一般化相互割当問題のための分散ラグランジュ緩和プロトコル

平山, 勝敏

(Citation)
電子情報通信学会論文誌. D-I, 情報・システム, I-情報処理,88(9):1269-1277

(Issue Date)
2005-09-01

(Resource Type)
journal article

(Version)
Version of Record

一般化相互割当問題のための分散ラグランジュ緩和プロトコル

平山　勝敏[†a)]

Distributed Lagrangean Relaxation Protocol for the Generalized Mutual Assignment Problem

Katsutoshi HIRAYAMA[†a)]

あらまし　容量制約がある複数の機械（エージェント）に仕事（ジョブ）を適切に割り当てるという問題は，オペレーションズリサーチの分野では一般化割当問題として比較的古くから研究されている．本研究では，この一般化割当問題を拡張し，複数のエージェントが相互にジョブを適切に割り当てることを目指す一般化相互割当問題を導入する．また，それを解く分散型の解法である分散ラグランジュ緩和プロトコルを提案し，その性質を実験により明らかにする．

キーワード　分散問題解決，一般化割当問題，ラグランジュ緩和

1. まえがき

容量制約がある複数の機械（エージェント）に仕事（ジョブ）を適切に割り当てるという問題は，オペレーションズリサーチの分野では組合せ最適化問題の一種である一般化割当問題（Generalized Assignment Problem：GAP）として比較的古くから研究されている．GAPでは，割り当てるべきジョブをもつ1人のコーディネータが，すべてのエージェントの容量制約を満たしながら，効用が最大になるように各ジョブをどれか一つのエージェントに割り当てる．GAPはNP困難問題に属することが知られており，厳密解法だけでなく発見的解法に関する研究も多い[5]．

本研究では，まず，マルチエージェントシステム（Multi-Agent System：MAS）における問題に適合するようGAPを拡張した一般化相互割当問題（Generalized Mutual Assignment Problem：GMAP）を導入する．直観的には，GAPは，1人のコーディネータが複数のエージェントにジョブを適切に割り当てるという問題であるのに対し，GMAPは，すべての参加者がコーディネータ兼エージェントとなり，それぞれが自分のジョブを自身を含む他の参加者に適切に割り当てるという問題である．

GMAPでは，参加者がもつすべてのジョブをいわゆるスーパコーディネータにいったん集め，スーパコーディネータが全体の問題をGAPとして解くという解法も可能であり，また，場合によってはそのような解法の方が効率的でさえある．しかしながら，MASでは，参加者のセキュリティやプライバシーの問題，計算サーバとしてのスーパコーディネータの管理コストの問題等，主として効率以外の面から，そのようないわゆる集中型の解法は好まれない．そこで本研究では，そのようなスーパコーディネータを用いない分散型の解法である分散ラグランジュ緩和プロトコルを提案し，その性質を実験により明らかにする．

MASの一分野である分散問題解決では，この種の分散タスク割当問題に対する解法は，有名な契約ネットプロトコル[7]以来，様々な定式化をもとにしたものが提案されているが，筆者の知る限り，GAPをもとにした解法を提案した研究は皆無である．また，最適化問題に対する分散型の解法に関する研究としては，Androulakisらによる先駆的な研究[1]や西によるサプライチェーン計画問題に対する分散協調型解法の研究[4]などがあるが，全般的にまだ数は少なく十分に研究がなされているとはいえない．

以下，本論文は次のように構成される．**2.**では，GMAPを互いに制約条件の一部を共有する整数計画問題の集合として定義し，また，それぞれの割当制約

† 神戸大学海事科学部，神戸市
Faculty of Maritime Sciences, Kobe University, Kobe-shi, 658-0022 Japan
a) E-mail: hirayama@maritime.kobe-u.ac.jp

を緩和して得られるラグランジュ緩和問題を導入する．次に，**3.** では，分散ラグランジュ緩和プロトコルの基本的なアイデアを述べた後，その詳細を示す．**4.** では，分散ラグランジュ緩和プロトコルの性質を実験により評価し，**5.** で本研究のまとめを示す．

2. 問題の定式化

2.1 一般化相互割当問題

一般化割当問題は，次のような整数計画問題として定式化できる．

$$
\begin{aligned}
&\mathcal{GAP} \quad (\text{decide } x_{ij},\ \forall i \in A,\ \forall j \in J):\\
&\max. \quad \sum_{i\in A}\sum_{j\in J} p_{ij}x_{ij} && (1)\\
&\text{s. t.} \quad \sum_{i\in A} x_{ij} = 1,\ \ \forall j \in J, && (2)\\
&\qquad \sum_{j\in J} w_{ij}x_{ij} \le c_i,\ \ \forall i \in A, && (3)\\
&\qquad x_{ij} \in \{0,1\},\ \forall i \in A,\ \forall j \in J, && (4)
\end{aligned}
$$

ここで，$A = \{1,\ldots,m\}$ はエージェント集合，$J = \{1,\ldots,n\}$ はジョブ集合，p_{ij} はジョブ j をエージェント i に割り当てた場合の効用，w_{ij} はエージェント i がジョブ j を実行する場合の資源消費量，c_i はエージェント i が消費できる資源の総量（資源容量）である．また，x_{ij} は，エージェント i にジョブ j を割り当てる場合は 1，そうでない場合は 0 に設定される決定変数である．GAP の目的は，各ジョブがただ一つのエージェントに割り当てられ（割当制約 (2)），どのエージェントもその資源消費量の合計が資源容量を超えない（ナップサック制約 (3)），かつ，どのジョブもエージェントに割り当てられるか割り当てられないかのどちらかである（01 制約 (4)）という制約条件のもとで割当の効用の総和を最大化することであり，その最大値をこの問題の最適値，また，最適値を実現する割当をこの問題の最適解という．なお，GAP は NP 困難な問題に属する．

一般化相互割当問題は，それぞれ独自のエージェント集合 A_k，ジョブ集合 J_k からなる局所的な GAP をもつエージェント $k \in \{1,\ldots,m\}$ で構成され，大域的にはエージェント集合を $\bigcup_{k=1}^{m} A_k$，ジョブ集合を $\bigcup_{k=1}^{m} J_k$ とする GAP として定式化できる．なお，以降，$\bigcup_{k=1}^{m} A_k$ を A，$\bigcup_{k=1}^{m} J_k$ を J と簡略化して表記する．

[例 1] 二つのエージェント 1 と 2 が存在し，エージェント 1 はジョブ 1，エージェント 2 はジョブ 2 とジョブ 3 をもつ（$J_1 = \{1\}, J_2 = \{2,3\}$）．エージェント 1 はジョブ 1 を自分で実行するか，あるいは，エージェント 2 に実行してもらいたいと考えている（$A_1 = \{1,2\}$）．一方，エージェント 2 は，ジョブ 2 とジョブ 3 のそれぞれについて自分で実行するか，あるいは，エージェント 1 に実行してもらいたいと考えている（$A_2 = \{1,2\}$）．今，エージェント 1 の資源容量は 4（$c_1 = 4$），エージェント 2 の資源容量は 3（$c_2 = 3$）とし，ジョブ j がエージェント i に割り当てられた場合の資源消費量 w_{ij} と効用 p_{ij} が，ジョブ 1: $w_{11} = 2$, $p_{11} = 5$, $w_{21} = 2$, $p_{21} = 4$, ジョブ 2: $w_{12} = 2$, $p_{12} = 6$, $w_{22} = 2$, $p_{22} = 2$, ジョブ 3: $w_{13} = 1$, $p_{13} = 5$, $w_{23} = 2$, $p_{23} = 2$ のように与えられているものとする．この問題は大域的には次のような GAP として定式化できる．

$$
\begin{aligned}
&\mathcal{GAP} \quad (\text{decide } x_{11},x_{12},x_{13},x_{21},x_{22},x_{23}):\\
&\max. \quad 5x_{11}+6x_{12}+5x_{13}+4x_{21}+2x_{22}+2x_{23}\\
&\text{s. t.} \quad x_{11}+x_{21} = 1,\\
&\qquad x_{12}+x_{22} = 1,\\
&\qquad x_{13}+x_{23} = 1,\\
&\qquad 2x_{11}+2x_{12}+x_{13} \le 4,\\
&\qquad 2x_{21}+2x_{22}+2x_{23} \le 3,\\
&\qquad x_{ij} \in \{0,1\},\ \forall i \in \{1,2\},\ \forall j \in \{1,2,3\}.
\end{aligned}
$$

しかし，分散型の解法を目指す場合，このような大域的な定式化ではなく，個々のエージェントが全体の問題のうちのどの部分を担当するかを明記すべきである．本論文では，前述の定式化における決定変数 x_{ij} の値はエージェント i が決定する —— すなわち，ジョブの割当側ではなく受け手側に決定権がある —— ものとし，GMAP を，エージェント k に対する整数計画問題

$$
\begin{aligned}
&\mathcal{GMP}_k \quad (\text{decide } x_{kj}, \forall j \in R_k):\\
&\max. \quad \sum_{j\in R_k} p_{kj}x_{kj}\\
&\text{s. t.} \quad \sum_{i\in S_j} x_{ij} = 1,\ \forall j \in R_k,\\
&\qquad \sum_{j\in R_k} w_{kj}x_{kj} \le c_k,
\end{aligned}
$$

$$x_{kj} \in \{0,1\},\ \forall j \in R_k,$$

の集合 $\{\mathcal{GMP}_k | k \in A\}$ として定式化する．ここで，$R_k(\subset J)$ は，k 及び k 以外の他のエージェントが k に割り当てようとしているジョブの集合であり，$S_j(\subset A)$ はジョブ j が割り当てられる可能性のあるエージェントの集合である．他の要素については $\mathcal{GAP}$ の定式化で用いたものと同じである．なお，$\mathcal{GMP}_k$ の 1 番目の制約条件（割当制約）には，k 以外のエージェントによって決定される変数 $x_{ij}(i \neq k)$ が含まれることに注意されたい．

[例 2]　例 1 の問題は，次の二つの整数計画問題として定式化できる．

$\mathcal{GMP}_1$　(decide x_{11}, x_{12}, x_{13})：

$$\begin{aligned}
\text{max.}\quad & 5x_{11} + 6x_{12} + 5x_{13} \\
\text{s. t.}\quad & x_{11} + x_{21} = 1, \\
& x_{12} + x_{22} = 1, \\
& x_{13} + x_{23} = 1, \\
& 2x_{11} + 2x_{12} + x_{13} \leq 4, \\
& x_{1j} \in \{0,1\},\ \forall j \in \{1,2,3\};
\end{aligned}$$

$\mathcal{GMP}_2$　(decide x_{21}, x_{22}, x_{23})：

$$\begin{aligned}
\text{max.}\quad & 4x_{21} + 2x_{22} + 2x_{23} \\
\text{s. t.}\quad & x_{11} + x_{21} = 1, \\
& x_{12} + x_{22} = 1, \\
& x_{13} + x_{23} = 1, \\
& 2x_{21} + 2x_{22} + 2x_{23} \leq 3, \\
& x_{2j} \in \{0,1\},\ \forall j \in \{1,2,3\}.
\end{aligned}$$

これらの整数計画問題の集合には次の性質がある．

[補題 1]　すべてのエージェントが $\mathcal{GMP}_k$ の最適解を得て，かつ，これらの最適解において変数 x_{ij} の値が整合するならば，それらは $\mathcal{GAP}$ の最適解を構成している．

(証明)　すべての $\mathcal{GMP}_k$ の実行可能領域の交わりが $\mathcal{GAP}$ の実行可能領域であり，また，すべての $\mathcal{GMP}_k$ の目的関数の値を最大にすれば $\mathcal{GAP}$ の目的関数の値が最大になることから明らかである．　□

2.2　ラグランジュ緩和問題

$\mathcal{GAP}$ において割当制約を緩和したラグランジュ緩和問題は次のようになる．

$\mathcal{LGAP}(\mu)$　(decide x_{ij}, $\forall i \in A$, $\forall j \in J$)：

$$\begin{aligned}
\text{max.}\quad & \sum_{i \in A}\sum_{j \in J} p_{ij}x_{ij} + \sum_{j \in J} \mu_j \left(1 - \sum_{i \in A} x_{ij}\right) \\
\text{s. t.}\quad & \sum_{j \in J} w_{ij}x_{ij} \leq c_i,\ \forall i \in A, \\
& x_{ij} \in \{0,1\},\ \forall i \in A,\ \forall j \in J,
\end{aligned}$$

ここで，$\mu = (\mu_1, \ldots, \mu_n)$ は実数値をとるラグランジュ乗数ベクトルである．$\mathcal{LGAP}(\mu)$ の最適値が $\mathcal{GAP}$ の上界値を与えることは周知のとおりである．

2.1 で $\mathcal{GAP}$ をエージェント k に対する $\mathcal{GMP}_k$ に分解したときと同様の方法により，ラグランジュ緩和問題 $\mathcal{LGAP}(\mu)$ を分解して，以下に示すエージェント k に対するラグランジュ緩和問題 $\mathcal{LGMP}_k(\mu)$ を得る．

$\mathcal{LGMP}_k(\mu)$　(decide $x_{kj}, \forall j \in R_k$)：

$$\begin{aligned}
\text{max.}\quad & \sum_{j \in R_k} p_{kj}x_{kj} + \sum_{j \in R_k} \mu_j \left(\frac{1}{|S_j|} - x_{kj}\right) \\
\text{s. t.}\quad & \sum_{j \in R_k} w_{kj}x_{kj} \leq c_k, \\
& x_{kj} \in \{0,1\},\ \forall j \in R_k.
\end{aligned}$$

なお，$\mathcal{LGMP}_k(\mu)$ には，k 以外のエージェントによって決定される変数は含まれていないことに注意されたい．

[例 3]　例 2 におけるエージェント 1 とエージェント 2 に対するラグランジュ緩和問題はそれぞれ次のようになる．

$\mathcal{LGMP}_1(\mu)$　(decide x_{11}, x_{12}, x_{13})：

$$\begin{aligned}
\text{max.}\quad & 5x_{11} + 6x_{12} + 5x_{13} + \mu_1\left(\frac{1}{2} - x_{11}\right) \\
& + \mu_2\left(\frac{1}{2} - x_{12}\right) + \mu_3\left(\frac{1}{2} - x_{13}\right) \\
\text{s. t.}\quad & 2x_{11} + 2x_{12} + x_{13} \leq 4, \\
& x_{1j} \in \{0,1\},\ \forall j \in \{1,2,3\};
\end{aligned}$$

$\mathcal{LGMP}_2(\mu)$　(decide x_{21}, x_{22}, x_{23})：

$$\begin{aligned}
\text{max.}\quad & 4x_{21} + 2x_{22} + 2x_{23} + \mu_1\left(\frac{1}{2} - x_{21}\right) \\
& + \mu_2\left(\frac{1}{2} - x_{22}\right) + \mu_3\left(\frac{1}{2} - x_{23}\right) \\
\text{s. t.}\quad & 2x_{21} + 2x_{22} + 2x_{23} \leq 3, \\
& x_{2j} \in \{0,1\},\ \forall j \in \{1,2,3\}.
\end{aligned}$$

これらのラグランジュ緩和問題の集合には次の性質

がある．

[補題 2] すべてのエージェントが $\mathcal{LGMP}_k(\mu)$ の最適解を得た場合，それらは $\mathcal{LGAP}(\mu)$ の最適解を構成している．

(証明) 補題 1 と同様． □

[定理 1] すべてのエージェントの $\mathcal{LGMP}_k(\mu)$ の最適値の和は，$\mathcal{GAP}$ の上界値を与える．

(証明) 補題 2，及び，$\mathcal{LGAP}(\mu)$ の最適値が $\mathcal{GAP}$ の上界値を与えることから明らか． □

こうして得られる $\mathcal{GAP}$ の上界値はラグランジュ乗数ベクトルの値により変わる．上界値は小さい（最適値に近い）方が望ましいのだが，この上界値を最小化するようなラグランジュ乗数ベクトルの値を探す問題はラグランジュ双対問題と呼ばれている．この問題を解く際に次の性質は有用である．

[定理 2] すべてのエージェントが $\mathcal{LGMP}_k(\mu)$ の最適解を得て，かつ，それらが割当制約（$\sum_{i\in A} x_{ij} = 1,\ \forall j \in J$）を満たしているとき，それらの最適解は $\mathcal{GAP}$ の最適解を構成している．

(証明) 定理 1 より，それらの最適解は $\mathcal{GAP}$ の上界値を与える．それらが更に割当制約（緩和された等式制約）を満たすということは，それらが $\mathcal{GAP}$ の実行可能解を構成し，その下界値を与えることを意味している．すなわち，それらの最適解は，上界値と下界値の両方を与えることから，$\mathcal{GAP}$ の最適解を構成している． □

3. プロトコル

すべてのエージェントが，互いの暫定的な解を交換しながら並行して，ラグランジュ緩和問題 $\mathcal{LGMP}_k(\mu)$ とラグランジュ双対問題を交互に解く分散ラグランジュ緩和プロトコルの概要と詳細を示す．なお，以降，前者の問題を主問題，後者の問題を双対問題と呼ぶ．

3.1 近傍構造と通信モデル

主問題を解くために，エージェント k は，R_k（k 及び k 以外のエージェントが k に割り当てようとしているジョブの集合）内の各ジョブ j について，ラグランジュ乗数 μ_j の値と集合 S_j（j が割り当てられる可能性のあるエージェントの集合）のサイズを知る必要がある．一方，詳細は後述するが，双対問題を解く際の劣こう配計算のために，エージェント k は，R_k 内の各ジョブ j について，他エージェント i がもつ決定変数 x_{ij} の値を知る必要がある．これらを得るためにエージェント k は，集合 $\bigcup_{j\in R_k} S_j$ 内のエージェントと通信する必要があるが，この集合をエージェント k の近傍と呼ぶ．本研究では，各エージェントは近傍のそれぞれのエージェントと 1 対 1 のメッセージ通信を行うものとし，任意の 2 エージェント間で，メッセージは消失しないこと，また，メッセージの送受信の順序が保存されることを仮定する．

3.2 主問題の解法

関連するラグランジュ乗数の値が決定されると，エージェント k は主問題 $\mathcal{LGMP}_k(\mu)$ を解き，決定変数 x_{kj} $(\forall j \in R_k)$ の値を決める．これは，R_k 内のジョブ（各ジョブ j の効用は $p_{kj} - \mu_j$，資源消費量は w_{kj}）を取捨選択して，容量 c_k の「容器」に効用和が最大になるように詰めるナップサック問題を解くことに相当する．ナップサック問題は NP 困難であり効率的な厳密解法の存在は期待できないが，同問題に対する厳密解法の近年の発展は目覚ましく，1 万個程度のジョブからなる問題でも実際上は難なく解くことができる [3]．

3.3 双対問題の解法

近傍及び自身の決定変数の値が決定されると，エージェント k は，R_k 内の各ジョブ j に対応するラグランジュ乗数 μ_j の値を更新する．本研究では，双対問題を解く代表的な手法である劣こう配法 [6] を用いて μ_j の値を更新する．本研究の劣こう配法では，決定変数に対する現在の値のもとで，

$$G_j = 1 - \sum_{i\in S_j} x_{ij}$$

で計算される，緩和されたジョブ j の割当制約に対する劣こう配 G_j と，離散的な時間（反復回数）t とともに一定の割合 $r(0 < r \leq 1)$ で減少するステップ長 $l^{(t)}$（すなわち，$l^{(t+1)} = rl^{(t)}$），及び，集合 S_j のサイズを用いて，μ_j を次のように更新する．

$$\mu_j^{(t+1)} \leftarrow \mu_j^{(t)} - l^{(t)} \frac{G_j}{|S_j|} \tag{5}$$

ここで注意すべきことは，μ_j の値はジョブ j に固有のものであり，S_j に属するすべてのエージェントで μ_j に共通の値を代入することが，**2.2** の定理 1 と 2 を成立させるための必要条件だということである．これを実現するために本研究では，すべてのエージェントが，ラグランジュ乗数の初期値，及び，ステップ長の初期値 $l^{(0)}$ とその減少率 r にそれぞれ共通の値を設定するものとし，また，任意のエージェントは，その近傍及び自分の t 回目反復後の決定変数の値を見な

がら，次の $(t+1)$ 回目のためのラグランジュ乗数の値を劣こう配法により決定するものとする．これにより，S_j に属するエージェント間で明示的に通信を行って μ_j に共通の値を代入しなくても，S_j に属する全員が共通の初期値と共通の規則に従ってラグランジュ乗数を更新することになるため，一貫して μ_j に共通の値を代入することができる．

一方で，S_j に属するエージェントが μ_j に共通の値を代入しないとすると，定理 1 と 2 は成立しない．すなわち，例えば，そのような条件のもとに求められた各エージェントの主問題に対する最適解はもはや $\mathcal{GAP}$ の上界値を与える保証はない．ところが，エージェントによるこのような「勝手な」ラグランジュ乗数の更新が，(最適解ではないかもしれないが) 実行可能解への比較的早い収束をもたらす．詳しくは **3.6** で説明する．

3.4 終了判定

μ に対するある値のもとで，全エージェントの主問題の最適解が割当制約 $\sum_{i\in A} x_{ij} = 1, \quad \forall j \in J$ を満たした場合，定理 2 より $\mathcal{GAP}$ の最適解が得られているため反復手続きを終了してよい．この判定問題は分散制約充足問題 [8] の終了判定の問題に他ならない．分散制約充足問題の終了判定は，分散ブレイクアウト法 [2] などの局所同期型の解法では比較的容易に実現でき，本研究でもそれと同じ方法を用いる．概要は次のとおりである．

各エージェント k は，自分の決定変数の値とともに，論理変数 cs_k の値と非負の整数値をとる変数 tc_k (初期値は 0) の値を近傍に送る．エージェント k は近傍から決定変数の値を受信することにより，全体の割当制約の一部，すなわち，$\sum_{i\in S_j} x_{ij} = 1, \forall j \in R_k$ が充足されているか否かを局所的に判定することができる —— R_k 内の任意のジョブ j の劣こう配 G_j がゼロであれば充足されている —— が，それらがすべて充足されていれば論理変数 cs_k を真とし，そうでなければ偽とする．一方，近傍から受け取った論理変数の値がすべて真で自分の論理変数 cs_k の値も真ならば，tc_k の値を，近傍プラス自分の範囲での現在の tc の最小値プラス 1 に設定する．そうでなければ tc_k の値をゼロに戻す．なお，この結果，tc_k の値が全体のエージェント数に等しくなれば，すべてのエージェントの割当制約が充足されている．この手続きで終了判定可能なことは，[2] と全く同様に証明できる．

3.5 実行例

例 3 のラグランジュ緩和問題に対する実行例を示す．ラグランジュ乗数ベクトルの初期値を $\mu = (\mu_1, \mu_2, \mu_3) = (0, 0, 0)$ とする．また，エージェント 1 とエージェント 2 はともに，ステップ長の初期値 $l^{(0)}$ を 1，その減少率 r を 1 に設定するものとする．すなわちステップ長は 1 に固定する．

まず，エージェント 1 は，ジョブ 1 (効用 5，資源消費量 2)，ジョブ 2 (効用 6，資源消費量 2)，ジョブ 3 (効用 5，資源消費量 1) を容量 4 の容器に詰めるナップサック問題を解き，自分への割当としてジョブ 1 とジョブ 2 を選び，その旨を近傍 (エージェント 2) に伝える．一方，エージェント 2 は，ジョブ 1 (効用 4，資源消費量 2)，ジョブ 2 (効用 2，資源消費量 2)，ジョブ 3 (効用 2，資源消費量 2) を容量 3 の容器に詰めるナップサック問題を解き，自分への割当としてジョブ 1 を選び，その旨を近傍 (エージェント 1) に伝える．

近傍の全エージェントからの割当を受け取ると，エージェント 1 は，まず，現在の割当のもとでの各ジョブの劣こう配を $(G_1, G_2, G_3) = (-1, 0, 1)$ のように計算し，ラグランジュ乗数ベクトルを $\mu = (0.5, 0, -0.5)$ に更新する．エージェント 2 も全く同様に μ を更新する．次に，この μ のもとで，エージェント 1 は，ジョブ 1 (効用 4.5，資源消費量 2)，ジョブ 2 (効用 6，資源消費量 2)，ジョブ 3 (効用 5.5，資源消費量 1) を容量 4 の容器に詰めるナップサック問題を解き，自分への割当としてジョブ 2 とジョブ 3 を選び，その旨をエージェント 2 に伝える．一方，同じ μ のもとで，エージェント 2 は，ジョブ 1 (効用 3.5，資源消費量 2)，ジョブ 2 (効用 2，資源消費量 2)，ジョブ 3 (効用 2.5，資源消費量 2) を容量 3 の容器に詰めるナップサック問題を解き，自分への割当としてジョブ 1 を選び，その旨をエージェント 1 に伝える．

この割当を受け取るとエージェント 1 は，劣こう配が $(G_1, G_2, G_3) = (0, 0, 0)$ —— すなわち割当制約が満たされている —— ゆえ，ラグランジュ乗数ベクトルを更新しない．したがって，次の割当もジョブ 2 とジョブ 3 のままであり，論理変数 cs_1 を真に設定して，割当とともにエージェント 2 に送る．一方，エージェント 2 も同様にジョブ 1 の割当を維持し，論理変数 cs_2 を真に設定して，エージェント 1 に送る．

これらのメッセージを受け取ると，エージェント 1 とエージェント 2 はともにそれぞれの tc_1，tc_2 の値

```
procedure init
1.  round_k := 1;
2.  cs_k := false;
3.  tc_k := 0;
4.  leng := someCommonValue;
5.  r := someCommonRatio;
6.  δ := someCommonRatio;
7.  for each job j ∈ R_k do μ_j := 0 end do;
8.  jobs_k := optimal solution to a knapsack problem;
9.  send assign(k, round_k, cs_k, tc_k, jobs_k) to neighbors;
10. waitL := neighbors;
11. cs_N := true;
```

図 1 init 手続き
Fig. 1 Init procedure.

```
when k receives assign(i, round_i, cs_i, tc_i, jobs_i) from i do
1.  if round_k < round_i then
2.    add this message in deferredL;
3.  else
4.    if cs_i then
5.      tc_k := min(tc_k, tc_i);
6.    else
7.      cs_N := false;
8.      update agentView with jobs_i;
9.    end if;
10.   delete i from waitL;
11.   if waitL is empty then
12.     stop := localcomp;
13.     if stop then
14.       terminate the procedure;
15.     else
16.       waitL := neighbors;
17.       cs_N := true;
18.       process deferredL;
19.     end if;
20.   end if;
21. end if;
end do
```

図 2 メッセージ受信後の手続き
Fig. 2 Message processing procedure.

を 0 から 1 に増やし，同様に同じ割当を送り合う．その結果，次の反復で両者の tc が 2（全エージェント数）に達し，全体の割当制約が満たされていることを両者が知って終了する．得られた割当は，エージェント 1 がジョブ 2 とジョブ 3，エージェント 2 がジョブ 1 を担当するというものであり，これは最適解になっている．

3.6 実行可能解への収束

以上の手続きで各反復ごとに得られる主問題の最適値の和は，$\mathcal{GAP}$ の上界値を与えることは先の定理 1 で示したとおりである．このときの割当が，更に割当制約（$\sum_{i \in A} x_{ij} = 1, \forall j \in J$）を満たしたときには，定理 2 よりそれは $\mathcal{GAP}$ に対する最適解となるが，そのような割当が必ず見つかるという保証はない．そのため，$\mathcal{GAP}$ に対する実行可能解が必要な場合には，上

```
procedure localcomp
1.  round_k := round_k + 1;
2.  leng := leng * r;
3.  if round_k > cutOffRound then
4.    return true;
5.  else
6.    cs_k := true;
7.    for each job j ∈ R_k do
8.      calculate subgradient G_j based on agentView;
9.      if G_j ≠ 0 then
10.       cs_k := false;
11.       ε := randomly chosen value from [−δ, δ];
12.       μ_j := μ_j − (1 + ε) * leng * G_j/|S_j|;
13.     end if;
14.   end do;
15.   if cs_k ∧ cs_N then
16.     tc_k := tc_k + 1;
17.     if tc_k = #agents then return true end if;
18.   else
19.     tc_k := 0;
20.     jobs_k := optimal solution to a knapsack problem;
21.   end if;
22.   send assign(k, round_k, cs_k, tc_k, jobs_k) to neighbors;
23.   return false;
24. end if;
```

図 3 localcomp 手続き
Fig. 3 Localcomp procedure.

界値を与える実行不可能解を実行可能解に変形する何らかの仕組みを解法に組み込む必要がある．このような仕組みは一般にラグランジアンヒューリスティックと呼ばれ，通常，問題ごとに固有のものが考案される．

本研究では，$\mathcal{GAP}$ の実行可能解を求めるための簡単な手法を導入する．本手法は，実行不可能解を実行可能解に明示的に変形するのではなく，前述の一連の手続きのうち双対問題を解く劣こう配法に少し変形を加え，(最適解ではないかもしれないが) 実行可能解へ直接収束させようというものである．

3.3 でも述べたとおり，双対問題を解く場合，任意のラグランジュ乗数 μ_j に対し，関連するエージェントが常に同じ値を代入しておくことが，エージェントを $\mathcal{GAP}$ の最適解に収束させるための必要条件である．一方で，μ_j の値が共通だと，S_j 内の一部のエージェント同士が，ジョブ j に対して集合・離散を繰り返すといういわゆる振動が起こりやすく，最適解に収束しないことが多い．そこで本研究では，μ_j の値は共通であるという条件を緩め，S_j 内の各エージェントは μ_j に異なる値を代入してもよいことにする．ここでは単純に，μ_j の更新式 (5) における変化分 $l^{(t)} \frac{G_j}{|S_j|}$ をプラスマイナス最大 $\delta (0 \leq \delta \leq 1)$ の割合でエージェントごとにランダムに変動させる．すなわち，更新式 (5) を次で置き換える．

$$\mu_j^{(t+1)} \leftarrow \mu_j^{(t)} - (1+N_\delta)l^{(t)}\frac{G_j}{|S_j|} \qquad (6)$$

ここで，N_δ は $[-\delta,\delta]$ 上で一様分布する確率変数である．なお，$\delta=0$ のとき，更新式 (6) は更新式 (5) に等しくなる．以上の分散ラグランジュ緩和プロトコルの手続きをまとめると図 1〜図 3 のようになる．

4. 実　　験

筆者の知る限り，GMAP は新しい問題であり，本プロトコルと直接比較可能な既存の解法は存在しない．そこで，本研究では，提案したプロトコルの優位性を示すためではなく，その性質 —— 現時点では実験的にしか示し得ない性質 —— を明らかにするための実験を行う．具体的には，**3.6** で説明した δ の設定値と本プロトコルの性能との関係を実験により示す．

この実験では，エージェント数が $m \in \{3,5,7\}$，ジョブ数が $n=5m$ —— 各エージェントが五つのジョブをもつ —— とし，エージェント間のジョブを割り当てる/割り当てられるという関係が次に示す特殊な形状を構成する例題をランダムに生成した．

鎖（*chain*）　i 番目のエージェントは，各ジョブを $i-1$ 番目のエージェント，$i+1$ 番目のエージェント，または自分のうちのどれかに割り当てようとする．ただし，1 番目のエージェントは，2 番目か自分，m 番目のエージェントは，$m-1$ 番目か自分に割り当てようとする．

環（*ring*）　i 番目のエージェントは，各ジョブを $i-1$ 番目のエージェント，$i+1$ 番目のエージェント，または自分のうちのどれかに割り当てようとする．ただし，1 番目のエージェントは，2 番目のエージェント，m 番目のエージェント，または自分に，m 番目のエージェントは，$m-1$ 番目，1 番目のエージェント，または自分に割り当てようとする．

完全（*cmplt*）　各エージェントは，各ジョブを自分を含むすべてのエージェントのうちのどれかに割り当てようとする．

ランダム 3（*rndm3*）　各エージェントは，各ジョブを，ジョブごとにランダムに選んだ三つのエージェントのうちのどれかに割り当てようとする．ただし，この三つのエージェントには必ず自分を含めるものとする．

各例題では，エージェント i にジョブ j を割り当てた場合の資源消費量 w_{ij}，及び，効用 p_{ij} には 1 以上 10 以下の整数をランダムに設定し，各エージェント i の資源容量 c_i はすべて 20 とした．なお，このように例題を生成した後に，その例題が実行可能解をもつかどうかを集中型のアルゴリズムにより検査し，実行可能でない例題は除外した．

本プロトコルを実行するエージェントは Java で実装されており，エージェント同士はソケットを介して通信を行う．今回の実験では，1 台の計算機上に m 個のエージェントを置き，ローカルなポートを使って互いに通信を行わせた．プロトコルのパラメータの設定値は，反復回数（ラウンド数）の上限 $cutOffRound=100n$，ステップ長の初期値 $l^{(0)}=1.0$，ステップ長の減少率 $r=1.0$，及び，ラグランジュ乗数の値の変化分の最大変動率 $\delta \in \{0.0,0.3,0.5,1.0\}$ である．各例題に対して，δ のそれぞれの値につき計 20 回の試行を行い，

Opt.Ratio：　最適解が得られた試行回数の割合

Fes.Ratio：　実行可能解が得られた試行回数の割合

Avg/Bst.Quality：　得られた実行可能解の評価値の最適値に対する割合の平均値/最良値

Avg.Cost：　実行可能解が得られるまでのラウンド数の平均値（ただし，上限以内に実行可能解を求めることができなかった場合は *cutOffRound* の値を用いる）

をそれぞれ求めた．ただし，$\delta=0.0$ の場合には，本プロトコルに非決定性はないため試行回数は 1 とした．結果を表 1 に示す．なお，表中の Pr.ID の欄の値は，左から，形状，エージェント数，ジョブ数，各エージェントの資源容量，問題識別子を表す．

$\delta=0.0$ の場合，前述のとおり，更新式 (6) は更新式 (5) に等しくなるため，本プロトコルの終了条件が満たされた場合，全体の $\mathcal{GAP}$ に対する最適解が得られることになる．しかし，今回の実験では，ほとんどの例題において，設定した上限ラウンド数以内に最適解は得られていないことが分かる．最適解が得られない場合の本プロトコルの動作を詳しく調べた結果，やはり **3.6** で述べたような振動が起きていることが観測された．

一方で，δ に 0.0 以外の値を設定することによる効果は総じて劇的である．δ に 0.0 以外の値を設定すると，各ジョブ j のラグランジュ乗数 μ_j を，S_j 内のエージェントはそれぞれ異なる値に更新する．その結果，直観的には，j に関する割当制約がなかなか満たされない場合に，それに対する S_j 内の各エージェントの「見解」が次第に分かれていくことになる．これにより，前述のような振動が回避され，本プロトコルは比較的速く実行可能解に収束できたと想像される．

表1 実験結果
Table 1 Experimental results.

Pr.ID	δ	O.R.	F.R.	A.Q.	B.Q.	A.C.	Pr.ID	δ	O.R.	F.R.	A.Q.	B.Q.	A.C.
chain-3-15-20-002	0.0	0/1	0/1	N/A	N/A	1500	rndm3-5-25-20-000	0.0	0/1	0/1	N/A	N/A	2500
	0.3	13/20	20/20	0.990	1.000	82.0		0.3	8/20	20/20	0.977	1.000	527.3
	0.5	8/20	20/20	0.984	1.000	93.2		0.5	6/20	19/20	0.965	1.000	428.8
	1.0	6/20	20/20	0.962	1.000	69.8		1.0	2/20	20/20	0.951	1.000	288.6
chain-3-15-20-004	0.0	1/1	1/1	1.000	1.000	22	rndm3-5-25-20-001	0.0	0/1	0/1	N/A	N/A	2500
	0.3	16/20	20/20	0.988	1.000	70.5		0.3	19/20	20/20	0.998	1.000	40.6
	0.5	14/20	20/20	0.993	1.000	32.7		0.5	17/20	20/20	0.994	1.000	66.6
	1.0	6/20	20/20	0.943	1.000	74.1		1.0	10/20	20/20	0.957	1.000	53.8
cmplt-3-15-20-000	0.0	0/1	0/1	N/A	N/A	1500	chain-7-35-20-000	0.0	0/1	0/1	N/A	N/A	3500
	0.3	4/20	20/20	0.974	1.000	224.3		0.3	3/20	19/20	0.969	1.000	810.7
	0.5	1/20	20/20	0.977	1.000	93.4		0.5	1/20	19/20	0.965	1.000	505.3
	1.0	1/20	20/20	0.952	1.000	57.2		1.0	0/20	19/20	0.959	0.986	473.8
cmplt-3-15-20-001	0.0	0/1	0/1	N/A	N/A	1500	chain-7-35-20-001	0.0	0/1	0/1	N/A	N/A	3500
	0.3	6/20	20/20	0.972	1.000	216.9		0.3	5/20	20/20	0.983	1.000	359.0
	0.5	8/20	20/20	0.978	1.000	93.4		0.5	2/20	20/20	0.977	1.000	223.1
	1.0	0/20	20/20	0.933	0.980	91.7		1.0	3/20	20/20	0.966	1.000	283.5
chain-5-25-20-000	0.0	0/1	0/1	N/A	N/A	2500	ring-7-35-20-000	0.0	0/1	0/1	N/A	N/A	3500
	0.3	5/20	20/20	0.989	1.000	269.6		0.3	3/20	18/20	0.959	1.000	993.7
	0.5	2/20	20/20	0.978	1.000	177.9		0.5	2/20	20/20	0.955	1.000	359.5
	1.0	1/20	20/20	0.964	1.000	148.0		1.0	0/20	20/20	0.946	0.996	164.0
chain-5-25-20-001	0.0	0/1	0/1	N/A	N/A	2500	ring-7-35-20-001	0.0	0/1	0/1	N/A	N/A	3500
	0.3	11/20	20/20	0.995	1.000	84.2		0.3	1/20	20/20	0.949	1.000	1013.0
	0.5	13/20	20/20	0.993	1.000	146.3		0.5	0/20	20/20	0.939	0.992	479.1
	1.0	3/20	20/20	0.985	1.000	72.4		1.0	0/20	20/20	0.935	0.983	278.3
ring-5-25-20-000	0.0	0/1	0/1	N/A	N/A	2500	cmplt-7-35-20-000	0.0	0/1	0/1	N/A	N/A	3500
	0.3	1/20	19/20	0.966	1.000	832.6		0.3	0/20	20/20	0.929	0.977	559.3
	0.5	1/20	20/20	0.966	1.000	478.9		0.5	0/20	20/20	0.919	0.984	298.8
	1.0	1/20	20/20	0.957	1.000	362.9		1.0	0/20	20/20	0.865	0.928	151.4
ring-5-25-20-001	0.0	0/1	0/1	N/A	N/A	2500	cmplt-7-35-20-001	0.0	0/1	0/1	N/A	N/A	3500
	0.3	5/20	20/20	0.970	1.000	373.6		0.3	0/20	20/20	0.938	0.987	488.4
	0.5	2/20	20/20	0.968	1.000	176.8		0.5	0/20	20/20	0.911	0.970	288.1
	1.0	1/20	20/20	0.939	1.000	161.6		1.0	0/20	20/20	0.883	0.960	179.9
cmplt-5-25-20-000	0.0	0/1	0/1	N/A	N/A	2500	rndm3-7-35-20-001	0.0	0/1	0/1	N/A	N/A	3500
	0.3	0/20	20/20	0.934	0.980	423.9		0.3	1/20	18/20	0.971	1.000	1213.0
	0.5	0/20	20/20	0.923	0.980	208.6		0.5	0/20	18/20	0.962	0.996	735.2
	1.0	0/20	20/20	0.881	0.959	182.5		1.0	1/20	18/20	0.951	1.000	966.3
cmplt-5-25-20-001	0.0	0/1	0/1	N/A	N/A	2500	rndm3-7-35-20-002	0.0	0/1	0/1	N/A	N/A	3500
	0.3	1/20	20/20	0.954	1.000	245.7		0.3	3/20	16/20	0.967	1.000	1507.0
	0.5	0/20	20/20	0.944	0.981	121.3		0.5	0/20	20/20	0.939	0.996	735.2
	1.0	0/20	20/20	0.908	0.953	111.4		1.0	1/20	20/20	0.916	1.000	507.9

しかしながら，$\delta \neq 0.0$ の場合，本プロトコルが最適解に収束するという保証はない．実際，今回の実験では，本プロトコルは一部の試行を除いて最適解ではない実行可能解に収束している．しかし，どの例題においても比較的質の高い実行可能解（$\delta = 0.3$ のとき最適値に対して平均 92.9%以上/最良 97.7%以上，$\delta = 0.5$ のとき平均 91.1%以上/最良 97.0%以上，$\delta = 1.0$ のとき平均 86.5%以上/最良 92.8%以上の質をもつ実行可能解）に収束しており，δ の値を適切に設定することにより低コストで高品質の解が得られることが分かる．

また，多くの場合，δ の値を大きくすると，得られる実行可能解の質が平均的には低下するが，同時に，そのような実行可能解を得るための平均コストは小さくなるという定性的な傾向が見られる．この結果は，δ の値により，得られる実行可能解の質とそれを得るためのコストのバランスをある程度調整できることを示唆しており，興味深い結果といえる．

5. むすび

本論文では，一般化割当問題を拡張し，複数のエージェントが相互にジョブを適切に割り当てることを目指す一般化相互割当問題を導入した．また，それを解く分散型の解法である分散ラグランジュ緩和プロトコ

ルを提案し，その性質を実験により明らかにした．

本論文では，問題に対する実行可能解を求めることのみを目指したが，最大化問題を解く場合には，問題に対する最小の上界値が一般に非常に有益な情報となる．そこで今後は，1対1メッセージ通信モデルのもとでそのような上界値を効率良く求める方法を検討していきたい．

文　献

[1] I.P. Androulakis and G.V. Reklaitis, "Approaches to asynchronous decentralized decision making," Computers and Chemical Engineering, vol.23, pp.341–355, 1999.

[2] K. Hirayama and M. Yokoo, "The distributed breakout algorithms," Artif. Intell., vol.161, no.1-2, pp.89–115, 2005.

[3] S. Martello, D. Pisinger, and P. Toth, "New trends in exact algorithms for the 0-1 knapsack problem," European J. Oper. Res., vol.123, pp.325–332, 2000.

[4] 西　竜志，"サプライチェーンにおける分散協調型最適化技術，" 人工知能誌，vol.19, no.5, pp.571–578, 2004.

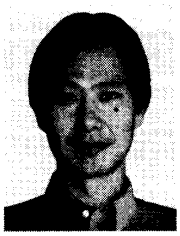
[5] I.H. Osman, "Heuristics for the generalised assignment problem: Simulated annealing and tabu search approaches," OR Spektrum, vol.17, pp.211–225, 1995.

[6] C.R. Reeves（編），横山隆一，奈良宏一，佐藤晴夫，鈴木昭男，荻本和彦，陳　洛南（訳），モダンヒューリスティックス—組合せ最適化の先端手法，日刊工業新聞社，1997.

[7] R.G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," IEEE Trans. Comput., vol.29, no.2, pp.1104–1113, 1990.

[8] M. Yokoo, E.H. Durfee, T. Ishida, and K. Kuwabara, "The distributed constraint satisfaction problem: Formalization and algorithms," IEEE Trans. Knowl. Data Eng., vol.10, no.5, pp.673–685, 1998.

（平成16年11月19日受付）



平山　勝敏　（正員）

1990阪大・基礎工卒．1992同大大学院基礎工学研究科博士前期課程了．1995同大学院基礎工学研究科博士後期課程了．1995神戸商船大学商船学部助手．1997神戸商船大学商船学部講師．1999～2000カーネギーメロン大学ロボティクス研究所客員研究員（文部省在外研究員）．2001神戸商船大学商船学部助教授．2003神戸大学海事科学部助教授（神戸大学と神戸商船大学の統合による）．主にマルチエージェントシステム，及び，制約充足問題に関する研究に従事．情報処理学会，人工知能学会，日本ソフトウエア科学会，日本オペレーションズリサーチ学会，AAAI各会員．