多層一般化相互割当問題の定式化とその解法

花田, 研太
平山, 勝敏

(Citation)
電子情報通信学会論文誌. D, 情報・システム,96(12):2908-2919

(Issue Date)
2013-12-01

(Resource Type)
journal article

(Version)
Version of Record

多層一般化相互割当問題の定式化とその解法

花田　研太[†a)]　　平山　勝敏[†b)]

Multi-layered Generalized Mutual Assignment Problem : Formulation and Solutions

Kenta HANADA[†a)] and Katsutoshi HIRAYAMA[†b)]

あらまし　本論文では，一般化相互割当問題（GMAP）の従来の定式化と過制約な状況を扱う定式化の齟齬を解消する新しい問題として多層一般化相互割当問題（MLGMAP）を提案する．MLGMAPは，割り当てられない財の数を最小化した上で，効用最大，若しくはコスト最小となる財の割当てを求める問題である．また，MLGMAPを解く手法として2段階最適化手法とペナルティコスト最適化手法を提案し，それぞれに対して既存のプロトコルであるDisLRP-DAとDisLRP-IBFを適用した．提案手法を実験的に評価した結果，ペナルティコスト最適化手法におけるDisLRP-DAが最も有効であるが，ペナルティコストの設定によって，割り当てられない財の数を最小化できる問題例の数と得られる実行可能解の質にトレードオフの関係があることが分かった．

キーワード　マルチエージェントシステム，分散協調問題解決，分散最適化，ラグランジュ緩和

1. まえがき

分散最適化問題は，地理的に分散した情報を保持したエージェントが協調的に解の探索を行うことによって，大域最適解を求める問題である．汎用的な分散協調問題解決の枠組みとしては分散制約最適化問題[1]や分散施設配置問題[2]が挙げられる．応用例としてはマルチロボットタスク割当て[3]や分散ターゲット追跡[4]などが挙げられる．

一般化相互割当問題（GMAP: Generalized Mutual Assignment Problem）は，古くから組合せ最適化分野において研究されている一般化割当問題（GAP: Generalized Assignment Problem）を分散環境へ拡張した問題である[5]～[7]．GMAPでは，以下に示す制約を満たすように機械（エージェント）に対して仕事（財）を割り当てる．まず，各財はいずれか一つのエージェントに必ず割り当てられなければならない．これを割当制約と呼ぶ．また，エージェントは財が割り当てられたときに資源を消費するが，消費される資源量の総和がエージェントの資源容量を超えてはならない．これをナップサック制約と呼ぶ．GMAPの目的は，これらの制約を満たす財の割当て（許容領域）の中で，割当てに伴うエージェントの効用の総和が最大となる財の割当てを求めることである．

GMAP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）が提案されている[5]～[7]．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り，どのエージェントがどの財を得るかをエージェント間で調整し，各エージェントは財の割当てを協調的に決定する．

従来のDisLRP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容量をもつと暗黙のうちに仮定していた．しかし現実には，エージェントが十分な資源容量をもたないような，いわゆる過制約な問題例もあり得ると考えられる．このような問題例に対応するために，従来とは異なる定式化が提案されており，その解法としてdisposalエージェントを用いたDisLRP（DisLRP-DA）と不等式制約緩和に基づくDisLRP（DisLRP-IBF）の二つが提案されている[8]．

両プロトコルの基本的な振る舞いは従来のDisLRPと同じであるが，以下の点で異なる．DisLRP-DAでは，資源制約のない（無限の資源をもつ）仮想的な

† 神戸大学大学院海事科学研究科，神戸市
Graduate School of Maritime Sciences, Kobe University, 5–1–1 Fukaeminamimachi, Higashinada-ku, Kobe-shi, 658–0022 Japan
a) E-mail: kenta_hanada@stu.kobe-u.ac.jp
b) E-mail: hirayama@maritime.kobe-u.ac.jp

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なかった財を disposal エージェントに割り当てる．DisLRP-IBF では，GMAP の割当制約である「各財は，いずれか一つのエージェントに必ず割り当てられる」を緩和し，「各財は，高々一つのエージェントに割り当てられる」とする．

以上のように，DisLRP-DA 及び DisLRP-IBF では，エージェントを追加，若しくは制約を緩和することによって許容領域を変更し，従来の DisLRP では扱えなかった過制約な問題例に対処できるようになった．しかし DisLRP-DA と DisLRP-IBF には，許容領域を変更することによって生じる問題点が二つある．

1 点目は，過制約でない問題例の扱いについてである．従来の DisLRP では，そのような問題例に対し，全ての財をエージェントに割り当てる．しかし，過制約でない問題例を DisLRP-DA 及び DisLRP-IBF で解くと，全ての財をエージェントに割り当てるとは限らない．なぜなら，ある財に関してそれを割り当てない方が効用が高いという状況が存在するからである．

2 点目は，従来の DisLRP は最小化問題にも自然に対応可能だが，DisLRP-DA 及び DisLRP-IBF はそうではないという点である．すなわち，DisLRP-DA 及び DisLRP-IBF で最小化問題を解こうとすると，許容領域が緩和されているため最適値が常に 0，すなわち財を割り当てないのが最適解となってしまう．

そこで本研究では，GMAP の許容領域にレイヤーという概念を導入し，これら二つの問題点に対処できる多層一般化相互割当問題（Multi-layered GMAP）を新たに提案する．MLGMAP では，緩和された割当制約とナップサック制約に加えて，割り当てられない財の数（レイヤー数）を制約条件に追加した新たな許容領域（レイヤー）を考える．MLGMAP の目的は，レイヤー数が最小という条件のもとで，効用の総和が最大，若しくはコストの総和が最小となる財の割当てを求めることである．また，この問題を解く新しい手法を二つ提案し，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．

一つ目の手法は 2 段階最適化手法である．この手法では，MLGMAP を 2 段階の最適化問題として解く．1 段階目は，レイヤー数が最小となるレイヤーを求める．2 段階目は，1 段階目で求まった最適なレイヤーに対して，全体の効用が最大，若しくはコストが最小となる財の割当を求める．

二つ目の手法はペナルティコスト最適化手法である．この手法は，2 段階最適化手法と異なり，レイヤー数が最小となるレイヤーと，全体の効用が最大，若しくはコストが最小となる財の割当てを同時に求める．基本的なアイディアは，割り当てられない財が発生した場合，目的関数にペナルティコストがかかるようにする．ペナルティコストを適切な定数値に設定した場合，求められる最適解は必ずレイヤー数最小のレイヤーに含まれていることが保証できる．

以下，本論文は次のように構成される．**2.** では GMAP の従来の定式化とプロトコル，及び，それらの問題点を述べる．**3.** では MLGMAP におけるレイヤー及びレイヤー数を定義し，MLGMAP の目的を述べる．**4.** では，2 段階最適化手法に基づく DisLRP-DA 及び DisLRP-IBF を提案し，一方，**5.** では，ペナルティコスト最適化手法に基づく DisLRP-DA 及び DisLRP-IBF を提案する．続く **6.** 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，**7.** 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

2.1 従来の定式化

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．

$$\mathcal{GAP} \quad (\text{decide } x_{kj},\ \forall k \in A,\ \forall j \in J):$$

$$\max. \quad \sum_{k\in A}\sum_{j\in J} p_{kj}x_{kj}$$

$$\text{s. t.} \quad \sum_{k\in A} x_{kj} = 1,\ \ \forall j \in J, \tag{1}$$

$$\sum_{j\in J} w_{kj}x_{kj} \le c_k,\ \ \forall k \in A, \tag{2}$$

$$x_{kj} \in \{0,1\},\ \forall k \in A,\ \forall j \in J. \tag{3}$$

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，$p_{kj} \in \mathbb{R}$ はエージェント $k \in A$ が財 $j \in J$ を得た場合の効用，$w_{kj} \in \mathbb{R}^+$ はエージェント k が財 j を得た場合の資源消費量，$c_k \in \mathbb{R}^+$ 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），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．式 (1)，式 (2)，式 (3) を満たす解ベクトル $x = (x_{11}, \ldots, x_{mn})$ の集合 X を $\mathcal{GAP}$ の許容領域と呼ぶ．なお，この問題は NP 困難な問題に属する．以降，紙面の都合上 0-1 制約を省略する．

2.2 過制約な問題例のための定式化

2.1 では，問題例が実行可能である，すなわち，割当制約とナップサック制約を全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，全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，いわゆる過制約な問題例もあり得る．そこで，そのような問題例に対処できる定式化が二つ提案されている [8]．

2.2.1 disposal エージェントを用いた定式化

第 1 の方法は，資源制約のない（無限の資源をもつ）仮想的な 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を disposal エージェントに割り当てるというものである．なお，disposal エージェントが財を得た場合の効用は 0 である．

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

$$
\begin{aligned}
\mathcal{GAPD}\quad & (\text{decide } x_{kj},\ \forall k \in A \cup \{d\},\ \forall j \in J): \\
\max.\quad & \sum_{k \in A} \sum_{j \in J} p_{kj} x_{kj} \\
\text{s. t.}\quad & \sum_{k \in A \cup \{d\}} x_{kj} = 1,\ \ \forall j \in J, && (4) \\
& \sum_{j \in J} w_{kj} x_{kj} \leq c_k,\ \ \forall k \in A, && (5)
\end{aligned}
$$

と記述できる [8]．式 (4)，式 (5)，そして 0-1 制約を満たす解ベクトル $x_D = (x_{11}, \ldots, x_{mn}, x_{d1}, \ldots, x_{dn})$ の集合 X_D を $\mathcal{GAPD}$ の許容領域と呼ぶ．これは，前述の $\mathcal{GAP}$ の許容領域 X とは明らかに異なる．

また，この問題を解く分散解法として，disposal エージェントを用いた DisLRP（DisLRP-DA）が提案されている [8]．

2.2.2 不等式制約緩和に基づく定式化

もう一つの方法は，GMAP の割当制約である「各財は，いずれか一つのエージェントに必ず割り当てられる」を緩和し，「各財は，高々一つのエージェントに割り当てられる」とするものである．系全体の問題は，以下のように記述できる [8]．

$$
\begin{aligned}
\mathcal{GAPI}\quad & (\text{decide } x_{kj},\ \forall k \in A,\ \forall j \in J): \\
\max.\quad & \sum_{k \in A} \sum_{j \in J} p_{kj} x_{kj} \\
\text{s. t.}\quad & \sum_{k \in A} x_{kj} \leq 1,\ \ \forall j \in J, && (6) \\
& \sum_{j \in J} w_{kj} x_{kj} \leq c_k,\ \ \forall k \in A. && (7)
\end{aligned}
$$

式 (6)，式 (7)，そして 0-1 制約を満たす解ベクトル x の集合 X_I を $\mathcal{GAPI}$ の許容領域と呼ぶ．$\mathcal{GAPI}$ は，$\mathcal{GAP}$ や $\mathcal{GAPD}$ とは異なる割当制約をもつため，その許容領域 X_I はそれぞれの許容領域 X や X_D とは明らかに異なる．

また，この問題を解く分散解法として，不等式制約緩和に基づく DisLRP（DisLRP-IBF）が提案されている [8]．

2.2.3 分 解 法

各定式化に対する解法では，割当制約をラグランジュ緩和して問題を分解する．ここでは $\mathcal{GAPD}$ に対する分解の手順を示す．

$\mathcal{GAPD}$ 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．

$$
\begin{aligned}
\max.\quad & \sum_{j \in J} p_{kj} x_{dj} + \sum_{j \in J} \mu_j \left(1 - \sum_{k \in A \cup \{d\}} x_{kj} \right) \\
\text{s. t.}\quad & \sum_{j \in J} w_{kj} x_{kj} \leq c_k,\ \forall k \in A.
\end{aligned}
$$

$\mu = (\mu_1, \ldots, \mu_n) \in \mathbb{R}^n$ はラグランジュ乗数ベクトルと呼ばれ，この問題の最適値及び最適解は財 j に対するラグランジュ乗数 μ_j の値に依存する．ラグランジュ緩和問題の最適値を μ の関数として $L : \mathbb{R}^n \mapsto \mathbb{R}$ と定義する．なお，等式制約をラグランジュ緩和したときは μ は無制約となるが，$\mathcal{GAPI}$ の割当制約 (6) をラグランジュ緩和するときは，式 (6) が不等式なので $\mu_j \geq 0, \forall j \in J$ という非負制約が伴う．

この問題は，制約集合ごとに通常のエージェント k に関するナップサック問題

$$
\begin{aligned}
\max.\quad & \sum_{j \in J} (p_{kj} - \mu_j) x_{kj} \\
\text{s. t.}\quad & \sum_{j \in J} w_{kj} x_{kj} \leq c_k,
\end{aligned}
$$

disposal エージェントに関する最大化問題

$$
\max. \sum_{j \in J} (-\mu_j) x_{dj},
$$

及び，ラグランジュ乗数のみからなる項

$$\sum_{j\in J}\mu_j,$$

に分解することができる．$\mathcal{GAPD}$ と $\mathcal{GAPI}$ を分解した後の部分問題を表 1 の DisLRP-DA 及び DisLRP-IBF の欄に示す．

2.2.4 ラグランジュ双対問題と劣勾配法

ラグランジュ緩和問題の最適値は，緩和前の問題（以降，原問題と呼ぶ）が最大化問題の場合，原問題に対する上界値を与えることが知られている．より良い上界値を求めることは，最適解を探索する上で重要な情報をもたらす．また，後述するラグランジュヒューリスティックにも良い影響を与えるとされる．この上界値を最小化する問題をラグランジュ双対問題と呼び，$\mathcal{GAPD}$ の場合，

$$L(\mu^*)=\min .L(\mu),$$

という無制約最小化問題となる．

この問題の最適解 μ^* は，目的関数が微分不可能なので解析的に求めることができない．そこで，DisLRP では劣勾配法という山登りアルゴリズムを用いて μ を更新する．劣勾配法の具体的な更新規則は，**2.2.6** で示す．

2.2.5 ラグランジュヒューリスティック

ラグランジュ緩和問題の最適解から，原問題の実行可能解を求める手法を一般にラグランジュヒューリスティックと呼ぶ．ラグランジュヒューリスティックを用いて生成された原問題の実行可能解は，多くの場合良い下界値をもつとされる．なお，ラグランジュヒューリスティックは定式化ごとに個別に設計される．具体的な手法は，**2.2.6** で示す．

2.2.6 プロトコルの手続き

本項では，DisLRP-DA の手続きを述べる．

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

(Step 2) 現在の μ の値のもとで，全てのエージェントがそれぞれのナップサック問題を解く．各エージェントは最適値及び最適解を，原問題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) 原問題の割当制約が全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 回の繰り返しをラウンドと呼び，処理の 1 単位とみなす．

また，Step 3 から Step 5 の計算には，本来，系全体の大域情報が必要である．そのためには，系全体をモニター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得ることから，[7] 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．

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，ラグランジュヒューリスティックを用いて原問題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．DisLRP-DA のラグランジュヒューリスティックは以下のとおりである．

- 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．

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

ここで，右辺第 2 項の劣勾配 $g_j^{(t)}$ にかかる係数をステップ長と呼ぶ．この更新規則では，エージェントはその時点での $BestUB$ と $BestLB$ を知る必要があるが，これらは大域情報であるため，[7] と同様に生成木を用いてそれらを収集する．また，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．

DisLRP-IBFの基本的な手続きは，DisLRP-DAと同様であるが，Step 3の終了条件，ラグランジュヒューリスティック，劣勾配及び劣勾配法の更新規則が異なっている．詳細は[8]を参照されたい．

2.3 DisLRP-DA及びDisLRP-IBFの問題点

DisLRP-DA及びDisLRP-IBFでは，従来のDisLRPで解けないような過制約な問題例を扱うとされる[8]．一方，過制約でない問題例に対しては，DisLRP-DA及びDisLRP-IBFは従来のDisLRPと同じ最適値及び最適解を与えるとは限らない．これは，過制約な問題例を扱えるように割当制約を緩和したため許容領域が広がり，結果として最適値及び最適解が変わる可能性があるからである．

また，DisLRP-DA及びDisLRP-IBFでは最大化問題しか扱えない．従来のDisLRPでは，効用 p_{kj} をコストとみなすだけで最小化問題にも対応できるが，DisLRP-DA及びDisLRP-IBFで同様の手段を取った場合，最適値が0，すなわち財を割り当てない自明な解しか得られない．これは，過制約な問題例を扱うために許容領域を拡張した結果，財を全く割り当てなくても良いことを許しているためである．

そこで本論文では，従来の定式化と過制約な問題例を扱うための定式化の齟齬を解消し，また，最小化問題にも対応できる新たな問題として，多層一般化相互割当問題（MLGMAP）を提案する．また，この問題を解く手法を二つ提案する．

3. 多層一般化相互割当問題

本章では，MLGMAPで用いる語句を定義し，MLGMAPの目的を述べる．

3.1 定　　義

MLGMAPでは，過制約な問題例の許容領域 X_D 及び X_I を割り当てられない財の数ごとに分割する．ここで，割り当てられない財の数をレイヤー数，分割された許容領域をレイヤーと呼ぶことにする．レイヤーと呼ぶ理由は，分割された許容領域に対して，割り当てられない財の数で順位を付けるからである．すなわち，分割された許容領域は階層化される．

まず，レイヤー及びレイヤー数を定義する．

[定義1] $l \in J \cup \{0\}$ をレイヤー数とし，disposalエージェントを用いた定式化におけるレイヤー数を，

$$l = \sum_{j \in J} x_{dj},$$

不等式制約緩和に基づく定式化におけるレイヤー数を，

$$l = \sum_{j \in J} \left(1 - \sum_{k \in A} x_{kj}\right),$$

と定義する．集合 X_D^l を，

$$X_D^l = \left\{ x_D \middle| l = \sum_{j \in J} x_{dj}, x_D \in X_D \right\},$$

とし，disposalエージェントを用いた定式化のレイヤーと定義する．また，集合 X_I^l を，

$$X_I^l = \left\{ x \middle| l = \sum_{j \in J} \left(1 - \sum_{k \in A} x_{kj}\right), x \in X_I \right\},$$

とし，不等式制約緩和に基づく定式化のレイヤーと定義する．

許容領域を分割しているので，$s \in \{D, I\}$ として，

$$X_s = \bigcup_{l=0}^{n} X_s^l,$$

が成り立つ．

なお，X_D^l 及び X_I^l は空となる場合がある．例えば，過制約な問題例においては必ず $X_D^0 = \emptyset, X_I^0 = X = \emptyset$ である．

3.2 目　　的

MLGMAPの目的は，非空なレイヤーの中でレイヤー数が最小となるレイヤーを求めること，また，そのレイヤーにおいてエージェントの効用の総和が最大，もしくコストが最小となるような財の割当てを求めることである．以降，非空なレイヤーの中でレイヤー数が最小となるレイヤーを最適なレイヤーと呼ぶ．また，そのときのレイヤー数を最適なレイヤー数と呼ぶ．

この目的に従えば，過制約でない問題例をMLGMAPで定式化して解いたとき，従来のDisLRPと同じ最適値及び最適解を導出できる．まず，過制約でない問題例において X_D^0 及び X_I^0 は必ず非空であり，0はレイヤー数の中で最も小さいので，非空なレイヤーの中でレイヤー数が最小となることが保証されている．次に，レイヤー数0において，disposalエージェントに関する決定変数 x_{dj} は全て0となるため，許容領域 X_D^0 は X と対応する．すなわち，通常のエージェントの解ベクトル x について，同じ最適値及び最適解を導

出する．また，定義1から明らかに $X = X_I^0$ であるから，同様に同じ最適値及び最適解を導出する．

一方，過制約な問題を最小化問題として従来のDisLRP-DA及びDisLRP-IBFで解いた場合，常に X_D^n 及び X_I^n の最適値，すなわち0が求まる．しかし，MLGMAPではレイヤー数が最小のレイヤーが選ばれるため，X_D^n 及び X_I^n 以外のレイヤーが全て空でない限り，最適値が0になることはない．

なお，過制約な問題例をMLGMAPで定式化して解いた場合，従来のDisLRP-DA及びDisLRP-IBFとは異なる最適値及び最適解が得られることになる．

MLGMAPは最大化問題と最小化問題の両方に対応できるが，紙面の都合上，以降の章では効用 p_{kj} をコストとみなして最小化問題の定式化のみ記述する．

4. MLGMAPのための2段階最適化手法

本章では，MLGMAPを解く一つ目の手法である2段階最適化手法について述べる．この手法では，まず最適なレイヤーを求め，その後，最適なレイヤー内でコストの総和が最小となる財の割当てを求める．

2段階最適化手法におけるMLGMAPを解く手順は以下のとおりである．

（1） 1段階目として，最適なレイヤーとそのレイヤー数 l_{opt} を求め，そのレイヤーにおける実行可能解を求める．

（2） もし，l_{opt} が財の総数 n に一致するなら，コスト最小の財の割当ては自明なため手順を終了する．

（3） もし，l_{opt} が n より小さければ，最適なレイヤーにおける全体のコストを最小化する財の割当てを求める2段階目の最適化問題を解く．このとき，初期の実行可能解は1段階目で求まったものを使用する．

4.1 1段階目の定式化と解法

1段階目の目的は最適なレイヤーを求めることであるから，定式化は目的関数をレイヤー数とし，制約式は $\mathcal{GAPD}$ または $\mathcal{GAPI}$ と同じにすればよい．よって，disposalエージェントを用いた定式化は $\mathcal{GAPD}$ の目的関数を以下の式

$$l_{opt} = \max. - \sum_{j \in J} x_{dj},$$

に置き換える(注1)．不等式制約緩和に基づく定式化の場合は $\mathcal{GAPI}$ の目的関数を以下の式

$$l_{opt} = \max. - \sum_{j \in J} \left(1 - \sum_{k \in A} x_{kj} \right),$$

に置き換える．

2.2.3 と同様に割当制約をラグランジュ緩和し，部分問題へ分解する．分解後の部分問題を表1の $\text{DisLRP-DA}_{phase1}$ と $\text{DisLRP-IBF}_{phase1}$ の欄にそれぞれ示す．

2.2.1 及び **2.2.2** の定式化と本節の定式化をそれぞれ比較すると，目的関数はもとの定式化の特殊形であり，制約式は同じである．不等式制約緩和に基づく定式化に関しては定数項に変化があるが，これは解法に対して何ら影響を与えない．したがって，1段階目の問題は[8]と同じ解法を用いて解くことができる．

なお，DisLRPは発見的解法であるため l_{opt} が求まる保証はない．つまり，この手順だと1段階目で l_{opt} が求まらず2段階目に移れない可能性がある(注2)．

4.2 2段階目の定式化と解法

2段階目の目的は，1段階目で求まった最適なレイヤー内でコストが最小となる財の割当てを求めることである．したがって，まず最大化問題から最小化問題へと変更する(注3)．具体的には，目的関数を

$$\max. \sum_{k \in A} \sum_{j \in J} (-p_{kj})\, x_{dj}$$

とする．

次に，1段階目で求まった最適なレイヤー数 l_{opt} を用いて，$\mathcal{GAPD}$ の制約式に，

$$\sum_{j \in J} x_{dj} = l_{opt}, \tag{8}$$

$\mathcal{GAPI}$ の制約式に，

$$\sum_{j \in J} \left(1 - \sum_{k \in A} x_{kj} \right) = l_{opt}, \tag{9}$$

をそれぞれ追加する．以後，式(8)及び(9)をレイヤー制約と呼ぶ．

分散環境で問題を解くためには問題を分解しなけ

(注1)：$\min. \sum_{j \in J} x_{dj}$ と表現するのが自然であるが，今後の議論のため最大化問題として記述している．以降の節も同様である．

(注2)：1段階目では最適なレイヤー数の上界値である l_{ub} が求められるので，レイヤー制約を不等式として2段階目を解く方法も考えられる．しかし，本論文では後に述べるペナルティコスト最適化手法との比較のため今回は考えない．

(注3)：p_{kj} を効用とみなして最大化を目的とするならば，この変更をする必要はない．

ればならないが，2段階目定式化ではレイヤー制約が新たに追加されている．しかし，式(8)にはdisposalエージェントに関する決定変数しか存在しないため，割当制約をラグランジュ緩和して分解する方法において分解可能性を壊さない．したがって，disposalエージェントを用いた定式化では**2.2.3**と同様に割当制約のみをラグランジュ緩和し，部分問題へ分解すればよい．一方，不等式制約緩和に基づく定式化では，式(9)に全ての決定変数が使われているため，分解可能性を壊している．よって，部分問題への分解を行うためには割当制約に加えて式(9)を新たにラグランジュ緩和する必要がある．

不等式制約緩和に基づく定式化に対して式(9)及び割当制約を緩和すると，

$$\begin{aligned}\max.\quad & \sum_{k\in A}\sum_{j\in J}(-p_{kj})\,x_{kj}+\sum_{j\in J}\mu_j\left(1-\sum_{k\in A}x_{kj}\right)\\ & +\lambda\left(1+\frac{1}{l_{opt}-n}\sum_{k\in A}\sum_{j\in J}x_{kj}\right)\\ \text{s. t.}\quad & \sum_{j\in J}w_{kj}x_{kj}\le c_k,\ \forall k\in A,\\ & \mu_j\ge 0,\ \forall j\in J,\end{aligned}$$

というラグランジュ緩和問題が得られる．λはレイヤー制約に対応するラグランジュ乗数である．以降，部分問題への分解は**2.2.3**と同様である．分解後の部分問題を表1のDisLRP-IBF$_{phase2}$及びDisLRP-DA$_{phase2}$の欄に示す．

2.2.1及び**2.2.2**の定式化と本節の定式化を比較する．まず，2段階目の定式化ではレイヤー制約がそれぞれ追加されたため，許容領域が$\mathcal{GAPD}$及び$\mathcal{GAPI}$と異なっている．したがって，ラグランジュヒューリスティックを変更する必要がある．不等式制約緩和に基づく定式化では，そのレイヤー制約もラグランジュ緩和されているので，新たに追加されたλのために劣勾配法を変更する必要がある．

ラグランジュヒューリスティックは次のように変更する．まず[8]の解法と同様のラグランジュヒューリスティックを用いて，実行可能解の候補を作成する．次に，その実行可能解の候補がレイヤー制約を満たしているかを調べる．もし満たしていれば実行可能解とし，満たしていなければ候補を破棄し，そのラウンドにおける実行可能解はなしとする(注4)．

不等式制約緩和に基づく定式化における劣勾配法は次のように変更する．ラウンドtにおける各財j及びレイヤー制約の劣勾配を，

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

$$h^{(t)} \leftarrow 1+\frac{1}{l_{opt}-n}\sum_{k\in A}\sum_{j\in J}p_{kj}x_{kj},$$

また，ステップ長を，

$$e=\frac{\pi(BestUB^{(t)}-BestLB^{(t)})}{(h^{(t)})^2+\sum_{j\in J}(g_j^{(t)})^2},$$

として，次のような更新規則を用いる．

$$\begin{aligned}\mu_j^{(t+1)} &\leftarrow \max\{\mu_j^{(t)}-e\cdot g_j^{(t)},0\},\\ \lambda^{(t+1)} &\leftarrow \lambda^{(t)}-e\cdot h^{(t)}.\end{aligned}\qquad(10)$$

不等式制約緩和に基づく定式化の場合，$\mu_j\ge 0$でなければならないので，式(10)によってそれを保証する．

5. MLGMAPのためのペナルティコスト最適化手法

本章では，MLGMAPを解くための二つ目の手法であるペナルティコスト最適化手法について述べる．ペナルティコスト最適化手法のアイディアを説明するために，以下のような問題を考える．

$$\begin{aligned}f(l)=\max.\quad & \sum_{k\in A}\sum_{j\in J}(-p_{kj})\,x_{kj}-r\cdot l\\ \text{s. t.}\quad & \sum_{j\in J}x_{dj}=l,\\ & \sum_{k\in A\cup\{d\}}x_{kj}=1,\ \forall j\in J,\\ & \sum_{j\in J}w_{kj}x_{kj}\le c_k,\ \forall k\in A.\end{aligned}\qquad(11)$$

ここで，$r\in\mathbb{R}^+$はペナルティコストを表す定数である．この定式化は，割り当てられない財が発生した場合，重み$-r$が目的関数値にかかるようになっている．

ペナルティコスト最適化手法では，lを定数ではなく決定変数とみなす．しかし，式(11)より，lは同じく決定変数であるx_{dj}で表現できるため，目的関数及び制約式からlを削除することができる．そのときの

(注4)：毎ラウンドにおいて実行可能解が得られるようなラグランジュヒューリスティックを構成するのが理想であるが，それを実現するためには許容領域の探索が必要となる．この探索は最悪の場合指数時間となり，全体の処理性能を大きく低下させるおそれがある．したがって，今回はこれ以上の探索を行うことはせず，今後の課題とする．

目的関数は以下のようになる．

$$f^* = \max. \sum_{k\in A}\sum_{j\in J}(-p_{kj})\,x_{kj} - r\sum_{j\in J}x_{dj}.$$

5.1 ペナルティコスト r の設定

f^* が最適なレイヤーにおける最適値であるには，$f^* = f(l_{opt})$ であればよい．そこで，$f^* = f(l_{opt})$ となるための r の十分条件を求める．

$f^* = f(l_{opt})$ であるためには，l_{opt} における最適値が他の任意のレイヤー数における最適値よりも小さければよい．したがって，$l' \in \{l_{opt}, \ldots, n-1\}$ として，十分条件は次のように表すことができる．

$l_{opt} \neq n$ のとき $f(l_{opt}) < f(l'),\ \forall l'.$

$l_{opt} = n$ のとき r の値は任意．

$f(l)$ の最適解を $x^{(l)}$ とする．

$$b = \sum_{k\in A}\sum_{j\in J} p_{kj}x_{kj}^{(l_{opt})} - \sum_{k\in A}\sum_{j\in J} p_{kj}x_{kj}^{(l')}$$

として式変形すると，最終的に十分条件は

$$r > \frac{b}{l' - l_{opt}},\ \forall l' \tag{12}$$

となる．

この十分条件 (12) を満たす r を求めるには全てのレイヤーにおける最適値が必要となる．しかし，最適値は問題を解く前には知り得ようのない情報なので，十分条件 (12) を満たす r は事前に計算できない．したがって，これらの最適値を使わない r の十分条件を考える必要がある．

本研究では，次のような十分条件を提案する．

[定理 1]

$$r > \sum_{k\in A}\sum_{j\in J} p_{kj} \tag{13}$$

は，$f^* = f(l_{opt})$ であるための十分条件である．

証明　$x_{kj} \in \{0,1\}$ より，任意の x_{kj} で

$$\sum_{k\in A}\sum_{j\in J} p_{kj} \geq \sum_{k\in A}\sum_{j\in J} p_{kj}x_{kj}$$

が成り立つ．したがって $l_{opt} \neq n$ のとき，任意の l' において $l' - l_{opt} \geq 1$ だから，

$$\frac{b}{l'-l_{opt}} \leq b \leq \sum_{k\in A}\sum_{j\in J} p_{kj}x_{kj}^{(l_{opt})} \leq \sum_{k\in A}\sum_{j\in J} p_{kj}$$

が成り立つ．つまり，$r > \sum_{k\in A}\sum_{j\in J} p_{kj}$ ならば，

$$r > \sum_{k\in A}\sum_{j\in J} p_{kj} \geq \frac{b}{l' - l_{opt}},\ \forall l'$$

が成り立つから，題意は証明された．　□

なお，r の値は大域情報となるため，[7] の方法に基づき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し，問題を解く前にあらかじめ r を算出しておく．

5.2 定式化と解法

ペナルティコスト最適化手法における disposal エージェントを用いた定式化は，$\mathcal{GAPD}$ の目的関数を以下の式

$$\max. \sum_{k\in A}\sum_{j\in J}(-p_{kj})\,x_{kj} - r\sum_{j\in J}x_{dj}$$

に置き換えればよい．また，不等式制約緩和に基づく定式化は，$\mathcal{GAPI}$ の目的関数を以下の式

$$\max. \sum_{k\in A}\sum_{j\in J}(-p_{kj})\,x_{kj} - r\sum_{j\in J}\left(1 - \sum_{k\in A}x_{kj}\right)$$

に置き換えればよい．

制約式はもとの問題と同じであるから，**2.2.3** と同様に割当制約をラグランジュ緩和し，部分問題へ分解する．分解後の部分問題を表 1 の DisLRP-DA$_{pnt}$ 及び DisLRP-IBF$_{pnt}$ の欄に示す．

2.2.1 及び **2.2.2** の定式化と本節の定式化をそれぞれ比較すると，目的関数はもとの定式化の特殊形であり，制約式は同じである．不等式制約緩和に基づく定式化に関しては定数項に変化があるが，これは解法に対して何ら影響を与えない．したがって，これらの問題は [8] と同じ解法を用いて解くことができる．

6. 実　　験

4. 及び **5.** で示した四つの解法を実装して実験を行った．MLGMAP 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，評価実験では提案した四つの解法の比較のみを扱う．

実験では過制約な状況が含まれた最小化問題の問題例を与える必要があるため，OR-Library [9] に掲載されている GAP のベンチマーク問題例（gapa, gapb, gapc）の 18 問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 $x \in \{0.1, 0.2, \ldots, 1.0\}$ をかけて小数点以下を切り捨て

表 1 既存手法及び提案手法の部分問題別の比較
Table 1 A comparison among protocols.

protocol	agent k （制約式は省略）	disposal agent （01 制約は省略）	constant
DisLRP-DA	max. $\sum_{j\in J}(p_{kj}-\mu_j)x_{kj}$	max. $\sum_{j\in J}(-\mu_j)\,x_{dj}$	$\sum_{j\in J}\mu_j$
DisLRP-IBF	max. $\sum_{j\in J}(p_{kj}-\mu_j)x_{kj}$	-	$\sum_{j\in J}\mu_j$
DisLRP-DA$_{ph1}$	max. $\sum_{j\in J}(-\mu_j)\,x_{kj}$	max. $\sum_{j\in J}(-1-\mu_j)\,x_{dj}$	$\sum_{j\in J}\mu_j$
DisLRP-DA$_{ph2}$	max. $\sum_{j\in J}(-p_{kj}-\mu_j)x_{kj}$	max. $\sum_{j\in J}(-\mu_j)\,x_{dj}$ s. t. $\sum_{j\in J}x_{dj}=l_{opt}$,	$\sum_{j\in J}\mu_j$
DisLRP-IBF$_{ph1}$	max. $\sum_{j\in J}(1-\mu_j)x_{kj}$	-	$-n+\sum_{j\in J}\mu_j$
DisLRP-IBF$_{ph2}$	max. $\sum_{j\in J}\left(-p_{kj}-\mu_j+\frac{\lambda}{l_{opt}-n}\right)x_{kj}$	-	$\sum_{j\in J}\mu_j+\lambda$
DisLRP-DA$_{pnt}$	max. $\sum_{j\in J}(-p_{kj}-\mu_j)\,x_{kj}$	max. $\sum_{j\in J}(-r-\mu_j)\,x_{dj}$	$\sum_{j\in J}\mu_j$
DisLRP-IBF$_{pnt}$	max. $\sum_{j\in J}(-p_{kj}+r-\mu_j)\,x_{kj}$	-	$\sum_{j\in J}\mu_j-\sum_{j\in J}r$

表 2 問題例の一覧と r_{s1} と r_{s2} に関する比較
Table 2 A list of instances and a comparison among r_{s1} and r_{s2}.

カテゴリー	問題例名	エージェント数	財数	r_{s1}										r_{s2}
				0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	
a	c5100-1	5	100	79	136	163	241	129	80	59	44	40	40	4462
a	c5200-2	5	200	131	225	168	180	202	74	53	49	39	39	8788
a	c10100-3	10	100	38	141	188	71	44	37	30	28	28	28	4706
a	c10200-4	10	200	122	236	201	95	55	44	37	33	31	30	9413
a	c20100-5	20	100	58	106	110	39	29	26	23	22	20	20	4860
a	c20200-6	20	200	32	117	191	45	31	28	26	26	26	26	9669
b	c5100-1	5	100	59	104	110	133	174	119	165	174	81	59	4420
b	c5200-2	5	200	107	170	243	166	239	206	94	64	55	49	8774
b	c10100-3	10	100	26	111	268	131	210	68	55	45	34	28	4679
b	c10200-4	10	200	29	90	163	223	285	102	62	47	39	33	9372
b	c20100-5	20	100	22	32	171	165	52	38	32	26	22	22	4880
b	c20200-6	20	200	24	77	150	392	84	41	29	26	22	21	9710
c	c5100-1	5	100	154	119	152	180	228	162	114	75	57	53	4480
c	c5200-2	5	200	85	200	222	172	243	187	126	64	55	50	8674
c	c10100-3	10	100	35	83	157	154	186	114	69	45	40	35	4649
c	c10200-4	10	200	32	136	329	214	364	143	57	44	36	34	9351
c	c20100-5	20	100	38	53	89	121	202	63	34	34	26	26	4866
c	c20200-6	20	200	30	54	205	242	111	59	42	31	26	22	9716

た数値を求め，一つのベンチマーク問題例に対して 10 個の問題例を新たに作成した．すなわち，全部で 180 個の問題例を作成した．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，過制約度を一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．

実験環境には，デスクトップ PC（Core i7-870 2.93GHz，4 コア 8 スレッド，メモリ 8GB，Windows 7 Professional）を用いた．実験は，各手法の動作を模擬するシミュレータを Java 1.6.0_33 で実装し，上記の実験環境上で行う．各エージェントのナップサック問題を解くソルバーには，整数計画問題を解く汎用的なソルバーである IBM ILOG CPLEX 12.4 を使用した．なお，ペナルティコスト r の導出や実験の評価のために必要な最適値も，全て CPLEX を使用して計算した．

6.1 ペナルティコスト r の導出

本節では，**5.1** で示したペナルティコスト r に関する比較を行う．r は本来，エージェントが通信を行って決定すべき値であるが，本研究では通信遅れの影響はあまりないという実験結果 [7] に基づき，あらかじめ全エージェントが知っているものとした．

比較のため，十分条件 (12) の右辺値に近い値 r_{s1} を，以下の数式を用いてあらかじめ求めた．

$$r_{s1}=1+\left\lfloor \max_{l'}.\frac{b}{l'-l_{opt}}\right\rfloor .$$

また，式 (13) の右辺値 r_{s2} もあらかじめ計算を行った．その結果を表 2 に示す．なお，r_{s1} は容量係数 x

ごとにその値が変わるため表 2 では容量係数ごとに示した．また，r_{s2} はもとの問題例と実験のために新たに作成した問題例で変わらないため，もとの問題例のみを示した．

表 2 から，全ての問題例，全ての容量係数において，r_{s1} は r_{s2} より十分小さいことが分かる．

6.2 実験結果と考察

今回，提案手法を評価するため 2 種類の実験を行った．

6.2.1 最適なレイヤーに関する実験

一つ目の評価は，より多くの問題例で最適なレイヤーを求められたのはどの解法かを比較することである．比較対象は，2 段階最適化手法 1 段階目の DisLRP-DA$_{ph1}$ 及び DisLRP-IBF$_{ph1}$ とペナルティコスト最適化手法の DisLRP-DA$_{pnt}$ 及び DisLRP-IBF$_{pnt}$ である．なお，ペナルティコスト r は，r_{s1}（グラフ中では r_s1 と表記），r_{s2}（グラフ中では r_s2 と表記），それらよりも十分大きい定数値 100000 を用いた．

実験結果を図 1，図 2 及び図 3 に示す．これらの図は全て同じ実験結果から作成されており，図 1 は disposal エージェントを用いた解法と不等式制約緩和に基づく解法の比較，図 2 及び図 3 はそれぞれ DisLRP-DA と DisLRP-IBF の容量係数ごとの比較となっている．なお，縦軸は全ての図において最適なレイヤーが求まった問題例の数である．

まず，disposal エージェントを用いた解法と不等式制約緩和に基づく解法の比較を行う．図 1 より，全ての項目において disposal エージェントを用いた解法の方が良いことが分かる．また，disposal エージェントを用いた解法において，最適なレイヤーを求められた問題例の数が一番多いのは $r = 100000$ で，その次に多いのは僅差で r_{s2} であることが分かる．値の小さい r_{s1} では最適なレイヤーが求められた問題例の数が極端に少ない．これは r の値が小さくなればなるほど結果が悪くなることを示唆しており，r_{s2} 以上の値を用いる正当性を与えるものと考えられる．

次に，容量係数による比較を行う．図 2 と図 3 より，DisLRP-DA では，容量係数 0.3 から 0.6 付近で求まる問題例が少なくなる傾向にあり，DisLRP-IBF では容量係数が大きくなるにつれて求まる問題例の数が少なくなる傾向にある．理由としては，ラグランジュ乗数 μ_j の探索空間の差異が挙げられる．μ_j の値が大きくなれば，その財はエージェントに選ばれにくくなり，逆に μ_j の値が小さくなれば，その財はエージェントに選ばれやすくなる．DisLRP-DA は μ_j に制約がないため，財の選ばれやすさの調整が容易に可能である．一方，DisLRP-IBF は μ_j に非負制約が伴うため，財の選ばれやすさの調整には制限が加わる．μ_j の値によってラグランジュ緩和解が決まるので，ラグランジュ緩和解をもとに作られる実行可能解は，上

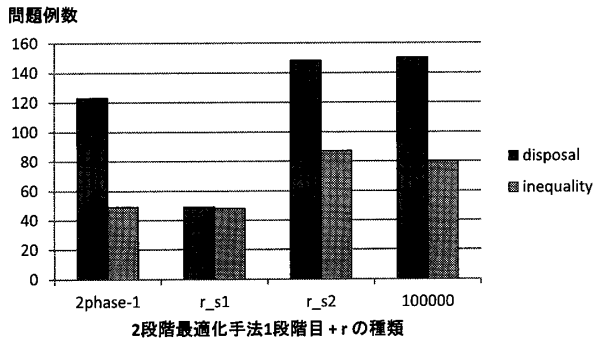


図 1 実験 1 の disposal エージェントと不等式制約緩和の比較

Fig. 1 A comparison between a disposal agent and inequality on an experiment 1.

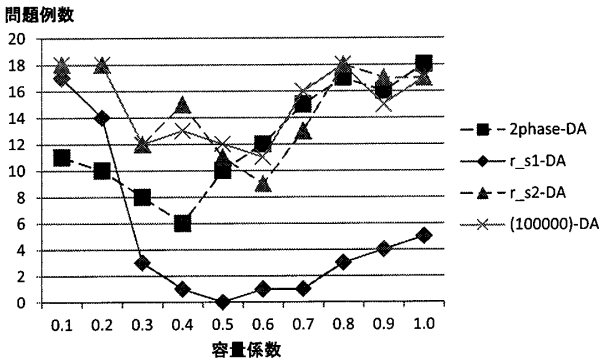


図 2 容量係数による比較（disposal エージェント）

Fig. 2 A comparison among the methods (only disposal agent).

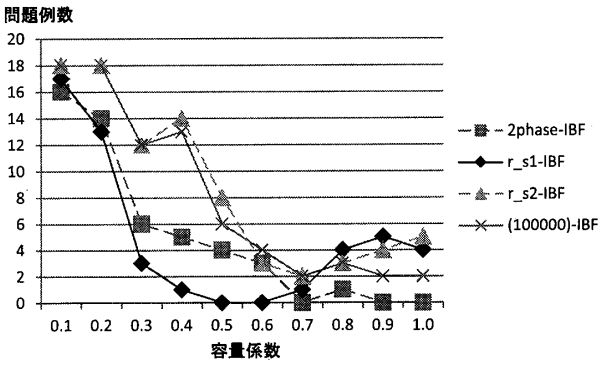


図 3 容量係数による比較（不等式制約緩和）

Fig. 3 A comparison among the methods (only inequality formulation).

記の影響を受けると考えられる．

r_{s1} を用いたペナルティコスト最適化手法では，DisLRP-DA と DisLRP-IBF 共にほぼ同じ値を推移しており，容量係数 0.3 以降の数値が非常に悪いことが分かる．これは，r_{s1} の値が小さいため，たとえ良い目的関数値をもつ実行可能解が得られたとしても，最適なレイヤーに含まれる実行可能解の目的関数値と他のレイヤーに含まれる実行可能解の目的関数値が近いので，他のレイヤーに属する実行可能解が求まってしまうからだと考えられる．

全体的な比較をすると，容量係数 0.3 から 0.6 にかけてどの手法でも最適なレイヤーが求められた問題例数が少なくなっている．これは，0.3 から 0.6 が問題例が過制約かどうかの境界であり，実行可能解の数が少なくなることから，実行可能解を見つけられたとしても更新が難しいからだと考えられる．

6.2.2 コスト最小の割当てに関する実験

二つ目の評価は，提案手法がどれだけ良い質の実行可能解を得られたかである．具体的には，問題の上界値（求められた最良の目的関数値）/最適値で定義される実行可能解の質が最も良い解法はどれかを比較した．比較対象は，2 段階最適化手法 2 段階目の DisLRP-DA$_{ph2}$ 及び DisLRP-IBF$_{ph2}$ とペナルティコスト最適化手法の DisLRP-DA$_{pnt}$ 及び DisLRP-IBF$_{pnt}$ である．なお，2 段階目最適化手法 2 段階目は 1 段階目において最適なレイヤーが求められた問題例のみを扱った．また，ペナルティコスト最適化手法では，目的関数のペナルティ項を取り除き，コスト p_{kj} の項のみで解の質を評価した．実験結果を図 4 に示す．縦軸が解の質を表しており，1 に近づけば近づくほど良い．

図 4 より，全てのペナルティコスト r において，平均値においても中央値においても，DisLRP-IBF の方が DisLRP-DA よりも良い解の質が得られていることが分かる．また，r の値が小さいほど解の質が良くなる傾向も読み取れる．

r の値が小さいほど解の質が良くなる理由については，r の値が小さい方がラグランジュ乗数ベクトルを更新するステップ長が小さくなり，下界値の収束が速くなると考えられ，下界値をもとに計算している上界値も良くなると考えられる．

図 4 は，最適なレイヤー数が求まった問題例のみで評価を行っている．しかし，図 1 からも分かるように，最適なレイヤー数が求まった問題例の数は各手法ごとに異なる．そこで，r_{s1} の DisLRP-DA$_{pnt}$ と

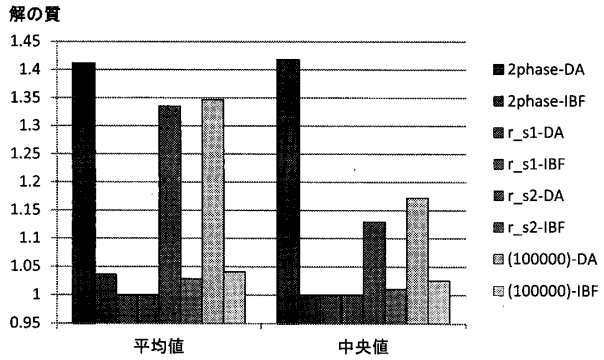


図 4 コスト最小の割当てに関する実行可能解の質の評価
Fig. 4 An evaluation for a quality of feasible solution.

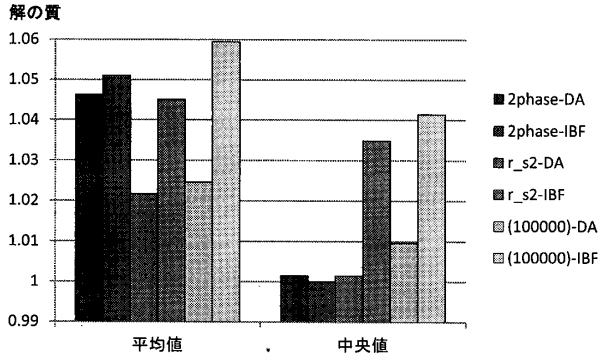


図 5 全ての解法で最適なレイヤーが求められた問題例 30 問の評価
Fig. 5 An evaluation for a quality of feasible solution (30 instances).

DisLRP-IBF$_{pnt}$ を除く(注5)全ての手法で最適なレイヤーが求められている問題例 30 問のみを抽出してグラフを作成した．その結果を図 5 に示す．図 5 より，全ての問題例の平均値において，DisLRP-DA の方が DisLRP-IBF よりも良い解の質が求められていることが分かる．これは図 4 とは逆の結果になっている．前節でも指摘したとおり，DisLRP-DA では高い容量係数（0.6 から 1.0）においても最適なレイヤー数が求まりやすいが，DisLRP-IBF は高い容量係数では最適なレイヤー数が求まりにくい．したがって，図 5 で図 4 と評価が逆転するのは，DisLRP-DA では主に高い容量係数における解の質が悪いが，DisLRP-IBF では最適なレイヤーが求まっていないため評価から除外された結果，評価が改善されたためだと考えられる．

7. むすび

GMAP の従来の定式化と過制約な問題例を扱う定

(注5)：最適なレイヤーが求まった問題例の数が他の手法に比べて極めて少ないので，評価から除外した．

式化の齟齬を解消するため，MLGMAP を新たに提案した．また，MLGMAP を解く手法として 2 段階最適化手法とペナルティコスト最適化手法の二つを提案し，それぞれに DisLRP-DA 及び DisLRP-IBF を適用した．

実験の結果，最適なレイヤーの求解及びコスト最小の割当ての求解共に，十分大きな r を用いたペナルティコスト最適化手法の disposal エージェントを用いた定式化が最も効果的であると分かった．しかし，r の値が大きいとより多くの問題例で最適なレイヤーが求められるが，得られる実行可能解の質が悪くなる傾向にある．逆に r の値が小さいと，得られる実行可能解の質は良くなるが，最適なレイヤーを求めにくくなる傾向があることが分かった．

今後の課題は，より多くの問題例で最適なレイヤーを求められるような実行可能解の改善アルゴリズムの提案が挙げられる．また，MLGMAP に対する厳密解法の提案が挙げられる．GMAP や MLGMAP に対する分散環境の厳密解法はまだ存在しない．分散環境の厳密解法は，集中環境においてメモリに乗り切らないような大規模な問題例を扱うことができるので，有用だと考えられる．

文　献

[1] P.J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo, "Adopt: asynchronous distributed constraint optimization with quality guarantees," Artifi. Intelli., vol.161, no.1-2, pp.149–180, 2005.

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

[3] M.B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," Proc. IEEE, vol.94, no.7, pp.1257–1270, July 2006.

[4] S. Bhatti and J. Xu, "Survey of target tracking protocols using wireless sensor network," 5th International Conference on Wireless and Mobile Communications, pp.110–115, Aug. 2009.

[5] K. Hirayama, "A new approach to distributed task assignment using lagrangian decomposition and distributed constraint satisfaction," Proc. 21st National Conference on Artificial Intellignce (AAAI-2006), pp.660–665, 2006.

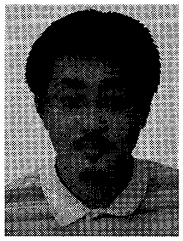
[6] K. Hirayama, "An α-approximation protocol for the generalized mutual assignment problem," Proc. 22nd AAAI Conference on Artificial Intelligence (AAAI-2007), pp.744–749, 2007.

[7] K. Hirayama, T. Matsui, and M. Yokoo, "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.

[8] K. Hanada and K. Hirayama, "Distributed lagrangian relaxation protocol for the over-constrained generalized mutual assignment problem," Proc. 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174–186, 2011.

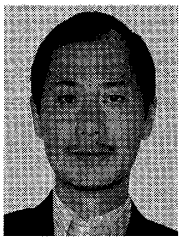
[9] J.E. Beasley, "Welcome to or-library." http://people.brunel.ac.uk/~mastjjb/jeb/info.html

（平成 25 年 2 月 10 日受付，6 月 25 日再受付）



花田　研太

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．2013 年同大大学院海事科学研究科海事科学専攻博士前期課程卒．現在，同専攻博士後期課程在学中．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　（正員）

1995 年大阪大学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）．2013 年神戸大学大学院海事科学研究科教授．1999 年–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．情報処理学会，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．