# 特別研究報告書

# 複合 Web サービスにおける収益配分法の考案

指導教員　　松原 繁夫 准教授

京都大学工学部情報学科

古白川　亮太

平成 21 年 2 月 2 日

# 複合 Web サービスにおける収益配分法の考案

古白川　亮太

**内容梗概**

Web サービスとはネットワークを介して相互運用可能な計算機対計算機のインタラクションができるように設計されたソフトウェアシステムである．Web サービスは，単体で利用するだけでなく，複数の Web サービスをワークフローを用いて連携させて利用することもできる．前者は原子サービス，後者は複合サービスと呼ばれる．例えば，言語サービスにおいて，日英翻訳サービスと専門用語辞書サービスが原子サービスとして提供されるとき，この二つを組み合わせて専門用語日英翻訳サービスを実現することが可能である．このように，サービスを適切に組み合わせて利用者の要求を満たすサービスを提供する点が Web サービスの特徴である．

現在利用可能な Web サービスは，無償で提供されているものが多い．ここで，経済的なインセンティブを与える，つまり，サービス提供者に金銭的な報酬を与えることを考えれば，より広範な層から Web サービスが提供され，より利用者の要求に即したサービスが提供できると期待される．しかし，Web サービスのもう一つの特徴である，個々のサービス提供者が異なるという点が，収益配分に関する問題を引き起こす．

収益配分の一つの方法は，個々のサービス提供者が価格を付け，その総和を複合サービスの価格とする方法である．しかし，この提供者主導の方法では，利用者にとって価格が高くなり過ぎ，利用者が利用をあきらめる危険性がある．一方，価値ベースの価格付け，つまり，利用者がサービスに対して持つ価値相当額を課金し，それを提供者に配分する方法が考えられる．しかし，複合 Web サービスを構成する各原子サービスは，そのコストなどが異なり, 単純に利益を等分する方法では，それに納得しないサービス提供者が現れ，結局複合サービスとしての利用ができなくなる恐れがある．

この問題の解決には，シャープレイ値の適用が有望である．シャープレイ値は，ゲーム理論で提案されている概念で，貢献度に応じた配分を行おうとするものである．しかし，実際に配分決定にシャープレイ値を用いようとすると問題が生じる．本研究で取り組む課題は以下の二つである．

**特性関数値の欠損**　シャープレイ値の計算では，全てのサービスの組み合わせ

に対して，利用者が持つ価値（特性関数値）が与えられることを前提にしている．しかし，Web サービスでは，サービスの組み合わせ数が非常に多くなることが想定され，すべての特性関数値が得られるとは限らない．特性関数値に欠損がある場合，シャープレイ値を計算することができない．

**Web サービスにおける実装形態**　仮に利用者があるサービスの組み合わせに対する特性関数値を知っているとしても，まず，システムはその情報を獲得しなければならない．それをどのように実現するかは，明白ではない．

上記の課題を解決するため，特性関数値が欠損している場合に，その値を推定する方法を提案する．本稿では，単純推定と類似推定の 2 つの推定法を検討する．単純推定は，複合サービスと，そこに含まれる原子サービスの特性関数値の制約関係を利用して値を推定する方法である．類似推定は，特性関数値が不明なサービスの類似サービスの特性関数値を元に推定する方法である．Web サービスでは，同じ機能を果たすサービスが利用可能である場合が多くあるため，この方法を提案した．この方法の有効性を示すために，シミュレーションを行った．そこでは，特性関数値に関する代表的な場合に関して，特性関数値データ欠損の程度と配分計算に生じる誤差の関係を調べた．

また，後者の課題に対しては，利用者がサービス検索をするときに，同時にその価値を入力させることで，特性関数値のデータを収集する方法を検討する．また，上記の類似推定においては，サービス間の類似度の情報を得ることが必要である．これに対しては，サービスプロファイルの情報から計算することを検討する．

本研究の貢献は以下の 2 点となる．

**特性関数値に欠損がある場合のシャープレイ値の計算法の提案**　特性関数値に欠損がある場合でも，適切に推定を行うことで，シャープレイ値の計算を可能とした．特性関数値の欠損の割合が 6 割程度になっても，配分額計算の誤差は 2%に収まることをシミュレーションにより確認した．

**Web サービスにおける実装形態の提案**　利用者の Web サービス検索時に特性関数値が得られる可能性があること，また，サービス間の類似度を Web サービスのプロファイル情報から得られる可能性があることを確認した．

# Designing a Profit Sharing Method for Composite Web Services

Ryota Koshirakawa

**Abstract**

Web service is a software system that is designed that can interact between a computer and a computer through the network. Web service can be used not only with the unit but also cooperate with plural Web services by using work flow. The former is called an atomic service, the latter is called composite services. For example, in language services, Japanese-English translation service and Technical term service are offered as atomic services. It is possible that combine these services and be used as Technical term Japanese-English translation service. Like this, this is a feature of Web service that offer services that meet the user's demand by combining services appropriately.

A lot of web services that can be used now is offered free of charge. Now,if we give economical incentive, or give reward to donor of service, it is expected that more people offer services and can offer services that meet the user's demand further than before. But the problem occurs about profit sharing because of another feature of Web service, or a different donor provides services.

One of Profits sharing methods is each donor makes prices and those sum totals are assumed to be price of the composite services. But in this method, it is possible that the price rises and users can not use the service. Another Profit sharing method is making price based on value, or users pay the price that he can pay and distributed that to donors. But as for each atomic services, the cost is different, so in the method that the profit is evenly distributed, some donors don't agree and don't offer services.

To solve this problem, it can be expected that the use of Shapley value is effective. Shapley value is a concept of Game Theory, that decide the distribution ratio according to the contribution level. But the problem occurs when use shapley value to decide the distribution ratio. The problems of this paper are the following.

**Loss of values of characteristic functions** When calculate Shapley value, Values that user have(values of characteristic functions)is needed about all

combinations of services. But in Web services, the number of all combinations of services grows very much, and can not get all values of characteristic functions. If all values of characteristic functions is not got, Shapley value is not calculated.

**Mounting form in Web service** Even if users have values of characteristic functions of some combinations of services, the system have to know that. The method of that is not clearly.

To solve these problems, if loss values of characteristic functions, I suggested the method that guess the value. In this paper, I suggested two guess methods, simple guess and similarity guess. Simple guess is the guess method that guess the value from restriction about values of atomic services and composite services. Similarity guess is the guess method that guess the value from values of similar services. Because of there is many similar service in Web services, I suggested this method. This method was evaluated by using simulations. I evaluated relation between loss ratio of values of characteristic functions and error margin of distribution.

And about the latter problem, I suggested the method that We get values of characteristic functions from users by be inputed values when users search services. About Similarity guess, We have to get degree of similarity between services. For this, I suggested the method that calculate from data of service profiles. The contributions of this paper are following.

**The suggestion of the method of calculate Shapley value** Even if values of characteristic functions is loss, I enabled Shapley value to be calculated from appropriate guess is done. I confirmed if loss ratio of values of characteristic functions become 60%, error margin of distribution become 2% by using simulations.

**The suggestion of mounting form in Web service** I confirmed that when users search Web services, we can get values of characteristic functions and we can get degree of similarity between services from data of Web service profiles.

**複合 Web サービスにおける収益配分法の考案**

# 目次

**第 1 章　はじめに** **1**

**第 2 章　既存手法とその問題点** **3**

2.1　シャープレイ値 . . . . . . . . . . 3

2.2　シャープレイ値が満たす性質 . . . . . . . . . . 6

2.3　シャープレイ値の問題点 . . . . . . . . . . 7

**第 3 章　収益配分計算における特性関数値推定法の提案** **7**

3.1　推定誤差の影響 . . . . . . . . . . 7

3.2　単純推定 . . . . . . . . . . 9

3.3　類似推定 . . . . . . . . . . 10

**第 4 章　Web サービスへの提案配分法の適用** **12**

4.1　利用者からの価値情報の獲得 . . . . . . . . . . 12

4.2　類似度の計算 . . . . . . . . . . 13

4.3　推定関数の設定 . . . . . . . . . . 14

**第 5 章　評価** **15**

5.1　シミュレーション方法 . . . . . . . . . . 15

5.2　評価基準 . . . . . . . . . . 17

5.3　シミュレーション結果 . . . . . . . . . . 17

**第 6 章　終わりに** **23**

**謝辞** **24**

**参考文献** **25**

# 第1章　はじめに

Web サービスとはネットワークを介して相互運用可能な計算機対計算機のインタラクションができるように設計されたソフトウェアシステムである．これは，メッセージング Simple Object Access Protocols (SOAP)，サービス記述 Web Services Description Language (WSDL)，サービス発見 Universal Description Discovery, and Integration (UDDI) の３つの基本技術によって実現される [1]．サービスインタフェースをサービスの実装と分離することによって，一旦サービスを実装すれば，様々な応用プログラムからそのサービスを利用することが可能となる．

Web サービスは，単体で利用するだけでなく，複数の Web サービスをワークフローを用いて連携させて利用することもできる．前者は原子サービス，後者は複合サービスと呼ばれる．例えば，言語サービス基盤である言語グリッドにおいては，日英翻訳サービスと専門用語辞書サービス，例えば，ライフサインエス辞書サービスが原子サービスとして提供されており，この二つを組み合わせて複合サービスとし，ライフサイエンス用日英翻訳サービスを実現することが可能である [2]．このように，利用者の要求に応じて，複合サービスを構成し利用する点が Web サービスの特徴である．

Web サービスのもう一つの特徴は，個々のサービス提供者が異なるかもしれないという点である．現在利用可能な Web サービスは，Google や Amazon などによって提供されているもので，無償で利用可能なものがほとんどである．ここで，経済的なインセンティブを与える，つまり，サービス提供者に金銭的報酬を与えることで，大企業だけからではなく，より広範な層から Web サービスが提供されると期待できる． 利用可能な Web サービスが増えれば，より利用者の要求に即した複合サービスを構成できることになる．

しかし，価格が適切に設定されなければ，かえって利用が減退するかもしれない．例えば，個々のサービス提供者が独自に価格を設定し，複合サービスを利用する場合の価格を，そこに含まれる原子サービスの価格の和とすれば，複合サービスの価格が非常に高価なものとなって，利用者がサービスの利用をあきらめるかもしれない．多くの場合，Web サービスの限界費用，つまり，一人の利用者に追加的にサービスを提供する場合の費用は 0 に近いと考えられる．よって，全体で少し価格を下げることで，新たな利用者を獲得できる．しかし，

サービス提供者にとっても，どのようなサービスと組み合わせて利用されるかといったことを事前に知ることは難しく，適切な価格を決定することは難しい．

そこで，費用ベースではなく，利用者に対する価値ベースの価格設定を考える．これは，複合サービスに対して利用者が持つ価値相当額を支払ってもらい，それを提供者に配分する方法である．ここで，配分額決定はある程度自動化できることが望ましい．なぜなら，Web サービスにおいては，既存サービスの停止や新規サービスの提供といったことが頻繁に生じる可能性があり，また，組み合わせ方に関しても多様で，その都度，提供者間で調整することは大きな負荷となるからである．

配分決定を自動的に行う単純な方法は，サービス提供者に均等配分する方法である．しかし，この方法に対してサービス提供者から同意を得るのは難しいかもしれない．例えば，JIS，SJIS，EUC，Unicode などの文字コードを変換するサービスと，日英翻訳サービスがあったとしよう．日英翻訳サービス単体で 1 回 6 円の収入が見込めるとする．このとき，複合サービス 1 回の利用で 10 円の収入があった場合，5 円ずつ配分するというのでは，日英翻訳サービス提供者は納得しないであろう．つまり，日英翻訳サービス提供者は別のサービスと結合して利用することを禁止し，単体で収入を得ることを考えるであろう．こうして，利用者は複合サービスを利用できなくなってしまう．

この問題の解決にはシャープレイ値の利用が有望である．シャープレイ値とは，協力ゲーム理論において収益配分の方法として提案されているもので，各々の貢献度に応じて，収益を配分しようとするものである [3]．シャープレイ値は計算機科学の分野でも，マルチキャストの課金の問題や [4]，エージェントの交渉問題などに応用されている [5]．このシャープレイ値を利用すれば，均等配分で生じる問題を緩和できる．しかし，Web サービスでの収益配分にそのまま適用しようとすれば，問題が生じる．

シャープレイ値の計算では，サービスの組み合わせに対して利用者が持つ価値がすべて与えられていることを前提にしている．例えば，サービス $A$，$B$，$C$ があったとき，原子サービス $A$ の価値，$B$ の価値，$C$ の価値，複合サービス $(A, B)$ の価値，$(B, C)$ の価値，$(C, A)$ の価値，$(A, B, C)$ の価値が与えられていることを仮定している．しかし，収益配分の場面では，すべての値が与えられているとは限らず，その場合には配分計算ができなくなる．

本研究での課題をまとめると以下となる．

**特性関数値の欠損**　シャープレイ値の計算では，全てのサービスの組み合わせに対して，利用者が持つ価値（特性関数値）が与えられることを前提にしている．しかし，Web サービスでは，サービスの組み合わせ数が非常に多くなることが想定され，すべての特性関数値が得られるとは限らない．特性関数値に欠損がある場合，シャープレイ値を計算することができない．

**Web サービスにおける実装形態**　仮に利用者があるサービスの組み合わせに対する特性関数値を知っているとしても，まず，システムはその情報を獲得しなければならない．それをどのように実現するかは，明白ではない．

この問題を解決するために，本研究では，価値が未知のサービスに対して，類似の機能を持つサービスから価値を推定し，そのもとでシャープレイ値を計算し，収益配分を決定する方法を提案する．さらに，シミュレーションを用いて，値に欠損がある場合とない場合のシャープレイ値を比較し，提案方法が有効となる条件を確認する．

以降，本稿の構成は以下の通りである．第 2 章で既存手法としてシャープレイ値と実応用における問題点を説明し，第 3 章では特性関数値の推定法を提案する．第 4 章で Web サービスに提案方法を適用する場合の，実現可能性について検討し，第 5 章で提案方法の評価を行う．第 6 章をまとめとして本稿を締めくくる．

# 第 2 章　既存手法とその問題点

本章では，本研究で用いるシャープレイ値について説明する．本研究でシャープレイ値に着目するのは，以下で述べるように，シャープレイ値が理論的に多くの望ましい性質を満たすからである．一方で，実応用を考えると，シャープレイ値計算の前提が成立しない場合がある．それを問題点として指摘する．

## 2.1　シャープレイ値

シャープレイ値とは，譲渡可能な効用を持つ提携形ゲームの解概念で，ゲームにおいて各プレイヤーが獲得できると予想される利得を表すものとして提案された値である．協力ゲームにおいて，プレイヤー集合 $N$ の部分集合を提携といい，各提携について，その提携で獲得できる利得を対応させる関数 $v$ を特性関数という．例えば，プレイヤー $A$ とプレイヤー $B$ の組み合わせを提携 $\{A, B\}$

と表し，その提携によって獲得できる利得を，$v(\{A, B\})$ と表現する．あるプレイヤーについて，そのプレイヤーが含まれる提携の利得と，そのプレイヤーが含まれない提携の利得の差は，そのプレイヤーによって追加的に付加される利得であり，それをそのプレイヤーの限界貢献度という．限界貢献度の厳密な定義は後に述べる．シャープレイ値とは，各プレイヤーの限界貢献度の期待値である．すなわち，シャープレイ値とはそのプレイヤーが利得獲得に対して，どの程度貢献出来るかを表す値である．

複合サービスの収益配分において，特性関数値は，提携に対応する原子／複合サービスに対して，利用者が支払っても良い金額を表す．ある複合サービスが利用され，収入が得られた場合，各サービスのシャープレイ値の比に従って配分するとは，各サービスの貢献度の比で配分するということを意味する．

プレイヤー $i$ のシャープレイ値は以下で定義される．

$$\phi_i(v) = \frac{1}{n!} \sum_{\pi \in \Pi} \{v(S_{\pi,i} \cup \{i\}) - v(S_{\pi,i})\}$$

ここで，$\Pi$ は，1 から $n$ までの数字の並べ方の集合，$S_{\pi,i}$ は，並べ方 $\pi$ において $i$ より前に並んでいるプレイヤーの集合を表す．この定義式は，次のように解釈できる．$n$ 人のプレイヤーが一人ずつ集まってくる状況を考える．今，ある $\pi \in \Pi$ という並べ方の順番でプレイヤーが集まるとする．プレイヤー $i$ が現れたとき，特性関数値は $v(S_{\pi,i} \cup \{i\}) - v(S_{\pi,i})$ だけ増加する．この値を並べ方 $\pi$ におけるプレイヤー $i$ の限界貢献度と定義する．$n$ 人のプレイヤーがランダムに選ばれた順番で集まるとすると，一つの並べ方が生起する確率は $1/n!$ と考えられる．上記の $\phi_i(v)$ は，そのような場合におけるプレイヤー $i$ の限界貢献度の期待値を表している．

具体例を図 2.1 に示す．

プレイヤー数が 3 の場合のシャープレイ値の計算方法を以下に示す．

1. プレイヤー $A$，$B$，$C$ の並べ方を全て求める．
2. $ABC$ という並べ方の場合，$A \to B \to C$ という順番でプレイヤーが集まると考えるので，$A$ の限界貢献度は，$v(\{A\})$，$B$ の限界貢献度は $v(\{A, B\}) - v(\{A\})$，$C$ の限界貢献度は $v(\{A, B, C\}) - v(\{A, B\})$ と計算される．
3. 全ての並べ方に対し，同様にして各プレイヤーの限界貢献度を求める．
4. 各プレイヤーについて，各並べ方における限界貢献度の平均をそのプレイ

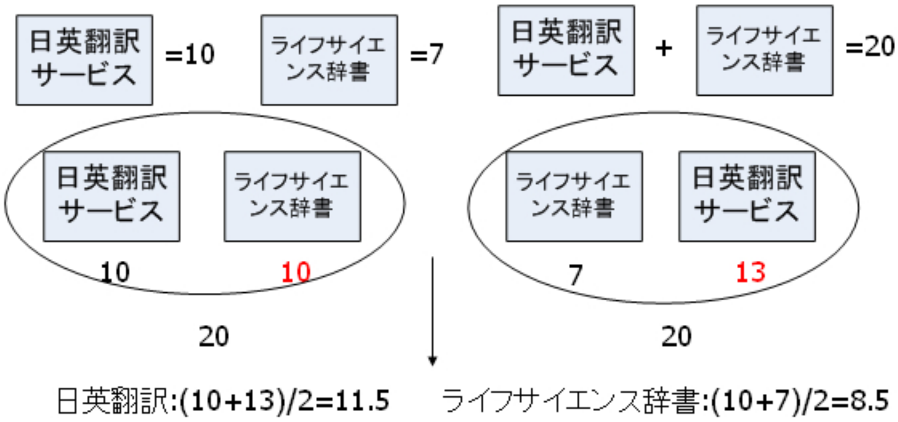



図 2.1: シャープレイ値の計算例

ヤーのシャープレイ値とする．

具体的な値を与えてシャープレイ値がどう計算されるかを示す．3 人のプレイヤー $N = \{A, B, C\}$ に対する各提携の特性関数値を，$v(\{A, B, C\}) = 10$，$v(\{A, B\}) = 8$，$v(\{B, C\}) = 5$，$v(\{A, C\}) = 4$，$v(\{A\}) = v(\{B\}) = v(\{C\}) = 1$，$v(\{\emptyset\}) = 0$ とする．このとき，各プレイヤーの限界貢献度は表 2.1，シャープレイ値と配分は表 2.1 のようになる．

表 2.1: プレイヤーの限界貢献度の計算法

| | $A$ の限界貢献度 | $B$ の限界貢献度 | $C$ の限界貢献度 |
|---|---|---|---|
| $ABC$ | $v(\{A\}) - v(\{\emptyset\}) = 1$ | $v(\{A, B\}) - v(\{A\}) = 7$ | $v(\{A, B, C\}) - v(\{A, B\}) = 2$ |
| $ACB$ | $v(\{A\}) - v(\{\emptyset\}) = 1$ | $v(\{A, B, C\}) - v(\{A, C\}) = 6$ | $v(\{A, C\}) - v(\{A\}) = 3$ |
| $BAC$ | $v(\{A, B\}) - v(\{B\}) = 7$ | $v(\{B\}) - v(\{\emptyset\}) = 1$ | $v(\{A, B, C\}) - v(\{A, B\}) = 2$ |
| $BCA$ | $v(\{A, B, C\}) - v(\{B, C\}) = 5$ | $v(\{B\}) - v(\{\emptyset\}) = 1$ | $v(\{B, C\}) - v(\{B\}) = 4$ |
| $CAB$ | $v(\{A, C\}) - v(\{C\}) = 3$ | $v(\{A, B, C\}) - v(\{A, C\}) = 6$ | $v(\{C\}) - v(\{\emptyset\}) = 1$ |
| $CBA$ | $v(\{A, B, C\}) - v(\{B, C\}) = 5$ | $v(\{B, C\}) - v(\{C\}) = 4$ | $v(\{C\}) - v(\{\emptyset\}) = 1$ |
| Total | 22 | 25 | 13 |

表 2.2: シャープレイ値と配分

| | $A$ | $B$ | $C$ |
|---|---|---|---|
| シャープレイ値 | $\frac{22}{6}$ | $\frac{25}{6}$ | $\frac{13}{6}$ |
| 配分 | 37% | 42% | 22% |

## 2.2 シャープレイ値が満たす性質

プレイヤーを $N$，特性関数型ゲームを特性関数 $v$ で表したとき，ゲーム $v$ に対して $n$ 個の値を対応させる関数

$$\theta(v) = (\theta_1(v), \cdots, \theta_n(v))$$

をゲームの値と呼ぶ．シャープレイ値もゲームの値の一つである．ゲームの値が，協力ゲームにおいて各プレイヤーが得られる利得を表すのであれば，そのようなゲームの値が満たすべき四つの性質がある．以下にそれらを述べる．

**全体合理性：** $\sum_{i \in N} \theta_i(v) = v(N)$

提携全体で獲得した利得がすべて各プレイヤーに分け与えられるという性質を表す．

**ナルプレイヤーのゼロ評価：** 任意のナルプレイヤー $i$ に対して $\theta_i(v) = 0$

プレイヤー $i$ と，任意の提携 $S$ について，$v(S \cup \{i\}) = v(S)$ が成り立つとき，$i$ をナルプレイヤーと呼ぶ．これは，ナルプレイヤー $i$ がどんな場合も追加的な利得を生み出さないことを意味している．ナルプレイヤーのゼロ評価は，常に利得を生まないプレイヤーが得る利得は常にゼロであるという性質を表す．

**対称性：** 対称なプレイヤー $i$ と $j$ に対して $\theta_i(v) = \theta_j(v)$

プレイヤー $i, j$ と，$i, j$ を含まない任意の提携 $S$ に対して，$v(S \cup \{i\}) = v(S \cup \{j\})$ が成り立つとき，$i$ と $j$ は対称であるという．対称性は，対称なプレイヤーは同じ利得を得るという性質を表す．

**加法性：** 任意の特性関数ゲーム $v, w$ について，$\theta_i(v + w) = \theta_i(v) + \theta_i(w)$ $(i = 1, \cdots, n)$

二つのゲーム $v, w$ に対し，二つのゲームを一括して一つのゲームとしてとらえたものをゲーム $v+w$ で表す．$v+w$ におけるゲームの値は，二つのゲーム $v, w$ におけるゲームの値の合計でなければいけないという性質を表す．

シャープレイ値は，これら 4 つの性質を全て満たす．また，これらの 4 つの性質を全て満たすゲームの値はシャープレイ値だけであることも証明されている．これらの性質は，複合 Web サービスの収益分配において満たされているべき性質である．また，シャープレイ値は特性関数の値が与えられればつねに解が一意に定まるという性質も持っている．これらの性質から，複合 Web サービスの収益配分においてシャープレイ値の適用が適切であると考えられる．

## 2.3 シャープレイ値の問題点

実際に複合 Web サービスの収益配分決定にシャープレイ値の適用を考えると，以下の問題が生じる．シャープレイ値の計算は，全ての提携の特性関数値が得られることを前提としている．しかし，実応用の場面では，全ての提携の特性関数値を得ることは困難である．一つは，サービスとして利用されない組み合わせがある点である．例えば，専門語日英翻訳サービスは，形態素解析サービス，専門辞書サービス，日英翻訳サービスから構成される．ここで，形態素解析サービスは，専門語辞書を検索するために必要なものであり，形態素解析サービスと日英翻訳サービスを組み合わせても，一般の利用者にとっては，日英翻訳の品質を向上させるものではないため，そのような組合わせで使用されることはない．よって，形態素解析サービスと日英翻訳サービスからなる複合サービスに対して，利用者が持つ価値を獲得することは難しい．

また，Web サービスは必要なサービスを組み合わせて利用者の要求を実現するというものであるため，広範な要求に応えようとすると，その数が多くなる．サービスの数が 3 個の場合，提携の数は 7 個であるが，サービスの数が 10 個になると，提携の数は 1023 個となる．一般には，サービスの数が $n$ 個になると，提携の数は $2^n - 1$ 個となる．さらに状況を悪くする要因として，提供されるサービスは固定的でなく，頻繁に新規サービスが提供されるという点が挙げられる．よって，すべての組み合わせに対して，利用者が持つ価値を獲得することは難しい．つまり，シャープレイ値を用いて収益の配分を決定することが困難になる．

# 第 3 章　収益配分計算における特性関数値推定法の提案

## 3.1 推定誤差の影響

まず，特性関数値の推定誤差が配分にどのような影響を与えるかを見る．サービス $\{A, B, C\}$ が存在し，特性関数値 $v(\{A, B\})$ の値が未知であるとする．ここで，本来の値 $v(\{A, B\})$ を $v(\{A, B\}) + \alpha$ と推定したとしよう．このとき，各サービスが受け取る配分 $\phi'$ は以下のように計算される．$\phi$ は本来の値を用いた

ときの配分を表す．

$$
\begin{aligned}
\phi'_A &= \frac{1}{3!}((v(\{A\}) - v(\{\emptyset\})) + (v(\{A\}) - v(\{\emptyset\})) + (v(\{A,B\}) + \alpha - v(\{B\})) \\
&\quad + (v(\{A,B,C\}) - v(\{B,C\})) + (v(\{A,C\}) - v(\{C\})) \\
&\quad + (v(\{A,B,C\}) - v(\{B,C\}))) \\
&= \phi_A + \frac{1}{3!}\alpha
\end{aligned}
$$

同様に計算して，以下を得る．

$$
\begin{aligned}
\phi'_B &= \phi_B + \frac{1}{3!}\alpha \\
\phi'_C &= \phi_C - \frac{2}{3!}\alpha
\end{aligned}
$$

さらに，特性関数値 $v(\{A,C\})$ の値が未知であるとし，本来の値 $v(\{A,C\})$ を $v(\{A,C\}) + \beta$ と推定したとしよう．このとき，各サービスが受け取る配分 $\phi''$ は以下のように計算される．

$$
\begin{aligned}
\phi'_A &= \frac{1}{3!}((v(\{A\}) - v(\{\emptyset\})) + (v(\{A\}) - v(\{\emptyset\})) + (v(\{A,B\}) + \alpha - v(\{B\})) \\
&\quad + (v(\{A,B,C\}) - v(\{B,C\})) + (v(\{A,C\}) + \beta - v(\{C\})) \\
&\quad + (v(\{A,B,C\}) - v(\{B,C\}))) \\
&= \phi_A + \frac{1}{3!}(\alpha + \beta)
\end{aligned}
$$

同様に計算して，以下を得る．

$$
\begin{aligned}
\phi'_B &= \phi_B + \frac{1}{3!}\alpha - \frac{2}{3!}\beta \\
\phi'_C &= \phi_C - \frac{2}{3!}\alpha + \frac{1}{3!}\beta
\end{aligned}
$$

$\alpha$ や $\beta$ の値は，正負いずれの場合も考えられる．本来得られるべき配分よりも少ない配分しか得られないサービス提供者は，不満を持つことになる．特性関数値で未知の部分が増えたからといって，単調にサービス提供者の不満が増大するわけではない．しかし，未知部分が増えると，配分計算の妥当性が弱まることになり，より精密に推定することが求められる．

## 3.2 単純推定

提携の特性関数値が不明な場合，特性関数値を推定して配分を決定する方法を提案する．従来のシャープレイ値計算のフローチャートを図 3.1 に，推定を用いたシャープレイ値計算のフローチャートを図 3.2 に示す．

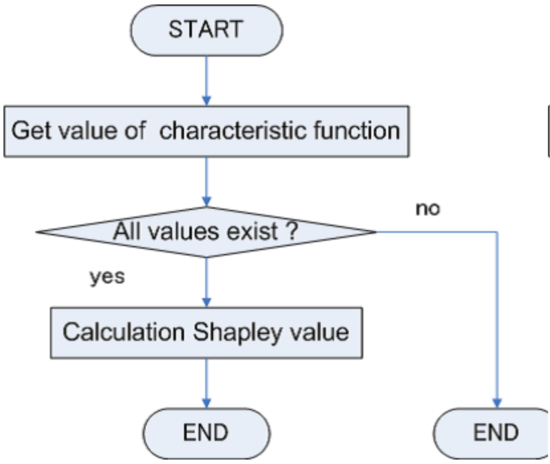



図 3.1: シャープレイ値計算のフローチャート

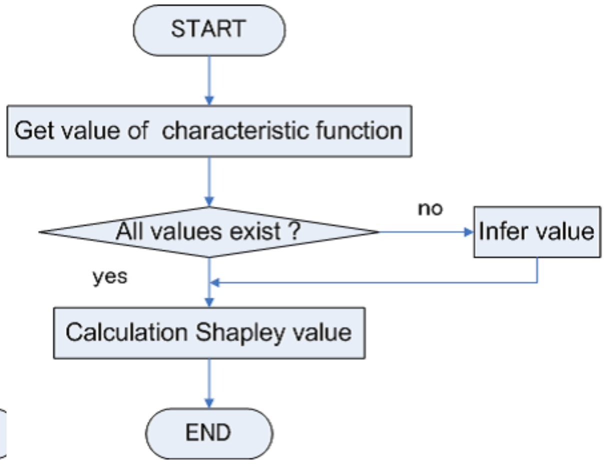



図 3.2: 推測を用いたシャープレイ値計算のフローチャート

サービス $S$ の特性関数値が未知であるとして，その値の推定について，2 つの方法を考える．

**単純推定：** サービス $S$ の構成サービス，被構成サービスの評価値からの制約

**類似推定：** 類似サービスの評価値

本節で単純推定について，次節で類似推定について説明する．

複合サービス $\{A, B\}$ はワークフローを用いて結合される．よって，サービス $A$ のみを機能させる，サービス $B$ のみを機能させることが可能である．よって，複合サービスの価値は，少なくともその任意の部分サービスの価値よりも大きくなる．

$$v(\{A, B\}) \geq max\{v(\{A\}), v(\{B\})\}$$

サービス $\{A\}$ とサービス $\{B\}$ の両機能を必要とする利用者の存在を仮定すれば，以下の関係が導ける．

$$v(\{A, B\}) \geq v(\{A\}) + v(\{B\})$$

原子サービス $A$ については，以下の関係が成り立つ．

$$v(\{A\}) \geq 0$$

また，複合サービス $\{A, B\}$ は，サービス $C$，あるいは，サービス $D$ と結合されて，別の複合サービス $\{A, B, C\}$，あるいは，$\{A, B, D\}$ を構成するかもしれない．このとき，以下の関係が成立する．

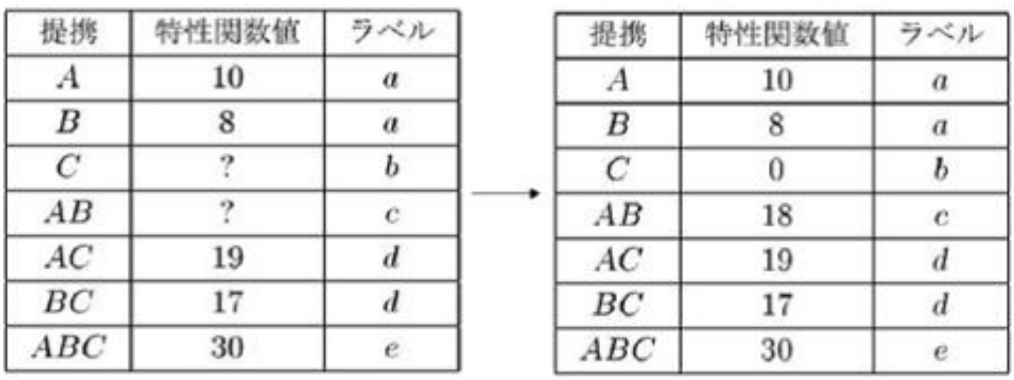
$$v(\{A, B\}) \leq min\{v(\{A, B, C\}), v(\{A, B, D\})\}$$

以上より，特性関数値を推定したい提携に関して，任意の 2 つの提携に分割し，部分提携の特性関数値の和が最大となる組み合わせの特性関数値を，その提携の推定特性関数値とする．これは，以下のように表現される．

$$v(S) = \max_{S' \in S}(v(S'), v(S/S'))$$

$S$ は複合サービスであり，$S'$ は $S$ の任意の部分集合である．また，$S/S'$ は，部分集合 $S'$ に対する補集合を表す．原子サービスの価値が不明な場合，その価値は 0 とする．

単純推定の例を図 3.3 に示す．



| 提携 | 特性関数値 | ラベル |
| --- | --- | --- |
| $A$ | 10 | $a$ |
| $B$ | 8 | $a$ |
| $C$ | ? | $b$ |
| $AB$ | ? | $c$ |
| $AC$ | 19 | $d$ |
| $BC$ | 17 | $d$ |
| $ABC$ | 30 | $e$ |

| 提携 | 特性関数値 | ラベル |
| --- | --- | --- |
| $A$ | 10 | $a$ |
| $B$ | 8 | $a$ |
| $C$ | 0 | $b$ |
| $AB$ | 18 | $c$ |
| $AC$ | 19 | $d$ |
| $BC$ | 17 | $d$ |
| $ABC$ | 30 | $e$ |

図 3.3: 単純推定の例

## 3.3 類似推定

単純推定は，計算に別の付加的な情報を必要とせず，いつでも適用可能な点が利点である．ただし，この推定法では，複合サービスが付加価値を生まないということになり，推定値は実際の特性関数値と等しいか，より小さくなって

しまう．そもそも，Web サービスの世界では，原子サービスでは利用者の要求を満たさない状況で，原子サービスを組み合わせることで解決を図ろうとしている．つまり，複合サービスが利用者にとって大きな価値をもたらすという世界を対象にしている．よって，特性関数値を過小に見積り過ぎることは問題である．

そこで，本研究では，特性関数値が不明なサービスに対し，その類似サービスに着目する．類似サービスが存在し，またその類似サービスの特性関数値がわかっている場合に，その値で代用しようとする考え方である．

まず本稿で取り扱う問題のモデルを与える．2 つのサービス $s_i$ と $s_j$ の間には，類似度 $sim(s_i, s_j)$ が定義される．ここで，関数 $sim$ は，2 つのサービスが同一である場合には 1 を返し，まったく異なる場合には 0 を返すとする．つまり，$0 \leq sim(s_i, s_j) \leq 1$ が満たされる．これを持って，特性関数値 $v$ の推定値 $v'$ を以下のように与える．

$$v'(S) = \begin{cases} v(S) & v(S) \text{ が既知のとき} \\ f(S) & v(S) \text{ が未知のとき} \end{cases}$$

$f$ は推定関数である．$f$ としては，様々なものが考えられる．例えば，以下は，類似度が最大となるものを一つ選んで，その値を推定値とするものである．

$$f(S) = \max_{S'}(v(S')|sim(S, S') \text{ が最大})$$

類似推定の例を図 3.4 に示す．これは，類似度が 0 が 1 の 2 値で表される場合に対応する．

| 提携 | 特性関数値 | ラベル |
|---|---|---|
| $A$ | 10 | $a$ |
| $B$ | ? | $a$ |
| $C$ | 5 | $b$ |
| $AB$ | 20 | $c$ |
| $AC$ | ? | $d$ |
| $BC$ | 17 | $d$ |
| $ABC$ | 30 | $e$ |

$\longrightarrow$

| 提携 | 特性関数値 | ラベル |
|---|---|---|
| $A$ | 10 | $a$ |
| $B$ | 10 | $a$ |
| $C$ | 5 | $b$ |
| $AB$ | 20 | $c$ |
| $AC$ | 17 | $d$ |
| $BC$ | 17 | $d$ |
| $ABC$ | 30 | $e$ |

図 3.4: 類似推測の例

この推定値が計算されると，プレイヤー $i$ のシャープレイ値は以下で計算さ

れることになる．

$$\phi_i(v') = \frac{1}{n!} \sum_{\pi \in \Pi} \{v'(S_{\pi,i} \cup \{i\}) - v'(S_{\pi,i})\}$$

先と同じく，$\Pi$ は，1 から $n$ までの数字の順列の集合，$S_{\pi,i}$ は，並べ方 $\pi$ においてi より前に並んでいるプレイヤーの集合を表す.

ここで，シャープレイ値の計算量について述べておく．シャープレイ値の計算はサービス数 $N$ に対して，それから得られる順列の数だけの計算が必要である．Web サービスにおいては，サービス数が多くなることが想定されるため，特性関数値の推定ができたとしても，シャープレイ値の計算は現実的でなくなる．しかし，様々な近似計算の手法が提案されており [6]，それらを適用することで，計算量の問題に対処できると考えている．この点に関する，より詳細な検討は今後の課題である．

# 第 4 章　Web サービスへの提案配分法の適用

第 3 章章では，未知の特性関数値を推定する方法を示した．提案

方法を，実際に Web サービスにおける収益配分に適用するには，例えば，類似度の求め方などを具体的に定める必要がある．本章では，それを検討する．

## 4.1　利用者からの価値情報の獲得

第 3 章章では，未知の特性関数値を推定する方法を示したが，既知とした特性関数値に関しても，利用者から獲得する必要がある．ここで，特性関数値とは，そのサービスに対する評価値，すなわち，利用者が支払っても良いという金額に対応する．本稿では，利用者は必要なサービスの仕様を入力し，Web サービス合成の技術を用いて，複合サービスを構成するとする．Web サービス合成に関しては，例えば，制約充足に基づく方法などが提案されている [7]．この際に，利用者に支払っても良い金額を入力させ，そのデータを蓄積することとする．

本研究では，評価値を，評価値そのものと評価者 ID で特徴づける．サービスの機能が同じといっても，実装上その品質が異なるかもしれない．複数の日英翻訳サービスがあるとして，日英翻訳としう機能は変わらないが，辞書の語彙数などに異なりがあるかも知れない．つまり，検索時に入力された評価値は，利用者がサービスに対して持つ正確な値ではないかもしれない．これに対して

は，例えば，提示されたサービスを，当該の利用者が再度利用すれば，その価格を支払うことに納得したと考えることができ，特性関数値計算において，その値の重みを増すといった処理が考えられる．

さて，実装上の品質だけではなく，同一のサービスであっても，利用者が異なれば，その評価値は異なる可能性がある．例えば，英語の知識がほとんどない者にとっては，日英翻訳サービスが簡単な文章の翻訳しかできなくても利用する価値はあるであろう．一方，英会話に堪能な者にとっては，簡単な文章の翻訳しかできなければ，そのサービスを利用しようとは思わないであろう．これは，特性関数値に関して利用者毎に異なる値を保持すべきか，利用者全体で集約した値を保持すべきかという，システム設計上に関する問題をもたらす．

課金する立場から見ると，利用者毎に課金額が変われば，利用者が混乱するかもしれず，一定の金額を課金するのが望ましいと言える．そこで，各サービスに対して，一つの特性関数値を保持する，つまり，一つの価格を保持することにする．なお，一つの価格に決めるとした場合，提供者の利益を最大化する価格は利用者の評価値分布に依存する．本研究では，簡単のため，あるサービスに関しては，利用者から得られた評価値を平均した額をそのサービスに対する特性関数値として用いることにする．

## 4.2 類似度の計算

類似度に基づく特性関数値の推定を行うためには，まず 2 つのサービス間の類似度の計算ができること，つぎに，定義された類似度に基づいて，類似 Web サービスを検索できることが必要である．サービス間の類似度の判定には，サービスプロファイルの記述を用いる．例えば，言語グリッドにおいては，各サービスに対して，以下の情報を獲得可能である．

例えば，日英翻訳サービス J-Server と WEB-Transer は，Resource Type が同一で，Language にともに，Japanese と English の言語対を含むため，類似しているとみなせる．この重なりに応じて，0 から 1 の間の値を与えることが可能であるが，ここでは，類似しているか，していないかの 2 値で考えることとする．

一般に，どのような形でプロファイル記述が与えられるか規定することは難しいため，適応領域に応じた形で，類似度判別関数を定義することになる．さらに言えば，プロファイル記述が機会可読な形で与えられている保証もない．WSDL はつねに与えられ，そこにはどのような方法でサービスを呼び出せばよ

表 4.1: サービスプロファイル記述の例

J-Server

| | |
|---|---|
| Resource Type | Machine Translator |
| Languages | Japanese⟨-⟩English, Japanese⟨-⟩Korean, Japanese⟨-⟩Chinese |
| Copyright | KODENSHA, Co., Ltd. |
| License | – |
| Requirements | OS: Windows2000/2003 |

WEB-Transer

| | |
|---|---|
| Resource Type | Machine Translator |
| Languages | Japanese⟨-⟩English, Japanese⟨-⟩Korean,Japanese⟨-⟩Chinese, English⟨-⟩Italian, English⟨-⟩German, English⟨-⟩French, English⟨-⟩Spanish, English⟨-⟩Portuguese |
| Copyright | Cross Language Inc. |
| License | – |
| Requirements | OS: Windows2000/2003 |

Life Science Dictionary

| | |
|---|---|
| Resource Type | Multilingual Dictionary |
| Languages | Japanese, English |
| Copyright | Shuji Kaneko |
| License | http://lsd.pharm.kyoto-u.ac.jp/en/about/policy/index.html |
| Requirements | None. |

いかが定義されている．しかし，WSDL の中に，どのようなサービスであるかの定義が十分に与えられているとは限らない．これらの点は，Web サービスの技術動向を観測し，それに対応していく必要がある．

## 4.3 推定関数の設定

類似度が与えられれば，つぎは，推定関数を設定する必要がある．第 3 章では，一例として，類似度が最大となるものを一つ選んで，その値を推定値とする推定関数を定義した．ここで，考えるべきは，計算の対象とする範囲である．

あるライフサイエンス向け日英翻訳サービス $A$ の特性関数値が未知であるとして，別のサービスの特性関数値を用いて推定するとする．ここで，別のライフサイエンス向け日英翻訳サービス $B$ が存在すれば，類似度という点では問題がない．しかし，その評価値がごく少数の利用者から与えられたものであれば，サービス $A$ が本来持つ価値と大きく掛け離れたものである可能性がある．このような場合には，類似の基準を下げ，対象とする範囲を，一般向け日英翻訳サービスまで広げ，その平均値を計算するといった方法が考えられる．ただし，あまりに対象とする範囲を広げすぎると，元のサービスの特徴を捉え損ねて，過度に平均化された値を特性関数値としてしまう恐れがある．

この問題を一般的に扱うのは難しいため，本稿では，類似度が 0 か 1 の 2 値で与えられ，類似サービスの評価値の平均値をもって，推定することとする．より，厳密な取り扱いは，今後の課題である．

# 第 5 章　評価

本章では，推定の評価を行うためのシミュレーションの方法と，シミュレーションの結果を示し，それについて考察する．

## 5.1　シミュレーション方法

まず，具体的なシミュレーション方法を示す．

1. 各原子サービスと複合サービス全てに特性関数値を与える．
2. 各サービスについて，それぞれの特性関数値を元にラベルを与える．
3. 各プレイヤーについてシャープレイ値を求める．
4. $n$ 個の特性関数値を欠損させる．($0 < n < m \quad m$ は提携の総数)
5. 欠損している特性関数値を推定する．
6. 推定された特性関数値を用いて，各プレイヤーについて推定シャープレイ値を求める．
7. 真のシャープレイ値と推定シャープレイ値の誤差を求め，特性関数値の欠損数 $n$ に対するシャープレイ値の誤差を求める．

2 の，特性関数値を与えるときに，様々な場合の値の与え方，例えば各プレイヤーに対して価値が一様である場合や，一つのサービスだけ価値が大きい場合などを考え，それぞれの場合に特性関数値の欠損数とシャープレイ値の誤差が

どのような相関を示すかを調べる．3 のラベル付けであるが，本来ならば，各サービスの類似度に応じてラベルを付けるべきであるが，ここでは問題の単純化のため，類似のサービスが生み出す価値は近い値になるであろうという推定から，特性関数値が近いサービスに同じラベルを与えるとした．6 の推定は，第 3 章で説明した推定法を用いる．

なお，具体的な特性関数値の与え方であるが，原子サービスについては，$[0,1]$ の一様乱数に何らかの定数値をかけた値を与えた．複合サービスについては，その部分サービスの特性関数値に関わらず優加法性が満たされるような値を与え，そこに更に付加価値が発生するとして，$[0,1]$ の一様乱数に何らかの定数値をかけた値を加えたものを特性関数値として与えた．ここで，$V$ を，そのプレイヤーが取りうる特性関数値の最大値を表す関数と定義する．例えば，$k_A$ は定数，$r$ は $[0,1]$ の一様乱数とし，

$$v(\{A\}) = k_A r$$

とすると，

$$V(\{A\}) = k_1$$

となる．$v(\{A,B\}) \geq V(\{A\}) + V(\{B\})$ となるように $v(\{A,B\})$ の値を与えれば，$v(\{A,B\}) \geq v(\{A\}) + v(\{A\})$ の優加法性を常に満たす．

プレイヤー数が 3 の場合の特性関数値の与え方の例を以下に示す．

$$\begin{aligned}
v(\{A\}) &= k_A r \\
v(\{B\}) &= k_B r \\
v(\{C\}) &= k_C r \\
v(\{A,B\}) &= V(\{A\}) + V(\{B\}) + k_{AB} r = k_A + k_B + k_{AB} r \\
v(\{B,C\}) &= V(\{B\}) + V(\{C\}) + k_{BC} r = k_B + k_C + k_{BC} r \\
v(\{A,C\}) &= V(\{A\}) + V(\{C\}) + k_{AC} r = k_A + k_C + k_{AC} r \\
v(\{A,B,C\}) &= max\{V(\{A,B\}) + V(\{C\}), V(\{A,C\}) + V(\{B\}), \\
&\quad V(\{B,C\}) + V(\{A\})\} + k_{ABC} r \\
&= max\{k_A + k_B + k_{AB} + k_C, k_A + k_C + k_{AC} + k_B, k_B + k_C + k_{BC} + k_A\} \\
&\quad + k_{ABC} r
\end{aligned}$$

但し，$k_i \ \ (i = A, \ldots, ABC)$ は定数，$r$ は $[0,1]$ の一様乱数である．

## 5.2　評価基準

シミュレーションの評価において，欠損率と誤差率を定義する．欠損率は，特性関数値が与えられていない提携の個数 ÷ 全ての提携の個数で定義する．誤差率は，真のシャープレイ値と推定シャープレイ値の差を，各プレイヤーのシャープレイ値の総和で割ったものと定義する．これは，推定を用いて計算した各プレイヤーが得られる利得が本来得られる利得に対し，どの程度乖離しているかを表している．例えば，誤差率が 0.1 とは，各プレイヤーについて，推定により計算された配分が，本来の配分に対し，全体で得られる利益の 10% 程度の差が生じるということを意味する．

## 5.3　シミュレーション結果

本稿では，サービス数を 6 としてシミュレーションを行った．シミュレーション回数は各欠損率に対して 10000 回行った．また，推定が有効な状況を調べるために，様々な特性関数値の与え方を考察した．具体的には，先述の特性関数値の与え方の例で，$k_i$ の値を変化させることで様々なケースを調べた．

まず，各サービスの価値が一様である場合について調べた．このとき，特性関数値は以下の条件に従う．

**case1：** 各サービスの価値が一様である場合

**特性関数値設定：** $k_i = 1 \ (i = A, \ldots, ABCDEF)$

この場合の欠損率と誤差率の相関を，図 5.1 に示す．

図 5.1 では，特性関数値の欠損が大きいほどシャープレイ値の誤差が大きくなるということが言える．この結果は，推定の元となるデータが少ないほど，類似推定の精度は低下するという自明な相関関係を表している．誤差率は，欠損率が小さい間は線形的に増加するが，欠損率が大きくなってくると幾何級数的に増加する．このことから，欠損率が大きくなると，類似推定を用いるのは適切ではないと言える．

次に，原子サービスの半分は価値が大きく，半分は価値が小さいという場合について調べた．これは，価値の大きい原子サービスと価値が小さい原子サービスが同程度の割合で混在しているという状況を想定した．

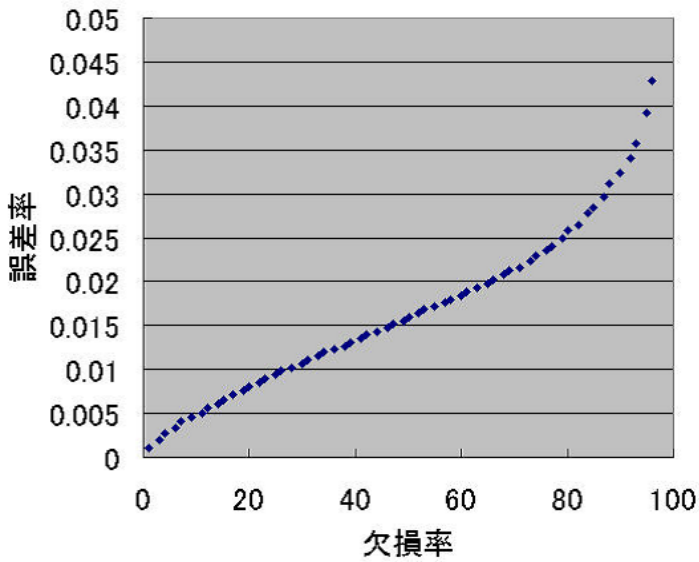



図 5.1: 各サービスの価値が一様である場合

**case2：** 価値の大きい原子サービスと価値の小さい原子サービスが同程度の割合で混在している場合

**特性関数値設定：**

$$k_i = \begin{cases} 1 & i = AB, AC, \ldots, ABCDEF \\ 1.8 & i = A, B, C \\ 0.2 & i = D, E, F \end{cases}$$

この場合の欠損率と誤差率の相関を，図 5.2 に示す．

比較のために，case1 のグラフも表示している．

case1 と比較して，全体的に誤差率が大きく，また，欠損率が小さいうちから誤差率が幾何級数的に増加している．case1 では誤差率が 80% 付近から幾何級数的に増加しているが，case2 では 60% 付近から幾何級数的に増加している．case2 では，欠損率が大きい場合には推定の適用は適切ではないと言える．これは，複合サービスが，価値の大きい原子サービスを含む提携のグループと，価値の小さい原子サービスを含む提携のグループの二つに分かれるため，欠損率が大きくなると，推定の元となる類似サービスが存在しない可能性が高まるためであると考えられる．

次に，一つの原子サービスの価値が大きく，他の原子サービスの価値が小さいという場合について調べた．これは，一つの価値の大きいサービスに，小さ

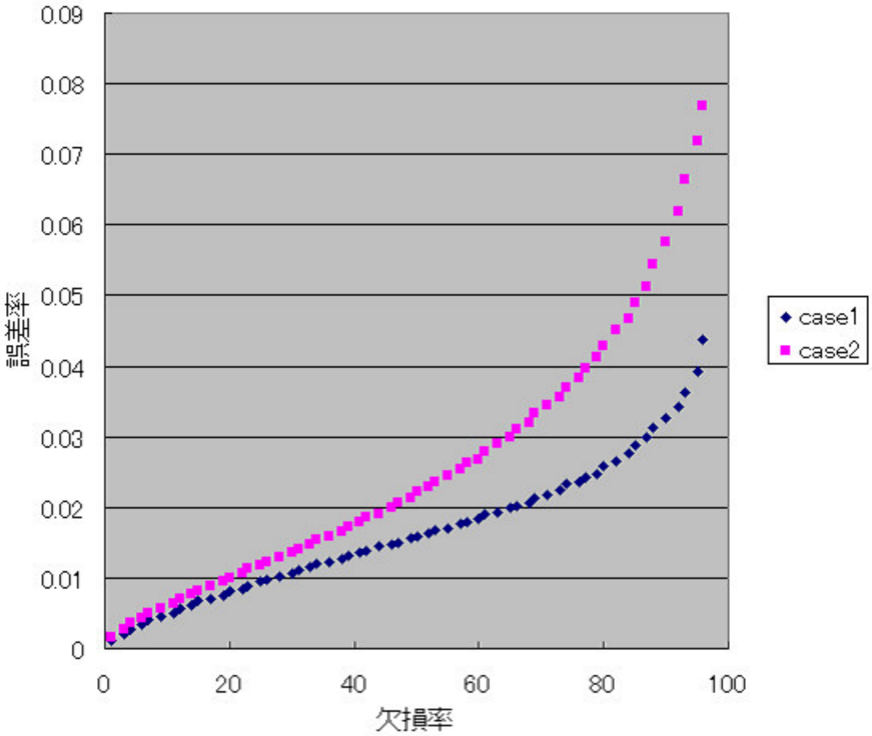



図 5.2: 価値の大きい原子サービスと小さい原子サービスが同程度の割合で混在している場合

いサービスがいくつか付属しているという状況を想定した．

**case3：** 一つの原子サービスの価値が大きく，他の原子サービスの価値が小さい場合

**特性関数値設定：**

$$k_i = \begin{cases} 1 & i = AB, AC, \ldots, ABCDEF \\ 1.8 & i = A \\ 0.2 & i = B, C, D, E, F \end{cases}$$

この場合の欠損率と誤差率の相関を，図 5.3 に示す．

case1 と比較して，全体的に誤差率が大きく，また，case2 と同様に欠損率が小さいうちから誤差率が幾何級数的に増加している．case3 では 65% 付近から幾何級数的に増加している．case3 でも，case2 の場合と同様に，欠損率が大きい場合には推定の適用は適切ではないと言える．しかし，case2 よりは誤差率は抑えられている．これは，case3 の場合も case2 の場合と同様に，価値の大きい

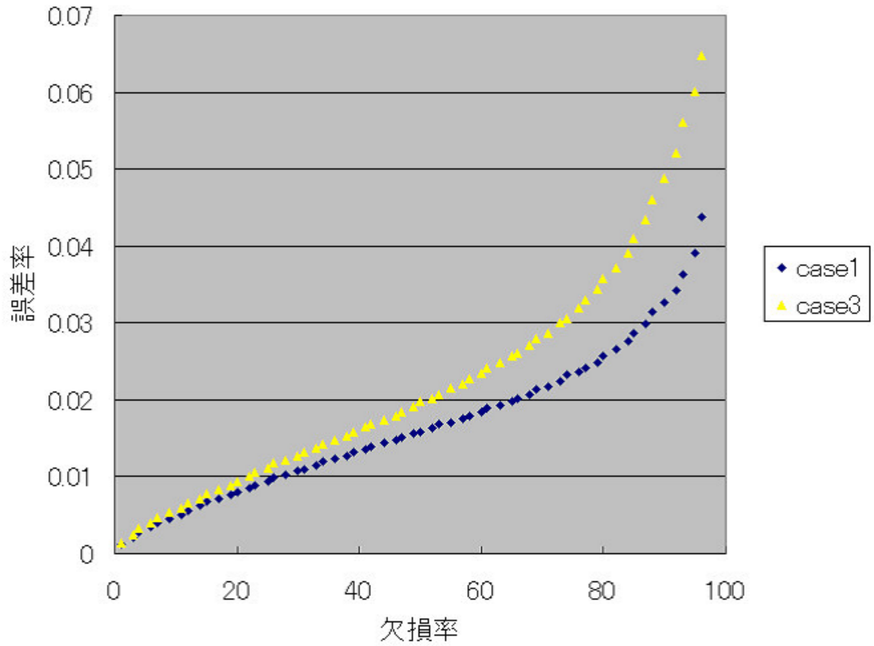



図 5.3: 一つの原子サービスの価値が大きく，他の原子サービスの価値が小さい場合

原子サービスを含む提携のグループと，価値の小さい原子サービスを含む提携のグループの二つに分かれるが，case3 では価値の小さいグループに 5 つの原子サービスが含まれ，推定の元となるデータが case2 の場合より多くなるからであると考えられる．

次に，一つの原子サービスの価値が小さく，他の原子サービスの価値が大きいという場合について調べた．これは，価値の大きいいくつかのサービスがあるところに，小さいサービスを一つ付属しているという状況を想定した．

**case4：** 一つの原子サービスの価値が小さく，他の原子サービスの価値が大きい場合

**特性関数値設定：**

$$k_i = \begin{cases} 1 & i = AB, AC, \ldots, ABCDEF \\ 0.2 & i = A \\ 1.8 & i = B, C, D, E, F \end{cases}$$

この場合の欠損率と誤差率の相関を，図 5.4 に示す．

case3 とほぼ同様のグラフとなった．これは，case4 では case3 の原子サービ

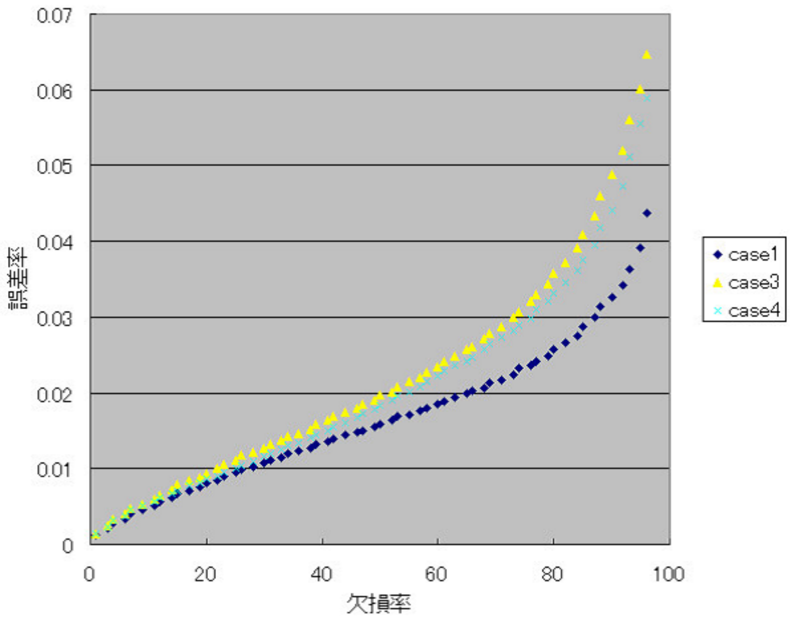



図 5.4: 一つの原子サービスの価値が小さく，他の原子サービスの価値が大きい場合

スの価値を逆にしただけであり，case3 では，価値の小さいグループに 5 つの原子サービスが含まれたのと同様に，case4 では価値の大きいグループに 5 つの原子サービスが含まれたので，推定の元となるデータが case3 と同程度存在し，推定の精度も同様のものとなったからであると考えられる．

次に，サービスを組み合わせることによって生じる付加価値が，原子サービスの価値に対して大きい場合について調べた．これは，単体では価値は大きくないが，組み合わせることによって大きな価値を生むサービスがある場合を想定した．

**case5：** 複合サービスの価値が原子サービスの価値と比較して大きい場合

**特性関数値設定：**

$$k_i = \begin{cases} 1 & i = A, B, C, D, E, F \\ 2 & i = AB, AC, \ldots, ABCDEF \end{cases}$$

この場合の欠損率と誤差率の相関を，図 5.5 に示す．

case1 とほぼ同様のグラフとなった．これは，case5 では case1 と同様に，各サービスの価値が一様であるとしているためである．サービスの組み合わせによって生じる付加価値が，原子サービスの価値に対して大きくなっても，推測

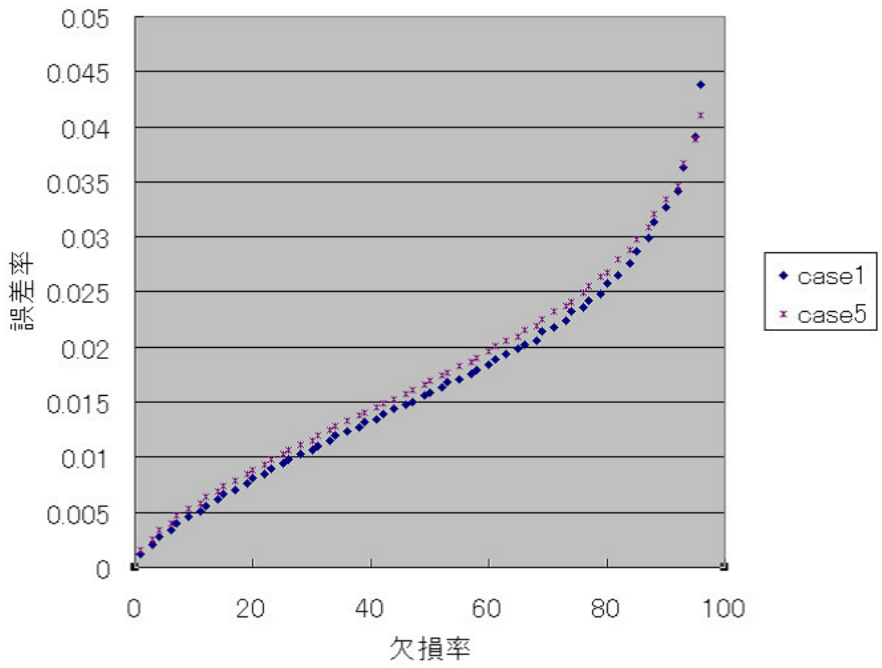



図 5.5: 複合サービスの価値が原子サービスの価値と比較して大きい場合

の精度に大きな変化はないと言える．

次に，サービスを組み合わせることによって生じる付加価値が，原子サービスの価値に対して小さい場合について調べた．これは，組み合わせても大きな付加価値は生じないサービスがある場合を想定した．

**case6：** 複合サービスの価値が原子サービスの価値と比較して小さい場合

**特性関数値設定：**

$$k_i = \begin{cases} 10 & i = A, B, C, D, E, F \\ 1 & i = AB, AC, \ldots, ABCDEF \end{cases}$$

この場合の欠損率と誤差率の相関を，図 5.6 に示す．

case1 の場合と比較して，誤差率は小さくなった．また，誤差率が幾何級数的に増加し始める欠損率は case1 と同様になった．この結果から，サービスを組み合わせてもわずかな付加価値しか生じない場合，複合サービスの価値は原子サービスの価値から推測することが比較的容易であるということが言える．

以上の結果を以下にまとめる．

1. 各サービスの価値が一様な場合，欠損率が 80% 程度までは，誤差率は欠損率に対して線形に増加し，また，欠損率が 60% 程度でも誤差率は 2% 程度

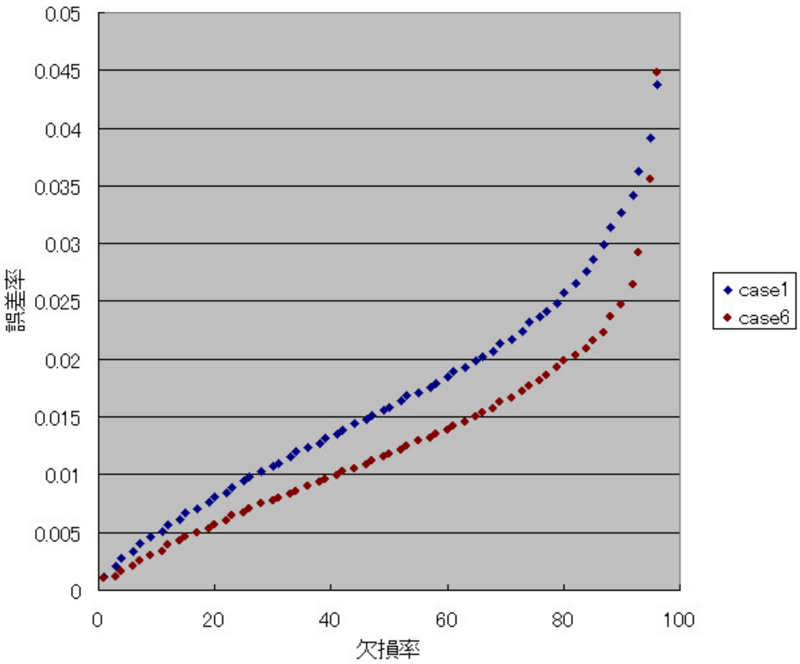



図 5.6: 複合サービスの価値が原子サービスの価値と比較して小さい場合

に抑えられる．

2. 価値の大きいサービスと価値の小さいサービスが混在している場合は，一様な場合に比べて推定の精度が低下し，欠損率が 60% あたりから誤差率が幾何級数的に増加するので，欠損率が大きい場合は推定は適さない．
3. サービスの組み合わせによって生じる付加価値が，原子サービスの価値に対して大きくなっても，推定の精度への影響は少ない．
4. サービスの組み合わせによって生じる付加価値が，原子サービスの価値に対して小さくなる場合，推定の精度は向上する．

# 第6章　終わりに

本研究では，複合 Web サービスにおける収益配分法に関して，以下のような問題について取り扱った．

**特性関数値の欠損**　シャープレイ値の計算では，全てのサービスの組み合わせに対して，利用者が持つ価値（特性関数値）が与えられることを前提にしている．しかし，Web サービスでは，サービスの組み合わせ数が非常に多くなることが想定され，すべての特性関数値が得られるとは限らない．特

性関数値に欠損がある場合，シャープレイ値を計算することができない．

**Web サービスにおける実装形態**　仮に利用者があるサービスの組み合わせに対する特性関数値を知っているとしても，まず，システムはその情報を獲得しなければならない．それをどのように実現するかは，明白ではない．

上記の課題を解決するため，特性関数値が欠損している場合に，その値を推定する方法を提案した．本稿では，単純推定と類似推定の 2 つの推定法を検討した．この方法の有効性を示すために，シミュレーションを行った．そこでは，特性関数値に関する代表的な場合に関して，特性関数値データ欠損の程度と配分計算に生じる誤差の関係を調べた．また，後者の課題に対しては，利用者がサービス検索をするときに，同時にその価値を入力させることで，特性関数値のデータを収集する方法を検討した．また，上記の類似推定においては，サービス間の類似度の情報を得ることが必要である．これに対しては，サービスプロファイルの情報から計算することを検討した．

本研究の貢献は以下の 2 点となる．

**特性関数値に欠損がある場合のシャープレイ値の計算法の提案**　特性関数値に欠損がある場合でも，適切に推定を行うことで，シャープレイ値の計算を可能とした．特性関数値の欠損の割合が 6 割程度になっても，配分額計算の誤差は 2%に収まることをシミュレーションにより確認した．

**Web サービスにおける実装形態の提案**　利用者の Web サービス検索時に特性関数値が得られる可能性があること，また，サービス間の類似度を Web サービスのプロファイル情報から得られる可能性があることを確認した．

上記二点の貢献により，複合 Web サービスの収益配分において，推定を用いた配分決定の有用性と，適用可能範囲を示すことが出来た．

今後の課題としては，本稿では類似推定における類似度を 0 と 1 の 2 値で与えたが，類似度を実数値で与え，類似度が最大となるものを元に推定を行いというより厳密な推定法の考案がある．

# 謝辞

本研究を行うにあたり，熱心なご指導，ご助言を賜りました松原繁夫准教授に厚くお礼申し上げます．また，日頃より，有益な御助言を与えてくださりました石田亨教授に心から感謝致します． そして，日頃から様々な御助言，御協

力を頂きました石田・松原研究室の皆様に感謝致します.

# 参考文献


[1] 社団法人電子情報通信学会: Web サービスコンピューティング, 電子情報通信学会 (2005).

[2] Ishida, T.: Language Grid: An Infrastructure for Intercultural Collaboration, *Proceedings of IEEE/IPSJ Symposium on Applications and the Internet (SAINT-06)*, pp. 96–100 (2006). keynote address.

[3] Shapley, L. S.: A Value for n-person Games, *Annals of Mathematical Studies*, Vol. 28, pp. 307–317 (1953). In Contributions to the Theory of Games, volume II by by H.W. Kuhn and A.W. Tucker, editors.

[4] Feigenbaum, J., Papadimitriou, C. H. and Shenker, S.: Sharing the Cost of Muliticast Transmissions (preliminary version), *Proceedings of the Thirty-second Annual ACM Symposium on Theory of Computing (STOC 2000)*, pp. 218–227 (2000).

[5] Gilad Zlotkin, J. S. R.: Coalition, Cryptography, and Stability: Mechanisms for Coalition Formation in Task Oriented Domains, *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pp. 432–437 (1994).

[6] Fatima, S. S., Wooldridge, M. and Jennings, N. R.: A Linear Approximation Method for the Shapley Value, *Artificial Intelligence*, Vol. 172, No. 14, pp. 1673–1699 (2008).

[7] Ben Hassine, A., Matsubara, S. and Ishida, T.: A Constraint-based Approach to Horizontal Web Service Composition, *Proceedings of the Fifh International Semantic Web Conference (ISWC2006)*, pp. 130–143 (2006).