平成 28 年度

法政大学 卒業論文

低コストで柔軟性を持つインターコネクトトポロジーの研究

An interconnection Topology with Low-Cost and Flexibility

法政大学 情報科学部 コンピュータ科学科
学籍番号 13K0120
佐藤 智文
Tomofumi Sato
E-mail: 13k0120@stu.hosei.ac.jp

指導教員: 李 亜民 教授

目次

図目次 **ii**

表目次 **iii**

Abstract **iv**

概要 **iv**

1 まえがき **1**

2 先行研究 **2**
2.1 Hypercube 2
2.2 Crossed Cube 2
2.3 (n, k)-Star Graph 4
2.4 Star Graph 5
2.5 Star Cube 6
2.6 Generalized-Star Cube 6

3 Generalized-Star Crossed Cube とその特徴 **8**
3.1 Generalized-Star Crossed Cube 8
3.2 基礎用語 9
3.3 証明 9

4 最短経路問題 **11**
4.1 Crissed Cube 部分の最短経路の概要 11
4.2 (n, k)-Star Graph 部分の最短経路の概要 12
4.3 最短経路探索単解 12
4.4 最短経路探索全解 13

5 BroadCasting on the Single-Port Model **18**
5.1 Spanning Binary Tree 18

6 比較 **20**

7 結び **23**

謝辞 **24**

参考文献 **25**

図目次

図 1 HQ(3) 2
図 2 HQ(4) 3
図 3 CQ(3) 4
図 4 CQ(4) 4
図 5 (3,2)-Star Graph 5
図 6 (4,2)-Star Graph 5
図 7 (3,2)-Star Graph x HQ(3) 7
図 8 HQ(3) x (3,2)-Star Graph 7
図 9 CQ(3) x (3,2)-Star Graph 8
図 10 (3,2)-Star Graph x CQ(3) 9
図 11 spanning binomial tree on single port CQ(4) 19
図 12 トポロジーのノード数と直径の比較 21
図 13 トポロジーのノード数とコストの比較 21
図 14 Hypercube との相対性コスト比較 (RCP) 22

表目次

表 1　ONE_STEP_ROUTE の適否 13
表 2　トポロジーの比較 20

Abstract

This paper presents a new interconnection network topology, called the Generalized-Star Crossed Cube (GSCC(n, k, m)), which is a product graph of the m-dimentional Crossed Cube and the (n, k)-Star Graph. GSCC(n, k, m) has three parameters: n, k, and m. So, the network size of GSCC(n, k, m) is more flexibly than a single graph: Crossed Cube or (n, k)-Star graph. And the diameter of this graph is smaller than one of GSC(n, k, m). GSC(n, k, m) is a product graph of the m-dimentional Hypercube and (n, k)-Star Graph. The diameter of GSCC(n, k, m) is shorter than the one of GSC(n, k, m) because the diameter of Hypercube is shorter than the one of Crossed Cube. This paper describes a shortest-path routing algorithm. The algorithm is varified by the execution time.

概要

この論文では Generalized-Star Crossed Cube(GSCC(n, k, m)) と呼ばれる m 次元の Crossed Cube と (n, k)-Star Graph との積グラフについて説明する．GSCC(n, k, m) は n, k, m の 3 つのパラメーターを所持している．だから，GSCC(n, k, m) のネットワークサイズは Crossed Cube，(n, k)-Star Graph 単体だけのときよりも柔軟性がある．そして，直径は GSC(n, k, m) よりも小さくなる．GSC(n, k, m) とは m 次元の Hypercube と (n, k)-Star Graph との積グラフである．GSCC(n, k, m) の直径が GSC(n, k, m) の直径が GSC(n, k, m) より小さくなるのは Crossed Cube の直径が Hypercube よりも小さくなるからである．この論文では最短経路アルゴリズムについて述べてあり，そのアルゴリズムの実行時間を検証している．

1. まえがき

近年，コンピュータを並列につなぎ，巨大なデータを高速に処理することが必要である．特にスーパーコンピュータでは接続する方法で処理速度が大きく変化する．例として，コンピュータ同士を全接続すると高速で処理できるが，リンク数が多くなり費用がかかる．しかしながら，コンピュータをリング型に接続すると費用は抑えられるが，直径が大きくなり処理に時間がかかる．これら両方のバランスを取り，できるだけ費用を抑え，かつ高速に処理できるトポロジーを設計することが求められている．また，既存の研究で提案されたトポロジーの例として Hypercube [1] が挙げられる．これは，ノード数が 2^m で次数と直径が m のグラフである．このグラフはスーパーコンピュータなどでよく用いられている．しかし，このグラフが設計できるノード数は 2 の累乗数のみで，それ以外は不可能である．この問題の解決策の 1 つとして，グラフと他のグラフとの積グラフを設計する方法がある．これは，あるグラフのノードの中に他のグラフを埋め込む方法である．これを使用することで，2 つのトポロジーの成分両方を所持し，ノード数はそれぞれのグラフが持つノード数の積で，次数と直径は 2 つのグラフの和で表すことができる．このようにノードがどんな数でもグラフを設計できる柔軟性のあるトポロジーを考える．

そのトポロジーの例として，Hypercube と (n,k)-Star Graph [2] の積グラフである GSC(m,n,k) (Generalized-Star Cube) [3] がある．これを使用することで Hypercube が持つパラメータ m と (n,k)-Star Graph が持つパラメータ n と k 2 つの合計 3 つのパラメータを持ち，Hypercube・(n,k)-Star Graph の 1 つだけよりも柔軟性があるグラフが設計できた．しかし，Hypercube であるとノード数は 2^m で直径が m と，ノード数が $n!/(n-k)!$ であり直径が $n-1$ である (n,k)-Star Graph と比較すると大きい．そこで，今回 Hypercube の代わりに Crossed Cube [4] を使う．これは，Hypercube と比較すると次数は m と変化しないが，直径が $\lceil (m+1)/2 \rceil$ となるグラフである．このグラフを (n,k)-Star Graph との積グラフにすることで，Hypercube が Crossed Cube に変化した分，直径を減らすことができる．本研究では，GSC(n,k,m) から直径を減らした GSCC(n,k,m) (Generalized-Star Crossed Cube) を提案する．

この論文では，2 節では先行研究・関連研究について述べ，3 節で GSCC(n,k,m) について，4 節では GSCC(n,k,m) の最短経路アルゴリズムとその実行時間について検証し，5 節では Crossed Cube 部分の Single-port model のブロードキャストについて紹介し，6 節で先行研究・関連研究と GSCC(n,k,m) のトポロジーの直径や次数などを比較し，7 節で結論を述べる．

2. 先行研究

この節では，GSCC(n, k, m) に関連したトポロジーの先行研究とそれの特徴や利点などについて紹介する．

2.1. Hypercube

Hypercube (HQ(m)) とは，ノード数が 2^m であり，次数と直径が m のグラフである．また平均距離は $m/2$ で，直径の半分である．アドレスは図 1 のように m ビットの 2 進数で表す．また，000 の隣接ノードは 001 や 010，100 の 3 つであり，そのアドレスと 1 ビット異なるものに接続する．この Hypercube はスーパーコンピュータのインターコネクトでもよく使われている．例えば，1985 年 7 月に発表された iPSC (Intel Personal Subercomputer) [5] では，128 個のノード数 (HQ(7)) で構成されている．この 128 個のノードには 16 ビットの 80286/287 の CPU が搭載されていて，ノード 1 個ずつが 0.1MFROPS で実行すると，128 個のノード iPSC はおよそ 12MFROPS のスループット能力であったが，当時のベクトルスーパーコンピュータの 160MFROPS と比べると非常に遅かった．その後，12 月に HQ(10) で 1024 個のノードを持つ NCUBE/ten が登場し，それぞれのノードが 32 ビットのカスタムプロセッサを搭載することによって各ノード 0.5MFROS となり，このシステムのピーク時で 500MFROS を達成した．当時は非常に高い入出力伝達ノードとされた．

このグラフの特徴として，左右同型，正則グラフ，最短経路アルゴリズムが簡単，などがあげられる．

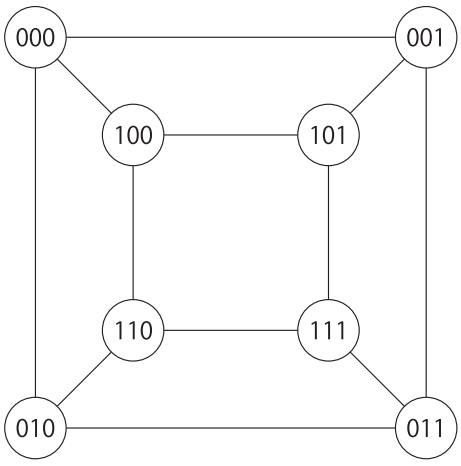

図 1: HQ(3)

2.2. Crossed Cube

Crossed Cube (CQ(m)) とは，ノード数の 2^m と次数の m は Hypercube と同じであるが，直径が $\lceil (m+1)/2 \rceil$ で Hypercube のおよそ半分からなるグラフである．このグラフも Hypercube と同様に m ビットの 2 進数で表す．このグラフは図 3 のように表すことができ

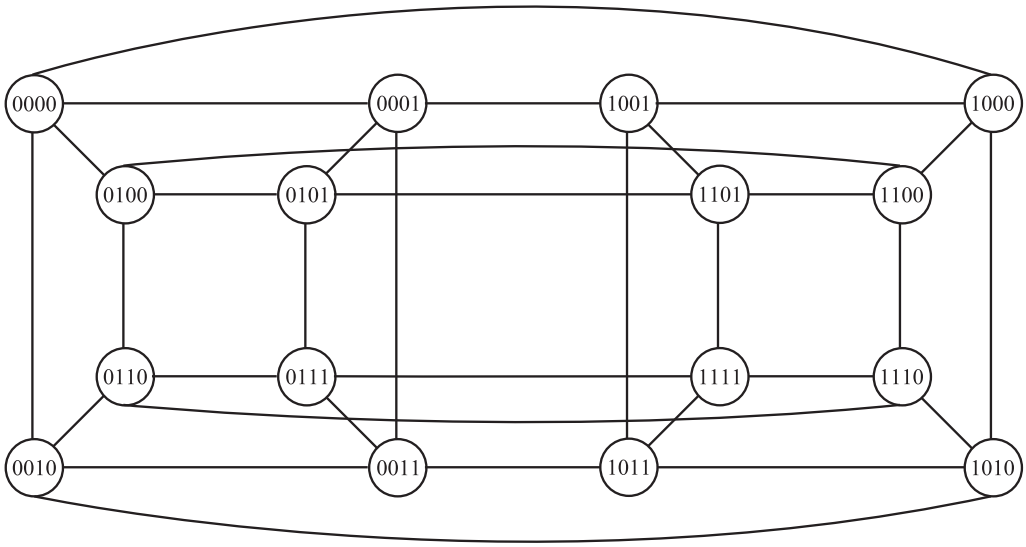

図 2: HQ(4)

る．図 3 との違いの例として，アドレス 110 の隣接ノードは図 1 の HQ(3) では 010,100,111 であるが，CQ(3) の隣接ノードは 001,100,111 である．また，アドレス 000 から 111 への経路探索として Hypercube の最短経路は 3 であるが，Crossed Cube の最短経路は 2 となり，1 回少ない探索回数で到達することができる．このようにリンクの接続する場所を変えることで直径を小さくすることができる．また図 2 と図 4 を比較すると，先程の隣接部分の他に，HQ(4) では (0101,1101)，(0011,0011)，(0111,1101)，(0001,1001) となっているが，CQ(4) では，(0101,1111), (0011,1001), (0111,1101), (0001,1011) の 4 つ異なる．このように m が大きくなるとリンクのつながり方が複雑になる．また，CQ(4) の次数は 4 で直径は 3 となり，HQ(4) よりも直径が 1 小さい．このグラフの特徴として，正則グラフ，短い直径，左右非同型などがあげられる．

また，CQ(m) の隣接ノードは，CQ($m-1$) を 2 つ組み合わせたものであるから，$\mathrm{CQ}^{(0)}_{m-1}\,(u_{m-2},...,u_1,u_0)$ と $\mathrm{CQ}^{(1)}_{m-1}\,(v_{m-2},...,v_1,v_0)$ を定義する．次に， 隣接ノードの条件の 1 つとなる $R = \{(00,00),(10,10),(01,11),(11,01)\}$ を定義する．これを満たすものを Crossed Cube のペア関係 (pair related) が成り立つという．これらから，隣接ノードは次の条件からなる．条件 1 はビット数が偶数である場合，最大ビットの 1 つ下のビットは変化しないといったものである．条件 2 は下から 2 ビットずつ見て，$u_{2j+1}u_{2j} \in \{01,11\}$ のとき上のビットを入れ替え，$u_{2j+1}u_{2j} \in \{00,10\}$ のときはそのままになる．また，経路の詳細は 4 節で示す．

1) m が偶数のとき， $u_{m-2} = v_{m-2}$
2) $(u_{2j+1}u_{2j}, v_{2j+1}v_{2j}) \in R\ (0 \le j < \lfloor \frac{m-1}{2} \rfloor)$

例えば，CQ(8) のアドレス 01001101(s) の隣接ノード t は，次数と同じで 8 つあり，01001100 ($m=1$)，01001111 ($m=2$)，01001110 ($m=3$)，01000111 ($m=4$)，01000101 ($m=5$)，01100111 ($m=6$)，00000111 ($m=7$)，11000111 ($m=8$) である．これは，m より大きいビットは変化しない特徴がある．ここで，$m=8$ のときの隣接ノードを見ていく．条件 1 は m が偶数であるので，t_5 は $s_5=1$ となる．条件 2 は，まず $0 \le j < (\lfloor (8-1)/2 \rfloor = 3)$ と

なる．$j = 0$ のとき $s_1 s_0 = 01$ であるので上のビットを入れ替えて $t_1 v_0 = 11$ になる．$j = 1$ のときも同様に $s_3 s_2 = 11$ であるので上のビットを入れ替えて $s_3 s_2 = 01$ となり，$j = 2$ のとき $s_5 s_4 = 00$ であるのでそのまま入れて $t_5 t_4 = 00$ となる．よって，$m = 8$ の隣接ノード v は 11000111 となる．

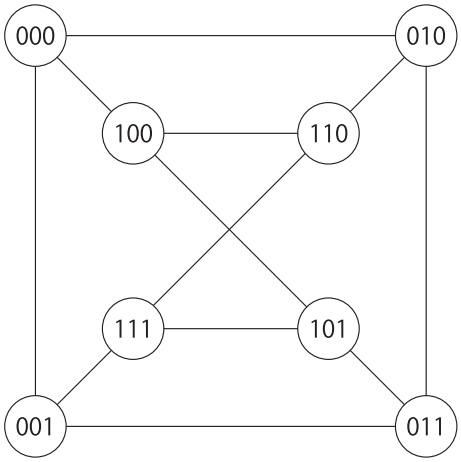

図 3: CQ(3)

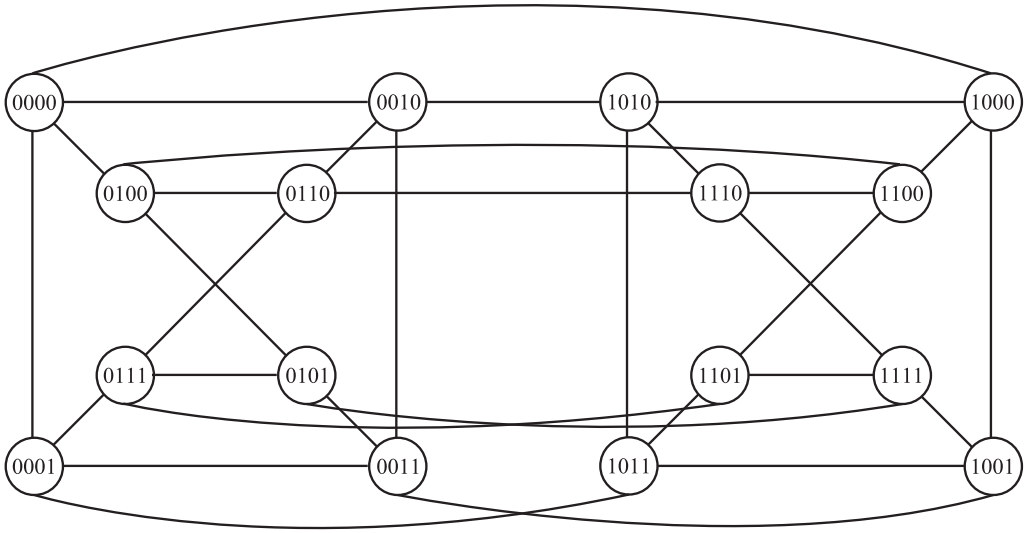

図 4: CQ(4)

2.3. (n, k)-Star Graph

(n, k)-Star Graph ((n, k)-Star) とは，ノード数が $n!/(n-k)!$ で表すことができるグラフである．次数は k の値に依存せず常に $n-1$ である．直径は n と k のパラメータで次の 2 つの状態に場合分けされる．$(1 \leq k \leq \lfloor n/2 \rfloor)$ の場合の $2k-1$ と，$(\lfloor n/2 \rfloor + 1 \leq k \leq n-1)$ の場合の $k + \lfloor (n+1)/2 \rfloor$ である．次数は k の値では変化しないため，n と k の値次第で直径とコストが非常に小さくなる．このグラフは図 5 と図 6 のように表すことができる．図 5 のノード数は 8，次数は 2 で直径は $k + \lfloor (n-1)/2 \rfloor = 3$ であり，6 のノード数は 12，次数

は 3，直径は $2k-1=3$ となる．また，アドレス番号は 1 から n の数値を k 個並べたものからなる．特に，$k=n-1$ のときは次に述べる Star Graph(n-Star) と同型になる．このグラフの特徴として，少ない次数，短い直径，低コスト，左右同型などが挙げられる．

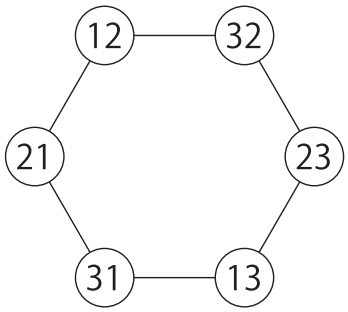

図 5: (3,2)-Star Graph

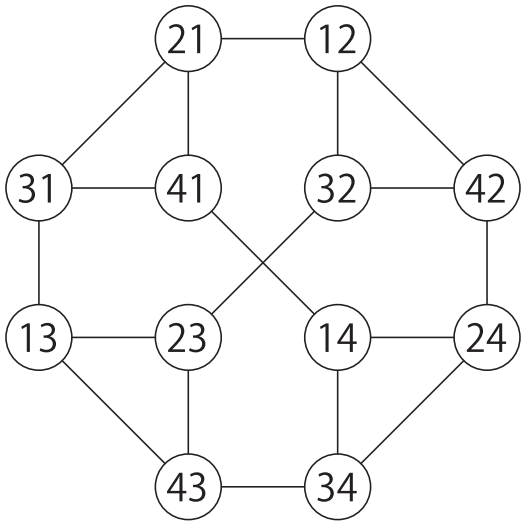

図 6: (4,2)-Star Graph

2.4. Star Graph

n-dimentional Star Graph(n-Star) は (n,k)-Star Graph から特に $(k=n-1)$ のときに作ることができるグラフである．このグラフのノード数は $n!$ のように n の階乗で表すことができる．次数は $n-1$ であり，直径は $\lfloor 3(n-1)/2 \rfloor$ である．図 5 は $k=n-1$ を満たしていて，3-dimentional Star Graph と表すこともできる．次数は (n,k)-Star Graph と同じ $n-1=2$ で，直径は $\lfloor 3(n-1)/2 \rfloor = 3$ となり，これらの条件全て満たしていることが確認できる．アドレス番号は 1 から n までの数からすべての数を 1 つずつ使っている．アドレス番号は，図 5 とは異なるが，$3! = 3!/(3-2)!$ となるため問題ない．このグラフの特徴として，正規グラフ，左右同型，短い直径，少ない次数などが挙げられる．

2.5. Star Cube

Star Cube とは Hypercube と Star Graph との積グラフであり，Hypercube が持つパラメータ m と Star Graph が持つパラメータ n の 2 つのパラメータを所持している．このグラフは，2 つを積グラフにすることによって少なくても 2 の累乗数，または n の階乗，またはその 2 つの積のどちらかを満たしたときこのグラフを設計することができる．それによって，グラフを設計することができる柔軟性を高めることができる．例えばノード数が 96 個の場合，2 の累乗数ではないので Hypercube で設計することは不可能である．また，n の階乗でもないため，Star Graph で設計することも不可能である．しかし，Star Cube であると，$m = 2$，$n = 4$ のとき，$2^2 \times 4! = 96$ となる．このように 2 の累乗数と n の階乗を組み合わせることにより，ノード数が 96 でも設計することができるようになる．このグラフの直径は 2 つのグラフの直径の和 $m + \lfloor 3(n-1)/2 \rfloor$ で，次数は $m + n - 1$ となる．このグラフは Hypercube 部分と Star Graph 部分の 2 つのアドレスを所持していて，Hypercube 部分の m ビットの 2 進数，Star Graph 部分の 1 から n までから 1 つずつ使うものである．このグラフの特徴として，正規グラフ，頂点と辺が左右同型，単純な最短経路探索などが挙げられる．

2.6. Generalized-Star Cube

このグラフは Hypercube と (n, k)-Star Graph の積グラフである．このグラフは Hypercube の m，(n, k)-Star Graph 部分の n と k の 3 つの変数を所持している．これら 3 つの変数を変えることで様々なサイズのグラフを設計することができる．これらの積グラフで，2 の累乗数または n-元集合から k-順列を取り出すもののどちらかを満たすことができる場合にこのグラフを設計することができる．Star Cube と比較して，パラメータ k を用いることができるので GSC(n, k, m) よりも高い柔軟性を保持している．次数はこれら 2 つの和 $m + n - 1$ である．直径は (n, k)-Star Graph と同様に k の値で 2 つの状態に場合分けされる．図 7 は，GSC(3,2,3) のグラフを示している．(3,2)-Star Graph に 6 つの HQ(3) を埋め込んでつくられたグラフになる．また，このグラフは Hypercube に (n, k)-Star Graph を埋め込んで作ることもできる．このグラフは図 8 で示す．これは，HQ(3) のノード部分に 8 つの (4,2)-Star Graph を埋め込んでつくることができる．これらのノード数は $2^3 \times 3!/1! = 48$ で，次数は $3 + 3 - 1 = 5$，直径は $3 + 2 + \lfloor 2/2 \rfloor = 6$ となる．このグラフのアドレスは Star Cube と同様に 2 つのアドレスを持つ．Hypercube 部分が持つ m ビットの 2 進数，(n, k)-Star Graph 部分が持つ 1 から n の数値を k 個並べたものからなる．

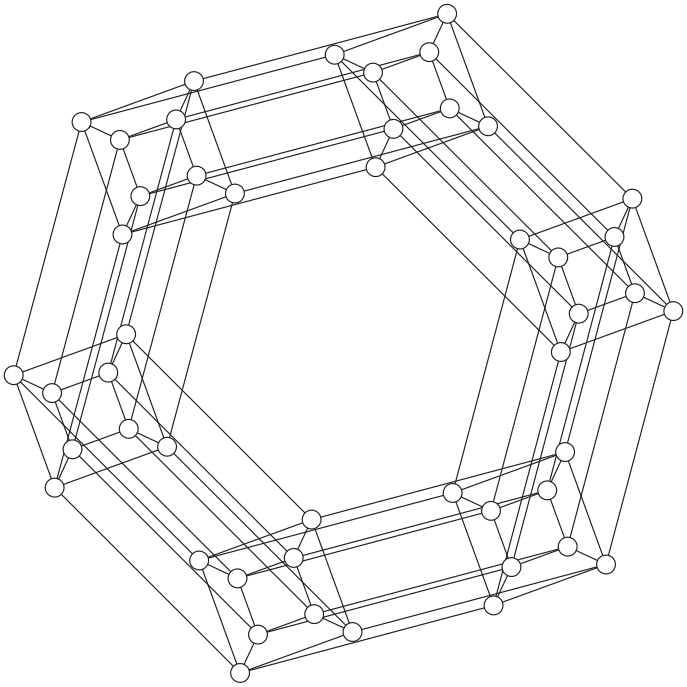

図 7: (3,2)-Star Graph x HQ(3)

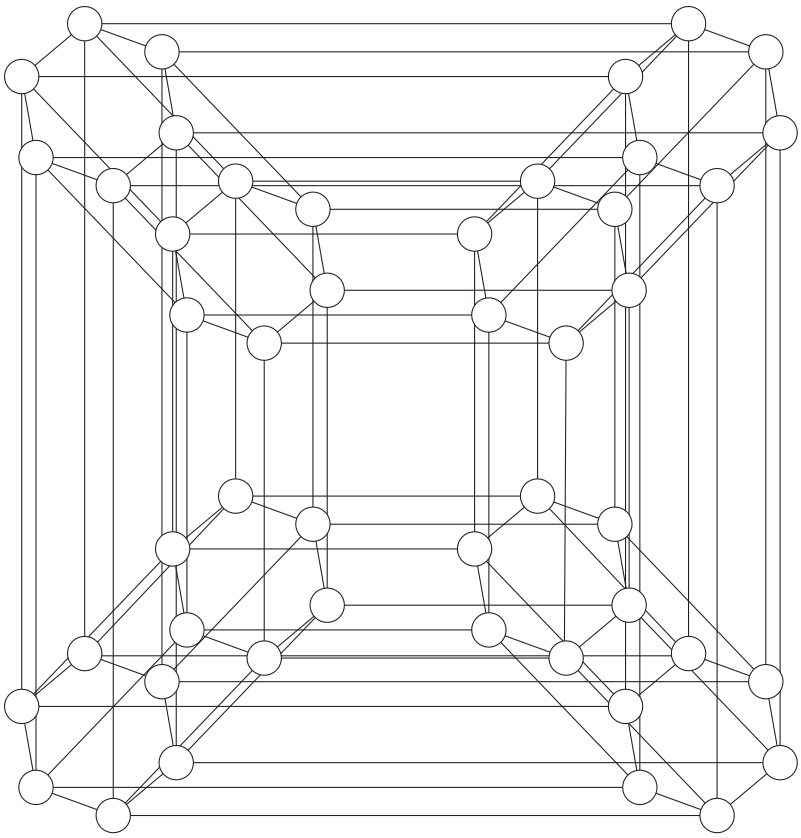

図 8: HQ(3) x (3,2)-Star Graph

3. Generalized-Star Crossed Cube とその特徴

この節では新しいトポロジーである Generalized-Star Crossed Cube について述べ，そのトポロジーの特徴について述べる．

3.1. Generalized-Star Crossed Cube

このグラフは，GSC(n, k, m) の Hypercube 部分を Crossed Cube に変化することで設計することができる．これは次数は変化しないが，Hypercube 部分の直径 m から Crossed Cube が持つ直径 $\lceil (m+1)/2 \rceil$ と小さくすることができる．図 9 は GSCC(3,2,3) のグラフを示していて，図 7 との違いは 6 個の HQ(3) が CQ(3) に変化している点である．またこのグラフは GSC(n, k, m) と同様に Crossed Cube に (n, k)-Star Graph を埋め込んでも同じグラフを作ることができる．そのグラフは図 10 に示す．このグラフは CQ(3) のノード部分に 8 つの (3,2)-Star Graph を埋め込んでいる．このグラフのアドレスは GSC(n, k, m) と同様に 2 つの部分のアドレスを持つ．Crossed Cube 部分が持つ m ビットの 2 進数，(n, k)-Star Graph 部分が持つ 1 から n の数値を k 個並べたものからなる．

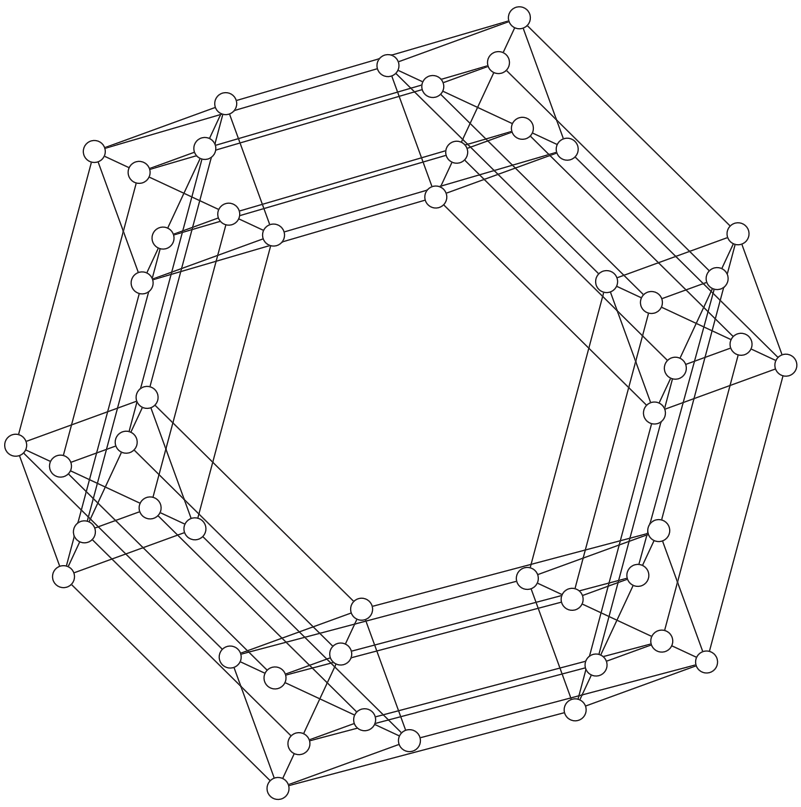

図 9: CQ(3) x (3,2)-Star Graph

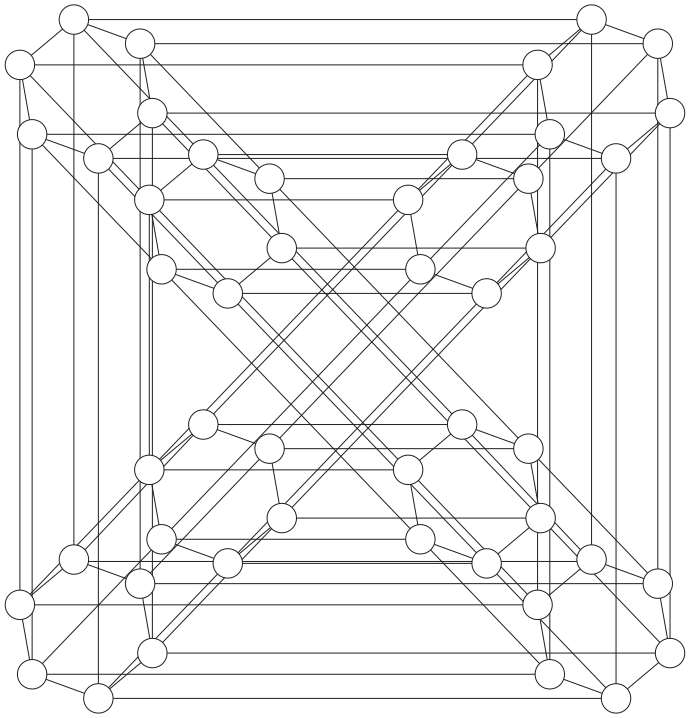

図 10: (3,2)-Star Graph x CQ(3)

3.2. 基礎用語

GSCC(n,k,m) の特徴を紹介する前に，下記に相互接続ネットワークについての基礎用語について説明する．この論文では、相互接続ネットワークは無向グラフとしている．したがって，頂点はプロセッサーに対応し，辺は双方向通信リンクに対応している．

定義 *1.* 相互接続ネットワークは有限グラフ $\boldsymbol{G} = \{\boldsymbol{V}, \boldsymbol{E}\}$ であり，$\boldsymbol{V}$ と $\boldsymbol{E}$ はそれぞれ頂点またはノードの集合，辺またはリンクの集合を表す．

定義 *2.* グラフ $\boldsymbol{G}$ における頂点 v の次数は，辺の数が v 上に示している物に等しい．

定義 *3.* グラフ $\boldsymbol{G}$ の直径 D_G は $\max\{d_G(u,v)|u,v \in \boldsymbol{V}\}$ であり，ここでの d_G は u と v の間の距離である．

定義 *4.* 全ての頂点が同じ次数ならば，それは正規グラフである．

3.3. 証明

定理 *1.* GSCC(n,k,m) のノード数は $2^m \times n!/(n-k)!$ である．

証明 *1.* Crossed Cube のノード数は 2^m である．また，(n,k)-Star Graph のノード数は $n!/(n-k)!$ である．GSCC(n,k,m) はこれらの積グラフであるから，ノード数は 2 つのグラフの積 $2^m \times n!(n-k)!$ となる． ■

定理 *2.* GSCC(n,k,m) の直径は $1 \leq k \leq \lfloor n/2 \rfloor$ のとき $\lceil (m+1)/2 \rceil + 2k - 1$ で，$\lfloor n/2 \rfloor + 1 \leq k \leq n-1$ のとき $\lceil (m+1)/2 \rceil + k + \lfloor (n-1)/2 \rfloor$ である．

証明 *2.* Crossed Cube の直径は $\lceil (m+1)/2 \rceil$ である．また，(n,k)-Star Graph の直径は $(1 \leq k \leq \lfloor n/2 \rfloor)$ のとき $2k-1$ である．GSCC(n,k,m) はこれらの積グラフであるから，直径は 2 つのグラフの和で表す事ができる．，$(1 \leq k \leq \lfloor n/2 \rfloor)$ のとき $\lceil (m+1)/2 \rceil + 2k - 1$ で，$(\lfloor n/2 \rfloor + 1 \leq k \leq n-1)$ のとき $(\lceil (m+1)/2 \rceil + k + \lfloor (n-1)/2 \rfloor)$ となる．

■

定理 *3.* GSCC(n,k,m) の次数は $m+n-1$ である．

証明 *3.* Crossed Cube の次数は m である．また，(n,k)-Star Graph の次数は $n-1$ である．GSCC(n,k,m) はこれらの積グラフであるから，次数は 2 つのグラフの和 $m+n-1$ となる． ■

定理 *4.* GSCC(n,k,m) のリンク数の合計は $2^{m-1}n!/(n-k)!(m+k+1)$ である．

証明 *4.* GSCC(n,k,m) は，(n,k)-Star Graph の型に $k!/(n-k)!$ 個の Hypercube が埋め込まれていると考える．したがって，Crossed Cube 部分のリンク数の合計は $(m2^m)/2 \times n!/(n-k)! = m2^{m-1}n!/(n-k)!$ となる．$(m2^m)/2$ は 2^m の各ノードが m 個隣接しているが，全ての辺が 2 倍に数えられているので，2 で割る．次に各 Crossed Cube 部分は $n-1$ 個隣にリンクされていて，それは (n,k)-Star Graph 部分を通して同じ Crossed Cube 部分と繋がっている．したがって，(n,k)-Star Graph の辺は $(n-1)2^m/2 \times n!/(n-k)! = (n-1)2^{m-1}n!/(n-k)!$ となる．ゆえに，GSCC(n,k,m) のリンク数は合計で $m2^{m-1}n!/(n-k)! + (n-1)2^{m-1}n1(n-k)! = 2^{m-1}n!/(m-k)!(m+n-1)$ となる． ■

定理 *5.* GSCC(n,k,m) のコストは $(1 \leq k \leq \lfloor n/2 \rfloor)$ のとき $(m+n-1)(\lceil (m+1)/2 \rceil + 2k + 1)$ で、$(\lfloor n/2 \rfloor + 1 \leq k \leq n-1)$ のとき $(m+n-1)(\lceil (m+1)/2 \rceil + 2k - 1)$ である．

証明 *5.* ネットワークのコストは コスト = 次数 × 直径 である．定理 3 から，GSCC(n,k,m) の次数は $(m+n-1)$ である．また，定理 2 から，直径は $(1 \leq k \neq \lfloor n/2 \rfloor)$ のとき $(m+n-1)(\lceil (m+1)/2 \rceil + n - 1)$ で，$(\lfloor n/2 \rfloor + 1 \leq k \leq n-1)$ のとき $(\lceil (m+1)/2 \rceil + k + \lfloor (n-1)/2 \rfloor)$ である．従って GSCC(n,k,m) のコストは，$(1 \leq k \leq \lfloor n/2 \rfloor)$ のとき $(m+n-1)(\lfloor (m+1)/2 \rfloor + 2k - 1)$ で，$(\lfloor n/2 \rfloor + 1 \leq k \leq n-1)$ のとき $(m+n-1)(\lceil (m+1)/2 \rceil + k + \lfloor (n-1)/2 \rfloor)$ となる． ■

定理 *6.* GSCC(n,k,m) は正規グラフである．

証明 *6.* Crossed Cube と (n,k)-Star Graph は正規グラフである．ゆえに GSCC(n,k,m) は積グラフであり，定義 4 から，GSCC(n,k,m) は正規グラフである． ■

4. 最短経路問題

GSCC(n,k,m) の最短経路の探索は，次の方法からなる．

1) Crossed Cube 部分の対象ノード s を宛先ノード t へ移動する．
2) (n,k)-Star Graph 部分の対象ノード u を宛先ノード v へ移動する．

4.1. Crissed Cube 部分の最短経路の概要

このアルゴリズムは探索回数を減らす方法として，Crossed Cube は 2 ビットペア関係であることをうまく使用し，上位ビットを使う順番を考え，余計な探索をしないようにする．例えば，$u = 0000, v = 0111$ のとき m=3 を実行すると u' = 0111 となる．このように上位ビットをうまく使うことによって探索回数を減らすことができる．

最初に，Crossed Cube 部分の最短経路の条件として l を s と t で異なる最大ビットとする．次に Crossed Cube の隣接ノードはビットのペア関係で遷移するため，全体のビットの半分の定数である $i^* = \lfloor \frac{l}{2} \rfloor$ を定義する．$j \geq i^*$ のとき，探索回数のステートである ρ と対象ノードと宛先ノード s,t の関係は式 (1), 式 (2) のようになる．s と t で異なる最大ビットより大きいビット ($j > i^*$) は探索する必要がないため，ρ は 0 になる．また，異なる最大ビット l が奇数で，かつそのビットとその下位ビットがどちらも異なる場合，ρ は 2 となり，それ以外は 1 となる．

$$\rho_{i^*}(s,t) = 0 \quad (j \geq i^* + 1) \tag{1}$$

$$\rho_j(s,t) = \begin{cases} 2 & (s_{i^*+1}s_{i^*} = \bar{t}_{i^*+1}\bar{t}_{i^*}) \\ 1 & (Otherwise) \end{cases} \tag{2}$$

次に，$j < i^*$ のときを考える．Crossed Cube はビットが変化すると下位ビットも変化する．そのために，下の 3 つ条件のいずれかを満たす s と t を距離維持ペア関係 (distance-preserve pair related) が成り立つとして，その式を $s_{2j+1}s_{2j} \overset{\text{d.-p.}}{\sim} t_{2j+1}t_{2j}$ のように表す．また，$j < i^*$ のときの ρ と s,t の関係は式 (3) のようになる．

1) $(s_{2j+1}s_{2j}, t_{2j+1}t_{2j}) \in \{(01,01),(11,11)\}$ かつ $\sum_{i=j+1}^{\lfloor \frac{m-1}{2} \rfloor} \rho_i(s,t)$ が偶数
2) $(s_{2j+1}s_{2j}, t_{2j+1}t_{2j}) \in \{(01,11),(11,01)\}$ かつ $\sum_{i=j+1}^{\lfloor \frac{m-1}{2} \rfloor} \rho_i(s,t)$ が奇数
3) $(s_{2j+1}s_{2j}, t_{2j+1}t_{2j}) \in \{(00,00),(10,10)\}$

$$\rho_j(s,t) = \begin{cases} 0 & (s_{2j+1}s_{2j} \overset{\text{d.-p.}}{\sim} t_{2j+1}t_{2j}) \\ 1 & (Otherwise) \end{cases} \tag{3}$$

ONE_STEP_ROUTE(i,Q)(Algorithm 3) は特定の条件を満たしたときに m を探索する．特定の条件は表 1 からなる．偶数，奇数というものは，$\sum_{k=j+1}^{\lfloor \frac{m-1}{2} \rfloor} \rho_k(s,t)$ が奇数か偶数かということを示している．また，yes はそのステートを維持したまま回を出すことができる。

no は維持されず，他の状態に変化することが必要になる．-は d.-p. pair related が成り立っていてその部分はアルゴリズムを実行することなく $s_{2i+1}s_{2i} = t_{2i+1}t_{2i}$ になる．

4.2. (n, k)-Star Graph 部分の最短経路の概要

(n, k)-Star Graph 部分の隣接ノードは，先頭のアドレスだけが異なる値である，あるいは先頭と n 番目にあるアドレスが交換されたもののみと接続されている．最短経路の探索方法として，先頭部分が異なる場合，それが宛先ノードにあるときはそれと交換する．宛先ノードに無い場合は，宛先ノードに存在して対象ノードにない最小の値を入れている．また，対象ノードと宛先ノードの先頭が同じであるが他の値が異なる場合，先頭とその異なる場所の中から最も小さい値と入れ替える．

4.3. 最短経路探索単解

最短経路アルゴリズムは次の Algorithm 1，Algorithm 3 に示す．まずは，1 つだけの最短経路探索の解を見つける．例として，GSCC(7,5,8) の大きさで Crossed Cube 部分のアドレス $s = 00101110$，$t = 00011001$，(n, k)-Star Graph 部分のアドレス $u = 73215$， $v = 12345$ とする．まず Crossed Cube 部分を探索する．はじめに探索回数とステートの $\rho_i(s, j)$ と T，Q を定義する．式 (1),(2),(3) から $\rho_j(s, t) = \{1, 1, 2, 0, 0\}$，Algorithm 1 から $Q = \{0, 1, 2\}$，$T = \{0, 1\}$ が導かれる．¯はビット反転であり，例として $\overline{s}_{2j+1}s_{2j} \overset{\text{d.-p.}}{\sim} t_{2j+1}t_{2j}$ ならば次の条件，

1) $(s_{2j+1}s_{2j}, t_{2j+1}t_{2j}) \in \{(11, 01), (01, 11)\}$ かつ $\sum_{i=j+1}^{\lfloor \frac{m-1}{2} \rfloor} \rho_i(s, t)$ が偶数
2) $(s_{2j+1}s_{2j}, t_{2j+1}t_{2j}) \in \{(11, 11), (01, 01)\}$ かつ $\sum_{i=j+1}^{\lfloor \frac{m-1}{2} \rfloor} \rho_i(s, t)$ が奇数
3) $(s_{2j+1}s_{2j}, t_{2j+1}t_{2j}) \in \{(10, 00), (00, 10)\}$

のいずれかを満たした場合成立する．探索を開始するためにまず Step1 へ移動する．$T \neq \phi$ であるから，$T \in T = 0, 1$ となる j を低い値から順番に ONE_STEP_ROUTE(i, Q) を実行する．ここでは，ONE_STEP_ROUTE$(0, Q)$ と ONE_STEP_ROUTE$(1, Q)$ の順で実行する．1 回目の実行では，$\rho_0 = 1$ で $(s_1s_0, t_1t_0) = (10, 01)$，$\sum_{k=1}^{\lfloor \frac{m-1}{2} \rfloor} = 3$ であるから，$s_1\overline{s}_0 \overset{\text{d.-p.}}{\sim} t_1t_0$ の条件を満たす．よって $2j = 0$ の隣接ノードを参照する． $m = 1$ が探索されて，$u = 00101111$，$Q = \{1, 2\}$ となる．2 回目で，$\rho_1 = 1$ で $(s_3s_2, t_3t_2) = (11, 10)$，$\sum_{k=2}^{\lfloor \frac{m-1}{2} \rfloor} \rho_k(s, t) = 1$ であるから，$s_3\overline{s}_2 \overset{\text{d.-p.}}{\sim} t_3t_2$ の条件を満たす．よって $2j = 2$ の隣接ノードを参照する．$m = 3$ が探索されて，$u = 00101001$，$Q = \{2\}$ となる．$T = \phi$ であるため Step2 へ移動する．ここでは Q が空になるまで Step3 に移動するものになっている．$Q \neq \phi$ であるから，Step3 へ進む．Step3 では，条件を満たす Q の最小値は 2 であるから，$i = 2$ が選択され，ONE_STEP_ROUTE$(2, Q)$ が実行される．$\rho_2 = 2$ であるから，$2j+1$ 番目または $2j$ 番目の隣接ノードを参照する．今回は $2j + 1 = 5$ 番目の隣接ノードを探索するために $m = 6$ を探索して，$s = 00001011$ になる．$\rho_2(s, t) = 1$，$Q = 2$ であり，また Step3 の条件を満たすため再度 ONE_STEP_ROUTE$(2, Q)$ が実行される．$\rho_2 = 1$ で $(s_5s_4, t_5t_4) = (00, 01)$，$\sum_{k=3}^{\lfloor \frac{m-1}{2} \rfloor} \rho_k(s, t) = 0$ であるから，$s_1\overline{s}_0 \overset{\text{d.-p.}}{\sim}$

t_1t_0 の条件を満たす．よって $2j = 2$ の隣接ノードを参照する．$m = 5$ を探索する．再び Step2 へ移動するが，$Q = \phi$ となり S が出力されて探索が終了する．Crossed Cube 部分の探索回数は 4 回で終了する．その探索経路 S は以下に示す．二重下線は m でビットが変化した部分であり，一重下線はペア関係によってビットが変化した部分を示している．

$$00101110 \rightarrow 00101111 \rightarrow 00101001 \rightarrow 00001011 \rightarrow 00011001$$

次に，(n, k)-Star Graph 部分を探索する．(n, k)-Star Graph はノードの先頭部分が変化するから，まず対象ノードの先頭の値を見る．$u_0 = 7$ であるが，$u_0 \notin v$ である．よって，$u \not\subset v$ である最小値 4 を u_0 へ代入して $u = 43215$ になる．2 回目は，$u_0 \neq v_0$ であるから，$u_0 = v_j$ となる j を探す． $u_0 = v_3$ であるから，u_0 と u_3 を入れ替えて $u = 13245$ となる．3 回目は，$u_0 = v_0$ であり，$u_j \neq v_j$ となる j の最小値は 1 であるため，u_0 と u_1 を入れ替えて $u = 23145$ が導かれる．4 回目は，2 回目と同様に $u_0 = v_j$ となる j を探す．$u_0 = v_1$ であるから，u_0 と u_1 を入れ替えて $u = 32145$ となる．最後に $u_0 = v_2$ であるから u_0 と u_2 を入れ替えて $u = v$ となる．対象ノードと宛先ノードが同じになり，(n, k)-Star Graph 部分の探索は 5 回で終了する．(n, k)-Star Graph 部分の経路は下に示す．また，この最短経路はこれら 2 つのグラフの和の 9 である．

$$73215 \rightarrow 43215 \rightarrow 13245 \rightarrow 23145 \rightarrow 32145 \rightarrow 12345$$

Crossed Cube は，最大で $\lceil (m+1)/2 \rceil$ 回で到達するため実行時間は $\boldsymbol{O}(m)$ となる．また，(n, k)-Star Graph 部分の実行時間は $\boldsymbol{O}(k)$ となる．このアルゴリズムの実行時間は平均で 2 つのグラフの和 $\boldsymbol{O}(m + k)$ となる．よって，Crossed Cube 部分の探索が多くなくなるときは $\boldsymbol{O}(m)$ で，(n, k)-Star Graph 部分の探索が多くなるときは $\boldsymbol{O}(k)$ となる．

表 1: ONE_STEP_ROUTE の適否

		$s_{2i+1}s_{2i}$			
		00	10	11	01
$t_{2i+1}t_{2i}$	00	-	yes	no	yes
	10	yes	-	yes	no
	11	(偶数,no)	(偶数,yes)	(偶数, -)	(偶数,yes)
		(奇数,yes)	(奇数,no)	(奇数,yes)	(奇数, -)
	01	(偶数,yes)	(偶数,no)	(偶数,yes)	(偶数, -)
		(奇数,no)	(奇数,yes)	(奇数, -)	(奇数,yes)

4.4. 最短経路探索全解

GSCC(n, k, m) の最短経路は 1 つだけでなく 2 つ以上存在することもある．4.3 節で示した例 (Crossed Cube 部分の $(s, t) = (00101110, 00011001)$，$(n, k)$-Star Graph 部分の $(u, v) = (73215, 12345)$) をまた見ていく．まず，Crossed Cube 部分の Algorithm 1 の Step1 までは 4.3

Algorithm 1 Routing Algorithm

Require: CQ(m) と (n,k)-Star の対象ノード s，u
Require: CQ(m) と (n,k)-Star の宛先ノード t，v
Require: Crossed Cube 部分の経路 S.
  $\rho_i(s,t)$ を定義. /* Crossed Cube 部分 */
  $Q \leftarrow \{j|\rho_j(s,t) \neq 0\}$,
  $T \leftarrow \{\rho_j(u,v) \neq 0, j < i^*$ かつ
    $\overline{s}_{2j+1}s_{2j} \overset{\text{d.-p.}}{\sim} t_{2j+1}t_{2j}$ or $s_{2j+1}\overline{s}_{2j} \overset{\text{d.-p.}}{\sim} t_{2j+1}t_{2j}\}$
  /***** Step1 *****/
  if $T = \phi$ **then**
    Step2 に進む.
  else
    $j \in T$ を探し，*ONE_STEP_ROUTE(i,Q)* を実行.
  end if
  /***** Step2 *****/
  if $Q = \phi$ **then**
    Crossed Cube 部分の最短経路 S を出力する.
    他の最短経路探索の為に S を消去し，Step4 へ進む.
  else
    Step3 へ進む.
  end if
  /***** Step3 *****/
  if $i \in Q$ を満たす $\overline{s}_{2i+1}s_{2i} \overset{\text{d.-p.}}{\sim} t_{2i+1}t_{2i}$ または $s_{2i+1}\overline{s}_{2i} \overset{\text{d.-p.}}{\sim} t_{2i+1}t_{2i}$ が存在する **then**
    最小の i を選ぶ.
  else
    $i \leftarrow max\{j|j \in Q\}$
  end if
  ONE_STEP_ROUTE(i,Q) を実行して Step2 へ戻る.
  while $u \neq v$ **do** /* (n,k)-Star Graph 部分 */
    u の最初のアドレスをみる.
    if $u_0 \neq v_0$ **then**
      $u_0 = v_j$ となる j ($j \neq 0$) を探す.
      u_0 と u_j を入れ替える.
    else
      $u_j \neq v_j$ となる最小の j を探す.
      if $j \neq null$ **then**
        u_0 と u_j を入れ替える.
      else
        $u \not\subset v$ である v の最小値 min を探す.
        $u_0 \leftarrow min$
      end if
    end if
  end while

Algorithm 2 Routing_Algorithm2

$\rho(s,t)$ を再定義.
$Q_2 \leftarrow \{j|\rho_j(s,t) \neq 0\}$
/***** Step4 *****/
if $Q_2 = \phi$ **then**
　最短経路 S を出力して終了.
else
　Step5 へ移動.
end if
/***** Step5 *****/
$a \leftarrow \max\{j|j \in Q_2\}$
$b \leftarrow \max\{j|j \in Q_2 \text{ and } j < a\}$
if $\rho_j(u,v) = 1$ and $u_{2b+1}\overline{u}_{2b} \overset{\text{d.-p.}}{\sim} v_{2b+1}v_{2b}$ **then**
　ONE_STEP_ROUTE(b, Q_2) を実行.
else
　if $\rho_j(u,v) = 1$ and $\overline{u}_{2b+1}u_{2b} \overset{\text{d.-p.}}{\sim} v_{2b+1}v_{2b}$ **then**
　　ONE_STEP_ROUTE(b, Q_2) または *ONE_STEP_ROUTE*(a, Q_2) を実行.
　else
　　ONE_STEP_ROUTE(a, Q_2) を実行.
　end if
end if
Step4 へ戻る.

Algorithm 3 ONE_STEP_ROUTE(j, Q)

if $\rho_j(s,t) = 2$ **then**
　$2j$ 番目または $2j+1$ 番目の探索.
　$\rho_j(s,t) \leftarrow \rho_j(s,t) - 1$;
else
　if $\overline{s}_{2j+1}s_{2j} \overset{\text{d.-p.}}{\sim} t_{2j+1}t_{2j}$ **then**
　　$(2j+1)$ 番目の探索.
　else if $s_{2j+1}\overline{s}_{2j} \overset{\text{d.-p.}}{\sim} t_{2j+1}t_{2j}$ **then**
　　$2j$ 番目の探索.
　end if
　$Q \leftarrow Q - \{j\}$
　$\rho_j \leftarrow \rho_j(s,t) - 1$
end if
$S \leftarrow S + \{u'\}$
$u \leftarrow u'$

節と同様に進め，$s = 00101001$ とする．Step3 の部分で ONE_STEP_ROUTE$(2, Q)$ を実行する部分で，$\rho_j(s,t) = 2$ のときに $2j$ 番目または $2j+1$ 番目の探索をして，1 では $2j+1 = 5$ 番目の探索をしたが，ここで $2j = 4$ 番目を探索する．その次に $2j+1 = 5$ 番目の探索をすることによって Crossed Cube 部分の最短経路の総数は 2 つに増える．この 2 つの計算する順序を入れ替えても変化しないのは m が奇数で $m+1$ が偶数のときに $m+1$ 番目を実行し

ても $u_{m-1} = v_{m-1}$ となり m 部分は変化しないからである．

$$S_1 = \{00101110, 00101111, 00101001, 00001011, 00011001\}$$
$$S_2 = \{00101110, 00101111, 00101001, 00111011, 00011001\}$$

また，他の Crossed Cube 部分の最短経路アルゴリズムの解として，Algorithm 2 がある．これによって他の探索方法で最短経路を求めることができる．$\rho(s,t)$ を再定義し，Q_2 は Q と同じ定義をする．まず，Step4 へ移動して，ここでは $Q_2 \neq \phi$ であるため，Step5 へ移動する．Step5 では a に Q_2 の最大値 2，b に Q_2 で 2 番目に大きい値 1 を代入する．$\rho_2(s,t) = 2$ であるので，ONE_STEP_ROUTE$(2, Q_2)$ を実行する．ここでも，先程と同様に $2j+1 = 5$ 番目と $2j = 4$ 番目の探索どちらから実行しても最短経路を求めることができる．これら 2 回実行した後の s と Q_2 はどちらも $s = 00011110$，$Q_2 = \{0, 1\}$ となる．次の Step5 では $a = 1$，$b = 0$ となり，$rho_1 = 1$ かつ $s_{2j+1}\overline{s}_{2j} \overset{\text{d.-p.}}{\sim} t_{2j+1}t_{2j}$ を満たす．よって，ONE_STEP_ROUTE$(0, Q_2)$ で，$m = 1$ が実行され，$s = 00011111$，$Q_2 = \{1\}$ となる．最後は $a = 1$，$m = null$ となり，ONE_STEP_ROUTE$(1, Q_2)$ で $rho_1 = 1$ かつ $s_{2j+1}\overline{s}_{2j} \overset{\text{d.-p.}}{\sim} t_{2j+1}t_{2j}$ を満たす．$m = 3$ が実行されて Step4 へ戻り，$Q_2 = \phi$ となり S が出力されて探索が終了する．この Algorithm 2 によっても最短経路を求めることができる．これによって求められる Crossed Cube 部分の最短経路の解は 2 つある。これらから，Crossed Cube 部分の最短経路の解は先程述べた 2 つを加えて 4 つになる．

$$S_3 = \{00101110, 00000110, 00011110, 00011111, 00011001\}$$
$$S_4 = \{00101110, 00110110, 00011110, 00011111, 00011001\}$$

(n,k)-Star Graph 部分も 1 つだけでなく複数の最短経路の解が存在する．Algorithm 1 の (n,k)-Star Graph 部分で，$u_j \neq v_j$ となる最小の j を探す．とあるが，ここを任意の j とすることで複数の最短経路を求めることができる．また，$u \not\subset v$ である v の最小値 min を探す．とあるが，ここも最小値ではなく任意の値にすることでここでも複数の最短経路の解を求めることができる．ここで，4.3 の (n,k)-Star Graph 部分の例 $u = 73215$，$v = 12345$ で検証する．2 回目までの探索は 4.3 の (n,k)-Star Graph 部分と同様に進めて，$u = 13245$ となる．3 回目で，先頭が 1 になるので 4.3 では $u_j \neq v_j$ の最小値 1 を求め，u_0 と u_1 を入れ替えたが，ここでは $u_j \neq v_j$ を満たす任意の値 2 として，u_0 と u_2 を入れ替える．4 回目は，$u_0 \neq v_0$ であり，$u_0 = v_2$ となるので u_0 と u_2 を入れ替えて $u = 21345$ が導かれる．5 回目も同様に，$u_0 = v_1$ となり u_0 と u_2 を入れ替える．ここで，$u = v$ となり探索は終了する．(n,k)-Star Graph 部分の最短経路総数は 2 つあり，それらは下記に示す．

$$73215 \to 43215 \to 13245 \to 23145 \to 32145 \to 12345$$
$$73215 \to 43215 \to 13245 \to 31245 \to 32145 \to 12345$$

これら 2 つを組み合わせることによって，GSCC(n,k,m) の全解を求めることができる。Crossed Cube 部分と (n,k)-Star Graph 部分は独立しているので，これら 2 つの積で求めることができる．GSCC(n,k,m) の最短経路探索の解は合計で $4 \times 2 = 8$ 通り存在する．なお，

Crossed Cube 部分と (n, k)-Star Graph 部分を交互に探索しても最短経路になるが，ここでは考えないものとする．

5. BroadCasting on the Single-Port Model

ここでは，single-port model についてのアルゴリズムについて説明する．アルゴリズムも最短経路探索アルゴリズムと同じように Crossed Cube 部分と (n,k)-Star Graph 部分に分けて考える．最初に，Crossed Cube 部分のブロードキャストについて考える．この部分では，spanning binary tree を用いて実行できるようにする．(n,k)-Star Graph 部分は Neighborhood broadcasting を用いて実行する．GSCC(n,k,m) のアルゴリズムはこれらを合わせることによって実行することができる．この方法は [6] と同じである．

5.1. Spanning Binary Tree

Crossed Cube 部分は，Crossed Cube の隣接ノードが再帰的に定義されるため，spanning binomial tree 通信方式を用いることができる．一般的に二項木は次のように定義される．

1) 次数 0 の二項木は 1 つのノードをもつ．
2) 次数 m の二項木は 1 つの根を持ち，その子はそれぞれ次数 $m-1, m-2, ..., 2, 1, 0$ の二項木の親である．

これらから，二項木は次数 m で 2^m 個のノードを持ち，高さは m である．一般的な二項木におけるブロードキャストアルゴリズムは Algorithm 4 に示す．Crossed Cube でこれを用いるには $mask$ の部分などを変更すればこのアルゴリズムを使うことが可能である．また，このアルゴリズムは 4 つのパラメータを使用している．d は Crossed Cube 部分の次元，$\boldsymbol{my\_id}$ は各ノードにつながった ID で，s は目的ノードの ID で，$\boldsymbol{X}$ はメッセージである．$\boldsymbol{s}$ によって送られたメッセージ $\boldsymbol{X}$ が到着するとすぐに全てのノードは並列にこのアルゴリズムが実行される．$\boldsymbol{my\_virtual\_id}$ と $\boldsymbol{mask}$ はノードにメッセージが送られているかか送られていないかを決定するために使われる．

Algorithm 4 ONE_TO_ALL_BC_single-port(d,$\boldsymbol{my\_id}$,$\boldsymbol{s}$,$\boldsymbol{X}$)

```
my_vertual_id = my_id XOR s;
mask = 2^d − 1;
for i = d − 1 to 0 do
    if (my_vertual_id & mask) = 0 then
        if (my_vertual_id & 2^i) = 0 then
            virtual_destination = my_virtual_id XOR 2^i;
            send(virtual_destination, X);
        end if
    end if
end for
```

図 11 は Crossed Cube の目的ノード $\boldsymbol{s} = 0000$ からのブロードキャストである．これは 4 回で到達することができる．また，$\boldsymbol{s} = 0000$ や $\boldsymbol{s} = 1000$ などのときは Hypercube と同じブロードキャストになる．

- Step1:
  $0000 \rightarrow 1000$

- Step2:
  $0000 \rightarrow 0100$, $1000 \rightarrow 1100$
- Step3:
  $0000 \rightarrow 0010$, $0100 \rightarrow 0110$, $1000 \rightarrow 1010$, $1100 \rightarrow 1110$,
- Step4:
  $0000 \rightarrow 0001$, $0010 \rightarrow 0011$, $0100 \rightarrow 0101$, $0110 \rightarrow 0111$,
  $1000 \rightarrow 1001$, $1010 \rightarrow 1011$, $1100 \rightarrow 1101$, $1110 \rightarrow 1111$

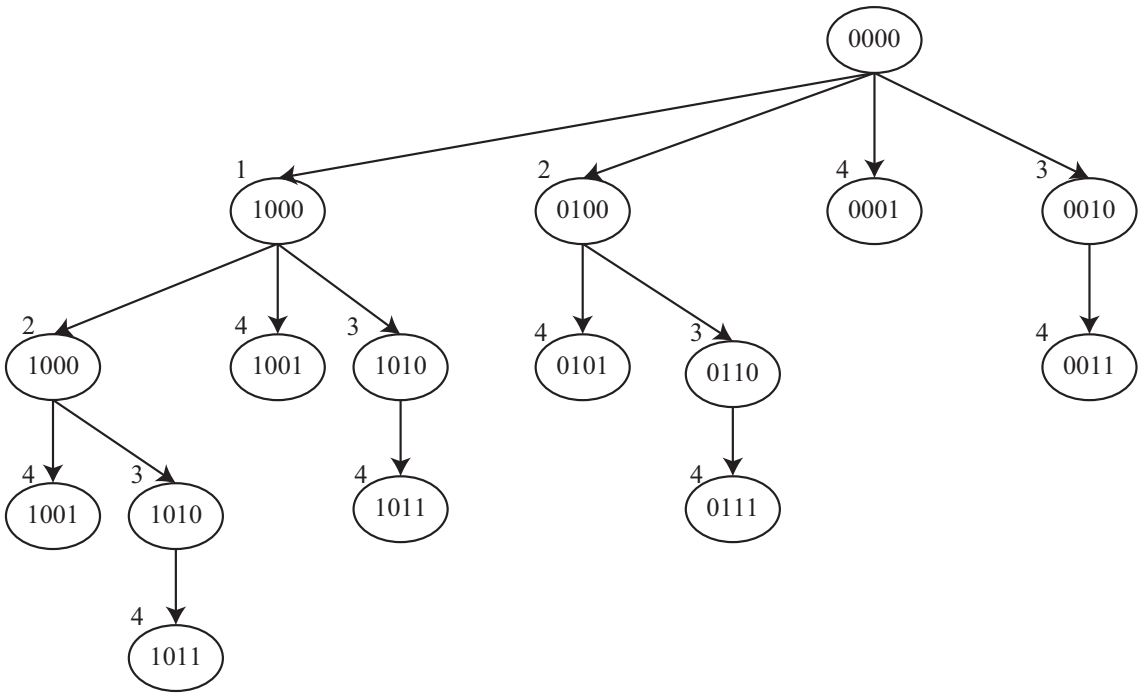

図 11: spanning binomial tree on single port CQ(4)

6. 比較

表 2: トポロジーの比較

	HQ(m)	CQ(m)	(n,k)-Star	GSC(n,k,m)	GSCC(n,k,m)
ノード数	2^m	2^m	$\frac{n!}{(n-k)!}$	$2^m\frac{n!}{(n-k)!}$	$2^m\frac{n!}{(n-k)!}$
次数	m	m	$n-1$	$m+n-1$	$m+n-1$
リンク	$m2^{m-1}$	$m2^{m-1}$	$\frac{n!}{(n-k)!}\frac{n-1}{2}$	$2^{m-1}\frac{n!}{(n-k)!}(m+n-1)$	$2^{m-1}\frac{n!}{(n-k)!}(m+n-1)$
直径	m	$\lceil\frac{m+1}{2}\rceil$	$2k-1$ $(1\le k\le\lfloor\frac{n}{2}\rfloor)$ $k+\lfloor\frac{n-1}{2}\rfloor$ $(\lfloor\frac{n}{2}\rfloor+1\le k\le n-1)$	$m+2k-1$ $(1\le k\le\lfloor\frac{n}{2}\rfloor)$ $m+k+\lfloor\frac{n-1}{2}\rfloor$ $(\lfloor\frac{n}{2}\rfloor+1\le k\le n-1)$	$\lceil\frac{m+1}{2}\rceil+2k-1$ $(1\le k\le\lfloor\frac{n}{2}\rfloor)$ $\lceil\frac{m+1}{2}\rceil+k+\lfloor\frac{n-1}{2}\rfloor$ $(\lfloor\frac{n}{2}\rfloor+1\le k\le n-1)$
コスト	m^2	$m\times\lceil\frac{m+1}{2}\rceil$	$(n-1)(2k-1)$ $(1\le k\le\lfloor\frac{n}{2}\rfloor)$ $(n-1)(k+\lfloor\frac{n-1}{2}\rfloor)$ $(\lfloor\frac{n}{2}\rfloor+1\le k\le n-1)$	$(m+n-1)(m+2k-1)$ $(1\le k\le\lfloor\frac{n}{2}\rfloor)$ $(m+n-1)(m+k+\lfloor\frac{n-1}{2}\rfloor)$ $(\lfloor\frac{n}{2}\rfloor+1\le k\le n-1)$	$(m+n-1)(\lceil\frac{m+1}{2}\rceil+2k-1)$ $(1\le k\le\lfloor\frac{n}{2}\rfloor)$ $(m+n-1)(\lceil\frac{m+1}{2}\rceil+k+\lfloor\frac{n-1}{2}\rfloor)$ $(\lfloor\frac{n}{2}\rfloor+1\le k\le n-1)$

表 2 は，それぞれのトポロジー HQ(m)，CQ(m)，(n,k)-Star，GSC(n,k,m) と GSCC(n,k,m) のノード数，次数，リンク数，直径とコスト (コスト = 直径×次数) を示している．GSC(m,n,k) と比較すると，GSCC(m,n,k) は $(1\le k\le\lfloor n/2\rfloor)$ のとき $m+2k-1$ から $\lceil(m+1)/2\rceil+2k-1$ へと減少していることが確認できる．また，コストも $(1\le k\le\lfloor n/2\rfloor)$ のとき $(m+n-1)(m+2k-1)$ から $(m+n-1)(\lceil(m+1)/2\rceil+2k+1)$ に減少している．

図 12，図 13 はそれぞれのトポロジーの直径とコストをグラフに示している．グラフの点はそのトポロジーを設計することができるノード数を示している．ここでは，本提案手法の GSCC(n,k,m) を他のトポロジーと比較する．Hypercube と比べると，ほとんどの場合でコストが小さく抑えられている．また，Crossed Cube と比較するとノード数が増えていくごとにコストが小さくなることが確認できる．また，柔軟性も Hypercube と Crossed Cube と比べても高い．(n,k)-Star Graph と比較すると GSCC(n,k,m) よりも直径が小さくなるが，作ることができるノード数は多く，柔軟性で勝ることが確認できる．また，GSC(n,k,m) と比較すると柔軟性に変化はないが，直径が減少してコストが抑えられている．特にノード数が 737,280 個のときの直径は GSC(n,k,m) は最大で 19 であるが，GSCC(n,k,m) は 15 と 4 減少し，コストも 323 から 255 と 68 減少していることが確認できる．

また，他の評価方法として李教授が提案した RCP(Relative Cost Performance) というものがある．これは，Hypercube を基準としてその RCP が常に 1 になり，他のグラフが Hypercube と比べてコストパフォーマンスが相対的に確認することができる．この計算は，図 13 で示した次数や直径の積と簡単な計算ではなく，ルーターコスト係数 λ や直結ポート数 p なども含めて複雑なコストパフォーマンスを比較できる．ルータコスト係数とはクロスバーの面積であり，制御部の構造部分で値が 1 から 2 の間に変化する．直結ポート数とはルータ

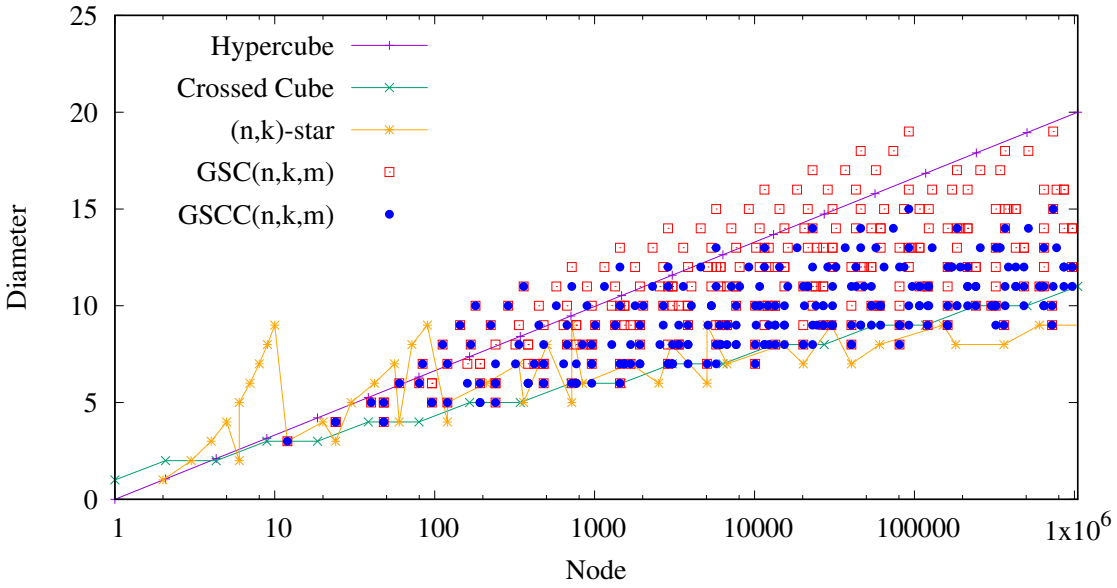

図 12: トポロジーのノード数と直径の比較

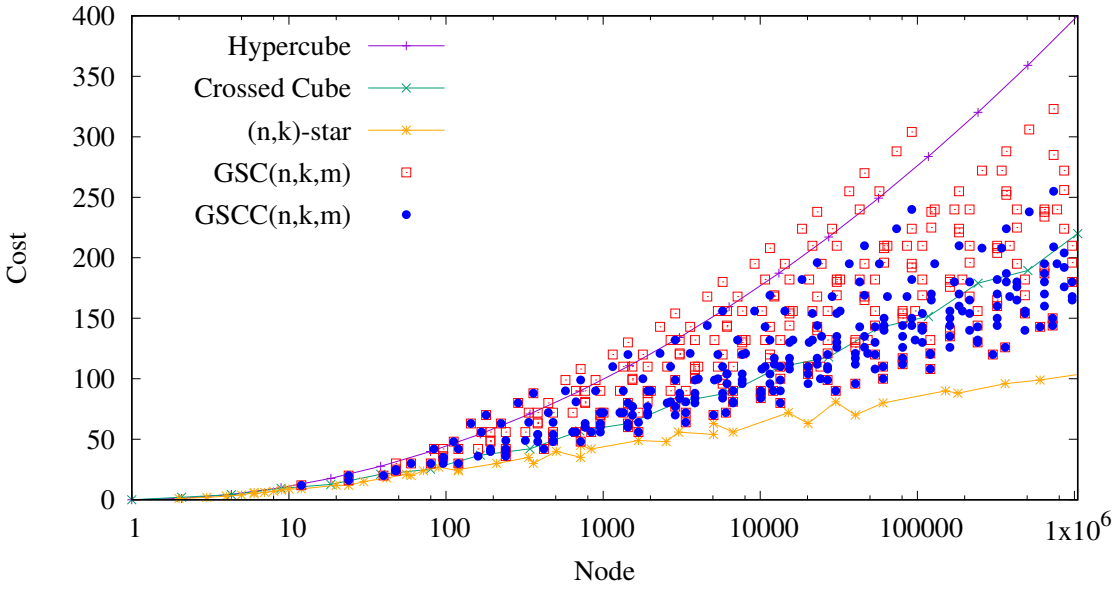

図 13: トポロジーのノード数とコストの比較

に接続するノード数のことである．これらを用いた RCP の式は下の式 (4) に示す．ここでは，直径 D，次数 d，ノード数 N としている．

$$RCP = \frac{(d+p)^{\lambda} D}{(\log_2 N + p)^{\lambda} \log_2 N} \tag{4}$$

ここで，この RCP を用いてそれぞれのトポロジーを比較したものは図 14 に示す．今回，ルータコスト係数 $\lambda = 1$，直結ポート数 $p = 1$ とした．Hypercube は RCP の基準となるため，

常に 1 になる．Crossed Cube は直径が大きくなるにつれて RCP が減少しコストパフォーマンス良くなることが分かる．ノード数が大きくなるにつれて 0.5 に収束していく．また，一度上昇してから再び減少するのは直径が $\lceil (n+1)/2 \rceil$ であり m の値が大きくなっても 1 回変化しないからである．(n,k)-Star Graph は Crossed Cube とほぼ同じ程度に推移するが，$n = k-1$ のときに著しくコストが低下している．特に $n = 3, k = 2$ のときに 1.6 以上になり，全体を見ても最もコストパフォーマンスが悪いことが確認できる．GSC(n,k,m) と GSCC(n,k,m) をみていくと (n,k)-Star Graph と同様に $n = k-1$ のときにコストが低下している．GSC(n,k,m) と比較すると，GSCC(n,k,m) はコストパフォーマンスの増減の幅が少なく，安定していることが分かる．のまた，ノード数が大きくなるにつれて GSCC(n,k,m) が他のトポロジーと比べて RCP が (n,k)-Star Graph よりも RCP が小さくなるときが増え，最もコストパフォーマンスが良くなることが確認できる．

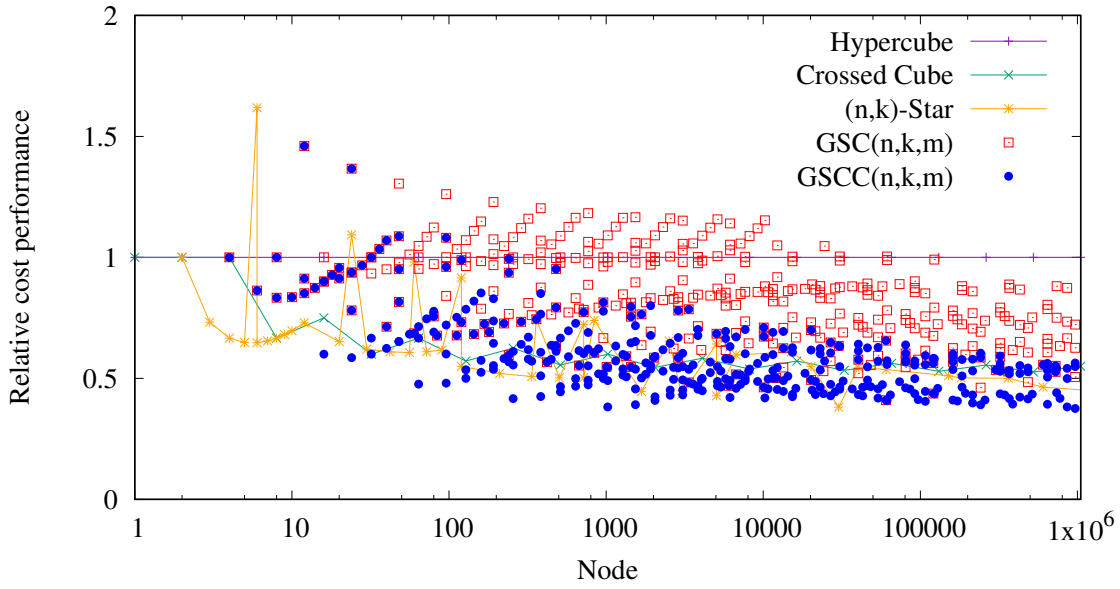

図 14: Hypercube との相対性コスト比較 (RCP)

7. 結び

本研究では，Generalized-Star Crossed Cube の利点，最短経路アルゴリズムとその実行時間について述べた．このグラフは柔軟性があり，さらに直径が減少することで，低コストで設計することができた．このグラフの更なる問題，例えば Broadcasting All-Port や故障率問題などにも検証することが今後の課題となる．

謝辞

本研究を進めるにあたり，ご指導を頂いた指導教員の李亜民教授に感謝致します．また，日常の議論を通じて多くの指摘を頂いた李研究室の先輩ならび同期に感謝します．

参考文献

[1] Y. Saad and M. H. Schultz, “Topological properties of hypercubes,” *IEEE Transactions on Computers*, vol. 37, no. 7, pp. 867–872, Jul 1988.

[2] W.-K. CHIANG and R.-J. CHEN, “Topological properties of the (n,k)-star graph,” *International Journal of Foundations of Computer Science*, vol. 09, no. 02, pp. 235–248, 1998. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0129054198000167

[3] D. Arai and Y. Li, “Generalized-star cube: A new class of interconnection topology for massively parallel systems,” in *2015 Third International Symposium on Computing and Networking (CANDAR)*, Dec 2015, pp. 68–74.

[4] C.-P. Chang, T.-Y. Sung, and L.-H. Hsu, “Edge congestion and topological properties of crossed cubes,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 1, pp. 64–80, Jan 2000.

[5] J. P. Hayes, T. Mudge, Q. F. Stout, S. Colley, and J. Palmer, “A microprocessor-based hypercube supercomputer,” *IEEE Micro*, vol. 6, no. 5, pp. 6–17, Oct 1986.

[6] D. Arai and Y. Li, “Routing and broadcasting algorithms for generalized-star cube,” *International Journal of Networking and Computing*, vol. 6, no. 2, pp. 368–394, 2016. [Online]. Available: http://www.ijnc.org/index.php/ijnc/article/view/133