数理解析研究所講究録 1409 巻 2005 年 167-180

最適意思決定問題のDNAコンピューティングによる解法

早稲田大学　大学院　情報生産システム研究科
和多田淳三 (Junzo Watada)、小島智 (Satoshi Kojima)
Graduate School of Information, Production and Systems
Waseda University

1. はじめに

近年、最適化問題、特に組み合わせ最適化問題の解法が種々研究されている。これらの問題の難しさは、解法の効率化を図ったとしても解を得るための計算量が膨大にならざるを得ない。このため、計算機アルゴリズムでは並列計算法が研究されており、ハードウエアーの開発が行われており、von Neuman 型計算機の限界を超える努力がなされている。

本論分では、DNA のシークエンシングの機能を用いて解を説く方法が、1994 年にアドルマンがハミルトン問題を解いたことで、関心が高まっている。本論分では、DNA によるコンピューティング手法を用いて組み合わせ型最適化問題の一例として、エレベータの群管理を実現する試みを行った。もちろん DNA コンピューティングの問題点は入出力の部分がマニュアルベースで必ずしも、一般に言われている計算機とは現段階では距離がある。

建築物の高層化が進んでいる現在、階層間の移動手段としてエレベータが広く利用されている。規模の大きな建築物では複数台のエレベータが設置されていることが多く、それらの効率的な運行が求められている。

効率的なエレベータの運行を決定する問題はエレベータ郡管理システムと呼ばれており、利用者全体の満足度を高くし、システム運用効率を高くすることを目的として、利用者をどのようにエレベータに乗せるか、エレベータをどのような順序で適切な階に停止させるかを決定する。

近年では、人工知能付きエレベータの開発など改善が進んでいる。しかしながら、例えば、朝のラッシュ時など４機が４機共、同じ方向に向かったり、4 機が同時に来るといった効率の悪さを改善する必要性がある。これらの問題を解消するためには、時々刻々発生する乗場呼びに対して最適なエレベータを割り当てることにあるが、このほかにも郡管理システムは交通流の変動に応じた運転パターンの選択や災害時の管理運転など幅広い機能を持ち、快適で安全かつ経済的なエレベータ運行を実現している[5]。

2. 群管理システム

1 階から N 階までの建設物があり、エレベータの台数をｍとする。このとき、ある時点ｔでの、エレベータの位置、エレベータ内部の乗客の行きたい階、エレベータ外部の乗客の行きたい方向を表すことができる。

つまり、入力情報は

(1) エレベータ 1～ｍの現在の位置
(2) エレベータ内にある目的階を押す（籠呼び）
(3) エレベータ外からの要求（乗場呼び）

である。これらの情報を基にいかに効率よくエレベータを運行させるかを考える。

定義

Ⅰ．エレベータの動作の制限

(1) エレベータ内の乗客の目的地にすべて到着するまで、(エレベータが空の状態になるまで) 一定方向に向かうものとする。つまり、エレベータ内に人が乗っている場合、乗客の目的地と

逆方向には向かうことができない。
(2) エレベータの定員などの乗車制限は設けないこととする。
(3) 上昇と下降が同じ階数移動するなら移動時間は同じであるとする。
Ⅱ. エレベータが行う作業
(1) 階iから階jへの移動（$1 \leq i < j \leq N$）
(2) 階jに到着後扉を開き、目的の階が押され、閉ボタンが押されるのを待つか一定の時間が経過すれば自動的に扉を閉める。
(3) 次の階に移動する。
Ⅲ. 記　号
時刻tにおいて、待ち行列とエレベータの初期位置が与えられたとき、待ち時間を最短とするm台のエレベータの作業ダイアグラムを決定する。
i ：現在エレベータがある階
j ：次にエレベータが向かう階
T_E ：各階にエレベータが停止している時間
$T(i,j)$ ：エレベータが階i～階jまでの移動時間
とおくと、$T(i,j)=T(j,i)=f(|j-i|)$となる。
結局、エレベータが階i～階jまでの移動時間$T(i,j)$は、j,iの差の関数$f(|j-i|)$で表すことができる。これにより、枝の重みが決定する。

$$\psi_1 = f(1) + T_E$$
$$\psi_2 = f(2) + T_E$$
$$\vdots$$
$$\psi_9 = f(9) + T_E$$

つまり、$|j-i|=k$とおくと

$$\psi_k = f(k) + T_E$$

$1 \leqq k \leqq j-1$と置ける。
ここで、エレベータの動きをグラフで表すと、エレベータが一階にあり、目的の階j（$2 \leqq j \leqq$ N）までの経路はN-1通り存在し、移動時間は$\psi_k = f(k) + T_E$（$1 \leqq k \leqq$N-1）である。
エレベータは2階から3階、5階からN階なども到達可能であり、全方向への経路をグラフ化すると図2.2.4のように表すことができる。

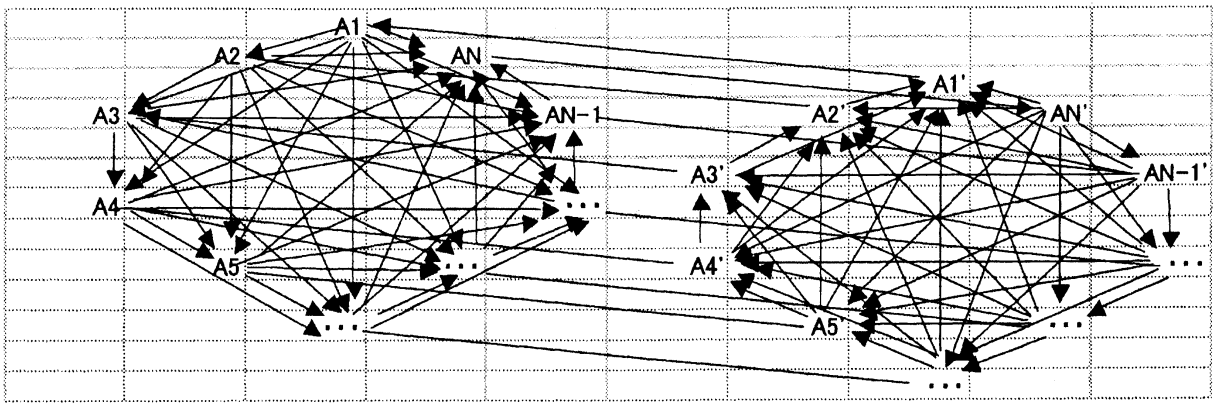


図 2.2.4　エレベータＡの全経路のグラフ

ここで、A1←A1'はエレベータの下から上への一方向転換を表している。逆に、AN→AN'はエレベータの上から下への一方向転換を表している。また、A2⇔A2', A3⇔A3', ・・・・AN-2⇔AN-2', AN-1⇔AN-1' は双方向の方向転換が可能であることを表している。
それぞれの枝の重みは、前述した通り

$$\psi_k = f(k) + T_E \quad (1 \leqq k \leqq \mathrm{N\text{-}1})$$

となる。また、エレベータは一台でなくｍ台存在しているので図 2.2.4 のグラフをｍ個重ねてすべての頂点を直線で結んだグラフを書くことができる。(図 2.2.5 参照)

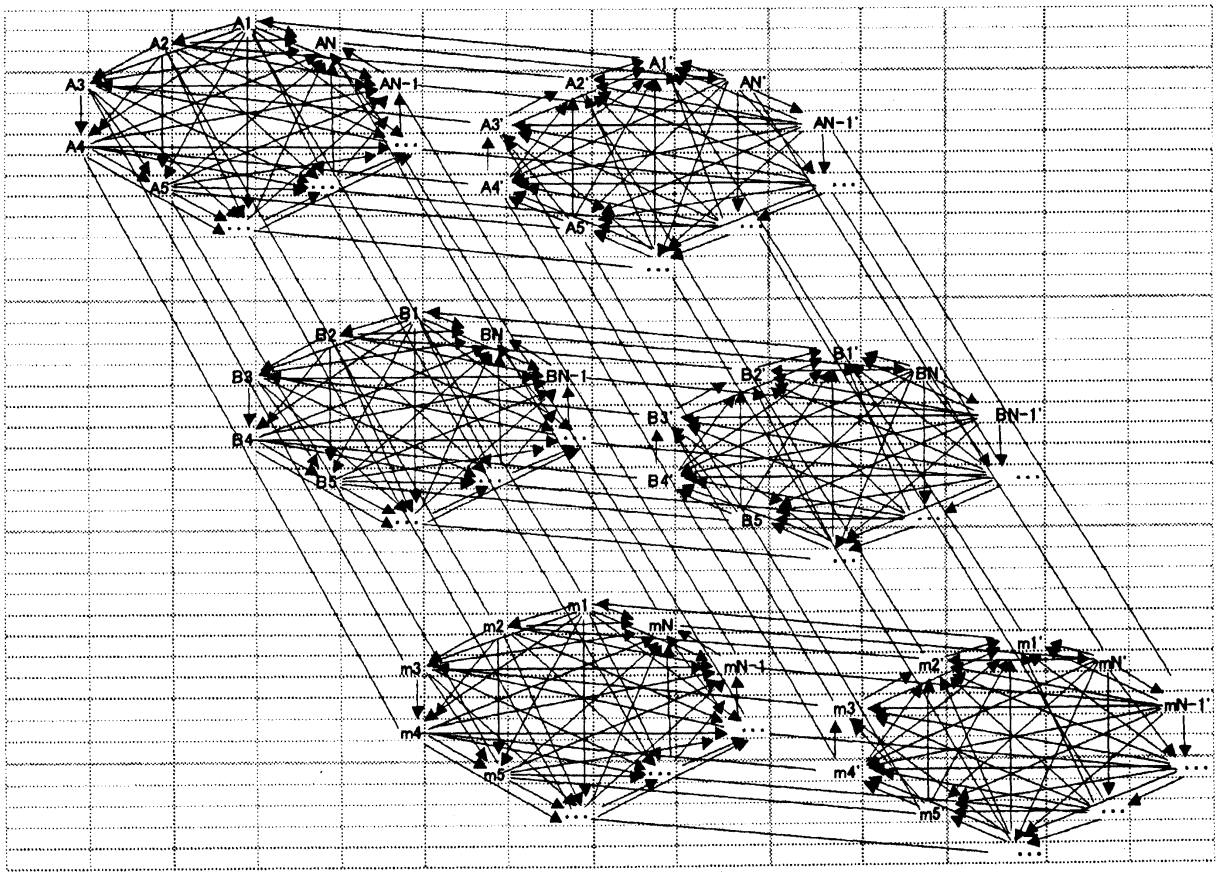


図 2.2.5　エレベータがｍ台の場合のグラフ

グラフ A,B…m の直線上の点にはm台のエレベータの内一台でも外部からの要求階へ行けば良いことになる。エレベータ外部からボタンが押された時、その階にAが行く場合、Bがいく場合、mがいく場合の場合分けを行いグラフを作成する。

そのすべての組み合わせ全てで、グラフ A,B…m の最短経路を計算し、グラフ A,B…m の中で最も大きい値を $f(A,B,\cdots,m)$ とおき、組み合わせの数だけ $f_x(A,B,\cdots,m)$ を計算する。エレベータ A,B…m の最適な割り当ては $f_x(A,B,\cdots,m)$ の最小の値の組み合わせを求めればよい。

階	目的	A↑	B↓
6			2,3
5	↓		
4	↑		
3	↓		
2			
1		3,5	

図 3.1　エレベータの状況

ただし、xは組み合わせの数であり、例えば、エレベータが3台で外部からの要求が3の場合 2^3 通りの組み合わせができる。

3. 最適階の例

建物が6階（1～6階）でエレベータが2基（A・B）で図3.1に示す条件が与えられた時の最適なエレベータのスケジューリングを考える。ある時点tで図 3.1 の条件が与えられたとして最適な運行スケジュールを求める。

エレベータの経路をグラフ化するとAが上方向に向かう場合と、下方向に向かう場合の2種類のグラフが考えられる。

また、上方向から下方向への方向転換（2階から6階）、下方向から上方向への方向転換（1階から5階）も可能であるので、エレベータAの全経路のグラフは、Ai,Aj（$1\leqq i,j\leqq 6$）間の直線または矢印はエレベータの方向転換を表している。ただし、$A1,A1'$と$A6,A6'$間は一方向にしか進むことはできない（下から上または上から下への方向転換のみ）。Ai,Aj（i,j,2,3,4,5）の時は双方向への方向転換が可能である。また、エレベータBも同様にグラフ化することができる。

現在、エレベータAが1階に停止中であり、必ず行かなければならない階は3階と5階である。エレベータBも同様に2階と3階には必ず行かなければならない。現在、4階では上方向、3階、2階では下方向の待ち行列が発生している。しかしこれらはエレベータA，Bどちらか一方が行けばいいことになる。つまり、この2台のエレベータを最も効率よく運行させるには、エレベータ内部ではなく、エレベータ外部で待っている人に効率よくエレベータを割り当てる問題と考えることができる。

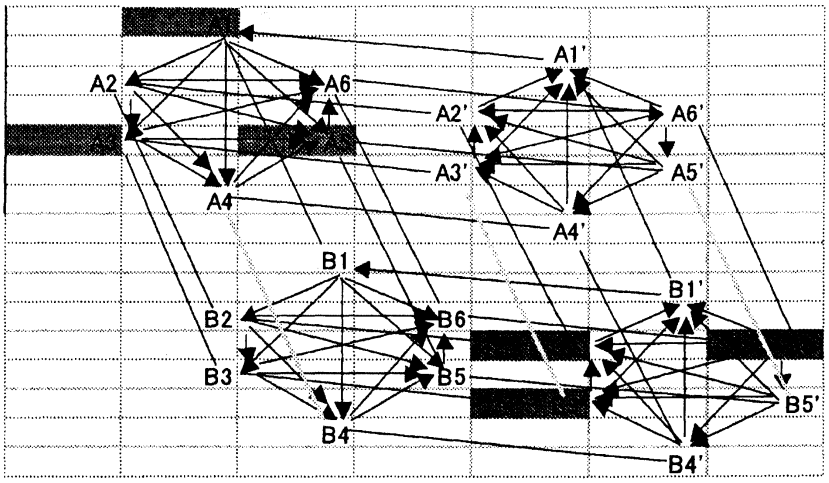
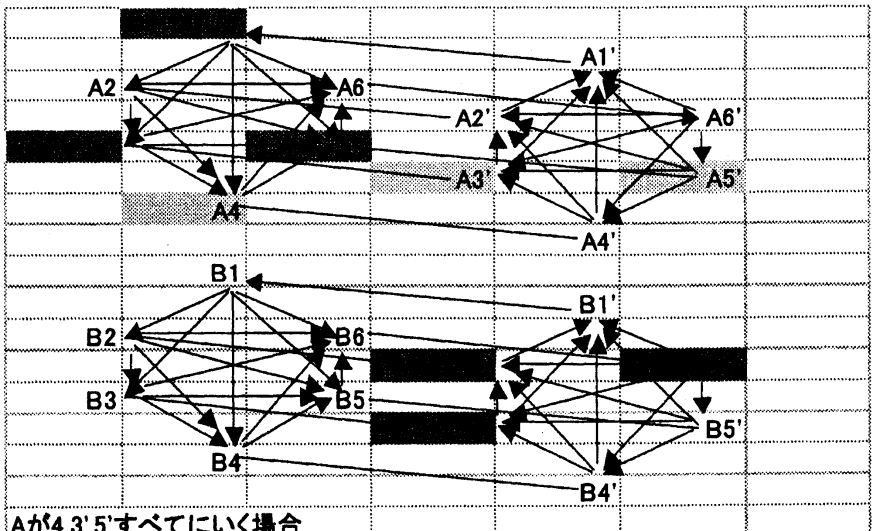


図 3.6　エレベータ 2 台の全経路のグラフⅡ

図 3.6 の黄色の線上の点は、エレベータA，Bのどちらかのグラフで通ればよい。また、色のついている階には必ず到達しなければならない。ただし、エレベータA，Bの内、片方の行く階が決まれば、もう片方のエレベータが行く階も決定する。結局、$2^3=8$ 通りの組合せそれぞれで、エレベータAとB、2 種類のグラフの最短経路を求めることになる。そして、エレベータA，Bの大きい方を $f_x(A,B)$ とおく。それを、この場合は 8 通りの $f_x(A,B),(x=1,\cdots,8)$ を計算する。最後にその最小の値を選ぶ $\min(f_x(A,B),(x=1,\cdots,8))$ の問題となる。この値の組み合わせがエレベータA，Bの 2 台のエレベータの最適スケジューリング問題である。
図 3.6 から 8 通り全ての組み合わせを求めると以下の 16 種類のグラフが求められる。



A	A1	A3	A4	A5	A5'	A3'
B	B6'	B3'	B2'			

図 3.7.1　エレベータA，Bの動きの例 1

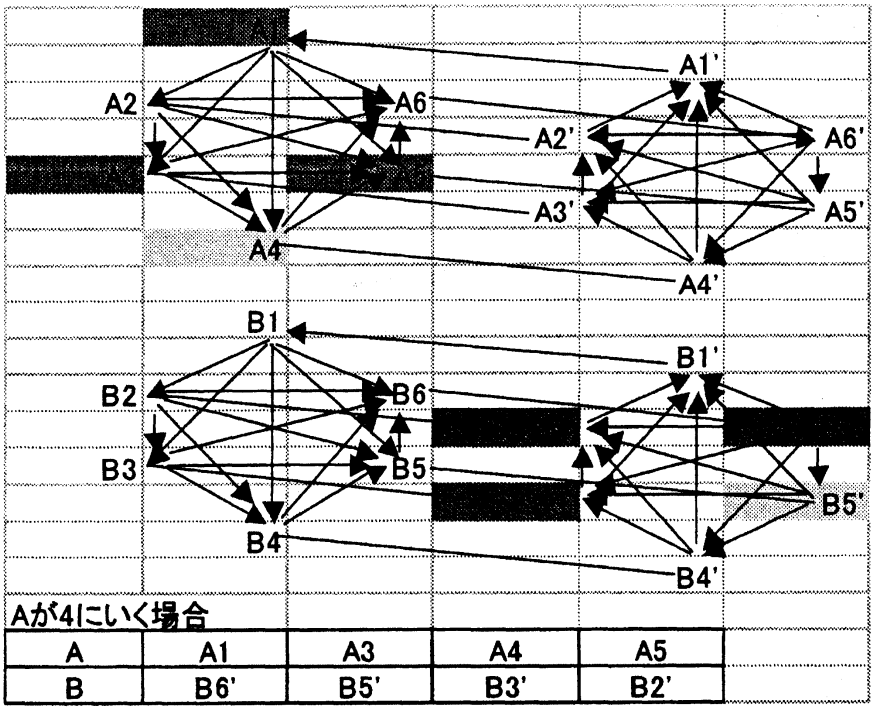


図 3.7.2　エレベータA，Bの動きの例2

5章でこれらのグラフのDNAを用いた最短経路を求める。

4. DNAコンピュータでの群管理問題の解法事項

アデルマンの計算モデル

DNA分子を用いた有効ハミルトン経路問題

［ネットワークのアルゴリズム－郵政研究所研究厳書，p38,p207］

の解法がアデルマン［IT&バイオ入門－杉野昇,p76］

によって提案されたことによって、DNA分子による汎用的な計算機を構築するための理論モデルの研究が行なわれるようになった。リプトンによる充足可能性問題の解法は、その第一歩とみなすことができる。アデルマンはこれらの結果を基礎にして、有限アルファベット上の文字列をDNA分子を用いて符号化し、文字列の多重集合に対する次のような演算をDNA分子に対する実験操作を用いて実現する計算モデルを考案した。

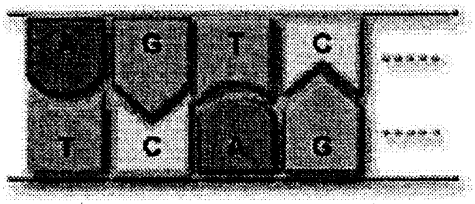
① 分離：与えられた集合Tを文字列sを部分文字列として含む文字列の集合+(T,s)と、含まない文字列の集合－(T,s)に分割する。これはDNA分子の抽出実験を試験管Tに対して行なうことに対応する。

② 混合：集合T_1とT_2に対して、その和集合$T_1 \vee T_2$をとる。これは試験管T_1とT_2の内容物を混合させる操作に対応する。

③ 検出：Detect(T)はTが空集合でないならばYESを返し、Tが空集合ならばNOを返す。これは電気泳動法などを使って、DNA分子の存在の有無を確認する実験操作に対応する。

④ 増幅：与えられたTに対して、内容がTと等しい新しい多重集合T_1とT_2を生成する。これはPCRを用いてDNA分子の量を増幅する実験操作に対応する。

ここで、DNA 分子の量を増幅する実験操作に対応する。ここで、DNA 計算機を実現するためにもっとも重要な DNA 分子の性質は、ワトソン・クリックの相補性にある。
DNA は A(アデニン)・T(チミン)・G(グアニン)・C(シトシン)の 4 つの塩基と呼ばれる分子構造の羅列から出来ている。(図 4.1.1　DNA の結合法則)
A は T と、G は C としか結合できない（ワトソン・クリックの相補性)。この性質は、上記の分離命令を実現するうえで本質的となる。つまり、ad を部分文字列は、ad を表す DNA 配列に相補的な DNA 配列を特別な指標をつけて試験管に注入し、ハイブリダイズ（二本鎖化）させた後に抽出することによって分離される。また、この性質を利用して、ある規則をもった文字列集合をランダムに生成することもできる。



A:アデニン(Adenine) T:チミン(Thymine)
G:グアニン(Guanine) C:シトシン(Cytosine)

図 4.1.1　DNA の結合法則（ワトソン・クリックの相補性）

アデルマンのモデルでは、最初に、ハイブリダイゼーションを利用して生成した文字列集合が与えられ、上記の 4 種類の命令で記述されたプログラムを適用することにより計算が実行される。これにより、NP 完全問題などを生成検査法に基づくアルゴリズムで超並列に解くことができる。DNA コンピュータは、この性質を利用して現実の問題を解決するコンピューティング手法である。

5. DNA での郡管理問題の解法

DNA コンピュータでの解法では、まず下記の表 1 の階対応表に示す塩基配列を各階に対応させる。

表 1　階対応表

1	2	3	4	5	6
AAAA	CCCC	TTTT	ATAT	GAGA	GGGG
1'	2'	3'	4'	5'	6'
CACA	TCTC	TGTG	GCCG	CAGT	GATC

次に各階に割り当てた塩基配列を前半と後半に分けて、それをつないだものをそれぞれの移動ルートとする(移動ルート図の黒字部分)。
また、枝の重みにＤＮＡの配列の長さを割り当てる

$$\psi_k = f(k) + T_E$$

$$\psi_1 = f(1) + T_E = \text{ランダムな 2 文字の配列（ZZ）}$$

$\psi_2 = f(2) + T_E$ ＝ランダムな 4 文字の配列(WWWW)

・
・
・

$\psi_5 = f(5) + T_E$ ＝ランダムな 10 文字の配列(YYYYYYYYYY)

これらを各階の配列の間に挟み時間を表すものとする。

表 2　　DNA 配列で経路の置き換え

1	→	2	AAZZCC	6'	→	5'	TCZZCA
			TTZZGG				AGZZGT
1	→	3	AAWWWWTT	6'	→	4'	TCWWWWGC
			TTWWWWAA				AGWWWWCG
1	→	4	AAVVVVVVAT	6'	→	3'	TCVVVVVVTG
			TTVVVVVVTA				AGVVVVVVAC
1	→	5	AAXXXXXXXXGA	6'	→	2'	TCXXXXXXXXTC
			TTXXXXXXXXCT				AGXXXXXXXXAG
1	→	6	AAYYYYYYYYYYGG	6'	→	1'	TCYYYYYYYYYYCA
			TTYYYYYYYYYYCC				AGYYYYYYYYYYGT
2	→	2'	CCTG	5'	→	5	GTCA
			CCAG				CAGT
2	→	3	CCZZTT	5'	→	4'	GTZZGC
			CCZZAA				CAZZCG
2	→	4	CCWWWWAT	5'	→	3'	GTWWWWTG
			CCWWWWTA				CAWWWWAC
2	→	5	CCVVVVVVGA	5'	→	2'	GTVVVVVVTC
			CCVVVVVVCT				CAVVVVVVAG
2	→	6	CCXXXXXXXXGG	5'	→	1'	GTXXXXXXXXCA
			CCXXXXXXXXCC				CAXXXXXXXXGT
3	→	3'	TTTG	4'	→	4	CGAT
			AAAC				GCTA
3	→	4	TTZZAT	4'	→	3'	CGZZTG
			AAZZTA				GCZZAC
3	→	5	TTWWWWGA	4'	→	2'	CGWWWWTC
			AAWWWWCT				GCWWWWAG
3	→	6	TTVVVVVVGG	4'	→	1'	CGVVVVVVCA
			AAVVVVVVCC				GCVVVVVVGT
4	→	4'	ATGC	3'	→	3	TGTT
			TACG				ACAA
4	→	5	ATZZGA	3'	→	2'	TGZZTC
			TAZZCT				ACZZAG
4	→	6	ATWWWWGG	3'	→	1'	TGWWWWCA
			TAWWWWCC				ACWWWWGT
5	→	5'	GACA	2'	→	2	TCGC
			CTGT				AGCG
5	→	6	GAZZGG	2'	→	1'	TCZZCA
			CTZZCC				AG
6	→	6'	GGGA	1'	→	1	CAAA
			CCCT				GTTT

この場合、移動ルートは全部で 40 ルートである。それの対となる DNA 断片を作成する(移動ルート図 n の赤字部分)。次に「表 1　階対応表」の DNA 断片と「表 2　　DNA 配列で経路の置き換え」の赤字部分の DNA 断片を大量に作る。あとはそれぞれの DNA 断片と、それを結合する酵素を同じ容器に入れて、適温にして掻き混ぜることで自動的に各種の組合せが生成される。

すると「各階対応表」と「移動ルート図の赤字部分」の断片が勝手につながり合い、いろいろな DNA 配列の断片が出来上がる。

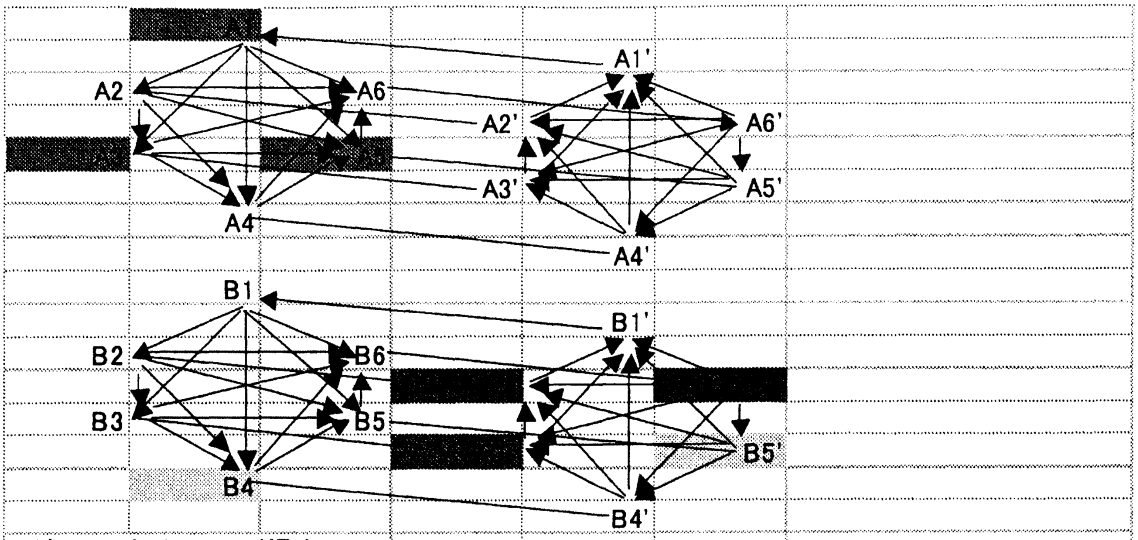


図 3.7.3　エレベータＡ，Ｂの動き例１

図 3.7.3のグラフを解く場合、多数のＤＮＡ配列の中から、再度各種の酵素の力を借りて、最初が「AA」(上行き１階の前半２文字)と最後が「GA」(上行き５階の後半２文字)の塩基を持つDNA列だけを取り出す。

この結果、「1階上向き→5階上向き」の配列だけが選択出来きる。その中には行かなければならない階に行ってないものや、同じ階に二度行っているものなど、雑多なものが含まれている。

しかし経由すべき階は判っているので、ＡＡ＊ＴＴＴＴ＊ＧＡの配列だけを選択する。(ＡＡで始まり、途中ＴＴＴＴという配列を含み、ＧＡで終わるもの）の中で最も配列の長さの短いものを選択すると結局、下記の１階からスタートし３階をへて６階に到達するものの内で最短時間のものが選ばれる。

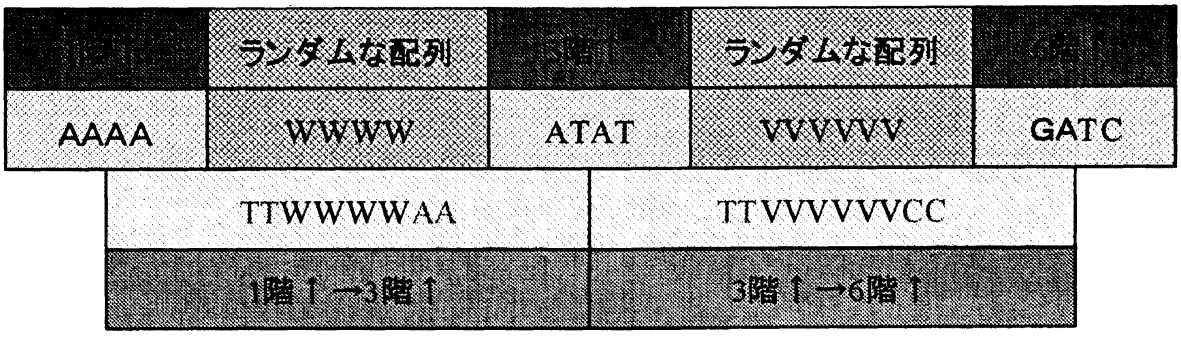


図 5.2　　経路の DNA 配列へ割当Ⅱ

これは DNA 断片の重さを利用して抽出することが出来る。(長いものは重く、短いものは軽い)後は DNA 配列を調べて、各階に置き換えるだけである。

同様にしてエレベータＢを計算すると

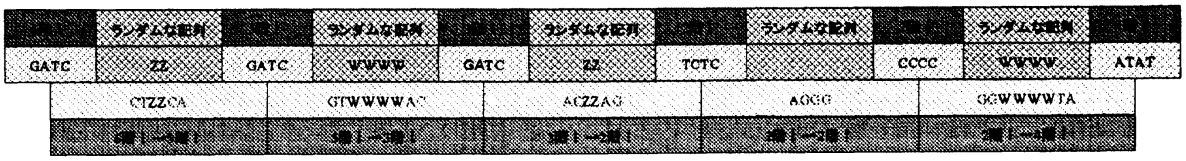


図 5.3　経路の DNA 配列へ割当Ⅲ

という配列が計算できる。

$f(A,B)$=max(4+4+4+6+4,4+2+4+4+4+2+4+4+4+4)=max(20,36)=36

同様にして２．２．１～２．２．８のグラフの最短経路を求めると以下のようなＤＮＡ配列の長さが求められる。

表 5.1　エレベータの動きⅠ

Bが4,3',5'すべてにいく場合						
A	A1	A3	A5			AA*TTTT*GA
B	B6'	B5'	B3'	B2'	B4	GA*CAGT*TGTG*TCTC*AT

$f(A,B)$=max(4+4+4+6+4,4+2+4+4+4+2+4+4+4+4) =max(20,36)=36

表 5.2　エレベータの動きⅡ

Aが3'にいく場合						
A	A1	A3	A5	A3'		AA*TTTT*GAGA*TG
B	B6'	B5'	B3'	B2'	B4	GA*CAGT*TGTG*TCTC*AT

$f(A,B)$ =max(32,36)=36

表 5.3　エレベータの動きⅢ

Aが5'にいく場合						
A	A1	A3	A5	A5'		AA*TTTT*GAGA*GT
B	B6'	B3'	B2'	B4		GA*TGTG*TCTC*AT

$f(A,B)$ =max(24,32)=32

表 5.4　エレベータの動きⅣ

Aが4にいく場合						
A	A1	A3	A4	A5		AA*TTTT*ATAT*GA
B	B6'	B5'	B3'	B2'		GA*CAGT*TGTG*TC

$f(A,B)$ =max(24,24)=24

表 5.5　エレベータの動きⅤ

Aが4,3'にいく場合						
A	A1	A3	A4	A5	A3'	AA*TTTT*ATAT*GAGA*TG
B	B6'	B3'	B2'	A5'		GA*TGTG*TCTC*GT

$f(A,B)$ =max(36,24)=36

表 5.6　エレベータの動きVI

Aが4,5'にいく場合						
A	A1	A3	A4	A5	A5'	AA*TTTT*ATAT*GAGA*GT
B	B6'	B3'	B2'			GA*TGTG*TC

$f(A,B)$ =max(28,20)=28

表 5.7　エレベータの動きVII

Aが3',5'にいく場合						
A	A1	A3	A5	A5'	A3'	AA*TTTT*GAGA*CAGT*TG
B	B6'	B3'	B2'	B4		GA*TGTG*TCTC*AT

$f(A,B)$ =max(32,32)=32

表 5.8　エレベータの動きVIII

Aが4,3',5'すべてにいく場合						
A	A1	A3	A4	A5	A5'	AA*TTTT*ATAT*GAGA*GT
B	B6'	B3'	B2'			GA*TGTG*TC

$f(A,B)$ =max(28,20)=28

この中で最も配列の短い組み合わせはエレベータAが 4 階に行き、エレベータBが 3 階と 5 階に向かう組み合わせが選ばれるわけである。

つまりは、「図 3.7.2　エレベータA，Bの動き例 2」が現時点でのエレベータ外部で待っている人の待ち時間を最小にする組み合わせであり、A,B2 台のエレベータの現時点での最適な割り当てであることが分かる。この計算を新たにボタンが押された時に再度計算し続ければ、エレベータA,Bは常に乗客の待ち時間を最小にする経路を再計算し続けることで、常に乗客の待ち時間を最小にすることができる。

6. 検証

エレベータ 1 階分の移動時間を 3 秒

（エレベータの階i～階jまでの移動時間が$T(i,j)$より$f(1)$=3 秒）

また、乗車・下車時間を 3 秒とおく。

（T_E=6 秒）

6.1　方法

1200 回乱数を発生させランダムな待ちを得る。

乱数の計算 1 回分を 3 秒とおき 1200 回の乱数発生で 3600 秒間（一時間）のシミュレーションを行った。

1 分ごとの待ち数をヒストグラムで表すと以下のグラフが得られる。

以上の分布を待ちが発生してからエレベータに乗るまでの時間を 1 分ごとに整理する。

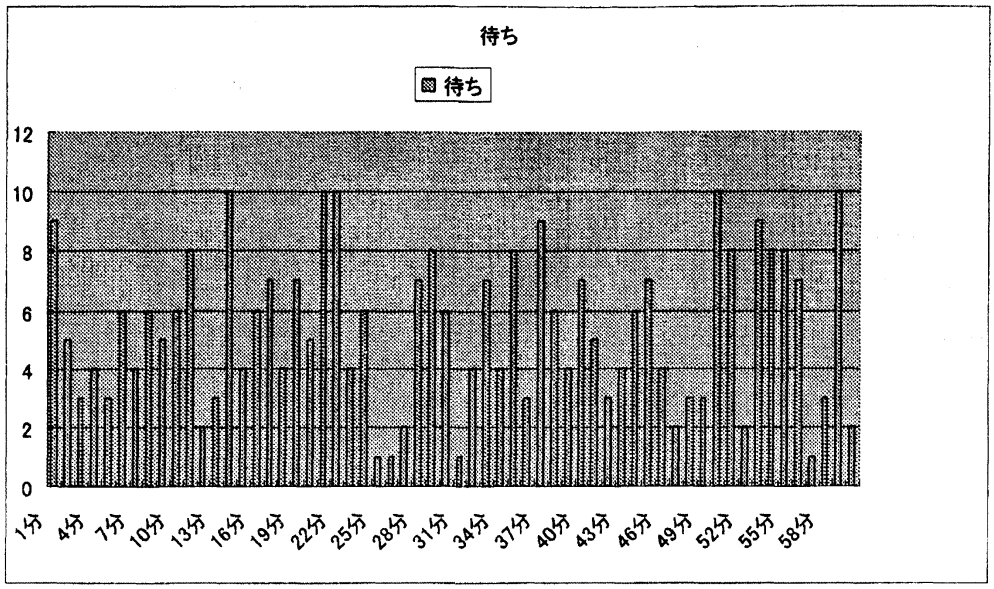


図 6.1.1　　1 分ごとの待ち数の分布

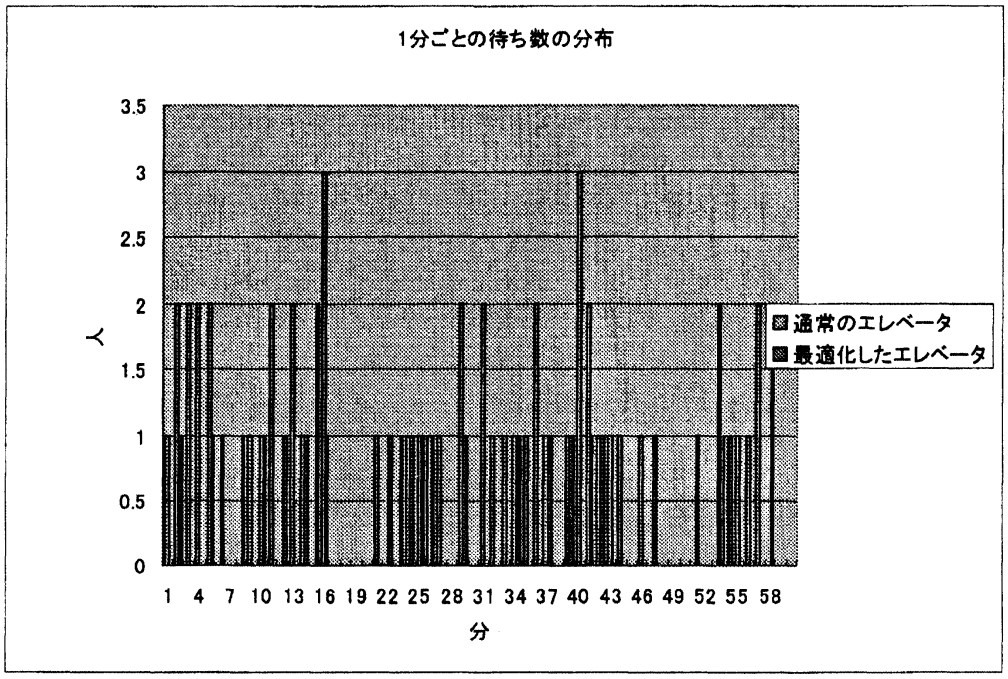


図 6.1.2　1 分ごとの待ち数の分布

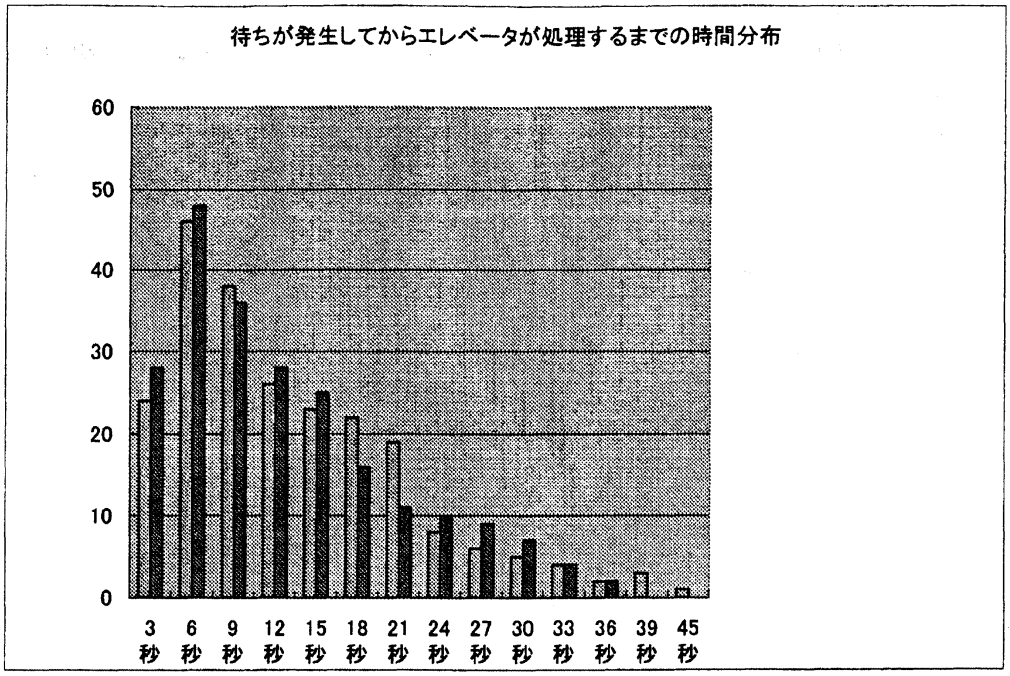


図 6.13　待ちが発生してからエレベータが処理するまでの時間分布

7. 考察

1 時間にランダムな乗場呼びが 320 回発生し、待ちが発生してからエレベータが処理するまでの時間が 18 秒までは最適スケジューリングも通常のエレベータも差がでなかったが、これは移動時間はどちらも変わらないためである。

しかし、18 秒を超えたデータに関しては最適スケジューリングの方がより短い時間で処理を行っていることが明確となっている。今回の問題は規模が小さいため DNA コンピュータの動きをプログラミングとして計算できた。しかし、実際の DNA コンピュータは再計算に時間が掛かるため実用的でない。

8. むすび（DNA コンピュータの将来）

本論文で紹介した DNA コンピュータの考えは、現在のコンピュータ（フォン・ノイマン型コンピュータ）の限界である「逐次処理」に対する挑戦であり、パラレル処理である。人間の脳は、まさにこの典型で、自然界、生物はパラレル処理を実行している。

それは、0 と 1 を使い、すべての可能性を順次処理していく、現在のコンピュータよりはるかに効率がよく、合理的な情報の処理を行なっている点である。　　つまり、DNA は 2 進法の代わりに 4 進法のデジタル処理を行い、かつ、その組み合わせが限られた性質を使用し、超並列敵に処理を行っている。現在のコンピュータが指数関数的に計算時間が増加するような問題が、線形的な時間増加で済むことが期待されており、セールスマン問題と同様、暗号に対する総当り攻撃（全数探索）などに対しても非常に有効な手段になり得るのではないかと考えられている。

参考文献

[1] J.Watada, S.Kojima, S.Ueda and O.Ono: "DNA Computing Approach to Optimal Decision Problems", FUZZ-IEEE2004, IEEE International Conference on Fuzzy Systems at Budapest, 25-29 July 2004

[2] 杉野昇：IT&バイオ入門，Vol.1,pp.1-43, 2004
[3] G.パウン訳者：横森貴：DNA コンピューティング，Vol.1, No.3,2001 年 5 月 28 日
[4] アラン・ドーラン,ジョーン・オールダス：よくわかるネットワークのアルゴリズム, Vol.1, No.1, pp.95-190, 2003 年 3 月 10 日
[5] 電気学会：あいまいとファジィ-その計測と制御-，Vol.1, No.1, pp.33-46, 1991 年 6 月 20 日
[6] 萩原将文：ニューロ・ファジィ・遺伝的アルゴリズム，Vol.1, No.7, pp.93-111, 2000