平成31年度令和元年度修士論文

マルチプロセッサシステムのヒープメモリ消費量を削減するリアルタイムスケジューリング手法の提案

広島市立大学

大学院情報科学研究科システム工学専攻組込みデザイン研究室

氏名 町頭　優輝

指導教員 中田　明夫　教授

令和2年1月28日　提出

目次

1. はじめに **1**
   1. 研究背景 *. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .* 1
   2. 研究目的 *. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .* 2
   3. 研究概要 *. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .* 3
   4. 本論文の構成 *. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .* 5
2. マルチタスクシステム **6**
   1. 概要 *. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .* 6
   2. ヒープメモリ管理 *. . . . . . . . . . . . . . . . . . . . . . . . . . . . .* 7
   3. 時間制約について *. . . . . . . . . . . . . . . . . . . . . . . . . . . . .* 10
3. マルチプロセッサシステム **11**
   1. 概要 *. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .* 11
   2. システムモデル *. . . . . . . . . . . . . . . . . . . . . . . . . . . . . .* 11
   3. マルチプロセッサシステムにおけるスケジューリング *. . . . . . . . .* 12
4. **Least Memory Consumption First 15**
   1. LMCF スケジューリングの概要 *. . . . . . . . . . . . . . . . . . . . .* 15
   2. LMCF スケジューリングの理論的正当性 *. . . . . . . . . . . . . . . .* 15
      1. シングルプロセッサ環境下 *. . . . . . . . . . . . . . . . . . . .* 15
      2. マルチプロセッサ環境下 *. . . . . . . . . . . . . . . . . . . . .* 19
5. **Least Memory, remaining Computation-time, and Laxity First 23**
6. プロセッサを意図的に休ませることによるメモリ削減 **25**
   1. マルチプロセッサ環境下の問題点と改良方針 *. . . . . . . . . . . . . .* 25
   2. LMCF スケジューリングの改良 *. . . . . . . . . . . . . . . . . . . . .* 26
   3. LMCLF スケジューリングの改良 *. . . . . . . . . . . . . . . . . . . .* 28
7. メモリ削減スケジューリングと他のスケジューリングの併用 **29**
   1. 定義と理論的正当性 *. . . . . . . . . . . . . . . . . . . . . . . . . . .* 29
   2. メモリ削減スケジューリングと LLF スケジューリングの併用 *. . . . .* 30
8. 評価実験 **33**
   1. 実験方法 *. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .* 33
   2. タスクセットの生成方法 *. . . . . . . . . . . . . . . . . . . . . . . . .* 33
   3. 周期タスクに対する LLF スケジュール可能性判定 *. . . . . . . . . . .* 34
   4. 実験結果 *. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .* 35
9. 考察 **38**
10. 結論 **40**

参考文献 **41**

発表業績 **44**

謝辞 **45**

# **1** はじめに

## **1.1** 研究背景

スマートフォン，家電製品，医療機器など代表される組込みシステムは大量生産されることが多いため，製造コストの削減は重要な課題である．そのため，組込みシステムの開発目標の一つとして，メモリ消費量を削減することが挙げられる [1]．なぜなら，メモリ消費量を削減することができれば，組込みシステムのメモリ搭載量を削減でき，製造コストの削減に結びつくと考えられるからである．

リアルタイム組込みシステムは，複数の外部入力に対する応答性を向上させるため，複数のタスクを切り替えることによって並行処理するマルチタスクシステムで構成されることが多い．しかしながら，マルチタスクシステムではタスクが切り替わることによって一時停止するとき，タスクにヒープメモリが割り当てられたままとなるため，ヒープメモリ消費量が増加してしまう．従って，一般に，マルチタスクシステムのヒープメモリ消費量は，同じ機能を実現するシングルタスクシステムのヒープメモリ消費量よりも増加する傾向にある．

マルチタスクシステムのメモリ消費量を静的解析によって削減する手法が提案されている[2][3]．しかしながら，実際に市場に流通しているマルチタスクシステムで動作する製品には，数千個のタスクを扱う製品も存在する．それらの製品において，他のタスクへの切り替えが多くなるとメモリ消費量は増加する．また，タスク単体が持つ状態（ヒープメモリの割り当て箇所）数が少なくても，複数のタスクの並行処理時には各タスクの状態で全ての組み合わせを取り得るため，状態数が爆発的に増加する．よって，既存の静的解析に基づくメモリ削減手法ではそのような解析は困難である．

リアルタイムスケジューリングの研究は多く存在する[4][5][6][7][8][9]．しかしながら，メモリ消費量の削減を目的としたリアルタイムスケジューリングの研究は，我々の知る限りまだ存在しない．

そこで，我々は，マルチタスクシステムのヒープメモリ消費量を動的解析によって削減するスケジューリングアルゴリズムである，Least Memory Consumption First

(LMCF) スケジューリングを提案している [10][11][12]．LMCF スケジューリングでは，各タスクの次のステップのヒープメモリの割り当て（以下，消費メモリ増分）は予測可能[13]であると仮定する．プロセッサ数を*p*とすると，次のステップの消費メモリ増分が小さいタスクから *p* 個選択し，選択したタスクに優先度を付与する．これにより，ヒープメモリ消費量が最大となる状態（以下，最大メモリ消費状態）を回避可能なタスクセットに対しては，LMCF スケジューリングによって必ず最大メモリ消費状態を回避可能である [11]．

一方，LMCFスケジューリングでメモリ消費量の削減はできるが，マルチタスクシステムによる効果（応答性の向上など）が得られない可能性がある．そこで，LMCF スケジューリングと他のスケジューリングを組み合わせることによってメモリ消費量の削減と他の制約（時間制約など）との両立を試みる[11][12]．特に，リアルタイム組込みシステムは，消費電力の削減と性能の向上の両立を図るため，マルチプロセッサ環境下で実行されることが多い．マルチプロセッサ環境下でのリアルタイムスケジューリングアルゴリズムは，一般的に，デッドラインが最も早いタスクに最高優先度を与える Earliest Deadline First (EDF) スケジューリング [14] よりも，余裕時間の最も短いタスクに最高優先度を与えるLeast Laxity First (LLF)スケジューリングの方が，スケジュール可能性が高いことが知られている [15]．従って，マルチプロセッサリアルタイムシステムにおいてヒープメモリ消費量を削減するためには，LLFスケジューリングとLMCFスケジューリングを併用することが有効と考えられる．

## **1.2** 研究目的

LMCF スケジューリング [11][12] は，必ずしもメモリ最適なスケジュールを構築するわけでない．例えば，デッドラインや余裕時間にばらつきがあるタスクセットの場合を考える．この場合，EDF スケジューリングや LLF スケジューリングではタスクが逐次的に実行されるため，結果として，比較的メモリ消費が少なくなる可能性がある．一般に，同時起動中のタスク数が少なければ少ないほど，メモリ消費量も小さくなるため，できるだけ早くタスクを終了させることが効果的であると考えられる．したがって，残余実行時間の短いタスクをより優先する方針をとることで，メモリ消費量のさらなる削減が期待できる．また，残余実行時間が同じタスクに対しては，EDFスケジューリングやLLFスケジューリングと同様の方針でデッドライン制約を考慮し，同時起動中のタスク数を少なくすることが効果的であると考えられる．

一方で，リアルタイム組込みシステムは，高性能化（スループットの改善など）と低消費電力化の両立を図るため，マルチプロセッサ環境下で実行されることが多くなっている．しかしながら，マルチプロセッサ環境下では，一般的に，複数のタスクを並列処理するため，ヒープメモリ消費量が増加するという問題点がある．したがって，本研究では，ヒープメモリ消費量を削減するためのスケジューリングアルゴリズムを考案し，リアルタイム組込みシステムのヒープメモリ消費量を可能な限り削減することを目的とする．

## **1.3** 研究概要

本研究では，まず，消費メモリ増分だけでなく，残余実行時間と余裕時間を考慮した Least Memory, remaining Computation-time, and Laxity First（LMCLF）スケジューリングを提案する．LMCLF スケジューリングでは，(*α*× 消費メモリ増分 + 残余実行時間 × 余裕時間 )の値が小さいタスクから順に優先度を付与する．（ただし，*α* は時間とメモリの換算レートであり，設計者が任意に定める．）LMCLF スケジューリングでは，同時起動中のタスク数が減ることによって，LMCF スケジューリングよりもメモリ消費量を削減することが期待できる．

次に，意図的にタスクをプロセッサに割り当てないことでヒープメモリ消費量のさらなる削減を試みるIntentionally-Idling LMCF (II-LMCF)スケジューリングを提案する．LMCFと同様，II-LMCFスケジューリングにおいても，各タスクの次のステップの消費メモリ増分は予測可能[13]であると仮定する．II-LMCFスケジューリングでは，まず，整数 *p*′ = {1*,...,p*} において，消費メモリ増分が小さいタスクから *p*′ 個選択し，選択したタスク集合に含まれるタスクの消費メモリ増分の総和（以下，総消費メモリ増分）を比較する．その後，選択したタスク集合のうち，総消費メモリ増分が最小のタスク集合 *T* に含まれるタスク全てに最高優先度を付与する．

（以下，タスク集合に含まれる全てのタスクをスケジュールすることを，「タスク集合をスケジュールする」と呼ぶ．）しかしながら，II-LMCFスケジューリングでは利用可能なプロセッサを意図的に休ませるため，性能低下を引き起こす可能性がある．そこで，この問題を解決するために以下のような方法をとる．

1. 現在の状態において，II-LMCF の方針でスケジュールするタスク集合を選択し，そのタスク集合をスケジュールしたときの状態を探索する．
2. その状態においてもう一度 II-LMCF の方針でスケジュールするタスク集合を選択する．

この操作をプロセッサ数回または同じタスクが二度選択されるまで繰り返す．そして，このアルゴリズムが停止するまでに選択されたタスク集合の和集合をスケジュールする．これによって，プロセッサ利用率の改善を図る．このスケジューリングを，

Accelerated Intentionally-Idling LMCF (AcII-LMCF) スケジューリングと呼ぶ．ただし，このスケジューリングでは，各タスクの 2 ステップ先の消費メモリ増分は予測可能であると仮定する [13]．

また，LMCF スケジューリングを LMCLF スケジューリングに置き換えた

Intentionally-Idling LMCLF (II-LMCLF) スケジューリングおよび Accelerated

Intentionally-Idling LMCLF (AcII-LMCLF) スケジューリングを提案する．

本研究では，メモリ消費量の削減と他の制約の両立を図るため，メモリ削減スケジューリング（LMCF, LMCLF, AcII-LMCF, AcII-LMCLF）と他のスケジューリングを組み合わせる方針 [11][12] を採用する．本研究では，マルチプロセッサリアルタイムシステムにおいて，メモリ消費量の削減と時間制約の充足の両立を図るために，メモリ削減スケジューリングとLLFスケジューリングを組み合わせたLLF併用メモリ削減スケジューリングを提案する．LLF 併用メモリ削減スケジューリングでは，任意のスケジューリングサイクルで LLF スケジュール可能性判定 [15] を行い，その結果に応じて，メモリ削減スケジューリングとLLFスケジューリングを切り替える．

最後に，ランダムに生成されたタスクセットに対して，シミュレーション実験を行い，提案したメモリ削減スケジューリングと従来手法[14][15][16] との比較評価を行う．

## **1.4** 本論文の構成

本論文の構成について説明する．まず，2 章で，マルチタスクシステムの概要や特徴について説明し，3 章で，マルチプロセッサシステムの概要の説明と，システムモデルの定義を行う．次に，4 章で，マルチタスクシステムにおいてヒープメモリ消費量を削減するためのスケジューリングアルゴリズムであるLMCFスケジューリングについて説明し，5 章で，消費メモリ増分だけでなく残余実行時間と余裕時間を考慮してヒープメモリ消費量を削減を図る LMCLF スケジューリングについて説明する．その後，6 章では，プロセッサを意図的に休ませることによってメモリ削減を行うスケジューリング手法について説明し，7 章では，これまでに提案してきたメモリ削減スケジューリング（LMCF, LMCLF, AcII-LMCF, AcII-LMCLF）と他のスケジューリング（LLFスケジューリングなど）を組み合わせることによって，他の制約（時間制約など）とメモリ削減の両立を図るメモリ削減併用スケジューリングについて説明する．最後に，8 章で評価実験を行い，9 章で考察し，10 章でまとめと今後の課題について記述する．

# **2** マルチタスクシステム

本章では，マルチタスクシステムの概要，マルチタスクシステムにおけるヒープメモリ管理，時間制約の特徴について説明する．最初に，2.1節で一般的なマルチタスクシステムの動作について説明する．次に，2.2節でマルチタスクシステムのヒープメモリ管理における特徴について，2.3節ではマルチタスクシステムの時間制約における特徴について説明する．

## **2.1** 概要

マルチタスクシステムとは，リアルタイム OS （一定時間内に与えられた処理を完了させる制約を持つ OS）が提供する機能の 1 つであり，中でも，OS が複数タスクを切り替えながら処理するシステムのことを指す．

マルチタスクシステムの実行環境では，並行プログラムの手法が利用可能となり，機能をタスクとして分割することで，ソフトウェアの再利用性が向上する．通常の CPUが1つのみのコンピュータでは，ある瞬間には1つの処理しかできない．しかし，タスクの CPU 処理時間を数 10 ミリ秒程度の短い区間で区切ることで，タスク間で 1 つの CPU をタイムシェアリングすることができる．これによりユーザからは，複数のアプリケーションが同時に処理されているように見える．

マルチタスクシステムは，タスク切り替え時に余分な処理を行う必要があり，キャッシュのミスヒット率の上昇などのコストがかかる．しかし，入出力待ちなどによって，あるタスクの処理が止まっても他のタスクが処理されるため，全体としてのスループットの改善が期待できる．なお，マルチタスクシステムには，CPUの実行権限を全て OS が管理し，強制的に処理の切り替えを行うプリエンプティブマルチタスクと，処理の切り替えが個々のアプリケーションに任せられているノンプリエンプティブマルチタスクが存在する．

マルチタスクシステムの一例を図 1 に示す．図中の四角形はタスクの処理を示し，その中に各タスクが起動した回数を記入する．タスクの優先度は Task 1*>*Task

2*>*Task 3とする．上向きの矢印はリリース時刻，下向きの矢印は絶対デッドラインを示し，横軸は時間を示す．

1

1

1

2

2

2

3

Task 1

Task 2

Task 3

図 1: マルチタスクシステムの一例

## **2.2** ヒープメモリ管理

各タスクに与えられた優先度によりタスクの切り替えを行うマルチタスクスケジューリングが存在する．ここでは，複数のタスクを並行処理するため，優先度の高い他のタスクへの切り替えが発生すると，切り替えられたほうのタスクは，作業用に確保したメモリを保持したまま一時停止する．その様子を図 2に示す．ただし，図 2 のタスクの優先度は図 1 と同様，Task 1*>*Task 2*>*Task 3 とする．

図 2 より，Task 3 から Task 2 へ切り替わることによって，Task 3 はメモリを確保したまま一時停止する．同様に，Task 2 から Task 1 へ切り替わることによって，

Task 2 はメモリを確保したまま一時停止する．このとき，Task 2 と Task 3 の 2 つのタスクで確保された分のメモリを保持し続けなければならない．また，タスク単体が持つ状態数が少なくても，複数のタスクをマルチタスクシステムで扱うと，コンパイル時にタスクの状態数全てを組み合わせたものを生成するため，状態数が爆発的に増加する．

この例として，5 つの状態数を持つ 2 つのタスクを図 3，図 4 に示す．図 3，図 4 に示す 2 つのタスクを用いて構成したマルチタスクシステムの状態遷移図を図 5 に示す．図 3，図 4 は，簡単な C 言語プログラムを状態遷移図に変換した図である．状態遷移図の円は，状態を表す．各状態に記載されている番号は，タスクが何番目

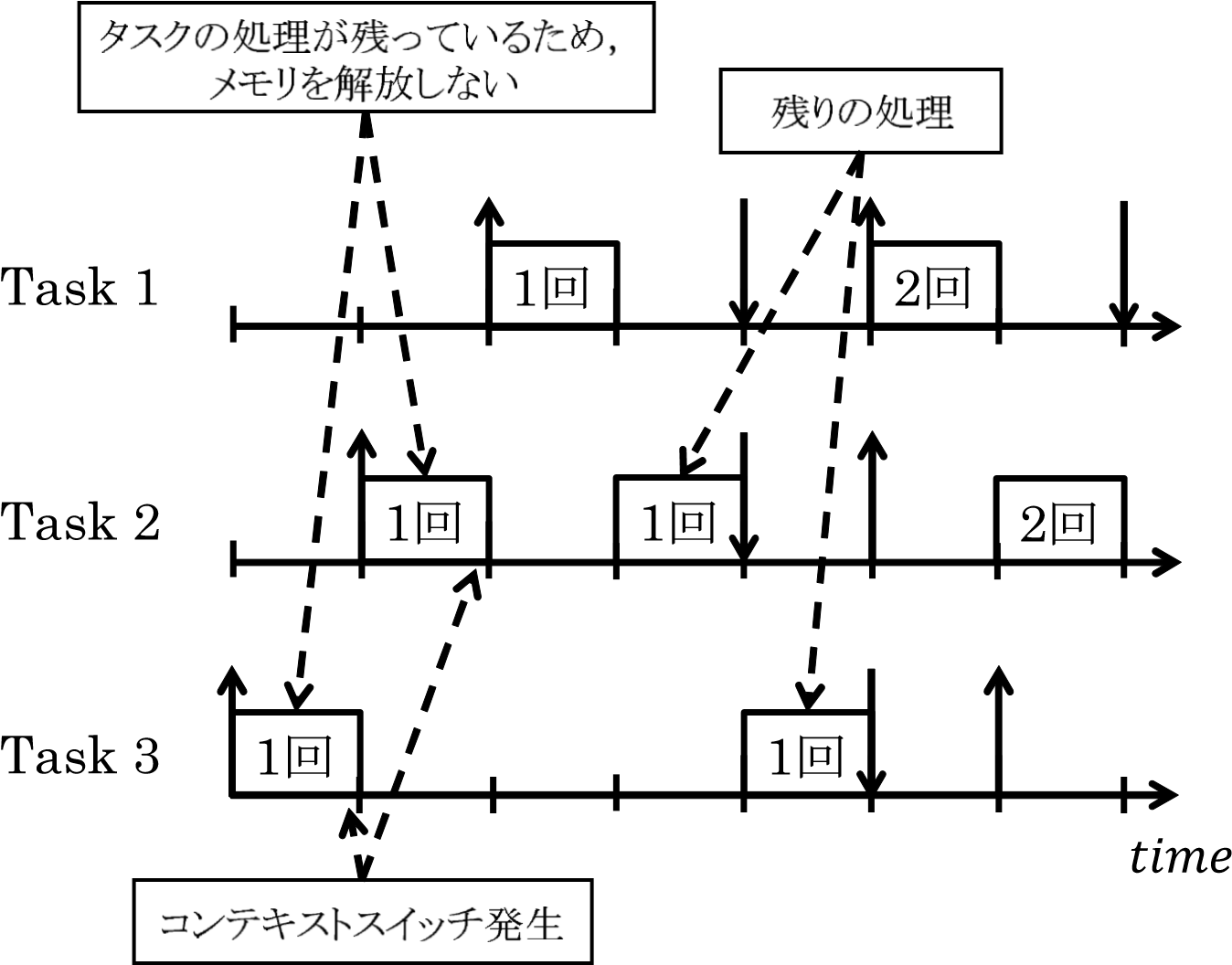


図 2: コンテキストスイッチ発生時における各タスクのメモリ消費状況

の状態にあるかを指す．これらの遷移図では，各状態で確保している変数を記載する．また，図中の最初の状態 state1 は初期状態，最後の状態 state5 は終了状態を表している．図 5の矢印は，左向きはTask 1，右向きはTask 2がスケジュールされた場合の状態遷移を表す．各状態に記載されている番号は，各タスクが何番目の状態にあるかを指す．左側の数字は Task 1 の状態，右側は Task 2 の状態とする．

並行処理を行う際には，全てのタスクの状態を組み合わせるため，図 5 のように状態数が増加する．状態を組み合わせた中には，各タスクで確保している変数を同時に保持している状態が存在する．変数を多く確保している状態は，メモリ消費量が大きい動作を行うと考えられる．図 5では，状態(3*,*3)と状態(3*,*4)が最も多く変数を確保しているため，メモリ消費量が大きい動作を行うと考えられる．一般的に，マルチタスクシステムは様々なスケジューリング方針でタスクを切り替える．よって，図 5の状態(3*,*3)と状態(3*,*4)のように，メモリ消費量の大きい状態を通過する可能性を持つ．

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Task 1{ int x, y; |  | //state1 | | x = 1; |  | //state2 | | y = x + 1; |  | //state3 | | printf(“%d”, y); |  | //state4 | | } |  | //state5 | | 1  2  3  5  4 |

x

x , y

y

図 3: Task 1 の状態遷移

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Task 2{ int s, t; |  | //state1 | | s = 3; |  | //state2 | | t = 2; |  | //state3 | | s = s \* t; |  | //state4 | | } |  | //state5 | | 1  2  3  5  4 |

s

s , t

s , t

図 4: Task 2 の状態遷移

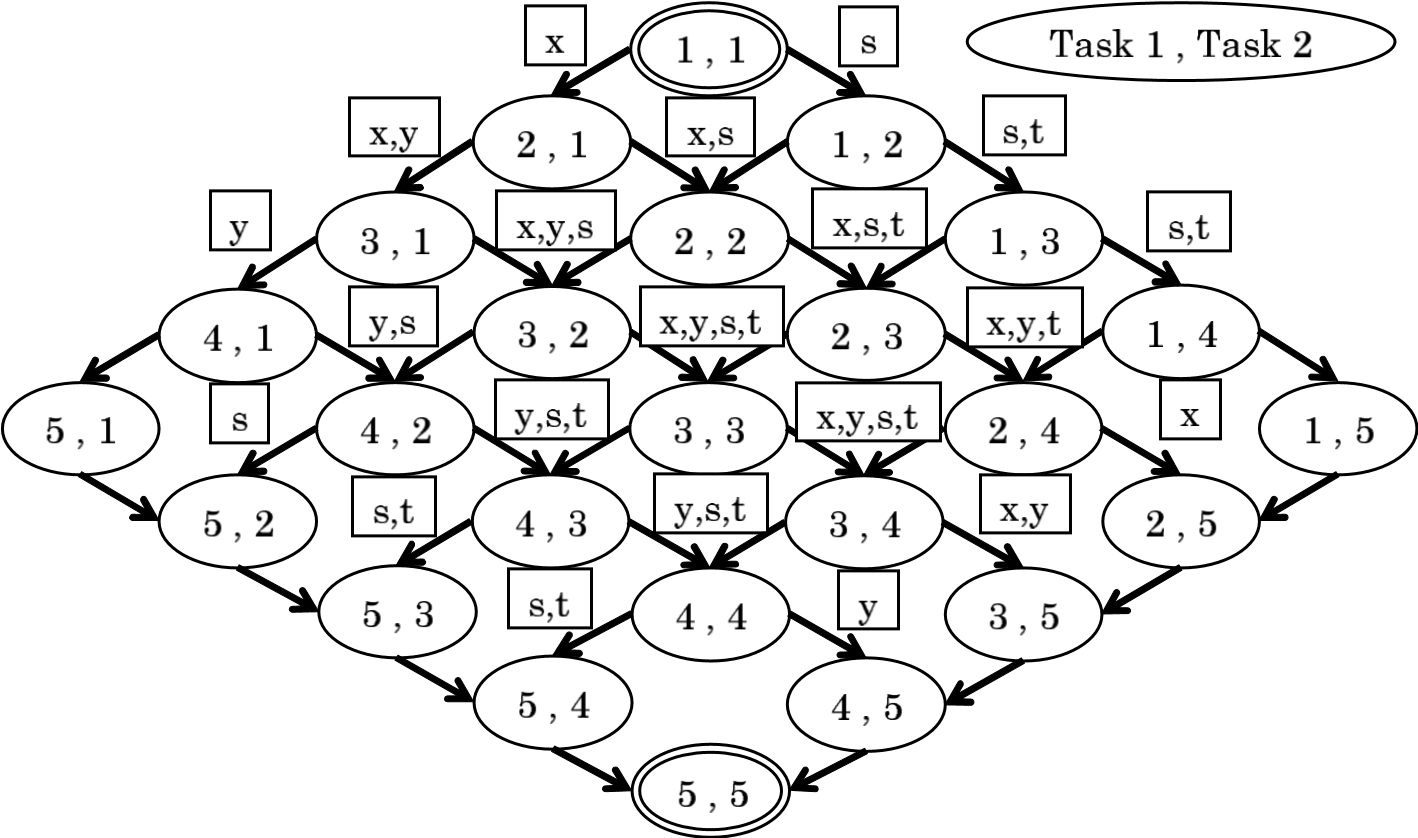


図 5: Task 1 と Task 2 の状態遷移

## **2.3** 時間制約について

システムの中には，処理がデッドラインまでに終了しなかった場合，システムにとって致命的なダメージを受けるものがある．そのため，システムの設計・開発の際に，システムの時間制約を満たすという保証が求められることがある．そこで，システムの時間制約が満たされることを保証するための技術が必要となる．この技術には，各タスクの最悪実行時間を導出するための技術と，複数のタスクに関して，処理の順番を考えるリアルタイムスケジューリング理論がある．

各タスクの最悪実行時間の導出には，あらゆる入力に対する処理の実行時間の導出が必要である．しかし，これはキャッシュや分岐などの機構によって，処理の実行時間の導出が困難となっている．現実的には，処理の実行時間を複数回測定し，結果に安全係数を見込んで，最悪実行時間を定める方法を用いる．

リアルタイムスケジューリング理論とは，複数のタスクがデッドラインを持つ場合に，それらをスケジューリングする方法と，各タスクが時間制約を満たすかを数学的に証明するための方法を与えるものである．

最後にマルチタスクシステムの特徴をまとめる．マルチタスクシステムにおける各タスクは，タスクが切り替わることで待ち状態となり，メモリを確保し続けてしまう．それにより，全体のメモリ消費量が増加してしまう．また，複数のタスクに時間制約がある場合に，各タスクが時間制約を満たすかどうかを数学的に証明する方法が存在することがマルチタスクシステムの特徴であるといえる．

# **3** マルチプロセッサシステム

本章では，マルチプロセッサシステムの概要，マルチプロセッサシステムにおけるスケジューリングについて説明する．3.1節では，一般的なマルチプロセッサシステムの特徴を説明し，3.2 節では，システムモデルを定義し，3.3 節では，マルチプロセッサシステムにおけるスケジューリングについて説明する．

## **3.1** 概要

マルチプロセッサシステムとは，コンピュータに複数の CPU を搭載して一連の処理を行うシステムのことである．マルチプロセッサシステムは，CPU ごとに異なる処理を並行できるという特徴がある．マルチプロセッサシステムには，複数の CPU で 1 つの主記憶を共有する密結合と，1 つの CPU につき 1 つの主記憶で構成される疎結合がある．また，各CPUが対等でどのCPUでも同じ処理を行えるものを対称型マルチプロセッサ（SMP：Symmetric Multiprocessing）といい，CPUごとに役割が決められている（異なる機能を持つ）ものを非対称型プロセッサ（ASMP：

Asymmetric Multiprocessing）という．

## **3.2** システムモデル

本研究で取り扱うシステムモデルは，[17]で定義されているものと同様である．最

小リリース間隔を *Ti*，最悪実行時間を *Ci*（Worst-case Execution Time, WCET），相対デッドラインを *Di* とし，このモデルにおけるタスク *τi* = (*Ti,Ci,Di*) はタスク

セット *TS* に含まれる（*τi* ∈ *TS*）．タスク *τi* は一連のジョブを呼び出し，各ジョブは，前のジョブと少なくとも*Ti* 時間単位だけ間隔をとる．また，タスクのひとつのジョブは並列に実行されないと仮定する．

タスク*τi* の時刻*t*における相対デッドラインと残余実行時間をそれぞれ*Di*(*t*)，*Ci*(*t*) とし，タスク *τi* の時刻 *t* における余裕時間を *Li*(*t*) とし，式 (1) から計算する．

*Li*(*t*) = *Di*(*t*) − *Ci*(*t*) (1)

タスク数の合計は *n* とし，システムの利用率を *Usys* = ∑*τi*∈*TS Ci/Ti* とする．また， *p* 個のプロセッサが利用可能であるとし，プロセッサの性能は同等のものとする．タスクセットを以下のように定義する．

#### 定義 **1.** タスクセット

*n*(≥ 1) を任意の自然数とする．任意の *i* ∈ {1*,...,n*} に対して，タスク *τi* は 1 ステップ目から *ei*（1 以上の任意の自然数）ステップ目まで状態を変化させる有限状態機械であるとし，*τi* の *j*(1 ≤ *j* ≤ *ei*) ステップ目の状態を *sij* とする．

タスクの集合（以下，タスクセットと呼ぶ）を *TS* = {*τ*1*,τ*2*,...,τn*} とする．*i* ∈

{1*,...,n*} に対して，*ji*(1 ≤ *ji* ≤ *ei*) を自然数とし，*TS* の状態は各タスクの状態の

直積(*s*1*j*1*,s*2*j*2*,s*3*j*3*,...,snjn*)であると定義する．*TS* の初期状態はすべてのタスクの初期状態（1 ステップ目の状態 *si*1）の直積，*TS* の最終状態はすべてのタスクの最終状態（*ei* ステップ目の状態 *siei*）の直積と定義する．

時刻 *t* における *τi* の状態が *sij* の時，状態 *sij* の *τi* の相対デッドラインと残余実行時間をそれぞれ，*D*(*sij*) と *C*(*sij*) とする．

*Ti* に対して，∑1≤*j*≤*ei*−1 *m*(*sij*) = 0を満たすような各状態*sij* への整数値*m*(*sij*)の割り当てを状態 *sij* における「消費メモリ増分」と呼ぶ．

## **3.3** マルチプロセッサシステムにおけるスケジューリング

マルチプロセッサシステムにおいて，EDF スケジューリング [14] よりも LLF スケジューリングの方がより適している [15]．

例えば，以下のような 3 つのタスクのスケジューリング問題を考える．タスクの

パラメータは，*τ*1 = (3*,*3*,*3)，*τ*2 = (2*,*1*,*2)，*τ*3 = (2*,*1*,*2) とする．このスケジューリング問題では 2 つのプロセッサが利用可能であると仮定し，プロセッサの性能は同等のものとする．このスケジューリング問題に EDF スケジューリングを適用したときの例と LLF スケジューリングを適用したときの例をそれぞれ図 6 と図 7 に示す．図中の四角はタスクの処理を表し，四角の中の数字はプロセッサの識別子を示す．上矢印がタスクのリリース時刻，下矢印が絶対デッドライン，横軸を時間とする．

2

1

2

1

1

2

1

3

4

2

2

1

3

1

1

2

2

4

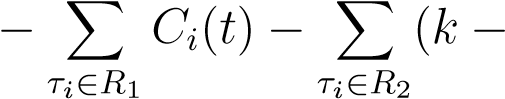
1 2 1 2

図 6: EDF スケジューリング 図 7: LLF スケジューリング

図 6から分かるように，EDFスケジューリングでは，*τ*2 と*τ*3 が優先的にプロセッサに割り当てれ，結果として*τ*1 がデッドラインミスする．しかしながら，このスケジューリング問題は LLF スケジューリングによって解決できる（図 7）．LLF スケジューリングでは，式 (1) から各タスクの余裕時間 *Li* を計算する．このとき，*τ*1， *τ*2，*τ*3 の余裕時間はそれぞれ，*L*1 = 0，*L*2 = 1，*L*3 = 1 となる．その結果，LLF スケジューリングでは，*τ*1 と*τ*2 が優先的にプロセッサに割り当てられ，*τ*1 のデッドラインミスを回避でき，このスケジューリング問題を解決できる．このことから，マルチプロセッサ環境下では EDF スケジューリングよりも LLF スケジューリングのほうがより適しているといえる．

時刻 *t* で，LLF スケジューリングがデッドライン制約を満たすかどうかについては，プロセッサの余剰計算力を計算することで判定することができる[15]．まず，*n* 個のタスクについて，タスク *i*(*i* = 1*,*2*,...,n*) の余裕時間 *Li*(*t*) を式 (1) から計算する．次に，整数*k*(*>* 0)を用いて，*R*1(*k*) = {*τi* | *Di* ≤ *k*}*,R*2(*k*) = {*τi* | *Li* ≤ *k*∩*Di* ≥ *k*}*,R*3(*k*) = {*τi* | *Li* ≥ *k*} の 3 つの集合にタスクを分類する．*R*1(*k*) は，*k* 単位時間後までに処理を終了しなければならないタスクの集合，*R*2(*k*)は，*k* 単位時間後に余裕がなくなる（処理を終了させる必要はない）タスクの集合，*R*3(*k*)は，*k* 単位時間後もまだ余裕がある（*k* 単位時間処理をしなくてよい）タスクの集合である．プロセッサ数を *p* とすると，プロセッサの余業計算力 *F*(*k,t*) は式 (2) で求められ，すべ

ての整数 *k*(*>* 0) で *F*(*k,t*) がすべて 0 以上であれば，時刻 *t* で LLF スケジューリング可能となる [15]．

*F*(*k,t*) = *k* × *p*  *Li*(*t*)) (2)

# **4 Least Memory Consumption First**

この章では，マルチタスクシステムにおいてヒープメモリ消費量を削減するためのスケジューリングアルゴリズムであるLeast Memory Consumption First (LMCF) スケジューリング[11][12] について記述する．まず，4.1節で，LMCFスケジューリングの概要について記述する．その後，4.2 節で，LMCF スケジューリングの定義し，LMCF スケジューリングがメモリ削減できることを理論的に証明する．

**4.1 LMCF**スケジューリングの概要

LMCF スケジューリングでは，スケジューリングサイクルごとに各タスクの次のステップの消費メモリ増分を比較し，消費メモリ増分が最小のタスクに最高優先度を与える．

以下に，2 つのタスクをシングルプロセッサ環境下で並行処理したときの例を示す．図 8 は，図中の Task 1 と Task 2 をマルチタスクで実行した時の，メモリ消費量の変化の全状態を示している．図中の左下向きの矢印はTask 1が，右下向きの矢印はTask 2がスケジュールされた時の状態遷移である．図 8のマルチタスクシステムが任意の OS スケジューラで処理されていると仮定すると，初期状態から最終状態までの経路上で，最大メモリ消費状態（メモリ消費量が 9 となる状態）を通る経路を構築する可能性がある．一方，LMCF スケジューリングによる状態遷移を図 9 に示す．図 9 から，LMCF スケジューリングの，最悪メモリ消費量（スケジュール構築時の総メモリ消費量の最大値）は 6 となり，LMCF スケジューリングが最大メモリ消費状態を回避していることが分かる．

**4.2 LMCF**スケジューリングの理論的正当性

### **4.2.1** シングルプロセッサ環境下

シングルプロセッサ環境下のLMCFスケジューリングを以下のように形式的に定義する．

#### 定義 **2.** シングルプロセッサ環境下の**LMCF**スケジューリング

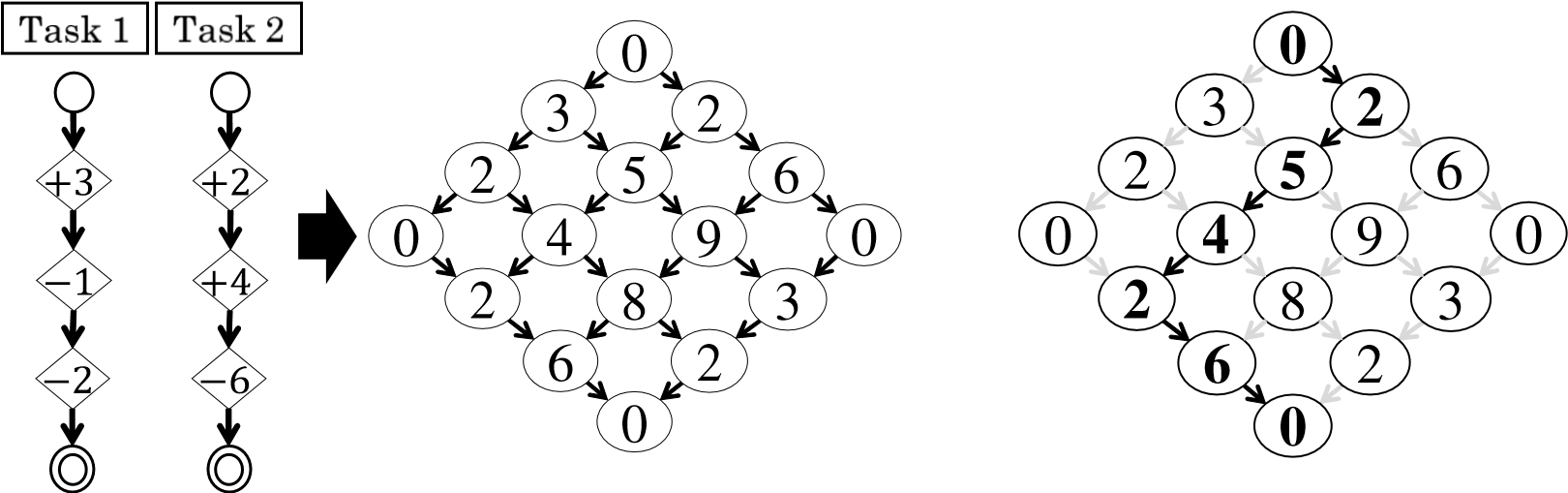


図 8: 各状態における総メモリ消費量 図 9: LMCF スケジュール

*TS* の任意の状態 (*s*1*j*1*,...,siji,...,snjn*) において，*Ti* がスケジュールされたときの *TS* の次状態を (*s*1*j*1*,...,si*(*ji*+1)*,...,snjn*) と定義する．ある *Ti* が存在し，状態

(*s*1*j*1*,...,siji,...,snjn*) に対して，*Ti* をスケジュールした次状態となっている状態

(*s*1*j*1*,...,si*(*ji*+1)*,...,snjn*) を，単に状態 (*s*1*j*1*,...,siji,...,snjn*) の次状態と呼ぶ．

*TS* のスケジュールとは，*TS* の初期状態に始まり，最終状態で終了し，初期状態

以外が直前の状態の次状態であるような有限シーケンスであると定義する．

*TS* の任意の状態 (*s*1*j*1*,...,siji,...,snjn*) における「総メモリ消費量」

*M*(*s*1*j*1*,...,siji,...,snjn*) を以下のように帰納的に定義する．

*M*(*s*11*,...,si*1*,...,sn*1) = 0 （初期状態）

*M*(*s*1*j*1*,...,si*(*ji*+1)*,...,snjn*)

= *M*(*s*1*j*1*,...,siji,...,snjn*) + *m*(*siji*)

*TS* の任意の状態(*s*1*j*1*,...,siji,...,snjn*)の「LMCFスケジューリングによる次状態」とは式 (3) となる *Ti* がスケジュールされた時の次状態であると定義する．

*m*(*siji*) = min *m*(*skjk*) (3)

1≤*k*≤*n*

「*TS* の LMCF スケジュール」とは，初期状態以外の任意の状態が直前の状態の LMCF スケジューリングによる次状態となっている *TS* のスケジュールであると定義する．

以下の定理は任意のLMCFスケジュールが総メモリ消費量上位の状態を回避できることを示す．

定理 **1.** 任意の正の整数 *h*(≥ 1) に関して，総メモリ消費量上位 *h* 位以内のいずれの状態も通らないスケジュールが存在するならば，任意のLMCFスケジュールは総メモリ消費量上位 *h* 位以内のいずれの状態も通らない．

*Proof.* 総メモリ消費量上位 *h* 位までのいずれかの状態を通る LMCF スケジュールが存在すると仮定し，矛盾を導く．

総メモリ消費量上位 *h* 位までのいずれかの状態を通る LMCF スケジュールを *SL*，総メモリ消費量上位 *h* 位までのいずれの状態も通らないスケジュールを *SA* とする．仮定より，*SL* が総メモリ消費量上位*h*位までのいずれかの状態を通ることに注意すると，*SL* には *SA* のどの状態よりも総メモリ消費量が大きい状態が存在する．

*SA* に関して次状態がLMCFに従っていない状態(*s*1*j*1*,...,siji,...,snjn*)を初期状態から探索する．その後，次状態をLMCFに従うように組み替え，それ以降は全てのタスクのスケジュールを後方に1ステップずつずらしたスケジュール*SA*′ を構築す

る．このとき，状態 (*s*1*j*1*,...,siji,...,snjn*) において *SA*′ では *τu* が，*SA* では *τv* がスケジュールされたとすると，式 (3)より*m*(*suju*) ≤ *m*(*svjv*)であるから，次状態の総メモリ消費量の関係は以下のようになる．

*M*(*s*1*j*1*,...,su*(*ju*+1)*,...,snjn*)

= *M*(*s*1*j*1*,...,suju,...,snjn*) + *m*(*suju*)

≤ *M*(*s*1*j*1*,...,svjv,...,snjn*) + *m*(*svjv*)

= *M*(*s*1*j*1*,...,sv*(*jv*+1)*,...,snjn*)

この操作を最終状態に到達するまで繰り返し，最終状態に到達したときに構成されたスケジュールを *Snew* とする．（ここで，LMCF スケジュールへの組み換えを初期状態から最終状態まで行ったため，*Snew* は LMCF スケジュールとなる．）このとき，前述の議論より，*Snew* の任意の状態における総メモリ消費量は対応する *SA* の状態の総メモリ消費量以下となる．さらに，*SL* は LMCF スケジュールであるため前述の議論と同様，*SL* の任意の状態の総メモリ消費量は対応する *SA* の状態の総メモリ消費量以下となる．以上のことから，*SL* には *SA* のどの状態よりも総メモリ消費量が大きい状態が存在するという仮定に矛盾する．

3

3

0

0

0

0

0

0

図 10: 総メモリ消費量上位 3*,*4 位が存在しないタスクセット

3

3

3

3

3

3

0

0

図 11: 総メモリ消費量上位 3 位を避けることができないタスクセット

よって，総メモリ消費量上位 *h* 位以内のいずれの状態も通らないスケジュールが存在するならば，任意のLMCFスケジュールは総メモリ消費量上位*h*位以内のいずれの状態も通らない．

タスクセットにタスクが *n* 個存在する時を考える．このとき，最悪の場合で，*n* 通りの次状態が存在する．このケースにおいて，LMCF スケジューリングでは，消費メモリ増分が*n*位のタスクをスケジュールするため，総メモリ消費量上位(*n*−1) 位までの状態を回避できると考えられる．さらに，タスク数 *n* のタスクセット *TS* における次状態の消費メモリ増分は，ほとんどの場合で上位 *n* 位まで存在することから，LMCF スケジューリングは多くのタスクセットでメモリ削減の効果が期待できる．

しかしながら，総メモリ消費量上位 (*n* − 1) 位までの状態を回避できないタスクセットも存在する．そのタスクセットの例を，図 10と図 11に示す．ただし，図 10 と図 11 のタスクセットの例は現実的にあまり扱うことがなく，LMCF スケジューリングの有効性への影響は少ないと考えられる．

図 10 のタスクセットで LMCF スケジュールを構築すると，LMCF スケジュールのメモリ時系列変化は，0 → 3 → 0 となり，定理 1 によれば，総メモリ消費量上位 (4 − 1) = 3 位までの状態を回避する．しかしながら，図 10 のタスクセットにおける総メモリ消費量上位1位は3，総メモリ消費量上位2位は0となり，総メモリ消費量上位 4 位までの状態が存在しないため，LMCF スケジュールが総メモリ消費量上位 3 位までの状態を回避することはできない．

図 11 のタスクセットで LMCF スケジュールを構築すると，LMCF スケジュールのメモリ時系列変化は，0 → 3 → 0 → 3 → 0 → 3 → 0 となり，定理 1によれば，総メモリ消費量上位 (4 − 1) = 3 位までの状態を回避する．図 11 のタスクセットにおける総メモリ消費量上位1位は9，総メモリ消費量上位2位は6，総メモリ消費量上位3位は3，総メモリ消費量上位4位は0となる．しかしながら，LMCFスケジュールが総メモリ消費量上位 3 位までの状態を回避できていないことが分かる．

### **4.2.2** マルチプロセッサ環境下

マルチプロセッサ環境下のLMCFスケジューリングを以下のように形式的に定義する．

#### 定義 **3.** マルチプロセッサ環境下の**LMCF**スケジューリング

プロセッサ数を*p*(≥ 2)とし，空いている利用可能なプロセッサには必ずタスクを割り当てると仮定する．*p*プロセッサ環境において，*n*個のタスク群から*p*個のタスクをスケジュールする場合，任意のスケジュールされるタスク群の集合は，|*I*| = *p* であるような添え字の集合 *I* ⊆ {1*,...,n*} に対して，*TSI* = {*Ti*|*i* ∈ *I*} と定義できる．一般に，*TSI* には，*n*C*p* 通りの選び方がある．

*TSI* の全てのタスクをスケジュールすることを「*TSI* をスケジュールする」と呼ぶ．*TS* の任意の状態(*s*1*j*1*,...,snjn*)に対して，*TSI* をスケジュールした次状態とは

(*s*1*j*1*I,...,snjnI*) であると定義する．ただし，

*jiI* def= { *ii* ∈ *j* + 1 *if i I*

*j otherwise.*

とする．ある *TSI* が存在し，状態 (*s*1*j*1*,...,snjn*) に対して，*TSI* をスケジュールした次状態となっている状態を，単に状態 (*s*1*j*1*,...,snjn*) の次状態と呼ぶ．

*TS* の任意の状態(*s*1*j*1*,...,snjn*)および添え字集合*I* に対する「総消費メモリ増分」を *m*((*s*1*j*1*,...,snjn*)*,I*) = ∑*i*∈*I m*(*siji*) と定義する*.*

*TS* の任意の状態(*s*1*j*1*,...,snjn*) に対して，「総メモリ消費量」*M*(*s*1*j*1*,...,snjn*) を以下のように帰納的に定義する．

*M*(*s*11*,...,sn*1) = 0 (初期状態)

*M*(*s*1*j*1*I,...,snjnI*)

= *M*(*s*1*j*1*,...,snjn*) + *m*((*s*1*j*1*,...,snjn*)*,I*)

*TS* の任意の状態 (*s*1*j*1*,...,snjn*) の「LMCF スケジューリングによる次状態」とは，式 (4) となる *TSI* をスケジュールした次状態であると定義する．

*m*((*s*1*j*1*,...,snjn*)*,I*)

= min *m*((*s*1*j*1*,...,snjn*)*,I*′) (4)

*I*′⊆{1*,...,n*}∧|*I*′|=*p*

マルチプロセッサ環境下のスケジュールで到達可能な状態の集合は，一般に，シングルプロセッサ環境下のスケジュールで到達可能な状態の集合の部分集合である．従って，LMCF スケジューリングはシングルプロセッサ環境下と同じように，消費メモリ増分の小さいタスク群の部分集合 *TSI*（ただし，|*I*| = *p*）を選択することによって，*p* プロセッサ環境下に拡張することができる．

一般的に，定理 1 同様，任意の正の整数 *h*(≥ 1) に対して，総メモリ消費量上位 *h* 位以内のいずれの状態も通らないスケジュールが存在するならば，任意の *p* プロセッサ環境下のLMCFスケジュールは総メモリ消費量上位*h*位以内のいずれの状態も通らない．

定理 **2.** 任意の正の整数*h*(≥ 1)に関して，*p*プロセッサ環境下において，総メモリ消費量上位 *h* 位以内のいずれの状態も通らないスケジュールが存在するならば，任意の LMCF スケジュールは総メモリ消費量上位 *h* 位以内のいずれの状態も通らない． *Proof.* 総メモリ消費量上位 *h* 位以内の任意の状態を通る LMCF スケジュールが存在すると仮定し，矛盾を導く．

総メモリ消費量上位 *h* 位以内のいずれかの状態を通る LMCF スケジュールを *SL*，総メモリ消費量上位 *h* 位以内のいずれの状態も通らないスケジュールを *SA* とする．仮定より，*SL* が総メモリ消費量上位*h*位以内のいずれかの状態を通ることに注意すると，*SL* には *SA* のどの状態よりも総メモリ消費量が大きい状態が存在する．

定理 1同様，*SA* に関して次状態がLMCFに従っていない状態(*s*1*j*1*,...,snjn*)を初期状態から探索する．その後，次状態をLMCFに従うように組み替え，それ以降は全てのタスクのスケジュールを後方に1ステップずつずらしたスケジュール*SA*′ を構築する．このとき，状態(*s*1*j*1*,...,snjn*)において*SA*′ では*TSU* が，*SA* では*TSV* がスケ

ジュールされたとすると，式 (4) より *m*((*s*1*j*1*,...,snjn*)*,U*) ≤ *m*((*s*1*j*1*,...,snjn*)*,V* ) であるから，次状態の総メモリ消費量の関係は以下のようになる．

*M*(*s*1*j*1*U,...,snjnU*)

= *M*(*s*1*j*1*,...,snjn*) + *m*((*s*1*j*1*,...,snjn*)*,U*)

≤ *M*(*s*1*j*1*,...,snjn*) + *m*((*s*1*j*1*,...,snjn*)*,V* )

= *M*(*s*1*j*1*V ,...,snjnV* )

この操作を最終状態に到達するまで繰り返すことで，定理 1 と同様に証明できる．

タスクセットにタスクが *n* 個存在する時を考えると，最悪の場合で，*n*C*p* 通りの次状態が存在する．このケースにおいて，LMCF スケジューリングでは，総メモリ

1

1

1

1

0

0

0

0

図 12: 総メモリ消費量上位 4*,*5*,*6 位が存在しないタスクセット

1

1

1

1

1

1

0

0

図 13: 総メモリ消費量上位 3 位を避けることができないタスクセット

消費増分が *n*C*p* 位のタスクをスケジュールするため，総メモリ消費量上位(*n*C*p* −1) 位までの状態を回避できると考えられる．したがって，以降は *n* を *n*C*p* と置き換えて，シングルプロセッサと同様に，LMCF スケジューリングによってメモリ削減できないタスクセットについて議論する．特に，*p* = 1 のとき (*n*C1 − 1) = (*n* − 1) となるため，*n*C*p* へ置き換えは自然な拡張となっている．

総メモリ消費量上位 (*n*C*p* − 1) 位までの状態を回避できないタスクセットの例を，図 12と図 13に示す．ただし，シングルプロセッサ同様，図 12と図 13のタスクセットの例は現実的にあまり扱うことがなく，LMCF スケジューリングの有効性への影響は少ないと考えられる．

図 12 のタスクセットで LMCF スケジュールを構築すると，LMCF スケジュールのメモリ時系列変化は，0 → 0 → 2 → 0 となり，定理 2 によれば，総メモリ消費量

上位(4C2 −1) = 5位までの状態を回避する．しかしながら，図 10のタスクセットにおける総メモリ消費量上位1位は2，総メモリ消費量上位2位は1，総メモリ消費量上位3位は0となり，総メモリ消費量上位6位までの状態が存在しないため，LMCF スケジュールが総メモリ消費量上位 5 位までの状態を回避することはできない．

図 13 のタスクセットで LMCF スケジュールを構築すると，LMCF スケジュールのメモリ時系列変化は，0 → 2 → 0 → 1 → 0となり，定理 2によれば，総メモリ消

費量上位 (4C3 − 1) = 3 位までの状態を回避する．図 11 のタスクセットにおける総メモリ消費量上位 1 位は 3，総メモリ消費量上位 2 位は 2，総メモリ消費量上位 3 位は 1，総メモリ消費量上位 4 位は 0 となる．しかしながら，LMCF スケジュールが総メモリ消費量上位 3 位までの状態を回避できていない．

# **5 Least Memory, remaining Computation-time, and Laxity First**

この章では，消費メモリ増分だけでなく，残余実行時間と余裕時間を考慮した新しいスケジューリングアルゴリズムを提案する．

図 14の3つのタスクのスケジューリング問題を考える．*τ*1，*τ*2，*τ*3 のパラメータ

はそれぞれ，*τ*1 = (24*,*4*,*24)，*τ*2 = (19*,*6*,*19)，*τ*3 = (18*,*4*,*18) とする．この時，各タスクの余裕時間はそれぞれ *L*1 = 20，*L*2 = 13，*L*3 = 14 となる．図 14 のタスクセットをLMCFスケジューリングによってスケジュールした時の総メモリ消費量の時系列変化を図 15に示す．同様に，LLFスケジューリングによってスケジュールした時の総メモリ消費量の時系列変化を図 15 に示す．LMCF スケジューリングでは，最悪メモリ消費量が25であるのに対して，LLFスケジューリングでは，最悪メモリ消費量が 20 である．このことから，LMCF スケジューリングが LLF スケジューリングよりもメモリ削減できない場合が存在する．これは，残余実行時間が短いタスクがメモリを確保したまま一時停止してしまうことが原因であると考えられる．

したがって，消費メモリ増分だけでなく，残余実行時間と余裕時間も考慮したスケジューリング手法を提案する．このスケジューリング手法を，Least Memory, remaining

Computation-time, and Laxity First (LMCLF) スケジューリングと呼ぶ．LMCLF スケジューリングでは，次のスケジューリングサイクルにおいて，*θi*(*siji*) を式 (5) から計算し，*θi*(*siji*) の値が小さいタスクから順に優先度を付与する．

*θi*(*siji*) = *α* × *m*(*siji*) + *Ci*(*siji*) × *Li*(*siji*) (*α >* 0) (5)

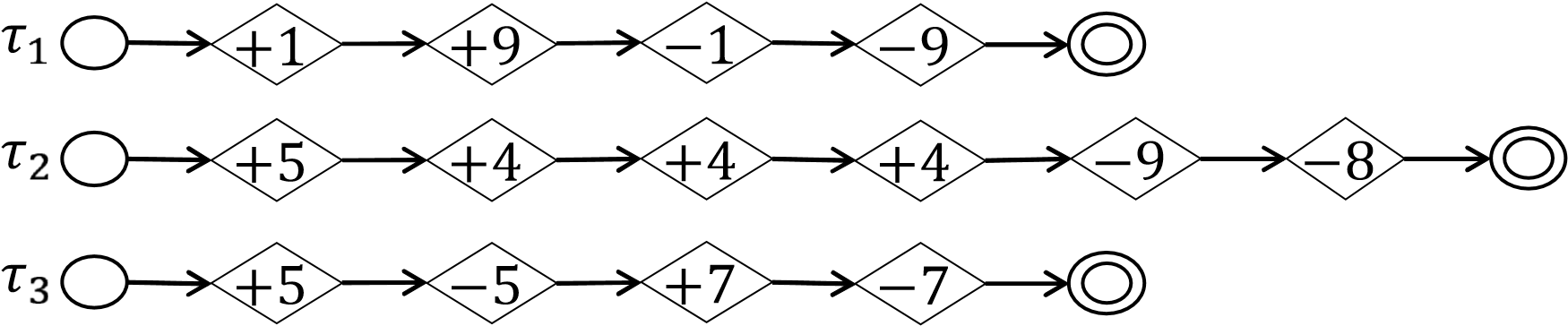


図 14: 3 つのタスクのスケジューリング問題

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| .  1  )  LMCF  (  2  .  LLF  )  (  .  3  4.  5  .  6  . | (LMCLF) | 0615142591090 0109201791090  0109201791090 066171427170  0109142591090  061016059131780 | 25  20  20  27  25  17 |

.

図 15: 総メモリ消費量の時系列変化

図 14のタスクセットをLMCLFスケジューリングによってスケジュールした時の総メモリ消費量の時系列変化を図 15に示す．LMCLFスケジューリングでは，最悪メモリ消費量が17 となり，LMCLF スケジューリングによってメモリ消費量を削減することができる．

*θi*(*siji*) を式 (5) から計算する際に，用いられる *α* は，メモリと時間の換算レートであり，*α*は，設計者が任意に定めるパラメータである．*α*の値をどのようにして決定するか，最適値が存在するかどうかは，本研究の本質的な問題ではないため，今後の課題とする．

# **6** プロセッサを意図的に休ませることによるメモリ削減

この章では，プロセッサを意図的に休ませることによってメモリ削減を図るスケジューリングについて記述する．まず，6.1節では，マルチプロセッサ環境下の問題点と，改良方針について説明する．その後，6.2 節と 6.3 節で，プロセッサを意図的に休ませることによってメモリ削減を図るスケジューリングについて定義し，改良したスケジューリングがメモリ削減できることを理論的に証明する．

## **6.1** マルチプロセッサ環境下の問題点と改良方針

マルチプロセッサ環境下のLMCFスケジューリングでは，複数のタスクが並列に実行されるため，複数のタスクが同時にメモリを確保する．さらに，タスクの切り替えが増加すると，複数のタスクがメモリを確保したまま一時停止することになり，結果として，ヒープメモリ消費量が増大してしまう可能性がある．この問題を解決するため，たとえタスクをプロセッサに割り当てることが可能であっても，意図的にタスクをプロセッサに割り当てないことによって，ヒープメモリ消費量を削減する改良型 LMCF スケジューリングを提案する．改良型 LMCF スケジューリングでは，各タスクの 2 ステップ先の消費メモリ増分は予測可能 [13] であると仮定する．改良型スケジューリングでは，消費メモリ増分が最小のタスク集合を選択し，選択したタスク集合をスケジュールした次状態を探索する．その後，探索した次状態において，もう一度，消費メモリ増分が最小のタスク集合を選択する．この操作を，高々 *p*（プロセッサ数）回繰り返す．ただし，これまでに選択されたタスク集合の和集合の要素数が *p* を超えた時，または，同じタスクが二度選択された時は，この操作を終了する．最後に，選択した全てのタスク集合の和集合に含まれる全てのタスクに最高優先度を付与する．

プロセッサを意図的に割り当てないことを考慮し，図 8 のタスクセットを 2 プロセッサ環境下でスケジュールする時の，総メモリ消費量の変化の全状態を図 16に示す．図中の下向き矢印は，Task 1 と Task 2 が並列処理された時の状態遷移である．図 16 のタスクセットを改良型 LMCF スケジューリングでスケジュールした時の状態遷移を図 17に示す．図 17から，改良型LMCFスケジューリングが最大メモリ消

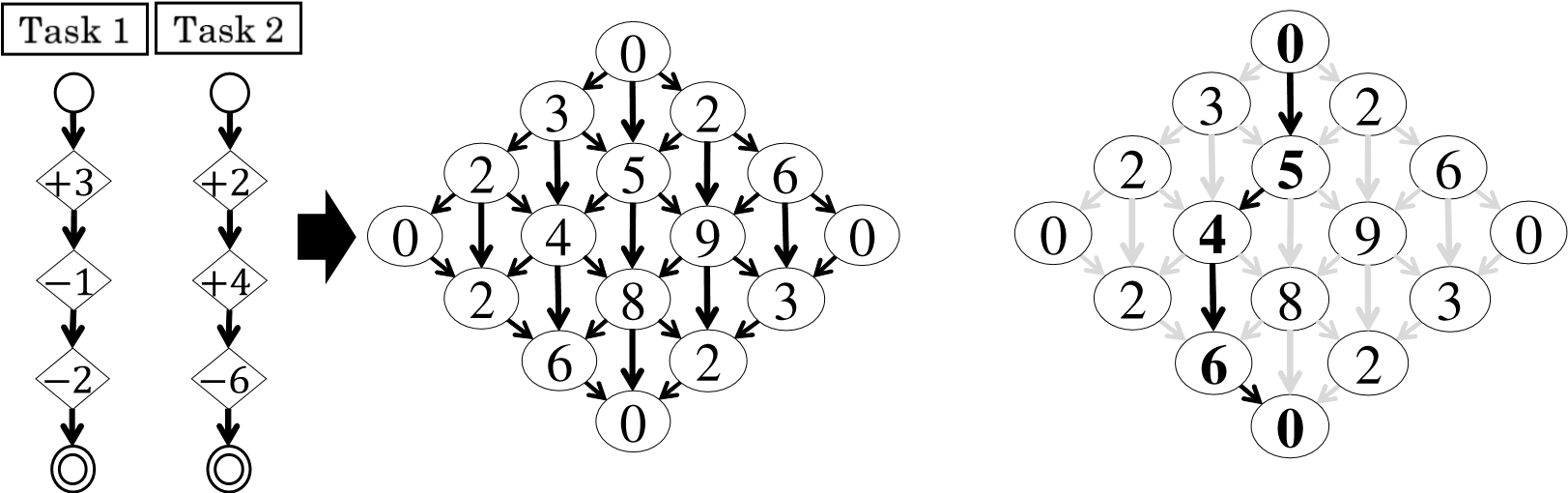


図 16: 各状態における総メモリ消費量 図 17: 改良型 LMCF スケジュール

費状態（総メモリ消費量が 9 の状態）を回避していることがわかる．さらに，最大で2つのプロセッサが利用可能であるため，図 17は，図 9よりも短い状態遷移で最終状態に到達している．

**6.2 LMCF**スケジューリングの改良

改良型 LMCF スケジューリングについて，以下に 2 段階で形式的に定義する．

定義 **4. Intentionally-Idling LMCF (II-LMCF)** スケジューリング

*p* プロセッサ環境下の「*TS* の Intentionally-Idling LMCF (II-LMCF) スケジュール」とは，「*n*個のタスク群から*p*′ ∈ {1*,...,p*}個のタスク群を選択する」という点以外は，定義 3（*p* プロセッサ環境下の LMCF スケジューリング）と同様である．つまり，合計*p*個の利用可能なプロセッサがある場合でも，*p*（′ 1 ≤ *p*′ ≤ *p*）プロセッサのみにスケジュールするタスクを割り当てることができる．

定理 **3.** 任意の正の整数 *h*(≥ 1) に関して，*p* プロセッサ環境下において，総メモリ消費量上位 *h* 位以内のいずれの状態も通らないスケジュールが存在するならば，任意の II-LMCF スケジュールは総メモリ消費量上位 *h* 位以内のいずれの状態も通らない．

*Proof.* 定理 2 と同様に証明できる.

定義 **5. Accelerated Intentionally-Idling LMCF (AcII-LMCF)** スケジューリング

**Accelerated Intentionally-IdlingLMCF**

Input :

Output :

Procedure:

Return ′,

′,

Input :

Output :

Procedure:

II-LMCF !

1. "#$%& #'(

)\*+#′ , ∪ , -

1. ./$01&, -

図 18: Accelerated Intentionally-Idling LMCF スケジューリングアルゴリズム

*TS* の AcII-LMCF スケジュールとは，初期状態以外の任意の状態*s* が，直前の状態で，図 18 のアルゴリズム *P*(*s*) によって得られたタスク集合をスケジュールした次状態となっている *TS* のスケジュールと定義する．

補題 **1.** *TS* の任意の状態***s***に対して，AcII-LMCF スケジューリングによる次状態 ***next*(*s,P*(*s*))**は，II-LMCFスケジューリングによって状態***s***から到達可能である．

*Proof.* 図 18のアルゴリズムによって***s***から到達可能なすべての状態は，II-LMCFスケジューリングによって順に到達可能である．タスク集合*P*(***s***)を同時にスケジュールした時の次状態は，図 18のアルゴリズムが停止した時に，最後に到達した状態と同じである．したがって，補題が成り立つ．

定理 **4.** 任意の正の整数 *h*(≥ 1) に関して，*p* プロセッサ環境下において，総メモリ消費量上位 *h* 位以内のいずれの状態も通らないスケジュールが存在するならば，任意の AcII-LMCF スケジュールは総メモリ消費量上位 *h* 位以内のいずれの状態も通らない．

*Proof.* 補題 1 より，任意の AcII-LMCF スケジュール *S* に対して，状態遷移を補完することによって対応するII-LMCFスケジュール*S*′ を求めることができる．定理 3 から，総メモリ消費量上位 *h* 位以内のいずれの状態も通らないスケジュールが存在するならば，*S*′ は総メモリ消費量上位 *h* 位以内のいずれの状態も通らない．*S* の任意の状態は必ず *S*′ に含まれることから，定理が成り立つ．

**6.3 LMCLF**スケジューリングの改良

Intentionally-Idling LMCLF (II-LMCLF) スケジューリングと Accelerated Intentionally-Idling LMCLF (AcII-LMCLF)スケジューリングは，LMCFスケジューリングを LMCLF スケジューリングに置き換えることで，定義 4 と定義 5 と同様に定義する．ただし，LMCLFスケジューリングに置き換えたとき，先に示した定理 3 と定理 4 が必ずしも成り立つとは限らない．

**7** メモリ削減スケジューリングと他のスケジューリングの併用

この章では，マルチプロセッサリアルタイムシステムにおいて，他の制約とメモリ消費量の削減を両立するためのスケジューリング手法について説明する．7.1でメモリ削減スケジューリングと他のスケジューリングの併用について定義し，理論的正当性を証明する．その後，7.2 で LLF スケジューリングとメモリ削減スケジューリングを併用したスケジューリング手法について提案する．

## **7.1** 定義と理論的正当性

提案したメモリ削減スケジューリング（LMCF，LMCLF，AcII-LMCF，AcIILMCLF）は，メモリ消費量の削減はできるが，マルチタスクシステムの効果（応答性の向上など）が得られない可能性がある．また，リアルタイムシステムにおいては，時間制約の充足が必要不可欠となる．そこで，多くの場合では，メモリ削減スケジューリングと他のスケジューリングを組み合わせたメモリ削減併用スケジューリングによって，メモリ消費量の削減と他の制約（時間制約など）との両立を図る．一般的に，メモリ削減併用スケジューリングでは任意のスケジューリングサイクルにおいて，メモリ削減スケジューリングと他のスケジューリングのどちらかを選択する．

メモリ削減併用スケジューリングについて以下のように定義する．

定義 **6.** メモリ削減併用スケジューリング

任意のタスクセット*TS* において，*TS* の状態を引数とし，ブール値（真または偽）を返す関数をスケジュール可能条件 *EC* と定義する．「スケジュール可能条件 *EC* に関するメモリ削減併用スケジュール」とは，*TS* のスケジュールで，その状態のうち *EC* が真である任意の状態に対しては，その次状態がメモリ削減スケジューリングによるものと同じであるようなスケジュールであると定義する．

LMCF 併用スケジューリングと AcII-LMCF 併用スケジューリングの理論的正当性を以下に示す．

定理 **5.** 任意の正の整数 *h*(≥ 1) とスケジュール可能条件 *EC* において，総メモリ消費量上位 *h* 位以内のいずれの状態も通らないスケジュールが存在するならば，総メモリ消費量上位 *h* 位以内のいずれの状態も通らない *EC* に関する LMCF 併用スケジュールが存在する．

*Proof.* 総メモリ消費量上位*h*位以内のいずれの状態も通らないスケジュールを*S* と

する．*EC*(*s*1*j*1*,...,snjn*) = *true*となるような*S* の任意の状態(*s*1*j*1*,...,snjn*)に対して，次にスケジュールされるタスク群をメモリ削減併用スケジューリングの方針に従うように組み替えることで，*EC* に関するメモリ削減併用スケジュール *S*′ を構築

することができる．（ここで，*S* の全ての状態において*EC*(*s*1*j*1*,...,snjn*) = *false*であれば，*S*′ は *S* と一致する．）*S*′ の任意の状態における総メモリ消費量が，*S* の対応する状態における総メモリ消費量以下となることは明らかである．

定理 **6.** 任意の正の整数 *h*(≥ 1) とスケジュール可能条件 *EC* において，総メモリ消費量上位 *h* 位以内のいずれの状態も通らないスケジュールが存在するならば，総メモリ消費量上位 *h* 位以内のいずれの状態も通らない *EC* に関する AcII-LMCF 併用スケジュールが存在する．

*Proof.* 定理 5 と同様に証明できる．

ただし，定理 5 と定理 6 は，定理 1 や定理 2，定理 4 （単なるメモリ削減スケジュール）とは違い，任意のメモリ削減併用スケジュールが総メモリ消費量上位*h*位以内の状態を回避することを必ずしも保証できない．また，LMCLF併用スケジューリングやAcII-LMCLF併用スケジューリングの理論的正当性については，証明することができていないため，本研究では，実験的に評価する．

**7.2** メモリ削減スケジューリングと**LLF**スケジューリングの併用

マルチプロセッサリアルタイムシステムにおいてヒープメモリ消費量を削減するために，LLF スケジューリングとメモリ削減スケジューリング（LMCF，LMCLF， AcII-LMCF，AcII-LMCLF）を組み合わせたスケジューリングを提案する．これを，

LLF併用メモリ削減スケジューリングと呼ぶ．LLFスケジューリングと組み合わせ

**LLF**

Inputs :

Outputs:

Procedure:

1. !"#$%#&'()\*+,
2. \*-.\*/0
3. 123\*4-567 \*89
4. : ;-LLF#&'<=

>?@>\*ABCLLF<=DABE

* 1. F4<=GHI\*9J
  2. K@GL3I ;-!"#$%

#&'()\*+,4M3NO

;-LLF#&'()

\*+,4 \*9J

図 19: LLF 併用メモリ削減スケジューリングアルゴリズム

る理由は，マルチプロセッサ環境下でのリアルタイムスケジューリングアルゴリズムにおいては，一般的に，EDFスケジューリング[14]よりもLLFスケジューリングの方が，スケジュール可能性が高いことが知られているからである[15]．LLF併用メモリ削減スケジューリングでは，次のスケジューリングサイクルのLLFスケジュール可能性判定 [15] の結果，LLF スケジューリング可能と判定された場合は，メモリ削減スケジューリングし，そうでなければ LLF スケジューリングする．

LLF スケジュール可能性判定は [17] でも提案されている．しかしながら，[17] で提案された LLF スケジュール可能性判定では，時刻 *t* におけるスケジュール可能性を判定できない．したがって，[15] の LLF スケジュール可能性判定を使用する．

LLF併用メモリ削減スケジューリングアルゴリズムを図 19に示す．まず，現在時刻*t*に対して，式 (2)から*F*(*k,t*+1)を計算する．この時，スケジューリングサイクルは 1 単位時間と仮定する．したがって，時刻 (*t* + 1) は次のスケジューリングサイクルを意味する．任意の整数*k*(*>* 0)で*F*(*k,t*+1) ≥ 0となるとき，LLFスケジュール可能と判定する．ただし，*Dmax* をタスクの相対デッドラインの最大値とすると， *k > Dmax* の時は，すべてのタスクの処理が終了しているはずなので，*k* は高々*Dmax* まで調べればよい．したがって，*k* の範囲は0 *< k* ≤ *Dmax* となる．次に，LLFスケジュール可能性判定の結果，次のスケジューリングサイクル（時刻 (*t* + 1)）において，LLF スケジュール可能と判定されたときは，現在のスケジューリングサイクル（現在時刻*t*）でメモリ削減スケジューリングし，そうでなければ，LLF スケジューリングする．このアルゴリズムは，タスクセットが時間制約に対して余裕がある場合は，メモリ削減の方針でスケジューリングし，時間に余裕がなくなるとき，LLF スケジューリングに切り替えることで時間制約を充足する．したがって，LLF スケジューリングでデッドラインミスが発生しないときは，このアルゴリズムでおいても，デッドラインミスは発生しない．

# **8** 評価実験

この章では，提案したメモリ削減スケジューリングアルゴリズム（LMCF スケジューリング，AcII-LMCFスケジューリング，LMCLFスケジューリング，AcII-LMCLF スケジューリング，LLF併用LMCLFスケジューリング，LLF併用AcII-LMCLFスケジューリング）がヒープメモリ消費量を削減することを評価するための評価実験について説明する．8.1節では，実験の方法について説明し，8.2節では，タスクセットの生成方法について説明し，8.4 節では，実験結果を示す．

## **8.1** 実験方法

提案したメモリ削減スケジューリングアルゴリズムを評価するため，ランダムなタスクセットを100個生成した．そのタスクセットに対して，EDFスケジューリン

グ，Earliest Deadline first until Zero-Laxity (EDZL) スケジューリング[16]，LLFスケジューリング，および，提案したメモリ削減スケジューリングアルゴリズムの最悪メモリ消費量とデッドラインミス発生率を測定する．

## **8.2** タスクセットの生成方法

本実験では，文献 [18] に基づいたタスクセットの生成方法を用いる．さらに，文献 [18] のタスクセット生成方法は，文献 [19] で提案されているタスクセット生成方法に基づいており，文献 [19] のタスクセット生成方法は様々な研究の評価実験において用いられている（例えば，[4][7][17][20]）．

タスクセットを生成するため，以下のパラメータを与える．

1. プロセッサ数 *p*(2 or 4)
2. [−10000*,*10000]の一様分布で決定した各タスクの消費メモリ増分の時系列変化
3. パラメータ(0*.*1*,*0*.*3*,*0*.*5*,*0*.*7*,*0*.*9)の指数分布で決定した個々のタスクのプロセッサ利用率 *δi* = (*Ci/Ti*)，[100*,*1000] の一様分布で決定した最小リリース間隔 *Ti*

（ただし，本実験ではデッドラインに焦点をあてるため，*Ti* = *Di* する．）

1. 与えられた *δi* と *Ti* から算出した実行時間 *Ci*

上記のパラメータに対して，以下のStepに従い，タスクセットを100個生成する．

Step 1 初期値として，*p* + 1 個のタスクを生成する．

Step 2 スケジュール不可能なタスクセットを取り除くために，生成されたタスクセッ

トが必要条件（∑*τi*∈*TS δi* ≤ *p*）を満たすかどうかを判定する [21]．

Step 3 もし必要条件を満たさなければ，そのタスクセットを破棄し，Step 1 に戻る．もし必要条件を満たせば，そのタスクセットを実験するタスクセットに含め，そのタスクセットに新しいタスクを一つ追加して，Step 2 に戻る．

本実験で用いた CPU は，Intel(R) Xeon(R) CPU E5-2643v2 @3.50GHz であり，

OS は，Windows 10 Pro 64 bit である．

**8.3** 周期タスクに対する**LLF**スケジュール可能性判定

[15] で提案されている LLF スケジュール可能性判定は，周期タスクを対象としていないので，周期タスクをスケジュールする時には，8.3 節の LLF スケジュール可能性判定を行う．

周期タスクに対して LLF スケジュール可能性判定を行うとき，将来起動するタスク既に起動したと仮定する．まず，任意の整数 *r*(*>* 0) に対して，*r* 回目に起動するタスク *τi* を *τir*，*τir* の起動時刻を *Air* とする．次に，全てのタスクが時刻 0 で起動したと仮定する．このとき，タスク*τir* の相対デッドライン*Dir*(*t*)，残余実行時間

*Cir*(*t*)，余裕時間*Lir*(*t*)は以下の式 (6)から計算する．最後に，*Di*(*t*)を*Dir*(*t*)，*Ci*(*t*) を *Cir*(*t*)，*Li*(*t*) を *Lir*(*t*) と置き換えて，3.3 節と同様に LLF スケジュール可能性判定を行う．

|  |  |
| --- | --- |
| *Dir*(*t*) = *Ti* + *Air* − *t rr* *Cri*(0)− (*t < Ar ir*)  *Ci* (*t*) =  *Ci*(*t*) (*otherwise*)  *Li* (*t*) = *Di* (*t*) *Ci* (*t*) | (6) |

## **8.4** 実験結果

2プロセッサ環境下と4プロセッサ環境下でそれぞれ実験を行った．EDFスケジューリング，EDZLスケジューリング，LLFスケジューリング，本研究で提案したメモリ削減スケジューリングにおける最悪メモリ消費量とデッドラインミス発生率を図 21 と図 22 にそれぞれ示す．横軸は，図 20 に示したスケジューラである．タスクセットに含まれるタスク数の平均 *~~n~~*，個々のタスクのプロセッサ利用率 *δi* をそれぞれ計算した．

|  |  |
| --- | --- |
| **No.** |  |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | EDF  EDZL  LLF  LMCLF  LMCLF  AcII-LMCF  AcII-LMCLF  AcII-LMCLF  AcII-LMCLF  LLFLMCLF  LMCF  LMCLF  LLF  AcII-LMCLF  LLF  AcII-LMCLF  LLF  AcII-LMCLF  LLFLMCLF  LLFLMCLF |

図 20: 実験で用いたスケジューラ

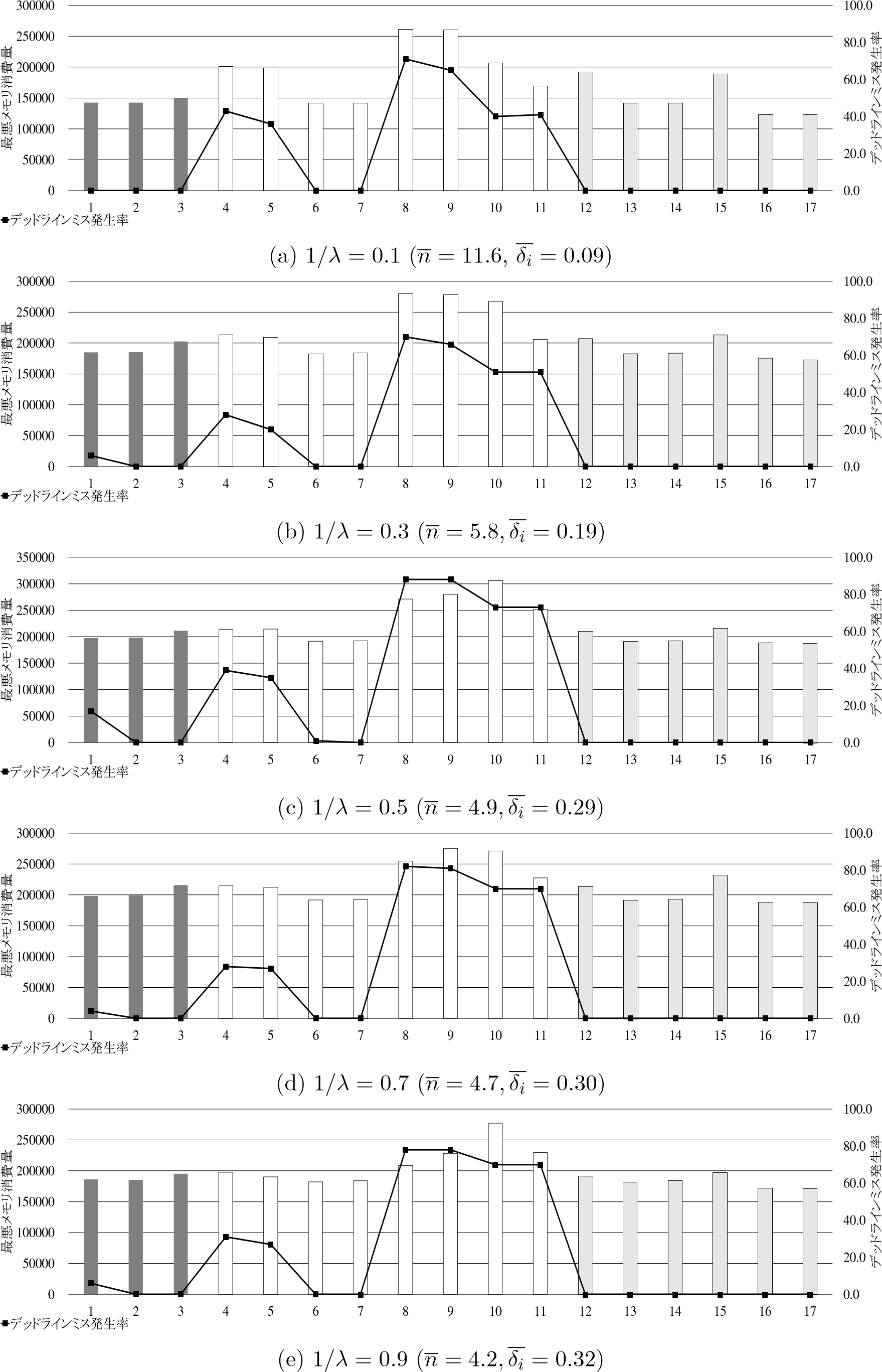


図 21: 2 プロセッサ環境下

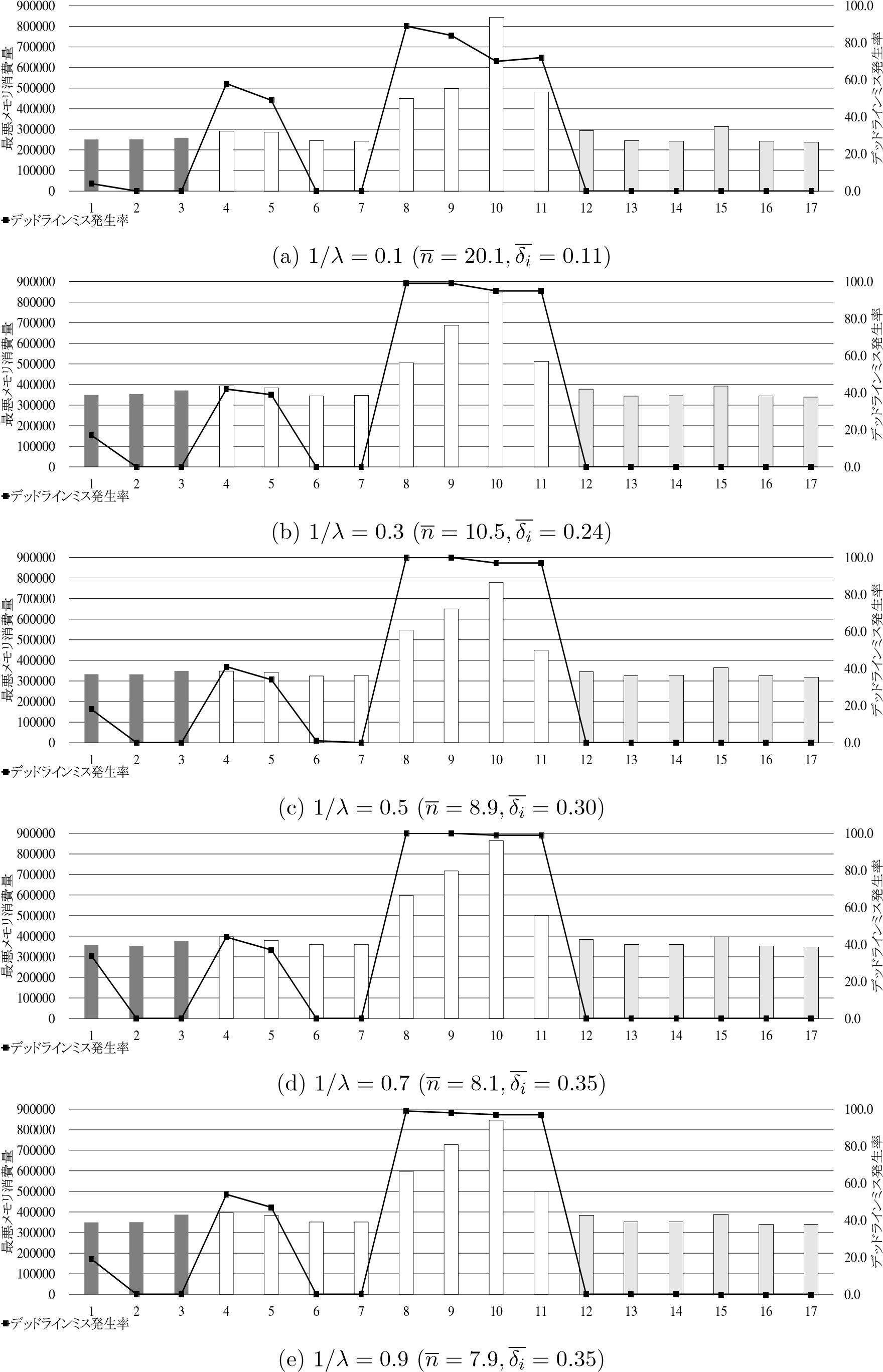


図 22: 4 プロセッサ環境下

# **9** 考察

図 21 と図 22 から，LMCLF スケジューリング (No.6, No.7)，LLF 併用 LMCLF スケジューリング(No.13, No.14)，LLF併用AcII-LMCLFスケジューリング(No.16,

No.17)は，EDFスケジューリング(No.1)，EDZLスケジューリング(No.2)，LLFスケジューリング (No.3) に比べて平均的にメモリ削減できるとわかる．特に，2 プロセッサ環境下では，No.6，No.7，No.13，No.14はLLFスケジューリングよりも5.4% から11.1%メモリ削減し，No.16とNo.17はLLFスケジューリングよりも10.7%から

17.9%メモリ削減している．4プロセッサ環境下では，No.6，No.7，No.13，No.14は

LLF スケジューリングよりも 4.5%から 9.1%メモリ削減し，No.16 と No.17 は LLF スケジューリングよりも 5.9%から 11.9%メモリ削減している．これらのことから，プロセッサ数に関係なくメモリ消費量を削減できると考えられる．

さらに，LLF 併用メモリ削減スケジューリング (No.12 から No.17) では，LLF スケジューリングと組み合わせることによって，デッドライン制約を充足していることがわかる．

No.5，No.12，No.15 は，それぞれ *α* = 1*.*0 の LMCLF スケジューリング，LLF 併用 LMCLF スケジューリング，LLF 併用 AcII-LMCLF スケジューリングである． *α* = 1*.*0 とは，残余実行時間と余裕時間よりも消費メモリ増分を重要視することを意味する．図 21と図 22から，No.5，No.12，No.15は，LLFスケジューリングよりもメモリ削減ができていないことがわかる．したがって，メモリ削減するためには，消費メモリ増分だけでなく，残余実行時間と余裕時間を考慮する必要があると実験的に示された．ただし，*α* = 1 と *α* = 0 のどちらが適切かどうかは，タスクセットによって依存するため，本実験からは断定できない．

図 21a と図 22a は，タスク数が多く，個々のタスクのプロセッサ利用率が低いタスクセットの実験結果であり，図 21e と図 22e は，タスク数が少なく，個々のタスクのプロセッサ利用率が高いタスクセットの実験結果である．これらの実験結果から，No.6，No.7，No.13，No.14，No.16，No.17は，タスク数やタスクのプロセッサ利用率にかかわらず，平均的にメモリ削減できるとわかる．

一方で，No.8からNo.11のスケジューリングでは，メモリ削減に失敗していることがわかる．No.8からNo.11は，プロセッサを意図的に休ませるというスケジューリング方針である．これは，プロセッサを意図的に休ませることにより，より多くのタスクが終了せずに並行してスケジューリングされる傾向があるためであると考えられる．

# **10** 結論

本研究では，マルチプロセッサハードリアルタイムにおいて，ヒープメモリ消費量を削減するリアルタイムスケジューリング手法を提案した．提案したメモリ削減スケジューリングアルゴリズム（LMCLF スケジューリング，LLF 併用 LMCLF スケジューリング，LLF併用AcII-LMCLFスケジューリング）が，ヒープメモリ消費量を削減することを理論的に証明した．さらに，本研究では，消費メモリ増分だけでなく残余実行時間と余裕時間を考慮した LMCLF スケジューリングを提案した．実験結果から，提案したメモリ削減スケジューリングは，EDF スケジューリング，

EDZL スケジューリング，LLF スケジューリングよりも少ないメモリ消費量となることが示された．特に，LLFスケジューリングに比べて，2プロセッサ環境下と4プロセッサ環境下で，約 5%から 20%程度メモリ削減した．また，LLF スケジューリングと併用した場合，ランダムに生成したすべてのタスクセットにおいて，デッドライン制約を充足した．これらの結果から，LLF 併用メモリ削減スケジューリングは，メモリ削減とデッドライン制約の充足を両立することができると結論できる．提案したメモリ削減スケジューリングアルゴリズムでは，次の状態の消費メモリ増分は予測可能であると仮定している．これについては，事前にプログラムを実行し，メモリ割り当て箇所の消費メモリ増分を記録（プロファイリング）することで，前回の消費メモリ増分に基づいて，現在実行している同じプログラムに対するメモリ割り当て箇所の消費メモリ増分を予測する [13]．また，WCET 解析[22] のような静的プログラム解析によって，予測精度はさらに向上する可能性がある．

今後の課題として，LMCLF スケジューリングがヒープメモリ消費量を削減することを理論的に証明することや，LMCLF スケジューリングで使用する換算レート *α* の決め方を考えることなどが挙げられる．

# 参考文献

1. R. Zurawski, “Embedded Systems Handbook, Second Edition: Embedded Systems Design and Verification”, CRC Press, 2009.
2. N. Vasudevan and S. A. Edwards, “Buffer Sharing in CSP-like Programs”, In

Proc. of 7th ACM/IEEE Int. Conf. on Formal Methods and Models for CoDesign (MEMOCODE 2009), 2009.

1. S. A. Edwards and O. Tardieu, “SHIM: A deterministic model for heterogeneous embedded systems”, In Proc. of 5th ACM Int. Conf. on Embedded Software (EMSOFT 2005), pp.37–44, 2005.
2. M. Bertogna, M. Cirinei, and G. Lipari, “Schedulability Analysis of Global

Scheduling Algorithms on Multiprocessor Platforms”, IEEE Transactions on Parallel and Distributed Systems, Vol. 20, No. 4, pp.553–566, 2009.

1. R. I. Davis and A. Burns, “A Survey of Hard Real-Time Scheduling for Multiprocessor Systems”, ACM Computing Surveys, Vol. 43, No. 4, Article 35, pp1–44, 2011.
2. G. Yao, R. Pellizzoni, S. Bak, E. Betti, and M. Caccamo, “Memory-centric scheduling for multicore hard real-time systems”, Real-Time Systems, Vol. 48, Issue 6, pp.681–715, 2012.
3. J. Lee and I. Shin, “EDZL Schedulability Analysis in Real-Time Tasks”, IEEE Transactions on Software Engineering, Vol. 39, No. 7, pp.910–916, 2013.
4. A. Alhammad, S. Wasly, and R. Pellizzoni “Memory Efficient Global Scheduling of Real-Time Multicore Scheduling”, 21th IEEE Real-Time and Embedded Technology and Applications Symposium, pp.285–296, 2015.
5. J. Lee, J. Lee, K. M. Phan, A. Easwaran, and I. Shin, “Global EDF Schedulability Analysis for Parallel Tasks on Multi-Core Platforms”, IEEE Transactions on Parallel and Distributed Systems, Vol. 28, No. 5, pp1331–1345, 2017.
6. 船瀬広岐中田明夫, 「変数の生存期間を考慮してヒープメモリ使用量削減を行うマルチタスクスケジューリング手法の検討」, 電子情報通信学会ソフトウェア

サイエンス研究会技術研究報告 (SS2012), 信学技報 (SS2012–2), Vol. 112, No.

23, pp.7–12, 2012.

1. Y. Machigashira and A. Nakata, “An improved LLF scheduling for reducing maximum memory consumption by considering laxity time”, In Proc. of 12th

Int. Symp. on Theoretical Aspects of Software Engineering, pp.144–149, IEEE Computer Society Press, 2018.

1. 町頭優輝中田明夫, 「余裕時間の考慮によりマルチプロセッサリアルタイムシステムのヒープメモリ消費量を削減する改良LLFスケジューリング」, 電子情報

通信学会ソフトウェアサイエンス研究会報告 (SS2018), 信学技報 (SS2018–55),

Vol. 118, No. 471, pp.19–24, 2019.

1. 中山敬基中田明夫,「ヒープメモリ使用状況の予測によりメモリ使用量削減を行うマルチタスクスケジューリング手法の提案」, 電子情報通信学会ソフトウェア

サイエンス研究会技術研究報告 (SS2016), 信学技報 (SS2016–39), Vol. 116, No.

426, pp.19–24, 2017.

1. C. L. Liu and J. W. Layland, “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment”, Journal of the ACM, Vol.20, No.1, pp.46–61, 1973.
2. M. L. Dertouzos and A. K. Mok, “Multiprocessor On-Line Scheduling of Hard-

Real-Time Tasks”, IEEE Tran. on Software Engineering, Vol.15, No.12, 1989.

1. S. K. Lee, “On-line Multiprocessor Scheduling Algorithms for Real-Time Tasks”

In Proc. of IEEE Region 10’s 9th Annual Int. Conf., pp.607-611, 1994.

1. J. Lee, A. Easwaran, and I. Shin, “Laxity Dynamics and LLF Schedulability Analysis on Multiprocessor Platforms”, Real-Time Systems, Vol. 48, Issue 6, pp716–749, 2012.
2. J. Lee, “Time-Reversibility for Real-Time Scheduling on Multiprocessor Systems”, IEEE Transactions on Parallel and Distributed Systems, Vol. 28, No. 1, pp230–243, 2017.
3. T. P. Baker, “Comparison of Empirical Success Rates of Global vs. Partitioned

Fix-Priority and EDF Scheduling for Hard Real Time”, Technical Report TR-

050601, Department of Computer Science, Florida State University, pp.1–14, 2005.

1. J. Lee, A. Easwaran, and I. Shin, “Maximizing Contention-Free Executions in

Multiprocessor Scheduling”, In Proc. of 17th IEEE Real-Time and Embedded Technology and Applications Symposium, pp.235–244, 2011.

1. S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, “Proportionate

Progress: A Notion of Fairness in Resource Allocation”, Algorithmica, Vol. 15, Issue 6, pp600–625, 1996.

1. C. Zoubek and P. Trommler, “Overview of worst case execution time analysis in single- and multicore environments”, In Proc. of 30th International Conference on Architecture of Computing Systems (ARCS2017), pp.146–150, 2017.

# 発表業績

1. Y. Machigashira and A. Nakata, “An improved LLF scheduling for reducing maximum memory consumption by considering laxity time”, In Proc. of 12th

Int. Symp. on Theoretical Aspects of Software Engineering, pp.144–149, IEEE Computer Society Press, 2018.

1. 町頭優輝中田明夫, 「余裕時間の考慮によりマルチプロセッサリアルタイムシステムのヒープメモリ消費量を削減する改良LLFスケジューリング」, 電子情報通信学会ソフトウェアサイエンス研究会技術研究報告(SS2018), 信学技報

(SS2018–55), Vol. 118, No. 471, pp.19–24, 2019. （平成 30 年度電子情報通信

学会ソフトウェアサイエンス研究会研究奨励賞受賞）

1. 町頭優輝中田明夫, 「ヒープメモリの確保・解放量と実時間制約を共に考慮しマルチプロセッサシステムのメモリ消費量を削減するリアルタイムスケジューリング」, 電子情報通信学会ソフトウェアサイエンス研究会技術研究報告 (SS2019). （2020 年 3 月発表予定）
2. Y. Machigashira and A. Nakata, “Global Scheduling for Reducing Heap Memory Consumption in Multiprocessor Hard Real-time Systems”, Real-time Systems Journal, 2019.（投稿中）

# 謝辞

本研究を行うにあたってご指導いただいた，中田明夫教授に心より感謝申し上げます．また，本研究へ貴重な助言をいただいた村田佳洋准教授，双紙正和准教授，島和之准教授，佐藤康臣助教に心より感謝いたします．

最後に，お世話になった広島市立大学組込みデザイン研究室の皆様にお礼申し上げます．