TM-1043

タスク割り当て問題向け並列協調方式

進藤　静一、炭田　昌人（三菱）

May, 1991

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191 ~5
Telex ICOT J32964

Institute for New Generation Computer Technology

タスク割り当て問題向け並列協調方式

三菱電機株式会社 中央研究所
進藤静一　　炭田昌人

1991年3月25日

要旨

並列推論マシンのアプリケーションとしてタスク割り当て問題向け並列協調方式を開発した。この方式は、適当にタスクを初期割り当てされたリソース間でお互いにタスク割り当てを替え合いながら、段階的により良いタスク割り当てをみつけてゆく。探索を制御する知識が協調方式に組み込まれているのでそのような知識が獲得困難であるような問題向けとなっている。協調方式は同期式と非同期式の2方式からなる。これらの方式を評価する為、配送計画問題をドメインとしたシステムをマルチPSIで実現し、実験を行なった。本報告は両方式のアルゴリズムレベルでの説明、及び、実験を通じての両方式の評価を含む。使用したPEの台数と処理時間との関係についての議論も載せる。

1　概要

ハードウェア技術の進歩に伴い、我々の使用できる計算機環境は、分散化、並列化の傾向を強めている。この計算機環境の変化に対応するべく、人工知能の分野でも分散人工知能の研究が盛んに行なわれるようになってきた[Bond 88]。

人工知能がひとりの人間の知的活動を計算機で実現することを目的とするとすれば、分散人工知能は複数の人間からなる集合による知的活動を実現の対象、または、メタファの拠り所とすると捉えることができる。一言に人間の集合といっても、構成メンバー数、発言力の強さ、行動目標の多様性（個人中心か全体中心か）、メンバー間の共有情報量、メンバー間の情報伝達速度、要求される応答時間等、その属性により、異なったあり方を示す。例えば、軍隊では、明確な階層構造、上官への絶対服従という強力な制御機構を背景に、その軍に与えられた指令を達成することを集合の活動動機とする。そこでは個人の目標達成は許されない。一方、株式市場では、各メンバーは同格であり、各メンバーによる自己利益の増大という極めて個人的な目標達成意欲が集合の活動動機となっている。このような人間の集合の機能の多様性と同様、分散人工知能の所産である協調方式やシステムも様々な特徴や仮定を持ち、この多様性を統一的に扱える枠組は得られていない。

このような状況の中で、注目を浴びているのが、自律的協調[Hewitt 86][伊藤 90]の概念である。即ち、ある程度の処理能力を持つ計算主体（以下、エージェントと呼ぶ）が複数個集まり、全体を統制する制御部からの命令に服従するのではなく、お互いに情報交換をし、その情

報に基づき各々が自律的に行動しながらも、全体の目標を達成したり、全体の制約を満たすように振舞う方式である。この方式の特徴は、システムの拡張 / 縮退容易性、状況の変化への適応性、問題記述の高いモジュラリティ、にあり、また、並列処理の導入が容易であることが期待される。

我々は自律協調方式の研究の対象としてタスク割り当て問題を選んだ。なぜなら、ジョブショップスケジューリングや負荷分散等、タスク割り当て問題は基本的な工学的問題のひとつであり、更には、この種の問題は組合せ問題であることからその探索空間が膨大となり並列処理をするのに値するだけの計算量が必要となると考えたからである。

我々が対象とする問題を以下に示す。以下の条件が与えられたとする。

リソースの集合：$R = \{r_1, r_2, \ldots, r_m\}$
タスクの集合：$T = \{t_1, t_2, \ldots, t_n\}$
リソース r_i の担当タスクの集合：T_i （但し、$1 \leq i \leq m$、$\bigcup T_i = T$、$\forall i, \forall j (i \neq j)$ に対して $T_i \cap T_j = \phi$）
リソース r_i の評価関数：$f_i : T_i \vdash$ 実数（但し、$1 \leq i \leq m$）

この時、$\sum_{k=1}^{m} f_k(T_k)$ の値をなるべく小さくする $T_i (1 \leq i \leq m)$ を求めることである。この問題では、組合せ爆発を起こし、その回避の為には探索をうまく制御するヒューリスティクスが必要である。しかし、問題が複雑な場合、そのような知識を獲得、表現することが困難であることもある。我々は探索知識を埋め込むことにより、このような状況に適用できるような方式を考案することとした。

タスク割り当て問題に対する有効な自律協調方式としてコントラクトネット [Davis 83] がある。コントラクトネットは、自分が実行できないタスクを他のどのエージェントに実行して貰うかを決定する問題に対して、タスクの公示、入札、落札を基本とし、タスクを誰に依頼するかという依頼者側の選択と誰のタスクを実行してやるかという被依頼者側の選択の両者の相互選択権を実現することにより、制御と情報が分散した環境でのタスク割り当て方式を与える。また、並列動作単位を1エージェントの動作とみなすことにより、並列処理の導入に対して見通しが良い。

我々が対象とする問題でのリソースをエージェントに対応付け、タスクをエージェントに適当に初期割り当てしておきエージェント間でその割り当てを替え合うことにより段階的により最適な解を求めてゆくというアプローチをとることにより、コントラクトネットの枠組を有効に利用できると考えられる。具体的には、各エージェントの持つ情報を自分の担当タスクと評価関数に限定し、エージェント間で以下の動作を1ステップとし、このステップを繰り返す方式である。

1. **タスク公示**：エージェントに評価関数値の基準を設け、基準値以上のエージェント（この状態を「過負荷である」と呼ぶ）が自分の担当タスクの内の1個を替わりに行なってくれるエージェントを募集する。

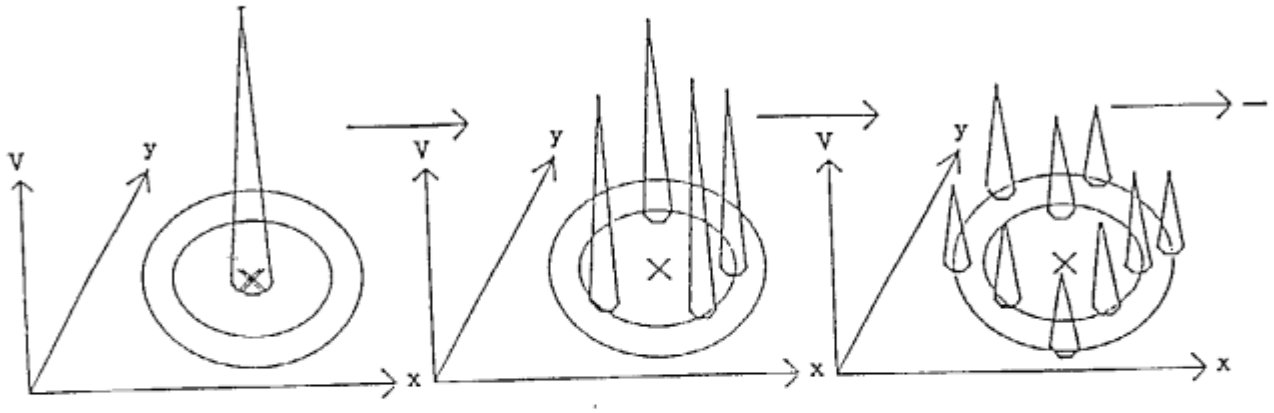
2. **入札**：公示を受けたエージェントは公示されたタスクを自分が行なった場合の評価関数値を求めることにより公示されたタスクの評価を行ない、それにより、適宜、公示に応募する。

3. **落札**：入札してきたエージェントの中から、入札状況に応じて依頼するエージェントを選択する。

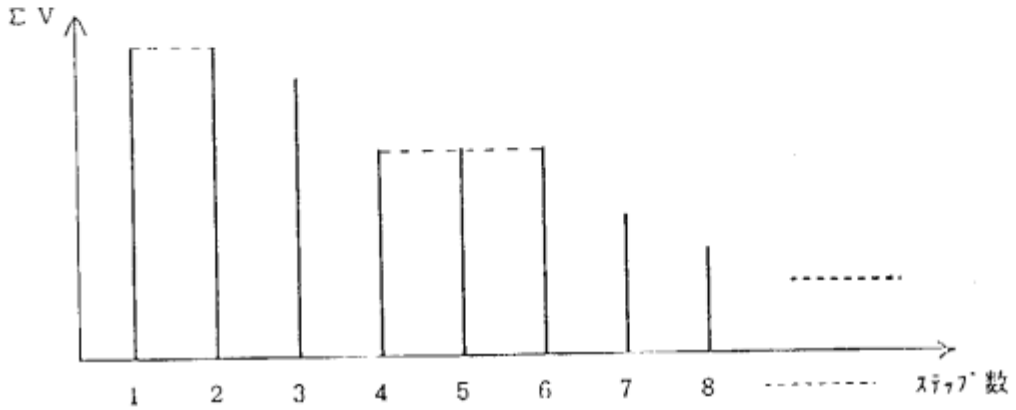
一般に過負荷となるエージェントは複数個存在する。そのような場合、どのエージェントからステップを開始するかの指定はコントラクトネットではされていない。そこで、我々はこの複数過負荷エージェントによる並列ステップ実行の方式を開発することにより、並列協調方式を明確化することとし、エージェントのステップ実行間で同期を取る方式と、取らない方式を開発した。前者を同期式協調方式、後者を非同期式協調方式と呼ぶ。我々の方式はコントラクトネットの並列処理版と位置付けることができる。これらの方式を評価する為に、配送計画問題をドメインとしたシステムをマルチ PSI で実現し、実験を行なった。

本報告は両方式のアルゴリズムレベルでの説明、及び、実験を通じての両方式の評価を含む。使用した PE の台数と処理時間との関係についての議論も載せる。以下、第2章で協調方式の説明を、第3章でシステム構成を、第4章で例題として用いた配送計画問題の説明と協調方式との対応を、第5章で実験結果と評価を、述べる。

なお、本報告内容は、ICOT の委託業務の研究成果に基づくものである。



（a）エージェントの変化



（b）全エージェントでの評価関数値の和の変化

図 1: 協調方式の動作の概略

2　並列協調方式

2.1　概略

我々が開発した協調方式の動作を直感的に説明する。前章で述べたように、この協調方式は、全体を統制する制御部の介在無く、エージェントがお互いに情報をやりとりすることによって、予め割り当てられたタスクの担当者を替えあってゆく。その基本となるのは、タスクの公示、入札、落札である。その結果、評価関数値のエージェント全体での総和が徐々に改良されてゆく。

この状況を図 1に示す。同図 (a) は各エージェントの評価関数値の変化のイメージである。ステップが進むにつれ左から右に変化する。 xy 平面にエージェントが分布しており、 v 軸はエージェントの評価関数値の大きさを表す。最初は 1 個のエージェントのみが過負荷である（当然最初から複数個のエージェントが過負荷であってもよい）。ステップが進むにつれて、一般に、過負荷であるエージェントの数が増えるが各エージェント間の評価関数値の差は縮

まってゆく。

同図 (b) は全エージェントでの評価関数値の和の変化である。縦軸が和を、横軸はステップを表す。この図が示すように、和は単調減少[1]してゆく。但し、その先が最小値に到達するとは限らない。

協調方式には、同期式と非同期式がある。次節以降でその各々を説明する。

2.2 同期式協調方式

同期式協調方式では、過負荷エージェントのステップ間で同期をとることによってタスク割り当てが決定してゆく。即ち、一斉に過負荷エージェントがタスク公示を行ない、その返答結果を全過負荷エージェントが共有することにより、公示に出されたタスクの割り当てを全エージェントが公認した後に、次のタスク公示を全ての過負荷エージェントが一斉に行なう。同期式協調方式に於けるエージェント間のコミュニケーションとエージェントの状態の概略を図2に示す。

同期式協調方式の長所は誰のタスクを誰がやればどの程度全体の評価関数値が変化するかの情報を過負荷エージェント全員が共有することにより1ステップ内での最適割り当て、即ち、評価関数値の全エージェントでの総和が最も減少する割り当てが得られることであり、短所は同期の為の無駄な待ち時間（例えば、タスク割り当てに関係のないエージェントが次のステップに進めない）が発生することである。

同期式協調方式では、同じステップを繰り返し行なう。以下、使用されるメッセージの種類、1ステップ内のアルゴリズム、及び、ステップの繰り返しの終了条件を説明する。

2.2.1 メッセージの種類

- 公示メッセージ
  - 意味：公示するタスクをメッセージの受け手が請け負うことによる評価関数値の変化量の見積依頼を表す。
  - 送信内容：メッセージの送り手のid（この項目は以下のいかなるメッセージに於いても含まれるので以下の説明では省略する）、ステップ番号、タスクのidと仕様、及び、このメッセージの送り手がそのタスクを実行しないことによる評価関数値の変化量。
- 不公示メッセージ
  - 意味：現ステップではこのメッセージの送り手がタスク公示を行なわないことを表す。

[1]値が増加しないことが保証されているという意味である。即ち、直前のステップでの値と同じであるか、小さいかの何れかである。

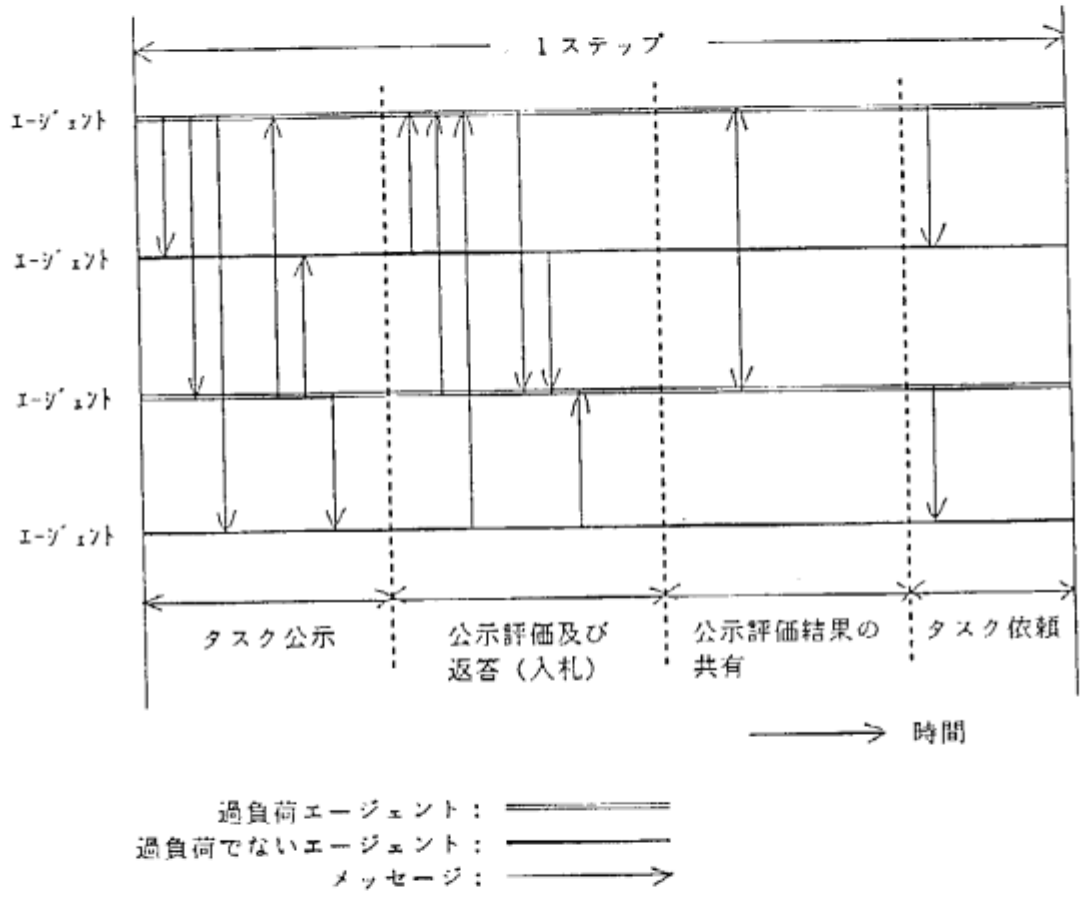


図 2: 同期式協調方式に於けるエージェント間コミュニケーションとエージェントの状態

- 送信内容：ステップ番号、及び、不公示メッセージであることが分かる atom。

- **入札メッセージ**
  - 意味：公示の返事。公示されたタスクの実行依頼に応じることを表す。
  - 送信内容：ステップ番号、対応する公示番号、及び、公示されたタスクを請け負うことによる評価関数値の変化量。

- **共有メッセージ**
  - 意味：誰から出された公示タスクを誰が実行すれば両者の評価関数値の和がどれくらい変化するかを表す。
  - 送信内容：ステップ番号、タスクを実行するエージェントの id[2]、及び、両者の評価関数値の和の変化量。

- **落札メッセージ**
  - 意味：入札結果の通知。公示に出したタスクを誰に依頼するかを表す。
  - 送信内容：ステップ番号、対応する公示番号、及び、落札先のエージェントの id。但し、誰にも落札しない場合はそのことが分かる値が落札先のエージェント id に代入される。

2.2.2 1ステップ内のアルゴリズム

同期式協調方式の1ステップ内での処理の流れを図3に示す。以下、説明を与える。

(i) **終了条件の判定【図中 (1)】**

2.2.3「終了条件」の節にて説明する。

(ii) **態度表明**

- **過負荷であるエージェントの場合【図中 (3)(4)(5)】**：各エージェント毎に公示に出して良いタスクの集合が定義されている（詳細は 2.2.3「終了条件」の節で説明される）。その集合の中から、そのタスクを担当しないことにより一番自分の評価関数値が減少する1個のタスクを選択し、そのタスクとその評価関数値変化量をブロードキャストする（公示メッセージ）。公示に出してよいタスクが無い場合、又は、どのタスクを公示に出しても評価関数値が減少しない場合は、このステップでは自分が公示を出さないことをブロードキャストする（非公示メッセージ）。
- **過負荷でないエージェントの場合【図中 (14)】**：このステップでは、自分は公示を出さないことをブロードキャストする（非公示メッセージ）。

[2]「誰から出された公示タスクを誰が実行すれば」の内の「誰が」に相当する。ちなみに、「誰から」はメッセージの送り手であり、公示メッセージの説明にある通りこのメッセージにも含まれる。

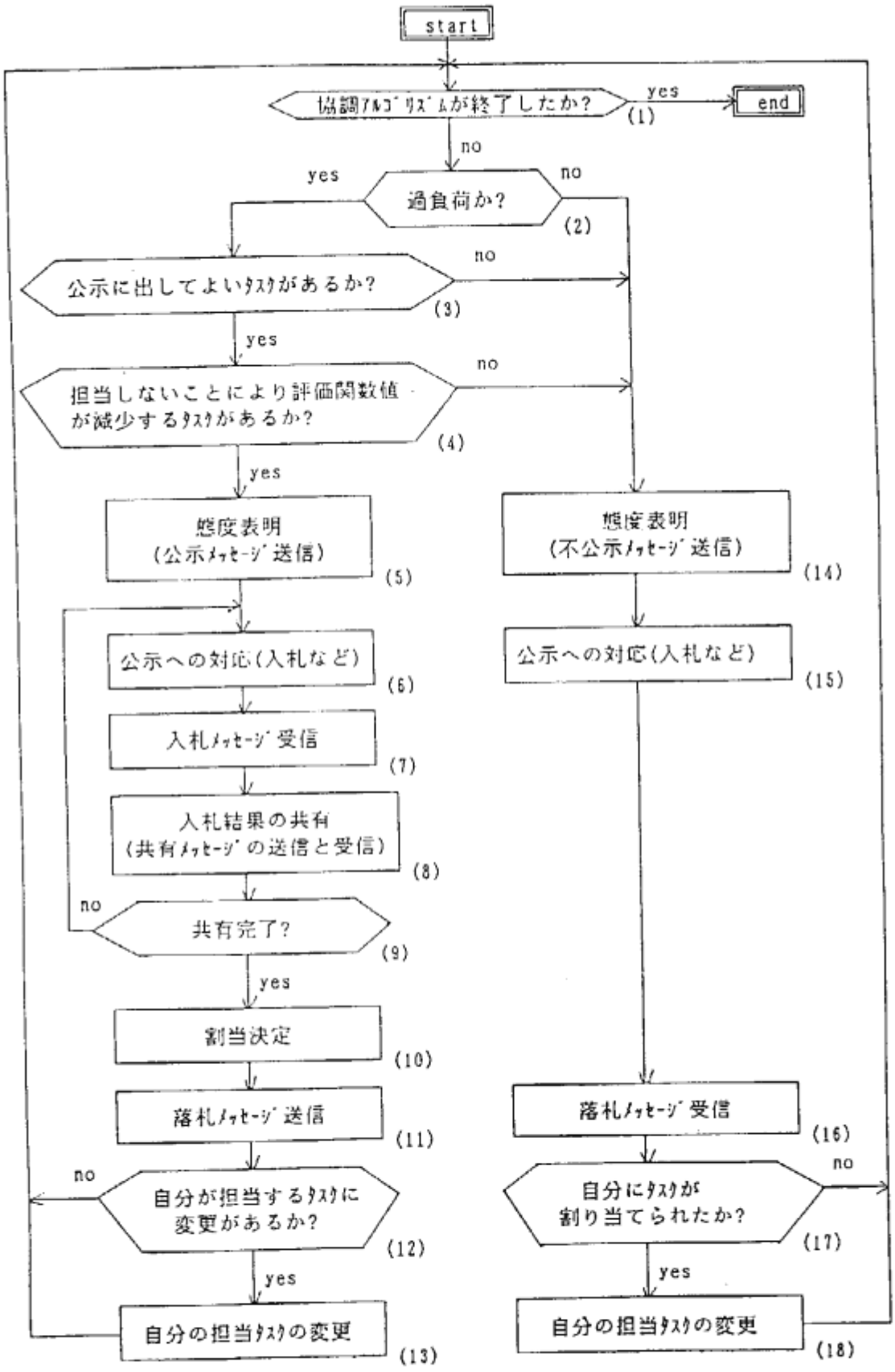


図 3: 同期式協調方式 1 ステップ内の処理の流れ

(iii) **公示への対応 [図中 (6)(15)]**

全エージェントは態度表明のメッセージを受信する。その各々に対して以下の反応をする。

- **それが公示メッセージである場合**：公示されたタスクを引き受けることによる両者の評価関数値の変化量を公示を出したエージェントに返答する。
- **それが不公示メッセージである場合**：そのメッセージの送信主が公示を出さないことを記録する。メッセージに対する返答はなし。

これにより、全エージェントは他の全エージェントの態度を知ることができる。公示を出した（マネージャと呼ぶ）か、公示を出さないか（コントラクタと呼ぶ）の何れかである。マネージャであっても公示メッセージに対して返答することに注意。

(iv) **情報共有 [図中 (7)(8)(9)]**

1ステップ内での最適、かつ、衝突の無いタスク割当を行う為に、「誰から出された公示タスクを誰が実行すれば両者の評価関数値の和がどれくらい変化するか」という情報をマネージャ間で共有する。この変化量をef値と呼ぶ。その為に、マネージャは公示の返答（他のどのエージェントが自分の公示タスクを実行した場合、ef値はどれだけか）を全過負荷エージェントに通知する（共有メッセージ）。各マネージャは他の全マネージャに関するef値をマトリクスの形で蓄積する。このマトリクスをefマトリクスと呼ぶ。マネージャはどのエージェントがマネージャでどのエージェントがコントラクタであるかが前段階（公示への対応）で分かっているので、efマトリクスのどの項目が揃えば情報共有が完了するかを独自に判断できる。

(v) **タスク依頼 [図中 (10)(11)]**

各マネージャはefマトリクスの内容が全て揃った時点で、その情報より、最適、且つ、衝突の無い割当を決定し、自分は誰にタスクを割り当てるのかをブロードキャストする。マネージャの中には公示タスクを他のエージェントに依頼したくともefマトリクスを評価した結果それを諦めざるを得ないものがある。そのようなマネージャでも、自分がタスク依頼を行なわないことをブロードキャストする。これらは落札メッセージにて、通知される。

(vi) **タスク割り当て変更認識**

- **マネージャの場合 [図中 (12)(13)]**：自分がこのステップでタスク割り当ての変更に関与しているか否かを自分で判断できる。関与していれば、それに従って自分の担当タスクを変更し、関与していなければ変更しない。その後、次のステップに移行する。
- **コントラクタの場合 [図中 (16)(17)(18)]**：自分がこのステップでタスク割り当ての変更に関与しているか否かが自分で判断できない。そこで、
  1. どのエージェントが過負荷であるかを (iii) で認識した後、
  2. マネージャからのタスク依頼通知を (v) で全て受信した後、
  3. 自分がタスク割り当ての変更に関与していれば、それに従って自分の担当タスクを変更し、関与していなければ変更しない。
  4. その後、次のステップに移行する。

2.2.3 終了条件

全ての過負荷エージェントは、公示に出してよいタスクの集合を持つ。その集合が、全ての過負荷エージェントに関して空集合となった時点で、このアルゴリズムは終了する。この基準は一言で表せば、全くタスク割り当てに変更の無いステップが連続して続けばこれ以上協調手続きを続行しても全体の評価関数値の和の減少に貢献しないから中止するというものである。

アルゴリズムの終了判定を行なう為には、各過負荷エージェントは自分の公示に出してよいタスクの集合が空になったか否かを常に把握しなければならない。その為、自分の担当タスク毎にフラグをつける。フラグの値は0、又は、1である。0はそのタスクを公示に出してよいことを、1はいけないことを表す。公示に出してよいタスクの集合とはフラグの値が0であるタスクの集合である。

全タスクのフラグの初期値は0である。ステップの終りで、全エージェント間で全くタスクの割り当て替えが発生しなければ、各マネージャはそのステップで公示に出したタスクのフラグを1に替える。何らかのタスク割り当て変更が発生すれば、自分がそのタスク割り当てに関係していようといまいと、そのステップで公示に出したタスクをも含めて、自分の担当タスクのフラグを全て0にリセットする。ステップでタスクの割り当て変更が発生したか否かは、2.2.2の(v)で説明した通り、タスク依頼が成立するしないに関わらず、落札結果を全エージェントで共有しているので各エージェントが独立に判定できる。

2.3 非同期式協調方式

非同期式協調方式では、過負荷エージェントのステップ間で同期をとらないことによってタスク割り当てが決定してゆく。

過負荷エージェントは1タスクを公示に出す。出すタイミングはエージェントによってバラバラでもよい。公示を受けたエージェントはなんらかの基準で応じるべき公示を1個選択し（ここでは、早いもの順、且つ、効果のあるものとしている）、公示タスクを実行することによる評価関数値の変化量を返すことにより、入札する。過負荷エージェントは公示に応じない。その後、落札通知（当選／落選）が来るまで、ロックを掛ける。即ち、それまでに到着する公示に対して入札拒否をする。落札通知がきた後、ロックを解除して次の公示を評価する。公示を出した側はある者からは入札を、あるものからは入札拒否を受ける。その内、入札してきた者の中から、自分と相手の過負荷量の和が一番減少する相手を選択しタスクをその者に落札する。非同期式協調方式に於けるエージェント間のコミュニケーションとエージェントの状態の概略を図4に示す。

非同期式協調方式の長所は、同期式協調方式と比較すれば、同期の為の無駄な時間が発生せず、それだけ、時間単位当たりにより多くの公示を評価できることであり、短所は同期式協調方式と比較すれば、タスク割り当てに用いる情報がより局所的であるのでステップ内での最適割り当てが得られないことと、得られる解が、非決定的に変化することである。

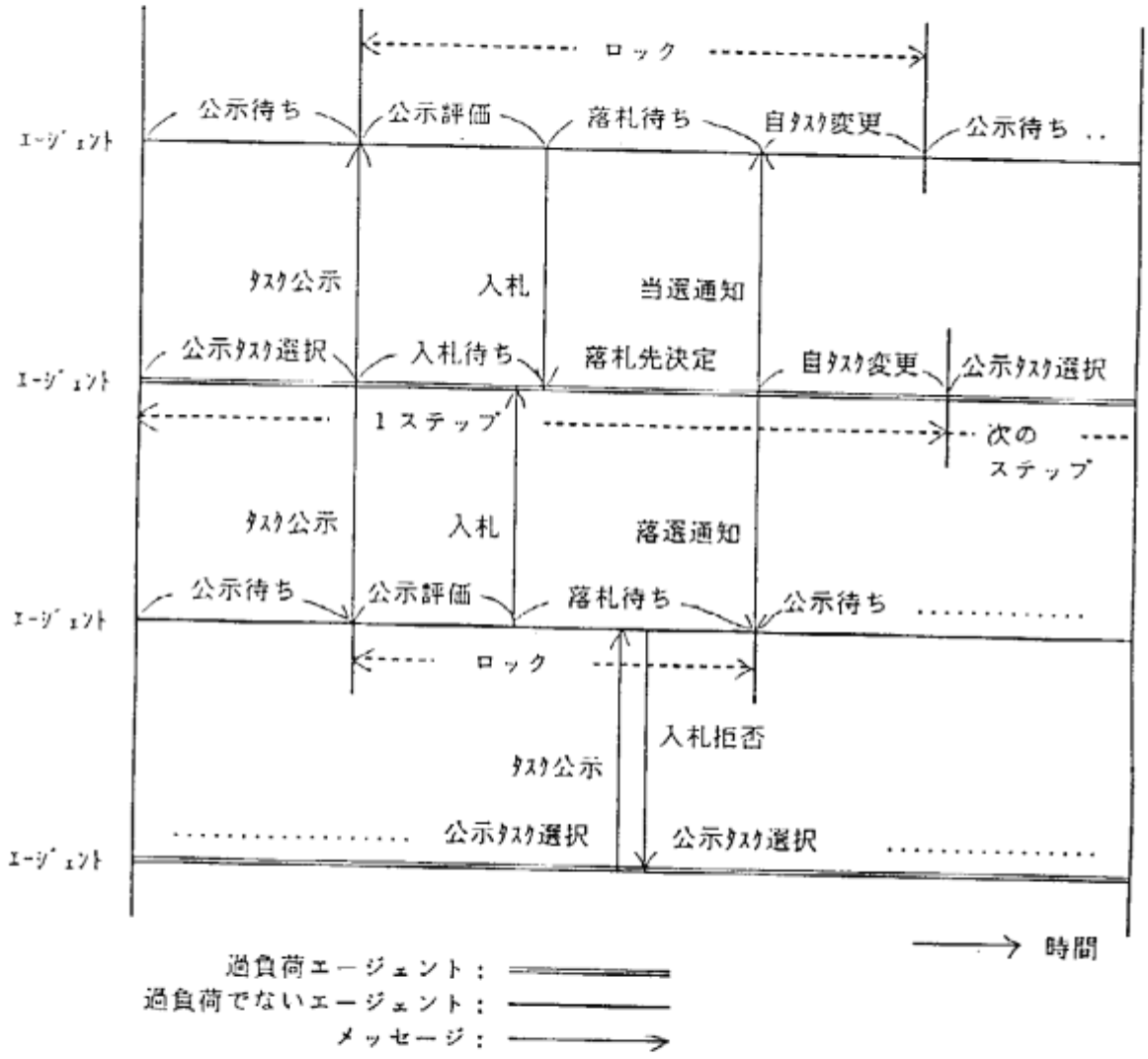


図 4: 非同期式協調方式に於けるエージェント間のコミュニケーションとエージェントの状態

非同期式協調方式も、同じステップを繰り返す。このステップはエージェントが過負荷であるかないかにより異なり、先程述べたようにエージェント間で同期のとれたものではない。同期式協調方式と同様に、メッセージの種類や使用するデータ、1ステップのアルゴリズム、及び、ステップの繰り返しの終了条件を説明する。

2.3.1 メッセージの種類、及び、主なデータ

(1) メッセージの種類

- 公示メッセージ
  - 意味：公示するタスクをメッセージの受け手が請け負うことによる評価関数値の変化量の見積依頼を表す。
  - 送信内容：メッセージの送り手のid（この項目は以下のいかなるメッセージに於いても含まれるので以下の説明では省略する）、公示番号、タスクのidと仕様、及び、メッセージの送り手がそのタスクを実行しないことによる評価関数値の変化量。
- 入札メッセージ

  意味：公示の返事。公示されたタスクの実行依頼に応じることを表す。
  - 送信内容：対応する公示番号、及び、公示されたタスクを請け負うことによる評価関数値の変化量。
- 入札拒否メッセージ
  - 意味：公示の返事。公示されたタスクを評価したが評価関数値の減少に貢献できないので、そのタスクを請け負わないことを表す。
  - 送信内容：対応する公示番号、及び、入札拒否メッセージであることが分かるatom。
- 多忙メッセージ (busy0)
  - 意味：公示の返事。他のエージェントに入札しておりその落札結果が返ってきていないので、この公示の評価を見送ることを表す。
  - 送信内容：対応する公示番号、及び、多忙メッセージ (busy0) であることが分かるatom。
- 多忙メッセージ (busy1)

  意味：公示の返事。自分が過負荷であるので、この公示の評価を見送ることを表す。
  - 送信内容：対応する公示番号、及び、多忙メッセージ (busy1) であることが分かるatom。

- 落札メッセージ
  - 意味：入札結果の通知。公示に出したタスクを誰に依頼するかを表す。
  - 送信内容：対応する公示番号、及び、落札先のエージェントの id。
- 状態変化メッセージ
  - 意味：自分の担当タスクに変化があったことを表す。

    送信内容：変化が起こるまでに、このメッセージの送信先に入札拒否メッセージを返したタスク id の集合。

(2) 使用される主なデータ

- noBidHistory：過負荷でないエージェントのみが保有する。今までに、どのエージェントのどのタスクに対して入札拒否メッセージを返したかを保有する。
- commHistory：過負荷エージェントのみが保有する。自分が出してきた公示に対してどのエージェントがどのような反応（入札、入札拒否、多忙）を返したかを保持する。詳細は終了条件の項を参照のこと。

2.3.2 1 ステップ内のアルゴリズム

非同期式協調方式の 1 ステップ内での処理の流れを図 5 に示す。以下、説明を与える。図中、左半分が過負荷でないエージェントの振舞いを、右半分が過負荷エージェントの振舞いを表す。以下、図中の各詳細処理を説明する。説明に於いては、以下の各項目の頭の数字が図中のボックスの番号と対応する。

(i) 過負荷でないエージェントの振舞い

- 公示入手［図中 (1)］：その時点で到着している最初の公示メッセージを取り出す。
- 公示評価［図中 (2)］：公示メッセージに含まれているタスクを引き受けることによる自分の評価関数値の変化量を求める。
- 入札するか？［図中 (3)］：(2) で求めた値がある値（パラメータとして指定される）以下である場合、公示に対して入札すると判断する。
- 入札すると判断した場合［図中 (4)］：入札メッセージを公示発信元に送る。
- 入札しないと判断した場合［図中 (5)］：入札拒否メッセージを公示発信元に送る。更に、「どのエージェントのどのタスクに対して入札しなかったか」を noBidHistory に記録する。

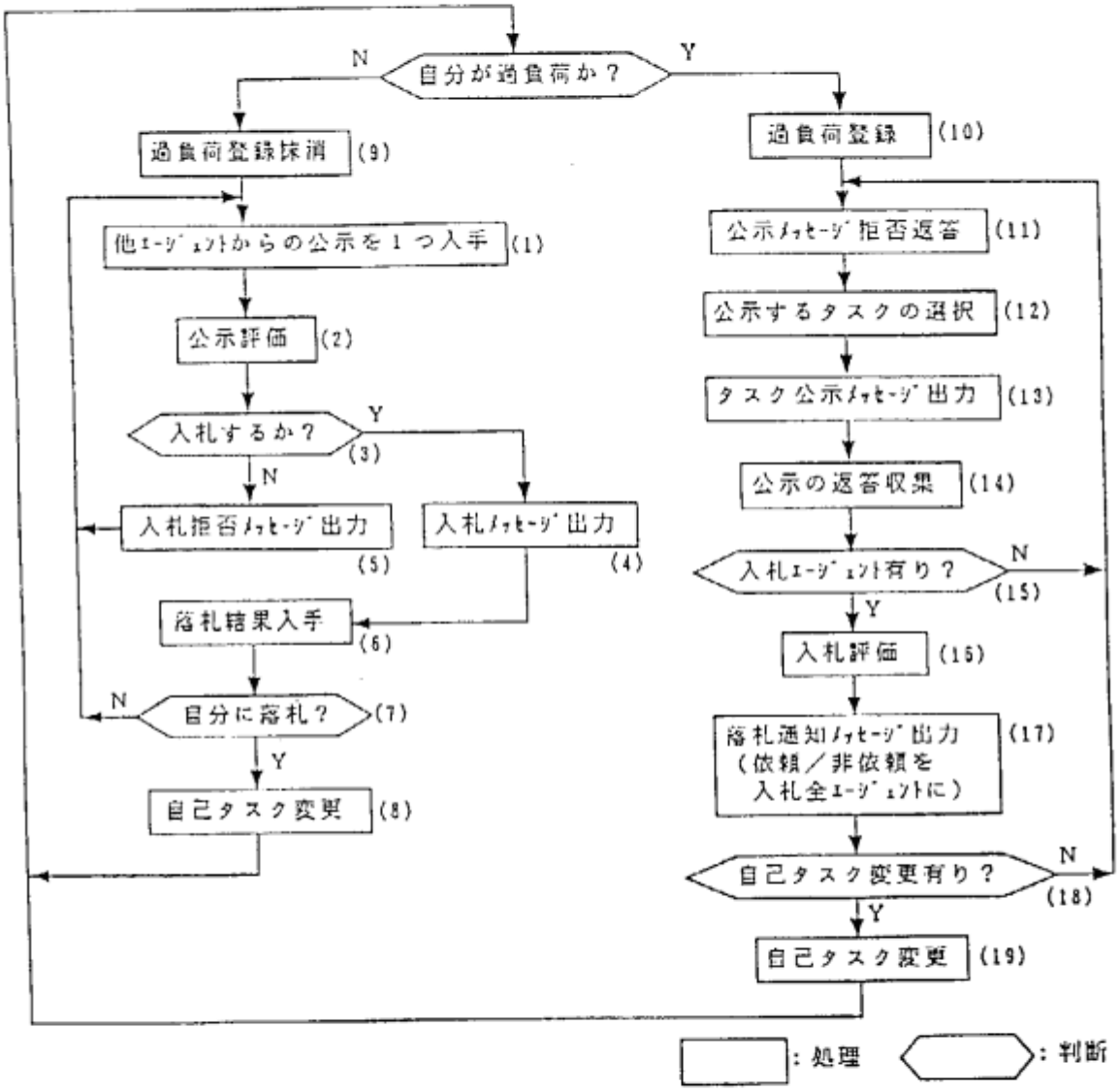


図 5: 非同期式協調方式 1 ステップ内の処理の流れ

- **落札結果入手［図中 (6)］**：自分が入札したエージェントからの落札メッセージが到着するのを待つ。それまでに来た公示メッセージに対しては、多忙メッセージ（busy0）を返すことにより公示評価を見送る旨を返答する。それまでに来た状態変化メッセージに対しては、単に読みとばす。

- **自己タスク変更［図中 (8)］**：自分が落札したのであれば、それに従って自分のタスクを変更する。タスク変更後も過負荷でない場合は、noBidHistory に記録されている、かつて入札拒否メッセージを返したエージェントに、入札拒否したタスクを状態変化メッセージとして送る。これにより、今度はそのタスクに入札できるかも知れない旨を告げる。なぜならば、タスクを引き受けることにより自分の状態が変化し、その結果、かつては仮に請け負ったとしても評価関数値が減少しないタスクでも、今度は評価関数値が減少するかもしれないからである。その後 noBidHistory を空にリセットする。

- **過負荷登録抹消［図中 (9)］**：(19) で過負荷状態から過負荷でない状態に変化した場合にのみ、自分は新たに過負荷でなくなったことを告げる。

(ii) 過負荷エージェントの振舞い

- **過負荷登録［図中 (10)］**：(8) で過負荷でない状態から過負荷状態に変化した場合にのみ、自分は新たに過負荷になったことを告げる。

- **公示メッセージ拒否返答［図中 (11)］**：その時点で到着している公示メッセージに対しては、多忙メッセージ（busy1）を返すことにより公示評価を見送る旨を返答する。その時点で到着している状態変化メッセージに対しては、その変化を commHistory に反映する。タスクが無い場合は読みとばす。

- **公示するタスクの選択［図中 (12)］**：commHistory を参照することにより、まだ一度も公示を出していないタスク、又は、かつて出したが相手に評価されていないタスクの中から、自分がそのタスクを担当しないことによって自分の評価関数値が減少するタスクを選択する。

  初めて公示に出す前に、自分の担当タスクの各々に対して、そのタスクを担当しないことによる自分の過負荷量の変化量を求める。それをソートする。初めての公示では、その内、一番過負荷量が減少するタスクを公示に出す。2 回目以降では、前ステップで自分のタスクに変化があれば、評価関数値の変化量を求めなおし、それをソートし直して、一番評価関数値が減少するタスクを公示候補とする。前ステップで自分のタスクに変化がなければ次に評価関数値が減少するタスクを公示候補とする。次に、公示候補の commHistory を参照する。そのタスクがかつて公示に出されており、且つ、そのタスクに対する反応が未評価 (busy0) であるエージェントが無い場合はそのタスクは公示に出さない。この場合、次に過負荷量が減少するタスクをサイクリックに選択する。

  過負荷量が減少するタスクの全てが公示に出せなければ、その旨を告げ、その後、このエージェントは公示を受ける一方となる。公示に出すタスクがあれば、そのタスクを初めて公示に出す場合は公示をブロードキャストし、かつて公示に出した場合は未公示 (busy0) であるエージェントのみに公示を送る。

- **タスク公示メッセージ出力［図中 (13)］**：(12) で選択されたタスクがまだ一度も公示に出していない場合はブロードキャストを、かつて公示に出した場合は commHistory を参照することにより前回までで評価してもらっていない相手に、公示を行なう。
- **公示の返答収集［図中 (14)］**：自分が出した公示に対する返答をすべて収集する。収集し終るまでに来た公示メッセージに対しては、多忙メッセージ（busy1）を返すことにより公示評価を見送る旨を返答する。収集し終るまでに来た状態変化メッセージに対しては、その変化を commHistory に反映する。タスクが無い場合は読みとばす。
- **入札評価［図中 (16)］**：公示の返答が全部揃った時点で、入札メッセージを返してきたエージェントの中から、自分とその相手の評価関数値の和の減少量が一番大きいものに入札する。入札エージェントが無い場合もある。
- **落札通知メッセージ出力［図中 (17)］**：(14) で入札メッセージを送ってきた全てのエージェントに、落札メッセージを送る。

2.3.3 終了条件

公示に出したタスクに対して、少なくとも 1 回は他の過負荷でないエージェントによって入札されたものの、自分と相手の評価関数値の和の減少がみられず、結局誰にもそのタスクが落札されなければ、そのタスクは他のエージェントに再び公示に出しても無駄である。公示に出す候補の全タスクに関して、そのようなことがいえれば、そのエージェントが公示を出しても過負荷量の減少には貢献しない。過負荷である全エージェントがそのような状態（公示に出すタスクがなくなった状態）になったときにこのアルゴリズムは終了する。終了を判定するには、(1) 全過負荷エージェントが公示に出すタスクがなくなったことを知る、(2) 1 過負荷エージェントが公示に出すタスクがなくなったことを知る、必要がある。以下、その各々について説明する。

(1) 全過負荷エージェントが公示に出すタスクがなくなったことを知る方法

どのエージェントが過負荷であるかを把握するところをエージェント以外に設定する。これを管理部と呼ぶ。管理部が把握している過負荷エージェントがすべて公示に出すタスクがなくなれば、管理部がエージェントの動作を終了させる。そのため、エージェントは、管理部に、

- 初めて過負荷になった時にその旨を通知する。
- 過負荷から過負荷でなくなった時にその旨を通知する。
- 公示に出すタスクがなくなった過負荷エージェントはその旨を通知する。

これらは図 5 の処理 (10)、(9)、(12) で各々行なわれる。

(2) 1過負荷エージェントが公示に出すタスクがなくなったことを知る方法

担当タスクの各々に対して、そのタスクに対する他のエージェントの今までの反応を保持する。反応とは、かつて出したそのタスクの公示に対して他の各エージェントがどの様な返答を返したかを意味する。commHistoryに保持される。以下のように分類される。

- **未公示**：まだ1回もそのタスクを公示に出していない。
- **入札済み**：そのエージェントから入札メッセージが返ってきた。
- **入札拒否済み**：そのエージェントから入札拒否メッセージが返ってきた。
- 未評価：以下の2種類に分類される。
  - busy0：過負荷でないエージェントから多忙中（busy0）メッセージが返ってきた。
  - busy1：過負荷エージェントから多忙中（busy1）メッセージが返ってきた。

かつて入札拒否を返したエージェントが、その後担当タスクに変化があり、且つ、過負荷でない場合は、今度は同じタスクでも入札できるかも知れない。そのためには、公示を出した側がそのことを認識し、再び公示をそのエージェントに出し直す必要がある。そこで、そのようなエージェントは、変化が起こるまでに入札拒否メッセージを返したエージェントid、タスクidをnoBidHistoryで保持し、それらに対してかつて自分が出した入札拒否メッセージを打ち消すことを意味する状態変化メッセージを出す（図5の処理(8)にて行なわれる）。状態変化メッセージを受けた過負荷エージェントは対象となるエージェント、タスクの反応を「入札拒否済み」から「busy0」に書き換える。そのタスクはそのエージェントにとって公示可能なものとなる。

過負荷エージェントにとって、公示に出してよいタスクとは、それを実行しないことにより自分の評価関数値が減少し、且つ、そのタスクの他エージェントの反応が「未公示」、又は、「busy0」であるエージェントが少なくとも1個存在するタスクである。逆に、自分の各担当タスクに関して、他の全エージェントからの反応が、「入札済み」か、「入札拒否済み」か、「busy1」であった場合、自分は公示に出すタスクが無いと判断する。この場合、その旨を管理部に告げる。

3　システム構成

ここでは、協調方式をアルゴリズムレベルで説明する際に必要となる、このシステムの構成を説明する。システム構成を図6に挙げる。

以下、各部・各モジュールの概要について説明する。

(1) 管理部

管理部はプログラム実行の初期化、ファイル・ウィンドウへの入出力管理、交渉の開始と終了の管理、プログラムの終了処理などを行う。同期式と非同期式の2つの実行モジュールが管理部に含まれる。それぞれ同期式と非同期式の協調方式を実行する場合に、上記のような管理を行うためのモジュールである。

- **同期式実行モジュール**：同期式協調方式の実行管理を司る。協調方式での交渉とは直接関係のない、インタフェース部を通じたファイルへの入出力の管理や、ウィンドウ入出力の管理を行い、必要なデータをエージェント部に渡す。さらに、交渉開始合図を各エージェント部に送るとともに、あるエージェント部からの終了メッセージを全体に伝達する。終了判定自身はエージェント部が行う。
- **非同期式実行モジュール**：非同期式協調方式の実行管理を司る。同期式実行モジュールと同様なファイル・ウィンドウへの入出力管理を行う。交渉開始の処理は同期式と同じであるが、終了判定は本モジュールが行う。

(2) エージェント部

エージェント部は協調方式の実際の処理を実現する部分である。処理の制御は同期式と非同期式の協調方式で異なるので、それぞれ同期式制御モジュールと非同期式制御モジュールで行う。通信モジュールは両制御モジュールで共通に使用する、メッセージ送受信のためのモジュールである。また計画評価モジュールはドメインに固有な処理を記述するものであり、配送計画固有の処理を記述する。

- **同期式制御モジュール**：同期式協調方式の実行時に、方式に基づて処理を制御するためのモジュールである。実際には外部から送信されてきたメッセージを処理し、必要なメッセージを生成し送信する処理を通して、同期式協調方式の制御を行う。外部とのメッセージの送受信は、本モジュールが直接行うのではなく、通信モジュールを通して間接的に実行する。
- **同期式制御モジュール**：非同期式協調方式の実行時に、方式に基づて処理を制御するためのモジュールである。同期式制御モジュールと同様、メッセージの送受信処理を通して、非同期式協調方式の制御を行う。外部とのメッセージの送受信は通信モジュールを通して間接的に実行する。

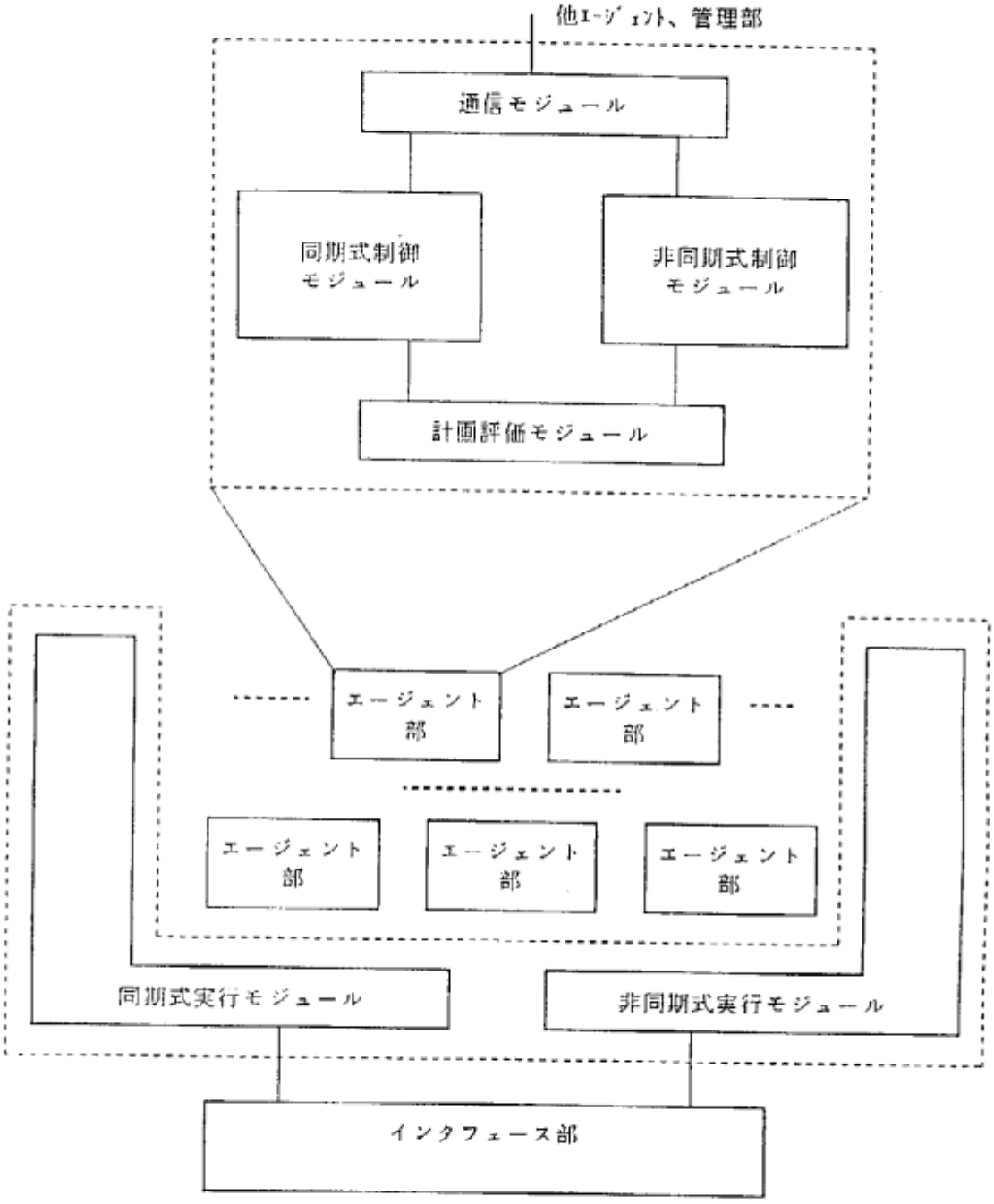

他ｴｰｼﾞｪﾝﾄ、管理部
通信モジュール
同期式制御
モジュール
非同期式制御
モジュール
計画評価モジュール
エージェント
部
エージェント
部
エージェント
部
エージェント
部
エージェント
部
同期式実行モジュール
非同期式実行モジュール
インタフェース部

図 6: システム構成

- **通信モジュール**：協調方式の実行に必要なメッセージを複数のエージェント部の間で送受信するための種々の機能を実現する。実行モジュール・制御モジュールを完全結合し、相互にメッセージの送受信を可能にする。
- **計画評価モジュール**：ドメインに固有な処理を実行するモジュールであり、協調方式に必要な評価関数値の算出や計画の変更などを行う。ここで対象とするドメインは配送計画問題である。

(3) インタフェース部

インタフェース部ではＦＥＰへのウィンドウ入出力およびファイル入出力を実行する。ＫＬ１との間は、ストリング入出力デバイス機構により接続され、ＫＬ１からの要求により動作する。

4　配送計画問題への適用

前節まで説明した協調アルゴリズムを評価する為に例題として取り上げた配送計画問題について説明する。

4.1　問題設定

(1) 配送計画問題

ここで取り上げた配送計画問題は、1つの配送拠点から納期指定された複数個の荷物を納期遅れがないように指定された複数個のサイトに複数台のトラックで配送することを目的とする。以下の設定を置く。

- 無線などを使って、任意の2台のトラック間で交信が可能である。
- 各トラックは、全サイトの位置、全トラックのid、自分の担当配送オーダー、を知っている。配送オーダーは配送荷物、配送サイト先、納期から成る。
- 配送荷物の識別はその種類のみである。共通の単位が設定されており、1オーダーの荷物は全て1単位量である。
- 1サイトからは1つのオーダーしか出されない。
- トラックは幾つでも荷物を運ぶことができる。

このような設定の基で、複数台のトラックが自分に割り当てられた配送オーダーを実行するべく移動している。この時点では各トラックには納期送れは発生していない。ここで1台のトラックに故障が発生し、そのトラックの荷物が永久に運べなくなったとする。このまま放置すればこれらの荷物の納期送れ時間は無限大となる。そこで、他のトラックがこれらの荷物を代理で配送することにより、荷物の納期遅れ時間を他のトラック全体でなるべく小さくするとする。

解くべき問題は、どの荷物をどのトラックに割り当てればよいかを決定することである。これに対して、以下に示す2段階からなる方法を用いる。第1段階では、故障トラックが中心となって、自分の配送オーダーを正常トラックに分散する。その結果、幾つかの正常トラックで納期遅れが発生する。第2段階では、全トラック間での納期送れ時間の和をより小さくする為、正常トラック同士で配送オーダーの割り当て替えを行なう。我々が実現した部分はこの内の第2段階に当たる部分である。

(2) 協調アルゴリズムとの対応

協調アルゴリズムと上で述べた配送計画問題との対応は以下の通りである。

- **タスク**：配送オーダーの実行。
- **エージェント**：トラック。
- **評価関数**：トラックが担当しているオーダーを実行することによる納期送れ時間。
- **エージェントが過負荷である**：そのエージェントに対応するトラックが担当しているオーダーの納期送れ期間の和が正である。

4.2 タスクの評価方法

我々の提案する協調方式は、公示タスクの評価、その値に基づく入札、落札が中心となる。配送計画問題では、公示タスクの評価には、依頼方式と、交換方式の2つの方式が取られる。依頼方式では、公示に出された配送オーダーの荷物そのものを替わりに運ぶことにより、オーダーの割り当てを変更する。その為には、どこかでその荷物を受け渡さなければならない。交換方式では、2台のトラックが同種類の荷物を持っている場合にのみ成立し、それらの荷物の配送先を変えることによりオーダーの割り当てを変更する。2台のトラックが荷物の授受を行なう必要はない。

依頼方式では、公示タスクの評価値として、

「公示送り手が公示タスクを担当しないことによる納期送れ時間の変化量」＋
「公示受け手が公示タスクを担当することによる納期送れ時間の変化量」

を求める。しかし、この値は、荷物の公示送り手がどこにいつ置くかに依存する。そこで、公示に荷物の置く場所とその時刻を含ませ、公示評価側がその中で一番上で定義した評価値が小さくなる場所、時刻を選択し、公示返答に含ませる。荷物を置く場所は公示送り手の配送先のサイトに、置く時刻はそのサイトへの到着予定時刻とする。

交換方式では、公示の送り手、受け手両者がタスクを交換するので、受け手のみでは両者の納期送れ時間の変化量を求めることはできない。そこで、以下の通信プロトコルを用いる。

(1) トラックSがタスクt(S)を公示に出す。

(2) トラックRが公示を受ける。Rはt(S)に関わる荷物と同じ種類の荷物を持っているかをチェックする。無い場合は、依頼方式のみでそのタスクを評価する。有る場合（一般に複数個ある）は、その荷物に関わるタスクの各々（t(R)とする）に対し、

t(S)を実行しt(R)を実行しないことによる自分の納期送れ時間の変化量

を求める。公示に入札する時は、t(R)、及び、この値を返す。

(3) Sが入札を受ける。交換の可能性がある場合は、Sは、各t(R)に関して、

t(R)を実行しt(S)を実行しないことによる自分の納期送れ時間の変化量

を求め、その値と、入札に含まれる、

t(S) を実行し t(R) を実行しないことによる自分の納期送れ時間の変化量との和が最小となるタスク t(R) を交換タスクとする。

5 実験評価

5.1 実験内容

協調方式の評価を行なう為に、配送計画問題を例題とした配送計画システムをマルチPSI上に実現し、各種実験を行なった。ここではその内容を説明する。

(1) 初期データ

協調アルゴリズムは、同期式、非同期式を問わず、エージェントに初期タスクを割り当て、エージェント間で協調することによりそれを変更してゆく。従って、アルゴリズムの実行に先だって各エージェントにタスクを初期割り当てしなければならない。我々の実験では、平成元年度に得られた2段階からなる段階的過負荷浸透方式の第1段階で得られたタスク割り当て結果をこれに充てる。具体的には以下のようなものである。

ある1台のトラックが故障し、自分の配送オーダーを他のトラックに替わりに実行して貰うことを想定する。以下のアルゴリズムに従ってタスクを分配する。

(i) **タスク公示**：故障トラックが自分の担当配送オーダーをすべて公示に出す。

(ii) **公示評価**：正常トラックが公示された配送オーダーの各々を独立に実行した場合の納期送れ時間を故障トラックに返す。

(iii) **割り当て通知**：故障トラックが、正常トラックの返答結果を基に、自分の配送オーダーを実行することによる正常トラックの納期送れ時間の総和が最小となるように配送オーダーを割り当てる。

初期割り当ての結果、あるトラックは自分の配送オーダーを実行することにより納期送れ時間が生じる。これら過負荷エージェントが公示を出すことにより、協調アルゴリズムは開始される。

なお、実験に使用した例題は、トラック数（即ち、エージェント数）が、15、30、60、120の4種類を用いた。

(2) PE割り付け

実験は16PE構成、及び、64PE構成のマルチPSIを使用した。1PEは、管理部に充て、残りをエージェント部に充てた。エージェント部には、PEを1、2、4、8、15、30、60台充てて並列度の評価を行なう。エージェントへのPEの割り付けは、サイクリックに行なうこととした。過負荷エージェントと過負荷でないエージェントでは、計算量が異なる。理想的には過負荷エージェントと過負荷でないエージェントを適切に異なるPEに割り付けることにより各PEの計算量を均一化できればよい。しかし、協調中に過負荷からそうでなくなるエージェ

表 1: 同期式協調方式・解の質

エージェント数	評価関数値の和の変化	総メッセージ数	落札数
15	1,642 → 479	9,077	21
30	823 → 276	22,745	16
60	464 → 168	68,877	12
120	248 → 84	118,440	5

ント、また、逆の変化をとるエージェントが発生し、それらを静的に検出するのは不可能である。また、動的割り付けを行なうにしても、そのためのオーバーヘッドの見積りも困難である。そこで、静的に各 PE の担当エージェント数が均一となるように PE 割り付けを行なうこととした。

(3) 測定項目

アルゴリズムの機能評価、及び、実行性能評価を行なう。機能評価は、結果の解の質により行なう。具体的には、協調開始前、後の評価関数値（即ち、納期送れ時間）の全エージェントでの和の変化を計測する。また、有効な協調がどの程度行なわれたかを知る為、落札回数（タスクの依頼回数）を計測する。実行性能評価は、処理時間により行なう。PE 数を変えた場合の評価のみならず、処理時間と関係が深いメッセージ数を計測する。

5.2 実験結果

同期式協調方式を取った場合の解の質を表 1に、処理時間を表 2に、まとめる。 同期式では、PE 数に関係無く、得られる結果（評価関数値の和）、及び、メッセージ数は変わらないので、解の質の実験結果（表 1）には、エージェント用に 15PE 使用したものを載せ、又、メッセージ数を同表に入れた。処理時間に関しては、各実験を 2 回ずつ行ないその結果を表 2に載せる。

非同期式協調方式を取った場合の解の質、及び、処理時間を表 3にまとめる。非同期式協調方式では、エージェントの動作間に同期を持たせていないので、過負荷でないエージェントにタスク公示がいつ届くかのタイミングが同じエージェント数、同じ PE 構成であっても、実行するたびに異なる。これにより、終了までには各エージェントがとったステップ数や落札数が異なり、その結果、得られる解、及び、処理時間も異なる。従って異なる PE 台数による処理時間から台数効果を評価しても無意味である。ここでは、エージェント部用に 15PE を使用し、各実験に関して 5 回行なった結果を載せる。この結果より、実際、実行する度に結果（解の質、及び、処理時間）が異なっていることが確認できる。

表 2: 同期式協調方式・処理時間

表の各項目の内容は以下の通り。

第 1 行目：第 1 回目処理時間 (単位は秒)

第 2 行目：第 2 回目処理時間 (単位は秒)

第 3 行目：第 1 回目と第 2 回目の平均値 (単位は秒)

第 4 行目：台数効果 (PE 台数が 1 の場合の処理時間との比)

エージェント数	PE 数						
	1	2	4	8	15	30	60
15	2,073	1,236	738	609	478	-	-
	2,067	1,237	738	619	480	-	-
	2,070	1,237	738	614	479	-	-
	1	*1.67*	*2.80*	*3.37*	*4.32*	-	-
30	2,880	1,651	1,127	856	702	642	-
	2,890	1,661	1,117	853	703	640	-
	2,885	1,656	1,121	855	703	641	
	1	*1.74*	*2.57*	*3.37*	*4.10*	*4.50*	-
60	5,197	3,886	2,788	1,713	1,148	1,037	978
	5,387	3,912	2,795	1,695	1,145	1,046	971
	5,292	3,899	2,792	1,704	1,147	1,042	975
	1	*1.36*	*1.89*	*3.10*	*4.61*	*5.08*	*5.43*
120	4,476	2,569	1,755	1,329	789	667	620
	4,469	2,625	1,759	1,319	784	667	620
	4,473	2,597	1,757	1,324	787	667	620
	1	*1.72*	*2.54*	*3.37*	*5.68*	*6.71*	*7.21*

表 3: 非同期式協調方式・計測結果

（注）表の横 1 行が 1 回の実行結果を表す。

エージェント数	評価関数値の和の変化	総メッセージ数	落札数	処理時間（秒）
15	1,642 → 713	5,465	28	101
	1,642 → 789	4,318	25	102
	1,642 → 763	4,637	25	130
	1,642 → 789	4,526	25	101
	1,642 → 956	4,670	25	168
30	823 → 172	9,766	28	150
	823 → 126	10,897	26	168
	823 → 160	7,549	24	94
	823 → 160	5,854	26	121
	823 → 160	5,594	25	89
60	464 → 174	5,288	11	120
	464 → 224	4,345	9	101
	464 → 224	4,244	9	101
	464 → 224	4,304	9	102
	464 → 224	4,282	9	95
120	248 → 84	1,050	2	44
	248 → 84	1,453	3	54
	248 → 84	1,551	3	53
	248 → 85	1,749	3	79
	248 → 84	1,555	3	52

5.3 評価

(1) 解の質について

このアルゴリズムが適用される場合として、多くの処理時間をかけて良い解を求める場合と、時間が限られていてその範囲でとにかく何らかの解を求める場合が考えられる。前者の評価基準として、最終結果、及び、全処理時間を用いる。後者の評価基準としては、同期式、非同期式、共に、タスク割り当てが発生すると確実に解は改良されるので、割り当て結果が1回替わるのに要する時間を用いる。

最終結果は、エージェント数が15、60の場合は同期式の方が、30の場合は非同期式の方がよく、120の場合は同じである。また、処理時間を比較しても、非同期式に要する時間は同期式の1/5から1/11と、短い時間ですんでいる。

どの程度の頻度でタスク割り当てが発生するかは、同期式の場合は1ステップにかかった時間を計測することにより得られるが、非同期式の場合は、ステップの切れ目がエージェントにより異なるので計測が困難である。そこで、近似値として、 全処理時間 ÷ 落札回数を採用した。PE数が15の場合のこの値は、同期式の場合、25.3（15エージェント）、51.0（30エージェント）、110.9（60エージェント）、192.6（120エージェント）であり、非同期式の場合、3.6～6.7（15エージェント）、3.6～5.4（30エージェント）、10.6～11.2（60エージェント）、17.3～22.0（120エージェント）、である。これより、非同期式が同期式の1/4から1/11の短い時間でタスク割り当てを変えている。但し、1回のタスク割り当てによる解の改善度は、評価関数値の変化量から判断して同期式の方が良いと判断される。

以上のことから、解の質に関しては、同期式と非同期式では優劣付けがたく、処理速度に関しては、非同期式が優れてると結論できそうであるが、ひとつ問題がある。即ち、非同期式は同じPE構成、同じエージェント数、同じ初期タスク割り当てで行なっても、行なう度に結果が異なるということである。更に実験を試みることにより、その分布がどのようなものであるかを明らかにしなければならない。

(2) 処理時間について

同期式協調方式に関する評価を行なう。同期式協調方式の通信時間を除いた処理時間は、1PEの場合には全エージェントの処理時間の合計になる。1＜PE数＜エージェント数 の場合には、各ステップでの処理時間最大のPEの処理時間を、全ステップで合計したものになる。ここで、PEの1ステップの処理時間はそのPEに割り付けられたエージェントの処理時間の合計になる。各ステップでエージェントはマネージャ（タスクを公示したエージェント）か潜在的コントラクタのどちらかとして働く。あるマネージャは他マネージャの潜在的コントラクタとしても働くので、マネージャ独自の処理の分、マネージャの処理時間は潜在的コントラクタとしてしか働かないエージェントよりも長くなる。そこで、あるステップで最も処理時間の長くなるPEは、そのPEに割り付けられたエージェントでマネージャとして働くエージェン

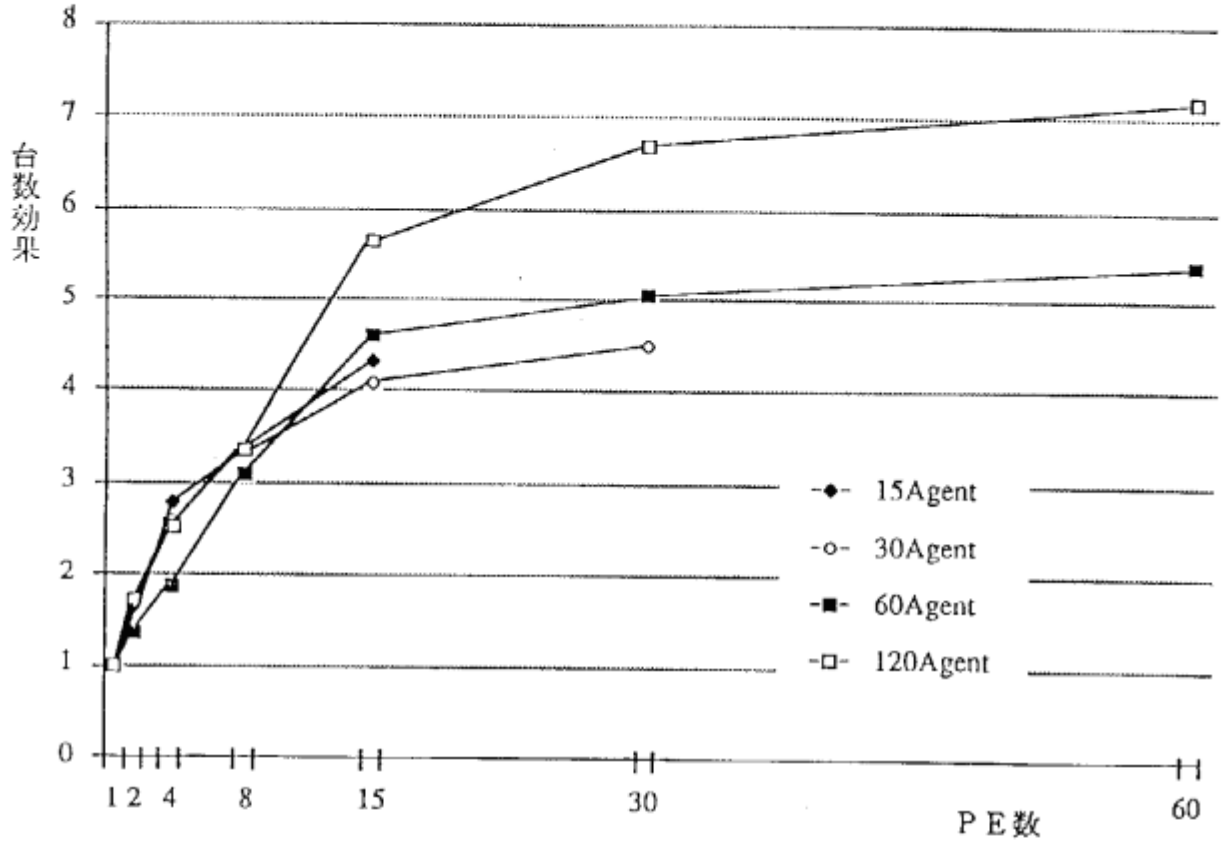


図 7: 同期式協調方式に於ける台数効果

トの数が最も多いPEになると考えられる。PE数＝エージェント数の場合には、通信時間を除いた処理時間は1ステップのマネージャの処理時間をステップ数倍した時間になる。

1PEの処理時間は問題により異なるが、全エージェント数、マネージャ数の割合、協調終了までのステップ数に処理時間は支配される。エージェント数が多い程、全エージェントに占めるマネージャの割合が高い程、そして交渉ステップ数が多い程、処理時間は長くなる。また1＜PE数＜エージェント数 の場合には、更には、エージェントのPE割付によっても処理時間が異なってくる。即ち、同じPEに割り付けられたマネージャの数が処理時間に大きな影響を及ぼす。理想的には全交渉ステップを通して同PEのマネージャ数が最小になるように割り付けられれば、処理時間を最短にできるはずである。しかしマネージャの分布は動的に変化するので、予めそのようなPE割付をとることは不可能である。そこで今回の実験ではエージェントを単にサイクリックにPEに割り付けている。その結果、4つの問題における同期式協調方式の台数効果は、図7のようになった（表2から作成）。この図からも分かるように、PE数が15以下の場合は概ね使用PE数のルート程度の台数効果が得られているが、PE数が増加するとかなり劣化する。また、60エージェントの問題で2PE、4PEの場合の台数効果が他と比べ

て悪い。交渉の全過程をトレースしてマネージャの分布を調査した結果、マネージャの分布が特定のPEに片寄っていることが判明し、これが台数効果を低くしている原因と考えられる。そこでマネージャの分布が出来るだけ均等になるようにトレース結果からPE割付を決定し、処理時間を測定すると台数効果は良くなることが確認できた。従って、1＜PE数＜エージェント数 の時、台数効果を高めるためには、静的に分析できる範囲でもマネージャの分布を均等になるようにPE割付を行なうべきである。

PE数＝エージェント数 の場合には、通信時間を無視した理想的な台数効果は次式で与えられる。

$$\frac{C_{ann}T_{mngr}+C_{step}N_aT_{cntr}}{C_{step}(T_{mngr}+T_{cntr})}$$

ここで、T_{mngr} ：1ステップでのマネージャとしての処理時間、T_{cntr} ：1ステップでの潜在的コントラクタとしての処理時間、N_a ：全エージェント数－1、C_{step} ：協調終了までの全ステップ数、C_{ann} ：交渉中の公示されたタスク数の総和（＝交渉中マネージャとして働いたエージェント数の総和）、とする。この式の左辺に15エージェントの問題で15PEを使用した場合の台数効果の値を適用して、右辺のT_{mngr}とT_{cntr}の比を求めると以下の値になる。

$$T_{mngr}:T_{cntr}=1:0.12$$

実際にはメッセージ通信時間やGCの影響等があるので、この値は正確なものではないが、マネージャとしての処理時間と比べて、潜在的コントラクタとしての処理時間はかなり短いものと考えられる。従って、1エージェント1PEのPE割付が可能な場合でも、潜在的コントラクタとしてしか働かないエージェントが割り付けられたPEは何もしていない時間がかなり存在することになる。問題の性質（T_{mngr}、T_{cntr}、N_a、C_{step}、C_{ann}により定まる）によってもこの傾向は多少異なるかもしれないが、一般に全エージェントに占めるマネージャの割合が高い程、1エージェント1PEのPE割付をとった場合の台数効果は高くなると考えられる。実際この傾向は実験によって確認された（結果は割愛）。結局、1エージェント1PEのPE割付をとる限り、問題自体に内在する性質により台数効果が決まってしまい、全エージェントに占めるマネージャの割合が低い場合には、高い台数効果を期待できない。従って、さらに台数効果を高めるためには、潜在的コントラクタしか存在していないPEに対して、マネージャの処理の一部を動的に割り付ける工夫が必要である。

6 おわりに

並列推論マシンのアプリケーションとしてタスク割り当て問題を対象とした、大域的制御や情報を持たないエージェントによる並列協調アルゴリズムとして、同期式協調方式と非同期式協調方式を開発した。これらの方式は、タスクの公示、入札、落札を1ステップとしこのステップを繰り返すことにより段階的により良いタスク割り当てを得てゆく方式である。探索を制御する知識が協調方式に組み込まれているのでそのような知識が獲得困難であるような問題向けとなっている。協調方式は同期式と非同期式の2方式からなる。

同期式協調方式では、ステップがエージェント間で同期を取って行なわれる。まず、タスク公示が一斉に出される。一般にエージェントは複数個のタスク公示を受ける。その各々に対して、自分がそのタスクを実行することによる評価関数値の変化量を求め、その値を返すことにより入札する。その後、公示を出したエージェント間でこれらの値を共有することにより、そのステップ内での最適割り当て、即ち、評価関数値の全エージェントでの総和が一番減少する割り当てを決定した後、次のステップに移行する。

非同期式協調方式では、ステップがエージェント間で非同期に行なわれる。タスク公示がエージェント毎に異なったタイミングで出される。一般にエージェントは複数個のタスク公示を受ける。それらから、ある基準で応じるべき公示を1個選択しその公示タスクを実行することによる評価関数値の変化量を返すことにより、入札する。その後、落札通知（当選／落選）が来るまで、ロックを掛ける。即ち、それまでに到着する公示に対して入札拒否をする。落札通知がきた後、ロックを解除して次の公示を評価する。公示を出した側は、入札してきた者の中から自分と相手の過負荷量の和が一番減少する相手を選択し、タスクをその者に落札する。

これらの方式を評価する為、配送計画問題をドメインとしたシステムをマルチPSIで実現し、実験を行なった。その結果、同期式並列協調方式は非同期式並列協調方式に比べ、エージェント間で共有する情報が多く、その分、より最適なタスク割り当てが実現できるものの、その分、より多くの通信を用いているので、処理時間はより長くかかること、また、非同期式協調方式は得られる結果や処理時間が非決定的に変化することが確認できた。

今後の課題としては、更に多くのドメインへの適用実験を行なうことにより、今年度に得られた両方式の適性を明確化すること、及び、PE数が大きくなった場合の両方式の処理効率を実験を通じて確認することにより、大規模問題への適用の可能性や改良点を明確化することが挙げられる。

謝辞

本研究を推進するにあたって有益な助言を頂いた、新田室長、市吉室長代理を始めとするICOT第7研究室の皆様に深謝します。

参考文献

[Bond 88] Bond,A.H. et.al. *"Readings in Distributed Artificial Intelligence"* MORGAN KAUFMANN PUBLISHERS,INC., 1988.

[Davis 83] Davis,R. et.al. *"Negotiation as a Metaphor for Distributed Problem Solving"* Artificial Intelligence Vol.20, 1983, pp.63-109.

[Hewitt 86] Hewitt,C. *"Offices are open systems"* ACM Tracsactions on Office Information Systems, Vol.4, No.3, July 1986, pp.271-287.

[伊藤 90] 伊藤「自律分散システムはいかにして構成されるか」計測と制御, Vol.29, No.10, 1990年10月, pp.1-5.