情報処理学会論文誌 Vol.53 No.2 857–867 (Feb. 2012)

# 時間制約を持つ寄り道経路探索システムの実現と評価

鈴木 源吾[1,a)] 榎本 俊文[1] 小林 伸幸[1] 山室 雅司[1] 鬼塚 真[1]

受付日 2011年5月30日, 採録日 2011年11月7日

**概要**：近年，カーナビ，インターネット等で地図検索，特にルート探索等のサービスが多数提供されている．本論文は，これらのルート探索において，経由地に時間制約があるような最短経路探索問題である「タイムセール寄り道探索」の概念を示すとともに，その解を求める手法を提案する．従来の寄り道探索手法をそのまま利用できる「基本導出法」を示し，その課題を指摘し，さらに，グラフの探索を行いつつ制約条件をチェックし解を導出する「動的導出法」を提案した．動的導出法は，グラフの探索範囲と候補となる解の個数を抑制し，性能を改善させることを特徴に持つ．グラフデータベース上に構築した実験システムを用いて提案方式の比較評価を行い，動的導出法が基本導出法に比べて，特にサービス密度（サービスを実施しているノードの割合）が低い場合に性能的に優れており，探索対象が大規模となる場合において適用性が高いことを示した．

**キーワード**：グラフデータベース，経路探索

# Time Constrained Trip Planning Search System

Gengo Suzuki[1,a)] Toshifumi Enomoto[1] Nobuyuki Kobayashi[1]
Masashi Yamamuro[1] Makoto Onizuka[1]

Received: May 30, 2011, Accepted: November 7, 2011

***Abstract:*** We propose and formalize “time-sale trip planning search”, which is a variation of shortest path problem. This is a trip planning which includes time-constrained services. We clarify “basic method” of this search, which is a simple extension of existing trip planning method, and “dynamic method” which can restrict search range and number of candidate answers using time constraints. We implemented these methods on a graph database system, and showed that dynamic method has advantages in performance under low service density (ratio of nodes in service), so that can be applicable for very large graph databases.

***Keywords:*** graph database, route search

## 1. はじめに

近年，カーナビゲーション，スマートフォン，地図サービス等の普及により，旅行計画作成に関するニーズが増大しており，様々なサービスが実用化されている．最も単純な旅行計画作成の例は，最短経路探索であり，鉄道網や道路網から所望のルートを探索するサービスに活用されている．これは，ノード間の移動コスト（例：所要時間）が与えられたグラフにおいて，合計コストが最小となる最短のルートを求める問題であり，ダイクストラ法，A*法等のグラフ探索技術が確立している [1], [2], [3].

一方，ユーザの位置情報に応じて，お勧めの場所を推薦するレコメンドサービスがある．ユーザの位置（GPS 等により取得）の近くにあり，ユーザの希望，旨向にあった店を推薦したり，目的地までのルートを探索したとき，そのルート沿いのガソリンスタンドやファーストフード店を推薦したり等，交通やグルメ等のインターネット上の検索サービスの分野 [4], [5] や，カーナビ等のナビゲーションシステムで用いられている．

また，上記サービスに関係するが，少し目的が違う，次のような要求もある．たとえば，ユーザがどこかに出張す

1 NTT サイバースペース研究所
NTT Cyber Space Laboratories, Yokosuka, Kanagawa 239–0847, Japan
a) suzuki.gengo@lab.ntt.co.jp

© 2012 Information Processing Society of Japan

るときに，必ずしも最短経路でなく，多少遠回りしてもよいから，そのルート上で自分の好みの昼食をとってから目的地に到達したい，という要求である．この要求を満たす探索を「寄り道探索」と呼ぶことにする．このとき経由地を POI（Point of interest）と呼ぶ．ユーザからは，直接具体的な POI ではなく，「寿司が食べたい」とか「銀行に行きたい」というような行きたい場所のカテゴリで指定されることを想定する．具体的な POI が事前に決まっているのであれば，出発点–POI・POI–到達点の 2 つのルートを最短経路探索法で求めればよいので，技術的には既存の技術で解決できる．しかし，寄り道探索では，あらかじめ分からない POI を探索しつつ，できるだけ最短経路に近いルートを求めることが必要になる．

寄り道探索において，寄り道先に時間制約がある場合を考える．たとえば，開店時間には 12 時〜22 時のように制限がある．その開店時間に到着できるようなルートだけを結果としてほしい．たとえば，夜遅めに出発するときには，閉店に間に合うように出発点に近い POI に寄り道する必要がある．また，開店時間だけでなく，「12 時〜13 時，ランチ 2 割引」や「17 時以降，閉店前 3 割引」のような時間限定のタイムセールのときに店に行きたいという要求も同じ課題であるといえる．このような，時間制約を持つ寄り道探索を「タイムセール寄り道探索」と呼ぶ．上記では，開店時間等のサービスの時間制約を例としてあげたが，「その店で 2 時間ゆっくりしたい」「銀行でお金をおろすのでぎりぎりに間に合えばよい」というような，ユーザにとっての時間制約の要求もあり，これらも満たすような探索を行いたい．

まず，時間制約を満たす解を導出する方法が課題となる．通常の寄り道探索では総所要時間は，出発点から POI までの所要時間と，POI から到達点までの所要時間を単純に足せばよいが，タイムセール寄り道探索の場合，待ち時間も許容することがサービス的に有効である．待ち時間も含めたスケジュールを決定し，その総所要時間で最終的にランキングする必要がある．また，寄り道探索は POI を探すために，通常の最短経路探索以上に多くのノードを探索する必要があるため，できるだけノードの展開を減らし，探索速度を向上させることも課題である．

本論文では，大沢ら [6] で提案されている寄り道探索のための逐次拡大法をそのまま利用し，それにタイムセール寄り道探索解を導出する方法を組み合わせた基本導出法と，さらに時間制約を積極的にグラフ探索に利用する動的導出法を提案している．動的導出法は，探索範囲と解候補の個数を制限し性能を改善できる．

本論文は以下のように構成されている．まず，2 章で過去の研究との関連を示し，3 章でタイムセール寄り道探索を定義し，4 章で既存の寄り道探索の手法と基本導出法について述べる．5 章で動的導出法を示す．6 章で提案手法をグラフデータベースを用いて実装した実験システムとその評価を述べ，7 章で結論と今後の課題を述べる．

## 2. 関連研究

本論文で扱っているのは，グラフ上で POI を探索し，条件付きの最短経路を求める問題であり，グラフデータベース・空間データベースの分野との関連が深い．空間データベース検索の分野では，k-最近傍探索等の分野で類似の POI を探索する方式が提案されている [7], [8]．その代表例として，Sharifzadeh ら [7] による Optimal Sequenced Route 探索があげられる．これは，複数の異なる POI を指定された順序で探索するという一種の最近傍探索でありユークリッド距離を用いている．本論文のようなグラフ上の最短経路探索ではない．また，Optimal Sequenced Route 探索は準最適解であることに対し，本論文で提案する手法では最適解を求めることが可能である．**表 1** にその違いをまとめた．一方，ユークリッド距離ではなくグラフ上の距離で，最短経路探索等を効率良く求める技術の研究が行われており [9]，特にネットワークボロノイ図等を前処理で作成し効率化をはかる研究がさかんである [10]．しかし，本論文では頻繁に変更される時間制約があるような応用を想定する．たとえば，我々が想定している応用として，レストランやスーパマーケットのタイムセールの探索がある．これは，事前にスケジュールが決まっておらず，当日の在庫状況，顧客状況等によって実施が決定され，それを POI にいる従業員が twitter でつぶやくと探索の対象として登録されるというように，実時間性がある．その場合，前処理が必要であるとそのやり直しが頻繁に発生してしまうので適しておらず，前処理を必要としないことが望ましい．たとえば，Sharifzadeh ら [7] や蒲原ら [10] は，ネットワークボロノイ図を事前に作成しておく必要があり，時間制約を利用する問題に適さない．前処理を必要としない手法として，大沢ら [6] による逐次拡大法がある．これは，POI が 1 つである制約はあるが，前処理なしで寄り道経路探索を実現している．本論文ではその手法を拡張している．

時間の制約という意味では，時間概念を組み込んだ時制データベース [11], [12] が研究されているが，関係モデルや XML 等の一般的なデータモデルに時間概念を組み合わせ

**表 1** ルート探索手法の比較
**Table 1** Comparison of route search method.

| | Sequenced Route 探索 | タイムセール寄り道探索 |
|---|---|---|
| 利用する距離 | ユークリッド距離 | グラフ上の距離 |
| 出発点指定 | あり | あり |
| 到着点指定 | なし | あり |
| 寄り道先の数 | 複数 | 1 |
| 順序性 | あり | なし |
| 時間指定 | なし | あり |
| 解の最適性 | 準最適解 | 最適解 |

© 2012 Information Processing Society of Japan

る研究が主流になっている．本論文は，グラフデータベースに時間概念を取り込んでいることに特徴があるが，サービスの概念に特化しているという意味で，一般的な時制グラフデータベースとまではいえない．しかし，今後，一般的な時制グラフデータベースの研究を進めるうえでの最初の1歩であり，重要な応用例になるものと考える．

本論文では，ユーザやサービスによって指定された時間制約を満たす時間区間を求めているが，これは区間スケジューリング問題 [13] に関連がある．区間スケジューリング問題では，要求する区間に重なりがなく，できるだけ多く（長く）要求を実行できる区間を選ぶことが目標になるが，本論文ではPOIを1つに制約し，答えとなる時間区間の個数が3つに限られていることから，問題に特化した初等的な解法で時間区間を求めている．今後，POIを複数に拡張するときに，より一般的な区間スケジュール問題の成果を利用できる可能性がある．

ユーザの嗜好を反映したルートを提示するような研究としては，松田ら [14] があるが，「歩道のある道を歩きたい」等の要求に応じて，コストに重みを設定しルートを計算する方法であり，本論文で問題としているカテゴリに合致するような寄り道を行う探索には利用することができない．

グラフ情報を蓄積・検索する手段として，グラフのデータモデルを直接扱うことができるデータベース管理システムが一般的になりつつあり，グラフデータベースと呼ばれている [15]．実用的に利用できる製品も近年でてきている [16]．本論文でも，システムの実現と評価に利用している．

## 3. タイムセール寄り道探索

### 3.1 タイムセール寄り道探索の定義

以下にタイムセール寄り道探索を定義する．寄り道先は1カ所に限定している．$G = (V, E)$ を重みつきグラフとする．$V$ が頂点（ノード）集合，$E$ が辺（エッジ）集合である．エッジへの重みは移動にかかる時間を表すこととする．

**定義1（カテゴリ）** ノードの種別を表すラベルをカテゴリと呼ぶ．カテゴリが付与されたノードをPOIと呼ぶ．カテゴリは，寄り道先として指定するために使う「コンビニ」「銀行」「中華料理屋」等の指定を意味する．

**定義2（時間区間）** 時間区間を [開始時刻, 終了時刻] と表現する．以降の説明では，分単位で時刻と時間を表記し，時刻を表すときに年月日を省略する．

**定義3（サービス条件）** POIにおいてサービスを識別するラベルをサービス内容と呼ぶ（例：営業時間，ランチサービス，商品2割引）．サービス内容とそのサービスが実施される時間区間の組をサービス条件と呼ぶ．一般的にはサービス実施時間は1つの区間とは限らないが（例：昼と夜に営業），以降の解の導出法では1つのサービス実施は1つの区間であることを前提としている部分がある．一般化は今後の課題である，

**定義4（ユーザ条件）** 以下の5つのユーザ側の時間制約をユーザ条件と呼ぶ．

- 出発点条件：出発点を出発し探索を開始する時刻を含む時間区間
- POI開始条件：POIでサービスを開始する時刻を含む時間区間
- POI終了条件：POIでサービスを終了する時刻を含む時間区間
- 滞在長条件：POIでサービスを受ける時間の長さの最大値・最小値
- 到着点条件：到着点に到着し探索を終了する時刻を含む時間区間

**定義5（タイムセール寄り道探索）** カテゴリ・サービス条件が与えられている重みつきグラフ $G = (V, E)$ に対して，タイムセール寄り道探索とは，指定した出発点・到着点・カテゴリ・サービス内容・ユーザ条件・解の個数kに対して，以下の条件を満たすグラフ上のルートと結果となる時間区間であるS区間（出発点からPOIまで移動する時間区間），P区間（POIに滞在する時間区間），E区間（POIから到着点へ移動する時間区間）の組について，総所要時間（E区間の最大値 − S区間の最小値）が最小となる上位k組を求めることをいう．

- ルートは，指定した出発点を出発し，指定したカテゴリを付与されたPOIを経由し，指定した到着点に到着する．
- S区間の最小値は出発点条件に含まれる．
- S区間の長さは出発点からPOIまでの移動時間に一致する．
- P区間の最小値が，POI開始条件に含まれる．
- P区間の最大値が，POI終了条件に含まれる．
- P区間はPOIにおける指定したサービス内容のサービス条件に含まれる．
- P区間の長さが，滞在長条件の最小値と最大値の間にある．
- E区間の最大値が到着点条件に含まれる．
- E区間の長さはPOIから到着点までの移動時間に一致する．

**図1**に，ユーザ条件・サービス条件とそれを満足する解の例を示す．タイムセール寄り道は以下にあげる特徴を持ち，単に時間制約を満たすルートを求めるだけでなく，「スケジュールを立案する」といった性格を持っている．

- 区間の位置は一意的に決まるとは限らない．P区間は，この条件であれば，図1に示したように19時半〜21時半でもよいし，19時〜21時としてもよい．移動時間が長ければ，それにより制約を受けるが，決定には任意性がある．その決定のためには，スケジュールに関する何らかの戦略（できるだけ早く物事を済ます，

© 2012 Information Processing Society of Japan

等）が必要となる．

- POI 滞在区間の前後の待ちを許容している．S 区間と E 区間が POI 滞在時間と重なってはならないという条件はあるが，S 区間の最大値は POI 開始条件に含まれる必要がないし，E 区間の最小値は POI 終了条件に含まれる必要がない．これは，たとえば，POI がレストランとなるような応用において自然な仮定である．

ユーザ条件を細かく定義しているのは，できるだけ一般的な状況に対応するためであり，それらには実用上対応する意味がある．仕事終わりに友人と宴会をする場面で図 1 の値を例にして，以下に説明する（以降の説明でも POI への滞在を宴会の比喩で説明することがある）．

- 出発点条件の最小値：仕事が 17 時までなので，それ以前は出発できない．
- 出発点条件の最大値：18 時に会社が閉まるので，その前に出発する．
- POI 開始条件の最小値：友人が 19 時前に来られないので，それ以降に宴会を開始する．
- POI 開始条件の最大値：宴会を遅くとも 20 時より前に始めたい．
- POI 終了条件の最小値：21 時まではゆっくり宴会をしたい．
- POI 終了条件の最大値：22 時以降は太るので宴会を終わらせる．
- 到着条件の最小値：23 時にならないと家のカギが開かない．
- 到着条件の最大値：門限が 24 時である．
- 滞在長条件の最小値：2 時間は最低ゆっくり宴会をしたい．
- 滞在長条件の最大値：長く滞在すると腰が痛くなり 2 時間が限界．

また，チケット購入のように，窓口に滑り込みで間に合えばよいという場合，滞在長条件を 0 に設定すればよい．

今後の説明のために，ユーザ条件と解の区間の最大値と最小値に名前を付ける（図 1）．

- 出発点条件 $= [start_min, start_max]$
- POI 開始条件 $= [poi_start_min, poi_start_max]$
- POI 終了条件 $= [poi_end_min, poi_end_max]$

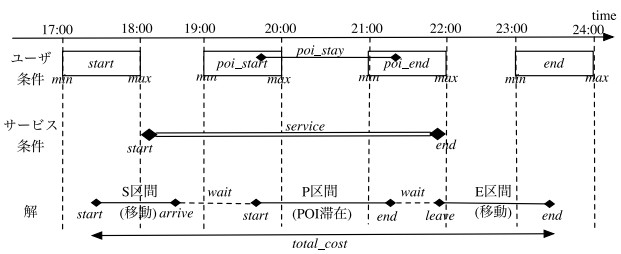


図 1　タイムセール寄り道探索の解
**Fig. 1**　Answer of time-sale trip planning search.

- 滞在長条件の最小値 $= poi_stay_min$
- 滞在長条件の最大値 $= poi_stay_max$
- S 区間 $= [total_start, poi_arrive]$
- P 区間 $= [poi_stay_start, poi_stay_end]$
- E 区間 $= [poi_leave, total_end]$

総所要時間を $total_cost$，出発点から POI までの時間を $cost_s$，POI から到着点までの時間を $cost_e$ と表すこととする．以下の自明な関係が成り立つ．

- $total_cost = total_end - total_start$
- $poi_arrive = total_start + cost_s$
- $total_end = poi_leave + cost_e$

## 4.　タイムセール寄り道探索の基本導出法

### 4.1　逐次拡大法による寄り道探索

基本導出法の基となる寄り道探索のための逐次拡大法（大沢ら [6]）について説明する．上位 k 個の解を求める逐次拡大法の基本的な手順を**図 2** に示す．ダイクストラ法に相当する部分は簡略化している．出発点を $S$・到着点を $E$ とする．基本的なアイディアは，出発点側（$S$ 側）・到着点側（$E$ 側）の両方からダイクストラ法を実行することである．指定したカテゴリのノードに到達したら，そのノード（これが見つかった候補 POI．複数ありうる）までのルートを候補テーブルに登録し，候補テーブルの中で，候補

**while** 展開すべきノード集合が空ではない **do**
　$N_S \leftarrow S$ 側の次の最小コストノード（ダイクストラ法）
　$N_E \leftarrow E$ 側の次の最小コストノード（ダイクストラ法）
　**if** 終了条件を満たす **then**
　　**return**　結果テーブル
　**end if**
　**if** $N_S$ がユーザ指定カテゴリを持つノード **then**
　　候補テーブルに $S \rightarrow N_S$ のルートを登録
　**end if**
　**if** $N_E$ がユーザ指定カテゴリを持つノード **then**
　　候補テーブルに $N_E \rightarrow E$ のルートを登録
　**end if**
　**if** 候補テーブルに両側から同じ POI に到着するルートがある **then**
　　見つかった S 側・E 側のルートをつなげて，結果テーブルに登録
　　候補テーブルから，見つかった S 側・E 側のルートを削除
　**end if**
　結果テーブルを全体コストでソートし，上位 k 個のみを保持
　コストテーブルの更新（ダイクストラ法）
**end while**

図 2　逐次拡大法のアルゴリズム（概要）
**Fig. 2**　Overview of incremental algorithm.

© 2012 Information Processing Society of Japan

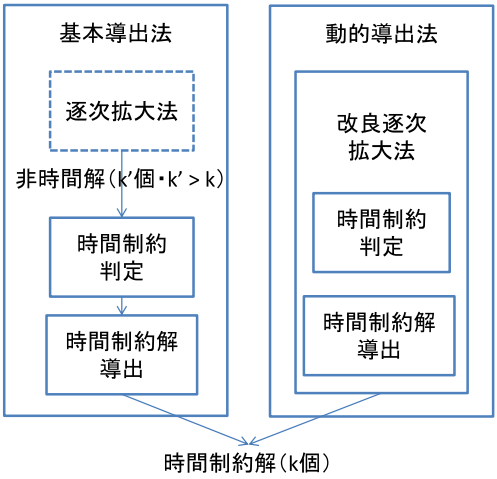


図 3　基本導出法と動的導出法の概要
**Fig. 3**　Overview of basic method and dynamic method.

POI に両側から到達するルートが登録されたら寄り道が完成したことになるので，それをつなげて結果テーブルに登録する．

繰返しの終了条件であるが，通常のダイクストラ法ならば，到着点のコスト決定時点でよいが，寄り道探索では到着点ではなく POI を探すため，以下の条件とする必要がある．結果テーブルを $RT$，候補テーブルを $CT$，ルート $R$ の全体コストを $cost(R)$，出発点（到着点）から POI までのコストを $cost_s(R)$（$cost_e(R)$），探索において次に展開するノードのコストを $cost_s_current$（S 側），$cost_e_current$（E 側）とするとき，終了判定条件は，k 個の解が求められ，かつ以下の 2 つが両方成り立つことである．この式はよりコストの小さいルートが今後得られる可能性がないことを意味している．

$$max_{R \in RT}(cost(R)) \leq min_{R \in CT}(cost_s(R)) + cost_e_current$$

$$max_{R \in RT}(cost(R)) \leq min_{R \in CT}(cost_e(R)) + cost_s_current$$

ただし，POI 密度（全体ノード数に対する POI 数の割合）が低い場合，k 個の解が見つからないと，上記条件だけでは終了せずに延々と遠回りして解を探してしまう．よって，応用目的にあった現実的な値（例：レストラン探索なら 180 分以内）で上限を設けることも必要になる．

### 4.2　タイムセール寄り道探索解の基本導出法

基本導出法の概要を図 3 の左側に示す．まず既存の逐次拡大法を用いて，時間制約のない解を（最終的に求める個数よりも多めに）求める（非時間解と呼ぶ）．そして，非時間解が時間制約を満たすかを判定し，最終的な解（時間制約解と呼ぶ）を導出する．この方法のメリットは既存の寄り道探索手法をそのまま利用できることにある．一方，問題点は，非時間解を多めに求めたとして，時間制約解が

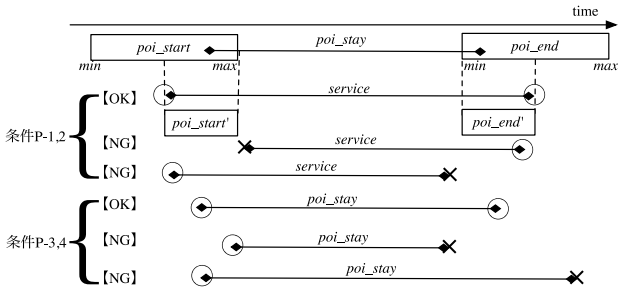


図 4　P 区間の制約チェック
**Fig. 4**　Constraint check of P-interval.

その中に含まれる保証が必ずしもないことである．たとえば，POI 密度が高いが時間制約を満たす POI が少ない場合，逐次拡大法は近くにある POI ばかりを非時間解として求めてしまって，与えられた時間制約を満たす POI に到達しない可能性がある．よって，非時間解は多めに求めておく必要があるが，その個数については 6 章で評価している．

### 4.3　基本導出法における時間制約の判定

非時間解が時間制約を満たすかの判定について述べる．3 つという限られた時間区間の判定なので，(a) 単独区間の制約判定，(b) 隣接 2 区間の制約判定，(c) 全体（3 区間）の制約判定，を数直線上でチェックする初等的な方法でよい．また，方針としては，最初にサービス条件と POI 開始条件と POI 終了条件の制約をチェックする．それが満たされるのであれば，それらの交わりを新たな POI 開始条件と POI 終了条件としてその後のチェックに進む．このことで，実質的にサービス条件のチェックが終了するため，その後，サービス条件を考慮する必要がなくなる．つまり最初に P 区間単独の制約判定を行うことになる．

P 区間に関するチェックの例を図 4 に示す．最初に，POI 開始・終了条件とサービス条件に矛盾がないことをチェックする．POI への滞在開始時刻と滞在終了時刻はサービス時間に含まれる必要がある．以下に条件を示す．

- 条件 P-1：$service_start \leq poi_start_max$
- 条件 P-2：$poi_end_min \leq service_end$

次に，POI 開始・終了条件を補正する．これはサービス条件を POI 開始・終了条件に反映させる処理である．POI 開始・終了条件とサービス条件で交わりをとり，それを補正された POI 開始条件・終了条件として今後の処理で使用する（図 4 の $poi_start'$ と $poi_end'$）．

次に，補正された POI 開始・終了条件と POI 滞在時間に矛盾がないことをチェックする．図 4 に示すように，滞在時間が POI 開始・終了条件に対して短い場合と長い場合は矛盾である．制約を満たす条件を以下に示す．条件 P-3 が POI 滞在が短い場合の排除，条件 P-4 が POI 滞在が長い場合の排除である．これで P 区間単独としての制約チェッ

© 2012 Information Processing Society of Japan

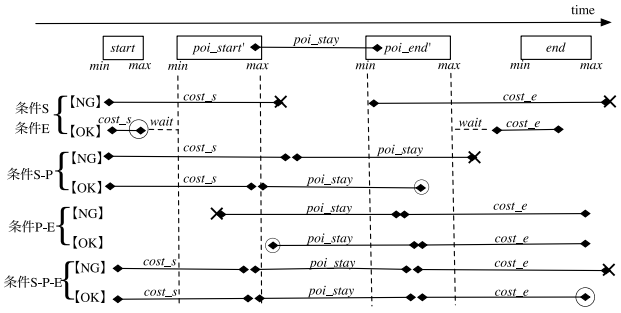


図 5 S 区間と E 区間の制約チェック
**Fig. 5** Constraint chek of S-interval and E-interval.

クが完了する．

- 条件 P-3：$poi_end'_min < poi_start'_max + poi_stay_min$
- 条件 P-4：$poi_start'_min + poi_stay_max \leq poi_end'_max$

次に，S 区間と E 区間についての制約チェックを行う．非時間解で求まった出発点から POI までの時間（$cost_s$）と POI から到着点までの時間（$cost_e$）を利用する．**図 5** にチェックの例を示す．まず，S 区間と E 区間の単独のチェックであるが，待ちを許容するから小さい分には制約はない（POI 開始条件（終了条件）内に到着（出発）する必要はない）ので，大きい場合のチェックのみでよい．S 区間は POI 開始条件を超えなければよく，E 区間は POI 終了条件を超えなければよい．以下に条件を示す．

- 条件 S：$start_min + cost_s < poi_start'_max$
- 条件 E：$poi_end'_min + cost_e < end_max$

次に，隣接 2 区間の制約判定であるが，それぞれ 2 つをつなげた区間が大きすぎて，POI 滞在終了・開始を超えるということがなければよい（小さい場合については，単独区間の制約のみでかまわない）．以下に条件を示す．

- 条件 S-E：$start_min + cost_s + poi_stay_min \leq poi_end'_max$
- 条件 P-E：$poi_start'_min \leq end_max - (cost_e + poi_stay_min)$

最後に 3 区間全体の整合であるが，出発から到着までのトータルの長さが区間の最大幅を超えなければよい．以下に条件を示す．

- 条件 S-P-E：$cost_s + poi_stay_min + cost_e < end_max - start_min$

これで制約条件のチェックは完了である．

### 4.4 基本導出法における時間制約解の導出

次に，時間制約解の導出について説明する．前節のチェックによって，「時間（の長さ）」について矛盾するような条件は排除されているが，これまで与えられた制約だけでは，いつ出発し，POI に滞在し，目的地に到着するという「時刻」は一意には決定されない（総所要時間は一意に決まらない）．時刻を決定するためには，スケジュールの戦略を前提としておく必要がある．ここでは，代表的な「できるだけ待ちをなくし，かつ早く行動する」という戦略に基づく方法を提示する．戦略の特徴を以下にあげる．

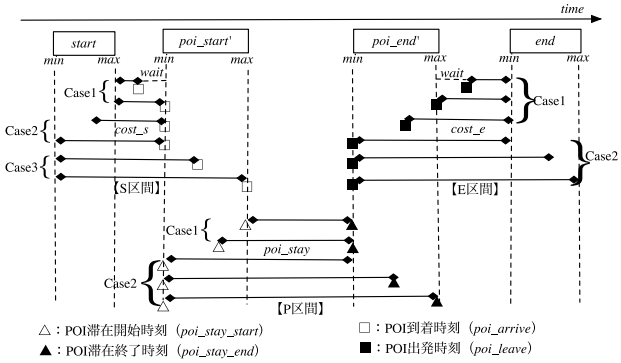


図 6 個別値の決定イメージ
**Fig. 6** Overview of each value determination.

- できるだけ早く POI 滞在を開始する．
- できるだけ早く帰宅しようとする．
- 待ち時間はできるだけ発生させない．
- POI でのサービスに間に合う範囲で，出発点の出発時間を遅らせる．

この戦略は単純化されており，現実の適用領域によっては適さない場合もあるが，それは下記に述べる方法を基本として修正すれば実現できる．

以下の導出では単純化のため P 区間の長さは固定であると仮定する（この値を $poi_stay$ とする）．

時間制約解の導出は 2 つのフェーズによって構成される．

- 個別値決定フェーズ
- 調整フェーズ

個別値決定フェーズとは，全体としてのスケジュールの整合性は無視し，3 つの区間の位置をそれぞれ戦略に合うように（この場合はできるだけ早めに行動するように）決定する．調整フェーズでは，前フェーズで決まった区間が重ならないように調整する．

**図 6** に個別値決定フェーズのイメージを示し，決定するための場合分け条件を**表 2** に示す．この表の条件ですべての値が決定される．この表に入らないような区間が長すぎるケース，短すぎるケースは事前に除外されている．

S 区間は，$cost_s$ の値が小さいときは，許容されるぎりぎり（$start_max$）に出発し，POI 到着後宴会開始まで待つことになる（Case1）．$cost_s$ がだんだん大きくなり待ちが短くなり，$poi_arrive$ が $poi_start_min$ に到達すると，次に $total_start$ が前倒しされる（Case2）．$total_start$ が $start_min$ に到達した時点で，$poi_arrive$ が後ろにずれていく（Case3）．$cost_s$ の最大値は $poi_start_max - start_min$ である．

P 区間は，POI 滞在時間（$poi_stay$）が小さいときは，許容されるぎりぎり（$poi_end_min$）に終了するように宴

© 2012 Information Processing Society of Japan

表 2　個別値の決定
**Table 2**　Each value determination.

| 場合 (Case) | | 条件 | 変数 | 値 | 備考（意味解釈） |
|---|---|---|---|---|---|
| S 区間 | 1 | *cost_s =< poi_start_min - start_max* | *total_start* | *start_max* | 往路が短く待ち発生 |
| | 2 | *poi_start_min - start_max < cost_s*<br>*cost_s <= poi_start_min - start_min* | *total_start* | *poi_start_min - cost_s* | 待たずに最も早く宴会開始 |
| | 3 | *poi_start_min - start_min < cost_s*<br>*cost_s <= poi_start_max - start_min* | *total_start* | *start_min* | 往路が長く宴会が後ろにずれる |
| P 区間 | 1 | *poi_end_min - poi_start_max <= poi_stay*<br>*poi_stay < poi_end_min - poi_start_min* | *poi_stay_start* | *poi_end_min - poi_stay* | 宴会は最も早く終る |
| | 2 | *poi_end_min - poi_start_min <= poi_stay*<br>*poi_stay <= poi_end_max - poi_start_min* | *poi_stay_start* | *poi_start_min* | 宴会が長いため終了が後ろにずれる |
| E 区間 | 1 | *cost_e =< end_min - poi_end_min* | *poi_leave* | *end_min - cost_s* | 最も帰宅が早い |
| | 2 | *end_min - poi_end_min < cost_e*<br>*cost_e <= end_max - poi_end_min* | *poi_leave* | *poi_end_min* | 復路が長く帰宅が後ろにずれる |

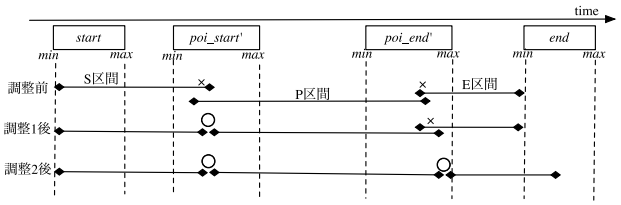


図 7　調整フェーズのイメージ
**Fig. 7**　Overview of adjustment phase.

会が行われる（Case1）．POI 滞在時間がだんだん大きくなり，*poi_stay_start* が *poi_start_min* に到達すると，次に *poi_stay_end* が後ろにずれる（Case2）．POI 滞在時間の最大値は $poi_end_max - poi_start_min$ である．

E 区間は，*cost_e* の値が小さいときは，許容されるぎりぎり（*end_min*）に到着するように POI を出発する．宴会終了後 POI 出発まで待つことになる（Case1）．*cost_e* がだんだん大きくなり待ちが短くなり，*poi_leave* が *poi_end_min* に到達すると，次に *total_end* が後ろにずれる（Case2）．*cost_e* の最大値は $end_max - poi_end_min$ である．

次に調整フェーズについて述べる．上記パターンの組合せによって全体が決定されるが，調整が必要な場合として，一般的には以下の 2 つがある．

- 各区間の間に空きがある場合
- 各区間に重なりがある場合

しかし，POI 滞在時間を固定としたために，空きがある場合は，それは待ち時間となり，調整の余地はない．重なりがある場合の解決方法であるが，イメージを**図 7** に示す．個別値決定では可能なかぎり，前倒しになるようにスケジュールが決定されているので，重なりが発生した場合は，後の区間を後ろにずらせばよい．S 区間と P 区間が重なったら，P 区間を後ろにずらし，さらに P 区間と E 区間が重なったら E 区間を後ろにずらす．その結果，*total_end* が *end_max* を超えるケースは事前に除外されているため正しい解が得られる．

## 5.　タイムセール寄り道探索解の動的導出法

動的導出法は，時間制約条件を積極的に利用して，時間制約解を探索中に完成させ，事前に候補を絞り込み性能向上をはかる方式である．図 3 の右側にイメージを示しているが，基本導出法で用いた時間制約判定と時間制約解導出を逐次拡大法の中に取り込んでいる．単純に取り込むだけではなく，できるだけ早めに絞り込みを行うための修正を行っている．

基本導出法における時間制約判定は，最初に P 区間のチェックを行い，POI 開始・終了条件の補正を行ったために，このままではすべての条件が POI 確定後ではないとチェックできない．しかし，条件 S，条件 E，条件 S-P，条件 P-E は，補正前の POI 開始・終了条件を用いてチェックすることができ，POI 確定前にチェックできる条件である．以下に条件を示す（それぞれ元の条件を補正前に変更したものである）．

- 条件 S’：$start_min + cost_s \leq poi_start_max$
- 条件 E’：$poi_end_min + cost_e \leq end_max$
- 条件 S-P’：$start_min + cost_s + poi_stay_min \leq poi_end_max$
- 条件 P-E’：$poi_start_min \leq end_max - (cost_e + poi_stay_min)$

この 4 条件を含んだ 13 条件をできるだけ早い段階でチェックすることにより性能向上をはかる．その処理フローを，**図 8** に示す（図の (b) は (a) における「S 側（E 側）で確定したノードが POI かチェック」の中のフローである）．元となっている逐次拡大法に対する修正点を，図中で二重線で示している．以下の 3 つのステップで候補を絞り込む．それぞれでチェックできる条件を括弧内に書く．

- Step1：ノードへの最短距離が確定する時点（条件 S’，E’，S-P’，P-E’）
- Step2：片側から POI に到達する時点（条件 S，E，P-1〜4，S-P，P-E）
- Step3：POI への両側からのパスが見つかる時点（条

© 2012 Information Processing Society of Japan

件 S-P-E）

Step1 のチェックは，探索範囲の限定に効果がある．Step2 のチェックは，解の候補テーブルを小さくする効果がある．Step3 のチェックは，最終的に全体としての制約を満たすため必要なチェックである．条件 S’，E’，S-P’，P-E’ は，POI に到達する前にチェックできるので，Step1 でのチェックが可能である．チェックの結果 NG になった場合は，そのノードに至るコストを無限大に設定し，そのノードの先を展開しないようにする．一方，条件 S，E，P-1〜4，S-P，P-E は，POI に到達しないとチェックができないが，両側からのコストを要求していないので，Step2 でのチェックが可能である．チェックの結果 NG になった場合は，候補テーブルへの追加を行わない．条件 S-P-E は両側からのコストが必要なので，Step3 でのチェックが必要である．チェックの結果 NG になった場合は結果テーブルへの追加を行わない．

チェックが終わってから，S，P，E 区間を導出するが，これは基本導出法と同じ方法である．そこで求まった総所要時間によって，結果テーブルのソートを行う．

また，終了条件において，POI 滞在時間を考慮する必要があるため，4.1 節で述べた逐次拡大法における終了条件を以下のように修正する必要がある．

$$max_{R \in RT}(cost(R)) \leq min_{R \in CT}(cost_s(R)) + cost_e_current + poi_stay_min$$

$$max_{R \in RT}(cost(R)) \leq min_{R \in CT}(cost_e(R)) + cost_s_current + poi_stay_min$$

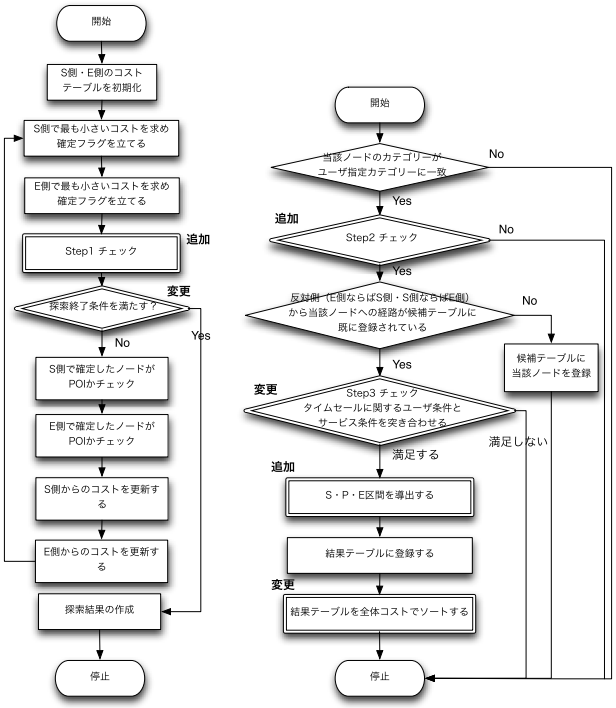


図 8　動的導出法のフロー図

**Fig. 8**　Flow of dynamic method.

## 6.　システムの実装と評価

前章までに述べた基本導出法，動的導出法を実装し評価した．グラフ情報を実装するために，グラフデータベースである Neo4j [16] を利用した．首都圏の実際の鉄道網の情報と，POI 情報をグラフデータベースとして構築した（ノード数 64 万件，エッジ数 60 万件）．グラフデータの例を**図 9** に示す．鉄道網に関しては，駅をノード，路線をエッジとして表現し，POI 情報はノードとして表現し，駅と POI の間は徒歩による移動時間を入れてエッジとして表現している．また，今回の実験システムでは，サービス情報と時間情報もグラフデータベース内のノードとして構築した．サービス情報と時間情報に対するエッジは，鉄道網と POI 情報のエッジと種別を区別することによって，グラフ探索の範囲から除外している．サービス情報と時間情報はできるだけ汎用的になるように設計している．1 つの POI が複数のサービス（例：ランチサービス，宴会サービス）を持つ場合，それは別ノードとして表現する．時間情報は 1 つの区間に対応しており，1 つのサービスに対して，複数区間の営業時間がある場合（例：ランチ後に休憩あり），時間情報のノードを複数持たせることによって表現することができる．時間情報にはサービス実施，非実施の識別子があり，サービスを実施している区間とサービスを実施していない区間という表現もできる．

このグラフデータベースに対して，タイムセール寄り道探索の基本導出法，動的導出法を実装し，Web からもアクセス可能な，タイムセール寄り道探索の実験システムを構築した（実験環境：Apple MacPro（Xeon 2.8 GHz × 2），メモリ 12 GB，SSD128 GB，Mac OS X 10.6.7．プログラミング言語：Java（JDK 6））．その画面例を**図 10** に示す．ユーザは出発地，到着地，寄り道先のカテゴリ，サービス

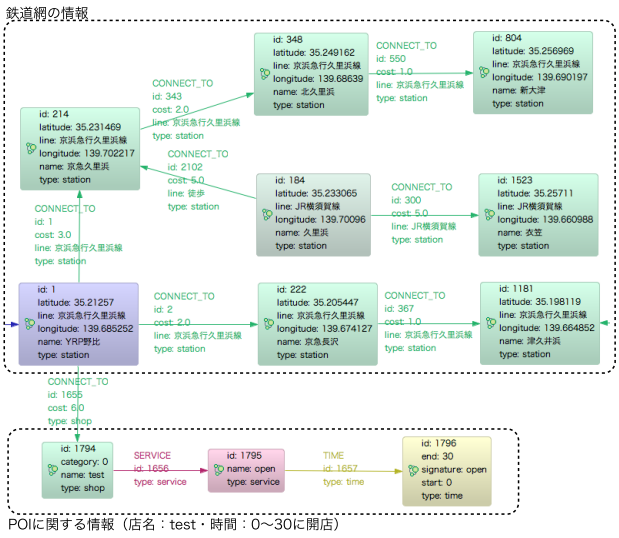


図 9　グラフデータベースのデータ例

**Fig. 9**　Sample of graph database data.

© 2012 Information Processing Society of Japan

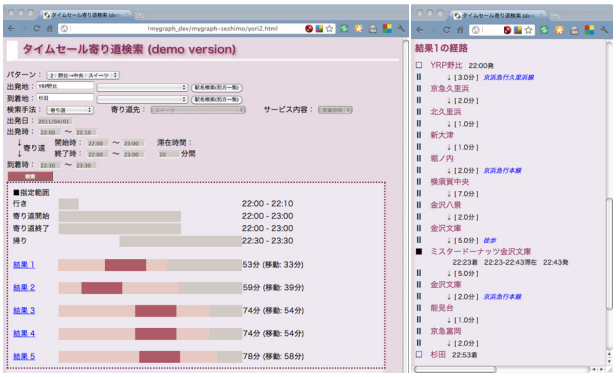


図 10　寄り道探索システムの画面
**Fig. 10**　Sample window of trip planning search system.

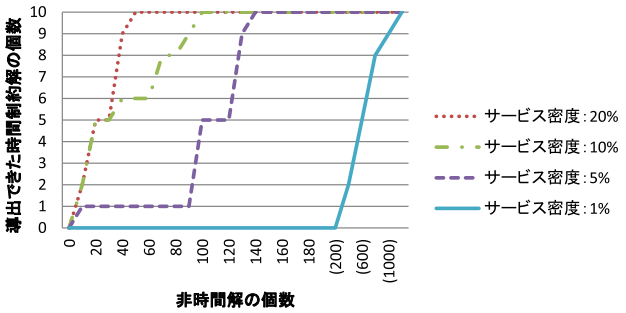


図 11　基本導出法における非時間解の個数と時間制約解の個数の関係
**Fig. 11**　Relationship of no constrained answers and time constrained answers.

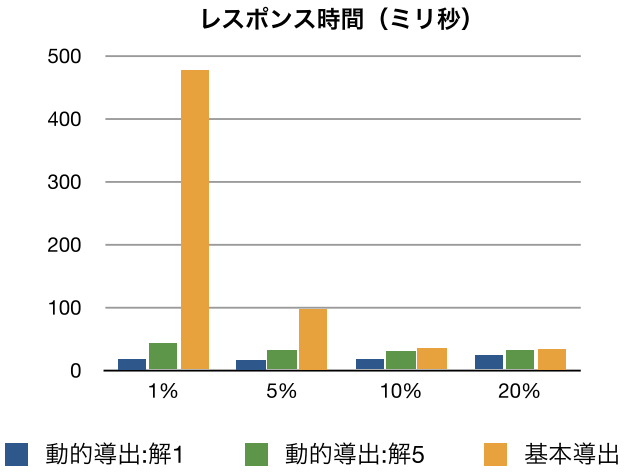


図 12　基本導出法と動的導出法の比較（レスポンス時間）
**Fig. 12**　Evaluation of response time.

内容，ユーザ条件となる時間制約を指定して探索実行すると，探索結果のスケジュールとルートを返却する．画面は実験目的であるので，すべての時間制約を入力可能にしているが，実際のアプリケーションでは，時間制約をスケジュールシステムから取得したり，個人ごとに事前設定したりするやり方も考えられる．

まず，基本導出法において，中間解として生成される非時間解の個数をどの程度にすべきかを見積もった．データとしては上記の鉄道網データに加えて，サービス密度（交通網のノード数に対して，指定されるサービス内容を実施している POI の数の割合）を変化させ，評価用に自動生成した POI データを用いている（ノード数とエッジ数はともに約 4,000 件）．また，POI は各駅に隣接して 1 つずつ設定した．よって，サービス密度が 100%とは探索するすべての駅ごとにサービスが実行されているという意味となる．

我々は，本論文の手法の適用範囲を，サービス密度が 20%以下の比較的サービス密度が低い場合を想定し，評価した．サービス密度が高い場合，本論文で述べたような手法を用いずに，出発点と到着点の間の最短経路を求め，その経路上にある POI を列挙する簡易な手法でも，ある程度実用的な解を求められることが，その理由である（POI の分布が偏っていれば，正解を見つけられない場合はある）．しかし，サービス密度が 20%の場合とは，今回の評価データである鉄道網であれば，5 駅のうち 1 つでサービスを実施することに相当し，道路網においては，エッジの長さが信号間の距離 200 m 程度であると想定すれば，約 1 km ごとにサービスがあることに相当する（エッジが直線的に連続する場合）．これは，サービスへの適用範囲としては十分に広いと考えた．実際の実用的なサービスにおけるサービス密度の調査については今後の課題である．

まず，動的導出法によって正解となる 10 個の時間制約解を求めておいて，非時間解の個数を変化させて，同じ条件で基本導出法を実行したときに正解となる時間制約解を 10 個中何個導出できるかを調べた．評価結果を**図 11** に示す．サービス密度が低くなる場合，非時間解を多く生成しないと十分な時間制約解を求めることができない．以降の評価では，最終的に返却する解の個数を 1，5 の 2 通りについて評価するために，5 個の正解を得ることができるように，非時間解の個数を 1%：500，5%：100，10%：20，20%：20，と設定した．

次に，基本導出法と動的導出法の探索を実行し，展開された総ノード数，候補テーブルの要素の最大個数，レスポンスタイムを測定した．ユーザの利用シーンとして，以下を想定した．

- 移動途中の 1 時間の空き時間に，タイムセールで安くなったケーキをお土産として買う．
- 買い物時間は 10 分．
- タイムセールの時間は 1 時間．

この利用シーンに相当する時間制約を与え探索を実行し，測定した結果を**図 12**，**図 13**，**図 14** に示す（出発点，到着点を 2,000 組ランダムに選び，その中で解が見つかったものの平均値を示している）．時間制約解の個数は 1，5 の 2 通り測定したが，基本導出法についてはどちらの結果もほぼ同じである．特にサービス密度が低い 1%，5%の場合に，動的導出法の結果がすぐれている．たとえば，サービス密度が 1%の場合は，動的導出法を用いることにより，総

© 2012 Information Processing Society of Japan

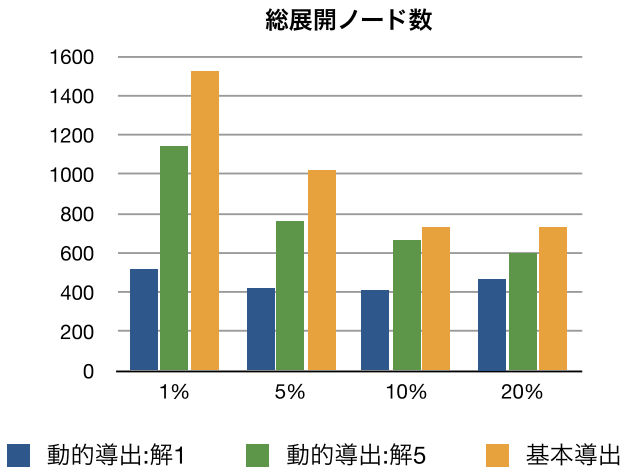


図 13　基本導出法と動的導出法の比較（総展開ノード数）
**Fig. 13**　Evaluation of number of expanded nodes.

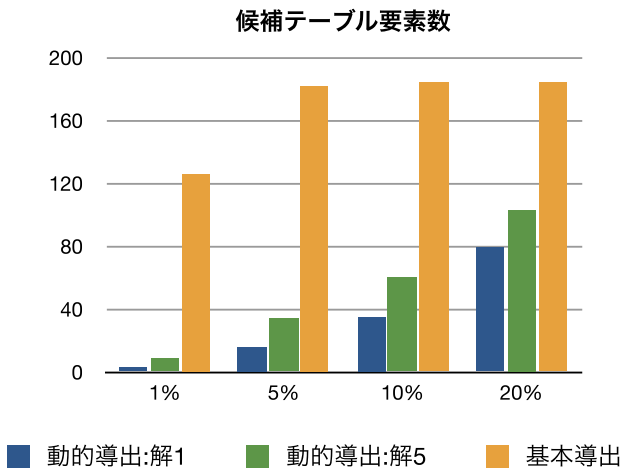


図 14　基本導出法と動的導出法の比較（候補テーブル要素数）
**Fig. 14**　Evaluation of number of candidate table elements.

展開ノード数が 34〜75%に，候補テーブルの要素数が 2〜7%に減り，レスポンスタイムが 11〜24 倍に改善されている．特に時間制約解が 1 つの場合に改善効果が大きい．また，図 13，図 14 から，総展開ノード数と候補テーブル要素数が動的導出で大幅に削減されており，性能向上に貢献していると推測される．また，サービス密度が高い 20%の場合には，動的導出法の優位性は小さくなるものの，そのオーバヘッドは大きくないことが分かる．今回の測定では，鉄道網というグラフとしては小規模な例で評価したが，道路網等のより大きいデータになれば，この差は広がるため，動的導出法は大規模グラフデータベースに対してはより有効であると考えられる．

## 7.　おわりに

本論文では，タイムセール寄り道探索の考え方を提案し，その実現方法として基本導出法と動的導出法を明らかにした．提案方式をグラフデータベースを利用して実装，評価し，動的導出法がサービス密度が低い場合に特に有効であることを示した．今後の技術的な課題としては，複数の POI への拡張がある．ダイクストラ法を基本とする本手法では複数 POI へ拡張する場合，中間解における組合せが巨大になるという問題が発生する．よって，本手法のように最適解を求めることが難しいため，非最適解を精度，効率良く求める手法が必要になる．また，本技術を適用する観点では，本論文では鉄道網をベースとした典型的な利用シーンを想定して評価したが，本手法は汎用的であるため，より広い利用シーンで利用可能である．タイムセールというと，現状だと夕方限定ということが多いかもしれないが，今後，twitter 等を利用して積極的な時間限定セールのマーケティングを行うとすると，本技術との組合せで新しいサービスが実現できる可能性もある．今後，サービスイメージや利用シーンをさらに検討し，その要求条件に沿ったモデルによる評価が課題となる．

## 参考文献

[1] Dijkstra, E.W.: A note on two problems in connexion with graphs, *Numerische Mathematik*, Vol.1, pp.269–271 (1959).
[2] Hart, P.E., Nilsson, N.J. and Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Trans. on Systems Science and Cybernetics*, Vol.SSC4, No.2, pp.100–107 (1968).
[3] 五十嵐健夫：データ構造とアルゴリズム，数理工学社 (2007).
[4] 駅前探検倶楽部，入手先 ⟨http://ekitan.com/⟩（参照 2011-05-23）.
[5] 食ベログ，入手先 ⟨http://tabelog.com/⟩（参照 2011-05-23）.
[6] 大沢　裕，藤野和久：前処理を必要としない道路ネットワーク上での最短寄り道経路探索アルゴリズム，電子情報通信学会論文誌 D，Vol.J93-D, No.3, pp.203–210（オンライン）(2010)，入手先 ⟨http://search.ieice.org/⟩.
[7] Sharifzadeh, M., Kolahdouzan, M. and Shahabi, C.: The optimal sequenced route query, *VLDB J.*, Vol.17, pp.765–787 (2008).
[8] Li, F., Cheng, D., Hadjieleftheriou, M., Kollios, G. and Teng, S.-H.: On trip planning queries in spatial databases, *Proc. SSTD 2005*, pp.273–290 (2005).
[9] Papadias, D., Zhang, J., Mamoulis, N. and Tao, Y.: Query processing in spatial network databases, *Proc. 29th VLDB*, pp.790–801 (2003).
[10] 蒲原智也，上島紳一：道路網応用のための空間索引木の提案と最短経路探査への応用，情報処理学会論文誌　データベース，Vol.2, No.2, pp.10–28（オンライン）(2009)，入手先 ⟨http://www.bookpark.ne.jp/ipsj/⟩.
[11] Tansel, A.U., Clifford, J. and Gadia, S.: *Temporal Databases: Theory, Design and Implementation*, Benjamin-Cummings (1993).
[12] Norvag, K.: Temporal Query Operators in XML Databases, *Proc. ACM Symposium on Applied Computing 2002*, pp.402–406 (2002).
[13] Kleinberg, J. and Tardos, E.：アルゴリズムデザイン，共立出版 (2008).
[14] 松田三恵子，杉山博史，土井美和子：歩行者の経路への嗜好を反映した経路生成，電子情報通信学会論文誌 A，Vol.J87-A, No.1, pp.132–139（オンライン）(2004)，入手先 ⟨http://search.ieice.org/⟩.

© 2012 Information Processing Society of Japan

[15] Cheng, J., Ke, Y. and Ng, W.: Efficient Query Processing on Graph Databases, *ACM Trans. Database Systems*, Vol.34, No.1, pp.1–48 (2009).
[16] Neo4j WEB Page, available from ⟨http://neo4j.org⟩ (accessed 2011-05-23).

**鈴木 源吾** （正会員）

1990 年東北大学大学院理学研究科修士課程修了．同年日本電信電話株式会社入社．データベース設計，マルチデータベースシステム，XML データ管理，グラフデータ管理等の研究開発に従事．電子情報通信学会，日本データベース学会各会員．

**榎本 俊文** （正会員）

1994 年大阪大学大学院基礎工学研究科修士課程修了．同年日本電信電話株式会社入社．エージェント技術，情報検索，XML データ管理等の研究開発に従事．電子情報通信学会，人工知能学会各会員．

**小林 伸幸** （正会員）

1989 年京都大学大学院応用システム科学研究科修士課程修了．同年日本電信電話株式会社入社．データベース管理システム，XML データ管理等の研究開発に従事．電子情報通信学会，日本データベース学会各会員．

**山室 雅司** （正会員）

1987 年早稲田大学大学院理工学研究科修士課程修了．同年日本電信電話株式会社入社．1990 年コロンビア大学大学院電気工学研究専攻修士課程修了．以来ネットワーク設計法，データベース設計・可視化，マルチメディア情報検索，データストリーム管理，XML データ管理，グラフデータ管理の研究開発に従事．博士（工学）．1994 年電子情報通信学会学術奨励賞．電子情報通信学会シニア会員，日本ソフトウェア科学会，日本データベース学会，IEEE-CS 各会員．本会マルチメディア通信と分散処理研究会幹事，本会財務理事．

**鬼塚 真** （正会員）

1991 年東京工業大学工学部情報工学科卒業．同年日本電信電話株式会社入社．2000～2001 年ワシントン州立大学客員研究員．博士（工学）．ACM，電子情報通信学会，日本データベース学会各会員．

© 2012 Information Processing Society of Japan