Vol.2014-SLDM-165 No.1
Vol.2014-EMB-32 No.1
2014/3/15

情報処理学会研究報告
IPSJ SIG Technical Report

環境電力駆動組込みシステムの動作シナリオのモデル化と動作品質最大化アルゴリズム

青野 和巳[1,a)]　岩田 淳[1]　高瀬 英希[1]　高木 一義[1]　高木 直史[1]

概要：環境発電で駆動する組込みシステムは，電力を得られる機会が安定している場合は，恒久的にシステムを稼働させることが可能である．しかし，その機会が安定していない場合には適切な消費電力管理が必要となる．本稿では，まず，環境発電のシナリオが既知の下で，環境電力で駆動する組込みシステムを，ある一定周期の実行候補点で実行するかを選択するものとしてモデル化する．ただし，システムの安定稼働を保証するため，最大実行間隔のうちで必ず 1 回は実行されるものとする．さらに，モデル化したシステムの動作品質を最大化するアルゴリズムを提案する．ここで，動作品質とは，実行回数と定義する．アルゴリズムは，最大実行間隔およびバッテリ容量の制約を満たすように，実行候補点から可能な限り多く実行点を求める．提案するアルゴリズムは最適な動作シナリオが求められることを証明し，さらに実験により全数探索よりも大幅に高速にそれが求められることを示す．

An Operation Scenario Model for Energy Harvesting Embedded Systems and an Algorithm to Maximize the Operation Quality

Aono Kazumi[1,a)]　Iwata Atsushi[1]　Takase Hideki[1]　Takagi Kazuyoshi[1]　Takagi Naofumi[1]

Abstract: Energy harvesting embedded systems may work permanently when the chance to harvest energy is steady. However, operation strategies to utilize the limited energy are needed when the chance is not steady. In this report, we present an operation scenario model for these systems defined as a set of execution points selected from periodical candidate points to execute the application task. In order to ensure the system stability, the application has to be executed at least once within a given maximum interval. We define the operation quality as the execution frequency . We propose an efficient algorithm to maximize the operation quality, i.e., to find a maximum set of execution points. The proposed algorithm selects as many execution points as possible while satisfying the constraints of the battery amount and the system stability. We describe a proof to show that we can find an optimum solution by the proposed algorithm. We show that the proposed algorithm is much faster than exhaustive search by evaluation using sample scenarios.

1. はじめに

無線通信端末などの携帯型の組込みシステムは，一般的にバッテリを持ち，バッテリから供給される電力によって駆動する．近年，バッテリの電力量を補うもの，さらにはバッテリに代わる電力源として，環境電力が注目されている．環境電力とは，外部環境や機器の動作から副次的に発生する再生可能エネルギーから生み出される電力である．電力を得ることのできる機会が安定している場合には，環境電力のみによって組込みシステムを恒久的に駆動させることが可能である [1].

これまでの消費電力管理に関する研究で対象とされてきたシステムは，ワイヤレスセンサ・ノード端末のような要求される処理が単純なものであった [2]．しかし，マルチメディア系の携帯型情報端末のような組込みシステムについては，環境電力の応用は未だ検討されていない．これらの組込みシステムは，複雑な処理が要求されるため消費電力がより大きくなる [3]．このため，従来研究における環境電力の管理手法ではシステムを恒久的に稼働できないおそれがある．また，環境電力による恒久的な稼働を保証し

[1] 京都大学
Kyoto University
[a)] aono@lab3.kuis.kyoto-u.ac.jp

Vol.2014-SLDM-165 No.1
Vol.2014-EMB-32 No.1
2014/3/15

た上で，動作品質を可能な限り高くすることが望まれる．

本稿では，アプリケーションの実行が一定周期でない組込みシステムに適用できる，環境電力で駆動することを前提とした動作シナリオのモデルを確立する．さらに，そのシステムモデルにおける動作品質最大化問題を定義する．ここで，システムの動作品質は，実行候補点のうちでアプリケーションが実行される回数とする．

本稿ではさらに，このシステムモデルの上で最適な動作シナリオを高速に求めることができるアルゴリズムを提案する．提案するアルゴリズムは，ある時刻を実行点にすると仮定した場合に，それ以降に制約を満たすことが可能であるか否かを判定する．制約とは，実行間隔が最大実行間隔以内であること，かつ，バッテリ量が上限および下限の範囲で値をとることである．制約を満たすことができる場合は，その時刻を実行点とする．この判定を稼働開始時刻から順に行い，環境電力駆動の組込みシステムにおいて動作品質が最も高くなる動作シナリオを得る．さらに，提案する動作品質最大化アルゴリズムで最適解が得られることを証明する．

本稿の構成は，次の通りである．まず，第 2 節では，環境電力で駆動する組込みシステムの動作シナリオのモデル化を行い，動作品質最大化問題を定義する．第 3 節では，全数探索よりも高速に最適な動作シナリオを求めることのできるアルゴリズムを提案し，その最適性を証明する．第 4 節では，提案したアルゴリズムをサンプルデータに適応して評価し，その結果を考察する．最後に，第 5 節にて，結論として本稿のまとめと今後の方針を述べる．

2. 動作シナリオのモデル化

本節では，まず，環境電力で駆動する組込みシステムの動作シナリオをモデル化する．モデル化の際には，従来研究で対象とされているシステムよりも複雑な処理が要求されるものにも対応できるようにする．さらに，提案するシステムモデルにおける動作品質最大化問題を定義する．

2.1 対象システム

本稿の対象とする組込みシステムは，環境発電素子とバッテリを持つ．環境発電素子は，時間とともに変動する発電量が，稼働開始時刻において既知であるものとする．また，バッテリは，蓄電できる容量に上限はあるが，充電効率 η は 1 であり，リーク電流の無いものとする．環境電力は，アプリケーションが実行されない場合は，容量の上限値まで全てバッテリに蓄えられ，実行される場合は直接アプリケーションの実行に使われる．余剰な発電量は，容量の上限値までバッテリに蓄えられる．アプリケーションの実行 1 回にかかる消費電力は，時刻に依らず一定のものとする．

対象システムのアプリケーションには，ある一定周期の実行候補点があるものとする．実行候補点では，アプリケーションを実行するか否かが選択される．システムの安定稼働を保証するため，アプリケーションを与えられた最大実行間隔のうちに必ず 1 回は実行させる．実行点，非実行点，および，動作シナリオを，次のように定める．

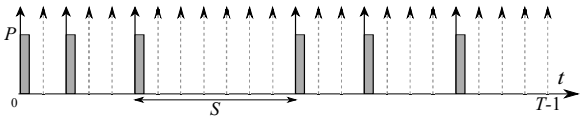

図 **1** 動作シナリオの例

実行点　実行候補点の中から実行すると選択したもの
非実行点　実行候補点の中から実行しないと選択したもの
動作シナリオ　実行点の集合

2.2 モデル化

2.1 節で述べた対象システムについて，その動作シナリオをモデル化する．まず，対象システムの動作シナリオを求めるのに必要な定数を，次のように定義する．

P　アプリケーションの実行 1 回にかかる消費電力
S　アプリケーションの最大実行間隔
B_{max}　バッテリの最大容量
$B[0]$　バッテリの初期値
T　実行候補点の個数

対象システムの実行候補点の周期 Δt を単位時間とする．システムの動作シナリオは，時刻 t を実行候補点の周期 $\Delta t = 1$ で分割することで離散値として考える．t に依存する変数を，次のように定義する．

$H[t]$　環境発電のシナリオ（時刻 t で利用できる発電量）
$B[t]$　時刻 t におけるバッテリ量

決定変数 $A[t]$ を式 (1) で定義する．

$$A[t] = \begin{cases} 1 & (\text{時刻 } t \text{ が実行点のとき}), \\ 0 & (\text{時刻 } t \text{ が非実行点のとき}). \end{cases} \tag{1}$$

動作シナリオ U は式 (2) で与えられる．

$$U = \{t | A[t] = 1\} \tag{2}$$

環境電力が少ないときにも対象システムの安定稼働を保証するため，最大実行間隔 S のうちで必ず 1 回は実行されるものとする．このことは，式 (3) で与えられる．

$$\forall t \in \{0, 1, \ldots, T - S\}, \sum_{i=t}^{t+S-1} A[i] \geq 1 \tag{3}$$

本稿では，動作品質は，アプリケーションの実行回数として定義する．つまり，動作品質 Q は，式 (4) で与えられる．

$$Q = \sum_{t=0}^{T-1} A[t] \tag{4}$$

$A[t] = \{1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0\}$ における動作シナリオの例を図 **1** に示す．図 1 では，動作品質 Q は 5 となる．

実際のアプリケーションを例にして，本モデルの妥当性

Vol.2014-SLDM-165 No.1
Vol.2014-EMB-32 No.1
2014/3/15

を考える．ここでは，監視カメラを環境電力で駆動させることを考える．まず，監視カメラでは，求められる最大実行間隔のうちに必ず 1 回は実行されないと意味をなさないため，安定的な稼働が求められる．ここで，環境から電力が得られる機会が少ないときは，サンプリング頻度を少なくしてフレームレートを下げる，つまり，実行候補点において実行点と選択する機会を少なくし，消費電力を抑える．これにより，安定稼働を保証することができる．環境から電力を得られる機会が多いときは，サンプリング頻度を多くしてフレームレートを上げる，つまり，実行候補点において実行点と選択する機会を多くできる．サンプリング頻度が多いときは，よりなめらかで鮮明な映像を録画できる．つまり，監視カメラの動作品質が高くなることになる．

2.3 動作品質最大化問題

環境電力で駆動する組込みシステムで得られた電力を最大限に使える動作シナリオを得るために，動作品質の最大化問題を定義する．

まず，各変数の値域を以下のように定義する．

$$t \in \{0, 1, \ldots, T-1\} \tag{5}$$

$$H[t] \in [0, \infty] \tag{6}$$

$$B[t+1] \in [0, B_{max}] \tag{7}$$

$$A[t] \in \{0, 1\} \tag{8}$$

バッテリ量 $B[t]$ に関する漸化式を式 (9) に示す．バッテリ量 $B[t]$ の容量に関する制約式を式 (10)，最大実行間隔での動作を保証するための制約式を式 (11) に示す．

$$B[t+1] = \min(B_{max}, B[t] - A[t] \times P + H[t]) \ (0 \le t \le T-1) \tag{9}$$

$$B[t+1] \ge 0 \ \ (0 \le t \le T-1) \tag{10}$$

$$\sum_{i=t}^{t+S-1} A[i] \ge 1 \ \ (0 \le t \le T-S) \tag{11}$$

最大化する目的関数は，式 (4) で与えられる．

動作品質最大化問題の最適解は，安定稼働の制約を満たす動作シナリオの集合，および，バッテリに関して実行可能な動作シナリオの集合の積集合のうち，実行回数が最大となる動作シナリオである．実行候補点の個数 T に対して動作シナリオは 2^T 個存在するため，T が大きくなると，全数探索によって最適解を求めるのは困難となる．

3. 動作品質最大化アルゴリズム

本節では，2.3 で定義した動作品質最大化問題について，その最適解を高速に求めるアルゴリズムを提案する．そして，アルゴリズムの最適性を証明する．

3.1 アルゴリズム

提案するアルゴリズムの方針として，まず，ある時刻における実行候補点の判定について考える．ある時刻を，実行点にすると仮定した場合に，その時刻以降に制約を満たすことが可能であるかを判断する．制約は，式 (10) および式 (11) で与えられる．制約を満たせる場合は，その時刻を実行点にすると判定する．つまり，実行点とすることができるとき，必ず実行点とする．この判定を稼働開始時刻から順に行うことで動作シナリオを得る．

Algorithm 1 Calculate $A[t]$ when Q can be maximized

Given: $H[t],\ B[0] > 0,\ B_{max} > 0,\ T > 0,\ S > 0,\ P > 0$
Require: $\forall t \in \{0, 1, \ldots, T-S\}, \sum_{i=t}^{t+S-1} A[i] \ge 1,\ 0 \le B[t] \le B_{max}$

```
 1: Q ← 0
 2: for t = 0 to T − 1 do
 3:     if t ≥ S ∧ Q < 1 then
 4:         break                                   ▷ No Solution
 5:     else
 6:         if Decision(t, B[0, ..., T]) then
 7:             A[t] ← 1
 8:             Q ← Q + 1
 9:         else
10:             A[t] ← 0
11:         end if
12:         B[t + 1] ← min(B_max, B[t] − A[t] × P + H[t])
13:     end if
14: end for
```

時刻 t^* における実行候補点の判定方法を述べる．ここで，時刻 $0, \ldots, t^*-1$ における部分動作シナリオは，すでに制約を満たしていると判定されていることに注意されたい．時刻 t^* が実行点であるとしたときに，時刻 $t^*, \ldots, T-1$ において，制約を満たす部分動作シナリオが存在すれば，時刻 t^* は実行点と判定できる．

なお，式 (9) より，時刻 t までの動作シナリオが与えられたとき，時刻 t において，$B[t] - A[t] \times P + H[t] > B_{max}$ となったとき，$(B[t] - A[t] \times P + H[t]) - B_{max}$ だけの電力が無駄となってしまう．以下，この電力を損失電力 $W[t]$ と呼ぶ．損失電力 $W[t]$ は，式 (12) で示される．$W[t] > 0$ のときは $B[t+1] = B_{max}$ となる．提案するアルゴリズムでは，損失電力 $W[t]$ に注意して実行候補点の判定を行う．

$$W[t] = \max\{0, (B[t] - A[t] \times P + H[t]) - B_{max}\} \tag{12}$$

提案する動作品質最大化アルゴリズムを，Algorithm 1 および Algorithm 2 に示す．

Algorithm 1 では，時刻 t^* における実行候補点の判定を，稼働開始時刻 0 から $T-1$ まで順に行うことで動作シナリオを得る手順を示している．また，この判定は Algorithm 2 に示す判定関数 Decision$(t', B[0, \ldots, T])$ によって行われる (6 行目)．ここで，$\forall t^* \in \{0, \ldots, S-1\},\ A[t^*] = 0$ であると判定されたときは，制約を満たす動作シナリオは得られないと判定して，停止する (4 行目)．なぜなら，時刻 $\exists t^* \in \{0, \ldots, S-1\},\ A[t^*] = 1$ であると判定されたとき，時刻 $t^*, \ldots, T-1$ について制約を満たした部分動作シナリオが存在することが保証されるためである．

Algorithm 2 では，判定関数 Decision$(t', B[0, \ldots, T])$ が再帰的に呼び出される．Decision$(t', B[0, \ldots, T])$ では，時刻 t' が実行点であり，かつ，時刻 $t', \ldots, t'+S-1$ において制約

Vol.2014-SLDM-165 No.1
Vol.2014-EMB-32 No.1
2014/3/15

Algorithm 2 Decision($t, B[0,\ldots,T]$)

Given: $t,\ B[0,\ldots,T],\ H[t],\ B_{max} > 0,\ T > 0,\ S > 0,\ P > 0$
Require: $0 \le B[t] \le B_{max}$
1: $B[t+1] \leftarrow \min(B_{max},\ B[t] - P + H[t])$
2: $W[t] \leftarrow (B[t] - P + H[t]) - B[t+1]$
3: **if** $B[t+1] < 0$ **then**
4: **return** false
5: **end if**
6: **if** $t \ge T - S$ **then**
7: **return** true
8: **end if**
9: **for** $i = t+1$ **to** $t+S-1$ **do**
10: $B[i+1] \leftarrow \min(B_{max},\ B[i] + H[i])$
11: $W[i] \leftarrow (B[i] + H[i]) - B[i+1]$
12: **if** $W[i] > 0$ **then**
13: **if** Decision($i, B[0,\ldots,T]$) **then**
14: **return** true
15: **end if**
16: **end if**
17: **end for**
18: **return** Decision($t+S, B[0,\ldots,T]$)

を満たした部分動作シナリオが存在するかどうかを判定する．このとき，時刻 $t'+1,\ldots,t'+S-1$ をすべて非実行点としても，安定稼働の制約を満たすことができるため，

$$A[t] = \begin{cases} 1 & (t = t'), \\ 0 & (t = t'+1, t'+1, \ldots, t'+S-1). \end{cases} \tag{13}$$

と仮定する．

まずは，式 (13) で仮定した部分動作シナリオが，バッテリの制約を満たしているかどうかを判定する．バッテリ量が負の値を取りうるのは $B[t'+1]$ のみである．$B[t'+1] < 0$ となったときは，時刻 t' を実行点とすることができないため，Decision($t', B[0,\ldots,T]$) は false を返す（7 行目）．

次に，時刻 $t'+1,\ldots,t'+S-1$ において損失電力が発生するとき，つまり

$$\exists t_i'' \in \{t'+1,\ldots,t'+S-1\}, W[t_i''] > 0 \tag{14}$$

となるような時刻 $t_i''(t_i'' < t_{i+1}'')$ について考える．時刻 $0,\ldots,t_i''-1$ における部分動作シナリオは制約を満たしていることに注意されたい．ここで，時刻 t_i'' を実行点と仮定する判定 Decision($t_i'', B[0,\ldots,T]$) を同様の手順で行う（13 行目）．ここで，Decision($t_i'', B[0,\ldots,T]$) が false を返した場合は，Decision($t', B[0,\ldots,T]$) の判定の続きを再開する．つまり，式 (14) を満たす，新たな時刻 $\forall t_{i+1}'' \in \{t_i'+1,\ldots,t'+S-1\}$ を探索し，Decision($t_{i+1}'', B[0,\ldots,T]$) の判定を行う．以上の手順を，図 **2** に示す．式 (14) を満たすすべての時刻 t_i'' に対して判定関数が false を返す場合は，時刻 $t'+S$ を実行点と仮定する判定を行う（18 行目）．

以上のように，Algorithm 1 において Decision($t^*, B[0,\ldots,T]$) を実行するとき，判定関数は，時刻 t^* 以降の時刻に関して再帰的に呼び出される．ここで，時刻 $t' \in \{T-S,\ldots,T-1\}$ を実行点と仮定する判定では，時刻 $t'+1$ 以降がすべて非実行点であっても，安定稼働の制約を満たすことができる．さらに，時刻 $0,\ldots,t'-1$ における部分動作シナリオは制約を満たしている．よって，時刻 $t'+1$ におけるバッテリ残量 $B[t'+1]$ が負の値を取らなければ，制約を満たした部分動作シナリオが存在すると判定される．このとき，判定関数は true を返すことで呼び出しを終了し，時刻 t^* は，実行点と判定できる（7 行目，14 行目）．また，Decision($t^*, B[0,\ldots,T]$) において false を返すとき，時刻 t^* は実行点と判定できない（4 行目，18 行目）．

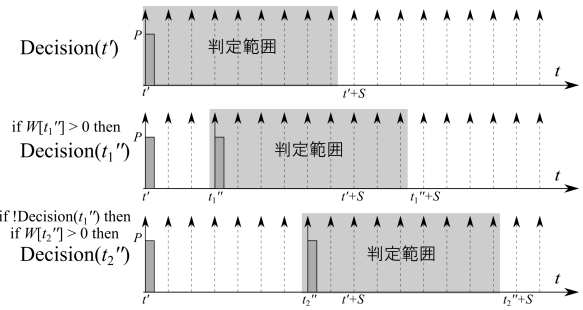

図 2　部分動作シナリオの探索による実行候補点の判定

バッテリの上限が存在しない場合（$B_{max} = \infty$）について，提案したアルゴリズムの効率について考える．このとき，判定関数が呼び出される回数は式 (15) で示される．

$$\sum_{t=0}^{T-1} \left\lfloor \frac{T-t}{S} \right\rfloor \tag{15}$$

このため，計算時間は全数探索で要する指数時間よりも小さい．

3.2　最適性の証明

提案した動作品質最大化アルゴリズムにより，モデル化したシステムでの最適解が得られることを証明する．つまり，以下の定理 1 が正しいことを示す．

定理 1
3.1 節で述べた動作品質最大化アルゴリズムで，システムモデルにおける最適な動作シナリオを得られる．

まず，次の補題 1 を考える．

補題 1 (実行点の後方移動における性質)
n 個の実行点集合のうち $n-1$ 個が等しく 1 個だけ異なる実行点をもつ 2 つの動作シナリオ U および V が

$$U = \{t_1, t_2, \ldots, t_i, \ldots, t_n\} \tag{16}$$
$$V = \{t_1, t_2, \ldots, t_j, \ldots, t_n\} \tag{17}$$
$$t_i < t_j \tag{18}$$

であるとき，各動作シナリオにおけるバッテリ量を $B_U[t]$ および $B_V[t]$ とする．このとき，式 (19) が成立する．

$$\forall t \in \{t_j,\ldots,T-1\},\ B_U[t+1] \ge B_V[t+1] \tag{19}$$

証明 1 (補題 1 の証明)
各動作シナリオにおける損失電力を $W_U[t]$ および $W_V[t]$ と

する．動作シナリオ U および V の時刻 $0,\ldots,t_i-1$ における部分動作シナリオは等しいため，式 (20) が成り立つ．

$$\forall t \in \{0,\ldots,t_i-1\},\ B_U[t+1] = B_V[t+1] \tag{20}$$

$t_i \in U, \notin V$ であるため，式 (9) および式 (20) より，

$$0 \le B_V[t_i+1] - B_U[t_i+1] \le P \tag{21}$$

という関係が成り立つ．続けて，動作シナリオ U および V の時刻 $t_i+1,\ldots,t_j-1$ における部分動作シナリオは等しい．式 (12) および式 (21) より，式 (22) が成り立つ．

$$\forall t \in \{t_i,\ldots,t_j-1\},\ W_U[t] \le W_V[t] \tag{22}$$

ここで，時刻 t_i 以降について場合分けして考える．

(i) $\forall t \in \{t_i,\ldots,t_j\},\ W_V[t]=0$ のとき

式 (22) より，$\forall t \in \{t_i,\ldots,t_j\},\ W_U[t]=0$ である．さらに，$t_j \notin U, \in V$ であるため，式 (23) が成り立つ．

$$B_U[t+1] = \begin{cases} B_V[t+1]+P & (t_i \le t < t_j), \\ B_V[t+1] & (t_j \le t \le T-1). \end{cases} \tag{23}$$

(ii) $\exists t' \in \{t_i,\ldots,t_j-1\},\ W_U[t'] > 0$ のとき

式 (22) より，$W_V[t'] > 0$ である．このとき，

$$B_U[t+1] = \begin{cases} B_V[t+1] = B_{max} & (t=t'), \\ B_V[t+1] & (t' < t \le t_j-1). \end{cases} \tag{24}$$

が成り立つ．さらに，$t_j \notin U, \in V$ であるため，

$$\forall t \in \{t_j,\ldots,T-1\},\ B_U[t+1] \ge B_V[t+1] \tag{25}$$

が成り立つ．

(iii) (i) および (ii) 以外のとき

$$\forall t \in \{t_i,\ldots,t_j-1\},\ 0 \le B_V[t+1] - B_U[t+1] \le P \tag{26}$$

が成り立つ．さらに，$t_j \notin U, \in V$ であるため，

$$B_U[t_J+1] = B_U[t_j] + H[t_J] \le B_{max} \tag{27}$$

$$B_V[t_j+1] = \min(B_{max}, B_V[t_J] - P + H[t_j]) \tag{28}$$

が成り立つ．また，式 (26) より，$0 < B_V[t_j] - B_U[t_j] \le P$ であるため，式 (29) が成り立つ．

$$B_U[t_j+1] \ge B_V[t_J+1] \tag{29}$$

ここで，各動作シナリオの時刻 $t_i+1,\ldots,t_j-1$ における部分動作シナリオは等しいため，

$$\forall t \in \{t_j,\ldots,T-1\},\ B_U[t+1] \ge B_V[t+1] \tag{30}$$

が成り立つ．

よって，全ての場合で式 (19) が成り立つ．

以上より，補題 1 が正しいことが示された． □

以下の定義 1 および定義 2 と，補題 1 を用いて，定理 1 が正しいことを示す．

定義 1 (E 解) バッテリに関する制約を満たす動作シナリオの集合を E 解とする．

定義 2 (I 解) 安定稼働の制約を満たす動作シナリオの集合を I 解とする．

証明 2 (定理 1 の証明)

3.1 節で述べた動作品質最大化アルゴリズムで得られた n 個の実行点をもつ動作シナリオを U，また，m 個の実行点をもつ動作シナリオを $V \in$ E 解 $\cap$ I 解 とする．動作シナリオ U および V は，次式で表される．

$$U = \{t_{U_1}, t_{U_2}, \ldots, t_{U_n}\} \tag{31}$$

$$V = \{t_{V_1}, t_{V_2}, \ldots, t_{V_m}\} \tag{32}$$

$n < m$ と仮定する．$t_{U_i} \in U$ および $t_{V_j} \in V$ を，

$$t_{U_i} = \min\{t \mid t \in U \cap t \notin V\} \tag{33}$$

$$t_{V_j} = \min\{t \mid t \notin U \cap t \in V\} \tag{34}$$

で定義し，t_{U_i} および t_{V_j} について場合分けして考える．

(i) t_{U_i} が存在しないとき（$\forall k \in \{1,\ldots n\},\ t_{U_k} = t_{V_k}$ のとき）

提案したアルゴリズムによって，時刻 $0,\ldots,t_{V_j}-1$ における U の部分動作シナリオ $\in$ E 解 $\cap$ I 解 に対して，時刻 t_{V_j} は実行点と判定されていない．

よって，$t_{V_j} \in V$ であることは，動作シナリオ V が E 解 $\cap$ I 解 であることと矛盾する．

(ii) $t_{U_i} > t_{V_j}$ のとき

(i) と同様に，提案したアルゴリズムによって，時刻 t_{V_j} は実行点と判定されていない．

よって，$t_{V_j} \in V$ であることは，動作シナリオ V が E 解 $\cap$ I 解 であることと矛盾する．

(iii) $t_{U_i} < t_{V_j}$ のとき

動作シナリオ U における時刻 t_{U_i} における実行点を，より後方の時刻 t_{V_j} に変更した動作シナリオ U' は次式で表される．

$$U' = \{t_{U_1}, t_{U_2}, \ldots, t_{U_{i-1}}, t_{U_{i+1}}, \ldots, t_{V_j}, \ldots, t_{U_n}\} \tag{35}$$

動作シナリオ U' におけるバッテリ量を $B_{U'}[t]$ とすると，補題 1 より，式 (36) が成り立つ．

$$\forall t \in \{t_{V_j}+1,\ldots,T-1\},\ B_U[t] \ge B_{U'}[t] \tag{36}$$

ここで，動作シナリオ U' および V に対して，$t_{U'_i}$ および $t_{V'_j}$ を式 (33) および式 (34) で同様に定義する．

この定義を (i) または (ii) の場合となるような $t_{U_i^{(n)}} \in U^{(n)}$ および $t_{V_j^{(n)}} \in V$ が得られるまで繰り返し行う．この繰り返しの間，常に式 (37) が成り立つ．

$$\forall t \in \{t_{V_j^{(n)}}+1,\ldots,T-1\},\ B_U[t] \ge B_{U^{(n)}}[t] \tag{37}$$

時刻 $0,\ldots,t_{V_j^{(n)}}-1$ における $U^{(n)}$ の部分動作シナリオおよび V の部分動作シナリオは等しい．よって，動作シナリオ V が E 解$\cap$I 解 であるとき，時刻 $0,\ldots,t_{V_j^{(n)}}-1$ における $U^{(n)}$ の部分動作シナリオも E 解 $\cap$ I 解 である．このとき，時刻 $t_{V_j^{(n)}}$ における実行候補点の判定を考える．式 (37) より，動作シナリオ $U^{(n)}$ について時刻 $t_{V_j^{(n)}},\ldots,T-1$ で利用することのできる電力は，動作シナリオ U が時刻 $t_{V_j^{(n)}},\ldots,T-1$ で利用することのできる電力以下である．

一方で，提案したアルゴリズムによって，時刻 $0,\ldots,t_{V_j^{(n)}}-1$ における U の部分動作シナリオ $\in$ E 解 $\cap$ I 解 に対して，時刻 $t_{V_j^{(n)}}$ は実行点になり得な

Vol.2014-SLDM-165 No.1
Vol.2014-EMB-32 No.1
2014/3/15

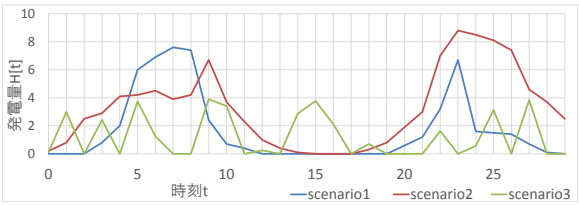

図 3　環境発電のシナリオ

いと判定されている．よって，動作シナリオ $U^{(n)}$ においても，時刻 $t_{V_j^{(n)}}$ は実行点になり得ないと判定される．

以上から，$t_{V_j^{(n)}} \in V$ であることは，動作シナリオ V が E 解 ∩ I 解 であることと矛盾する．

よって，全ての場合で $n \geq m$ となる．

以上より，定理 1 が証明された．　□

4. 評価

4.1 評価環境

本稿で提案した動作品質最大化アルゴリズムをサンプルデータに適応することで，アルゴリズムの有効性を評価した．全数探索および提案するアルゴリズムをそれぞれ Ruby で記述し，そのプログラムに環境発電のシナリオおよびシステムの設定値を入力し，最大の動作品質となる動作シナリオを導出するのに掛かる実行時間を計測した．評価環境としては，以下のような計算機を用いた．OS は Windows7 Professional 64bit SP1，プロセッサは Intel Xeon W3565 3.2GHz，実装メモリは 24GB である．

評価に用いた発電シナリオとして，実際の太陽光発電システムの測定結果 [4] を基に作成したものと，乱数によって作成したものを用いた．図 **3** に示す scenario1，scenario2，および，scenario3 は，それぞれ，冬，夏の測定結果，および，乱数のものである．なお，scenario1 および scenario2 は，実行候補点の周期を 1 時間としてデータを引用した．

評価に用いたシステムの設定値は，表 **1** で示す setting1，setteing2，および，setting3 を用いた．実行候補点の個数 T は 20，25，および，30 の 3 通りで実行した．

4.2 結果

提案したアルゴリズムでは，すべての発電シナリオと設定の組に対して，1μs 未満で単一の最適解を求めることができた．例として，scenario2・setting2 に対して得られた動作シナリオを，図 **4** に示す．

全数探索によって最適解を求めるのに要した計算時間は，以下のとおりであった．$T = 20$ のときは，7.706s から 12.891s，$T = 25$ のときは，281.710s から 1105.945s，$T = 30$ のときは，9744.461s から 83969.909s であった．

表 **1**　システムの設定値

	setting1	setting2	setting3
最大実行間隔 S	3	4	4
バッテリ初期値 $B[0]$	10	10	0
バッテリ上限値 B_{max}	20	20	20
アプリケーションの消費電力 P	3	5	3

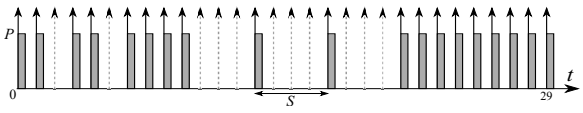

図 4　scenario2・setting2 に対して提案アルゴリズムで得られた解

4.3 考察

実験結果から，提案した動作品質最大化アルゴリズムは，全数探索よりも大幅に高速に最適解を得られることが確認できた．ただし，得られた動作シナリオは単一であり，図 4 から分かるように，実行点が数ヶ所に固まっている．これは，稼働開始時刻から順次実行点となり得る実行候補点を，すべて実行点としたことによるものである．

式 (4) のとおり，本稿では動作品質を合計実行回数と定義した．しかし，システムモデルに応じて目的関数を適切に設定する必要があると考えられる．例えば，最大実行間隔を最小化することが挙げられる．この場合，本稿のアルゴリズムにおいて最大起動間隔 S を 1 ずつ減らすことで，目的関数の最適解を得ることができる．

5. おわりに

本稿では，まず，アプリケーションの実行周期が一定でない組込みシステムに対する，環境電力で駆動するための動作シナリオのモデルを確立した．次に，そのモデルにおける動作品質最大化問題を定義した．さらに本稿では，最適な動作シナリオを高速に求めることができるアルゴリズムを提案した．提案する動作品質最大化アルゴリズムは，実行点の後方移動に関する性質の補題を用いて最適解が得られることを証明した．提案する動作品質最大化アルゴリズムを実装し，サンプルデータに適応することで，アルゴリズムの有効性を評価した．その結果，提案手法は全数探索よりも高速に最適解を導出することが確かめられた．

今後の方針としては，本稿とは異なる動作品質を定義して環境電力駆動の組込みシステムをモデル化することや，発電量が未知である場合にも対応することが挙げられる．

参考文献

[1] Raghunathan, V., et al.: Design Considerations for Solar Energy Harvesting Wireless Embedded Systems, *Proc. of the 4th ISPN*, pp. 457–462 (2005).
[2] Wan, Z., et al.: Review on Energy Harvesting and Energy Management for Sustainable Wireless Sensor Networks, *Proc. of 13th ICCT*, pp. 362–367 (2011).
[3] McIntire, D., et al.: The Low Power Energy Aware Processing (LEAP) Embedded Networked Sensor System, *Proc. of the 5th ISPN*, pp. 449–457 (2006).
[4] 太陽光発電システム測定結果収集システム [online], `http://www1.sbenergy.co.jp/PVMS/` (2014.2.13).