Vol.2017-MPS-112 No.13
2017/2/27

Web システムの性能評価に基づく クラウド資源割当最適化モデルの提案

齋藤 篤志[†1]　山下 雅喜[†2]　三浦 克宜[†3]　棟朝 雅晴[†4]

概要：エンジニアは世界中に無数にあるクラウドサービスを吟味して Web システムを構築するが，その作業は煩雑である．クラウドブローカーシステムではエンジニアの要求要件に最適な Web システムを提示することで，エンジニアの負担を軽減できる．しかし従来研究では実際のシステムの性能評価に基づく目的関数の検討を行っていなかった。本論文では現実的なシステムを提示するための多目的最適化数理モデルを提案する．

キーワード：クラウドコンピューティング, Web 三層モデル，クラウドブローカー，最適化，数理モデル

1. はじめに

現在クラウドコンピューティングは広く浸透し，無数のクラウドサービスが提供されている．一方でエンジニア/ユーザーは無数のクラウドサービスから自身の要求要件に見合うクラウドサービスを見つけ出すのが難しい状況にある．そこでクラウドブローカーサービスが求められている．クラウドブローカーサービスとは最適なクラウドリソースの選択や，複数のクラウドサービスの統合管理・運用をサポートするサービスのことである．本論文ではクラウドブローカーサービスにおいて，要求要件に対して，最適な Web 三層システムを提示する際の数理モデルについて検討する．クラウドブローカーサービスに関する論文は数多く存在しているが，特徴的な研究を４つ紹介する．クラウドブローカーサービスに関する研究において，ユーザーの求めるインスタンスを単独で選択するものと，アーキテクチャ全体に対してユーザーの求める複数のインスタンスを選択するものがある．

まず，単独のインスタンスを評価する K.Greg ら[1]の論文では CSMIC が提案する Service Mesurement Index(SMI)というクラウドサービス評価基準を用い，それぞれが階層的に示された 7 つの性質に基づいて評価を行なう．それぞれのサービスを SMI で評価した上で，階層分析法(AHP)を用いてインスタンスの評価を行なっている．AHP は階層構造を探索するために優れているため，SMI とともに用いる必然性があり，定性的な QoS を定量的に評価している点が独創的である．

次に，S. Sundareswaran ら[2]の研究では，無数にあるクラウドサービスを符号化して管理する手法を述べている．この論文の主眼は効率的なインデックスと探索を行なうことであった．確かに CSP-index を用いてクラウドサービスをバイナリ符号化し，CSS-query algorithm でハミング距離を用いて探索をすることは高速である．しかし，ハミング距離が近いからといって，それが最適解とは言えないことが問題である．上記２つの研究では，単独のインスタンスを選択する方法ではあるが，最適解を選択できない点で，本研究との差異がある．

次に，アーキテクチャ全体を考慮して，複数のインスタンスを選択する研究であるが，川勝ら[3]の研究では，世界的な計算機資源に焦点を当て，Web 三層システムの最適化を行なっている．主眼は，データセンター内の計算機資源を最適化することであり，クラウドブローカーサービスにおいて主眼となるユーザー(あるいはエンジニア)目線での目的関数は設定されていない．本論文ではユーザー目線でクラウドサービスを選ぶ目的関数を明確に議論している点で優位性がある．またこの論文[3]では NSGA-Ⅱで最適解を計算しているが，クラウドブローカーサービスでは数多くの目的関数を扱うため，目的関数の増加によって計算量が跳ね上がる．その点で，本論文で用いた NSGA-Ⅲの方が適当であるだろう．

次に，クラウドブローカーに特化した研究では P. Pawluk ら[4]の研究がある．この研究では，クラウドブローカーサーサービスを実装し，ユーザーのリクエストに答える最適なアーキテクチャの構築を目指している．解くべき問題を正しく定式化できているが，ユーザーが実際にクラウドサービスを選択する目的関数は十分に議論されておらず，最適化手法も荷重和法が取られていて不十分である．本論文で利用した多目的最適化手法を用いるべきである．

2. 数理モデル

クラウドブローカーシステムが解くべき問題は制約条件つき多目的最適化問題として定義できる．なぜならば第一に，ユーザーからの要求要件によって複数の制約がつくこ

†1 北海道大学 情報科学研究科
Graduate School of Informaiton Science and Technology, Hokkaido University
†2 SCSK 株式会社
SCSK Corporation
†3 北見工業大学 情報処理センター
Infomation Processing Center, Kitami Institute of Technology
†4 北海道大学 情報基盤センター
Information Initiative Center, Hokkaido University

とが普通である．例えばユーザーはクラウドを設置する地域に関して，国外に置くことに抵抗があることもあるだろう．このような場合，要求要件の一部は最適化の対象とならず，解が必ず満たすべき制約条件になる．第二に，クラウドサービスを選択する場合，選択する基準がコストやパフォーマンス，稼働率など複数存在し，トレードオフ関係が見られるものもあるが，その中でユーザーはどの基準でもできる限り良いものを選択したいと考える．よって多目的最適化問題である．以上より，クラウドブローカーシステムの数理モデルは制約条件つき多目的最適化問題である．

数式で表すならば，変数空間の次元を n，目的関数の個数を m とし，制約充足付き多目的最適化問題の解を $\boldsymbol{x} = (x_1, x_2, \cdots x_n)^T \in R^n$，目的関数を $\boldsymbol{f} = (f_1, f_2, \cdots, f_m)^T$，可能領域を $S \subset R^n$ とすると，次のように記述される．

$$minimize\ \boldsymbol{f_i(x)}\ (i = 1,2,\cdots,m) \quad \text{ただし}\ \boldsymbol{x} \in S$$

また可能領域Sは一般にL個の制約条件を満たす領域として以下のように記述される．

$$\boldsymbol{g}_j(\boldsymbol{x}) \leq 0\ (j = 1,2,\cdots,L)$$

つまり，目的関数 $\boldsymbol{f_i(x)}$の解 $\boldsymbol{x}$ は可能領域 S の内部で選択される．$\boldsymbol{g_j(x)}$>0 の場合，解 $\boldsymbol{x}$ は可能領域 S の外部に属するため，最適化アルゴリズムで利用されることはあるが，選択はされない．

3. 目的関数

本論文では最適化に用いる 7 つの目的関数を検討する．目的関数には，ユーザー視点での目的関数とエンジニア視点での目的関数があると考えられる．なぜなら同じシステムに関してみても，ユーザーとエンジニアでは重要視する点が異なると思われる．例えば，システムのパフォーマンスに関してみても，ユーザー視点ではシステムに対するリクエストが素早く確実に応答されると嬉しいが，エンジニア視点では，多少のリクエストが応答できなくてもシステムが落ちないことが重要だろう．これら 2 つの視点から，今回提案する目的関数を分類すると以下のようになる．

ユーザー視点：レスポンスタイム，棄却率
エンジニア視点：オペレーションコスト，構成変更コスト，スループット，稼働率，複雑性

3.1 オペレーションコスト

エンジニア視点では，できる限り低いコストでシステムを運用できる構成にしたい．目的関数 f_{op_cost} では，あるシステム構成 c_i の場合，一ヶ月あたりにかかるコスト（金額）を求める．目的関数 f_{op_cost} は最小化を目指す目的関数である． Web サーバー，アプリケーションサーバー，データベースサーバー，ロードバランサーの料金の各合計をそれぞれ P_{Web}，P_{App}，P_{DB}，P_{LB}，VM の監視料金の合計を P_{watch} とすると以下のように記述される．

$$f_{op\_cost}(c_i) = P_{Web} + P_{App} + P_{DB} + P_{LB} + P_{watch}$$

つまり単純に，一ヶ月あたりのコストの総和として表される．

3.2 構成変更コスト

あるシステムに変更を加える場合，構成の変更が大きいほどエンジニアの負担は大きくなるため，できる限り小さな変更で済ませたい．目的関数 f_{reconf_cost} では，あるシステム構成 c_i からある構成 c_j に，構成を変更した場合にかかるコスト（金額）を求める．目的関数 f_{reconf_cost} は最小化を目指す目的関数である．Web サーバー，アプリケーションサーバー，データベースサーバー，ロードバランサーの構成の変更にかかるコストをそれぞれ C_{Web}，C_{App}，C_{DB}，C_{LB} とすると以下のように記述される．

$$f_{reconf\_cost}(c_i, c_j) = C_{Web} + C_{App} + C_{DB} + C_{LB}$$

以上のように，変更部分のコストの総和として表される．

3.3 スループット

エンジニア視点では，構成するシステムができる限り多くのリクエスト数を処理できるようにしたい．目的関数 f_{perf} では，あるシステム構成 c_i が 1 秒あたり何リクエストを処理できるかを求める．目的関数 f_{perf} は最大化を目指す目的関数である．本論文では待ち行列ネットワークを用いてスループットの計算を行う．待ち行列ネットワークを用いた多層システムの性能予測では[4]の研究があるが，推定結果は芳しくなかった．しかし B. Urgaonkar ら[6]による多層システムの分析的モデル（図 1）では，様々な実験において，95%の精度でスループットを推定することに成功しており，本論文では[6]に依ってスループット推定を行う．

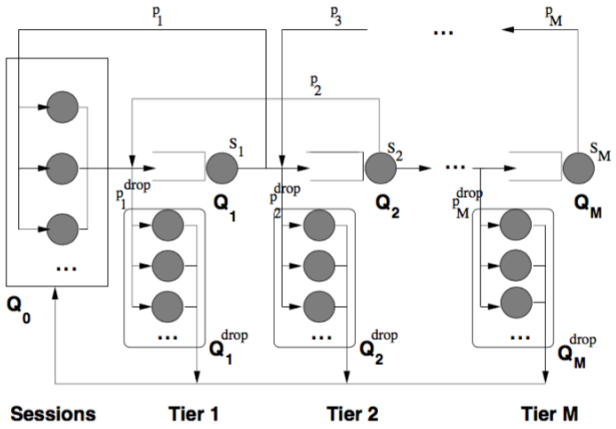

図 1：多層システムの分析的モデル（[6]より）

この分析的モデルの特徴は同時セッションの制限によって，リクエストが棄却される確率 P_x^{Drop} を考慮した点にある．これにより，リクエスト数が増大した際の Web サーバーや App サーバーの動作を現実的に推測することが可能になったと考えられる．

この分析的モデルからスループット τ を算出するためには，

３つのステップが必要である．

1, あるシステム構成において同時セッション制限がないものとして（$P_x^{Drop}=0$），スループットλを MVA で計算する(表 1).

2, Q_x を M/M/1(K_x)の上限付き待ち行列とし，λ V_x を到着率として，同時セッションを行ったときに棄却される確率 P_x^{Drop} を計算する．

3, P_x^{Drop} を考慮して再度 MVA を行い，スループットτを算出する．

input : $N, \bar{S}_m, V_m, 1 \le m \le M; \bar{Z}$
output : $\bar{R}_m$ (avg. delay at Q_m), $\bar{R}$ (avg. resp. time)

initialization:

$\bar{R}_0 = \bar{D}_0 = \bar{Z}; \bar{L}_0 = 0;$

for $m = 1$ ***to*** M **do**
  $\bar{L}_m = 0;$
  $\bar{D}_m = V_m \bar{S}_m$ /* service demand at each queue */;
end
/* introduce N customers, one by one */
for $n = 1$ ***to*** N **do**
  for $m = 1$ ***to*** M **do**
    $\bar{R}_m = \bar{D}_m(1 + \bar{L}_m)$ /* avg. delay at each que.*/;
  end
  $\tau = \left(\frac{n}{\bar{R}_0 + \sum_{m=1}^{M} \bar{R}_m} \right)$ /* throughput */;
  for $m = 1$ ***to*** M **do**
    $\bar{L}_m = \tau \cdot \bar{R}_m$ /* update queue lengths (little's law) */;
  end
  $\bar{L}_0 = \tau \cdot \bar{R}_0;$
end
$\bar{R} = \sum_{m=1}^{m=M} \bar{R}_m$ /* response time */;

表 1：多層システムにおける MVA アルゴリズム（[6]より）

Symbol	Meaning
M	Number of application tiers
N	Number of sessions
Q_m	Queue representing tier T_m $(1 \le m \le M)$
Q_0	Inf. server system to capture sessions
$\bar{Z}$	User think time
$\bar{S}_m$	Avg. per-request service time at Q_m
$\bar{L}_m$	Avg. length of Q_m
τ	Throughput
$\bar{R}_m$	Avg. per-request delay at Q_m
$\bar{R}$	Avg. per-request response time
$\bar{D}_m$	Avg. per-request service demand at Q_m
V_m	Visit ratio for Q_m
$\bar{A}_m$	Avg. num. customers in Q_m seen by an arriving customer

表 2：使われる記号の意味（[6]より）

以上より，目的関数 f_{thrpt} は以下のようになる．

$$f_{thrpt}(c_i) = \tau$$

目的関数 f_{thrpt} は最大化を目指す目的関数である．

3.4 レスポンスタイム

3.3 において，MVA アルゴリズムを用いてスループットの推定を行ったが，このアルゴリズムでは同時にレスポンスタイムの算出も行うことができる．ユーザーはシステムの内部を覗くことはないため，自分のリクエストに対するレスポンスタイムが短いことが，システムに対する一つの評価となる．よって，目的関数 f_{rsp_time} は次のようになる．

$$f_{rsp\_time}(c_i) = \bar{R}$$

目的関数 f_{res_time} は最小化を目指す目的関数である．

3.5 棄却率

3.4 と同様に，ユーザー視点では，自身のリクエストが棄却される場合はシステムに対する評価が下がると考えられる．3.3 で算出した棄却率 P_x^{Drop} はリクエストが棄却される確率であり，この値が小さいほどユーザーの評価は高くなる．よって，目的関数 f_{reject} は以下のようになる．

$$f_{reject}(c_i) = \sum_{x=1}^{M} P_x^{Drop}$$

目的関数 f_{reject} は最小化を目指す目的関数である．

3.6 動作率

エンジニアにとって，障害に対して強いシステムを構成することは非常に重要である．目的関数 f_{avail_ratio} ではあるシステム構成 c_i の動作率を求める．動作率とは，システムの動作が期待される時間のうち，実際にシステムが動作した時間の割合である．目的関数 f_{avail_ratio} は最大化を目指す目的関数である．本論文では，ベイジアンネットワークを用いて，システムが動作する確率事象を分析し，動作率を計算する．システムが動作する確率を $P(X)$とすると，システムの動作率は動作している割合と等しいため，以下のように記述される．

$$f_{avail\_ratio}(c_i) = P(X)$$

Web 三層システムの特徴として，Web 層，App 層，DB 層のどの層がダウンしても，システム全体がダウンすることが挙げられる． Web 層，App 層，DB 層が動作する確率をそれぞれ $P(W)$，$P(A)$，$P(D)$とすると，全ての層が動作する確率であるため，以下のように記述される．

$$P(X) = P(W)P(A)P(D)$$

各層が動作しない確率$P(\overline{W})$，$P(\overline{A})$，$P(\overline{D})$はそれぞれベイジアンネットワークにおけるレイヤーを３つに分けることで分析できる．すなわち，(i)同じレイヤーの事象群の影響で停止する確率$P(\overline{W_S})$，(ii)下のレイヤーの事象群の影響で停止する確率$P(\overline{W_U})$, (iii)下のレイヤーの事象群が同じレイヤーの事象群に影響を与えて停止する確率$P(\overline{W_{U\to S}})$である．(i)と(ii)を合わせた確率から(iii)の確率を引くと，P(W), P(A), P(D)の確率を計算できる．例えば Web 層においては以下の

ように記述される．

$$P(\overline{W}) = P(\overline{W_S}) + P(\overline{W_U}) - P(\overline{W_{U\to S}})$$

$P(\overline{W_S}), P(\overline{W_U}), P(\overline{W_{U\to S}})$を一つずつ分析する．システム構成要素停止の事象群では，App 層に不具合がある場合，Web 層で正しい情報が提示されない不具合が予想される．さらに DB 層に不具合がある場合，App 層で正しい計算処理が行われず，結果的に Web 層に正しい情報が提示されないことが予想されるため，Web 層は App 層に依存し，App 層は DB 層に依存するといえる．よって，Web 層，App 層が独立してダウンする確率を P(w),P(a)とすると，以下のように記述される．

$$P(\overline{W_S}) = P(w) + P(\overline{W}|\bar{A})P(\bar{A})$$
$$P(\overline{A_S}) = P(a) + P(\bar{A}|\bar{D})P(\bar{D})$$
$$P(\overline{D_S}) = P(d)$$

次に，インスタンス停止の事象群では各 VM が停止した場合にシステム構成要素に与える影響を示している．システム構成要素である各層は，アベイラビリティゾーン停止の事象群で各層中の VM が一定台数停止した場合にダウンしたものと見なされる．インスタンス n 台のうち r 台が動作する場合 (n > r) に Web 層が動作する確率は，

$$P(\overline{W_U}) = 1 - \left(\sum_{i\in\{r,\dots,n\}} \left(\sum_{S\in C(n,i)} \left(\prod_{s\in S} p(w_s) \right) \left(\prod_{t\in\{1,\dots,n\}\setminus S} p(\overline{w_t}) \right) \right) \right)$$

と表せる．

また今回，Web 三層システムにおいてインスタンスの共有を想定していないため，$P(\overline{W_{U\to S}}) = 0$である．以上より，$P(\overline{W_S}), P(\overline{W_U}), P(\overline{W_{U\to S}})$が分析できた．さらに，インスタンス停止よりも下のレイヤーに関しても分析する．

アベイラビリティゾーンとは，同一地域に位置するデータセンター群を指す（例えば，バージニア州）．また同一アベイラビリティゾーン内に設置された VM は同時にダウンする確率が高いと推察される．よって，例えば，w_1 がダウンする確率を P(w_1)として，以下のように記述される．

$$P(w_1) = 1 - P(\overline{w_1})$$
$$P(\overline{w_1}) = P(\overline{w_{1S}}) + P(\overline{w_{1U}}) - P(\overline{w_{1U\to S}})$$

ここで，インスタンス停止の事象群において，Web 層，App 層ではインスタンスの共有は想定していないため，

$$P(\overline{w_{1S}}) = 0$$
$$P(\overline{w_{1U\to S}}) = 0$$

アベイラビリティゾーンの停止$P(\overline{Az_1})$によってインスタンスが停止する確率を考慮し，インスタンスが独立して停止する確率を$P(\overline{w_1})$とすると，以下のように記述される．

$$P(\overline{w_{1U}}) = P(\overline{w_1}|\overline{Az_1})P(\overline{Az_1}) + P(\overline{w_1})$$

ただし，DB 層では，マスターノードとスレーブノードなど，依存関係をもつ形態も少なくない．例えば，D1 が D2 に依存している場合，$P(\overline{d_1^*})$を d_1 が独立して停止する確率として以下のように記述される．

$$P(\overline{d_{1S}}) = P(\overline{d_1}|\overline{d_2})P(\overline{d_2}) + P(\overline{d_1^*})$$
$$P(\overline{d_{1U}}) = P(\overline{d_1}|\overline{A_{Z_3}})P(\overline{A_{Z_3}})$$
$$P(\overline{d_{1U\to S}}) = P(\overline{d_1}|\overline{d_2})P(\overline{d_2}|\overline{A_{Z_3}})P(\overline{A_{Z_3}})$$

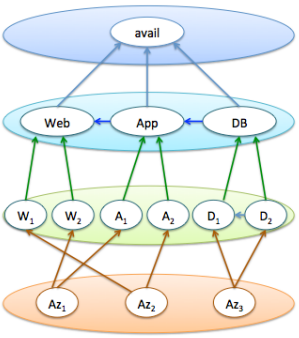

図 2：ベイジアンネットワークによるシステムがダウンする事象の分析

3.7 複雑性 Complexity

エンジニアにとって，システムの構成が複雑になるほど保守・管理にかかる労力が増大する．本論文では，ある層 n において，サーバー台数 S_n が増えるほど，また利用するインスタンスタイプの違い D_n があるほど複雑性が増大するものとし，複雑性の目的関数f_{cmpl}を次のように定式化した．

$$f_{cmpl}(c_i) = \sum_{n=1}^{N} w_1 * S_n + w_2 * D_n$$

ここで w_1，w_2 は複雑性の増大に関する重みである．複雑性の目的関数f_{cmpl}は最小化を目指す目的関数である．

4. 多目的最適化アルゴリズム

2.において検討した制約条件付き多目的最適化問題を解くためには，十分に解空間の探索がなされる必要があるが，従来の手法では，多数の目的関数を同時に最適化する点で非常に難しい問題である．本論文では進化的計算手法である遺伝的アルゴリズムを拡張した NSGA-Ⅲアルゴリズムを用いて多目的最適化を行う．NSGA-Ⅲアルゴリズムは多目的最適化手法の中でも計算負荷が小さい特徴がある．

4.1 NSGA-Ⅲアルゴリズム

NSGA-Ⅲアルゴリズムは，NSGA-Ⅱアルゴリズムを計算負荷の点で改善したアルゴリズムである．NSGA-Ⅱアルゴリズムから高速非優越ソートとエリート主義を引き継ぎ，NSGA-Ⅱアルゴリズムで計算のボトルネックになっていた混雑度の計算は，参照点を用いた選択に変更されている(図 N)．これらの工夫によって，パラメータが不要で，計算負荷の小さいアルゴリズムとなっている．以下，高速非優越ソート，エリート主義，参照点による選択について簡単に説明する．

4.1.1 高速非優越ソート

高速非優越ソートでは各個体に対して，優越している個体と優越されている個体の数を同時に数える．優越されている個体がゼロの個体をランク 1, ランク 1 の個体を取り除いて優越されている個体がゼロの個体をランク 2, と全個体がなくなるまで繰り返す．優越・被優越の数を数えることで高速なランク分けを実現している．

4.1.2 エリート主義

エリート主義とは常に優良個体を保存する親母集団と探索用の子母集団を使い，探索で発見した優れた解の消失を防ぐ手法である．手順は次のようになっている．世代 t において，親母集団をP_t，子母集団をQ_t，P_tとQ_tを合わせた合成集団をR_tとする．P_tからQ_tを選択し，選択されたQ_tに選択，交叉，突然変異という遺伝的操作を加えてQ_tを更新する．そしてQ'_tと親母集団P_tを組み合わせて R_tを生成する．R_tから P_{t+1}を選択し，次の世代へと移行する．

4.1.3 参照点による選択

参照点とはパレートフロント上に規則的に配置され，最も近い個体が選択される点のことである．参照点の数Hはおおよそ可能解の数と同等とし，目的関数の数を M，任意の数をpとして $H = {}_{M+p-1}C_p$ で表される．これにより，目的関数の数が増加したとしても，パレートフロント上で 広範囲かつ均等に最適解群が得られ，かつ計算時間は短くて済む．

Input: H reference points Z^r, parent population P_t
Output: P_{t+1}
1: $S_t = \emptyset, i = 1$
2: Q_t = Recombination+Mutation(P_t)
3: $R_t = P_t \cup Q_t$
4: $(F_1, F_2, \ldots)$ = Non-dominated-sort(R_t)
5: **repeat**
6: $S_t = S_t \cup F_i$ and $i = i + 1$
7: **until** $|S_t| \geq N$
8: Last front to be included: $F_l = F_i$
9: **if** $|S_t| = N$ **then**
10: $P_{t+1} = S_t$, break
11: **else**
12: $P_{t+1} = \cup_{j=1}^{l-1} F_j$
13: Points to be chosen from F_l: $K = N - |P_{t+1}|$
14: Normalize objectives
15: Associate each member $\mathbf{s}$ of S_t with a reference point: $[\pi(\mathbf{s}), d(\mathbf{s})]$ =`Associate`(S_t, Z^r) % $\pi(\mathbf{s})$: closest reference point, d: distance between $\mathbf{s}$ and $\pi(\mathbf{s})$
16: Compute niche count of reference point $j \in Z^r$: $\rho_j = \sum_{\mathbf{s} \in S_t/F_l} ((\pi(\mathbf{s}) = j) \ ? \ 1 \ : \ 0)$
17: Choose K members one at a time from F_l to construct P_{t+1}: `Niching`$(K, \rho_j, \pi, d, Z^r, F_l, P_{t+1})$
18: **end if**

図 3：NSGA-Ⅲアルゴリズムの手順（[7]より）

5. 実験

今回提案した目的関数と最適化手法の妥当性を確認する．

5.1 パラメータの設定

今回，AWS の VM のメタデータを用いたが，各々の VM の性能は，AWS の m3.large における ApachBench の平均サービス率の結果を基点として，平均サービス率を計算している．表 1 では遺伝的アルゴリズム，パフォーマンス（スループット，レスポンスタイム，棄却率），動作率のパラメータを設定した．

遺伝的アルゴリズム	③④⑤パフォーマンス	⑥アベイラビリティ	⑦複雑性
世代数:1000世代	顧客数:100人	p(w),p(a),p(d):0.001	w_1: 0.5
個体数:1000個体	思考時間:7.5s	$p(w_k)$:0.001	w_2: 0.5
交叉確率:0.01	到着率:1req/s	$p(w_k\|Az_{JP})p(Az_{JP})$: 0.008	
突然変異確率:0.03	平均サービス率:3.318/vCPU	$p(w_k\|Az_{US})p(Az_{US})$: 0.04	
	棄却上限:100req	$p(w_k\|Az_{EU})p(Az_{EU})$: 0.03	

表 1 ：実験で用いたパラメータ

5.3 実験

上記のパラメータにおいて，最終個体群と選択された解群の評価値を得た．次の図 4 ～ 9 の小さな+(青色)は最終個体群を示し，大きな×(黄色)は選択された解群（上位 1 0 個体)を示す. 横軸は全てオペレーションコストで統一した．

図 4 は縦軸がコンフィグレーションコストのグラフである．オペレーションコスト，コンフィグレーションコストともに最小化の目的関数であるから，左下に向かって解が最適化されていくはずである．選択された解群は左下に集中しており，この 2 つの目的関数において最適解が選択されていることが分かる．

図 6， 7， 9 の最小化される目的関数では図 4 と同様に考えられる．左下に最終個体群が最適化されてゆき，左下から選択されれば良い．これらのグラフは選択されるべきところから選択されていることが分かった．

また図 5， 8 の最大化される目的関数では，左上に最終個体群が最適化されてゆき，左上から選択されていれば良い．これらのグラフでも選択されるべきところから選択されていることが分かった．

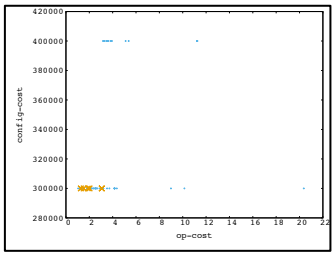

図 4 ：コンフィグレーションコストの最終個体群と選択された解群

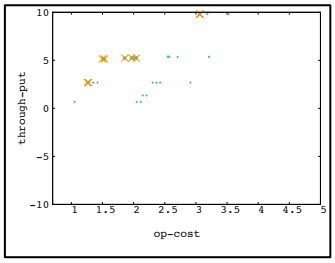

図 5 ：スループットの最終個体群と選択された解群

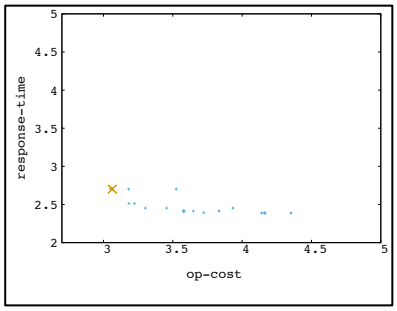

図６：レスポンスタイムの最終個体群と選択された解群

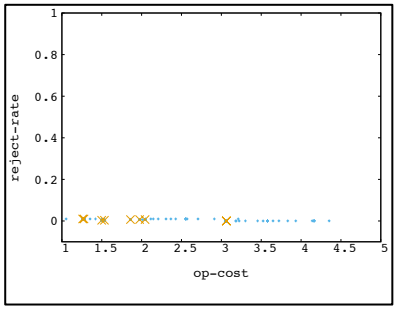

図７：棄却率の最終個体群と選択された解群

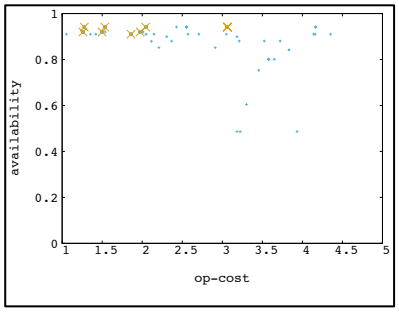

図８：動作率の最終個体群と選択された解群

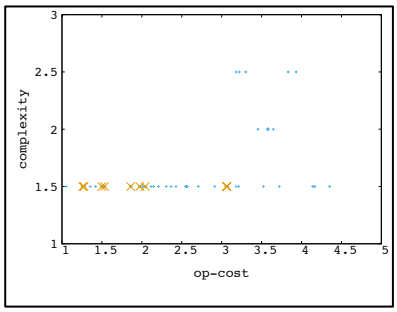

図９：複雑性の最終個体群と選択された解群

次に選択された解群の特徴を考察する．図１０は選択された解の中でランク１に属する解の一例である．サーバ構成が１台ずつであるのは，コンフィグレーションコストの目的関数と複雑性の目的関数に効いているからであろう．しかしそれでは動作率が低減してしまう．そのためサーバ設置地域に Az レベルでの動作率が最も高い(ap-northeast-1, 東京)が選ばれたのだろう．パフォーマンスに関する３つの目的関数に対しては 16vCPU をもつ c3.4xlarge が選択されており，高パフォーマンスが担保されている．以上のように７つの目的関数が同時に解決されていることが分かった．

```
Webサーバ : [{location=ap-northeast-1, type=c3.4xlarge}]
Appサーバ : [{location=ap-northeast-1, type=c3.4xlarge}]
DBサーバ : [{location=ap-northeast-1, type=c3.4xlarge}]
```

図１０：選択された解群の例

6. おわりに

本論文では，クラウドサービス選択問題の解決策となるクラウドブローカーサービスに適した目的関数と数理モデルを提案し，実験において目的関数と最適化手法の妥当性を確認した．

謝辞

本稿作成にあたりましては，SCSK 株式会社様との共同研究「インタークラウドにおける多目的最適化手法を用いた IT サービスの継続的最適配置」において多くの助言を頂きました．ここに感謝を申し上げます．

参考文献

1) Garg, Saurabh Kumar, Steve Versteeg, and Rajkumar Buyya. "A framework for ranking of cloud computing services." Future Generation Computer Systems29.4 (2013): 1012-1023.

2) Sundareswaran, Smitha, Anna Squicciarini, and Dan Lin. "A brokerage-based approach for cloud service selection." Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. IEEE, 2012.

3）川勝崇史，棟朝雅晴．"分散クラウド環境における SLA を考慮した WEB システムの多目的資源割当最適化"．情報処理学会研究報告，Vol. 2013-MPS-96, No. 9, pages 1-6, 2013.

4) P. Pawluk, B. Simmons, M. Smit, M. Litoiu, and S. Mankovski. "Introducing stratos: A cloud broker service". IEEE CLOUD '12, pages 891–898, 2012.

5) 原田雅史，河村美嗣．"待ち行列ネットワークモデルを使用した Web3 階層システムの性能予測." 全国大会講演論文集 72 (2010): 289-290.

6) Urgaonkar, Bhuvan, et al. "An analytical model for multi-tier internet services and its applications." ACM SIGMETRICS Performance Evaluation Review. Vol. 33. No. 1. ACM, 2005.

7) Jain, Himanshu, and Kalyanmoy Deb. "An improved adaptive approach for elitist nondominated sorting genetic algorithm for many-objective optimization." International Conference on Evolutionary Multi-Criterion Optimization. Springer Berlin Heidelberg, 2013.