分散ラグランジュ緩和プロトコルとその応用に関する研究

花田，研太

(Degree)
博士（工学）

(Date of Degree)
2016-03-25

(Date of Publication)
2017-03-01

(Resource Type)
doctoral thesis

(Report Number)
甲第6664号

(URL)
https://hdl.handle.net/20.500.14094/D1006664

博士論文

分散ラグランジュ緩和プロトコルとその応用に関する研究

(Distributed Lagrangian Relaxation Protocol and its Applications)

平成28年1月

神戸大学大学院 海事科学研究科

花田　研太

内容梗概

分散最適化問題は，地理的に分散した情報を保持した複数のエージェントが協調的に解の探索を行うことによって，大域的な最適値および最適解を求める問題である．本研究では，エージェント間の局所通信のみによって分散最適化問題を解く分散ラグランジュ緩和プロトコル (DisLRP: Distributed Lagrangian Relaxation Protocol) に注目する．DisLRP は分散最適化問題を解くためのヒューリスティック解法であり，エージェント間の通信や処理方法を規定したプロトコルである．DisLRP の特徴は，どのような問題に対しても，分散最適化問題が最大化問題の場合は最適値に対する良い質の上界値，最小化問題の場合は最適値に対する良い質の下界値をプロトコルの変更無しに求めることが出来ることである．また，問題に対して適切なアルゴリズムの実装を行えば，良い質の実行可能解（分散最適化問題が最大化問題の場合は下界値，最小化問題の場合は上界値）を求めることも可能である．

本研究の目的は， (1) DisLRP に新しいアルゴリズムを適用し，最大化問題に対するより良い質の上界値（最小化問題に対するより良い質の下界値）を導出できるようにすることで DisLRP の性能を向上させること，(2) 分散最適化問題に対する新しい定式化を提案し，その定式化に対する実行可能解（上界値または下界値）を求めるためのアルゴリズムを DisLRP に適用して，その有効性を検証すること，(3) DisLRP を応用して実社会の問題に対する新しいアプリケーションを提供することの 3 点で構成される．

まず本研究では，バンドル法と DisLRP を組み合わせた新しいプロトコルであるバンドル法を用いた分散ラグランジュ緩和プロトコル (BDisLRP: Bundle DisLRP) を提案する．この研究の目的は，目的 (1) に該当する．DisLRP では，Dantzig-Wolfe 分解またはラグランジュ分解と呼ばれるテクニックを用いて原問題を部分問題に分解し，ラグランジュ乗数と呼ばれるパラメータを変更しながら部分問題を繰り返し解くことによって，原問題の最良の上界値または下界値を探索する．この問題はラグランジュ双対問題と呼ばれる．従来の DisLRP では，ラグランジュ双対問題の最適値（最良の上界値または下界値）を探索するアルゴリズムとして劣勾配法を使用していた．劣勾配法では，エージェント間の局所的な同期のみでラグランジュ乗数を更新することができ，大域情報が全エージェントに行き渡るのを待つ必要がないという利点がある．しかし，劣勾配法は計算コストが小さい反面，上界値または下界値がラグランジュ双対問

題の最適値に収束したかどうかを判定できない，解の振動が発生するなどの問題がある．そこで本研究では，ラグランジュ乗数の更新にバンドル法を用いる．既存のバンドル法――安定化切除平面法は，集中環境において目的関数が凸関数である無制約最適化問題を解く効率的なアルゴリズムである．しかし，バンドル法を分散環境で使用した先行研究は存在しない．分散環境でバンドル法を用いる場合，ラグランジュ乗数を更新するためには必ず大域情報が必要となる．従って，エージェント間の局所的な同期のみではラグランジュ乗数を更新することができず，大域情報が全エージェントに行き渡るのを待つ必要がある．そこで本研究では，すでに行き渡った古い大域情報でラグランジュ乗数の更新を行うことによってプロトコルの遅延発生を防ぎ，最適値の効率的な探索を可能とした．本研究では，Dantzig-Wolfe 分解を用いた定式化で表現される一般化相互割当問題 (GMAP: Generalized Mutual Assignment Problem) と，ラグランジュ分解を用いた定式化で表現される重み付き最大充足化問題 (WMax-SAT: Weighted Maximum Satisfiabilit Problem) の問題例を用いて実験を行い，BDisLRP と従来の DisLRP との間で比較を行った．その結果，GMAP の問題例を用いた実験では，大域情報収集の時間遅れがあったとしても，BDisLRP が劣勾配法を使用した従来の DisLRP に比べて良い質の上界値が得られることが分かった．また，WMax-SAT の問題例を用いた実験では，制約式の分割方法によって得られる下界値の質が大きく変わり，問題例の構造に強く依存することが分かった．

次に本研究では，従来の GMAP では扱えない問題例を扱えるようにした過制約な GMAP (OC-GMAP: Over-Constrained GMAP) を提案し，OC-GMAP の実行可能解導出に対応する 2 つの DisLRP を提案する．この研究の目的は，目的 (2) に該当する．GMAP は，オペレーションズリサーチの分野で古くから研究されている一般化割当問題 (GAP: Generalized Assignment Problem) を拡張した問題であり，財をも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．従来の GMAP では，エージェントは財を割り当てるのに十分な資源容量を持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持たない，いわゆる過制約な状況も当然考えられる．そこで，エージェントの資源容量が著しく少ない過制約な最大化問題を新たに 2 つの方法で定式化し，それぞれの定式化に対する実行可能解を導出する DisLRP をそれぞれ開発した．

1 つ目の定式化では，割り当てられない財を全て引き受ける disposal エージェントを新たに加え，全ての財は disposal エージェントを含めてただ 1 つのエージェントに割り当てられる．この定式化で実行可能解を求めるための DisLRP を，disposal エージェントを用いた DisLRP (DisLRP-DA: DisLRP with a Disposal Agent) と呼ぶ．disposal エージェントは資源制約がなく（資源容量が無限大），仮に財が disposal エージェントに割り振られても得られる効用は 0 である．disposal エージェントの導入は，既存の手法を変更無しに適用できるというメリットがあるが，一方でエージェントが 1 つ増えることによる通信量増大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，仮想化することが可能である．これにより，エージェント数を disposal エージェント導入前と同じ数にすることができ，通信量増大を回避することができる．2 つ目の定式化では，割当制約と呼ばれる制約式を不等式として記述して問題を解く．すなわち，全ての財がただ 1 つのエージェントに選ばれるか，全くエージェントに選ばれないという制約に変更するものである．この定式化で実行可能解を求めるための DisLRP を，不等式制約を用いた DisLRP (DisLRP-IBF: DisLRP with Inequality-Based Formulation) と呼ぶ．DisLRP-IBF では，定式化そのものが変更されているため，実行可能解を求める手順に対して若干の変更が必要になる．OC-GMAP は本研究で初めて扱う問題であり比較対象が存在しないため，DisLRP-DA と DisLRP-IBF の性能を比較する実験を行った．エージェントの資源容量を加工した 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に対して有効であることがわかった．

次に本研究では，従来の GMAP と OC-GMAP の齟齬を解消する多層一般化相互割当問題 (ML-GMAP: Multi-Layered Generalized Mutual Assignment Problem) を提案し，ML-GMAP の実行可能解導出に対応する DisLRP を提案する．この研究の目的は，目的 (2) に該当する．GMAP と OC-GMAP には 2 つの問題点がある．まず，OC-GMAP において，従来の GMAP の問題例をどう扱うかという問題点である．OC-GMAP のプロトコル（DisLRP-DA もしくは DisLRP-IBF）で通常の GMAP の問題例を解くと，従来の DisLRP で解いた最適値及び最適解とは違う答えを導出する可能性がある．また，OC-GMAP のプロトコルで最小化問題を解くと最適値が必ず 0 となり，意味のある答えを導出することができない．そこで ML-GMAP では，これらの問題点を解消するために，まず割り当てられ

ない財をできるだけ減らした上で，コスト最小（または効用最大）となる財の割当てを求める．本研究では，割り当てられない財の数の最小化を行う定式化とコスト最小となる財の割当てを行う定式化をそれぞれ行い，2 つの定式化を用いて ML-GMAP を解く 2 段階最適化手法と，割り当てられない財の数の最小化と財の割当てコスト最小化を 1 つの定式化で行うペナルティコスト最適化手法の 2 つを提案する．また，DisLRP-DA と DisLRP-IBF に基づいて，2 段階最適化手法に対する DisLRP とペナルティコスト最適化手法に対する DisLRP をそれぞれ開発した．ベンチマーク問題例を用いて両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の最小化における達成度と財の割当てコスト最小化の達成度にトレードオフの関係があることがわかった．

最後に，本研究では実社会における分散最小化問題として増床計画付き患者搬送問題 (PTP: Patient Transportation Problem with bed capacity management) を新たに提案し，PTP の実行可能解導出に対応する DisLRP を提案する．この研究の目的は，目的 (3) に該当する．PTP の目的は，救急車による患者の搬送コストの最小化と，受け入れ病院がベッドを増床した場合にかかるコストの最小化を同時に行うことである．本研究では，Dantzig-Wolfe 分解を用いた定式化を使用し，GMAP と施設配置問題を組み合わせて PTP の定式化を行った．また，PTP の実行可能解を求めるアルゴリズムを開発して DisLRP に適用した．本研究は比較できる先行研究が存在しないため，ラグランジュ乗数の更新に大域情報を必要としない DisLRP_L と，乗数の更新に大域情報を必要とする ADisLRP の間で比較を行った．病院の増床コストと単位病床数の設定を付加した GAP のベンチマーク問題例で実験を行った結果，DisLRP_L の方が少ない通信量でより良い質の実行可能解を早く導出できることが分かった．

以上より，DisLRP は様々な分散最適化問題に対して適用可能であり，それぞれの問題に対して良い質の上界値または下界値を得られることが実験的に確かめられた．

目 次

第1章 序論

1.1 はじめに

分散協調問題解決 (Distributed Corporative Problem Solving) は，人工知能やマルチエージェントシステムの分野において，複数のロボット間における協調動作の決定や複数の組織間における意思決定といった応用例が存在する，重要な研究課題の 1 つである．分散協調問題解決では，ある 1 つの問題を複数の**エージェント** (Agent) が協力し合って解くことを目的としている．エージェントとは，実問題におけるロボットや組織を抽象化した呼称である．

分散協調問題解決を導入する動機は，主に 2 つある．

1 つ目は，非常に大きな問題例を扱えることである．具体的には，ある問題の問題例を計算機で解かせようと考えたとき，そもそも問題例のファイル容量が大きすぎてメモリに乗り切らないような状況を指す．分散協調問題解決の手法を導入すれば，問題例を複数の計算機に分割して割当て，協調的に問題を解くことが可能となる．

2 つ目は，プライバシーの確保である．集中型の問題解決手法においては，分散された情報は 1 カ所に集められる．しかし，例えばあるエージェントが他のエージェントに公開したくない情報が存在したとすれば，それはプライバシーを侵害されたことになる．分散協調問題解決を用いて適切なプロトコルを使用すれば，そのようなプライバシーの侵害は発生しない．

この分野での最も古い例として，契約ネットプロトコル [Smith 90] が挙げられる．契約ネットプロトコルは，エージェントが受け持つ仕事（財）をどのように割り当てるかを決定する**割当問題** (Assignment Problem) を解くためのプロトコルである．しかし，契約ネットプロトコルは欲張り型探索を行なうため，一般に割当結果の最適性については何も保証しない．

契約ネットプロトコルで扱える割当問題は，ある財をあるエージェントに割り当てたとき，他の財をそのエージェントが受け持つことができないといった制約が存在しない．このような制約が存在する場合は，分散制約充足問題 (DisCSP: Distributed Constraint Satisfaction Problem) として定式化できる．DisCSP を解くためのプロトコルとして，非同期バックトラッキングアルゴリズム (ABT: Asynchronous Backtracking. Algorithm) が提案されている [Yokoo 98]．ABT は，1 つの決定変数が 1 つのエージェ

ントであり，制約は1対1のエージェント間に記述される．ABTは全ての制約を満たした決定変数の割当てが見つかれば，アルゴリズムを終了する．また，DisCSPを最適化問題へ拡張した分散制約最適化問題 (DCOP: Distributed Constraint Optimization Problem) が提案されており，それを解くプロトコルとしてADOPT[Modi 05]やBnB-ADOPT[Yeoh 08]が提案されている．

DisCSPやDCOPは汎用的な定式化を持ち，様々な実問題に対して応用されている．一方で，数理計画モデルとして定式化できる組合せ最適化問題を分散環境で解く手法として，**分散分枝限定法** [佐々木 05]や**分散ラグランジュ緩和プロトコル** (DisLRP: Distributed Lagrangian Relaxation Protocol) [Hirayama 06, Hirayama 07, Hirayama 09]などが提案されている．

分散分枝限定法は，組合せ最適化問題に対する厳密解法である．しかし，そもそも分散分枝限定法は非常に大きな問題例を扱うことだけを想定しており，負荷分散の手法として分散協調問題解決が用いられるに過ぎない．従って，プライバシーは考慮されない場合が多い．

一方，DisLRPは組合せ最適化問題に対する近似解法である．DisLRPは，組合せ最適化問題として定式化される一般化相互割当問題 (GMAP: Generalized Mutual Assignment Problem) を解くための手法として提案された．

DisLRPの特徴は，Dantzig-Wolfe分解またはラグランジュ分解というテクニックを用いて組合せ最適化問題を部分問題へ分割し，各部分問題をエージェントが担うことである．すなわち，非常に大きな問題例に対しても，分割数を増やす，すなわちエージェント数を増やすことによって近似解の導出が可能となる．また，各部分問題を解くための情報をそのエージェントのみが保持しているとみなせば，エージェントは外部のエージェントに対して情報を送る必要が無く，情報のプライバシー性が保たれる．従って，DisLRPは分散協調問題解決を導入する2つの動機を満たしていると言える．

また，組合せ最適化問題は一般にNP困難に属し，実用時間内に最適値及び最適解を求めるのは困難とされる．しかし，実際には，実用時間内に精度の良い近似解が得られれば十分という場合も多い．従って本研究では，分散環境で質の良い近似解を導出できるDisLRPに注目する．

1.2 研究の目的

本研究の目的は，次の 3 点で構成される．

1. 1 つ目の目的は，DisLRP そのものの性能を向上させることである．これは，DisLRP のに対して新しいアルゴリズムを適用することによって，得られる上界値の質または下界値の質を向上させることにより達成される．

2. 本研究の 2 つ目の目的は，分散最適化問題の新しい定式化を提案し，提案した新しい定式化に対する実行可能解（最小化問題に対する上界値または最大化問題に対する下界値）を求めるアルゴリズムを DisLRP に適用して，その有効性を確かめることを目的とする．

3. 3 つ目の目的は，DisLRP を実社会の問題に対して応用することである．実社会の問題に対して新しい定式化を提案し，提案した新しい定式化に対する実行可能解（最小化問題に対する上界値または最大化問題に対する下界値）を求めるアルゴリズムを DisLRP に適用して，その有効性を確かめる．

1.2.1 分散ラグランジュ緩和プロトコルの性能向上

DisLRP では，Dantzig-Wolfe 分解またはラグランジュ分解と呼ばれるテクニックを用いて原問題を部分問題に分解し，ラグランジュ乗数と呼ばれるパラメータを更新しながら部分問題を繰り返し解くことによって原問題の最良の上界値（最大化問題）または下界値（最小化問題）を探索する．この問題はラグランジュ双対問題と呼ばれる．従来の DisLRP では，最良の上界値または下界値，すなわちラグランジュ双対問題の最適値を探索するアルゴリズムとして劣勾配法を使用していた．劣勾配法では，エージェント間の局所的な同期のみでラグランジュ乗数を更新することができ，大域情報が全エージェントに行き渡るのを待つ必要がないという利点がある [Hirayama 06]．しかし，劣勾配法は計算コストが小さい反面，上界値または下界値がラグランジュ双対問題の最適値に収束したかどうかを判定できない，解の振動が発生するなどの問題がある．

本研究は，劣勾配法の代わりにバンドル法 [Schramm 92] を使用した新しいプロトコルであるバンドル法を用いた分散ラグランジュ緩和プロトコル (BDisLRP: Bundle DisLRP) を提案し，バンドル法が分散環境にも適

用可能であることを示す．BDisLRP は，最大化問題に対する上界値または最小化問題に対しル下界値を求めることに特化したプロトコルである．

バンドル法は安定化切除平面法とも呼ばれ，目的関数が凸（凹）関数である無制約最適化問題を解く効率的なアルゴリズムである．バンドル法では，ラグランジュ乗数を更新するために必ず大域情報が必要となる．従って，エージェント間の局所的な同期のみではラグランジュ乗数を更新することができず，大域情報が全エージェントに行き渡るのを待つ必要がある．そこで本研究では，すでに行き渡った古い大域情報でラグランジュ乗数の更新を行うことによってプロトコルの遅延発生を防ぎ，最適値の効率的な探索を可能とした．本研究では，大域情報収集の時間遅れがあったとしても，BDisLRP が劣勾配法を使用した従来の DisLRP に比べて良い質の上界値を得ることを実験的に確かめる．

1.2.2 分散最適化問題に対する新しい定式化の提案と DisLRP の適用

1.2.2.1 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 前述の通り，DisLRP は GMAP を解くための手法として提案された．GMAP は，オペレーションズリサーチの分野で古典的な問題の 1 つである一般化割当問題 (GAP: Generalized Assignment Problem) [Savelsbergh 97] を分散環境へと拡張した問題である．

GMAP は，財の集合をエージェントに割り当てることを考える．問題の目的は，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），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，どの財もエージェントに割り当てられる，または，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．

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も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もたないような，いわゆる過制約な状況もあり得ると考えられる．本研究では，このような過制約な状況に対処する新たな問題を**過制約な一般化相互割当問題** (OC-GMAP: Over-Constrained GMAP) として提案する．また，OC-GMAP に対応した 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．

1 つ目は，disposal エージェントを用いた DisLRP (DisLRP-DA: DisLRP with a Disposal Agent) である．DisLRP-DA の基本的なアイデアは，資源

制約のない（無限の資源をもつ）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，他の全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．

2 つ目は，不等式制約緩和に基づく DisLRP (DisLRP-IBF: DisLRP with Inequality Based Formulation) である．DisLRP-IBF では，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すむ反面，従来の定式化を変更することに伴い 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 点が挙げられる．

1.2.2.2 多層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

OC-GMAP では，エージェントを追加，も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 GMAP では実行可能解が存在しないような過制約な問題例にも対処できるようになった．しかし OC-GMAP には，許容領域を変更することによって生じる問題点が 2 つある．

1 つ目は，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つ目は，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2つの問題点に対処できる多層一般化相互割当問題 (ML-GMAP: Multi-Layered GMAP) を新たに提案する．ML-GMAP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．ML-GMAPの目的は，レイヤー数が最小という条件の下で，効用の総和が最大，も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．

1つ目の手法は2段階最適化手法である．この手法では，ML-GMAP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，もしくはコストが最小となる財の割当てを求める．

2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，も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．

1.2.2.3 増床計画付き患者搬送問題に対する分散ラグランジュ緩和プロトコル

最後に，本研究ではDisLRPを実社会の問題に対して応用することを目的として，増床計画付き患者搬送問題 (PTP: Patient Transportation Problem with Bed Capacity Management) を提案する．大規模な災害が発生した場合，多数の負傷者が発生する．これらの負傷者は速やかに病院へ搬送し，適切な治療を施す必要がある．その際，どの患者をどの病院へ搬送するか――割り当てるかが重要な問題となる．また，大規模な災害時においては，現在の病院の病床数では足りないほどの負傷者が発生する可能性があると考えられる．通常，病院における病床数は医療法や

厚生労働省による省令によって定められており，簡単に増床することはできない．しかし，災害拠点病院と呼ばれる病院においては，災害時などにおいては特例的に増床することが可能である．

そこで本研究は，患者の搬送計画と病院の増床計画を同時に考慮するPTPを提案する．PTPは，GMAPと施設配置問題 (Facility Location Problem) を組み合わせることによって汎用的な問題として定式化が可能である．また，増床計画付き患者搬送問題に対する実行可能解を求めるアルゴリズムをDisLRPに適用する．

この問題では，災害に対応するためにより早くより良い質の実行可能解が求められることが重要である．また，大規模な災害が発生した場合，通信インフラの障害などが発生することが予想され，通信量はできるだけ少ない方が良いと考えられる本研究では，DisLRPがPTPに対して少ない通信量と短い実時間で良い質の実行可能解が求められることを，実験的に確かめる．

1.3 論文の構成

本論文は次のように構成される．

第2章では，DisLRPの概要について述べる．DisLRPで扱える問題について汎用的な定式化を与え，本研究で使用される基本的な数学的概念を概説する．また，具体的な定式化の例としてGMAPとWMax-SATを挙げる．また，プロトコルの基本的な動作について説明し，DisLRPの内部で使用されているアルゴリズムについて述べる．

第3章では，バンドル法を用いたDisLRPについて述べる．まず，汎用的な定式化を用いてバンドル法の概要について述べる．また，バンドル法を用いた新しいプロトコルについて説明する．最後に，GMAPとWMax-SATの問題例を用いた実験結果を示し，既存のプロトコルと比較，評価する．

第4章では，OC-GMAPについて述べる．まず，既存のGMAPに対して過制約な状況を定義し，OC-GMAPの定式化を与える．次に，DisLRPをベースとしたOC-GMAPを解く2つのプロトコル，DisLRP-DAとDisLRP-IBFについてそれぞれ述べる．本研究では，比較できる先行研究が存在しないため，DisLRP-DAとDisLRP-IBFの間で実験的な比較を行い，結果を考察する．

第5章では，ML-GMAPについて述べる．まず，既存のGMAPとOC-

GMAP の齟齬について述べ，ML-GMAP の目的の定義を行う．次に，ML-GMAP の目的を達成するための 2 つの手法として，2 段階最適化手法とペナルティコスト最適化手法について，それぞれ述べる．本研究では，比較できる先行研究が存在しないため，2 段階最適化手法とペナルティコスト最適化手法の間で実験的な比較を行い，結果を考察する．

第 6 章では，PTP について述べる．まず，PTP に対する定式化について述べる．次に，DisLRP をベースとした PTP を解くためのプロトコルを提案する．本研究では比較できる先行研究が存在しないため，ラグランジュ乗数の更新方法が異なる 2 つの DisLRP で実験を行い，結果を考察する．

第 7 章では，本論文のまとめと今後の課題を示す．

第2章　分散ラグランジュ緩和プロトコル

2.1　はじめに

分散ラグランジュ緩和プロトコル (DisLRP: Distributed Lagrangian Relaxation Protocol) は，組合せ最適化問題を分散環境で解くための処理と通信を規定したプロトコルである．本節では，本研究で用いる DisLRP の概要について述べる．

まず，DisLRP が適用可能な 2 種類の組合せ最適化問題を定義する．1 つ目は，Dantzig-Wolfe 分解を用いた定式化である．2 つ目は，ラグランジュ分解を用いた定式化である．また，DisLRP で用いられる命題や定理についても述べる．

次に，DisLRP におけるエージェントの基本的な振る舞いについて述べる．DisLRP では，大域情報の収集といった全エージェントが関係する動作についても，局所通信のみによって行われる．その仕組みについて概説する．

最後に，既存の DisLRP のエージェントが内部で使用するアルゴリズムである劣勾配法について述べ，劣勾配法を使用することによる問題点を指摘する．

2.2　問題の定義

本節では，まず DisLRP が適用可能な 2 つ定式化を示す．

1 つ目は，Dantzig-Wolfe 分解を用いた定式化である．本研究で扱う一般化相互割当問題 (GMAP: Generalized Mutual Assignment Problem)，過制約な一般化相互割当問題 (OC-GMAP: Over-Constrained Generalized Mutual Assignment Problem)，多層一般化相互割当問題 (ML-GMAP: Multi-Layered Generalized Mutual Assignment Problem)，増床計画付き患者搬送計画問題 (PTP: Patient Transportation Problem with bed capacity management) の定式化は，この手法が用いられる．

2 つ目は，ラグランジュ分解を用いた定式化である．本研究で扱う重み付き制約最大化問題 (WMax-SAT: Weighted Maximum Satisfiabilit Problem) は，この手法が用いられる．

なお，本章では全て最大化問題として記述するが，一般に最小化問題にも適用可能である．

2.2.1　共通の表記

本節では，Dantzig-Wolfe 分解を用いた定式化及びラグランジュ分解を用いた定式化の両方で用いられる表記について述べる．

定義 1 集合 A, J_i, K_i, L，ベクトル x_i を次のように定義する．

- $A = \{1, \ldots, m\}$ は，エージェントの識別子集合である．
- $J_i = \{1, \ldots, n_i\}$ は，エージェント $i \in A$ の決定変数の識別子集合である．$x_i = (x_{i1}, \ldots, x_{in_i}) \in D_i^{n_i}$ は，エージェント i の決定変数ベクトルである．ここで，$x_{ij} \in D_{ij}$ $(j \in J_i)$ はエージェント i の決定変数，D_{ij} は決定変数 x_{ij} の値域集合である．エージェントは，変数 x_{ij} の値を値域 D_{ij} から選択する．なお，値域 D_{ij} は連続集合でも離散集合でもよい．
- $K_i = \{1, \ldots, o_i\}$ は，エージェント i の内部制約関数の識別子集合である．$C_i = \{C_i^1, \ldots, C_i^{o_i}\}$ は，エージェント i が保持する内部制約関数 $C_i^k : D_i^{n_i} \to \mathbb{R}$ $(k \in K_i)$ の集合である．
- $L = \{1, \ldots, p\}$ は，エージェント間制約関数の識別子集合である．$C_{trans} = \{C_{trans}^1, \ldots, C_{trans}^p\}$ は，全エージェントまたは一部のエージェントが保持するエージェント間制約関数 $C_{trans}^l : D_1^{n_1} \times \cdots \times D_m^{n_m} \to \mathbb{R}$ の集合である．また，$C_{trans}^{i,l}$ を $C_{trans}^{i,l} : D_i^{n_i} \to \mathbb{R}$ と定義し，

  $$C_{trans}^l(x_1, \ldots, x_m) = \sum_{i \in A} C_{trans}^{i,l}(x_i),$$

  という線形結合で表せるものとする．

2.2.2　Dantzig-Wolfe 分解を用いた定式化

Dantzig-Wolfe 分解は，エージェントが個々に保持する制約とエージェント間にまたがる制約が存在する場合，エージェント間にまたがる制約式を緩和し，主問題を部分問題へと分解する手法である．

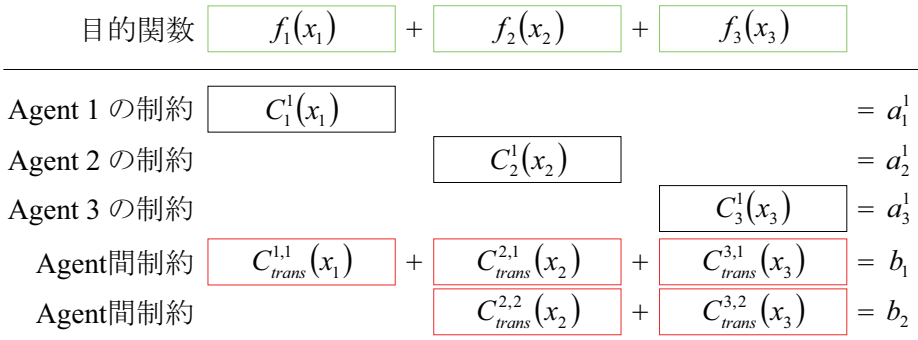

図 2.1: Dantzig-Wolfe 分解の制約構造の例

2.2.2.1　主問題　Dantzig-Wolfe 分解を用いた主問題 ($\mathcal{P}_{\mathcal{DW}}$) は，次のように表される．

$$
\begin{aligned}
\mathcal{P}_{\mathcal{DW}} \quad & (\text{decide } x_i,\ \forall i \in A) : \\
\max. \quad & \sum_{i \in A} f_i(x_i) && (2.1) \\
\text{s. t.} \quad & x_i \in X_i, \forall i \in A, \\
& (x_1, \ldots, x_m) \in X_{trans}. && (2.2)
\end{aligned}
$$

ここで，f_i はエージェント i が保持する目的関数 $f_i : D_i^{n_i} \to \mathbb{R}$，$X_i$ は $\{x_i | C_i^k(x_i) = a_i^k, \forall k \in K_i\}$，$X_{trans}$ は $\{(x_1, \ldots, x_m) | C_{trans}^l(x_1, \ldots, x_m) = b_l, \forall l \in L\}$，$a_i^k, b_l \in \mathbb{R}$ は所与のスカラーである．

なお，エージェント内制約やエージェント間制約が不等式で表されたとしても，等式標準形に変形することが可能であるため一般性は失われない．

図 2.1 に，Dantzig-Wolfe 分解を適用できる問題の例を示す．図 2.1 の例では，エージェント間制約が 2 つ存在する．図から分かる通り，エージェント間制約は全てのエージェントにまたがっている必要は無い．また，エージェント間制約を取り除けば，制約はブロック構造を持つことが分かる．

2.2.2.2　ラグランジュ緩和問題　主問題 ($\mathcal{P}_{\mathcal{DW}}$) の特徴は，目的関数がエージェント間の線形結合で表されることである．従って，エージェント間制約 (2.2) を取り除けば，主問題 ($\mathcal{P}_{\mathcal{DW}}$) を部分問題へと分解することが出来る．

DisLRP では，エージェント間制約 (2.2) をラグランジュ緩和する．ラグランジュ緩和とは，取り除く制約式が違反した場合，目的関数にペナルティコストを足し込むことで取り除いた制約式をできるだけ満たす手法のことを指す．

エージェント間制約 (2.2) をラグランジュ緩和して得られるラグランジュ緩和問題 ($\mathcal{D}_{\mathcal{DW}}$) は，次のように表される．

$$\begin{aligned} \mathcal{D}_{\mathcal{DW}}(\mu) \quad & (\text{decide } x_i,\ \forall i \in A) : \\ \max. \quad & \sum_{i \in A} f_i(x_i) + \sum_{l \in L} \mu_l \left(b_l - C^l_{trans}(x_1, \ldots, x_m) \right) \\ \text{s. t.} \quad & x_i \in X_i, \forall i \in A. \end{aligned}$$

ここで，$\mu_l \in \mathbb{R}$ はエージェント間制約 (2.2) に対する実数値パラメータで，エージェント間制約 (2.2) の**ラグランジュ乗数**と呼ばれる．また，$\mu = (\mu_1, \ldots, \mu_l)$ はラグランジュ乗数ベクトルと呼ばれる．

エージェント間制約 (2.2) はエージェントごとの決定変数を用いた線形結合で表すことができるので，ラグランジュ緩和問題 ($\mathcal{D}_{\mathcal{DW}}$) は次のように書き換えることが出来る．

$$\begin{aligned} \max. \quad & \sum_{i \in A} \left(f_i(x_i) - \sum_{l \in L} \mu_l \cdot C^{i,l}_{trans}(x_i) \right) + \sum_{l \in L} \mu_l b_l \\ \text{s. t.} \quad & x_i \in X_i, \forall i \in A. \end{aligned}$$

目的関数 (2.1) もエージェント間制約 (2.2) と同様に線形結合で表すことが出来るので，ラグランジュ緩和問題 ($\mathcal{D}_{\mathcal{DW}}$) をエージェントごとに分解することが出来る．分解後のエージェント i の部分問題は，次のように表すことが出来る．

$$\begin{aligned} \max. \quad & f_i(x_i) - \sum_{l \in L} \mu_l \cdot C^{i,l}_{trans}(x_i) + \sum_{l \in L} \frac{\mu_l b_l}{m} \\ \text{s. t.} \quad & x_i \in X_i. \end{aligned}$$

図 2.2 に，図 2.1 の例題をラグランジュ緩和した例を，図 2.3 にエージェント 1 が受け持つラグランジュ緩和問題の部分問題をそれぞれ示す．図 2.2 おいて，2 つのエージェント間制約関数が取り除かれ，それぞれラグランジュ乗数 μ_1, μ_2 を乗じたものが目的関数に追加されている．図 2.3 においては，エージェント 1 は 2 つ目のエージェント間制約とは関係ないため，1 つ目のエージェント間制約に関する部分のみが目的関数に追加されていることが分かる．

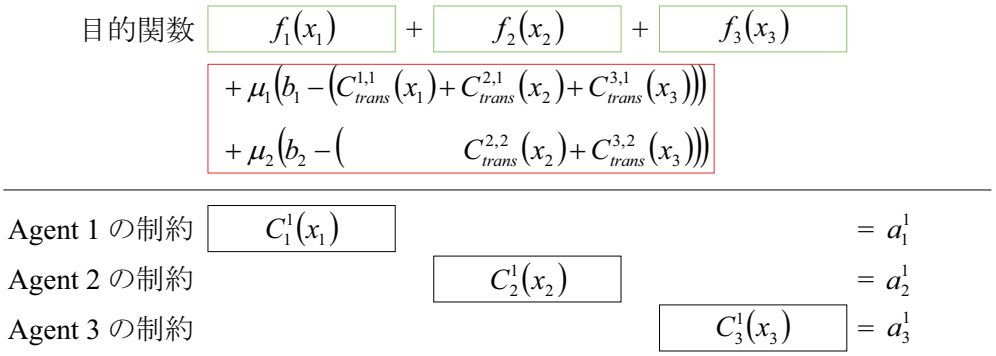

図 2.2: ラグランジュ緩和問題

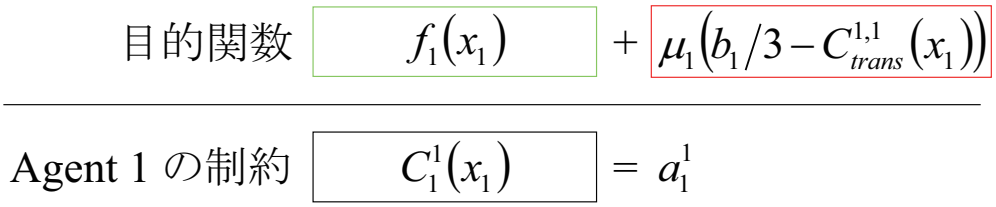

図 2.3: エージェント 1 が受け持つラグランジュ緩和問題

2.2.2.3　例：一般化相互割当問題

本節では，Dantzig-Wolfe 分解を用いた定式化の例として，一般化相互割当問題 (GMAP: Generalized Mutual Assignment Problem) を示す [Hirayama 06].

GMAP は，オペレーションズリサーチの分野で古典的な問題の 1 つである一般化割当問題 (GAP: Generalized Assignment Problem) [Savelsbergh 97] を分散環境へと拡張した問題である．GAP は分散環境におけるマルチロボットタスク割当 [Luo 13] などに応用されており，GAP を拡張した GMAP は多くの実用問題に適用できると考えられる．例えば，無線ネットワークに関する資源管理問題を GMAP にモデル化して解くという研究が報告されている [Aristomenopoulos 12].

GMAP は，財の集合をエージェントに割り当てることを考える．問題の目的は，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），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，どの財もエージェントに割り当てられる，または，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．

GMAP を 0-1 整数計画問題 (IP 問題) として定式化すると，以下のようになる．

$$
\begin{aligned}
\mathcal{P}_{\mathcal{DW}}^{\mathcal{GMP}} \quad & (\text{decide } x_i,\ \forall i \in A) : \\
\max. \quad & \sum_{i \in A} f_i(x_i) = \sum_{i \in A} \sum_{j \in J'} p_{ij} x_{ij} \\
\text{s. t.} \quad & x_i \in \left\{ x_i | \sum_{j \in J'} w_{ij} x_{ij} \leq c_i \right\}, \forall i \in A, \qquad (2.3) \\
& (x_1, \ldots, x_m) \in \left\{ (x_1, \ldots, x_m) | \sum_{i \in A_j} x_{ij} = 1, \forall j \in J' \right\}, \qquad (2.4) \\
& D_{ij} = \{0, 1\},\ \forall i \in A,\ \forall j \in J'.
\end{aligned}
$$

ここで，$J' = \{1, \ldots, n\} (= J_1 = \cdots = J_m = L)$ は財の集合，$A_j \subseteq A$ は財 $j \in J'$ を選択することのできるエージェントの識別子集合，p_{ij} はエージェント i に財 j を割り当てた場合の効用，w_{ij} はエージェント i に財 j を割り当てた場合の資源消費量，c_i はエージェント i の資源容量である．GMAP の場合，ナップサック制約がエージェント内制約 (2.3)，割当制約がエージェント間制約 (2.4) となる．また，値域集合 D_{ij} は，エージェント i に財 j を割り当てた場合は 1，そうでない場合は 0 を表す．

GMAP のラグランジュ緩和問題は，次のようになる．

$$
\begin{aligned}
\mathcal{D}_{\mathcal{DW}}^{\mathcal{GMP}}(\mu) \quad & (\text{decide } x_i,\ \forall i \in A) : \\
\max. \quad & \sum_{i \in A} \sum_{j \in J'} p_{ij} x_{ij} + \sum_{j \in J'} \mu_j \left(1 - \sum_{i \in A_j} x_{ij} \right) \\
\text{s. t.} \quad & x_i \in \left\{ x_i | \sum_{j \in J'} w_{ij} x_{ij} \leq c_i \right\}, \forall i \in A.
\end{aligned}
$$

$\mathcal{D}_{\mathcal{DW}}^{\mathcal{GMP}}$ をエージェントごとに分解して表すと，エージェント i の部分問題は次のように表される．

$$
\begin{aligned}
\max. \quad & \sum_{j \in J'} (p_{ij} - \mu_j) x_{ij} + \sum_{j \in J'} \frac{\mu_j}{m} \\
\text{s. t.} \quad & x_i \in \left\{ x_i | \sum_{j \in J'} w_{ij} x_{ij} \leq c_i \right\}.
\end{aligned}
$$

2.2.3　ラグランジュ分解を用いた定式化

ラグランジュ分解は，Dantzig-Wolfe 分解とは違い，制約式に特徴的な構造がない問題にも適用可能である．すなわち，ほとんどの組合せ最適化問題は，ラグランジュ分解を用いた定式化を用いれば分散環境で求解可能である．ラグランジュ分解を用いた定式化では，コピー変数と呼ばれる決定変数を用意し，制約式をエージェントが分割して受け持つことによって分解を実現する．

2.2.3.1　主問題　ラグランジュ分解を用いた主問題 ($\mathcal{P}_{\mathcal{LD}}$) は，次のように表される．

$$
\begin{aligned}
\mathcal{P}_{\mathcal{LD}} \quad & (\text{decide } x_i,\ \alpha_i^s,\ \forall i, s \in A) : \\
\max. \quad & \sum_{i \in A} f_i(x_i) \\
\text{s. t.} \quad & x_i \in X_i, \forall i \in A, \\
& (\alpha_1^i, \ldots, x_i, \ldots, \alpha_m^i) \in X_{trans}^i, \forall i \in A, \\
& x_i = \alpha_i^s, \forall i \in A, \forall s \in A \setminus \{i\}.
\end{aligned} \tag{2.5}
$$

ここで，$\alpha_i^s = \left(\alpha_{i1}^s, \ldots, \alpha_{in_i}^s\right) \in D_i^{n_i}$ はエージェント s に値の決定権があるエージェント i に関する決定変数ベクトルであり，**コピー変数**ベクトルと呼ばれる．ただし，$s = i$ の場合はエージェント i 自身の決定変数ベクトルを表すことから，$s \neq i$ として除外する．式 (2.5) はコピー制約または一致制約と呼ばれ，エージェント間にまたがっている α_i^s を全て一致させる．これにより，主問題 ($\mathcal{P}_{\mathcal{LD}}$) は主問題 ($\mathcal{P}_{\mathcal{DW}}$) と同様の最適値を得ることが出来る．

ラグランジュ分解を用いた定式化では，エージェント間制約をエージェントが分割して受け持つ．$X_{trans}^i \subseteq X_{trans}$ は，エージェント i が受け持つエージェント間制約の集合を表す．なお，$\bigcup_{i \in A} X_{trans}^i = X_{trans}$ が満たされておればよく，同じエージェント間制約を複数のエージェントが保持しても一般性は失われない．

この定式化の特徴は，エージェントが他のエージェントの決定変数をコピーして保持していることである．α_i^s はエージェント i に関係する変数であるにもかかわらずエージェント s のみが保持しており，決定権もエージェント s にある．従って，最適解を探索する過程では，$\alpha_i^s (i \neq s)$ は必ずしも x_i と一致する必要は無い．しかし，式 (2.5) の一致制約によっ

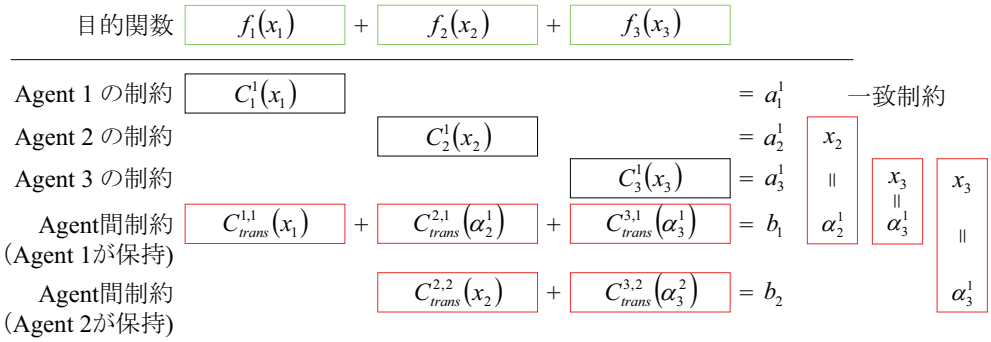

図 2.4: ラグランジュ分解の制約構造の例

て，最終的には全てのコピー変数はエージェント i による意思決定と同一の値となる．

図 2.4 に，ラグランジュ分解を適用できる問題の例を示す．図 2.4 の制約構造は，図 2.1 の例と全く同じである．前節と異なるのは，Dantzig-Wolfe 分解が 2 つのエージェント間制約を直接緩和することに対して，ラグランジュ分解は 2 つのエージェント間制約をいずれかのエージェントに保持させる．図 2.4 の例においては，1 つ目のエージェント間制約をエージェント 1 が，2 つ目のエージェント間制約をエージェント 2 が保持すると仮定する．

ここで，エージェント 1 が保持するエージェント間制約に注目する．エージェント 1 が保持するエージェント間制約には，エージェント 1，エージェント 2，エージェント 3 の決定変数が含まれている．ここに，エージェント 2 やエージェント 3 の意思に関係なく，エージェント 1 が自由に値を決定できるコピー変数 α_2^1, α_3^1 を導入する．また，エージェント間の変数の決定を同一のものとするため，エージェント 1 が保持するエージェント 2 に関する決定変数と，エージェント 2 自身が保持する決定変数を一致させる必要がある．この例では，$x_2 = \alpha_2^1$ などが，それにあたる．

2.2.3.2　ラグランジュ緩和問題　一致制約 (2.5) をラグランジュ緩和して得られるラグランジュ緩和問題 ($\mathcal{D}_{\mathcal{LD}}$) は，次のように表される．

$$
\begin{aligned}
\mathcal{D}_{\mathcal{LD}}(\lambda) \quad & (\text{decide } x_i,\ \alpha_i^s,\ \forall i, s \in A) : \\
\max. \quad & \sum_{i \in A} f_i(x_i) + \sum_{i \in A} \sum_{s \in A \setminus \{i\}} \lambda_i^s \left(x_i - \alpha_i^s\right) \\
\text{s. t.} \quad & x_i \in X_i, \forall i \in A, \\
& (\alpha_1^i, \ldots, x_i, \ldots, \alpha_m^i) \in X_{trans}^i, \forall i \in A,
\end{aligned}
$$

ここで，$\lambda = (\lambda_1^1, \ldots, \lambda_1^m, \lambda_2^1, \ldots, \lambda_2^m, \ldots, \lambda_m^m)$ は一致制約に対するラグランジュ乗数ベクトル，λ_i^s はエージェント i とエージェント s 間の一致制約に対するラグランジュ乗数である．

ラグランジュ緩和問題 ($\mathcal{D}_{\mathcal{LD}}$) をエージェントごとに分解すると，次のようになる．

$$
\begin{aligned}
\max. \quad & f_i(x_i) + \sum_{s \in A \setminus \{i\}} \lambda_i^s x_i - \sum_{s \in A \setminus \{i\}} \lambda_s^i \alpha_s^i \\
\text{s. t.} \quad & x_i \in X_i, \\
& (\alpha_1^i, \ldots, x_i, \ldots, \alpha_m^i) \in X_{trans}^i,
\end{aligned}
$$

ラグランジュ分解を用いた定式化の特徴は，分解後のラグランジュ緩和問題 ($\mathcal{D}_{\mathcal{LD}}$) に，主問題の制約式の構造が残っていることである．

図 2.5 に，例題におけるラグランジュ緩和問題を，図 2.6 にエージェント 1 が受け持つラグランジュ緩和問題の部分問題をそれぞれ示す．一致制約が緩和されたことによって，コピー変数 α_i^s は元のエージェントの決定変数である x_i に一致させる必要がなくなる．従って，コピー変数の値はラグランジュ乗数 λ_i^s の値によって変化する．また，図 2.6 には，エージェント 1 が値を決定できる決定変数 $x_1, \alpha_1^2, \alpha_1^3$ のみが残っていることが確認できる．

2.2.3.3　例 : 重み付き Max-SAT 問題　本節では，ラグランジュ分解を用いた定式化の例として，重み付き最大充足化問題 (WMax-SAT: Weighted Maximum Satisfiabilit Problem) を示す．

WMax-SAT は，知能情報学や論理学において重要な研究対象である充足可能性判定問題 (SAT: Satisfiabilit Problem) を最適化問題へ拡張した問題である．真か偽で表された論理変数をリテラルと呼び，リテラルの選言を節と呼ぶ．WMax-SAT は重み付きの標準連言形式（WCNF 形式）で与

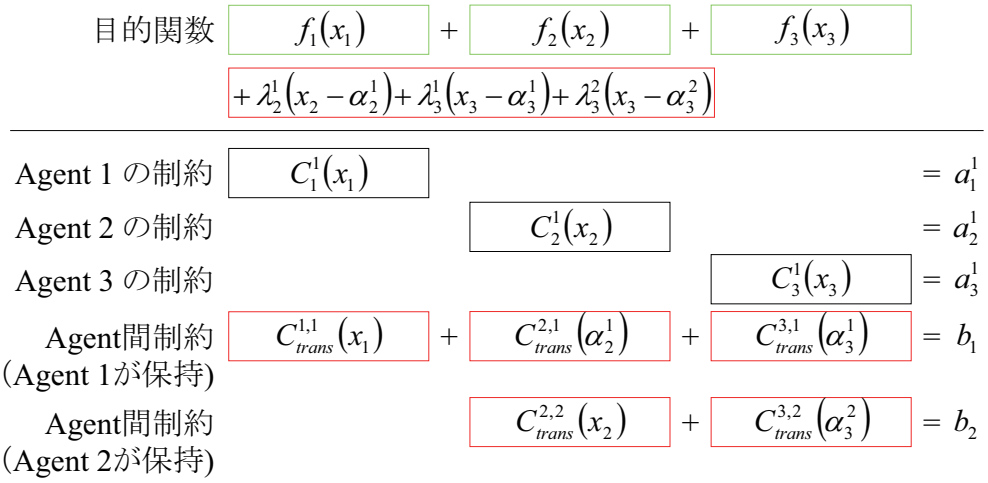

図 2.5: ラグランジュ分解を用いた定式化におけるラグランジュ緩和問題

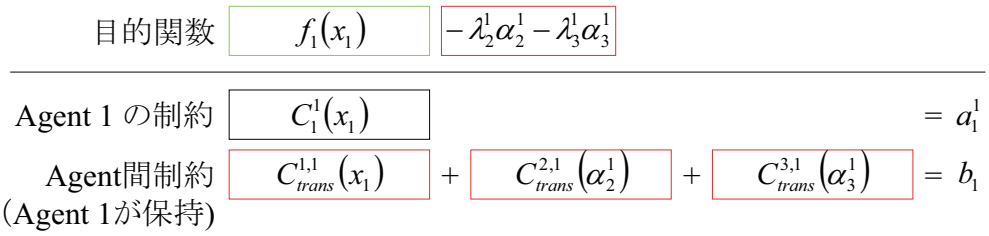

図 2.6: エージェント 1 が受け持つラグランジュ緩和問題

えられ，重みの付いた節の連言で表現される．WMax-SAT の目的は，偽となる節から発生する重みの総和を最小化することである．

ここで，WMax-SAT に対するラグランジュ分解を用いた定式化を示す前に，通常の 0-1 整数計画問題を用いた定式化を示す．

$$\begin{aligned} \text{min.} \quad & \sum_{e\in E} w_e(1-y_e) \\ \text{s. t.} \quad & \sum_{j\in L_e^+} x'_j + \sum_{j\in L_e^-} (1-x'_j) \geq y_e,\ \forall e\in E, \\ & \alpha'_j \in \{0,1\},\ \forall j \in J'', \\ & y_e \in \{0,1\},\ \forall e \in E. \end{aligned} \tag{2.6}$$

ここで，$J''=\{1,\ldots,n\}$ は論理変数の添字集合，E は節の添字集合，$L_e^+\subseteq J''$ は節 $e\in E$ に含まれる論理変数のうち真のリテラルの集合，$L_e^-\subseteq J''$

は節 e に含まれる論理変数のうち偽のリテラルの集合，$w_e \in \mathbb{N}$ は節 e が真である場合にかかる重みである．また，x'_j は論理変数 $j \in J''$ が真のとき 1，そうでない場合は 0 に設定される決定変数，y_e は節 e が真の場合に 1，そうでない場合は 0 に設定される補助決定変数である．式 (2.6) は節を表す制約式である．

ラグランジュ分解を用いた WMax-SAT の定式化では，節を m 分割してエージェントが保持すると仮定する．ここで，エージェント i が保持する節の添字集合を $E_i \subseteq E$ とする．$\emptyset \neq E_i \subset E$ とし，節はいずれか 1 つの E_i に必ず属している．すなわち，E_i は互いに素な集合である．また，エージェント i が保持する論理変数の添字集合を $J''_i \subseteq J''$ とする．ここで，J''_i にも $J''_s (s \in A)$ にも含まれる論理変数のことを共有変数と呼ぶ．

ラグランジュ分解を用いた定式化を用いると，WMax-SAT は以下のように定式化できる．

$$
\begin{aligned}
\mathcal{P}_{\mathcal{LD}}^{\mathcal{WMS}} \quad & (\text{decide } x_i,\ \forall i \in A,\ y_e,\ \forall e \in E) : \\
\min . \quad & \sum_{i \in A} f_i(x_i) = \sum_{i \in A} \sum_{e \in E_i} w_e (1 - y_e) \\
\text{s. t.} \quad & x_i \in \left\{ x_i | \sum_{j \in L_e^+} x_{ij} + \sum_{j \in L_e^-} (1 - x_{ij}) \geq y_e,\ \forall e \in E_i \right\}, \forall i \in A, \qquad (2.7) \\
& x_{ij} = x_{sj}, i < s, \forall i \in A, \forall s \in A \setminus \{i\}, \exists j \in J''_i \cup J''_s, \qquad (2.8) \\
& D_{ij} = \{0, 1\},\ \forall i \in A,\ \forall j \in J''_i. \qquad (2.9)
\end{aligned}
$$

ここで，決定変数 x_{ij} はドメインを表す式 (2.9) より，エージェント i が保持する論理変数 $j \in J''_i$ が真のとき 1，そうでない場合は 0 に設定される．

一致制約である式 (2.7) は，本来コピー変数 α_{ij}^s を用いて $x_{ij} = \alpha_{ij}^s$ となる．しかし，論理変数 j は全てのエージェントで共有しているため，エージェント s が決定すべき論理変数 j の値の割当ては，全てのエージェントに対して同じとなる．すなわち，$\alpha_{ij}^s = x_{sj}, \forall i \in A$ が成り立つ．従って，$x_{ij} = \alpha_{ij}^s = x_{sj}, \forall i \in A$ となるため，α_{ij}^s を用いずに表記している．

なお，節を表す制約である式 (2.7) はエージェント内部制約となるため，WMax-SAT はエージェント間制約が存在しない．また，補助決定変数 y_e は節 j を保持するエージェントに決定権があるものとする．

WMax-SAT のラグランジュ緩和問題は，次のようになる．

$$
\begin{aligned}
\mathcal{D}_{\mathcal{LD}}^{\mathcal{WMS}} \quad & (\text{decide } x_i,\ \forall i \in A,\ y_e,\ \forall e \in E) : \\
\min . \quad & \sum_{i \in A} \sum_{e \in E_i} w_e(1 - y_e) + \sum_{\substack{i \in A \\ (i<s, \exists j \in J_i'' \cup J_s'')}} \sum_{s \in A \setminus \{i\}} \lambda_i^s \left(x_{ij} - x_{sj}\right) \\
\text{s. t.} \quad & x_i \in \left\{ x_i | \sum_{j \in L_e^+} x_{ij} + \sum_{j \in L_e^-} (1 - x_{ij}) \geq y_e,\ \forall e \in E_i \right\}, \forall i \in A.
\end{aligned}
$$

$\mathcal{D}_{\mathcal{LD}}^{\mathcal{WMS}}$ をエージェントごとに分解して表すと，エージェント i の部分問題は次のように表される．

$$
\begin{aligned}
\min . \quad & \sum_{e \in E_i} w_e(1 - y_e) + \sum_{\substack{s \in A \setminus \{i,\ldots,m\} \\ (\exists j \in J_i'' \cup J_s'')}} \lambda_i^s x_{ij} - \sum_{\substack{s \in A \setminus \{1,\ldots,i\} \\ (\exists j \in J_i'' \cup J_s'')}} \lambda_s^i x_{ij} \\
\text{s. t.} \quad & x_i \in \left\{ x_i | \sum_{j \in L_e^+} x_{ij} + \sum_{j \in L_e^-} (1 - x_{ij}) \geq y_e,\ \forall e \in E_i \right\}.
\end{aligned}
$$

2.2.4　プロトコルで利用する命題

Dantzig-Wolfe 分解を用いた定式化およびラグランジュ分解を用いた定式化は，双対分解法を利用するという点で本質的に同じである．以降，DisLRP で解ける主問題を $(\mathcal{P})$，q 個の制約式をラグランジュ緩和したラグランジュ緩和問題を $(\mathcal{D})$ と表す．また，主問題 $(\mathcal{P})$ の最適値を Opt，あるラグランジュ乗数ベクトル $\mu \in \mathbb{R}^q$ の下でのラグランジュ緩和問題 $(\mathcal{D})$ の最適値を $L(\mu)$，エージェント i のラグランジュ緩和問題を $(\mathcal{D}_i(\pi_i(\mu)))$，$\pi_i(\mu)$ をエージェント i に関係するラグランジュ乗数とする．

ここで，プロトコルが利用する 2 つの命題について紹介する．

命題 1 任意の μ において，全てのエージェントのラグランジュ緩和問題 *$(\mathcal{D}_i)$* の最適値の和は，*$(\mathcal{P})$* の最適値の上界を与える．すなわち，

$$Opt \leq L(\mu) = \sum_{i \in A} L_i(\pi_i(\mu)),\ \forall \mu,$$

が成り立つ．

命題 2 ある μ において，全てのエージェントのラグランジュ緩和問題 *$(\mathcal{D}_i)$* の最適解を得て，かつ，それらが緩和したエージェント間制約を全て満たしているとき，それらの最適解は主問題 *$(\mathcal{P})$* の最適解を構成している．

2.2.5　ラグランジュ双対問題

一般に，ラグランジュ緩和問題を解いて得られる上界値は，原問題を解く上で重要な情報を与える．最も良い上界値を得る問題をラグランジュ双対問題と呼び，

$$L(\mu^*) = \min_{\mu} . L(\mu) = \min_{\mu} . \sum_{i \in A} L_i(\pi_i(\mu)),$$

という q 次元実ベクトル空間上の無制約最小化問題となる．ここで，μ^* はラグランジュ双対問題の最適解を表す.

ラグランジュ双対問題の目的関数 $L(\mu)$ は，凸関数になることが知られている [Korte 05]．従来の DisLRP で使用されている劣勾配法や，本研究で使用するバンドル法は，この凸関数性を利用したアルゴリズムである．

ここで，劣勾配法やバンドル法で使用される劣勾配（ベクトル）を導入する．

定義 2 凸関数 $L : \mathbb{R}^q \to \mathbb{R}$ に対してある点 $\mu^{(t)}$ における劣勾配ベクトルとは，任意の μ において，

$$L(\mu) \geq L(\mu^{(t)}) + g^{\mathsf{T}}(\mu - \mu^{(t)}), \tag{2.10}$$

が成り立つようなベクトル g である．また，g の要素を劣勾配と呼ぶ.

ラグランジュ双対問題における劣勾配は主問題で緩和した制約式になることが知られている．Dantzig-Wolfe 分解を用いた定式化では，あるラグランジュ乗数ベクトル μ のエージェント間制約 l に対する劣勾配 g_l は，

$$g_l = b_l - C^l_{trans}(x^*_1, \ldots, x^*_m),$$

と表すことができる．ここで，x^*_i は，ある μ における $\mathcal{D}_{\mathcal{DW}}$ の最適解である．また，ラグランジュ分解を用いた定式化では，あるラグランジュ乗数ベクトル λ におけるエージェント s に決定権のあるエージェント i に関する決定変数と，エージェント i 自身の決定変数を一致させる一致制約に対する劣勾配 g_l は，

$$g^s_i = {x_i}^* - \alpha^{t*}_i,$$

と表すことができる．ここで，α^{s*}_i は，ある μ における $\mathcal{D}_{\mathcal{LD}}$ の最適解である.

最適解は μ の値によって決定されるため，劣勾配を $g_i(\mu)$，劣勾配ベクトルを $g(\mu) = (g_1(\mu), \ldots, g_n(\mu))$ と表記する．また，$\rho_i(g(\mu))$ は，エージェント i に関係する劣勾配ベクトルを返す関数と定義する．

なお，命題 2 は劣勾配を用いると次のように書き直すことができる．

命題 3 ある μ において，全てのエージェントの $\mathcal{D}_i(\pi_i(\mu))$ の最適解を得て，かつ，それらを基に計算された劣勾配 g_j が全ての j において *0* のとき，それらの最適解は *(P)* の最適解を構成している．

ここで，Dantzig-Wolfe 分解を用いた定式化とラグランジュ分解を用いた定式化の間に存在する定理を紹介する．Dantzig-Wolfe 分解を用いた定式化は，ラグランジュ分解を用いた定式化でも表現できることは先に述べた．ここで，Dantzig-Wolfe 分解を用いた定式化で求まるラグランジュ緩和問題の最適値を $L_{\mathcal{DW}}$，ラグランジュ分解を用いた定式化で求まるラグランジュ緩和問題の最適値を $L_{\mathcal{LD}}$ として，両者の間には次の関係が成り立つ [Guignard 87].

定理 1 原問題を最大化問題とし，それぞれの定式化におけるラグランジュ双対問題の最適解を μ^*, λ^* とする．このとき，以下の不等式が成り立つ．

$$Opt \leq L_{\mathcal{LD}}(\lambda^*) \leq L_{\mathcal{DW}}(\mu^*). \tag{2.11}$$

すなわち，ラグランジュ分解を用いた定式化を使用すれば，Dantzig-Wolfe 分解を用いた定式化を使用するより良い質の上界値が得られることが期待できる．これは，Dantzig-Wolfe 分解がエージェント間制約を直接緩和してしまうのに対して，ラグランジュ分解を用いた定式化ではエージェント間制約がラグランジュ緩和後の問題にも残っていることに起因する．

なお，GMAP もラグランジュ分解を用いた定式化を使用することができる．ここで，GMAP をラグランジュ分解を用いた定式化で表した場合，

次のように表すことができる．

$$
\begin{aligned}
\max. \quad & \sum_{i\in A} f_i(x_i) = \sum_{i\in A}\sum_{j\in J'} p_{ij}x_{ij} \\
\text{s. t.} \quad & x_i \in \left\{ x_i | \sum_{j\in J'} w_{ij}x_{ij} \le c_i \right\}, \forall i \in A, \\
& (\alpha_1, \ldots, \alpha_m) \in \left\{ (\alpha_1, \ldots, \alpha_m) | \sum_{i\in A_j} \alpha_{ij} = 1, \forall j \in J' \right\}, \\
& x_i = \alpha_i, \forall i \in A, \\
& D_{ij} = \{0, 1\},\ \forall i \in A,\ \forall j \in J'.
\end{aligned}
\tag{2.12}
$$

ここで，式 (2.12) はこの定式化における一致制約である．式 (2.12) をラグランジュ緩和すると，次のようなラグランジュ緩和問題を得る．

$$
\begin{aligned}
\max. \quad & \sum_{i\in A}\sum_{j\in J'} p_{ij}x_{ij} + \sum_{i\in A}\sum_{j\in J'} \lambda_{ij}\left(x_{ij} - \alpha_{ij}\right) \\
\text{s. t.} \quad & x_i \in \left\{ x_i | \sum_{j\in J'} w_{ij}x_{ij} \le c_i \right\}, \forall i \in A, \\
& (\alpha_1, \ldots, \alpha_m) \in \left\{ (\alpha_1, \ldots, \alpha_m) | \sum_{i\in A_j} \alpha_{ij} = 1, \forall j \in J' \right\}.
\end{aligned}
$$

これを部分問題へと分解すると，各エージェント i に関する部分問題,

$$
\begin{aligned}
\max. \quad & \sum_{j\in J'} (p_{ij} - \lambda_{ij})x_{ij} \\
\text{s. t.} \quad & x_i \in \left\{ x_i | \sum_{j\in J'} w_{ij}x_{ij} \le c_i \right\},
\end{aligned}
$$

及び，財 j の割当制約に関する部分問題,

$$
\begin{aligned}
\max. \quad & -\sum_{i\in A_j} \lambda_{ij}\alpha_{ij} \\
\text{s. t.} \quad & \sum_{i\in A_j} \alpha_{ij} = 1,
\end{aligned}
$$

へと分解することが出来る．

ここで，各財に関する部分問題の制約式を行列で表した場合，完全ユニモジュラー行列となる．よって，各財に関する部分問題は必ず整数解となる [Korte 05]．式 (2.11) の等号が成立しないための必要十分条件は，全ての部分問題が整数解を持たないことである [Guignard 87] から，Dantzig-Wolfe 分解を用いて定式化された GMAP とラグランジュ分解を用いて定式化された GMAP で求まる上界値は同じである．従って本研究では，GMAP に対して部分問題の数がより少ない Dantzig-Wolfe 分解を用いた定式化を使用する．

2.3　プロトコル

2.3.1　プロトコルの基本動作

DisLRP の基本的なプロトコルは次の通りである．

Step 0: 初期化　各エージェント i が，パラメータ t を 1，ラグランジュ乗数ベクトル $\pi_i(\mu^{(t)}) = \pi_i(\mu^{(1)})$ の要素を全て 0 で初期化する．

Step 1: 部分問題の求解　現在の $\pi_i(\mu^{(t)})$ の値のもとで，各エージェント i は，ラグランジュ緩和問題 ($\mathcal{D}_{\mathcal{LD}}$) を任意の厳密解法を用いて最適解 $x_i^{(t)}$ を求める．

Step 2: 近傍通信　各エージェント i は，Step 1 で求めた $x_i^{(t)}$ を各自の近傍に送る．ここで，エージェント i の近傍 N_i とは，J_i 内の財の割当制約に現れる変数を持つ（k 以外の）エージェントの集合であり，$N_i = \bigcup_{j \in J_i} A_j \setminus \{i\}$ で表される．

Step 3: 劣勾配の計算　各エージェント i は，自身の最適解及び近傍から送られてきた最適解を基に劣勾配ベクトル $\rho_i(g(\mu^{(t)}))$ を計算する．

Step 4: $\mu^{(t)}$ の更新　各エージェント i は，ラグランジュ乗数ベクトルを $\mu^{(t+1)}$ に更新する．また，$t \leftarrow t+1$ として **Step 1** へ戻る．

プロトコルは **Step 0** で各パラメータを初期化した後，終了条件を満たすまで **Step 1** から **Step 4** を繰り返す．この **Step 1** から **Step 4** の 1 回の繰り返しをラウンドと呼び，その繰り返し回数 t をラウンド数と呼ぶ．

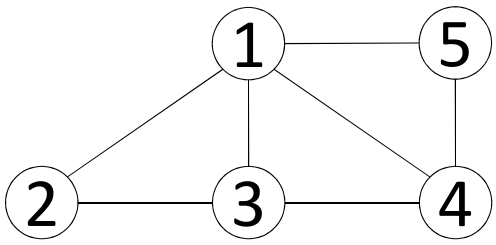

図 2.7: ある問題例の近傍構造

2.3.2　生成木を用いた大域情報の収集

Step 2 において，各エージェント i は劣勾配を計算するために近傍と局所情報を送受信する．劣勾配は近傍の構造によらず近傍同士の情報交換のみで計算できるが，$L(\mu)$ の値および命題 3 が成立しているか否か等は系全体の大域情報であり，近傍を越えた情報交換が必要である．図 2.7 は，5 エージェントの例題とその近傍関係を示すグラフである．接点がエージェント，枝が近傍関係を表す．この例題では，エージェント 1 の近傍が $N_1 = \{2, 3, 4, 5\}$，エージェント 2 の近傍が $N_2 = \{1, 3\}$ となる．エージェント 1 は，全てのエージェントと近傍関係にあるので，近傍通信のみで大域情報を得ることができる．一方，エージェント 2 の近傍はエージェント 1 とエージェント 3 のみであるので，それ以外のエージェントの局所情報を得るには近傍を越えた情報交換が必要である．

既存の DisLRP では，生成木を用いて大域情報を収集するプロトコルが提案されている [Hirayama 09]．このプロトコルでは，近傍関係を示すグラフ上に何らかの分散アルゴリズム [Gallager 83, Bui 04] により生成木を事前に構築し，各エージェント i は各ラウンド t において大域情報の計算に必要な自身の局所情報を生成木の近傍にのみ送信及び転送を行う．

ここで，$GI^{(t)}$ をラウンド t における大域情報の集合とする．例えば，$GI^{(t)} = \{L(\mu^{(t)})\}$ である．次に，$LI_i^{(t)}$ を $GI^{(t)}$ を計算するのに必要なエージェント i の局所情報とする．例えば，$LI_i^{(t)} = \{L_i(\pi_i(\mu))\}$ である．エージェント i における生成木上の近傍を $S_i \subseteq N_i$ と定義する．

図 2.8 に，図 2.7 の例題における幅優先 (BFS: Breadth First Search) の生成木，図 2.9 に深さ優先 (DFS: Depth First Search) の生成木の例を示す．太く示された枝が生成木を構成する枝を表す．例えば，図 2.8 におけるエージェント 1 の生成木上の近傍は $S_1 = \{2, 3, 4, 5\}(= N_1)$，図 2.9 におけるエージェント 1 の生成木上の近傍は $S_1 = \{2\}(\subset N_1)$ となる．

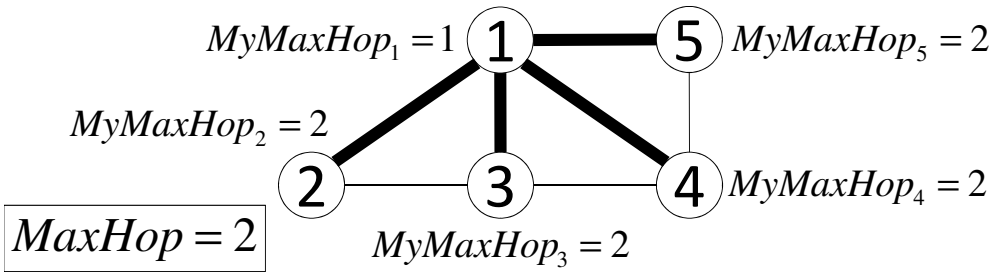

図 2.8: 幅優先の生成木の例

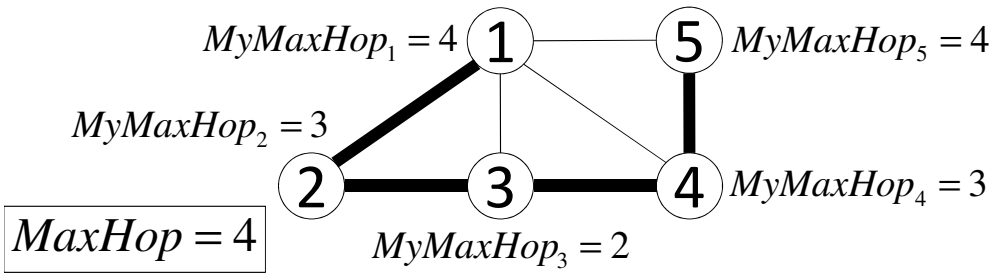

図 2.9: 深さ優先の生成木の例

大域情報を収集する場合，2.3.1 節における **Step 2** の後に，次のような手続きを挿入する．

Step 2′: 大域情報の収集 各エージェント i は，$LI_i^{(t)}$ をメッセージとして生成木上の各自の近傍に送る．また，各エージェント i は，生成木上の近傍であるエージェント $k' \in S_i$ から送られてきたメッセージを，エージェント k' 以外の生成木上の近傍，すなわち $S_i \setminus \{k'\}$ に転送する．

各エージェント i が，あるラウンド t における $GI^{(t)}$ を計算するのに必要な全ての $LI^{(t)}$ が得られたかどうかを知る方法は，次のように行う．各エージェント i はラウンド t において，**Step2′** に従い，自身の局所情報 $LI_i^{(t)}$ を記録するとともに，近傍内の生成木上の全てのエージェントにそれを送信する．それを受け取った近傍のあるエージェントは $LI_i^{(t)}$ を記録し，次のラウンドでそれを（k 以外の）生成木上のエージェントに転送する．この要領で，ラウンド t における各エージェント i の $LI_i^{(t)}$ を生成木上に伝播させる．一方，転送すべきラウンド t に関する局所情報がない

場合，エージェントは転送終了を意味する $\sharp^{(t)}$ を転送先に送る．やがて，エージェントは自信の近傍内の生成木上の全てのエージェントから $\sharp^{(t)}$ を受け取ることになる．すなわち，$m-1$ 個の $\sharp^{(t)}$ を受け取った時点で，全てのエージェントのラウンド t の局所情報が届いたことになる．

詳細は [Hirayama 09] を参照されたい．

2.3.3　大域情報利用のための同期化

DisLRP の最新のプロトコルである Adaptive DisLRP (ADisLRP) では，**Step 4** において μ を更新するために $GI^{(t)}$ を利用する．しかし，あるラウンド t における $GI^{(t)}$ を計算するのに必要な全ての $LI_i^{(t)}$ が得られるタイミングは，エージェントによって異なる．ラウンド t における局所情報 $LI_i^{(t)}$ がどのように伝播するか，図 2.8 の場合を例に挙げる．

まず，ラウンド t において，全てのエージェントは $LI_i^{(t)}$ を計算し，それを生成木上の全ての近傍のエージェントに送信する．

ラウンド $t+1$ において，エージェント 1 はその他のエージェントから全ての $LI_i^{(t)}$ が送られてくる．従って，エージェント 1 は受け取ったその他のエージェントの $LI_i^{(t)}$ をそれぞれ転送する．例えば，エージェント 2 に対してはエージェント 2 の局所情報を除く $LI_3^{(t)}, LI_4^{(t)}, LI_5^{(t)}$ を転送する．一方，エージェント 1 を除くその他のエージェントは，ラウンド $t+1$ において $LI_1^{(t)}$ しか送られてこない．また，$LI_1^{(t)}$ を転送すべきエージェントが存在しないので，エージェント 1 に対して $\sharp^{(t)}$ を送信する．

ラウンド $t+2$ において，エージェント 1 は全てのエージェントから $\sharp^{(t)}$ を受け取る．受け取った $\sharp^{(t)}$ の数が 4 であるため，エージェント 1 はこの時点で全ての局所情報を得られたと知ることができる．また，送られてきた $\sharp^{(t)}$ を 4 個全てその他のエージェントに転送する．一方その他のエージェントは，全ての局所情報をこのラウンドで得る．しかし，必要な数の $\sharp^{(t)}$ をまだ得ていないので，全ての局所情報を得られたかどうかこの時点で判断することはできない．

ラウンド $t+3$ において，その他のエージェントは 4 つの $\sharp^{(t)}$ を受け取る．従って，この時点でようやく全ての局所情報を得られたと知ることができる．

また，μ の更新は各エージェントで独立に行われる．しかし，命題 1 より，上界値を保証するためには各エージェントの μ の更新結果は必ず同じでなければならない．従って，μ の更新に $GI^{(t)}$ を利用するためには，$GI^{(t)}$ を利用するタイミングを同期化しなければならない．

ADisLRP では，大域情報を利用するための同期化プロトコルが提案されている [Hirayama 09]．以下にその概要を示す．このプロトコルでは，各エージェント i は次の 2 種類の静的なパラメータを持つ．

- $MyMaxHop_i$: エージェント i から生成木上で最も遠いエージェントまでの枝の数．
- $MaxHop$: $MyMaxHop_i$ の最大値．

$GI^{(t)}$ を利用するタイミングを同期化するために，各エージェント i は次の規則に従う．

- あるエージェント i が，あるラウンド $t_i(> t)$ に $GI^{(t)}$ を知ったとする．エージェント i は，$GI^{(t)}$ をラウンド $(t_i + MaxHop - MyMaxHop_i)$ で利用する．

上述のように，図 2.8 におけるエージェント 1 が $GI^{(t)}$ を知るのはラウンド $t + 2(= t_1)$ である．従って，エージェント 1 は $GI^{(t)}$ をラウンド $(t_1 + MaxHop - MyMaxHop_1) = ((t+2)+2-1) = (t+3)$ で利用する．その他のエージェントが $GI^{(t)}$ を知るのはラウンド $t+3(= t_i)$ である．従って，その他のエージェントは $GI^{(t)}$ をラウンド $(t_i + MaxHop - MyMaxHop_i) = ((t+3)+2-2) = (t+3)$ で利用する．このように，この規則に従うことによって，全てのエージェントは大域情報 $GI^{(t)}$ を必ず同じラウンドで利用することができる．

紙面の都合上，$MyMaxHop_i$ 及び $MaxHop$ をエージェントが知る方法は省略する．詳細は [Hirayama 09] を参照されたい．

2.3.4　劣勾配法

既存の DisLRP では，劣勾配法を用いた μ の更新に以下の式を用いる [Hirayama 09]．

$$\mu^{(t+1)} \leftarrow \mu^{(t)} - l^{(t)} \cdot g(\mu^{(t)}). \tag{2.13}$$

ここで，$l^{(t)}$ はステップ長と呼ばれる非負のスカラーである．最初に提案された DisLRP である DisLRP$_L$ では，$l^{(t)}$ には静的な値を用いていた [Hirayama 06]．最新のプロトコルである ADisLRP では，以下の式を用いて $l^{(t)}$ を動的に変化させる [Hirayama 09]．

$$l^{(t)} \leftarrow \frac{\pi(\min\{ub\} - \max\{lb\})}{\|g(\mu^{(t)})\|^2}.$$

ここで，$\|\cdot\|$ はユークリッドノルム，$\min\{ub\}$ は今まで得られた上界値 ub の中で一番低い値，$\max\{lb\}$ は今まで得られた推定下界値の中で一番高い値，$\pi \in \mathbb{R}^+$ は制御パラメータである．この式は最適値に収束する理論的な保証は無いが，実際には多くの問題で非常に有効であることが知られている [Yagiura 07].

劣勾配法は計算コストが小さい反面，上界値がラグランジュ双対問題の最適値に収束したかどうかを判定できない，解の振動が発生するなどの問題がある．

2.3.5　ラグランジュヒューリスティック

ラグランジュヒューリスティック (Lagrangian heuristic) とは，ラグランジュ緩和問題の解から，原問題の実行可能解に変換する手法のことである．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持つと期待される．

ラグランジュヒューリスティックは問題毎に設計されるため，一般的に記述することはほとんどできない．各問題に対するラグランジュヒューリスティックは，それぞれ後に述べる．

2.4　本節のまとめ

分散ラグランジュ緩和プロトコル (DisLRP) を適用可能な問題について，Dantzig-Wolfe 分解を用いた方法およびラグランジュ分解を用いた方法で問題を定義した．DisLRP は，エージェント間の局所的な通信のみによって動作が規定されており，全ての情報を収集するような管理者エージェントは存在しない．また，DisLRP の基本的なアルゴリズムについて述べ，問題点について取り上げた．

第3章　バンドル法を用いた分散ラグランジュ緩和プロトコル

3.1　はじめに

分散ラグランジュ緩和プロトコル (DisLRP: Distributed Lagrangian Relaxation Protocol) [Hirayama 06, Hirayama 07, Hirayama 09] は，エージェント間の局所的な通信のみを利用して一般化相互割当問題 (GMAP: Generalized Mutual Assignment Problem) を解くヒューリスティック解法として提案された．

DisLRP には，いくつかの種類が存在する．まず，DisLRP として最初に提案された DisLRP$_L$ は，分散環境で GMAP の実行可能解を求めるプロトコルである [Hirayama 06]．適応的な価格更新を用いた DisLRP (ADisLRP: Adaptive DisLRP) は，最大化問題に対する上界値を求めることに特化した最新の DisLRP であり，その他のプロトコルに比べて良い質の上界値を求めることができる [Hirayama 09].

2 章で述べたように，DisLRP は，Dantzig-Wolfe 分解またはラグランジュ分解と呼ばれるテクニックを用いて原問題を部分問題に分解し，ラグランジュ乗数と呼ばれるパラメータを更新しながら部分問題を繰り返し解くことによって，原問題に対する最良の上界値を探索する．この問題はラグランジュ双対問題と呼ばれる．従来の DisLRP では，最良の上界値，すなわちラグランジュ双対問題の最適値を探索するアルゴリズムとして劣勾配法を使用していた．劣勾配法では，エージェント間の局所的な同期のみでラグランジュ乗数を更新することができ，大域情報が全エージェントに行き渡るのを待つ必要がないという利点がある [Hirayama 06]. しかし，劣勾配法は計算コストが小さい反面，上界値がラグランジュ双対問題の最適値に収束したかどうかを判定できない，解の振動が発生するなどの問題がある．

本研究は，劣勾配法の代わりにバンドル法 [Schramm 92] を使用した新しいプロトコルであるバンドル法を用いた DisLRP (BDisLRP: Bundle DisLRP) を提案し，バンドル法が分散環境にも適用可能であることを示す．BDisLRP は，ADisLRP と同じく上界値を求めることに特化したプロトコルである．

バンドル法は安定化切除平面法とも呼ばれ，目的関数が凸関数である

無制約最適化問題を解く効率的なアルゴリズムである．バンドル法では，ラグランジュ乗数を更新するために必ず大域情報が必要となる．従って，エージェント間の局所的な同期のみではラグランジュ乗数を更新することができず，大域情報が全エージェントに行き渡るのを待つ必要がある．そこで本研究では，すでに行き渡った古い大域情報でラグランジュ乗数の更新を行うことによってプロトコルの遅延発生を防ぎ，最適値の効率的な探索を可能とした．本研究では，大域情報収集の時間遅れがあったとしても，BDisLRP が劣勾配法を使用した従来の DisLRP に比べて良い質の上界値を得ることを実験的に確かめる．

3.2　バンドル法

3.2.1　切除平面法

まず，バンドル法の基礎となる切除平面法について述べる．

劣勾配法では，あるラウンド t における劣勾配ベクトルの計算結果は，μ の更新に利用された後は必要ないので全て破棄される．一方，切除平面法では計算結果を破棄せず，全て記憶する．これは，式 (2.10) の等号が成り立つとき，式 (2.10) の右辺が関数 $L(\mu)$ に接する超平面をなすという性質を利用するためである．切除平面法では，以下のように定義される**バンドル**を全ラウンドで記憶する．

定義 3 ラウンド *1* からあるラウンド T までにおいて，ラグランジュ乗数ベクトル μ，ラグランジュ緩和問題の最適値 $L(\mu)$ と劣勾配 $g(\mu)$ をまとめたもの，すなわち $\{\langle \mu^{(t)}, L(\mu^{(t)}), g(\mu^{(t)})\rangle | t \in \{1, \ldots, T\}\}$ をバンドルと呼ぶ．

切除平面法はこのバンドルを用いるので，バンドル法の一種とも言える．

ここで，式 (2.10) の右辺を $f^{(t)}(\mu)$ とおく．すなわち，

$$f^{(t)}(\mu) = L(\mu^{(t)}) + g^{\mathsf{T}}(\mu - \mu^{(t)}), \tag{3.1}$$

である．前述のように，$f^{(t)}(\mu)$ は $L(\mu)$ に接する超平面となるため，$f^{(t)}(\mu)$ はラグランジュ双対問題の目的関数値の空間を最適値 $L(\mu^*)$ を含む領域と含まない領域に分割することができる．$f^{(t)}(\mu)$ を逐次追加して $L(\mu^*)$ を含まない領域を取り除きながら，ラグランジュ双対問題の最適解 μ^* を求める方法が切除平面法である．

切除平面法では，μ の更新の際に以下の線形計画問題を解く．

$$\begin{aligned} \mathcal{CP} \quad & (\text{decide } \mu_j, \forall j \in J, r) : \\ \min. \quad & r \\ \text{s. t.} \quad & f^{(t)}(\mu) \leq r,\ \forall t \in \{1, \ldots, T\}. \end{aligned}$$

$\mathcal{CP}$ の最適値を r_1^* とおく．もし，あるラウンド t で $r_1^* = L(\mu^{(t)})$ となれば，$\mu^{(t)}$ がラグランジュ双対問題の最適解 μ^* となる．

残念ながら，切除平面法は分散環境で使用することが困難である．DisLRP においては，エージェントがそれぞれ μ の更新を行う．しかし，μ は全てのエージェントで同じ値でなければ上界値を保証することができないため，更新結果は常に全エージェントで同じでなければならない．すなわち，μ を更新するアルゴリズムには解の一意性が必要である．

切除平面法は，μ を更新するために線形計画問題を解かなければならないが，一般に線形計画問題の解は一意とは限らない．線形計画問題を解く通常のソルバーは，最適解を 1 つ発見した段階でアルゴリズムが終了する．一方，解が複数存在するような問題例を解いた場合，たとえ全エージェントが同じソルバーを使用していたとしても，各々のソルバーの設定によって最初に発見する最適解が異なる可能性がある．その場合，$\mathcal{CP}$ を解いた後にエージェント間で μ の値が一致しているかどうか確認する処理が必要となる．しかし，そのような処理は通信コストの増大につながるので望ましくない．

従って本研究では，切除平面法ではなく解の一意性が保証された後述のバンドル法（安定化切除平面法）を利用する．

3.2.2　バンドル法（安定化切除平面法）

バンドル法（安定化切除平面法）は，切除平面法を拡張した近接点法の一種で，中心点と呼ばれるパラメータを未知の最適解に近づけることによって最適値へと収束させる手法である．直感的には，上述の $\mathcal{CP}$ の目的関数に 2 次の安定化項を加えた切除平面法と理解できる．

バンドル法では，μ の更新の際に以下の 2 次計画問題を解く．

$$\begin{aligned} \mathcal{BM} \quad & (\text{decide } \mu_j, \forall j \in J, r) : \\ \min. \quad & r + \frac{1}{2h}\|\mu - \mu_{cp}\|^2 \\ \text{s. t.} \quad & f^{(t)}(\mu) \leq r,\ \forall t \in \{1, \ldots, T\}. \end{aligned}$$

表 3.1: 各解法の特徴

	劣勾配法	（切除平面法）	バンドル法
計算コスト	低	（中）	高
反復回数	高	（高/中）	低
最適性の保証	無	（有）	有

ここで，$\mu_{cp} \in \mathbb{R}^n$ は所与の中心点，h は正の実数値制御パラメータである．$\mathcal{BM}$ の最適値を r_2^* とおく．

$\mathcal{BM}$ の目的関数は狭義凸関数であるため，解の一意性が保証される．従って，分散環境において各エージェントが任意のソルバーを使ったとしても，同じ計算結果を得ることができる．

3.2.3　解法の比較

表 3.1 に，各解法の特徴を示す．劣勾配法は，ベクトルを更新するだけなので計算量は $O(d)$ である．切除平面法は線形計画問題を解くが，これは多項式時間で解ける．バンドル法は 2 次計画問題を解くので，切除平面法より計算コストがかかる．

一方，反復回数は劣勾配法，切除平面法は収束するまでに非常に多くの反復を要するのに対し，バンドル法はそれらに比べて少ない反復回数で収束することが報告されている．

また，劣勾配法はラグランジュ双対問題が収束したかどうか最適性の保証ができないのに対し，切除平面法とバンドル法は最適性の保証が可能であるのが大きな違いである．

3.3　プロトコル

バンドル法は，μ を更新するために必ず大域情報が必要だが，序盤のラウンドにおいては大域情報が全エージェントには行き渡ってはいない．従って，常にバンドル法によって μ を更新する場合，序盤のラウンドでは常に同じ切除平面によって $\mathcal{BM}$ を解くことになり，μ は同じ値となってしまう．この間，事実上プロトコルは μ の更新を行わないということになり無駄となる．そこで，切除平面が更新されるようになるまで従来の劣

勾配法で更新することによってプロトコルの遅延を防ぎ，最適値の効率的な探索を可能とした．

中心点 μ_{cp} を更新するかどうかを判定するパラメータとして $\kappa \in (0,1)$ を用いる．μ_{cp} が更新される場合をシリアスステップ，更新されない場合をヌルステップと呼ぶ．シリアスステップの場合，次のラウンドで求まる $L(\mu)$ の値は必ず改善する．一方，ヌルステップの場合，次のラウンドで求まる $L(\mu)$ の値が改善するとは限らないが，解の振動を抑える働きがある．

また，$\delta \in \mathbb{R}^+$ を終了条件に対する許容誤差とする．すなわち，$\delta = 0$ の場合は真の最適値を得ることができる．

ラグランジュ乗数の更新にバンドル法を用いた BDisLRP のプロトコルは，次の通りである．

Step 0: 初期化 各エージェント i が，パラメータ t を 1，パラメータ h, κ, δ のそれぞれについて共通する適当な値，ラグランジュ乗数ベクトル $\mu^{(1)} = \mu_{cp}$ の要素を全て 0 で初期化する．

Step 1: 部分問題の求解 現在の $\pi_i(\mu^{(t)})$ の値のもとで，各エージェント i は，$\mathcal{LGMP}_i(\pi_i(\mu^{(t)}))$ を 0-1 ナップサック問題に対する任意の厳密解法を用いて最適解 $x_{ij}^{(t)}$ を求める．

Step 2: 近傍通信 各エージェント i は，Step 1 で求めた $x_{ij}^{(t)}$ を各自の近傍 N_i に送る．

Step 2′: 大域情報の収集 各エージェント i は，$LI_i^{(t)}$ をメッセージとして生成木上の各自の近傍 S_i に送る．また，各エージェント i は，生成木上の近傍であるエージェント $k' \in S_i$ から送られてきたメッセージを，エージェント k' 以外の生成木上の近傍，すなわち $S_i \setminus \{k'\}$ に転送する．

Step 3: 劣勾配の計算 各エージェント i は，自身の最適解及び近傍から送られてきた最適解を基に劣勾配ベクトル $\rho_i(g(\mu^{(t)}))$ を計算する．もし，全ての劣勾配 $g_j(\mu^{(t)})$ が 0 ならば，命題 3 より最適解が得られているため終了する．

Step 4 もし $MyMaxHop_i$ 及び $MaxHop$ が求まっていない場合，**Step 4S** へ進む．それ以外は **Step 4B.1** へ進む．

Step 4B.1: バンドル法による $\mu^{(t)}$ の更新 t' を $t_i+MyMaxHop-MyMaxHop_i$ とする．各エージェント i は t' までの大域情報を用いて $\mathcal{BM}$ を解き，$r_2^*, \mu^{(t+1)}$ を求める．

Step 4B.2: バンドル法の停止条件 $\delta' \leftarrow L(\mu_{cp}) - r_2^*$ とする．もし $\delta' \leq \delta$ なら，ラグランジュ双対問題の最適解が得られているとみなして，プロトコルを終了する．

Step 4B.3: 中心点の更新 もし $L(\mu_{cp}) - L(\mu^{(t')}) \geq \kappa\delta'$ なら，$\mu_{cp} \leftarrow \mu^{(t')}$ とする．$t \leftarrow t+1$ として，**Step 1** へ戻る．

Step 4S: 劣勾配法による $\mu^{(t)}$ の更新 ラグランジュ乗数ベクトルをステップ長 1 の劣勾配法によって $\mu^{(t+1)}$ に更新し，$t \leftarrow t+1$ として **Step 1** へ戻る．

Step 2′ では，各エージェント i は $LI_i^{(t)}$ として $\{\pi_i(\mu^{(t)}), L_i(\pi_i(\mu^{(t)})), \rho_i(g(\mu^{(t)}))\}$ を送信する．

Step 4B.2 は，DisLRP の新しい停止条件である．バンドル法は上界値が最適値に収束したかどうかを判定できるため，BDisLRP ではラグランジュ双対問題の最適値を得られた時点で終了する．ここで，原問題の最適値とラグランジュ双対問題の最適値との間には整数性ギャップが生じる可能性があるため，両者の値は必ずしも同じであるとは限らないことに注意されたい．

Step 4S は，劣勾配法によってラグランジュ乗数を更新するためのステップである．

3.4　実験

BDisLRP の性能を評価するため，2 種類の実験を行った．

1 つ目の実験は，Dantzig-Wolfe 分解を用いた定式化である一般化相互割当問題 (GMAP: Generalized Mutual Assignment Problem) の問題例を用いて，上界値を求める最新のプロトコルである ADisLRP との比較実験，及び大域情報収集による時間遅れの影響を評価する比較，評価を行った．

2 つ目の実験は，ラグランジュ分解を用いた定式化である重み付き最大充足化問題 (WMax-SAT Problem: Weighted Maximum Satisfiabilit Problem) の問題例を用いて，制約式の分割の仕方による上界値の質の影響について比較，評価を行った．

3.4.1　一般化相互割当問題の問題例を用いた実験

本節では，Dantzig-Wolfe 分解を用いた定式化である GMAP の問題例を用いた実験結果を示す．GMAP は最大化問題であるため，BDisLRP が導出するのは上界値である．

実験の目的は，本研究で提案した BDisLRP と上界値を求めることに特化した最新のプロトコルである適応的な価格更新を用いた DisLRP (ADisLRP: Adaptive DisLRP) と比較，評価すること，及び大域情報収集による時間遅れの影響を評価することである．

問題例は，OR-Library[Beasley] に掲載されている GAP のベンチマーク問題例の中から，最適値が判明していない問題例を含み，問題例の規模が大きいカテゴリー d 及び e を使用した．なお，カテゴリー d 及び e の問題例は全て最小化問題である．従って，目的関数に -1 を乗じて等価な最大化問題に変換して実験を行った．

実験は，デスクトップ PC（Core i7-2600K 3.40GHz，メモリ 12GB，Windows 7 Professional）上のシミュレータで実施した．また，BDisLRP 及び ADisLRP は Java で実装し，JDK 1.8.0-25 によってコンパイルした．各エージェントのナップサック問題及びバンドル法の 2 次計画問題を解くソルバーには，IBM ILOG CPLEX 12.6.0 を使用した．

全ての実験において，問題例のサイズや構造に依存した BDisLRP のパラメータ設定は困難だったため，予備実験により様々なパラメータを検討した結果，全体的に良い実験結果の傾向を示したパラメータ（$h = 128, \kappa = 0.1, \delta = 10^{-6}$）を使用した．

プロトコルの停止条件は BDisLRP と ADisLRP で少し異なる．まず，BDisLRP 及び ADisLRP 共通の終了条件を設けるため，実行時間の上限を 30 分 (1800 秒) とした．BDisLRP は，ラグランジュ双対問題の最適値（原問題の最小の上界値）が得られたかどうかが判定できるため，ラグランジュ双対問題の最適値が得られた時点で終了する [1]．一方，ADisLRP はラグランジュ双対問題の最適値が得られたかどうかが判定できないため，劣勾配法のパラメータ π が最終的に $\pi < 10^{-6}$ となった時点で終了する．なお，π は [Hirayama 09] の設定に従い，初期値を 2 とし，各ラウンドまでの最小の上界値 ub が 100 ラウンド連続で更新されなければ，π の値を 1/2 倍するものとする．

評価指標には，プロトコルがいずれかの停止条件で停止した時に得ら

[1] ラグランジュ双対問題の最適値が原問題の最適値と一致するとは限らないことに注意されたい．

表 3.2: BFS を用いた BDisLRP，DFS を用いた BDisLRP 及び BFS を用いた ADisLRP の比較

Instance	Opt (Best)	Quality of Bounds			Round			Time [sec]		
		Bund BFS	Bund DFS	Adap BFS	Bund BFS	Bund DFS	Adap BFS	Bund BFS	Bund DFS	Adap BFS
d5100	−6353*	**1.0004**	1.0005	**1.0004**	1138	1734	5581	1545	>1800	1487
d5200	−12742*	1.0002	1.0005	**1.0001**	1046	1126	4429	>1800	>1800	>1800
d10100	−6347*	1.0011	1.0024	**1.0008**	873	1530	4013	>1800	>1800	>1800
d10200	−12430*	1.0010	1.0038	**1.0003**	727	952	3324	>1800	>1800	>1800
d10400	−24961*	1.0013	1.0079	**1.0001**	581	709	2828	>1800	>1800	>1800
d15900	−55414	1.0099	1.0556	**1.0010**	323	457	1532	>1800	>1800	>1800
d20100	−6185*	1.0033	1.0282	**1.0014**	554	1019	2902	>1800	>1800	>1800
d20200	−12244	1.0045	1.0653	**1.0012**	489	853	2151	>1800	>1800	>1800
d20400	−24585	1.0084	1.0587	**1.0011**	370	667	1639	>1800	>1800	>1800
d201600	−97837	1.0475	1.1473	**1.0036**	227	323	952	>1800	>1800	>1800
d30900	−54868	1.0619	1.2138	**1.0096**	237	333	937	>1800	>1800	>1800
d40400	−24417	1.0457	1.2823	**1.0041**	305	440	1045	>1800	>1800	>1800
d401600	−97113	1.2916	1.6146	**1.1956**	153	196	577	>1800	>1800	>1800
d60900	−54606	1.2304	**1.1389**	11.8554	154	210	584	>1800	>1800	>1800
d801600	−97052	**1.3540**	Infinit	12.2139	78	66	383	>1800	>1800	>1800
e5100	−12681*	**1.0006**	**1.0006**	**1.0006**	454	1097	4676	223	325	879
e5200	−24930*	**1.0001**	**1.0001**	**1.0001**	711	992	5534	605	739	1073
e10100	−11577*	**1.0007**	1.0010	**1.0007**	913	1607	4243	1727	>1800	1179
e10200	−23307*	1.0003	1.0006	**1.0002**	854	1364	5153	>1800	>1800	1567
e10400	−45746*	1.0002	1.0010	**1.0000**	761	899	4498	>1800	>1800	>1800
e15900	−102421*	1.0005	1.0253	**1.0000**	505	509	2745	>1800	>1800	>1800
e20100	−8436*	1.0009	1.0078	**1.0005**	629	1235	3271	>1800	>1800	>1800
e20200	−22379*	1.0009	1.0098	**1.0001**	560	908	3516	>1800	>1800	>1800
e20400	−44877*	1.0010	1.0397	**1.0000**	481	596	2852	>1800	>1800	>1800
e201600	−180645*	1.0026	1.1427	**1.0006**	312	352	1581	>1800	>1800	>1800
e30900	−100427*	**1.0039**	1.3391	18.6980	299	393	1507	>1800	>1800	>1800
e40400	−44561*	**1.0039**	1.2632	18.6525	332	462	1651	>1800	>1800	>1800
e401600	−178293*	**1.0245**	1.3830	1.0832	182	269	1001	>1800	>1800	>1800
e60900	−100149*	**1.0296**	1.2254	18.7544	196	320	1105	>1800	>1800	>1800
e801600	−176820*	**1.2031**	Infinit	18.7031	71	124	819	>1800	>1800	>1800

れる最小の上界値を用いた．

3.4.1.1　上界値の解の質の比較　実験結果を表 3.2 に示す．Opt (Best) は，集中型解法 [Posta 12] によって得られた問題例の最適値（最適値が判明していない問題例は知られている最良の下界値），Quality of Bounds はプロトコルが停止した時点で得られた最小の上界値を最適値（最良の下界値）で割った上界値の質を表している．太文字の値は，各指標の比較においてもっとも良い値を示している．Infinit と表示されている欄は，制限時間内に上界値を計算するための局所情報を全て収集できなかったことを表す．Time は実行時間を秒で表している．Time の欄に > 1800 と書かれて

いる場合は，実行時間が制限時間である 30 分（1800 秒）を超えたことを意味する．Bund は BDisLRP，Adap は ADisLRP，BFS は幅優先木，DFS は深さ優先木をそれぞれ表している．

d5100（5 エージェント 100 財），e5100（5 エージェント 100 財），e5200（5 エージェント 200 財）及び e10100（10 エージェント 100 財）の問題例の場合，BFS を用いた BDisLRP と ADisLRP は同じ上界値を得ている．これらの問題例については，BDisLRP は制限時間内にプロトコルが終了して最適性を保証できているため，ラグランジュ双対問題の最適値であるという理論的保証を与えているという点で，BDisLRP の方が ADisLRP よりも優れていると言える．また，e5100 と e5200 の問題例は，BDisLRP の方が ADisLRP よりもラウンドと実行時間双方の点でより速く求めている．実験結果によると，e5100 と e5200 の問題例において ADisLRP が最後に上界値を更新したラウンド数はそれぞれ 4575 と 5434 である．これは，BDisLRP がラグランジュ双対問題の最適値を発見できるので，最適性が保証された段階でプロトコルを終了できるのに対し，ADisLRP は最適性が保証できないため，ラウンドと実行時間を無駄に消費してしまっているためである．

ここで，BDisLRP は μ の更新のために 2 次計画問題を解いていることに注意されたい．μ を更新するための計算量を比較すると，ベクトルの要素の値を加算もしくは減算するだけの ADisLRP に対して，BDisLRP は 2 次計画問題を解くため明らかに計算量が大きい．しかし，計算時間は BDisLRP の方が短いので，2 次計画問題を解く効果がそのコストを上回っていると考えられる．

d60900（60 エージェント 900 財），d201600（20 エージェント 1600 財），及び e30900（30 エージェント 900 財）から e801600（80 エージェント 1600 財）にかけての大きい問題例の場合，BDisLRP は制限時間内でラグランジュ双対問題の最適値を求めることに失敗している．それにも関わらず，BDisLRP は ADisLRP と同じ実行時間内かつ短いラウンドでより良い上界値を求めていることが分かる．これは，ADisLRP が良い上界値に収束する前に解の振動を起こしていることが原因だと考えられる．図 3.1 に，問題例 d60900 におけるラウンド数と上界値の関係をプロットしたグラフを示す．横軸はラウンド数，縦軸はそのラウンド数において求まった上界値の質を表す．図 3.1 より，BDisLRP でも振動は見られるが，中心点の更新により振動が小さく抑えられていることが分かる．一方 ADisLRP では，序盤のラウンドで大きく解の質が低下し，その後も大きな振動を繰

り返しながらゆっくりと解の質が上昇していることが分かる．

上記で挙げた問題例以外の問題例は，ADisLRP の方が BDisLRP より良い上界値が得られている．これらの問題例について，BDisLRP は財数が増えるにつれてラウンド数は減る傾向にある．これは，財数が増えることによって 2 次計画問題の求解時間が長くなってしまったためと考えられる．また，d201600（20 エージェント 1600 財），d30900（30 エージェント 900 財），d201600（40 エージェント 400 財），d401600（40 エージェント 1600 財）の問題例の場合，得られる上界値の質の差が大きいことが分かる．これらは，最適値が分かっていない問題例であり，問題例の構造が原因であると考えられる．それ以外の問題例については，得られる上界値の質の差は 1%未満であり，BDisLRP は ADisLRP に比べて十分良い上界値を得ていると言える．

最後に，通信コストについて考察する．本実験ではシミュレータ上にプロトコルを実装しており，TCP/IP 等のプロトコルを用いた実際の通信は行っていない．しかし，エージェント同士の局所情報の交換及び転送は実際に行っており，情報量が大きければその分実行時間も大きくなる．近傍のエージェントに送らなければならない局所情報の数は，BDisLRP の方が ADisLRP よりも多い．しかし，ラウンド数を比較すると BDisLRP は ADisLRP に比べて少ないので，情報量の大きさが実験の結果に及ぼした影響は少ないものと考えられる．

3.4.1.2　大域情報収集による時間遅れの影響　次に，大域情報収集による時間遅れの影響について考察する．BDisLRP 及び ADisLRP 共に大域情報を収集するために生成木を用いる．大域情報収集による時間遅れの影響を評価するため，BDisLRP では幅優先探索 (BFS) 及び深さ優先探索 (DFS) の両方で実験を行った．なお，ADisLRP は実験結果の知見 [Hirayama 09] により生成木の形状によって性能が変わらないため，BFS のみで実験を行った．

BFS を用いた BDisLRP と DFS を用いた BDisLRP を比較すると，全ての問題例において BFS を用いた BDisLRP の方がより良い上界値を求めており，DFS の方が大域情報収集による時間遅れの影響が大きいことがわかる．これは，DFS が BFS に比べて大域情報が全エージェントに行き渡るラウンド数が大きく，バンドルの更新が遅いことが原因だと考えられる．従って，BDisLRP を使用する場合は，できるだけ木の深さが小さい生成木を使用することが望ましいと考えられる．

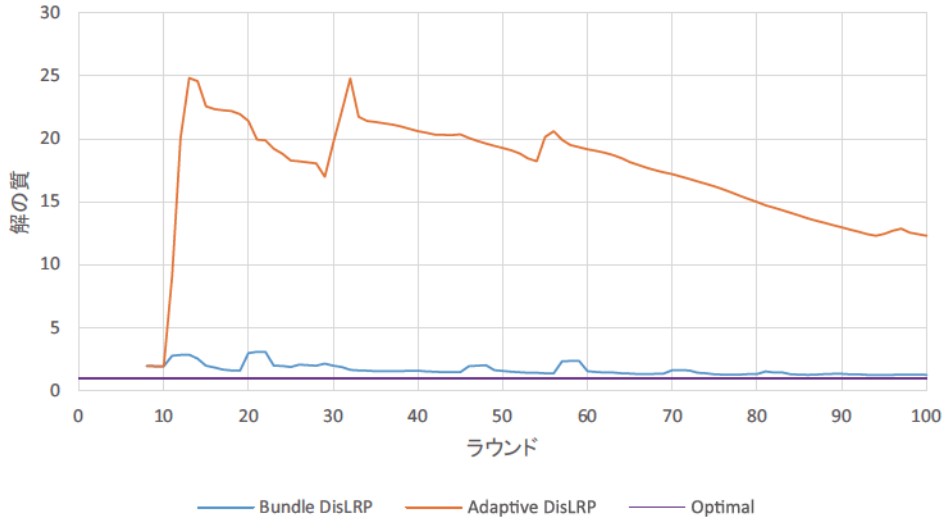

図 3.1: d60900 におけるラウンド数と上界値の質

以上より，BFS を用いた BDisLRP は財数が小さい問題例及びエージェント数，財数共に大きな問題例において ADisLRP より良い性能を持つと言える．

3.4.2　重み付き最大充足化問題の問題例を用いた実験

本節では，ラグランジュ分解を用いた定式化である WMax-SAT の問題例を用いた実験結果を示す．WMax-SAT は最小化問題であるため，BDisLRP が導出するのは下界値である．

実験の目的は，制約式の分解の仕方による下界値の質の影響を比較，評価することである．また，共有変数の割合による下界値の質の変化を比較，評価することである．

BDisLRP の性能を評価するため，3 種類の実験を行った．WMax-SAT の下界値を分散環境で導出できる先行研究は，筆者の知る限り存在しない．しかし，DisLRP は集中環境でも使用することが出来るので，本実験では集中環境で下界値を導出できる Max-SAT Resolution[Li 09] を比較対象とした．全ての実験において，最適値が分かっている問題例に対しては最適値/下界値，最適値が不明な問題例は現在知られている最も良い目的関数値/下界値で定義される下界値の質で評価する．定義から明らかな

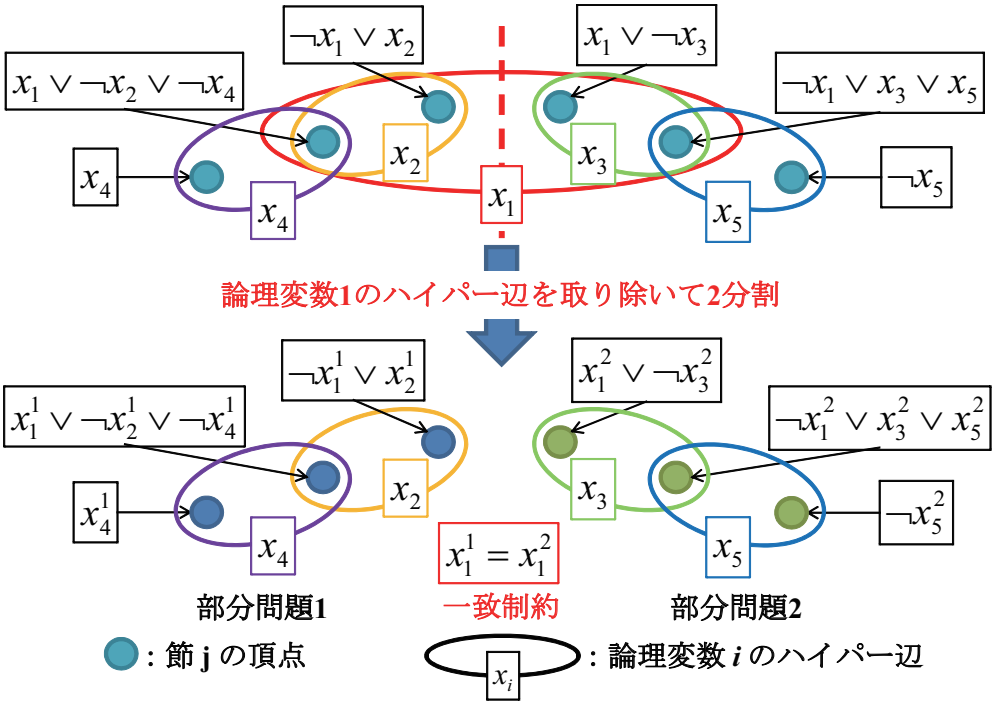

図 3.2: 節集合（ハイパーグラフ）2 分割問題の例

ように，下界値の質は 1.0 に近ければ近いほど良い．

部分問題を解く重み付き Max-SAT ソルバーには akmaxsat[Kugel 10] を使用した．また，線形計画問題及び 2 次計画問題を解くソルバーとして CPLEX12.4 を使用した．実験環境は CPU: Intel Core i7-2600@3.4GHz, Memory: 8GB, OS: Ubuntu 11.04, Compiler: Java 1.6.0_27 である．分割数は実験 1，2 では $k \in \{2, 3, 4, 5, 10\}$，実験 3 では $k = 10$ に固定して実験を行った．全ての実験において 5 分でタイムアウトとし，制限時間内に得られた最良の下界値を評価に用いた．

3.4.2.1　節集合の分割方法　先行研究において，共有変数が少ないほど良い質の実行可能解が得られることがわかっている [黒田 09]．従って，共有変数が少ないほど良い質の下界値も得られることが予想され，どの節をどの部分問題に割り当てるかが重要な問題となる．

できるだけ少ない共有変数で節集合を k 個に分割する問題は，節を頂点，論理変数をハイパー辺と見なせばハイパーグラフ k 分割問題と見なすことができる．ハイパーグラフ k 分割問題とは，いくつかのハイパー辺

表 3.3: 実験 1：重み付き Max-2SAT の結果

分割数	Hypergraph	Random	Resolution
2	**0.9419***	0.8948	0.8069
3	**0.9049**	0.7704	0.8069
4	0.7686	0.6647	**0.8069**
5	0.8052	0.7036	**0.8069**
10	0.7012	0.1966	**0.8069**

表 3.4: 実験 1：重み付き Max-3SAT の結果

分割数	Hypergraph	Random	Resolution
2	**0.4381***	0.3447	0.0000
3	**0.1917**	0.0268	0.0000
4	**0.0195**	0.0000	0.0000
5	**0.0452**	0.0000	0.0000
10	0.0000	0.0000	0.0000

を取り除いて k 個の部分ハイパーグラフに分割するとき，取り除くハイパー辺の重みを最小化する問題である．この問題を解いた結果取り除かれた論理変数（ハイパー辺）が共有変数となる．図 3.2 に，節集合を 2 分割する例を示す．図 3.2 の例題では，例えば論理変数 1 のハイパー辺を取り除けば，2 つの部分ハイパーグラフに分割できることが容易に分かる．

本研究ではハイパーグラフ k 分割問題を解く手法として，ヒューリスティック解法 [Karypis 97] を実装したソフトウェアである hMetis を使用した．

3.4.2.2　節集合分割の方法による比較　1 つ目の実験は，節集合分割の方法による下界値への影響を評価する．比較対象はランダムに節集合を分割する方法と，ハイパーグラフ k 分割問題で節集合を分割する方法である．それぞれ分割方法で 10 回の試行を行い，下界値の質の平均を比較する．使用した問題例は，Max-SAT Evaluation 2012 で使用された WMax-SATRandom 部門の Max-2SAT 問題 1 問（100 変数 1200 節）及び Max-3SAT 問題 1 問（70 変数 700 節）である．

表 3.3 と表 3.4 にそれぞれ Max-2SAT 問題と Max-3SAT 問題の結果を示す．Hypergraph はハイパーグラフ k 分割を，Random はランダム分割をそれぞれ示している．太字になっている数値は，その分割数において一

表 3.5: 実験 2：重み付き Max-2SAT の結果

分割数	BDisLRP	ADisLRP	Resolution
2	0.9335	**0.9337***	0.7994
3	**0.8994**	0.8956	0.7994
4	0.7705	0.7638	**0.7994**
5	**0.8138**	0.8064	0.7994
10	0.7070	0.6984	**0.7994**

表 3.6: 実験 2：重み付き Max-3SAT の結果

分割数	BDisLRP	ADisLRP	Resolution
2	**0.6054***	0.5949	0.0000
3	**0.3420**	0.3113	0.0000
4	**0.0902**	0.0334	0.0000
5	**0.1105**	0.0364	0.0000
10	**0.0068**	0.0000	0.0000

番良い下界値の質が得られていることを示す．また，*がついている数値はその表の中で最も良い下界の質の平均を示す（表 3.8 を除く）．なお，Max-SAT Resolution も分割数毎に表示しているが，Max-SAT Resolution は分割数に影響されないことに注意されたい．

表 3.3 と表 3.4 より，全ての分割数でハイパーグラフ k 分割の方がランダムより同じかより良い下界値の質が得られていることが分かる．Max-2SAT 問題の場合，分割数が 4 を超えると Max-SAT Resolution よりも下界値の質が悪くなる．これは，部分問題が保持する節数が少なくなることによって部分問題の最適値が低くなることが原因だと考えられる．

この結果より，以降の実験では全てハイパーグラフ k 分割問題で節集合を分割する方法を採用する．

表 3.7: 実験 2:重み付き Max-3SAT における問題例別の結果

分割数	2		3		4		5		10		-
Instance	Bund	Adap	Bund	Adap	Bund	Adap	Bund	Adap	Bund	Adap	Res
C700_0	**0.4627**	0.4505	0.1641	0.1531	0.0199	0.0000	0.0443	0.0000	0.0000	0.0000	0.0000
C700_1	**0.5453**	0.5367	0.2578	0.2124	0.0000	0.0000	0.0443	0.0000	0.0000	0.0000	0.0000
C700_2	**0.4483**	0.4258	0.2307	0.2103	0.0000	0.0000	0.0544	0.0000	0.0000	0.0000	0.0000
C700_3	**0.5683**	0.5599	0.2751	0.2383	0.0121	0.0000	0.0450	0.0000	0.0000	0.0000	0.0000
C700_4	**0.4897**	0.4684	0.2597	0.2170	0.0000	0.0000	0.0304	0.0000	0.0000	0.0000	0.0000
C700_5	**0.5024**	0.4866	0.2356	0.1657	0.0000	0.0000	0.0323	0.0000	0.0000	0.0000	0.0000
C700_6	**0.5341**	0.5044	0.2013	0.1370	0.0000	0.0000	0.0658	0.0000	0.0000	0.0000	0.0000
C700_7	**0.4832**	0.4721	0.2507	0.2091	0.0431	0.0000	0.0713	0.0000	0.0000	0.0000	0.0000
C700_8	**0.6875**	0.6752	0.2981	0.2160	0.0377	0.0000	0.0656	0.0000	0.0000	0.0000	0.0000
C700_9	**0.5070**	0.4889	0.2206	0.1646	0.0887	0.0000	0.0456	0.0000	0.0000	0.0000	0.0000
C800_0	**0.5784**	0.5622	0.3008	0.2745	0.0485	0.0000	0.0645	0.0000	0.0009	0.0000	0.0000
C800_1	**0.5905**	0.5764	0.2955	0.2580	0.0422	0.0000	0.1195	0.0145	0.0000	0.0000	0.0000
C800_2	**0.6041**	0.5909	0.3569	0.3089	0.0246	0.0000	0.1051	0.0000	0.0000	0.0000	0.0000
C800_3	**0.7106**	0.6953	0.3949	0.3655	0.0580	0.0000	0.1121	0.0149	0.0000	0.0000	0.0000
C800_4	**0.6016**	0.5992	0.2913	0.2590	0.0055	0.0000	0.0865	0.0000	0.0000	0.0000	0.0000
C800_5	**0.6071**	0.6000	0.3264	0.3080	0.1188	0.0240	0.0850	0.0000	0.0000	0.0000	0.0000
C800_6	**0.5585**	0.5414	0.3237	0.2765	0.1053	0.0074	0.0895	0.0000	0.0000	0.0000	0.0000
C800_7	**0.5899**	0.5692	0.3100	0.2901	0.0764	0.0000	0.0923	0.0136	0.0000	0.0000	0.0000
C800_8	**0.6395**	0.6303	0.3592	0.3173	0.0761	0.0000	0.1068	0.0000	0.0000	0.0000	0.0000
C800_9	**0.5979**	0.5801	0.3217	0.2966	0.0789	0.0000	0.1389	0.0666	0.0000	0.0000	0.0000
C900_0	**0.6265**	0.6163	0.3661	0.3493	0.0694	0.0000	0.1421	0.0382	0.0003	0.0000	0.0000
C900_1	**0.6108**	0.6091	0.3405	0.3283	0.0845	0.0000	0.0313	0.0680	0.0000	0.0000	0.0000
C900_2	**0.6256**	0.6196	0.3374	0.3071	0.0744	0.0000	0.1456	0.0980	0.0001	0.0000	0.0000
C900_3	**0.6623**	0.6621	0.3830	0.3642	0.1004	0.0000	0.0491	0.0491	0.0000	0.0000	0.0000
C900_4	**0.6021**	0.5936	0.3369	0.3025	0.0935	0.0000	0.1067	0.0420	0.0000	0.0000	0.0000
C900_5	**0.6354**	0.6287	0.3840	0.3585	0.0879	0.0000	0.1703	0.0439	0.0007	0.0000	0.0000
C900_6	**0.6411**	0.6372	0.3350	0.2961	0.1976	0.1521	0.1604	0.0230	0.0171	0.0000	0.0000
C900_7	**0.6043**	0.5894	0.3536	0.3268	0.0761	0.0000	0.1163	0.0155	0.0008	0.0000	0.0000
C900_8	**0.6070**	0.6030	0.3560	0.3206	0.0964	0.0000	0.1148	0.0112	0.0078	0.0000	0.0000
C900_9	**0.6155**	0.6147	0.3471	0.3227	0.1549	0.0942	0.1241	0.0544	0.0000	0.0000	0.0000
1000_0	**0.6627**	0.6522	0.4589	0.4432	0.1464	0.0773	0.1379	0.1031	0.0142	0.0000	0.0000
1000_1	**0.6964**	0.6873	0.4455	0.4315	0.1480	0.0276	0.1979	0.1152	0.0198	0.0000	0.0000
1000_2	0.6213	**0.6242**	0.4152	0.3995	0.1225	0.0000	0.1270	0.0332	0.0281	0.0000	0.0000
1000_3	**0.6367**	0.6337	0.4263	0.4064	0.1755	0.1281	0.1862	0.1205	0.0265	0.0000	0.0000
1000_4	**0.6578**	0.6549	0.4173	0.3914	0.1579	0.1357	0.1402	0.0248	0.0278	0.0000	0.0000
1000_5	**0.7045**	0.6975	0.4618	0.4468	0.1477	0.0244	0.1946	0.1024	0.0324	0.0000	0.0000
1000_6	**0.6474**	0.6453	0.4199	0.3932	0.2656	0.2324	0.1952	0.0924	0.0256	0.0000	0.0000
1000_7	**0.7024**	0.7002	0.4868	0.4733	0.1649	0.0804	0.2149	0.1226	0.0220	0.0000	0.0000
1000_8	**0.7328**	0.7289	0.4931	0.4780	0.1501	0.1082	0.2119	0.1453	0.0311	0.0000	0.0000
1000_9	**0.6159**	0.6035	0.4413	0.4330	0.2594	0.2426	0.1560	0.0442	0.0159	0.0000	0.0000

3.4.2.3　下界値の解の質の比較　2 つ目の実験は，下界値の解の質の比較を行う．比較対象は BDisLRP，ADisLRP 及び Max-SAT Resolution である．実験の環境を統一するため，各問題例における節集合分割の構造（部分問題）は同一とした．各手法において 1 回の試行を行い，下界値の質の平均を比較する．問題例は Max-SAT Evaluation 2012 で使用された

表 3.8: 実験 3 の結果

割合	BDisLRP	ADisLRP	Resolution
0.1	**0.9629**	0.9539	0.0004
0.2	**0.8561**	0.8355	0.0017
0.3	**0.7444**	0.7163	0.0045

WMax-SATRandom 部門の問題例全 141 問を使用した．なお，問題例の内訳は Max-2SAT 問題が 121 問，Max-3SAT 問題が 40 問である．

表 3.5 と表 3.6 にそれぞれ Max-2SAT 問題と Max-3SAT 問題の結果を示す．また，表 3.7 は問題例別の結果であるが，紙面の都合上，Max-3SAT 問題の結果のみを示す．Bund が BDisLRP を，Adap が ADisLRP を，Res が Max-SAT Resolution をそれぞれ示している．

Max-2SAT 問題の問題例では，分割数 2 を除いて BDisLRP の下界値の質の平均が ADisLRP のそれを上回ったが，差は僅かである．Max-3SAT 問題の問題例では，1000_2 の問題例を除いて，分割数において BDisLRP が ADisLRP 及び Resolution を上回った．また，分割数が小さいほど差が小さくなる傾向は両方のカテゴリで同じであった．

3.4.2.4　共有変数の割合による比較　[黒田 09] の方法に倣い，共有変数の割合に関する性能の比較を行う．共有変数の割合とは，各部分問題に含まれる論理変数の数に対する共有変数の割合である．今回，部分問題の数（分割数）を 10 で固定し，全ての共有変数は全ての部分問題が保持するようにした上で，以下のような問題例を新たに作成した．

- 各部分問題に含まれる論理変数の数：50
- 各部分問題に含まれる節の数：500
- 各部分問題の性質：重み付き Max-3SAT（Random）
- 共有変数の割合：$\{0.1, 0.2, 0.3\}$

問題例全体としては，455 変数 5000 節，410 変数 5000 節，365 変数 5000 節というサイズの問題例である．各共有変数の割合につき 10 問，合計で 30 問を作成した．

表 3.8 に結果を示す．割合は共有変数の割合を示している．

表 3.8 より，全ての共有変数の割合において BDisLRP の下界値の質の平均が ADisLRP 及び Resolution のそれを上回ることがわかった．また，

共有変数の割合が小さくなるほど得られる下界値の質が良くなることもわかった．

特筆すべきなのは，構造を持たない Random な重み付き Max-3SAT 問題ではそれほど良い下界値の質が得られなかったのに対し，構造を持った問題例では非常に良い下界値の質が得られたことである．

3.5　本節のまとめと今後の課題

バンドル法を用いた分散ラグランジュ緩和プロトコルである Bundle DisLRP (BDisLRP: Bundle DisLRP) を提案した．無制約凸関数最適化の手法であるバンドル法と分散環境で上界値を求めることが出来る DisLRP を組み合わせ，バンドル法が分散環境でも適用可能であることを示した．

一般化相互割当問題 (GMAP) と重み付き最大充足化問題 (WMax-SAT) の 2 つの問題で実験を行い，比較，評価した．まず，GMAP の問題例を使用した実験では，エージェント数，財数共に大きな問題例において，劣勾配法を用いた既存の DisLRP である Adaptive DisLRP (ADisLRP) よりも良い上界値がより速く求まることが分かった．また，BDisLRP は大域情報の収集に使用する生成木の形状に性能が依存し，木の深さが小さい生成木の方が性能が良いことが分かった．

次に，WMax-SAT の問題例を使用した実験では，問題の種類や構造及び節集合の分割数に依存するものの，BDisLRP の方が ADisLRP に比べて良い質の下界値を得られることが分かった．また，共有変数が少ないほど良い質の下界値が得られることが分かった．

BDisLRP は生成木の形状に性能が依存するが，ADisLRP は依存しないことが示されている．今後，劣勾配法とバンドル法を組み合わせたハイブリッド手法を開発することによって，BDisLRP の性能を向上させたいと考えている．

バンドル法は，ラグランジュ乗数 μ を更新するために 2 次計画問題を用いる．既存のバンドル法では，他にもトラストリージョン法などが提案されており，それらの手法を BDisLRP に適用することが今後の課題に挙げられる．

第 4 章　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

4.1　はじめに

近年，マルチエージェントシステムの分野において，分散割当は主要な問題の一つとなっている．分散割当とは，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．分散割当の特徴は，割当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．集中型の割当では，割当を行う特別なエージェント（コーディネータ）が存在し，全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 [Smith 90] が挙げられる．しかし，契約ネットプロトコルは欲張り型探索を行なうため，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 [Dias 06] や分散ターゲット追跡 [Bhatti 09] 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 [Modi 05]，分散施設配置問題 [Frank 07] 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も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) [Hirayama 06, Hirayama 07, Hirayama 09] が提案されている．

GMAP は，財をも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．

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も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もたないような，いわゆる過制約な状況もあり得ると考えられる．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（無限の資源をもつ）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，他の全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すむ反面，従来の定式化を変更することに伴い 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 点が挙げられる．

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得ると考えられる．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．

4.2 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（無限の資源をもつ）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．

4.2.1 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一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．

以上を踏まえて，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，

$$
\begin{aligned}
\mathcal{GAP}' \quad & (\text{decide } x_{ij},\ \forall i \in A \cup \{d\},\ \forall j \in J) : \\
\max. \quad & \sum_{i \in A} \sum_{j \in J} p_{ij} x_{ij} \\
\text{s. t.} \quad & \sum_{i \in A \cup \{d\}} x_{ij} = 1, \quad \forall j \in J, \\
& \sum_{j \in J} w_{ij} x_{ij} \leq c_i, \quad \forall i \in A, \\
& x_{ij} \in \{0, 1\},\ \forall i \in A \cup \{d\},\ \forall j \in J,
\end{aligned}
\tag{4.1}
$$

と記述できる．前節の$\mathcal{GAP}$との違いは，全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 (4.1) を緩和したラグランジュ緩和問題は次のようになる．

$$\begin{aligned} L'(\mu) \;=\; & \max. \sum_{i\in A}\sum_{j\in J} p_{ij}x_{ij} \\ & \qquad + \sum_{j\in J}\mu_j \left(1 - \sum_{i\in A\cup\{d\}} x_{ij}\right) \\ & \text{s. t.} \sum_{j\in J} w_{ij}x_{ij} \le c_i,\ \forall i \in A, \\ & \qquad x_{ij} \in \{0,1\},\ \forall i \in A\cup\{d\},\ \forall j \in J. \end{aligned}$$

2.2.2.3 節と同様に，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

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

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iに関するナップサック問題

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

disposal エージェントに関する最大化問題

$$L'_d(\mu) \;=\; \max. \sum_{j\in J}(-\mu_j)x_{dj} \tag{4.2}$$

$$\text{s. t. } x_{dj} \in \{0,1\},\ \forall j \in J, \tag{4.3}$$

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．従って，先のラグランジュ双対問題は，

$$\min. \quad \sum_{i \in A} L'_i(\mu) + L'_d(\mu) + L'_{const}(\mu),$$

となる．2.2.2.3 節との違いは，$\mathcal{GAP}'$ 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．

4.2.1.1　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一つ増加する．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 (4.2) および制約条件 (4.3) によれば，$-\mu_j \geq 0$，すなわち $\mu_j \leq 0$ 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 エージェントの振るま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，ほとんど無視して良い．

具体的な仮想化の方法については，次に述べる．

4.2.2　解法とラグランジュヒューリスティック

本解法の基本的な手続きは以下の通りである．

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 $i \in A$ はそれぞれのナップサック問題を解いて $L'_i(\mu)$ 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 [Reeves 97][Yagiura 07] 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，本研究では，[Hirayama 09] 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通りである．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．

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

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．

4.3 不等式制約緩和に基づく DisLRP

もう一つの方法は，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．

4.3.1 定式化

系全体の問題は次のような整数計画問題となる．

$$\begin{aligned}
\mathcal{GAP}'' \quad & (\text{decide } x_{ij},\ \forall i \in A,\ \forall j \in J) : \\
\text{max.} \quad & \sum_{i \in A} \sum_{j \in J} p_{ij} x_{ij} && (4.4) \\
\text{s. t.} \quad & \sum_{i \in A} x_{ij} \leq 1,\ \ \forall j \in J, && (4.5) \\
& \sum_{j \in J} w_{ij} x_{ij} \leq c_i,\ \ \forall i \in A, && (4.6) \\
& x_{ij} \in \{0, 1\},\ \forall i \in A,\ \forall j \in J. && (4.7)
\end{aligned}$$

2.2.2.3 節の $\mathcal{P}_{\mathcal{DW}}^{\mathcal{GMP}}$ との違いは，$\mathcal{P}_{\mathcal{DW}}^{\mathcal{GMP}}$ 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 (4.5) を緩和すると

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

というラグランジュ緩和問題が得られる．なお，2.2.2.3 節における $\mathcal{P}_{\mathcal{DW}}^{\mathcal{GMP}}$ 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 は任意の実数を取ることができるが，$\mathcal{GAP}''$ 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 には非負制約が伴う．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

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

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 i のナップサック問題

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

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．従って，ラグランジュ双対問題は，

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

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．

4.3.2 解法とラグランジュヒューリスティック

本解法の基本手続きは以下の通りである．

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 i はそれぞれのナップサック問題を解いて $L_i''(\mu)$ 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，

$$\mu_j(1 - \sum_{i \in A} x_{ij}) = 0,\ \forall j \in J \tag{4.8}$$

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 [Reeves 97][Yagiura 07] を用いて更新し，Step 2 へ戻る．なお，更新の際，$\mu_j \geq 0$ を常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．(4.8) 式の左辺は，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．(4.8) 式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 (4.8) 式は必ず 0 になるので，終了条件に (4.8) 式を加える必要が無い．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 (4.8) 式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 (4.8) 式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．

$$\begin{aligned} g_j^{(t)} &\leftarrow 1 - \sum_{i \in A} x_{ij}, \\ temp &\leftarrow \mu_j^{(t)} - \frac{\pi^{(t)}(\mathit{BestUB}^{(t)} - \mathit{BestLB}^{(t)}) g_j^{(t)}}{\sum_{j \in J} (g_j^{(t)})^2}, \\ \mu_j^{(t+1)} &\leftarrow \max\{temp, 0\}. \end{aligned}$$

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．

4.4 実験

4.2 節および 4.3 節で述べた 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速く 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[Beasley] 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_i を減らすため c_i に係数 $x \in \{0.1, 0.2, ..., 0.9\}$ 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毎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ーには，LP-Solve[Eikland] 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 2.93GHz，4 コア 8 スレッド，メモリ 8GB，Windows 7 Professional）を用いた．

表 4.1: 容量係数毎における解の質の平均及び中央値

	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 4.1 及び表 4.2 に，容量係数毎における解の質とラウンド数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 10000 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 4.1 より，平均値，中央値共に容量係数が 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 4.2 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わかる. 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わかる.

表 4.1 及び表 4.2 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} オーダー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わかりづらい．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順位和検定 [岩崎 06] 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 $(x_1, y_1), ..., (x_n, y_n)$ 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，

表 4.2: 容量係数毎におけるラウンド数の平均及び中央値

	Average		Median	
x	Disposal	Inequality	Disposal	Inequality
0.1	199.1833	27.9333	1	1
0.2	1291.3833	613.2000	34	5
0.3	2543.7167	1254.6333	117	13
0.4	2344.9833	1942.4500	259	176
0.5	5685.4000	4599.9000	10000	1423
0.6	5277.1667	5256.5500	5935	6006
0.7	7873.1833	8096.9833	10000	10000
0.8	8084.8667	9673.7833	10000	10000
0.9	7609.7119	10000.0000	10000	10000

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 4.3 に示す．なお，表 4.3 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言えないことがわかる．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言える．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

表 4.3: 全問題例のウィルコクソン符号付順位和検定結果

	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 4.4: 容量係数 0.2〜0.5 の検定結果

	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

るとは言えないが，不等式制約緩和に基づく DisLRP の方が速く収束していることがわかる．

表 4.1 及び表 4.2 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[2]，先ほどと同様にウィルコクソン符号付順位和検定を実施した．その結果を表 4.4 と表 4.5 に示す．表 4.4 と表 4.5 に示された中央値も，表 4.3 と同様，検定対象となったデータに対する値である．

表 4.4 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言える．また表 4.5 より，2 つの解法による解の質に差があるとは言えないが，収束の速さには差があると言える．中央値を見ると，容量

[2]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．

表 4.5: 容量係数 0.6〜0.9 の検定結果

	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

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 の方が良いことがわかる．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速く最適値が求められると言える．

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，全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 (4.8) 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 (4.8) 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．

4.5 本節のまとめと今後の課題

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速く収束する．

- 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速く収束する．

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及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，結果として最適値及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及び最適解を出力し，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挙げられる．

また，今回は選択されなかった財のその後について全く考慮していない．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一つである．

第5章　多層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

5.1　はじめに

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 [Hirayama 06, Hirayama 07, Hirayama 09]．GMAP の目的は，効用が最大となる財の割当てを決定することである．GMAP には 2 種類の制約があり，1 つ目は財はただ 1 つのエージェントに割り当てられるという割当制約，2 つ目は財が割り当てられる際にエージェントが消費する資源量がエージェントの資源容量を超えてはならないというナップサック制約である．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．

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，系全体の財集合に対して，エージェントが十分な資源容量をもたないような，いわゆる過制約な状況もあり得ると考えられる．4 章では，このような状況に対応するために過制約な一般化相互割当問題 (OC-GMAP: Over-Constrained GMAP) を提案した．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詰め込み（パッキング）を求める，いわゆる資源制約付きの集合パッキング問題と見なすことができる．また，OC-GMAP を解くプロトコルとして 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 2 つを提案した．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 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．DisLRP-IBF では，GMAP の割当制約である「各財は 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．

両プロトコルの基本的なアイディアは，割当制約を緩和する，つまり，解空間を拡張することによって過制約な問題を扱えるようにすることだが，これによって生じる問題点が 2 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 2 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．ML-GMAP 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．ML-GMAP の目的は，レイヤー数が最小という条件の下で，効用の総和が最大，も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 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．

1 つ目の手法は 2 段階最適化手法である．この手法では，ML-GMAP 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，もしくはコストが最小となる財の割当を求める．

2 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，も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．

5.2 多層一般化相互割当問題

本節では，ML-GMAP で用いる語句を定義し，ML-GMAP の目的を述べる．

5.2.1 定義

ML-GMAP 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 4 $l \in J \cup \{0\}$ 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，

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

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，

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

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．

5.2.2 目的

ML-GMAP 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 ML-GMAP 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 4 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，ML-GMAP 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 ML-GMAP で定式化して解いた場合，従来の DisLRP-DA および DisLRP-IBF とは異なる最適値および最適解が得られることになる．

ML-GMAP 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，以降の節では効用 p_{ij} をコストと見なして最小化問題の定式化のみ記述する．

5.3 ML-GMAP のための 2 段階最適化手法

本節では，ML-GMAP を解く 1 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 ML-GMAP を解く手順は以下の通りである．

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．

5.3.1 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 $\mathcal{GAPD}$ 又は $\mathcal{GAPI}$ 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 [3]．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

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

に置き換える．

2.2.2 節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 5.1 の DisLRP-DA$_{phase1}$ と DisLRP-IBF$_{phase1}$ の欄にそれぞれ示す．

[3] $\min. \sum_{j \in J} x_{dj}$ 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．

4.2 節及び 4.3 節の定式化と本節の定式化をそれぞれ比較すると，目的関数は元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．従って，1 段階目の問題は 4 章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} が求まる保証は無い．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 [4]．

5.3.2 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．従って，まず最大化問題から最小化問題へと変更する [5]．具体的には，目的関数を

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

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，

$$\sum_{j \in J} x_{dj} = l_{opt}, \tag{5.1}$$

$\mathcal{GAPI}$ の制約式に，

$$\sum_{j \in J} \left(1 - \sum_{i \in A} x_{ij} \right) = l_{opt}, \tag{5.2}$$

をそれぞれ追加する．以後，式 (5.1) 及び (5.2) 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 (5.1) 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．

[4] 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．

[5] p_{ij} 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無い．

従って，disposal エージェントを用いた定式化では 2.2.2 節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良い．一方，不等式制約緩和に基づく定式化では，式 (5.2) 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 (5.2) 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 (5.2) 及び割当制約を緩和すると，

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

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 2.2.2 節と同様である．分解後の部分問題を表 5.1 の DisLRP-IBF$_{phase2}$ 及び DisLRP-DA$_{phase2}$ の欄に示す．

4.2 節及び 4.3 節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．従って，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，4 章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 [6]．

[6] 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恐れがある．従って，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．

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，

$$\begin{aligned} g_j^{(t)} &\leftarrow 1 - \sum_{i \in A} x_{ij}, \\ h^{(t)} &\leftarrow 1 + \frac{1}{l_{opt} - n} \sum_{i \in A} \sum_{j \in J} p_{ij} x_{ij}, \end{aligned}$$

また，ステップ長を，

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

として，次のような更新規則を用いる．

$$\begin{aligned} \mu_j^{(t+1)} &\leftarrow \max\{\mu_j^{(t)} - e \cdot g_j^{(t)}, 0\}, \\ \lambda^{(t+1)} &\leftarrow \lambda^{(t)} - e \cdot h^{(t)}. \end{aligned} \tag{5.3}$$

不等式制約緩和に基づく定式化の場合，$\mu_j \geq 0$ でなければならないので，式 (5.3) によってそれを保証する．

5.4 ML-GMAP のためのペナルティコスト最適化手法

本節では，ML-GMAP を解くための 2 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．

$$\begin{aligned} f(l) = \max. \quad & \sum_{i \in A} \sum_{j \in J} (-p_{ij})\, x_{ij} - r \cdot l \\ \text{s. t.} \quad & \sum_{j \in J} x_{dj} = l, \\ & \sum_{i \in A \cup \{d\}} x_{ij} = 1, \ \ \forall j \in J, \\ & \sum_{j \in J} w_{ij} x_{ij} \leq c_i, \ \ \forall i \in A. \end{aligned} \tag{5.4}$$

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見なす．しかし，式 (5.4) 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．そのときの目的関数は以下のようになる．

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

5.4.1 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，$f^* = f(l_{opt})$ であれば良い．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良い．従って，$l' \in \{l_{opt}, \ldots, n-1\}$ として，十分条件は次のように表すことができる．

$$\begin{aligned} l_{opt} &\neq n \text{ のとき } f(l_{opt}) < f(l'),\ \forall l'. \\ l_{opt} &= n \text{ のとき } r \text{ の値は任意.} \end{aligned}$$

$f(l)$ の最適解を $x^{(l)}$ とする．

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

として式変形すると，最終的に十分条件は

$$r > \frac{b}{l' - l_{opt}},\ \forall l' \tag{5.5}$$

となる．

この十分条件 (5.5) 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 (5.5) を満たす r は事前に計算できない．従って，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 2

$$r > \sum_{i \in A} \sum_{j \in J} p_{ij} \tag{5.6}$$

は，$f^* = f\left(l_{opt}\right)$ であるための十分条件である．

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

$$\sum_{i \in A} \sum_{j \in J} p_{ij} \geq \sum_{i \in A} \sum_{j \in J} p_{ij} x_{ij}$$

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

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

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

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

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，[Hirayama 09] 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．

5.4.2　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

$$\max . \sum_{i \in A} \sum_{j \in J} \left(-p_{ij}\right) x_{ij} - r \sum_{j \in J} x_{dj}$$

に置き換えれば良い．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

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

表 5.1: 既存手法及び提案手法の部分問題別の比較

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

に置き換えれば良い.

制約式は元の問題と同じであるから，2.2.2 節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 5.1 の DisLRP-DA$_{pnt}$ 及び DisLRP-IBF$_{pnt}$ の欄に示す.

4.2 節及び 4.3 節の定式化と本節の定式化をそれぞれ比較すると，目的関数は元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．従って，これらの問題は 4 と同じ解法を用いて解くことができる.

5.5 実験

5.3 節及び 5.4 節で示した 4 つの解法を実装して実験を行った．ML-GMAP 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 4 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 [Beasley] 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_i を減らすため c_i 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た数値を求め，1 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毎に設定することも可

表 5.2: 問題例の一覧と r_{s1} と r_{s2} に関する比較

カテゴリ	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

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．

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．

5.5.1 ペナルティコスト r の導出

本節では，5.4.1 節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無いという実験結果 [Hirayama 09] 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 (5.5) の右辺値に近い値 r_{s1} を，以下の数式を用

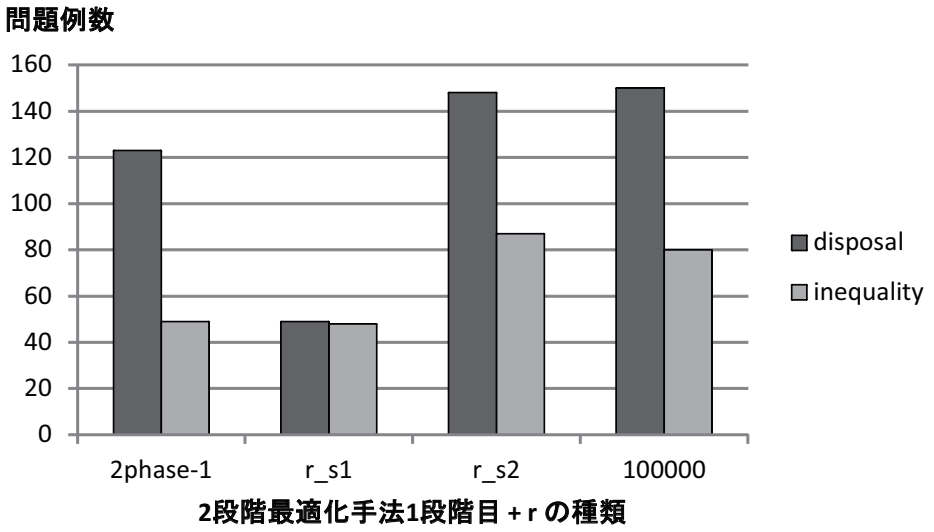

図 5.1: 実験 1 の disposal エージェントと不等式制約緩和の比較

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 (5.6) の右辺値 r_{s2} もあらかじめ計算を行った．その結果を表 5.2 に示す．なお，r_{s1} は容量係数 x 毎にその値が変わるため表 5.2 では容量係数毎に示した．また，r_{s2} は元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，元の問題例のみを示した.

表 5.2 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わかる.

5.5.2 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.

5.5.2.1 最適なレイヤーに関する実験 1 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$_{ph1}$

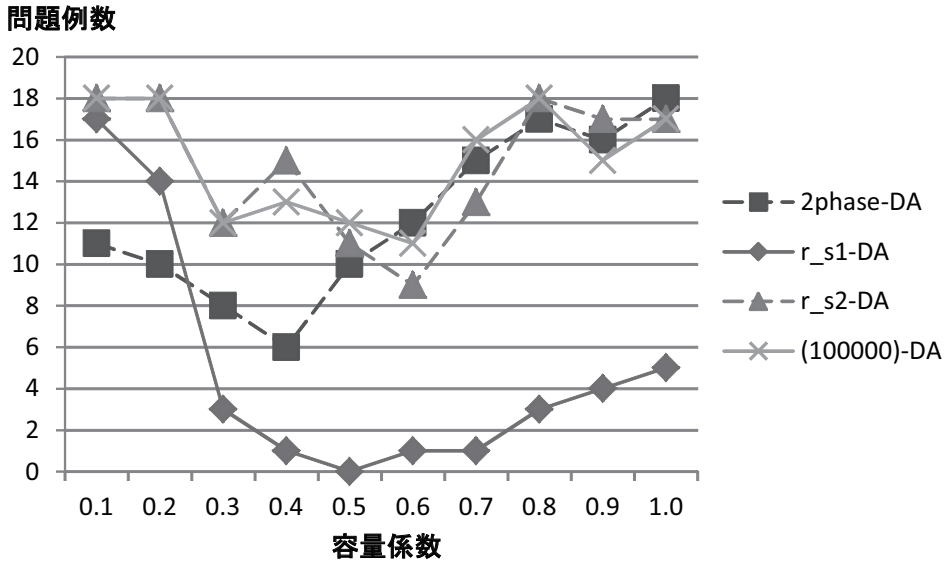

図 5.2: 容量係数による比較（disposal エージェント）

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 5.1，図 5.2 及び図 5.3 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 5.1 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 5.2 及び図 5.3 はそれぞれ DisLRP-DA と DisLRP-IBF の容量係数毎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 5.1 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わかる．また，disposal エージェントを用いた解法において一番良い値を示しているのが $r = 100000$ で，僅差で r_{s2} であることがわかる．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くなっている．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 5.2 と図 5.3 より，DisLRP-DA で

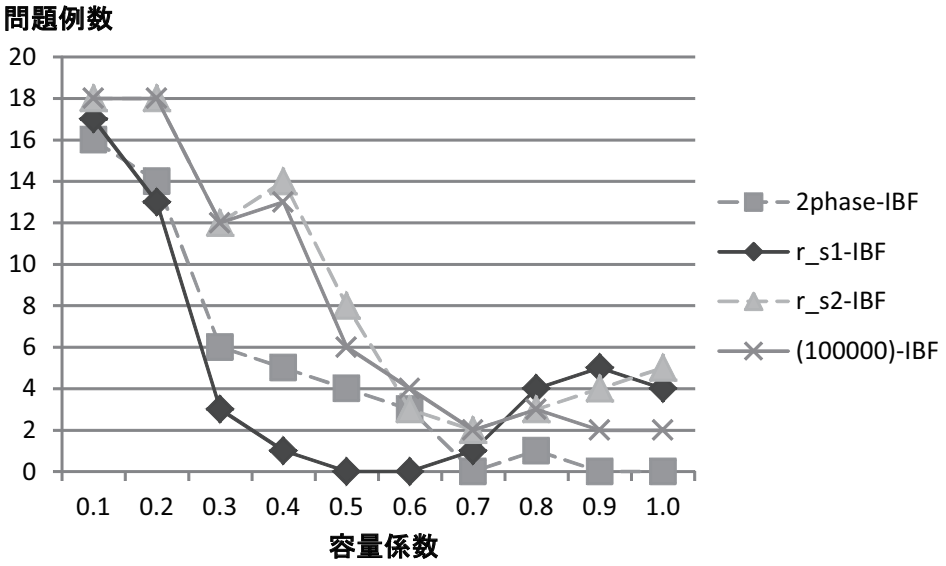

図 5.3: 容量係数による比較（不等式制約緩和）

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基に作られる実行可能解は，上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わかる．これは，r_{s1} の値が小さいため，たとえ良い目的関数値を持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

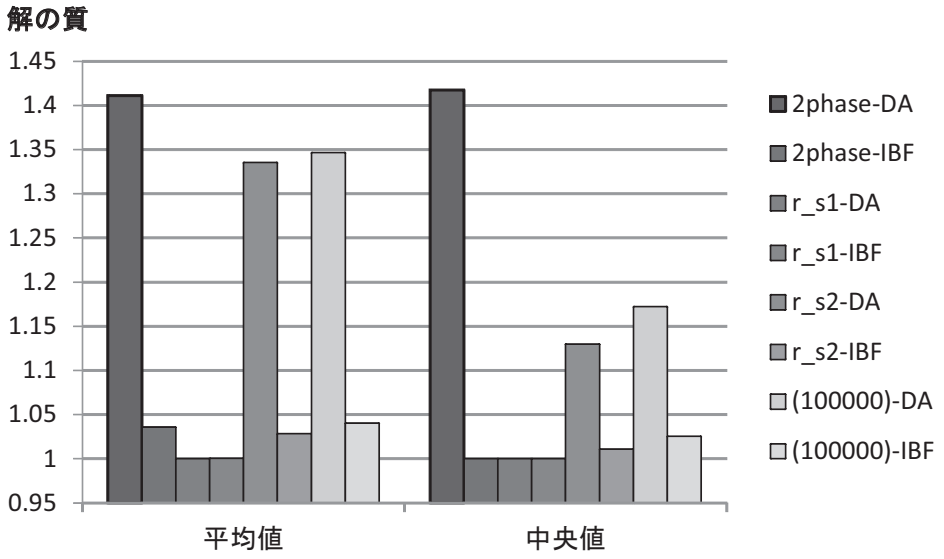

図 5.4: コスト最小の割当てに関する実行可能解の質の評価

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．

5.5.2.2　コスト最小の割当てに関する実験　2 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_{ij} の項のみで解の質を評価した．実験結果を図 5.4 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 5.4 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わかる．また，r の値が小さいほど解の質が良くなる傾

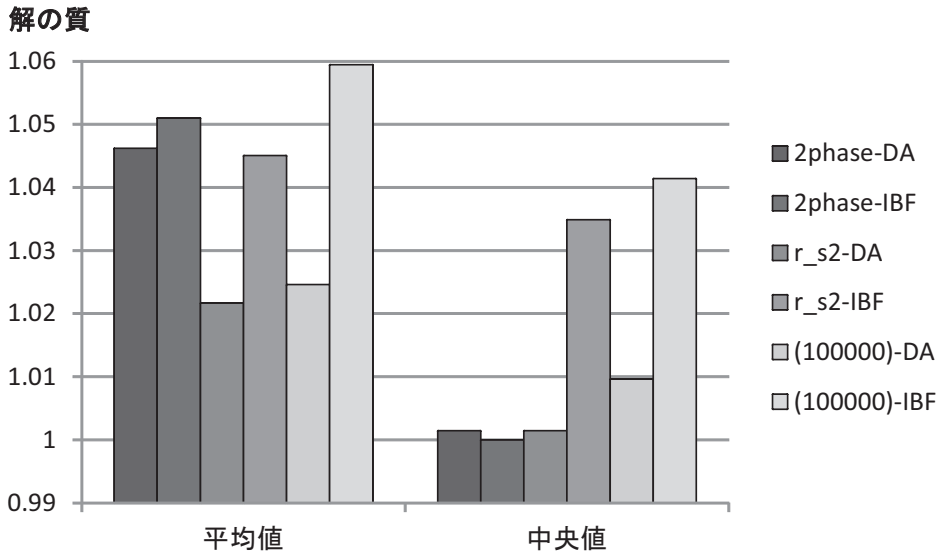

図 5.5: 全ての解法で最適なレイヤーが求められた問題例 30 問の評価

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，下界値を基に計算している上界値も良くなると考えられる．

図 5.4 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 5.1 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毎に異なる．そこで，r_{s1} の DisLRP-DA$_{pmt}$ と DisLRP-IBF$_{pmt}$ を除く [7] 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 5.5 に示す．図 5.5 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わかる．これは図 5.4 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．従って，図 5.5 で図 5.4 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

[7] 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．

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．

5.6 本節のまとめと今後の課題

GMAP の従来の定式化と過制約な問題例を扱う定式化の齟齬を解消するため，ML-GMAP を新たに提案した．また，ML-GMAP 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 2 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わかった．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わかった．

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，ML-GMAP に対する厳密解法の提案が挙げられる．GMAP や ML-GMAP 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．

第6章　増床計画付き患者搬送問題に対する分散ラグランジュ緩和プロトコル

6.1　はじめに

前章までは，汎用的な分散最適化問題である一般化相互割当問題 (GMAP: Generalized Mutual Assignment Problem) の問題点を解消するための新しい定式化を提案し，組合せ最適化問題を分散環境で解く分散ラグランジュ緩和プロトコル (DisLRP: Distributed Lagrangian Relaxation Protocol) を用いて，提案した新しい定式化に対する実行可能解（上界値または下界値）を求める研究について述べてきた．

本章では，DisLRP を実社会の問題に対して応用することを考える．本研究の目的は，実社会の問題に対して新しい定式化を提案し，提案した定式化に対する実行可能解（上界値または下界値）を求めるアルゴリズムを DisLRP に適用して，その有効性を確かめることである．

近年，2011 年に発生した東日本大震災や 2014 年に発生した御嶽山の噴火など，各地で記録的な犠牲者や負傷者を出す自然災害が多数発生している．大規模な災害時においては，患者を速やかに病院へと搬送する必要があるが，どの患者をどの病院に運ぶかを決定する必要がある．このとき，患者の搬送コスト（時間）はできるだけ小さい方が良いことは明らかである．また，大規模な災害時においては，現在の病院の病床数では足りないほどの負傷者が発生する可能性があるとも考えられる．通常，病院における病床数は医療法や厚生労働省による省令によって定められており，簡単に増床することはできない．しかし，災害拠点病院と呼ばれる病院においては，災害時などにおいては特例的に増床することが可能である．

そこで本研究では，患者の搬送計画と病院の増床計画を同時に考慮する増床計画付き患者搬送問題 (PTP: Patient Transportation Problem with Bed Capacity Management) を提案する．この問題の目的は，患者を病院へ搬送するコストと病院が病床数を増やすためにかかるコストを同時に最小化することである．

このような問題を解く場合，全ての情報を一カ所に集めて解く集中型の解法が一般的である．しかし，災害時においては混乱した状況が想定

されるため，全ての情報を収集することは困難であると考えられる．そこで本研究では，全ての情報を送信する必要が無い DisLRP を用いて，定式化に対する実行可能解を求めるアルゴリズムを提案し，実験によりその性能を評価する．

6.2 増床計画付き患者搬送問題

増床計画付き患者搬送問題 (PTP: Patient Transportation Problem with Bed Capacity Management) の目的は，患者を病院へ搬送するコストと病院が病床数を増やすためにかかるコストを同時に最小化することである．本研究では，病院をエージェントとみなし，病院間で通信を行うことによって患者の割当てを決定する．

6.2.1 定式化

PTP は，Dantzig-Wolfe 分解を用いた定式化を用いて，次のように定式化できる．

$$
\begin{aligned}
\min. \quad & \sum_{h\in H}\sum_{p\in P} f_{hp}x_{hp} + \sum_{h\in H}\sum_{b\in B_h} g_{hb}y_{hb} \\
\text{s. t.} \quad & \sum_{h\in H} x_{hp} = 1, \ \ \forall p \in P, && (6.1)\\
& \sum_{b\in B_h} y_{hb} \le 1, \ \ \forall h \in H, && (6.2)\\
& \sum_{h\in H} x_{hp} \le c_h + \sum_{b\in B_h} b y_{hb}, \ \ \forall h \in H, && (6.3)\\
& x_{hp} \in \{0,1\}, \ \forall h \in H, \ \forall p \in P, && (6.4)\\
& y_{hb} \in \{0,1\}, \ \forall h \in H, \ \forall b \in B_h. && (6.5)
\end{aligned}
$$

ここで，$H = \{1,\ldots,m\}$ は病院の集合，$P = \{1,\ldots,n\}$ は患者の集合，B_h は病院 h が増床できる単位病床数の集合である．例えば，病院 h が病床数を 10 ずつ最大 60 床まで増床可能なとき，$B_h = \{0,10,\ldots,60\}$ となる．f_{hp} は病院 h に患者 p を搬送する場合のコスト，g_{hb} は病院 h が病床数 b を増床する場合のコスト，c_h は病院 h の当初保有している既定の病床数である．また，x_{hp} は，病院 h に患者 p が割り当てられる場合は 1，そうでない場合は 0 に設定される決定変数，y_{hb} は，病院 h が病床数を b 増

床する場合は 1，そうでない場合には 0 に設定される決定変数である．問題の目的は，(6.1) 各患者はただ 1 つの病院に割り当てられ（割当制約），(6.2) 病院は増床しないか選択可能な増床数のうちいずれか 1 つを選択し，(6.3) 病院は受け入れ患者の和が既定病床数と増床数の和を超えてはならない（容量制約）という条件の下で，患者の搬送コストと病院の増床コストの和を最小化することである．

この問題は，式 (6.1) を一般化割当問題 [Savelsbergh 97] における割当制約，式 (6.3) を容量制約付き施設配置計画 [Geoffrion 78] の容量制約を組み合わせた定式化となっている．

なお，PTP をラグランジュ分解を用いた定式化で表すことも可能である．しかし，2.2.5 節で例に挙げた GMAP の場合と同様，分解後における一部の部分問題が必ず整数解となり，Dantzig-Wolfe 分解を用いて定式化した場合とラグランジュ分解を用いて定式化した場合と，求まる下界値が同じとなる．従って本研究では，より部分問題の数が少ない Dantzig-Wolfe 分解を用いた定式化を使用する．

以降，紙面の都合上，0-1 制約 (6.4)(6.5) を省略する．

割当制約 (6.1) を緩和して得られたラグランジュ緩和問題は次のようになる．

$$
\begin{aligned}
L(\mu) = \min . \quad & \sum_{h\in H}\sum_{p\in P}(f_{hp}-\mu_p)\,x_{hp} \\
& \qquad + \sum_{h\in H}\sum_{b\in B_h} g_{hb}y_{hb} + \sum_{p\in P}\mu_p \\
\text{s. t.} \quad & \sum_{b\in B_h} y_{hb} \le 1, \ \ \forall h \in H, \\
& \sum_{h\in H} x_{hb} \le c_h + \sum_{b\in B_h} b y_{hb}, \ \ \forall h \in H.
\end{aligned}
$$

ここで，μ_p は患者 p に対する実数値パラメータで患者 p のラグランジュ乗数である．また，$\mu = (1, \ldots, \mu_n)$ はラグランジュ乗数ベクトルである．

このラグランジュ緩和問題を病院ごとに分解すると，次のような部分

問題を得ることができる．

$$
\begin{aligned}
L_i(\pi_i(\mu)) \ = \min . \quad & \sum_{p\in P} (f_{hp} - \mu_p)\, x_{hp} + \sum_{b\in B_h} g_{hb} y_{hb} \\
\text{s. t.} \quad & \sum_{b\in B_h} y_{hb} \leq 1, \\
& \sum_{h\in H} x_{hb} \leq c_h + \sum_{b\in B_h} b y_{hb}.
\end{aligned}
$$

6.2.2 解法とラグランジュヒューリスティック

PTP に対する DisLRP の概略は，次の通りである．

Step 0 各病院 $h \in H$ が，パラメータ t を 1，ラグランジュ乗数ベクトル $\pi_i(\mu^{(t)}) = \pi_i(\mu^{(1)})$ の要素を全て 0 で初期化する．

Step 1 現在の $\pi_i(\mu^{(t)})$ の値のもとで，各病院 h は，ラグランジュ緩和問題を任意の厳密解法を用いて最適解を求める．

Step 2 各病院 h は，Step 1 で求めた最適解を各自の近傍に送る．

Step 3 各病院 h は，自身の最適解及び近傍から送られてきた最適解を基に劣勾配ベクトル $\rho_i(g(\mu^{(t)}))$ を計算する．

Step 4 ラグランジュヒューリスティックにより上界値を計算する．

Step 5 各病院 h は，ラグランジュ乗数ベクトルを $\mu^{(t+1)}$ に更新する．また，$t \leftarrow t+1$ として **Step 1** へ戻る．

基本的な動作は，2.3 節で述べたものと同じである．

本研究では，**Step 5** におけるラグランジュ乗数を更新に劣勾配法を使用する．更新式は，式 (2.13) を用いる．劣勾配は緩和した制約式となるので，PTP の緩和問題に対する劣勾配は次のようになる．

$$
g_i^{(t)} = 1 - \sum_{p\in P} x_{hp}^*.
$$

ここで，x_{hp}^* は $\mu^{(t)}$ におけるラグランジュ緩和問題の最適解である．

既存の DisLRP である DisLRP$_L$ は，ステップ長に固定長を用いる．この場合，μ を更新するのに大域情報が必要では無いため生成木を用いた大

域情報の収集を行う必要が無く，通信量を低く抑えることが出来る．一方，適応的な価格更新を用いた DisLRP (ADisLRP: Adaptive DisLRP) では，ステップ長に以下の式を用いる．

$$l^{(t)} = \frac{\pi(\min\{ub\} - \max\{lb\})}{\|1 - \sum_{p \in P} x^*_{hp}\|^2}.$$

ここで，$\pi \in \mathbb{R}^+$ は制御パラメータ，$\min\{ub\}$ はラウンド t までに知られている最小の上界値，$\max\{lb\}$ はラウンド t までに知られている最大の下界値である．ADisLRP の手法で μ を更新するためには，*ub* や *lb* などの大域情報が必要である．従って，生成木を用いた大域情報の収集が必要となるため，通信量が増大する．

ある μ の値の元で，制約 (6.1) から (6.5) を満たすような実行可能解を求めるアルゴリズムをラグランジュヒューリスティックと呼ぶ．本研究では，次のような増床計画付き患者搬送問題に対するラグランジュヒューリスティックを提案する．

Step 1 各患者を，割当制約を満たしている場合は集合 P_T に，重複して病院に割り当てられている場合は集合 P_F に，どの病院にも割り当てられていない場合は集合 P_N に振り分ける．$P = P_T$ の場合は，全ての割当制役が満たされているため終了する．

Step 2 P_F に含まれる各患者について，最も搬送コストが低い病院に割り当て，$P_T \leftarrow P_T \cup P_F$ とする．

Step 3 病院に患者 P_T のみが割り当てられたと仮定して，各病院の残り病床数を求める．

Step 4 P_N に含まれる患者 p について，割当可能な病院の集合 $H'_p \subseteq H$ を考える．$P_N = \emptyset$ の場合，全ての患者を割り当てられたので，アルゴリズムを終了する．$H'_p = \emptyset$ の場合，解を求めることに失敗したとして，アルゴリズムを終了する．

Step 5 患者 p にとって最も搬送コストが低い病院 $h \in H'_p$ を選ぶ．容量制約 (6.3) を満たしている場合，患者 p を病院 h に割り当てる．容量制約 (6.3) を違反し，かつ増床可能な場合，割り当てられた病院は最も少ない増床コスト分だけ増床を行い，患者 p を病院 h に割り当てる．容量制約 (6.3) を違反し，かつ増床も不可能な場合，$H'_p \leftarrow H'_p \setminus \{h\}$ として，**Step 4** に戻る．

このヒューリスティックでは，すでに割当制約を満たしている患者についてはそのままの割当てとし，まず重複して病院に割り当てられている患者を優先して考える．**Step 2** において，重複して割り当てられている患者は，最も搬送コストが低い病院に割り当てる．重複して割り当てられている患者を病院から取り除くことにより，病院は空き病床数を増やすことが出来る．

次に，どの病院にも割り当てられていない患者を考える．まず，患者にとって最も搬送コストが低い病院に割当てることを試みる．もし，病院に空き病床がある場合，病院は患者を受け入れる．もし，病院に空き病床がない場合，病院は増床を行って患者を受け入れる．このとき，病院は増床コストを支払わなければならない．もし，病院に空き病床がなくこれ以上増床も行えない場合は，病院は患者の受け入れを拒否する．患者は，次に搬送コストが低い病院を選び，受け入れてもらえるかどうかの判定を行う．仮に，全ての病院で患者が受け入れられない場合は，解なしとしてヒューリスティックを終了する．

このヒューリスティックは，患者の搬送コスト最小化を優先したヒューリスティックである．これは，患者の治療を早く始められるようにすることを目的としている．

6.3 実験

6.2.2 節で述べた解法を実装して実験を行った．PTP 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．そこで本研究では，大域情報を使用しないステップ長 $l^{(t)}$ が固定長の DisLRP$_L$ と，$l^{(t)}$ に大域情報を使用する ADisLRP の間で比較を行った．実験の目的は，プロトコル終了時点で最も良い上界値を最も良い下界値で割った *BestUB*/*BestLB* で定義される解の質がより速く 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．

実験に使用した問題例は，[Beasley] 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に対して，目的関数に病院の増床コストをランダムに設定し，単位病床数を 10 として病院は最大 60 床追加できるものとして，新たに作成した．

実験環境には，デスクトップ PC (CPU: Intel Core i7-2600K@3.4GHz, Memory: 12GB, OS: 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 1.8.0 25 で実装し，上記の実験環境上で行う．各病院の部分問題を解くソルバーには，整数計画問題を解

表 6.1: ADisLRP と DisLRP$_L$ の比較結果

		Quality of Values		Elasped Time(s)		BestUB Time(s)	
Cat.	Instance	Adap	Stat	Adap	Stat	Adap	Stat
a	c5100	1.0169	1.0169	107	**50**	58	**16**
a	c5200	1.0100	**1.0098**	183	**112**	46	**26**
a	c10100	1.0501	**1.0496**	365	**146**	156	**24**
a	c10200	**1.1197**	1.1221	742	**357**	**0**	209
a	c20100	**1.0965**	1.1644	201	**93**	187	**51**
a	c20200	1.0573	**1.0570**	1476	**1014**	1234	**878**
b	c5100	**1.0197**	1.0201	253.133	**156**	53	**31**
b	c5200	1.0093	**1.0039**	278	**179**	**93**	171
b	c10100	**1.0489**	1.0499	262	**115**	108	**23**
b	c10200	1.0450	**1.0228**	403	**250**	**184**	208
b	c20100	1.0859	1.0859	360	**96**	261	**28**
b	c20200	1.0870	**1.0287**	2135	**1121**	2025	**1035**
c	c5100	1.0189	1.0189	94	**51**	39	**10**
c	c5200	1.0101	**1.0094**	512	**159**	335	**151**
c	c10100	1.0517	1.0517	208	**135**	67	**57**
c	c10200	1.0462	**1.0229**	**384**	415	109	**92**
c	c20100	**1.1685**	1.1702	420	**76**	246	**47**
c	c20200	1.0961	**1.0646**	1879	**1116**	1576	**858**
d	c5100	1.0103	1.0103	250	**107**	241	**72**
d	c5200	1.0056	1.0056	461	**123**	179	**41**
d	c10100	**1.0253**	1.0328	217	**74**	96	**8**
d	c10200	**1.0525**	1.0574	339	**155**	**0**	23
d	c20100	**1.0736**	1.0765	355	**88**	193	**31**
d	c20200	1.0646	**1.0267**	1672	**891**	1482	**594**
e	c5100	**1.0064**	1.0075	1041	**959**	**200**	626
e	c5200	1.0032	1.0032	1310	**966**	327	**152**
e	c10100	1.0061	**1.0058**	4701	**3099**	**410**	660
e	c10200	**1.0220**	1.0225	2842	**2046**	1164	**299**
e	c10400	1.0057	**1.0053**	6017	**3127**	3462	**926**
e	c15900	1.0029	**1.0022**	12717	**10011**	**5790**	9596
e	c20100	**1.0328**	1.0416	1695	**627**	1530	**94**
e	c20200	**1.0132**	1.0920	6608	**904**	5010	**900**

く汎用的なソルバーである IBM ILOG CPLEX 12.6 を使用した．

固定長のステップ長として，$l^{(t)} = 1$ を使用した．可変長のステップ長は，[Hirayama 09] の設定に従い，π の初期値を 2 とし，各ラウンドまでの最小の上界値 ub が 100 ラウンド連続で更新されなければ，π の値を 1/2 倍するものとする．

プロトコルの終了条件は，劣勾配法のパラメータ π が最終的に $\pi < 10^{-6}$ となった時点で終了する．なお，固定長のステップ長では劣勾配の更新に π を使用しないが，終了条件を同一とするために計算を行っている．

実験結果を表 6.1 に示す．Cat. は問題例のカテゴリー，Instance は問題例の名前を表す．例えば，問題例の名前が c5100-1 の場合，病院数が 5,

患者数が 100 であることを表す．Adap は ADisLRP，Stat は DisLRP$_L$ の実験結果である．また，Quality of Values は，*BestUB*/*BestLB* で定義される解の質，Elasped Time (S) はプロトコルが終了した CPU 時間（秒），BestUB Time (s) は最も良い実行可能解を発見したときの CPU 時間（秒）である．

表 6.1 より，解の質は両手法であまり差は見られなかった．また，どちらの手法が良いという傾向は見られなかった．一方，プロトコルの実行時間は DisLRP$_L$ の方が短く，最も良い実行可能解を発見した時間も DisLRP$_L$ の方が早い結果となった．これは，ADisLRP が生成木を用いて大域情報を伝播させるのに処理の時間が必要なのに対して，DisLRP$_L$ はその必要がないからだと考えられる．

また，μ の更新に大域情報を使用しなくても十分良い質の実行可能解を早く求められることが分かった．大規模な災害においては，通信インフラが遮断されるなど大きな混乱が予想される．従って，通信量が比較的少なくてすむ DisLRP$_L$ の方が，PTP に対しては有効であると考えられる．

6.4 本章のまとめと今後の課題

患者の搬送計画と病院の増床計画を同時に考慮する増床計画付き患者搬送問題を定式化した．また，増床計画付き患者搬送問題の実行可能解を求めるラグランジュヒューリスティックアルゴリズムを開発した．ラグランジュ乗数の更新に大域情報を使用しない DisLRP$_L$ と大域情報を使用する ADisLRP に対してラグランジュヒューリスティックを適用して実験した結果，DisLRP$_L$ の方が早く質の良い実行可能解を発見できることが分かった．

大規模な災害においては，通信インフラの断絶といった状況が発生する可能性がある．混乱状況においては，正しい情報が伝わってこない可能性も考慮する必要がある．従って，通信の途絶や通信のノイズに頑健なプロトコルの開発が，今後の課題の 1 つである．

本研究では，患者の搬送コスト（時間）と病院の増床コスト（お金）が 1 つの目的関数で同列に扱った．しかし，時間とお金を同次元で比較することは本質的に困難である．今後，搬送コストと増床コストをそれぞれ別の目的関数とした多目的最適化問題へ拡張していきたい．

また本研究では，静的な状況についてしか考慮されていない．しかし，現実には状況は刻一刻と変化するため，本研究のプロトコルでは対応できない可能性が高い．従って，今後は動的な環境に対応できるプロトコ

ルが必要であると考えられる.

第7章　おわりに

7.1　本研究のまとめ

本研究は，組合せ最適化問題を分散環境で解くためのプロトコルである分散ラグランジュ緩和プロトコル (DisLRP: Distributed Lagrangian Relaxation Protocol) に注目し，バンドル法を用いることによって DisLRP の性能を向上させる研究を行った．また，新しい分散最適化問題を 3 種類提案し，それぞれに対して DisLRP をベースとした解法を提案した．

以下に，本論文のまとめと今後の課題について述べる．

1 章では，本研究の背景と目的について述べた．

2 章では，本研究で注目した分散ラグランジュ緩和プロトコル (DisLRP: Distributed Lagrangian Relaxation Protocol) について概説した．まず，Dantzig-Wolfe 分解を用いた定式化とラグランジュ分解を用いた定式化について定義し，それぞれについて例を示した．また，DisLRP で使用する命題や定理について述べた．次に，DisLRP の基本的な動作と，既存の DisLRP が抱えている問題点について指摘した．

3 章では，既存の DisLRP とバンドル法を組み合わせた新しいプロトコルである Bundle DisLRP (BDisLRP) を提案した．BDisLRP は，最大化問題に対する上界値または最小化問題に対する下界値を導出することに特化したプロトコルである．バンドル法は，集中環境で凸最適化問題を解くために使用されるアルゴリズムであり，本研究では分散環境の解法である DisLRP にバンドル法を適用した．その際，集中環境では考慮されない，通信の遅れに対応する工夫を行った．Dantzig-Wolfe 分解を用いた定式化を持つ一般化相互割当問題 (GMAP: Generalized Mutual Assignment Problem)，およびラグランジュ分解を用いた定式化を持つ重み付き最大充足化問題 (WMax-SAT: Weighted Maximum Satisfiabilit Problem) の両問題について実験を行い，既存のプロトコルである Adaptive DisLRP (ADisLRP) と比較した．GMAP の問題例を使用した実験では，エージェント数，財数共に大きな問題例において，劣勾配法を用いた既存の DisLRP である Adaptive DisLRP (ADisLRP) よりも良い上界値がより速く求まることが分かった．また，BDisLRP は大域情報の収集に使用する生成木の形状に性能が依存し，木の深さが小さい生成木の方が性能が良いことが分かった．WMax-SAT の問題例を使用した実験では，問題の種類や構造及び節集合の分割数に依存するものの，BDisLRP の方が ADisLRP に比べて良い質の下界値

を得られることが分かった．また，共有変数が少ないほど良い質の下界値が得られることが分かった．

4 章では，GMAP に対して過制約な状況を定義し，新しい分散最適化問題として過制約な一般化相互割当問題 (OC-GMAP: Over-Constrained Generalized Mutual Assignment Problem) を提案した．また，DisLRP をベースとした OC-GMAP を解くための解法を 2 種類の解法，disposal エージェントを用いた DisLRP (DisLRP-DA: DisLRP with a Disposal Agent) と，不等式制約緩和に基づく DisLRP (DisLRP-IBF: DisLRP with Inequality Based Formulation) を提案した．両プロトコルを実験により比較した結果，過制約度が低い問題例に対しては DisLRP-DA が，過制約度が高い問題例に対しては DisLRP-IBF の方が良い結果を示すことが分かった．

5 章では，既存の GMAP と OC-GMAP の齟齬を解消するための新しい分散最適化問題である多層一般化相互割当問題 (ML-GMAP: Multi-Layered Generalized Mutual Assignment Problem) を提案した．また，ML-GMAP を解くための手法として，DisLRP をベースとした 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た．両手法を実験により比較した結果，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，十分大きなペナルティコスト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わかった．しかし，ペナルティコストの値によって，得られるレイヤーと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．

6 章では，実世界の応用例として，増床計画付き患者搬送問題 (PTP: Patient Transportation Problem with Bed Capacity Management) を提案した．また，DisLRP をベースとした実行可能解を求めるプロトコルを提案した．実験の結果，ラグランジュ乗数の更新に大域情報を必要としなくても，早く質の良い実行可能解を得られることが分かった．

7.2 今後の課題

本節では，本研究に対する今後の課題について述べる．

7.2.1 プロトコルの非同期化

DisLRP は，基本的に同期的なプロトコルである．同期的プロトコルは，ネットワークにボトルネックが発生した場合，プロトコル全体の性能が

低下する可能性がある．

今後の展開として，ラグランジュ乗数 μ の非同期的な更新が挙げられる．プロトコルの非同期化によって，ボトルネックの発生に対して頑健なプロトコルが実現可能であると考えられる．

7.2.2 完全なプライバシーの確保

DisLRP は，目的関数や制約式の係数を他のエージェントに対して公開しなくて良いという点で，プライバシー性が確保されている．しかし，プロトコルの性質上，あるラグランジュ乗数 μ の下での割当てや目的関数値は他のエージェントに送信しなければならず，完全にプライバシーを保障されているわけではない．

今後の課題として，このような割当てなども他のエージェントに知られなくてすむような，完全なプライバシーが確保されるプロトコルの開発が挙げられる．

7.2.3 多層一般化相互割当問題に対する多目的最適化への拡張

ML-GMAP では，まず割り当てられない財の数を最小化する．次に，割り当てられない財の数を制約で固定し，効用を最大化またはコストを最小化するような最適解を求める．

今後の課題は，割り当てられない数と効用の最大化（コストの最小化）の両方を目的関数とする，多目的最適化問題への拡張が挙げられる．多目的最適化問題へ拡張することによって，問題を解くユーザに対してより多くの選択肢を提供することが可能となる．

7.2.4 増床計画付き患者搬送問題に対する種々の拡張

増床計画付き患者搬送問題は，実問題への応用を強く意識した問題である．しかし本研究では，静的な状況についてしか考慮されていない．現実には状況は刻一刻と変化するため，本研究のプロトコルでは対応できない可能性が高い．従って，今後は動的な環境に対応できるプロトコルが必要であると考えられる．

患者の搬送コストは時間で定量化でき，病院の増床コストはお金で定量化が可能である．本研究で提案した PTP は，この搬送コストと増床コストを同一の目的関数で扱っている．しかし，時間とお金を比較するの

は本質的に難しい．今後，搬送コストと増床コストをそれぞれ別の目的関数とした多目的最適化問題へ拡張していきたい．

また，大規模な災害においては，通信インフラの断絶といった状況が発生する可能性がある．混乱状況においては，正しい情報が伝わってこない可能性も考慮する必要がある．従って，通信の途絶や通信のノイズに頑健なプロトコルの開発も，今後の課題の 1 つである．

謝辞

本研究を進めるにあたり，指導教官である平山勝敏教授に深く感謝致します．平山先生には，研究の内容に関してはもちろんのこと，研究の進め方，研究及び研究者のあるべき姿まで，丁寧にご指導くださいました．また，基礎研究の重要性と楽しさを教えてくださいました．先生にご教授頂いたことを深く胸に刻み，今後の研究生活に役立てていきたいと考えております．ここに改めて，深い感謝の意を表します．

次に，沖本天太准教授に感謝いたします．沖本先生とは，先生がまだ学生の頃から親交があり，研究の議論などをしてまいりました．先生が神戸大学大学院海事科学研究科の准教授になられてからは，研究室の指導教官として非常に的確なアドバイスを何度も頂きました．改めて，深く御礼申し上げます．

また，山村三朗名誉教授には，研究の進捗状況や普段の生活に関していつも気にかけて頂きました．ここに，深く感謝いたします．

本論文の査読をお願い致しました今井昭夫教授および堀口知也教授に感謝いたします．先生方には，数多くの有意義なコメントをいただきました．改めまして，本論文を査読して下さった先生方に感謝致します．

共同研究のミーティングや国際・国内学会において貴重な質問，コメントをくださった神戸大学大学院 田村直之教授，番原睦則准教授，宋剛秀助教，国立情報学研究所 井上克己教授及び九州大学大学院 横尾真教授にも深く感謝いたします．

また，同研究室の卒業した先輩同期の仲間，後輩達に感謝致します．ここまで研究を遂行できたのは，時に能力を高めあい，時に協力しあえたからこそです．

最後に，研究者の道を歩むという私のわがままを暖かく見守ってくださり，また辛抱強く支援してくださった両親に対して深く感謝の意を表します．

参考文献

[Aristomenopoulos 12] Aristomenopoulos, G., Kastrinogiannis, T., and Papavassiliou, S.: Multiaccess Multicell Distributed Resource Management Framework in Heterogeneous Wireless Networks, *IEEE Transactions on Vehicular Technology*, Vol. 61, No. 6, pp. 2636–2650 (2012)

[Beasley] Beasley, J. E.: Welcome to OR-Library

[Bhatti 09] Bhatti, S. and Xu, J.: Survey of Target Tracking Protocols Using Wireless Sensor Network, in *5th International Conference on Wireless and Mobile Communications*, pp. 110–115 (2009)

[Bui 04] Bui, M., Butelle, F., and Lavault, C.: A Distributed Algorithm for Constructing a Minimum Diameter Spanning Tree, *Journal of Parallel and Distributed Computing*, Vol. 64, No. 5, pp. 571–577 (2004)

[Dias 06] Dias, M., 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)

[Eikland] Eikland, K. and Notebaert, P.: lpsolve — Free Development software downloads at SourceForge.net

[Frank 07] Frank, C. and Römer, K.: Distributed facility location algorithms for fl xible configuration in *Proc. International Conference on Distributed Computing in Sensor Systems (DCOSS-2007)*, pp. 124–141 (2007)

[Gallager 83] Gallager, R. G., Humblet, P. A., and Spira, P. M.: A Distributed Algorithm for Minimum-Weight Spanning Trees, *ACM Transactions on Programming Languages and Systems*, Vol. 5, No. 1, pp. 66–77 (1983)

[Geoffrion 78] Geoffrion, A. and Bride, R. M.: Lagrangean relaxation applied to capacitated facility location problems, *AIIE transactions*, Vol. 10, No. 1, pp. 40–47 (1978)

[Guignard 87] Guignard, M. and Kim, S.: Lagrangean decomposition: A model yielding stronger lagrangean bounds, *Mathematical Programming*, Vol. 39, pp. 215–228 (1987)

[Hirayama 06] Hirayama, K.: A new approach to distributed task assignment using Lagrangian decomposition and distributed constraint satisfaction, in *AAAI-2006*, pp. 660–665 (2006)

[Hirayama 07] Hirayama, K.: An α-approximation protocol for the generalized mutual assignment problem, in *AAAI-2007*, pp. 744–749 (2007)

[Hirayama 09] Hirayama, K., Matsui, T., and Yokoo, M.: Adaptive price update in distributed Lagrangian relaxation protocol, in *AAMAS-2009*, pp. 1033–1040 (2009)

[Karypis 97] Karypis, G., Aggarwal, R., Kumar, V., and Shekhar, S.: Multilevel hypergraph partitioning: applications in VLSI domain, in *DAC '97*, pp. 526–529 (1997)

[Korte 05] Korte, B. and Vygen, J.: *Combinatorial Optimization: Theory and Algorithms*, Springer Publishing Company, Incorporated, 3rd edition (2005)

[Kugel 10] Kugel, A.: Improved Exact Solver for the Weighted Max-SAT Problem, in *PoS '10*, pp. 15–27 (2010)

[Li 09] Li, C., Manyá, F., Mohamedou, N., and Planes, J.: Exploiting Cycle Structures in Max-SAT, in *SAT '09*, pp. 467–480 (2009)

[Luo 13] Luo, L., Chakraborty, N., and Sycara, K.: Distributed algorithm design for multi-robot generalized task assignment problem, in *IEEE/RSJ International Conference on Intelligent Robots and Systems 2013 (IROS 2013)*, pp. 4765–4771 (2013)

[Modi 05] Modi, P. J., Shen, W.-M., Tambe, M., and Yokoo, M.: Adopt: asynchronous distributed constraint optimization with quality guarantees, *Artificia Intelligence*, Vol. 161, No. 1-2, pp. 149–180 (2005)

[Posta 12] Posta, M., Ferland, J. A., and Michelon, P.: An exact method with variable fixin for solving the generalized assignment problem, *Computational Optimization and Applications*, Vol. 52, No. 3, pp. 629–644 (2012)

[Reeves 97] Reeves, C. R. ed.: *Modern heuristic techniques for combinatorial problems*, Blackwell (1997)

[Savelsbergh 97] Savelsbergh, M.: A branch-and-price algorithm for the generalized assignment problem, *Operations Research*, pp. 831–841 (1997)

[Schramm 92] Schramm, H. and Zowe, J.: A Version of the Bundle Idea for Minimizing a Nonsmooth Function: Conceptual Idea, Convergence Analysis, Numerical Results, *SIAM Journal on Optimization*, Vol. 2, No. 1, pp. 121–152 (1992)

[Smith 90] Smith, R. G.: The contract net protocol: high-level communication and control in a distributed problem solver, in *IEEE Transactions on Computers*, pp. 29(2):1104–1113 (1990)

[Yagiura 07] Yagiura, M. and Ibaraki, T.: *Handbook of Approximation Algorithms and Metaheuristics*, chapter 48, Chapman & Hall/CRC (2007)

[Yeoh 08] Yeoh, W., Felner, A., and Koenig, S.: BnB-ADOPT: an asynchronous branch-and-bound DCOP algorithm, in Padgham, L., Parkes, D. C., Müller, J. P., and Parsons, S. eds., *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008), Estoril, Portugal, May 12-16, 2008, Volume 2*, pp. 591–598, IFAAMAS (2008)

[Yokoo 98] Yokoo, M., Durfee, E. H., Ishida, T., and Kuwabara, K.: The Distributed Constraint Satisfaction Problem: Formalization and Algorithms, *IEEE Trans. Knowl. Data Eng.*, Vol. 10, No. 5, pp. 673–685 (1998)

[岩崎 06] 岩崎 学：ノンパラメトリック法, 東京図書 (2006)

[黒田 09] 黒田 陽之, 平山 勝敏 : Multi-MaxSAT : ラグランジュ分解・調整法を用いた Weighted Max-SAT 問題の解法, 電子情報通信学会論文誌 D, Vol. 92, No. 1, pp. 51–60 (2009)

[佐々木 05] 佐々木 淳, 櫟 粛之, 増山 繁：負荷分散のための非同期分散分枝限定法, 電子情報通信学会技術研究報告. COMP, コンピュテーション, Vol. 104, No. 642, pp. 23–32 (2005)

本論文に関する原著論文

国内学術論文誌

1 花田研太, 平山勝敏，沖本天太: 分散ラグランジュ緩和プロトコルにおけるバンドル法, 人工知能学会論文誌, Vol.31, No.2, pp.C-F75_1-10, 2016 年 2 月.

2 花田研太, 平山勝敏: 多層一般化相互割当問題の定式化とその解法, 電子情報通信学会論文誌 D，Vol.J96-D, No.12, pp.2908–2919, 2013 年 12 月.

3 花田研太, 平山勝敏: 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, 情報処理学会論文誌, Vol.53, No.11, pp.2370–2378, 2012 年 11 月.

国際会議（査読あり）

1 ◯Kenta Hanada, Katsutoshi Hirayama, Tenda Okimoto: Effect of Bundle Method in Distributed Lagrangian Relaxation Protocol, AAAI-15 Workshop on Planning, Search, and Optimization (PlanSOpt-15), January 2015, Austin, USA.

2 ◯Kenta Hanada, Katsutoshi Hirayama: Distributed Lagrangian Relaxation Protocol for the Over-constrained Generalized Mutual Assignment Problem, Proceedings of the 14th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA-2011), pp.174–186, November 2011, Wollongong, Australia.

国内会議(査読あり)

1 ◯花田研太，平山勝敏，沖本天太: 分散ラグランジュ緩和プロトコルにおけるバンドル法の効果，合同エージェントワークショップ & シンポジウム 2014 (JAWS-2014) 講演論文集，pp.245–248, 2014 年 10 月，宮崎.

2 ◯花田研太，平山勝敏: 多層一般化相互割当問題の定式化とその解法，合同エージェントワークショップ & シンポジウム 2012 (JAWS-2012) 講演論文集，2012 年 10 月，掛川.

3 ◯花田研太，平山勝敏 : 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，合同エージェントワークショップ & シンポジウム 2010 (JAWS-2010) 講演論文集，2010 年 10 月，富良野.

国内会議 (査読なし)

1 花田研太，平山勝敏，沖本天太: 増床計画付き患者搬送問題の定式化とヒューリスティック解法の提案, 合同エージェントワークショップ & シンポジウム 2015 (JAWS-2015) 講演論文集, pp.16–17, 2015 年 9 月 30 日, 加賀, ポスター発表.

2 ◯花田研太，平山勝敏，沖本天太: Max-SAT に対する非厳密解法を用いたラグランジュ分解・調整法, 2015 年度人工知能学会全国大会（第 29 回） (JSAI-2015) 講演論文集，2015 年 5 月 31 日，函館.

3 ◯花田研太，平山勝敏: Multi-MaxSAT におけるバンドル法の効果, 2013 年度人工知能学会全国大会（第 27 回） (JSAI-2013) 講演論文集，2013 年 6 月，富山.

口頭発表

1 ◯Kenta Hanada, Katsutoshi Hirayama, Tenda Okimoto: Patient Transportation Problem with Bed Capacity Management: Formalization and Solution, Workshop for Resilience and Scheduling, November 2015, Nagasaki, Japan.

2 ◯Kenta Hanada, Katsutoshi Hirayama, Tenda Okimoto: Lagrangian Decomposition and Adjustment Method with Incomplete Solver for Max-SAT Problem, Workshop for Computational Techniques for Sustainability and Resilience, March 2015, Okinawa, Japan.

3 ◯花田研太: 分散ラグランジュ緩和プロトコルとその応用，FUN-AI-15，2015 年 3 月.

4 ◯花田研太，平山勝敏，沖本天太 : 分散ラグランジュ緩和プロトコルにおけるバンドル法の効果，平成 26 年度第 1 回 DCR 研究会，2014 年 9 月 .

5 ◯Kenta Hanada, Katsutoshi Hirayama: Distributed Lagrangian Relaxation Protocol for Generalized Mutual Assignment Problem, JFLI-LIP6-NII Meeting on Efficien Methods for Reasoning and Problem Solving 2013, October, 2013.

6 ◯花田研太，平山勝敏: Multi-MaxSAT を拡張した Weighted Partial Max-SAT Solver，第 3 回 CSPSAT2 研究会，2013 年 7 月 .

7 ◯花田研太，平山勝敏: 多層一般化相互割当問題の定式化とその解法，平成 24 年度第 2 回 DCR 研究会，2012 年 9 月 .