特別研究報告書

未知の選好を含む最大安定度マッチング問題

指導教員　　松原 繁夫 准教授

京都大学工学部情報学科

境　良太

平成 22 年 2 月 3 日

未知の選好を含む最大安定度マッチング問題

境　良太

内容梗概

タスクを人に割り当てる問題において，誰にどのタスクを割り当てるかというのは大きな問題となる．なぜならば，タスクには必要な能力があり，より能力のある人にタスクを割り当てたい一方で，タスクの請負人は，それぞれ得意なタスクや好むタスクを持っており，双方の希望を満たす必要があるからである．これを，一人に一つのタスクを割り当てる問題と考えると安定結婚問題として広く知られた問題に帰着して解くことができる．

大規模なタスク割り当て問題では，新しい請負人が次々と参加することも少なくない．しかし，これは安定結婚問題に帰着するうえで問題を引き起こす．安定結婚問題を解くためには，すべてのタスクと請負人の選好が必要となるためである．タスクの側はタスクに必要な能力により選好を決めることができる．また，古くからタスクを行っている請負人に関してもそれぞれのタスクの選好はよくわかっているため問題はない．しかし，新しく参入した請負人はそれぞれのタスクのことをよく知らないため，タスクの選好順を述べることができない．さらに問題となるのは，選好の分からない新しい請負人も実際には選好を持っているため，マッチングが終わってタスクを行っている間に，それぞれのタスクの選好に気づくことがあることである．結果的にタスクにミスマッチが起こっていることが分かれば，タスク依頼者，請負人双方にとって，マッチング結果に不満が生ずる．そのため，選好が未知の請負人の参加するマッチングの問題において，安定となるようなマッチングを求めることが必要である．

このような問題を解決するために，選好が未知の参加者がいる場合の安定結婚問題について研究を行った．安定結婚問題では一般に安定マッチングを求めることを目指すが，この問題においては，安定マッチングが存在しない問題例も起こりうるため，不安定となったとしてもできるだけ安定に近いマッチングを考え，最大安定度マッチング (almost stable matching) を目指すこととする．本研究の目的は，未知の選好が含まれる場合における安定結婚問題を定式化し，最大安定度マッチングを求めるアルゴリズムを作成することである．なお，本研究では上で述べたタスク割り当て問題の例にならい，タスク側には選好が未

知のタスクは含まないとする．

この問題を解くにあたり，以下のような課題が存在する．

- これまでの安定結婚問題の研究においては，全員の選好が分かることが仮定されていた．そのため，未知の選好に対応するためには，未知の選好の定式化が必要となってくる．
- 同順位を許す安定結婚問題などの，安定結婚問題の拡張は様々なされてきたため，これを利用して新しく参入した人の選好を無視することで，それなりによいマッチングを求めることはできるかもしれない．しかし，参入者も実際には選好を持っていることから，不明な選好も含めて結果にミスマッチが起こらないようにしなければならない．

これらの課題の解決のために，未知の選好をほぼ同順位とみなして，同順位を許す安定結婚問題と同様の形で未知の選好を表現することとした．また，未知の選好を，実際に起こりうるすべての選好の集合と捉え，起こりうるすべての選好に場合分けをして安定結婚問題を解くことで，一定の条件のもとでは，最大安定度マッチングを探すことができることが判明した．安定結婚問題の解法を改良して，場合分けを必要になるまで遅延することで，場合分けの数を減らすことができる．これにより，指数時間で最大安定度マッチングを求めるアルゴリズムを作成した．

本研究により，選好の一部が未知の安定結婚問題の定式化をすることができた．未知の選好の定式化により，他のマッチング関連の問題に対しても，未知の選好を含む場合に関して，マッチングの評価ができるようになった．

また，未知の選好が含まれる場合に，一定の条件のもとで，不安定度を最小化するマッチングを求めるアルゴリズムを提案した．結果を弱安定マッチングに限定しなければ，近似アルゴリズムとなる．そのため，参入者が問題となるタスク割り当ての問題などにおいても，不安定度の少ない，比較的性質の良いマッチングが求められるようになった．計算量は指数時間となったが，全探索に必要な階乗時間と比べると高速となっている．

Almost Stable Matching Problem Including Unknown Preferences

Ryota SAKAI

Abstract

In task assignment problems, it is a crucial problem that to decide to which contractor a particular task should be assigned. Task clients desire that a contractor who has higher ability performs the task. On the other hand, contractors have preferences of tasks and abilities for tasks. Therefore it's necessary to satisfy both requests. If you regard the task assignment problem as the problem of assigning one task to one person, it becomes a widely known problem, the stable marriage problem.

New contractors often participate in the matching one after another in a large-scale task assignment problem. That causes a problem when it results in the stable marriage problem, because preferences of all tasks and contractors are necessary for solving the stable marriage problem. Task clients have their preferences from the ability to perform the task. Contractors who did tasks before also have their own preferences because they understand tasks well. Therefore there is no problem for them. However new contractors can't state their preferences order of tasks, because they don't know the respective tasks well. Furthermore, because the contractors whose preferences are unknown have actual preferences, they may realize their own preferences. After the matching, if clients and contractors notice there are mis-matchings, they will not be satisfied with the result. Therefore it's necessary to acquire a stable matching in the matching problem with contractors whose preferences are unknown.

In order to solve the above problem, I studied about the stable marriage problem with participants whose preferences are unknown. In general, the purpose of the stable marriage problem is to find a stable matching. However there are the cases that there is no stable matching in this problem. Therefore this paper does not focus on stable matchings but almost stable matchings in which the result is as stable as possible even if it becomes unstable. The purpose of this research is to formulate the stable marriage problem including unknown preferences and to make an algorithm by which we can find an almost stable

matching. Like the example of the task allocation problem explained above, I assume that unknown preferences aren't included in the task side in my study.

This problem contains these two following problems.

- All the members' preferences are assumed in the stable marriage problem. Therefore a formulation corresponding to unknown preferences is needed.
- The stable marriage problem is expanded variously to problem such as the stable marriage problem with tie. It may be possible to ignore preferences of new contractors by using this. Then good matchings will be acquired to some degree. However, actually participants do have their own preferences, so there should be no mis-matching.

In order to solve these problems, I regarded unknown preferences as tie preferences, then expressed the problem as the stable marriage problem with tie. Unknown preferences are considered to be the set of all possible preferences. It reveals that it's possible to find out almost stable matching under the fixed condition, solving the stable marriage problem in all case of each possible preferences. I improved the solution to the stable marriage problem in which case analysis is delayed until it is needed. Consequently it's possible to reduce the number of cases. This algorithm I made can find almost stable matching with exponential time.

The stable marriage problem with partially unknown preferences became to be formulated in this research. Because of this formulation, the instability of matching can be evaluated in other matching problems with unknown preferences.

I made an algorithm that can find almost stable matching in a case that unknown preferences are included under the fixed condition. When the result is not limited to the fixed condition, it becomes an approximation algorithm. Therefore in the task allocation problem with unknown preferences, the better matching which is less unstable can be acquired. Complexity is exponential time, but it is faster compared with full search which needs factorial time.

未知の選好を含む最大安定度マッチング問題

目次

第1章　はじめに

タスクを人に割り当てるという状況は様々な場面に存在する．例えば企業では，従業員に仕事を割り当てるという状況が日常的に発生している．近年では，インターネットなどを使って不特定多数の人に仕事を委託するクラウドソーシングという形態も広まっているが，これもタスク割り当ての例となっている．クラウドソーシングとは，実行しなければいけないタスクを持っている人が，タスクをクラウドソーシングに依頼し，タスクの請負人は，その中からタスクを選んで実行するというものである．クラウドソーシングは，数多くのタスクや請負人がいることが特徴となっている．

このようにタスクを請負人に割り当てようとする時，誰にどのタスクを割り当てるかというのは大きな問題となってくる．なぜなら，それぞれのタスクには必要な能力があり，より能力のある人にタスクを割り当てたい一方で，それぞれの請負人はそれぞれ得意なタスクや好むタスクを持っており，タスクの依頼者と請負人双方の希望を満たす必要があるためである．これを一人に一つのタスク割り当てる問題だと考えると，タスクと請負人の一対一マッチングの問題となり，安定結婚問題 [1] として広く知られた問題に帰着して解くことができる．しかし，安定結婚問題として問題を解くためには，すべてのタスクと請負人の選好が分かっている必要がある．選好が分からないタスクや請負人がいる時には，安定結婚問題に単純に帰着させることはできない．例えば，クラウドソーシングにおけるタスク割り当てのような問題では，新しい請負人が次々と参加することも少なくないが，新しく参入した請負人は，それぞれのタスクのことをよく知らないため，タスクの選好順を述べることができない．そのため，その請負人の選好が分からず，マッチングの問題を解くことができない．

この時さらに問題となるのは，選好の分からない参入者も実際には選好を持っており，タスクを行っている間にそれぞれのタスクの選好に気付くかもしれないということである．もし，未知であった選好が全く判明する可能性がないのであれば，未知の選好を無視したマッチングを行ってミスマッチが起こったとしても，ミスマッチの起こった本人たちはそのことに気が付かないため問題はない．しかし，未知であった選好が後から判明すると，タスク依頼者と請負人はミスマッチが起こったことに気付き，マッチングの結果に不満が生ずる．そのため，選好が未知である参入者を無視するわけにはいかない．

未知の選好を含むタスクと請負人のマッチングを行うためには，未知の選好を含む安定結婚問題を解くことが必要となってくる．そのため，本研究では選好が未知の請負人を含む安定結婚問題について取り上げた．ところで，請負人は選好が未知の可能性があるが，新しいタスクに関しては，タスクに必要な能力がはっきりしているため，選好が分からないという問題は生じづらい．このように，マッチングの対象はしばしば非対称であり，片方の集団のみに選好が未知の人がいる場合に関する議論も有用である．また，問題の簡単のためも考えて，本研究では選好が未知であるのは請負人側のみである場合を中心に議論を行った．

それでは，今度は安定結婚問題のこれまでの研究を概観し，未知の選好に対しどのようなアプローチをとることができるかについて述べる．安定結婚問題はGale，Shapley[1] によって提唱された問題で，男性と女性，タスクと請負人などの二つの異なる集団を一対一マッチングさせる問題である．男性，女性などのそれぞれの集団は相手の選好順序を持っており，自分がより好む相手とペアになりたいと考えている．マッチングが与えられた時，もし現在の自分のパートナーよりもお互いをより好むペアが存在した場合，この二人は与えられたマッチングを無視して，新しいペアを作ってしまう．このようなペアが存在しないマッチングを，安定なマッチングという．このように，二つの集団の選好順序が与えられた時に安定なマッチングを求める問題が安定結婚問題である．安定結婚問題は様々な拡張を含め広く研究されている．

この安定結婚問題では，全員が相手全員の順序付けをしていることが仮定されているが，実際には全員に対して完全な順序付けを行うことは不可能な場合が多い．この問題を解決するにあたっても，様々な研究がなされてきた．例えば，安定結婚問題の緩和として，選好順序に同順位を許したり，選好リストに含まれない相手を許したりする拡張が知られている [2][3]．これを用いて，未知の選好を同順位とみなすことで，タスクと請負人のマッチングを求めることができるかもしれない．しかし，単純に同順位とみなすというのは未知の選好を無視するということであり，未知であった選好が判明した際に問題が起こってしまうため，工夫が必要である．

選好の知識のあり方に関しては，違った考え方もできる．ここまでの問題では主に，選好を一ヶ所に集めて，すべての選好を用いて中央集権的に解く問題として考えられてきたが，それぞれのタスクや請負人をエージェントとみなして，

エージェント同士の通信を通じて安定マッチングを求めるというアプローチも考えられいる．それぞれのエージェントは自分の選好しか知らず，他人の選好の知識は持っていない．それぞれのエージェントにとって他人の選好は未知である．この場合でも，安定マッチングを得ることは可能であるが [4]，例えば通信の回数に制限がある場合などには，必ずしも安定マッチングを得ることができなくなってしまう [5]．しかし，不安定マッチングとなってしまったとしても，その中でもできるだけ安定なマッチングである最大安定度マッチング (almost stable matching)[6] を求める必要がある．タスクと請負人のマッチングの問題の場合，自分の選好すら分からない人がいることが問題である一方，中央集権的に解くことができるため問題が簡単となり，このアプローチを用いることは適さない．しかし，選好が未知であると必ずしも安定マッチングが求められないため，その場合は最大安定度マッチングを求めなければならないことは考慮しなければならない．

タスクと請負人のマッチングなどを解くために，未知の選好含む安定結婚問題において，より安定なマッチングを行うことが望まれていることをここまで述べた．そのため，本研究の目的は，片側の集団に選好に未知であるような人を含む安定結婚問題において，最大安定度マッチングを求めるアルゴリズムを求めることとする．

この問題の解決のために，未知の選好をほぼ同順位とみなして，同順位を許す安定結婚問題と同様の形で未知の選好を表現することとし，安定結婚問題の解法を改良することで，最大安定度マッチングを求めるアルゴリズムを作成した．

第 2 章では関連研究として，安定結婚問題とその応用について述べる．第 3 章では問題を定式化し，その性質について論ずる．第 4 章で最大安定度マッチングを求めるアルゴリズムを提案する．第 5 章でアルゴリズムの評価を行い，最後に第 6 章でまとめを行う．

第 2 章　関連研究

本章では安定結婚問題とその拡張の同順位を許す安定結婚問題について，問題の定義と性質，解法について述べ，最後に，最大安定度マッチングについて述べる．

2.1 安定結婚問題

安定結婚問題とは，男性と女性，タスクと請負人のような二つの異なる集団を1対1マッチングさせる問題である．ここでは，男女のマッチングを例に，安定結婚問題の定義と，その基本的な性質について述べる．

男女それぞれ n 人いるとし，男性の集合を M，女性の集合を W とする．それぞれの男女は，異性に対して選好順序を持っている．$m \in M$ が $w_1 \in W$ を $w_2 \in W$ より好むことを $w_1 <_m w_2$ と表わすこととする．選好順序は異性全員に対するもので，同程度に好むということは許されていない．つまり，ある，$m \in M$，$w_1, w_2 \in W$ に対して，$w_1 <_m w_2$ または $w_2 <_m w_1$ でなければならない．男性と女性を一対一で結びつけたものをマッチングといい，μ などの文字で表す．あるマッチング μ において男性 $m \in M$ とペアになっている女性を $\mu(m)$（$\mu(m) \in W$），女性 $w \in W$ とペアになっている男性を $\mu(w)$（$\mu(w) \in M$）と表わすこととする．あるマッチングが成立した時，そのマッチングに含まれないペアが，もし現在のマッチングのパートナーよりお互いにお互いを好むならば，その二人でペアを作ることが，その二人にとってより望ましい状態となる．そのため二人で結託して新しいペアができてしまい，その結果マッチングが不安定となってしまう．このようなペアをブロッキングペアという．これを式で表すと，ブロッキングペアとは，μ に含まれないようなペア (m, w) で，$w <_m \mu(m)$ かつ $m <_w \mu(w)$ が成り立つものである．ブロッキングペアが存在しないマッチングを，安定マッチングという．Gale，Shapley[1] によって，それぞれの男女がどのような選好順序を持っていたとしても必ず安定マッチングが存在することが証明されている．

また，安定マッチングの存在の証明の過程から，安定マッチングを求めるアルゴリズムも知られている．そのアルゴリズムは以下のとおりである．

- 各男性はまず最も選好している女性にプロポーズをする．プロポーズを断られたら，その次に選好する女性にプロポーズをする．プロポーズが成立するまでこれを繰り返す．
- 各女性は，最初に受けたプロポーズは必ず受理する．二人目以降の男性からプロポーズを受けた際には，これまでにプロポーズをした男性も含め，最も選好する男性のプロポーズのみ受け入れ，残りの男性のプロポーズは破棄する．

男性の選好	女性の選好
$m_1 : w_1 < w_2 < w_2$	$w_1 : m_1 < m_2 < m_3$
$m_2 : w_1 < w_2 < w_3$	$w_2 : m_3 < m_1 < m_2$
$m_3 : w_3 < w_1 < w_2$	$w_3 : m_1 < m_2 < w_3$

図 1: 安定結婚問題の問題例

このアルゴリズムは必ず全員がペアとなり，得られたマッチングは安定マッチングとなることが証明されている．このアルゴリズムは，Gale，Shapley の名前から Gale-Shapley のアルゴリズムなどと呼ばれる．

なお，どの男性から順にプロポーズを行ったとしても，同じマッチングが得られる．これは，女性からプロポーズを行っても同様の議論ができるが，男性がプロポーズをした場合と女性がプロポーズした場合では，結果として得られるマッチングは異なる場合がある．男性がプロポーズした場合，すべての安定マッチングの中で男性にとって最も有利なマッチングが得らる．これを男性優位安定マッチングという．なお，どの男性にとっても，男性優位安定マッチングにおけるパートナーよりも選好する女性とペアになれる安定マッチングは存在しないことが知られている．

安定結婚問題の問題例を図 1 に示す．この問題の安定マッチングは $((m_1, w_1), (m_2, w_2), (m_3, w_3))$，$((m_1, w_1), (m_2, w_3), (m_3, w_2))$ の二つで，Gale-Shapley のアルゴリズムを適用すると，このうち $((m_1, w_1), (m_2, w_2), (m_3, w_3))$ のマッチングが得られる．これが男性優位安定マッチングである．$((m_1, w_1), (m_2, w_2), (m_3, w_3))$ は，男性全員が $((m_1, w_1), (m_2, w_3), (m_3, w_2))$ と比べて同等以上の選好の女性とマッチングされていることが分かる．

2.2 同順位を許す安定結婚問題

安定結婚問題は，様々な拡張がなされて研究されているが，そのひとつに選好に同順位を許す場合がある [2]．もとの安定結婚問題では，異性全員に対して厳密に選好順を持つ必要があったのに対し，同順位を許す安定結婚問題では，同程度好むことが許される．ある $m \in M$ が，$w_1 \in W$ と $w_2 \in W$ を同程度好むことを $w_1 =_m w_2$ と等号で表し，$m \in M$ が，$w_1 \in W$ を $w_2 \in W$ と同程度以上好むことを，$w_1 \leq_m w_2$ と表わすこととする．

選好に同順位を許すことに伴い，ブロッキングペアの定義も見直す必要がある．ブロッキングペアは3通りの定義がなされ，それぞれの定義に対して安定性が定義されている．

一つめのブロッキングペアの定義は，もとの問題と同じように，お互いをお互いに厳密に好む場合のみをブロッキングペアとし，自分と同順位の相手とするものである．つまり，マッチング μ に含まれないペア (m, w) で，

$$w <_m \mu(m) \text{ かつ } m <_w \mu(w) \tag{1}$$

を満たすものをブロッキングペアとする．このようなブロッキングペアが存在しない安定状態を弱安定という．同順位の選好を任意に順位付けして，その順位付けのもとでGale-Shapleyアリゴリズムを行うことで，弱安定マッチングを求めることができる．そのため，弱安定マッチングは必ず存在する．

二つめのブロッキングペア定義は，ある男性は女性を厳密に好むが，女性はその男性と今のパートナーと同じくらい好むペアをブロッキングペアとする場合である．つまり，ブロッキングペアの定義を

$$\begin{aligned} &w <_m \mu(m) \text{ かつ } m \leq_w \mu(w) \text{ または} \\ &w \leq_m \mu(m) \text{ かつ } m <_w \mu(w) \end{aligned} \tag{2}$$

とする．このときの安定性を強安定という．

三つ目の定義は，お互いに今のパートナーと同順位のペアもブロッキングペアと定義する場合で，この時の安定性を超安定という．強安定，超安定マッチングは必ずしも存在するとは限らないことが知られている．また，定義より，超安定であれば強安定であり，強安定であれば弱安定であることが分かる．

同順位を許す安定結婚問題の例を，図2に示す．同順位の選好順を図に書かれている順に順位付けすると，図1と同じ問題になり，これを解くと，$((m_1, w_1), (m_2, w_2), (m_3, w_3))$，$((m_1, w_1), (m_2, w_3), (m_3, w_2))$ のマッチングが得られる．これらは，弱安定マッチングとなっている．なお，同順位の選好順を別の順に順位付けしても，弱安定マッチングとなるため，これ以外の弱安定マッチングも存在する．この問題例では，強安定マッチング，超安定マッチングは存在しない．

男性の選好	女性の選好
$m_1 : w_1 = w_2 < w_2$	$w_1 : m_1 < m_2 < m_3$
$m_2 : w_1 < w_2 < w_3$	$w_2 : m_3 < m_1 < m_2$
$m_3 : w_3 < w_1 = w_2$	$w_3 : m_1 = m_2 = w_3$

図 2: 同順位を許す安定結婚問題の問題例

2.3 マッチングの不安定度

中央集権的でない分散的な安定結婚問題などにおいては，必ずしも安定なマッチングが得らるとは限らないことから，不安定なマッチングに対する不安定度についても研究がされている [7]．不安定度の尺度の候補としては，ブロッキングペアの数や，ブロッキングペアに含まれる男女の人数などが考えられるが，ブロッキングペアの数 (を正規化したもの) を尺度にするのがよい．ブロッキングペアに含まれる男女の人数を不安定度の尺度とした場合，ブロッキングペアに含まれる男女の人数が同じならば同じ不安定度とされるが，人数が同じでもブロッキングペアが多ければ多いほど，互いにブロッキングペアになっていると発覚する可能性が高くなり不安定になるため，ブロッキングペアに含まれる男女の人数よりも，ブロッキングペアの数を尺度に用いるのがよいことが分かる．

そのため，マッチング μ の不安定度 (Instability) として，マッチング μ のブロッキングペアの個数を $Bp(\mu)$ としたとき，ブロッキングペアの数を正規化した次のような式を用いるのがよい．

$$Bp(\mu)/n^2 \tag{3}$$

不安定度は，安定マッチングならば 0 で，不安定になればなるほど大きな値となる．

この方法は選好構造が異なる場合のマッチング間で比較する場合には問題があることも指摘されている．そのため，マッチングの relative instability という尺度が提案されている．本研究では選好構造が異なる場合のマッチングを扱う場面があり，そこでは relative instability を用いた方がよい．しかし，relative instability の計算は計算量が多く扱いづらいため，本研究では不安定度のみを用いる．

第3章　未知の選好を含むマッチング

クラウドソーシングにおけるタスクと請負人のマッチングの例をもとに，一部の請負人の選好が未知である場合に最大安定度マッチングを求める問題を定式化する．これは，選好が未知の請負人を含めることで安定結婚問題を拡張している．

3.1　問題の定式化

n 個のタスク T と，n 人の請負人 C が存在し1対1マッチングを行う問題を考える．各タスクは，すべての請負人に対する完全な選好順序をもっているが，請負人の中には，選好の一部が未知である人も存在する．ある請負人 $c \in C$ にとって $t_1 \in T$ と $t_2 \in T$ の選好関係が未知であるとは，$t_1 <_c t_2$ であるのか $t_2 <_c t_1$ であるのか分からないことであると定義する．そして，これを $t_1 \sim_c t_2$ と表わす．$t_1 \sim_c t_2$ であるとき c の選好は未知であるが，実際には厳密な選好を持っており，$t_1 <_c t_2$ または $t_2 <_c t_1$ であることに注意しなければならない．なお，問題を解くにあたっては，簡単のため，未知の選好を含む請負人 c がすべてのタスクの選好関係について未知である場合に限定して議論を行うが，限定しない場合でも問題の性質はほとんど変わらないため，しばらくは選好の一部が未知である場合に関して議論を行う．

まず，実際の問題の適用場面を考えると，請負人 c にとって選好が未知なタスク t_1，t_2 は，実際には選好がある程度近いことが予想される．なぜならば，もしあまりにも選好が違う場合には容易に選好関係が見出せると考えられるからである．そのため，$t_1 \sim_c t_2$ かつ $t_2 \sim_c t_3$ ならば $t_1 \sim_c t_3$ という推移律が成り立つと仮定できる．この仮定をおくと，未知の選好を含む問題の $\sim$ と同順位を許す安定結婚問題の$=$を入れかえることによって，同順位の選好と同じ形式で書くことができる．すると，各タスクと請負人の選好順序は，図3の例のように，$<$ と $\sim$ を用いて示すことができる．同順位の選好と許す安定結婚問題との類推ができるため，同順位の選好と許す安定結婚問題の概念の一部を利用する．

この問題の特徴は，実際の選好が存在するにも関わらず，マッチングに選好の情報を用いることができないことにある．その一方で，マッチングが成立した後に選好が判明する可能性があり，もしミスマッチが存在していたとすれば，タスクと請負人の双方が不満を持つこととなってしまう．これらをふまえ，ブ

男性の選好	女性の選好
$m_1 : w_1 \sim w_2 < w_2$	$w_1 : m_1 < m_2 < m_3$
$m_2 : w_1 < w_2 < w_3$	$w_2 : m_3 < m_1 < m_2$
$m_3 : w_3 < w_1 \sim w_2$	$w_3 : m_1 \sim m_2 \sim w_3$

図 3: 未知の選好を含む安定結婚問題の問題例

ロッキングペアの定義と安定性に関して論ずる．

まず，お互いがお互いをマッチングのパートナーより選好することが既知であるペアが存在する場合，未知の選好の議論とは無関係に，もとの安定結婚問題と同様，そのペアはマッチングをブロックする．これをブロッキングペアと考えると，ブロッキングペアは，あるマッチングに含まれないペア (t, c) に対して，

$$\{(t, c) | t \in T, c \in C, t <_c \mu(c) \text{ かつ } c <_t \mu(t)\} \tag{4}$$

が成り立つものと定義することができる．このようなブロッキングペアが存在しないマッチングを弱安定マッチングと定義する．なお，同順位を許す安定結婚問題と同様に，既知の選好順を壊さないように未知の選好を任意に順位付けすることによって，弱安定マッチングを求めることができる．従って，弱安定マッチングは必ず存在する．

弱安定マッチングには，タスク t と請負人 c のペアで，タスク t はマッチングのパートナー $\mu(t)$ よりある請負人 c を好み，請負人 c はタスク t とパートナー $\mu(c)$ のどちらを好むか分からない $(t \sim_c \mu(c))$ ペアが含まれていてもかまわない．しかし，実際には，請負人 c の選好は $t <_c \mu(c)$ となる可能性も，$\mu(c) <_c t$ となる可能性もあるが，$\mu(c) <_c t$ であった場合ペア (t, c) は特に問題を起こさないのに対し，選好が $t <_c \mu(c)$ である場合にはペア (t, c) はお互いにお互いを好むことになってしまう．$t <_c \mu(c)$ であった場合，もし選好が判明すれば，タスク t の依頼者と請負人 c は不満を持ってしまう．そのため，このような状況が起こらないようにすることを考えなければならない．

$t_1 \sim_c t_2$ または $t_1 < t_2$ であることを，$t_1 \lesssim_c t_2$ と表わすとき，ブロッキングペアの定義を，

$$\{(t, c) | t \in T, c \in C, t \lesssim_c \mu(c) \text{ かつ } c <_t \mu(t)\} \tag{5}$$

のようにすることで，選好が判明した後でもお互いにお互いを好むペアが現れ

男性の選好	女性の選好
$m_1 : w_1 < w_2$	$w_1 :$ unknown$(m_1 \sim m_2)$
$m_2 : w_1 < w_2$	$w_2 : m_1 < m_2$

図 4: 強安定マッチングの存在しない問題例

るのを防ぐことができる．このブロッキングペアがないマッチングを強安定マッチングと定義する．しかし，弱安定マッチングと違い，強安定マッチングは必ずしも存在するとは限らない．図 4 の問題例では，男性では m_1，m_2 ともに w_1 を好んでいるが，m_1 と w_1 がペアになれば m_2 と w_1 がブロッキングペアとなり，m_2 と w_1 がペアになれば m_1 と w_1 がブロッキングペアとなるため，どのようにマッチングをしても強安定とはならないということが分かる．

選好が既知になっても安定とするためには弱安定では安定性としては弱く，強安定マッチングをとることが必要となってくる．しかし，強安定マッチングは必ずしも存在するとは限らない．そのため，強安定においてのブロッキングペアの数を不安定度の尺度に用い，強安定におけるブロッキングペアの数を最小化する問題を考える．この定義の妥当性に関しては，次の節で詳しく論ずる．

しかし，強安定マッチングにおけるブロッキングペアを最小化するのは難しい問題である．そこで，解を弱安定マッチングに限定することを考える．例えば，全員が他人の選好を知っている応用例がある場合，弱安定すら満たされていないマッチングでは直ちにマッチングがブロッキングペアができてしまい，マッチングに不満が起こる．この場合は，解が弱安定の条件を満たしている必要がある．そうでなかったとしても，強安定マッチングであれば弱安定を満たすため，弱安定マッチングに限定して強安定におけるブロッキングペアが最小のマッチングを選んでも，よい近似になっていると考えられる．また，弱安定マッチングは必ず存在する．これらをふまえて，本研究では弱安定が満たされるマッチングに限定して，不安定度を最小化する最大安定度マッチングを求める問題を考える．

なお，本研究では選好が未知であるのは請負人のみで，タスクの選好はすべて既知であるが，タスク側にも未知の選好を含めると，同順位を許す場合との類推から，強安定マッチング，超安定マッチングを考えることができる．この時，お互いに選好が不明なペアができる可能性があるが，お互いがお互いをよ

り好む可能性は低くなるため，あまり考慮する必要はない．そのため，タスク側にも未知の選好がある場合でも，強安定でのブロッキングペアの数を不安定度として最大安定度マッチングを求めるという方針を用いることができる．また，本研究では同順位を許す安定結婚問題に帰着させるため，$\sim$ の性質として推移律を挙げているが，これを取り除いて半順序の選好を用いることも有用である場合がある．その場合も，同様の考え方をすることができる．

3.2 マッチングの不安定度

マッチングの不安定度として，マッチングのブロッキングペアの数を用いればよいことが関連研究で示されているため，前節では，強安定におけるブロッキングペアの数を不安定度とした．しかし，複数の種類のブロッキングペアを定義したため，どのブロッキングペアを用いるかは議論が必要である．

弱安定は安定性として弱すぎるため，強安定におけるブロッキングペアの数を用いたほうがよいことは前節にて説明した．ここまでの議論では，未知の選好を単に未知のものとして扱ってきたが，実際には選好が未知であっても何らかの選好を持っていることも考えなければならない．選好の分からない人の選好を完全にランダムとみなし，すべての選好順が同じ確率で起こると考えると，すべての起こりうる選好それぞれに対してマッチングのブロッキングペアの数を数えることができる．そこで，すべての選好順に対するブロッキングペアの数の期待値をマッチングの不安定度と考えることもできる．

マッチングを弱安定マッチングに限定すると，強安定におけるブロッキングペアの数と，ブロッキングペアの期待値は比例関係にあることが証明できる．そのため，弱安定マッチングにおいて，強安定におけるブロッキングペアの数を最小化することは，ブロッキングペアの期待値を最小化することに等しい．ここでは，弱安定マッチングが満たされている場合，強安定のブロッキングペアの数と，ブロッキングペアの期待値が比例関係にあることを示す．

弱安定マッチングであれば，ブロッキングペアのうち，既知の選好においてお互いにお互いを好むということが起こらない．つまり，前節の式 (5) を満たすペアのうち，式 (4) を満たすものは存在しないため，ブロッキングペアは常に

$$\{(t,c)|t \in T, c \in C, t \sim_c \mu(c) \text{ かつ } c <_t \mu(t)\} \tag{6}$$

となる．

$$t_1 < t_2 < t_3,\ t_1 < t_3 < t_2,\ t_2 < t_1 < t_3$$
$$t_2 < t_3 < t_1,\ t_3 < t_2 < t_1,\ t_3 < t_1 < t_2$$

図 5: 起こりうるすべての選好順

この時未知の選好をランダムとみなすと，請負人 c は現在のパートナーとタスク t のそれぞれを好む確率は $1/2$ ずつとなる．そのため，このペアは $1/2$ の確率でブロッキングペアとなる．また，選好順をランダムとみなしているため，あるタスクとそのタスクが選好する請負人のペアと，他のタスクとその請負人とのペアで，ブロッキングペアになるかどうかは独立となる．また，ある請負人の選好と別の請負人の選考も独立とみなせる．式 (6) のようなペアはすべてが独立に $1/2$ の確率でブロッキングペアをなすため，そのようなペアの数の $1/2$ がブロッキングペアの数の期待値となる．従って，弱安定マッチングの中では，強安定のブロッキングペアの数の $1/2$ がブロッキングペアの期待値となる．

例えば，選好が未知の請負人 c にとって，タスク t_1，t_2，t_3 の選好順は図 5 に示す 6 通りである．この時，t_1 と t_2 のみが請負人 c を好むとすると，(t_1, c)，(t_2, c) の二つが強安定におけるブロッキングペアとなるが，6 通りの選好順の中で，$t_1 <_c t_2$ となる選好順や $t_1 <_c t_3$ となる選好順はどちらも 3 通りあり，どちらも全 6 通りの $1/2$ の数であることが分かる．また，$t_1 <_c t_2$ と $t_1 <_c t_3$ の数の合計を計算しても，2 つとなるのが 2 通り，1 つとなるのが 2 通り，ひとつもないのも 2 通りで，平均すると 1 となり，強安定におけるブロッキングペアの数の $1/2$ の 1 がブロッキングペアの期待値となっていることが確認できる．

そのため，弱安定マッチングの中において，強安定の意味でのブロッキングペアの数を最小化することは，ブロッキングペアの期待値を下げることと同値である．強安定に関して最大安定度マッチングをとることは，結果として得られるマッチングの不安定度の期待値を最小化するという観点からも適していると考えられる．

3.3 不安定度の最小化

ここまでで，ブロッキングペアと安定の定義について論じてきたが，この節では，弱安定マッチングの中で強安定の意味でのブロッキングペアが最も小さくなる最大安定度マッチングの性質について論じる．

まず，選好が未知である場合の弱安定マッチングの性質について述べる．選

好が未知である場合の弱安定マッチングでは，次の命題が成り立つ[1)]．

命題　選好が未知である場合の弱安定マッチングの集合と，未知の選好を任意に順位付けした場合に起こりうる選好順のいずれか対して安定となるマッチングは一致する．

証明　同順位を許す安定結婚問題において，同順位の選好を強制的に順位付けすることで弱安定マッチングが得られることは，関連研究の章で述べている．同様の議論で，未知の選好の場合でも，未知の選好を強制的に順位付けすることで，弱安定マッチングが得られる．そのため，未知の選好の任意の順位付けにおいておこる安定マッチングは，すべて弱安定マッチングである．

一方，未知の選好を含む安定結婚問題に対してある弱安定マッチングが与えられた時，そのマッチングのパートナーとの選好関係が未知である相手の中で，マッチングのパートナーを最も選好するように順位付けをすることで，その選好順における安定結婚問題において，そのマッチングは安定となる．なぜならば，あるペアがブロッキングペアであるかどうかは，お互いにお互いの選好が未知である場合には弱安定と順位付けした際では変わらず，また，選好が未知である場合には，与えられたマッチングの中でそのペアのパートナーが最も選好するように順位付けしているため，それ以外の人とはブロッキングペアとならないためである．そのため，弱安定マッチングならば，ある順位付けにおいて安定となる．

命題より，弱安定マッチングの中で不安定度の最も小さなマッチングを求めるためには，起こりうるすべての順位付けにおいて安定マッチングを求め，その中から最大安定度となるマッチングを求めても同じ結果が得られることが分かる．また，起こりうる順位付けの中からひとつの順位付け選び，選好順を固定して考えると，弱安定マッチングの中で最大安定度マッチングとなる可能性があるのは，固定された選好順による安定マッチング中での最大安定度マッチングのみである．そのため，すべての順位付けに対して，それぞれにおける不安定度の最も小さなマッチングを集めれば，その中に最大安定度マッチングが存在する．

これを，図 6 の例を用いて説明する．ある未知の選好について，起こりうる

[1)] 同順位を許す安定結婚問題においても同様の命題が成り立つ．

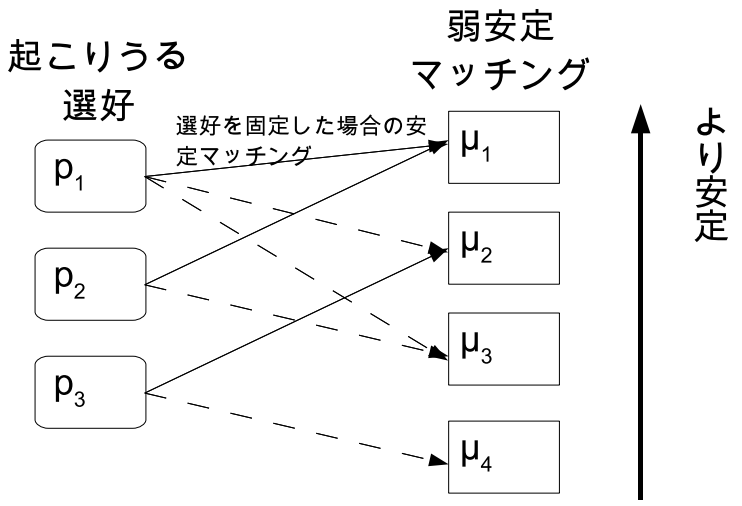

図 6: 起こりうる選好と弱安定マッチングの関係

選好順が p_1，p_2，p_3 の 3 通りあったとする．図の矢印は，選好を固定した場合に安定となるマッチングを指している．例えば，p_1 に選好を固定した場合の安定マッチングは μ_1，μ_2，μ_3 の三つである．それぞれの選好順に対する安定マッチングの集合が弱安定マッチングとなるため，弱安定マッチングは μ_1，μ_2，μ_3，μ_4 のみである．μ_4，μ_3，μ_2，μ_1 の順により安定となるものとする．この時，p_1 に選好を固定した場合の安定マッチングに限定すると，μ_1 のブロッキングペアが最も少ないため，μ_2 と μ_3 は最大安定度マッチングとはならない．そのため，p_1 においては，μ_1 のみが最大安定度マッチングの候補となる．図では最大安定度マッチングの候補となるもののみ実線で表し，残りは破線で表している．p_2，p_3 でも同様に考えることができ，それぞれについて最大安定度マッチングの候補を μ_1，μ_2 のように挙げることができる．もしそれぞれの選好での安定マッチングから最も不安定度が小さいマッチングを選ぶことができれば，最大安定度マッチングの候補を μ_1 と μ_2 に絞ることができる．この中から不安定度の最小となるマッチングを選んでも，確実に最大安定度マッチングを求めることができる．

ところで，本研究では選好が未知である可能性があるのを請負人側だけに限定して考えている．さらに，選好の分からない請負人の選好が完全に未知である場合，マッチングを弱安定マッチングに限定すると，ブロッキングペアとなるのは，選好の完全に分からない請負人と，その請負人を好むタスクのペアである．そのようなペアを少なくするためには，そのタスクが今のパートナーよ

り好む請負人の数を減らした方がよい．そのため，タスク側の選好順位をできるだけ上げた方がよい．タスク側の選好順位を上げるためには，タスク優位安定マッチングを求めるのがよい．これは，2.2 節で述べたように，すべてのタスクに対して，タスク優位安定マッチングでのパートナーとなる請負人よりよい請負人とパートナーになれる安定マッチングは存在しないからである．このことから，順位づけを固定した場合の不安定度の最も小さなマッチングは，タスク優位安定マッチングである．

これらを総合すると，すべてのとりうる順位付けに対してタスク優位安定マッチングを求め，その中からブロッキングペアの数が最も少ないマッチングを取り出すことで，必ず最大安定度マッチングを求めることができることが分かる．

第 4 章　アルゴリズム

この章では，これまでの議論を踏まえ，選好の未知である請負人が存在するときに，弱安定マッチングの中で最大安定度マッチングを求めるアルゴリズムを提案する．第 3 章で述べたように，選好の未知である請負人は，全タスクの選好関係が未知であるとする．

4.1　アルゴリズム

前の章で述べたように，すべてのとりうる順位付けに対してタスク優位安定マッチングを求め，その中から最大安定度マッチングをとることで，最大安定度マッチングを求めることができる．しかし，それをこのまま行おうとすると，選好が完全に未知な請負人の選好順が $n!$ 通りあるように，非常に計算量が多くなってしまう．そのため，この方針で最大安定度マッチングを求めるには工夫が必要となる．

選好の順位付けがされている場合は，2.2 節で述べたように，Gale-Shapley のアルゴリズムによりタスク優位安定マッチングを求めることができる．これを改良して，選好の順位付けが必要になるまで，選好順による場合分けを行わないという工夫を行った．

例えば，ある請負人 c が 2 つのタスク t_1，t_2 のどちらを好むか分からない時には $(t_1 \sim_c t_2)$，c が t_1 を好んだとしても，t_2 を好んだとしても，t_1 が c にプロポーズを行うことには変わりがない．あるいは，別のタスクがプロポーズをし

たり，選好が既知の請負人がプロポーズの採否を決めたりするのも，c の選考には関係せず，この時も c の選好順が実際によらず同じようにアルゴリズムを進めることができる．そのため，アルゴリズム開始時には，すべての順位付けの場合すべてを想定してアルゴリズムを行う．しかし，$t_1 \sim_c t_2$ であるときに，t_1 と t_2 がともに c にプロポーズすると，c の選好順によって結果が異なってくる．c の選好順が $(t_1 < t_2 < t_3)$ や $(t_1 < t_3 < t_2)$ などで c が t_1 を好む場合と，c は t_1 をパートナーに選んで t_2 のプロポーズを拒否するが，c の選好順が $(t_2 < t_3 < t_1)$ や $(t_3 < t_2 < t_1)$ などで c が t_2 を好む場合，c はパートナーに t_2 を選ぶ．そのため，選好順を $t_1 <_c t_2$ となる選好順と，$t_2 <_c t_1$ となる選好順に分け，c の選好で場合分けを必要がある．そのため，この時点で場合分けを行って，それぞれの場合においてアルゴリズムを続行する．このように必要な時まで場合分けを行わないのがこのアルゴリズムの特徴である．

場合分けを遅らせるメリットは，実際の選好順が違っても結果が同じになる場合に，場合分けをしなくて済むようになることである．例えば，あるタスク t_3 が選好の不明な請負人 c にプロポーズをしなければ，c の選好順 $(t_1 < t_2 < t_3)$ も $(t_1 < t_3 < t_2)$ も $(t_3 < t_1 < t_2)$ も同じように扱うことができる．さらに重要なことは，本当に必要な場合分けは，どのタスクを最も好むかの場合分けのみであるということである．t_1，t_2 の場合分けがあった後で 3 つめのタスク t_3 が請負人 c にプロポーズする場合，t_1，t_2 の場合分けで c が選んだ側のタスクと，タスク t_3 のどちらかとしか，c はマッチングされない．そのため，t_1 または t_2 と t_3 の二者の選好順序で場合分けをすればそれで十分である．このような 2 通りの場合分けは最大でもタスク数の分しかない．計算量を簡単に計算すると，すべての場合分けが n の階乗通りあるのに対し，このアルゴリズムでは，2 通りの場合分けが高々 n 回で，2^n 個の場合分けしか起こらない．全通りの選好を試す場合に比べ，計算量を抑えることもできていることが分かる．

これらをふまえて，提案するアルゴリズムをアルゴリズム 1 に示す．アルゴリズムの動作に関して説明を行う．まず，アルゴリズム 1 では実際のタスクの選好を指定し，再帰呼び出しであるアルゴリズム 2 の $almostStable$ を呼び出している．アルゴリズム 2 の $almostStable$ では，Gale-Shapley のアルゴリズムとほぼ同様にマッチングを行っていく．パートナーのいないタスクが存在すれば，そのタスクは最も選好する請負人にプロポーズをする．すべてのタスクにパートナーがいれば，タスク優位安定マッチングが成立し，マッチングを $almostStable$

アルゴリズム 1 最大安定度マッチングを求めるアルゴリズム

$pr \leftarrow$ all tasks' preference lists

return $almostStable(pr)$

アルゴリズム 2 Procedure $almostStable$(PreferenceList p)

while there exist task t not entered a company **do**
  task t proposes to its most preferred contractor c
  if c has a partner tp **then**
    if c know which task c prefers **then**
      c refuses its less preferred task $t\prime$
      delete c from $t\prime$'s preference list
    else
      copy from p to $p\prime$
      delete c from t's preference list in p
      delete c from tp's preference list in $p\prime$
      $matching1$ = almostStable(p)
      $matching2$ = almostStable($p\prime$)
      return more stable matching from $matching1$ and $matching2$
    end if
  end if
end while
return the matching

タスクの選好	請負人の選好
$t_1 : c_3 < c_1 < c_2$	c_1 : unknown
$t_2 : c_1 < c_3 < c_2$	$c_2 : t_1 < t_2 < t_3$
$t_3 : c_3 < c_1 < c_2$	$c_3 : t_2 < t_3 < t_1$

図 7: 問題例

の結果として返す．プロポーズを受けた請負人は，まだパートナーがいなければプロポーズを受諾する．もし請負人にパートナーがいれば，より選好するタスクのみプロポーズを受諾して，もう片方のタスクのプロポーズは断り，断った側のタスクの選好からその請負人を削除するのであるが，今回の問題ではどちらのタスクを好むか未知である場合がある．そのためその場合には，現在のパートナーのより好む場合と，新しくプロポーズしたパートナーを好む場合の両方の場合に場合分けをする必要がある．それぞれの場合においてアルゴリズムを継続するために，それぞれの場合における現在の状態を $almostStable$ に渡して，再帰的に呼び出している．それぞれの場合で呼び出しが終了すると，その状態で最も安定なマッチングが求められるため，得られたマッチングの中でより安定な方のマッチングを結果として返す．なお，n^2 通りのすべてのペアに対してブロッキングペアであるかを判定することで，ブロッキングペアの数は計算することができる．これにより，すべての起こりうる選好順に関するタスク優位安定マッチングの中から，最も安定となるマッチングを得ることができる．

今度は図 7 の例を用いて，アルゴリズムの動作を具体的に説明する．選好の不明な請負人 c_1 の選好は，$(t_1 < t_2 < t_3)$，$(t_1 < t_3 < t_2)$，$(t_2 < t_1 < t_3)$ など 6 通りの選好が起こりうる．しかし，実際にどの選好であれ安定結婚問題を解く手順は同じである．そのため，6 通りの選好順すべてを想定してアルゴリズムを始める．ますは，t_1 から t_3 までのそれぞれのタスクは最も好む請負人にプロポーズを行う．c_1 は t_2 のみからプロポーズを受けるためプロポーズを受諾する．一方，c_3 は t_1 と t_3 の二つからプロポーズを受けているため，より好む t_3 のみプロポーズを受諾し，t_1 のプロポーズは拒否する．プロポーズを断られた t_1 は，今度は c_3 の次に好む c_1 にプロポーズをする．この時点で，c_1 は t_1 と t_2 の両方からプロポーズを受けているが，c_1 は t_1 と t_2 のどちらを好むは未知である．選好順序が，$(t_1 < t_2 < t_3)$，$(t_1 < t_3 < t_2)$ または $(t_3 < t_1 < t_2)$ で t_1 を好む場

合と，それ以外の選好順序で t_2 を好む場合の両方の場合に場合分けを行う．c_1 が t_2 を好む場合には t_1 が断られるため，t_1 はその次に好む c_2 にプロポーズを行う．すると，マッチングが成立して，マッチングは $((t_1, c_2)(t_2, c_1)(t_3, c_3))$ となる．この時のブロッキングペアを計算すると，(t_1, c_1) のみがブロッキングペアとなり，ブロッキングペアの数は 1 になる．一方，c_1 が t_1 を好む場合には t_2 が断られれ，t_2 はその次に好む c_3 にプロポーズを行う．このように繰り返していくと，もう一度 2 通りの場合分けが起こり，ブロッキングペアの数が 2 のマッチングが二つ得られる．ここで得られた 3 つのマッチングのうち最もブロッキングペアが少ないのは，ブロッキングペアが 1 つの，最初に示した $((t_1, c_2)(t_2, c_1)(t_3, c_3))$ である．そのため，このマッチングが最大安定度マッチングとなる．

4.2　アルゴリズムの最適化

アルゴリズムを高速にするための簡単な最適化を考える．どのようなマッチングが得られるかは，プロポーズする順番に関係ないため，選好の不明な請負人がプロポーズを受けてもしばらくプロポーズを受理せず，男性全員がプロポーズを完了した後にそのプロポーズを受け入れて，採否を決めても構わない．分岐をするタイミングを遅らせることで同じ処理をする回数を減らし，計算時間を減らすことができる．

このアルゴリズムでは，より安定となるマッチングを選択するために，ブロッキングペアの数を数える必要がある．単純にブロッキングペアを数えるならば，すべてのペアに対してブロッキングペアの定義を満たすかどうかを調べればいいため $O(n^2)$ の時間がかかる．しかし，ブロッキングペアが起こるのは，選好が未知の請負人に 2 つ以上のタスクがプロポーズをした際であるのでプロポーズをするたびに随時これを計算することで，計算時間を抑えることができる．

第 5 章　アルゴリズムの評価

3.3 節で述べたように，このアルゴリズムは，弱安定マッチングの中で最もブロッキングペアの数が少ないものを選ぶことができることが保証されている．また，順位が必要となるまで順位付けによる場合分けをしていないため，すべての順位の可能性から選ぶよりは計算回数が少なくなっている．ここでは，不安定度や計算量が実際にはどの程度になるかについて論ずる．また，選好表明

の戦略に関する評価にも触れる．

ここで，選好が不明な請負人の数を $k(1 \leq k \leq n)$ とし，全請負人のうちの選好が不明である請負人の割合を $p = k/n$ とする．これを用いて評価を行う．

5.1 不安定度の評価

まずは，アルゴリズムで得られたマッチングの不安定度に関して考察する．このアルゴリズムでは，弱安定マッチングの中で最もブロッキングペアの数が少ないものを選ぶことができることは保証されているが，ここでは，それが実際にどのくらい値になるかについて述べる．

まず，選好を固定してタスク優位安定マッチングを求めると，これはただの安定結婚問題となる．安定結婚問題では，それぞれのタスクは平均して $\ln n$ 番目の請負人とペアになることができることが知られている [8]．すると，あるタスクと，そのタスクが $\ln n$ 番以内の選好が未知の請負人のペアのみがブロッキングペアとなる．$\ln n$ 番以内のタスクの中で，選好が未知であるタスクの割合は平均して p になると考えられるため，選好を固定した場合の一つのタスクあたりのブロッキングペアは，$p \ln n$ となる．そのため，全体ではブロッキングペアの数は $pn \ln n$ となる．よって，これを正規化して，不安定度は

$$\frac{p}{n} \ln n \tag{7}$$

となる．実際には，選好を固定したタスク優位安定マッチングの中から，最も不安定度の低いマッチングをとるため，不安定度はこれよりもさらに小さくなる．

今度は，実際に選好をランダムに生成してマッチングを行い，実際の不安定度がどの程度になるか測定を行う．また，考えられる簡単なアルゴリズムと比較して，どの程度よいアルゴリズムであるかを考察する．なお，問題設定自体が新しいため，既存の方法と比較することはできない．そのため，簡単なアルゴリズムとして，選好が未知の請負人の選好をひとつに決めて固定してしまうという方法を考える．比較対象とする素朴なアルゴリズムではまず，選好が未知の請負人の選好について，選好順をランダムに選んで固定をする．選好順を固定することで安定結婚問題となるため，Gale-Shapley のアルゴリズムを用いてタスク優位安定マッチングを求める．これを素朴なアルゴリズムとする．

素朴なアルゴリズムの不安定度は，提案アルゴリズムにおける不安定度と同様の議論から，式 (7) の $p/n \ln n$ という値を出すことができる．提案アルゴリズ

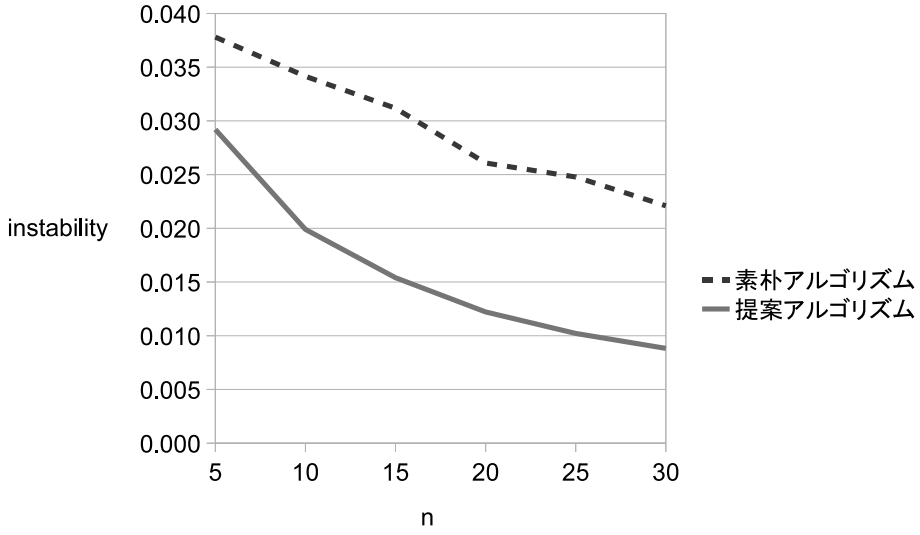

図 8: 問題の大きさによる不安定度の変化

ムは弱安定マッチングの中で最も安定度の小さいマッチングを求めることができるため，素朴なアルゴリズムより提案アルゴリズムの方が不安定度が小さくなると考えられる．これを確かめるとともに，どの程度の違いが出るか調べる．

実験方法は，n と p の値をそれぞれ変えながら，それぞれの n と p に関してランダムな選好を生成して，それぞれのアルゴリズムでマッチングを求め，不安定度を求めた．n を 5 から 30 まで 5 刻みで，p を 0 から 1 まで 0.1 刻みで変えていき，それぞれの n と p の組み合わせに対して 100 回ずつ問題を生成して，それぞれの方法で不安定度を求めた．実験結果の詳細は付録に記載する．

まず，p の値を 0.5 に固定した時の n の値による不安定度の変化を図 8 に示す．図の縦軸は不安定度で，横軸は問題の大きさ n である．はじめに述べたとおり，素朴なアルゴリズムよりも提案手法の方が不安定度が低いという結果が出た．また，n が大きくなればなるほど，素朴なアルゴリズムと比べた際に不安定度が速く 0 に近づくことが判明した．なお，p の値が 0 に近い値でなければ，p を変えても同様の傾向となっている．

今度は，p の値を変えた際の不安定度の変化を図 9 に示した．n は 30 に固定している．図の縦軸は不安定度で，横軸は p である．先ほどと同様に，素朴なアルゴリズムよりも提案手法の方が不安定度が低いという結果が出た．また，どちらの場合も p が不安定度が大きくなる．グラフからは分かりづらいが，提案

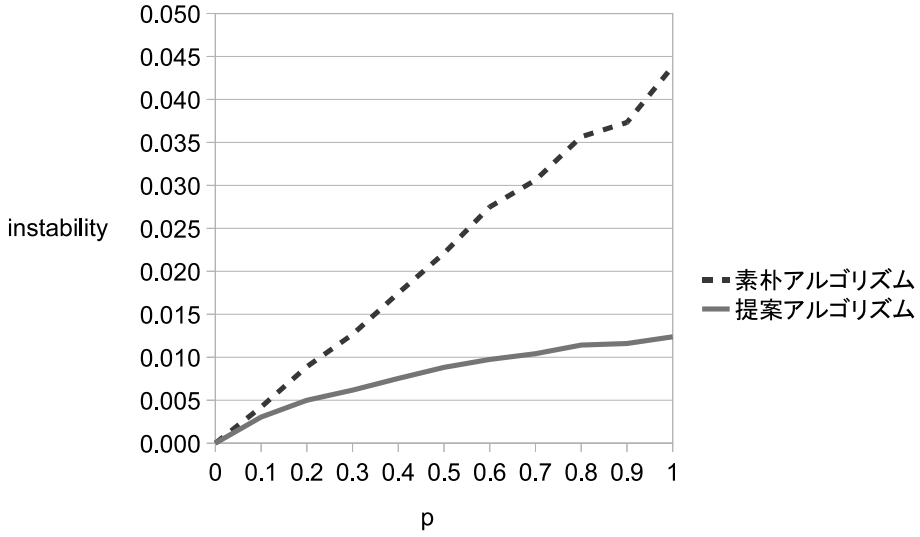

図 9: 選好の不明な請負人の割合による不安定度の変化

アルゴリズムでは，p に完全に比例をしているわけではなく，p が大きくなればなるほど，不安定度の増え方が小さくなっている．そのため，p が大きくなればなるほど，提案アルゴリズムの方が不安定度が小さな値となっている．これも，n の値によらず同じ傾向を示す．

以上から，n や p が大きくなる程，不安定度の差に大きな違いが現れてくることが分かる．n や p が大きい時，この提案アルゴリズムを使うことが有効である．しかし一方で，次の 5.2 節で詳しく議論するが，n や p が大きくなればなるほど，計算量は指数的に増えていく．一方，素朴なアルゴリズムは Gale-Shapley のアルゴリズムを解けばよいため，計算量は $n \ln n$ となる [8]．そのため，n や p が大きいほど不安定度に関しては提案アルゴリズムの方が有利となるが，その分計算時間がかかってしまい，実用的ではなくなってしまう．

なお，不安定度の数値がどのような意味を持つかは議論が必要である．n や p が大きい時に，不安定度の数倍の差が大きいとみるのであれば，今回のアルゴリズムは有効であると考えることができるが，数倍程度の差が誤差の範囲内と捉える事ができるのであれば，簡単なアルゴリズムの方を多項式時間で解ける近似アルゴリズムとしてとらえることができるかもしれない．

5.2 アルゴリズムの計算量

今度は，本研究で提案したアルゴリズムの計算量に関して論ずる．また，全探索と計算量を比較する．

まずは，すべての選好に対して全探索を行った場合に関して計算量を求める．選好が不明な請負人の選好は全部で $n!$ 通り存在する．それが k 人いるため，全部で $(n!)^k$ 通りの選好が存在する．それぞれの選好に対し Gale-Shapley のアルゴリズムを適用し，ブロッキングペアの数を数える．GaleShapley のアルゴリズムの計算量は $O(n \ln n)$ であることが知られている [8]．また，ブロッキングペアを数えるためには，n 通りのタスクと n 通りの請負人に対してブロッキングペアとなっているか数えればよいため，$O(n^2)$ となる．そのため，全選好に対する全探索を行った場合の計算量は，$O(n^3(n!)^k \ln n)$ となる．こうなると，すべてのマッチングに対してブロッキングペアを数えた方がまだよい．マッチングは，全部で $n!$ 通り存在し，それぞれに対してブロッキングペアを数えると $O(n^2)$ となるため，$O(n^2 n!)$ で求めることができる．それでも，階乗を含む計算量となってしまう．

今度は，本論文で提案した手法の計算量に関して述べる．提案したアルゴリズムにおいて，最悪の場合，全タスクがほぼすべての請負人に対してプロポーズが行われる．このとき，すべての請負人が，すべてのタスクからプロポーズを受けるとすると，二人目以降のタスクからプロポーズを受けるたびに，現在のパートナーをより好む時と，新しくプロポーズしたタスクを好む時の場合分けが発生する．この場合分けは，一人の請負人当たり $n-1$ 回起こる．選好の分からない請負人は k 人であるため，全部で $k(n-1)$ 回場合分けが起こり，$2^{k(n-1)}$ 通りの場合分けが起こる．

なお，今回提案したアルゴリズムは，場合分けが起こらなければ Gale-Shapley のアルゴリズムそのものとなり，それに場合分けが加わるため，計算量は最悪ケースで，

$$2^{k(n-1)} n \ln n \tag{8}$$

となる．

全探索で計算すると計算時間が階乗のオーダーであるのに対し，今回提案したアルゴリズムでは最悪の場合でも指数時間である．提案したアルゴリズムが総当たりと比べて，計算量が改善されていることが分かる．ただし，計算量が

タスクの選好	請負人の選好
$t_1 : c_1\ c_3\ c_2\ c_4$	$c_1 : (t_1\ t_2\ t_3\ t_4)$
$t_2 : c_1\ c_2\ c_3\ c_4$	$c_2 : t_2\ t_1\ t_3\ t_4$
$t_3 : c_3\ c_1\ c_2\ c_4$	$c_3 : t_1\ t_4\ t_3\ t_2$
$t_4 : c_4\ c_1\ c_3\ c_2$	$c_4 : t_2\ t_4\ t_1\ t_3$

図 10: 問題例

改善されているとはいえ指数時間でも十分大きな計算量であるため，現実に適用することを考えるとさらなる高速化が必要となる．

5.3 戦略的な評価

この方法では，中央集権的なマッチングで，各タスクに対して，中央のサービスに選好を報告させている．そのため，各タスクと請負人が正確な選好を報告することが重要である．

既存の研究では，Gale-Shapley のアルゴリズムを用いて安定マッチングを求める時，タスク側がプロポーズをしてタスク優位安定マッチングを求める時，それぞれのタスクにとっては選好を正直に報告するのが最適な戦略であるが，請負人には虚偽の選好を報告するインセンティブがある可能性があることが分かっている．

未知の選好を含む問題の特殊例として全員の選好が既知の問題を考えると，安定結婚問題そのものとなるため，この問題でも同様に，請負人側には虚偽報告のインセンティブが場合がある．しかし，選好が未知の場合はそれだけではなく，タスク側にも虚偽報告のインセンティブがある場合があることが判明した．図 10 に示す問題例より，タスクが持つ虚偽報告のインセンティブに関して説明する．

図 10 に示す問題において，今回提案したアルゴリズムを適用してみる．各タスクは，最も選好する請負人にプロポーズをするが，t_1 と t_2 が c_1 にプロポーズした時，c_1 は t_1 と t_2 から，より安定となるマッチングが得られるタスクを選ぶ．c_1 が t_1 を選ぶとマッチングは $((t_1, c_1), (t_2, c_2), (t_3, c_3), (t_4, c_4))$ となり，ブロッキングペアの数は 1 となるに対し，c_1 が t_2 を選ぶと $((t_1, c_3), (t_2, c_1), (t_3, c_2), (t_4, c_4))$ などのマッチングとなり，ブロッキングペアの数が 2 となる．従って，アルゴリズムではよ

りブロッキングペアの少ないt_1が選ばれ，マッチング$((t_1, c_1), (t_2, c_2), (t_3, c_3), (t_4, c_4))$が選ばれる．しかし，$t_2$が$(c_1\ c_4\ c_3\ c_2)$という虚偽の選好を報告すると，$c_1$が$t_2$を選ぶと先ほどと同様にブロッキングペアの数が2となるに対し，c_1がt_1を選ぶとブロッキングペアの数が3となってしまうため，t_2が選ばれる．その結果，t_2はc_2より好むc_1とペアになることができる．そのため，t_2には虚偽報告をするインセンティブが働いてしまう．このことから，タスク側にも虚偽報告のインセンティブが働く可能性があることが説明できる．

このようなインセンティブが働くのは，Gale-Shapleyのアルゴリズムでは，すべてのタスクにとって最適なマッチングを求めているのに対し，未知の選好を含む場合では，最大安定度という全体の利益のために，一部のタスクが犠牲となってより選好順の低い請負人をパートナーにされているためだと考えることができる．そのようなタスクが犠牲となるのは，そのタスクが犠牲になった時がもっとも安定となるためであるが，虚偽報告をしてそのタスクが犠牲となった時の安定度が大きく損なわれるようにすることで，他のタスクに犠牲をなすりつけることができる．そのような見方をすると，図10の例では，c_1を好むt_2が犠牲となってc_2とマッチングされることでブロッキングペア1の最大安定度マッチングとなるのに対し，t_2がそれを嫌って，c_1とマッチングされなかった際にブロッキングペアがより多い3となるように虚偽の選好を報告して，t_2とc_1がマッチングされなかったときに最大安定度とならないようにすることで，t_2はc_1ペアになることができる．

第6章　結論

本研究では，安定結婚問題を拡張して，未知の選好を含めることができるように定式化を行った．また，未知の選好を含む安定結婚問題に対して，最大安定度マッチングを求めるためのアルゴリズムを提案した．これにより，タスクと請負人のマッチング問題のように未知の選好を含む可能性がある場合でも，できるだけ不安定とならないようなマッチングを求めることができるようになった．

本研究では，アルゴリズムに関しては請負人の選好が完全に未知の場合に限定したが，請負人の選好が同順位のように表わされる場合においても，最大安定度マッチングを求めることが必要となってくる．さらに問題を一般化すると，タスクと請負人の両者が未知の選好を持つ可能性がある問題を考えることがで

きる．その場合も，未知の選好を同様に表現することができ，同順位を許す安定結婚問題との類推からブロッキングペアを考えることもできる．このように，本研究によって未知の選好を含むマッチング問題に対して，問題の定式化の方法を与えることができるようになった．ただし問題を一般化すると，本論文におけるアルゴリズムはそのまま用いることはできないため，さらなるアイデアが必要となってくる．

本研究にて提案しているアルゴリズムは，弱安定マッチングに限定すると不安定度を最小化することが保証されている．計算量を考察すると，全探索が階乗のオーダーであるのに対し，提案アルゴリズムでは指数時間と若干抑えられている．現実のサイズの大きなマッチング問題に適用するためには，より計算量の少ないアルゴリズムが求められる．

現実のタスク割り当ての問題においては，選好が未知である請負人であっても，タスクを精査することで正確な選好を得ることができる．タスクの精査にはコストがかかるため，精査できるタスクは少数であるが，選好が未知の場合において，単に今得られている選好のみを使うのではなく，少数でも請負人にタスクを提示して精査してもらうことで，選好情報を獲得してより安定なマッチングを得ることができると考えられる．本研究結果によって，あるタスクを提示して正確な選好が得られた時のマッチングの評価方法を与えることができるようになったが，多数あるタスクの中からどのタスクを提示するべきかの問題に関しては依然解決されていない．最大安定度マッチングを求めるために，どのタスクの精査すればいいかを求めるのは今後の課題である．

謝辞

研究に際していつも的確な助言を下さいます石田 亨 教授に感謝申し上げます．また，日頃の研究や本論文の執筆においてご指導頂きました松原 繁夫 准教授に大変厚く感謝を申し上げます，最後に，普段からお世話になっている石田・松原研究室の皆さまに感謝いたします．

参考文献

[1] Gale, D. and Shapley, L. S.: College admission and the stability of marriage, *The American Mathematical Monthly*, Vol. 69, No.1, pp. 9–15 (1962).

[2] Irving, R. W.: Stable marriage and indifference, *Discrete Applied Mathematics*, Vol. 48, pp. 261–272 (1994).

[3] 久保幹雄, 田村明久, 松井知己: 応用数理計画ハンドブック, 朝倉書店, 東京都新宿区新小川町 6-29　アクロポリス東京 10F (2002).

[4] Brito, I. and Meseguer, P.: Distributed Stable Marriage Problems, *P. van Beek (Ed.) CP 2005*, Vol. LNCS 3709, pp. 152–166 (2005).

[5] Floréen, P., Kaski, P., Polishchuk, V. and Suomela, J.: Almost Stable Matchings in Constant Time, *CoRR abs/0812.4893* (2008).

[6] Abraham, D. J., Biró, P. and Manlove, D. F.: "Almost stable" matchings in the roommates problem, *Proceedings of WAOA 2005. Springer Lecture Notes in Computer Science*, Vol. 3879, pp. 1–14 (2006).

[7] Eriksson, K. and Häggström, O.: Instability of matchings in decentralized markets with various preference structures, *International Journal of Game Theory*, Vol. 36, pp. 409–420 (2008).

[8] Knuth, D. E.: *Stable Marriage and Its Relation to Other Combinatorial Problems*, the American Mathmatical Societiy, Providence (1997).

付録：実験データ

5.1 節の不安定度の評価で用いた実験データを付録として掲載する．

実験方法は，n と p の値をそれぞれ変えながら，それぞれの n と p に対して，提案アルゴリズムと，素朴アルゴリズムにて，ブロッキングペアの数，不安定度を求めた．実験では，n を 5 から 30 まで 5 刻みで，p を 0 から 1 まで 0.1 刻みで変えていき，それぞれの n と p の組み合わせに対して 100 回ずつ問題を生成して，それぞれの方法で不安定度を求めた．ここでは，100 回の平均値を表 A.1 に表わす．

表 A.1: 実験結果

n	p	素朴アルゴリズム		提案アルゴリズム	
		ブロッキングペアの数	不安定度	ブロッキングペアの数	不安定度
5	0	0	0	0	0
5	0.1	0	0	0	0
5	0.2	0.36	0.0144	0.33	0.0132
5	0.3	0.405	0.0162	0.365	0.0146
5	0.4	0.725	0.029	0.6	0.024
5	0.5	0.945	0.0378	0.73	0.0292
5	0.6	1.22	0.0488	0.835	0.0334
5	0.7	1.24	0.0496	0.895	0.0358
5	0.8	1.24	0.0496	0.88	0.0352
5	0.9	1.54	0.0616	1.04	0.0416
5	1	1.555	0.0622	0.965	0.0386
10	0	0	0	0	0
10	0.1	0.62	0.0062	0.56	0.0056
10	0.2	1.385	0.01385	1.06	0.0106
10	0.3	2.235	0.02235	1.485	0.01485
10	0.4	2.77	0.0277	1.78	0.0178
10	0.5	3.415	0.03415	1.99	0.0199

表 A.1: 実験結果

n	p	素朴アルゴリズム		提案アルゴリズム	
		ブロッキングペアの数	不安定度	ブロッキングペアの数	不安定度
10	0.6	4.385	0.04385	2.38	0.0238
10	0.7	5.075	0.05075	2.63	0.0263
10	0.8	4.91	0.0491	2.45	0.0245
10	0.9	6.825	0.06825	3.22	0.0322
10	1	6.44	0.0644	2.975	0.02975
15	0	0	0	0	0
15	0.1	0.97	0.004311111	0.84	0.003733333
15	0.2	2.87	0.012755556	1.845	0.0082
15	0.3	3.605	0.016022222	2.23	0.009911111
15	0.4	5.63	0.025022222	3	0.013333333
15	0.5	7.015	0.031177778	3.465	0.0154
15	0.6	9.065	0.040288889	3.875	0.017222222
15	0.7	9.085	0.040377778	4.015	0.017844444
15	0.8	10.5	0.046666667	4.245	0.018866667
15	0.9	12.665	0.056288889	4.915	0.021844444
15	1	13.78	0.061244444	5.11	0.022711111
20	0	0	0	0	0
20	0.1	2.22	0.00555	1.695	0.0042375
20	0.2	4.565	0.0114125	2.72	0.0068
20	0.3	6.45	0.016125	3.465	0.0086625
20	0.4	8.42	0.02105	4.175	0.0104375
20	0.5	10.43	0.026075	4.885	0.0122125
20	0.6	13	0.0325	5.415	0.0135375
20	0.7	16.11	0.040275	6.275	0.0156875
20	0.8	15.37	0.038425	5.89	0.014725
20	0.9	19.39	0.048475	6.85	0.017125

表 A.1: 実験結果

n	p	素朴アルゴリズム		提案アルゴリズム	
		ブロッキングペアの数	不安定度	ブロッキングペアの数	不安定度
20	1	20.425	0.0510625	7.045	0.0176125
25	0	0	0	0	0
25	0.1	2.33	0.003728	1.76	0.002816
25	0.2	5.86	0.009376	3.475	0.00556
25	0.3	8.22	0.013152	4.43	0.007088
25	0.4	11.4	0.01824	5.325	0.00852
25	0.5	15.485	0.024776	6.385	0.010216
25	0.6	17.15	0.02744	6.815	0.010904
25	0.7	21.095	0.033752	7.57	0.012112
25	0.8	23.22	0.037152	8.315	0.013304
25	0.9	27.245	0.043592	8.81	0.014096
25	1	28.365	0.045384	9.255	0.014808
30	0	0	0	0	0
30	0.1	3.775	0.004194444	2.735	0.003038889
30	0.2	8.005	0.008894444	4.495	0.004994444
30	0.3	11.39	0.012655556	5.555	0.006172222
30	0.4	15.715	0.017461111	6.78	0.007533333
30	0.5	19.89	0.0221	7.94	0.008822222
30	0.6	24.74	0.027488889	8.77	0.009744444
30	0.7	27.55	0.030611111	9.37	0.010411111
30	0.8	32.095	0.035661111	10.275	0.011416667
30	0.9	33.585	0.037316667	10.44	0.0116
30	1	39.465	0.04385	11.14	0.012377778