一般社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

メモリ使用量の動的変化に対する適応的制御によりデッドラインを充足し総メモリ使用量を削減するマルチタスクスケジューリング手法の提案

新井 諒介† 中田 明夫‡

†, ‡広島市立大学 大学院情報科学研究科 システム工学専攻

〒713-3194 広島県広島市安佐南区大塚東 3-4-1

E-mail: †mg68002@e.hiroshima-cu.ac.jp , ‡nakata@hiroshima-cu.ac.jp

あらまし 筆者らは従来,メモリ使用量と実時間制約の両方を考慮した優先度を設定することにより実時間制約を守りながらシステム全体の総メモリ使用量を削減することができるマルチタスクスケジューリング手法 LMCLF を提案している.LMCLF は,設計者が任意に定めるパラメータαを導入し,各タスクの「α ×（メモリ使用量の増分）＋（余裕時間）×（残余実行時間）」の値が小さい順に優先度を高くすることで,メモリ使用量削減とデッドライン充足の両者を勘案したスケジューリングを実現している.しかし,余裕時間や残余実行時間の数値と比較してメモリ使用量の数値が極端に大きい場合やその逆の場合,ほぼメモリ削減かデッドライン充足の一方のみが優先されてしまう可能性がある.本研究では,LMCLF におけるパラメータαのより適切な値をスケジューラ動作中に自動推定することにより,任意のタスクセットに対してメモリ使用量削減とデッドライン充足の両者を実現するスケジューリング手法を提案する.シミュレーション実験により評価を行った結果,提案手法は従来手法よりもメモリ使用量削減の効果が大きく,かつ,LLF スケジューリングと同等にデッドラインを充足可能であることが分かった.

キーワード リアルタイム組込みシステム,マルチタスクシステム,マルチプロセッサシステム,リアルタイムスケジューリング,メモリ削減技術

Multitask Scheduling for Reducing Total Memory Consumption while Satisfying Deadlines using Adaptive Control according to Dynamic Change of Task Memory Consumption

Ryosuke ARAI† Akio NAKATA‡

†, ‡ Graduate School of Information Sciences, Hiroshima City University 3–4–1 Ozuka-Higashi, Asa-Minami-ku, Hiroshima, 731–3194 Japan

E-mail: †mg68002@e.hiroshima-cu.ac.jp, ‡nakata@hiroshima-cu.ac.jp

Abstract The authors have previously proposed a multi-task scheduling method LMCLF, which can reduce the total memory usage of the entire system while satisfying the real-time constraints by setting priorities that take both memory usage and real-time constraints into account. LMCLF introduces a parameter α, which is arbitrarily set by the designer, and gives higher priority to each task in the decreasing order of the value of "α × (incremental memory usage) + (laxity time) × (remaining computation time)," thereby realizing scheduling that considers both memory usage reduction and deadline guarantee. However, if the memory usage value is extremely large compared to the laxity time and/or remaining computation time values or vice versa, the scheduling will consider almost only one of either memory reduction or deadline guarantee. In this paper, we propose a scheduling method that achieves both memory usage reduction and deadline guarantee for an arbitrary task set by automatically estimating a more appropriate value of the parameter α in LMCLF during the scheduler operation. As a result of simulation experiments, we found that the proposed method is more effective in reducing memory usage than the existing method, and is as effective as LLF scheduling.

Keywords Embedded and Real-time Systems, Multitasking Systems, Multiprocessor Systems, Real-time Scheduling, Memory Saving Technology

1. まえがき

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，メモリ使用量を削減することが挙げられる [1]．なぜなら，メモリ使用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．

一方で，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，ヒープメモリ使用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使用量は，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使用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使用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使用量は増加する.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]．先行研究では,マルチタスクシステムのヒープメモリ使用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) スケジューリングが提案されている [8][9]．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，次のステップのメモリ使用量の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，ヒープメモリ使用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．

一方，LMCF スケジューリングは実時間制約を考慮しないため，メモリ使用量の削減はできるが，応答性の向上などのマルチタスクシステムによる効果が得られない可能性がある．そこで，先行研究ではメモリ使用量の増分だけでなく，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されている[10]．LMCLF スケジューリングでは，(α × メモリ使用量の増分 + 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高くすることで,総メモリ使用量削減とデッドライン充足の両立を目指す．（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.）しかし,メモリ使用量とデッドラインに大きなばらつきがある場合は,事前に適切なαの値を設定するのは困難である.もしαの値が最適でない場合,最適である場合よりもデッドラインミスが増加するか，総メモリ削減量が減ってしまう可能性がある．本研究では,スケジューラ動作中に残余実行時間や余裕時間の値に対してメモリ使用量の増分が非常に大きくなる場合やその逆の場合でも,デッドラインを充足し十分なメモリ使用量の削減ができるように,スケジューラが動作中に観測する各タスクのメモリ使用量増分に対して,パラメータαの値を適応的により最適な値に変更する変化させる手法を提案する.提案手法により，αを事前に定める必要が無いかつ，αの値が最適でない場合の従来手法よりもデッドラインを充足しかつ総メモリ使用量を削減することが期待できる．

2. システムモデル

本研究で取り扱うシステムモデルは，[10]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T_i,最悪実行時間をC_i(Worst-case Execution Time, WCET）,相対デッドラインをD_iとし，このモデルにおけるタスク$\tau_i = (T_i, C_i, D_i)$はタスクセット TS に含まれる$(\tau_i \in TS)$．タスクτ_i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T_i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τ_i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$D_i(t), C_i(t)$とし,タスク τ_iの時刻 t における余裕時間を$L_{i(t)}$とし,式 (1) から計算する．

$$L_i(t) = D_i(t) - C_i(t) \quad (1)$$

タスク数の合計はnとし,システムの利用率を$U_{sys} = \sum \tau_i \in TS \frac{C_i}{T_i}$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

$n(\geq 1)$を任意の自然数とする．任意の $i \in \{1, \ldots, n\}$に

対して，タスク τ_iは 1 ステップ目から e_i（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，τ_iの $j(1 \leq j \leq e_i)$ステップ目の状態を s_{ij}とする．

タスクの集合（以下，タスクセットと呼ぶ）を $TS = \{\tau_1, \tau_2, \ldots, \tau_n\}$とする．$i \in \{1, \ldots, n\}$に対して，$ji(1 \leq ji \leq e_i)$ を自然数とし，TS の状態は各タスクの状態の直積 $(s_1j_1, s_2j_2, s_3j_3, \ldots, s_nj_n)$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 s_{i1}）の直積，TS の最終状態はすべてのタスクの最終状態（e_iステップ目の状態 s_ie_i）の直積と定義する．

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

T_iに対して，$\Sigma_{1 \leq j \leq e_i - 1} m(s_{ij}) = 0$を満たすような各状態 s_{ij} への整数値 $\mathrm{m}(s_{ij})$の割り当てを状態 s_{ij}における「消費メモリ増分」と呼ぶ．

3. LMCLF スケジューリング

LMCLF スケジューリング[10]は，マルチタスクシステムにおいて，デッドライン制約を充足しつつメモリ使用量の削減をするスケジューリング手法である．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，$\theta_i(s_{ij})$を式 (2) から計算し，$\theta_i(s_{ij})$の値が小さいタスクから順に優先度を付与する．

$$\theta_i(s_{ij}) = \alpha \times m(s_{ij}) + C_i(s_{ij}) \times L_i(s_{ij}) \ (\alpha > 0) \quad (2)$$

$\theta_i(s_{ij})$を式(2) 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，適切な値を設計者がスケジュール対象のタスクセットに合わせて事前に定める必要がある．

しかし，メモリ使用量の増分が非常に大きくなる場合やその逆の場合は，事前に適切なαの値を設定するのは非常に困難である．

4. パラメータ α の適応的決定

本研究では，LMCLF スケジューリングにおけるパラメータ αの値をタスクセットによって適応的に決定する手法を提案する．

一般に遠い未来のメモリ使用量の予測は困難であるため，提案手法では現在時刻から高々2 ステップ先のメモリ使用量増分の予測は可能と仮定し，2 ステップ先までのスケジュールにおいて最適，すなわち，メモリ使用量が最小となり，かつ，デッドラインを充足するパラメータαの決定を目指す．しかし，マルチプロセッサスケジューリングにおいては，1 ステップのスケジュールの可能な場合の数でさえ，全タスクのプロセッサ数分の部分集合の総数となり，タスク数に応じて指数関数的に増大するため，2 ステップ分の全ての可能なスケジュールの候補を対象に最適となるαの探索をするのは，計算量が大きくなりすぎて現実的ではない．そこで提案手法では，αの暫定的な初期値を定め（以下，α'とする）α'に基づいて各タスクの優先度を算定し，優先度が上位となる一部のタスク群（本研究では上位プロセッサ数×2 位のタスク群とする）を対象に探索を行い，その結果に基づいて最適なαの値を決定する．その際に，実行中タスクの余裕時間の最小値の桁数（以下，$LaxityUrgency$とする）を求め，その値がメモリ使用量の桁数に比べて非常に小さいならばαの値も小さくするように調整することで，余裕時間が 0 に近いタスクの優先度を上げてデッドライン充足を図る．具体的な方法としては，スケジュールされるタスクの余裕時間の対数の逆数($LaxityUrgency$)を取り，その対数の底の $LaxityUrgency$のべき乗で重みづけを行う．

このように決定したαの値を次のスケジューリング

- 入力:タスク$\tau i(i = 1, \ldots, n)$の時刻$t(t = 0,1)$における残余実行時間$c_i(t)$,余裕時間$L_i(t)$,及び$q(q = 0,1)$ステップ目におけるメモリ増分$m_i(q)$,プロセッサ数を$P(p >= 1)$,前の周期で決定したα'
- 出力:LMCLF でスケジュールした場合に,2 ステップ後のメモリ消費量が最小となるようなα

minVal=∞とする．

1 評価値$(\alpha' * m_i(0) + c_i(0) * L_i(0))$の値が小さい下位 P*2 個分の任意のタスク i の集合 I ⊂ {1, …, n}(ただし$|I| = P * 2$)に対して、以下を繰り返す

1.1 任意のi ∈ Iに対して、以下を繰り返す

1.1.1 タスクiが LMCLF アルゴリズムでタスク集合Iに含まれない任意のタスクjよりも優先してスケジュールされるための α の上限$\alpha^U_{i,j}$または下限$\alpha^L_{i,j}$を求める

1.1.2 1.1.1 求めた$\{\alpha^U_{i,j}, \alpha^L_{i,j}\}_\{i \in I, j \in \{1, \ldots, n\} \setminus I\}$から、LMCLF により 1 ステップ目にタスク集合 I がスケジュールされるための α の上限$\alpha^u_I = \min_{i \in I}\{\min_j\{\alpha^u_{ij}\}\}$および下限$\alpha^l_I = \max_{i \in I}\{\max_j\{\alpha^l_{ij}\}\}$を求める

1.2 評価値$(\alpha' * m_k(0) + c_k(0) * L_k(0))$の値が小さい下位 P*2 個分の任意のタスク k の集合 K ⊂ {1, …, n}(ただし$|K| = P * 2$)に対して、以下を繰り返す

1.2.1 任意のk ∈ Kに対して、以下を繰り返す

1.2.1.1 タスク k が LMCLF アルゴリズムでタスク集合Kに含まれない任意のタスクlよりも優先してスケジュールされるための α の上限$\alpha^U_{k,l}$または下限$\alpha^L_{k,l}$を求める

1.1.1 上記で求めた{α^u_{kl},α^l_{kl}}_{k ∈ K, l ∈ {1, …, n} \ K}から、LMCLF により 2 ステップ目にタスク集合 K がスケジュールされるための α の上限$\alpha^U_{I,K} = \min[\alpha^U_I, \min_{k \in K}\{\min_l\{\alpha^U_{kl}\}\}]$および下限$\alpha^L_{I,K} = \max[\alpha^L_I, \max_{k \in K}\{\max_l\{\alpha^L_{kl}\}\}]$を求める

1.2.2 $\alpha^U_{I,K} > 0$ かつ $\alpha^U_{I,K} > \alpha^L_{I,K}$ を満たすαが存在するならば

1.2.2.1 1 ステップ目でスケジュールされたタスクの評価値の合計及び，1 ステップ目，2 ステップ目でスケジュールされたタスクの評価値を足したものの合計，のうち大きい値 $Val = \max\{\Sigma_{i \in I}\{(\alpha' * m_i(0) + c_i(0) * L_i(0))\}, \Sigma_{k \in K}\{(\alpha' * m_k(qk) + c_k(1) * L_k(1))\}\}$を求める

1.2.2.2 $LaxityUrgency(I, K) = max[\max_{i \in I}\{f(L\_i(0))\}, \max_{k \in K}\{f(L\_k(1))\}$ (ただし，$x > 0$のとき$f(x) = (1/\log_b x)$ かつ $f(0) = \infty$)}を求める

1.2.2.3 minVal > Val ならば

1.2.2.3.1 minVal=Val とする

1.2.2.3.2 $\alpha^L_{I,K} < \infty$ ならば

1.2.2.3.2.1 $\alpha = (\alpha^u_I + \alpha^l_u) * b^{-LaxityUrgency\ (I,K)} / 2$

1.2.2.3.3 さもなければ

1.2.2.3.3.1 $\alpha = (\alpha^l_u) * b^{-LaxityUrgency\ (I,K)}$

図 1：提案手法のアルゴリズム

周期におけるα'の値として用いることにより,スケジュールが進むにつれてαの値を最適値に近づける.

以上の方針に基づいて具体化したアルゴリズムを図1に示す.

5. 評価実験

5.1. 実験目的・方法

提案手法の有効性を評価するために,ランダムに生成されたタスクセットを用いたシミュレーション実験を行った.実験では従来手法の LMCLF(α=1,100,1000), 余裕時間が少ないタスクから高い優先度を与える LLF スケジューリング[11],および,提案手法のスケジューリングの最悪メモリ使用量と平均デッドラインミス回数を測定した.提案手法がαの値が最適でないときの従来手法よりもメモリ削減できるか,また,既存手法のうち最もデッドラインを充足しやすい手法として知られる LLF スケジューリングと比較することにより,デッドラインミスが起こらないかを確認することを実験目的とする.タスクセットの生成法については文献[12]と同様に行う.

5.2. タスクセットの生成方法

本実験では文献[10]と同様に，文献 [5] のタスクセットの生成方法に基づき，各タスクのメモリ使用量の情報を追加してタスクセットをランダム生成する．文献 [5] のタスクセット生成方法は，文献 [12] 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，様々なリアルタイムスケジューリング手法の評価実験において用いられている（例えば，[6][7]）．提案手法がどのようなタスクセットでも対応できることを示すため上記のタスクセット生成法と同様の方法でメモリ使用量とプロセッサ利用率のパラメータを変更する．

タスクセットを生成するため，以下のパラメータ群を与える

- タスクセット

  プロセッサ数 P4

  [−1000000,1000000]のそれぞれ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

  パラメータ 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 $\delta i = (Ci/Ti)$

  [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.）

  与えられた δi と Ti から算出した実行時間 Ci

パラメータ群に対して，以下の Step に従い，10 個のタスクからなるタスクセットを生成する.

- 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 $\sum \tau i \in TS\ \delta i \leq p$）を満たすかどうかを判定する [9].
- Step 3 もし満たさなければ,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.

本実験で用いた計算機の CPU は，Intel(R) Xeon(R) CPU E5-2643v2 @3.50GHz ,メモリ容量は 16305MB であり OS は，Windows 10 Pro 64 bit である.

5.3. 実験結果

実験結果を図 2 に示す.最悪メモリ使用量が箱ひげ図でデッドラインミス回数の平均が折れ線グラフで示されている.

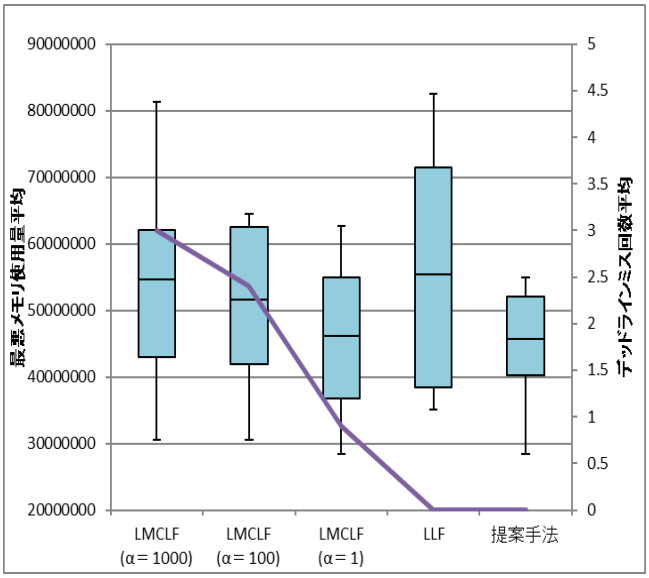

図 2：実験結果

6. 考察

図 2 の箱ひげ図や折れ線グラフから提案手法のほうが従来手法の LMCLF および LLF スケジューリングよりも,デッドラインミス回数,メモリの削減量の双方において優れていることが分かった.折れ線グラフを見てわかる通りデッドラインミスの観点では余裕時間優先の LLF と同じでデッドラインミス回数が 0 でありデッドライン制約を充足できている.また箱ひげ図より最悪メモリ使用量の平均値がどのスケジュールよりも優れているため,提案手法はデッドライン制約を充足しかつ,総メモリ使用量の削減もできていることがわかる.

7. 結論

本論文では,[10]で提案された LMCLF スケジューリ

ングを改良し，残余実行時間や余裕時間の値に対してメモリ使用量の増分が非常に大きくなる場合やその逆の場合でもデッドラインを充足し十分なメモリ使用量の削減ができるように，スケジューラが動作中に観測する各タスクのメモリ使用量増分に対して，パラメータαの値を適応的により最適な値に変更する手法を提案した.実験的評価により，提案手法はランダム生成したタスクセットに対して，従来手法よりもデッドライン充足とメモリ使用量削減を両立したスケジュールを行うことができることが分かった.

一方で，本研究では余裕時間によるパラメータαの調整を対数関数により定義したが，線形関数など他の関数で定義した場合との比較評価は行っていない．また，スケジューラは一般に非常に短い周期で動作する必要があるため，オーバヘッド低減のために優先度を浮動小数点演算より高速な整数演算で行う方が望ましい．今後の課題としては，パラメータαの調整を対数関数以外で行う場合の比較評価や，優先度の計算を固定小数点演算やシフト演算などを用いて効率化することなどが挙げられる．

文　献

[1] R. Zurawski, “Embedded Systems Handbook, Second Edi-tion:Embedded Systems Design and Verification”, CRC Press, 2009.

[2] S. A. Edwards and O. Tardieu, “SHIM: A deterministic model for heterogeneous embedded systems”, In Proc. of 5th ACM Int. Conf. on Embedded Software, pp.37–44, 2005.

[3] 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 Co-Design, 2009.

[4] 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.

[5] J. Lee, “Time-Reversibility for Real-Time Scheduling on Multiprocessor Systems”, IEEE Transactions on Parallel and Distributed Systems, Vol. 28, No. 1, pp.230-243, 2017.

[6] 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.

[7] 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.

[8] 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.

[9] 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使用量を削減する改良 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.

[10] 町頭優輝，中田明夫,「ヒープメモリ確保・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使用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報告(SS2019), 信学技報 (SS2019–45), pp.25–30, 2020.

[11] M. L. Dertouzos and A. K. Mok, “Multiprocessor On-Line Scheduling of Hard-Real-Time Tasks”, IEEE Tran. on Soft-ware Engineering, Vol.15, No.12, 1989.

[12] 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, Depart-ment of Computer Science, Florida State University, pp.1– 14, 2005.