修士論文

学術ネットワークの帯域予約サービスにおけるスケジューリングアルゴリズム

同志社大学大学院 工学研究科 情報工学専攻
博士前期課程 2008 年度 718 番

川　　考蔵

指導教授　三木 光範 教授

2010 年 1 月 23 日

Abstract

Quality of Service (QoS) is essential for satisfying various types of Internet application requirements by making best use of limited bandwidth. In particular, layer-1 bandwidth-on-demand (Layer-1 BoD) service, which dynamically allocates a certain amount of bandwidth to specific users according to their requests, has attracted a great deal of attention as an ideal Internet backbone QoS, and it has been employed in the Science Information networks, which emphasizes various aspects of service: Virtual Private Networks (VPN), QoS, and bandwidth on demand. In this service, when the user requires a certain amount of bandwidth between an arbitrary pair of source and destination nodes, part of the full bandwidth is temporarily assigned to the user, similar to dedicated links in traditional telecommunications. The scheduler is thus required to efficiently and fairly allocate network resources, and this strongly affects user satisfaction. At present, simple algorithms, such as First In, First Out (FIFO) policies, have been employed, which causes imbalanced utilization of network resources and unfairness. There have been few studies focusing on Layer-1 BoD. Here, we propose sophisticated scheduling algorithms that decide whether each request should be accepted or not, to make the best use of network resources on the Internet. The priority of user requests is optimized for various objectives, such as fairness, or maximum total bandwidth used.

目 次

1 序論

学術情報ネットワークなどの先進的なインターネットでは，増大するユーザからの需要 (遠隔会議，遠隔医療などのリアルタイム性が要求される大規模データ転送など) により，高品質サービスとして柔軟な資源配分による QoS，Bandwidth-on-Demand (BoD) などが求められている．これらの先進的インターネットにおいては，資源を効率的に利用することが求められる．そこで，任意の通信拠点間のネットワークを専用線のように利用する事ができるレイヤ 1 帯域オンデマンドサービス (Layer-1 BoD) が注目されている．Layer-1 BoD では，ユーザに任意の拠点間におけるネットワーク帯域を任意の時間帯で提供する．ユーザは，使用したい拠点間，帯域，日時などをリクエストとして指定することにより，その時間帯でそのユーザ専用のネットワーク帯域を専用線と同じように利用できる[1)2)]．このようなサービスにおいては，ユーザの満足度や公平性を考慮し，ネットワーク資源を効率的に運用することが求められる．一方で，大規模ネットワークのパケット通信を対象にネットワーク資源を効率的に運用する研究が盛んに行われている[3)4)]．しかしながら，Layer-1 BoD においては従来のパケットトラフィックにおける資源割当ての最適化ではなく，各ユーザの帯域確保要求であるリクエストに対して，どのリンクを割り当てるのか，また全リクエストを受理できない状況において，どのユーザのリクエストを受理，または却下するのかを公平性，資源利用効率の両面において考慮する必要がある．本研究は Layer-1 BoD を対象とし，そのネットワークリソースを複数のユーザに効率的に分配する方法を検討する．本章では，本論文の概略について述べた．2 章では Layer-1 BoD の内容とその必要性について述べる．3 章では既存のスケジューリングシステムの概要とそのアルゴリズムについて述べる．4 章で具体的な資源分配アルゴリズムを提案し，5 章において提案アルゴリズムを最適化の観点から実現する方法を提案する．また 6 章で提案アルゴリズムの検証を行い，7 章で結論を述べる．

2　Layer-1 BoD

本章では，はじめに Layer-1 BoD の概要を説明する．次に，そのサービスがどのように構成されているかを述べ，最後にどのようなプロセスをたどって帯域割り当てが行われるかを述べる．なお，本章においては実際に Layer-1 BoD を提供している学術ネットワークである SINET3 の実装を述べる[5)1)]．

2.1　Layer-1 BoD の概要

Layer-1 BoD は，利用者側から任意の対拠点間での波長接続や帯域指定専用線接続を設定可能とするサービスである．SINET3 においてはネットワーク (リンク) で接続された，全国 75 の大学や研究機関を拠点として Layer-1 BoD を提供する．各拠点間を結ぶリンクの限界帯域は最小で 1Gbps，最大で 40Gbps である．SINET3 の拠点とリンクのイメージを Fig. 2.1 に示す．

ユーザは，Fig. 2.1 で示される拠点の中から利用したい 2 つの拠点を指定した後，ネットワークの利用時間や利用帯域を指定することにより，ネットワークを利用することが可能となる．これにより，ユーザは超大容量のデータ転送や超高品質な通信を必要な日時に行うことができる．また接続拠点だけでなく，そのネットワーク経路 (ルーティング) を指定することも可能であり，これにより通信の遅延時間を最小とすることもできる．

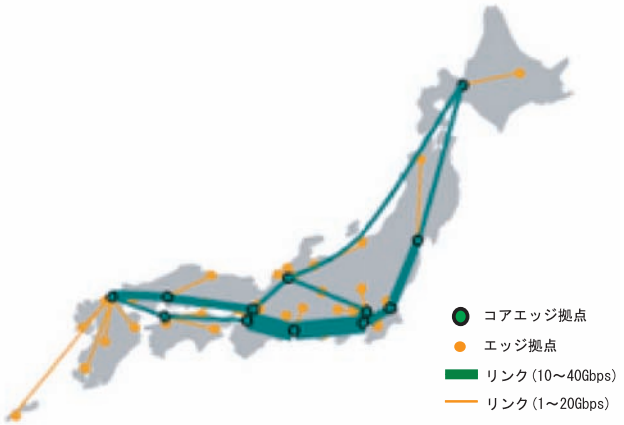

Fig. 2.1　SINET3 のネットワーク

2.2　Layer-1 BoD の構成とサービスの処理の流れ

Layer-1 BoD の構成を Fig. 2.2 に示す．Layer-1 BoD は，主にユーザ，ユーザ装置，SINET3 ネットワーク，オンデマンドサーバで構成される．

ユーザは，Layer-1 BoD の利用者であり，ユーザ装置を介して Web ベースのインタフェースにより帯域確保の要求 (リクエスト) をサブミットすることができる．SINET3 ネットワークは，Layer-1 BoD における通信資源であり，ユーザはこのネットワークを利用することによりデータ通信を行う．オンデマンドサーバは，Layer-1 BoD のシステム管理を行う．オンデマンドサーバがユーザから受け

付けたリクエストを元に，どのように使用するリンクとパスを決定するかの処理の流れを以下に示す．

(1)　ユーザからの希望パスのリクエストの受付

Web ベースのインタフェースを介して，希望パスや希望する利用開始日時と終了日時，使用帯域などのリクエストを受けつける．

(2)　採用するリクエストの決定

各リンクの時間毎の使用可能な帯域を計算し，リソース情報を元に使用するリンクを決定する．リンクの限界帯域が理由で，全てのリクエストを確立させることができない時は，どのリクエストを優先するかの抽選を行う．抽選に関するアルゴリズムは後ほど説明する．

(3)　各リンクにおけるパスの確立

抽選により決定したリクエストが希望する使用時間に，そのリクエストを満足させる接続対地間における各リンクのパスを確立する．

(4)　リソースの管理

リンクの利用状況を元にリソース情報を更新する．

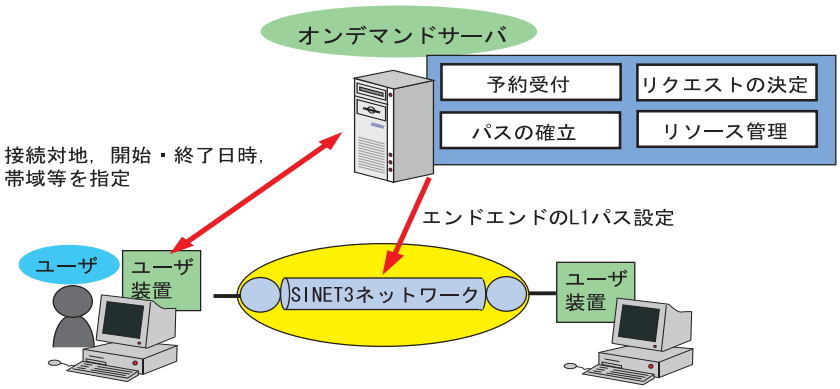

Fig. 2.2　SINET3 における Layer-1 BoD の構成

2.3　Layer-1 BoD におけるユーザへの帯域割り当てまでのプロセス

希望パスのリクエスト予約からパス確立までのプロセスを Fig. 2.3 に示す．

Fig. 2.3 の各プロセスについての説明を以下で述べる．

予約開始

ユーザからの予約受付は，Web ベースのインタフェースを介してオンデマンドサーバで行う．

発着ノード・日時設定

ユーザは，パスを確立する発着拠点と，そのパスを使用してネットワーク通信を行いたい開始時間および終了時間をリクエストとして設定し，オンデマンドサーバに送信する．

利用可能帯域の提示

オンデマンドサーバは，ユーザが希望する接続対地と，ネットワーク通信を行いたい日時から，使用可能帯域を提示する．

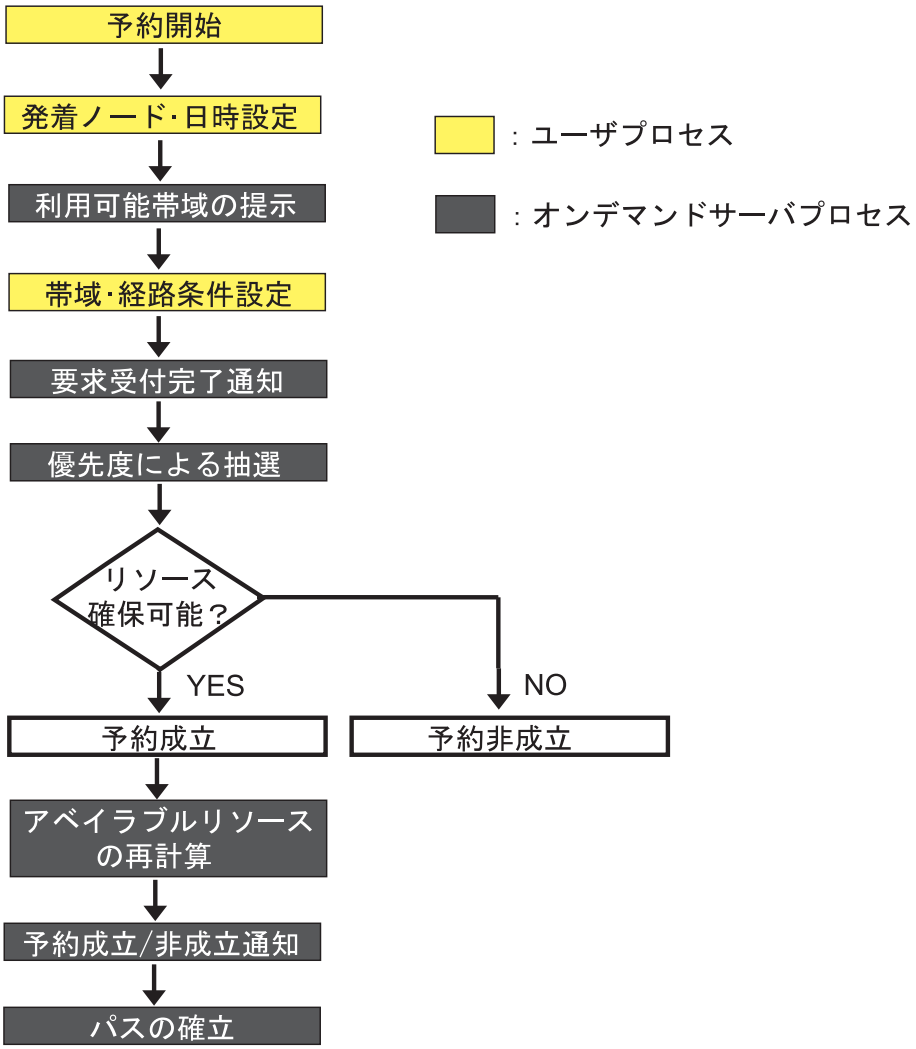

Fig. 2.3　ユーザへの帯域割り当てのプロセス

帯域・経路条件設定

ユーザは，提示された使用可能帯域の範囲内で，使用帯域 ($150Mbps \times n$) と，経路条件を設定する．経路条件によってネットワーク通信の遅延時間を最小にするネットワーク経路を希望するか，遅延時間が最小ではない経路で通信を行うことを許可するかを決定する．

要求受付完了通知

オンデマンドサーバは，上記の一連のプロセスを終了すると，ユーザの要求受付を完了し，その旨をユーザに通知する．

抽選

オンデマンドサーバは，サブミットされた複数のリクエストをスケジューリング契機まで保持する．スケジューリング契機となると，保持しているリクエストに関して，要求している時間，経路，帯域でパスの確保が可能かどうかを判断する．もし，パス確立が不可能なリクエストが出現した場合は，ある特定のアルゴリズムによって，保持したリクエスト群の中から受理するリクエストを決定し，その他のリクエストは却下する．

予約成立または非成立

オンデマンドサーバは，あるリクエストについて予約の成立・非成立の旨をユーザに通知する．

アベイラブルリソースの再計算

オンデマンドサーバは，ある予約が受理されると各リンクの使用予定状況を再計算する．

パスの確立

オンデマンドサーバは，受理した各リクエストのパス確立の希望使用時間となれば，そのリクエストが希望する接続対地間のリンクのパスを希望する帯域分だけ確立する．

” 優先度による抽選” のプロセスにおいて，パスの確立が不可能な場合について述べる．オンデマンドサーバは，サブミットされており，かつ予約の可否待ちのリクエストについて，それぞれ希望する対拠点間のリンクで希望する帯域のパスを確立することが可能かを確認する．その際，あるリクエストの希望するリンク間で，既に利用が確定しているリクエストと，新たに確定したリクエストの影響により，希望する使用開始日時にパスを確立することができないという状況が起こりうる．

このような状況が発生した場合，複数のリクエストの中からどのリクエストを優先するのかを決定しスケジューリングを行うことが重要となる．しかし，既存の Layer-1 BoD においては，非常にシンプルなスケジューリングアルゴリズムが採用されており，全てのユーザの要望を満足できない状況が生じる場合があると考えられる．そこで，本論文では Layer-1 BoD における効率的なスケジューリングアルゴリズムを提案する．

3　代表的なスケジューリングシステム

Layer-1 BoD においてリクエストのスケジューリングを行う際，受理または却下するリクエストは，後にスケジューリングされるリクエストの受理，却下，受理した場合のネットワーク使用経路に大きく影響を与える．そのため，どのリクエストを優先して受理するべきかを決定するアルゴリズムが重要となる．例を挙げると公平性という観点から見て，ある特定のユーザのリクエストに受理が偏って

しまい，それが原因となり，他のユーザのリクエストが却下され続けるという状況が発生するべきではない．このような理由から各リクエストをスケジューリングし，どのリクエストを優先するかを決定することは極めて重要となる．

そこで，Layer-1 BoD におけるリクエストスケジューリングを検討するにあたり，関連研究としてオペレーティングシステム (OS) と分散コンピューティングにおける既存のジョブスケジューリングシステムについてその概要とアルゴリズムについて述べる．

3.1 OS におけるスケジューリング

3.1.1 OS におけるスケジューリングの定義

OS におけるスケジューリングとは，優先度無し，または優先度有りのプロセスを制御する手法のことである．また，実行可能なプロセス全体を監視し，どのプロセスに実行権を与えるかを判断するソフトウェアをスケジューラと呼ぶ．

スケジューラは OS が提供するソフトウェアであり，いつ，どのプロセスに実行権を与えるかを判断し，プロセス切り替えを行うソフトウェア (ディスパッチャ) に実行プロセスの切り替え (プリエンプト) 要求を出す．

プロセスには，実行状態，実行可能状態，待機状態の 3 つの状態がある．実行状態のプロセスは CPU によって実行されているプロセスのことである．実行可能状態のプロセスは，実行権を与えられのを待っているプロセスである．また，待機状態のプロセスは，入出力要求等のシグナルを待っているプロセスである．OS におけるスケジューリングはプロセスにこれらの 3 つの状態をどのように割り当てるかが重要となる．

3.1.2 OS によるスケジューリングの処理

OS によるスケジューリングでは，基本的にはプロセスの処理は FIFO(First In, First Out) の原則に従って処理が行われる．まずプロセスが新たに立ち上がると，そのプロセスはキューに挿入される．そしてキューに挿入されたプロセスから順に処理が行われることになるが，そこに新たにプロセスが立ち上がった場合は，実行中のプロセスとの優先度の比較が行われる．このとき，新規プロセスの優先度が，実行中のプロセスよりも低い場合は実行プロセスはそのまま処理されることになるが，高い場合はプリエンプト要求が出される．この場合，実行中のプロセスは実行状態から，実行可能状態または待機状態へと変更される．最後に，プロセスはある実行割り当て時間 (タイムスライス) が与えられ，実行される．

3.1.3 Linux カーネルによるスケジューリング

ここでは，OS によるスケジューリングの例として，UNIX 系 OS である Linux カーネルを例に挙げる[1]．Linux カーネルで使用されているスケジューリングは主に以下の項目によって，実装されている．

[1]高橋浩和（VA Linux Systems Japan）著, UNIX USER2004 年 6 月号「Linux カーネル 2.6 解読室」より転載, URL 参照 http://www.itmedia.co.jp/enterprise/articles/0406/08/news002.html

- RUN キュー

  Linux カーネルでのキューは RUN キューと呼ばれ，2 種類の RUN キューが存在する．一つは active キューと呼ばれ，実行可能状態にあり，タイムスライスを持っているプロセスが挿入されるキューである．もう一方は expired キューと呼ばれ，実行可能状態ではあるがタイムスライスを使い果たしたプロセスが挿入される．

- 優先度

  各プロセスは実行優先度として固定優先度と変動優先度を持つ．固定優先度は「nice コマンド」によってユーザが決定することができる優先度である．変動優先度は時間経過と共に変動する優先度である．スケジューラは固定優先度と変動優先度の和が最も大きいプロセスに実行権を与える．より多く CPU を利用したプロセスほど低い変動優先度を持つようになっており，シェルのような対話型プロセスはあまり CPU を利用しないため，相対的に高い変動優先度が与えられる．

- スケジューリング契機

  スケジューリングが行われる契機は要求が提出されれば即座に行われるものと，要求が提出されてもある時期まで待機させられるものの 2 つがある．前者は，ある実行中のプロセスがシグナル待ちの待機状態になったときである．この場合，スケジューラは即座に実行可能状態のプロセスから最適なものに実行権を与える．後者は，実行中のプロセスがタイムスライスを使い果たした場合，もしくは実行中のプロセスよりも優先度の高いプロセスがキューに挿入された場合である．この場合，スケジューラはシステムコール処理や割り込み処理などの Linux カーネルが行わなければならない処理が完了するまで起動しない．

3.1.4 Linux カーネルのスケジューリングアルゴリズム

Linux カーネルにおけるスケジューリングアルゴリズムを述べる．

実行状態のプロセスが待機状態に変わると，スケジューラは，そのプロセスを active キューから外す処理を行う．ただし，シグナル待ちの待機状態で，シグナルを受信したときは active キューに戻す．

次に，スケジューラは active キューを先頭から検索し，次に実行権を与えるプロセスとして，最も優先度の高いプロセスを選択する．もし，active キューが空ならば，active キューと expired キューを交換し，再度実行すべきプロセスを検索する．

もし，expired キューにもプロセスが存在しない場合は，アイドル状態であるアイドルプロセスを実行する．

次に，動作すべきプロセスが決定すればプロセスディスパッチャを呼び出し，次に実行されるプロセスと実行されていたプロセスを切り替える．

スケジューリング処理の間に発生した新しいスケジューリング要求に対しては，スケジューリング処理が完全に終了した時点で，再度スケジューリング処理を行うことによって対応する．

プロセスの優先度を決めるためのパラメータである変動優先度は，スケジューリングの際に計算される．変動優先度は，プロセスの実行時間と待機時間を考慮して求められる．実行時間が長いプロセスほど変動優先度は低く設定され，待機時間が短いプロセスほど変動優先度は高く設定される．

もし，次に実行されるプロセスが，シグナル待ちの状態から起床したばかりであれば，実行優先度の再計算を行う．なぜなら，シグナル待ちの待機状態から起床したプロセスは応答性確保のために一時的に高めの変動優先度が与えられるからである．そのため，本来の優先度に戻す処理が必要となる．

3.2 分散コンピューティングにおけるスケジューリング

分散コンピューティングでは，遊休資源を効率的に使用することが全体の利用効率の向上につながるため，スケジューリングは重要となる．そこで，分散コンピューティングにおけるスケジューラとして代表的な Condor について述べる．本節では Condor の概要，構成及びアーキテクチャを述べた後，Condor システムにおけるジョブスケジューリングアルゴリズムについて述べる．

3.2.1 Condor の概要

Condor は米国ウィスコンシン大学マディソン校における Condor Research Project によって開発，配布が行われているフリーのジョブスケジューラである[6)]．High Throughput Computing を念頭においており，また遊休ノードを効率的に利用することも目的としたスケジューリングシステムである．

Condor システムは，様々なリソース (パーソナルコンピュータや PC クラスタ) が混在している分散コンピューティングを環境を想定している．分散コンピューティングの環境では，多様なリソースに加え，多様なニーズを持つユーザが存在している．そのため各ユーザは使用可能なリソースの状態の把握が困難である．Condor は，それぞれのユーザのジョブの要件及び，リソースの状態からジョブの実行に最も適したリソースを決定し，そのリソースにおいてジョブを実行する．

また，Condor ではある特定のユーザにリソースを占有されることを防ぐため，独自の優先度決定アルゴリズムを採用している[2]．

3.2.2 Condor の構成及びアーキテクチャ

Condor は主にクライアントマシン，セントラルマネージャ，Condor プールによって構成される．Condor の構成を Fig. 3.1 に示す．クライアントであるユーザは処理したいジョブを，Condor システムに投入 (サブミット) する．Condor プールは，Condor システムにおける計算ノード群である．ユーザがサブミットしたジョブは，Condor プールのいずれかのリソースによって実行される．セントラルマネージャは Condor プール内のノード，ユーザの状態を逐次把握し，各ユーザのジョブのスケジューリングを行う．

3.2.3 Condor システムにおけるジョブスケジューリングアルゴリズム

Condor システムにおけるジョブスケジューリングアルゴリズムについて述べる．ユーザはサブミットマシンから，ジョブのサブミットを行う．サブミットされたジョブの情報はセントラルマネージャに送信される．Condor システムにおいてスケジューリングは一定間隔で行われており，スケジューリング契機となるとセントラルマネージャは優先度順に処理待ちを行っているジョブを並び替える．次に，優先度の高いリクエストから最適なリソースを選出し，そのリソースでジョブの実行を行う．

[2]CondorVersion 7.5.0 Manual, Copyright 1990-2009 Condor Team, Computer Sciences Department, University of Wisconsin-Madison, URL 参照 http://www.cs.wisc.edu/condor/manual/v7.5/

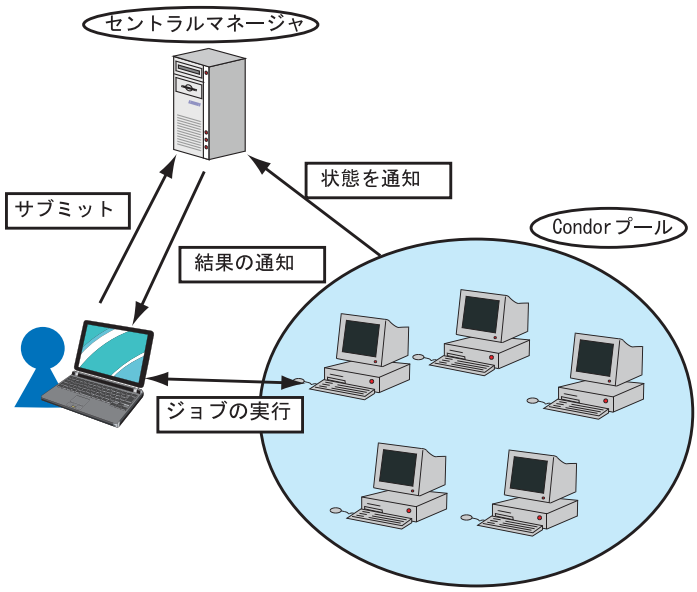

Fig. 3.1　Condor の構成

セントラルマネージャが，実行待ちのジョブを並び替える際，それぞれのジョブの優先度を計算する．優先度は一般に，以下の式 3.1 で表される Real User Priority(RUP) により与えられる．

$$\pi(u,t) = (0.5)^{\delta t/h} \times \pi(u, t-\delta t) + (1-(0.5)^{\delta t/h}) \times \rho(u,t) \tag{3.1}$$

上式で，$\pi(u,t)$ はユーザ u の時刻 t における RUP の値を表している．δt はスケジューリング間隔を表し，h は半減期を表す．また $\rho(u,t)$ はユーザ u が時刻 t に使用しているリソース数を表す．RUP は，スケジューリング時に，前回のスケジューリングにおける RUP 値をもとに新たに計算される．

Condor では，RUP の初期値は 0.5 であり，最小の値も 0.5 と設定されている．それぞれのユーザの RUP 値は 0.5 からはじまり，しだいにユーザが使用しているリソース数に収束する．そして全てのリソースを解放すると，半減期の設定により RUP の値はしだいに 0.5 に近づく．例えば，半減期を 86400 秒 (1 日) として，ある RUP=0.5 のユーザが，常に 10 のリソースを使用しているならば，そのユーザの RUP 値はしだいに 10 に近づいていく．RUP 値が 10 となると，リソースを解放しない限り，RUP 値は変化しない．そのユーザがリソースを全て解放すると半減期 h 後には RUP の値が半分の 5 となり，しだいに 0.5 に収束する．

セントラルマネージャは，キューに格納されているジョブを，RUP の昇順に並び替える．そして，RUP の低いものから優先的にリソースを与えられ，RUP が同値のときは，早くサブミットされたジョブ程優先的にリソースを割り当てる．

また，特殊なユーザにおける優先度の計算方法として Effective User Priority(EUP) という優先度が設定されている．EUP は以下の式 3.2 により与えられる．

$$\pi'(u,t) = \pi(u,t) \times f(u,t) \tag{3.2}$$

$\pi'(u,t)$ はユーザ u の時刻 t における EUP の値であり，$\pi(u,t)$ はユーザ u の時刻 t における RUP

の値である．EUP は RUP とユーザ u の時刻 t における特殊要素 $f(u,t)$ との積により与えられる．特殊要素 $f(u,t)$ は Condor の管理者が設定することができる．例えば，ジョブの優先度を低くすることにより，他のユーザが使用していない遊休状態の計算機で処理を行いたいユーザがいるとすると，そのユーザは $RUP \geq 1$ の大きな値 との積で優先度が計算される．

4 Layer-1 BoD に適したスケジューリングアルゴリズム

Layer-1 BoD は不特定多数のユーザを想定したシステムである．そのため，複数のユーザで帯域を確保したい拠点間の使用リンク，使用日時が重複すると，全ユーザの希望した通りにネットワークを確保することが不可能な状況が生じる．そのような状況が発生した場合，どのユーザのリクエストを受理し，またどのユーザのリクエストを却下するか，受理するリクエストに対してどのようにネットワークリンクを割り当てるかを決定するリクエストスケジューリングが重要となる．本章ではこのリクエストスケジューリングを行う上で指針となる要件を定義し，その要件を満たすためのスケジューリングアルゴリズムを提案する．

4.1 スケジューリングアルゴリズムの要件

本論文においては，サービスの質，公平性の観点から下記のことを考慮した上でリンクを効率的に利用し，各ユーザの満足度を高く保つことを目標にしたスケジューリングアルゴリズムを検討する．

- 全体のスループットを高く保つ

  常にリクエストの処理数や， 各リンクの使用率を高く保つことによって， サービスのスループットを高く保つことが望まれる．そのためには，どのユーザのリクエストを受理し，それぞれのリクエストに対し，どのようにネットワークリソースを提供するかということが非常に重要となる．

- 全ユーザが平等にネットワークリソースを利用する

  Layer-1 BoD では複数のユーザがネットワークリソースであるリンクの帯域を共有して利用する． そのため， ある特定のユーザによって常にリソースが占有され， 他のユーザがリンクを利用することができないという状況は避けるべきである．また，サブミットされた時期が早いリクエストほど優先されるべきである．このことは Layer-1 BoD において，その処理プロセスの特性から既に考慮されていると言える．なぜならば，オンデマンドサーバはある一定の間隔でスケジューリングを行うからである．スケジューリングの結果，リクエストが受理されたなら，その時点で確実な希望時間にパスが確立されることになる．よってある時点でスケジューリングされるリクエストは，次回以降にスケジューリングされるリクエストよりも必ず優先されることになる．

4.2 提案スケジューリングアルゴリズム

本論文では，2.3 節の SINET3 におけるユーザへの帯域割り当てまでのプロセスを考慮し，4.1 節で述べた要件を満たすためのアルゴリズムとして Fig. 4.1 に示すアルゴリズムを提案する．また，本

アルゴリズムはLayer-1 BoDにおけるオンデマンドサーバに適用することを想定している．そのため，ユーザ情報及び，ユーザのリクエスト管理，ネットワークの各リンク，拠点の状態は常に提案アルゴリズムに対してオープンに提示することが可能であると仮定する．

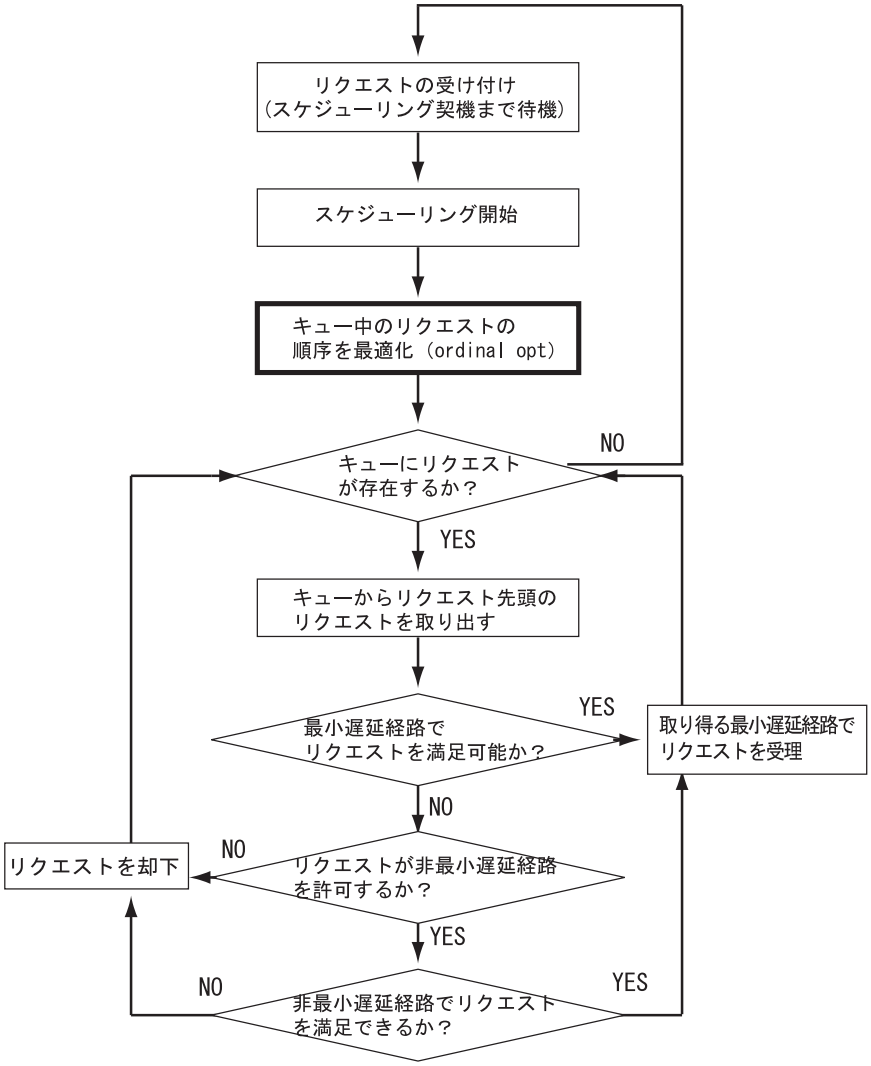

Fig. 4.1　提案スケジューリングアルゴリズム

次に，それぞれのプロセスについて説明する．

リクエストの受け付け

オンデマンドサーバはスケジューリング契機までリクエストの受付を行う．その際，あるユーザがオンデマンドサーバに対してリクエストをサブミットした場合，オンデマンドサーバはそのリクエストを自身が持つキューに格納する．同様にしてスケジューリングを行う時間となるまでオンデマンドサーバはリクエストの受付を行う．

スケジューリング開始

スケジューリングを開始するために必要なユーザ情報やリンク状況，リクエスト等をインプットする．それが終了すればスケジューリングを開始する．

キュー中のリクエストの順序を最適化 (ordinal opt)

オンデマンドサーバは自身が持つキューに格納されているリクエスト郡を処理する順序を決定する．本論文ではこのプロセスを”ordinal opt” とする．

キューから先頭のリクエストを取り出す

次にリクエストをキューの先頭から一つ取り出し，それを処理するリクエストとして受理判定を行う．もしその時点で処理するリクエストが一つも無いならば，リクエストの受け付けのプロセスに戻る．

取り得る最小遅延経路でリクエストを受理

もし処理しているリクエストの内容で利用可能な経路が存在するならば，その経路のうち，最も遅延の少ない経路をリクエストに割り当てる．

処理しているリクエストに経路を割当て，受理すれば，そのリクエストは処理済みとなる．

一方で，Layer-1 BoD においてはリアルタイム性の重要なアプリケーションをサポートするために，リクエストの開始拠点から目的拠点までの時間を最小遅延となる経路に固定して割り当てるオプションを提供している．このオプションを利用すると開始拠点と目的拠点間で遅延の少ない高品質なデータ通信が行える．しかし，最小遅延経路におけるリンクで一つでも利用不可能なものがあればそのリクエストはリジェクトされる．

リクエストを却下

もし処理しているリクエストの内容で利用可能な経路が存在しないならば，そのリクエストをリジェクトする．そして，そのリクエストを処理済みとする．

4.3 提案最適化手法への適用

本論文におけるスケジューリングアルゴリズムは，Fig. 4.1 における”ordinal opt” のプロセスにおいて処理するリクエストの順序を決定し，決定した処理順序によって各リクエストを受理するか否かが決定される．また，リクエストが両端拠点まで，どのリンクを利用するかを決定するルーティングルールとして，リクエストを処理する時点で取り得る最小遅延経路を選択している．このため，”ordinal opt” のプロセスにおいて決定するリクエストの処理順序は非常に大きくサービスの性質に影響を与える．

そのため”ordinal opt” のプロセスにおいてオンデマンドサーバが処理するリクエストの順序を最適化し，4.1 節において述べた要件をより高い精度で満足することを目指す．

まずはじめに，4.1 節における「全体のスループットを高く保つ」という要件について考える．ここでスループットを各ユーザのアクセプトされたリクエストにおけるネットワーク利用の大きさと定義し，それを可能な限り大きくすることを目標とする．ユーザのネットワーク利用度を以下の式 4.1 で定義する．

$$\rho_i = D_i \times T_i \tag{4.1}$$

式 4.1 において，ユーザ i が要求する帯域を D_i，使用時間 (使用終了時間 - 使用開始時間) を T_i とする．ネットワーク利用度 ρ が大きくなるほど，各ユーザが利用するネットワークの帯域及び時間が大きくなることを意味する．ネットワークを効率的に利用するという観点においては，各ユーザのネットワーク利用度の和を最大となるスケジューリングが求められる．しかし，各ユーザのネットワーク利用度を最大化するスケジューリングでは，ネットワークを利用できるユーザに偏りが生じ，平等性を保つという目的に反してしまう状況が発生する可能性がある．これは 4.1 節における「全ユーザが平等にネットワークリソースを利用する」という要件に反してしまう．そこで，各ユーザの公平性を維持するために，各ユーザに重みを設定する．平等性の実現方法においては，過去の研究により提案，検証を行い，有効性を確認している[7]．本論文では，過去の研究を基に，ユーザの過去のネットワーク利用度を考慮に入れた重み W_i を導入する．重みを以下の式 4.2 で定義する．

$$W_i(t) = 0.5(dt/h) \times W_i(t - dt) + 1 - 0.5(dt/h) \times \rho_i \tag{4.2}$$

W_i の導入には式 3.1 にて示した Condor の優先度決定アルゴリズムを参考にした[6]．Condor は各ユーザに対し平等にリソースを分配するため，過去のリソース利用を考慮した優先度決定メカニズムを用いている[8]．このメカニズムは，Layer-1 BoD のユーザにリソースを平等に分配する上で非常に有用であると考えられる．$W_i(t)$ は時刻 t におけるユーザ i の重みであり，この値は，前回のスケジューリング時刻におけるユーザ i の重み $W_i(t - dt)$ と，時刻 t において得たネットワーク利用度 ρ_i に大きく依存する値である．この値はユーザがネットワークを利用した際に上昇し，時刻を経るごとに減少する．またこの減少する割合を半減期 h により決定することが可能である．

式 4.1 と式から，$Fitness$ を以下の式 4.3 により決定する．

$$Fitness = \sum_{i=1}^{n} W_i^{-1}(t) \times \rho_i \tag{4.3}$$

$Fitness$ は，各ユーザのネットワーク利用度とその重みの積で得た値の総和によって決定し，本論文では，この $Fitness$ の値を高くするリクエストスケジューリングを目指す．これにより，各ユーザがネットワークを効率的かつ平等に利用することが可能であると予想される．

5 Layer-1 BoD に適用する最適化アルゴリズム

本章では，4 章で述べたスケジューリングアルゴリズムにおける，”ordinal opt” プロセスの最適化手法を検討する．本論文では，”ordinal opt” プロセスにおいて First In, First Out，Local Search アルゴリズム，貪欲法，遺伝的アルゴリズム，分散遺伝的アルゴリズムの適用を考慮し，その実装について述べる．

5.1 First In, First Out

5.1.1 First In, First Out の概要

First In, First Out(FIFO) は待ち行列理論により定義されているキューの動作手法であり，QoS(Quality of Service) や CPU 内部に命令や計算結果を格納するときなどに代表的に利用される手法である．FIFO

では最初に入ってきたデータやプロセスを最初に処理し，次に入ってきたものは最初の処理が終わるまで待機させる．そのため一番早く格納されたデータやプロセスが最も優先される．

5.1.2 First In, First Out での実装

Layer-1 BoD においては，CPU で採用されている FIFO と同様に，早くサブミットしたユーザのリクエストから順次リンクの割当てを行う．このため，早くサブミットしたユーザほど，他のユーザのリクエストとの競合が生じる可能性が低くなり，アクセプトされやすくなる．本手法は SINET3 の Layer-1 BoD において採用されている．

5.2 貪欲法

5.2.1 貪欲法の概要

貪欲法 (Greedy Algorithm:GR) とは問題の要素を複数の部分問題に分割し，それぞれを独立に評価を行い，評価値の高い順に取り込むことで解を得る手法である[9)]．GR では一度選択した要素を再考することはない．そのため，非常に高速に解を得ることが可能であるが，適合度の低い局所解に陥ってしまう可能性が高い．

5.2.2 貪欲法での実装

Layer-1 BoD における GR の実装として，部分問題をサブミットされた各リクエストの適合度値を検討する．適合度値はそれぞれのリクエストがアクセプトされた際の適合度の上昇値である．GR においては適合度値が高いリクエストから順に処理していく．これにより適合度値の高いリクエスト程アクセプトされやすくなると考えられる．

各リクエストの適合度値は 4.3 節で述べた適合度関数である式 4.3 から以下の式のように定義する．

$$RequestValue = W_i^{-1}(t) \times \rho_i \tag{5.1}$$

式 5.1 は，時刻 t におけるユーザ i のリクエストの適合度値である．W_i 及び ρ は 4.3 章で述べたパラメータであるため説明を省略する．

5.3 Local Search アルゴリズム

5.3.1 Local Search アルゴリズムの概要

Local Search(LS) は進化計算の中でも最も単純なアルゴリズムの一つである[10)11)]．LS はある解からその解の近傍（次解候補）のうち一つをある条件で選び近傍解とする最適化手法である．解の近傍の設定や，その近傍から解を選定する方法は多岐にわたる．

また，進化計算においては，受理している解が目的にどの程度有効かを示す指標として適合度関数が与えられる．一般的に，適合度関数は適合度の高い解ほど良い性能を持つように設定される．そのため，適合度関数の設計は非常に重要となる．

5.3.2 Local Search アルゴリズムの基本動作

LS の基本動作のフローチャートを Fig. 5.1 に示す．

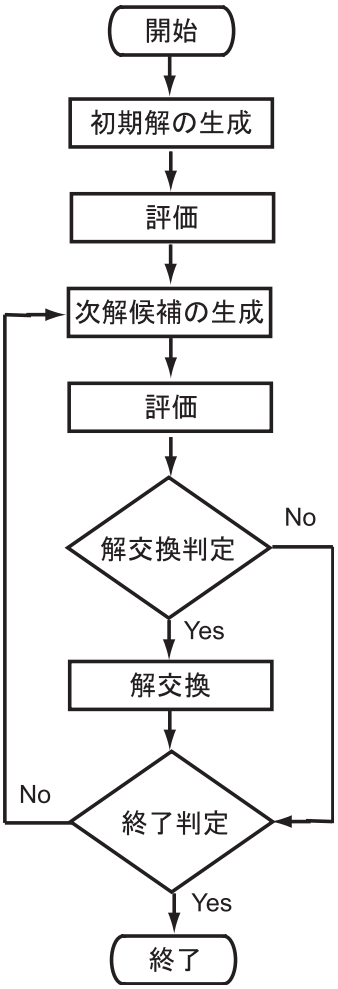

Fig. 5.1　LS の基本動作

LS のアルゴリズムの各プロセスの内容は以下のとおりである．

初期解の生成 (Initialization)

　初期解を生成し，解として受理する．

評価 (Evaluation)

　解の適合度を計算する．適合度の高い解ほど，良好な解といえる．

次解候補の生成 (Create Solution)

　現在受理している解を元に次解候補を生成する．

解交換判定 (Judge)

　現在受理している解の適合度と，次解候補の適合度関数による解の適合度を比較する．

解交換 (Exchange)

　解交換判定にて次解候補の適合度が現在受理している解の適合度を上回っていれば，解交換を行い，次解候補を解として受理する．

終了判定 (Terminal criterion)

あらかじめ定められた終了条件に基づいて，LS の処理を終了する．この時点で受理している解を最適解とする．

5.3.3 Local Search アルゴリズムでの実装

LS の Layer-1 BoD の実装として，”ordinal opt” によって生成されたあるリクエスト処理順序を一つの” 解” と表現する．

初期解はランダムにリクエスト処理順序を決定することにより生成する．

” 評価” に用いる適合度関数は 4.3 節で述べた式 4.3 における $Fitness$ とする．” 解交換判定” の際は，$Fitness$ の値が高いリクエスト処理順序を解として判定する．

” 次解候補” は離散最適化問題において代表的な 2-Change[9] を用いて生成する．

5.4 遺伝的アルゴリズム

5.4.1 遺伝的アルゴリズムの概要

遺伝的アルゴリズム (Genetic Algorithm:GA) は生物の進化の過程を工学的に摸倣した確率的な最適化手法である[12)13)]．GA ではある世代 (Generation) を形成しているいくつかの個体 (Individual) の集合を母集団 (Population) と呼ぶ．また GA では自然界の進化過程と同様に，環境への適合度 (Fitness) の高い個体が高い確率で選択 (Selection) される．そして，その個体に対して，交叉 (Crossover) や突然変異 (Mutation) がある確率で発生することにより次世代の母集団が形成され，最後に得られた母集団の中で，最も適合度が高い個体を最適解として採用する．

個体は，染色体 (Chromosome) によって特徴付けられており，染色体は複数の遺伝子 (Gene) で構成されている．通常 GA では 1 染色体で 1 個体を表現する．染色体上で各遺伝子の置かれている位置を遺伝子座 (Locus) といい，実数最適化においては対立遺伝子 (Allele) として，Fig. 5.2 のように，2 進数のビット {0,1} を用いることが多い．一方で離散最適化においては，遺伝子型に実数値を適用することが多い．適合度が大きいものほど子孫を残しやすく，小さいものほど死滅しやすいようになっている．このことにより，次世代の各個体の適合度は前世代よりも良いことが期待される．

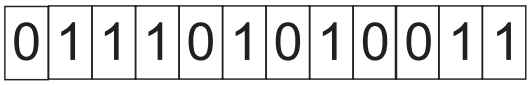

Fig. 5.2　遺伝子型のビット表現

5.4.2 遺伝的アルゴリズムの基本動作

GA の基本動作のフローチャートを Fig. 5.3 に示す.

GA のアルゴリズムの各プロセスの内容は以下のとおりである．

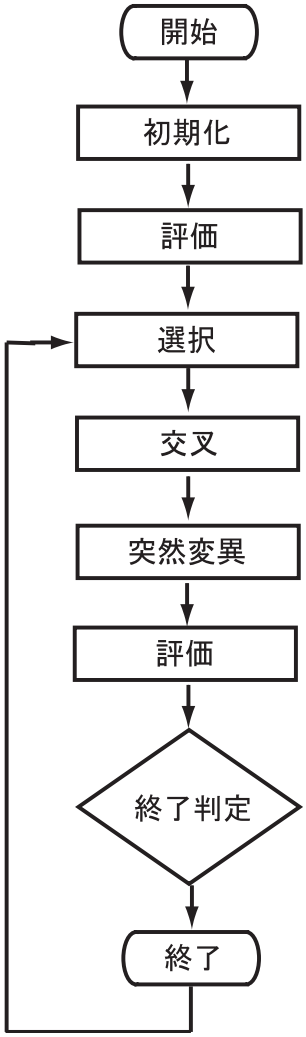

Fig. 5.3　GA のフローチャート

初期化 (Initialization)

あらかじめ設定された数だけの個体を生成する．生成される個体の数を母集団サイズ (Population size)，または個体数と呼ぶ．

評価 (Evaluation)

各個体に適合度を設定する．適合度は，一般的に非負であり，適合度の高い個体ほど，良好な個体といえる．

選択 (Selection)

生物の適者生存を模倣したものである．適合度に依存した一定の規則に従い，次世代に残す個体を選択する．選択により，適合度の低い幾つかの個体は淘汰され，その個体数だけ適合度の高い個体が増殖する．代表的な選択の方法として，ルーレット選択 (Roulette selection)，トーナメント選択 (Tournament selection) などがある．ルーレット選択は個体の適合度に比例した割合で選択する手法である．トーナメント選択は，集団の中から重複を許して，あらかじめ定め

られた数の個体をランダムに選び出し，その中で最も適合度の高い個体を選択する操作を，設定された個体数が選ばれるまで繰り返す手法である．この 2 つの選択手法は，適合度の低い個体も選ばれる可能性もある．またこの他に，適合度の高い個体を無条件に次世代に残すエリート保存戦略 (Elitism) も選択手法の 1 つとして考えられる．

交叉 (Crossover)

生物の有性生殖を模倣したものである．交叉では，親個体の優れた部分形質を子個体に継承することを目的としている．交叉の仕組みは，定められた交叉率 (Crossover rate) に基づき，個体の遺伝子と別の個体の遺伝子を入れ替えることにより，新しい子個体を生成するものである．優れた個体同士が交叉すると，それぞれの個体の優れた部分解を構成する部分が結合し，より優れた個体が生成されることが期待される．

突然変異 (Mutation)

選択，交叉のみでは，初期母集団内の遺伝子に依存するような限られた範囲の子個体しか生じない．そのため，一般にあまり望ましくない解に収束することが多い．従って，突然変異によって選択・交叉だけでは生成できない子個体を生成し，個体群の多様性を維持することが必要である．突然変異では，染色体上のある遺伝子座を，他の対立遺伝子に置き換える．突然変異は突然変異率 (Mutation rate) で定められた確率で起こる．これは自然界における DNA 複写の際のコピーミスに当たる．

終了判定 (Terminal criterion)

あらかじめ定められた終了条件に基づいて，GA の処理を終了する．この時の母集団で適合度の最も高い個体を最適解として採用する．

5.4.3 遺伝的アルゴリズムにおける交叉手法

遺伝的アルゴリズムにおける交叉は，子が親のそれぞれのどのような形質をどの程度受け継ぐかを決める操作である．この操作により次世代に生じる個体の特徴が決定される．このため交叉は評価に大きく影響を与える重要な操作である．

離散最適化，特に順列最適化においては，問題によっては順番のみならず，その位置が重要となる場合があり，様々な交叉手法が提案されている．本項では，離散最適化において代表的な 3 手法を挙げて説明する．

5.4.3.1 部分写像交叉

部分写像交叉（Partially-mapped crossover:PMX）は一方の親からその部分経路をそのまま受け継ぎ，他の親から残りの遺伝子座について，可能な限り親の順番を受け継ぐことを目的としている[14)]．そのため，2 つのランダムに選んだ切断点間の部分経路を選び，以下の例のように交叉を行う．

$$p1 = (123|4567|89)$$

$$p2 = (452|1876|93)$$

まず 2 つの交叉点で挟まれた部分を交叉すると次に示す子が得られる．ここで”*” は今のところ未決定であることを示す.

$$c1 = (***|1876|**)$$

$$c2 = (***|4567|**)$$

このことは次の入れ替えを定義していることとなる.

$$1 \leftrightarrow 4 \quad , 8 \leftrightarrow 5 \quad , 7 \leftrightarrow 6 \quad , 6 \leftrightarrow 7$$

残りの未決定の部分について，既定のものと衝突を起こさないものはそのまま挿入する．最後に残った部分については前記の入れ替えの定義を参照して衝突が生じないように挿入する．この操作により以下の子が得られる.

$$c1 = (423|1876|59)$$

$$c2 = (182|4567|93)$$

この手法は類似性と順序を可能な限り保存している点が特徴である.

5.4.3.2 順序交叉

順序交叉 (Order crossover:OX) は片方の親からは一部をそのままに受け継ぎ，残りの部分については相対的な順番は保持しつつ受け継ぐ方法である[14)]．例として次の 2 つの親からの交叉を考える.

$$p1 = (123|4567|89)$$

$$p2 = (452|1876|93)$$

はじめに 2 切断点間はそのままにコピーする.

$$c1 = (***|4567|**)$$

$$c2 = (***|1876|**)$$

次に第 2 切断点から開始して右回りに順序を保ちながら他の親をコピーする．その際に既定のものと衝突するものがあれば除く．例えば，$c1$ について見てみると，第 2 の親 $p2$ の順序は

$$9 - 3 - 4 - 5 - 2 - 1 - 8 - 7 - 6$$

となるが子 $c1$ と衝突するもの (4，5，6，7) を除くと

$$9 - 3 - 2 - 1 - 8$$

となるので，上記のリストを挿入すると次の子が得られる.

$$c1 = (218|4567|93)$$

$$c2 = (345|1876|92)$$

この手法は，ある一定の連続した遺伝子座を親から受け継ぐ部分ではPMXと同じである．しかし，PMXは遺伝子座の位置を保持しているのに対し，OXは相対的な順番を保持するのが特徴である．

5.4.3.3 循環交叉

循環交叉 (Cyclic crossover:CX) は部分的に親の遺伝子座を継承する交叉手法である[14)]．例として次の2つの親からの交叉を考える．

$$p1 = (123|4567|89)$$

$$p2 = (452|1876|93)$$

はじめに順番に残す遺伝子をランダムに決める．子$c1$について，親$p1$の最初の遺伝子”1” を選択したとすると，次は親$p2$の一番目の遺伝子は”4” であるので，親$p1$の”4” の遺伝子がある遺伝子座を子$c1$に継承する．この結果次の子が得られる．

$$c1 = (1 * *4 * * * 8*)$$

同様の操作を施し，$c1$，$c2$は以下のようになる．

$$c1 = (123376985)$$

$$c2 = (412856739)$$

この手法では，交差点が存在せず，連続した遺伝子座ではなく，各親の部分的な順序を継承する．このため，交差点に依存しない進化を行うことが可能である．

5.4.4 遺伝的アルゴリズムでの実装

GAのLayer-1 BoDの実装として，”ordinal opt” によって生成されたあるリクエスト処理順序を一つの” 個体” として表現する．そしてランダムなリクエスト処理順序を母集団サイズ分生成し母集団とする．

評価に用いる適合度関数はLSと同様に4.3で述べた式4.3における$Fitness$とする．

” 選択” においては効率的に母集団を進化させるために，各個体の適応度に大きく依存するルーレット選択と，適応度の高い個体を残すエリート保存戦略を併用する．

” 交叉” 手法は離散最適化問題を解決するうえで代表的な手法から適用する環境に適した手法を利用する．

” 突然変異” はある一つの個体におけるある2つの遺伝子をランダムに選択し，それらの位置を交換する手法とする．これはある個体のリクエスト処理順序の中からリクエストを2つだけランダムに選びそれらの順番を入れ替えることを意味する．

5.5　分散遺伝的アルゴリズム

分散遺伝的アルゴリズム (Distributed Genetic Algorithm:DGA) は GA と同様に生物の遺伝と進化を模擬した多点探索手法である[15]．DGA では，GA における母集団を複数のサブ母集団に分割し，各サブ母集団 (Sub-Population) ごとに独立に遺伝的操作を行い，一定世代ごとに異なるサブ母集団間で移住と呼ばれる複数個体の交換を行う．結果として，全ての個体が一つの母集団を形成するよりも多様性が大きくなり，より効率的な探索を進めることが可能である．ここで，移住 (Migration) を行う世代間隔を移住間隔 (Migration interval) と呼び，サブ母集団の個体数に対する移住個体の割合を移住率 (Migration rate) と呼ぶ．DGA と移住の概念を Fig. 5.4 に示す．

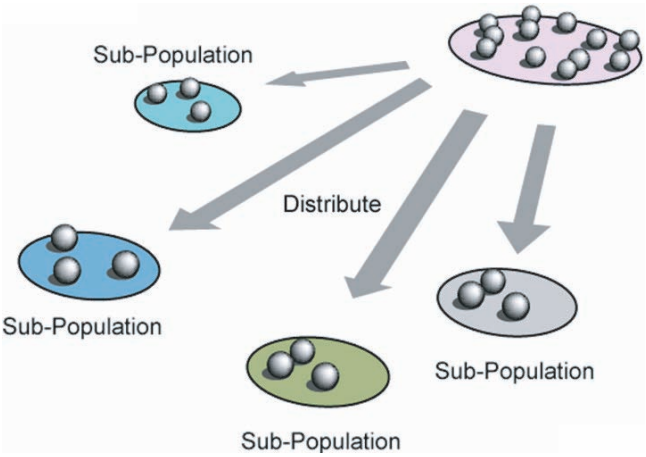

Fig. 5.4　DGA の概念図

DGA の特徴として，母集団の分割による利点がある．母集団を分割することにより各島で個体の異なった遺伝子が進化している場合，移住によって他の島の個体と混ざり合い交叉を行うことで，優れた遺伝子同士が集まった，非常に優れた個体が誕生する可能性がある．

5.5.1　分散遺伝的アルゴリズムの基本動作

DGA の基本動作のフローチャートを Fig. 5.5 に示す．

GA との違いは，” 選択” の後に” 移住間隔” であれば” 移住” 操作を行うことである．移住は，数世代に一度，各サブ母集団内で選ばれた１つまたは複数個の個体（移住個体：Migrant）を別のサブ母集団と交換することで実現される．移住には，移住元と移住先を結んだ線が１つのリングになるように移住先を定める $Randomring$[16] 手法や，各サブ母集団に ID を付与し，その上で各サブ母集団の ID が隣り合っている２つのサブ母集団と移住を行う $bi-DirectionalRing$[17] など様々な手法が存在する．

5.5.2　分散遺伝的アルゴリズムでの実装

GA と同様に”ordinal opt” によって生成されたあるリクエスト処理順序を一つの” 個体” として表現する．そしてランダムなリクエスト処理順序を各サブ母集団サイズ分生成しサブ母集団とする．移住には $Randomring$ を採用する．その他のプロセスにおける実装は GA と同様とする．

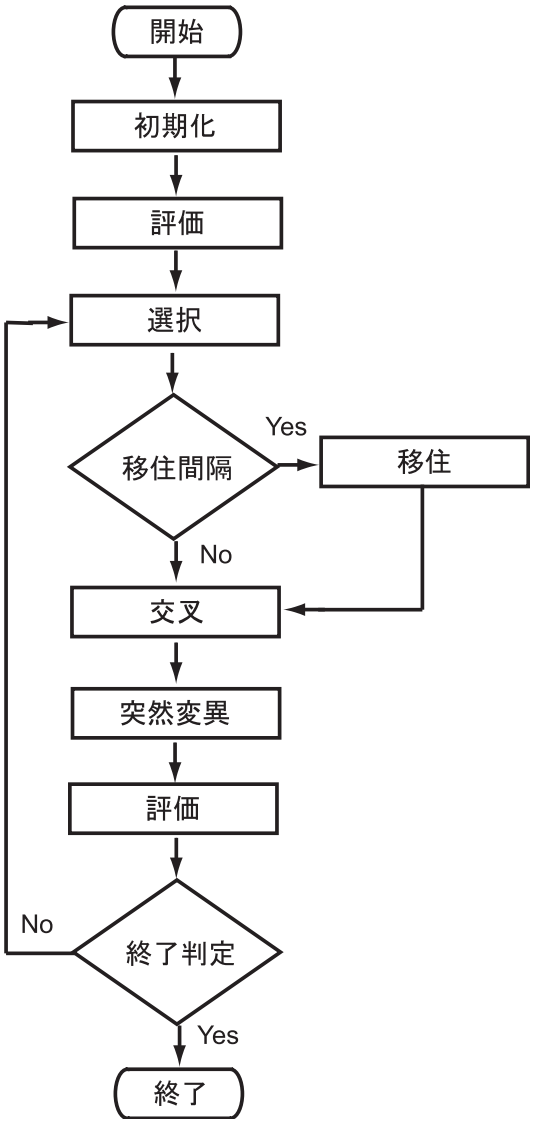

Fig. 5.5　DGA のフローチャート

6　提案アルゴリズムの検証

本章では，4，5 章において提案した Layer-1 BoD のリクエストスケジューリングアルゴリズムの検証を行う．まず，検証に利用した自作のシミュレーションプログラムについて述べ，次に予備実験として GA，DGA における交叉手法の検討を行う．そして提案アルゴリズムにおけるネットワーク利用性能の評価と考察を行い，最後にネットワーク利用性能と公平性の両面における評価と考察を行う．

6.1　Layer-1 BoD シミュレータの実装

我々は提案アルゴリズムの検証を行うために，Layer-1 BoD を模したシミュレーションプログラムを作成した．本シミュレータの機能を以下の項目で述べる．

- オンデマンドサーバの機能 Fig. 4.1 において述べた処理プロセスをプログラム上で仮想的に行うことが可能である．また，スケジューリング間隔等のパラメータを変更することが可能であ

る．また，最小遅延経路及び非最小遅延経路を設定することができる．ユーザの重みに関しては以下の値を設定できる．

- 各ユーザの重み上昇速度 (Duplation-user's Weight Period:DWP)
  ネットワーク資源未使用ユーザの重みを倍にするまでの早さ

- ユーザ設定機能ユーザを任意の人数で設定し，シミュレートを行うことが可能である．また，ユーザの希望する使用帯域やリクエストのサブミット頻度，目的接続拠点の設定等の行動制御を行うことができる．ユーザの行動制御におけるパラメータを以下に示す．

  - 開始拠点 (Source)
    通信を行う拠点の一つである．この拠点を始点としてデータ通信を行う．
  - 終端拠点 (Destination)
    通信を行う拠点の一つである．この拠点を終点としてデータ通信を行う．
  - 非最小遅延経路の許可・非許可 (Allowance Latency)
    各ユーザは非最小遅延によるデータ通信を許可するか，許可しないかのオプションを有する．
  - 希望利用帯域 (Bandwidth)
    各ユーザの希望する帯域をそれぞれ設定することが可能である．
  - 利用開始時間 (Linkup Time)
    各ユーザの希望するサービスの利用開始時間をそれぞれ設定することが可能である．
  - 利用時間 (Communication Time)
    各ユーザの希望するサービスの利用時間をそれぞれ設定することが可能である．
  - リクエスト送信間隔 (Submit Interval)
    各ユーザが送信するリクエストの時間間隔を設定することが可能である．

- 拠点及びリンクの設定複数の拠点とリンクからなるネットワークを仮想的に作成することが可能である．これにより，拠点とリンクの数やそのトポロジーをシミュレータの利用者が任意に設定することが可能である．

6.2 遺伝的アルゴリズムにおける交叉手法の検討（予備実験）

GA 及び DGA において，各個体同士がどのような交叉を行うかによって，母集団の進化，ひいては解の進化がどのように行われるか変化する．また，子が親のそれぞれの形質をどの程度受け継ぐかが評価に影響を及ぼす．

本節では，GA や DGA において，より良い探索を行うために，予備実験として，5.4.3 項にて述べた，順序最適化において代表的な CX，OX，PMX の交叉手法を GA に適用し，6.1 節にて述べたシミュレータを利用してそれぞれ数値実験を行う．それにより，以降の実験に適当な交叉手法の検討を行う．

6.2.1 実験環境

Fig. 6.1 に示すようなトポロジーにおいて，GA によるスケジューリングを解析する．また，各リンクは共通して最大利用帯域が 100Mbps であり，レイテンシは 3msec とした．なお，SINET3 ではユーザは 150Mbps 単位で必要とする帯域を要求できる．またリンク（データセンター間）は 2.4G〜40Gbps の帯域をもち，L1 と L2/3 サービスとの間で利用帯域を動的に変更することができる．ただし，本評価では解析を簡素化するため，L1 の利用帯域を 100Mbps，レイテンシは一律に 3msec とした．また本節における GA のパラメータを Table 6.1 に示す.

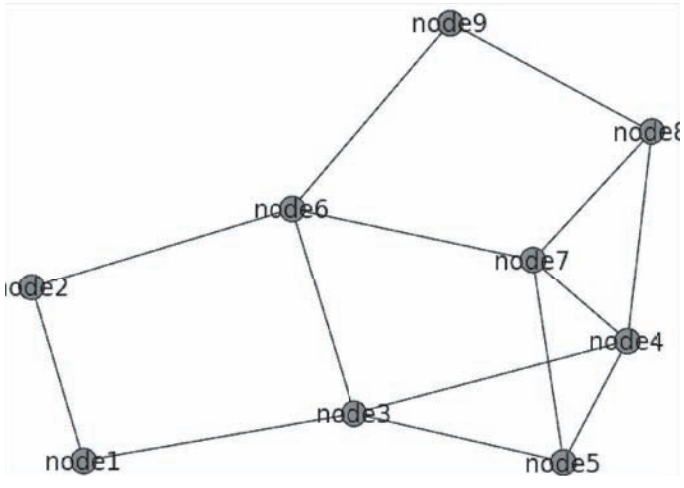

Fig. 6.1　数値実験におけるトポロジー

Table 6.1　GA におけるパラメータ

Population Size	20
Generation Num	500
Clossover Rate	0.7
Mutation Rate	0.3

次に，ユーザの行動規則を表したデータを Table 6.2 に示す.「Source」,「Destination」により通信を行う拠点間を決定している，また,「Allowance Latency」は通信遅延時間が非最短の場合でも許可するかを示すものである.「Link Up Time」はサブミットした時点から何 step 後に通信を開始したいかを表した数値である.「Submit Interval」はサブミットを行う時間間隔を表している．本論文では，スケジューリングが必ず発生するように「Submit Interval」」は全ユーザ共通とする．また，より分析を行いやすいよう，ユーザを「Bandwidth」と「Communication Time」により大きく 4 つのグループに分類した．Fig. 6.2 における点線はそれらのグループを表す．上から「使用帯域」「使用時間」共に小さいユーザ群 (GroupI),「使用帯域」は大きいが「使用時間」が短いユーザ群 (GroupII),「使用帯域」「使用時間」共に大きいユーザ群 (GroupⅢ),「使用帯域」は小さいが「使用時間」が

大きいユーザ群 (GroupIV) に分類した.

Table 6.2　各ユーザの行動制御

Group	UserName	Source	Destination	Allowance Latency	Bandwidth (Mbps)	LinkUp Time(step)	Communication Time(step)	Submit Interval(step)
	user0	node6	node9	FALSE	9	6	6	100
	user1	node9	node4	TRUE	8	8	7	100
	user2	node3	node8	TRUE	9	5	7	100
	user3	node1	node4	FALSE	10	8	2	100
I	user4	node2	node1	FALSE	6	6	11	100
	user5	node6	node4	TRUE	6	10	6	100
	user6	node4	node1	TRUE	10	10	12	100
	user7	node2	node9	FALSE	7	10	2	100
	user8	node7	node8	FALSE	9	10	7	100
	user9	node6	node8	FALSE	58	6	6	100
	user10	node5	node1	TRUE	60	9	9	100
	user11	node4	node7	FALSE	56	7	6	100
	user12	node2	node5	TRUE	51	7	9	100
II	user13	node5	node7	TRUE	50	8	9	100
	user14	node6	node4	TRUE	57	6	5	100
	user15	node7	node6	FALSE	56	8	12	100
	user16	node9	node3	TRUE	55	6	6	100
	user17	node9	node4	FALSE	60	9	12	100
	user18	node1	node5	FALSE	66	5	25	100
	user19	node4	node5	FALSE	80	7	24	100
	user20	node8	node4	TRUE	71	9	13	100
	user21	node5	node2	FALSE	76	9	15	100
III	user22	node1	node6	FALSE	63	9	22	100
	user23	node9	node1	TRUE	69	9	12	100
	user24	node1	node7	FALSE	73	5	23	100
	user25	node6	node8	FALSE	80	6	24	100
	user26	node7	node4	TRUE	65	7	19	100
	user27	node1	node7	FALSE	28	9	13	100
	user28	node6	node3	TRUE	15	9	8	100
	user29	node3	node4	TRUE	27	7	13	100
	user30	node4	node2	FALSE	17	7	11	100
IV	user31	node3	node8	FALSE	29	9	9	100
	user32	node7	node6	FALSE	30	10	11	100
	user33	node6	node3	FALSE	20	5	17	100
	user34	node1	node5	FALSE	26	9	12	100
	user35	node7	node3	TRUE	24	5	15	100

6.2.2　実験結果

CX(循環交叉)，OX(順序交叉)，PMX(部分写像交叉) のそれぞれの世代数における適合度の 5 試行における平均値を Fig. 6.2 に示す．Fig. 6.2 において横軸は世代数を表し，縦軸は適合度値を表している．また，Fig. 6.3 に最終世代の適合度値の最大値，最小値，中央値を示す.

Fig. 6.2 から，500 世代で得られた解の適合度値の平均値は CX が最も良いことが分かる．また，解の収束も早く，70 世代目には最良解に到達していることが確認できる．一方で，OX は 500 世代で得られる適合度値が最も低く，また収束に比較的長い世代数を必要としていることから，本実験環境には適さないと考えられる.

Fig. 6.3 から，全交叉手法において 5 試行の最良解の値が同じことが分かる．また，OX は低い適合度値で収束する場合があることが確認できる．一方で，平均値は CX が最も高く，良い解が出現する可能性が高いことが分かる．これらのことから，本実験環境においては CX が最も適当であるとして，本交叉手法を用いて以降の実験を行う.

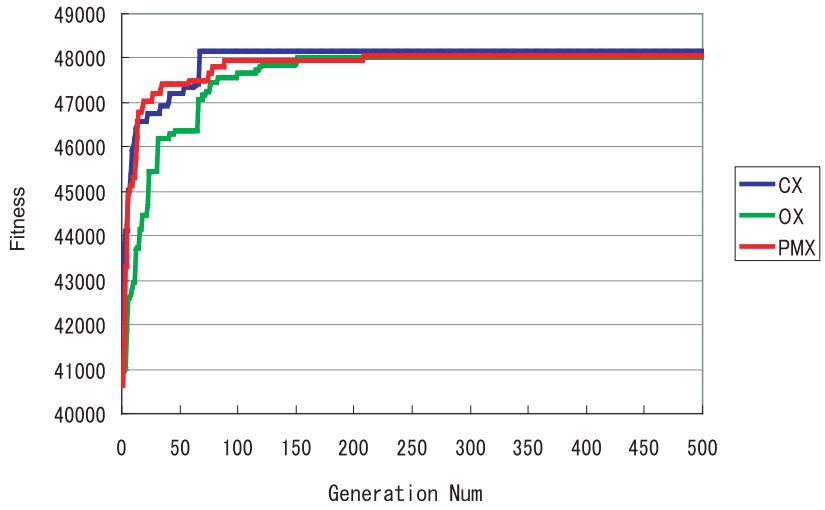

Fig. 6.2　3 交叉手法の評価 1

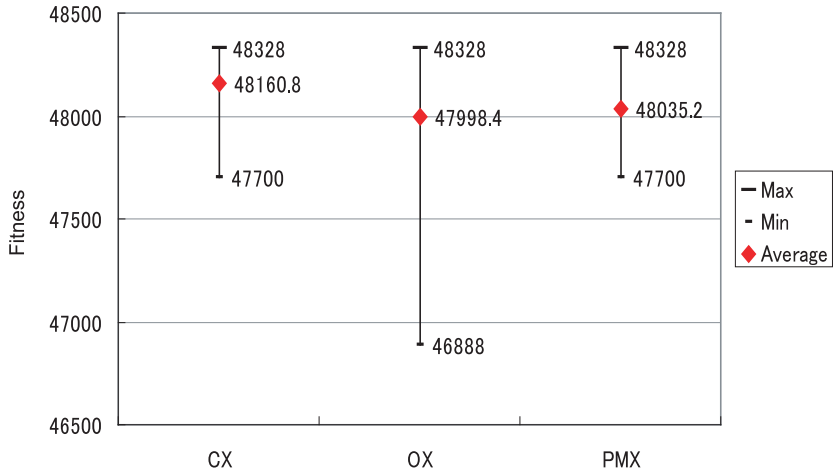

Fig. 6.3　3 交叉手法の評価 2

6.3　提案アルゴリズムにおけるネットワーク利用性能に関する有効性の検証

6.3.1　実験目的

本実験では提案アルゴリズムのネットワーク利用性能，即ち各ユーザのリクエストに対する処理能力 (スループット) に焦点を当て，最適化手法が得る適合度の比較を行う．更に，提案アルゴリズムを適用したオンデマンドサーバがスケジューリングを行った結果，受理するリクエスト，及び受理リクエストのルーティングがどのように決定されるかを検証する．また，効率の良いスケジューリングがどのようなものであるかを検証する．

6.3.2　実験環境

実験に使用するネットワークトポロジ，及びユーザの行動制御は 6.2.1 項において示した Fig. 6.1 及び，Table 6.2 を利用する．また，各最適化アルゴリズムにおいて使用したパラメータを Table 6.3

に示す．また，未使用ユーザの重み上昇速度は $DWP = 5$ とした．なお，評価を行うスケジューリングは各最適化手法において 1 回のみであるが，その試行を 5 回行う．

Table 6.3　各最適化アルゴリズムにおけるパラメータ

LS	Trial Num	10000
GA	Population Size	20
	Generation Num	500
	Clossover Rate	0.7
	Mutation Rate	0.3
DGA	Sub population Size	20
	Generation Num	100
	Island Num	5
	Clossover Rate	0.7
	Mutation Rate	0.3
	Elite Num	2
	Migration Rate	0.2
	Migration Interval	5

6.3.3　実験結果と考察

スケジューリングの結果，各最適化アルゴリズムが得た適合度値を Table 6.4 に示す．Table 6.4 では各最適化アルゴリズムを 5 試行行い，適合度値の最大値，最小値，平均値を表した表である．Table 6.4 では最も良い適合度を得られるのは DGA であり，最も悪い適合度を得るのは FIFO であることが分かる．またこの適合度値は，本実験環境においては全ユーザの重みが同値 (0.5) であるため，スループットと対応している．即ち Table 6.4 で示した表は各最適化アルゴリズムのネットワーク利用効率を示している．このことから，本実験環境においては DGA が最もスループットが高く，かつ探索能力において優れていることが確認できる．

Table 6.4　各最適化アルゴリズムが得た適合度値

	FIFO	GR	LS	GA	DGA
Max	38092.0	38324.0	47700.0	48328.0	48328.0
Min	26764.0	38324.0	42984.0	48120.0	48120.0
Average	32867.2	38324.0	44934.4	48244.8	48286.4

また，ある試行における各最適化アルゴリズムにおける受理の可否とルーティングを Table 6.5 に示す．Table 6.5 から DGA，GA や LS は GR や FIFO と比較して受理したリクエストが多いことが分かる．これは，DGA，GA，LS は GR，FIFO より効率的に各ユーザに資源を分配した結果である．Table 6.5 における GroupI，Ⅳ の各順序最適化アルゴリズムのスケジューリングを見ると，評価値の最も高い DGA，GA で最も多くのユーザのリクエストを受理しているが，GR や FIFO では多くの棄却が発生している．また，user2 や user6，user29 等のリクエストは DGA，GA では経由ノードが最小になっているが，評価値の低い GR や FIFO では経由ノードが多く，使用リンク数が多いことが分かる．GroupⅡ を見ると全てのアルゴリズムで受理できるリクエストが少ないことが分かる．これは GroupⅡ のリクエストタイプのように使用帯域が大きく，使用時間が短いユーザは他のユーザと競合した際，受理されにくいことを意味する．GroupⅢ のユーザはネットワーク利用度が大きいユー

ザ郡である．そのため，GR では最も優先される．しかしながら，GroupⅢ における GR の受理するリクエスト数は DGA，GA よりも少ない．これは，リクエストを受理した際のルーティングが大きく影響しているためである．具体的には，user26 のリクエストを見ると，経由ノード数が最小で 2 であるにも関わらず，GR では 7 つのノードを経由している．このようなネットワーク利用度の大きいユーザのリクエストに多数の資源を与えるルーティングはネットワーク全体を圧迫するため，非効率な資源割り当てである．

Table 6.5　各最適化アルゴリズムにおける受理の可否とルーティング

Group	User	FIFO	GR	LS	GA, DGA
Ⅰ	user0	node6, node9		node6, node9	node6, node9
	user1	node9, node8, node4	node9, node8, node4	node9, node8, node4	node9, node8, node4
	user2	node3, node1, node2, node6, node7, node8	node3, node5, node7, node8	node3, node4, node8	node3, node4, node8
	user3				node1, node3, node4
	user4		node2, node1	node2, node1	node2, node1
	user5	node6, node9, node8, node4	node6, node7, node5, node4	node6, node3, node4	node6, node3, node4
	user6	node4, node7, node6, node2, node1	node4, node5, node7, node6, node2, node1		node4, node3, node1
	user7	node2, node6, node9	node2, node6, node9	node2, node6, node9	node2, node6, node9
	user8	node7, node8	node7, node8	node7, node8	node7, node8
Ⅱ	user9				
	user10				
	user11				
	user12				
	user13	node5, node7		node5, node7	node5, node7
	user14	node6, node3, node4			
	user15				
	user16			node9, node6, node3	node9, node6, node3
	user17				
Ⅲ	user18				node1, node3, node5
	user19		node4, node5	node4, node5	node4, node5
	user20			node8, node4	node8, node4
	user21			node5, node3, node1, node2	
	user22	node1, node2, node6	node1, node2, node6		node1, node2, node6
	user23	node9, node8, node4, node5, node3, node1			
	user24		node1, node3, node4, node7		
	user25	node6, node7, node8	node6, node7, node8	node6, node7, node8	node6, node7, node8
	user26	node7, node4	node7, node5, node3, node6, node9, node8, node4	node7, node4	node7, node4
Ⅳ	user27				
	user28	node6, node3		node6, node3	node6, node3
	user29	node3, node5, node4	node3, node6, node9, node8, node4	node3, node4	node3, node4
	user30	node4, node3, node1, node2		node4, node3, node1, node2	node4, node3, node1, node2
	user31				
	user32				
	user33	node6, node3		node6, node3	node6, node3
	user34		node1, node3, node5		
	user35	node7, node4, node3	node7, node4, node3	node7, node4, node3	node7, node4, node3

6.4　提案アルゴリズムにおけるネットワーク利用性能と公平性に関する有効性の検証

6.4.1　実験目的

6.3 節においては，スループットのみに注目し一回のみのスケジューリングの結果，及び効率的な受理・ルーティングを確認した．しかし，Layer-1 BoD においてそれぞれのユーザは通信する必要がある度にリクエストをサブミットする．そのため提案アルゴリズムにおいて複数回のスケジューリングが発生するとき，スループットと公平性においてどのような性能を示すかを確認する．

6.4.2　実験環境

実験に使用するネットワークトポロジ，及びユーザの行動制御は 6.2.1 項において示した Fig. 6.1 及び，Table 6.2 を利用する．また，各最適化アルゴリズムにおいて使用したパラメータは 6.3.2 項の Table 6.3 に示したものを利用する．また，スケジューリングは各最適化アルゴリズムにおいて 12 回連続で行った．また，未使用ユーザの重み上昇速度は DWP の値を $DWP = 1, 5, 10, 50$ でそれぞれ実験を行った．

6.4.3　実験結果と考察

DWP の値が 1，5，10，50 のときの各最適化アルゴリズムにおける全ユーザのネットワーク利用度の総和を Fig. 6.4 に示す．Fig. 6.4 は 12 回のスケジューリングを通しての各アルゴリズムのスループットを表している．Fig. 6.4 から，全 DWP の場合において，FIFO が最低の値となっている．このことから，既存のアルゴリズムよりも最適化を利用した提案アルゴリズムがよりスループットの面において有効であることが分かる．また DWP の値が大きい程，FIFO，GR を除く全最適化アルゴリズムにおいてスループットが高くなることが分かる．特に $DWP = 50$ のとき DGA で最も高いスループットを得ることができた．

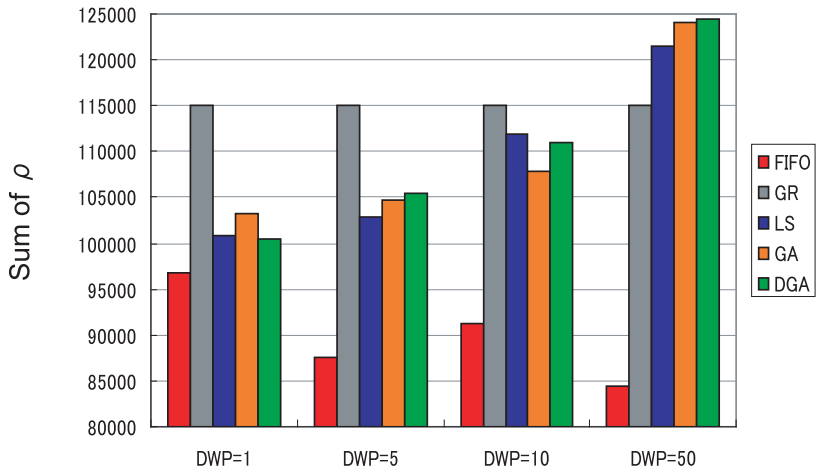

Fig. 6.4　各ユーザのネットワーク利用度の総和

次に，DWP の値が 1，5，10，50 のときの各最適化アルゴリズムにおける全ユーザのネットワーク利用度の分散値を Fig. 6.5 に示す．Fig. 6.5 は 12 回のスケジューリングを通しての各最適化アルゴリズムの公平性を表している．値が少ない手法ほど公平なスケジューリングを行っており，値が大きい程ネットワークを利用するユーザに偏りが生じている．Fig. 6.5 から全 DWP の場合において GR が最も分散値が高く，不公平なスケジューリングを行っていることが分かる．一方で DWP が 1 のとき DGA で最も分散値が低く，公平なスケジューリングを行えていることが確認できる．また Fig. 6.5 と Fig. 6.4 での結果から DWP の値が 1,5 において DGA は FIFO よりもスループットが高く，更に公平性も高いスケジューリングを行っていることから，既存の手法よりも効率的にネットワークを利用していることが分かる．しかし DWP が大きくなると，分散値が逆転し，スループットが高くなる一方で，FIFO よりも不公平なスケジューリングを行ってしまうことが分かる．

なぜこのような結果になるのかを確認するために，$DWP = 1, 5, 10, 50$ における各最適化アルゴリズムの各スケジューリング時のユーザの重み変化を Fig. 6.6，Fig. 6.7，Fig. 6.8，Fig. 6.9 にそれぞれ示す．それぞれの図において，横軸は時間をスケジューリング番号に対応させて示しており，縦軸は各ユーザの重みの値を示している．各図において，ユーザの重み減少しているタイミングがあるが，これはその時間にリクエストの受理が行われたことが原因で生じる．このことから，重みが上昇し続けているユーザはアクセプトが却下され続けていることを意味する．

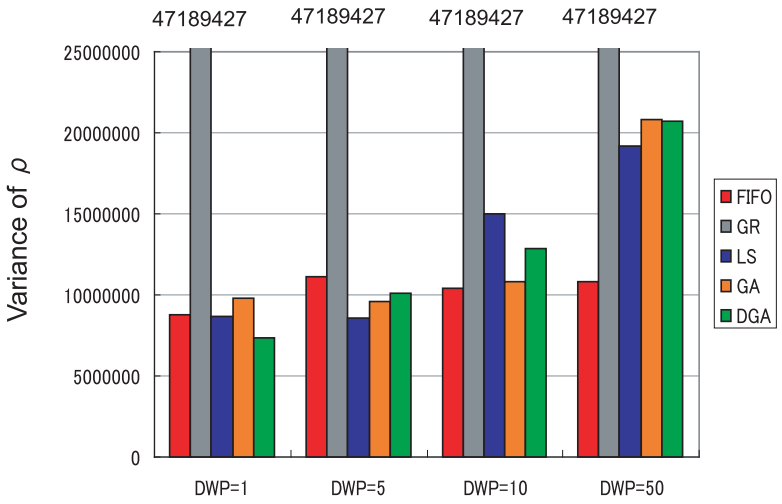

Fig. 6.5　各ユーザのネットワーク利用度の総和分散値

Fig. 6.6 から $DWP = 1$ において，FIFO では最大で 11 回目のスケジューリングまでアクセプトされていないユーザが存在していることが分かる．また一方で，LS，GA，DGA では，長くとも 5 回目のスケジューリングまでに全てのユーザのリクエストをアクセプトしていることが確認できる．また，LS，GA，DGA では多くのユーザが 1 回または 2 回のスケジューリングの間隔でアクセプトされている．これらのことから，$DWP = 1$ においては，LS，GA，DGA で重みが適当に作用し，公平なスケジューリングを行っていると考えられる．

次に，Fig. 6.9 から $DWP = 50$ において全ての最適化アルゴリズムで最終のスケジューリング (12 回目) までに，それぞれのユーザの重みに差が生じていることが確認できる．特に LS，GA，DGA では GroupⅢ の user18 や user19 は積極的にリクエストが受理され，GroupI や GroupⅡ の user4 や user10 では一度もアクセプトされない，またはアクセプトされるまで多くの時間を必要としていることが分かる．これは，ネットワーク資源未使用ユーザの重みの上昇が遅いため，ネットワークを利用することによる重みの減少よりも，ネットワーク利用度が大きくなるユーザを優先しているからである．そのために相対的に見て，LS，GA，DGA よりも FIFO の方が公平性の高い処理となっていると思われる．

これらのことから，スループットと公平性にはトレードオフの関係があり，適当なスケジューリングを行うためには環境に適した DWP の設定が重要であることが分かる．

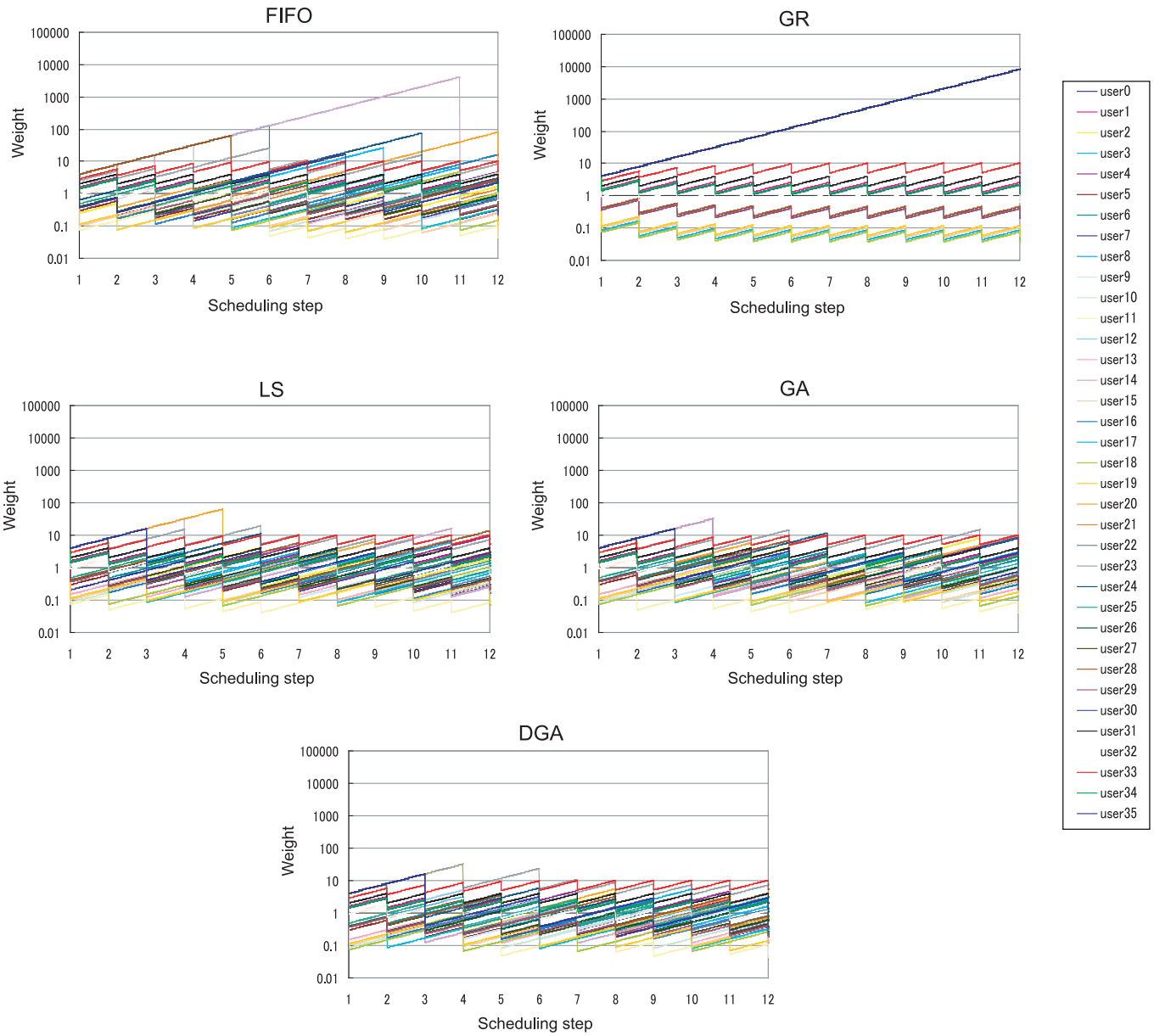

Fig. 6.6　DWP=1 における各最適化アルゴリズムのユーザの重みの変化

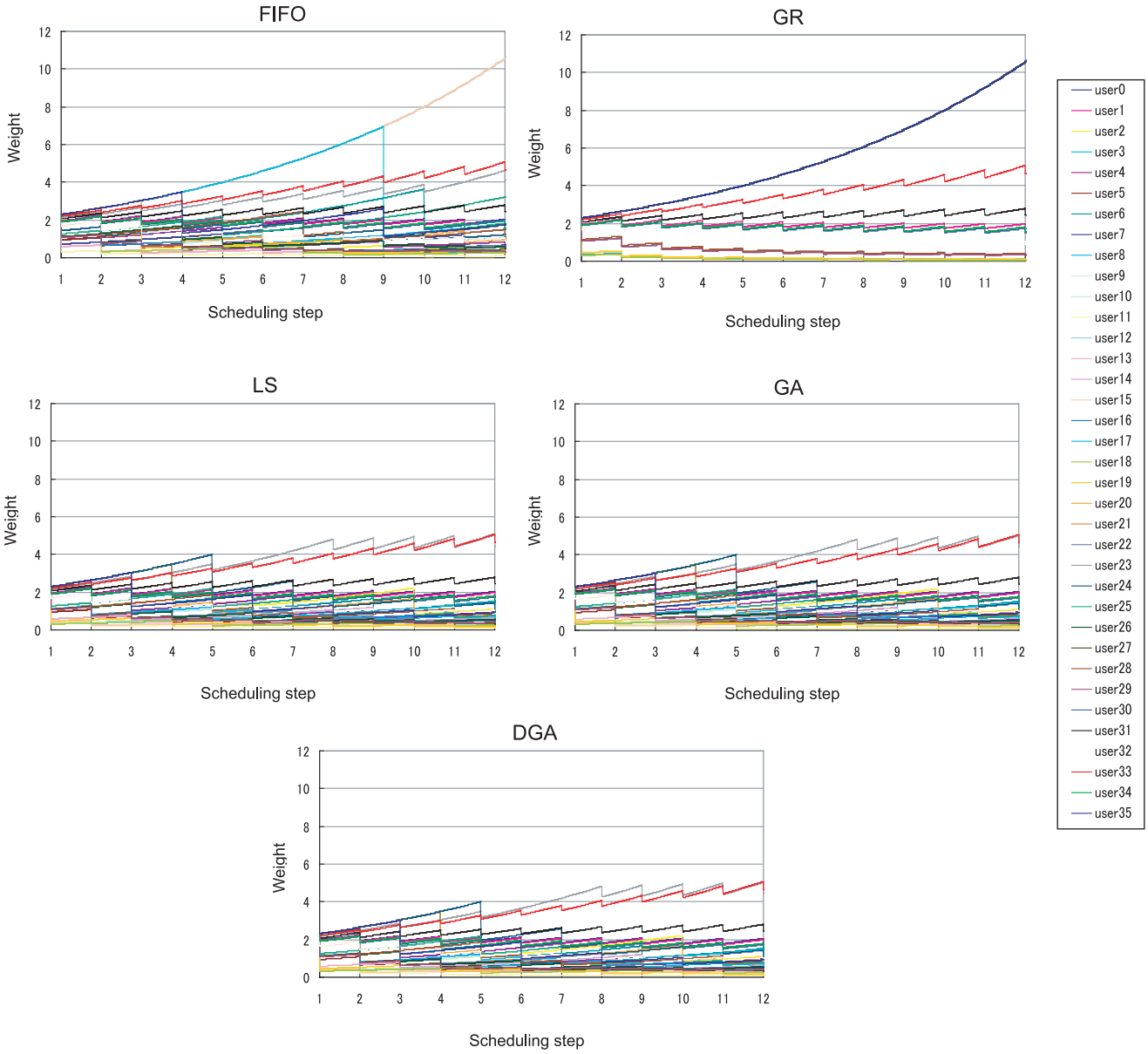

Fig. 6.7　DWP=5 における各最適化アルゴリズムのユーザの重みの変化

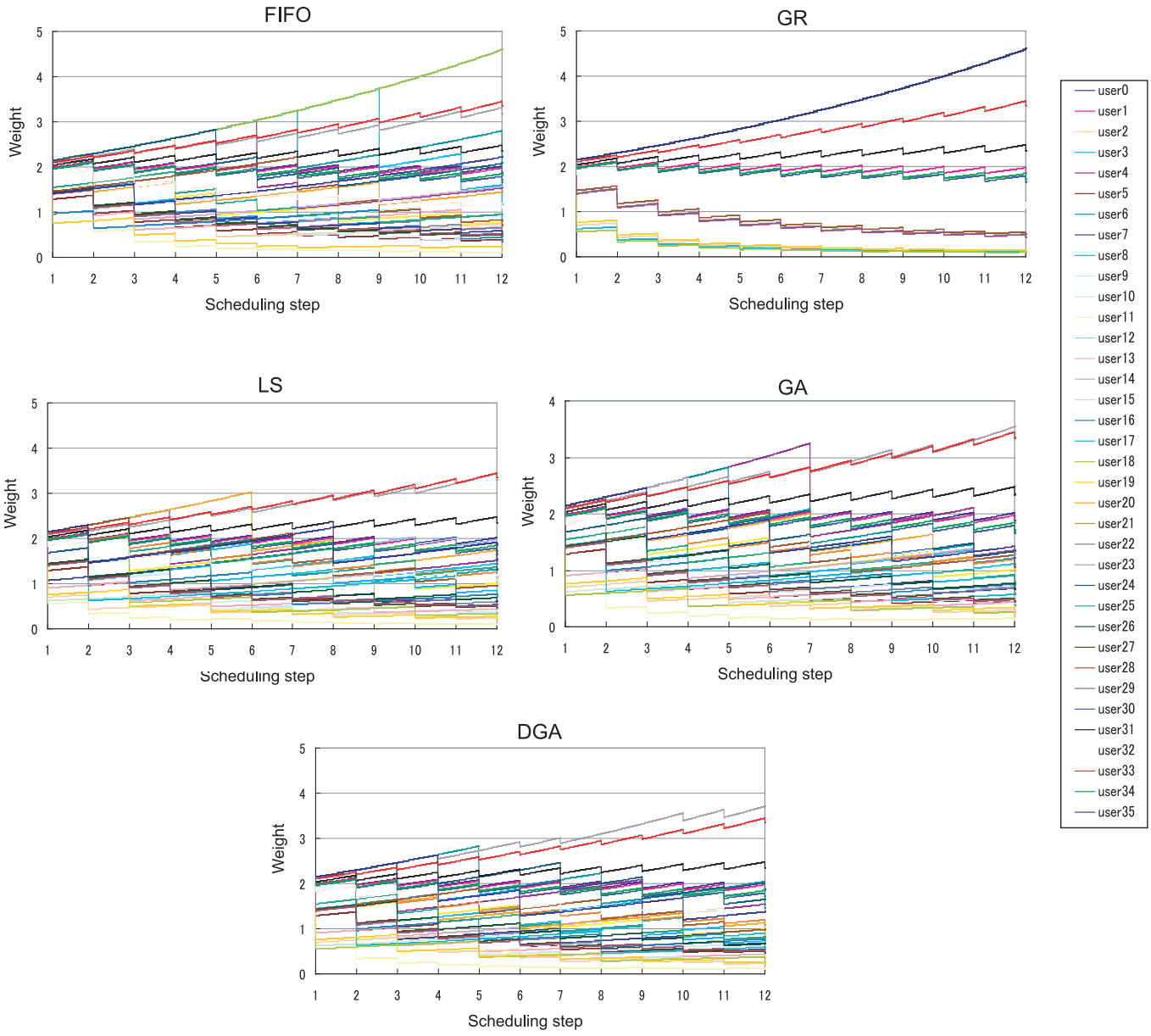

Fig. 6.8　DWP=10 における各最適化アルゴリズムのユーザの重みの変化

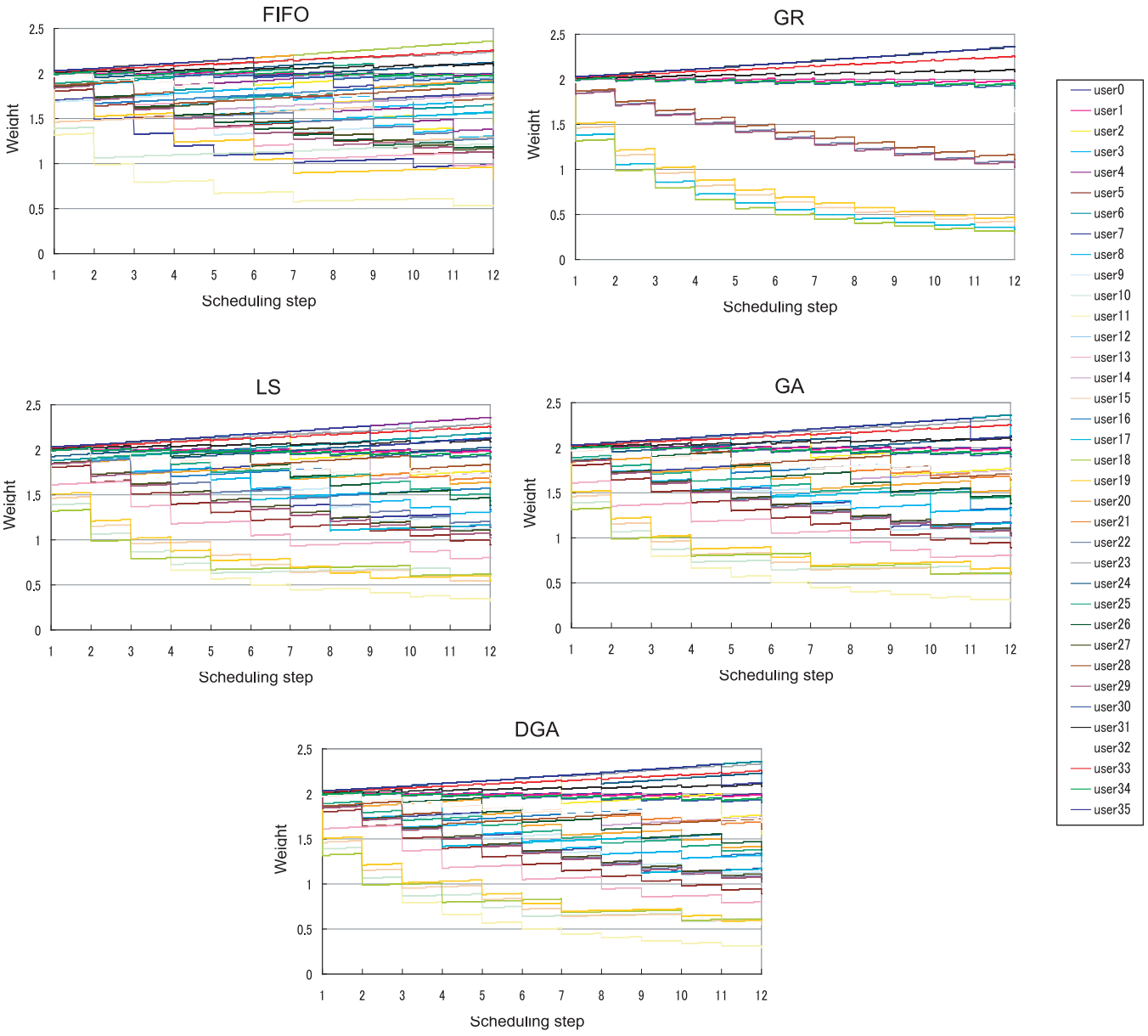

Fig. 6.9　DWP=50 における各最適化アルゴリズムのユーザの重みの変化

7　まとめ

本研究では，学術情報ネットワークなどの先進的なインターネットにおける，帯域予約型のサービスである Layer-1 BoD を対象に，その効率的なリクエストスケジューリングを行う手法を提案した．

Layer-1 BoD におけるスケジューリングアルゴリズムの要件として,「全体のスループットを高く保つ」,「全ユーザが平等にネットワークリソースを利用する」という 2 項目を挙げた．そしてそれらを満たすためのスケジューリングアルゴリズムを提案した．

本研究における提案アルゴリズムは，SINET3 で提供されている Layer-1 BoD のユーザへの帯域割当てのプロセスをベースに，ネットワークリソースを効率的に運用することを目的としている．また，本研究では，CPU スケジューラやジョブスケジューラで広く利用されているキューの概念を取り入れ，リクエストの処理順序を最適化することにより，受理の可否と受理する際に割り当てるネットワークリソース（ルーティング）を決定する．

最適化の手法として，以下の 5 つの最適化手法を適用した．

- FIFO

  早くサブミットされたリクエストから順に受理していく手法．最もシンプルな既存手法である．

- GR

  問題の要素を複数の部分問題に分割し，それぞれを独立に評価を行い，評価値の高い順に取り込むことで解を得る手法

- LS

  ある解からその解の近傍（次解候補）のうちある条件で一つを選び近傍解とする最適化手法

- GA

  生物の進化の過程を工学的に摸倣した確率的な最適化手法

- DGA

  GA における母集団を複数のサブ母集団に分割し，各サブ母集団 (Sub-Population) ごとに独立に遺伝的操作を行う最適化手法

上記の最適化手法によるスケジューリングアルゴリズムの検証を行うために，Layer-1 BoD シミュレータを作成し，提案手法を実装し数値実験を行った．実験の結果，パラメータを適切に設定することによって DGA，GA，LS で FIFO よりもスループット，公平性の両面で良い性能を発揮することが分かった．また，評価の高いスケジューリングでも，スループットまたは公平性のどちらか一方で FIFO よりも悪い性能となる場合があることが分かり，これによりスループットと公平性のトレードオフ関係を確認した．

謝辞

本研究を遂行するにあたり，主担当教員として 3 年間に渡り多大なる御指導そして御協力を頂きました，同志社大学生命医科学部の廣安知之教授に心より感謝いたします.

また，副担当教員として様々な指摘や助言をして頂き，様々な研究活動の機会を与えて頂きました，同志社大学工学部の三木光範教に心より感謝いたします.

また，研究の方向性などたくさんの助言を下さった同志社大学工学部の吉見真聡助教に心より感謝いたします.

これまで三木先生，廣安先生のご指導の下，3 年間の研究活動に尽力して今に至りましたことを，この場を借りて重ねて厚く御礼申し上げます．本当にありがとうございました.

そして，共同研究として Layer-1 BoD における様々な情報をご教授して頂いた NII の鯉渕道紘先生に敬意を込めて感謝いたします，本研究は鯉渕先生のご協力なしには成し得ませんでした．誠にありがとうございました.

日頃から研究内容や研究の進め方などについて相談をして頂いた卒業生の同志社大学大学院 工学研究科 知識工学専攻 博士前期課程の渡辺崇史さんと木浦正博さんに心より感謝いたします.

また，本論文の構成や内容に関して多くの助言を頂いた同志社大学大学院 工学研究科 知識工学専攻 博士前期課程の山下尊也さん，岡田典子さんに感謝いたします．そして知的システムデザイン研究室の皆様に心より感謝いたします．特に，3 年間同じ環境で多くの経験を共有した 11 期生の皆様に心より感謝いたします．非常に意欲的で活動的な皆さんと素晴らしい環境の下で研究ができたことを心から嬉しく思います.

最後に，私の研究活動に様々な面で支えとなってくれた家族，そして私を支えてくれた全ての方々に深く感謝します．本当にありがとうございました.

参考文献

1) Shigeo Urushidani, Jun Matsukata, Kensuke Fukuda, Shunji Abe, Yusheng Ji, Michihiro Koibuchi, Shigeki Yamada, Kaori Shimizu, Tomonori Takeda, Ichiro Inoue, and Kohei Shiomoto. Layer-1 bandwidth on demand services in sinet3. *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pp. 2286–2291, Nov 2007.

2) Shigeo Urushidani, Shunji Abe, Yusheng Ji, Kensuke Fukuda, Michihiro Koibuchi, Motonori Nakamura, Shigeki Yamada, Kaori Shimizu, Rie Hayashi, Ichiro Inoue, and Kohei Shiomoto. Design of versatile academic infrastructure for multilyaer network services. *IEEE Journal on Selected Areas in Communications*, Apr 2009.

3) Hidehiro Kobayashi and Masaharu Munetomo. Bandwidth allocation using genetic algorithms. *Joho Shori Gakkai Kenkyu Hokoku*, Vol. 99, pp. 91–96, Apr 1999.

4) Louis Anthony Cox. Dynamic anticipatory routing in circuit-switched telecommunications networks. *Handbook of Genetic Algorithms*, Vol. 99, pp. 124–143, 1991.

5) Shigeo Urushidani. Design of versatile academic infrastructure for multilyaer network services. *Proceedings of SPIE*, Vol. 6784, , Apr 2007.

6) A Bricker, M Litzkow, and M Livny. Computer sciences department, university of wisconsin. *Computer Sciences Techical Report*, 1992.

7) 川崎考蔵, 鯉渕道紘, 漆谷重雄, 廣安知之, 三木光範, 吉見真聡. インターネットの帯域オンデマンドサービスにおけるスケジューリングアルゴリズム. 信学技報, Vol. 109, No. 168, pp. 73–78, Aug 2009.

8) Fran Berman, Geoffrey Fox, and Anthony J. G. Hey. *Grid computing: making the global infrastructure a reality.* John Wiley and Sons Inc, May 2003.

9) P Preux and EG Talbi. Towards hybrid evolutionary algorithms. *International Transactions in Operational Research*, pp. 224–230, 1999.

10) S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *International Transactions in Operational Research*, Vol. 21, No. 2, pp. 498–516, Mar 1973.

11) Glover, Fred W. Kochenberger, and Gary A. *Handbook of Metaheuristics*, Vol. 57 of *International Series in Operations Research and Management Science*. SPRINGER NETHERLANDS, 2003.

12) D.E.Goldberg. *Genetic Algorithms in Search Optimization and Machine Learnig.* Addison-Wesley, 1989.

13) John H. Hollandy. *Adaptation in Natural and Artificial Systems.* The MIT Press, April 1992.

14) Michalewicz Z. *Genetic Algorithm + Data Structures = Evolution Programs,.* Springer-Verlag, 1992.

15) Reiko Tanese. Distributed genetic algorithms. *Proc.3rd ICGA*, pp. 434–439, 1989.

16) J.Nang and K.Matsuo. A survey on the parallel genetic algorithms. *J.SICE*, Vol. 33, No. 6, 1994.

17) Erick Cantu Paz. A survey of parallel genetic algorithms. *Calculateurs Paralleles*, 1998.

付 図

付 表

付録：発表論文リスト

1. **川崎 考蔵**，鯉渕 道紘，漆谷 重雄，廣安 知之，三木 光範
   レイヤ 1 帯域オンデマンドサービスにおけるスケジューリングアルゴリズムの基礎的検討並列/分散/協調処理に関するサマー・ワークショップ (SWoPP 佐賀 2008)（2008.8.7）

2. **川崎 考蔵**，鯉渕 道紘，漆谷 重雄，廣安 知之，三木 光範
   帯域予約型サービスにおける効率的な資源分配アルゴリズムの検証並列/分散/協調処理に関するサマー・ワークショップ (SWoPP 仙台 2009)（2009.8.5）

3. **Kozo Kawasaki**, Tomoyuki Hiroyasu, Michihiro Koibuchi, Shigeo Urushidani, Mitunori Miki, Masato Yoshimi
   Efficient Scheduling Algorithms on Bandwidth Reservation Service of Internet using Metaheuristics 9th International Conference on Intelligent Systems Design and Applications (ISDA'09)（2009.11.29）