社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－AL－110 (8)
2007／1／23

コールアドミッションアルゴリズムに適した経路探索法の提案

山本 正樹[1] 上土井 陽子[2] 若林 真一[2]
広島市立大学大学院 情報科学研究科[1]
広島市立大学 情報科学部[2]

概 要

コール (通信要求) が頻繁に発生し，ネットワーク容量を超えてしまうようなネットワークでは，QoSを保証するために新しいコールを受理すべきか判断する必要がある．QoSを保証しつつ，コールを受理して得られる利益が高まるように，コールの受理または拒否を決定するアルゴリズムをコールアドミッションアルゴリズムと呼ぶ．本稿では，オンラインに発生するコール系列を入力とするコールアドミッションアルゴリズムの１つである$SAAP\mu$について考察し，コールアドミッションに適した効率的な経路探索手法を提案する．提案手法を用いた$SAAP\mu$とDijkstra 法を経路探索手法とする$SAAP\mu$をC 言語で実現し，提案手法を用いた$SAAP\mu$と従来手法を用いた$SAAP\mu$を比較し，提案手法の有効性を示す．

A QoS Routing Method for Online Call Admission

Masaki Yamamoto[1] Yoko Kamidoi[2] Shin'ichi Wakabayashi[2]
Graduate School of Information Sciences, Hiroshima City University[1]
Faculty of Information Sciences, Hiroshima City University[2]

Abstract

On an online network with limited edge capacities, we must decide whether or not to accept a new call to guarantee quality-of-service. A call admission algorithm decides whether or not to accept calls to guarantee quality-of-service and obtain many profits by accepting calls. We study the call admission algorithm $SAAP\mu$, which receives calls generated in online as its inputs, and propose an effective routing method for the call admission. We implemented $SAAP\mu$ with the proposed method and Dijkstra's algorithm with the C language, and compared $SAAP\mu$ with Dijkstra's algorithm with $SAAP\mu$ with the proposed routing method. Experimental results show the effectiveness of the proposed routing method.

1 はじめに

近年，インターネット利用者の爆発的な増加に伴いブロードバンド化が推進され，IP 電話，TV 電話，TV 会議のような帯域幅を大きく占有するデータをインターネット上で伝送するようになっている．利用者の増加と大容量データの伝送によって，すべてのコール (通信要求) を受け入れることは回線の混雑によるQoSの悪化につながる可能性がある．大容量データのコールが頻繁に発生し，ネットワーク容量を超えてしまうようなコールが発生するネットワークでは，QoSを保証するために新しいコールを受理すべきか決定する必要がある．QoSを保証しつつ，コールを受理することによって得られる利益の合計を高めるためにコールの受理，拒否を決定することをコールアドミッションと呼ぶ．また，コールが受理された場合，ネットワーク中の１つのルートをコールに割り当てるというルーティングを行う必要がある．本稿では，コールアドミッション/ルーティ

ング問題を単にコールアドミッション問題と呼ぶ．オンラインでコールが発生するオンラインコールアドミッション問題では，あらかじめすべてのコールの情報，発生時刻を知ることができないため，コールが発生したときのネットワークの状況に基づいてコールの受理，拒否を決定しなければならない．

本稿では，コールの受理，拒否を決定するコールアドミッションアルゴリズムの1つである$SAAP\mu$[1]について考察する．$SAAP\mu$の特徴は，枝の負荷の増加に応じて指数関数的に増加するコストを枝に割り当て，経路の枝コストの総和に上限(制約コスト)を設定し混雑している経路を避けることである．ここで，我々は制約コストを満たし，かつ **(a)** 枝コストの総和が最小,**(b)** ホップ数が最小，という2つの経路選択基準を考えた．基準(a)では，経路探索手法としてDijkstra法を用いる(従来手法)．本稿では，基準(b)を満たす経路を探索するためのアルゴリズムを提案する．提案経路探索法は，以前に探索した始点から各節点への全ての経路の中で枝コストの総和が最小となる経路を保持し，このコストを超える経路への探索と，制約コストを超える経路への探索を制限し条件付幅優先探索を行うことで，始点から終点へのコスト制約を満たす最小ホップ数の経路を探索する．提案経路探索法の有効性を示すために，提案手法を用いた$SAAP\mu$とDijkstra法を経路探索手法とする$SAAP\mu$を比較することで，実行時間とコールの受理，拒否が効率的に行われているかを実験的に評価する．また，提案経路探索法が制約コストを満たす経路の中で最小ホップ数の経路を求めることを証明する．

2 コールアドミッション

オンラインコールアドミッションでは，受理されたコールの利益の合計を高めることを目的に逐次的に発生するコールに対して，コールが発生したときのネットワークの状況からコールの受理，拒否を決定する．本稿におけるネットワークは節点数がnである無向グラフとし，各枝eは容量$u(e)$を持つとする．本稿では，基本的な場合を考えるためにコールの存続期間は無限と仮定し，i番目に発生したコールを$r_i = (s_i, t_i, b_i, p_i)$と表し，$s_i$はコールの始点，$t_i$は終点，$b_i$は$r_i$のバンド幅要求，$p_i$は$r_i$を受理することによって得られる利益とする．本稿ではb_iを1に固定する．

コールが発生したとき，受理，拒否を決定するためにネットワーク中の各枝eにどれくらい負荷がかかっているか計算する必要がある．そこで，j番目のコールを処理したときの枝eに対する負荷率$L_j(e)$を以下に定義する．

$$\mathrm{L}_j(e) = \frac{1}{u(e)} \sum_{\substack{k \in A_j \\ e \in P_k}} b_k \qquad (1)$$

ここで，A_jはj番目までに受理されたコールの添字の集合，b_kはk番のコールのバンド幅要求とし，P_kは$k \in A_j$となるコールに割り当てられた経路を表す．

コールアドミッション問題とは，$L_{j-1}(e)+(b_j/u(e)) < 1$を満たすようにコールの受理，拒否を決定し，受理したコールの利益の合計を最大にすることが目的である．本稿では，各コールの利益p_iを一定値pと仮定する．このとき，拒否率を最小にすることがコールの利益の合計を最大にすることを意味する．ここで，拒否率は(拒否したコール数/コールの総数)と定義する．

3 オンラインアルゴリズム$SAAP\mu$

コールを受理して得られる利益を高めるためにコールの受理，拒否を決定するアルゴリズムをコールアドミッションアルゴリズムと呼ぶ．本稿では，コールアドミッションアルゴリズムの1つである$SAAP\mu$[1]について考察する．

$SAAP\mu$は，コールが発生するとコールの受理または拒否を決定し，受理する場合はコールの始点と終点間の経路を決定する．$SAAP\mu$の特徴は，枝の負荷の増加に応じて指数関数的に増加するコストを枝に割り当て，始点と終点間の経路上の枝コストの総和が式(2)を満たし，経路上のすべての枝が式(3)を満たす経路を選択することである．本稿では，経路上の枝の本数をホップ数とし，枝の容量がコールの要求するバンド幅以上空いているならホップ数1のコールを受理するものとする．つまり，式(2)よりコールの利益を$p = b\mu$と設定する．

$$\sum_{e \in P} b_j \mu^{L_{j-1}(e)} \leq p_j \quad (2)$$

$$\mathrm{L}_{j-1}(e) + \frac{b_j}{u(e)} < 1 \quad (3)$$

コールアドミッション問題において，コストは枝容量の空いている度合いを表し，ホップ数は枝容量を帯域幅の分だけ占有する枝の数を表している．コストが低く，ホップ数の少ない経路を優先することは後に発生するコールの割り付けを効率的にすると考えられる．ここで，式 (2)，(3) を満たす経路の中で，**(a)** 枝コストの総和が最小，**(b)** ホップ数が最小，という 2 つの経路選択基準を考える．基準 (a) ではホップ数を考えていないため，枝コストの総和が最小であるがホップ数の長い経路を選択している可能性がある．ホップ数の長い経路はホップ数の短い経路に比べてより多くの枝容量を占有してしまう．そこで，ホップ数に制約を加えた基準 (b) を考えた．基準 (a) では，経路探索手法として Dijkstra 法を用いる (従来手法)．本稿では，基準 (b) を満たす経路を探索するためのアルゴリズムを提案する．

4 提案経路探索

4.1 条件付ホップ数最小経路探索問題

コールアドミッションでの経路探索

入力ネットワーク $N = (V, E)$ の各枝 e には容量 $u(e)$ と負荷が定義されているとする．ネットワーク N 上でのコールアドミッション問題に対し，$SAAP\mu$ では，枝の負荷の増加に応じて指数関数的に増加するコストを割り当て経路探索を行う．

条件付ホップ数最小経路探索問題

以下では，上記問題を枝に空き容量とコストをもつネットワーク $G = (V, E)$ での経路探索問題として考える．ここで，コール r_j が発生したとき，

$$\sum_{e \in P} b_j \mu^{L_{j-1}(e)} \leq p$$

上式を満たす，つまり，上式右辺の p を制約コストとし，$b_j \mu^{L_{j-1}(e)}$ を各枝 e の枝コストとするとき，以下の条件を満たすコール r_j の始点 s_j から終点 t_j までの経路 P を探索することを目的とする．条件 3 を満たす経路が存在する場合は経路を出力とし，条件 1, 2 を満たす経路がない場合は経路が存在しないことを出力とする．

条件 1 経路の枝コストの総和が制約コスト C_{limit} を超えない．

条件 2 経路の各枝の容量が b_j 以上空いている．

条件 3 条件 1, 2 を満たす経路の中で最小ホップ数の経路．

コールアドミッションの本来の問題における，枝に容量 $u(e)$ と負荷を持つネットワーク N を条件付ホップ数最小経路探索問題における，枝に空き容量とコストをもつネットワーク G への変換した例を図 1 に示す．また，枝に空き容量とコストをもつネットワーク G と 8.0 のコスト制約において，節点 a から節点 e への枝コストの総和が最小経路の例を図 2 に示し，条件 1，2，3 を満たす経路の例を図 3 に示す．

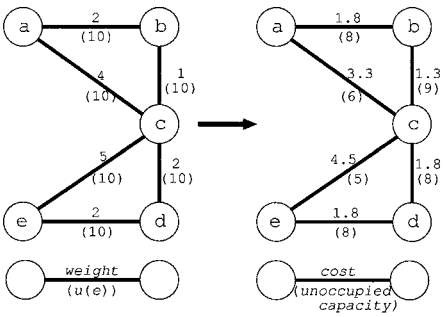

図 1: ネットワークの変換例

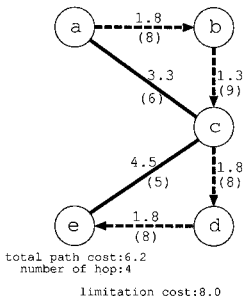

図 2: コスト最小経路

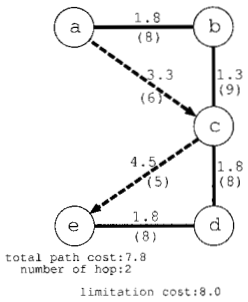

図 3: コスト制約付ホップ数最小経路

以降では，節点 s は始点，節点 t は終点，$EdgeC(v,w)$，$u(v,w)$ は節点 v，w 間の枝のコストと枝容量とする．

4.2 提案経路探索アルゴリズム

本節では，条件付ホップ数最小経路探索問題を解く条件付幅優先探索アルゴリズムを提案する．提案アルゴリズムは，以前に探索した始点から各節点への経路の中で枝コストの総和が最小のコストの経路のコストを保持し，このコストを超える経路への探索と，制約コストを超える経路への探索を制限し条件付幅優先探索をすることで，始点から終点へのコスト制約を満たす最小ホップ数の経路を探索する．

提案アルゴリズムでは，ホップ数 i の始点 s から各節点 v への制約コスト C_{limit} を超えない探索経路中で最も小さいコストを持つ経路の枝コストの総和をコスト関数 $Cost(i,v)$ へ記録し，以前に探索した始点 s から各節点 v への経路の中の枝コストの総和の最小値を配列 $min[v]$ へ記録する．節点名，ホップ数の二項組を要素とし，後に延長される候補となる経路を表現する要素を FIFO 型のデータ構造キュー Q に保持する．また，キュー Q へ要素を入れるとき，同じ要素がすでにキュー Q へ入っている場合は古い要素を削除する．

条件付幅優先探索アルゴリズム

入力：枝 e に容量 $u(e)$ とコスト $EdgeC(e)$ をもつネットワーク $N=(V,E)$，始点 s，終点 t，コスト制約 C_{limit}

出力：始点 s から終点 t へのコスト制約 C_{limit} を満たす最小ホップ数の経路，またはこの条件の経路が存在しなければ，経路が存在しないこと

初期設定

- $min[v] \leftarrow \infty$ $(\forall v \in V-\{s\})$，$min[s] \leftarrow 0$
- $Cost(i,v) \leftarrow \infty$ $(\forall v \in V-\{s\}, i=0,1,\ldots,n$ のすべての組合せに対して)
- $Cost(0,s) \leftarrow 0$
- $Q \leftarrow \emptyset$

STEP1: $i \leftarrow 0$，$u \leftarrow s$，

STEP2: 節点 u に隣接する各節点 w に対して，$C_{limit} \geq Cost(i,u) + EdgeC(u,w)$，かつ $min[w] > Cost(i,u) + EdgeC(u,w)$ を満たすならば，1，2，3 を実行する．

1. $Cost(i+1,w) \leftarrow Cost(i,u)+EdgeC(u,w)$，$min[w] \leftarrow Cost(i,u)+EdgeC(u,w)$
2. 要素 $(w,i+1)$ をキュー Q へ入れる．
3. 節点 w が t なら ***STEP6*** へ．

STEP3: 終了判定：キュー Q が空なら求める経路が存在しないことを出力し，終了．

STEP4: キュー Q の先頭の要素のホップ数が i ならば，キュー Q の先頭の要素 (i,v) を取り出し，$u \leftarrow v$ とし ***STEP2*** へ．
キュー Q の先頭の要素のホップ数が i でなければ ***STEP5*** へ．

STEP5: $i \leftarrow i+1$，***STEP4*** へ．

STEP6: 節点 t に隣接する節点で $Cost(i,u) + EdgeC(u,w)$ が最小となる節点に対し，$Cost(i+1,t) \leftarrow Cost(i,u)+EdgeC(u,t)$ とし，$Cost(i+1,t)$ からホップ数とコストを参照して始点 s の $Cost(0,s)$ まで辿ることによって経路を求め出力し，終了．

4.3 提案経路探索アルゴリズムの正当性の証明

本節では以下の定理を 2 つの補題に分けて証明する．

定理 1：提案経路探索アルゴリズムは入力が各枝 e に容量 $u(e)$ とコスト $EdgeC(e)$ をもつネットワーク $N=(V,E)$ と始点 s，終点 t，制約コスト C_{limit} が与えられたとき，始点 s から終点 t へのコスト制約を満たす経路が存在するならばそれらのうちの最小ホップ数をもつ経路の 1 つを出力し，存在しなければ経路が存在しないことを出力する．

証明

定理 1 を証明するため，以下の補題 1，補題 2 を証明する．補題 2 は枝コストの総和がコスト制約を満たす始点 s から各節点 v への経路の中でホップ数 l を持つ最小コスト経路のコストが $P(l,v)$ であるならば，提案探索アルゴリズムでは $Cost(l,v)=P(l,v)$ となることを示している．終点 t の $Cost(l,t)$ からホップ数とコストを参照して始点 s の $Cost(0,s)$ まで辿ることによって経路を探索することができる．

補題 1：$Cost(k,v)=Cost \Rightarrow$ ホップ数 k の始点 s から節点 v へのコスト制約を満たし，かつ，枝コストの総和が $Cost$ である経路がネットワーク N 上に存在する．

証明

アルゴリズムの **STEP2** より，節点 $v=v_k$ に隣接し，かつ $Cost(k-1,v_{k-1})=Cost-EdgeC(v_k,v_{k-1})$ となる節点 v_{k-1} が存在しなければならない．同様に v_{k-i} に隣接し，かつ $Cost(k-i-1,v_{k-i-1})=Cost-\sum_{j=0}^{k-1} EdgeC(v_{k-j},v_{k-j-1})$ となる節点 v_{k-i-1} が存在しなければならない．アルゴリズムの初期設定と動作より明らかに，$Cost(0,v_0)=0$ であり，始点 s はコスト関数が 0 となる唯一の節点であるから $v_0=s$ である．このようにして，$v_0-v_1-\ldots-v_{k-1}-v_k$ はホップ数 k の始点 s から節点 v へのコスト制約を満たし，かつコストが $Cost$ である経路である．

補題 2：提案探索アルゴリズムの試行において $i=l$ として ***STEP5*** に入ったとき，以下の命題が満たされる．ホップ数 l 以下の始点 s から節点 v へのコスト制約を満たす経路の中で最小コストの経路のコストを $MC_{l,v}$ とし，かつホップ数 l 未満の始点 s から節点 v へのコスト制約を満たす経路の枝コストが $MC_{l,v}$ 以下でないとき，$P(l,v)=MC_{l,v}$，そうでないとき $P(l,v)=\infty$ である．$\Rightarrow Cost(l,v)=P(l,v)$

証明

以降ではホップ数 l と節点 v を変数としたとき，ネットワーク N で，ホップ数 l 以下の始点 s から節点 v へのコスト制約を満たす経路中で最小コストの経路のコストを $MC_{l,v}$ としたとき，ホップ数 l 未満の始点 s から節点 v へのコスト制約を満たす経路の枝コストが $MC_{l,v}$ 以下でない，という述語を $X(l,v)$ で表す．

ホップ数 l と節点 v が与えられたとき，命題 $X(l,v)$ が成り立たないならば $P(l,v)=\infty$ である．このとき，提案アルゴリズムでは $Cost(l,v)$ はそれまでに探索した始点 s から節点 v への経路のコストの最小値 $min[v]$ 未満のコストを持つ経路においてのみ ***STEP2*** の 1 を実行し，幅優先で経路を探索することにより，命題 $X(l,v)$ を満たさないとき，$Cost(l,v)$ は更新されず ∞ のままである．よって，命題 $X(l,v)$ が成り立たないとき，補題 2 は成り立つ．以降ではホップ数 l と節点 v において，命題 $X(l,v)$ が成り立つ場合に補題 2 が成り立つことをホップ数 l の帰納法により証明する．

STEP1 で $Cost(0,s)=0$ となる．また，$P(0,s)=0$ であることより基本ステップが成り立つ．実際，始点 s から始点 s へのコストは 0 である．補題 2 が小さなホップ数で成り立つと仮定し，その仮定がホップ数 l のときも満たすことを示す．$s-v_1-v_2-\ldots-v_{l-1}-v$ は，命題 $X(l,v)$ を満たすアルゴリズムで最初に探索される経路とし，そのコストは $P(l,v)$ である．明白に，$s-v_1-v_2-\ldots-v_{l-2}-v_{l-1}$ は，$X(l-1,v_{l-1})$ を満たし，そのコストは $P(l,v)-EdgeC(v_{l-1},v_l)$ であり，このとき，$P(l-1,v_{l-1})=P_{l,v}-EdgeC(v_{l-1},v_l)$ である．それぞれの帰納的な仮定により，$Cost(l-1,v_{l-1})=P(l-1,v_{l-1})$ である．$i=l-1$ のとき，***STEP2*** では，節点 v は $Cost(l,v)+EdgeC(v_{l-1},v_l)=P(l,v)$ という値を受けとる．この後，もし，節点 v，ホップ数 l において $P(l,v)$ 未満のコストで $Cost(l,v)$ が値づけられていたなら，補題 1 よりホップ数 l の始点 s から節点 v へのコスト制約を満たし，かつ

コストが $P(l,v)$ 未満の経路が存在することになり，$P(l,v)$ の定義に矛盾する．もし，$Cost(l,v)$ の値の更新が行われていないとすると，***STEP2*** で節点 v が $Cost(l,v)+EdgeC(v_{l-1},v_l)$ なる値を受けとったときに，すでに $min[v] \leq P(l,v)$ が成り立っていることになる．したがって，経路 $s-v_1-v_2-\ldots-v_{l-1}-v$ の仮定から，ホップ数 l 未満の s から v へのコスト制約を満たし，かつコストが $P(l,v)$ 以下の経路が存在することになり命題 $X(l,v)$ が成り立つという仮定に矛盾する．よって，$Cost(l,v)=P(l,v)$ である．

5 実験的評価

従来手法の経路探索法と提案手法の経路探索法を用いた $SAAP\mu$ をそれぞれ C 言語で実現し，シミュレーション実験を行った．入力ネットワークとして，2 次元グリッド上に節点を発生させ，2 節点間の距離に反比例する確率関数によって枝を発生させたランダムグラフ [4] を用いた．また，始点，終点に対応する 2 節点間の距離に比例する確率を与えてコールの到着間隔とし，ポアソン過程を用いて発生させたコールの系列を入力とした [3]．実験で用いた計算機の CPU は UltraSPARC-IIIi 1062MHz，メモリは 2GB である．

実験 1 では節点数とネットワークの総容量を固定し，枝密度を変化させ，従来手法と提案手法の実行時間と拒否率を比較した．実験 2 ではコスト制約を満たす経路を発見しても，発見した経路と同じホップ数の間は探索すると終了条件を変更した提案手法の評価を行った．

5.1 実験 1

100×100 のグリッドを用いて節点数 100 個と枝密度に応じた枝数を発生させランダムグラフ [4] を作成した．ネットワーク総容量を約 10000 に固定し，コール数を 10000 個として，枝密度を変化させ実行時間と拒否率を計測した．実験結果を表 1，2 に示す．ただし，従来手法 (Dijkstra 法) において，始点 s から任意の節点への経路の最小コストを保持するためにヒープ木を用いている．節点名，経路の枝の合計コストの組をヒープ木の要素とし，ヒープ木の根は常にコストが最小の要素を持っている．従来手法における，in は，ヒープ木に保持した要素の総数，out はヒープ木から取り出した要素の総数とする．提案手法における，in は，キューに保持した要素の総数，out はキューから取り出した要素の総数とする．また，$Time$ は実行時間を表している．

表 1: 従来手法の枝密度に対する拒否率と実行時間

枝密度 [%]	拒否率	$Time$[s]	in	out
10	0.657	0.88	295,589	159,138
20	0.505	2.46	472,740	250,745
40	0.408	6.39	620,771	345,986
60	0.392	7.10	610,689	319,493
80	0.385	7.68	602,433	330,947

表 2: 提案手法の枝密度に対する拒否率と実行時間

枝密度 [%]	拒否率	$Time$[s]	in	out
10	0.673	0.70	224,245	64,987
20	0.505	1.45	364,125	102,867
40	0.406	2.43	533,172	132,496
60	0.373	2.61	562,661	125,683
80	0.355	2.83	636,877	113,776

表 1,2 より，提案手法は従来手法と比べて，枝密度が高くなるほどコールの拒否率を低く抑え，また計算を 1.2〜2.7 倍に高速化することができることがわかった．in，out の値が従来手法より提案手法のほうが小さいことから，計算が高速であったのは従来手法より提案手法のほうが探索する要素数が少なかったことが原因だと考えられる．両手法とも，枝密度が高くなると実行時間が増加するが従来手法の方が計算時間の増加率が大きく，ネットワーク規模が大きくなるほど実行時間の差が大きくなると考えられる．

5.2 実験 2

実験 1 の結果から，枝密度 10% の場合は従来手法と比べ提案手法のほうが拒否率が高いことがわかる．拒否率を低くするために，コスト制約を満たす経路を発見しても，発見した経路と同じホップ数の間は探索すると終了条件を変更した提案手法について実験を行った．従来手法と提案手法，終了条件変更後の提案手法それぞれにおいて，枝密度 10%，枝 1 本の容量を 10，コール数を総容量と同数としたとき，節点数を変化させ拒否率と実行時間を計測した．実験結果を表 3，4，5 に示す．

表 3，4，5 から，改良後の提案手法は従来手法と

表 3: 従来手法の節点数に対する拒否率と実行時間

節点数	拒否率	*Time*[s]	*in*	*out*
100	0.799	0.25	85,295	47,049
150	0.631	2.30	575,095	318,201
200	0.528	9.68	1,717,178	950,438
250	0.520	23.8	3,465,228	1,902,272
300	0.499	51.51	6,313,553	3,424,950

表 4: 提案手法の節点数に対する拒否率と実行時間

節点数	拒否率	*Time*[s]	*in*	*out*
100	0.800	0.19	65,322	17,455
150	0.642	1.35	394,340	97,863
200	0.538	4.62	1,312,216	273,095
250	0.520	11.0	2,671,295	544,818
300	0.510	26.6	5,250,004	1,170,489

表 5: 終了条件変更後の提案手法の節点数に対する拒否率と実行時間

節点数	拒否率	*Time*[s]	*in*	*out*
100	0.799	0.33	104,969	26,116
150	0.634	2.76	664,668	185,996
200	0.513	10.4	2,356,797	472,651
250	0.500	24.1	4,959,296	843,916
300	0.481	56.1	10,259,556	1,259,980

比較して，同程度の実行時間で，節点数 200，250，300 で拒否率の低い経路割当てを行っていることが確認できた．

6 おわりに

本稿ではコールアドミッションアルゴリズムの 1 つである $SAAP\mu$ を取り上げ，経路探索手法を提案し，Dijkstra 法と比較することで提案経路探索手法の有効性を示した．また，提案探索アルゴリズムはコスト制約を満たし，かつ，ホップ数最小の経路を探索可能なことを証明した．今後の課題は，より大きなネットワークにおいて提案手法を評価すること，コールの存続期間が有限な場合の評価である．

参考文献

[1] A. Borodin and R. El-Yaniv, "Online Computation and Competitive Analysis," *Cambridge University Press*, pp.226–237, 1998.

[2] Shimon Even, "Graph Algorithms," *Technion Institute, Computer Science Press*, 1979.

[3] S. M. Ross, "Simulation: Second Edition," *Academic Press*, pp.74–77, 1996.

[4] L. Wei and D. Estrin, "The Trade-offs of Multicast Trees and Algorithms" *Computer Science Department University of Southern California*, pp.1–15, 1995.