卒業論文　　2002年度(平成14年度)

VVF：モバイルリザベーションに適した Variable Value Function スケジューリング方式

指導教員

慶應義塾大学環境情報学部

徳田　英幸

村井　純

楠本　博之

中村　修

南　正樹

慶應義塾大学環境情報学部

鈴木 源太

卒業論文要旨 2002 年度 (平成 14 年度)

VVF：モバイルリザベーションに適した Variable Value Function スケジューリング方式

論文要旨

近年実現されつつあるユビキタスコンピューティング環境では，ユーザの周辺に “いつでも · どこでも” 計算機能とネットワークコネクティビティを有するサービスが存在する．ユーザは移動中もそれらを次々と利用できる．こうした時に，物理的な出力物をユーザに提供するプリンタや CD·DVD にデータをコピーするようなサービスでは，ハードウェアが提供物を出力している間，ユーザがその場で待たされることがある．本研究ではこれを待ち時間問題と呼ぶ．ハードウェア自体の性能向上を待つことなく，待ち時間問題を解決するために，移動中のユーザがハードウェアに対して携帯端末を使って遠隔から利用要求を送信する．加えて，ユーザが所持するセンサから取得したデータをもとに予測した到着時刻を送信して，その時刻までに必要な出力をハードウェアが完了させるという手法が考えられる．本論文では，この解決手法をモバイルリザベーションと呼ぶ.

モバイルリザベーションの実現にあたって，いまだ検討されていない機能として，どのユーザの処理から実現していくかを決定するスケジューリング機能がある．本論文では，モバイルリザベーションに適したスケジューリング方式として VVF（Variable Value Function）を提案する．VVF では，ユーザの予測到着時刻をデッドラインとし，デッドラインまでに必要な処理を完了するようスケジューリングする．デッドラインは，ユーザが移動経路や移動速度を変更したり，長い間ある場所に立ち止まっていると変動する．これをデッドライン変動と呼ぶ．デッドライン変動に対しては，既存のスケジューリング方式では対応できない．VVF では，想定するユーザの移動モデルに対応し，ユーザの残り移動距離によって Value Fucntion の傾きを変動させる．これに加え，デッドラインの更新を複数回行うことで，デッドライン変動に対応する．本論文では，VVF でサービス処理にかかる時間の長さが全ユーザ固定の場合について，少ない計算量でスケジューリングする近似を提案した.

本論文の最後で，待ち時間問題に対し，既存のスケジューリング方式と比べて，提案した近似の VVF がどれだけ待ち時間を減少させたかをシミュレーションによって評価した．その結果，ユーザがより遠隔から利用要求をを発行するときほど，既存のスケジューリング方式と比べ，VVF による待ち時間減少がみられた．また，サービス処理にかかる時間の長いサービスほど，VVF による待ち時間減少が大きかった.

キーワード :

1 待ち時間問題, 2 モバイルリザベーション，3 到着時刻予測，4 スケジューリング，5 デッドライン変動

慶應義塾大学 環境情報学部 **鈴木　源太**

Abstract of Bachelor's Thesis
Academic Year 2002

VVF: Variable Value Function Scheduling Algorithm for Mobile Reservation

In the ubiquitous computing environment, there are many services which have computing power and connect to network. The user can use them one after another during movement. But when he uses the service which hardware provides physical output (e.g. printer, data copy service to CD or DVD), he is forced to wait by the service during processing time. We called this problem as the "Waiting Time Problem".

To resolve Waiting Time Problem without imploving hardware's performance, we propose the solution which called "Mobile Reservation". In the Mobile Reservation, the user sends the request and the time of user's arrival to the remote service. The time of user's arrival is calculate by the wearable computer using the sensor data. By the time of user's arrival, hardware tries to finish processing.

To realize Mobile Reservation, it is important to decide the task scheduling algorithm. Most of existing scheduling algorithms which are called as Real-time scheduling use deadline. In the Mobile Reservation, we set the time of user's arrival to deadline. Deadline is changed by the user's route change decision, side trip decision, or speed change. Existing scheduling algorithms cannot deal with this deadline's change. In this paper, we propose the VVF (Variable Value Function) scheduling algorithm which can deal with deadline's change in the Mobile Reservation by dynamic slope change and domain change in Value Function . We also propose the approximation of VVF for the same task length service.

We simulate the mobile reservation. From simulation results, we show that the VVF can decrease user's waiting time as compared with existing scheduling algorithms when the user sends the request further away.

Summary

1 Waiting Time Problem, 2 Mobile Reservation, 3 Arrival Time Prediction, 4 Scheduling, 5 Deadline Change

Faculty of Environmental Information, Keio University.

Genta Suzuki

目 次

図 目 次

表 目 次

第1章

序論

本章では，本研究の意義および本論文の内容構成について述べる．

1.1 本研究の背景

近年，ユビキタスコンピューティング環境 [1] が実現しつつある．ユビキタスコンピューティング環境とは，あらゆる所に偏在した計算機を “いつでも · どこでも” 利用可能なコンピューティング環境である．本研究では，ユビキタスコンピューティング環境の以下の二点に注目する．

さまざまな機器や業務の情報化と設置場所の多様化

ユビキタスコンピューティング環境では，計算機がユーザの移動先のどこにでも存在し，ユーザはそれを利用できる．また，ユーザの周囲にあるさまざまな機器が “情報化” され，計算機能とネットワークコネクティビティを有す．家庭環境においては，これらの機能を持ったエアコン，冷蔵庫，AV 機器等が情報家電として提案されている．ユーザは家の外からでも，それらに接続して，情報が得られる．また，ユーザの移動先の環境にも，ユーザに利益をもたらす新しいサービスが提案されている．たとえば，i-mode[4] を使って購入できる飲料水の自動販売機や，写真印刷，MD への音楽コピー，バス発券，旅行予約などを行う多機能端末が既に実現され，繁華街やコンビニエンスストアに設置されている．さらに，銀行の窓口業務が ATM (Automatic Teller Machine) によって一部機械化されたように，近い将来，ファーストフード店や市役所等の公共機関の業務の一部が機械化され，さらに情報化された形でさまざまな場所に設置されると推測される．

ユーザの所持する携帯端末の高機能化

現在，IEEE 802.11x[2]，Bluetooth[3]，IrDA といった無線伝送方式が提案されている．ユーザはこれらを利用可能な携帯端末を所持して，どこからでもネットワークに接続できる．特に，携帯電話の i-mode サービスの普及と多様化がめざましく，ユーザは携帯電話を使って銀行口座の残高照会や，レンタカー予約，サービスクーポンの発行といった様々なサービスに接続できる．

さらに，GPS (Global Positioning System) や体温計，心拍計湿度計等のセンサを所持したり，Java Ring のような超小型の計算機を身につけるウェアラブルコンピューティング環境が提案されている．これらを利用して，ユーザは “移動しながら” 周囲のサービスに対して，自身の現在の位置情報や健康状態さらに身分証明等の情報提示や，認証を自動的に行える．

以上のユビキタスコンピューティング環境の実現によって，ユーザの移動中にも，ユーザとユーザ，ユーザとサービス，サービスとサービスが相互に接続され，ユーザに情報や物理的な出力をもたらすことができる（図 1.1）．ユビキタスコンピューティング環境では，ユーザが一つのサービスを円滑に利用開始，利用完了できることがより重要である．サービスの円滑な利用とは，サービスの利用に際してユーザにとって不必要な時間が少なく，さらに煩わしい手続きなしにサービスを利用することを指す．必要な時間に，必要なサービスを，即時に利用できることが望ましい．しかしながら，

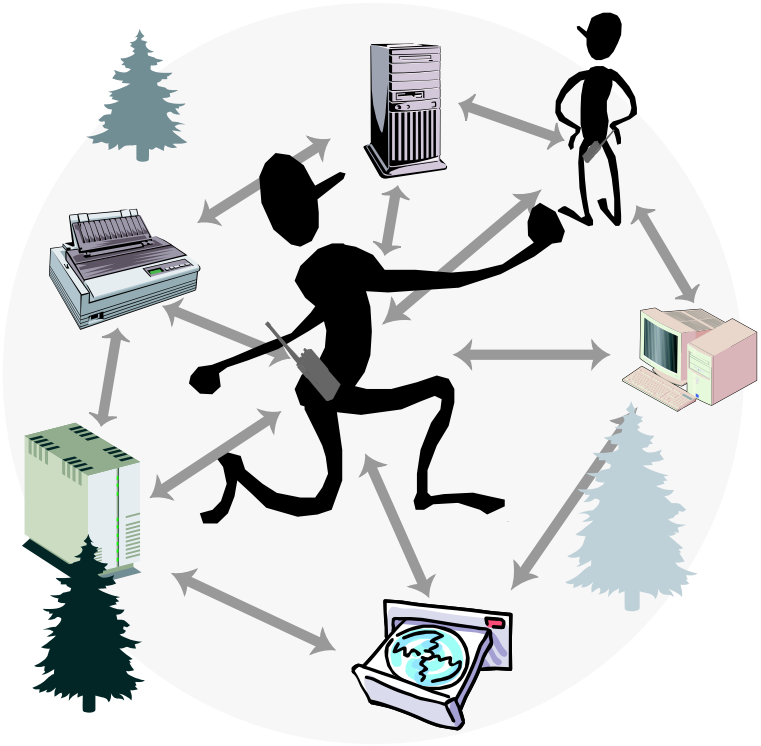

図 1.1: ユビキタスコンピューティング環境

ユーザにとって円滑なサービス利用を困難にする三つの要因がある．それを以下の 1 から 3 で示す．

1. 利用したいサービスを検索すること
2. 利用するサービスに対して必要な入力を行うこと
3. サービスの処理を待つ時間

円滑なサービス利用を妨げる最初の要因は，利用したいサービスを検索することである．周囲の環境に埋め込まれたサービス等の視覚的に発見しづらいサービスをユーザが発見し，その情報を得ることは困難である．このサービス検索の問題を解決するために，ASAMA[5] や LDAP[6] といったディレクトリサービスが提案されている．これらは，ユーザの現在の位置情報から周辺のサービスを検索できる．ディレクトリサービスを利用することによって，ユーザは利用したいサービスがある場合，その場ですぐに携帯端末や WC から周辺のサービスを検索し，その情報を得ることが容易となる．これによってユーザはより快適にサービスを利用できる．

二番目の要因は，利用するサービスに対する，ユーザの直接的な入力である．あるサービスを利用する際に，毎回同じ操作をユーザが直接行うことはユーザにとって煩わしいことである．この問題を解決するために，サービス利用にあたってのユーザの入力を減少させる研究がある．PRONA システム [7] は，情報家電に対するユーザの

直接的な入力を学習し，以後のユーザの直接的な入力を減少させる．また，Zero-stop Authentication[8] ではユーザの所持する携帯電話や PDA と認証サーバとの間でユーザ – アプリケーション間の認証をユーザの直接的な入力無しで即時に実行する．

円滑なサービス利用を困難にする最後の要因は，サービスの処理を待つ時間がユーザに課されることである．サービスを検索して，必要な入力を終えた後，サービス側がサービス利用に必要な処理を実行する時間がある．ユーザのサービス利用にとって，この時間が不必要なものであるならば，この時間は待たされる時間である．印刷などの時間のかかる処理を行うサービスや，複数のユーザの処理を実行しなければならないサービスにとっては，この待ち時間はサービスの円滑な利用を妨げる最大の要因となりうる．

サービスを提供するハードウェアがより高性能で処理速度が高速なものになれば，この待ち時間は減少する．しかし，すべてのハードウェアの性能向上には時間とコストがかかるうえ，限界もある．しかも，高速で処理できるハードウェアが誕生したところで，全てのサービス提供者がそれをすぐに用意するとは限らない．したがって，この解決策は得策ではない．サービスの処理速度向上に頼らず，既存の処理速度でユーザの待たされる時間が減少できれば，ユーザはより円滑にサービスを利用でき非常に有益である．

1.2 本研究の目的及び意義

本研究の目的は，ユビキタスコンピューティング環境において，サービスを提供するハードウェアの処理性能向上に頼ることなく，ソフトウェアからのアプローチによってによってユーザが待たされる時間を減少させることである．待ち時間の減少によって，ユーザはより円滑にサービスを利用できる．また，円滑に利用できるサービスを提供することによってユーザが増加すれば，サービス提供者側も大きな利益を得られる．

1.3 本論文の構成

本論文は，本章を含め全 6 章から成る．次章では，本研究が想定するサービス利用環境を述べ，そこで待ち時間問題を解決する方式を検討する．続く第 3 章では，適用する待ち時間解決方式における大きな問題点であるデッドライン変動問題をとりあげ，その詳細を述べ既存の解決方式の問題点を指摘する．また，第 4 章において，デッドライン変動問題を解決する VVF モデルを提案し，その詳細を述べる．第 5 章で VVF モデルの正当性を示すためのシミュレーションを説明し，VVF を評価する．最後に，第 6 章で，本論文をまとめ，今後の課題について言及する．

第2章

モバイルリザベーション

本章では，本研究で取り組む待ち時間問題を詳述する．本研究が取り上げるサービスは，機器がユーザに印刷物，調理物などの物理的な出力を提供する．ユビキタスコンピューティング環境では，ユーザの移動先にさまざまなサービスが存在する．ユーザがサービスを利用する際には，サービスが他のユーザの処理をしている時間，サービスが自分の処理を完了させるまでの時間がユーザに待ち時間が課される (2.1)．この待ち時間を減少させる方式として，ユビキタスコンピューティング環境においてはモバイルリザベーションが実現可能である．モバイルリザベーションとは，サービスを利用可能な遠隔からサービスへ利用要求を発行し，GPS，加速度計によって取得された到着時刻にあわせてユーザの処理を完了させておく方式である（2.2，2.3，2.4）．本章の最後に，モバイルリザベーション実現の機能要件を述べ，これを実現するうえでの問題点を明らかにする（2.5）．

2.1 想定するサービスと待ち時間問題

本研究が取り上げる**サービス**は，ユビキタスコンピューティング環境において，特殊なハードウェアがユーザに物理的な出力を提供する．サービスの例としては，プリンタや MD，CD へのコピーメディアサービス，機械化された役所やファーストフード店のドライブスルーを想定する．これらのサービスを利用するためには，ユーザはサービスを処理する機器（以後，**サービスプロバイダ**）によってサービスが提供される場所（以後，**サービススポット**）へ移動し，そこに設置されたタッチパネルやユーザが所持する携帯端末からメニューを選択し，選択したメニューにあった出力物（以後，**サービスアウトプット**）を受け取り，サービスの利用を完了する．この際，ユーザが希望するサービスの利用内容を本論文では**リクエスト**と呼ぶ．さらにリクエストをサービスプロバイダ側へ伝達する行為を**リクエストの発行**とする．また，ユーザからのリクエストに応じてサービスプロバイダ側で行うべき処理を**タスク**と呼ぶ．

サービスの例として銀行の ATM があげられる．ATM で現金を引き出す場合，ユーザはサービスプロバイダである ATM の前（サービススポット）まで移動し，ATM のタッチパネルやボタンで現金引き出しを指定（リクエストを発行）する．その後，ATM が現金（サービスアウトプット）を引き出すタスクを実行した後，ユーザは現金を受け取り，現金引き出しというサービスを完了する．

ここで，サービスをタスク長の観点から分類する（図 2.1）．タスク長とは，タスクの実行に必要な時間である．サービスの中には，タスク長が数十秒から数分以上の長いものがある．その例として，プリンタ，MD，CD などのメディアにデータをコピーしてユーザに提供するサービス，自動調理販売機があげられる．たとえば，プリンタの印刷は 30 秒程度，デジタル写真印刷は 60 秒程度，ファーストフード店のドライブスルーは 120 秒，CD へのデータコピーは 180 秒程度である．これらを利用する際，ユーザはリクエストを発行後，印刷物あるいはデータをコピーされたメディアを受け取ってサービスを完了するまでに，印刷やデータコピーといったタスク実行による待ち時間を課される．この時間経過を図 2.2 で示す．横軸が時間，白帯はタスク J の実行にかかる時間を示す．ユーザ到着後リクエストを発行し，サービス側はタスクを実行し，その完了後ユーザはサービスを利用する．タスク開始から終了までの白帯の時間，ユーザはその場で待たされる．また，ユーザの到着時に他のユーザがサービスを利用中であれば，タスクの開始時刻が遅れ，さらに待たされる時間が長くなる．ここで，ユーザがサービススポットに到着後からサービス完了までの間サービススポットで待たされる問題を，本論文では**待ち時間問題**と呼ぶことにする．待ち時間問題を解決できれば，ユーザはより快適にサービスを利用できる．

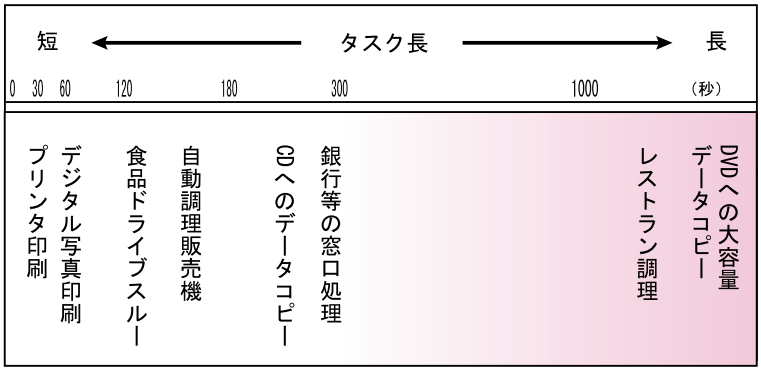

図 2.1: 想定サービスごとのタスク長

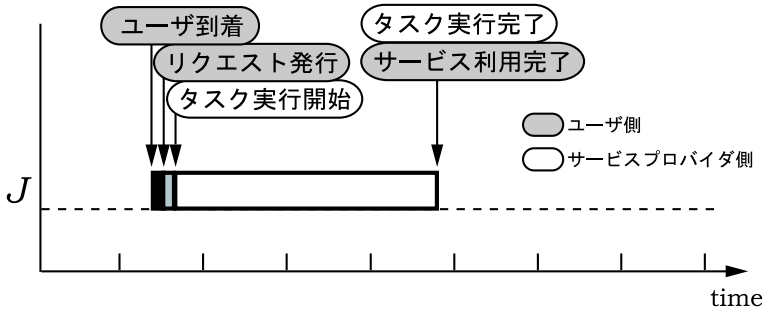

図 2.2: 従来のサービス利用の時間図

2.2 モバイルリザベーション

ユビキタスコンピューティング環境の普及により新たに実現できるモバイルリザベーションについて詳説する．本研究では，モバイルリザベーションは前節の待ち時間問題を解決する際に想定する方式として定義する．

2.2.1 モバイルリザベーション概要

ユビキタスコンピューティング環境では，ユーザはサービスを提供するハードウェアに対してネットワークを介し接続できる．本研究で想定するサービスは，ネットワークを介してユーザがサービススポットと遠隔からリクエスト発行可能なものである（図 2.3）．ただし，遠隔からリクエストを発行してもユーザがサービスを利用完了するためには，サービススポットへ移動して，サービスアウトプットを受け取らなければならない．しかし，サービスプロバイダにとって，複数の遠隔からのリクエストが到着した場合にどれを先に実行するかによって，全ユーザの待ち時間合計が異なった結果となる．サービスプロバイダにとっても，リクエストを受け付けたからといって，いつまでもサービススポットにやってこないユーザのタスクを実行するのは不利益である．

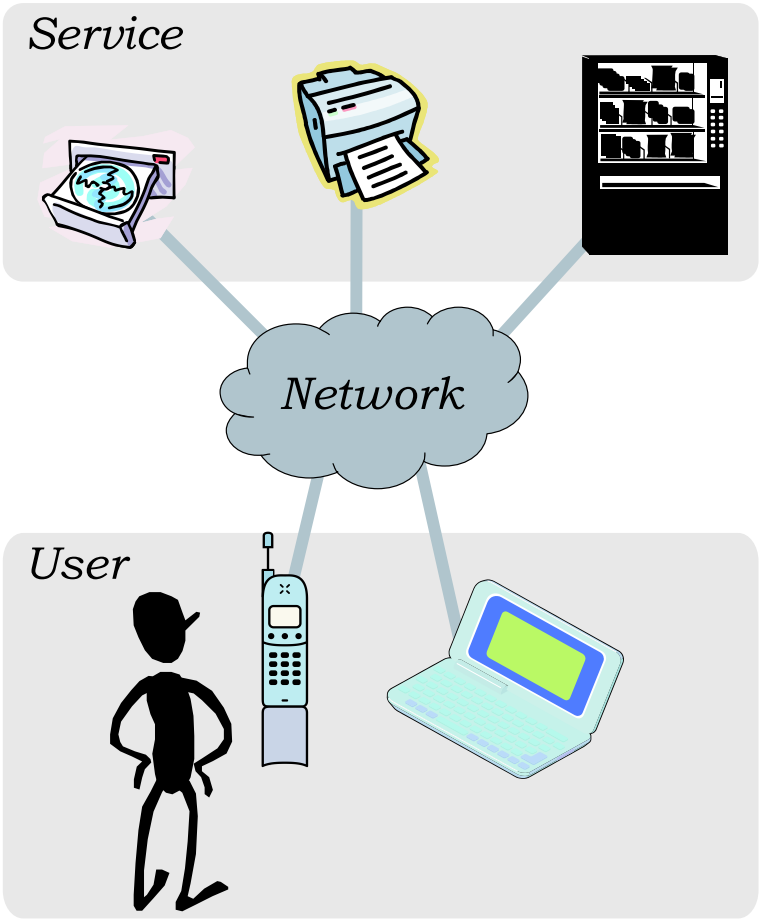

図 2.3: サービスへのネットワークを介した接続

こうしたなかで，GPS や，加速度計を使って，ユーザの現在の位置情報，速度を計算し，それをリクエストとともに送信する方法が考えられる．これをサービスへの**モバイルリザベーション**と呼ぶことにする．モバイルリザベーションによるユーザの待ち時間問題の解決アプローチを図 2.4 で示す．リクエストをユーザ到着前に発行することによって，サービス側はタスクをユーザ到着前に開始する．それによって図 2.2 で示された従来のサービス利用モデルに比べて，ユーザ到着後の待ち時間は減少すると考えられる．また，ユーザの到着後の待ち時間が減ることによって，DVD への映像コピーサービスのように，タスク処理が長い（2 時間のムービーデータ保存に 4 倍の書込速度で 30 分）ために実現されていないサービスが実用化できる．

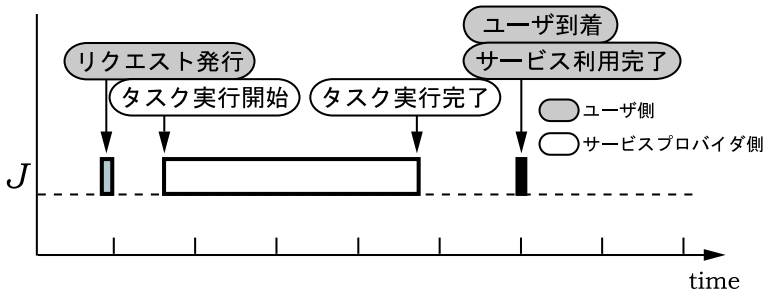

図 2.4: 想定するサービス利用の時間図

2.3 他の遠隔予約方式との比較

モバイルリザベーションとのサービスへの遠隔予約方式とを比較し，それらとの相違点を明らかにする．モバイルリザベーション以外の遠隔予約方式には二種類考えられる．一つ目は遠隔予約を受け付けるが，リクエストの到着順にタスクを実行していく方式，二つ目は遠隔予約を受け付け，さらにユーザが到着時刻を意識的に指定する方式である．本節では，一つ目の方式の例としてネットワークプリンタの lpr (line printer) をあげ，これをモバイルリザベーションと比較し，さらに二つ目の方式とモバイルリザベーションを比較する．

2.3.1 lpr (line printer) との比較

lpr はネットワークプリンタの印刷管理方式である．モバイルリザベーションでは，プリンタもサービス例として取り上げているため，プリンタへのモバイルリザベーションの適用と lpr と比較する．モバイルリザベーションで想定するサービスとユーザの利用モデルは，従来の利用モデルと比較して，サービススポット，ユーザがリクエストを発行する時の状況，リクエストを発行する範囲が異なる．表 2.1 でモバイルリザベーションによるプリンタの利用モデルと lpr の想定するプリンタの利用モデルとの違いをまとめる．

表 2.1: モバイルリザベーションと lpr のプリンタ利用モデルの違い

印刷管理方式	lpr	Mobile Reservation
サービススポット	オフィスや学校等の屋内	街角や公共施設で屋内外
リクエスト発行状況	ユーザが座席から	ユーザが移動しながら
リクエストを発行する範囲	部屋の中 20m 以内を想定	数百メートル以内を想定
到着時刻考慮	しない	する

lpr はオフィスや学校等の決められた空間内に設置され，プリンタに対しておよそ

20m 以内に位置しているユーザが座席端末からリクエストを発行する．リクエストを発行してからサービススポットへの移動にかかる時間はどのユーザも大差の無いものと考えられる．これに対して，モバイルリザベーションでは，繁華街や公共施設のような込み入った場所で，不特定多数の移動中のユーザが，携帯端末からリクエストを発行するモデルを想定している．この環境下では，移動中のユーザがサービススポットへの到着後の印刷処理の待ち時間を減少させるために，従来の lpr よりも離れた数百メートル以内の距離からリクエストを発行しようとする．それゆえ，複数のユーザが同時に利用しようとした時に，それぞれのサービススポットへの移動にかかる時間がユーザによって大きく異なる状況が発生する．

この状況で，従来の lpr を使用した時に，lpr ではリクエストの到着順に印刷を開始するため，遠くにいるユーザが近くにいるユーザよりも早くリクエストを発行すると，早く到着したユーザが余計に待たされるという問題が発生する．これに対し，モバイルリザベーションはユーザの到着を考慮して印刷を行うため，lpr に比べ利用するユーザの待ち時間が減少すると考えられる．

2.3.2 到着時刻指定予約との比較

サービスに対して遠隔からリクエストを発行する際に，同時にユーザが到着時刻を指定してそれを送信する方法が考えられる．本論文では，この方式を**到着時刻指定予約**と名づけ，モバイルリザベーションと比較する．

到着時刻指定方式では，ユーザは指定時刻に合わせてサービススポットに到着するように移動する．一方，モバイルリザベーションでは，指定時刻を意識することなくユーザは自由に動き回ってサービススポットへ到着する（図 2.5）．しかも，ユーザが到着時刻を意識してサービススポットへの移動を行なった場合でも，到着時刻指定予約と同様にタスクが実行される．

また，到着時刻指定予約において，当初に想定した時刻よりも早くユーザが到着した場合，到着時にタスクが完了しておらずユーザが待たされる結果となることがある．しかし，モバイルリザベーションでは，早くユーザが到着することが予測される場合には，タスクの完了時刻もそれに合わせてくりあがるため，到着時にユーザが待たされる時間が減少する．想定した時刻より遅く到着した場合は，従来の方式では，そのユーザが待たされることは少ないが，先に到着したユーザが待たされてしまう可能性がある．一方，モバイルリザベーションでは，ユーザが遅く到着する場合にも，その到着に合わせてタスクを完了させようとするため，そのユーザの到着より先に到着する別のユーザへの影響も少ない．表 2.2 では，サービスアウトプットがすぐに受けとることができ，待ち時間 0 が期待できる場合を ◯，待ち時間 0 が期待できないを × 待ち時間 0 が期待できるが他のユーザを余計に待たせることあるが場合を △ としてモバイルリザベーションと到着時刻指定予約とをまとめた．

表 2.2: モバイルリザベーションと到着時刻指定方式との比較

予約方式	到着時刻指定予約	Mobile Reservation
想定した時刻に到着	○	○
想定より早く到着	×	○
想定より遅く到着	△	○

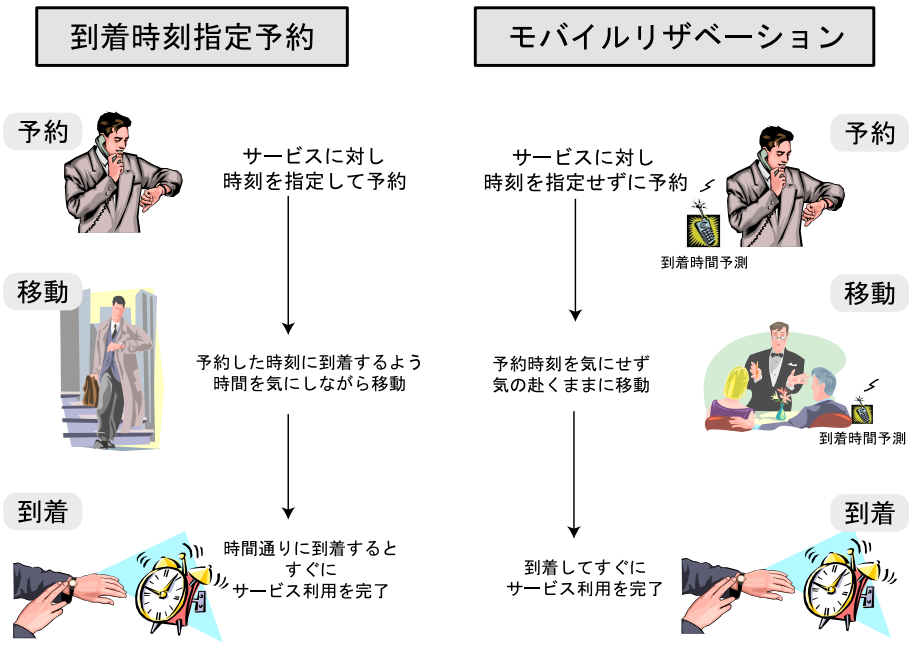

図 2.5: 従来の予約方式とモバイルリザベーション

2.4 想定シナリオ

前項までで述べた，モバイルリザベーションを利用するシナリオの一例を図 2.6 で示す．サービススポットから 離れた場所で，ユーザは PDA からリクエストを発行する．その後，ユーザは サービススポットへ歩行や自動車等任意の方法で移動する．ユーザのサービススポットへの到着に合わせて，サービスプロバイダはそのユーザのタスクを完了させておく．本シナリオ例では，サービスプロバイダとしてネットワークプリンタを挙げた．ネットワークプリンタには，ユーザの到着に合わせて印刷処理（タスク）を行なう機能が無いため，サーバを介してサーバがタスク実行を開始すべき時刻に印刷要求をプリンタへリダイレクトする．プリンタは印刷を開始し，印刷処理を終えると，リクエストを発行したユーザ以外が印刷物を取り出せないように，ソートしておく．ユーザが到着し，ユーザの PDA をプリンタに近付けると，近距離無線伝送方式，たとえば IrDA を利用して PDA とプリンタの間で認証が行われ，認証が正しければ，ユーザがリクエストした印刷物が即時にプリンタからユーザへ渡される．

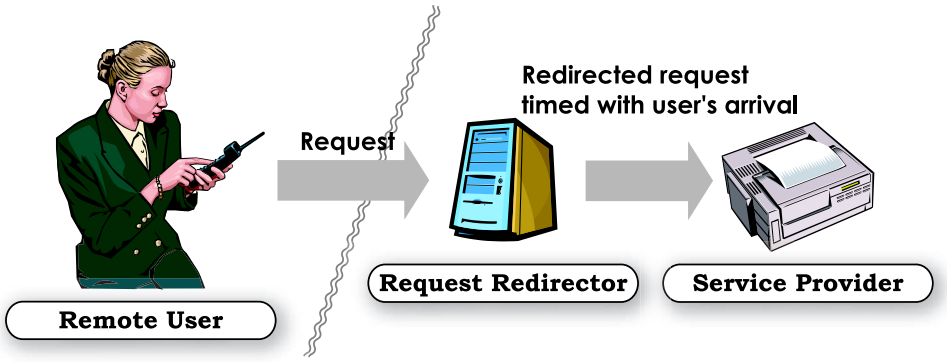

図 2.6: 想定するサービス利用のシナリオ例

2.5 モバイルリザベーションの機能要件

ここで，モバイルリザベーションでユーザ側，サービス側双方で想定するハードウェアとソフトウェアの要件を以下でまとめ，それぞれを説明し，解決方針を明らかにする．実現のための問題点を明らかにする．

- **ユーザ要件**
  - **ハードウェア要件**
    - 位置情報，速度取得機器
    - ネットワークコネクティビティ
  - **ソフトウェア要件**
    - リクエスト発行機能
    - 移動経路予測機能
    - 到着時刻予測機能
    - ユーザ認証機能
- **サービス要件**
  - **ハードウェア要件**
    - サービスアウトプットソート機能
    - ネットワークコネクティビティ
  - **ソフトウェア要件**
    - ユーザ認証機能
    - タスク長算出機能
    - タスクスケジューリング機能

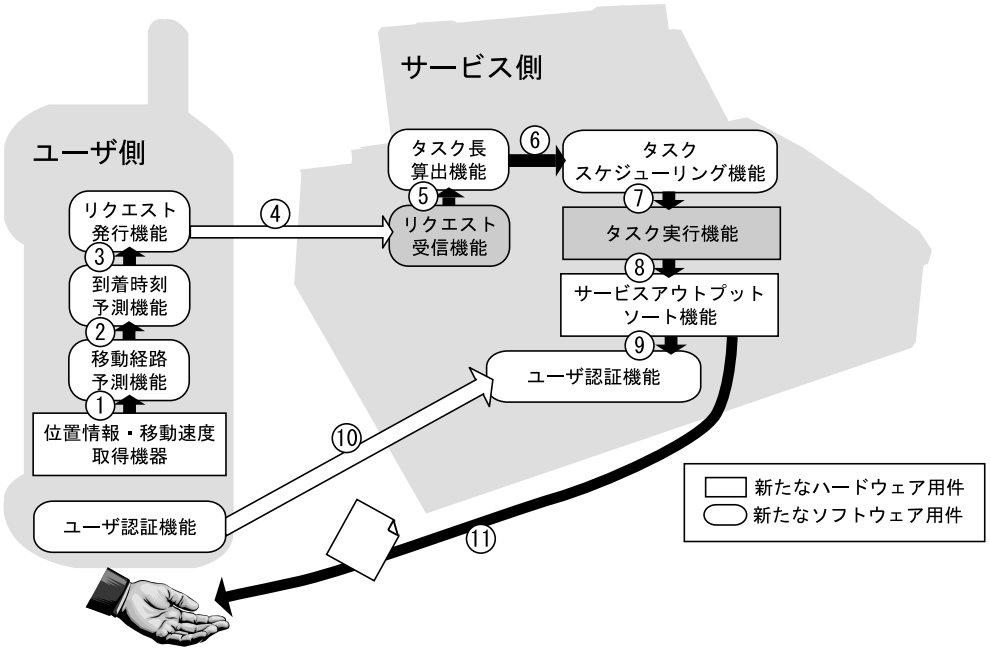

図 2.7: 機能要件

2.5.1 ユーザ機能要件

ユーザハードウェア要件：位置情報，移動速度取得機器

ユーザの位置情報や，移動速度を取得する機能が必要である．ユーザのサービススポットへの到着時刻を予測するためにこれらから取得されるデータが必要である．現在，GPS や，加速度計によってこの機能が実現されている．これらのデバイスをユーザが身につけ，ユーザの所持する携帯端末に接続されるか，ユーザの携帯端末に内蔵されることを想定する．

ユーザハードウェア要件：ネットワークコネクティビティ

リクエスト発行をユーザの移動中に行なうために，無線通信によるネットワークコネクティビティが確保されている必要がある．これは IEEE802.11x 等で実現可能である．

ユーザソフトウェア要件：リクエスト発行機能

遠隔からリクエストを発行する機能である．リクエストをサービス側へ送信することにより，サービス側は図 2.4 の白帯で示されるタスク実行時間が取得できる．利用するサービスに関する情報とリクエスト発行機能については，World Wide Web によって広く公開されているモデルや，ディレクトリサービスによって提供されているモデル，予めユーザ端末が所持しているモデルを想定する．ユーザが所持する携帯端末にはそれぞれのモデルに必要な機能が備わっている必要がある．

ユーザソフトウェア要件：サービススポットへの移動経路予測機能

位置情報取得デバイスから得られたユーザの現在位置と，サービス提供者によって公開されたサービススポットの位置情報を使って，サービススポットへのユーザの移動経路を予測する機能が必要である．複数の移動経路が想定された場合，最短距離の経路を選択することを想定する．現在，カーナビゲーションシステムでこの機能が実現されている．

ユーザソフトウェア要件：サービススポットへの到着時刻予測機能

移動経路予測機能と，移動速度取得デバイスから得られた移動速度を使って，サービススポットへのユーザの到着時刻を予測する機能が必要である．ユーザの到着時刻は図 2.4 の黒棒で示される時刻である．到着時刻予測は，ユーザの現在地からサービススポットまでの残り移動距離を移動平均速度で割って計算する．移動平均速度とは，リクエスト発行時からデッドライン予測時までの間毎秒取得した速度で，速度 0 で停止している時間を除いた速度の平均である．

リクエストの発行時（秒単位）の速度を $V(0)$ とし，現在の速度（秒単位）を $V(n)$，n 秒の間で速度 0 だった秒数を m とすると，n 秒間の移動平均速度 $\bar{V}$ は (2.1) 式で表せる．また，現在地からサービススポットまでの距離を L とすると，現在予測するデッドライン $D(n)$ は (2.2) 式で表せる．これを，リクエストの際や，サービス側からのデッドライン要求があった際，予測したデッドラインとして送信する．

$$\bar{V} = \frac{1}{n-m}\sum_{k=0}^{n} V(k) \tag{2.1}$$

$$D(n) = \frac{L}{\bar{V}} \tag{2.2}$$

ユーザソフトウェア要件：ユーザ認証機能

ソートされたサービスアウトプットを取り出すために，ユーザの到着時に認証する機能である．現在，IrDA 等の近距離無線方式を使った認証法がある．

2.5.2 サービス機能要件

サービスハードウェア要件：サービスアウトプットのユーザ別ソート

サービスプロバイダがユーザ到着前のタスクの実行を完了した際，サービスプロバイダは印刷物や調理物等のサービスアウトプットをユーザの到着までの間，他のユーザに渡すことなく，保存しておく必要がある．モバイルリザベーションでは，印刷物や調理物をユーザ別にソートしておくハードウェア機能を想定する．

サービスハードウェア要件：ネットワークコネクティビティ

遠隔からのユーザのリクエスト発行を受信するために，サービスプロバイダにはネットワークコネクティビティが確保されている必要がある．ただし，サービスプロバイダが固定の場所に設置されている場合，接続形態は無線，有線を問わない．

サービスソフトウェア要件：タスク長の算出機能

ユーザからのリクエストの内容に応じて，必要なタスクを割りだし，その実行時間を算出する機能である．これにより，タスク実行完了時刻が予測できる．それぞれのリクエスト内容とそのタスク長を，後述するタスクスケジューラにサービス提供者があらかじめ入力しておくことを想定する．

サービスソフトウェア要件：ユーザ認証機能

到着したユーザを認証し，どのサービスアウトプットをリクエストしたユーザかを判断する機能である．

サービスソフトウェア要件：タスクスケジューリング機能

受理したリクエストのタスクの実行順序をスケジューリングして，ユーザの到着に間に合うよう，必要な時刻にタスクを実行するタスクスケジューラである．モバイルリザベーションによって待ち時間を減少するためには，この機能が最も重要である．しかし，モバイルリザベーションに最適なスケジューリングアルゴリズムは検討されていない．

以上から，タスクスケジューリング機能以外の機能は，他システムで現在実現されていることがわかる．モバイルリザベーションに適した，優れたスケジューリングアルゴリズムが決まれば，モバイルリザベーションは実現できる．

2.6 本章のまとめ

本章では，待ち時間問題を詳説し，それを解決するためのモバイルリザベーションを提案した．モバイルリザベーションを詳説し，既存の予約方式との違いを明らかにした．そしてモバイルリザベーション実現の要件を述べ，タスクスケジューリングの必要性を明らかにした．

第3章

タスクスケジューリング

本章では，モバイルリザベーションでのタスクスケジューリングの性質について考察し，到着予測時刻が変動するというデッドライン変動問題をとりあげる（3.1）．そして，デッドライン変動によって生じる従来のスケジューリングアプローチの問題点を指摘する（3.2）．

3.1 モバイルリザベーションにおけるタスクスケジューリングの考察

前章にて，モバイルリザベーションによって待ち時間問題を解決する方法を述べ，モバイルリザベーションにおけるタスクスケジューリングの必要性を示した．本節では，モバイルリザベーションにおけるタスクスケジューリングについて考察する．

3.1.1 タスクスケジューリング

モバイルリザベーションで想定するスケジューリングはタスクの実行順序を決定するものである．図 3.1 でスケジューリングの概念図を示す．リクエストが到着すると，そのタスクがスケジューラに入る．スケジューラに既に他の実行すべきタスクがあった場合，決まったポリシーに基づいて，タスクの実行順序を決定し，実行順にタスクがサービスプロバイダのタスク実行機能（図 3.1 楕円）に渡される．

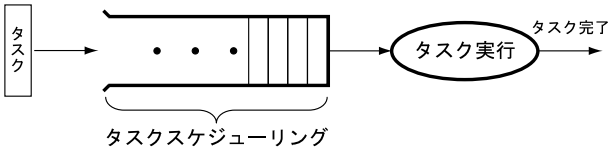

図 3.1: タスクスケジューリング

一箇所に同機能のサービスプロバイダが複数存在する場合，スケジューラがそれぞれのサービスプロバイダに設置された図 3.2 の並行方式よりも，図 3.3 のような一つのスケジューラが複数のサービスプロバイダにタスク実行を命令する直列方式の方が待ち時間が減少することが待ち行列理論 [15][16] で証明されている．したがって，モバイルリザベーションにおいて，複数のサービスプロバイダが一箇所に設置されている場合，スケジューリングは一箇所で行うべきである．

3.1.2 モバイルリザベーションでのリアルタイム特性

モバイルリザベーションでのタスクスケジューリングは，ユーザのサービス完了までの待ち時間を減少させることを目的とする．ユーザの待ち時間は，ユーザがサービススポットに到着した時刻とタスクを完了した時刻によって決められる．したがって，モバイルリザベーションにおけるスケジューリングの正しさは，ユーザがサービススポットに到着した時刻とタスクを完了する時刻に依存する．このようなスケジューリングは**リアルタイムスケジューリング**と呼ばれる．リアルタイムスケジューリングでは，タスク完了の目標時刻として，**デッドライン**を設定し，それを使用してタスクをスケジューリングする．リアルタイムスケジューリングには，デッドラインを過ぎた

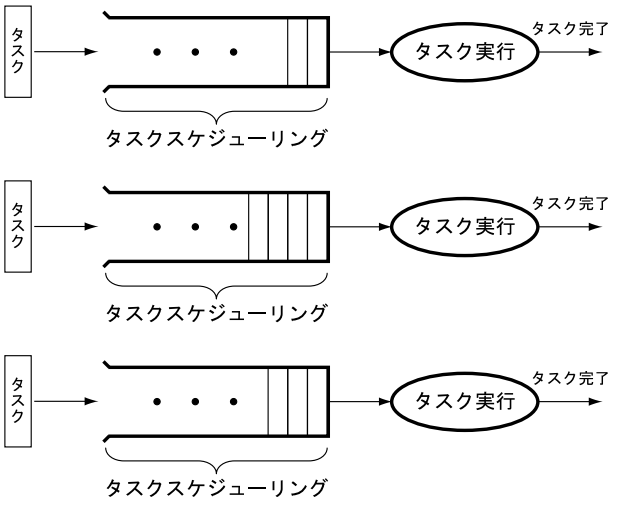

図 3.2: サービスプロバイダが複数の場合の並列タスクスケジューリング

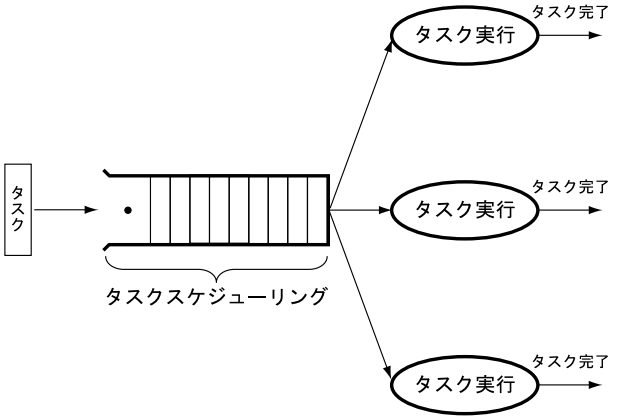

図 3.3: サービスプロバイダが複数の場合の直列タスクスケジューリング

場合のタスク完了価値の下がりかたによって二つに分類される．タスク完了の価値がデッドラインを境に無くなるものを**ハードリアルタイム**，下がるもののゆるやかでデッドライン後でもタスク実行価値があるものを**ソフトリアルタイム**と呼ぶ．

モバイルリザベーションでは，ユーザのサービススポットへの到着時刻をデッドラインに定める．到着時刻にタスクが完了していない場合は，サービススポットでタスク完了を待つことを想定する．そのため，モバイルリザベーションで適用するスケジューリングはソフトリアルタイムスケジューリングである．

3.1.3 デッドライン変動とその要因

前項で，モバイルリザベーションにおけるタスクスケジューリングは，到着時刻をデッドラインに定めたソフトリアルタイムスケジューリングであることを示した．到

着時刻であるデッドラインは，最短経路での残り移動距離と，移動平均速度を使って予測される．この場合，ユーザの残り移動距離や移動速度が変化すると，デッドラインが変動する．本項では，このデッドライン変動要因を分析する．

まず，ユーザのサービススポットへの移動手段としては，歩行を想定する．これらでの移動における主なデッドラインの変動要因として以下の三点をあげる．

- 移動経路変更
- 移動速度変更
- 長期停止（寄り道）

移動経路は，ユーザの現在位置からサービススポットまでの最短経路を使用する（2.5節）．ユーザがその経路から外れた場合に**移動経路変更**が起きる．予測に使用した移動経路は，最短経路であるため，経路が変更すると最短経路から外れることになる．図3.4で移動経路変更によるデッドライン変動を示す．黒線矢印はユーザの現在位置からサービススポットまでの最短経路を示し，これが予測経路として使われる．しかし，ユーザが次の交差点で経路変更すると，予測経路が破線矢印となり，残り移動距離が変化して，デッドラインが変動する．

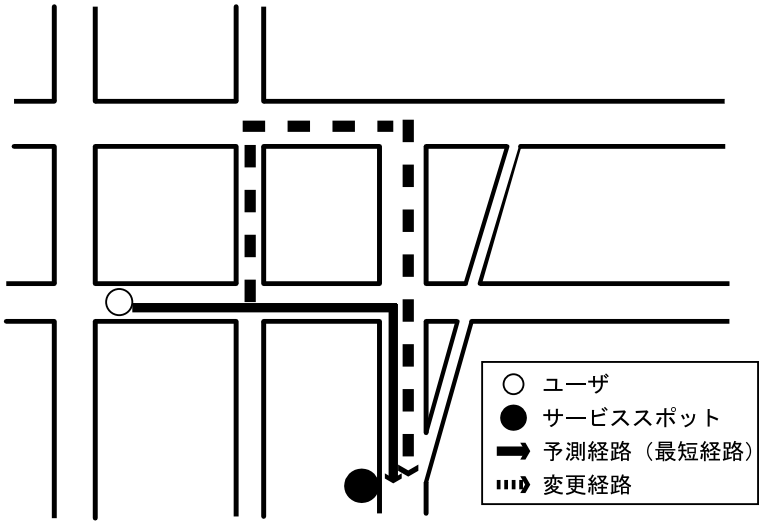

図 3.4: 移動経路変更

また，**移動速度変更**によってもデッドラインが変動する．移動速度が上昇すれば，移動平均速度も上昇し，デッドラインが前進する．移動速度が下降すると，移動平均速度も下降し，デッドラインが後退する．

モバイルリザベーションにおいては，ユーザ自身はデッドラインを意識することなく移動する．したがって，通常の予約方式にくらべて，ユーザは時間に縛られずに，自由な移動が行なわれると考えられる．自由な移動とは，たとえば，他の仕事をしてか

ら移動したり，移動しながらサービススポット以外の別の場所に寄り道したりすることである．これらによって，長い間残り移動距離が減少しないことを**長期停止**と呼ぶことにする．長期停止によって，デッドラインが変動する．

3.2 既存のスケジューリングアルゴリズム適用に関する考察

本節では，ソフトリアルタイムスケジューリングにおける代表的なスケジューリング方式を説明し，それらをモバイルリザベーションへ適用した場合に発生する問題点を考察する．

3.2.1 先行研究

ソフトリアルタイムスケジューリングの先行研究として，既存のスケジューリング方式である，EDF[9][10]，Least Slack Time[11]，Value Function[13][14] をあげる．

EDF (Earliest Deadline First)

デッドラインの早いタスクから順に実行していくスケジューリング方式である．図 3.5 で EDF スケジューリングの例を示す．タスク J_1 からタスク J_5 までのデッドライン d_1 から d_5 が黒棒で表され，タスク実行時間が白線で表される．$d_1 < d_2 < d_3 < d_4 < d_5$ ならば，$J_1 \rightarrow J_2 \rightarrow J_3 \rightarrow J_4 \rightarrow J_5$ の順序でタスクが実行される．

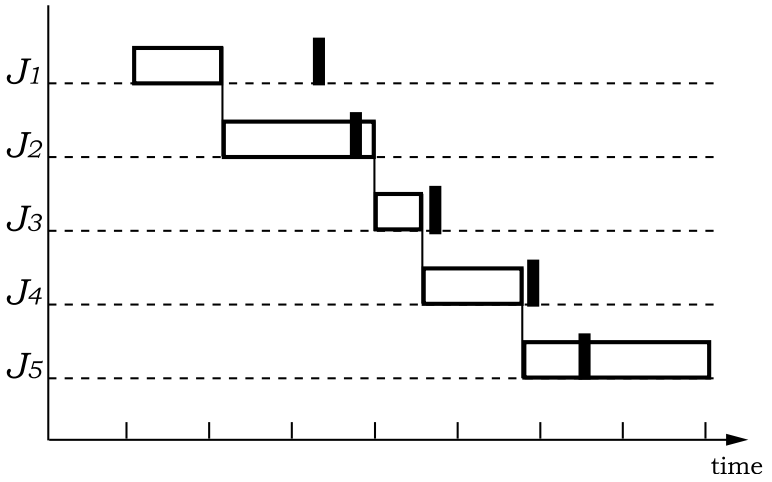

図 3.5: EDF (Earliest Deadline First) スケジューリング

Least Slack TIme

Least Slack Time は余裕時間の少ないタスクから順に実行していくスケジューリング方式である．余裕時間とは，現在時刻から，デッドラインをミスしないためにタスクを開始する時刻の最も遅い時刻までの時間で，デッドラインからタスク長と現在時刻を引いた時間であらわされる．全てのタスクのタスク長が皆同じ場合，EDF と同じ順序でスケジューリングされる．タスク長がばらばらの場合，EDF とは異なる順序でスケジューリングされることがある．

Value Function

Value Function によるスケジューリング方式では，タスクの終了価値を Value として表現し，これを最大にするような順序でタスクをスケジューリングする．モバイルリザベーションにおいて，Value を待ち時間によって減少するユーザの満足度とし，待ち時間 0 の場合の Value を k，予測したデッドラインを d，d を 1 秒超えてタスクが実行完了した時の Value 減少を p とすると，時刻 t にタスクが完了する場合の Value である $V(t)$ は (3.1) 式で表せる．このときの Value Function を示す横軸 時間，縦軸 Value のグラフは図 3.6 のようになる．

また Value Function は，ユーザによって 3.1 式の k を変更して，ユーザごとの優先度を変えられるという特徴がある．たとえば，あるユーザがサービスに対して通常のユーザよりも料金を多く払っている場合は，k の値を大きくすると，そのユーザを優先した方が Value が高くなるのでそのユーザのタスクが優先的にスケジューリングされる．

$$V(t) = \begin{cases} k & \text{if } t \leq d \\ k - p(t-d) & \text{otherwise} \end{cases} \tag{3.1}$$

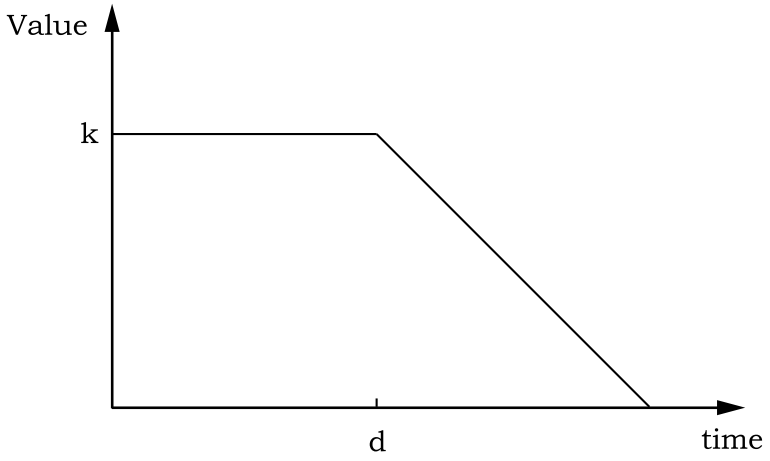

図 3.6: ソフトデッドライン Value Function

また，Value Function では特徴的なサービスを定式化してスケジューリングできる．たとえば，ファーストフード店のサービスが機械化された自動調理販売機では，調理を行うというタスクの完了がユーザのサービススポットへの到着より充分早い場合，調理したものが冷めてしまう．このように，待ち時間を減らすために早くタスクを完了すると，サービスアウトプットの品質が低下してしまうようなサービスについて，Value をサービス品質の低下によるユーザの満足度減少を考慮して，図 3.7 のような Value Function を設定できる．これにより，スケジューラ内にタスクが一つしかない場合でも，それをすぐには実行せず，Value が最大となるデッドラインの地点でタスクが実行完了するようにスケジューリングされる．

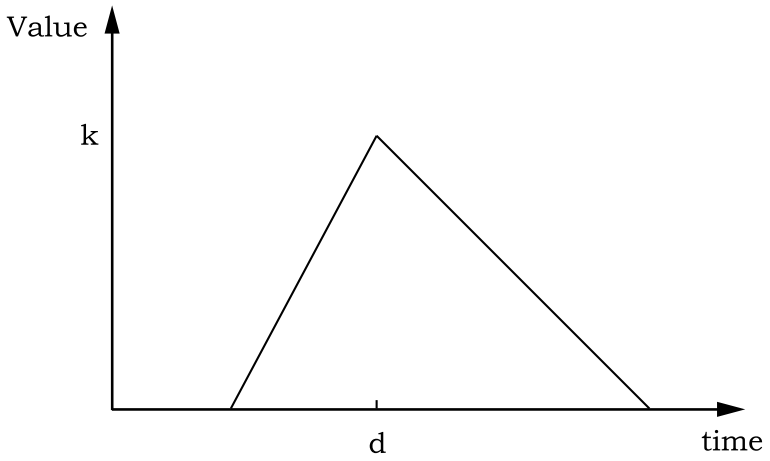

図 3.7: 自動調理販売機の Value Function

3.2.2 問題点

デッドラインをユーザのサービススポットへの到着時刻に設定して EDF，Least Slack Time，Value Function スケジューリングを適用すれば，リクエストの到着順にタスクを実行する FIFO (First In First) 方式に比べて待ち時間は減少する．しかし，モバイルリザベーションでは，3.1.3 項で説明したデッドラインの変動が起こる．これは，デッドラインを使用したスケジューリングにとって大きな問題である．たとえば，ある時刻にユーザ A，B のタスクがあり，デッドラインの早いユーザ A を優先して A→B とスケジューリングしても，その後ユーザ A のデッドラインが後退して，ユーザ B が先に到着してしまい，結果的に B→A とタスクを実行した方が待ち時間が減少するというケースがある．この現象に対応し，デッドラインが後退しすい A のタスクは優先度を下げて，それよりも遅いデッドラインであるがデッドラインが守られやすい B のタスクを優先的にスケジューリングして，B→A で実行することが望まれる．しかし，

EDF，Least Slack Time，Value Function で既存に提案されている方式では，ともにモバイルリザベーションにおけるデッドライン変動に対応しておらず，モバイルリザベーションに適したスケジューリング方式とは言えない．

3.3　本章のまとめ

本章では，モバイルリザベーションにおけるタスクスケジューリングについて説明した．特に，モバイルリザベーションではデッドライン変動が発生し，それを考慮したスケジューリングが提案されていないことを述べた．

第4章

VVFモデル設計

モバイルリザベーションに適したスケジューリングアルゴリズムであるVVF (Variable Value Function) を設計する．モバイルリザベーションにおけるユーザの移動をモデル化し，それにあったスケジューリング方式としてVVFを提案した（4.1，4.2）．VVFは一つのタスクについても，残り移動距離の変化によって動的にValue Functionの傾きを変え，さらに，デッドラインが変動するたび，新たなデッドラインを取得して，Value Functionの定義域を変更する（4.3）．本章の最後にVVFによるスケジューリングを，計算量の多い全解探索をせずに求める近似方法について検討する（4.4）．

4.1 VVF 概要

VVF はモバイルリザベーションにおけるデッドライン変動に対応して，Value Function 関数の定義域と傾きを変更するモデルである．デッドライン変動をモデル化し，それに対応して，あるタスクについて動的に Value Function の傾きと定義域を変更してスケジューリングする．

4.2 HaT (Hare and Tortoise) モデル

3.1.3 項にて，モバイルリザベーションにおける主なデッドライン変動要因は，**移動経路変更**，**移動速度変更**，**長期停止**であることを示した．本節では，これらのそれぞれの変動規則をモデル化する．

4.2.1 前提

モデル化の前提は以下の四点である．

- 移動手段は歩行
- 移動速度はユーザごとに異なる
- 全てのユーザはリクエスト発行後直接来るとは限らない
- サービススポットに到着したユーザはサービスが完了するまでサービススポットから離れない

移動手段は全ユーザ統一されているが，移動速度はユーザによって走ったり歩いたりしてばらばらであることを想定する．また，ユーザはサービススポットに到着する時間にとらわれずに自由に移動しながらサービススポットへ到着することを想定する．さらに，ユーザが一度サービススポットに到着するとそこから離れることはないと想定した．

4.2.2 デッドライン変動規則

デッドライン変動要因の発生について以下の規則を設定した．

移動経路変更規則

移動経路変更は，交差点や道の分岐点ごとに起こる可能性がある．本モデルでは，交差点や道の分岐点を経路変更点と呼ぶ．ユーザが経路変更点に遭遇し，移動経路を変更すると，デッドラインは後退する．それを図 4.1 を使って説明する．移動予測経路は，ユーザの現在値からサービススポットまでの最短経路であり，

図 4.1 では，黒線矢印である．図 4.1 の交差点でユーザが右折すると，右折してからの最短経路である破線矢印が移動予測経路となる．このとき，移動経路変更は，最短経路から外れることを意味するため，必ず経路変更前の残り移動距離よりも，経路変更後の残り移動距離のほうが長くなる．デッドラインは残り移動距離を平均移動速度で割って求められるため，平均移動速度が変わらないならば，残り移動距離が増えることによってデッドラインは後退する．

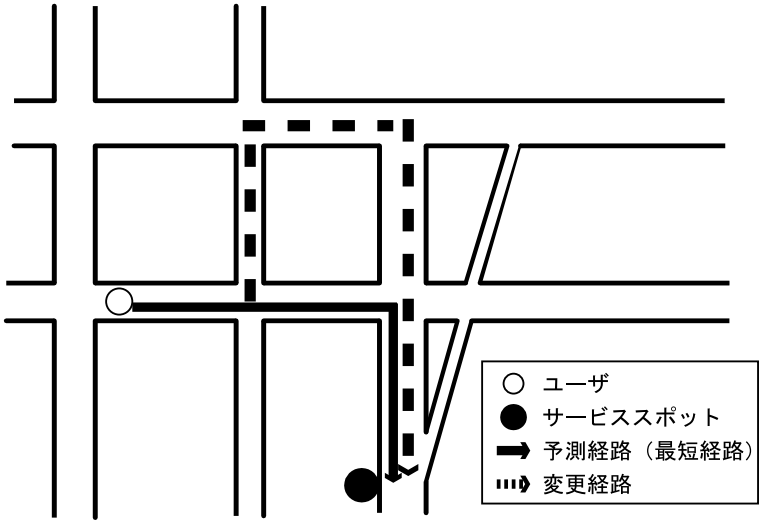

図 4.1: 移動経路変更によるデッドライン後退

どのユーザにとっても，移動経路のなかに経路変更点が多いほど，経路変更する確率が上がる．これは残りの移動経路のなかに経路変更点が多いほど，デッドラインが後退する確率があがることを意味する．そして，経路変更点の数は残り移動距離が長いほど多くなると考えられる（図 4.2）．このことから，**予測した残り移動距離が長いほど移動経路の変更が発生しやすく，デッドラインが後退しやすい**と定義した．

ユーザごとの移動経路の変更しやすさのモデルについては二通り提案する．一つ目は，ユーザごとに経路変更点で経路変更する確率が異なり，経路変更しやすいユーザは経路変更点に遭遇するたびに同確率で何度も経路変更を繰り返すモデルである．二つ目もユーザごとに経路変更点で経路変更する確率が異なるが，ユーザは経路変更するたびに次の経路変更点で経路変更する確率が落ちていくモデルである．

移動速度変動規則

移動速度変動は，移動距離にかかわらず，同速度での移動時間が増えると起こりやすくなることを想定した．そのうえで，ある時点での移動速度が移動平均速度と同じ時は，移動速度が下がると，移動平均速度も下がりデッドラインが後退し，移動速度が上がると，移動平均速度が上がってデッドラインが前進する．

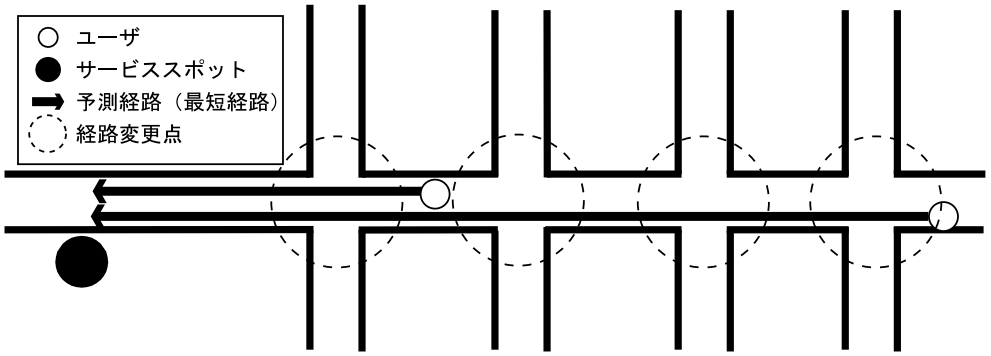

図 4.2: 経路変更点の距離による増加

また，ある時点での移動速度が移動平均速度を上回っている場合，移動速度が変わらなければ，平均移動速度が上がってデッドラインが前進し，移動速度が下がると，移動平均速度も下がりデッドラインが後退，移動速度が上がると，移動平均速度が上がってデッドラインが前進する．

逆に，ある時点での移動速度が移動平均速度を下回っている場合，移動速度が変わらなければ，平均移動速度が下がってデッドラインが後退し，移動速度が下がると，移動平均速度がさらに下がってデッドラインが後退，移動速が上がると，移動平均速度が上がってデッドラインが前進する．

この三個のパターンによってデッドライン変動が起こる確率をすべて同率として，**残り移動時間が長いほど移動速度変更が起きやすいがデッドラインが前進する確率，デッドラインが後退する確率は同率**と定義した．

長期停止発生規則

長期停止は，ユーザが移動中に寄り道することによって起こる．店や別のサービスなど，寄り道をする場所（以後，長期停止点）があった際に，寄り道が起こる可能性があり，長期停止点で長期停止する．長期停止によってデッドラインは後退する．長期停止点は残り移動距離が長いほど増加する（図 4.3）と想定し，**残り移動距離が長いほど長期停止が発生しやすく，デッドラインが後退しやすい**と定義した．

ユーザごとの長期停止のしやすさのモデルについては移動経路変更のしやすさモデルと同様の二通りを提案する．一つ目は，ユーザごとに長期停止点で長期停止する確率が異なり，長期停止しやすいユーザは長期停止点に遭遇するたびに同確率で何度も長期停止を繰り返すモデルである．二つ目もユーザごとに長期停止点で長期停止する確率が異なるが，ユーザは長期停止するたびに次の長期停止点で長期停止する確率が落ちていくモデルである．

以上の三点のデッドライン変動要素に関するモデルとそれがデッドラインに与える影響を，表 4.1 にまとめた．移動経路変更は，残り移動距離に比例して発生率が上が

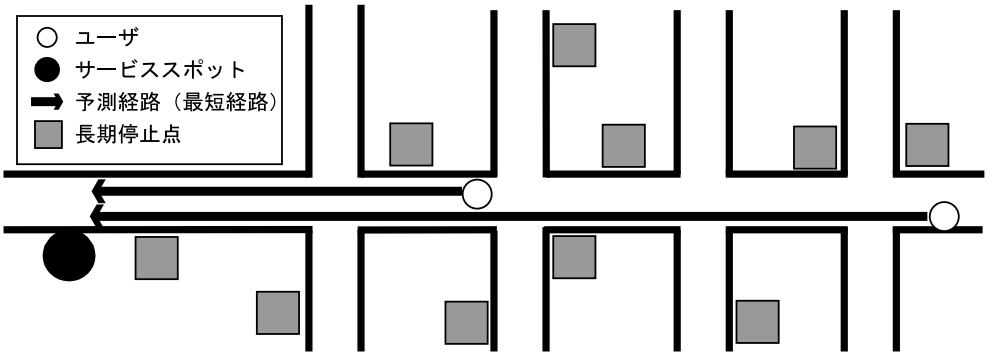

図 4.3: 長期停止点の距離による増加

り，デッドラインが後退しやすい．移動速度変更は，残り移動時間に比例して発生率が上がるが，デッドラインが前進，後退が同じ確率で起こる．長期停止は，残り移動距離に比例して発生率が上がり，デッドラインが後退する．以上より，**デッドラインは残り移動距離に比例して，後退しやすい**と結論づけた．

たとえば，同一のデッドラインを予測したユーザ A とユーザ B がいた場合，移動平均速度がユーザ A> ユーザ B で，残り移動距離がユーザ A> ユーザ B ならば，ユーザ A のデッドラインのほうがユーザ B のデッドラインよりも後退しやすい．移動平均速度が速いがデッドラインが後退しやすいユーザ A を “うさぎ”，移動平均速度が遅いがうさぎよりもデッドラインが後退しづらいユーザ B を “かめ” に見立て，本モデルを **HaT (Hare and Tortoise: うさぎとかめ) モデル**と呼ぶ．

表 4.1: HaT モデル

デッドライン d 変動要因	変動要因発生率	変動前のデッドライン d と変動後の d’ との関係
移動経路変更	**残り移動距離**に比例して発生率が上昇	全ての場合で d < d’
移動速度変更	**残り移動時間**に比例して発生率が上昇	d < d’ or d > d’ or d = d’ が同確率で発生
長期停止	**残り移動距離**に比例して発生率が上昇	全ての場合で d < d’

4.3 VVF の設計

モバイルリザベーションにおける HaT モデルを想定し，ユーザの移動変化によるデッドライン変動を考慮したスケジューリングモデルである VVF (Variable Value Function)

を設計した．まず，VVF の要件を整理し，その後 VVF の特徴である，Value Function モデルの傾き変更と定義域変更について述べる．

4.3.1 VVF 設計の要件

VVF 設計の要件をまとめる．

全ユーザの待ち時間の総計を減少させる

待たされるユーザの総数を減少させるのではなく，全ユーザの待ち時間総計を減少させるものである．

サービススポットへ到着後のユーザのタスクは到着順に実行される

ユーザごとの優先度が同じで，サービススポットに到着したユーザが複数いた場合，到着順で実行され，これが逆転することはない．

4.3.2 Value Function 基本形の設定

本節で，Value Function の基本形を設定する．Value Function の基本形とは，デッドラインが確定した場合の Value Function の形である．Value Function の基本形では，

Value：待ち時間とサービスアウトプットの品質低下によって減少するユーザの満足度

と設定する．待ち時間が無く，サービスアウトプットの品質が低下していない場合 Value は最大である．モバイルリザベーションでは，サービススポットに到着して初めてデッドラインが確定する．したがってサービススポットに到着後の Value Function の形を定義する．

1. **到着時刻より前のタスク実行完了でサービスアウトプットの品質が低下しないもの**

   想定するサービスとして，プリンタ，CD・DVD へのデータコピー機，証明書発行機がある．これらのサービスでは到着前にタスクを完了させておけば，Value は最大である．

   秒単位のタスク実行完了時刻を t，そのときの Value を $V(t)$，デッドラインを d，ユーザごとの優先度を k，1 秒の待ち時間による Value 減少率を p と設定し，デッドライン確定時の Value Function を，4.1 式および図 4.4 のように設定した．ユーザは，サービスを利用完了するまでは待ち続けることを想定した．

   全ユーザのタスク長が同一な場合，タスク長がユーザごとに異なっている場合，タスクの実行を途中で中断できない（タスクが**ノンプリエンプタブル**）場合，タ

スクを途中で中断して，他のタスクを実行可能（タスクが**プリエンプタブル**）な場合全ての場合で 4.1 および図 4.4 を適用する．

$$V(t) = \begin{cases} k & \text{if } t <= d \\ k - p(t-d) & \text{otherwise} \end{cases} \tag{4.1}$$

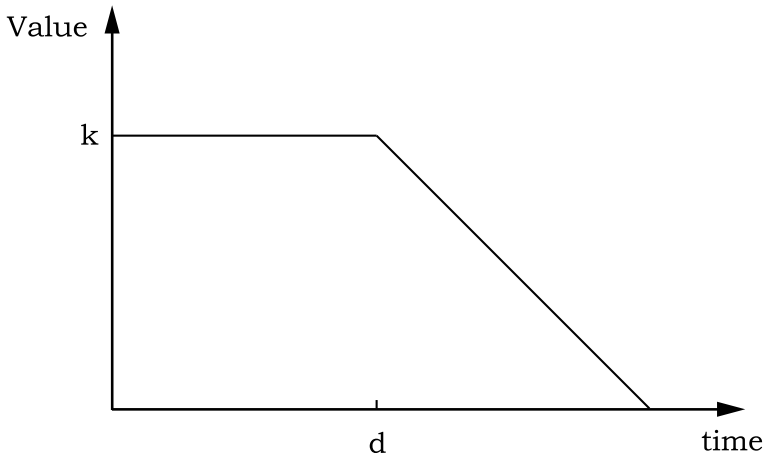

図 4.4: サービス アウト プットの品質が低下しない Value Function

2. **到着時刻より前のタスク実行完了でサービスアウトプットの品質が低下するもの**

   想定するサービスとして，自動調理販売機がある．サービスアウトプットの品質が時間の経過によって低下するため，到着時にタスクが実行完了することが望ましい．サービスアウトプットの品質低下が一定以上になるとサービスプロバイダはそれをユーザに提供できなくなり，サービスアウトプットを廃棄して新しいサービスアウトプット生成の処理をはじめる．

   この Value を数式表すと 4.2 となる．デッドライン d より一定時間前の時刻 S 以前でタスクを実行完了した場合，品質低下によってユーザにサービスアウトプットを提供できなくなるため，S 以前のタスク実行価値は $-\infty$ とし，S 以前でタスクを実行完了するスケジューリングをしないよう設定した．

$$V(t) = \begin{cases} -\infty & \text{if } t < S \\ k & \text{else if } S <= t <= d \\ k - p(t-d) & \text{otherwise} \end{cases} \tag{4.2}$$

4.3.3 Value Function の傾き変更モデル

モバイルリザベーションでは，ユーザが実際にサービススポットに到着するまでデッドラインは確定しない．この場合，予測したデッドラインに加え，“デッドラインの変

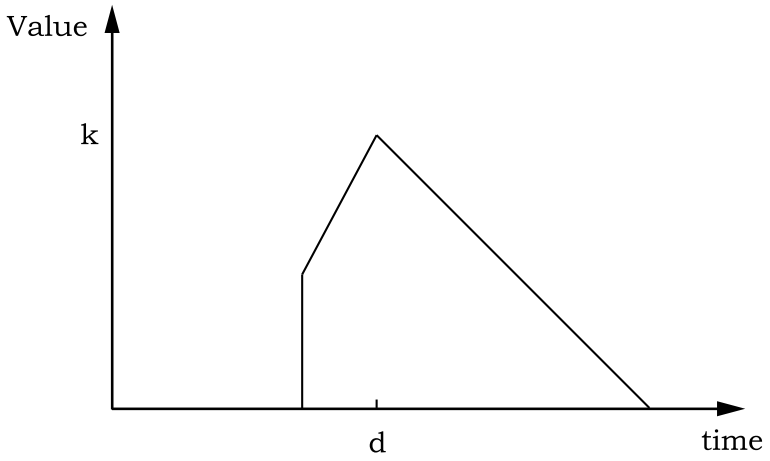

図 4.5: サービスアウトプットの品質が低下する Value Function

動しやすさ” といったパラメータを使ってスケジューリングしてデッドライン変動に対応する方法がある．VVF では，デッドラインの変動規則を示した HaT モデルにもとづいて，前項で設定した Value Function の傾きを変化させて，デッドライン変動に対応する．

以下で，Value Function の傾きを変化させる考え方を述べる．HaT モデルでは，残り移動距離の長いタスクほどデッドラインが後退しやすい．この規則にしたがって，VVF では残り移動距離にしたがって Value Function の傾きを変動させる．どのようにして傾きを変化させるかを図 4.6 で示す．図 4.6 の上図太線は，デッドライン d を持ち，残り移動距離 L_1 が 300m のタスク J_1 の Value Function を示し，下図太線は J_1 と同じデッドライン d を持ち，残り移動距離 L_2 が 1000m のタスク J_2 の Value Function を示す．どちらも同じデッドライン d を持つため，Value Function の基本形は同じである．しかし HaT モデルは，残り移動距離が長いほど，デッドラインは後退しやすいと言う性質を持つため，残り移動距離の長い J_2 のほうが J_1 よりもデッドラインが後退しやすい．図 4.6 の灰帯は，デッドラインが変動して Value 関数がずれることをあらわす．灰帯の濃い場所ほどその場所に Value 関数がずれやすい．タスク J_2 は残り移動距離がタスク J_1 よりも大きいため，デッドラインが後退して Value 関数が後ろにずれやすく，濃い灰帯がデッドライン以後に濃い灰帯の場所が多い．VVF ではこの灰帯の分布を反映して，より後ろにずれやすいタスクの Value Function の傾きを緩やかにする（図 4.6 破線）．この結果，Value Function の傾きが J_1 の傾き $<J_2$ の傾きとなり，同じデッドラインを持つ場合でも J_1 が優先される．このようにして VVF では Value Function の傾きをユーザの残り移動距離によって変化する．

以上のように，残り移動距離による優先度を傾きによって変化させるために，VVF では 4.1 式を基本形とした式である，4.3 式によって Value をあらわす．4.3 式での変数は，4.1 式で用いられた t，$V(t)$，d，k，p に加え，ユーザの残り移動距離として l，サービス側が想定するリクエスト発行の最長距離を L であらわした．そして $\frac{L-l}{L}$ を**デッドラインの信頼度**とした．デッドラインの信頼度は最小が 0，最大が 1 の間であらわされ

$l \geq L$ のときは 0，ユーザがサービススポットに到着して $l = 0$ となると信頼度が最大となり 1 となる．デッドライン信頼度は基本形の傾き p の係数である．信頼度が 0 の時は VVF の傾きは $\frac{L-l}{L}p = 0$ となり，いつタスクを実行完了してもよいと判断される．逆に，信頼度が 1 の時は，傾き $\frac{L-l}{L}p = p$ となり，基本形と同じ傾きとなる．VVF の基本形は，ユーザがサービススポットに到着してデッドラインが確定した時に適用されるため，$l = 0$ となったときに傾きが基本形と同じとなり，ユーザごとの優先度が皆同じならば完全に到着順でタスクが実行される．

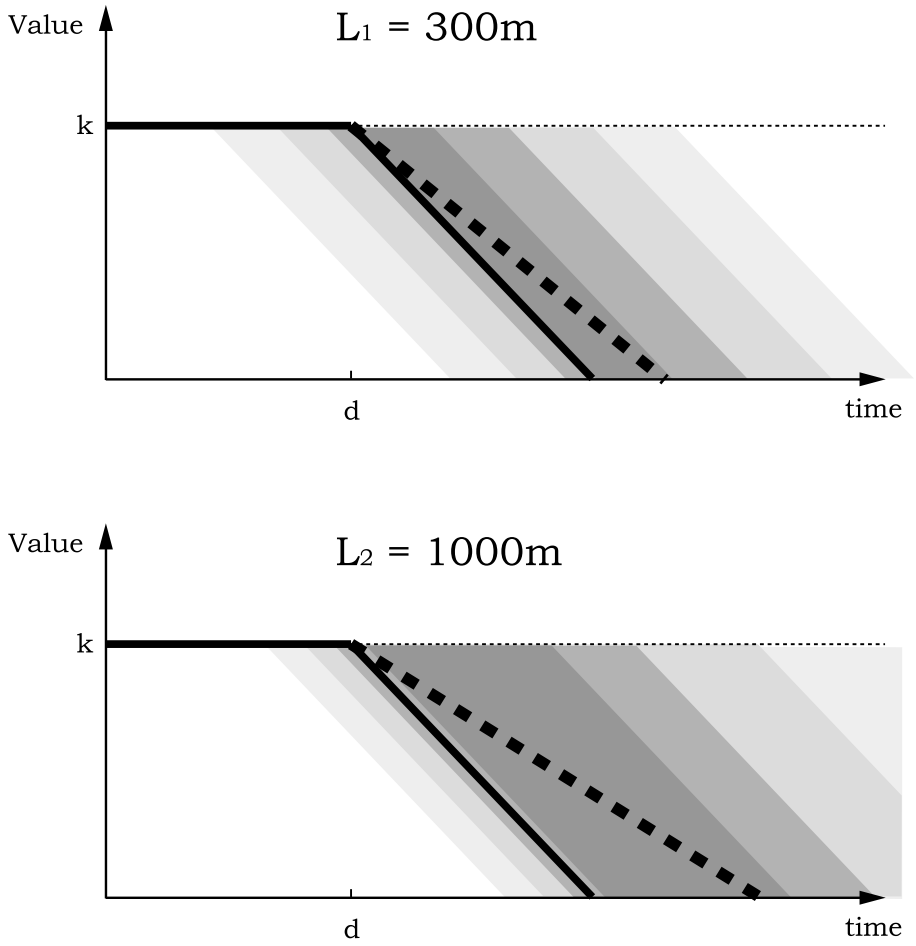

図 4.6: VVF の傾き変動

$$V(t) = \begin{cases} k & \text{if } t \leq d \quad \text{or if } L \leq l \\ k - \frac{L-l}{L}p(t-d) & \text{otherwise} \end{cases} \tag{4.3}$$

サービス提供者が定めたリクエスト発行の最長距離 $L = 500$ のときの，あるユーザ A についての VVF の状態遷移を図 4.7 を使って考える．ユーザ A の残り移動距離 l が 500 のとき，4.3 式よりデッドライン信頼度は 0 となり傾きが 1 である．ユーザ A がサービススポットに向かい移動して，l=400，300，200，100 となるにしたがい，デッ

ドライン信頼度が上がっていき，傾きが急になる．そして時刻 d に l=0 となり，サービススポットに到着するとデッドライン信頼度は 1 となり，Value Function の基本形である図 4.7 太線になる．このように，同じユーザについても時間の経過とともに Value Function が変化する．

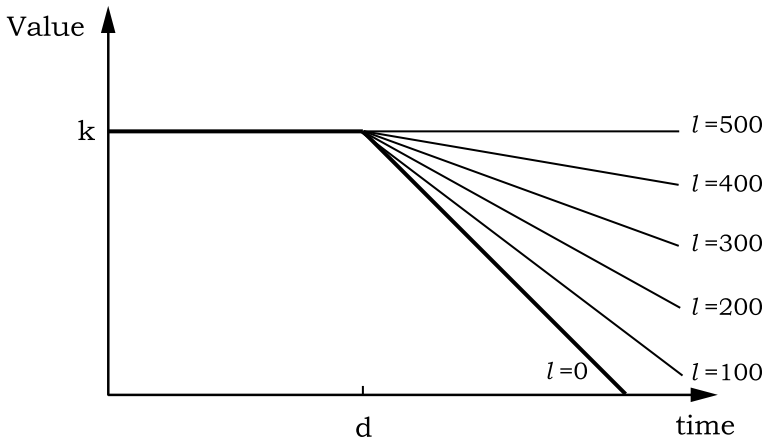

図 4.7: 残り移動距離変化による VVF の状態遷移

4.3 式によって，デッドラインが遅いタスクがデッドラインの早いタスクより優先される優先度逆転が起こる．優先度逆転を図 4.8 で示す．タスク J_1 のデッドライン d_1 よりも遅いデッドライン d_2 をもつタスク J_2 があり，タスク J_2 の残り移動距離 l_2 がタスク J_1 の l_1 よりも小さい時，Value Function の傾きは J_2 のほうが J_1 よりも急になる．このとき，デッドライン後の時刻で J_1 の Value Function と J_2 の Value Function が交わる．交点以降の時刻 t でタスク J_1 および J_2 を完了した場合，遅いデッドライン d_2 を持つ J_2 のほうが Value の損失が大きく J_2 の優先度が上がる．

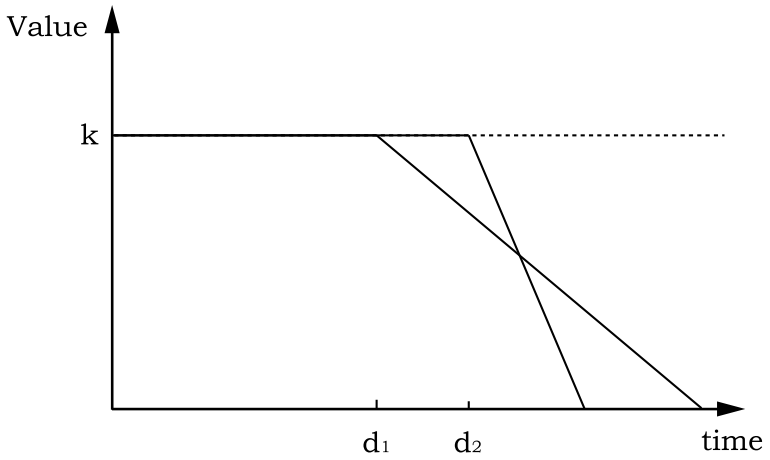

図 4.8: VVF の優先度逆転

4.3.4 Value Functionの定義域変更モデル

VVF ではValue Function の傾きを変更するほかに，デッドラインを頻繁に更新することによってもデッドライン変動に対応する．より最新に取得したデッドラインのほうが実際の到着時刻に近いと考えられるためである．VVF では，ユーザのデッドラインが変動した場合に，(1) ユーザからのデッドライン変動の通知，(2) 再スケジューリングによってValue Function の定義域を変更し，対応する．

ユーザからのデッドライン変動の通知

VVF では，ユーザのデッドラインが変動すると，それをユーザの携帯端末から再度送信することを想定する．

再スケジューリング

再スケジューリングは，タスクの実行順序を決め直す作業のことである．新たにリクエストが到着し，実行すべきタスクが増えた場合のほか，既存のタスクのデッドラインが変動した場合も行う必要がある．再スケジューリングを行うポリシは，タスクがノンプリエンプタブルかプリエンプタブルかによって異なる．

タスクがノンプリエンプタブルな場合，実行中のタスクを中断できないため，必ずタスク完了ごとに，最新のデッドラインを使って再スケジューリングを行う．タスクがプリエンプタブルな場合，実行中のタスクが完了した場合のほかに，新たなリクエストが到着した場合，既存のタスクのデッドラインが変動した場合にも再スケジューリングを行う．

4.4 VVFの近似解

VVF では，再スケジューリング毎に待ち行列にはいった全てのタスクの並び方について全解探索を行ない最もValue の高い並び方の先頭にあるタスクを次に実行すべきタスクとして決定した．しかし，全解探索はn 個の待ち行列があった場合に探索回数がn!のオーダとなり，n が増えるにしたがって計算量が爆発的に増大する．計算量が多く，最適解を見つけるのに5 分，10 分，1 時間，1 日とかかるのであれば，VVF はモバイルリザベーションに適用できない．そのため，スケジューリングアルゴリズムでは，全解探索をすることなく，できるだけ全解探索と同じ解を求める近似法が重要である．

VVF では，タスク完了毎に再スケジューリングを行なうため，スケジューリング時には次に実行すべきタスクを決定するのみで良い．この性質に着目し，VVF の全解探索の近似について，タスク長が固定の場合について以下の二方法を提案する．

4.4.1　タスク長固定の VVF 近似 1

次（一番目）に実行すべきタスクをとして，二番目に実行した場合の Value が最も少なくなり被害が大きいタスクを選ぶ方法である．

待ち行列のなかのタスク A から F が表 4.2 のようなデッドラインと傾きを持っていた場合を例としてこの近似法を説明する．タスク A から F のタスク長はすべて 10 とする．待ち時間 0 の場合の Value k を $k = 0$ とすると，タスク A，B，C，D，E，F を二番目に実行した場合の Value はそれぞれ以下のようになり，Value が最小となる A が実行される．この方法によって，次に実行するタスクを決定する際にかかる計算量は，待ち行列 n の場合 n-1 である．

$$\begin{aligned} V_A &= 0 - 4(20 - 7) \\ &= -52 \end{aligned}$$

$$\begin{aligned} V_B &= 0 - 5(20 - 12) \\ &= -40 \end{aligned}$$

$$\begin{aligned} V_C &= 0 - 2(20 - 15) \\ &= -10 \end{aligned}$$

$$\begin{aligned} V_D &= 0 - 3(20 - 10) \\ &= -30 \end{aligned}$$

$$V_E = 0$$

$$V_F = 0$$

表 4.2: タスクのデッドラインと傾き例

タスク	A	B	C	D	E	F
デッドライン d	7	12	15	10	20	22
傾き $-\frac{L-l}{L}p$	-4	-5	-2	-3	-1	-6

4.4.2　タスク長固定の VVF 近似 2

デッドラインが最も早いタスクとそれよりも傾きが急なタスクで，一，二番目の順序でタスクを実行した時の Value を検討する方法である．この方法は主に三個の手続きによって成る．以下でそれを説明する．

まず，手続き FindEarliestDeadline で全てのタスクのデッドラインを比較し最もデッドラインの早いものを選び出す．

```
FindEarliestDeadline()
        ed ← FirstTask()
        repeat until all deadline checked
                now ← NextTask()
                if Deadline(now)<Deadline(ed) then ed ← now
        return ed
```

次に，手続き FindEarliestDeadline によって選ばれたタスク ed よりも傾きの小さいタスクを取りだして優先度を決める手続き FindRapidSlope を行う．手続き FindRapidSlope では，まず現在の最優先タスクをあらわす max に最もデッドラインの早いタスクをいれる．そして，ed よりも傾きの大きなタスクをを探し，それを見つけると手続き CalAltOrder にそのタスクを現在の最優先タスクを手続き CalAltOrder にわたす．

最もデッドラインの早い ed に対して優先度逆転を起こす可能性があるのは ed よりも傾きの小さなタスクのみであるため，ed より傾きの大きなタスクを探す．ed よりも傾きの大きなタスクでは，ed のデッドラインが先のため，優先度逆転が起こり得ない（図 4.9）．

```
FindRapidSlope(ed)
        max ← ed
        repeat until all slope checked
                if(Slope(now) s < Slope(ed))
                        max ← CalAltOrder(now, max)
        execute max
```

```
CalAltOrder(a, b)
        aval ← ValueFirst(a)+ValueSecond(b)
        bval ← ValueFirst(b)+ValueSecond(a)
        if aval > bval return a
        else return b
```

手続き CalAltOrder では，渡された二つのタスクについて，それぞれを一番目，二番目に並べたときの Value を計算する手続き ValueFirst および手続き ValueSec-

OND を呼び，どちらのタスクを先に実行すると高い Value が得られるかを調べる．高い Value が得られる順番の VALUEFIRST() の引数を優先すべきタスクとして返す．もしどちらを先に実行しても得られる Value が同じならば，第二引数のタスクを返す．

手続き CALALTORDER によってどちらが優先すべきタスクか決められ，そのタスクが返却されると，手続き FINDRAPIDSLOPE ではそれを現在の最優先タスクとして max に代入する．この作業を，*ed* より傾きの小さなタスクが見つからなくなるまで繰り返す．これが完了すると現在の最優先タスク max を次に実行すべきタスクとして実行する．この近似法の実行にかかる計算量は，待ち行列 n 個の場合，最大 4n-5，最小 2n-3 である．

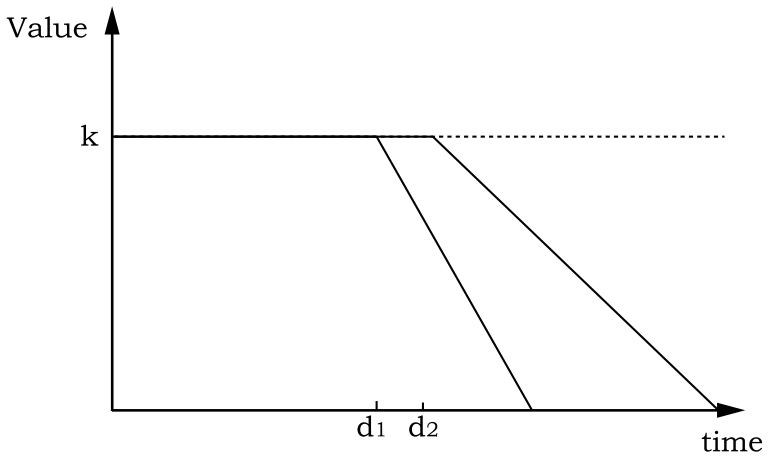

図 4.9: 優先度が逆転しないとき

4.5 本章のまとめ

本章では，モバイルリザベーションにおけるデッドライン変動に対応したスケジューリング方式である VVF を提案し，詳説した．モバイルリザベーションにおけるユーザの移動を HaT モデルとしてモデル化し，VVF をそれに対応させた．VVF は，Value Function の傾きをユーザの残り移動距離によって動的に変更し，さらに，複数回デッドラインを取得して，デッドラインが変動した場合は Value Function の定義域を変更する．本章の最後で，VVF でスケジューリングする際の全解探索の計算量を減らすために，タスク長が固定の場合の近似法を提案した．

第5章

シミュレーション

本論文では，シミュレーションによって VVF のスケジューリング性能を評価する．まず，前章で提案した VVF 近似方式を評価し，優れた結果を出した近似法を使って VVF と他のスケジューリング方式を比較するシミュレーションを行う．シミュレーションの概要を説明し，シミュレーション結果を述べ考察する．

5.1 シミュレーションの目的

本節で，VVF のスケジューリング性能を評価するためにシミュレーションを行う．VVF と他のスケジューリング方式とをシミュレーション結果から比較し，VVF の有用性を示す．

5.2 シミュレーション環境

VVF シミュレーションの実装には，C 言語を使用した．ただし，リクエスト発生時等で使用するランダム関数は，C 言語のものでは再現性が完全でないため，再現性が保証されている ns-2 (Network Simulator version 2)[18] の C++言語コードを移植した．再現性の保証は同一の状況における，複数のスケジューリング方式の比較要件である．

表 5.1 に VVF シミュレータの実装環境を示す．

表 5.1: 実装環境

項　　目	説　　明
ハードウェア	AMD Athlon (TM) XP 2000+ (1666.20-MHz)
オペレーティングシステム	FreeBSD-4.7 RELEASE
実装言語	C 言語 (一部 C++言語，gcc version 2.95.4)

5.3 VVF 近似評価

4.4 節で示した VVF の二つの近似法の精度を評価し，以後のシミュレーションで全解探索のかわりに使用するものを決める．

5.3.1 VVF 近似評価方法

VVF 近似 1，近似 2 が VVF の全解探索とどれだけ近い値を出すか評価を行った．評価方法としては，待ち行列に入っているタスクの数を VVF 全解探索で計算量が爆発しない 10 個，それぞれのタスクについてデッドラインを 0 から 100 までの整数，傾きを 0 から-1 までの負の小数の間で無作為に決定し，VVF 全解探索と VVF 近似 1，および VVF 近似 2 が導き出した次に実行すべき要素の解が等しいかを比べた．これを複数回行い，VVF 全解探索と VVF 近似とが異なる確率が一定となった時点での確率を VVF 近似誤差として使用する．VVF 全回探索との近似誤差が少ないほうの近似法をつかってシミュレーションを行う．

5.3.2 VVF 近似評価結果

評価を行った結果が安定したのは，試行数が 5000 回以上からであった．このことから，5000 回の試行実行時の VVF 全解探索と VVF 近似 1，VVF 近似 2 との誤差を評価した．評価結果を表 5.2 で示す．

表 5.2: VVF 近似評価（5000 回の試行による）

手法	VVF 全解探索と同一解の確率	待ちノード数 n 個の時の計算量
VVF 全解探索	100 (%)	$n!$
VVF 近似 1	75.38 (%)	$n-1$
VVF 近似 2	87.66 (%)	最小：$2n-3$ 最大：$4n-5$

VVF 近似 1 では，75.38%の確率で VVF 全解探索と同じ解を導き出した．これに対し VVF 近似 2 では，87.66%の確率で VVF 全解探索と同じ解を導き出した．以上の結果より VVF 近似 2 を近似法として以後のシミュレーションで全回探索のかわりに採用する．

5.4 シミュレータの基本設計

ユーザをノードとし，一次元の軸上で移動するシミュレータを作成した．

本節では，シミュレータでのリクエストの到着モデル（ノードの発生モデル）の設定を述べ，発生したノードの動作設計を述べる．動作設計では，シミュレータでのノードの基本動作と，HaT モデルに基づいた経路変更，移動速度変更，長期停止の実現方法を説明する．

シミュレーションで使われるパラメータを表 5.3 で示す．

5.4.1 リクエスト到着の設定

サービスに対するリクエストの到着モデルを**ポアソン到着**で表す．リクエストの到着とは，シミュレータでは新たなノードが発生することを意味する．

ポアソン到着は待ち行列理論においても，M/D/1，M/M/1 モデルや M/M/S モデルで採用されているリクエスト到着のモデルである．ポアソン到着の要件は以下の三点である．

- 定常性　　同一幅の時間区間あたりの到着は時刻に依存せず定常
- 独立性　　各到着は独立
- 希少性　　微小時間内の到着が希少

表 5.3: シミュレーションのパラメータ

パラメータ	説　　明	単　　位
NODE_NUM	ユーザ数	人
LAMBDA	リクエスト発生（到着）率	確率
TASK_LENGTH	タスク長（全ユーザ同じ）	秒（s）
FIELD_LENGTH	リクエスト発行距離最大値	距離（m）
MIN_SPEED	最低移動速度	速度（m/s）
MAX_SPEED	最高移動速度	速度（m/s）
ROUTE_CHANGE_M	経路変更点の平均設置間隔	距離（m）
ROUTE_CHANGE_MAX_M	経路変更での移動距離増加最大値	距離（m）
SIDE_TRIP_M	長期停止点の平均設置間隔	距離（m）
SIDE_TRIP_MAX_S	長期停止点での最長停止時間	時間（s）

シミュレーションではタスクを一個処理する間に新たに出現するノード数を LAMBDA とし，これを 0.1 から 1.0 の間に設定してシミュレーションを行った．この値が 1.0 を超えると，リクエスト到着数がタスク処理能力を超え，スケジューリングが破綻するためである．

5.4.2　ノードの動作設計

図 5.1 でノードの移動に関する node_action_type 構造体を示す．

```
struct node_action_type{
  int     distance;
  float   ave_speed   // average moving speed
  int     now_speed;  // speed now  walk MIN_SPEED-MAX_SPEED stop 0
  snipped...

  float   rc_prob;    // route change probability 0.0-1.0
  float   st_prob;    // side trip probability  0.0-1.0
  RNG     act_r;      // ns2 random struct type
};
```

図 5.1: node_action_type 構造体

ノードの基本動作

図 5.2 を使ってノードの基本動作を示す．ノード A はサービススポットである 0m から FIELD_LENGTH までの間の残り移動距離 l_0 の地点に移動速度 v_0 で出現し，リクエストを発行する．

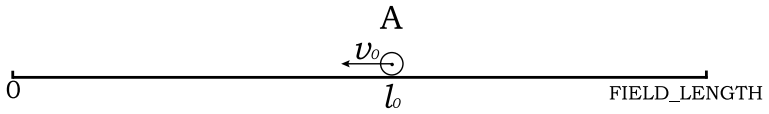

図 5.2: ノードの基本動作

シミュレータでは，ノード出現時に node_action_type 構造体の distance に 1 から FIELD_LENGTH までのいずれか，now_speed に MIN_SPEED から MAX_SPEED までのいずれか，経路変更のしやすさを示す rc_prob と長期停止のしやすさを示す st_prob にそれぞれ 0.0 から 1.0 までの数値のいずれかが一様分布によって決められ，初期値として入れられる．経路変更，長期停止などのイベントが起こらない時は，ノードは毎秒現在の移動速度 now_speed 分だけ残り距離 distance を減らして移動する．

経路変更

図 5.3 で経路変更のシミュレータでの再現法を示す．経路変更では，図 5.3 の (a) あらわされるように，ノードが交差点や道の分起点で予測していた最短経路である黒線からはずれる．はずれた経路の中の最短経路である図 5.3 破線の残り移動距離 l' が，それまでデッドライン予測に使われていた黒線の残り移動距離 l に変わって使われる．このとき必ず $l < l'$ である．経路変更によって，それまでの残り移動距離 l に新たに距離が足され，残り移動距離が l' となる（図 5.3(b)）．

シミュレータでは，経路変更点の平均設置間隔である ROUTE_CHANGE_M の距離を移動する間に一回の確率で経路変更点が発生する．経路変更点が発生すると，ノードは経路変更のしやすさを示す rc_prob の確率をもとに経路変更するか否かをきめる．経路変更が決まると 0 から ROUTE_CHANGE_MAX_M の間の整数が一様分布に基づいて決定され，残り移動距離の増加分として現在の残り移動距離 distance に加えられる．

長期停止

長期停止点の平均設置間隔である SIDE_TRIP_M の距離を移動する間に一回の確率で長期停止点が発生する．長期停止点が発生すると，ノードは長期停止のしやすさを示す st_prob の確率をもとに長期停止するか否かを決定する．長期停止すると，現在速度 now_speed が 0 となり，残り移動距離は減少しない．長期停止は，1 秒から最大 SIDE_TRIP_MAX_S 秒の間のいずれかの時間が一様分布によって決定され，その時間分続く．

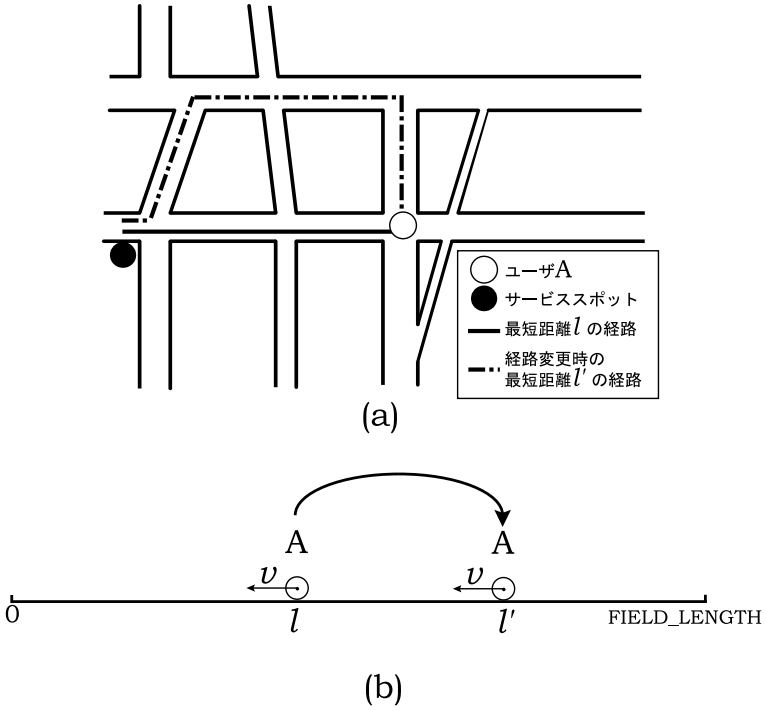

図 5.3: 経路変動 (a) のシミュレーションでの再現 (b)

5.5 評価方針

VVF の定義域変動と傾き変動による待ち時間減少効果を明らかにする．

シミュレーションでは，各パラメータに定数を設定して行った．一回のシミュレーションにおいて，ノード数は 200 個とし，これらのタスクが全て完了するまで行う．一回のシミュレーションを VVF，到着後のリクエスト発行，FIFO，Value Function の四パターンのスケジューリング方式適用について行う．この際，ns2 から移植したランダム関数に同じシードを入れて,「5 番目にリクエストを発行したノードが 10 秒後でスピードが 1 から 3 に変動する」といった状況を完全に再現して，スケジューリング方式以外は全て同じ状況にする．これをパラメータを変えるごとに 200 回づつ行い，**ノード一個当たりに課される平均待ち時間**を以下の比較対象と比較する．

比較対象

到着後のリクエスト発行（RAA：Request After Arrival）

モバイルリザベーションを行わない場合である．ユーザは到着後サービスにリクエストを発行する．ユーザがサービススポットに到着時に，サービスを完了していないユーザがいた場合，そのユーザのサービス完了を待ってリクエストを発行する．ユーザの到着順にスケジューリングされる．本論文ではこのスケジューリング方式を，以後 **RAA** と呼ぶ．

FIFO スケジューリング

遠隔からリクエストを発行し，リクエストの到着順にタスクを実行していく場合である．したがってユーザのサービススポット到着順ではない．

Value Function スケジューリング（VF）

遠隔からリクエストを発行し，リクエスト発行時にともに到着したデッドラインに基づいてスケジューリングする場合である．本シミュレーションでは，タスク長を固定，ユーザ自身の優先度は皆同じであるため，Value Function で決定されるスケジューリングは EDF と同じである．本論文では以後，**VF** とする．

5.6 評価

本節では，シミュレーションによって VVF 評価を行い，結果を考察する．サービスとしてはデジタル写真印刷，CD へのデータコピー，DVD へのデータコピーを想定する．

想定サービス：デジタル写真印刷

サービス例としてデジタル写真印刷を想定し，シミュレーションを行って評価した．デジタル写真印刷にかかる時間を 30 秒，ユーザの歩行スピードを 1 から 4m/s を想定した．これらを含めたパラメータ設定を表 5.4 で示し，シミュレーション結果を図 5.4 で示す．

また，表 5.4 でのパラメータ設定と比べ，リクエスト発行距離最大値 FIELD_LENGTH が大きい場合のシミュレーションを行った．この場合のパラメータ設定を表 5.5，シミュレーション結果を図 5.5 で示す．

さらに，表 5.5 でのパラメータ設定よりノードが経路変更，長期停止しやすい場合についてののシミュレーションを行った．現実世界では，サービスまでの距離が同じ場合でも郊外に設置去れたサービスよりも繁華街に設置されたサービスのほうがサービススポットへの到着までにユーザが経路変更，長期停止しやすいと考えるためである．この場合のパラメータ設定を表 5.6 で示し，シミュレーション結果を図 5.6 で示す．

表 5.4: デジタル写真印刷サービスパラメータ設定 1

パラメータ	設定値
TASK_LENGTH	30
LAMBDA	0.1-1.0
FIELD_LENGTH	300
MIN_SPEED	1
MAX_SPEED	4
ROUTE_CHANGE_M	100
ROUTE_CHANGE_MAX_M	30
SIDE_TRIP_M	100
SIDE_TRIP_MAX_S	30

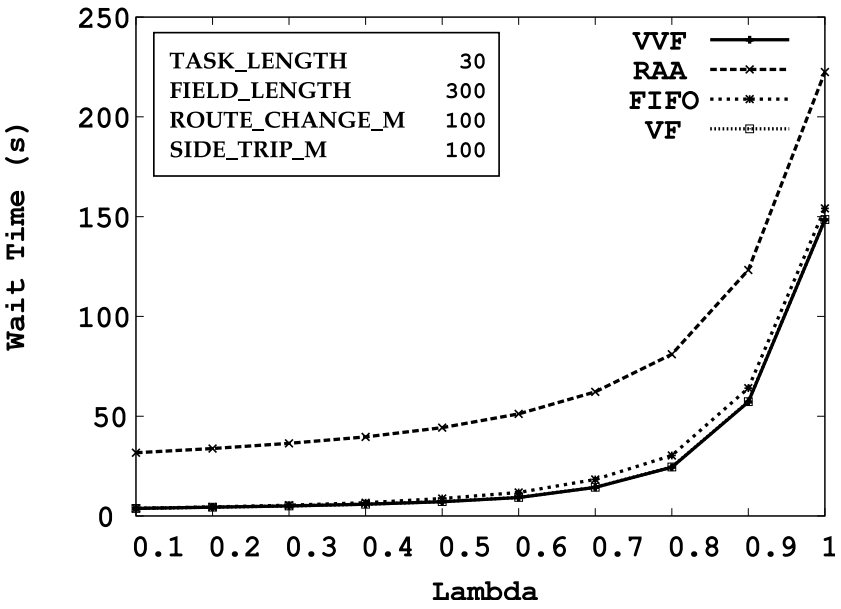

図 5.4: デジタル写真印刷サービスシミュレーション結果 1

表 5.5: デジタル写真印刷サービスパラメータ設定 2

パラメータ	設定値
TASK_LENGTH	30
LAMBDA	0.1-1.0
FIELD_LENGTH	600
MIN_SPEED	1
MAX_SPEED	4
ROUTE_CHANGE_M	100
ROUTE_CHANGE_MAX_M	60
SIDE_TRIP_M	100
SIDE_TRIP_MAX_S	30

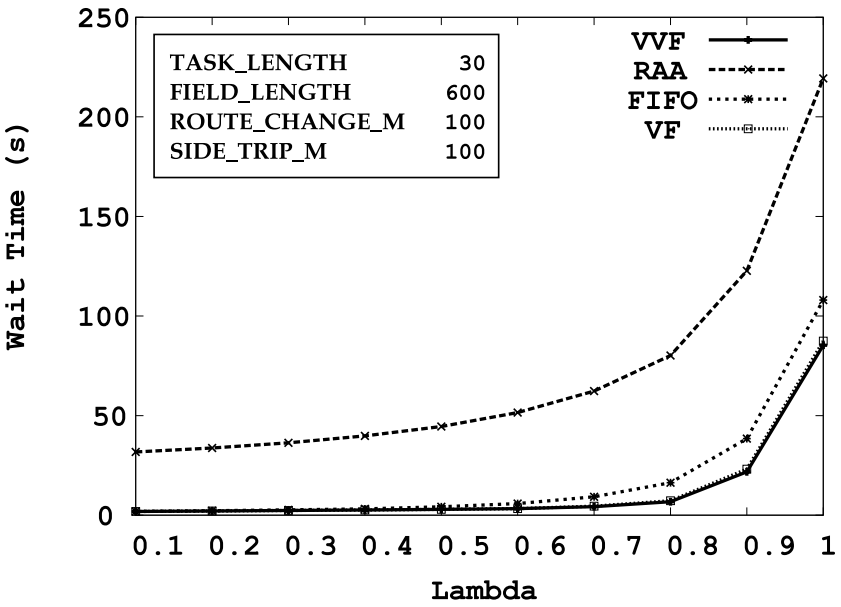

図 5.5: デジタル写真印刷シミュレーション結果 2

表 5.6: デジタル写真印刷パラメータ設定 3

パラメータ	設定値
TASK_LENGTH	30
LAMBDA	0.1-1.0
FIELD_LENGTH	600
MIN_SPEED	1
MAX_SPEED	4
ROUTE_CHANGE_M	50
ROUTE_CHANGE_MAX_M	60
SIDE_TRIP_M	50
SIDE_TRIP_MAX_S	30

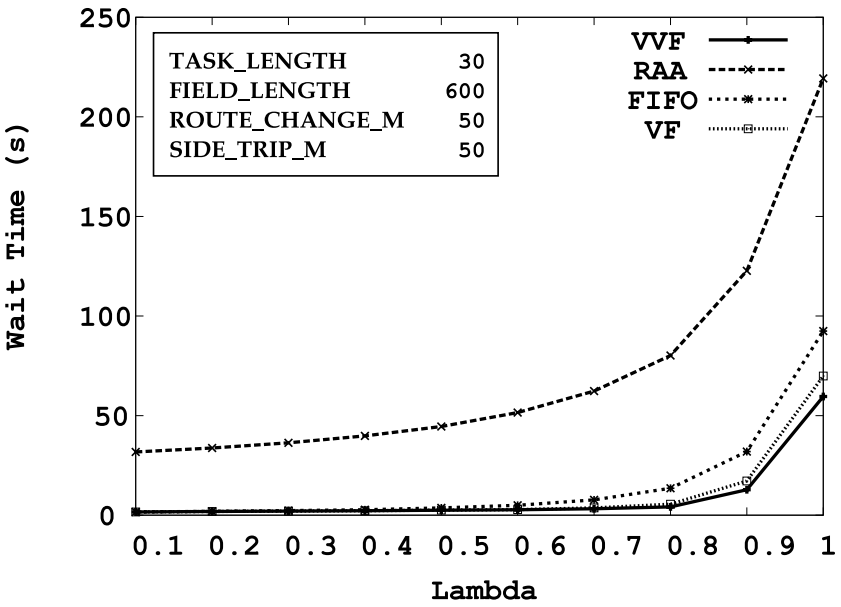

図 5.6: デジタル写真印刷シミュレーション結果 3

5.6.1 想定サービス：CD へのデータコピー

二つ目のサービス例として CD-R，CD-RW へのデータコピーを想定し，シミュレーションを行って評価した．この場合のパラメータ設定を 5.7、シミュレーション結果を図 5.7 で示す．

表 5.7: CD へのデータコピーサービスパラメータ設定

パラメータ	設定値
TASK_LENGTH	300
LAMBDA	0.1-1.0
FIELD_LENGTH	2000
MIN_SPEED	1
MAX_SPEED	4
ROUTE_CHANGE_M	100
ROUTE_CHANGE_MAX_M	200
SIDE_TRIP_M	100
SIDE_TRIP_MAX_S	300

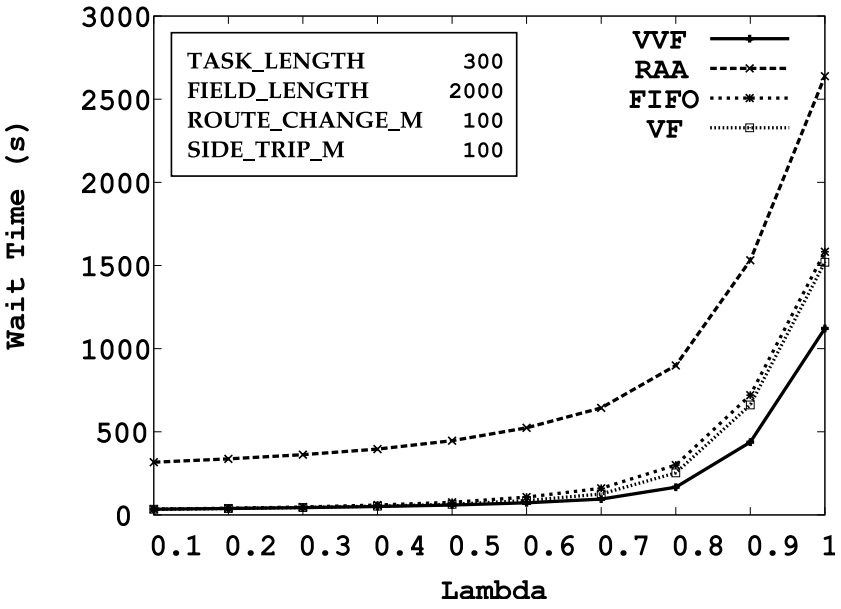

図 5.7: CD へのデータコピーサービスシミュレーション結果

5.6.2 想定サービス：DVD へのデータコピー

二つ目のサービス例として DVD-R 等へのデータコピーを想定し，評価した．本ケースはタスク長を 1800 秒に設定した．これを含めたパラメータ設定を 5.8 で示し，シミュレーション結果を図 5.8 で示す．

表 5.8: DVD へのデータコピーサービスパラメータ設定

パラメータ	設定値
TASK_LENGTH	1800
LAMBDA	0.1-1.0
FIELD_LENGTH	7200
MIN_SPEED	1
MAX_SPEED	4
ROUTE_CHANGE_M	100
ROUTE_CHANGE_MAX_M	720
SIDE_TRIP_M	100
SIDE_TRIP_MAX_S	1800

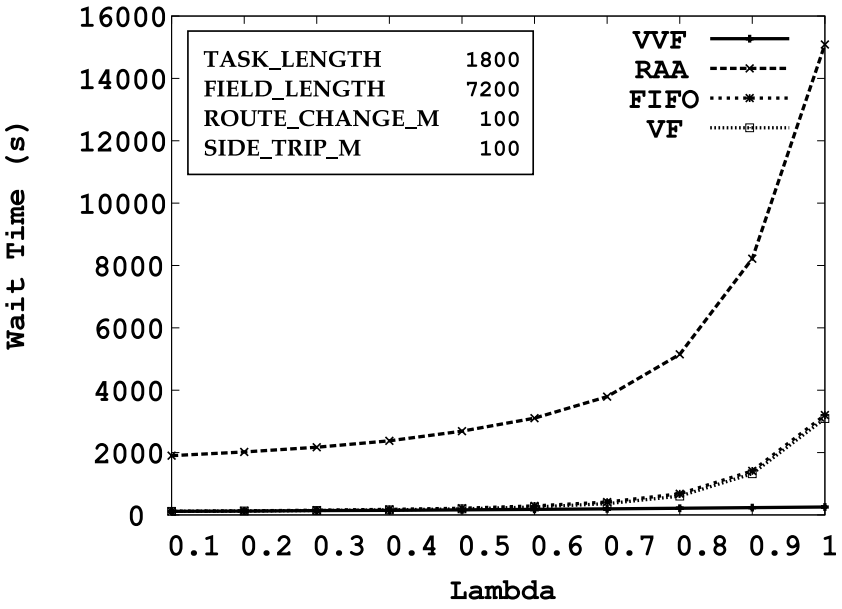

図 5.8: DVD へのデータコピーサービスシミュレーション結果

5.6.3　シミュレーション結果考察

サービススポット到着後のリクエスト発行では LAMBDA の大小に関わらず必ず TASK_LENGTH 以上は待たされる．一方，モバルリザベーションを行う FIFO，VF，VVF では，サービス到着後のリクエスト発行に比べ待ち時間が大幅に減少し，LAMBDA が小さい時には TASK_LENGTH を下回る．

モバイルリザベーションを行う VVF と FIFO，VF とを比較すると，図 5.5，図 5.6，図 5.7，図 5.8 全ての場合で，VVF の平均待ち時間減少が FIFO，VF よりも小さくなることはなく，これらと同等か，これらよりも待ち時間減少が大きかった．

VVF と FIFO，VF との間に待ち時間の差が生まれたのは，LAMBDA 0.7 からであった．このとき，図 5.4，図 5.5，図 5.6，図 5.7，図 5.8 全ての場合で，VVF は FIFO よりも待ち時間平均が減少している．また，図 5.4 と図 5.5，図 5.7，図 5.8 より，FIELD_LENGTH と TASK_LENGTH が大きいほど，VVF は FIFO よりも待ち時間減少が大きくなる．

VVF と VF を比較した場合，TASK_LENGTH が短いデジタル写真印刷サービスよりも TASK_LENGTH の長い CD や DVD へのデータコピーサービスにおいてより VVF のほうが待ち時間減少が大きくなる． これは，TASK_LENGTH が長いとノードがリクエストを発行してからサービススポットに到着するまでの間に経路変更や長期停止が起こりやすいためである．また，FIELD_LENGTH が同じ場合で，経路変更と長期停止のしやすく設定した図 5.6 は，図 5.5 よりも VVF の VF と比較した場合の待ち時間減少が大きい．

5.7　本章のまとめ

本章では VVF のスケジューリング性能を評価するためのシミュレーションについて説明し，シミュレーション結果を示した．

第6章

結論

本章では，本論文のまとめと今後の課題について述べる．

6.1 今後の課題

本節では，VVF の今後の課題をあげる．

6.1.1 サービスアウトプットの品質が低下するサービスへの VVF 近似

到着時刻より前のタスク実行完了でサービスアウトプットの品質が低下するものについて近似式を考案し，実際に VVF を利用できるようにする必要がある．

6.1.2 最大待ち時間の設定

今後の課題の二つ目はサービスを利用する時の待ち時間の最大値を設定した場合の Value Function への対応である．ある一定時間以上待つと，ユーザがサービスの利用をあきらめて利用をキャンセルする場合である．このときのユーザにとっての Value Function の形は図 6.1，図 6.2 であらわされる．この場合，サービス提供者側は自分の利益を守るために，ユーザのキャンセルを少なくするというポリシを新たに追加してスケジューリングする必要がある．また，ユーザごとにタスク長がばらばらの場合，タスク長の長いユーザ程最大待ち時間が長く，Value Function の基本形の傾きがゆるやかであると考えられる．VVF をこの最大待ち時間設定に対応させることも今後の課題とする．

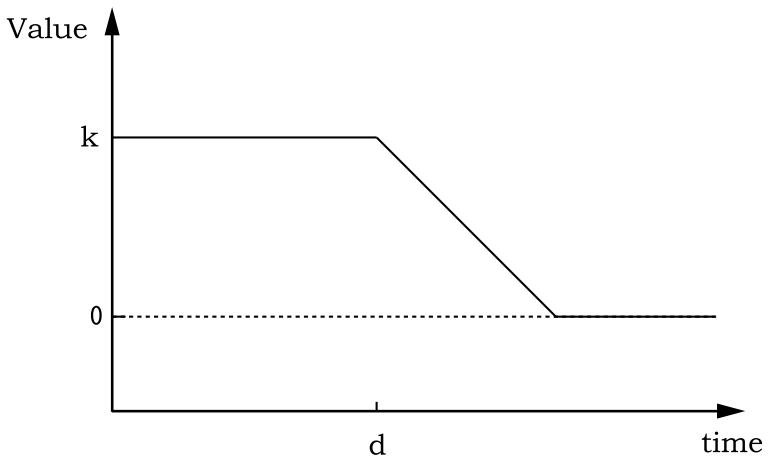

図 6.1: 最大待ち時間が設定された Value Function 1

6.1.3 最大ソート数問題への対応

最大ソート数を考慮したスケジューリングも VVF の課題である．最大ソート数とはユーザの到着前に完了し，生成されたサービスアウトプットをソートして保持でき

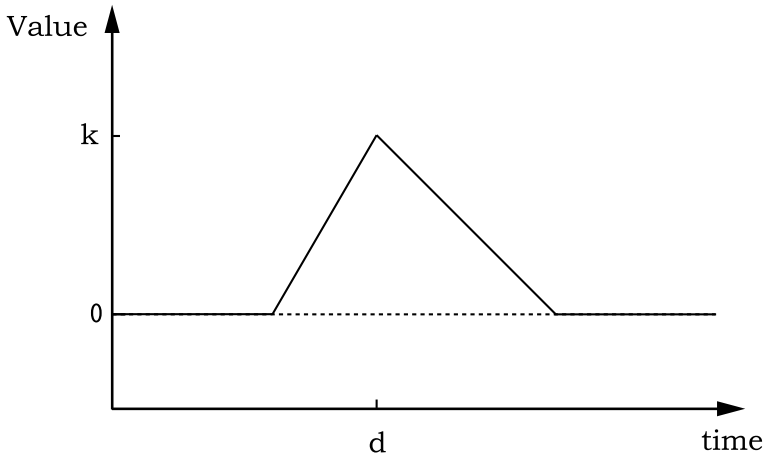

図 6.2: 最大待ち時間が設定された Value Function 2

る最大数である．モバイルリザベーションではハードウェアの機能要件としてソート機能をあげている＞最大ソート数が無限大のハードウェアが誕生するとは考えにくく，サービスアウトプットを保持できる箱（図 6.3 ではソート箱とした）の個数は限られていると考えられる．ソート箱の個数に考慮して，ソートできないサービスアウトプットを生成しないようにするスケジューリングポリシを付加することで，VVF はより現実的なスケジューリングアルゴリズムとなる，

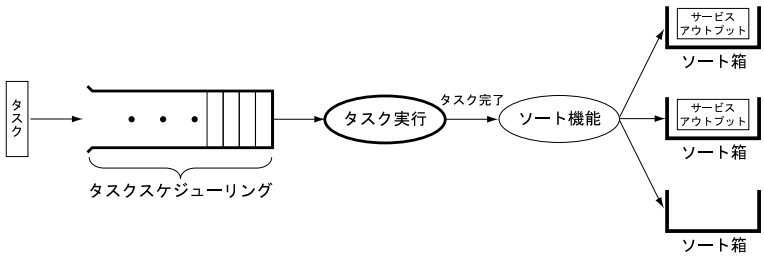

図 6.3: 最大ソート数が決められたタスクスケジューリング

6.2 まとめ

本論文では，ユビキタスコンピューティング環境において，ユーザの移動先に設置されたサービスに対して遠隔から予約を入れ，到着前にサービスの処理を開始させておくことでユーザの待ち時間を減少させるモバイルリザベーションを提案した．モバイルリザベーションでは，予約内容に加えて，ユーザの所持するセンサから取得したデータをもとに予測したユーザのサービス場所への到着時刻を送信する．サービス側で，その到着時刻予測をデッドラインとし，デッドラインに基づいてスケジューリングを行う．

ユーザの移動変化によってデッドラインは変動しやすい．しかし，既存のスケジューリング方式では，デッドラインの変動を考慮していない．本研究では，モバイルリザベーションにおけるユーザの移動状態変化によるデッドライン変動を HaT モデルとしてモデル化し，そのモデルに適したスケジューリングアルゴリズムである VVF を提案した．VVF では，Value Function の傾きと定義域を動的に変更することで，デッドライン変動に対応した．そして，VVF のスケジューリング決定の際の計算量を減らすために，タスク長が全て同じ場合について VVF の近似を提案し，VVF を実用に耐えうるアルゴリズムとした．

VVF の傾き変更と定義域変更による待ち時間減少を評価するためにシミュレーションを行った．シミュレーション結果として，パラメータを変更した全ての場合について，VVF ではモバイルリザベーションにおいて既存のスケジューリング方式をさいようした場合と比べ同等以上の待ち時間減少がみられた．特に，タスク長が長いサービスの場合，ユーザのリクエストを発行する距離が長い場合，経路変更や長期停止がしやすい場合で，多くのユーザが頻繁にサービスを利用する場合に，VVF は既存のスケジューリング方式よりも待ち時間減少が大きかった．

謝辞

本研究を進めるにあたり，御指導を頂きました慶應義塾大学環境情報学部教授徳田英幸博士に深く感謝致します．また，研究を進めるにあたり，貴重な御助言をいただきました慶応義塾大学政策・メディア研究科助教授西尾信彦氏に感謝します．

さらに，研究の議論のために多くの時間を割いていただいた徳田研究室 move!プロジェクトの永田 智大氏，村瀬 正名氏，権藤 俊一氏，堀江 裕隆氏，守分 滋氏，榊原 寛氏，小泉 健吾氏，柴田 悟郎氏，米澤 拓郎氏，米山 遼太氏に感謝します．また同様に本研究に多くの時間を割いていただき，叱咤激励をくださった徳田研究室 ACE プロジェクトの岩谷晶子氏に感謝致します．

2003 年 2 月 21 日
鈴木 源太

参考文献

[1] M. Weiser, "The computer for the twenty-century" Scientific American

[2] IEEE 802.11 Wireless LAN Working Group.
http://grouper.ieee.org/groups/802/11/.

[3] Bluetooth SIG, Inc. http://www.bluetooth.org/.

[4] NTTDoCoMo Inc.，"i モード全般"，http://www.nttdocomo.co.jp/i/index.html

[5] 永田 智大，西尾 信彦，徳田 英幸，"ASAMA: 適応的なサービス利用管理機構" 情報処理学会論文誌 6，2001，Vol.42, 6,P1557-1569

[6] T. Howes and M. Smith. "The ldap application program interface"
RFC 1823, 08/09, 1995.

[7] Yukihiko Kirihara, Jun'ichi Yura, Jin Nakazawa, and Hideyuki Tokuda
"PRONA: A Proactive Support System for Networked Appliances" IEEE International Workshop on Networked Appliances(IWNA) January 2002 pp.145-154

[8] Kenta Matsumiya, Soko Aoki, Masana Murase, and Hideyuki Tokuda
"Zero-stop Authentication: Sensor-based Real-time Authentication System",
Real-Time and Embedded Coputing Systems and Applications(RTCSA2003), February 2003

[9] R.W.Conway,W.L.Maxwell. and L.W.Miller. *Theory of Scheduling.*
Addison-Wesley, 1967.

[10] M.L.Dertouzos. Control Robotics: The Procedural Control of Physical Proccesses. In*Proceedings of the IFIP Congress*, pages 807-813, August 1974.

[11] A.K.Mok. *Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment* PHD thesis, Massachusetts Institute of Technology, May 1983.

[12] R.McNaughton. Scheduling with Deadlines and Loss Functions.
Management Science,6(1):1-12, October 1959.

[13] R.McNaughton. Scheduling with Deadlines and Loss Functions.
Management Science,6(1):1-12, October 1959.

[14] A.Schild and I.J.Fredman. Scheduling Tasks with Deadlines and Non-linear Loss Functions.
Management Science,9(1):73-81, October 1962.

[15] 萩原春生，中川健次 “情報通信理論 1 —符号理論 待ち行列理論—” 森北出版.

[16] 牧本直樹 “待ち行列アルゴリズム—行列解析アプローチ—” 朝倉書店 2001.

[17] Giorgio C. Buttazzo “HARD REAL-TIME COMPUTING SYSTEM
Predictable Scheduling Algorithms and Applications”

[18] Fall,K.,and Varadhan, K. *ns Manual*. The VINT project, UC Berkeley, LBL, USC/IDI, URL: http://www.isi.edu/nsnam/ns/,May 2001. Work in progress.