情報処理学会論文誌 Vol. 52 No. 10 2920–2933 (Oct. 2011)

同種サービスの併用を考慮した効率的なサービス選択

平塚信明[†1] 石川冬樹[†2] 本位田真一[†1,†2]

近年，同種の機能を持つ Web サービスが増加することを想定し，使用するサービスをユーザがアプリケーションに課した非機能要求を用いて選択する研究が進められている．ここで，複数の同種サービスを組み合わせることで単体のサービスでは実現不可能な非機能要求を得ることが考えられるが，可能なサービスの組合せの数が膨大になるため，サービス選択のための計算コストが非常に大きくなる．そこで本研究ではこの問題に対し，同種サービスの併用を考慮してのサービス選択における計算コストを削減するための手法を提案する．この手法においては，最適となりえない組合せや，他の組合せと類似している組合せを候補から外しながら組合せを行っていくことで，考慮する組合せを少なくしていく．

Efficient Service Selection with Combinational Use of Functionally-equivalent Services

Nobuaki Hiratsuka,[†1] Fuyuki Ishikawa[†2] and Shinichi Honiden[†1,†2]

Recently, due to the increase of the functionally-equivalent services, many studies address service selection problem based on nonfunctional or quality aspects. To compose application of higher quality by combining services, combinational use of the functionally-equivalent services could be considered. However, when such combinational use is introduced, the computational cost for the servcice selection becomes very large. In this work, we propose a set of methods that decrease the cost for the service selection considering combinational use. These methods decrease the number of combinations to examined by eliminating ones that cannot be optimal and ones similar to others.

†1 東京大学
The University of Tokyo
†2 国立情報学研究所
National Institute of Informatics

1. はじめに

Web サービスとは，インターネットを経由し，疎結合にシステムと接続するコンポーネントである．インターネット上の既存のサービスを使用することで，ユーザは容易，迅速にアプリケーションを作成することができる．近年，同一の機能を持つサービスがインターネット上に複数存在することを想定し，機能要求に合致した複数のサービスに対して，文献 1)–3) などでは非機能要求を用いて使用するサービスを選択する研究がなされている．非機能要求は QoS（Quality of Service）とも呼ばれ，料金，応答時間，可用性，スループット，セキュリティなどがある．これらは Web サービスの提供者とユーザ間の SLA（Service Level Agreement）や第 3 者機関による計測，評価によって定められた値である．このように機能要求を満たす複数のサービスが存在する環境を想定する背景として，以下のようなものがあげられる．

- Web サービスの増加
- 多段階の SLA
- 機能分割による要求実現の研究

ここ数年間，インターネット上で公開されている Web サービスは増加の傾向にある．文献 4) では 2007 年 10 月における Web サービスの数は 5,077 個だが，文献 5) では 2010 年 8 月には 28,558 個に増加している．この傾向から，同種の機能を持つサービスも増加することが考えられる．また，SLA の形態として，多段階にレベル分けした契約が考えられる（ゴールド，シルバー，ブロンズなど）．これは同一のサービスが異なる QoS で提供されていると見なすことができる．よってサービス選択時に契約レベルの決定も行う場合，複数の同種のサービスからの選択と考えることができる．さらに文献 6) などにおいて，要求される機能を単体ではなく複数の異なる機能を持ったサービスによって実現する研究が進められている．この研究が発展すれば，機能要求を満たすサービスがより多く認識できる環境になり，そして異なる QoS で同種の機能を満たすサービスが多く存在することになると考えられる．

本研究では，一般的な QoS に基づくサービス選択に対し，各タスクにおいて複数の同種サービスを併用することで以下の実現を考えている．

- アプリケーションの QoS の向上
- 複数のサービスをともなう要求の実現

まず，併用によって単体のサービス利用よりもよりユーザの非機能要求に合致したサービ

スの利用を考えている．想定するシナリオとして，たとえばゴールドの SLA で契約した高料金，高可用性のサービスの単体利用ではなく，ブロンズの SLA で契約した 3 つのサービスを同時利用で低料金，高可用性のアプリケーションを実現することがあげられる．次に，複数のサービス利用を各タスクへの要求として加えることで，単体の利用では実現しなかったサービス利用を実現することを考えている．たとえば，サービスの出力値の信頼を確保するために，3 つの同種のサービスを同時に呼び出し，そのうち 2 つのサービスの出力結果が一致した際に次のタスクに移るという利用方法が考えられる．

QoS に基づいてサービスを選択する際，選択に要する計算コストを小さくすることが求められる．これは，最適なサービスを選択するためにはアプリケーションの実行直前の選択が望まれるが，その計算コストが大きいとアプリケーションの実行時間に影響を及ぼしてしまうためである．サービスの併用を考慮することで，サービス選択時の計算コストは大きくなる．これは，いくつの，どの複数のサービスを，どのように併用するのか，といった検討すべき併用の状態の数が指数的に増加してしまうためである．また，得られた膨大な併用状態を QoS に基づく選択においてそのまま 2.1 節で述べる MMKP（Multi-choise Multi-demensionalKnapsack Problem）の要素として用いると，これも大きな計算コストを要することになる．本研究ではこの問題に対し，有効な併用のみをサービス選択時に検討し，さらに MMKP の要素を選別することで，併用を考慮した際のサービス選択にかかる計算コストを小さくすることを目的とする．

以下に，本論文の構成について述べる．2 章では本研究の動機として本研究で対象とする問題の具体例と関連研究，3 章ではこの問題を定式化したモデルについて説明する．4 章では本研究における併用の詳細について説明する．5 章では本研究の提案手法である同種サービスの併用を考慮した効率的なサービス選択手法を説明し，6 章ではその評価を行う．最後に，7 章で結論を述べる．

2. 動　　機

この章では，本研究の動機として 2.1 節において既存の問題である QoS に基づくサービス選択，2.2 節において本研究で対象とする問題について述べる．そして 2.3 節で本研究の関連研究について説明する．なお，本研究で用いる QoS に基づくサービス選択のモデルの詳細に関しては 3 章で説明する．

2.1 QoS に基づくサービス選択

QoS に基づいた一般的なサービス選択方法の説明に，**図 1** のような Web サービスを使用した商品購入のアプリケーションの例を用いる．これは言語や地域の問題を気にせずに目的とする商品を容易に購入できるアプリケーションを想定している．このアプリケーションは 2 つのユーザの入力と 5 つの Web サービスの利用を順次実行する形で構成される．図 1 において，前者を楕円，後者を四角で表している．このアプリケーションの詳細は以下のようになる．まず，ユーザが欲しい商品名，購入における上限金額，届け先を入力する．そしてサービスによって世界各国の為替相場に基づいて金額を変換，商品名を各国の言語に変換し，商品検索サービスによって上限金額以下の商品を検索する．次に配送検索サービスによって配送方法を決定し，商品を届け先に送る際の料金を検索した商品の価格に上乗せして商品の詳細とともにリストにしてユーザに提示する．最後にユーザが提示された商品の中から購入を選択すると，サービスによってクレジットカードによる決済が行われる．各タスクで使用するサービスには入出力情報などの機能要求，アプリケーション全体には QoS に関する要求が付加され，これを満たすように使用するサービスが決定される．ここでの QoS の要求は想定するシナリオによって異なる．たとえば，このアプリケーションを他の一般のユーザに公開して使用料を得る場合，ユーザからの満足を得るためには 2 つの入力間の応答時間を一定以内に抑えることが求められる．一方，このアプリケーションを個人的に使用する際には，多少の時間がかかってもサービスの利用によってかかる料金を抑えることが優先的に求められる．

図 1　商品購入のアプリケーション
Fig. 1　An application of merchandise purchase.

サービスの選択は機能要求，非機能要求の順に実行される．まず，システムが為替変換などのタスクごとに記述された機能要求を満たすサービス群を検索する．そしてこれらのサービス群の中からアプリケーション全体の QoS が要求を満足するように各タスクで使用するサービスを選択する．QoS に基づいたサービス選択は，アプリケーション全体が満たす QoS の制約条件内（使用料金が 10 円以内など）で各 QoS 項目への優先順位を反映した効用の値を最大化する方法が文献 1)–3) などで行われている．これは NP 困難な問題である MMKP（Multi-choise Multi-demensional Knapsack Problem）に帰着され，様々な手法を用いることでサービス選択における計算コストを小さくする研究が進められている（2.3 節）．小さい計算コストが求められる理由は，サービス選択の計算時間がアプリケーションの実行時間に与える影響を少なくするためである．サービスの選択は，サービスの QoS の更新や

新たなサービスの公開，利用する時間帯などで SLA で保証する QoS が変化するという状況が想定されるため，アプリケーションの実行直前に行われることが望まれる．そのため，先述の応答時間を一定内に抑えるシナリオなどを想定すると，実行直前のサービス選択における MMKP を効率的に解くことが実行時間への影響を抑えるために必要となる．

2.2　本研究で対象とする問題

本研究で対象とする問題について，2.1 節で述べた商品購入のアプリケーションの例を用いて説明する．本研究では，タスクごとに使用するサービスを 1 つずつ選択するのではなく，併用を考慮してタスクごとに複数のサービスを選択することで 1 章で述べたように以下の内容の実現を考えている．

- アプリケーションの QoS の向上
- 複数のサービスをともなう要求の実現

第 1 の点として，各タスクへにおいて単一ではなく複数のサービスを用いることで，ユーザの要求する QoS の要求により合致したアプリケーションを作成することを考える．たとえば，2.1 節の例において配送検索の機能要求を満たすサービスが普及し始めで開発段階のものが多く，信頼性が低い状況を想定する．このとき，既存研究のように「配送検索」のタスクに 1 つのサービスしか選択しないと，高い信頼性を要求されるアプリケーションを作成することができない．よって複数のサービスを同時に呼び出す，または障害時に用いるバックアップとして用意しておく，などの方法で信頼性を向上させることが考えられる．また，仮に高い信頼性を持つが高い使用料金を要求するサービスがあったとしても，それなりの信頼性を持つ安めのサービスを複数利用することによって，低料金で同等の高い信頼性が実現できる可能性がある．こういった利用に際しては，いくつ，どのサービスをどのような形で利用するかを適切に決定する必要がある．

第 2 の点として，単一のサービス利用で生じる問題を想定し，あらかじめ特定のタスクに対して複数のサービス利用の要求を付加した形で 2.1 節の QoS に基づくサービス選択を行うことを考える．たとえば，2.1 節の例において翻訳サービスの結果に誤訳があることは避けられないため，複数の翻訳サービスを 1 度に呼び出しそれらを人間に提示して適宜正しいものを選びながら統合してもらう，といった要求が考えられる．他の例としては，多様性による信頼性向上やロックイン回避のため複数提供者によるサービスを併用することがあげられる．また，ランキングとして書籍や Web サイトなどの推薦を行うサービスのように，唯一の正解がなく各サービスの提供する観点が異なる際に，それらを統合した結果を組み立てるという場合もある．こういった利用に際しては，特定タスクに対して一定数のサービスの併用要求を付加した状況下で，どのサービスを選択するかを適切に決定する必要がある．

2.3　関 連 研 究

近年，QoS に基づくサービス選択の研究が多くなされている．機能要求を満たす複数の Web サービスの中から QoS の条件に沿ったものを効率的に選択するために，文献 7) や 8) では線形計画問題，文献 9) では合成モデルとグラフモデルを用いたヒューリスティックな手法によってアプリケーション内のサービスを用いるタスクの数の影響を抑えて少ない計算コストでサービスの選択をする研究がなされた．また，これに加えて 2.1 節で述べた MMKP の要素である同種のサービスの数の影響を抑えて少ない計算コストを達成する研究もなされ，文献 2) や 3) では QoS の値に応じて同種のサービスを集合に分割して選択する手法が提案され，文献 1) ではスカイラインサービスによって有効なサービスを選別する手法が提案された（3.5 節）．本研究では文献 1) で提案された手法が QoS への多様な要求に応じて最適なサービスを効率的に選択できることから，この手法を採用し，これに複数の同種サービスによる併用の概念を加えている．

QoS の向上のために複数の同種サービスを併用する研究もいくつかなされている．文献 10) では分散コンピューティングにおける計算資源の併用の研究に基づき，サービスの併用に関して論じられている．文献 11) ではサービスを選択した後，制約条件として指定した QoS との余剰を用いてバックアップのサービスを選択する手法が提案されている．文献 12) では本研究でも紹介する様々な併用手法を考慮しながら，最適なサービスの組合せを少ない計算コストで選択する手法が提案されている．しかしこれら既存研究は，現実的でない強い仮定を前提とした手法となっている．具体的には，文献 11) では可用性に限定して議論を行っており，料金，応答時間など多様な（多次元の）QoS 項目を扱えない．文献 12) では同種の機能を持つサービスの QoS の値を一定値と仮定しており，「安いが遅い」といった様々な特徴を持つ提供者が存在する際，特に Web 上のサービスを対象とする際に不適切である．本研究ではこれに対し，多様な値のあらゆる QoS 項目を持つ同種サービスが存在するより現実的な環境を考慮し，ユーザの嗜好に合致したサービスを少ない計算コストで選択する手法を提案している．

また，本研究で取り上げる併用に含まれる 1 つの方式として，アプリケーション実行中の同種サービスの再呼び出しがある．これは使用中のサービスから応答がない場合に，他の同種サービスをバックアップとして呼び出すものである．本研究のように QoS を考慮したものとして，文献 13) があげられる．文献 13) は QoS の制約条件を満たすように再呼び出しを行う研究である．ここで，文献 13) は実行中に再呼び出し先を決めるが，本研究では実

行前に決めるという違いがある．本研究では実行前にあらかじめサービスの故障を考慮したサービスの利用計画を立てることで，実行中の再呼び出しよりも高い確率で QoS の制約条件を満たすことを考えている．端的な例として，競合と比べて他の QoS は同等だが格段に安いサービスが 1 つだけある場合を考えてみる．故障を想定しなければ，このサービスを選び，他の部分（他の機能を提供するサービス）により多くの金額を割り当てるようなサービス選択が最適かもしれない．しかし実行時にそのサービスが利用不可能になった際に，（他の部分に多くの金額を使ってしまった後では）代替サービスでは料金制約をオーバしてしまうかもしれない．これは，局所的な情報に基づきよさそうな解を絞り込む貪欲アルゴリズムの結果が最適とは限らないことと同様である．提案手法はこれに対し，故障があった場合の期待値まであらかじめ考慮，探索する．なお，文献 13) と比べて，他の併用方式の利用も考慮する点でも本研究はより一般的な手法となっている．

3. サービス選択モデル

この章では，本研究が解く問題を定式化したモデルについて説明する．3.1〜3.4 節では QoS に基づくサービス選択に用いる設定と詳細，3.5 節では本研究で使用する既存研究の技術について述べ，3.6 節で本研究が対象とする問題について説明する．これらは既存研究[1)–3)]における問題設定を参考にしている．

3.1 QoS

QoS には，料金，応答時間，可用性，スループット，セキュリティなどの項目が含まれる．本研究では，これらを数値評価したものを扱う．QoS は個々のサービスだけでなく，それらを組み合わせて形成されるアプリケーション全体に対しても定義される．これは使用するサービスの QoS から算出され，QoS 項目ごとに計算式が異なる．例として，あるアプリケーションでサービスが n 個連続して実行される場合，使用する各サービスの QoS 項目の値を $q(s_j)$，アプリケーション全体の QoS を $q(CS)$ とすると，その計算方法は**表 1** のように計算される．

3.2 QoS 制 約

QoS 制約とは，ユーザがアプリケーションに要求する必ず満足すべき QoS の条件である．例として，料金が 20 円以下，可用性が 98%以上といったものが設定される．3.1 節で示したアプリケーション全体の QoS の値が QoS 制約を満足するように，サービスが選択される．本研究では同種サービスの併用を考慮するにあたり，以下の 2 つの QoS 制約を定め，想定するシナリオに応じて使い分けるものとする．

表 1　アプリケーション全体の QoS 算出方法
Table 1　Calculation of QoS in an application.

QoS 項目	算出方法	計算式
料金，応答時間	総和	$q'(CS) = \sum_{i=1}^{n} q(s_i)$
可用性，信頼性	総積	$q'(CS) = \prod_{i=1}^{n} q(s_i)$
評価値	平均値	$q'(CS) = \frac{1}{n} \sum_{i=1}^{n} q(s_i)$
スループット	最小値	$q'(CS) = min_{i=1}^{n} q(s_i)$

- 強い QoS 制約
- 弱い QoS 制約

強い QoS 制約とは，サービス併用を考慮したアプリケーションの実行結果が必ず満足することを想定して課された制約である．たとえば，バックアップとしてサービスの失敗時に再呼び出しをする併用を考慮する際，再呼び出しされたサービスによって生じる料金などの QoS も含めて満足すべき制約を指す．つまり併用によって生じる QoS の最悪値が満たすことを考慮した制約である．弱い QoS 制約とは，サービス併用を考慮したアプリケーションの実行結果が平均的に満足することを想定して課された制約である．先述のバックアップによる併用において，バックアップのサービスは確率的に呼び出しを受ける．弱い QoS 制約は，この確率から併用によって生じる QoS を期待値で算出し，それが満足すべき制約を指す．よって必ずしもアプリケーションの実行結果がつねにこの制約を満たすとは限らない．

3.3 効　　用

効用とは，QoS 制約を満足するサービスの組合せが複数ある場合，その中から最適なものを選択する際に使用される値である．ユーザの各 QoS 項目に対する嗜好を反映し，高可用性よりも低料金を重視する組合せを選択するといったことを可能にする．効用の算出には，ユーザの嗜好を表現した各 QoS 項目に対する重みと，正規化された各サービスの QoS の値を使用する．重みは式 (1) のように定められる．r 個の QoS 項目がある場合，その合計値が 1 になるように 0〜1 の値の範囲で優先するものが大きくなるように各項目について定める．サービスの効用は式 (2)，(3)，(4) で求められる．r 個の QoS 項目を値が大きいことが望まれる QoS 項目 r_1 個，値が小さいことが望まれる QoS 項目 r_2 個に分割し，それぞれにおける効用の値（式 (2)，式 (3)）の合計値をサービスの効用の値とする（式 (4)）．式 (2)，(3) では，各タスクで選択候補となるサービスすべての各 QoS 項目の値の最大値（$Qmax$）や最小値（$Qmin$）を使用してサービスの QoS の値を正規化し，重みをかけたものの総和

を使用している．各タスクで使用するサービスの効用の合計値をアプリケーションの効用の値とし，この値が大きいほどユーザの嗜好を反映したものとする．なお，本研究では併用によって選択候補のサービスの QoS の最大値や最小値が大きく変動するため，これらの値をシナリオによって固定値にする場合がある．

$$\sum_{k=1}^{r} w_k = 1 \tag{1}$$

$$U(s+) = \sum_{k=1}^{r_1} \frac{Qmax(k) - q_k}{Qmax(k) - Qmin(k)} w_k \tag{2}$$

$$U(s-) = \sum_{k=1}^{r_2} \frac{q_k - Qmin(k)}{Qmax(k) - Qmin(k)} w_k \tag{3}$$

$$U(s) = U(s+) + U(s-) \tag{4}$$

3.4　QoS に基づくサービス選択

サービスの単体利用時の一般的な QoS に基づくサービス選択手法は以下のようになる．まず前提として，ユーザによってアプリケーション全体における QoS 制約と各 QoS 項目に対する重みが設定される．そのうえで，機能要求の合致した同種のサービスの中から，QoS 制約を満たして効用の値を最大化するサービスの組合せを選択する．本研究では，サービスの併用を考慮した QoS に基づくサービス選択においても，同様の方式で選択を行うものとする．

3.5　スカイラインサービス

スカイラインサービスとは，同種の機能を持つサービスの中で QoS 項目の値が他のサービスよりも少なくとも 1 つ優れているサービスを指す．QoS に基づくサービス選択ではスカイラインサービス以外が選択候補になることはなく，どのような QoS 制約や効用の算出の設定をしてもスカイラインサービスの中から最適なサービスが選択される．よってスカイラインサービスを選択候補にすることで，選択の最適性を保持したまま，2.1 節で述べた MMKP の要素を減らし，計算コストを下げることができる[1)]．

スカイラインサービスの例として，**表 2** のようにあるタスクの機能を満たすサービスが 4 つあると考える．低料金，短い応答時間，高可用性を持ったサービスの選択が望まれるが，Service4 はどのような QoS 制約や重みを課された状態でも選択候補になりえない．なぜなら Service1 が Service4 に対して，料金，応答時間，可用性のすべてにおいて優れているからである．一方，Service4 以外のサービスは他のサービスよりもいずれかの QoS 項目において優れているため選択候補となる．これをスカイラインサービスという．

表 2　スカイラインサービスの例
Table 2　Example of skyline services.

サービス名	料金	応答時間	可用性
Service1	$1.5	8 msec	96%
Service2	$1.2	10 msec	96%
Service3	$1.4	12 msec	98%
Service4	$1.6	10 msec	94%

先述したように，スカイラインサービスを選択候補とすることで，MMKP における計算コストを減らすことができる．スカイラインサービスの作成に要する計算コストは小さいため，この手法はとても有効である．本研究では，この有効性に着目し，このスカイラインサービスの手法を利用する．

3.6　本研究で対象とする問題

本研究では，3.1〜3.4 節で述べた QoS に基づくサービス選択問題に対し，同種サービスの併用の概念を加える．この際，2 章で述べたように，膨大な併用状態の検討と MMKP の要素の増加によって計算コストが非常に大きくなるという問題がある．この問題を解決するため，本研究では 3.5 節で述べたスカイラインサービスの概念を用いた手法を提案する．本研究では，4 章において併用の詳細を説明した後，5 章でこの問題に対する提案手法について述べる．

4.　サービスの併用

この章では，同種のサービスの併用について，本研究が対象とする構成とその QoS 算出方法について述べる（4.1，4.2 節）．また，併用状態を単体サービスと見なす仮想サービスについて説明する（4.3 節）．

4.1　構　　成

本研究では，同種のサービスの併用の構成として以下のものを考える．

- 並列呼び出し
- 直列呼び出し
- 上記の呼び出しの組合せ

並列呼び出しは，同時に複数のサービスを呼び出す構成である．直列呼び出しは，あるサービスを呼び出し，タイムアウト値の時間までに出力値が返ってこない場合，他のサービ

表 3　強い制約での並列呼び出しの QoS 計算
Table 3　Calculation of a parallel invocation in a strong constraint.

QoS 項目	計算式
料金	$c_v = c_1 + c_2$
応答時間	$t_v = t_2$
可用性	$a_v = 1 - (1 - a_1)(1 - a_2)$

表 4　強い制約での直列呼び出しの QoS 計算
Table 4　Calculation of a sequence invocation in a strong constraint.

QoS 項目	計算式
料金	$c_v = c_1 + c_2$
応答時間	$t_v = t_1 + t_2$
可用性	$a_v = 1 - (1 - a_1)(1 - a_2)$

スを呼び出す構成である．上記の呼び出しの組合せとは，並列と直列の呼び出しを組み合わせたものである．たとえば直列呼び出しの最初の呼び出すサービスに並列呼び出しをした複数サービスを使用するといった構成を指す．

併用の目的には，1 章で述べたように，アプリケーションの QoS の向上と複数利用の要求実現の 2 つがある．アプリケーションの QoS 向上を目的にした併用の場合，本研究では上記の各構成を自動的に決定することを想定している．これは，たとえば短い計算時間，高可用性が求められる場合には並列呼び出しをして最も早い出力を使用する，低料金，高可用性が求められる場合には直列呼び出しをする，といったことを自動的に決定することを指す．一方，複数利用の要求付加に関しては，シナリオに応じてあらかじめ併用の構成がタスクごとに決定されていることを想定している．

4.2　QoS 算出方法

サービスの併用状態における QoS 算出方法は，その構成によって異なる．また，3.2 節で述べたように，弱い QoS 制約と強い QoS 制約のどちらを用いるかによって算出方法が異なる．強い制約下では，最悪値を算出して併用状態の QoS とする．弱い制約下では，期待値を算出して併用状態の QoS とする．例として，2 つのサービスによって構成される強い制約下，弱い制約下における並列呼び出しと直列呼び出しの併用状態の QoS 算出方法を**表 3**，**表 4**，**表 5**，**表 6** にあげる．各 QoS 項目において，添字 1，2 は 2 つのサービスのそれぞれにおける QoS 項目の値を表し，添字 v は併用状態の値を表すものとする．たとえば料金については 2 つのサービスの料金をそれぞれ c_1，c_2 とし，併用状態の料金を c_v としている．応答時間 t，可用性 a についても同様に添字を用いる．ただし応答時間については $t_1 < t_2$ と仮定する．

表 5　弱い制約での並列呼び出しの QoS 計算
Table 5　Calculation of a parallel invocation in a weak constraint.

QoS 項目	計算式
料金	$c_v = c_1 + c_2$
応答時間	$t_v = t_1 \left(\frac{a_1}{a_v}\right) + t_2 \left(\frac{a_v - a_1}{a_v}\right)$
可用性	$a_v = 1 - (1 - a_1)(1 - a_2)$

表 6　弱い制約での直列呼び出しの QoS 計算
Table 6　Calculation of a sequence invocation in a weak constraint.

QoS 項目	計算式
料金	$c_v = c_1 \left(\frac{a_1}{a_v}\right) + (c_1 + c_2) \left(\frac{a_v - a_1}{a_v}\right)$
応答時間	$t_v = t_1 \left(\frac{a_1}{a_v}\right) + (t_1 + t_2) \left(\frac{a_v - a_1}{a_v}\right)$
可用性	$a_v = 1 - (1 - a_1)(1 - a_2)$

4.3　仮想サービス

同種のサービスの併用によって生じる QoS の値を仮想の単体サービスの QoS と見なし，この際のサービスを仮想サービスとする．これは，3 章で述べた QoS に基づく単体サービスの選択手法における MMKP の要素に併用状態を加えることを目的とする．これにより，QoS 制約下で効用の値を最大化するサービスを選択する際に，実在の単体サービスであるか併用状態により構成される仮想サービスであるかを問わず，同じ方法でいずれが最適かを判別することができる．

5.　併用を考慮した効率的なサービス選択手法

本研究の目的は，併用を考慮しながら QoS に基づくサービス選択における計算コストを減らすことである．これには併用による効果を得ながら，併用状態を検討する際の膨大な組合せの考慮を避けるとともに，MMKP の要素として加わる併用状態の数を抑えることが必要である．この章では，これらを達成する方法として 2 つの手法を提案する．まず 5.1 節においてこれらの手法の位置づけを説明し，5.2 節，5.3 節でそれぞれの手法の詳細について述べる．

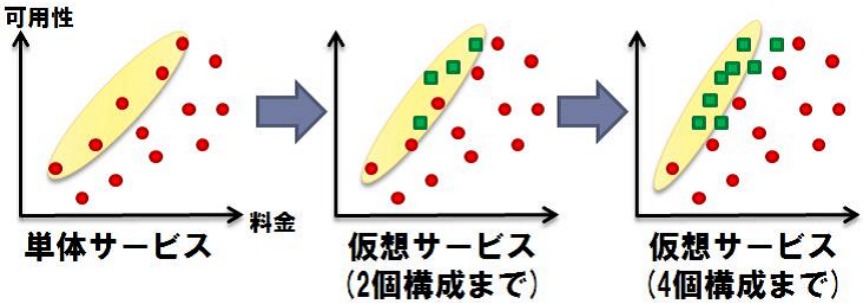

図 2　スカイラインサービスの逐次更新
Fig. 2　Gradual update of skyline services.

5.1　手 法 概 要

仮想サービスを考慮した効率的なスカイラインサービスの作成手法として，以下の 2 つの手法を提案する．

- スカイラインの逐次更新
- グルーピングによる組合せ削減

スカイラインの逐次更新は，併用状態の検討時に有効性の高いサービスの組合せのみを考慮し，さらに多くのサービスから構成される併用状態の検討には段階的な選別を加えることで，検討時の計算コストと MMKP を解く際の計算コストを抑えるものである．これには 3.5 節で述べたスカイラインサービスと 4.3 節で述べた仮想サービスの概念を利用する．グルーピングによる組合せ削減は，類似のサービスを 2 個の代表に抽象化し，スカイラインの逐次更新時の組合せの対象となるサービスの数を減少させる手法である．このとき，類似のサービスと判断する際に用いる設定値によって，スカイラインの逐次更新時に抑える計算コストと併用によって得られる効果のトレードオフをとることができる．

5.2　スカイラインの逐次更新

スカイラインの逐次更新とは，2 個のスカイラインサービスのみから仮想サービスを作成してスカイラインサービスを更新していく方法である．この際，4.1 節で述べた構成をすべて検討して仮想サービスを作成する．また，組合せの対象とするスカイラインサービスには実在サービスに加えて仮想サービスも考慮する．よって n 回の併用検討の試行で最大 2^n 個の実在サービスで構成される仮想サービスが作成される．

例として，考慮する QoS 項目が料金と可用性の 2 つのとき，同種の機能を持つサービスにおいて**図 2** のようにスカイラインサービスが更新される．ここで丸は実在サービス，正方形は作成された仮想サービスを表す．楕円で囲まれるサービス群がスカイラインサービスである．最大 4 個の実在サービスで構成される仮想サービスを作成する際，まずスカイラインサービスである実在サービスどうしの併用を考慮して構成数が 2 個の仮想サービスを作成する．そして作成した仮想サービスと実在サービスからスカイラインサービスを取得し，それらを組み合わせて構成数 3，4 個の仮想サービスを作成する．このように 2 個のスカイラインサービスから仮想サービスを作成していき，あらかじめ設定された構成数まで仮想サービスを作成した後，実在サービスと仮想サービスからスカイラインサービスをとり，これを MMKP の要素として用いる．

本手法は単純に 1 度に 3 個以上の併用を考慮する手法と比較して，少ない計算コストで有効な仮想サービスを効率的に作成できる．これには 2 つの理由がある．まず 1 つ目の理由として，スカイラインサービスのみを併用の構成対象とすることで，有効性の低いサービスを用いた併用の検討をしていない．2 つ目の理由として，2 つのスカイラインサービスの併用を繰り返す逐次更新により，途中段階で有効性の低い仮想サービスを併用の対象から外すことができる．たとえば，4 個の実在サービスで構成する仮想サービスを作成する際に，2 個で構成される仮想サービスの中でスカイラインサービスではない有効性の低いものを併用対象から外している．これらの理由から，本手法によって検討される仮想サービスは少なくなり，計算コストを下げることができる．

本手法は，スカイラインサービスのみを併用の対象とすることで計算コストを下げる代わりに，すべての有効な併用状態の検討は行っていない．これは，スカイラインサービス以外のサービスを併用対象として作成した仮想サービスが有効なものとなる可能性が低いためである．この有効性の評価に関しては付録の A.1 章で詳細を説明する．

なお，1 度に 3 個以上の併用によって作成した仮想サービスの構成は，2 個のサービスの併用の繰返しで表現でき，4 章で用いた式を用いることで QoS の値もこれらの間で差が生まれることはない．これらの証明は省略する．

5.3　グルーピングによる組合せ削減

グルーピングとは，近い QoS の値を持つサービス群を 2 個の代表サービスに抽象化する手法である．これは各項目の QoS の値が近いサービスからは，他のサービスとの併用によって近い QoS の値を持つ仮想サービスができることに基づく．逐次更新時のスカイラインサービスにこれを用いることで，併用の対象となるサービスの数を減らすことができる．そして検討する併用状態の数を減らすことで，この際の計算コストを下げることができる．

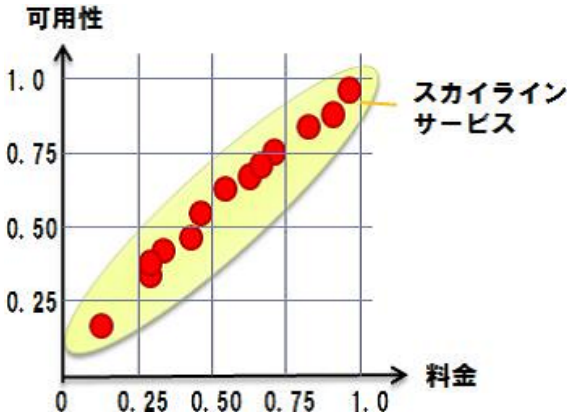

図 3　グルーピングによる領域作成
Fig. 3　Making area in grouping method.

近い QoS の値を持つサービスの判断基準となる QoS 間の距離のしきい値として，あらかじめ設定された類似判別値を使用する．この設定値が大きければ，それだけ併用対象のサービスが減って計算コストを大きく削減できるが，作成される仮想サービスが減るので併用による有効性が下がる．よってこの値に基づき，計算コストと有効性のトレードオフをとることができる．

サービスの抽象化は，正規化された QoS 項目を類似判別値によって等分割して作成された領域に基づいて行われる．例として，考慮する QoS 項目が料金と可用性で類似判別値が 0.25 である場合，それぞれの項目ごとに 4 等分して**図 3** のように 16 個の領域が作成される．こうして作成された領域内のサービス群を近い QoS を持つサービス群と見なす．そして各領域内に複数のサービスがある場合，その中から効用の値の大きい 2 個のサービスを代表サービスとして選別する．

各領域の代表サービスは，以下の併用の組合せに応じて下記のように用いられる．このため，同一領域内からの 2 個の代表サービスの選別が必要になる．

- 異なる領域間の組合せ
- 同一領域内の組合せ

異なる領域間の代表サービスの組合せには，2 個の代表サービスのうち効用の値が大きいものどうしを使用する．同一領域内の代表サービスの組合せには，領域内の 2 個の代表サービスどうしを使用する．

6. 評　　価

5 章の提案手法と併用の有効性を示すために，弱い制約下における併用考慮時の計算コストと併用の効果の評価をシミュレーションによって行う．計算コストの評価には作成した仮想サービスの数，計算時間，併用の効果の評価には効用，QoS 制約の満足率を用いた．

6.1　シミュレーション環境

文献 2) などの既存研究にならい，シミュレーションのモデルとして 6 つのタスクの順次実行によって構成されるアプリケーションを想定した．各タスクの機能を提供できるサービスを 20 個，または 200 個とする．各サービスの QoS には独自に作成したデータセットを使用し，考慮する QoS 項目を料金，応答時間，可用性とする．各サービスの料金には 0〜250 円，応答時間には 50〜600 msec，可用性には 0.5〜1.0 の乱数値を使用している．

このモデルに対し，QoS に基づくサービス選択のシミュレーションとして，以下のような問題設定を行う．ユーザが設定するアプリケーションの QoS 制約として，料金には 0〜1,500 円，応答時間には 300〜3,600 msec，可用性には 0〜1.0 の乱数値を使用する．これらはアプリケーション実行時に平均的に満足すべき QoS の値である．QoS 制約と各サービスの QoS の値が乱数なので，QoS 制約を満足できない環境も想定している．効用算出に用いる重みには，料金，応答時間，可用性の各項目の値の和が 1 になるように，0〜1.0 の乱数を使用する．また，仮想サービス作成の条件として料金，応答時間が選択候補となる実在サービスの最大値を超えない，可用性が 0.99999 を超えるサービスを構成対象として選択しないというものを課した．併用の構成には 4.1 節で述べた直列呼び出し，並列呼び出し，これらの組合せを使用する．このような問題設定に対して，仮想サービスの最大構成数と類似判別値の大きさを変えながら 200 回ずつシミュレーションを行った．

シミュレーションのプログラムには Java を使用した．また，MMKP に対して線形計画問題を高速に解くツールである IBM ILOG CPLEX を使用した．マシンには，Intel Pentium 4 CPU 3.80 GHz，2.00 GB の RAM，OS が Windows7 の 32 bit のものを使用している．

6.2　実 験 結 果

作成された仮想サービスの平均数の比較結果を**図 4**，**図 5** に示す．図 4 は各タスクのサービス数が 20 個のとき，図 5 は 200 個のときの結果である．類似判別値を 0，0.1，0.2 とし，大きいものほど使用する領域の幅を大きくしている．類似判別値 0 は，領域を作成せずスカイラインの逐次更新のみによる結果である．括弧で囲んでいる数値は仮想サービスを構

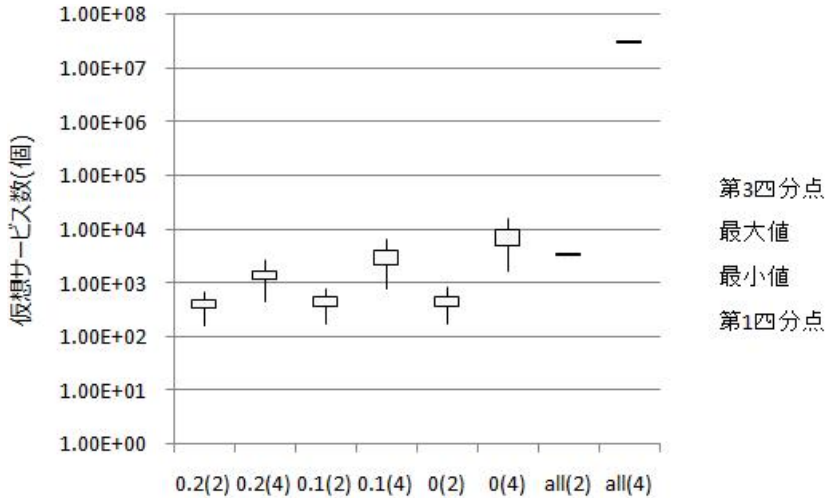

図 4　作成した仮想サービスの数（20）
Fig. 4　Number of virtual services (20).

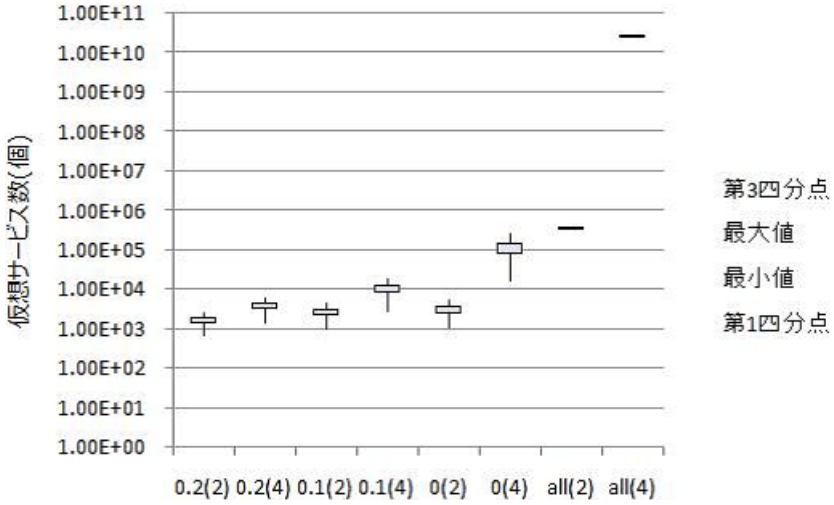

図 5　作成した仮想サービスの数（200）
Fig. 5　Number of virtual services (200).

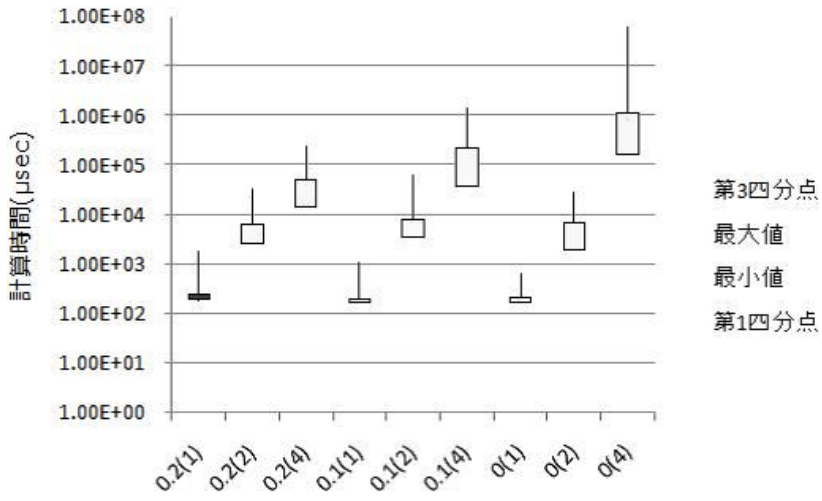

図 6　サービス選択にかかる計算時間（20）
Fig. 6　Computation time of service selection (20).

成する実在サービスの最大数を表す．all は提案手法を用いない場合の全組合せの結果であり，これはシミュレーションではなく理論値によって計算している．これらの結果から，提案手法が全組合せよりも仮想サービスの作成数を減らしていることが分かる．さらに，類似判別値を大きくすることで作成する仮想サービス数を減らしていることが確認できる．また，本手法を用いると QoS の値の傾向に関係なく仮想サービスの作成数を削減することができる．これは提案した 2 つの手法が影響している．スカイラインの逐次更新はサービスの QoS の値に傾向がないときに有効であり，グルーピングは QoS の値に傾向があり似たような QoS を持つサービスが多いときに有効である．このため，本手法によって仮想サービスの作成数は QoS の値の傾向に関係なく一定以下に抑えることができる．さらに類似判別値が 0.2 のときに各タスクのサービス数に関係なく一定以下の数の仮想サービスが作成できていることが確認できる．これはスカイラインサービスとグルーピングによって併用に有効なサービスの数を一定以下に抑えることができるからである．

仮想サービスを作成し，MMKP を解くまでの計算時間の結果を**図 6**，**図 7** に示す．図 6 は各タスクのサービス数が 20 個のとき，図 7 は 200 個のときの結果である．横軸には類似判別値，括弧で作成する仮想サービスの最大構成数を表している．図 6，図 7 と図 4，図 5

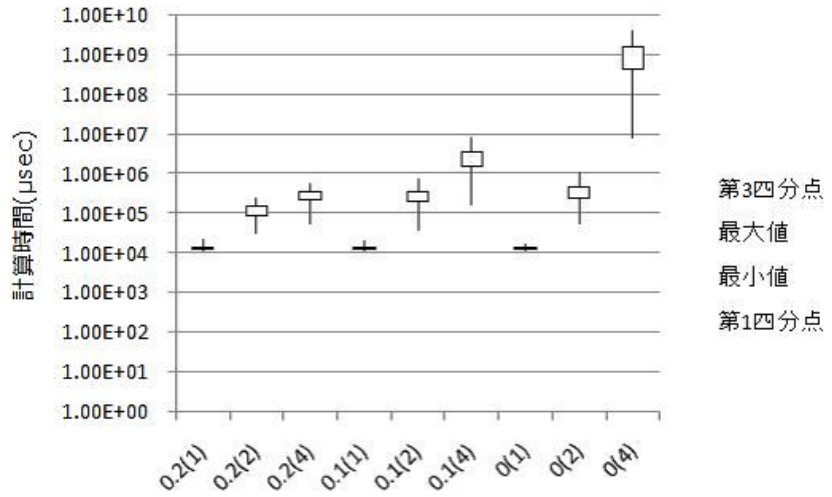

図 7　サービス選択にかかる計算時間（200）
Fig. 7　Computation time of service selection (200).

表 7　アプリケーションの効用
Table 7　Utility function of the application.

類似判別値	構成数：1	構成数：2	構成数：4
0.2（20）	4.3507	5.1508	5.2702
0.1（20）		5.1658	5.3000
0（20）		5.1701	5.3205
0.2（200）	5.0901	5.5961	5.6546
0.1（200）		5.6318	5.7052
0（200）		5.6384	5.7490

表 8　アプリケーションの QoS 満足率
Table 8　Rate of Satisfying QoS constraints of the application.

類似判別値	構成数：1	構成数：2	構成数：4
0.2（20）	0.46	0.82	0.86
0.1（20）			
0（20）			
0.2（200）	0.84	0.97	0.97
0.1（200）			
0（200）			

の結果から両者に相関関係があることが分かる．これより，以下のことが分かる．構成数の大きい仮想サービスを考慮することで作成する仮想サービスの数が多くなり，計算時間が多くかかっている．また，類似判別値を大きくすることで検討する仮想サービスの数を減らすことができ，計算時間の短縮が実現できている．さらに，各タスクのサービスの QoS の値の傾向やサービス数に関係なく一定以下の計算時間に抑えることができることが確認できる．ここで，各タスクのサービス数が 20 から 200 に 10 倍になった際の計算時間の増加は，たかだか 10^3 倍程度に抑えられている．本来各タスクに対し検討すべき組合せ数が 10^2 倍（構成数 2）または 10^4 倍（構成数 4）になっていること，さらにそれら 6 個（タスク数）の組合せを探索する MMKP を解いていることを考えると，提案手法により併用に起因する計算時間の爆発を抑えることができているといえる．

選択したサービスによって構成されるアプリケーションの QoS がユーザの嗜好に合ったものかどうかを表す効用の比較を**表 7** に示す．仮想サービスを構成する実在サービスの数の最大値が大きいほど，効用が上昇することが分かる．特に併用を考慮することにより（最大構成数 1 と最大構成数 2，4 を比べると），おおよそ 0.6 から 1.0 まで大きく上昇している．今回の 6 つのタスクを考えた環境では効用値の理論上の最大値（すべてのタスクにおいてすべての QoS 値がとりうる最も良い値となる場合）が 6 であるため，この上昇量は十分大きいと考えられる．一方で構成数を 2 から 4 に上げた場合の効用値上昇は 0.05 から 0.15 程度であり，特に選択候補となるサービスが元々多い（200 個の）場合にその効果は限られている．また，類似判別値を大きくすると 0.02 から 0.09 程度効用が下がることが分かるが，それでも多かれ少なかれ構成数を上げたことによる効用値の上昇の方が上回っており，またこの下がる値は併用を考慮すること自体による前述の上昇値に対して些細な量である．

アプリケーションの QoS がユーザの課した QoS 制約を満たした確率の比較を**表 8** に示す．表 7 で示した効用の結果と同じで，構成サービス数の最大値が大きいほど制約を満たす確率が上昇することが分かる．また，類似判別値を大きくすることで確率が下がることが考えられるが，わずかなために今回のシミュレーション結果には表れなかった．効用値の場合と同様に，選択候補となるサービスが元々多い（200 個の）場合に構成数を 2 から 4 に上げる効果は限られている．

以上の結果から，以下のことが考察される．まず，提案手法であるスカイラインの逐次更新とグルーピングによって計算時間の削減が確認できた．この際に削減できる計算時間は環境に依存するが，同種のサービス数や各サービスの QoS の値の傾向による影響を抑えて一

定時間以下にすることができ，本シミュレーションの環境では最大で 4 つのサービスの併用の考慮を 1 秒以下で計算できることが確認できた．また，サービスの併用によってユーザの要求により合致したサービスの利用方法が実現できることが効用や QoS 制約の満足率の結果から分かった．この際，特に選択サービス候補が少ない場合に，サービス構成数を上げることによる効果が大きくなる．この併用の効果は本手法を用いてもわずかに低下するだけであるため，計算コストを抑えて併用の効果を得るために本手法は有効であると思われる．これより，本手法によってアプリケーションの実行時間への影響を抑えて併用の有効性を得ることが確認できた．

7. 結　　論

本研究では，QoS に基づくサービス選択において同種の機能を提供するサービスの併用を考慮し，その際に要する計算コストを効率的に削減する手法を提案した．そしてスカイラインの逐次更新とグルーピングという手法により，作成する併用の数を減らすことで併用時の計算コストを抑えることを実現した．シミュレーションによって，提案手法が各タスクの機能要求を満たすサービスの数やその QoS の値の傾向に関係なく一定の計算時間に抑えることができ，さらに併用による効果が得られることを示し，本手法の有効性を示した．今後の研究方針として，強い制約下におけるサービスの併用を考慮した際の実行中のサービス再選択アルゴリズムの提案を考えている．これは，直列呼び出し時に先に呼び出したサービスが成功することで計画していた QoS から実際の値が変化することを想定し，その変化に応じて実行中に以降のサービスをより良いものに再選択するものである．

参 考 文 献

1) Alrifai, M., Skoutas, D. and Risse, T.: Selecting skyline services for QoS-based web service composition, *Proc. 19th International Conference on World Wide Web*, pp.11–20, ACM (2010).
2) Alrifai, M. and Risse, T.: Combining global optimization with local selection for efficient qos-aware service composition, *Proc. 18th International Conference on World Wide Web*, ACM New York, NY, USA, pp.881–890 (2009).
3) Qi, L., Tang, Y., Dou, W. and Chen, J.: Combining Local Optimization and Enumeration for QoS-aware Web Service Composition, *2010 IEEE International Conference on Web Services*, pp.34–41, IEEE (2010).
4) Al-Masri, E. and Mahmoud, Q.: Investigating web services on the world wide web, *Proc. 17th International Conference on World Wide Web*, pp.795–804, ACM (2008).
5) Web Services Search Engine @ seekda.com, available from ⟨http://webservices.seekda.com/⟩.
6) Chen, K., Xu, J. and Reiff-Marganiec, S.: Markov-htn planning approach to enhance flexibility of automatic web service composition, *IEEE International Conference on Web Services, ICWS 2009*, pp.9–16, IEEE (2009).
7) Zeng, L., Benatallah, B., Ngu, A., Dumas, M., Kalagnanam, J. and Chang, H.: QoS-aware middleware for web services composition, *IEEE Trans. Softw. Eng.*, Vol.30, No.5, pp.311–327 (2004).
8) Ardagna, D. and Pernici, B.: Adaptive service composition in flexible processes, *IEEE Trans. Softw. Eng.*, pp.369–384 (2007).
9) Yu, T., Zhang, Y. and Lin, K.: Efficient algorithms for Web services selection with end-to-end QoS constraints, *ACM Trans. Web* (*TWEB*), Vol.1, No.1, p.6 (2007).
10) Zheng, Z. and Lyu, M.: A distributed replication strategy evaluation and selection framework for fault tolerant web services, *2008 IEEE International Conference on Web Services*, pp.145–152, IEEE (2008).
11) Guo, H., Huai, J., Li, H., Deng, T., Li, Y. and Du, Z.: ANGEL: Optimal configuration for high available service composition, *IEEE International Conference on Web Services, ICWS 2007*, pp.280–287 (2007).
12) Stein, S., Payne, T. and Jennings, N.: Flexible provisioning of web service workflows, *ACM Trans. Internet Technology* (*TOIT*), Vol.9, No.1, p.2 (2009).
13) Zhai, Y., Zhang, J. and Lin, K.: SOA Middleware Support for Service Process Reconfiguration with End-to-End QoS Constraints, *IEEE International Conference on Web Services, ICWS 2009*, pp.815–822, IEEE (2009).

付　　録

A.1　スカイラインの逐次更新の有効性

この章では，スカイラインの逐次更新において組合せの対象をスカイラインサービスに限定することの有効性について述べる．ここで，スカイラインサービス以外のサービスを考慮するためにスカイラインレベルという指標を用いる．これは，仮想サービスの構成対象となるサービスの範囲を決めるものである．A.1.1 でスカイラインサービスのみを構成対象にすることで失われる有効性の説明をし，A.1.2 でスカイラインレベルについて述べ，A.1.3 でスカイラインレベルを用いた評価実験を行い，スカイラインの逐次更新において構成対象をスカイラインサービスのみに限定することの有効性について説明する．

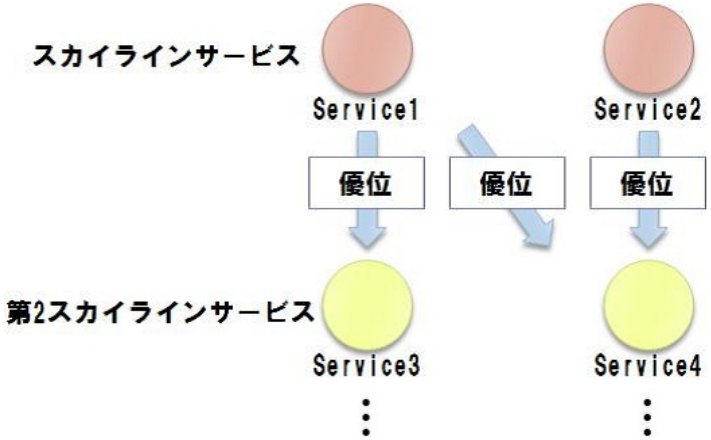

図 8　スカイラインサービスの関係
Fig. 8　Relationship of skyline services.

A.1.1　失われる有効性

スカイラインの逐次更新において，その構成対象をスカイラインサービスに限定することで失われる有効性について，**図 8** のような例を用いて説明する．あるタスクの機能を満たす同種サービスとして Service 1〜n があるとする．ここで Service1，Service2 はスカイラインサービスであり，これらを第 1 スカイラインサービスと表現する．そして第 1〜$k-1$ スカイラインサービスを除いた際にスカイラインサービスとなるサービス群を第 k スカイラインサービスとする．また，Service3 は Service1 に QoS の全項目において優位性を持たれるとし，同様の関係が図 8 のようにサービス間にあるとする．

図 8 の状況において構成数が 2 個の仮想サービスを作成することを考える．このとき，作成した仮想サービスがスカイラインサービスとなる可能性があるのは Service1 と Service2，Service1 と Service3 の組合せである．Service3 や Service4 を用いた組合せは，より優れた QoS を持つ Service1 や Service2 を用いた組合せが他に存在するため，スカイラインの性質上スカイラインサービスにはなりえない．よってスカイラインサービスとなる仮想サービスをすべて作成するには，第 1 スカイラインサービスどうしと，ある第 1 スカイラインサービスとそれのみに優位性を持たれる第 2 スカイラインサービスの組合せが必要である．

次に，図 8 の状況において構成数が 4 個の仮想サービスの作成を考える．このとき，構成数が 2 個の仮想サービスどうしの組合せが考えられるが，これは先ほどと同様に第 1，第 2 スカイラインサービス上のものが必要となる．ただ，このときの第 2 スカイラインサービスは前の試行で第 2，第 3 スカイラインサービスによって更新される可能性を持つ．さらに，組み合わせる仮想サービスの構成サービスに重複があってはならないため，第 1，第 2 スカイラインサービスの組合せに重複がある場合，第 1，第 3 スカイラインサービスの組合せが有効になる場合がある．よってこのときスカイラインサービスとなる構成数が 4 個の仮想サービスをすべて作成するには，前の試行で構成数の 2 個の仮想サービスを第 3 以下のスカイラインサービスを用いて作成する必要がある．

以上のように，スカイラインサービスとなりうるすべての仮想サービスの作成を検討すると，仮想サービスの最大構成数が増加することで，より下位のスカイラインサービスの組合せが必要となる．しかし，このような下位のスカイラインサービスの組合せが有効となるには，それより優位性を持つサービスの数が少ないという条件が必要であり，その可能性は下位になるほど小さくなる．本研究では，仮想サービスの構成対象を（第 1）スカイラインサービスのみに限定することでこの少ない可能性を無視している．この無視による影響を評価するため，以降の節でスカイラインレベルを用いた評価実験を行う．

A.1.2　スカイラインレベル

スカイラインレベルとは，仮想サービスを作成する際に考慮するスカイラインサービスの位の最大値である．たとえば，スカイラインレベルが 3 のときは第 1，第 2，第 3 スカイラインサービスの組合せによって仮想サービスを作成する．本研究では，スカイラインレベルに応じて仮想サービスを作成する際の各スカイラインサービスの組合せを式 (5) のように定める．ここでは，スカイラインレベルを i とし，構成対象の 2 つのサービスを含むスカイラインサービスの位をそれぞれ S_1，S_2 としている．たとえばスカイラインレベルが 3 のとき，第 1 と第 1〜3，第 2 と第 2 スカイラインサービスの組合せで仮想サービスを作成する．この仮想サービス作成に使用するスカイラインサービスの位と組合せは，逐次更新の各試行において同一とする．

$$S_1 + S_2 - 1 \leq i \tag{5}$$

式 (5) のように組合せを定めることで，仮想サービスの作成によって更新されるスカイラインサービスの位が次の試行での組合せで使用するものになる．たとえばスカイラインレベルが 3 のとき，最も有効性の低い第 1 と第 3，第 2 と第 2 スカイラインサービスの組合せによる仮想サービスは主に第 3 スカイラインサービスを更新する．これは，これらよりも優れた組合せがそれぞれ 2 つ考えられるからである．これにより次の試行で仮想サービスを作成する際，スカイラインレベルによって更新された第 1〜3 スカイラインサービスを選択

することができる．ただ，ここでは第 4 以下のスカイラインサービスによる第 3 以上のスカイラインサービスの更新を考慮していない．よって作成する仮想サービスの最大構成数が大きくなれば，このときの有効性は低くなる．

スカイラインレベルの値が大きくなれば，スカイラインサービスを更新する可能性を持つ有効な仮想サービスの網羅率が向上する．そして，このとき考慮する組合せの数が増大するため，計算コストが増大する．このようにスカイラインレベルは作成する仮想サービスの有効性と計算コストのトレードオフをとる．

A.1.3　評 価 実 験

スカイラインレベルによる併用考慮時の有効性を示すため，弱い制約下における併用考慮時の計算コストと併用の効果の評価をシミュレーションによって行う．計算コストの評価には作成した仮想サービスの数，計算時間，併用の効果の評価には効用，QoS 制約の満足率を用いた．

A.1.3.1　シミュレーション環境

6.1 節と同様の環境を想定している．6 つのサービスによって構成されるアプリケーションを想定し，各タスクの機能要求を満たすサービスの数を 20 個とする．考慮する QoS 項目を料金，応答時間，可用性とし，各値に関しては 6.1 節と同様な乱数値を用いた独自のデータセットを用いる．そして 6.1 節で用いた乱数値に基づく弱い制約下で乱数の重みを用いた効用の値を最大化する問題を解く．シミュレーションに用いた PC などの環境も 6.1 節と同様であり，スカイラインレベルの値を 1〜3 に変えながら 200 回ずつ評価を行った．なお，スカイラインレベルが 1 の場合は，本論文で使用しているスカイラインサービスどうしの組合せと同一の検討を行っている．

A.1.3.2　実 験 結 果

計算量の評価のため，作成した仮想サービスの数の比較結果を**図 9** に示す．スカイラインレベルによって検討される組合せの数が異なることが分かる．また，MMKP を解くまでの計算時間の比較結果を**図 10** に示す．図 9 における仮想サービスの数の関係と相関があり，スカイラインレベルが大きくなると計算時間がかかることが分かる．

アプリケーションの効用の比較結果を**表 9** に示す．仮想サービスの構成数が増加するほど効用は向上するが，スカイラインレベルの増加では微小な向上しか得られないことが分かる．また，アプリケーションの QoS 制約の満足率の比較結果を**表 10** に示す．これも表 9 の効用の比較結果と同じで，仮想サービスの構成数の増加によって大きな向上が見られるが，スカイラインレベルの増加ではあまり向上が見られない．

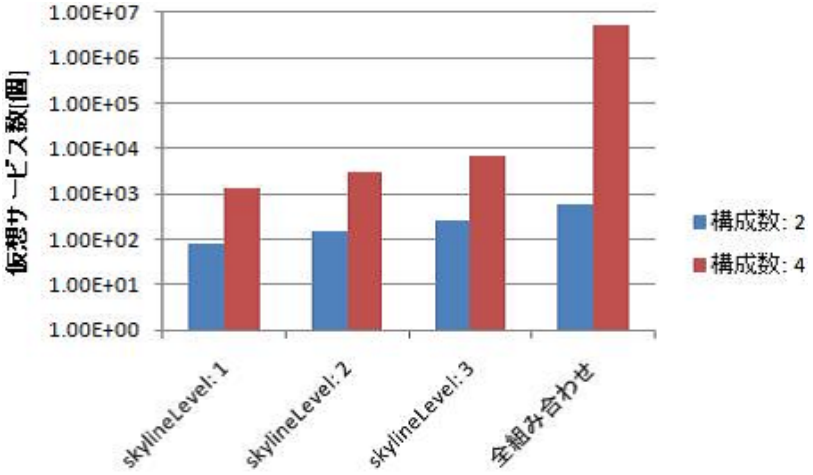

図 9　スカイラインレベルごとの作成した仮想サービス数の比較結果
Fig. 9　Number of virtual services in each skyline level.

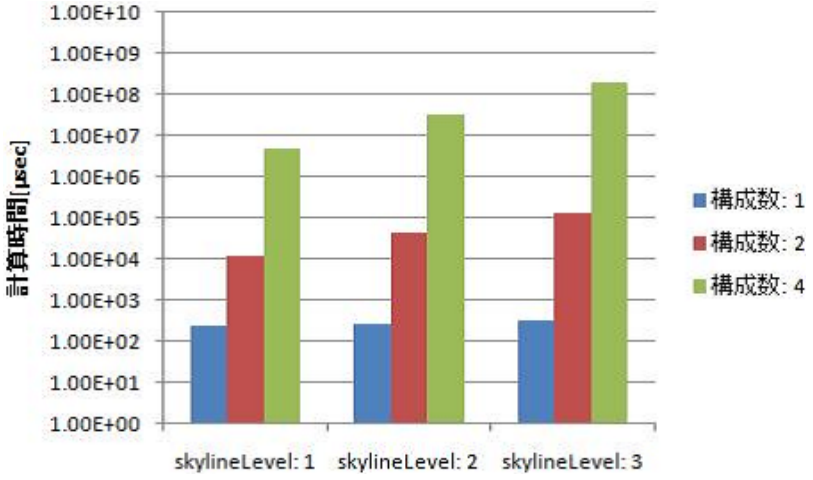

図 10　スカイラインレベルごとの計算時間の比較結果
Fig. 10　Computation time in each skyline level.

表 9　スカイラインレベルごとの効用の比較結果
Table 9　Utility function in each skyline level.

	構成数：1	構成数：2	構成数：4
skylineLevel:1	4.6643	5.2459	5.2921
skylineLevel:2		5.24670	5.2959
skylineLevel:3		5.2467	5.2964

表 10　スカイラインレベルごとの QoS 満足率の比較結果
Table 10　Rate of satisfying QoS constraints in each skyline level.

	構成数：1	構成数：2	構成数：4
skylineLevel:1	0.36	0.715	0.775
skylineLevel:2		0.72	0.78
skylineLevel:3		0.72	0.78

以上の結果から，スカイラインサービス以外のサービスを用いた仮想サービスの作成を検討してもアプリケーションの QoS にわずかな向上しか見られず，その代わりに計算コストが非常にかかることが分かる．よって，アプリケーションの QoS 向上の効果を得ながら小さい計算コストを得るためには，スカイラインサービスのみを用いて併用を検討することが有効であることが分かる．

(平成 22 年 12 月 29 日受付)
(平成 23 年 7 月 8 日採録)

平塚　信明
2011 年東京大学大学院情報理工学研究科創造情報学専攻修士課程修了．2011 年より野村総合研究所に所属．

石川　冬樹（正会員）
2007 年東京大学大学院情報理工学研究科コンピュータ科学専攻博士課程修了．博士（情報理工学）．2007 年より国立情報学研究所助教，総合研究大学院大学複合科学研究科助教兼任．現在に至る．サービスコンピューティング，ソフトウェア工学の研究に従事．2009 年より電子情報通信学会情報・システムソサイエティサービスコンピューティング時限専門研究委員会副委員長．

本位田真一（フェロー）
1978 年早稲田大学大学院理工学研究科修士課程修了．（株）東芝を経て 2000 年より国立情報学研究所教授，2004 年より同研究所アーキテクチャ科学研究系研究主幹を併任，現在に至る．2008 年より同研究所先端ソフトウェア工学・国際研究センター長を併任，現在に至る．2001 年より東京大学大学院情報理工学系研究科教授を兼任，現在に至る．現在，早稲田大学客員教授，英国 UCL 客員教授を兼任．2005 年度パリ第 6 大学招聘教授．工学博士（早稲田大学）．1986 年度情報処理学会論文賞受賞．日本ソフトウェア科学会理事，情報処理学会理事を歴任．ACM 日本支部会計幹事，情報処理学会フェロー，日本ソフトウェア科学会編集委員長，日本学術会議連携会員．