車両運用計画導出における島モデルGAの適用

平方 敦*　小高 知宏*　黒岩 丈介**　諏訪いずみ**　白井 治彦***

Application of island model GA in railway rolling stock operation plan

Atsushi HIRAKATA*,　Tomohiro ODAKA*, Jousuke KUROIWA**,
Izumi SUWA** and Haruhiko SHIRAI***

(Received February 2, 2018)

In this paper, we applicated island model GA for optimizing railway rolling stock operation plan. In island model GA, that performs evolutionary computation in each of many sub-populations. In the case of processing with the CPU, each sub-population is processed sequentially. GPU can be efficiently performed by distributing calculations within each sub-population to cores present in GPU.

We dealt with optimization of railway operation plan as one of optimization problems. When this method was used for deriving the railway operation plan with the GPU, the effect of shortening the calculation time was confirmed when processing was performed than using the CPU.

Key words : Railway operation plan, Genetic Algorithm, Graphic Processing Units, Island model

1. はじめに

我々が日常的に利用する公共交通機関として鉄道が挙げられる. 日本の鉄道技術は世界的に見ても先進的であり, 秒単位の定時性を維持しつつ毎日の運行を行っている. 近年注目される環境保全の面からも, 公共交通機関の利用は推進されており, 通勤や通学以外にも観光や貨物輸送への利用における再注目など, これからも我々の日々の生活に欠かせない存在になると考えられる.

日本の鉄道は高い定時運行制で知られているが, 時として, 事故や災害などによって列車ダイヤに乱れが生じることがある [1]. 人身事故や自然災害などによる列車の運休や遅れは全国で毎日のように発生している. 列車の運行が乱れた際には, 現在の状況から運行可能な区間を決定し, 列車の運行時刻や乗務員の手配, 充当する車両の手配を行なう必要がある. また, このような運転計画の策定は鉄道事業者が年に数回行なうダイヤ改正の際にも行われる. しかし, 数カ月前から準備できるダイヤ改正のような場合と異なり, 事故などの突発的な事象に対する列車の運転計画の策定には, なるべくダイヤの乱れを拡張させないという面から迅速な対応が求められる.

列車の運転計画における車両運用の策定は現在, 熟練の技術者が経験則に基づき行っている. そのため, 計画の策定には多くの時間を要し, 計画自体の品質についても計画作成者の技量に依存していることから不安定である. また, 運用作成技術を後継者に伝えるという点でも鉄道運用を担う事業者の負担となっている現状である.

そこで本研究では, 車両運用計画の導出をPCを用いて自動で行なう.PCにより計画を導出することで, 安定した品質の計画の導出が可能であることや, 計画の作成にかかる時間の短縮が見込める. その中でも本研究では処理の高速化に焦点を絞る.GPUを用いた島モデルGAを車両運用計画の導出に用いることで,CPU

* 大学院工学研究科 原子力・エネルギー安全工学専攻
** 大学院工学研究科 知能システム工学専攻
*** 工学部技術部
* Nuclear Power and Energy Safety Engineering Course, Graduate School of Engineering
** Human and Artificial Intelligence Systems Course, Graduate School of Engineering
*** Technical Division

を利用した場合と, 計算時間の短縮効果について比較し, 検証を行なう.

本論文では,2 章に車両運用計画の概要及び制約条件を示し,3 章にアルゴリズムの設計を述べる. また,4 章では車両運用計画の導出実験について述べる. 5 章では実験について考察し,6 章では本研究で提案した手法について, 総括する.

2. 車両運用計画の自動生成

本章では, 車両運用計画問題の概要について触れ, 自動生成する際に考慮しなくてはならない制約条件について述べる.

2.1 車両運用計画の概要

本節では現在の車両運用計画の概要と現状について述べる.

鉄道会社では , 列車の運行内容を列車ダイヤで定めているが , それを実行するために , 保有する各種の資源を効率良く運用することが求められる [2].

前提として, 鉄道の運行ダイヤは利用者である我々の生活ニーズを前提に取り決められることが通例となっている. 鉄道会社が保有する資源には様々なものがあり, 例えば運転士や車掌などの乗務員や線路設備, そして鉄道車両などがそれに当たる. 中でも鉄道車両は非常に高価な資源であり, その導入にかかるコストは 1 ユニット数十億円以上となる場合も少なくない. また, 車両のメンテナンスにかかる費用に関しても, 保有する鉄道車両の数に比例して大きくなることから, 出来る限り毎日の運行で使用される車両の数は少ないほうが望ましいと言える.

ここで, 毎日運行されるすべての列車に対して保有している鉄道車両を割り当てる作業を車両運用計画という. 現在, 車両運用計画については人手による導出が主となっている. しかし, 前述の通り人手による計画の導出は, 品質の不安定さや計画導出にかかる時間といった点で鉄道事業者の大きな負担となっている.

そこで, 車両運用計画問題を組合せ最適化問題として捉え,PC を用いて運用を導出するため, 現在様々な研究が行われている. 車両運用計画作成時における制約条件に対応させたシステムを用いた実験では, 実在する時刻表データに対してアルゴリズムを適応させ, 実際の運用に遜色ない運用導出を行なうことができていた [3]. しかし, 運行する列車の数が多い路線にアルゴリズムを適用させる際には, 必然的に計算量も多くなり実行時間が増えてしまうことが予想される. したがって, 車両運用計画の自動導出を実現するシステムにおいては, 運用導出に必要な計算を削減すると同時に, より高速な演算処理を行なうことが要求される.

2.2 車両運用計画における制約条件

車両運用計画を取り決める際には, いくつかの制約条件について考慮する必要がある.

まず, どの列車にもいずれかの一つ以上の車両が必ず割り当てられなければならないということが挙げられる. 複数の車両を増結して走行することもあるため, この場合には複数の車両をひとつの列車に対して割り当てることとなる. また, 列車によっては途中駅で編成の一部を切り離す場合もあるが, この場合にも, 切り離す車両と, 切り離される車両が互いに一両以上存在しなくてはならない. この条件により, 時刻表に記載されている全ての列車の運行が約束される.

また, 終着駅に列車が到着した際には, 折り返し列車として, 到着時刻以降に駅を発車する列車に割り当てることができるが, 到着してから発車するまでの時間 (折り返し時間) にも制約が存在する. 車両の車内清掃や, 運転士の交代などの折り返し列車の準備作業には少なからず時間を要するため, 折り返し時間を一定時間以上設けなければならない.

加えて, 鉄道車両についても自動車などと同様に, 車両検査が必要であり, 規定の日数の経過に応じて全般検査・仕業検査・交番検査などを行なう必要がある. 全般検査は車両の細かな部分まで丹念に検査するものであり, 一般的な通勤車両については,2 週間ほど検査に要する. このため, 一定期間, 検査中の車両を運用から外すことになる. したがって, 運用計画を考える上で全般検査の存在を考慮する必要は基本的にないが, 全般検査中の車両については運用に入ることができないため, 通常の運用に使用できる車両の数が減ってしまう. したがって運用で使用する車両数は, 保有車両数に対してになるべく少なくなるようにすべきである. 仕業検査については比較的短時間で検査を実施可能であることから, 運用の中に検査の時間を割くことが多い. したがって多くの日数を要する車両運用が組まれた場合には, 運用の中に仕業検査のための時間を設ける必要がある.

これらの制約条件に加えて, 各鉄道会社によってその地区特有の条件など, 更に車両運用計画作成時における制約条件は増える場合も多い. また, 制約条件ではないが, 鉄道車両をメンテナンスする上で, 単一の車両だけが走行距離が長くなってしまったり, または短くなってしまうと好ましくない. 保有している同一車種はなるべく均等な走行距離になるような運用を作成することにより, 各車両のメンテナンス回数を均

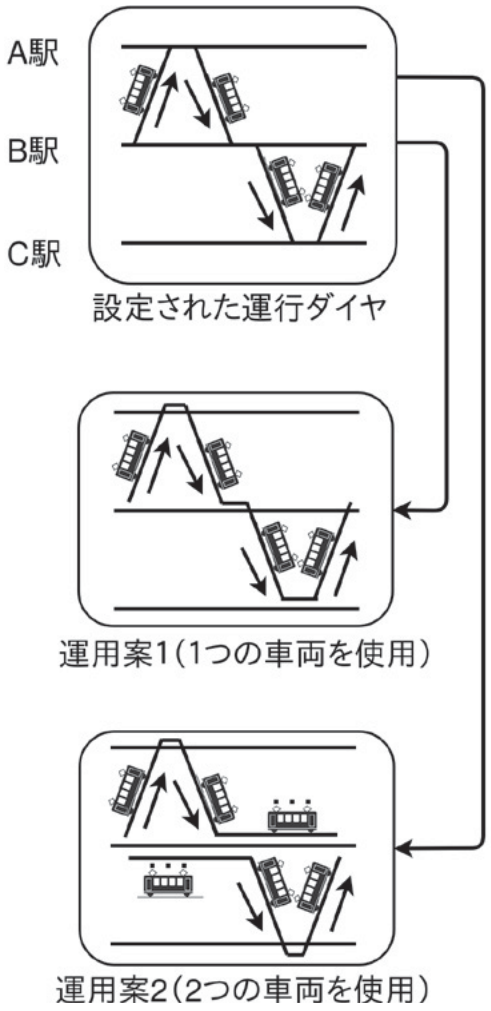

図 1 列車ダイヤと運用計画の例

一化することができ, 管理しやすくなる. 各車両の走行距離を均一化するためには,1 日ごとに割り当てる仕業を変えていき, 各車両が運用計画内の全仕業に循環的に割り当てるようにすればよい. この複数日にわたる車両の循環的な使用計画を交番という [4]. 交番を実現するためには, 前日の仕業における最後の列車の終着駅と, 翌日の仕業で最初に割り当てられる列車の始発駅が一致しなければならない.

上記のような制約条件を考慮した上で, 車両運用計画を作成していくこととなる. 全ての列車に対して列車を割り当てるという条件と, 交番を実現するようにすることから, 列車を頂点とみなし, 列車が走っていない時間をアークとするネットワークモデルを作成し, 各列車を一度ずつ通る巡回路を作成する. このような巡回路の導出が, 交番を満たした車両運用計画の作成と同義である. このとき, 総コストが小さくなるような巡回路を探索することにより, 無駄の少ない効率の良い運用を導くこととなる.

図 1 は運用計画を作成する列車ダイヤをダイヤグラムで示したものと, 列車ダイヤに対する 2 つの運用案を示している. ダイヤグラムにおける横軸は時間, 縦軸は位置を示している. 斜めの線については列車を表しており, この列車ダイヤにおいては 4 本の列車が存在していることがわかる. 右に示す運用案で色分けされている折れ線は車両の動きを意味しており, 折れ線がそれぞれ車両運用に対応している. 運用案 1 では全ての列車が 1 本の折れ線で表されていることから,1 つの車両を用いて左のダイヤグラムのように設定された列車の運行を賄っているということになる. また, 運用案 2 では全ての列車が 2 本の折れ線で表されていることから,2 つの車両を用いて左のダイヤグラムのように設定された列車の運行を賄っているということになる. 車両運用内で折れ線が水平になっている部分は車両の待機時間である. 図 1 から, 間合い時間の合計が長い車両運用の方が列車運行に用いる車両の数が多いことがわかる. したがって, 間合い時間の合計が短い車両運用計画の方が車両の稼働率が効率化された運用ということになる.PC を用いて車両運用計画を導出する際には, 間合い時間に着目して最適な運用を探索していけばよい.

3. アルゴリズムの設計

本章では, 車両運用計画を自動導出する際に用いるアルゴリズムについて述べ, 計算時間の短縮のために, 具体的にどのような手法をとるかについて解説する.

3.1 遺伝的アルゴリズムの導入

車両運用計画を PC で自動導出する際には, 計算量の削減と処理の効率化が重要である. そこで本研究では, 確率的探索手法として遺伝的アルゴリズムを用いる. 遺伝的アルゴリズムとは,1975 年に John Henry Holland によって提唱された近似解導出のためのアルゴリズムである [5]. 通常, 生物というものは, 交叉, 突然変異, 淘汰といったことを繰り返し, 世代を重ねていくごとに周囲の環境に適合するように進化していく. 遺伝的アルゴリズムでは, このような自然界での進化方法を元に, 交叉, 突然変異, 淘汰といった処理をデータに対して行なう. これらの操作を任意の世代数行なうことにより, 問題に適した解を探索することが可能である. このアルゴリズムは, 組合せ最適化問題や NP 問題などの様々な導出困難な問題に対して適用可能である.

そこで本研究では遺伝的アルゴリズムを用いて, 車両運用計画問題を導出する. 車両運用内での間合い時間を個体の環境適応度とし, 進化的計算を繰り返して

表 1 上り列車の時刻表

列車名	A 駅	B 駅
0	600	630
2	800	830
4	1000	1030
6	1200	1230
8	1400	1430

表 2 下り列車の時刻表

列車名	B 駅	A 駅
1	700	730
3	900	930
5	1100	1130
7	1300	1330
9	1500	1530

最良の車両運用計画を探索する. その際, 導出される車両運用計画については, 前述の各制約条件が遵守されるものとする. 個体の環境適応度である間合い時間の導出方法については次節で解説する.

また処理の効率化として,GPU を用いた並列処理を遺伝的アルゴリズムに適用する. この際, 並列処理に特化した遺伝的アルゴリズムの一つである島モデル GA を用いるものとした.

3.2 運用作成時における間合い時間の算出

車両運用を作成する際には, 時刻表データから間合い時間を算出し, 各列車間の間合い時間表を作成する必要がある. ここでは任意の二つの列車が運用上連続する場合に, 二つの列車を「つなぐ」と表現する. 今回は, 列車が到着する時刻から折り返しの列車が発車するまでの時刻を間合い時間として時間の差分を計算し, 車両運用を作成するアルゴリズム上で参照できるものとした.

本研究で用いた時刻表のデータの例を表 1, 表 2 に示す. ここで, 表 1 はある路線の上り列車の時刻表のデータであり, 表 2 はある路線の下り列車の時刻表のデータである. この例に示す路線では A 駅から B 駅までの間で列車を運行しており, 上下 5 本ずつ計 10 本の列車が運行されている. 実際にプログラムに与える時刻表データに含まれる情報は, 各列車の列車名, 始発駅の発車時刻, 終着駅の到着時刻とした. ここで列車名というものは, 運行される列車を識別するための名称であり, ここでは便宜上数字で表現している. 基本的に始発駅を発車する時刻の早い列車から順に小さな数字が割り当てられるものとし, 下り列車は奇数, 上り列車は偶数で列車名を設定した. これらの取り決めは実際の鉄道においても適用される例が多い. これら 3 種類の情報 (始発駅初時刻, 終着駅着時刻, 列車名) をテキストファイルとし, 実行時に読み込まれるものとした. この時刻表のデータを差し替えることで, 他の路線の車両運用計画についても, 基本的にはアルゴリズムを変更することなく導出可能である.

表 3 時刻表データから算出した間合い時間

前＼後	1	3	5	7	9
0	1800	9000	16200	23400	30600
2	81000	1800	9000	16200	23400
4	73800	81000	1800	9000	16200
6	66600	73800	81000	1800	9000
8	59400	66600	73800	81000	1800

表 3 は時刻表のデータである表 1 と表 2 から間合い時間を算出した例である. 表の縦軸については, つなぐ前の列車, 横軸についてはつなぐ後の列車を表している. また, 間合い時間については秒単位で計算している. 基本的には, つなぐ場合には上り列車から下り列車につなぐことが, 着発駅が一致するという観点から通例となっている. また, 表 3 では上り列車から下り列車につなぐ場合の全通りの間合い時間を示している. この表にある間合い時間に加えて実際には, 下り列車から上り列車につなぐ場合の間合い時間の算出も必要となる. ここで, つなぐ前後の列車の上下は確実に逆でなければならないことはない. 同一方向の列車同士であっても, つなぐ前の列車の終着駅からつなぐ後の列車の始発駅までの回送列車を設定することにより, つなぐことが可能となる. この場合は間合い時間に回送として走行する時間を加えることとなるが, 回送列車は乗客を乗せずに走る列車なので, 電力や人経費などの運行にかかるコストに対する利益がほぼないといえる. したがって回送を積極的に多用するような運用計画は望ましくないといえる.

ここで表 3 において, 間合い時間を算出する例を簡単に述べる. 例えば 2 列車から 3 列車につなぐ場合には,2 列車が終着駅である B 駅に到着する時刻が 08 時 30 分であるのに対し,3 列車が始発駅である B 駅を発車する時刻が 09 時 00 分であるので, 両時刻の差は 30 分であることがわかる. ここでは秒単位で計算を行なうので,

$$30 \times 60 = 1800(\text{秒}) \tag{1}$$

の間合い時間が存在するということとなる. もし,2 列車から 5 列車につなぐ場合には同様に,2 列車の終着

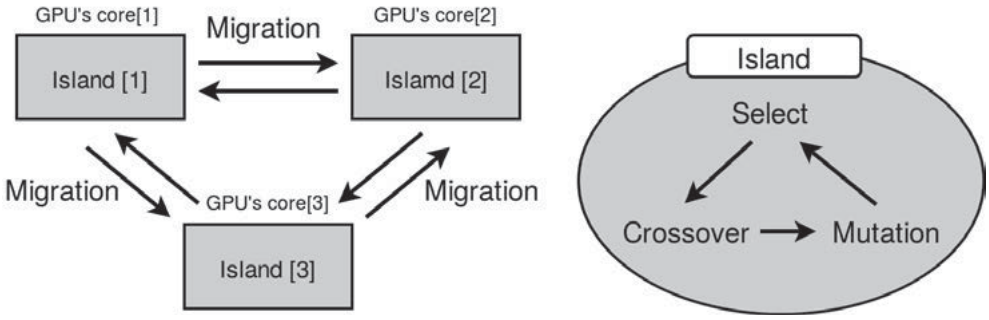

図 2 実装した島モデル GA

駅着時刻と,5 列車の始発駅初時刻の差が 2 時間 30 分であることから,

$$2 \times 60 \times 60 + 30 \times 60 = 9000(秒) \qquad (2)$$

の間合い時間となる. 式 (1) と式 (2) を比較すると, 前者のほうが間合い時間は短くなっていることがわかる. したがって, 二つの列車のみで見た場合は,2 列車からつなぐべき列車は 5 列車ではなく 3 列車のほうが適切であるといえる. このような手法で車両運用計画を作成する路線で運行される列車の全てのつなぎの組み合わせについて, 間合い時間を算出した.

3.3 島モデル GA の実装

今回実装するアルゴリズムでは, 時刻表のデータから列車間の間合い時間を算出した後, 遺伝的アルゴリズムを用いて最適運用を求める. 遺伝的アルゴリズムでは, 個体 (解) の特徴を示す遺伝子配列と, 環境適応度の設定が必要となるが, 今回遺伝子配列については車両が担当する列車の順序, 環境適応度 (評価関数) については列車間の間合い時間を用いるものとした. したがって, それぞれの個体の一つ一つの遺伝子要素については, 運用の中の列車一つ一つが対応していることとなる. 初期の解集団を乱数を用いて作成し, 環境適応度の優劣をもとに, 交叉, 突然変異, 淘汰の処理を複数回繰り返すことにより, 解の改善を行なう. また, 本アルゴリズムでは, 擬似乱数の生成手法として, 高精度で高速な乱数生成が可能であるメルセンヌ・ツイスター (Mersenne Twister)[6] を用いるものとした.

遺伝的アルゴリズムを用いて組合せ最適化問題を解く際に, 探索空間を広める方法としては, 集団内に存在する個体の数を増やすこと [7] や集団そのものの数を複数設定することが挙げられる. そこで本研究では, 後者の手法を島モデル GA として実装する. 図 2 に本研究で実装する島モデル GA を示す. 通常の遺伝的アルゴリズムのように, 単一の母集団内で進化的計算を行なうのではなく, 島モデル GA では複数の比較的小規模な母集団を設定し, それぞれの集団の中で進化的計算を行なう. 本研究では CPU または GPU 上で, それぞれの母集団内での進化的計算を設定回数行なう. 母集団内の進化的計算を CPU 上で行なう際には, それぞれの母集団の処理を逐次的に行なっていく. また, 母集団内の進化的計算において GPU を併用する場合は, 複数の母集団を GPU のコア 1 つ 1 つに割り振ることで, 並列的に進化的計算を行なうものとした.

また, 進化的計算を設定回数行なった後, 移民と呼ばれる操作を行なう. この操作では, 各母集団の中で環境適応度が最高となっている個体を他の母集団の環境適応度が最低の個体と入れ替える. この操作は, 候補解が局所解に陥るのを防ぐ目的がある. 確率的探索手法の短所である局所解への収束を防ぐことで, より最適解に近い解を得ることができる. したがって, 並列処理に特化しているだけでなく, 探索効率についても優れていると言える. 最終的に全ての母集団内で環境適応度が最良である個体を参照し, その中でも環境適応度の値が最も優良である運用を出力し, 実行を終えるものとした.

GPU に計算処理を担当させる際には, 専用の書式で命令文を CPU から GPU に転送する必要がある. 本研究において,GPU 上で動作する島モデル GA を実装する際には,NVIDIA 社が提供する CUDA という統合開発環境を用いる. この開発環境では C++がベースとなる独自の言語で GPU での処理を記述する. 通常,GPU のどのコアにどの処理を割り振るかといった定義は開発者が行なう必要があるが, この CUDA においてはコンパイル時に自動で処理が割り振られる. よって, 煩雑な記述を開発者が行なう必要がなく, 並列処理を伴うプログラムをスムーズに実装することが可能である. 一般的に GPU では, 複数のスレッドに対して並列的に計算を実行させることができる. したがって, 逐次的に処理する CPU に比べて, 高速に処理を実行できるケースが多い. 本研究では CPU 上で処理を行なうアルゴリズムについては C++,GPU 上で処理を行な

うアルゴリズムについてはCUDAを用いて実装を行なった.

4. 計画の導出実験

本章では, 前章で実装したアルゴリズムを用いて, 実際に車両運用計画の導出を行なう.

4.1 島モデルGAの並列処理による高速化実験

本節の実験では,12本の列車が運行される時刻表のデータと,68本の列車が運行される時刻表のデータを用いた. また, 架空の時刻表データを用いるのではなく, 実際に存在する路線の時刻表データとした. また, 実行環境として,CPUはintel core i7 2600K,GPUはNVIDIA社のGTX-980を用いた. 使用したCPU及びGPUの仕様については, 表4と表5に示す.

また, 今回の実験では,CPUによる処理とGPUとCPUを併用した場合の処理の速度を比較するため, 進化させる世代数は, 各サブ集団とも1000世代固定とした. どのサブ集団においても,30世代分進化処理を行なった後に, 集団内での優良個体を交換する移民操作を行なう. 実行開始から1000回目の進化処理が完了し, その時点での最良の運用を出力するまでの要した時間を記録した.

4.2 島モデルGAの並列処理による高速化実験の結果

表6及び図3に12本の列車が運転される路線における車両運用計画の導出実験の結果を, 表7及び図4に68本の列車が運転される路線における車両運用計画の導出実験の結果をそれぞれ示す. 表6, 表7では左の列から設定した小規模な母集団(島)の数,CPU上で母集団内での演算を行なった場合の処理時間, GPU上で母集団内での演算を行なった場合の処理時間を表している. また図3, 図4は表6, 表7をそれぞれグラフに表したものであり, グラフの縦軸は処理に要した時間, 横軸は設定した世代数を示している. 実線で示されるのが, 各サブ集団の計算をCPU上で逐次行っていった場合の処理時間である. また, 破線で示されるのが, 各サブ集団の計算をGPUのコア上で並列的に行った場合の処理時間である.

表6及び図3から,12本の列車が運転される路線の車両運用計画の導出において, 単一の母集団で演算を行なう場合にはCPUを用いたほうが高速な処理となっているが, その他の50, 100, 150, 200個の母集団を設定した場合はGPUを用いた処理の方が高速で処理を行えている. 両者の差はサブ母集団の数が大きくなるほど大きくなっていくことがグラフから読み取れる. また, 今回実装したGPUを使用するアルゴリズムでは,200個より多くの母集団を設定した場合に共

表4 intel core i7 2600K 諸元表

項目	数値
コア数	4個
ベースクロック	3400MHz
ブーストクロック	3800MHz
メモリ量	32GB

表5 GeForce GTX-980 諸元表

項目	数値
コア数	2048個
ベースクロック	1126MHz
ブーストクロック	1216MHz
メモリ量	4GB

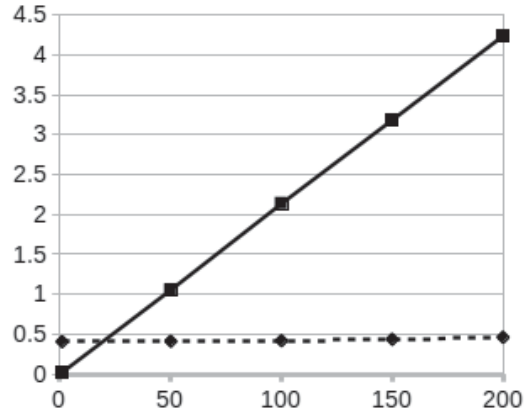

図3 各実行環境における処理時間の比較 (12本の列車が運行される場合)

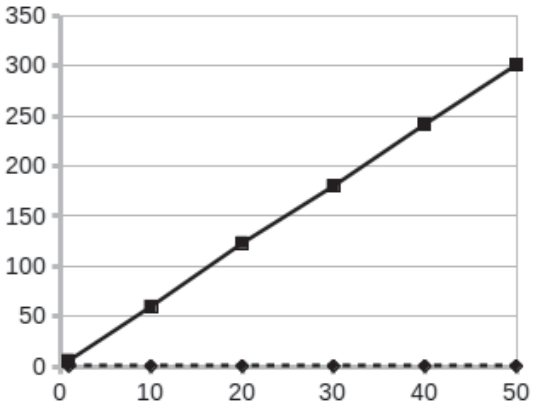

図4 各実行環境における処理時間の比較 (68本の列車が運行される場合)

表 6 島の数による処理速度の比較 (12 本の列車が運行される場合)

島の数	CPU を使用した場合の処理時間	CPU と GPU を併用した場合の処理時間
1	0.023 [s]	0.407 [s]
50	1.054 [s]	0.412 [s]
100	2.136 [s]	0.419 [s]
150	3.183 [s]	0.436 [s]
200	4.236 [s]	0.461 [s]

表 7 島の数による処理速度の比較 (68 本の列車が運行される場合)

島の数	CPU を使用した場合の処理時間	CPU と GPU を併用した場合の処理時間
1	5.859 [s]	0.437 [s]
10	59.860 [s]	0.437 [s]
20	123.131 [s]	0.453 [s]
30	180.149 [s]	0.453 [s]
40	241.809 [s]	0.453 [s]
50	301.079 [s]	0.453 [s]

有メモリの使用量が容量を超過してしまった. この場合, 適切な出力を得ることが出来ないため, 処理時間を計測することが出来なかった. したがって, 表 6 及び図 3 の実験結果では 200 個以上の母集団を設定した場合については省略している.

表 7 及び図 4 から,68 本の列車が運転される路線の車両運用計画の導出においては,1, 10, 20, 30, 40, 50 個の母集団を設定した場合に,GPU を用いた処理の方が高速で処理を行えている. 両者の差はサブ母集団の数が大きくなるほど大きくなっていくことがグラフから読み取れる. また, 今回実装した GPU を使用するアルゴリズムでは,50 個より多くの母集団を設定した場合に共有メモリの使用量が容量を超過してしまった. この場合, 適切な出力を得ることが出来ないため, 処理時間を計測することが出来なかった. したがって, 表 7 及び図 4 の実験結果では 50 個以上の母集団を設定した場合については省略している.

5. 考察

本章では前章の車両運用計画の導出実験についての考察を行なう.

はじめに,12 本の列車が運転される路線での車両運用計画の導出において, 母集団の数が 1 つの際に GPU を併用するよりも CPU を単独で用いた場合の方が処理時間が短い結果となったことについて考察する. GPU を用いる際には, あらかじめ CPU 上から計算に必要となるデータを GPU との共有メモリに格納する必要がある. また,GPU 内のどのコアにどのような計算を割り当てるかなどの定義も行なう必要がある. これらの処理には非常に時間がかかり, ボトルネックとなってしまう. したがって, これらの処理が必要ない CPU による処理の方が高速に処理を行なう結果となったと考えられる. この欠点を補うためには, 計算に用いるデータ量をなるべく小さい形式にして GPU 上に転送することや, より高速にデータ通信を行えるような実行環境を整える必要があると言える.

次に,12 本の列車が運転される路線での車両運用計画の導出において, 母集団の数が 50 個以上の際に GPU と CPU を併用した方が CPU を単独で用いた場合よりも処理時間が短い結果となったことについて考察する. この場合においても上述のボトルネックは存在する. しかし, 今回実装したプログラムにおいては母集団の数が並列度となる.50 個の母集団を設定した場合には GPU 内の 50 個のコアで並列に処理を行なう. 母集団の数が単一の場合は並列処理が行われないため, ボトルネックが足かせとなってしまうが, 並列計算を行なう場合には, その効率がボトルネックによる損失を上回るものとなるため, このような結果になったと考えられる.

また,68 本の列車が運転される路線における車両運用計画の導出においては, 設定したすべての島の数において,GPU を併用した場合に CPU を単独で用いた場合よりも処理時間が短い結果となったことについ

て考察する. 運転される列車の本数が多い場合には, 遺伝的アルゴリズム内での遺伝子配列が長くなる. その場合には個体の環境適応度の計算も複雑になるため, CPU による逐次処理よりも GPU を併用した並列処理のほうが高速に計算を行なう結果になったものと考えられる.

また,12 本の列車が運行される路線と 68 本の列車が運行される路線の両者において, 島の数を増やすことによって, 計算の並列度が上がり,CPU と GPU の処理に要する時間の差は更に広がるものと考えられるが, 前章の結果で示したように列車本数が 12 本の場合には 200 個以上, 列車本数が 68 本の場合には 50 個以上の母集団を設定した場合には, 適切な処理が行えない結果となったため, アルゴリズムの改善や, 実行する環境を変更する必要がある.

6. まとめと今後の課題

本章では車両運用計画を自動導出するにあたって提案した手法について総括し, 今後の課題点について述べる.

本研究では, 車両運用計画作成の自動化の手段として, 島モデル GA の提案, 及び設計と実装を行った. 大規模な車両運用計画の導出においては, 計算効率を高めることは非常に重要であり, その点において, 本研究で示したアルゴリズムは有用であると言える. このシステムは鉄道事業者が車両運用計画を作成する際に有用であると考え, 時刻表によって運行スケジュールが定められている他の交通機関の資源運用に対して応用することも期待できる.

今後の課題としては, 各路線に存在する様々な制約条件への対応が必要である. 本研究で構築したアルゴリズムでは, 他路線への乗り入れや, 逆に他路線から乗り入れてくる列車の存在を考慮していない. 他路線との相互乗り入れを考慮する場合には, 各路線の運用も考慮した上で効率的な運用を求めていかなければならない. また, 各列車の発着駅での留置車両の許容数についても路線によっては考慮する必要がある. 発着駅の規模が小さ い場合には, 1 本や 2 本しか列車が存在できず, このような場合には, 車両運用を計画する際にも多数の車 両が発着駅に存在することのないようにしなければならない. 加えて, 運用計画を作成する路線で複数の車種を使 用している場合には, それぞれの車種ごとに運用を決 める必要がある. このとき, 通勤通学時間帯には乗客収容力が高い車両を使用したり, 保有している車両の 編成数の割合によって, 一日の中の列車の担当数を決めければならない. 他にも現状では, 路線ごとに様々な 運用上の制約条件というものが存在しているが, こう いった制約条件を考慮した運用を計画するアルゴリズ ムの構築には至っていない. 制約条件を加えることによって各路線に適応した車両運用計画を生成するアルゴリズムを構築する必要がある.

参考文献

[1] 椎名航一, 田村啓, 富井規雄: 利用者の視点から最適な運転整理案を迅速に導出するアルゴリズムの開発 (鉄道技術連合シンポジウム (J-Rail) 講演論文集, Vol.19), 電気学会交通 · 電気鉄道技術委員会, pp.53-56(2012).

[2] 今泉淳, 山岸雄樹, 森戸晋: 二段階数理計画アプローチによる鉄道車両運用計画の策定 (日本オペレーションズ · リサーチ学会和文論文誌, Vol.53), 日本オペレーションズ · リサーチ学会, pp.14-29(2010).

[3] 福村直登, 中村達也, 西森進矢, 坂口隆: 車両運用計画自動作成アルゴリズムの開発 (鉄道総研報告, Vol.22), 鉄道総合技術研究所, pp.5-10(2008).

[4] (財) 鉄道総合技術研究所運転システム研究室: 鉄道のスケジューリングアルゴリズム, エヌ · ティー · エス (2005).

[5] 倪永茂: 遺伝的アルゴリズムによる巡回セールスマン問題の一解法 (宇都宮大学国際学部研究論集, Vol.3), 宇都宮大学国際学部, pp.31-40(1997).

[6] Matsumoto Makoto, Nishimura Takuji: Mersenne twister a 623-dimensionally equidistributed uniform pseudo-random number generator(ACM Transactions on Modeling and Computer Simulation (TOMACS), Vol.8), ACM, pp3-30(1998).

[7] 平方敦, 小高知宏, 黒岩丈介, 白井治彦: 車両運用計画最適化への並列処理技術の適用 (情報処理学会第 78 回全国大会講演論文集,Vol.1), 情報処理学会,pp.449-450(2016).