Vol.2013-ARC-206 No.12
2013/7/31

高次元トポロジNoCの配線長最小化手法

藤原 一毅[1,a)]　鯉渕 道紘[1,2,b)]

概要：チップ内ネットワーク (NoC) の通信遅延を削減するには，ホップ数を抑えることができる高次元なネットワークトポロジを利用することが有望である．しかし，多くの典型的な高次元トポロジは，チップ上へどのように配置するべきかが明らかでない．本報告では，高次元トポロジを持つ NoC のコア配置を二次割り当て問題としてモデル化し，総配線長を最小化するような準最適配置を求める手法を検討する．実験の結果，おおむね 512 コアまでの NoC に対し，ロバストタブーサーチ法により，準最適なコア配置を現実的な時間内で求めることができた．このとき，単純手法と比較して総配線長が最大 45%削減された．

1. はじめに

半導体技術の進歩により，多数のプロセッサコアを単一のチップ上に作り込むメニーコア SoC (Systems-on-Chip) やチップマルチプロセッサが実用化され [1–4]，その規模は 1,000 コアに達しようとしている [5,6]．これらのコア間を結合するチップ内ネットワークを NoC (Network-on-Chip) という．NoC のネットワークトポロジはチップ上の物理レイアウトを強く意識して設計されることが多い．必然的に，多くの NoC は 2 次元メッシュのような単純な構造を持つ低次元トポロジを採用している．また，Flattened Butterfly [7] のような高次元トポロジも提案されているが，その物理レイアウトは固定されている．本来，チップ上には 1 リンクあたり 32〜256 本程度の配線を作り込む余地があり，この潤沢な配線資源を活かして任意の高次元トポロジを実装できる可能性がある．特に，アプリケーションに特化した SoC では，そのアルゴリズムに適したトポロジを採用することで最適なタスクマッピングが得られ，性能向上が期待できる．しかし，そういった任意の高次元トポロジをチップ上へどのように配置するべきかは自明でなく，チップ設計者は困難な問題に直面することになる．

我々は，ネットワークトポロジがまず存在し，それに合わせてチップ上の物理レイアウトを設計するアプローチを探究する．具体的には，コア間を結合するネットワークトポロジが与えられたとき，平面チップ上の区画に各コアを割り付ける操作を二次割り当て問題としてモデル化し，メタヒューリスティクスを用いて準最適解を得る．通信遅延はコア間の距離（配線長）と密接な関係があることから，総配線長の最小化を最適化の目的関数とする．本報告の貢献は次の 3 点である．(1) おおむね 512 コアまでの NoC に対し，実用的な計算時間でコア配置を準最適化できることを示した．(2) 解の質と計算時間のトレードオフの観点から，ロバストタブーサーチ法が最も適切な解法であることがわかった．(3) ロバストタブーサーチ法を用いた場合，単純手法と比べて高次元トポロジの総配線長を最大 45%削減できることがわかった．

本報告の構成は次のようになっている．はじめに第 2 章で問題を定式化し，解法を紹介する．続く第 3 章では，解の質と計算時間の両面から各解法を評価する．第 4 章で関連研究を紹介し，最後に第 5 章で本報告を総括する．

2. 二次割り当て問題

ネットワークで結合されたコアをチップ上の区画へ割り付ける操作は二次割り当て問題 (QAP, quadratic assignment problem) としてモデル化される．本章では，その数学的表現と最適化アルゴリズムを紹介する．

2.1 定式化

いま，チップ上に N 箇所の区画があり，そこに N 個のコアを割り付けることを考える．区画 i と区画 j の物理的距離が d_{ij} で，コア i とコア j を結ぶ配線の本数が w_{ij} で，それぞれ与えられるものとする．コア i が割り付けられた区画の番号を $\phi(i)$ で表すと，目的関数である総配線長 L は

$$L = \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} d_{\phi(i)\phi(j)} \qquad (1)$$

[1] 国立情報学研究所
National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan 101-8430
[2] 総合研究大学院大学
The Graduate University for Advanced Studies
[a)] ikki@nii.ac.jp
[b)] koibuchi@nii.ac.jp

表 1　評価パラメータ
Table 1　Dimensions.

Physical dimensions		Topological dimensions				
#cores	#tiles	#cores	6-D Torus	7-D Torus	Hypercube	Random Ring [degree]
64	8×8	64	2×2×2×2×2×2	—	2×2×2×2×2×2	6
128	12×11	128	2×2×2×2×2×4	2×2×2×2×2×2×2	2×2×2×2×2×2×2	7
256	16×16	256	2×2×2×2×4×4	2×2×2×2×2×2×4	2×2×2×2×2×2×2×2	8
384	20×20	384	2×2×2×3×4×4	2×2×2×2×2×3×4	—	9
512	23×23	512	2×2×2×4×4×4	2×2×2×2×2×4×4	2×2×2×2×2×2×2×2×2	9
640	26×25	640	2×2×2×4×4×5	2×2×2×2×2×4×5	—	10

となり，この値を最小化する順列 $\Phi = \phi(1), \ldots, \phi(N)$ が QAP の最適解となる．

QAP はあらゆる組合せ最適化問題の中で最も困難であるとも言われ [8]，その ϵ-近似アルゴリズムすら存在しない[*1]ことが証明されている [9]．一般に 30 要素を超える QAP の厳密解を求めることは非現実的であり [10]，より大規模な問題を扱うには近似解法の利用が不可欠である．厳密求解が絶望的であるため，本報告では以後，「最適」という語を「準最適」と同じ意味で用いる．

2.2　解法

我々は，二次割り当て問題への適用実績がある 3 種類のメタヒューリスティクスを用いて求解を試みる．本報告で用いる具体的なアルゴリズムとその実装は次のとおりである．

- **Simulated annealing (SA)** は，徐々に下がる「温度」に従って確率的な探索を行う局所探索法の一種である．本報告では Connolly のアルゴリズム [11] の C 言語による実装 [12] を用いる．
- **Tabu search (TS)** は，探索過程で発見した解のリストを保持する局所探索法の一種である．本報告では Taillard のロバストタブーサーチ法 [13] の C++言語による実装 [12] を用いる．
- **Greedy randomized adaptive search procedure (GRASP)** [14] は, ステップごとに近似解を計算し, その中から最善の解を返す確率的反復解法の一種である．本報告では Recende の GRASP for sparse QAP [15] の Fortran 言語による実装 [16] を用いる．

また，比較のための**単純手法 (Baseline)** を以下のとおり定義する．チップ上の区画が格子状に並んでいると仮定し，(1) 各列ごとに左から右へ順番にコアを割り当てていく方法と，(2) 1 列目は左から右へ，2 列目は右から左へ，ジグザグにコアを割り当てていく方法の 2 通りを試し，どちらか総配線長が短くなった方を採用する．前者はトーラス系のトポロジに対し区画の並びに合致した配置を得る可能性があり，後者はリング系のトポロジに対し妥当な配置を得ると考えられる．

3.　評価

我々が用いる 3 種類のアルゴリズムはいずれも近似解法であり，解として得られる総配線長と求解に要する時間との間にトレードオフが存在すると考えられる．本章では，解の質と計算時間の観点から各アルゴリズムを評価する．

3.1　設定

本報告では次のように抽象化されたメニーコア SoC のモデルを想定する．**チップ**は 2 次元平面であり，その上に多数の正方形の区画が格子状に並んでいる．この区画を**タイル**と呼ぶ．例えば，8mm×8mm のチップ上に 1mm×1mm のタイルが 64 個並んでいる．1 個のタイルには 1 個の**コア**を割り付けることができる．各コアはルータを内包し（すなわち，コアとルータは一対一で存在し），ルータ同士はチップ上に形成される配線で相互結合される．本評価では物理的な長さの単位として tile を用いる．1tile はタイルの 1 辺の長さであり，上述の例では 1tile = 1mm である．

本評価で用いるパラメータを表 1 に示す．総コア数は $N \in \{32, 64, 128, 256, 384, 512, 640, 768, 896, 1024\}$，タイル数は $x \times y$（ただし $x = \lceil \sqrt{N} \rceil$，$y = \lceil N/x \rceil$）である．$N < xy$ の場合，チップの左上から順に N 個のタイルを使用し，残りの $xy - N$ 個のタイルは空き地とする．ネットワークトポロジは，n 次元トーラス ($n \in \{6, 7\}$)，n 次元ハイパーキューブ ($n \in \{6, 7, 8, 9\}$)，および d 次のランダムリング ($d \in \{6, 7, 8, 9, 10\}$) を用いる．ランダムリングとは，リングを基本として（リング自体は 2 次である），各ノードに $d - 2$ 本のランダムなショートカットリンクを追加したトポロジである [17].

我々は 2.2 節で紹介した各アルゴリズムの実装をシングルスレッドのプログラムとして実行した．使用した計算機のプロセッサは 3.47GHz Intel Xeon X5690，メインメモリは 144GB，OS は Debian GNU/Linux 7.0 である．Simulated annealing については，1 試行あたりの反復回数を 1 億回とし，10 試行の中から最善の解を選んだ．

[*1] P = NP でないかぎり．

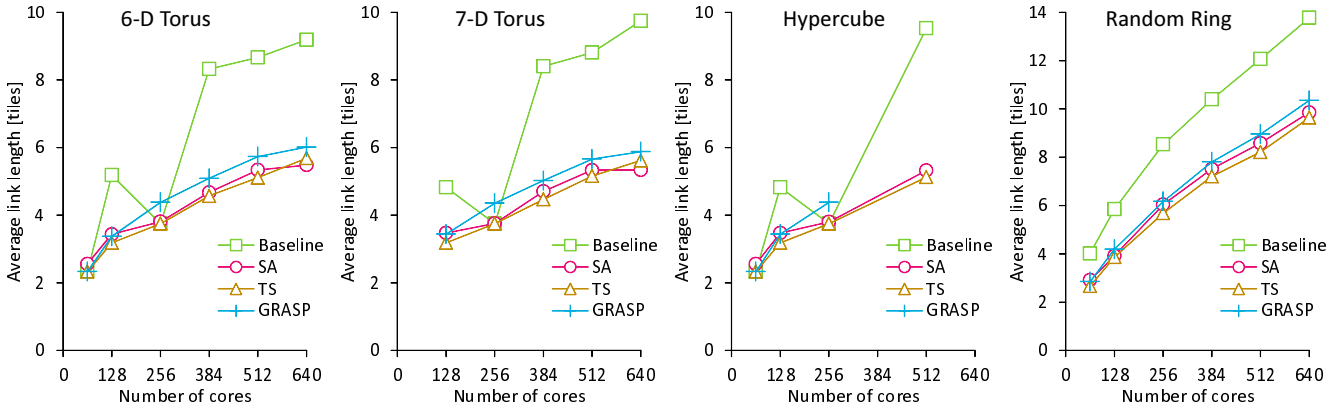

図 1 平均配線長

Fig. 1 Average link length vs. number of cores.

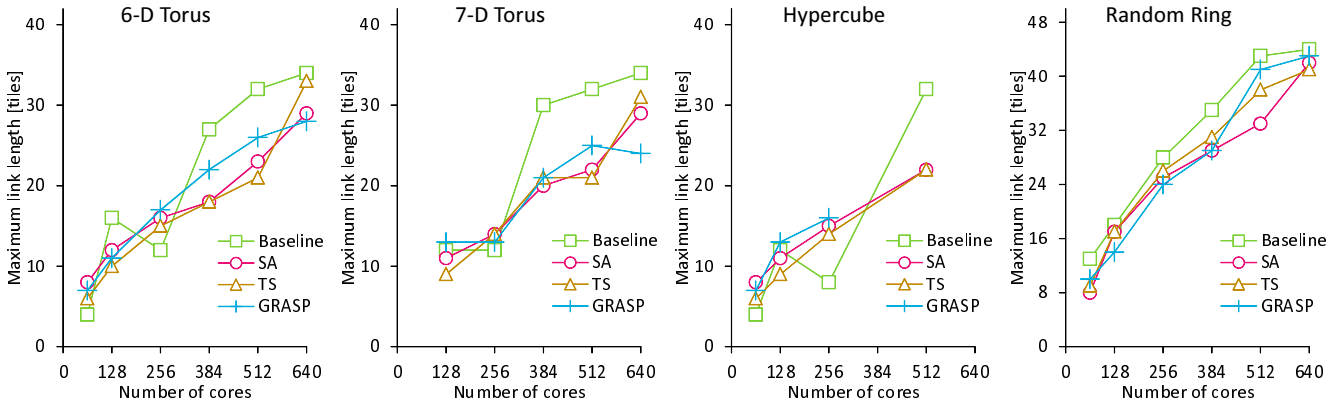

図 2 最大配線長

Fig. 2 Maximum link length vs. number of cores.

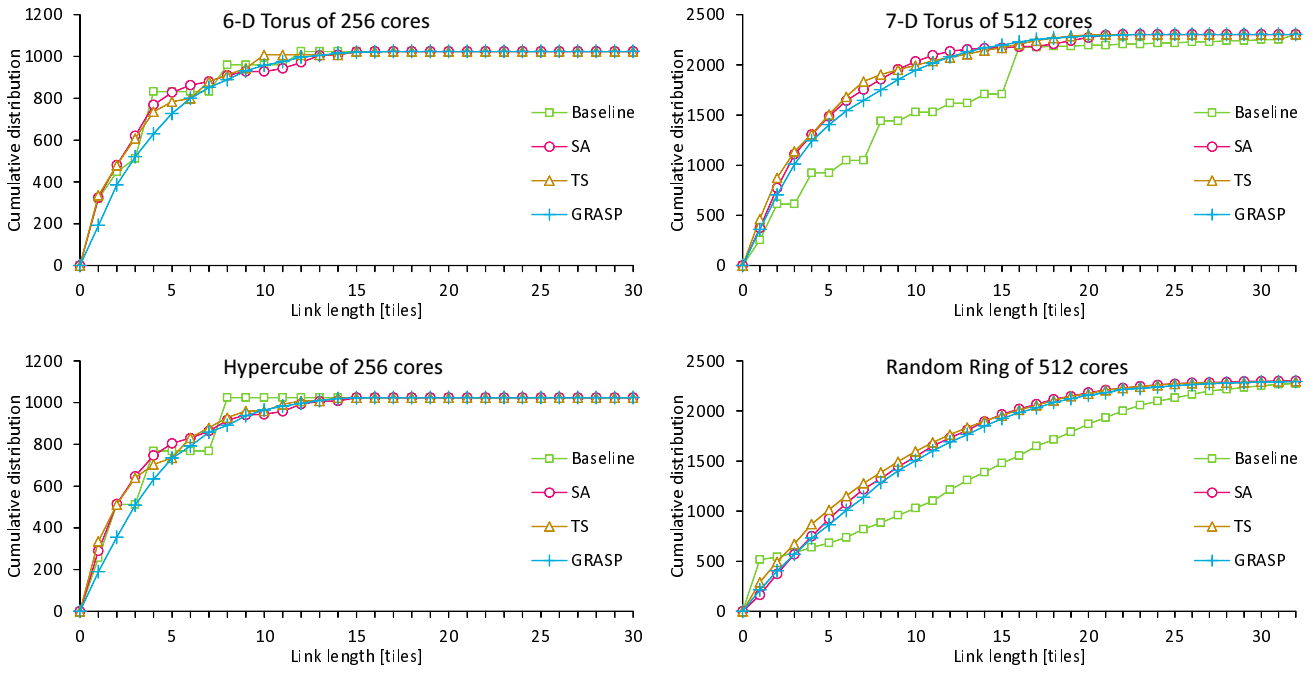

図 3 配線長の累積分布

Fig. 3 Cumulative distribution of link length.

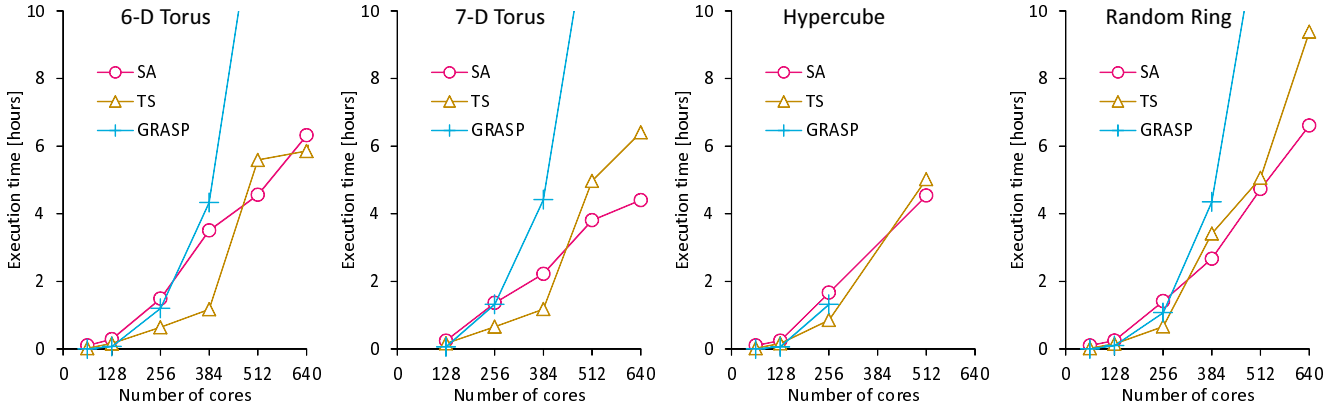

図 4 計算時間
Fig. 4 Execution time of solvers vs. number of cores.

3.2 コア数と配線長

コア数を 64 から 640 まで変化させたときの平均配線長を図 1 に，最大配線長を図 2 に，それぞれ示す．これらの結果から，SA と TS の両アルゴリズムがおおむね同様の配線長削減効果を達成していること，また，細かく見れば TS の効果が SA を上回る場合が多いことがわかる．例えば，640 コアの 7 次元トーラスの平均配線長は，SA で 45%，TS で 38%，それぞれ単純手法に比べて削減されている．SA で得られた平均配線長と TS で得られた平均配線長との差は 10%未満である．GRASP も配線長削減には成功しているが，その効果は SA や TS に比べて劣っている．なお，GRASP は 512 コアのハイパーキューブの計算時に実行時エラーが発生し，結果を得ることができなかった．単純手法 (Baseline) では，配線長が非常に長くなる場合が多い一方で，TS による最適解と同じ平均配線長が得られる「幸運な」場合もあることがわかる．図では 64 コアと 256 コアのトーラスとハイパーキューブがそれにあたる．この理由は配線長の累積分布を見ると明らかになる．

図 3 は，配線長の累積分布を示したものである．単純手法 (Baseline) のグラフには水平な階段状の部分が認められる．これは，チップの端とトポロジの次元境界がちょうど一致し，同じ長さの配線が多数生じたことを示している．階段状の部分はトーラスとハイパーキューブの「幸運な」場合において顕著に認められ（同図左：256 コア），「幸運でない」場合とランダムリングの場合には認められない（同図右：512 コア）．他方，SA, TS, GRASP のグラフはいずれも滑らかなカーブを描いている．これは，これらのアルゴリズムがコア配置を最適化するにあたり，チップの寸法とトポロジの次元境界との関係性に依存していないことを示している．

3.3 コア数と計算時間

コア数 N を $N = 64$ から $N = 640$ まで変化させたときの各アルゴリズムの計算時間を図 4 に示す．

まず SA について見ると，$N \leq 256$ の範囲では各アルゴリズムの中で計算時間が最も長く，$N > 256$ の範囲では逆に計算時間が最も短くなる傾向が認められる．前者は収束の速い問題に対しても必ず一定回数（本評価では 10 回）の試行を繰り返すことに起因し，後者は各試行における反復回数に上限（本評価では 1 億回）が設けられていることに起因すると考えられる．反復回数に上限を設けることは，計算時間の増加を抑える作用とともに，大域的最適解への到達可能性を下げる副作用を持つと考えられるが，結果として得られた配線長を見る限り，その副作用の大きさは限定的である．

次に TS について見ると，$128 < N < 512$ の範囲では各アルゴリズムの中で計算時間が最も短いが，$N \geq 512$ では SA よりも計算時間が長くなる傾向が認められる（ただし，ランダムリングではこの境界が $N \geq 384$ となる）．$N \leq 128$ では GRASP よりも計算時間が長いが，その差は 6 分未満である．また，$N = 512$ における SA の計算時間と TS の計算時間との差は最大 70 分 (23%) である．全体として，コア数 N に対する TS の計算時間は $O(N^2)$ にフィットしている．以上の結果から，おおむね $N < 512$ の範囲において，TS が最も妥当な最適化アルゴリズムであると言える．

最後に GRASP について見ると，コア数の増加とともに計算時間が指数関数的に増大し，$N = 512$ では計算時間が 12 時間を上回っている．第 3.2 節の結果と合わせて考えると，NoC のコア配置最適化問題に対して GRASP を用いることは不適切であると言える．

3.4 計算時間と配線長

各アルゴリズムの計算時間と配線長削減率の関係を図 5 に示す．縦軸は単純手法による総配線長を 1 とした比率である．この図から，SA と TS がほぼ同等の配線長削減率を達成していることが改めて確認できる．また，TS の計算時間はコア数が少なければ SA より短く，コア数が多ければ

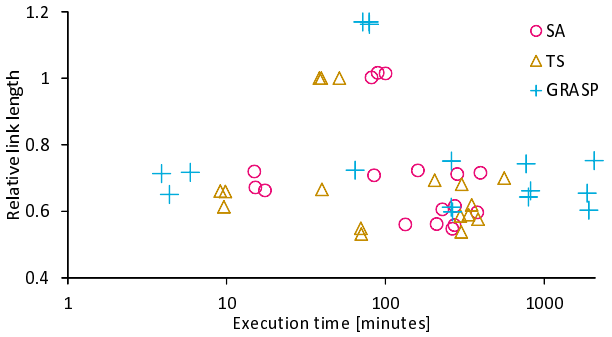

図 5 計算時間と配線長削減率

Fig. 5 Link length relative to baseline vs. execution time of solvers.

SA より長くなる傾向も確認できる．GRASP はプロットが縦に並ぶ傾向が認められる．これは，GRASP の計算時間が問題の本質的な難しさを反映しているのではなく，むしろ問題のサイズのみに依存していることを示している．640 コアの配置最適化に要する計算時間は，SA が最大 6.3 時間，TS が最大 9.4 時間，GRASP が最大 32 時間だった．

本章のすべての結果を総合すると，おおむね 512 コアを超えない範囲において，ロバストタブーサーチ法により，NoC のコア配置を効果的に最適化することが可能であると結論づけられる．

4. 関連研究

本章では，既存の NoC 向けトポロジを概観する．

すべてのルータがコアに接続されているネットワークを直接網と呼び，以下のようなトポロジが NoC に使われている．2 次元メッシュ (k-ary 2-mesh) と 2 次元折畳みトーラス (folded k-ary 2-torus) は，チップ上への配置が自明であり，配線長の均一性に優れることから，多くの NoC で採用されている．Butterfly [18] は，高次ルータを用いることでチップ上へ効率的に配置することができる．Flattened Butterfly [7] は，Butterfly を構成する多段ルータ群を一段のルータ群に置き換えたトポロジであり，ノード数に応じてルータの次数を変えることで柔軟な構成が可能である．Flattened Butterfly の次数を最も小さくした場合がハイパーキューブ (2-ary n-cube) に相当する．Spidergon [19] は，リングを基本として，その対角線上のノード間にショートカットリンクを追加したトポロジであり，メッシュと同様に短い配線でチップ上に配置することができる．以上のように，既存の NoC 向けトポロジはいずれもチップ上への最適な配置方法が既知である．一方，これらを性能面で上回る非 NoC 向けトポロジも種々存在するが，チップ上への配置が自明でないため，我々の知る限り NoC には使われてない．我々の手法を適用すれば，非 NoC 向けを含むあらゆるトポロジをチップ上へ最適に配置することが可能となる．

コアに接続されていないルータを持つネットワークを間接網と呼び，以下のようなトポロジが NoC に使われている．H-Tree は，根を除くすべてのルータが 1 本の上向リンクと 4 本の下向リンクを持つ四分木であり，平面を均等に分割するのに適していることから，多くの NoC で採用されている．Fat tree は，木構造を基本として，根に近い枝のリンク本数を増やすことで輻輳を緩和したトポロジである．Fat H-tree [20] は，2 つの H-tree を重ね合わせたトポロジであり，トーラス構造を内包しているという特徴を持つ．本報告の評価では間接網を扱っていないが，我々の手法はそのまま間接網にも適用可能である．

上述のような汎用的なトポロジとは別に，特定のアプリケーションの通信パターンに特化したトポロジを持つカスタム NoC も広く議論されている [21, 22]．ほとんどの汎用トポロジが規則的な構造を持つのとは対照的に，アプリケーションに特化したトポロジはしばしば不規則な構造を持つ．このような不規則なトポロジをチップ上に配置する方法は自明でなく，我々の手法を適用することで最適な配置を得ることが可能となる．

5. おわりに

本報告では，NoC で高次元なネットワークトポロジを利用することを目的として，その配線長を最小化するためのコア配置最適化手法を検討した．すなわち，コア間を結ぶネットワークトポロジを所与として，コア配置を二次割り当て問題としてモデル化し，3 種類のメタヒューリスティクスを用いて求解を試みた．実験の結果，おおむね 512 コアまでの NoC に対し，実用的な時間内でコア配置の最適化が可能であることが示された．このとき，単純手法と比べて総配線長が最大 45%削減された．また，解の質と計算時間のトレードオフの観点から，我々が用いた 3 種類のメタヒューリスティクスの中では，ロバストタブーサーチ法が最も適切なアルゴリズムであることがわかった．今後の課題として，より具象的なチップ製造上の諸条件を考慮し，実際の SoC 設計における我々のアプローチの有効性を検証していきたい．

謝辞 本研究の一部は科学研究費 #25280043 および #25730068 の助成を受けたものである．

参考文献

[1] W. J. Dally and B. Towles, “Route Packets, Not Wires: On-Chip Interconnection Networks,” in *Proceedings of the Design Automation Conference (DAC’01)*, Jun. 2001, pp. 684–689.
[2] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, John F. Brown III, and A. Agarwal, “On-Chip Interconnection Architecture of the Tile Processor,” *IEEE Micro*, vol. 27, no. 5, pp. 15–31, Sep. 2007.
[3] P. Gratz, C. Kim, K. Sankaralingam, H. Hanson, P. Shiv-

akumar, S. W. Keckler, and D. Burger, "On-Chip Interconnection Networks of the TRIPS Chip," *IEEE Micro*, vol. 27, no. 5, pp. 41–50, Sep. 2007.

[4] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz Mesh Interconnect for a Teraflops Processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sep. 2007.

[5] T. M. Pinkston and J. Shin, "Trends Toward On-Chip Networked Microsystems," *International Journal of High Performance Computing and Networking*, vol. 3, no. 1, pp. 3–18, Sep. 2005.

[6] N. Abeyratne, R. Das, Q. Li, K. Sewell, B. Giridhar, R. G. Dreslinski, D. Blaauw, and T. Mudge, "Scaling Towards Kilo-Core Processors with Asymmetric High Radix Topologies," in *Proc. of the International Symposium on High Performance Computer Architecture(HPCA)*, 2013.

[7] J. Kim, J. Balfour, and W. J. Dally, "Flattened Butterfly Topology for On-Chip Networks," in *Proceedings of the International Symposium on Microarchitecture (MICRO'07)*, Dec. 2007, pp. 172–182.

[8] M. Bayat and M. Sedghi, "Quadratic Assignment Problem," in *Facility Location*, R. Z. Farahani and M. Hekmatfar, Eds. Physica-Verlag, 2009, ch. 6, pp. 111–143.

[9] S. Sahni and T. Gonzalez, "P-Complete Approximation Problems," *Journal of the ACM*, vol. 23, no. 3, pp. 555–565, Jul. 1976.

[10] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," *European Journal of Operational Research*, vol. 176, no. 2, pp. 657–690, Jan. 2007.

[11] D. T. Connolly, "An improved annealing scheme for the QAP," *European Journal of Operational Research*, vol. 46, no. 1, pp. 93–100, May 1990.

[12] E. D. Taillard. Web page. [Online]. Available: http://mistic.heig-vd.ch/taillard/

[13] E. D. Taillard, "Robust taboo search for the quadratic assignment problem," *Parallel Computing*, vol. 17, no. 4-5, pp. 443–455, Jul. 1991.

[14] Y. Li, P. M. Pardalos, and M. G. Resende, "A greedy randomized adaptive search procedure for the quadratic assignment problem," in *Quadratic Assignment and Related Problems, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, P. M. Pardalos and H. Wolkowicz, Eds. AMS, 1994, vol. 16, pp. 237–261.

[15] P. M. Pardalos, L. S. Pitsoulis, and M. G. C. Resende, "Algorithm 769: Fortran subroutines for approximate solution of sparse quadratic assignment problems using GRASP," *ACM Transactions on Mathematical Software*, vol. 23, no. 2, pp. 196–208, Jun. 1997.

[16] M. G. Resende. Web page. [Online]. Available: http://www2.research.att.com/~mgcr/

[17] 鯉渕 道紘, 松谷 宏紀, 天野 英晴, D. F. Hsu, and H. Casanova, "高性能計算機インターコネクトにおけるランダムショートカットトポロジ," in **ハイパフォーマンスコンピューティングと計算科学シンポジウム** *(HPCS)*, Jan. 2012, pp. 85–92.

[18] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta, "Microarchitecture of a High-radix Router," in *Proceedings of the International Symposium on Computer Architecture (ISCA'05)*, Jun. 2005, pp. 420–431.

[19] M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, and A. Scandurra, "Spidergon: a novel on-chip communication network," in *Proceedings of the International Symposium on System-on-Chip (ISSOC'04)*, Nov. 2004, p. 15.

[20] 松谷 宏紀, 鯉渕 道紘, and 天野 英晴, "Network-on-Chip における Fat H-Tree トポロジに関する研究," **情報処理学会論文誌コンピューティングシステム**, vol. 48, no. SIG 12(ACS19), pp. 178–191, Sep. 2007.

[21] W. H. Ho and T. M. Pinkston, "A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns," in *Proceedings of the International Symposium on High-Performance Computer Architecture (HPCA'03)*, Feb. 2003, pp. 377–388.

[22] J. Hu and R. Marculescu, "Communication and Task Scheduling of Application-Specific Networks-on-Chip," in *IEE Proceedings Computers and Digital Techniques*, Sep. 2005.