修士論文

次数制約付き最小生成木問題への分散協調探索手法の適用

指導教員　松尾 啓志 教授
松井 俊浩 准教授

名古屋工業大学大学院工学研究科
修士課程 創成シミュレーション工学専攻
平成22年度入学 22413508番

伊藤 翼

平成24年2月3日

目 次

1　はじめに

近年，インターネット環境の通信網における一般ユーザによる動画像配信や次世代電力網における分散型電源を用いた送配電のようなネットワーク上の資源を配分するための分散協調型システムが注目されている．そのような分散協調型のシステムにおいては，競合解決や同時配信を実現するために配信木を利用することが，しばしば必要となる．一般に，配信木の木構造の多くは配信のコストを最小化する最小生成木 (MST: Minimum Spanning Tree) として研究されている [4, 6, 8, 10, 17]．MST はネットワークに参加する全ての頂点を辺で繋ぎ，かつ辺のコストの総和を最小化するため，配信木として有用である．

その一方で，配信を担当する各ノードの能力を考慮したデータやエネルギーの伝播も重要となる．実際的なネットワークにおけるノードは次数制約を課せられることが通例である．例えば，通信網においては，ある特定のノードに繋がる 1 本の物理的な通信線を複数の論理的な通信線が共有し，その共有数分だけ帯域が割かれることがあるため，接続数に制限が必要である．そこで，次数制約付き最小生成木 (d-MST: Degree-Constrained Minimum Spanning Tree) がネットワーク資源の消費を削減する重要な概念として研究されている [2, 9, 14, 20, 21]．一般的に d-MST は MST とは違い，多項式時間アルゴリズムを用いて解くことができないとされる．そのため，多くの近似解法が提案されているが，それらのほとんどは集中的なアルゴリズムであるため，大域的な情報の全てを単一ノードに集約するコストの増加や，そのノードが単一故障点となる問題が考えられる [2, 9, 14, 21]．その一方で，関連研究では，電力網におけるフィーダ木を構築する問題を分散制約最適化問題の枠組みで解く手法が提案されている [11]．この手法は送配電という特定の領域に特化しているが，これに類似する手法を d-MST のような，より一般的な制約付きの生成木構築問題に適用する余地があると考えられる．そこで，本稿ではネットワーク中のノードをエージェントとみなし，マルチエージェントの分散協調問題の枠組みで d-MST 問題を形式化 (分散 d-MST 問題) し，問題を解く．

分散 d-MST 問題に対して，従来研究における木構築問題のための分散制約最適化手法に類似する厳密解法，および近似解法を提案する．いずれの手法においても各エージェントが最大でも 1 本の辺を d-MST の辺として選択することで大域的な d-MST の解を得る．これらの 2 種類の手法を解の質やメッセージ通信のコスト，探索空間の規模といった観点から比較評価する．

2 章では背景について，3 章では，関連研究について述べる．4 章では分散 d-MST 問題の形式化について，5 章では提案手法について述べる．6 章では分散協調探索手法の比較・評価について述べ，7 章でまとめる．

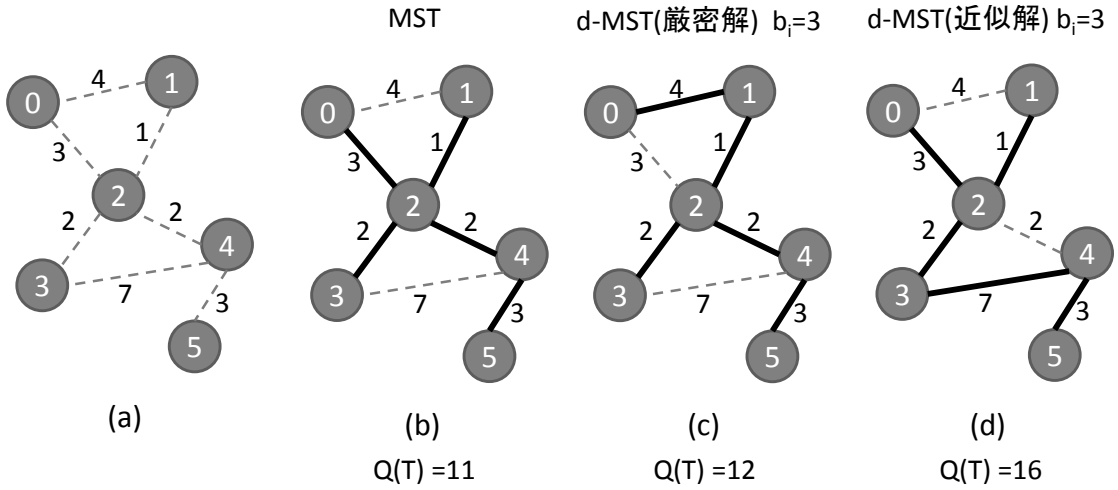

図 1: d-MST 問題

2 背景

通信網や電力網を構成する上で，コストを最小化する最小生成木 (MST: Minimum Spanning Tree) は重要な概念として研究されている [4, 6, 8, 10, 17]．その一方で，実際のネットワークにおいては，各ノードに接続する辺の接続数に制限があることが多いため，MST に次数の制約を加えた次数制約付き最小生成木 (d-MST: Degree-Constrained Minimum Spanning Tree) が有用であると考える．2.1 で d-MST 問題の詳細について述べ，2.2 では d-MST の適用分野について述べ，2.3 で分散協調問題解決について述べる．

2.1 次数制約付き最小生成木問題 (d-MST 問題)

d-MST 問題は，MST と同じく組み合わせ最適化問題の 1 つである．MST は，入力として重み付き無向グラフ $G = (V, E, c)$ が与えられ，そのグラフを構成する n 個の頂点 (エージェント) の集合 $V = \{i | i = 0, 1, \ldots, n-1\}$，および各頂点 i, j を結ぶ辺 (通信線) の集合 $E = \{(i, j) | i, j \in V \wedge i \neq j\}$，辺のコスト関数 $c \rightarrow R$ が与えられるときに，閉路を含まず，かつ，$\sum_{(i,j)\in E_T} c(i, j)$ を最小化する連結無向成分 $T = (V_T, E_T)$，$(E_T \subset E, V = V_T)$ である．それに加えて，d-MST 問題は，各頂点の次数 $d(i)$ に対し，次数の制限数 b_i が与えられ，構築する次数制約付き生成木 $T = (V_T, E_T)$ に含まれる各エージェント i の次数を $d_T(i)$ とするとき，$d_T(i) \leq b_i$ を満たし，かつ，$\sum_{(i,j)\in E_T} c(i, j)$ を最小化する T を求める問題である．解の質を $Q(T) = \sum_{(i,j)\in E_T} c(i, j)$ と定義する．

図 1 の (a) のようなグラフが与えられるとき，(b) のような MST と (c)，(d) のような d-MST を得る．(b) の MST は，次数制約がないため，最小の値 $Q(T) = 11$

をとる．(c) は d-MST の厳密解，(d) はコストに関する近似解であり，それぞれ $Q(T) = 12$，$Q(T) = 16$ となる．すなわち，同じ次数制約を満たす木でも $Q(T)$ に差がある．例えば，逐次的に辺を接続するような手法において，(d) のような解となる．仮に (a) において開始頂点を 0 として，そこから順々に辺を接続していくとする．まず，最初に頂点 0 は，隣接辺の中で最小のコスト値 3 を持つ辺 $(0, 2)$ を選択する．次に，頂点 0，2 の隣接辺の中で，まだ選択していない辺と頂点に対して，最小のコスト値 1 を持つ辺 $(2, 1)$ を接続する．同様に頂点 0，2，1 の隣接辺を選択するが，辺 $(2, 3)$ と辺 $(2, 4)$ が同等のコスト値 2 を持つため，どちらか 1 つを選択しなければならない．そこで，例えば，頂点番号による優先順序を付ける場合，辺 $(2, 3)$ が選択され，結果として，(d) とは違う木となる． MST 問題には，多くの決定的アルゴリズムが存在する [10, 17]．Prim[17] や Kruskal[10] のアルゴリズムは貪欲法を用いることで，多項式時間で MST 問題を解決可能である．それに対して，d-MST 問題は，一般的に $2 \leq b_i \leq n - 1$ のときに NP 困難な問題とされている [3]．そのため，多くの近似的な解法が提案されている [2, 9, 14, 21]．

d-MST 問題は，特に $b_i = 2$ のときに，最小ハミルトン路問題と同等な問題として扱われる [5]．また，d-MST 問題は，ユークリッド距離で辺のコストを表す問題と，ランダムな値で辺のコストを表す 2 種類に分かれる [9]．ユークリッド平面における d-MST は $b_i = 3$ の場合と $b_i = 4$ の場合，共に NP 困難であるとされるが，$b_i = 4$ の場合の方が比較的簡単に解けるとされている [15]．本研究ではランダムな値で辺のコストを表す問題を扱い，6 章の実験では $\forall i \in V$ に対して $b_i = 3$ とする．

また，次数制約付き最小生成木の近似解は制約の緩め方によって問題の性質が異なり，以下の 3 つがある．

次数制約付き準最小生成木 ： $d_T(i) \leq b_i$ を必ず満たすが，$Q(T)$ は必ずしも最小化されない．図 1 の (d) で示されるような木である．

緩次数制約付き最小生成木 ： $d_T(i) \leq b_i$ は満たさないが，Q(T) は最小化される．

次数制約付き部分最小生成木 ： 頂点集合 V の部分集合 R に対して，$v \in R$ を全て含む次数制約付き最小生成木を構築する．

本研究では，d-MST の近似解は次数制約付き準最小生成木 (DCST: Degree Constrained Spanning Tree) を基本的には扱うものとするが，5.2 において述べる近似解法の近似の度合いによっては，次数制約が緩和され，コストが最小化されない木を解として得る．

2.2 d-MST の適用分野

d-MST 問題の適用分野の 1 つとして，P2P ネットワークにおける動画像コンテンツの配信が挙げられる．P2P ネットワークでは，システム全体の中心となるノードが存在せず，各ノードが互いに対等な立場として扱われる．全てのノード

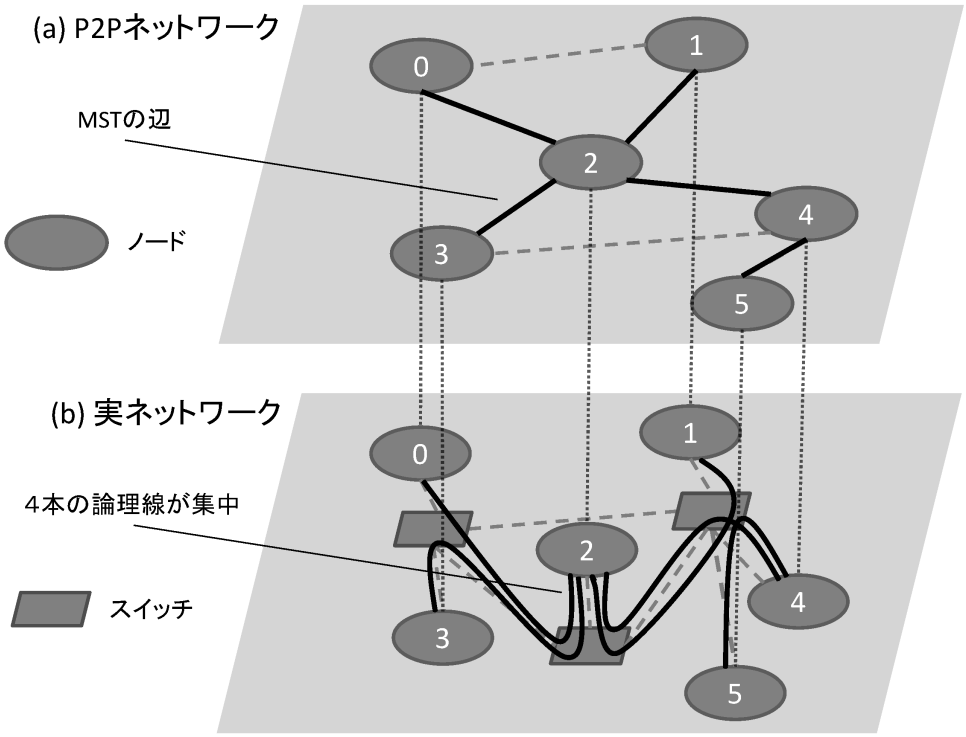

図 2: d-MST の適用分野

は，サービスをリクエストするクライントにもなり，リクエストに応えるサーバとしても機能する．この P2P ネットワークにおいて，例えば，動画像コンテンツの配信を行う場合，複数の端末間でデータを同期する必要がある．そこで，ALM(アプリケーションレベルマルチキャスト) が必要となり，多くの研究がなされている [7, 18, 19]．ALM においては，IP マルチキャストルータではなく，各エンドホストがルーティングやパケットの複製などの役割を担う．そのため，IP マルチキャストのようにネットワークインフラストラクチャを変更する必要がない．一般的には全てのエンドホストにパケットをルーティングするために，全てのエンドホストが論理的な通信線で結ばれるような生成木 (マルチキャスト木) を構築する．

例えば，図 2(a) の P2P ネットワークにおいて，MST をマルチキャスト木として構築すると，実線で示されるような全てのエンドホストを繋ぐ P2P の論理網が構築される．その一方で，図 2(b) の実際的な通信ネットワークにおいては，P2P の論理網は，下位層の物理ネットワークインタフェースを介するため，複数の P2P の論理線が，特定のエンドホストに繋がる 1 本の通信線を共有する状態や，特定のスイッチに過剰に接続するという状態が発生してしまう．1 本の通信線を複数の論理線が共有した場合，その通信線の帯域幅は共有された数だけ割かれることとなる．また，各エンドホストがパケットの複製を担うため，複製の数 (接続する論理線の数) が多ければ多いほど，複製の時間が大きくなってしまう．そこで，接続数 (次数) に制限を用いて，帯域幅の分割やノードの負荷の過剰な増加を抑制する

必要がある．従って，次数制約を考慮することができる d-MST の適用が有用であると考える．

2.3 分散協調問題解決

一般的に 2.2 のような分野では，ネットワーク中の各ノードが自律分散的に行動し，自身の情報と周辺のノードの情報を基に解を探索する分散協調探索が主体となる場合が多い．従って，本研究では d-MST 問題をマルチエージェントシステムの協調問題解決の枠組みとして捉える．マルチエージェントシステムにおいて協調問題を解決する基本的な枠組みとして分散制約充足/最適化問題 (DCOP: Distributed constraint Optimization Problem) が研究されている [11, 13, 16].

分散制約最適化問題は，複数のエージェントに変数と制約/評価関数が分散して配置された問題として形式化された，離散最適化問題である．分散制約最適化問題では，エージェントの状態/意思決定が変数として表現され，エージェント間の関係が制約/評価関数として表現される．各エージェントは互いに情報を交換しつつ自身の変数値を決定し，制約/評価関数の評価値を統合した値を最適化する変数値の割り当てを得る．このような表現は，分散システムにおける資源スケジューリングなどに含まれる，協調問題解決の本質的な問題を表すものとして重要である．基本的な形式は以下の通りである．

- マルチエージェントシステムに含まれるエージェントの集合を A で表す.
- A に含まれるエージェント $a_i \in A$ は変数の集合を持つ．a_i が持つ変数の集合を $X_i = \{x_i^1, x_i^2, \cdots, x_i^k\}$ で表す.
- エージェント a_i は X_i に含まれる変数 x_i^k の値のみ決定できる．すなわち，変数はエージェントの意思決定を表す.
- 変数間の関係は制約 c によって表す.
- 制約 c に関する評価関数 f_c を定義する．評価関数 f_c は変数値の組み合わせについての評価値を定義する.
- 制約の評価値の合計値が最適になる変数値を各変数に割り当てることが問題の目的である.

本研究では，この DCOP を用いて制約付き生成木の構築を扱う関連研究 [11] における形式化に類似した d-MST の形式化を行う．この関連研究の詳細については 3 章で述べる．

なお，グラフ $G = (V, E, c)$ は通信網を表し，頂点集合 V はマルチエージェントを表し，辺集合 E は通信線の集合を表すものとする．通信線はコスト c を持つ．各エージェントは自身と通信線で繋がれた隣接エージェントとメッセージを交換す

ることで相互作用し，解を探索する．また，通信網における通信線は全二重かつ対称的であり，隣接エージェント間では自由にメッセージの送受信が可能である．分散協調探索の開始時において，各エージェントはネットワーク中の自身の id と隣接辺のコスト値の情報については既知であるものとする．この形式化の詳細については 4 章で述べる．

3 関連研究

MST 問題および d-MST 問題に関する関連研究について 3.1，3.2 で述べる．これらを基礎とする手法を 6 章の評価における比較手法として用いる．また，制約付き生成木構築問題に類似する分散制約最適化問題の研究として，配電網におけるフィーダ木構築問題に関する研究について 3.3 で述べる．提案手法は，このフィーダ木構築問題の解法に類似すると考えられる．

3.1 MST 問題の解法

MST 問題の代表的な解法として Prim のアルゴリズム [17] と Kruskal のアルゴリズム [10] の 2 つがある．両手法ともグリーディな手法であるが，最適解を得ることが可能である．しかしながら，これらは集中型の解法であるため，2 章で述べたような協調問題解決の枠組みには直接的には適用できない．また，実際的な配信に適用するためには，管理ノードを設けてネットワークの大域的な情報を管理する必要がある．例えば，ネットワークトポロジや通信線の帯域幅などの情報を集約し，管理する必要があるため，ネットワークの規模が増大するほど，そのコストは増大する．それに対して，分散環境下で MST 問題の解を得る NNT アルゴリズムが提案されている．隣接ノードと隣接辺の情報のみを用いて，少ないメッセージ交換で解を得ることが可能であるが，得られる解は MST の近似解である．

Prim と Kruskal と NNT の詳細について，それぞれ 3.1.1，3.1.2，3.1.3 で述べる．

3.1.1 Prim

Prim のアルゴリズム [17] は，まず，任意の単一頂点のみを含む部分木から開始する．部分木に属する頂点は，部分木に含まれない頂点を端点に持つ辺の中でコストが最小となるものを選び，その頂点と辺を部分木に追加する．以下，同様の処理を行い部分木を拡大させていき，最終的に部分木が全ての頂点を含めば，終了する．$G = (V, E, c)$ が与えられるとき，出力する MST を $T = (V_T, E_T)$，部分木を $T' = V_{T'}, E_{T'}$ とする．Prim のアルゴリズムの処理の詳細は以下の通りである．

1. 部分木 $T' = (V_{T'} = \{\}, E_{T'} = \{\})$ を作成する．

2. G から任意の頂点を 1 つだけ選び，$V_{T'}$ に追加する．

3. $i \in V_{T'}$ と $j \in V - V_{T'}$ からなる 辺 (i, j) の中で，重みが最小となる (i, j) を選び，j を $V_{T'}$ に，(i, j) を $E_{T'}$ に追加する．

4. 2.，3. を繰り返し，$V_{T'} = V$ になれば，処理を終了する．T は最小全域木となる．

3.1.2 Kruskal

Kruskal のアルゴリズム [10] は，Prim のアルゴリズム [17] のように隣接した頂点を順に部分木に追加するのではなく，その時点で最もコストの小さい辺を閉路ができないように追加する．従って，複数の木を拡大させていき，最終的にはそれらの木が繋がって，1 つの木となるとき終了する．T の任意の部分グラフを $T' = (V_{T'}, E_{T'})$ とする $(E_{T'} \subseteq E')$．T' を構成する T の部分木の中で，頂点 v が属するものを T'_v と表すとする．

1. $T' = (V_{T'} = V, E_{T'} = \{\})$ を作成する．

2. E から重み $c(e)$ が最小となる 辺 e を取り出し，e を構成する 頂点 $i, j \in V$ の間に $i \notin T'_j$ かつ $j \notin T'_i$ のとき，e を E' に追加する．

3. T'_i と T'_j は T'_j もしくは T'_i のどちらかに統合される．

4. 2.，3. を繰り返して，T が連結となれば，処理を終了する．T は最小全域木となる．

3.1.3 NNT

NNT のアルゴリズム [8] は，MST を分散構築する近似解法である．分散配置されたセンサーが無線通信を用いてデータを同期することを想定し，データを同期するための配信経路として MST を利用する．センサーの電力や処理能力は限られているため，できる限り処理量や通信回数を減らし，電力消費を抑えるなど低コストな配信経路の決定が望ましい．そこで，NNT アルゴリズムによって少ないメッセージ交換で MST の構築を行う．そのため，NNT アルゴリズムによって，得られる木は準最適な MST である．

各エージェント i は，優先順位を付けるためのランダムな値 $rank(i)$ を持つ．リーダーとなるエージェント s から近傍のエージェント間でメッセージを交換することで，初期木が構築される．この初期木に基づいて，全てのエージェントは自身の rank 値を決定し，その rank 値を基に順序付けられる．リーダー以外の全てのエージェント i は，$rank(i) < rank(j)$ となる近傍エージェント j の中で， $c(i, j)$ が最小となる辺 (i, j) を 1 つだけ選択することで，コストを最小化できる可能性の

高い経路を選択する．これにより，閉路検出の処理を完全に省いて生成木を構築することができる．ランクの関係は以下のように定義される．

- $r(u) < r(v)$ のとき，$p(u) < p(v)$，もしくは，$p(u) = p(v)$ かつ $id(u) < id(v)$

基本的な NNT の処理は以下の通りである．

1. エージェント s をリーダーとする．s は，$p(s)$ という値を $[b-1, b]$(b はランダムな値) の範囲から選択し，自身のランク値 $r(s) = (p(s), id(s))$ を決定する．
   近傍のエージェント全てに $p(s)$ と自身の id 値 $id(s)$ を送信する．

2. $p(s)$ を受信したエージェント s の近傍エージェント v は，$[p(s)-1, p(s))$ の範囲から $p(v)$ を選択し，$r(v) = (p(v), id(v))$ を決定する．
   近傍のエージェント全てに $p(v)$ と $id(v)$ を送信する．

3. $p(v)$ を受信したエージェント v の近傍エージェント u は，同様にして，$[p(v)-1, p(v))$ の範囲から $p(u)$ を選択し，$r(u) = (p(u), id(u))$ を決定する．
   近傍のエージェント全てに $p(u)$ と自身の固有値 $id(u)$ を送信する．

4. 全てのエージェントは，3. の処理を行う．
   ただし，複数の近傍エージェントから，メッセージを受信するエージェントは一番最初に受信したメッセージに対してのみ値を決定する．
   (この処理によって，リーダー s 以外の全てのエージェントに対し，自身より高いランクを持つ隣接エージェントが少なくとも 1 つは存在することになる．)

5. $rank(i)$ を決定したエージェント i は，$rank(i) < rank(j)$ となり，最もコストの小さい辺 (i, j) の端点となるエージェント j に辺を接続する．

なお，NNT は無線通信網を想定したアルゴリズムであるため，rank 値を決定する際，無線を用いたブロードキャストによる rank 値情報のメッセージ伝播を行う．しかしながら，本研究で扱う通信網は，通信線を用いた有線での通信を前提とするため，そのためにアルゴリズムの変更が必要となる．また，d-MST 問題に適用したアルゴリズムとするために，次数の制限を考慮する辺の接続に関するヒューリスティクスを追加する必要もある．この変更の内容に関しては，5.3 で述べる．

3.2 d-MST 問題の解法

関連研究 [14] において，primal method，dual method，branch and bound procedure の 3 つの手法が提案されている．これら 3 つの手法は比較手法として最も代表的な手法であるが，3.1 で述べたような集中的なアルゴリズムによる解法であるため，分散協調の枠組みには直接的には適用できない．MST 問題に関しては，

3.1.3 で述べた NNT や関連研究 [4] の GHS アルゴリズムのような分散アルゴリズムを用いた解法が多数存在するが，d-MST には代表的な分散型の解法がない．そこで，primal-method を分散処理化し，分散協調探索手法の比較手法の 1 つとして 6 章において比較評価を行う．この分散処理の詳細については，5.4 で述べる．

3.2.1 primal method

primal method は，Prim のアルゴリズム [17] を d-MST の解法として拡張した手法 (d-prim) である． Prim のアルゴリズムと同様に任意の単一頂点のみを含む部分木からから処理を開始する．その部分木に属する頂点と，部分木に属さない頂点を繋ぐ辺のコストが最小となる頂点を部分木追加していく．ただし，その過程で Prim とは異なり，次数制約を満たす頂点のみを部分木に追加する．最終的に全ての頂点を追加したとき，低コストな次数制約付き生成木 T(DCST) を得る．その後，DCST における次数の制限を超えないように T の辺の組み換えを行い，$Q(T)$ を最適解に近付ける．DCST における辺の組み換えの処理は以下の通りである．削除した辺の集合を DE とする．

1. 辺 $e_{ij} =$ を T から削除し，部分木 T_i と T_j に分け，DE に e_{ij} を追加する．
2. $d_T(v)+1 \leq b_v$ かつ $d_T(w)+1 \leq b_w$ かつ $e_{vw} \leq cost(e_{ij})$ となる辺 e_{vw} を探す．ただし，$v \in T_i$ かつ $w \in T_j$ とする．
   もし，そのような e_{vw} が存在すれば 5. へ，そうでなければ 3 へ行く．
3. もし，$d_i = b_i$ かつ $d_j = b_j$ ならば 4. へ，そうでなければ 6. へ行く．
4. $d_T(v)+1 \leq b_v$ かつ $d_T(w)+1 \leq b_w$ かつ $e_{vw} = cost(e_{ij})$ となる辺 e_{vw} を探す．ただし，$v \in T_i$ かつ $w \in T_j$ とする．
   もし，そのような e_{vw} が存在すれば DE に e_{vw} を追加して 5 へ，そうでなければ 6. へ行く．
5. e_{ij} と e_{vw} を交換し 7. へ行く．
6. もし，$|DE| \leq |V|-1$ ならば 2. へ，そうでなければ処理を終了する．

2 章でも述べたように，この手法のように逐次的に辺を選択し，バックトラックを行わない近似アルゴリズムの場合，$Q(T)$ は処理を開始する頂点や，辺の選択順序に依存する．従って，$Q(T)$ は最適解である保証はない．

3.2.2 dual method

dual method は，primal method と同様に Prim のアルゴリズム [17] を d-MST 問題の解法として活用した手法である．まず最初に，Prim を用いて最小生成木

$T = (V_T, E_T)$ を作成する．その後，T に属する頂点の中で，次数制約を違反する頂点を含む辺を削除し，次数制約を満たし，かつ削除した辺のコストとの差が最小となるような辺を新たに T に追加することで $Q(T)$ を最適解に近付ける．辺の組み換えの処理は以下の通りである．あるノード v に隣接する頂点集合を V_v と表すとする．

1. MST を作成する $(T = (V, E))$.
2. $d_T(i) > b_i$ となる頂点を探す．ただし，$i \in V_T$ とする．
   もし，そのような i が存在すれば 3. へ，そうでなければ処理を終了する．
3. $j \in V_i$ について $p(j) = cost(e_{ij}) - cost(e_{rs}(j))$ を計算する．
   ただし，$e_{rs}(j)$ は辺 e_{ij} を削除することでできる部分木を e_{ij} の代わり繋ぐ辺の中で，最もコストが小さくなる辺とし，もし r または $s \neq j$ ならば $d_T(r)+1 \leq b_r(d_T(s)+1 \leq b_s)$ であり， r または $s = j$ ならば $d_T(r) \leq b_r(d_T(s) \leq b_s)$ である．
4. $p(j*) = min\{p(j)\}$ とする．e_{ij*} と $e_{rs}(j*)$ を交換し，$d_T(i) = d_T(i) - 1$，$d_T(j*) = d_T(j*) - 1$，$d_T(r) = d_T(r) + 1$，$d_T(s) = d_T(s) + 1$ とする．
   もし，$d_T(i) \leq b(i)$ ならば 2. へ，そうでなければ 3. へ行く．

この手法も primal method と同様に最適解を得る保証はない．

3.2.3 branch and bound procedure

branch and bound procedure は，分枝限定法を用いることで，部分解 (部分木) をバックトラックしながら探索し，d-MST 問題の最適解を得る厳密解法である．解の探索の際に，下界値として MST のコストの総和値，上界値として DCST のコストの総和値を用いることで枝刈りを行う．本研究では，この手法の様に分枝限定法を用いた手法を実装し，最適解を得る．そして，得た解の総コスト値 (最適値) と，各手法で得られた解の総コスト値との差を算出する．この誤差を用いて，6 章において，各手法の解の質を評価する．

本研究で用いる分枝限定法のアルゴリズムでは，d-MST の，ある部分問題の部分木を $T' = (V_{T'}, E_{T'})$ とし，$(i, j) \in E - E_{T'}$ を T' に追加することにより生成される最小生成木を T'' とするとき，下界値を $Q(T') + Q(T'')$ とする．また，それまでに探索して得た完全解のコストの最良値を上界値として用いる．処理の概要は以下の通りである．

1. 部分木 $T' = (V_{T'} = \{\}, E_{T'} = \{\})$ を作成する．
2. $d(i) < b_i$ かつ $d(j) < b_j$ を満たし，コストが最小な頂点 $i, j \in V - V_{T'}$ と辺 $(i, j) \in E - E_{T'}$ を 1 組選択する．

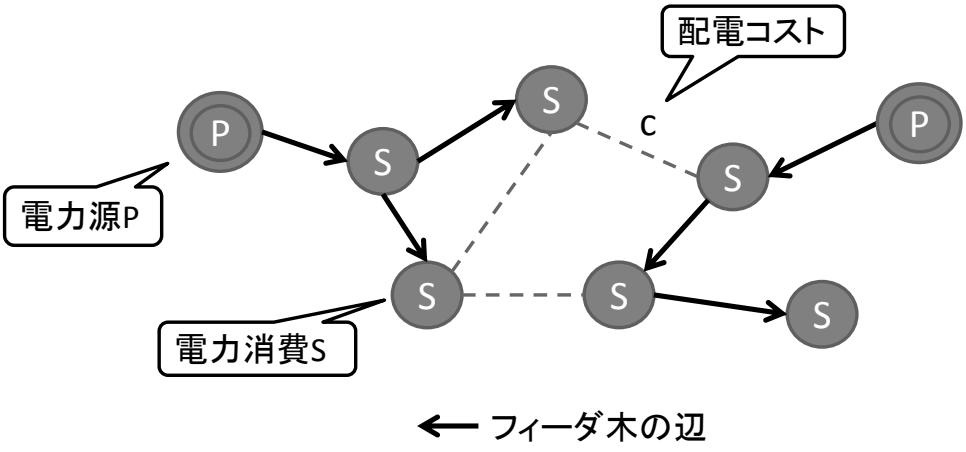

図 3: フィーダ木

3. もし，T' が閉路を含まず，かつ連結であれば，その頂点と辺を部分木 T' に追加する．

4. もし，$V_{T'} \neq V$ ならば 2. からの処理を繰り返す．$V_{T'} = V$ となれば処理を終了する．T は次数制約付き生成木となる．
$Q(T)$ が上界値より小さければ解と上界値を更新し，別の部分木が生成できる所までバックトラックし，2. からの処理を繰り返す．
ただし，その時点の下界値が現在の上界値以上ならば枝刈りする．

3.3 電力網の経路割り当て問題におけるフィーダ木の構築

3.3.1 形式化

関連研究 [11] において，分散制約充足/最適化問題 [11, 13, 16] を電力網の経路割り当て問題に適用する手法が提案されている．図 3 のように電力を供給する電力源 P と電力を消費する電力消費 S，配電コストの割り当てられた配電線からなるグラフが与えられたとき，電力網は矢印で表されるようなフィーダ木を構築する．電力源 P をソース，電力消費 S をシンクと呼ぶ[1]．全てのシンクに対して的確な電力を供給するために，制約条件が 3 つある．

- 木制約: 配電経路はソースを根とする生成木である
- KCL 制約: 電力フローはキルヒホッフの法則を満たす
- 資源制約: 配電線に流れる電力は配電線の最大容量を違反しない

[1]なお，関連研究 [11] の問題では，複数のソースがある場合には，複数の生成木にネットワークが分割される．

これらの制約条件を満たした上で，配電コストが最小となるフィーダ木を構築することが目的である．

このフィーダ木の木制約を満たすために，各エージェント i は，供給元変数 d_i を持つ．この共有元変数 d_i は生成木の有向辺の情報を持ち，電力の供給の方向を表す．$d_i = j$ のとき，エージェント i はエージェント j から電力を供給されることを表す．この変数 d_i に対して，評価関数 $f_i(d_i, d_j)$ が定義されている．d_i と f_i による評価は以下のように行う．

- $d_i = j \wedge d_j = i$ のとき，$f_i(d_i, d_j) = \infty$ となり，制約違反を表す．
- $d_j \neq i$ のとき，$f_i(d_i, d_j) = 0$ となり，配電コストが評価されないことを表す．
- $d_j = i$ のとき，$f_i(d_i, d_j) = c(i, j)$ となり，$c(i, j)$ 分の配電コストが評価されることを表す．

各エージェント i が，この共有元変数 d_i を矛盾なく決定することで，電力を供給するための木制約を満足したフィーダ木が構築可能である．

3.3.2 解法

3.3.1 の形式化に基づく [11] の解法では，分散制約最適化問題に対する最適解を求める厳密解法である DPOP アルゴリズム [16] を応用した解法を適用し，フィーダ木の生成問題のために特化させることで，電力網の経路割り当て問題を解く．

DPOP は，前処理として制約網に対する深さ優先探索木を生成する．また，深さ優先探索木に基づく擬似木 (pseudo-tree) を生成する．疑似木は部分木間に依存がないため，部分木間で並行して解探索の動作を行うことができる．そして，擬似木により定義される半順序関係にしたがって，メッセージ交換を伴う動的計画法に基づく部分解の伝播を行うことで最適解を得る．

3.3.1 の形式化に基づく関連研究 [11] においては，この DPOP に，配電線に流すことのできる最大の電力量に関する資源制約の評価を追加している．各エージェントは自身を含む閉路で消費する電力量がわからなければ配電線の容量制約を違反しているかどうかを評価することはできない．従って，生成された疑似木上に存在する自身を含む閉路において消費する電力を決定可能な閉路の最上位エージェントが資源制約の評価を行う．

この電力網の経路割り当て問題のような形式化を適用することで，d-MST 問題を分散協調問題解決の枠組みで定義できると考える．すなわち，電力網の経路割り当て問題におけるフィーダ木の構築に類似した形式化を，より一般的な木の構築問題である d-MST 問題に適用可能である．また，フィーダ木の構築において DPOP を応用したボトムアップな解法によって送配電のコスト値を最適化するように，d-MST の辺のコスト値を最適化することができると考える．そこで，本研究では，4 章において，辺を表現するための変数 d_i に類似する変数を用いた形式

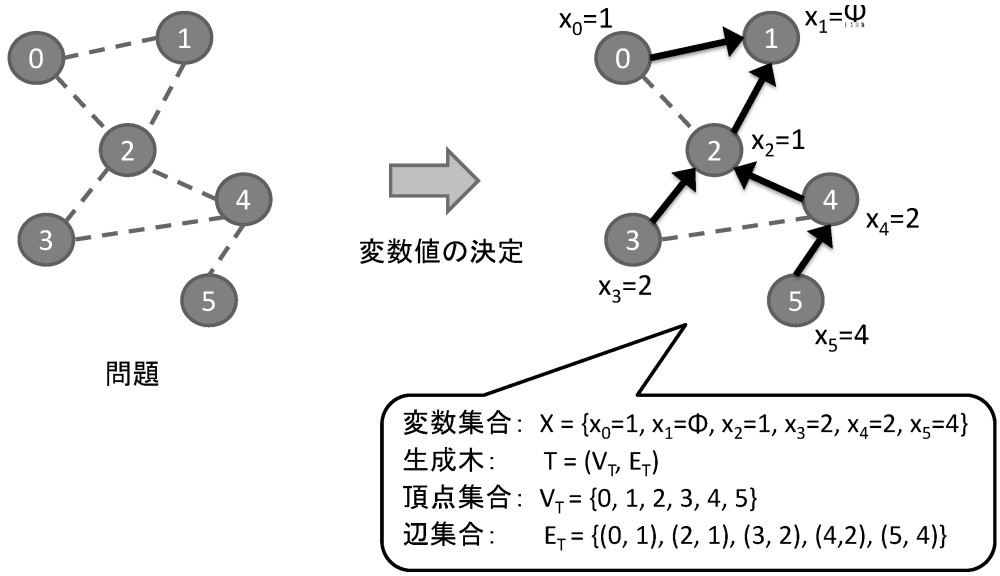

図 4: 変数値の決定により表される生成木

化を示し，5 章において，その形式化に基づく [11] の解法に類似する分散型の厳密解法と近似解法を提案する．

4 分散 d-MST 問題

d-MST 問題に対する多くの既存手法は集中型の解法であるが，本研究では d-MST 問題のような組み合わせ最適化問題は，例えば通信網の構築に適用する場合，分散された資源 (通信線) を基に問題を解くため，マルチエージェントシステムの協調問題解決の枠組みにおいて解くことが有用であると考える．そこで，各エージェントに変数と評価関数が分散して配置された分散 d-MST 問題として d-MST 問題の形式化を行う．なお，本研究の形式化はエージェントに分散して配置された変数と制約・評価関数の情報を基に分散アルゴリズムにより解を得る分散制約最適化問題への適用に向けた基礎的な位置付けとする．

4.1 形式化

3 章で述べたように，分散制約最適化問題を扱う関連研究 [11] に類似した形式化を行う．関連研究 [11] が電力網の経路割り当て問題におけるフィーダ木の構築に特化した形式化を行うのに対し，本研究では，より一般的な木の構築問題である d-MST 問題に類似の形式化を適用する．分散 d-MST 問題の構成要素と評価関数を以下のように定義する．

- マルチエージェントシステムに含まれる n 個のエージェントの集合を $A = \{i|i = 0, 1, \ldots, n-1\}$ で表す (V と同等).
- 変数の集合を $X = \{x_i|i = 0, 1, \ldots, n-1\}$ で表す．各エージェント $i \in A$ は，自身の選択した辺を表す変数 $x_i \in X$ を持つ.
- エージェント i の隣接エージェント集合を Nbr_i で表す.
- 変数 x_i の値域を D_i で表す．$D_i = Nbr_i \cup \{\emptyset\}$ とし，x_i の変数値 $d_i \in D_i$ は，エージェント i のみが決定可能であるとする.
- 各エージェント i は評価関数 f_i を持つ．f_i は自身と関連する変数値から評価値への写像を表す.

$$f_{cost_i}(i) = \begin{cases} c(i, x_i) & i \neq x_{x_i} \\ 0 & x_i = \emptyset \\ \infty & \text{otherwise} \end{cases} \tag{1}$$

$$f_{deg1_i}(i) = \begin{cases} 1 & f_{cost_i}(i) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$f_{deg2_i}(i, j) = \begin{cases} 1 & x_j = i \wedge x_i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

上記の構成要素と評価関数を用いて，以下の 2 つの制約を満たす必要がある.

- 木制約: $T = (A_T, E_T)$ は生成木を成す (T は，連結かつ閉路がなく，$A = A_T$).
- 次数制約: $\forall i \in A$ に対して，$d(i) \leq b_i$ を満たす.

この形式化においては，DCOP[11, 13] と同様に n 個のエージェントの集合 A を定義する．エージェント集合 A に属するエージェント i はそれぞれ 1 つの変数 x_i を持ち，自身の変数値のみ決定可能である．エージェント i がこの変数 x_i の値を決定するとき，d-MST の辺として辺 (i, x_i) が選択されることを表す．基本的にはエージェント i は自身の隣接辺を必ず 1 本選ぶこととする．ただし，n 個のエージェントが全ての変数値を決定した場合，n 本 の辺が選択されることを意味し，生成木であるための制約 $|E_T| = |A_T| - 1$ を満たさない．そのため，n 個のエージェントのうち，ある 1 つのエージェントは自身の変数値を $\emptyset$ と決定することで，辺を選択しない必要がある．従って，変数 x_i の値域 D_i は，d-MST の辺として選択すべき辺の端点となる隣接エージェント集合 Nbr_i と空集合 phi の和集合 $Nbr_i \cup \{\emptyset\}$ で表すこととする．また，あるエージェントと別のエージェントが，それぞれ同じ

辺を選択してしまった場合，辺の選択が競合するため，生成木であるための制約 $|E_T| = |A_T| - 1$ を満たさない．これを辺の整合性の矛盾と呼ぶこととする．この辺の整合性の矛盾は式 (1) にも表される．コスト関数 f_{cost_i} は，変数値の決定により選択される辺のコストを評価するものである．$i \neq x_{x_i}$ のとき，エージェント i が選択した辺 (i, x_i) と他のエージェント x_i が選択した辺が競合しないことを表し，辺のコスト $c(i, x_i)$ が評価される．また，$x_i = \emptyset$ のとき，エージェント i は辺を選択しないため，辺のコストは 0 として評価される．また，それ以外の場合は，各エージェントが異なる辺を選択せず，競合が発生することを表すため制約違反とする．この辺の整合性の矛盾については，4.2 で詳しく述べる．この形式化においては，ある 1 つのエージェントが辺を選択せず，それ以外の $n-1$ のエージェントが，それぞれ異なる辺を選択するとき，全てのエージェントの変数値の組み合わせは生成木を表すことができる．

図 4 の左側のグラフのような問題において，例えば，エージェント 2 の隣接エージェント $Nbr_i = 0, 1, 3, 4$ であるため，$D_2 = \{0, 1, 3, 4, \emptyset\}$ となる．エージェント 2 が $x_2 = 1$ と変数値を決定するとき，自身の選択する辺を $(2, 1)$ に決定することを意味する．エージェント 2 が $x_2 = \emptyset$ とするとき，辺を選択しないことを意味するため，エージェント 2 の隣接辺は d-MST の辺として選択されない．例えば，エージェント 1 が自身の変数値を $x_1 = \emptyset$ と決定し，それ以外の $n-1$ のエージェント $0, 2, 3, 4, 5$ が，自身の変数値をそれぞれ $x_0 = 1, x_2 = 1, x_3 = 2, x_4 = 2, x_5 = 4$ と決定するとき，図 4 の矢印で表される辺からなる生成木を表す．この生成木 $T = (V_T, E_T)$ においては，$V_T = \{0, 1, 2, 3, 4, 5\}, E_T = \{(0, 1), (2, 1), (3, 2), (4, 2), (5, 4)\}$ であり，図 1 の (c) の d-MST と同等である．

4.2 制約による部分解の候補の削除

4.1 の形式化に基づいて，各エージェントが周辺のエージェントの変数値の情報を集め，自身以外の他のエージェントの変数値を考慮する場合，d-MST 問題の部分問題における部分解は，各エージェントの変数値の組み合わせからなる部分木で表される．従って，変数値の組み合わせにより表される d-MST の解候補は，部分木の集合の形で表される．このとき，ある部分問題における各エージェント i の集合 $A_{T'}$ と，それらの変数値の組み合わせが表す部分木 $T' = (A_{T'}, E_{T'})$ に対して，制約による解候補の削除が可能である．この制約による解候補の削除条件は，以下の 4 つである．

1. 辺の整合性: 各エージェントの辺の選択に矛盾がないかどうか
2. 閉路の包含: 部分木が閉路を含むかどうか
3. 次数の超過: 次数制約の制限数を超えないかどうか
4. 部分木の重複: 部分木が重複していないかどうか

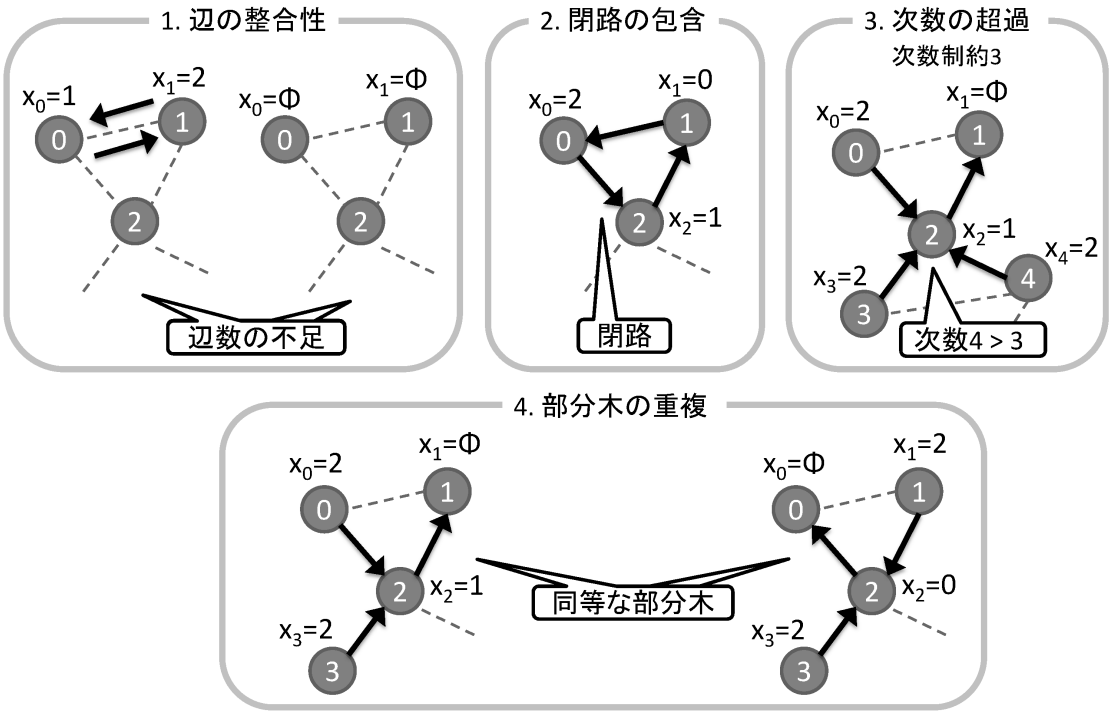

図 5: 制約による解候補の削除条件

1. の辺の整合性とは，ある 2 つのエージェントが変数値を決定した場合，その変数値の決定によって同一の辺を選択することで辺の選択が競合してしまったり，2 つ以上のエージェントが辺を選択しない状態となることによる矛盾が生じないかどうかである．この矛盾とは最終的に生成木構築するにあたり，辺数が不足しないかどうかを意味する．例えば，図 5 の 1. の左側のグラフのように，エージェント 1 が辺 $(1,2)$ を，エージェント 2 が辺 $(2,1)$ を選択した場合，部分木 $T' = (A_{T'} = \{0,1\}, E_{T'} = \{(0,1)\})$ が構築される．同一の辺を選択したことにより，この時点で，辺数が不足し木制約を満たさないことがわかる．この場合，4.1 の形式化における全てのエージェントが自身の変数値を決定し，$n-1$ 本の辺を必ず選択するという前提を満たせなくなる．従って，この T' のような部分解の候補の削除が可能である．この制約は式 (1) を用いて，式 (4) のように評価される．

$$\forall i \in A_{T'} \ ; \ f_{cost_i}(i) \neq \infty \tag{4}$$

また，図 5 の 1. の右側のグラフのように，エージェント 1 もエージェント 2 も変数値を $\emptyset$ と決定してしまう場合，部分木 $T' = (A_{T'} = \{0,1\}, E_{T'} = \{\emptyset\})$ が構築される．この部分木においても同様に，辺数が不足し木制約を満たさないことがわかるため，削除可能である．この制約は式 (2) を用いて，式 (5) のように評価される．

$$\sum_{i \in A_{T'}} f_{deg1_i}(i) \geq |A_{T'}| - 1 \tag{5}$$

2. の閉路の包含とは，部分木 T' が閉路を含むかどうかである．例えば，図 5 の 2. のようにエージェント $0, 1, 2$ が自身の変数値を $x_0 = 2, x_1 = 0, x_2 = 1$ と決定する場合，部分木 $T' = (A_{T'} = \{0, 1, 2\}, E_{T'} = \{(0, 2), (1, 0), (2, 1)\})$ が構築される．この部分木においては，$0 \rightarrow 2 \rightarrow 1 \rightarrow 0$ のような閉路が存在し，その時点で木制約を満たさないため，この部分解の候補の削除が可能である．

3. の次数の超過とは，部分木 T' の頂点 $i \in A_{T'}$ の次数が次数制約 b_i を超えないかどうかである．例えば，図 5 の 3. のようにエージェント $0, 1, 2, 3, 4$ が自身の変数値を $x_0 = 2, x_1 = 0, x_2 = 1, x_3 = 2, x_4 = 2$ と決定する場合，部分木 $T' = (A_{T'} = \{0, 1, 2, 3, 4\}, E_{T'} = \{(0, 2), (2, 1), (3, 2), (4, 2)\})$ が構築される．この部分木において，エージェント 2 の次数 4 は制限数 3 を超過し，次数制約を違反するため，この部分解の候補の削除が可能である．この制約は式 (2) と式 (3) を用いて，式 (6) のように表される．

$$\forall i \in A_{T'} \ ; \ f_{deg1_i}(i) + \sum_{j \in Nbr_i \cup A_{T'}} f_{deg2_i}(i, j) \leq b_i \tag{6}$$

4. の部分木の重複とは，各エージェントの変数値の決定により表される，ある部分木と別の部分木が同等の部分木を表すかどうかである．図 5 の 4. の左側のグラフのようにエージェント $0, 1, 2, 3$ が自身の変数値を $x_0 = 2, x_1 = \emptyset, x_2 = 1, x_3 = 2$ と決定する場合と，右側のグラフのように $x_0 = \emptyset, x_1 = 2, x_2 = 0, x_3 = 2$ と決定する場合，これらの表す部分木は，どちらの場合も $T' = (A_{T'} = \{0, 1, 2, 3\}, E_{T'} = \{(0, 2), (2, 1), (3, 2)\})$ となる．このような変数値の組み合わせを解候補として部分木集合に追加すると，同等の部分木が重複し冗長となってしまう．そのため，1 つの部分木だけを残し，あとの同等な部分木は削除が可能である．

上記の制約を全て満たし，全てのエージェントの変数値の組み合わせの候補を決定するとき，最終的に残る部分木集合の部分木は，生成木となる．その上で，式 (7) に示す $Q(T)$ を得ることができる木 T を目的とする．

$$Q(T) = \min \sum_{i \in A} f_{cost_i}(i) \tag{7}$$

問題の形式化における木が生成木であることの正当性について述べる．生成木の条件は，全ての頂点を含み，かつ連結であり，かつ閉路を含まないことである．この問題の形式化では，式 (5) を用いて，ある 1 つのエージェントが辺を選択せず，それ以外の $n - 1$ のエージェントが辺を選択するため，最終的に構築される木においては，どの頂点も必ず隣接辺を持ち，かつ辺数は必ず $n - 1$ となる．これは木が全ての頂点を含み，かつ連結であることを意味する．また，閉路を形成する辺の追加については，その時点で解候補から除外するため，必ず閉路は含まない．従って，生成木の条件を満たす．

5 提案手法 –分散協調探索手法–

本章では，4 章の形式化に基づく分散型 d-MST の厳密解法 (dd-mst) を提案する．また，近似解法として，dd-mst の探索空間を削減する手法と NNT のアルゴリズムに基づく手法と d-prim を分散処理化した手法についても示す．

5.1 dd-mst(厳密解法)

dd-mst は各エージェントがメッセージ交換を伴う協調探索により d-MST 問題の厳密解を得る手法である．関連研究 [11] の手法に類似する手法を d-MST の構築に適用させたものといえる．各エージェントは，隣接エージェントから受信する部分解 (部分木集合) と自身の変数値との組み合わせを新たに部分解として計算し，次のエージェントに引き継ぐ．2 章で述べた関連研究 [11, 16] の解法のように，各エージェントの部分解は，あらかじめ順序付けられた構造に従ってボトムアップに集計される．この順序付けの構造は，前処理として順序木を構築することで得る．順序木において親，子，葉，根にあたるエージェントをそれぞれ親エージェント，子エージェント，葉エージェント，根エージェントと呼ぶこととする．最終的には，順序木に沿って全ての部分解がボトムアップに最上位のエージェントに伝播され，最上位のエージェントがその情報から大域的な解 (d-MST) を計算する．そして，最上位のエージェントが最適解を下位のエージェントに伝播させることで，全てのエージェントがとるべき変数値を決定する．この手法は（1）順序木の生成,（2）部分解の伝播,（3）最適解の伝播の 3 つのフェーズからなる．擬似コード 1 に dd-mst の処理を示す．擬似コード 1 は，エージェント i を主体とした処理である．Phase1，Phase2，Phase3 は，それぞれ（1）順序木の生成,（2）部分解の伝播,（3）最適解の伝播の 3 つのフェーズを表す．擬似コード 1 においてエージェント i の持つ変数，送受するメッセージ，関数の概要は以下の通りである．

＜変数＞

prt_i: エージェント i の親エージェント

$Child_i$: エージェント i の子エージェント集合

ST_i: エージェント i の生成する部分木集合

$T_i(k)$: 部分木集合 ST_i 内の k 番目の部分木

ST_{tmp}, ST_{new}: エージェント i が子エージェントの部分木集合を統合したものを保持するための集合

$OPT(k)$: エージェント k の変数値を格納する配列

$value$: エージェント i の変数の最適値

＜メッセージ＞

PartialSolution(ST): 部分木集合 ST を送信するためのメッセージ

OptimalAnswer(OPT): 最適解の配列 OPT を送信するためのメッセージ

擬似コード 1: dd-mst

```
1  /* The algorithm is excuted by each agent i */
2
3  Phase1 : ordered tree generation
4   select root agent and generate ordered tree;
5   afterwards ∀i ∈ A knows prt_i and Child_i;
6
7  Phase2 : partial solution message propagation
8   if |Child_i| == 0 (that means agent i is the leaf) then
9     foreach d_i ∈ D_i do
10      T_i(k) ← (V_{T_i(k)} ← {i}, E_{T_i(k)} ← {(i, d_i)});
11      ST_i ← ST_i ∪ T_i(k);
12  else
13    execute Upon receipt of PartialSolution(ST_j) message;
14
15 Phase3 : optimal answer message propagation
16  execute Upon receipt of OptimalAnswer(OPT);
17
18 Upon receipt of set PartialSolution(ST_j) from agent j:
19  if the message is first recieved one then
20    ST_tmp ← ST_j;
21  else
22    ST_new ←Unify_Subtrees(ST_tmp, ST_j);
23    ST_tmp ← ST_new;
24  if PartialSolution messages from all children arrived then
25    ST_i ←Add_My_Domain(ST_new);
26    if prt_i != NULL (that means agent i is the root) then
27      send PartialSolution(ST_i) message to agent prt_i;
28    else
29      OPT ← Make_Optimal(ST_i);
30      x_i ← Choose_Optimal(OPT);
31    foreach k ∈ Child_i do
32      send OptimalAnswer(OPT) message to agent k;
33
34 Upon receipt of OptimalAnswer(OPT) from agent j:
35  x_i ← Choose_Optimal(OPT);
36  if |Child_i| != 0 then
37    foreach k ∈ Child_i do
38      send OptimalAnswer(OPT) message to agent k;
39  halt;
40
```

```
41 procedure Unify_Subtrees(ST_tmp, ST_j):
42  foreach T_tmp(k) ∈ ST_tmp = {T_tmp(k)|k = 0, 1, ..., |ST_tmp| − 1} do
43    foreach T_j(m) ∈ ST_j = {T_j(m)|m = 0, 1, ..., |ST_j| − 1} do
44      T_new ← T_tmp(k) ∪ T_j(m);
45      if T_new satisfies all constraints then
46        ST_new ← ST_new ∪ T_new;
47  return (ST_new);
48
49 procedure Add_My_Domain(ST_new):
50  foreach T_new(k) ∈ ST_new = {T_new(k)|k = 0, 1, ..., |ST_new| − 1} do
51    foreach d_i ∈ D_i do
52      T_i(k) ← (V_{T_i(k)} ← V_{T_new(k)} ∪ {i}, E_{T_i(k)} ← E_{T_new(k)} ∪ {(i, d_i)});
53      if T_i(k) satisfies all constraints then
54        ST_i ← ST_i ∪ T_i(k);
55  return (ST_i);
56
57 procedure Make_Optimal(ST_i):
58  foreach T_i(k) ∈ ST_i do
59    if there is T_i(k) that has minimum Q(T_i(k)) then
60      foreach v, u such that (v, u) ∈ V_{T_i(k)} do
61      OPT(v) ← u;
62  return (OPT);
63
64 procedure Choose_Optimal(OPT):
65  foreach k that is index of OPT do
66    if i == k then
67      value ← OPT(k);
68  return (value);
```

5.1.1 (1) 順序木の生成

（1）順序木の生成では，(2）部分解の伝播，(3）最適解の伝播のための前処理を行う．先に述べたように，本手法では，部分解の伝播を行うために何らかの順序を必要とする．従って，あらかじめ生成木のような初期構造を構築することで，問題を解くための順序をエージェントに与える．本手法では，これを順序木と呼ぶ．例えば，図 6 の問題から，(a) の実線で示されるような生成木を作成する．

全てのエージェント i は，dd-mst の開始時に擬似コード 1 の 3-5 行目に示す ordered tree generation を実行し，自身の prt_i，$Child_i$ を決定することで，順序性を得る．例えば，図 6 の順序木の場合，エージェント 2 について $prt_2 = 0$ であり，エージェント 0 は根エージェントであるため，$prt_0 = \emptyset$ である．また，エージェント 2 について $Child_2 = \{1, 4\}$ である．この順序木を用いて，ボトムアップなメッセージ伝播を行うとき，まず，葉エージェントであるエージェント $1, 3, 5$ が，それぞれの親エージェント $2, 4$ にメッセージを送信する．次にエージェント $3, 5$ から

メッセージを受信したエージェント 4 は，エージェント 2 にメッセージを送信する．エージェント 1, 4 からメッセージを受信したエージェント 2 は，エージェント 0 にメッセージを送信する．

なお，dd-mst のように順序木に基づいてエージェントの順位付けを行う場合，順序木の種類によっても探索の内容が変化することが考えられる．例えば，木の探索手法には，代表的なものとして深さ優先探索や幅優先探索がある．深さ優先探索は，根の頂点から探索できる頂点がある限り探索を行う．具体的には，探索先の頂点に 1 つでも隣接頂点が存在する場合はその頂点を探索し，探索先の頂点にそれ以上探索可能な頂点が存在しない場合は 1 つ前の頂点に戻り，また新たな頂点を探索する．最終的には，全ての頂点を探索し終えるまでこれを繰り返すことで，根の頂点から全頂点への経路を求める．従って，dd-mst における順序木として利用する場合，構築される木の深さは大きくなるため，メッセージ伝播による時間的コストは大きくなると考えられる．メッセージ伝播による時間的コストとは例えば，葉から根へのメッセージホップ数である．反対に，各頂点の次数は小さくなる傾向にあるため，各エージェントの担当する部分解の計算の負担は平均的に小さくなることが考えられる．また，深さ優先探索が頂点を辿れる限り縦に進むのに対して，幅優先探索は隣接している頂点がある限り横に探索を繰り返す．まず，根の頂点から隣接した全ての頂点に対して探索を行う．同様の処理を探索先に対しても繰り返していき，根の頂点から全頂点への経路を求める．従って幅優先探索を用いた順序木を dd-mst に利用する場合，深さ優先探索を用いた場合とは逆の特徴がある．幅優先探索木の幅が大きくなるため，メッセージ伝播のコストは大きくなる．反対に，各頂点の次数は大きくなる傾向にあるため，メッセージ伝播のコストは小さくなると考えられる．

また，本研究では分散協調問題解決の枠組みで問題を解くため，本来ならば順序木の生成の処理も分散処理化することが望ましいと考える．分散処理化を行う場合，メッセージ伝播による木の構築コストを見積もる必要がある．これに関しては，例えば，分散型の深さ優先探索 [1, 12] の様な探索手法を用いれば，$O(n)$ で構築可能であることが研究されている．これらも含めて，分散型の木の探索手法には様々なものが存在する．しかしながら，それら全ての探索の処理について内容を議論すると，d-MST の分散協調探索手法の議論の本質からは少し外れてしまう可能性がある．そこで，本研究では順序木の生成に関する厳密な議論は行わない．その代わりに，5.1.4 において，木の特徴に基づく探索の変化についての考察を示す．また，6.1 において，3 種類の異なる順序木を用いた場合の探索の比較について予備評価を行う．

5.1.2 (2) 部分解の伝播

（2）部分解の伝播では，図 6(b) で示されるように，順序木の最上位のエージェントまでボトムアップに部分解のデータを伝播していく．各エージェントは子エー

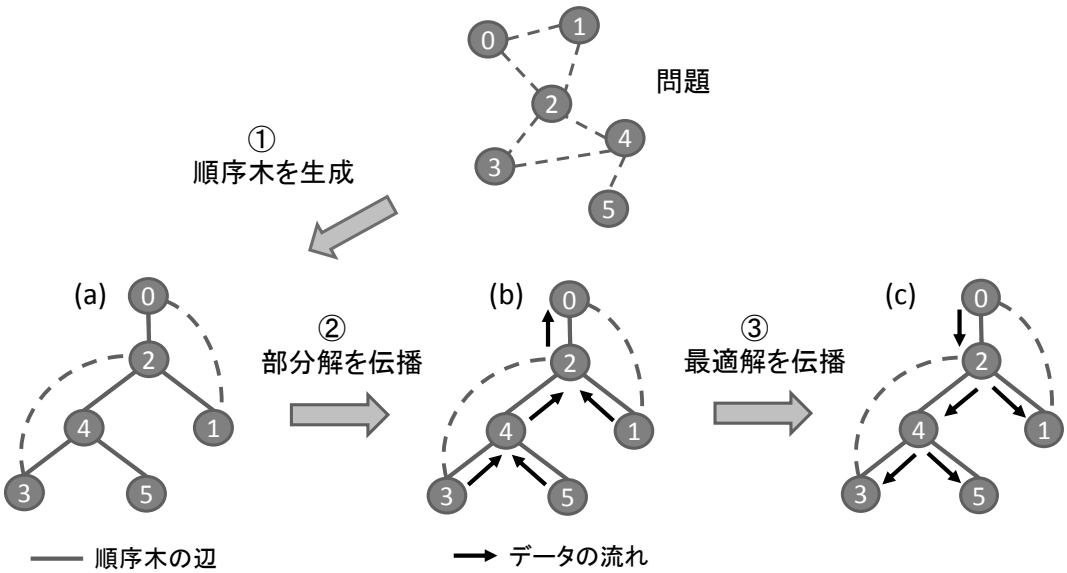

図 6: dd-mst の処理の概要

ジェントから受信した部分解 (部分木集合) と自身の変数値の組み合わせを基に新たに部分解 (部分木集合) を計算する．そして，得た部分解を親エージェントに送信することで，部分解の伝播を行う．部分解 ST の伝播は PartialSolution(ST) メッセージを用いて行う．

部分解の伝播は，順序木の葉エージェント l から開始される．葉エージェント l は，擬似コード 1 の 8-11 行目に示すように，自身の変数 x_l のとりうる値の組み合わせから $|D_l|$ 通りの部分木を計算し，新たな部分木集合 $ST_l = \{T_l(k)|k = 0, 1, \ldots, |D_l| - 1\}$ を計算し，新たな部分木集合を生成する．ここで，部分木集合内の部分木の要素はいずれも，自身の id が示す頂点と，自身と隣接したエージェントの id を端点とした辺のみである．これらの部分木の全ては 4.2 で述べた辺の整合性，閉路の包含，次数の超過，部分木の重複に関する解候補の削除条件には該当せず，d-MST に関する全ての制約を必ず満たすため，制約による解候補の削除条件の判定は行わない．従って，生成される部分木 $T_l(k)$ の，全てを部分木の集合 ST_l に追加し，ST_l を親エージェント prt_l へ送信する．

また，葉エージェント以外で親エージェントでないエージェント i は，全ての子エージェントから部分解を受信し，部分解の統合を行う必要がある．その上で，得られた部分解と自身の変数値との組み合わせから，新たな部分解を計算する．簡単な例として，ある節のエージェント i が部分解 ST_i を生成する場合を考える．エージェント i が，子エージェント a, b, c の部分解 ST_a, ST_b, ST_c を統合し，その統合した結果 S_{abc} と自身の変数値の組み合わせを掛けあわせて，部分解を生成するとき，以下の処理を行う．

1. $ST_{abc} = ST_a \oplus ST_b \oplus ST_c$

2. $ST_i = ST_{abc} \oplus D_i$

1.2. の $\oplus$ は直和を表し，共通部分集合が空集合であるような 2 つの集合の和集合を意味する．各エージェントの部分解は部分木の集合で表されるため，その部分木集合内の部分木の要素の全ての組み合わせに対して，和集合を計算することとなる．全ての子エージェントの部分解の直和を計算し，自身の値域との和集合をとることで，最終的な部分解を得る．

擬似コード 1 の 19-23 行目に示されるように，葉エージェント以外で親エージェントでないエージェント i は，子エージェント j から部分解 ST_j を受信する度に，その受信した部分解 ST_j と，前回までに統合した部分解 ST_{tmp} との直和を計算する．子エージェントの部分解の統合の計算は 41-47 行目に示される **procedure** Unify_Subtrees を用いて行い，自身の値域を考慮した最終的な部分解の計算は 49-55 行目に示される **procedure** Add_My_Domain を用いて行う．

procedure Unify_Subtrees では，エージェント i は，子エージェント j から受信した部分解 ST_j と，前回までに統合した部分解 ST_{tmp} との直和を計算する．このとき $|ST_{tmp}| \times |ST_j|$ 通りの部分木の和集合を計算し，その部分木集合 $ST_{new} = \{T_{new}(k) | k = 0, 1, \ldots, |ST_{tmp}| \times |ST_j| - 1\}$ を新たに生成する．ただし，部分木集合 ST_{new} には，45 行目の all constraints を満たす部分木のみが追加される．この all constraints を満たすことは，4.2 で述べた辺の整合性，閉路の包含，次数の超過，部分木の重複に関する解候補の削除条件には該当せず，d-MST に関する全ての制約を必ず満たすということを意味する．従って，実際には制約条件を満たさない部分木は削除されていくため，$|ST_{tmp}| \times |ST_j|$ のような掛け算で表される組み合わせの数よりも少ない数の部分木を含む集合が生成されることとなる．

そして，**procedure** Add_My_Domain では，全ての子エージェント $Child_i$ から受信した部分解を統合したエージェント i は，統合した部分木集合 S_{new} の部分木と，自身の変数 x_i のとりうる値との和集合から $(|S_{new}| \times |D_i|) - 1$ 通りの部分木を計算し，部分木集合 $ST_i = \{T_i(k) | k = 0, 1, \ldots, |S_{new}| \times |D_i| - 1)$ を生成する．53 行目の all constraints も 45 行目と同様の制約条件を意味し，もし，部分木 $T_i(k)$ が，全ての制約を満せば，その部分木を部分木集合 ST_i に追加する．従って，この場合も実際には制約条件を満たさない部分木は削除されるため，$(|S_{new}| \times |D_i|)$ よりも少ない数の部分木を含む集合が生成される．最後に，PartialSolution メッセージを用いて，その集合 ST_i を親エージェント prt_i へ送信する．子エージェント i から部分解 S_i を受信したエージェントも同様の処理を繰り返し，根エージェントまで部分解を伝播する．

5.1.3 (3) 最適解の伝播

(3) 最適解の伝播では，図 6(c) のように，(2) 部分解の伝播により導き出される最適解を根エージェントから葉エージェントへ伝播する．根エージェント r は，擬似コード 1 の 28-30 行目に示されるように，まず **procedure** Make_Optimal を用

	分枝係数 (大)	分枝係数 (小)
サイクル数	小	大
節での計算量	大	小

表 1: 分枝係数による違い

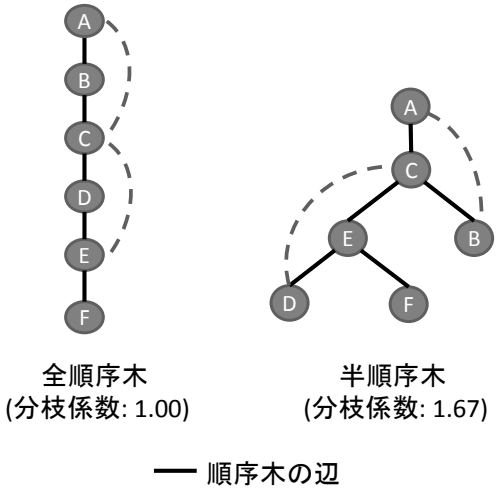

図 7: 順序木の種類

いて最適解を計算し，次に **procedure** Choose_Optimal を用いて，自身の最適な変数値を決定する．最後に全ての子エージェントに対して OptimalAnswer(OPT) メッセージを送信する．

procedure Make_Optimal では 57-62 行目に示すように，次数制約を満たし，かつ部分木の辺の総コストを最小化する生成木を最適解 (d-MST) として選び，その d-MST を構成する子孫のエージェントのとるべき変数値の配列 $OPT(k) = \{k|k = 0, 1, \ldots, n-1\}$ を生成する．ここで，次数制約や木制約を判定する必要がないのは，45 行目と 53 行目の all constraints の条件によって，制約を満たす木のみを選択するからである．

そして，**procedure** Choose_Optimal では，64-68 行目に示されるように，根エージェント r は自身の変数 x_r を $OPT(r)$ に決定する．すなわち，自身の id である r と変数値 x_r を端点とする辺 (r, x_r) を選択することを意味する．そして，OptimalAnswer(OPT) メッセージを用いて，子エージェント $Child_r$ へ OPT を送信する．prt_i から OPT を受信したエージェント i は，同様にして自身の変数 x_i を $OPT(i)$ に決定し，OPT を $Child_i$ へ送信する．同様の処理を繰り返し，最終的に，葉エージェント l が x_l を決定したとき，全てのエージェントが辺を選択したこととなるため，処理を終了する．

5.1.4 計算の複雑度

5.1.2 で述べたような部分解の計算方法を用いるため，問題の複雑度は順序木の性質や辺の密度に影響する．

順序木に関しては，直感的には木の分枝係数によって，問題の複雑度が変化する．分枝係数とは，その木において節の頂点が子の頂点をどれくらい持つかとい

う度合いの指標である．分枝係数は子を 1 つ以上持つ頂点の子の頂点数の和を子を 1 つ以上持つ頂点数で割った数である．この分枝係数の大小により異なると考えられる特徴を表 1 に示す．サイクルとはエージェントがメッセージを受信し，そのメッセージに基づく局所的な処理を実行し，その処理に基づくメッセージを他エージェントに送信する一連の動作の単位である．分枝係数が大きければ，エージェント間でのメッセージ交換のサイクル数が小さくなるが，節のエージェントの計算量は大きくなる．反対に，分枝係数が小さければ，サイクル数が大きくなるが，節のエージェントの計算量は小さくなると考えられる．例えば，図 7 に示す分枝係数の異なる全順序木と半順序木では，解探索の内容に差が出ると考える．順序木の中でも分枝係数が最も小さい全順序木では，最もサイクル数が大きくなる．それよりも分枝係数が大きい半順序木は，各エージェントにおける計算の並列性が増し，サイクル数が小さくなると考えられる．

また問題の辺の密度が各エージェントの計算量に大きく影響する．それは，制約を考慮しない場合の組み合わせ数が，エージェント i の隣接辺と空集合の和集合で表される値域 Di を用いて，以下のような式で表されるためである．

- 制約を考慮しない場合の組み合わせ数 $= \prod_{i \in A} |D_i|$

このような掛け算で表される制約を考慮しない場合の組み合わせ数は，各エージェントの隣接辺数に依存するため，辺密度が大きいときに組み合わせ数は膨大となる．また，部分解の伝播においては下位のエージェントの部分解を引き継ぐため，メッセージホップ数が増加すると，組み合わせの掛け算の回数が増加する．

5.2 dd-mst を基礎とする優先探索を用いた近似解法

dd-mst への貪欲的手法の適用のための検討として，探索空間削減手法を示す．5.1.4 で述べたように dd-mst は，仮に制約を一切考慮しない場合，探索すべき最大の組み合わせは，$\prod_{i \in A} |D_i|$ となり，指数関数的に増加する．4.2 で述べたように，各エージェントにおいて制約を違反する部分木を削除することにより，その組み合わせ数 (部分木集合における要素数) は削減される．しかしながら，削減が不十分であれば，組み合わせ数は爆発的に増大する．従って，実際には各エージェントの送受するメッセージの容量や，各エージェントのデータの記憶領域の容量を考慮し，送信する部分解の容量と，保持する部分解の容量を制限することが望ましい．すなわち，送信，保持する部分木の数を k 個に制限する．そこで，本研究では，dd-mst において探索空間を削減するために，部分木の特徴に基づく優先探索を用いる 5 つの手法を示す．これらの手法については，それぞれ 5.2.1，5.2.2，5.2.3，5.2.4，5.2.5 において詳しく述べる．なお，これらの手法は，分散探索手法における探索空間の大きさと，解の質，精度とのトレードオフの関係に対する基本的なヒューリスティクスとして用いる．

また，dd-mst は厳密解法であるため，必ず実行可能解を得ることができるが，部分解の容量の制限を加えて近似解法とする場合，各エージェントが k を超える組み合わせを切り捨てることにより，最終的な解が実行不可能解に陥る可能性がある．そこで実行不可能が発生する場合は，エージェントの順序を定める生成木が次数制約を満たすものであれば，ベースラインとして，その木を流用できるため，$x_i = prt_i$ とすることで解を確保する．この実行不可能解に陥った際にベースラインとして生成木を流用する処理は擬似コード 1 には示していない．実行不可能解に陥ったことを検知するには，例えば，エージェント i が部分解 ST_i を生成する際，$|ST_i|$ の値を調べればよい．もし，$|ST_i| = 0$ の場合，実行不可能解に陥ってしまったことを検知して，他の全てのエージェントに，その旨を通知することで，順序木の辺を d-MST の辺として選択させることが可能である．ただし，流用した順序木が次数制約を満たしていなければ，2.1 で述べたような次数制約が緩和され，コストが最小化されない木を解とすることになる．従って，順序木の流用は，あくまで生成木を得るための最終手段としての位置付けとする．なお，厳密に次数制約を満たしたい場合は，順序木の生成において d-prim などの近似解法を適用し，次数制約を満たすことを保証した上で解の計算を行うなどの方法が考えられるが，本研究の本質からは外れるため，これ以上の議論はしない．

5.2.1 部分木の計算順序に基づく優先探索

各エージェントの計算する部分木の計算順序に基づく優先探索による組み合わせ削減手法である．5.1.2 で述べた擬似コード 1 の **procedure** Add_My_Domain と **procedure** Unify_Subtrees の処理のように，各エージェントは受信した部分木集合を統合し，その結果と自身の変数値との組み合わせを計算し，新たな部分木を生成していく．この部分木を順に生成するとき，先頭から $k-1$ 番目までの部分木のみを部分木集合として保持し，k 個を超える組み合わせは切り捨てる．

単純に計算順序に関して部分木を残した場合，問題の規模が増大したり，パラメータ k の値が小さくなると，$Q(T)$ が大きくなるか，もしくは実行不可能解が発生する可能性が増大すると考えられる．問題の規模は各エージェントのグラフの辺密度に大きく依存する．また，5.1.4 で述べたように探索すべき最大の組み合わせは $\prod_{i \in A} |D_i|$ で表され，指数関数的に増大する．従って，辺密度が大きい場合，部分解の伝播が進むにつれて，各エージェントにおいて将来的に最適解もしくは準最適解となる部分木が計算される順番が k よりも遅くなる可能性が急激に増大する．その場合，最終的に求まる d-MST 問題の解の $Q(T)$ は大きくなるか，実行不可能解となってしまうと考えられる．そのような解の質の低下はパラメータ k の値を大きく設定することで，ある程度は回避できると考えるが，k の値が大きい場合は各エージェントの組み合わせの計算量やメッセージサイズが増大してしまうという問題がある．

5.2.2 部分木の総コストに基づく優先探索

各エージェントの計算する部分木の総コストに基づく優先探索による組み合わせ削減手法である．5.2.1 の計算順序に基づく優先探索では，解の質を簡単に低下させてしまう可能性があるため，部分木の総コスト値に基づいて，部分解として残す部分木の優先順位を決定することを考える．

この優先順位付けは，dd-mst の部分解の計算の過程において実現しなければならない．5.1.2 で述べた擬似コード 1 の **procedure** Unify_Subtrees における子エージェントから受信した全ての部分木集合の各部分木の和集合の計算と，**procedure** Add_My_Domain における統合した部分集合に含まれる部分木と自身の変数値の候補が表す辺との和集合の計算では，集合内の部分木をそれぞれ 1 つずつ徐々に演算する．この過程において総コスト値の小さい順に和集合演算を行うことで，部分木の総コストに基づく優先順位付けを実現する．また，部分木集合内の部分木の総コスト値の大小を比較する際，その部分木の辺数が異なる場合がある．この場合，辺数の少ない部分木の方が総コスト値が小さい可能性が高いため，そのまま単純にコスト値のみを比較するのは望ましくないと考える．そこで，部分木の総コスト値を辺数で割った値を優先順位付けに用いることとする．具体的には，ある部分木 T_A が，別の部分木 T_B よりも優先されるとき，それぞれの総コスト $Q(T_A)$, $Q(T_B)$ と辺数 $|E_{T_A}|, |E_{T_B}|$ に対して，$Q(T_A)/|E_{T_A}| < Q(T_B)/|E_{T_B}|$ という関係が成り立つ．

計算順序に基づく優先探索に比べて，最終的に求まる d-MST の $Q(T)$ が小さくなることと，実行不可能解が発生する確率を，ある程度下げることが期待できると考える．しかしながら，この手法はあくまでコスト値だけを優先するため，次数の制約も考慮しなければならない d-MST 問題において，実行不可能解の発生確率を抑えることに対しては十分でないと考える．

5.2.3 部分木の次数に基づく優先探索

各エージェントの計算する部分木の次数に基づく優先探索による組み合わせ削減手法である．次数に基づくとは，例えば，部分木の次数の余剰数の最小値や，次数のばらつき度合いを考慮することである．5.2.2 で述べた部分木の総コスト値と同様に，これらの値を基準とした部分木の優先順位を付けることが可能である．

部分木の次数の余剰数とは，頂点 i のと次数制約 b_i と次数 $d(i)$ の差 $b_i - d(i)(> 0)$ である．部分木の各頂点の余剰数の中の最小値を優先順位付けの指標として用いる．具体的には，ある部分木 T_A が，別の部分木 T_B よりも優先されるとき，それぞれの次数の余剰数の集合 $R(T_A), R(T_B)$ に対して，$min\{R(T_A)\} > min\{R(T_B)\}$ という関係が成り立つ．従って，この次数の余剰数の最小値を基準とする場合に関しては，より大きい値をもつ部分木が優先される．次数の余剰数の最小値が大きい方が直感的には部分木の各頂点に対してより辺を接続しやすい部分木となり，将来的に実行可能解となる可能性が大きいと考えられるためである．

一方，次数のバラつき度合いは，部分木の次数の分散や，中央値を用いることが考えられる．これらを用いる場合，各頂点にどれくらい均等に辺が接続されているかという目安を計ることができる．この値が小さければ，直感的には各頂点の次数が均一な部分木となり，将来的に実行可能解となる可能性が大きいと考える．6 の実験においては，次数のばらつきを考慮するための指標として四分位数範囲 (IQR) を用いることとする．四分位数とはデータを小さい順に並べたとき，データを 4 つに分ける場所の数である．例えば，第一四分位数とは下から 4 分の 1 のところのデータを意味する．四分位数範囲とは第 1 四分位数と第 3 四分位数の差である．この数は標準偏差や分散の代わりに用いられることがあり，データに偏りがある際の平均値のずれの影響を受けにくい数である．

次数に基づく優先探索は総コスト値に基づく優先探索に比べて，最終的に求まる d-MST の $Q(T)$ は大きくなるが，実行不可能解が発生する確率を下げることが期待できると考える．

5.2.4 部分木の総コストと次数に基づく優先探索

5.2.2 と 5.2.3 で述べたような部分木の総コストと次数の両方に基づく優先探索を用いた組み合わせた削減手法である．5.2.2 と 5.2.3 の評価指標を同時に考慮できるような評価指標を用いて，同様に部分木優先順位付けを行うことができる．

例えば，5.2.2 の部分木の総コスト値を辺数で割った値と 5.2.3 の部分木の次数の余剰数の最小値を用いる場合を考える．ある部分木 T_A が，別の部分木 T_B よりも優先されるとき，総コスト値を辺数で割った値をそれぞれ $QE(T_A), QE(T_B)$，次数の余剰数の集合を $R(T_A), R(T_B)$ とすると，$QE(T_A)/min\{R(T_A)\} < QE(T_B)/min\{R(T_B)\}$ という関係が成り立つ．すなわち，部分木の総コストを辺数で割った値が小さく，次数の余剰数の最小値が大きいほど，この評価値は小さくなる．

総コストと次数の両方を考慮するで，最終的に求まる d-MST の $Q(T)$ を小さくすることと，実行不可能解が発生する確率を下げることが期待できると考える．

5.2.5 部分木の総コストと次数の分布に基づく優先探索

部分木の総コストや次数に関する値の分布に基づく優先探索による組み合わせ削減手法である．5.2.2 や 5.2.3 のように部分木の総コストや次数の分散の値に基づいて優先探索を行う以外にも，部分木の特徴に関する分布に基づいて部分木を採用する手法も考えられる．

例えば，部分木の特徴に応じて，部分解として採用する数の頻度を変える手法が考えられる．dd-mst のように各エージェント単位で部分木の演算を徐々に行う計算では，この手法に対するヒューリスティクスの実現が難しい．さらに，部分木の様々な特徴を考慮し部分解として採用する頻度を変えるポリシーを厳密に実現しようとすると，部分解を詳細に解析する必要もあるため，優先探索のための

計算量が増大し，逆に計算量が増えてしまう可能性も考えられる．そこで，6 章においては，簡単のために総コスト値に基づいて優先する部分木と次数に基づいて優先する部分木をそれぞれ半分ずつ考慮して優先探索を行うこととする．

この優先の度合いが，その問題に順応すれば，実行可能解が発生する確率を大幅に下げることが期待できると考える．

5.3 NNT を基礎とする近似解法

dd-mst とは異なる分散協調探索を用いる比較手法として，3.1.3 で述べた NNT アルゴリズム [8] に，次数制約を追加した手法を示す．NNT は MST を構築する手法であるが，これに簡単なヒューリスティクスを適用することで次数制約を満たす分散 d-MST 問題の近似解法とする (d-nnt)．

3.1.3 で述べたように NNT アルゴリズムでは，リーダー以外の全てのエージェントがランクを用いて，辺の選択先を 1 つだけ選ぶ．そのため，4.1 で述べた辺を選択するための変数を用いる形式化を適用できる．各エージェントが受信した部分木集合と自身の値域との組み合わせを計算する dd-mst に対し，d-nnt では各エージェントが自身の値域に関してのみ計算を行うことで，探索空間の増大を線形に保つことが可能である．

擬似コード 2 に d-nnt の処理を示す．擬似コード 2 は，エージェント i を主体とした処理である．エージェント i の持つ変数，送受するメッセージ，関数の概要は以下の通りである．

＜変数＞

$rank$**:** エージェント i の持つランク

$setrank\_flag$**:** エージェント i が $rank$ を設定したかどうかを表すフラグ

deg_count**:** エージェント i の次数のカウンター

$SE(i,j)$**:** 隣接辺 (i,j) の状態を保持する配列．$Basic$(初期状態)，$Branch$(辺 (i,j) が選択された状態)，$Rejected$(辺 (i,j) が選択されない状態) の 3 状態をとる

＜メッセージ＞

SetRank(r)**:** 各エージェントにランクの設定を開始させるためのメッセージ

RankOK: ランクの設定を完了した旨を通知するためのメッセージ

Connect(r)**:** 辺を選択することを要求するためのメッセージ

Available: 選択候補の辺が利用可能であることを通知するためのメッセージ

UnAvailable: 選択候補の辺が利用不可能であることを通知するためのメッセージ

StartConnect: 各エージェントに辺の探索と選択を開始させるためのメッセージ

SubtractDegree: 親エージェントの次数を減らすためのメッセージ

SubtractOK: エージェントの次数を減らしたこと旨を通知するためのメッセージ

擬似コード 2: d-nnt

```
1  /* The algorithm is excuted by each agent i */
2
3  procedure Init_Process:
4   if prt_i == NULL(that means agent i is the root) then
5     deg_count ← |Child_i|;
6     rank ← random number;
7     setrank_flag ← TRUE;
8    foreach k ∈ Child_i do
9      send SetRank(rank) message to agent k;
10  else
11    deg_count ← |Child_i| + 1;
12
13 Upon receipt of SetRank(r) message from agent j:
14  if setrank_flag == FALSE then
15    if |Child_i| != 0 then
16      setrank_flag ← TRUE;
17      rank ← random number chosen from [r − 1, r);
18      foreach k ∈ Child_i do
19        send SetRank(rank) message to agent k;
20  else
21    send RankOK message to agent prt_i;
22
23 Upon receipt of RankOK message from agent j:
24  if RankOK messages from all children arrived then
25    if prt_i != NULL then
26      send RankOK message to agent prt_i;
27    else
28      foreach k ∈ Child_i do
29        send StartConnect message to agent k;
30
31 Upon receipt of Connect(r) message from agent j:
32  if j ∈ Child_i then
33    SE(i, j) ← Branch;
34    send Available message to agent j;
35  else
36    if r > rank and deg_count < b_i then
37      deg_count ← deg_count + 1;
38      SE(i, j) ← Branch;
39      send Available message to agent j;
40    else
```

```
41      SE(i,j) ← Rejected;
42      send Unavailabe message to agent j;
43
44  Upon receipt of Available message from agent j:
45   if j != prt_i then
46     send SubtractDegree message to agent prt_i;
47   else
48     foreach k ∈ Child_i do
49       send StartConnect message to agent k;
50     halt
51
52  Upon receipt of UnAvailable message from agent j:
53   SE(i,j) ← Rejected;
54   if there is agent k such that edge(i,k) has the minimum weight then
55     send Connect(rank) message to agent k;
56   else
57     SE(i,prt_i) ← Branch;
58     send Connect(rank) message to agent prt_i;
59
60  Upon receipt of StartConect message from agent j:
61   if prt_i != NULL  then
62     if there is minimum weight edge such that SE(i,k) == Basic  then
63       send Connect(rank) message to agent k;
64
65  Upon receipt of SubtractDegree message from agent j:
66   deg_count ← deg_count − 1;
67   send SubtractOK message to agent j;
68
69  Upon receipt of SubtractOK message from agent j:
70   foreach k ∈ Child_i do
71     send StartConnect messsage to agent k;
72   halt
```

5.3.1 順序木の生成

d-nnt は，NNT と同様に辺の選択を行うために，ランク値に基づく順序関係を必要とする．そこで，dd-mst のように順序木を利用して，順序木に沿ったメッセージ伝播によりランク値を決定する．この順序木として，次数制約付き生成木を構築する．この次数制約付き生成木は，あらかじめ次数制約を満たすため，その状態から，辺のコストの和をできるだけ最小化できるように辺の選択を行う．なお，順序木は dd-mst と同様に前処理によって既に構築されているものとして処理を開始する．従って，擬似コード 2 の処理の開始時点で，各エージェント i は親エージェント prt_i と子エージェント集合 $Child_i$ を設定済みであるとする．

この順序木に基づいてランク値を得るために，擬似コード 2 の 3-29 行目に示さ

れる処理によって，根エージェントから葉エージェントにランク値を伝播していく．まず，各エージェントは，3-11 行目に示される **procedure** Init_Process を用いて初期化を行う．このとき自身の次数 deg_count を順序木に基づく次数に設定する．順序木の根エージェントは，ランダムな数を自身の $rank$ に設定し，全ての子エージェントに SetRank($rank$) メッセージを送信することでランクの設定を開始する．親エージェントから SetRank(r) メッセージを受信したエージェントも同様に，自身の $rank$ を設定し，子エージェントに送信する．そして，葉エージェントまで SetRank(r) メッセージが伝播されたとき，全てのエージェントが自身のランク値 $rank$ の設定を完了する．この過程で，3.1.3 で述べた NNT においては，親エージェント v からランダムな値 $p(v)$ を受信した近傍エージェント u は，$[p(v)-1, p(v))$ から $p(u)$ を選択し，ランク値 $r(u) = (p(u), id(u))$ を決定するというランク値の選択方法を用いるが，擬似コード 2 においては，簡単のために，受信したランク r を用いて $[r-1, r)$ の範囲から値を選択する．従って，あるエージェントは順序木における自身より上位のエージェントよりも小さいランク値を必ず持つ．もし，順序木において，あるエージェントと同位のエージェントが存在する場合，それらのエージェントのランク値は，自身のそれよりも大きい場合と小さい場合の両方がある．

$rank$ を設定し終えた葉エージェントは，24-26 行目に示されるように，親エージェントに対して，RankOK メッセージを用いて $rank$ の設定を完了した旨を通知する．これを根エージェントまで伝播することでランクの設定を完了する．

5.3.2 辺の選択と制限

d-nnt は，辺の選択を決定する過程で $d(i) = b_i$ となるエージェント i は，次数の制限数を超えないようにするヒューリスティクスを NNT[8] に対して追加した手法である．自身の次数が制限数に達したエージェントは，それ以降，他エージェントに自身の隣接辺を選択されることを拒否し，拒否されたエージェントは別の辺の選択を行うことで，次数制約を満たす．ただし，最後まで辺の選択を決定できないエージェントが存在する可能性があるため，その場合は 5.2 で述べたように，あらかじめ生成した順序木の関係に基づき，$x_i = prt_i$ とすることで，解を確保する．

擬似コード 2 において，全てのエージェントが $rank$ を設定し終えた後，31-63 行目に示される処理によって，d-MST の辺として選択する辺の決定を行う．この辺の選択の開始もランクの設定と同様に根エージェントから開始する．RankOK メッセージを受信した根エージェントは，27-29 行目に示されるように，全ての子エージェントに対して StartConnect メッセージを送信する．根エージェントから StartConnect メッセージを受信したエージェント i は，60-63 行目に示されるように，自身と隣接し，最小のコストを持つ $Basic$ な状態の辺の端点となるエージェント k に対して，Connet($rank$) メッセージを送信する．また，$SE(i,k) \leftarrow Branch$ とすることで辺 (i,k) を d-MST の辺として選択する．

Connect(r) メッセージを受信したエージェント i は，Connect($rank$) メッセージを送信したエージェント j が $j \in Child_i$ でなければ,，35 行目に示されるように受信したランク値 r が自身のランク値 $rank$ より大きいかどうかと，自身の次数 deg_count が制限数を超えていないかどうかを確認する．この条件を満たしていれば，Available メッセージを用いて，辺 (i, j) を選択候補とすることを許可する旨を通知する．もし，この条件を満たさなければ，UnAvailable メッセージを用いて，辺 (i, j) を選択候補とすることを拒否する旨を通知する．Available メッセージを送信するときはエージェント u と同様に $SE(i, j) \leftarrow Branch$ とすることで辺 (i, j) を d-MST の辺として選択することを決定し，UnAvailable メッセージを送信するときは $SE(i, j) \leftarrow Reject$ とすることで辺 (i, j) を d-MST の辺として選択しないことを決定する．

Available メッセージを受信したエージェントは，44-50 行目に示されるように，もし親からのメッセージでなければ，自身の子エージェントへ StartConnect メッセージを送信し，処理を終了する．もし親からのメッセージであるならば，SubtractDegree メッセージを返信する．これは，順序木の辺ではなく，別の辺を d-MST の辺として選択することを確定したため，その順序木の辺に対して数えていた分の次数を引くためのメッセージである．SubtractDegree メッセージを受信したエージェントは自身の次数を $deg\_count \leftarrow deg\_count - 1$ とし，$SubtractOK$ メッセージ を用いて，次数を減らした旨を返信する．

また，UnAvailable メッセージを受信したエージェントは，52-58 行目で示されるように，もし自身と隣接し，最小のコストを持つ $Basic$ な状態の端点となるエージェント k があれば，エージェント k に Connect($rank$) メッセージを送信する．辺の選択が決定するか，もしくは他に選択するべき辺の候補が無くなるまで，これを繰り返す．もし，選択するべき辺の候補が無い場合は $SE(i, prt_i) \leftarrow Branch$ として，もとの順序木の辺を d-MST の辺として決定する．最終的に，全てのエージェント i が $SE(i, j) == Branch$ となる辺 (i, j) の選択を決定したとき，処理を終了する．

5.4 Prim を基礎とする近似解法

dd-mst の比較対象として，Prim のアルゴリズム [17] を基礎として分散処理化した手法 (dd-prim) を用意する．dd-prim では Prim と同様に部分木を拡大してくことで解を得るが，部分木に属するエージェントの中でリーダーを設ける．このリーダーは部分木に属するエージェントに周辺の辺を探索させ，d-MST の辺の候補を報告させる．リーダーは集約された辺の候補の情報をもとに，その時点で最適な辺を選択する．なお，このリーダーとなるエージェントは固定ではなく，辺の選択に応じて変わる．

例えば，エージェント s から処理を開始するとき，まずは s をリーダーとして，s のみからなる d-MST の部分木 T' を構築する．そして，s は自身の隣接辺の中で

最もコストの小さい辺 (s,v) を選択し，その端点となるエージェント v と辺 (s,v) を部分木に追加する．このとき，新たに追加されたエージェント v が新しいリーダーとなる．リーダーとなったエージェント v は，エージェント s に隣接辺を探索させ，自身も隣接辺を探索することで周囲の辺の情報を集める．そして，集約された辺の情報をもとに最適な辺を選択し，新たに部分木 T' に追加する．次のリーダーのエージェントも同様にして T' に属するエージェント全てに辺の探索要求を送信するこの探索要求を受けたエージェント i は，自身と隣接し，T' に属さない隣接エージェント $j \in Nbr_i$ とを繋ぐ辺 (i,j) の中で，$c(i,j)$ が最小となり，かつ次数制約を満たす辺を選択候補として選ぶ．そして，その情報をリーダーのエージェントに集約する．各エージェントから情報を得たリーダーのエージェントは，その中でコストが最小の辺を選択する．このように辺の探索と情報の集約を行い，逐次的に辺を選択する処理を反復的に繰り返すことで，d-MST を構築する．

擬似コード 3 に dd-prim の処理を示す．擬似コード 3 においてエージェント i の持つ変数，送受するメッセージ，関数の概要は以下の通りである．

＜変数＞

$flagment\_flag$**:** エージェント i が部分木に属するかどうか判定するフラグ

$leader\_flag$**:** エージェント i がリーダーかどうか判定するフラグ

deg_count**:** エージェント i の次数のカウンター

$find\_count$**:** エージェント i の隣接エージェントに対するメッセージ数のカウンター

$test\_edge$**:** エージェント i の探索する辺の端点となるエージェント

$best\_edge$**:** エージェント i 辺の選択候補を報告してきたエージェント

$best\_wt$**:** 選択候補となる辺のコスト

$in\_baranch$**:** エージェント i に探索要求を出してきたエージェント

$ElectTable$**:** 選択候補となる辺の端点のエージェントとコストを記録する表

$SE(i,j)$**:** 隣接辺 (i,j) の状態を保持する配列．$Basic$(初期状態)，$Branch$(辺 (i,j) が選択された状態)，$Rejected$(辺 (i,j) が選択されない状態) の 3 状態をとる

＜メッセージ＞

Joint: 辺の選択を決定することを要求するメッセージ

Search: 辺の探索を開始させるメッセージ

Looking: 隣接辺を選択候補としてよいか調査するメッセージ

Commit: 辺を選択候補とすることを許可するメッセージ

Abort: 辺を選択候補とすることを拒否するメッセージ

Advise(w): 発見した辺の選択候補の重み w をリーダーに報告するメッセージ

LeaderReduction: リーダーの決定した最適な辺を選択させるためのメッセージ

擬似コード 3: dd-prim

```
1  /* The algorithm is excuted by each agent i */
2
3  procedure Init_Process:
4   find_count ← 0;
5   if prt_i == NULL (that means agent i is the root) then
6     flagment_flag ← TRUE;
7     let k denote the agent that adjacent to agent i and
8     edge(i,k) has the minimum weight
9     SE(i,k) ← Branch;
10    deg_count ← 1;
11    send Joint message to agent k;
12  else
13    deg_count ← 0;
14    flagment_flag ← FALSE;
15
16 Upon receipt of Joint message from agent j:
17  leader_flag ← TRUE
18  flagment_flag ← TRUE;
19  SE(i,j) ← Branch;
20  deg_count ← deg_count + 1;
21  foreach k such that SE(i,k) == Branch do
22    send Search message to agent k;
23    find_count ← find_count + 1;
24  excute procedure Looking;
25
26 Upon receipt of Search message from agent j:
27  best_wt ← ∞
28  best_edge ← NULL
29  in_branch ← j
30  foreach k != j such that SE(i,k) == Branch do
31    send Search message to agent k;
32    find_count ← find_count + 1;
33  excute procedure Looking;
34
35 Upon receipt of Looking message from agent j:
36  if flagment_flag != TRUE then
37    send Commit message to agent j;
38  else
39    if SE(i,j) == Basic then
40      SE(i,j) ← Rejected;
```

```
41    if test_edge != j then
42      send Abort message to agent j;
43    else
44      excute procedure Looking;
45
46  Upon receipt of Commit message from agent j:
47   find_count ← find_count − 1;
48   if deg_count ≤ b_i then
49     add agent j and c(i, j) to ElectTable;
50   else
51     add agent NULL and cost ∞ to ElectTable;
52   excute procedure Advise(ElectTable);
53   if leader_flag == TRUE then
54     excute procedure LeaderReduction;
55
56  Upon receipt of Abort message from agent j:
57   if SE(i, j) == Basic then
58     SE(i, j) ← Rejected;
59   excute procedure Looking;
60
61  Upon receipt of Advise(w) message from agent j:
62   find_count ← find_count − 1;
63   add agent j and cost w to ElectTable;
64   excute procedure Advise(ElectTable);
65   if leader_flag == TRUE then
66     excute procedure LeaderReduction;
67   else if best_wt == ∞ and cost w == ∞ and test_edge == NULL then
68     halt;
69
70  Upon receipt of LeaderReduction message from agent j:
71   excute procedure LeaderReduction;
72
73  procedure Advise(ElectTable):
74   if find_count == 0 and test_edge == NULL then
75     foreach agent k and cost w in ElectTable
76       if w is the minimum cost then
77         best_edge ← k;
78         best_wt ← w;
79     if leader_flag != TRUE then
80       send Advise(best_wt) message to agent in_branch;
81
82  procedure Looking:
83   if there are adjancent edges such that SE(i, k) == Basic and
84      edge(i, k) has the minimum weight then
85     test_edge ← k;
86     send Looking messege to agent k;
87   else
```

```
88    test_edge ← NULL;
89    if leader_flag != TRUE then
90      excute procedure Advise(ElectTable);
91    else
92      if find_count == 0 and test_edge == NULL then
93        excute LeaderReduction;
94
95  procedure LeaderReduction:
96   if find_count == 0 and test_edge == NULL then
97    if SE(i, j) == Branch then
98      send LeaderReduction message to agent best_edge;
99    else
100     SE(i, j) ← Branch;
101     deg_count ← deg_count + 1;
102     send Joint message to agent best_edge;
```

辺の情報の探索と集約は，Search，Looking，Commit，Abort，Advise(w) メッセージと，それらに対する処理である **procedure** Looking，Advise($ElectTable$) を用いて行われる．辺の選択の処理は，Join，LeaderReduction メッセージと，それらに対する処理である **procedure** LeaderReduction を用いて行われる．

まず，各エージェントは，3-14 行目に示される **procedure** Init_Process を用いて初期化を行う．もし，根エージェントならば部分木に属するか属さないかを判定する変数 $flagment\_flag \leftarrow TRUE$ として，自身を部分木に追加する．そして，隣接辺の中で最もコストの小さい辺の端点となるエージェント k に Joint メッセージを用いて，辺 (i, k) を選択することを要求する旨を通知する．このとき，$SE(i, k) \leftarrow Branch$ とすることで辺の選択を決定する．すると 1 本の辺が繋がることになるため，自身の次数 $deg\_count \leftarrow 1$ として次数を 1 とする．もし，根エージェント以外のエージェントならば，$flagment\_flag \leftarrow FALSE$ として，自身を部分木属さないものとする．まだ部分木には属さないため $deg\_count \leftarrow 0$ とする．

エージェント j から Joint メッセージを受信したエージェント i は，17-18 行目に示されるように，$leader\_flag \leftarrow TRUE$ とすることで自身を部分木の新たなリーダーとし，$flagment\_flag \leftarrow TRUE$ とすることで自身を部分木に追加する．また，19 行目に示されるようにエージェント j と同様に $SE(i, j) \leftarrow Branch$ とすることで辺 (i, j) を d-MST の辺とすることを決定する．ここで，1 本の辺が繋がることが決定するため，$deg\_count \leftarrow deg\_count + 1$ として次数を増やす．そして，21-24 行目に示されるように，既に部分木に属している隣接エージェント全てに Search メッセージを送信することで，新しく d-MST の辺として選択するべき辺の探索を開始させ，自身も **procedure** Looking を用いて隣接辺を調査する．調査する，とは隣接辺を d-MST の辺の候補としてよいのかどうかを調べることである．なお，このとき $find\_count \leftarrow find\_count + 1$ とすることで，Search メッセージを送信した隣接エージェントの数を数える．同様に，探索を終えて辺の情報を要求してきたエージェントに対しても $find\_count \leftarrow find\_count - 1$ として数を数え，全て

のエージェントから辺の情報を受信したかどうかを判断する．

Search メッセージを受信したエージェントは 27-28 行目に示されるように選択候補の辺に関する情報を保持するための変数 $best\_edge$ と $best\_wt$ を初期化する．29 行目に示されるように $in\_branch \leftarrow j$ とするのは，探索要求を出してきたエージェント j を記憶し，探索した辺の情報をそのエージェントに報告するためである．エージェント i は，30-32 行目に示されるように $SE(i,j) == Branch$ かつ，エージェント j ではない全てのエージェントに Search メッセージを送信し，辺の探索を開始させる．そして，**procedure** Looking を用いて，82-93 行目に示されるように自身も隣接辺の調査を行う．このとき $SE(i,k) == Basic$ で，コストが最小な辺の端点となる隣接エージェントに Looking メッセージを送信し，辺 (i,k) を選択候補としてよいかどうかを調査する．もし，そのような隣接エージェントが存在しなければ，自身がリーダーでなければ **procedure** Advise を実行する．もし，リーダーであれば，探索を要求中のエージェントがなく，かつ調査している辺がなければ，辺の選択を決定するために **procedure** LeaderReduction を実行する．

エージェント j から Looking メッセージを受信し，辺の探索の要求を受けたエージェント i は，36-40 行目に示されるように，もし自身がまだ部分木に属さなければ，辺 (i,j) を選択候補としてよい旨を Commit メッセージを用いてエージェント j に返信し，もし，既に部分木に属していれば $SE(i,j) \leftarrow Rejected$ とすることで，その辺を d-MST の辺として選択しないことを決定する．また，$test\_edge\,!= j$ とはエージェント j が自身が調査中の辺の端点でないことを示し，その場合は Abort メッセージを用いて，辺の選択候補としてはいけない旨を返信する．$test\_edge == j$ の場合は，エージェント j 側で $SE(i,j) \leftarrow Rejected$ とされるはずであるため，Abort メッセージは送信せず，次の辺を調査するために **procedure** Looking を実行する．

エージェント j から Commit メッセージを受信し，その辺を選択候補としてよいことを確認したエージェント i は，47-51 行目に示されるように，もし自身の次数 deg_count が制限数 b_i を満たしていいれば，相手の id である j と辺のコスト $c(i,j)$ を $ElectTable$ に登録し，そうでなければ id を $NULL$ に，コストを ∞ として $ElectTable$ に登録する．ここで，相手の id を登録するのは，後の辺の選択の決定の処理において，選択する辺 (i,j) を記憶するためである．次数を登録しないのは，部分木に属していないエージェントの次数は必ず 0 であり制限数を超えていることはなく，考慮する必要がないからである．

また，エージェント j から Abort メッセージを受信し，その辺を選択候補としていけないことを確認したエージェント i は，57-59 行目に示されるように，$SE(i,j) \leftarrow Rejected$ として，別の辺を新たに探索するために **procedure** Looking を実行する．

辺に関する情報の登録を終えたエージェントは 73-80 行目に示される **procedure** Advise($ElectTable$) を実行する．もし，Search メッセージを用いて辺を探索させた全てのエージェントから辺の情報を受信し，自身も隣接辺の探索を終えているならば，$ElectTable$ の中からコストが最小となるエージェント k とそのコスト値 w をそれぞれ，$best\_edge$ と $best\_wt$ に保持する．そして，もし自身がリーダーでなけれ

ば，Advise(w) メッセージを用いてエージェント in_branch に，その辺の情報を送信する．Advise(w) メッセージを受信したエージェントは，63-64 行目に示されるように，そのメッセージを送信してきたエージェント id とコスト w を $ElectTable$ に登録し，同様に **procedure** Advise($ElectTable$) を実行することで，最もコストが小さく，次数制約を満たすことが可能な辺を選択する．辺の探索を要求した全てのエージェントから，Advise(w) メッセージを受信したリーダーのエージェントも同様にして，最もコストが小さく，次数制約を満たすことが可能な辺を選択し，最終的な辺の選択の決定を **procedure** LeaderReduction で行う．このように各エージェントによって探索された辺の情報は，リーダーに伝播されていく過程で，各エージェントによって吟味されていくため，最終的にリーダーのエージェントが辺を選択する際には，探索された全ての辺を保持し計算する必要はない．

procedure LeaderReduction では，95-102 行目に示されるように各エージェントがエージェント $best\_edge$ に LeaderReduction メッセージを送信することで，辺の選択を決定する旨を新しく部分木に追加されるエージェントまで伝播する．新しく部分木に追加されるエージェントと辺で繋がるエージェント i がエージェント j から LeaderReduction メッセージを受信した場合，$SE(i, best\_edge_i) \leftarrow Branch$ として，辺 $(i, best\_edge_i)$ の選択を決定する．そして，自身の次数 $deg\_count \leftarrow deg\_count+1$ として次数を増やし，新しく部分木に追加するエージェントに Joint メッセージを送信することで，辺の選択を完了する．Joint メッセージを受信したエージェントは，次のリーダーとなり，再び部分木内で，辺の探索と辺の情報の集約を行う．最終的には，部分木内の全てのエージェントにおいて，探索するべき辺がなくなった場合，処理を終了する．

6 実験・評価

6.1 で提案手法における探索空間の削減に関する予備評価を行う．6.3 では，提案手法の探索空間に関する評価を，6.4 では，厳密解法と近似解法に関する比較評価を行う．問題設定は以下の通りである．

- エージェント数 n: $5 \leq n \leq 30$ の範囲
- 辺の割合: ${}_nC_2 \times p$ (p は辺の割合を決定する値)
- 辺のコストの種類は以下の 2 種類である
  - (L) 一様分布に従う 10 から 100 の範囲の整数値
  - (H) 一様分布に従う 10 から 500 の範囲の整数値

本実験では，分散型の厳密解法が問題を解ける範囲で実験を行うために，極めて規模の小さい (辺密度の小さい) 問題を扱う．そのため，完全グラフに対する辺の

割合を少なく設定し，問題を作成した．また，グラフの辺に割り当てるコスト値の種類は，一様分布に従う 10 から 100 の範囲の整数値を用いる (L) と一様分布に従う 10 から 500 の範囲の整数値を用いる (H) の 2 種類とした．前者はコスト値の分散が低いもの，後者はコスト値の分散が高いものとして扱う．各エージェント数 n に対して，10 個の問題インスタンスを用意し，10 個の問題を解いた解に関する各評価指標の平均値を評価する．いずれの問題グラフも 3 章で述べたような近似解法の解を近似解に誘導可能なグラフである．また，全てのエージェント i に対して，$b_i = 3$ の次数制約を与える．評価対象は，以下の通りである．

dd-prim: d-prim を分散処理化した近似解法

d-nnt: NNT に次数制約を追加した近似解法

dd-mst: 分散型 d-MST の厳密解法

dd-mst-cl: 部分木の計算順序に基づく優先探索を行う近似解法

dd-mst-tc: 部分木の総コストを辺で割った値に基づく優先探索を行う近似解法

dd-mst-sdeg: 部分木の次数の四分位数範囲に基づく優先探索を行う近似解法

dd-mst-mdeg: 部分木の次数の最小値に基づく優先探索を行う近似解法

dd-mst-tcmdeg: dd-mst-tc，mdeg の指標に基づく優先探索を行う近似解法

dd-mst-half: dd-mst-tc，mdeg の指標に基づく優先を半分ずつ行う近似解法

6.1 異なる順序木による探索

dd-mst に関して，異なる順序木を用いて探索を行った場合の，サイクル数と組み合わせ数に関する評価である．順序木の種類は，エージェント数 $n = 6$ に対しては 1A，1B，1C を用意し，エージェント数 $n = 10$ に対しては 2A，2B，2C の計 6 種類を用意する．解いた問題と順序木の詳細を表 2 と表 3 に示す．表 2 と表 3 の順序木 1A と 2A はいずれも全順序木であり，分枝係数は 1 である．その他の順序木は分枝係数が，1A，2A よりは大きくなるような半順序木である．

サイクル数に関する結果を図 8 と図 10 に示し，組み合わせ数に関する結果を図 9 と図 11 に示す．図 8 と図 10 のグラフの横軸は順序木の種類であり，縦軸はサイクル数である．図 9 と図 11 のグラフの横軸は順序木の種類であり，縦軸は組み合わせ数である．組み合わせ数とは，順序木における各節エージェントの保持する組み合わせ数の平均である．

図 8 と図 10 からわかるように順序木の分枝係数が増加するほど，サイクル数が減少する．そのため，順序木の分枝係数が大きいほど，計算の並列性が増大すると考えられる．ただし，図 10 の 2B と 2C のように分枝係数が異なっていても，同

表 2: 順序木の種類 1

問題 エージェント数:6 辺数:9	順序木		
	1A (全順序)	1B (半順序)	1C (半順序)
分枝係数	1.00	1.67	2.50

表 3: 順序木の種類 2

問題 エージェント数:10 辺数: 16	順序木		
	2A (全順序)	2B (半順序)	2C (半順序)
分枝係数	1.00	1.29	2.25

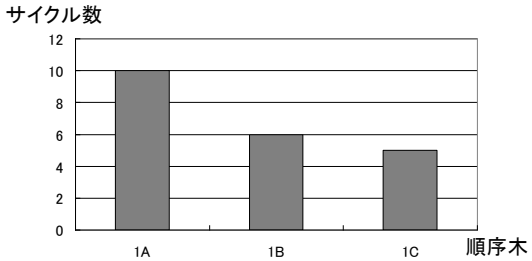

図 8: サイクル数

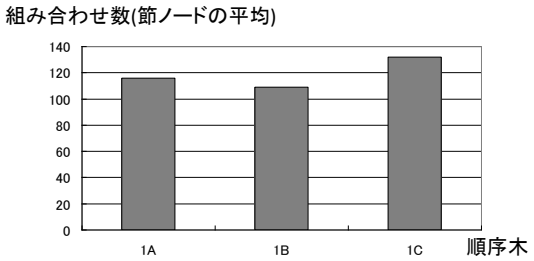

図 9: 組み合わせ数

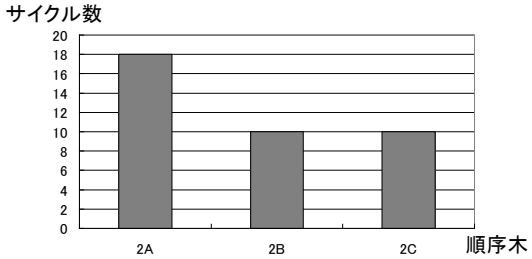

図 10: サイクル数

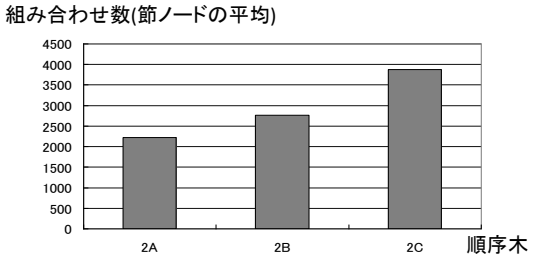

図 11: 組み合わせ数

じサイクル数になることがある．必ず全順序木よりも分枝係数が大きい半順序木を用いた場合，半順序木の方がサイクル数が必ず少なく済むが，分枝係数の異なる半順序木同士では分枝係数の大きい方が，必ずサイクル数が少ないとは限らない．サイクル数の大小関係が前後する可能性はあるが，分枝係数が増加するほど，サイクル数が減少する性質があると考えられる．

図 9 と図 11 からわかるように，サイクル数の傾向とは反対に，順序木の分枝係数が減少するほど，組み合わせ数が増大する．そのため，順序木の分枝係数が小さいほど，節のエージェントにおける部分木の演算に対する計算量が増大すると考えられる．しかしながら図 9 のように分枝係数の大きい 1A よりも分枝係数の小さい 1B の方が，組み合わせ数が大きくなることがある．これは問題の規模と，組み合わせの削除の度合いが関係するためであると考えるが，サイクル数の傾向と同様に，分枝係数が増加するほど，組み合わせ数は増加する性質があると考える．

なお，6.2，6.3，6.4 の実験において用いる順序木は，分枝係数が最も小さい全順序木や極端に大きい半順序木ではない順序木とする．従って，順序木に基づく計算順序や子エージェント数の極端な偏りが生じない．

6.2 異なる部分木の制限数による探索

dd-mst-cl, dd-mst-tc, dd-mst-sdeg, dd-mst-mdeg, dd-mst-tcmdeg, dd-mst-half の 6 手法に関して，異なる部分木の数の制限数 k を用いたときの解の探索空間の規模について実験を行った．エージェント数 $n = 30$ であり，問題のタイプは (H) と (L) の 2 つを用いる．パラメータ k には，$k = 500, 1,000, 2,500, 5,000, 10,000$ の 5 通りの制限数を用いる．得られた d-MST の総コストの和 $Q(T)$ と組み合わせ数と実行不可能解の平均値を評価した．組み合わせ数は，各エージェントが保持する組み合わせの平均値であり，実行不可能解はパラメータ k による部分木数の制限によって，実行不可能解に陥った回数である．なお，実行不可能解に陥った場合は，あらかじめ生成した順序木を解として，$Q(T)$ を算出する．問題のタイプが (L) の結果を表 4，表 5，表 6，表 7，表 8，表 9 に示し，問題のタイプが (H) の結果を表 10，表 11，表 12，表 13，表 14，表 15 に示す．

dd-mst-cl, dd-mst-tc, dd-mst-mdeg, dd-mst-sdeg, dd-mst-tcmdeg, dd-mst-half に関するいずれの表においても，組み合わせ数はパラメータ k の値が同数であれば，多少の差があるものの，ほとんど変わらないことがわかる．従って，優先探索のポリシーによって，組み合わせの削減度合いが大きく変化することはないと考える．ただし，優先探索の種類によっては，各エージェントの部分木の切り捨て方が，たまたま順応し組み合わせ数を多く削減できることも考えられる．

dd-mst-cl, dd-mst-tc, dd-mst-mdeg, dd-mst-sdeg, dd-mst-tcmdeg, dd-mst-half に関するいずれの表においても，パラメータ k の値を減少させると，組み合わせ数が減少することから，探索空間を削減できていることがわかる．その一方で，パラメータ k の値を減少させると，$Q(T)$ と実行不可能解の数は増加することから，解の質や精度が低下することがわかる．反対に，k の値を増加させると，組み合わせ数は増加するが，Q(T) と実行不可能解の数を減少させることが可能である．従って，各エージェントの記憶領域の容量や部分木に関する組み合わせの計算量やメッセージサイズを小さく抑えたい場合は，パラメータ k の値を小さくすれば良い．パラメータ k は出来る限り小さい方が望ましいが，解の質は簡単に悪化してしまう．反対に，解に高い質を求める場合，パラメータ k の値を大きくすれば良いが，k の値を大きくし過ぎると，各エージェントの記憶領域の容量や部分木に関する組み合わせの計算量やメッセージサイズも大きくなってしまうというトレードオフの関係がある．

また，問題のタイプによって $Q(T)$ の値に差があることがわかる．問題のグラフの辺のコスト値の分散が低い (L) では，$Q(T)$ は (H) の $Q(T)$ よりも小さくなり，辺のコスト値の分散が高い (H) では，$Q(T)$ は (L) における $Q(T)$ よりも大きくなる．表の結果では，(H) の方が問題が難しいため，全ての手法において (L) よりも実行不可能解の数が多くなっている．

表 4 と表 10 から，dd-mst-cl は，パラメータ k の値が大きくても小さくても実行不可能解の数が他手法に比べて平均的に多く発生することがわかる．これは dd-

表 4: dd-mst-cl

問題 n: 30 辺数: 40 タイプ: L	解法		
	dd-mst-cl		
k	組み合わせ数	Q(T)	実行不可能解
10,000	27,080	1,405	2
5,000	14,365	1,406	2
2,500	7,563	1,409	2
1,000	3,054	1,435	3
500	1,507	1,440	3

表 5: dd-mst-tc

問題 n: 30 辺数: 40 タイプ: L	解法		
	dd-mst-tc		
k	組み合わせ数	Q(T)	実行不可能解
10,000	27,264	1,290	0
5,000	14,014	1,293	0
2,500	7,249	1,331	1
1,000	3,067	1,378	2
500	1,588	1,443	4

表 6: dd-mst-sdeg

問題 n: 30 辺数: 40 タイプ: L	解法		
	dd-mst-sdeg		
k	組み合わせ数	Q(T)	実行不可能解
10,000	26,552	1,430	0
5,000	14,231	1,411	0
2,500	7,467	1,454	0
1,000	3,098	1,470	0
500	1,662	1,533	2

表 7: dd-mst-mdeg

問題 n: 30 辺数: 40 タイプ: L	解法		
	dd-mst-mdeg		
k	組み合わせ数	Q(T)	実行不可能解
10,000	27,639	1,421	0
5,000	13,929	1,467	1
2,500	7,776	1,430	0
1,000	3,159	1,470	0
500	1,618	1,481	0

表 8: dd-mst-tcmdeg

問題 n: 30 辺数: 40 タイプ: L	解法		
	dd-mst-tcmdeg		
k	組み合わせ数	Q(T)	実行不可能解
10,000	27,071	1,377	0
5,000	14,411	1,384	0
2,500	7,532	1,396	0
1,000	3,198	1,388	0
500	1,633	1,430	0

表 9: dd-mst-half

問題 n: 30 辺数: 40 タイプ: L	解法		
	dd-mst-half		
k	組み合わせ数	Q(T)	実行不可能解
10,000	27,186	1,414	0
5,000	14,460	1,428	0
2,500	7,565	1,434	0
1,000	3,198	1,430	0
500	1,660	1,451	0

表 10: dd-mst-cl

問題 n: 30 辺数: 41 タイプ: H	解法		
	dd-mst-cl		
k	組み合わせ数	Q(T)	実行不可能解
10,000	27,800	5,958	1
5,000	14,473	6,112	2
2,500	7,405	6,148	2
1,000	3,464	6,293	3
500	1,541	6,303	3

表 11: dd-mst-tc

問題 n: 30 辺数: 41 タイプ: H	解法		
	dd-mst-tc		
k	組み合わせ数	Q(T)	実行不可能解
10,000	27,853	5,845	1
5,000	14,312	5,861	1
2,500	7,360	6,050	2
1,000	3,054	6,408	4
500	1,619	6,482	4

表 12: dd-mst-sdeg

問題 n: 30 辺数: 41 タイプ: H	解法		
	dd-mst-sdeg		
k	組み合わせ数	Q(T)	実行不可能解
10,000	28,013	6,390	1
5,000	14,560	6,563	1
2,500	7,613	6,605	0
1,000	3,051	6,930	3
500	1,661	7,025	2

表 13: dd-mst-mdeg

問題 n: 30 辺数: 41 タイプ: H	解法		
	dd-mst-mdeg		
k	組み合わせ数	Q(T)	実行不可能解
10,000	27,773	6,445	0
5,000	14,930	6,561	0
2,500	7,753	6,582	0
1,000	3,148	6,858	1
500	1,620	6,995	3

表 14: dd-mst-tcmdeg

問題 n: 30 辺数: 41 タイプ: H	解法		
	dd-mst-tcmdeg		
k	組み合わせ数	Q(T)	実行不可能解
10,000	28,300	5,797	0
5,000	14,472	5,864	1
2,500	7,456	6,065	2
1,000	3,078	6,208	3
500	1,644	6,355	3

表 15: dd-mst-half

問題 n: 30 辺数: 41 タイプ: H	解法		
	dd-mst-half		
k	組み合わせ数	Q(T)	実行不可能解
10,000	28,332	6,382	0
5,000	14,834	6,429	1
2,500	7,625	6,469	1
1,000	3,153	6,596	1
500	1,646	6,757	1

mst-cl が単純に部分木を計算順序に従って切り捨てるためであると考える．それに対して，他の 5 手法は優先順位を用いて部分木の切り捨てを行うため，dd-mst-cl よりは平均的に実行不可能解の数が少ない．

表 5 と表 11 から，dd-mst-tc の誤差は他の 5 手法に比べて小さいことから，高い解の質を実現できていることがわかる．これは部分木の総コスト値に基づいて優先探索を行うためである．ただし，問題が (H) で k の値が小さい場合，dd-mst-tc の方が dd-mst-cl よりも実行不可能解の数が多くなり，$Q(T)$ が大きくなる．これは，実行不可能解に陥ったことにより，順序木を解として代用するためであり，問題のグラフのコスト値の分散が高い (H) の方が，実行不可能解に陥った際の解の質の悪化は顕著である．また，dd-mst-tc は総コスト値のみに基づく優先探索を用いるため，k の値が増加すると，実行不可能解の数も単調に増加する傾向がある．

表 6，表 7，表 12，表 13 から，次数に基づいて優先探索を行う dd-mst-sdeg と dd-mst-mdeg は，総コスト値のみに基づき優先探索を行う dd-mst-tc よりも実行不可能解の数を少なくできることがわかる．これは，部分木の各頂点の次数のばらつき度合いと部分木の各頂点の次数の余剰数の最小値が，ある部分木が，その後 d-MST として，どれくらい生き残りやすいかという指標になり得るためであると考える．なお，本来であれば k の値が大きい方が実行不可能解の数が少なくなると考えられるが，dd-mst-mdeg のように k の値が大きくても，実行不可能解の数が多くなる結果があった．これは本研究における近似手法が，あくまでも貪欲的な手法であり，k の値の増加に対して，完全に単調性を保証しているわけではないためである．これは，各エージェントの部分木の計算における計算順序の工夫によって，保証できると考えるが，本研究の本質ではないため，ここでの議論は省く．

また，表 8，表 9，表 14，表 15 から，同じ次数を考慮した優先探索でも，同時に総コストも考慮する優先探索である dd-mst-tcmdeg と dd-mst-half の方が $Q(T)$ を小さくできていることがわかる．なお，次数に基づく優先探索は dd-mst-sdeg と dd-mst-mdeg の 2 つを示したが，dd-mst-mdeg の方がより良い結果となったため，各部分木の次数の余剰数の最小値の中で大きいものを優先する手法と総コスト値を辺数で割った値が小さいものを優先する手法とを組み合わせた dd-mst-tcmdeg の結果のみを表に示した．特に，dd-mst-mdeg は，問題 (L) の場合，いずれのパラメータ k においても実行不可能解の数が 0 である．$k = 500$ のときに関しては，dd-mst-tc よりも小さい $Q(T)$ を得ることができている．dd-mst-half は，総コストと次数に基づく優先を半分ずつ行い，部分木の特徴をバランスよく考慮できるため，実行不可能解の数が少なくしつつ，$Q(T)$ を小さくできている．従って，部分木の分布に基づいて，部分木の削減を行う手法の有効性を示すことができたと考える．今回は半分ずつ考慮することで一定の効果を得たが，問題によっては部分木の分布に偏りがあることも考えられるため，それらを考慮すれば，より解の質を高めることが可能であると考える．

パラメータ k を増加させた場合，dd-mst-tcmdeg と dd-mst-half の 2 手法が他手法に比べて，より高い解の質を実現することができた．従って，総コストと次数

の両方に基づく優先探索が最も有効であると考える．6.3，6.4 の実験においては，次数に基づく優先探索として，dd-mst-mdeg と dd-mst-tcmdeg と dd-mst-half を用いることとする．また，6.3，6.4 におけるパラメータ k の値は，実行不可能解の数が 0 であり，かつ本研究の実験環境における実験時間とメモリ制限を考慮した値を設定する．dd-mst-cl，dd-mst-tc，dd-mst-mdeg，dd-mst-tcmdeg，dd-mst-half のいずれのパラメータ k も $30,000$ に設定するものとする．

6.3 探索空間の規模

dd-mst，dd-mst-cl，dd-mst-tc，dd-mst-mdeg，dd-mst-tcmdeg，dd-mst-half の 6 手法に関して，エージェント数 $n = 10, 15, 20$ に対する探索空間の規模の比較実験を行った．なお，6.2 で述べたように，いずれの手法においてもパラメータ $k = 30,000$ と設定する．組み合わせ数 (平均)，組み合わせ数 (最大)，削減総数 (平均)，削除数 (最大) を評価した．組み合わせ数 (平均) は各エージェントが保持した組み合わせの平均数，組み合わせ数 (最大) は各エージェントが保持する組み合わせの最大数であり，削減数 (平均) は各エージェントが一切制約を考慮しない場合の組み合わせ数から制約や部分木の制限数 k によって組み合わせを削減した平均数，削除数 (最大) は各エージェントが制約や k によって組み合わせを削減した最大数である．結果を表 16，表 17，表 18 に示す．なお，この実験においても，インスタンス数 10 個を解いた解に対して，各評価指標の平均値をとる．

表 16，表 17，表 18 のいずれの結果においても，dd-mst 以外の 5 手法は部分木の数の制限数 k によって組み合わせの削減が効き，dd-mst よりも組み合わせ数が少ないことがわかる．従って，純粋に d-MST に関する制約のみを考慮する dd-mst は，最も大きな記憶領域や計算する組み合わせの量や送受するメッセージサイズが必要となる．最大でも $30,000$ の組み合わせのみを保持する dd-mst-cl，dd-mst-tc，dd-mst-mdeg，dd-mst-tcmdeg，dd-mst-half は，dd-mst に比べて，それらの容量を大幅に削減することが可能である．その一方で 6.2 で述べたトレードオフの関係によって，解の質は，問題の規模の増大やパラメータ k の減少によって，低下してしまう可能性もある．その一方で，dd-mst-tc や dd-mst-mdeg のような部分木の総コストや次数に基づいた優先探索の厳密性と精度を高めたり，k の値を問題の規模に応じて設定することで，精度や実行可能性の低下は深刻ではなくなると考える．

また，エージェント数の増加に対する組み合わせ数の増加度合いも dd-mst の方が非常に大きいことから，パラメータ k を多少大きく設定したとしても，組み合わせの削減から得る計算量の減少の恩恵の方が大きいと考える．なお，d-nnt に関しては，各エージェントの探索空間の規模の増大は線形で，記憶領域は一定であるため，今回は評価を省いた．

表 16: 探索空間の比較 (エージェント数 10)

問題 n: 10 辺数: 18 タイプ: L	解法					
	dd-mst	dd-mst-cl	dd-mst-tc	dd-mst-mdeg	dd-mst-tcmdeg	dd-mst-half
組み合わせ数(平均)	6,133	2,633	2,825	2,823	2,793	2,958
組み合わせ数（最大）	60,168	25,166	27,090	27,222	26,762	28,415
削減数(平均)	6.0259 xE5	6.0608 xE5	6.0589 xE5	6.0589 xE5	6.0592 xE5	6.0575 xE5
削減数（最大）	6.0224 xE6	6.0574 xE6	6.0556 xE6	6.0554 xE6	6.0558 xE6	6.0542 xE6

表 17: 探索空間の比較（エージェント数 15）

問題 n: 15 辺数: 20 タイプ: L	解法					
	dd-mst	dd-mst-cl	dd-mst-tc	dd-mst-mdeg	dd-mst-tcmdeg	dd-mst-half
組み合わせ数(平均)	6,604	3,752	2,825	2,823	2,793	2,958
組み合わせ数（最大）	62,725	31,659	27,090	27,222	26,762	28,415
削減数(平均)	2.6915 xE7	2.6918 xE7	2.6918 xE7	2.6918 xE7	2.6918 xE7	2.6918 xE7
削減数（最大）	3.1754 xE8	3.1757 xE8	3.1757 xE8	3.1757 xE8	3.1757 xE8	3.1757 xE8

表 18: 探索空間の比較（エージェント数 20）

問題 n: 20 辺数: 26 タイプ: L	解法					
	dd-mst	dd-mst-cl	dd-mst-tc	dd-mst-mdeg	dd-mst-tcmdeg	dd-mst-half
組み合わせ数(平均)	15,688	5,400	5,459	5,410	5,423	5,564
組み合わせ数（最大）	172,010	45,124	46,086	45,668	45,820	46,517
削減数(平均)	4.0947 xE9	9.9279 xE9	9.9279 xE9	9.9279 xE9	9.9279 xE9	9.9279 xE9
削減数（最大）	6.4196 xE10	1.4193 xE11	1.4193 xE11	1.4193 xE11	1.4193 xE11	1.4193 xE11

6.4 解の質と構築サイクル数

dd-prim，d-nnt，dd-mst，dd-mst-cl，dd-mst-tc，dd-mst-mdeg，dd-mst-tcmdeg，dd-mst-half の 8 手法に関して，エージェント数 $n = 5$ から 30 まで 5 エージェントずつに対して，問題を解く比較実験を行った．問題は辺のコスト値の分散が高い (H) とコスト値の分散が低い (L) の 2 種類を用いて，誤差とサイクル数を評価した．誤差とは各手法によって求まる d-MST の辺の総コストの和と最適解の辺の総コストの和との差である．サイクルとはエージェントがメッセージを受信し，そのメッセージに基づく局所的な処理を実行し，その処理に基づくメッセージを他エージェントに送信する一連の動作の単位である．このサイクル数を分散協調探索のメッセージ送受による d-MST の構築コストを見積もるための評価指標として用いる．この評価においても 6.2 で述べたように dd-mst-cl，dd-mst-tc，dd-mst-mdeg，dd-mst-tcmdeg，dd-mst-half の上限数 k は実行不可能解の数が 0 となるように $k = 30,000$ と設定する．サイクル数に関する結果を表 19 に，解の質に関する結果を表 20 に示す．なお，表 19 の dd-mst に示される「-」は，本研究の実験環境において，現実的な計算時間による解の算出が不可能であったことを表す．

表 19 から，厳密解法である dd-mst の誤差は常に 0 となるため，最も解の質が高いことがわかる．しかしながら，探索空間の複雑度は各エージェントの順序木の深さと隣接辺数に対して指数関数的に増加するため，このような小規模な問題インスタンスを用いても現実的な時間で解が得られない場合が存在した．それに対して，6.2 でも述べたようにパラメータ k によって組み合わせ数の削減を行う dd-mst-cl，dd-mst-tc，dd-mst-mdeg，dd-mst-tcmdeg，dd-mst-half は，エージェント数 n が増加しても解を得ることができる．この 5 手法の誤差は，いずれのエージェント数 n においても，d-nnt より小さくなっている．また，エージェント数が小さいときに関しては，dd-prim より誤差は小さくなっている．その一方で，エージェント数が増加すると，dd-mst-cl と dd-mst-mdeg と dd-mst-half の誤差は，dd-prim のそれよりも大きな値をとってしまう．従って，考慮すべき組み合わせの上限数 k を設定し，それを上回る組み合わせを切り捨てるというような基本的なヒューリスティクスを用いる場合，優先探索のポリシーによっては，問題の規模の増大によって解の質が簡単に劣化する可能性が高まると考える．

dd-mst-tc と dd-mst-tcmdeg はいずれのエージェント数 n においても，平均的に小さい誤差であり，dd-prim よりも小さい誤差を実現できた．従って，探索空間の規模を小さく抑えつつ，ある程度高い解の質を得る手法の適用の有効性を示すことができた．dd-mst-tcmdeg と同様に部分木の総コストと次数の両方に基づく優先探索を行う dd-mst-half は，dd-mst-cl よりは小さい誤差であるため，優先順位の付け方の割合を，より柔軟に選択することで，解の質を高めることができると考える．また，d-nnt の誤差はいずれのエージェント数のときも，他手法に比べて大きくなったため，極端な探索空間の削減はあまり有効ではないと考える．

なお，本実験ではパラメータ $k = 30,000$ とすることで，dd-prim よりも高い精

表 19: 解の質

問題			解法							
			dd-prim	d-nnt	dd-mst	dd-mst-cl	dd-mst-tc	dd-mst-mdeg	dd-mst-tcmdeg	dd-mst-half
n	辺数	タイプ	誤差							
5	7	L	22	131	0	0	0	0	0	0
	8	H	80	355	0	0	0	0	0	0
10	18	L	14	148	0	5	0	1	0	1
	18	H	47	739	0	16	0	2	0	41
15	20	L	11	146	0	7	0	4	0	1
	21	H	36	820	0	107	0	19	13	28
20	25	L	9	134	0	13	4	21	5	10
	27	H	33	1,033	0	167	4	151	13	256
25	34	L	17	232	-	72	1	69	2	10
	33	H	94	991	-	353	5	220	6	265
30	42	L	11	329	-	129	4	113	5	98
	41	H	116	1,637	-	653	60	522	67	636

表 20: サイクル数

問題			解法							
			dd-prim	d-nnt	dd-mst	dd-mst-cl	dd-mst-tc	dd-mst-mdeg	dd-mst-tcmdeg	dd-mst-half
n	辺数	タイプ	誤差							
5	7	L	27	19	5	5	5	5	5	5
	8	H	31	17	5	5	5	5	5	5
10	18	L	105	28	6	6	6	6	6	6
	18	H	103	33	7	7	7	7	7	7
15	20	L	199	43	10	10	10	10	10	10
	21	H	195	42	11	11	11	11	11	11
20	25	L	325	50	13	13	13	13	13	13
	27	H	328	51	12	12	12	12	12	12
25	34	L	473	56	-	14	14	14	14	14
	33	H	467	60	-	15	15	15	15	15
30	42	L	626	67	-	15	15	15	15	15
	41	H	644	58	-	14	14	14	14	14

度を得たが，パラメータ k の値を大きくすればする分，厳密解法の精度に近付けることが可能であるため，解く問題の規模と各エージェントの処理能力に見合った k の値の設定が非常に重要となると考える．部分木の計算量と解の精度のトレードオフを意識した k の値の選択を柔軟に行うことができれば，計算量を増大させ過ぎず，かつ dd-prim よりも解の質を高めることが可能である．特に，dd-mst-tc や dd-mst-tcmdeg のように，解の質と実行可能性の両方を同時に考慮するような優先探索の併用が有効であるため，これらの優先探索のポリシーを問題の規模や得るべき解の精度の目標に応じて的確に適用し，より厳密に実装を行えば，問題の規模の増大に対しても十分に対応することが可能であると考える．

表 20 から，dd-mst，dd-mst-cl，dd-mst-tc，dd-mst-mdeg，dd-mst-tcmdeg，dd-mst-half のサイクル数は他の 2 手法と比べて常に少なくなっていることがわかる．これは dd-mst に関する 6 手法は順序木に沿って 1 往復のメッセージ伝播のみを行うため，メッセージ回数が少なく済むためである．従って，実際の通信ネットワークにおいて，この手法を適用する場合，各エージェントのメッセージの伝播による通信遅延は他手法に比べて，少く済むと考えられる．その一方で，なるべく小さい k の値を用いて組み合わせ数を抑制しなければ，各エージェントの計算すべき部分木集合のサイズが膨大となるため，個々のエージェントの計算時間や 1 つのメッセージに対する通信時間が大きくなることは不利な点であると考える．

また，dd-prim のサイクル数が最も多くなっている．これは dd-prim が辺の探索と情報の集約を反復的に繰り返すためである．従って，実際の通信網において，メッセージ回数が多くなり，メッセージの伝播による通信遅延が大きくなってしまうと考えられる．d-nnt に関しては，各エージェントは，最悪でも自身の隣接エージェント全てを，ひと通り探索すればよいため，dd-prim よりはサイクル数が少なくなっている．ただし，選択するべき辺を探索する際にランク値が自身よりも高く，次数制約を満たすエージェントをすぐに発見できない場合は，隣接エージェントとメッセージ交換を繰り返すため，サイクル数が増加すると考えられる．

7　おわりに

本研究では，分散協調問題解決の枠組みで，次数制約付き最小生成木問題を形式化した．それに対して分散協調探索手法を提案し，評価した．

比較的小規模な例題を用いた評価実験では，各エージェントの解探索において，ある程度の部分木を切り捨てて組み合わせを削減する近似解法を用いて，探索空間を削減しつつ一定の解の質を得ることができた．特に，部分木の辺の総コストと，次数の余剰数の最小値の両方に基づく優先探索を用いる手法の有効性が確認された．優先探索のポリシーを洗練し，各エージェントにおける部分木を，より効果的に削減することで，必要なメモリやメッセージサイズを抑制することが十分に可能であると考える．また，この優先探索に基づくボットアップな解法を基

礎として，部分木の辺の総コスト等をトップダウンに集計を伴う分枝限定法を併用するなどの手法の適用も考えられる．

謝辞

本研究を進めるにあたって，本当に御多忙の中，時には楽観的に時には厳しく研究方針を御指導してくださった松尾啓志教授に感謝致します．また，過密度日程が多く非常に御多忙の中において，常日頃から研究の進捗を熱心に気に掛けていただき，的確な助言と激励をくださった松井俊浩准教授に深く感謝致します．研究に関する助言をくださるとともに研究以外の相談にものってくださった津邑公暁准教授に感謝致します，齋藤彰一准教授にはミーティングの進捗発表の際に，研究に関するアドバイスをいただきました．有難うございます．また，常日頃から数々の助言や協力をして頂いた松尾・津邑研究室，齋藤研究室の皆様に深く感謝致します．特に，週に何度か研究室に泊まる中で，多くの励ましをくださり，精神的な支えになってくださった新部屋のメンバーである Thangaraja Vijitha 氏，池谷友基氏，浅井宏樹氏，祖父江宏裕氏，安井裕亮氏，柳田大輝氏，吉田健二氏，水野航氏，澤田晃平には感謝致します．ことに川東勇輝氏，熊崎宏樹氏は，研究に関して私と共に悩んでくださり，様々なアイデアをくださいました．有難うございます．最後になりますが，日頃から他愛もない会話で研究に対するやる気を向上させてくださった近藤勝彦氏，今井満寿巳氏，稲葉崇文氏，白井宏憲氏，秋山晋氏，加藤雄大氏，酒井衛氏に感謝致します．

本研究に関する発表

伊藤 翼, 松井 俊浩, 松尾啓志: ”次数制約付き最小生成木を構築する分散協調探索手法の検討 ”, 合同エージェントワークショップ&シンポジウム (JAWS2011) ロング発表論文 (Oct. 2011)

参考文献

[1] Baruch Awerbuch. A New Distributed Depth-First-Search Algorithm. *Information Processing Letters*, Vol. 20, No. 3, pp. 147–150, 1985.

[2] Bruce Boldon, Narsingh Deo, and Nishit Kumar. Minimum-Weight Degree-Constrained Spanning Tree Problem: Heuristics and Implementation on an SIMD Parallel Machine. *Parallel Computing*, Vol. 22, No. 3, pp. 369–382, 1996.

[3] Thang Nguyen Bui and Catherine M. Zrncic. An ant-based algorithm for finding degree-constrained minimum spanning tree. In *Genetic and Evolutionary Computation Conference*, pp. 11–18, 2006.

[4] Robert G. Gallager, Pierre A. Humblet, and Philip M. Spira. A Distributed Algorithm for Minimum-Weight Spanning Trees. *ACM Transactions on Programming Languages and Systems*, Vol. 5, No. 1, pp. 66–77, jan 1983.

[5] Michael R. Garey and David S. Johnson. Computers and Intractability: A Guide To the Theory of NP-Completeness. *a*, 1979.

[6] Lisa Higham and Zhiying Liang. Self-stabilizing minimum spanning tree construction on message-passing networks. In *DISC*, pp. 194–208, 2001.

[7] M. Hosseini, D. T. Ahmed, S. Shirmohammadi, and N. D. Georganas. A Survey of Application-Layer Multicast Protocols. *Communication Surveys & Tutorials, IEEE*, Vol. 9, No. 3, pp. 58–74, 2007.

[8] Maleq Khan, Gopal Pandurangan, and V.S. Anil Kumar. Distributed Algorithms for Constructing Approximate Minimum Spanning Trees in Wireless Networks. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 20, No. 1, pp. 124–139, jan 2009.

[9] Joshua D. Knowles and David Corne. A new evolutionary approach to the degree-constrained minimum spanning tree problem. *IEEE Transactions on Evolutionary Computation*, Vol. 125-134, No. 2, 2000.

[10] Joseph B. Kruskal. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. In *Proceedings of The American Mathmatical Society*, Vol. 7, pp. 48–48, feb 1956.

[11] Akshat Kumar, Boi Faltings, and Adrian Petcu. Distributed constraint optimization with structured resource constraints. In *Autonomous Agents & Multiagent Systems/Agent Theories, Architectures, and Languages*, pp. 923–930, 2009.

[12] S. A. M. Makki and George Havas. Distributed Algorithms for Depth-First Search. *Information Processing Letters*, Vol. 60, No. 1, pp. 7–12, oct 1996.

[13] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. Adopt: asynchronous distributed constraint optimization with quality guarantees. *Artif. Intell.*, Vol. 161, No. 1-2, pp. 149–180, 2005.

[14] Subhash C. Narula and Cesar A. Ho. Degree Constrained Minimum Spanning Tree. *Computers & Operations Research*, Vol. 7, No. 4, pp. 239–249, 1980.

[15] Christos H. Papadimitriou and Umesh V. Vazirani. On Two Geometric Problems Related to the Traveling Salesman Problem. *Jornal of Algorithms*, Vol. 5, No. 2, pp. 231–246, 1984.

[16] A. Petcu and B. Faltings. A Scalable Method for Multiagent Constraint Optimization. *9th Int. Joint Conf. on Artificial Intelligence*, pp. 266–271, 2005.

[17] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, Vol. 36, No. 1, pp. 1389–1401, dec 1957.

[18] Sherlia Y. Shi and Jonathan S. Turner. Multicast routing and bandwidth dimensioning in overlay networks. *Slected Areas in Communications, IEEE Journal on*, Vol. 20, No. 8, pp. 1444–1455, oct 2002.

[19] Knut-Helge Vik, Pal Halvorsen, and Carsten Griwodz. Multicast tree diameter for dynamic distributed interactive applications. In *IEEE International Conference on Computer Communications IEEE INFOCOM*, No. 3, pp. 1597–1605, 2008.

[20] Wamiliana. Solving the degree constrained minimum spanning tree problem using tabu and modified penalty search methods. *Journal Teknik industri*, Vol. 6, No. 1, pp. 1–9, 2004.

[21] Gengui. Zhou and Mitsuo Gen. A note on genetic algorithms for degree-constrained spanning tree problems. *Networks*, Vol. 30, No. 2, pp. 91–95, 1997.