TR-AC-0060

010

高次元アルゴリズムによる 0-1整数変数を含む系の最適化

寺前 裕之　斎藤 茂
柳 正秀*　植草 常雄*

*NTTファシリティーズ

2001. 9. 28

ATR環境適応通信研究所

目　次

高次元アルゴリズムによる
0-1 整数変数を含む系の最適化

寺前裕之、斎藤茂
ATR 環境適応通信研究所
柳正秀、植草常雄
NTT ファシリティーズ

要旨

0-1 整数変数を最適化変数として含む系に対し、最適化手法として高次元アルゴリズムを用いた最適化方法について報告する。実際の最適化問題としては建物エネルギーシステムにおいて、複数のヒートポンプ (空調用熱源) を用いて冷暖房をまかなう場合を考え、ヒートポンプ容量を評価関数として最小値を求めることを試みた。床面積 $10,000m^2$ の標準的なホテルの冷暖房需要データを用いて、各月各時刻における冷暖房の各出力を計算することにより、その最小値と最適な運用法ならびにヒートポンプ数と各々の容量が求まることを示した。このホテルのデータでは、2 台では最小値にならず、最低 3 台が必要となることがわかった。また専用機ならびに PC クラスターを使用して並列処理を試みたが、専用機では並列処理により、6CPU 使用時に最大 4.72 倍程度の加速率が得られることがわかった。

1 はじめに

2 種類の異なった出力を取り出せる装置を使用している場合を考える。この 2 種類の出力が同時には取り出せないとする。装置の例として空調用熱源 (ヒートポンプ) が挙げられる。冷房運転を行っている場合に暖房を同時に取り出すことは不可能である。このように同時出力が出来ない装置を含む系の数値シュミレーションを取り扱う場合には、いずれの出力が得られているかを表す 0-1 整数変数を導入する必要が生じる。

ある時刻 t における装置の異なった運転のモードを A、B とし、装置への入力・出力をそれぞれ x_A, x_B, y_A, y_B とし、装置の出力 A、B に対する変換効率を η_A、η_B として、0-1 整数変数 c_A を導入することで、次式のように表すことができる。

$$y_A = \eta_A c_A x_A \tag{1}$$

$$y_B = \eta_B(1 - c_A)x_B \tag{2}$$

これらの 0-1 整数変数を含めた出力全体に関して何らかの目的関数の最適化を行う場合を考えると、いわゆる混合 0-1 計画問題となる。従来から混合 0-1 計画問題に対しては分枝限定法に代表される厳密解法や [1]、遺伝的アルゴリズム [2]、シュミレーティッドアニーリング [3] といった近似解法が提案されているが、0-1 整数変数の数が多い大規模な問題の場合には効率良く解けるまでには至っていない。

近年、新上らによって提案された最適化手法である高次元アルゴリズムは、古典力学のダイナミクスの手法を用いて最適解を計算する近似解法である [5]。制御変数が多くなっても適用できること、局所解に捕らわれにくいこと、ベクトル計算機や並列計算機向けのアルゴリズムであること、などが特徴として挙げられる。パケットのルーティング最適化問題 [6] や光ファイバーの軸合わせ最適化問題 [7]、JPEG の量子化テーブルの最適化 [8] などに既に応用されており、いずれも良好な結果が得られている。

本論文では、0-1 整数変数を含んだ混合 0-1 計画問題の解法として高次元アルゴリズム [4] を用いる方法を提案し、得られた結果について報告する。具体的にはヒートポンプのみを用いた冷暖房システムをモデルとして取り上げ、ヒートポンプ 2 台以上の機器分割と冷暖房に対する割り当てに関する最適解を計算した。第 2 節ではとりあげたモデルと評価関数について詳しく述べ、第 3 節では高次元アルゴリズムについて述べる。第 4 節ではプログラムの並列処理について述べる。第 5 節では具体的に計算されたヒートポンプシステムの最適値について議論する。

2 モデルと最適化方法

ヒートポンプの最適化問題として冷暖房の割り当て（切替）を表す 0-1 整数変数ならびに各月、各時刻における冷暖房出力の最適化を考える。この場合、エネルギーのフローは以下のように表される。

$$\sum_{k=1}^{n} c_{hw}(m,k) \cdot \eta_{hw} \cdot x_{hw}(m,h,k) = d_w(m,h) \tag{3}$$

$$\sum_{k=1}^{n} (1 - c_{hw}(m,k)) \cdot \eta_{hc} \cdot x_{hc}(m,h,k) = d_c(m,h) \tag{4}$$

切替を表す 0-1 整数変数 $c_{hw}(m,k)$ が導入されている。ここで $d_w(m,h)$, $d_c(m,h)$ は m 月の時刻 h における暖房及び冷房の需要データである。$x_{hw}(m,h,k)$, $x_{hc}(m,h,k)$ は最適化変数でそれぞれヒートポンプ k 番目の暖房・冷房出力である。変数 $c_{hw}(m,k)$ は冷房と暖房の切替を表す変数、0 または 1 とし、1 の場合は暖房を、0 の場合は冷房を表す。計算途中では 0 から 1 までの値を自由に取る事を許し、最終的に 0 または 1 に最適化されるように評価関数を工夫する。時間毎に冷暖房の切り替えを行うのは実際的ではないので、同じ月は同一の出力モードでの運転を課している。そのため

$c_{hw}(m,k)$ は時間 h の関数にはなっていない。n は装置の台数である。一般には 2 種類の異なった需要を同時刻に満たす必要が生じるため装置は複数台必要となる。

簡単のため、ヒートポンプの冷暖房の効率 $\eta_{hw} = \eta_{hc}$ とし、

$$\max_{m,h}[d_w(m,h)/\eta_{hw}, d_c(m,h)/\eta_{hc}] = 1 \tag{5}$$

とする単位系で取り扱うこととし、η_{hw}, η_{hc} を消去する。

$$\sum_{k=1}^{n} c_{hw}(m,k) \cdot x_{hw}(m,h,k) = d_w(m,h) \tag{6}$$

$$\sum_{k=1}^{n} (1 - c_{hw}(m,k)) \cdot x_{hc}(m,h,k) = d_c(m,h) \tag{7}$$

評価関数はヒートポンプ容量の総和とし、これを最小とする最適化問題を解くプログラムを作成した。

$$U_0 = \sum_{k=1}^{n} \max_{m,h}[c_{hw}(m,k) \cdot x_{hw}(m,h,k) + (1 - c_{hw}(m,k)) \cdot x_{hc}(m,h,k)] \tag{8}$$

$$U_1 = a_1 \cdot c_{hw}(m,k)^2 (1 - c_{hw}(m,k))^2 \tag{9}$$

ここで、この評価関数は最小値がわかっているおり理論・プログラムの検証のために選んだ。式 (9) の 4 次関数は最適化終了時に $c_{hw}(m,k)$ を 0 または 1 にするためのペナルティ項として加えた。a_1 は適当な定数で最適化開始時には 0 に近く取っておき、徐々に増加させる。

変数の範囲は、

$$0 \leq c_{hw}(m,k) \leq 1 \tag{10}$$

$$0 \leq x_{hw}(m,h,k) \leq 1 \tag{11}$$

$$0 \leq x_{hc}(m,h,k) \leq 1 \tag{12}$$

である。

さらにいくつかの満たすべき制約条件があるため、これらを評価関数として加えた。式 (3) および式 (4) に与えられた d と x の関係式を最適化終了時に満たすための評価関数として、

$$U_2 = a_2 \sum_{m,h} [\sum_{k=1}^{n} c_{hw}(m,k) \cdot x_{hw}(m,h,k) - d_w(m,h)]^2 \tag{13}$$

$$U_3 = a_3 \sum_{m,h} [\sum_{k=1}^{n} (1 - c_{hw}(m,k)) \cdot x_{hc}(m,h,k) - d_c(m,h)]^2 \tag{14}$$

を加えた。

本来は変数を削減して関係式を厳密に成り立たせるべきだが、これらの関係式には $c_{hw}(m,k)$ および $(1-c_{hw}(m,k))$ が含まれ、変数を削減するためには、0 に近い数値による除算が生じ、誤差が無視できなくなる。そのため上式のような評価関数に対する工夫を行う必要がある。

$c_{hw}(m,k)=0$ なら $x_{hw}(m,h,k)=0$、$c_{hw}(m,k)=1$ なら $x_{hc}(m,h,k)=0$ であることを保証するためのペナルティ項として、

$$U_4 = a_4 | \sum_{m,h} [\sum_{k=1}^{n} x_{hw}(m,h,k) - d_w(m,h)] | \tag{15}$$

$$U_5 = a_5 | \sum_{m,h} [\sum_{k=1}^{n} x_{hc}(m,h,k) - d_c(m,h)] | \tag{16}$$

を加えた。

冷暖房同時負荷の場合、つまり、$d_w(m,h) \neq 0$ かつ $d_c(m,h) \neq 0$ ならば、最低1台のヒートポンプは冷房または暖房にそれぞれ割当てられている必要がある。つまり、

$$1 \leq \sum_{k=1}^{n} c_{hw}(m,k) \leq n-1 \tag{17}$$

この条件を満たすためのペナルティ項として、

$$U_6 = \begin{cases} -a_6 \cdot (\sum_{k=1}^{n} c_{hw}(m,k) - 1) & (\text{for } \sum_{k=1}^{n} c_{hw}(m,k) < 1) \\ 0 & (\text{for } 1 \leq \sum_{k=1}^{n} c_{hw}(m,k) \leq n-1) \\ a_7 \cdot (\sum_{k=1}^{n} c_{hw}(m,k) - (n-1)) & (\text{for } \sum_{k=1}^{n} c_{hw}(m,k) > n-1) \end{cases} \tag{18}$$

以上から最終的な評価関数は、

$$U = \sum_{i=0}^{6} U_i \tag{19}$$

となる。

a_1–a_7 まではパラメーターである。分割数 n によりまた他の条件などによって、適当な値を選ぶ必要がある。

3 高次元アルゴリズム

最適化問題に使用した高次元アルゴリズムを説明する。この方法は、運動の混合性 [13] という性質を利用して、力学系の自律的運動を生成することにより、目的関数の最小値を効率的に探索することを可能にする。すなわち、高次元アルゴリズムでは、最適値探索問題の目的関数値の変化を n 個の変数で指定された自由度を持つ力学系の運動

として表現し、その運動の性質 (混合性) を利用して、目的関数の最小 (最大) 値を探索する力学系の自律的運動を生成することによって最適値探索を高速化するものである。

運動に混合性がある場合、次元数 n が 3 以上になると、運動粒子の位置がポテンシャル最小値に近づくときに運動エネルギの増加とともに滞在時間は増加する。従って滞在時間はポテンシャル最小値で増大することになり、運動粒子がポテンシャル関数の最小値付近に集まりやすくなるという傾向を示すことになる [9]。このような依存性は、粒子の運動の混合性から生じる。混合性は粒子間で相互作用がない場合は、それぞれ単振動するだけであるが、運動エネルギの交換があると軌道が複雑になり、運動として様々な状態をとり得るようになることを言う。そこでポテンシャル関数を最適値探索問題の目的関数とみなし、各運動粒子のポテンシャルが最小に至る自律的運動を生じさせることにより、目的関数最小で決定される解を探索することが可能になる。多粒子系の運動を考えた場合、$2n$ 次元で質点系（運動粒子）の位置を $x_1, \cdots, x_n$、運動量を $p_1, \cdots, p_n$ で表すと、その自律的運動は次式によって得られる、

$$\frac{dx_i}{dt} = \frac{\partial H}{\partial p_i} \tag{20}$$

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial x_i} \tag{21}$$

ハミルトニアンの正準方程式で記述できる。ここで、ポテンシャル (位置エネルギー U) を受けている質量 1 の質点では、ハミルトンの特性関数 H は、次式で与えられる。

$$H = \frac{1}{2}\sum_i p_i^2 + U(x_1, x_2, \cdots, x_n) \tag{22}$$

ここで、ハミルトンの特性関数と運動量を消去すると次式が得られる。

$$\frac{d^2x}{dt^2} = -\frac{\partial U}{\partial x_i} \tag{23}$$

この式は、位置エネルギーから受ける力を加速度とした運動を示している。これをもとに、各変数が評価関数から受ける力をもとに、Verlet 法 [10, 11] により運動を計算し、評価関数の最小値の点に収束させる計算法を高次元アルゴリズムと呼んでいる。我々は複雑なシステムの最適解を求める手法として利用している。Verlet 法に必要となるコスト関数の微分については Appendix にまとめた。

上述のように適当な初期座標と初期速度を与えて、運動を追跡する場合には、時として初期値依存性が生じ、非常に長時間の運動を追跡したにもかかわらず評価関数の最小値付近を通らないという可能性が生じる場合がある。そのような場合には運動に均一性を持たせるような工夫が必要となる。高次元化に際して、仮に加えた運動量に対してミキシングというプロセスを加えた。求めたい最適値は、座標であって運動量ではないため運動量の変形は自由に行って構わない。

$$H = \frac{1}{2}\sum_{i,j} b_{ij} p_i p_j + U \tag{24}$$

適当な正の規格直交行列 b_{ij} を用いて運動量のミキシングを行う。

$$\ddot{x_i} = \frac{\partial H}{\partial p_i} = \sum_j b_{ij} p_j \tag{25}$$

$$p_i = -\frac{\partial H}{\partial x_i} = f_i \tag{26}$$

$$\ddot{x_i} = \sum_j b_{ij} f_j \tag{27}$$

となるので、実際の手順としては高次元アルゴリズムにおける各最適化変数にかかる力をミキシングすることになる。ここで b_{ij} は先に述べたように、正の規格直交行列、言いかえると正の固有値を持つ行列である必要がある。固有値が負となると、運動エネルギーが負の値を取ることが可能となり、無限遠へ運動してしまう可能性が出てしまい、最適化は失敗する。

ミキシングのための行列の生成法を以下に示す。実対称行列$\boldsymbol{A}$を疑似乱数によって作り対角化。その固有ベクトルを$\boldsymbol{C}$、固有値を対角要素とする行列を$\boldsymbol{\epsilon}'$ とすると、

$$\boldsymbol{CAC^T} = \boldsymbol{\epsilon}' \tag{28}$$

この固有ベクトル$\boldsymbol{C}$を利用して、ミキシングのための行列$\boldsymbol{B}$を作る。行列が positive definite になるように固有値を 1 近傍に値を分散させて選ぶ。

$$\epsilon_j = 1 + \delta_j \quad (\delta_j > -1) \tag{29}$$

以上から求める行列$\boldsymbol{B}$の満たすべき固有方程式は、次式のようになる。

$$\boldsymbol{BC} = \boldsymbol{C\epsilon} \tag{30}$$

ここで$\boldsymbol{C}$は定義より直行行列なので$\boldsymbol{CC^T} = \boldsymbol{1}$、右から$\boldsymbol{C^T}$を乗じることにより、

$$\boldsymbol{B} = \boldsymbol{C\epsilon C^T} \tag{31}$$

が求めるミキシングのための行列になる。δ の値を増減する事でミキシングの度合いをコントロールできるのでミキシングの度合いは対象とする系により適当な量を選ぶ。

4 プログラミングと並列処理

以上に述べてきた関係式を用いて実際にプログラムを作成した。ヒートポンプの分割数が増加してくると計算量がかなり増大してくることが見込まれたため、ループ変数としての月を利用することで並列処理を行った。ここでは並列化ライブラリとして MPI を用い図 1および図 2に示すような並列処理のプログラムを作成した。なおノード間の通信量を極力少なくするために、月を変数として並列処理を行ったため、最大でも 6 並列までにとどめた。ミキシングを行う場合には、図 2から容易にわかるように、ベクトル演算が大量に発生するため、並列化による大幅なパフォーマンス向上が期待できる。

実際の計算機環境としては以下の 2 種類を用いた。

```
      implicit real*8(a-h,o-z)
      parameter (m = 12, ih = 24)
      dimension xhw(m,ih,maxhp),xhc(m,ih,maxhp),cw(m,maxhp)
      dimension xhmax(m)
      include 'mpif.h'
      call mpi_init(ierr)
      icomm=mpi_comm_world
      call mpi_comm_rank(icomm,me,ierr)
      call mpi_comm_size(icomm,nproc,ierr)
      ...
      calculate the initial geometry and velocity
      ...
      if(imix.eq.1) then
        calculate mixing matrix
      endif

      do loop=1,nloop
      calculate xhmax=max(cw*xhw+(1-cw)*xhc)
      call mpi_allreduce(xhmax,xhmax,nhp,mpi_double_precision,
     1 mpi_max,icomm,ierr)
      do k=1,nhp
        do i=1,m
           if(mod(i,nproc).eq.me) then
            do j=1,ih
              calculate first derivative of cw, xhw, xhc
            enddo
           endif
        enddo
        enddo
        if(imix.eq.1) then
c        calculate mixing of derivatives
        call mpi_allreduce(dcw,dcw,m*nhp,mpi_double_precision,
     1 mpi_sum,icomm,ierr)
        call domix(vdcw,dcw,vscr,m,nhp)
        call mpi_allreduce(dxhw,dxhw,inum,mpi_double_precision,
     1 mpi_sum,icomm,ierr)
        call mpi_allreduce(dxhc,dxhc,inum,mpi_double_precision,
     1 mpi_sum,icomm,ierr)
        call domix2(vdxhw,dxhw,vscr1,m,ih,nhp)
        call domix2(vdxhc,dxhc,vscr1,m,ih,nhp)
       endif
       do k=1,nhp
       do i=1,m
          if(mod(i,nproc).eq.me) then
           do j=1,ih
             calculate next geometries and total energy
           enddo
          endif
      enddo
      enddo
      enddo
      call mpi_finalize(ierr)
      stop
      end
```

図 1: 並列化の概要 (1)

```
subroutine domix2(vdxhw,dxhw,vscr,m,ih,nhp)
include 'mpif.h'
integer status
common/pproc/nproc,me,icomm,status(mpi_status_size)
dimension vdxhw(m,ih,m,ih,nhp),dxhw(m,ih,nhp),vscr(m,ih,nhp)
do k=1,nhp
do i=1,m
if(mod(i,nproc).eq.me) then
do j=1,ih
vscr(i,j,k)=0.0d0
   do i1=1,m
   do j1=1,ih
     vscr(i,j,k)=vscr(i,j,k)+dxhw(i1,j1,k)*vdxhw(i1,j1,i,j,k)
   enddo
   enddo
 enddo
endif
enddo
enddo
do k=1,nhp
do i=1,m
if(mod(i,nproc).eq.me) then
do j=1,ih
dxhw(i,j,k)=vscr(i,j,k)
enddo
enddo
endif
enddo
return
end
subroutine domix(vdcw,dcw,vscr,m,nhp)
implicit real*8(a-h,o-z)
include 'mpif.h'
integer status
common/pproc/nproc,me,icomm,status(mpi_status_size)
dimension vdcw(m,m,nhp),dcw(m,nhp),vscr(m)
do k=1,nhp
do i=1,m
if(mod(i,nproc).eq.me) then
   vscr(i)=0.0d0
      do j=1,m
         vscr(i)=vscr(i)+dcw(j,k)*vdcw(j,i,k)
      enddo
   endif
   enddo
 do i=1,m
    if(mod(i,nproc).eq.me) then
        dcw(i,k)=vscr(i)
    endif
 enddo
 enddo
 enddo
return
end
```

図 2: 並列化の概要 (2)

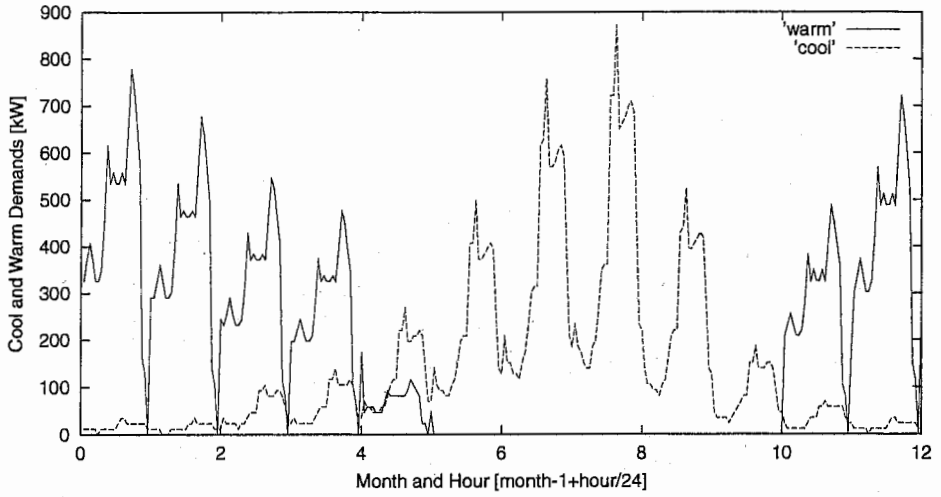

図 3: 延べ床面積 $10,000m^2$ の標準的なホテルの冷暖房負荷データ

1. 並列計算機 DEC Alpha Server 21164A 612MHz 8CPU 8GB メモリー、MPI ライブラリ DMPI(DEC 製)

2. PC クラスター Intel Pentium3 450MHz 6CPU、100BaseT イーサネットボード、FreeBSD 4.2Release、MPI ライブラリ LAM Ver. 6.2β[12]

上記、(1) は共有メモリー型の並列計算専用機であり、(2) は通常のパーソナルコンピューターをイーサネットを介してクラスター構成としたものである。

5 結果と考察

冷暖房負荷データとして延べ床面積 $10,000m^2$ の標準的なホテルのデータを使用して、計算された最適値を表 1に示す。またここで用いた標準的なホテルの負荷データは図 3 に示した。このホテルの負荷データでは、6-10 月は冷房負荷のみであるが、他の月は冷暖房負荷が同時に発生するという特徴的な負荷パターンとなっている。

高次元アルゴリズムにより計算された、2 分割時の各ヒートポンプの出力状況を図 4に示した。ここで重要なのは、ヒートポンプ 1 は 5 月のみ暖房に割り当てられ、他の月は全て冷房に割り当てられていることである。この冷暖房の切り替えを行うことにより、より大きな容量が必要となるのを回避している。実際この切り替えを考慮せず、ヒートポンプ 1 が常に冷房に割り当てられたとすると、ヒートポンプ容量は 1.0533 に対して 1.3 となる。ここでの容量の単位系は式 (5) に与えられている。なお最適化の初期値としてはこのような切り替えを意識していないものを用いた。ただし 2 台のヒートポンプでは切り替えによってこれ以上のヒートポンプ容量の節約は行えないが、より大きい分割数では、理論的最小値 (標準的ホテル負荷データでは 8 月の冷房最大負荷値)

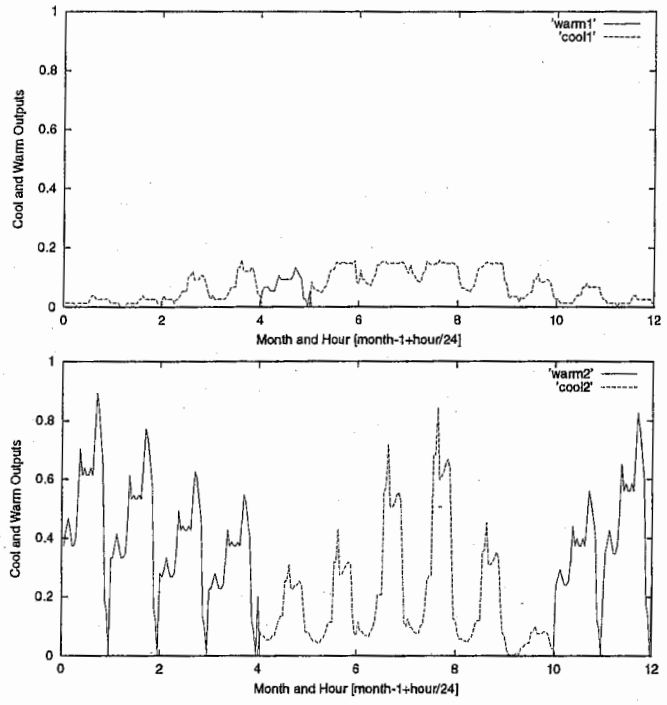

図 4: 2 分割時の各ヒートポンプの冷暖房出力

を取ることができ、この最小値は 1 となる。ここで用いた負荷パターンでは 3 分割以上でこの最小値をとることがわかる。なお理論最小値とは、冷暖房負荷の和の最大値であり、これを下回ると熱需要をまかなえなくなる。

実際の高次元アルゴリズムによる計算値は数値的な誤差を含むため理論値と完全には合致しないが、表 1に示したように 10 分割までについて最適化を行い、$n = 2$ の時に 1.0523、$n = 3 - 10$ の時 0.9995 または 0.9996 となっており、ほぼ一致しているとみなすことができよう。このうち計算された 3 分割時の各ヒートポンプの出力状況を図 5に示した。以上から高次元アルゴリズムによる最適化がうまく働いて最小値が求まっていることがわかる。

なお計算に必要となるパラメーター値 $a_1 - a_7$ については表 2にまとめた。第 2 節で述べたように、最適化開始時と終了時で値が異なるパラメーターは別に示した。また、ミキシング定数の最大値は 0.05 として各々の δ_j はこの値を越えないように乱数により生成した。3 台分割時にミキシングを行った場合と行わなかった場合について、表 3に示した。表 3から容易にわかるように、ミキシングのプロセスを行わなかった場合には、同じ 20 万回の最適化の繰り返し計算を行った後でも、まだ最適解に収束していない。ミキシングを行うことによって局所解近傍で一種のカオス的な運動が導入され [13]、局所解から抜け出すきっかけが与えられていると考えられる。

表 4および表 5に並列計算によるパフォーマンス向上について示した。ここではヒー

表 1: 高次元アルゴリズムによるヒートポンプ最小値

台数 n	理論最小値	最小値
2	1.0533	1.0523
3	1.0	0.9995
4	1.0	0.9996
5	1.0	0.9995
6	1.0	0.9996
7	1.0	0.9996
8	1.0	0.9996
9	1.0	0.9996
10	1.0	0.9995

表 2: 評価関数中のペナルティー項のパラメーター値

パラメーター	最適化開始時	最適化終了時
a_1	0.2	5
a_2	5	2000
a_3	5	2000
a_4	0.05	0.2
a_5	0.05	0.2
a_6	0.5	—
a_7	0.5	—

表 3: ミキシングによる計算された最小値の変化

ミキシング定数 δ	最小値
0.0	1.1299
0.05	0.9995

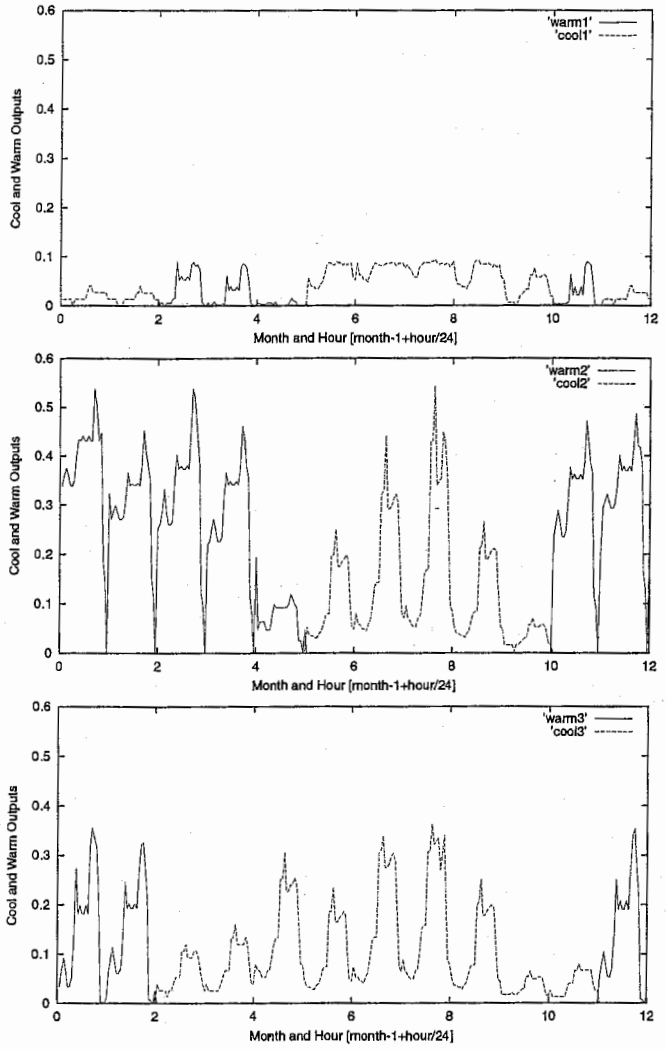

図 5: 3 分割時の各ヒートポンプの冷暖房出力

トポンプは 3 分割とし、繰り返し計算は 20 万回行った。表 4においては専用の並列計算機である Alpha サーバーの結果について示している。ミキシングを行った場合には、大規模な行列計算が必要となるため、並列化により、6CPU 使用時に 4.72 倍の加速率が得られており、大きな効果が得られていることがわかる。一方、ミキシングを行わなかった場合には、今回の計算ではコスト関数が非常に簡単であるため、ミキシングを行った場合ほどの効果ではないが、6CPU 使用時に加速率 1.63 倍と若干の並列化の効果が得られていることがわかる。ミキシングを行わなかった場合でも、評価関数がより複雑となった場合にはより大きな効果が得られることが期待される。

表 5では、PC クラスターによる結果を示したが、PC クラスターでは専用機の場合よりは得られる効果は小さい事がわかった。原因としては Alpha サーバーのように共有メモリーではなく、速度の遅いネットワーク経由でデータのやり取りをしているた

表 4: Alpha サーバーにおける並列計算によるパフォーマンスの向上

CPU 数	CPU 時間	加速率
ミキシング無し		
1	290	1.00
2	230	1.26
6	177	1.63
ミキシング有り		
1	7357	1.00
2	3861	1.95
6	1558	4.72

表 5: PC クラスターにおける並列計算によるパフォーマンスの向上

CPU 数	CPU 時間	加速率
ミキシング無し		
1	306	1.00
2	273	1.12
6	301	1.02
ミキシング有り		
1	3457	1.00
2	2193	1.58
6	1810	1.91

め、ネットワーク I/O 待ちが大きくなることが考えられる。図 1からわかるように、繰り返し計算一回あたりについて一度のデータのやり取りが行われる必要があり、このデータのやり取りに比して、並列化しうる計算量が少ないために、ネットワーク経由ではこのオーバーヘッド処理の時間が無視できなくなっているものと思われる。

以上のように本論文では、評価関数にペナルティー項を加え、最適化方法として高次元アルゴリズムを使用する事により 0-1 整数変数を含めた系の最適化を試みた。実際の最適化は建物エネルギーシステムにおいて、ヒートポンプ (空調用熱源) のみを用いて冷暖房をまかなう場合を考え、ヒートポンプ容量を評価関数として最小値を求めることを試みた。床面積 $10,000m^2$ の標準的なホテルの冷暖房需要データを用いて、その最小値と最適な運用法ならびに最適な分割数が求まることを示した。このホテルの需要データではヒートポンプが 2 台ではどのような運用法でも最小値をとることはできず、最低 3 台以上が必要とされることがわかった。

また専用の並列計算機ならびにPCクラスターを使用して並列処理を試みたが、専用機では並列処理により、6CPU使用時に最大4.72倍程度の加速率が得られることがわかった。今後より大規模な計算を行う際には並列処理を試みるのは有望であろう。

今後はヒートポンプ以外の熱源機器を用いた系であるとか、他の評価関数を用いた場合への適用が考えられ、現在検討中である。また本論文の結果を遺伝的アルゴリズムやシュミレーティッドアニーリングなどの他の近似解法から得られる結果との比較は大変興味深いと思われる。

参考文献

[1] R. S. Gerfinkel, G. L. Nemhauser, "Integer Programming", John Willey (1972).

[2] D. E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison-Wesley (1989).

[3] S. Kirkapatrck, *Science*, **220**, pp671-680 (1983).

[4] K. Shinjo, T. Sasada, *Phys. Rev.*, **E54**, 4686-4700 (1996).

[5] K. Shinjo, S. Shimogawa, J. Yamada, K. Oida, *Int. J. Mod. Phys.*, **C10**, 63-94 (1999).

[6] K. Oida, K. Shinjo, 情報処理学会論文誌, **Vol.40 No.SIG9**, 42 (1999).

[7] 水上雅人, 平野元久, 新上和正, 計測自動制御学会論文集, **33**, 709-715 (1997).

[8] K. Hirata, J. Yamada, K. Shinjo, *SPIE*, **3648**, 344-351 (1998).

[9] 新上和正、"高次元系のハミルトニアンダイナミクス"、物性研究、pp.269-282(1992).

[10] L. Verlet, "Computer Experiments on Classical Fluids. I. Thermodynamical Properties of Leonard-Jones Molecules.", *Phys. Rev.*, **159**, 98 (1967).

[11] C. W. Gear, "Numerical Initial Value Problems in Ordinary Differential Equations", Prentice-Hall, Englewood Cliffs, NJ (1971).

[12] G. D. Burns, R. B Daoud, J. R. Vaigl, Supercomputing Symposium '94 (Toronto, Canada June 1994); G. D. Burns, R. B. Daoud, MPI Developers Conference, University of Notre Dame (June 1995).

[13] G.M.Zaslavsky, 三島信彦訳:"カオス－古典及び量子力学系－", 現代工学社 (1989).

Appendix

ここでは Verlet 法で必要となる評価関数の微分について述べる。

$$\frac{\partial U}{\partial c_{hw}(m,k)} = \frac{\partial U_0 + \partial U_1 + \partial U_2 + \partial U_3 + \partial U_6}{\partial c_{hw}(m,k)} \tag{32}$$

$$\frac{\partial U}{\partial x_{hw}(m,h,k)} = \frac{\partial U_0 + \partial U_2 + \partial U_4}{\partial x_{hw}(m,h,k)} \tag{33}$$

$$\frac{\partial U}{\partial x_{hc}(m,h,k)} = \frac{\partial U_0 + \partial U_3 + \partial U_5}{\partial x_{hc}(m,h,k)} \tag{34}$$

ここで、

$$v(k) = \max_{m,h}[c_{hw}(m,k)x_{hw}(m,h,k) + (1 - c_{hw}(m,k))x_{hc}(m,h,k)] \tag{35}$$

を与える m, h を $m^*(k), h^*(k)$ とする。これ以外の m, h に関する U_0 の偏微分は 0 である。

$$\frac{\partial U_0}{\partial c_{hw}(m^*(k),k)} = x_{hw}(m^*(k),h^*(k),k) - x_{hc}(m^*(k),h^*(k),k) \tag{36}$$

$$\frac{\partial U_0}{\partial x_{hw}(m^*(k),h^*(k),k)} = c_{hw}(m^*(k),k) \tag{37}$$

$$\frac{\partial U_0}{\partial x_{hc}(m^*(k),h^*(k),k)} = 1 - c_{hw}(m^*(k),k) \tag{38}$$

$$\frac{\partial U_1}{\partial c_{hw}(m,k)} = 2a_1(c_{hw}(m,k) - 3c_{hw}(m,k)^2 + 2c_{hw}(m,k)^3) \tag{39}$$

$$\frac{\partial U_2}{\partial c_{hw}(m,k)} = 2a_2[\sum_{i=1}^{n} c_{hw}(m,i)x_{hw}(m,h,i) - d_w(m,h)]x_{hw}(m,h,k) \tag{40}$$

$$\frac{\partial U_2}{\partial x_{hw}(m,h,k)} = 2a_2[\sum_{i=1}^{n} c_{hw}(m,i)x_{hw}(m,h,i) - d_w(m,h)]c_{hw}(m,k) \tag{41}$$

$$\frac{\partial U_3}{\partial c_{hw}(m,k)} = -2a_3[\sum_{i=1}^{n}(1 - c_{hw}(m,i))x_{hc}(m,h,i) - d_c(m,h)]x_{hc}(m,h,k) \tag{42}$$

$$\frac{\partial U_3}{\partial x_{hc}(m,h,k)} = 2a_3[\sum_{i=1}^{n}(1 - c_{hw}(m,i))x_{hc}(m,h,i) - d_c(m,h)](1 - c_{hw}(m,k)) \tag{43}$$

$$\frac{\partial U_4}{\partial x_{hw}(m,h,k)} = \begin{cases} -a_4 & (\text{for } \sum_{i=1}^{n} x_{hw}(m,h,i) - d_w(m,h) < 0) \\ 0 & (\text{for } \sum_{i=1}^{n} x_{hw}(m,h,i) - d_w(m,h) = 0) \\ a_4 & (\text{for } \sum_{i=1}^{n} x_{hw}(m,h,i) - d_w(m,h) > 0) \end{cases} \tag{44}$$

$$\frac{\partial U_5}{\partial x_{hc}(m,h,k)} = \begin{cases} -a_5 & (\text{for } \sum_{i=1}^{n} x_{hc}(m,h,i) - d_c(m,h) < 0) \\ 0 & (\text{for } \sum_{i=1}^{n} x_{hc}(m,h,i) - d_c(m,h) = 0) \\ a_5 & (\text{for } \sum_{i=1}^{n} x_{hc}(m,h,i) - d_c(m,h) > 0) \end{cases} \tag{45}$$

$$\frac{\partial U_6}{\partial c_{hw}(m,k)} = \begin{cases} -a_6 & (\text{for } \sum_{i=1}^{n} c_{hw}(m,i) < 1) \\ 0 & (\text{for } 1 \leq \sum_{i=1}^{n} c_{hw}(m,i) \leq n-1) \\ a_7 & (\text{for } \sum_{i=1}^{n} c_{hw}(m,i) > n-1) \end{cases} \tag{46}$$