Vol.2015-OS-132 No.9
2015/2/27

# EDFを基にした スケジューリングアルゴリズムの提案

林　竜太[1]　兪　明連[1]　横山　考典[1]

**概要**：組み込みリアルタイムシステム向けのスケジューリングアルゴリズムの提案に関する論文．マルチプロセッサ環境下でのタスクの優先度変更が実行時に発生するシステムを想定し，動的優先度アルゴリズムである EDF を基にしたアルゴリズムに着目する．現在，EDZL，EDCL といったアルゴリズムが提案されてきているが実装やスケジュール成功率などの面でそれぞれ問題点を持っている．問題点の原因としてスケジューラの起動頻度あげ，スケジューラの起動頻度とスケジュール成功率の関係を分析するため，EDZL，EDCL の変型モデルである mEDZL，mEDCL を作成した．分析の結果から，本論分ではスケジューラの起動頻度を抑えつつ高いスケジュール成功率を実現するスケジューリングアルゴリズムの提案を目的とする．

## 1. はじめに

近年の組み込みリアルタイムシステムは，ロボット，ユビキタスアプリケーションなどの登場により高性能なプラットフォームを必要としてきている．組み込みシステムには発熱，消費電力の制約があるため従来のシングルプロセッサ環境で動作周波数を上げ，性能の向上を行うことは現実的ではない．そこで，マルチプロセッサ方式の利用が一般化してきている [1].

したがって，マルチプロセッサ環境での組み込みリアルタイムシステム向けのスケジューリングアルゴリズムが求められている．現在，タスクの優先度を実行前に決定する静的優先度アルゴリズムである RM(Rate Monotonic)[2] や，タスクの優先度が実行時に決定される動的優先度アルゴリズムである EDF(Earliest Deadline First)[2] を基にしたアルゴリズムが提案されてきている．本論文では，タスクの優先度が実行中に発生するシステムを想定し，動的優先度アルゴリズムである EDF に着目する．

EDF に基づくスケジューリングアルゴリズムとして EDZL(Earliest Deadline Zero Laxity)[3] がある.EDZL はスケジューリング成功率が高いことで知られているが，スケジューラの起動がに毎単位時間行われるため実装が困難になるという問題点がある．この問題点を解決するために，EDCL(Earliest Deadline Critical Laxity)[4] が提案された．EDCL はスケジューラの起動をジョブの起動時と完了時のみに制限することによって EDZL よりも実装を容易にしたアルゴリズムである．しかし，EDZL と比較して EDCL はスケジュール成功率が低いという問題点がある．このことから，スケジューラの起動頻度とスケジュール成功率はトレードオフの関係にあることがわかる．この関係を分析するために，EDZL と EDCL のスケジューラの起動タイミングをそれぞれ変更した場合の変型モデル mEDZL(modified Earliest Deadline Zero Laxity)[5]，mEDCL(modified Earliest Deadline Critical Laxity)[5] を作成した．mEDZL は EDCL と同様に，スケジューラの起動をジョブの起動時，完了時に制限するアルゴリズムである．mEDCL は EDZL と同様に，スケジューラの起動を毎単位時間行うアルゴリズムである．EDZL，EDCL，そして変型モデルである mEDZL，mEDCL のスケジュール成功率をそれぞれ比較した結果，スケジューラの起動が毎単位時間行われる EDZL，mEDCL はスケジュール成功率が高く，スケジューラの起動がジョブの起動時，完了時のみの EDCL，mEDZL はスケジュール成功率が低いという結果となった．

このことから，スケジュール成功率を上げるためにはスケジューラの起動頻度を増やす必要があることがわかる．しかし，スケジューラの起動を毎単位時間行うアルゴリズムは実装が困難なため，本論文ではスケジューラの起動頻度を抑えつつ，高いスケジュール成功率を実現したスケジューリングアルゴリズムを提案する．

[1] 東京都市大学
Tokyo City University, Setagaya, Tokyo 1-28-1, Japan

## 2. システムモデル

本論文では，周期的タスクからなるタスクセットのスケジューリングを対象とし，マルチプロセッサ環境での研究における一般的なシステムモデル [1] を仮定する．システムは $M$ 個の同一性能のプロセッサを持ち，$N$ 個のタスクから構成されるタスクセット $\tau = \{\tau_1, \tau_2, ...\tau_i...\tau_N\}$ が与えられる．各タスク $\tau_i$ は最悪実行時間 $C_i$ と周期 $T_i$ から，$\tau_i = (C_i, T_i)$ と表す．また，タスク $\tau_i$ のデッドラインを $d_i$ とする．タスク $\tau_i$ の利用率 (ひとつのタスクが実行を完了するまでに必要な CPU の割合) を $U_i = C_i/t_i$ で表し，そしてタスクセット $\tau$ に含まれるタスク利用率の合計を $U(\tau) = \sum_{i=1}^{N} U_i$ で表す．以後，$U(\tau)/M$ をシステム利用率と呼ぶ．各タスク $\tau_i$ は周期 $T_i$ の間隔でジョブを生成する．タスク $\tau_i$ が $k$ 番目に生成するジョブを $\tau_{ik}$ と表す．また，ジョブ $\tau_{ik}$ のデッドライン $d_{ik}$ は次のジョブの起動時刻 $r_{ik+1}$ に等しい．つまり，$d_{ik} = r_{ik+1} = r_{ik} + T_i$ となる．ある時刻 $t$ において，ジョブの残り実行時間を $C_{ik}(t)$ としたとき，$\tau_{ik}$ の余裕時間 (Laxity) を $L_{ik}(t) = d_{ik} - t - C_{ik}(t)$ と定義する．余裕時間とはそのジョブの緊急度を示しており，緊急度は余裕時間が少なくなるにつれ増加し，ゼロとなったとき最高となる．そして余裕時間が負であることはデッドラインミスを意味している．ジョブ $\tau_{ik}$ が実行可能状態であるときに全てのプロセッサが $\tau_{ik}$ よりも優先度の高いジョブによって使用されているとき，ジョブ $\tau_{ik}$ はそれらのジョブにブロックされているという．タスクはお互いに独立しており，割り込みが可能であるとする．また，いかなるタスクでも複数のプロセッサで同時に実行することができないとする．つまり，ある 1 つのタスクが並列に実行されることはないとする．そして一度実行が開始されたら新たなタスクが到着したり，すでに存在しているタスクが消滅することはないものとする．

## 3. 関連研究

本章では関連研究である EDF,EDZL,EDCL の概要について述べる．また，EDZL,EDCL の変型モデルである mEDZL,mEDCL の概要と，スケジューラの起動頻度とスケジュール成功率の関係についての分析結果について述べる．

### 3.1 EDF

EDF(Earliest Deadline First) は，デッドラインの早いタスクに高い優先度を与える動的優先度スケジューリングアルゴリズムである．図 1 に EDF のスケジューリング例を示す．ただし，実行するタスクセット及びプロセッサ数は $\tau = \{\tau_1 = (3,5), \tau_2 = (2,5), \tau_3 = (4,5), \tau_4 = (2,10)\}$, $M = 2$ である．図 1 より，時刻 0 において全てのタスクが実行可能状態であるがデッドラインの早いタスク $\tau_1$，$\tau_2$ を優先している．($\tau_3$ も $\tau_1$，$\tau 2$ とデッドラインが同じだが，このような場合はタスクのインデックスが若い順に優先する．)EDF によってスケジュールを行った結果，時刻 5，時刻 10 において $\tau_3$ がデッドラインミスを起こしている． EDF はジョブの起動時，完了時にスケジューラを起動するアルゴリズムである．

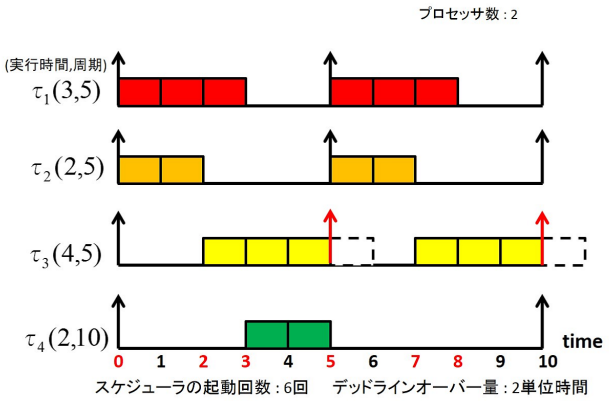


図 1 EDF の実行例

### 3.2 EDZL

EDZL(Earliest Deadline Zero Laxity) は，EDF にゼロ余裕時間ルールを適用したアルゴリズムである．ゼロ余裕時間ルールとは，余裕時間が 0 となったジョブに最高優先度を与えるというルールである．つまり，通常は EDF によってスケジューリングを行い，ジョブの余裕時間が 0 となったらそのジョブに最高優先度にするというアルゴリズムである．図 2 に EDZL の実行例を示す．ただし，実行するタスクセット及びプロセッサ数は $\tau = \{\tau_1 = (3,5), \tau_2 = (2,5), \tau_3 = (4,5), \tau_4 = (2,10)\}$, $M = 2$ である．図 2 より時刻 1 において，$\tau_3$ の余裕時間が 0 となり，最高優先度を獲得する．結果，$\tau_2$ がプリエンプトされている．EDZL は毎単位時間スケジューラを起動するアルゴリズムである．

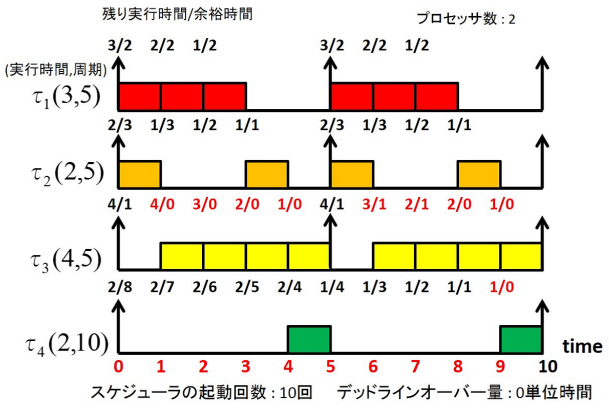


図 2 EDZL の実行例

### 3.3 EDCL

EDCL(Earliest Deadline Critical Laxity) は通常は EDF と同様の動作を行うが，余裕時間がクリティカル (危機的状況) となったジョブに最高優先度を与えるアルゴリズムである．「余裕時間がクリティカル」とは，ある時刻 $t_s$ における最高優先度を持つ $\tau_{hp}$ の最小の残り実行時間を $e_{hp}$ とし，その残り実行時間 $e_{hp}$ よりもタスク $\tau_i$ の余裕時間 $x_i(t_s)$ が小さい状態のことを指す．図 3 に EDCL の実行例を示す．ただし，実行するタスクセット及びプロセッサ数は $\tau = \{\tau_1 = (3,5), \tau_2 = (2,5), \tau_3 = (4,5), \tau_4 = (2,10)\}$, $M = 2$ である．図 3 より，時刻 0 において高い優先度を持つ $\tau_1$，$\tau_2$ のうち，最小の残り実行時間は $\tau_2$ の 2 である．その残り実行時間よりも小さい余裕時間は $\tau_3$ の 1 である．よって $\tau_3$ に高い優先度が与えられ実行している．EDCL はスケジューラの起動をジョブの起動時，完了時に行うアルゴリズムである．

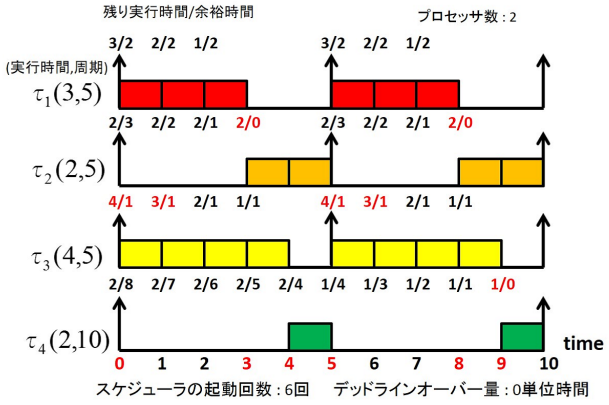


図 3　EDCL の実行例

### 3.4 利点と問題点のまとめ

表 1 に，EDZL,EDCL の利点と問題点をまとめる．表 1 より，スケジューラの起動頻度が多いと実装が困難となり，実用性が低下する．よって，スケジューラの起動頻度とスケジュール成功率はトレードオフの関係となっていることがわかる．この関係を調査するために，EDZL,EDCL のスケジューラの起動タイミングをそれぞれ変更した場合の変型モデル mEDZL,mEDCL を作成し，分析を行った．

表 1　EDZL,EDCL の利点と問題点

| | 利点 | 問題点 |
|---|---|---|
| EDZL | スケジュール成功率が高い | スケジューラを毎単位時間起動するため実用性が低い |
| EDCL | スケジューラの起動をジョブの起動時，完了時のみに制限するため実装が高い | EDZL と比べスケジュール成功率が低い |

### 3.5 分析モデル mEDZL

mEDZL(modefied Earliest Deadline Zero Laxity) は，EDZL のスケジューラの起動をジョブの起動時，完了時に制限するアルゴリズムである．図 4 に mEDZL の実行例を示す．ただし，実行するタスクセット及びプロセッサ数は $\tau = \{\tau_1 = (3,5), \tau_2 = (2,5), \tau_3 = (4,5), \tau_4 = (2,10)\}$, $M = 2$ である．図 4 より，時刻 1 において $\tau_3$ の余裕時間が 0 となっているが，時刻 1 ではスケジューラは起動されないためプリエンプションが発生することがなく実行されている．その結果，時刻 5，10 においてそれぞれデッドラインミスを起こしている．

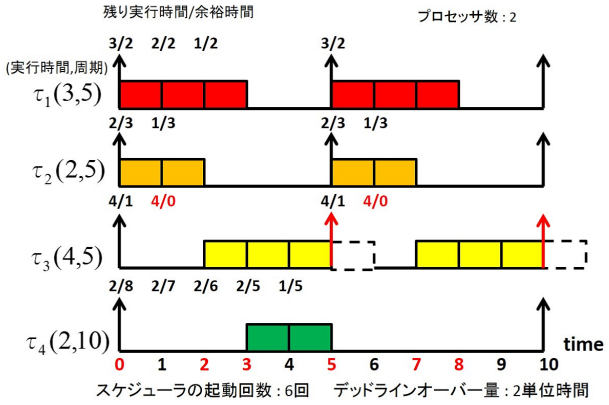


図 4　mEDZL の実行例

### 3.6 分析モデル mEDCL

mEDCL(modified Earliest Deadline Critical Laxity) は，EDCL のスケジューラの起動を毎単位時間行うアルゴリズムである．図 5 に mEDCL の実行例を示す．ただし，実行するタスクセット及びプロセッサ数は $\tau = \{\tau_1 = (3,5), \tau_2 = (2,5), \tau_3 = (4,5), \tau_4 = (2,10)\}$, $M = 2$ である．図 5 より，EDCL の実行例と同様に時刻 0 において $\tau_3$ の余裕時間がクリティカルである．また，時刻 2 においてクリティカルタスクが存在しなくなったため $\tau_3$ が $\tau_2$ にプリエンプトされて実行している．

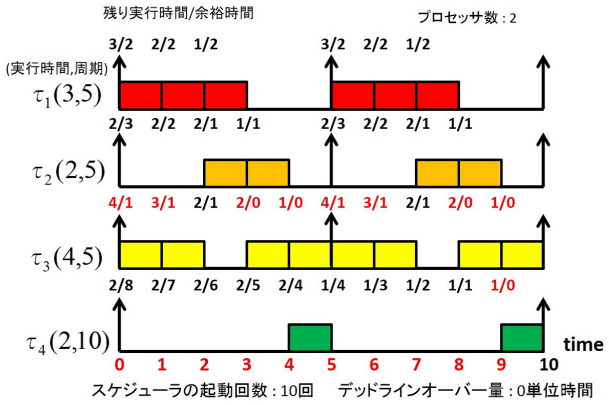


図 5　mEDCL の実行例

### 3.7 スケジューラの起動頻度とスケジュール成功率の関係

図 6, 図 7, 図 8 に EDF, EDZL, EDCL, mEDZL, mEDCL それぞれの $M = 4, 8, 16$ のときのスケジュール成功率の比較を示す．図はシステム利用率 70% から 100% の範囲で拡大したものである．それぞれの図より，スケジューラを毎単位時間起動する EDZL，mEDCL はスケジュール成功率が高く，スケジューラの起動がジョブの起動時，完了時のみの EDF，mEDZL はスケジュール成功率が低い．また，EDCL は EDF，mEDCL よりもスケジュール成功率が高いが，スケジューラの起動タイミングが EDF，mEDCL と同様であるため，EDZL，mEDCL よりはスケジュール成功率が低いという結果となった．実装面を考慮するとスケジューラを毎単位時間起動することは好ましくないため，スケジューラの起動ををジョブの起動時，完了時のみに制限しつつ高いスケジュール成功率を実現する必要がある．よって本論文ではスケジューラの起動がジョブの起動時，完了時のみであり，EDF よりもスケジュール成功率の高い EDCL を基に，EDZL に近いスケジュール成功率を実現するリアルタイムスケジューリングアルゴリズムを提案することが目的となる．

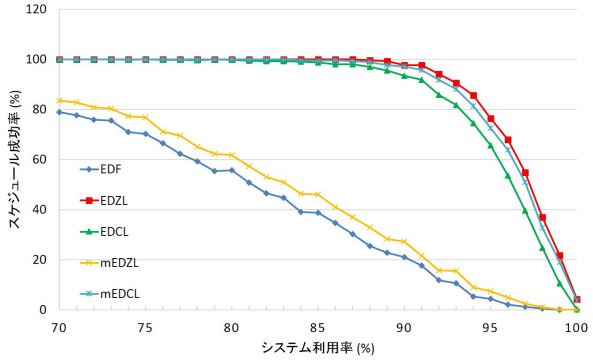


図 6　$M = 4$ のスケジュール成功率の比較

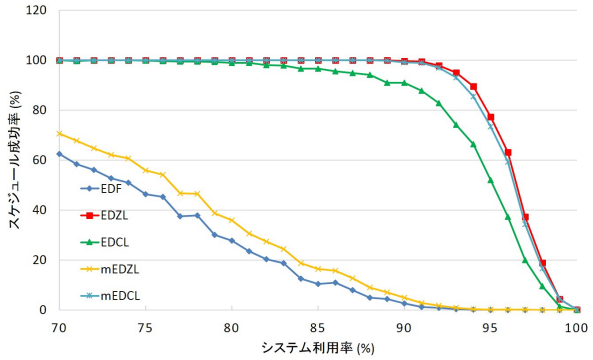


図 7　$M = 8$ のスケジュール成功率の比較

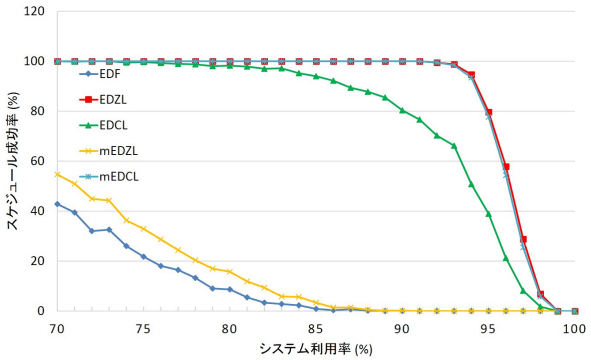


図 8　$M = 16$ スケジュール成功率の比較

## 4. 提案手法

提案手法は，余裕時間がクリティカルとなったジョブを最高優先度とし，次回にスケジューラが起動されるまでの時刻を早めるために，ある時刻 $t_s$ における高い優先度を持つ $\tau_{hp}$ のうち最小の残り実行時間 $e_{hp}$ を持つジョブに準最高優先度を与えるというアルゴリズムである．図 9 に提案手法の実行例を示す．ただし，実行するタスクセット及びプロセッサ数は $\tau = \{\tau_1 = (3,5), \tau_2 = (2,5), \tau_3 = (4,5), \tau_4 = (2,10)\}$, $M = 2$ である．図 9 より，時刻 0 において $\tau_3$ の余裕時間がクリティカルとなっているため最高優先度が与えられる．そして高い優先度を持つ $\tau_1$，$\tau_2$ のうち，最小の残り実行時間をもつ $\tau_2$ に準最高優先度が与えられ実行している．図 3 の EDCL と異なるのは，スケジューラの起動が時刻 3 から 2，時刻 8 から 7 へそれぞれ 1 単位時間分早まったことである．最小の残り実行時間を持つジョブに準最高優先度を与えることでスケジューラが起動する時刻を早め，余裕時間がクリティカルとなるジョブの発生をできるだけ早期に検出することができる．

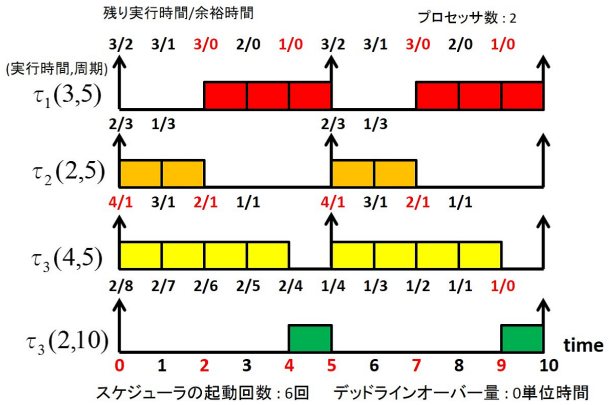


図 9　提案手法の実行例

## 5. 評価及び考察

本章ではシミュレーションによる評価と得られた結果の考察を述べる．

### 5.1 シミュレーション環境

評価はシミュレーションにより行う．シミュレーションにあたり，$U_{max}$，$U_{min}$，$U_{total}$ の 3 つのパラメータを定義する．$U_{max}$ と $U_{min}$ は与えられる各タスクセットに含まれているタスク利用率の最大値と最小値，$U_{total}$ は各タスクセットに含まれているタスク利用率の合計値である．また，$U_{total}/M$ をシステム利用率と定義する．このシステム利用率が 30% から 100% まで各々 1000 個のタスクセットを用意し，スケジュール成功率を計測する．また，$M$ は 4，8，16 の 3 通り，書くタスクの利用率の範囲を [$U_{min}$，$U_{max}$]=[0.01,1.0] とする．タスクセットは $U_{(\tau)} < U_{total}$ である限り新しいタスクを生成し，タスクセットを追加していく．新しく生成されるタスク $\tau_i$ の利用率 $U_i$ は [$U_{min}$，$U_{max}$] の範囲でランダムに決定する．タスクはハーモニックであるとし，周期 $T_i$ は [100,200,400,800,1600,3200] の中からランダムに選択する．実行時間 $C_i$ は $C_i = U_i T_i$ として得られる．スケジュール時間はタスクセットのパイパーピリオドまで，つまり $\mathrm{lcm}(T_i|\tau_i \in \tau)$ とする．

### 5.2 シミュレーション結果

本論文で示すスケジュール成功率は式 (1) で定義される値である．

$$\frac{\text{スケジュールに成功したタスクセット数}}{\text{スケジュールしたタスクセットの総数 (1000 個)}} \quad (1)$$

図 10，図 11，図 12 に $M$=4，8，16 のスケジュール成功率をそれぞれ示す．また，結果はシステム利用率 70% から 100% の範囲で拡大したものである．それぞれの図より，提案手法はスケジューラの起動をジョブの起動時，完了時のみに制限しつつ EDF，EDCL と比べ高いスケジュール成功率を実現している．これより，準最高優先度を加えスケジューラが起動する時刻を早めることで余裕時間がクリティカルとなるジョブの発生を早期に検出できる点が効果を発揮しているといえる．しかし，スケジューラを毎単位時間起動する EDZL を上回ることはなかった．これは余裕時間がクリティカルとなるジョブの検出を早めても検出のタイミングですでに余裕時間が負となるジョブが存在するためであると考えられる．よって，余裕時間がクリティカルとなったジョブが存在しない場合などの動作を改良することでより高いスケジュール成功率を実現できる可能性がある．

## 6. 結論及び今後の課題

本論文ではスケジューラの起動頻度を抑えつつ高いスケジュール成功率を実現するリアルタイムスケジューリングアルゴリズムを提案した．しかし，提案手法は準最高優先度を加えたことによるオーバーヘッドを考慮していない．また，本論分ではスケジューリングアルゴリズムの評価として必要不可欠なスケジュール可能性解析を行っていない．そして，提案手法が実用的なアルゴリズムであるか組み込みリアルタイム OS に実装し，評価を行う必要がある．これらは今後の課題とする．

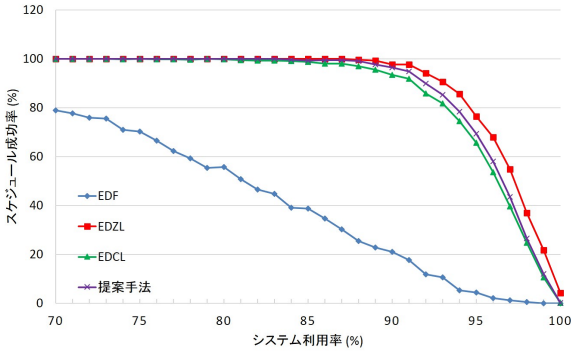


図 10　$M = 4$ のスケジュール成功率

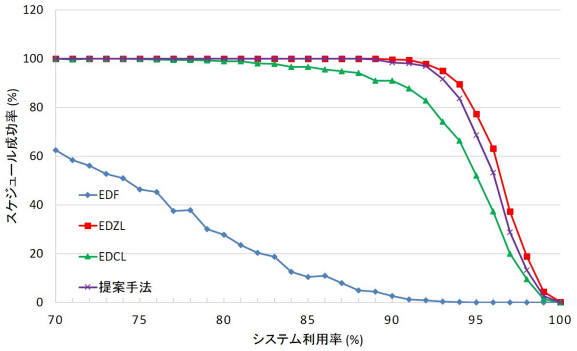


図 11　$M = 8$ のスケジュール成功率

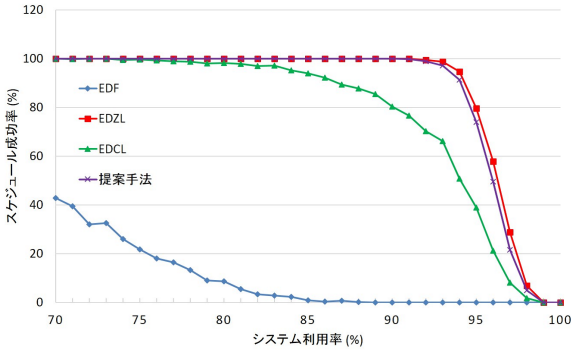


図 12　$M = 16$ のスケジュール成功率

**謝辞** 本研究の一部は JSPS 科研費 24500046 の助成を受けたものである．

## 参考文献

[1] 武田瑛，加藤真平，山崎信行，”Rate monotonic に基づくマルチプロセッサ用リアルタイムスケジューリング“, 情報処理学会論文誌コンピューティングシステム, Vol.2, No.1, pp. 64-74 (2009)

[2] C. L. Liu and J. W. Layland,“Scheduling Algorithms for Multiprogramming in a Hard- Real-Time Environment”, Journal of ACM, Vol. 20, No. 1, pp. 46-67 (1973)

[3] S. Cho, S. Lee, A. Han and K. Lin, “Efficient real-time scheduling algorithms for multiprocessor systems”, IEICE Transactions on Communications, Vol. E85-B, No. 12, pp. 2859-2867 (2002)

[4] S. Kato and N. Yamasaki, “Global EDF-based scheduling with laxity-driven priority promotion”, Journal of systems Architecture, Vol. 57, No. 5 pp. 498-517 (2011)

[5] 林竜太，兪明連，横山孝典，“EDF を基にしたスケジューリングアルゴリズムの比較分析”, 情報処理学会 第 76 回全国大会公演論文集, pp. 147-148 (2014)