【研究ノート】

複雑系としての時間割システムに係るある数理現象について

河本直紀*

On a Numerical Phenomenon Related to a Planning System of Lectures as a Complex System

Naoki Kawamoto

Abstract

In Japan Coast Guard Academy we operate the planning system to make a schedule of lectures. The schedule is rather complicated because of many factors to be considered: classes, faculty, dates, and so on. The system produces a lot of files from the above facts, but it is hard to find good indices which show the situation inside of the system. The most fundamental information is the number of lectures which are failed to be located by the system. We notice the numbers are random and chaotic for each try, but it seems that they suggest the existence of an underlying dynamical system.

Keywords: software, lecture, planning system, chaotic, dynamical system

1 はじめに

海上保安大学校では十数年前から各期の時間割の作成に自主開発の専用ソフトウェアを使用してきた．最近そのソフトウェアが興味ある動作をしていることに気付いたので報告したい．このような特定のシステムの性質について述べることは，自然現象等の解析とは異なっていて一般性に乏しいとも考えられる．しかしながら，ソフトウェアの仕組みの中には広く適用できる一般的な法則も存在していると思われるので，ここで述べるような現象もいろいろな局面で現れている可能性がある．すなわちソフトウェアの目的として当初から意図していたわけではないが，結果的にこのソフトウェアがある種の力学系のシミュレーションをしているのではと解釈できるような挙動をしているのでそのことについて述べたい．

時間割作成のソフトウェアは実務処理のために開発したものであり，矛盾の無い実行可能な時間割を作成することが主な目的であるが，これに加えてこのシステムではなるべく授業科目の配置に曜日や時限等の偏りが少なくなるように配慮している．そのためにシステムに簡単な自己学習機能を入れて，ある程度自律的に動作するような仕組みとなっている．その結果としてシステムの動作については予測困難な部分が生じた．システム内部でのプロセスの進行状況についてすべてを記録することは可能であるが，その記録をファイルとして出力するとデータ量が多過ぎてかえって全体的な状況の把握が難しくなる．そのため通常は基本的な1種類の数値を指標として参照しているのであるが，最近の見直しの際にその指標の変化に関して相転移とも思われる数理現象が起きていることに気付いたのでその概要について述べ，その要因についても考察を試みてみたい．

Received November 15, 2010

*海上保安大学校　kawamoto@jcga.ac.jp

2 時間割の概要

大学における時間割は通常は各クラスに対して曜日・時限からなる次のような2次元の表に講義を配列するという形でなされる.

	月曜日	火曜日	水曜日	木曜日	金曜日
1・2時限					
3・4時限					
5・6時限					
7・8時限					

すなわち上の20個の空欄へどのように講義を配置するかという問題となる. 複数の学年, クラスがある場合には重複しないように相互の関係も考慮する必要があって易しい問題ではないが, 基本的にはこの配置により半期15回の講義が設定される.

これに対して海上保安大学校では, 学期中に乗船実習や各種の訓練等が行われるために, 曜日と時限の指定のみでは規定の講義回数（通常は15回）が確保できない場合がしばしば起こる. そのようなときには不足している講義を別の曜日・時限に振り分ける必要が生じる. このことから実施日の指定が必要となり, 結果的に学期の最初の週から最後の週までの各週ごとの時間割をすべて作成することとなる. すなわち実施日を指定するということで時間軸のデータが加わり3次元配列への配置となってデータ量は10倍以上になる.

平成22年度前期における海上保安大学校のクラス数は数え方にもよるが24, 対象講義数はおよそ200である. 学期内にこれらを組み込んだ時間割のパターンの数は有限ではあるが, この中から実際に実行可能な時間割を探し出すことは容易ではない. 単なる総当たり方式では実際的でないことは明らかである. そこであらかじめ満たすべき条件を設定しておいて, それらの条件の下でそれぞれの講義の配置可能箇所を探すという方針で, これを約1万行からなるソフトウェアで実行している.

3 時間割の作成

合併講義や複数担当教官等の多くの条件を満たすように時間割を組み上げるのは極めて煩雑な作業となるのでコンピュータによる自動化を行っているのであるが, 使用するデータの整理・入力は人手で行わざるを得ない. これが最も時間のかかるところであるが入力データの形式等の詳細は省略する. コンピュータで処理する部分の方針は概略次の通りである（詳しくは[1]参照）.

(1) 乗船実習や訓練等の日程の確定している行事をまず配置する. 非常勤講師等による日程の確定した講義も優先的に配置する.

(2) その他の講義科目については各科目ごとに配置難易度を計算により定める. どのような評価点を与えるかは微妙な問題であるが, 受講学生数, 担当教官数, 実施期間等を考慮した評価式を用いて計算している.

(3) 難易度の高い講義科目から順番に, 最適と思われる曜日・時限に割り振る. このときにも各曜日・時限の評価点を空きコマの数等から計算してそれにより最適箇所を選ぶ.

(4) 上で所要の講義回数が確保できない場合には残った講義について(3)の作業を繰り返す. 最終的に配置されずに残った部分についてはエラーとしてその科目名とコマ数を記録して次の講義の処理に進む. このときにはこの講義の配置難易度を少し上げる. いくら加算するかは残りコマ数等から評価式により決める.

(5) すべての講義科目を処理した後でエラーが残っていれば, 上で更新した配置難易度を用いて(2) からの作業を再度繰り返す. すると, エラーを起こした科目の優先順位が上がり, 次回にはエラーが起きにくくなると思われる. この部分が簡単な自己学習機能として働いている.

ここで繰り返される(2)～(5)の一連の作業を以下では試行と呼ぶことにする.

(6) エラーが出なければ時間割は完成となるので結果を表示して終了する. エラーが消えないときは通常は30回程度の試行を繰り返したところでシステムが停止するよう設定しておき, 各種の条件を見直して再度実行する.

4 エラーの個数

システムは動作に関する様々なデータをファイルとして出力するように設定されているが, 時間割を作成する際に最も基本的でかつ分かりやすい指標として, 各試行ごとのすべての講義科目を処理した後に残るエラーの総数, すなわち配置されなかった全残り講義コマ数がある. これが0になれば時間割は完成, 0でなければ時間割は未完成である. このエラーの個数は毎回ランダムに変動して, 初期には減少傾向にあるが, その後はマルコフ過程のようにも思われて予測困難である.

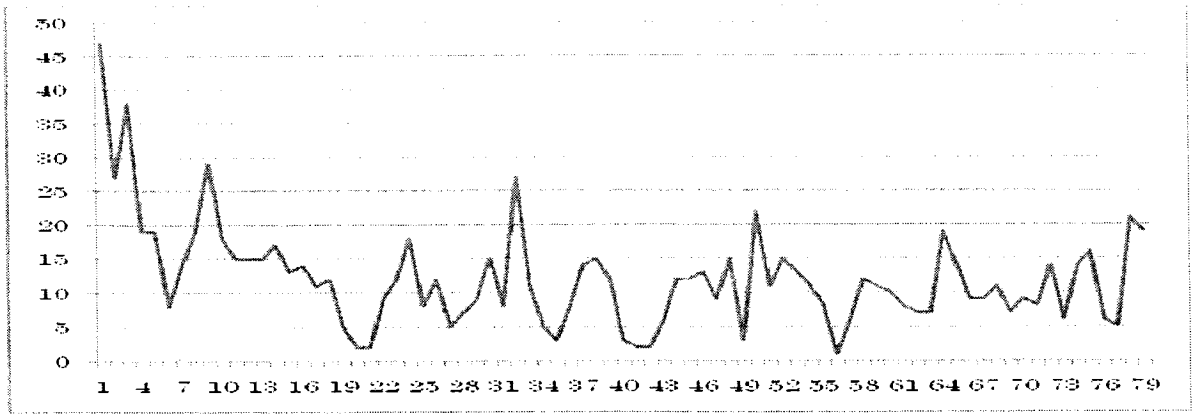

図1　試行ごとのエラーの個数（残り講義コマ数）

平成22年度前期の例を図1に示す．横軸は試行回数，縦軸は各試行でのエラーの個数（残り講義コマ数）である．科目の優先順位が少し変わると組み上がった時間割もかなり変動し，どのような配置になるかを予想することも困難である．

このシステムの過程はすべて決定論的であり，システムはエラーが0となるような組み合わせを見付けるまで，科目の順序を入れ替えて試行錯誤を行うよう設定されてはいるが，この方式により解が見付かるという理論的あるいはアルゴリズム的な保証は無い．時間割のパターンの数は有限であるからエラーの個数の最小値は存在するが，膨大な組み合わせの数の中からそれを効率よく見付ける方法は現在のところ不明であるので上のような試行錯誤を行っている．

時間割作成の作業を効率良く行いたいという実務的な面から，初期にエラーの個数が減少しても0にならずその後ある程度増加すると従来はその段階で試行を打ち切っていた．この度エラーの個数がどのように変化するのか，図1の試行をさらに長期間（600回）続けて結果を記録してみたところ図2のような結果を得た．横軸は試行回数，縦軸は各試行でのエラーの個数(残り講義コマ数)である．

このグラフには当初は予想していなかった点がいくつかある．

(1)　配置難易度の変更による修正効果についてはかなり長期間有効で，エラーの個数が何度か0に近付いていること．

(2)　その後は急速にエラーが増大して．ある種の相転移が生じたのではないかと推測されること．

(3)　システムを動かし続けると定常状態に陥って機能マヒを起こしたこと．

もう少し補足する．まず(1)については従来の短期の試行から経験的に，エラーの個数は当初は減少するがその後は増加に転ずるものと思っていた．試行回数を増やしてみると意外に長期間に渡ってエラーを0としようとするシステムの試行錯誤の効果が持続していた．このことは今後の運用の際に参考となる．

(2)についてはグラフの変化が急であることが

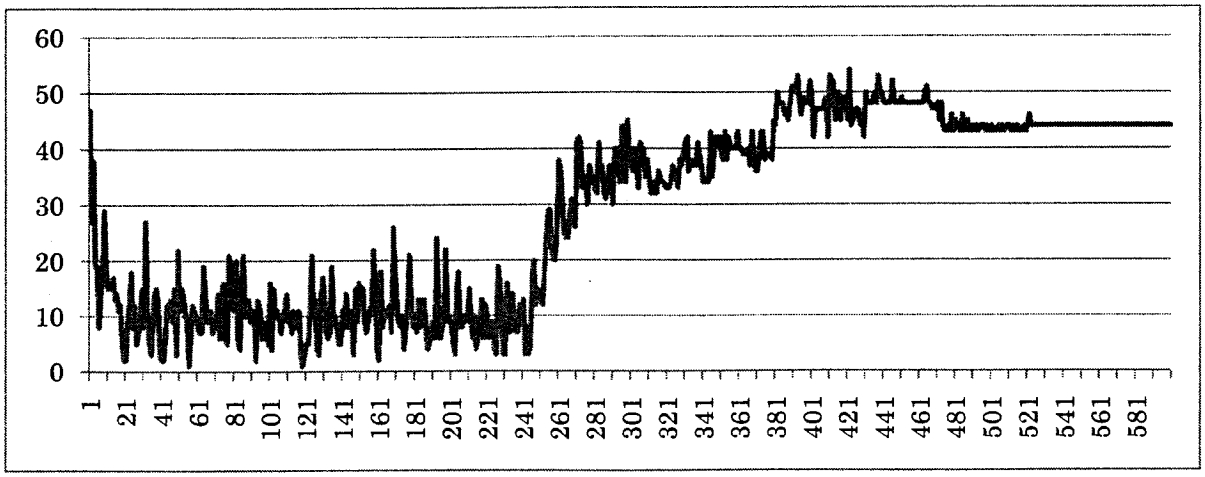

図2　長期の試行によるエラーの個数

予想外であった．他のケースについても調べたところではもう少し緩やかに増加した場合もあったが，いずれにしてもある時点から増加に向かっている．このことはシステムの内部で何らかのかなり急な変化が起きたことを推測させる．

次に，(3)については次のように考えられる．各科目の配置難易度は上で述べたように評価式により値を定めている．配置難易度は整数値としているが，実行時のオーバーフローを防ぐためにこの値にはシステム内で上限を設定している．そのためエラーとなった科目については配置難易度を上げてゆくのであるが，上限に達してしまうと以後は値が変化しない．エラーを起こす科目の多くがこの上限に達してしまうと，難易度による順序がほぼ固定化されて変化が起こりにくくなると思われる．

グラフの変動について詳しく見ると次のようになっている．まず100回あたりまでは収束期ともいうべき期間で次第にエラーの個数が減少してエラーが0の場合を見出すというシステムの目的に沿った結果を出している．次に250回あたりまでは巡回期ともいうべき期間となっている．ここでは他の出力データ等から見ると，科目の配置パターンに何種類かあって厳密に同じではないが，それらがほぼ繰り返し出力されているように見える．その後の後退期ではシステムがその効力を失い次第にエラーが増加する．これが500回あたりまで続く．それ以後の停止期ではシステムはほぼ機能停止状態になって出力はほとんど変化しなくなっている．上のグラフはかなり典型的な例であるが，データやプログラムの条件を変えてみてもほぼ同じような規則性が見られる．

変化の様子を表わすもう一つのデータを図3に挙げる．これは比較的配置難易度の変動の大きかった科目，すなわちエラー個数の変動の主な原因となっていると思われる科目の優先順位の和から計算したもので，算出方式の詳細は省略するが，横軸は試行回数，縦軸は各試行での該当科目の優先度の総計とでもいうべきものである．結果として図2のグラフに対応した変化が見られ，これも内部で何らかの変化が起きていることを示しているものと思われる．

システムはすべて決定論的であるのに，エラーの個数は予測不能なランダムな動きをしている．しかし何らかの規則性もうかがえる．しかも(2)のように相転移，ないしは解の爆発のような現象が起きているのではとも推測される．このようなことから，プログラムとして書かれている実際的な処理とは別の，何らかのメカニズムが奥に存在するのではとの思いが生じる．

5 1つの考察

それではプログラムに内蔵されている意図しなかったメカニズムとはどのようなものであろうか，ここでその可能性を少し検討してみたい．ただし，現時点ではあくまでも推測の段階であり明確な理由付けがあるわけではない．

1つの可能性は時間割を組む過程にあると思われる．このプログラムではそれぞれの科目をどこに配置するのが適当かを決めるときに，配置候補箇所の適合度を3(3)で述べたように評価式により計算してその値により選んでいる．すなわち，すでに配置した科目と重複しては困るので配置した科目の分布の様子が重要な要素となり，なるべく偏りを生じないような配置となるように評価点を与えて処理している．このことは配置科目の作るある種の場を考えて極値を求めているとも考えられる．つまり科目間の相互作用が極小となるような箇所を捜しているとも見なせる．この作用は現在のところ式として明確に定義できるようなも

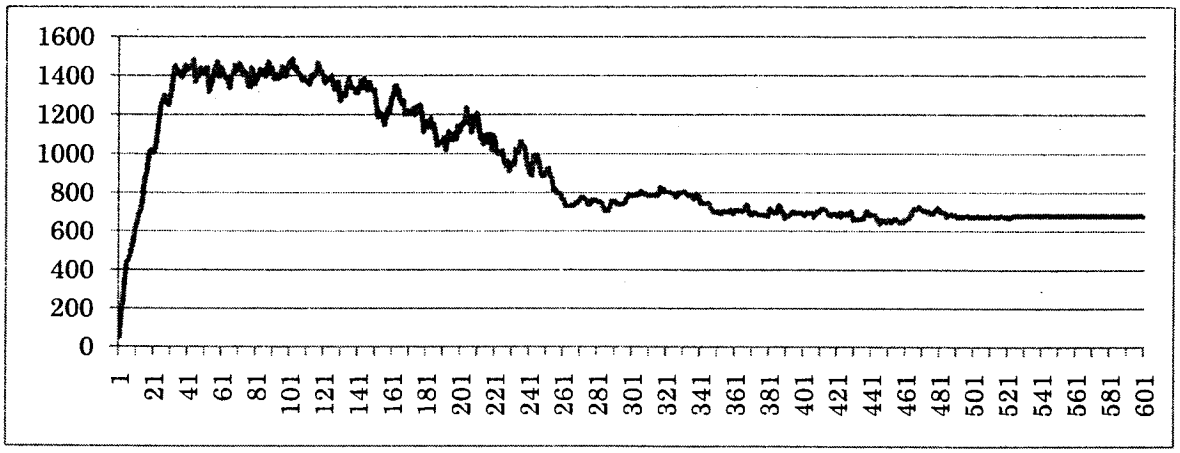

図3 優先順位の変動

のではないがある種のポテンシャルによるとも考えられる. 言い換えると, 重複が生じないように, 近づけば斥力が働いていると見なすことができる. このことにより各科目を粒子とする多体系が形作られ, ある種の力学系をなしていると考えられる. また, エラーにより配置難易度を修正し, 優先順位を変えて試行を繰り返すことは科目の配置を変えることとなるが, これはその科目にある種の力が働いて新たな分布に移行したものとも考えられる.

別の見方としては科目の優先順位を取り上げることも可能かもしれない. エラーの変動と科目配置の変化は主として配置難易度の変更による優先順位の変動から生じている. このように優先順位に着目すれば一列に並んだ粒子（科目）の相互作用による順位の入れ替えと見ることもできる. これを粒子の運動と思えば, ある確率で粒子が右あるいは左の粒子と入れ替わる現象が繰り返されていると見なすこともできそうである.

多体系においてはしばしばカオス的現象が観察されている（例えば[2], [3]等）. また多粒子系においては相転移を生じることがあることも知られている. 時間割の作成プログラムは実務のための処理系であって自然現象のシミュレーションを意図したものではないが, 結果的に内部では何らかの抽象的な離散力学系を動かしていることになっているのではと考えることができそうである. そうであれば出力データがカオス的で相転移が推測されるということもあり得るものと思われる. ただしその具体的な仕組みについては今のところ不明である. なお, このシステムでの計算はすべて整数計算であり, 計算誤差については考慮する必要は無い.

6 課題

多くの場合の組み合わせを含む現象については, その変化が離散的となることが多くて, 連続な関数による解析的な取り扱いは困難となることが多い. そこで個別にその変化を調べようとすると組み合わせの数が多過ぎて, 現象の統一的な把握が困難になることがある. 有限として個別に扱うには多過ぎるが, 無限として連続的に扱えるほどには多くないというのは, 具体的な問題ではしばしば起こる事柄である. ここで扱った問題についても現在のところ微分方程式等による定式化はできていないが, 適度な粗視化をすることにより連続的なモデル作成の可能性はあるように思われる（例えば[4]参照）. このシステムを力学系と見ることができるのではと述べたが, 微分方程式あるいは差分方程式によるシステムのモデル化ができればその検証も可能になるかもしれない. モデル化の際にはエラーが生じたときの配置難易度の変更をどのように考えるかが問題になると思われるが, 残念ながら現時点では具体的な方策については不明である.

このシステムを力学系とみる見方が可能になればこのプログラムの改良の方向性も見えてくるものと思われる. 良い時間割とはどのようなものであるかということに対しては現在のところ明確な基準は無いが, 例えば全エネルギーに相当する値を定めることができれば, その値は時間割の全体としての落ち着きの良さ（歪の少なさ）を示す指標になりうるであろう.

あるいは, 出力パターンの変化からエラーの最小値の推定, ないしは時間割作成可能な科目数の上限値について何らかの情報が得られるのではないか, との推測もできる. これが分かれば今後のカリキュラム改編の際の参考になるであろう.

ここで述べたように人工的なシステムを物理系と解釈することは, 新たな知見と手法をもたらす可能性があり興味深い見方と思われるが, モデル化にはかなりの困難が伴うものと予想される.

参考文献

[1] 河本直紀, 講義実施計画表の自動作成, 海上保安大学校研究報告, Vol. 45 (1999), No. 1, pp. 63-82.

[2] 川原琢治,「ソリトンからカオスへ　－非線形発展方程式の世界－」, 朝倉書店, 1993.

[3] 小室元政,「基礎からの力学系　分岐解析からカオス的遍歴へ」, 数理科学 SGC ライブラリ 17, サイエンス社, 2002.

[4] 大野克嗣,「非線形な世界」, 東京大学出版会, 2009.