情報処理学会論文誌 Vol.53 No.11 2370–2378 (Nov. 2012)

過制約な一般化相互割当問題に対する分散ラグランジュ緩和プロトコル

花田 研太[1,a)] 平山 勝敏[1,b)]

受付日 2012年1月27日, 採録日 2012年7月2日

概要：一般化相互割当問題（GMAP）において，エージェントの資源容量が著しく少ない過制約な問題を解く手法を 2 つ提案する．1 つは，割り当てられない財をすべて引き受ける disposal エージェントを新たに加え，標準の GMAP として解く手法，もう 1 つは割当制約を不等式として記述し問題を解く手法である．エージェントの資源容量を加工した一般化割当問題（GAP）のベンチマーク問題例を上記両手法で解き，得られたデータを比較，考察した．その結果，前者の方法は過制約度の低い問題例に対して，後者の方法は過制約度の高い問題例に対して有効であることが分かった．

キーワード：一般化相互割当問題，分散最適化，ラグランジュ緩和

Distributed Lagrangian Relaxation Protocol for the Over-constrained Generalized Mutual Assignment Problem

Hanada Kenta[1,a)] Hirayama Katsutoshi[1,b)]

Received: January 27, 2012, Accepted: July 2, 2012

Abstract: *The Generalized Mutual Assignment Problem* (GMAP) is a distributed combinatorial optimization problem in which, with no centralized control, multiple agents search for an optimal assignment of goods that satisfies their individual knapsack constraints. Previously, in the GMAP protocol, problem instances were assumed to be feasible, meaning that the total capacities of the agents were large enough to assign the goods. However, this assumption may not be realistic in some situations. In this paper, we present two methods for dealing with such "over-constrained" GMAP instances. First, we introduce a *disposal agent* who has an unlimited capacity and is in charge of the unassigned goods. With this method, we can use any off-the-shelf GMAP protocol since the disposal agent can make the instances feasible. Second, we formulate the GMAP instances as an Integer Programming (IP) problem, in which the assignment constraints are described with inequalities. With this method, we need to devise a new protocol for such a formulation. We experimentally compared these two methods on the variants of *Generalized Assignment Problem* (GAP) benchmark instances. Our results indicate that the first method finds a solution faster for fewer over-constrained instances, and the second finds a better solution faster for more over-constrained instances.

Keywords: generalized mutual assignment problem, distributed optimization, Lagrangian relaxation

1. 序論

近年，マルチエージェントシステムの分野において，分散割当ては主要な問題の 1 つとなっている．分散割当てとは，仕事（ジョブ）もしくは財を分散環境下でエージェントに割り当てることを指す．分散割当ての特徴は，割当てを行うエージェントが複数存在し，互いに独立していることにある．集中型の割当てでは，割当てを行う特別なエージェント（コーディネータ）が存在し，すべての仕事や財はコーディネータが一括して管理する．しかし分散割当てでは，仕事や財は複数のエージェントが分割して管理し，財の割当てはエージェント間で行われる．

この分野で最も古い例として，契約ネットプロトコル [1]

[1] 神戸大学海事科学研究科
Kobe University, Kobe, Hyogo 658–0022, Japan
a) kenta_hanada@stu.kobe-u.ac.jp
b) hirayama@maritime.kobe-u.ac.jp

があげられる．しかし，契約ネットプロトコルは欲張り型探索を行うため，一般に割当結果の最適性については何も保証しない．また最近では，マルチロボットタスク割当て [2] や分散ターゲット追跡 [3] といった問題が注目されており，問題に特化した様々な解法が提案されているが，解法間の比較を容易にするためにも問題を明確に定式化して汎用解法を適用するアプローチに期待する声は大きい．より一般化された問題としては，分散制約最適化問題 [4]，分散施設配置問題 [5] などがある．しかし，分散制約最適化問題では，その基本的な定式化において割当問題にとって本質的な資源制約が自然な形で導入されていないという問題がある．また，分散施設配置問題は，開設すべき施設およびその接続先を同時に決める最適化問題であり，開設コストを考慮できる点が重要である．しかし，局所問題が比較的疎な問題になりがちなため，効率的な分散型解法を設計することがやや困難だと予想される．

一方，分散環境において，資源制約を陽に扱うことができ，汎用的かつ明確な定式化を持つ組合せ最適化問題として一般化相互割当問題（GMAP: Generalized Mutual Assignment Problem），および，それを解く分散型の解法として分散ラグランジュ緩和プロトコル（DisLRP: Distributed Lagrangian Relaxation Protocol）[6], [7], [8] が提案されている．

GMAP は，財を持つ複数のエージェントが，それぞれの財を系内で相互に割り当てる—すなわち交換する—際，各エージェントの資源制約を満たしつつ系全体の効用和が最大となる割当てを求める分散最大化問題である．この問題は，財を割り当てられる側に選択の決定権があると考えれば，系全体の財集合に対し各エージェントの資源制約を満たす最適な分割を求める，いわゆる資源制約付きの集合分割問題と見なすことができる．

一方，DisLRP は，エージェント間の局所的な通信のみを利用して GMAP を解くプロトコルである．DisLRP では，ラグランジュ分解というテクニックを用いて，GMAP をエージェントごとに分割された部分問題の集合と見なす．この部分問題は，エージェントが自分に関係する財の集合から資源制約を満たす最適な部分集合を求める 0-1 ナップサック問題であり，そのときの各財の利得は，そのエージェントにとっての財の効用値からその時点での財の価格（ラグランジュ乗数の値）を引いたものとなる．

DisLRP におけるエージェントの振舞いの概略は次のとおりである．まず，財のある価格のもとで，エージェントはそれぞれの 0-1 ナップサック問題を並行して解き，その結果，どの財を選択したかを関連するエージェントに知らせる．次に，その暫定的な結果に基づいて，2 つ以上のエージェントに選択されている財についてはその価格を上げ，一方，誰にも選択されていない財についてはその価格を下げる．このように更新した新しい価格のもとで，エージェントはこの新しい 0-1 ナップサック問題を並行して解く．以下，すべての財が 1 エージェントのみに選択されるようになる—すなわち，系全体の財集合に対する正しい集合分割が得られる—まで同様の処理を繰り返す．

従来の GMAP では，系全体の財集合に対して，エージェントは十分な資源を持つと暗黙のうちに仮定していた．しかし，現実には，系全体の財集合に対して，エージェントが十分な資源を持たないような，いわゆる過制約な状況もありうると考えられる．本研究では，このような過制約な状況に対処する新たな DisLRP を 2 つ提案し，ベンチマーク問題例を用いた実験によりそれらを比較評価する．

第 1 の方法の基本的なアイデアは，資源制約のない（無限の資源を持つ）仮想的なエージェント（disposal エージェント）を導入し，通常のエージェントに割り当てることができずにあぶれてしまった財は disposal エージェントに割り当てるというものである．この方法では，GMAP の従来の定式化を基本的に変更しなくて済むため，既存の DisLRP をそのまま利用することができるというメリットがある．一方，disposal エージェントを系に加えて，それに計算をさせ，さらに，通常エージェントと通信させる必要があるため，追加のコスト負担があるというデメリットがある．しかし，disposal エージェントの挙動は非常に単純なので，他のすべてのエージェントが disposal エージェントの挙動を代替する—すなわち disposal エージェントを仮想化する—ことが可能である．仮想化することによってエージェント数を元の問題と同じにすることができ，追加のコスト負担を避けることができる．

一方，第 2 の方法は，GMAP の割当制約である「各財はどれか 1 エージェントに必ず割り当てられなければならない」を緩和し，「各財は，1 エージェントに割り当てられるか，あるいは，誰にも割り当てられなくてもよい」とするものである．この方法は，disposal エージェントのような特別なエージェントを導入しなくて済む反面，従来の定式化を変更することにともない DisLRP の手続きを一部変更する必要がある．技術的には，ラグランジュ双対問題が制約付きの問題となるためそれに合わせて双対空間の探索手続きを変更する必要があること，また，得られた解が最適解か否かをチェックするための条件を追加する必要があることの 2 点があげられる．

以下，本論文は次のように構成される．2 章では GMAP の定義，および DisLRP で用いるラグランジュ双対問題について述べ，3 章では前述の 2 つの方法を導くための背景となる理論の概要を述べる．続く 4 章では，ベンチマーク問題例を用いて両手法を実験的に比較評価し，5 章で本論文のまとめと今後の課題を示す．

2. 一般化相互割当問題（GMAP）とそのラグランジュ双対問題

GMAP [6] は，財を持つ複数のエージェントが，それぞれの財を系内で相互に割り当てる—すなわち交換する—際，各エージェントの資源制約を満たしつつ系全体の効用和が最大となる割当てを求める分散最大化問題である．系全体を見た場合，GMAP におけるエージェントは次のような整数計画問題 $\mathcal{GAP}$ を解く．

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

ここで，$A = \{1,\ldots,m\}$ はエージェントの集合，$J = \{1,\ldots,n\}$ は財の集合，p_{kj} はエージェント k が財 j を得た場合の効用，w_{kj} はエージェント k が財 j を得た場合の資源消費量，c_k はエージェント k の資源容量である．また，x_{kj} は，エージェント k が財 j を得た場合は 1，そうでない場合は 0 に設定される決定変数である．問題の目的は，(1) 各財がただ 1 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），かつ，(3) どの財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（0-1 制約）という制約条件のもとで効用の総和を最大化することである．なお，この問題は NP 困難な問題に属する．

割当制約 (1) を緩和したラグランジュ緩和問題は次のようになる．

$$
\begin{aligned}
L(\mu) = \text{max.} \ & \sum_{k\in A}\sum_{j\in J} p_{kj}x_{kj} + \sum_{j\in J}\mu_j\left(1 - \sum_{k\in A} x_{kj}\right)\\
\text{s. t.} \ & \sum_{j\in J} w_{kj}x_{kj} \le c_k,\ \forall k \in A,\\
& x_{kj} \in \{0,1\},\ \forall k \in A,\ \forall j \in J.
\end{aligned}
$$

この問題の最適値および最適解は，財 j に対する実数値パラメータ（ラグランジュ乗数）μ_j の値に依存する．ここでは，$\mu = (\mu_1,\ldots,\mu_n)$ とし，ラグランジュ乗数ベクトル μ に対する特定の値のもとでの上のラグランジュ緩和問題の最適値を $L(\mu)$ と表記した．なお，μ に対する任意の値のもとで，$L(\mu)$ は原問題である $\mathcal{GAP}$ の最適値の上界となるため，その上界値を最小化する μ を求めるラグランジュ双対問題は

$$\text{min.} \quad L(\mu),$$

という n 次元実ベクトル空間上の無制約最小化問題となる．

一方，$L(\mu)$ を与える最大化問題は，通常のエージェント k に関するナップサック問題

$$
\begin{aligned}
L_k(\mu) = \text{max.} \ & \sum_{j\in J}(p_{kj} - \mu_j)x_{kj}\\
\text{s. t.} \ & \sum_{j\in J} w_{kj}x_{kj} \le c_k,\\
& x_{kj} \in \{0,1\},\ \forall j \in J,
\end{aligned}
$$

および，ラグランジュ乗数のみからなる項

$$L_{const}(\mu) = \sum_{j\in J}\mu_j,$$

に分解することができる．したがって，先のラグランジュ双対問題は，

$$\text{min.} \quad \sum_{k\in A} L_k(\mu) + L_{const}(\mu),$$

となる．

3. 過制約な問題の定式化と解法

GMAP を解く従来のプロトコルでは，問題が実行可能である，すなわち，割当制約とナップサック制約をすべて満たすような決定変数への 0 または 1 の値の割当てが存在すると仮定されていた．しかし，現実には，系全体の財集合に対して，エージェントの資源が十分ではなく，すべての割当制約とナップサック制約を満たすことができない，いわゆる過制約な状況もありうると考えられる．本研究では，このような過制約な状況に対処する方法を 2 つ提案する．

3.1 disposal エージェントを用いた DisLRP

第 1 の方法の基本的なアイデアは，資源制約のない（無限の資源を持つ）仮想的なエージェント（disposal エージェント）を導入し，通常のエージェントに割り当てることができずにあぶれてしまった財は disposal エージェントに割り当てるというものである．

3.1.1 定式化

定式化にあたって，disposal エージェントを次のように考慮する．

- disposal エージェントが財を得たとしても効用はゼロである．
- 各財は，通常のエージェントと disposal エージェントのうち誰か 1 人に必ず割り当てられる．
- disposal エージェントにはナップサック制約がない．

以上をふまえて，disposal エージェントの識別子を $d \notin A$ とすれば，系全体の問題は，

$$
\begin{aligned}
&\mathcal{GAP}' \quad (\text{decide } x_{kj},\ \forall k \in A\cup\{d\},\ \forall j \in J):\\
&\text{max.} \quad \sum_{k\in A}\sum_{j\in J} p_{kj}x_{kj}
\end{aligned}
$$

$$\text{s. t.} \sum_{k\in A\cup\{d\}} x_{kj} = 1,\ \forall j \in J, \tag{4}$$
$$\sum_{j\in J} w_{kj}x_{kj} \le c_k,\ \forall k \in A,$$
$$x_{kj} \in \{0,1\},\ \forall k \in A\cup\{d\},\ \forall j \in J,$$

と記述できる．前節の $\mathcal{GAP}$ との違いは，すべての割当制約に対して disposal エージェントの決定変数 x_{dj} が追加されたことである．

この問題 $\mathcal{GAP}'$ に対し，割当制約 (4) を緩和したラグランジュ緩和問題は次のようになる．

$$L'(\mu) = \max. \sum_{k\in A}\sum_{j\in J} p_{kj}x_{kj} + \sum_{j\in J}\mu_j\left(1 - \sum_{k\in A\cup\{d\}} x_{kj}\right)$$
$$\text{s. t.} \sum_{j\in J} w_{kj}x_{kj} \le c_k,\ \forall k \in A,$$
$$x_{kj} \in \{0,1\},\ \forall k \in A\cup\{d\},\ \forall j \in J.$$

2 章のときと同様，μ に対する任意の値のもとで，$L'(\mu)$ は原問題である $\mathcal{GAP}'$ の最適値の上界となるため，その上界値を最小化する μ を求めるラグランジュ双対問題は

$$\min.\ \ L'(\mu),$$

という n 次元実ベクトル空間上の無制約最小化問題となる．

一方，$L'(\mu)$ を与える最大化問題は，通常のエージェント k に関するナップサック問題

$$L'_k(\mu) = \max. \sum_{j\in J}(p_{kj} - \mu_j)x_{kj}$$
$$\text{s. t.} \sum_{j\in J} w_{kj}x_{kj} \le c_k,$$
$$x_{kj} \in \{0,1\},\ \forall j \in J,$$

disposal エージェントに関する最大化問題

$$L'_d(\mu) = \max. \sum_{j\in J}(-\mu_j)x_{dj} \tag{5}$$
$$\text{s. t. } x_{dj} \in \{0,1\},\ \forall j \in J, \tag{6}$$

および，ラグランジュ乗数のみからなる項

$$L'_{const}(\mu) = \sum_{j\in J}\mu_j,$$

に分解することができる．したがって，先のラグランジュ双対問題は，

$$\min.\ \ \sum_{k\in A} L'_k(\mu) + L'_d(\mu) + L'_{const}(\mu),$$

となる．2 章との違いは，$\mathcal{GAP}'$ を各エージェントに分解するとき，disposal エージェントに関する最大化問題 $L'_d(\mu)$ が追加されている点にある．提案する解法では，disposal エージェントを含む各エージェントがこの（分解された）ラグランジュ双対問題を局所通信のみを用いて解く．

3.1.2 disposal エージェントの仮想化

disposal エージェントを用いた DisLRP では，disposal エージェントという特殊なエージェントを追加するため，エージェント数が元の問題より 1 つ増加する．これにより通信コストが増大するデメリットが考えられる．しかし，disposal エージェントの目的関数 (5) および制約条件 (6) によれば，$-\mu_j \ge 0$，すなわち $\mu_j \le 0$ のときに disposal エージェントが財 j を選ぶことは明らかである．そこで，disposal エージェントの振舞いを通常エージェントが代替することによって disposal エージェントを仮想化し，エージェント数を元の問題と同一にすることで，通信コストの増大というデメリットを回避することができる．また，新たに通常エージェントが負担する計算量は $O(|J|)$ であり，多項式時間で終了する．これはナップサック問題の計算量に比べてはるかに小さいので，ほとんど無視してよい．

具体的な仮想化の方法については，次に述べる．

3.1.3 解法

本解法の基本的な手続きは以下のとおりである．

(Step 1) 全エージェントがラグランジュ乗数ベクトル μ の値を 0 で初期化する．

(Step 2) 現在の μ の値のもとで，disposal エージェント以外の各エージェント $k \in A$ はそれぞれのナップサック問題を解いて $L'_k(\mu)$ を求める．それぞれの問題を解いて得た結果を，原問題 $\mathcal{GAP}'$ における割当制約を通じて関連するエージェントに送信する．このとき，ラグランジュ乗数 μ_j が負の財は disposal エージェントに選ばれているものとして全エージェントが扱う．

(Step 3) 原問題の割当制約がすべて満たされていれば最適解が得られているため終了する．

(Step 4) 上界値と下界値を求める．また，これまでに得られた最小の上界値 *BestUB* と最大の下界値 *BestLB* を求め，両者が一致すれば最適解が得られているため終了する．

(Step 5) 各財 j に対するラグランジュ乗数 μ_j を劣勾配法 [9], [10] を用いて更新し，Step 2 へ戻る．

この手続きでは，エージェントは Step 1 で初期化したのち Step 2 から Step 5 の手順を繰り返す．以下，この 1 回の繰返しをラウンドと呼び，処理の 1 単位と見なす．

Step 2 で，前節で述べた disposal エージェントの仮想化を行う．disposal エージェントが財を選ぶかどうかの判定は，全エージェントで共通の値を持つラグランジュ乗数ベクトル μ を用いているので，disposal エージェントの解がエージェントによって異なるということは起こらない．

また，Step 3 から Step 5 の計算には本来，*BestUB* および *BestLB* という系全体の大域情報が必要である．そのためには，系全体をモニタする特別なエージェントを利用することが簡単だが，そのようなエージェントは処理のボトルネックになりうることから，本研究では，文献 [8] の

方法に従い，生成木を用いてそれぞれのエージェントが必要な大域情報を収集するものとする．なお，大域情報を収集するために発生するメッセージ数は，1 ラウンドあたり $O(|A|)$ である．

Step 4 の上界値は $L'(\mu)$ であり，Step 2 で各エージェントが求めた最適値の和より計算できる．一方，下界値は，Step 2 で各エージェントが求めた最適解から原問題 $\mathcal{GAP}'$ の実行可能解を構成し，その目的関数値を計算して求める．なお，実行可能解の構成方法は次のとおりである．すなわち，Step 2 で各エージェントが求めた最適解において，

- 1 エージェントにのみ選択されている財は，そのエージェントへ割り当てる．
- 2 エージェント以上に選択されている財は，その財を選択したエージェントのうち最も効用の大きいエージェントへ割り当てる．
- どのエージェントにも選択されていない財は，disposal エージェントへ割り当てる．

Step 5 において，ラグランジュ乗数 μ_j を更新する際には劣勾配法を用いる．劣勾配法では，エージェントがラウンド t における各財 j の劣勾配

$$g_j^{(t)} \leftarrow 1 - \sum_{i \in A \cup \{d\}} x_{ij}$$

を求め，次の更新規則により，ラグランジュ乗数 μ_j を更新する．

$$\mu_j^{(t+1)} \leftarrow \mu_j^{(t)} - \frac{\pi^{(t)}(BestUB^{(t)} - BestLB^{(t)})g_j^{(t)}}{\sum_{j \in J}(g_j^{(t)})^2}.$$

π は制御パラメータであり，初期値は 2 で，$BestUB$ と $BestLB$ の両方が 30 ラウンド連続して更新されなければ半減される．

3.2 不等式制約緩和に基づく DisLRP

もう 1 つの方法は，GMAP の割当制約である「各財はどれか 1 エージェントに必ず割り当てられなければならない」を緩和し，「各財は，1 エージェントに割り当てられるか，あるいは，誰にも割り当てられなくてもよい」とするものである．

3.2.1 定式化

系全体の問題は次のような整数計画問題となる．

$$\begin{aligned}
&\mathcal{GAP}'' \quad (\text{decide } x_{kj},\ \forall k \in A,\ \forall j \in J):\\
&\text{max.} \quad \sum_{k \in A}\sum_{j \in J} p_{kj}x_{kj} && (7)\\
&\text{s. t.} \quad \sum_{k \in A} x_{kj} \le 1,\ \ \forall j \in J, && (8)\\
&\qquad \sum_{j \in J} w_{kj}x_{kj} \le c_k,\ \ \forall k \in A, && (9)\\
&\qquad x_{kj} \in \{0,1\},\ \forall k \in A,\ \forall j \in J. && (10)
\end{aligned}$$

2 章の $\mathcal{GAP}$ との違いは，$\mathcal{GAP}$ では割当制約が等式制約となるが，$\mathcal{GAP}''$ ではそれが不等式制約となる点である．

次に，$\mathcal{GAP}''$ に対して割当制約 (8) を緩和すると

$$\begin{aligned}
L''(\mu) = \text{max.} & \sum_{k \in A}\sum_{j \in J} p_{kj}x_{kj} + \sum_{j \in J}\mu_j\left(1 - \sum_{k \in A} x_{kj}\right)\\
\text{s. t.} & \sum_{j \in J} w_{kj}x_{kj} \le c_k,\ \forall k \in A,\\
& x_{kj} \in \{0,1\},\ \forall k \in A,\ \forall j \in J,\\
& \mu_j \ge 0,\ \forall j \in J,
\end{aligned}$$

というラグランジュ緩和問題が得られる．なお，2 章における $\mathcal{GAP}$ や前節における $\mathcal{GAP}'$ では，各財 j の割当制約は等式であるので，ラグランジュ乗数 μ_j は任意の実数をとることができるが，$\mathcal{GAP}''$ では，各財 j の割当制約は不等式なのでそれに対応するラグランジュ乗数 μ_j には非負制約がともなう．前節と同様に，$L''(\mu)$ は原問題 $\mathcal{GAP}''$ の最適値の上界を与えるため，その上界値を最小化する μ を求めるラグランジュ双対問題は

$$\text{min.} \quad L''(\mu) \qquad \text{s. t.} \quad \mu_j \ge 0,\ \forall j \in J,$$

という n 次元非負実ベクトル空間上の最小化問題となる．

一方，$L''(\mu)$ を与える最大化問題は，各エージェント k のナップサック問題

$$\begin{aligned}
L''_k(\mu) = \text{max.} & \sum_{j \in J}(p_{kj} - \mu_j)x_{kj}\\
\text{s. t.} & \sum_{j \in J} w_{kj}x_{kj} \le c_k,\\
& x_{kj} \in \{0,1\},\ \forall j \in J,
\end{aligned}$$

および，ラグランジュ乗数のみからなる項

$$L''_{const}(\mu) = \sum_{j \in J}\mu_j,$$

に分解することができる．したがって，ラグランジュ双対問題は，

$$\begin{aligned}
\text{min.} \quad & \sum_{k \in A} L''_k(\mu) + L''_{const}(\mu)\\
\text{s. t.} \quad & \mu_j \ge 0,\ \forall j \in J,
\end{aligned}$$

となる．提案する解法では，各エージェントがこの（分解された）ラグランジュ双対問題を局所通信のみを用いて解く．

3.2.2 解法

本解法の基本手続きは以下のとおりである．

(Step 1) 全エージェントがラグランジュ乗数ベクトル μ の値を 0 で初期化する．

(Step 2) 現在の μ の値のもとで，各エージェント k はそれぞれのナップサック問題を解いて $L''_k(\mu)$ を求める．それぞれの問題を解いて得た結果を，原問題 $\mathcal{GAP}''$ における割当制約を通じて関連するエージェントに送信

する．

(Step 3) 原問題の割当制約をすべて満たし，かつ，

$$\mu_j(1-\sum_{k\in A} x_{kj})=0,\ \forall j\in J \tag{11}$$

が成立すれば最適解が得られているため終了する．

(Step 4) 上界値と下界値を求める．また，これまでに得られた最小の上界値 $BestUB$ と最大の下界値 $BestLB$ を求め，両者が一致すれば最適解が得られているため終了する．

(Step 5) 各財 j に対するラグランジュ乗数 μ_j を劣勾配法 [9], [10] を用いて更新し，Step 2 へ戻る．なお，更新の際，$\mu_j \geq 0$ をつねに満たすようにする．

以上，本解法は，前節の解法と基本的な流れは同じだが，いくつか特筆すべき相違がある．

まず，Step 3 の終了条件が，前節の解法に比べてより厳しくなっている．式 (11) の左辺は，ラグランジュ緩和したときに目的関数へ追加した項を表している．式 (11) を満たしていなければ，上界値と下界値が一致せず，最適解である保証ができない．disposal エージェントを用いた DisLRP では，割当制約を満たせば式 (11) は必ず 0 になるので，終了条件に式 (11) を加える必要がない．しかし，不等式制約緩和に基づく DisLRP は，割当制約を不等式で満たしている制約において μ_j が 0 でない場合があり，その場合式 (11) を満たすことができず，最適性を保証できない．よって，不等式制約緩和に基づく DisLRP は，終了条件に式 (11) を追加する必要がある．

また，Step 4 の下界値計算で実行可能解を構成する際に，Step 2 で求めた最適解においてどのエージェントにも選択されていない財は単に無視すればよい．さらに，Step 5 において，ラグランジュ乗数の値が負にならないよう，劣勾配法におけるラグランジュ乗数の更新規則を変更する必要がある．よって本解法では更新規則を以下で置き換える．

$$g_j^{(t)} \leftarrow 1-\sum_{i\in A} x_{ij},$$

$$temp \leftarrow \mu_j^{(t)} - \frac{\pi^{(t)}(BestUB^{(t)}-BestLB^{(t)})g_j^{(t)}}{\sum_{j\in J}(g_j^{(t)})^2},$$

$$\mu_j^{(t+1)} \leftarrow \max\{temp, 0\}.$$

すなわち，従来の更新規則でラグランジュ乗数の値が負になる場合，強制的に 0 に置き換える．

4. 実験

3 章で述べた 2 つの解法を実装して実験を行った．過制約な一般化相互割当問題は本論文で初めて扱う問題であり比較対象が存在しないため，実験の目的は，$BestLB/BestUB$ で定義される解の質がより早く 1 に収束するのはどちらの解法かを比較することとした．

実験では過制約な問題例を与える必要があるため，OR-Library [11] に掲載されている一般化割当問題（GAP: Generalized Assignment Problem）のベンチマーク問題例 60 題を加工して使用した．具体的には，各エージェントの資源容量 c_k を減らすため c_k に係数 $x \in \{0.1, 0.2, \ldots, 0.9\}$ をかけて小数点以下を切り捨てた数値を求め，1 つのベンチマーク問題例に対して 9 個の問題例を新たに作成した．すなわち，全部で 540 個の問題例を作成した．以後，この係数 x を容量係数と呼ぶ．なお，容量係数はエージェントごとに設定することも可能だが，過制約度を 1 つのパラメータとして表現するために，全エージェントの容量係数を同じ値に設定した．

各エージェントのナップサック問題を解くソルバには，LP-Solve [12] を用いた．LP-Solve は C 言語で書かれた線形計画問題および混合整数計画問題を解くためのフリーソフトである．今回，2 つの解法を実装するにあたって JAVA を使用し，LP-Solve も JAVA の API を使用した．また，実験環境には，デスクトップ PC（Core i7-870 2.93 GHz, 4 コア 8 スレッド，メモリ 8 GB，Windows 7 Professional）を用いた．

表 1 および**表 2** に，容量係数ごとの解の質とラウンド数

表 1 容量係数ごとの解の質の平均および中央値
Table 1 Average and Median of $BestLB/BestLB$ for each capacity coefficient.

	Average		Median	
x	Disposal	Inequality	Disposal	Inequality
0.1	0.9996	1.0000	1.0000	1.0000
0.2	0.9998	0.9999	1.0000	1.0000
0.3	0.9992	0.9993	1.0000	1.0000
0.4	0.9993	0.9992	1.0000	1.0000
0.5	0.9935	0.9943	0.9993	1.0000
0.6	0.9919	0.9922	1.0000	1.0000
0.7	0.9886	0.9896	0.9913	0.9900
0.8	0.9878	0.9850	0.9913	0.9870
0.9	0.9882	0.9834	0.9919	0.9838

表 2 容量係数ごとのラウンド数の平均および中央値
Table 2 Average and Median of Round for each capacity coefficient.

	Average		Median	
x	Disposal	Inequality	Disposal	Inequality
0.1	199.1833	27.9333	1	1
0.2	1,291.3833	613.2000	34	5
0.3	2,543.7167	1,254.6333	117	13
0.4	2,344.9833	1,942.4500	259	176
0.5	5,685.4000	4,599.9000	10,000	1,423
0.6	5,277.1667	5,256.5500	5,935	6,006
0.7	7,873.1833	8,096.9833	10,000	10,000
0.8	8,084.8667	9,673.7833	10,000	10,000
0.9	7,609.7119	10,000.0000	10,000	10,000

の平均と中央値をそれぞれ示す．なお，カットオフラウンドは 10,000 ラウンドとし，これを超えても最適値が得られない場合は，そのときの解の質とラウンド数をデータとして用いている．表 1 より，平均値，中央値ともに容量係数が 0.1 から 0.4 まではほぼ同じであるが，0.5 以降にわずかな差が見られる．表 2 より，容量係数が 0.1 から 0.5 までは disposal エージェントを用いた DisLRP の方が，不等式制約緩和に基づく DisLRP よりも平均値が高く，0.6 以降は大小関係が逆転していることが分かる．また，中央値から，0.5 までは disposal エージェントを用いた DisLRP の方が数値が高く，0.6 以降はほぼ同じであることが分かる．

表 1 および表 2 から以上のことが読み取れるが，個々の問題例を比較すると数値のばらつきが非常に大きかった．また，解の質において，10^{-2} オーダまで数値がほぼ一緒なので，有意な差があるかどうかは一見して分かりにくい．よって，実験結果をより詳細に分析する必要があると判断し，解の質とラウンド数に対してウィルコクソン符号付き順位和検定 [13] を実施した．これは，2 つの母集団の中央値に有意な差があるかどうかを検定するノンパラメトリック手法である．n 組のデータ $(x_1, y_1), \ldots, (x_n, y_n)$ が与えられたとき，対応する 2 変数の差を $d_i = x_i - y_i$ とする．d_i を絶対値の小さい順に並べたときの d_i の順位を R_i とすると，統計検定量 T は，

$$T = \sum_{i=1}^{n} R_i$$

となる．なお，d_i が 0 の場合はデータから取り除き，d_i が 0 でない同じデータがある場合は，同順位（タイ）として扱う．今回扱うデータ量が 50 組以上あるので，検定統計量 T は正規分布 $N(0,1)$ に従うと見なしてよい．よって Z 値は，

$$Z = \frac{T - \dfrac{n(n+1)}{4}}{\sqrt{\dfrac{n(n+1)(2n+1) - \frac{1}{2}\sum_{j=1}^{g}(t_j - 1)t_j(t_j + 1)}{24}}}$$

となる．ここで g は 0 でない $|z_i|$ の中でタイが生じたグループ数，t_j はそのグループ内でタイとなった観測値の個数である．

検定結果を**表 3** に示す．なお，表 3 に示される中央値は，検定対象となったデータに対する値である．これより，解の質に対する仮説は棄却されない，すなわち 2 つの解法による解の質に差があるとはいえないことが分かる．また，ラウンド数に関しては仮説が棄却されるので，2 つの解法による収束の速さに差があるといえる．検定の対象となったデータの中央値とあわせて考えると最終的に，両解法による解の質に差があるとはいえないが，不等式制約緩和に基づく DisLRP の方が早く収束していることが分かる．

表 1 および表 2 から得られた知見の正当性を検証するため，データを容量係数 0.2〜0.5 と 0.6〜0.9 の 2 つのグループに分け[*1]，先ほどと同様にウィルコクソン符号付き順位和検定を実施した．その結果を**表 4** と**表 5** に示す．表 4 と表 5 に示された中央値も，表 3 と同様，検定対象となったデータに対する値である．

表 4 より，いずれの結果も棄却されるので，2 つのデータ系列に差があるといえる．また表 5 より，2 つの解法による解の質に差があるとはいえないが，収束の早さには差があるといえる．中央値を見ると，容量係数 0.2〜0.5 は解の質，ラウンド数のいずれも不等式制約緩和に基づく DisLRP の方が良く，容量係数 0.6〜0.9 のラウンド数は disposal エージェントを用いた DisLRP の方が良いこと

表 3 全問題例のウィルコクソン符号付き順位和検定結果
Table 3 Wilcoxon signed-rank test for all instances.

	BestLB/*BestUB*	Round
仮説	2 つのデータ系列に差はない	
検定統計量 T	67704.5	15658.5
$\|Z\|$ 値	0.3087	3.2182
結論	有意水準 5%で仮説は棄却されない	有意水準 1%で仮説は棄却される
中央値		
（disposal）	0.9998	285
（不等式緩和制約）	0.9990	174

表 4 容量係数 0.2〜0.5 の検定結果
Table 4 Wilcoxon signed-rank test for instances whose capacity coefficients range from 0.2 to 0.5.

	BestLB/*BestUB*	Round
仮説	2 つのデータ系列に差はない	
検定統計量 T	250	3122
$\|Z\|$ 値	3.17479	7.3127
結論	有意水準 1%で仮説は棄却される	有意水準 1%で仮説は棄却される
中央値		
（disposal）	0.9980	171.5
（不等式緩和制約）	1.0000	64

表 5 容量係数 0.6〜0.9 の検定結果
Table 5 Wilcoxon signed-rank test for instances whose capacity coefficients range from 0.6 to 0.9.

	BestLB/*BestUB*	Round
仮説	2 つのデータ系列に差はない	
検定統計量 T	2937	980
$\|Z\|$ 値	1.9490	2.6479
結論	有意水準 5%で仮説は棄却されない	有意水準 1%で仮説は棄却される
中央値		
（disposal）	0.9859	455
（不等式緩和制約）	0.9831	1220

[*1] 容量係数 0.1 の場合は最適値 0 の問題例が多かったので除外した．

が分かる．よって disposal エージェントを用いた DisLRP は，容量係数の大きい過制約度が比較的低い問題例に対して，不等式制約緩和に基づく DisLRP は容量係数の小さい過制約度が比較的高い問題例に対して早く最適値が求められるといえる．

不等式制約緩和に基づく DisLRP が過制約度の高い問題例に対して有効である理由は，以下のように考えられる．まず，割当制約を等式で満たすことが非常に困難なため，すべてのエージェントに選ばれない財が多いと思われる．そのような財におけるラグランジュ乗数 μ_j の値は，劣勾配法で更新しても，非負条件により 0 となる場合が多い．よって，不等式制約緩和の終了条件 (11) を容易に満たすことができるため，早く収束すると考えられる．逆に，過制約度の低い問題例ではエージェントが選ぶ財の重複が増え，終了条件 (11) を満たすことがより困難になるため，disposal エージェントを用いた DisLRP の方が良くなるものと考えられる．

5. 結論と今後の課題

過制約な一般化相互割当問題において，disposal エージェントを用いた DisLRP と不等式制約緩和に基づく DisLRP の 2 つの手法を提案した．各手法の性能面での特徴をまとめると次のようになる．

- disposal エージェントを用いた DisLRP
  - 各エージェントの 1 ラウンドあたりの送信メッセージ数：$O(|A|)$
  - 各エージェントの 1 ラウンドあたりの計算量：ナップサック問題の計算量，および，disposal エージェントの仮想化のための計算量 $O(|J|)$ の総和
  - 最適性と収束性：過制約度が比較的低い問題例に対して，不等式制約緩和に基づく DisLRP と同程度の解により早く収束する．
- 不等式制約緩和に基づく DisLRP
  - 各エージェントの 1 ラウンドあたりの送信メッセージ数：$O(|A|)$
  - 各エージェントの 1 ラウンドあたりの計算量：ナップサック問題の計算量
  - 最適性と収束性：過制約度が比較的高い問題例に対して，disposal エージェントを用いた DisLRP よりも良い解により早く収束する．

ここで留意しなければならないのは，今回提案した手法は過制約な一般化割当問題に対して適用されるということである．もし過制約でない，すなわち通常の一般化割当問題の問題例を今回の提案手法で解くと，既存の手法で解いた最適値および最適解と異なる結果を出力する可能性がある．これは，割当制約を緩和したことによって探索する解空間が広くなり，結果として最適値および最適解が変わる可能性があるからである．今後の課題としては，過制約でない問題例に対しては既存の手法で解いたときと同じ最適値および最適解を出力し，かつ過制約な問題例に対しては本論文で求めた解を出力することができる手法の開発があげられる．

また，今回は選択されなかった財のその後についてまったく考慮していない．通常，選択されなかった財は次の段階で再びエージェントに割り当てられると考えるのが自然だと思われる．このように，一般化相互割当問題を多段階の最適化問題に拡張することも今後の課題の 1 つである．

参考文献

[1] Smith, R.G.: The contract net protocol: High-level communication and control in a distributed problem solver, *IEEE Trans. Computers*, Vol.29, No.2, pp.1104–1113 (1990).

[2] Dias, M.B., Zlot, R., Kalra, N. and Stentz, A.: Market-based multirobot coordination: A survey and analysis, *Proc. IEEE*, Vol.94, No.7, pp.1257–1270 (2006).

[3] Bhatti, S. and Xu, J.: Survey of target tracking protocols using wireless sensor network, *Proc. 5th International Conference on Wireless*, pp.110–115 (2009).

[4] Modi, P.J., Shen, W.-M., Tambe, M. and Yokoo, M.: An asynchronous complete method for distributed constraint, *Proc. 2nd International Joint Conference on Autonomous Agents Multi-Agent Systems* (*AAMAS-2003*), pp.161–168 (2003).

[5] Frank, C. and Römer, K.: Distributed facility location algorithms for flexible configuration, *Proc. International Conference on Distributed Computing in Sensor Systems* (*DCOSS-2007*), pp.124–141 (2007).

[6] Hirayama, K.: A new approach to distributed task assignment using Lagrangian decomposition and distributed constraint satisfaction, *Proc. 21st National Conference on Artificial Intelligence* (*AAAI-2006*), pp.660–665 (2006).

[7] Hirayama, K.: An α-approximation protocol for the generalized mutual assignment problem, *Proc. 22nd AAAI Conference on Artificial Intelligence* (*AAAI-2007*), pp.744–749 (2007).

[8] Hirayama, K., Matsui, T. and Yokoo, M.: Adaptive price update in distributed Lagrangian relaxation protocol, *Proc. 8th International Joint Conference on Autonomous Agents Multi-Agent Systems* (*AAMAS-2009*), pp.1033–1040 (2009).

[9] Reeves, C.R. (Ed.): *Modern heuristic techniques for combinatorial problems*, Blackwell (1997).

[10] Yagiura, M. and Ibaraki, T.: Generalized assignment problem, *Handbook of Approximation Algorithms and Metaheuristics*, Gonzalez, T. (Ed.), Computer Information Science Series, Chapman Hall/CRC (2006).

[11] Beasley, J.E.: Welcome to OR-Library, OR-Library (2012) (online), available from ⟨http://people.brunel.ac.uk/˜mastjjb/jeb/info.html⟩.

[12] Eikland, K. and Notebaert, P.: lpsolve — Free Development software downloads at SourceForge.net, lpsolve (2012) (online), available from ⟨http://sourceforge.net/projects/lpsolve/⟩.

[13] 岩崎 学：ノンパラメトリック法，pp.54–55, 東京図書 (2006).

花田 研太

1985 年生．2011 年神戸大学海事科学部マリンエンジニアリング課程卒業．同年同大学大学院海事科学研究科修士課程入学．マルチエージェントシステム，組合せ最適化に関する研究に従事．2011 年 PRIMA-2011 Runner up for Best Student Paper Award 受賞．

平山 勝敏 （正会員）

1967 年生．1990 年大阪大学基礎工学部制御工学科卒業．1992 年同大学大学院基礎工学研究科博士前期課程修了．1995 年同大学院基礎工学研究科博士後期課程修了．工学博士．1995 年神戸商船大学助手．1997 年同講師．2001 年同助教授．2003 年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）．2007 年神戸大学大学院海事科学研究科准教授．1999～2000 年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）．マルチエージェントシステム，制約充足/SAT，組合せ最適化に関する研究に従事．2010 年 IFAAMAS Influential Paper Award 受賞．2011 年 CP-2011 Best Application Paper Award 受賞．2011 年 PRIMA-2011 Runner up for Best Student Paper Award 受賞．電子情報通信学会，人工知能学会，日本オペレーションズリサーチ学会，AAAI 各会員．