり，m はエージェントの数，L_k はエージェント k が通る経路長を示す．ρ と Q はそれぞれフェロモンの蒸発と分泌に関する係数であり，本研究では $\rho = 0.2, Q = 80$ とする．

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{1}$$

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{Ant k が (i, j) を通る} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

各エージェントの経路の探索では次に向かう都市を式 (3) に基づき確率的に決定する．なお，p_{ij}^{k} は都市 i にいるエージェント k が次に都市 j を選択する確率であり，d_{ij} は都市 i,j 間の距離，N^k はその時点でエージェント k がまだ訪れていない都市の集合である．本研究では $\alpha = 1, \beta = 2$ とする．

$$p_{ij}^{k} = \begin{cases} \frac{(\tau_{ij})^{\alpha}(1/d_{ij})^{\beta}}{\sum_{l \in N^k}(\tau_{il})^{\alpha}(1/d_{il})^{\beta}} & \text{if } l \in N^k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

蟻コロニー最適化の並列化による高速化手法も研究されている．本研究では過去に提案された MPI を用いた並列蟻コロニー最適化 [7] を Approximate ネットワークで動作させる．この手法ではそれぞれのノードで蟻コロニー最適化を行い得られた最適経路を集約し，その中での最適経路をさらに配布する．その後，フェロモンの状態を初期化し最適経路にのみフェロモンを追加する．本研究ではこの最適経路の共有を Approximate することで，実行時間と実行結果にどのような影響が及ぶかを調査した．

4.2 Approximate ネットワーク上での蟻コロニー最適化

本研究で用いる並列蟻コロニー最適化では，最適経路の共有は図2に示す手順で行われる．まずは，MPI_Gather で各々のノードでの最短経路を rank 0 に集約する．rank 0 で集めた最短経路の中で最も短い経路を最適経路とし，最適経路を MPI_Bcast で全ノードに配布する．この後，ノード毎に蟻コロニー最適化を継続する．

ここで共有される最適経路は途中の都市の番号の並びを 32bit 整数の配列として転送され，要素ごとに下位ビットの反転を許容して転送することで，伝送遅延の低下などを狙う．これは2. 2節で述べた仕組みで実現され，エラーを許容するビット幅を大きくすれば伝送遅延を低下させることができる．

本研究では rank 0 に最短経路を集約するときにのみエラー

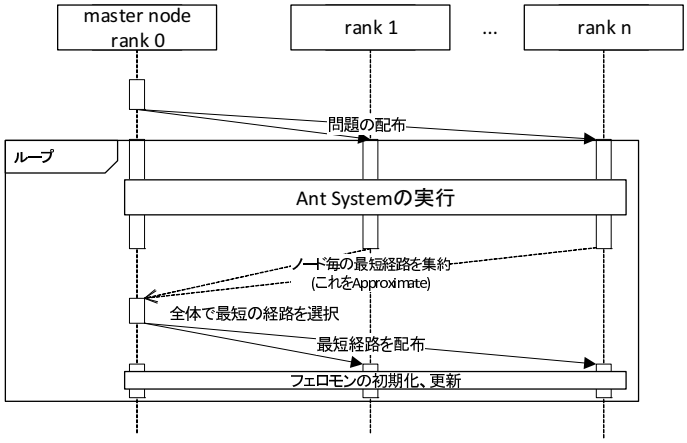


図 2: 並列蟻コロニー最適化の流れ

を許容した．これは繰り返し行われるノード間通信の半分のトラフィックに相当する．Approximate されたデータの受信側である rank 0 では受信したデータのエラーチェックを行う．これは受信したそれぞれの都市番号に対して都市の数 n で剰余を取り，都市が重複していないか確認するものである．これにより，巡回セールスマン問題の解として不正なものを取り除く．また，最初に 1 回以上は Approximate せずに最短経路を集約することで，プログラム全体で何らかの解を提示できるようにした．

5. 評　　価

5.1 評価環境

評価には先行研究 [3] でカスタムされた SimGrid v3.12 [8] を用いる．SimGrid は大規模分散システムの挙動を再現するシミュレータであり，MPI を用いた並列プログラムを設定した仮想の分散システムで動かすことができる．先行研究 [3] はこの MPI 通信の部分をカスタムし，指定したデータ型 (MPI_Datatype) に対して2. 2節で述べた手法を適用し Approximate ネットワークのシミュレーションを再現しており，本研究ではこの機能を用いて評価を取る．なお，MPI でのデータ型はプログラミング言語 (今回は C++) における型と一致している必要はなく，通信で送るデータサイズが一致していれば問題なく動作する．またサイズなどを指定して任意の名前の型を定義でき，これを利用して都市の巡回経路に対してつける名前を付け替えて Approximate の対象とするどうかを指定する．

本研究ではネットワークに接続するホスト数を 64 台とし，すべてのホストを同じスイッチに接続することで，互いに同じ帯域幅と遅延時間とする．それぞれのホストではプロセスを 1 つ実行する．これはなるべく対称的な構成とすることで，Approximate ネットワークが結果に与える影響をわかりやすくするためである．

本研究で用いた Approximate ネットワークではアプリケーション側でのデータ型に対して下位何 bit までのビットエラーを許容するかを指定する．本研究ではエラーを許容する下位 bit を 2bit,4bit,6bit,8bit の 4 パターン用意して評価した．通常通りデータが転送される場合の帯域幅は 100Gbps，遅延は 160nsec，Approximate 対象のデータが転送される場合の帯域幅は 1Tbps，遅延は 64nsec とする．

表 1: 評価に用いたパラメータ

ノード数	64
ノードの計算能力	100GFLOPS
シミュレーションする MPI 実装	MVAPICH2

本研究では TSPLIB [9] のうち 2 次元 TSP を抜粋して，評価プログラムの入力に使用した．使用した問題は a280, berlin52, ch130, ch150, eil101, eil51, eil76, kroA100, kroC100, kroD100, lin105, pcb442, pr1002, pr2392, pr76, rd100, st70, tsp225 である．問題名に含まれる数字は都市の数を示している．

蟻コロニー最適化に関し，各ノードで実行するエージェント (蟻) の数を 100，1 つのノード間通信の間にエージェントが経路を探索する回数は 10 とする．すなわち，通信と通信の間にシステム全体で 64 Node × 100 Ant × 10 Tour = 64000 個の経路を試行する．ノード間の通信は 100 回行い，通信のたびにその時点での最短経路を取得する．また，Approximate ネットワークを用いる際に，1 回目の通信のみエラーフリーで実行する．

蟻コロニー最適化を実施するに当たり，疑似乱数生成器のシードを同一にすることで，Approximate ネットワークが及ぼす影響を比較，分析可能にした．なお，同一のパラメータでのノード間ではシードはずらしている．その他のパラメータは表1に記す．

5.2 評価結果

まずは解の質と実行時間の関係を示す．図3の横軸に得られた経路のうち最短のものを 1 とした距離であり，縦軸は横軸の値を下回るのに掛かった時間が Approximate ネットワークによりどの程度変化したかを示す．このグラフにおける Approximate ネットワークでは下位 8bit のエラーを許容している．1.0 を下回っていれば，その解に到達するまでに Approximate ネットワークを利用したほうが早かったことを示している．なお，pr2392 は後述するエラーの影響で値がグラフから大きくはみ出ているため取り除く．

多くの線が 0.95 から 1.0 の間を推移している．これらの問題ではビットエラーを起こして使用不能になった探索結果がその時の最短距離でなかったために結果に影響を及ぼしていない．すなわち運良く Approximate の副作用を逃れ，Approximate の高速化の恩恵を数 % 受けることができたデータである．

一方，除外した pr2392 を含む一部のデータは上下に値が飛んでいる．値が上に飛び跳ねているものは，その時点で最適な経路を導き出したノードから rank 0 に経路を転送する際にエラーが発生したものである．下に飛び跳ねているものは，その直前のに上でエラーフリーで実行した場合とは異なるフェロモンの状態になり，別の局所解を導いた，もしくはソフトウェアのバグと考えられるが，どちらかの判断はできなかった．

これらより，同じ解を得るために問題によっては 5% 程度の

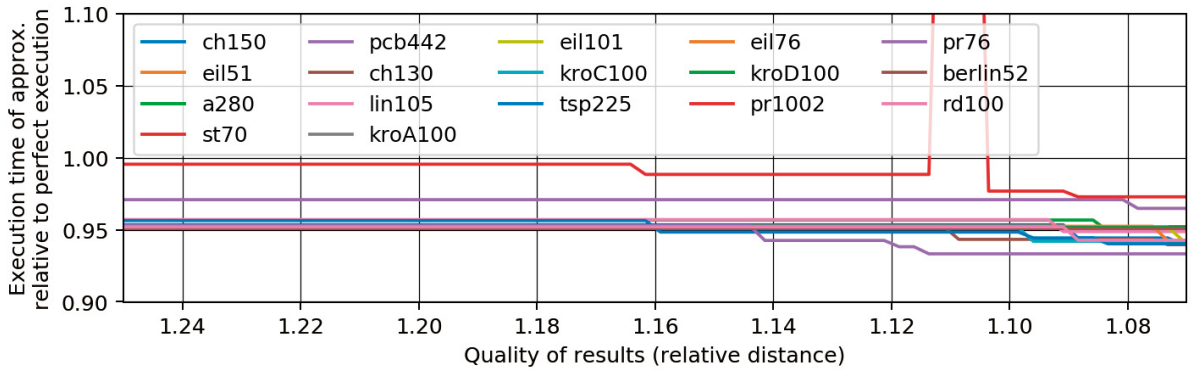


図 3: ある距離以下に解が到達した時間の比率

高速化が見込めるが，稀に大幅な時間増がありえることがわかった．

次に，問題ごとに蟻コロニー最適化を進めるに連れて，どのように見つけた最短経路が短くなっていったかを図4 に一部抜粋して示す．横軸はエラーフリーで 100 回目の通信時点を 1 とした実行時間であり，縦軸は得られた経路のうち最短のものを 1 とした距離である．

蟻コロニー最適化による探索では Ant System での探索を繰り返し，ある時点までに見つけた中で最も良い候補解がその時点での解となる．そのため，どのパターンでも階段状のグラフとして得られる．すでに見つかった最短経路よりも短い経路を発見できない場合はグラフの線は水平になる．

多くの問題では，図4の(a), (c), (d)の様に，Approximate した場合はしなかった場合と同じ推移を少し短い時間でたどっていた．これは図3において，グラフがほぼ水平だったものと同じであり，通信エラーによる最適解の転送が破壊を免れたものである．

一方，図4(b)ではエラーを許容するビット数が 6bit, 8bit であるものが，他とは異なる値を取っている．これは通信エラーによって最適解の転送が妨げられた場合である．本研究では，18 個の問題に対して 4 つのパラメータで Approximate ネットワークを実行し，そのうちこのような最適解の破壊があったのは約 11% の 8 つであった．100 回の通信を終えた段階でエラーを許容して転送されるデータの数 (バイト数ではなく MPI データ型の数) は，問題の都市数を n とすると $(64\ \mathrm{Node}-1)\times n\times 100$ である．これをすべての問題で合計すると 19,908,000 であり 8bit のエラーを許容した場合，Approximate ネットワークでの BER = 10^{-5} から 1500bit 前後が反転することになる．これだけのエラー見積もりに対して結果に影響が出ているパターンが少ないのは，本研究で行った並列化ではほとんどの MPI 通信が蟻コロニー最適化の最適解に直接貢献していないからと考えられる．この様な種類のソフトウェアでは Approximate ネットワークによるエラーが結果に与える影響は小さいと考える．

図5にエラーを許容するビット数と解の質の関係を示す．縦軸はそれぞれの問題でエラーフリーで蟻コロニー最適化を実行して得た最短経路を 1 とた時，エラーを許容するビット数を変えて得られた最短経路の比率を平均したものである．本研究の結果では変化は 0.1% を下回っており，全体的な解の質にはほとんど影響がなかったと言える．これは，前に示したデータにより多くのパターンではほとんど解が一致しているが，時たま外れた解になるということからも推測可能な結果である．

問題のサイズと Approximate ネットワークによって得られた速度向上率の関係を図6に示す．エラーを許容するビット数は 8bit，通信回数は 100 回である．問題サイズが大きくなると Approximate ネットワークによって得られる高速化の割合が小さくなっていくことがわかる．蟻コロニー最適化の各ノードでの計算量は $O(n^2)$ であるのに対し，Approximate ネットワークでの時間短縮は通信遅延低減がその主なものであり，問題サイズによる影響はほとんど無い．これらから本研究で用いた手法では大きな問題だとあまり高速化のメリットが得られなかったと考えられる．

6. おわりに

巡回セールスマン問題を近似アルゴリズムである蟻コロニー最適化で解くプログラムを Approximate ネットワーク上で実行することで，本研究で調べた範囲で解の精度にほとんど影響を与えずに，問題サイズが 100 以上の場合は 5% 前後の高速化ができることがわかった．

蟻コロニー最適化を並列化するに当たり，本研究では最適経路のみを共有するという手法を用いたため，解への影響が小さかった代わりにあまり高速化のインパクトが得られなかった．これはなるべく通信の量を抑える既存手法を元に Approximate ネットワークを適用したためである．

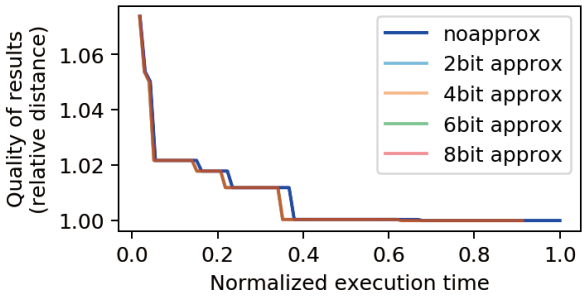
今後はフェロモン情報そのものをノード間で共有するなど，通信頻度が高いプログラムを Approximate ネットワークを利用して動作させた場合の傾向を見ていく．

文　献

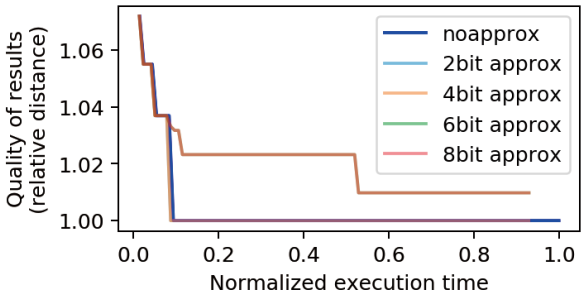
[1] M. Andrewartha, B. Booth, and C. Roth, “Feasibility and Rationale for 3m no-FEC server and switch DAC,” http://www.ieee802.org/3/by/public/Sept15/andrewartha_3by_01a_0915.pdf, Sept. 2015.

[2] Mellanox Technologies LLC, “New developments in data center cables and transceivers,” http://www.mellanox.com/data-center-cables-and-transceivers/, 2017.

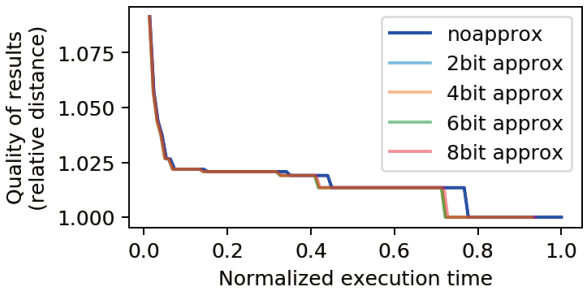
[3] D. Fujiki, K. Ishii, I. Fujiwara, H. Matsutani, H. Amano, H. Casanova, and M. Koibuchi, “High-bandwidth low-latency



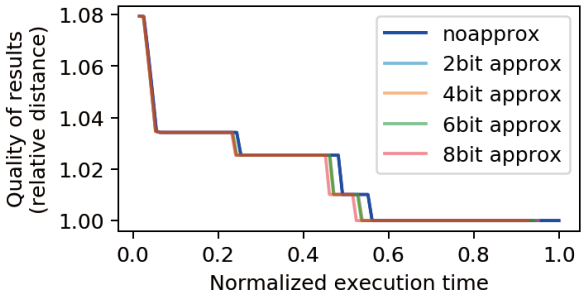
(a) eil76



(b) eil101



(c) lin105



(d) pcb442

図 4: 蟻コロニー最適化の実行時間と得られた最短経路の関係

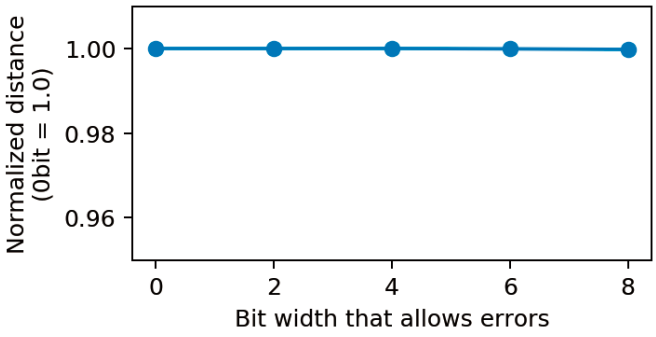


図 5: エラー許容するビット数と解の質の関係

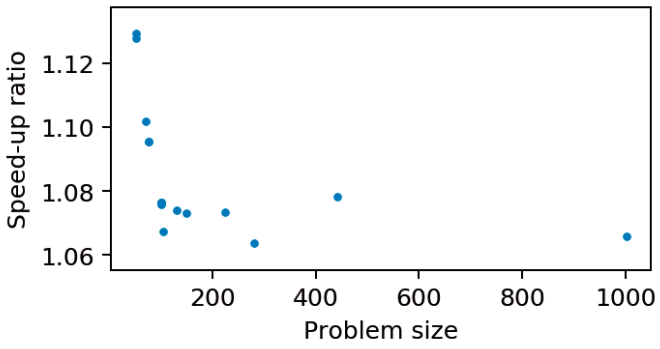


図 6: 問題の大きさと速度向上比率の関係

approximate interconnection networks," 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), pp.469–480, Feb. 2017.

[4] 平澤将一 and 鯉渕道紘 and others, "Approximate ネットワークに対する速性と計算精度の最適化基盤," 研究報告システムソフトウェアとオペレーティング・システム (OS), vol.2018, no.8, pp.1–8, 2018.

[5] M. Dorigo and M. Birattari, "Ant colony optimization," Encyclopedia of machine learning, pp.36–39, Springer, 2011.

[6] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol.26, no.1, pp.29–41, 1996.

[7] M. Manfrin, M. Birattari, T. Stützle, and M. Dorigo, "Parallel ant colony optimization for the traveling salesman problem," International Workshop on Ant Colony Optimization and Swarm IntelligenceSpringer, pp.224–234 2006.

[8] SimGrid: Versatile Simulation of Distributed Systems, http://simgrid.gforge.inria.fr/.

[9] G. Reinelt, "Tsplib–a traveling salesman problem library," ORSA journal on computing, vol.3, no.4, pp.376–384, 1991.