日本オペレーションズ・リサーチ学会和文論文誌
Transactions of the Operations Research Society of Japan
Vol. 53, 2010, pp. 30–55

車両運用計画問題に対する制約充足解法の提案

大槻 知史　　愛須 英之　　田中 俊明
（株）東芝

(受理 2009 年 08 月 10 日; 再受理 2009 年 12 月 15 日)

和文概要

現在多くの鉄道事業者では，経験豊富な熟練者が膨大な時間を割いて一定期間の車両運用の基本計画 を作成し，またダイヤ乱れが発生する度に計画修正している．本稿ではこの車両運用計画作成の自動化を目的とする制約充足解法を提案し，実問題に基づく評価では汎用ソルバー CPLEX よりも高速に求解できることを確認した．また提案解法は基本計画作成・修正計画作成のいずれにも利用可能であり，かつ評価関数の設計の自由度が高いため，多くの鉄道事業者に対し適用可能な汎用解法となる可能性がある．

キーワード: スケジューリング, 探索, アルゴリズム，意思決定, 組合せ最適化, 多品種フロー

1. はじめに

鉄道事業者では，日々の列車ダイヤに対し，所有するいずれかの列車編成を割当てる車両運用を実施している．通常，経験豊富な熟練者が 1ヶ月程度の基本運用計画作成にあたるが，複雑な制約条件を考慮しながらの手作業であるため，数日から数週間程度の膨大な時間を要するのが現状である．

またダイヤ乱れが発生した場合は，翌日以降の運用に支障を生じないように，既に作成済みの基本運用計画を基にして運用計画を修正する．通常，この修正作業は全編成が車両基地に入庫する深夜まで開始できず，かつ翌日の始発までに完了する必要があるため，迅速さと正確さとが共に要求される負荷の高い作業である．

近年，ダイヤの高密度化や鉄道事業者間の相互乗り入れの増加などに伴い運行計画が複雑化している．一方で留置場所の確保の困難さや，無駄な取得・保守コストの抑制の観点から，所有する編成数には余裕がない場合が多く，効率的な運用が求められる．また，経験豊富な熟練者の減少という構造的な問題もあり，作成者にかかる負荷は年々増大している．そのためコンピュータによる支援が強く求められている．

本稿では，この車両運用計画における基本運用計画作成と修正運用計画作成とを自動化することを目的とする制約充足解法を提案する．

以下 2 節では本稿で対象とする車両運用計画の作成の概要と従来研究について述べる．3 節ではモデル化について述べ，4 節では提案解法の詳細を説明する．5 節では，実問題において，多品種フロー問題として定式化し汎用ソルバー CPLEX を用いる場合と提案解法とによる性能の比較結果を示し提案解法の有効性を確認する．6 節では，提案解法の拡張により扱える問題の範囲について述べる．7 節はまとめである．

2. 車両運用計画の作成

2.1. 概要

車両運用計画とは，日々の列車ダイヤに対し，どの列車編成（以下**編成**と呼ぶ) を走らせるかを決定する計画である．通常，車両基地または駅を出庫し所定のダイヤを走行した後に車両基地または駅に入庫するまでの一連の運用単位 (以下，**運用**と呼ぶ) 群を予め作成することが多く，この運用に編成を割当てる形式を取ることが多い．

図 1 は，縦軸に駅，横軸に時刻を配した列車ダイヤ上に，運用の例を実線で示した．平日運用 1 は，車両基地 α を 6 時 15 分に出庫し所定のダイヤを走行した後に車両基地 β に 8 時 10 分に入庫する運用である．また平日運用 2 は，駅 γ を 6 時 20 分に出庫し車両基地 α に 7 時 55 分に入庫する運用である．

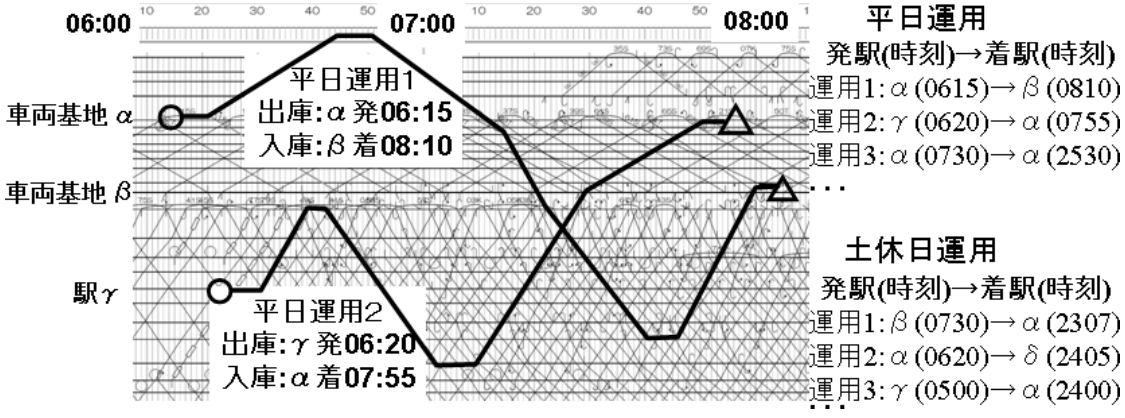

図 1: ダイヤと運用の例

車両運用計画は各編成に，平日または土休日ごとに定義された各運用を漏れなく重複なく割当てる形式をとり，1ヶ月程度ごとに基本計画を作成することが一般的である．これを**基本運用計画作成**と呼ぶ．基本運用計画作成は，図 2 のような，各行が編成を表し各列が日付を表す運用表に，運用や作業予定を記入することにより行う．

ある編成には，1 日に 2 個以上の運用が割当てられる場合があり，逆に 1 個も運用が割当てられない場合もある．なお 1 個も運用が割当てられない場合，編成は 1 日車両基地に滞在することとなり，この状態は**予備**と呼ばれる．

また，ある編成に検査・清掃等の作業が設定される場合は，作業可能な所定の時間帯に作業場所に滞在する必要があるため，当該編成に作業前後に割当て可能な運用が限定される．特に作業の種類によっては，その作業実施日は運用割当てができず，予備としなければならない場合もある．

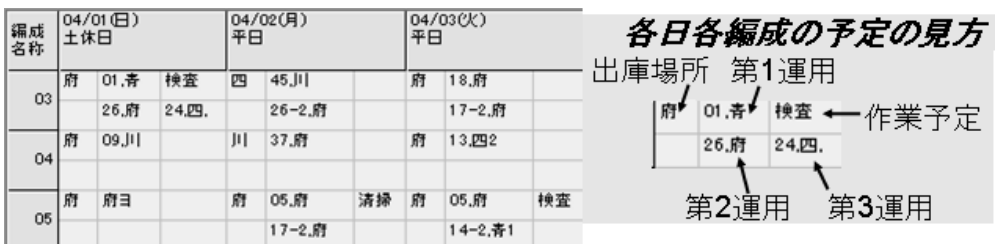

図 2: 運用表の例

一方ダイヤ乱れが発生した場合は，ダイヤの遅延の回復を優先し，一部運休や発着順序変更といった運転整理が行われることが多い [12] ため，ダイヤ上の正常運転を回復した後においても，車両運用計画は乱れたままである場合が多い．この結果として一部の編成が予定された入庫場所に入庫しないことで，翌日以降の運用や作業計画に支障が生じる場合がある．通常，全編成の当該日の最終入庫場所が確定した時点で翌日以降の車両運用計画の見直しが行われる，これを**修正運用計画作成** と呼ぶ．この場合既に作成済みの基本計画の変更コストは大きいため，できるだけ少ない日数で基本計画に戻すことが望まれる．

2.2. 車両運用計画問題

本節では，本稿で扱う車両運用計画問題の定義を与える．基本運用計画作成および修正運用計画作成のいずれも以下の共通の枠組で扱うことが可能である．

- 計画期間の初日を**計画初日**，最終日を**計画最終日**と呼び，計画初日から計画最終日までの計画を作成するものとする．
- 計画期間中に利用可能な編成全体の集合を $\mathcal{H} = \{h_1, h_2, \ldots\}$ とする．
- 各編成を留置可能な場所全体の集合を $\mathcal{R} = \{r_1, r_2, \ldots\}$ とする．これは駅や車両基地に相当する．
- 計画期間の各日には，平日/土休日などの区分に従った一定数の運用 (以下，**運用パターン**と呼ぶ) が定められており，計画初日から計画最終日までの運用全体の集合を $\mathcal{O} = \{o_1, o_2, \ldots\}$ とする．各運用 $o \in \mathcal{O}$ には出庫場所 $\mathbf{DEP}(o) \in \mathcal{R}$，入庫場所 $\mathbf{ARV}(o) \in \mathcal{R}$，出庫時刻 $\mathbf{dept}(o) \in \mathcal{T}$，入庫時刻 $\mathbf{arvt}(o) \in \mathcal{T}$ が定められている．なお $\mathcal{T}$ は時刻の集合である．
- 各編成 $h \in \mathcal{H}$ の計画初日に最初に出庫すべき場所 (**初日出庫場所**と呼ぶ) を $s_h \in \mathcal{R}$ とし，各編成は運用以外の方法では移動できないとする．よって各編成 h に割当てる一連の運用は，時間と場所が接続していなければならない．これを**つなぎ制約** と呼ぶ．
- 修正運用計画作成の場合に，各編成 $h \in \mathcal{H}$ の計画最終日に入庫すべき場所 (**最終日入庫場所**と呼ぶ) を $t_h \in \mathcal{R}$ とする．例えば各編成 h の元の基本計画における計画最終日翌日の出庫場所を t_h とすればよい．これを **最終日制約** と呼ぶ．
- 各編成 $h \in \mathcal{H}$ に割当不可能な運用 $o \in \mathcal{O}$ がある場合，この編成 h と運用 o との組合せ (h, o) 全体の集合を $\mathbf{NGpairs}$ と定める．これを**運用限定制約** と呼ぶ．

以上をまとめると，車両運用計画問題とは，編成 $\{h_1, h_2, \ldots\} \in \mathcal{H}$ のいずれかに，運用 $\{o_1, o_2, \ldots\} \in \mathcal{O}$ 全てを漏れなく重複なく割当て，かつ各編成に割当てた一連の運用が実行可能となる車両運用計画を作成する制約充足問題となる．

このとき，各編成 h の一連の運用列 $(p_{h,1}, p_{h,2}, \ldots, p_{h,N_h})$ は，(2.1) 式，(2.2) 式および (2.3) 式のつなぎ制約，(2.4) 式の最終日制約，および (2.5) 式の運用限定制約を満たさなければならない．なお N_h は編成 h に割当てられた総運用数とし，また時刻 $\tau_1, \tau_2 \in \mathcal{T}$ につき，$\tau_1 < \tau_2$ は時刻 τ_2 が時刻 τ_1 より後であることを表すとする．

$$
\begin{aligned}
s_h &= \mathbf{DEP}(p_{h,1}) && (2.1)\\
\mathbf{ARV}(p_{h,k}) &= \mathbf{DEP}(p_{h,k+1}) \quad (k = 1, 2, \ldots, N_h - 1) && (2.2)\\
\mathbf{arvt}(p_{h,k}) &< \mathbf{dept}(p_{h,k+1}) \quad (k = 1, 2, \ldots, N_h - 1) && (2.3)\\
\mathbf{ARV}(p_{h,N_h}) &= t_h && (2.4)\\
(h, p_{h,k}) &\notin \mathbf{NGpairs} \quad (k = 1, 2, \ldots, N_h) && (2.5)
\end{aligned}
$$

2.3. 従来研究

国外の車両運用計画の自動作成に関する研究は [4] にまとめられている．事例としては，車両運用のスケジューリングと機関車や等級の異なる客車による列車構成とを同時に計画する研究 [1] など，各国固有の問題設定である場合が多い．一方で，国内の鉄道事業者に多く見られる，ダイヤ乱れが多く発生し，かつ余裕のない編成数を運用する問題設定は，見られないようである．解法としては，客車などの構成の計画問題を数理計画に変形するもの [2] や，客車割当問題の性質に着目し Benders 分解法を適用するもの [3] などが提案されている．

国内の車両運用計画の自動作成に関する研究は [13] によくまとめられている．具体的な解法として [5] では，数日単位の周期的な基本運用計画作成を巡回セールスマン問題に帰着する手法について述べられている．この手法は周期的であることを前提としているが，実際の車両運用計画には周期的でない制約条件も少なくない．

一方 [6, 11] では列生成法を用いた解法により短期間の修正運用計画作成に成功する事例が示されている．列生成法は航空機の乗務員スケジュールのような集合被覆問題で有効な解法となることが知られており [9]，最近では，船舶スケジューリング [8] や構内作業計画問題 [7] などの小規模な集合分割問題においても成功例が報告されている．

この列生成法は，列生成部とコスト最小化部に分割することで，コスト最小化部における評価関数の設計に自由度のある点がメリットとして挙げられる．しかし，車両運用計画問題のような集合分割問題に列生成法を適用する場合は，分割条件を加味しながら列を生成するための一般的な手法が知られておらず，本稿のように大規模な問題を対象とする場合は，スケーラビリティーに課題があると考えられる．

3. モデル化

本節では車両運用計画問題を，運用をノード，つなぎ制約をアークで表現したネットワークを用いてモデル化する．このとき 1 つの編成に対する一連のつなぎ制約を満たす運用列はネットワーク上のパスとなる．全てのノードを漏れなく重複なく含む，パス群を構成することが運用計画の必要条件となる．

3.1. 運用ネットワーク

本節では**運用ネットワーク** $\mathcal{N} = (G = (V, E), V_{\mathcal{O}}, V^+_{\mathcal{H}}, V^-_{\mathcal{H}}, \mathbf{DEP}, \mathbf{ARV}, \mathbf{dept}, \mathbf{arvt})$ を次のように定義する．

まずノード集合 V には，各運用に対応する**運用ノード**の集合 $V_{\mathcal{O}}$ の他に，特別のノードとして各編成に対応する**初日ノード**の集合 $V^+_{\mathcal{H}}$ および**最終日ノード**の集合 $V^-_{\mathcal{H}}$ がある．また後の説明のため $V^+ \equiv V_{\mathcal{O}} \cup V^+_{\mathcal{H}}$ および $V^- \equiv V_{\mathcal{O}} \cup V^-_{\mathcal{H}}$ と定義しておく．

次に各ノードに対し，入出庫場所/時刻を定める関数 $\mathbf{DEP} : V \to \mathcal{R}$，$\mathbf{ARV} : V \to \mathcal{R}$，$\mathbf{dept} : V \to \mathcal{T}$，$\mathbf{arvt} : V \to \mathcal{T}$ を，$v \in V_{\mathcal{O}}$ に対しては対応する運用の入出庫場所/時刻の

値を流用して定め，$v_h^+ \in V_{\mathcal{H}}^+ (h = h_1, h_2, \ldots)$ に対しては $\mathbf{DEP}(v_h^+) = \mathbf{ARV}(v_h^+) = s_h$ および $\mathbf{dept}(v_h^+) + 1 = \mathbf{arvt}(v_h^+) = -M$，$v_h^- \in V_{\mathcal{H}}^- (h = h_1, h_2, \ldots)$ に対しては $\mathbf{DEP}(v_h^-) = \mathbf{ARV}(v_h^-) = t_h$ および $\mathbf{dept}(v_h^-) = \mathbf{arvt}(v_h^-) - 1 = M$ となるように定める．ここで M は十分大きい数とする．なお全ての $v \in V$ に対し，$\mathbf{dept}(v) < \mathbf{arvt}(v)$ が成り立つものとする．

さらにノード u とノード v が，$\mathbf{ARV}(u) = \mathbf{DEP}(v), \mathbf{arvt}(u) < \mathbf{dept}(v)$ を共に満たすとき，$u \to v$ 間に有向アーク $(u, v) \in E$ を張る．

以上のネットワークによるモデル化は車両運用計画問題において標準的なものの 1 つである (例えば [6])．

運用ネットワークの模式図を図 3 に示す．以下模式図では，各ノードの左上に出庫場所/時刻を，右上に入庫場所/時刻を表記する．ただし煩雑になる場合は出庫時刻，入庫時刻および有向アークについては適宜省略するものとする．

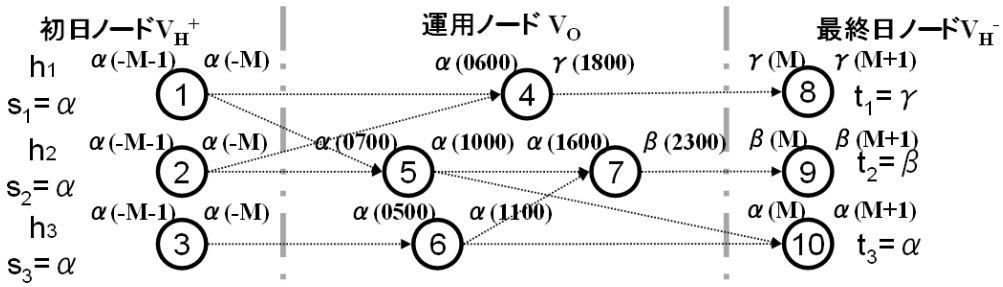

図 3: 運用ネットワークにおける $G = (V, E)$ の例

3.2. 運用ネットワーク上の運用パスと割当の定義

いま各編成 h の一連の運用列 $(p_{h,1}, p_{h,2}, \ldots, p_{h,N_h})$ の直前に，対応する編成 h の初日ノード $p_{h,0} (\equiv v_h^+ \in V_{\mathcal{H}}^+)$ を追加し，直後に対応する編成 h の最終日ノード $p_{h,N_{h+1}} (\equiv v_h^- \in V_{\mathcal{H}}^-)$ を追加したノード列を h の**運用パス** P_h と呼ぶ．このとき P_h は $G = (V, E)$ 上の $v_h^+ \in V_{\mathcal{H}}^+$ を始点とし $v_h^- \in V_{\mathcal{H}}^-$ を終点とするパスとなる．

また運用パス P_h 全体が V の分割となる場合，すなわち運用パスに互いに重複がなく，かつ全てのノード $v \in V$ がいずれかの編成の運用パスに含まれる場合，$v \in V^+$ に対し，

$$\mathbf{next}(v (\equiv p_{h,k})) = p_{h,k+1} \tag{3.1}$$

で定義される全単射 $\mathbf{next} : V^+ \to V^-$ を定めることができる．

この性質を用いて割当を次のように定義する．

定義 3.1. 運用ネットワーク $\mathcal{N}$ において，V^+ から V^- への全単射 $\mathbf{next} : V^+ \to V^-$ が，$\forall v \in V^+, (v, \mathbf{next}(v)) \in E$ を満たすとき，**next** は**割当**であるという．

この **next** を用いて，各編成 h ごとに v_h^+ を始点とし，$v = \mathbf{next}^N(v_h^+)$ が $v \in V_{\mathcal{H}}^-$ となる最小の自然数 N に対する v を終点とする運用パスを構成すると，運用パス群は V のノードを漏れなく重複なく含む．ここで $k \geq 2$ に対し $\mathbf{next}^k(v_h^+) = \mathbf{next}(\mathbf{next}^{k-1}(v_h^+))$ とした．

よって以下では，車両運用計画 A と A から得られる割当 $\mathbf{next}(v)$ を同一視し，$\mathbf{next}_A(v)$ と書くこととする．

例えば図 4 の割当 (A とする) の場合，$V_{\mathcal{H}}^-$ に属するノード $11, 12, 13$ が最終日ノードであり，各編成 h の最後に割当てられた運用ノード $8, 9, 10$ の次に $\mathbf{next}_A(8) = 11$，$\mathbf{next}_A(9) = 12$，$\mathbf{next}_A(10) = 13$ と，異なる最終日ノードを対応させると，$\mathbf{next}_A$ は全単射となる．

実際上段に $v \in V^+$ を下段に対応する $\mathbf{next}_A(v)$ を書く表記を用いると，

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 4 & 5 & 6 & 8 & 9 & 7 & 10 & 11 & 12 & 13 \end{pmatrix} \tag{3.2}$$

のように，下段には V^- の異なる要素が並び全単射となることを確認できる．

このとき，編成 h_1 の運用パス $1 \to 4 \to 8 \to 11$，編成 h_2 の運用パス $2 \to 5 \to 9 \to 12$ および 編成 h_3 の運用パス $3 \to 6 \to 7 \to 10 \to 13$ が得られ，これらの運用パス群は V の分割となる．

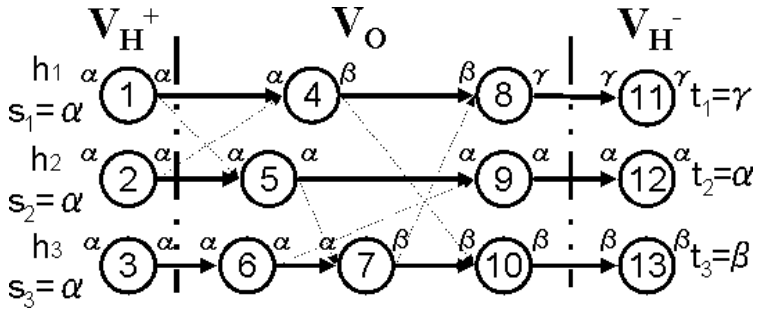

図 4: 運用ネットワーク上の割当の例

3.3. 割当に関連する定義

次に，$\mathbf{NG}_A(h)$ および $\mathbf{NGtot}(A)$ の定義を次のように与える．

定義 3.2. 割当 A において，$\mathbf{NG}_A(h)$ を編成 h の運用パス P_h 上の制約違反の合計数を返す関数とし，$\mathbf{NG}_A(h) > 0$ である編成を **NG 編成**と呼ぶこととする．さらに $\mathbf{NGtot}(A)$ を割当 A における制約違反の合計数，すなわち $\mathbf{NGtot}(A) = \sum_{h \in \mathcal{H}} \mathbf{NG}_A(h)$ のこととする．

割当 A は $\forall v \in V^+, (v, \mathbf{next}(v)) \in E$ であり，必ずつなぎ制約を満たすため，$\mathbf{NG}_A(h)$ は編成 h の最終日制約および運用限定制約の違反箇所数の合計となる．

さらに後の説明のため，次の定義を与える．

定義 3.3. 割当 A_0 と A とで運用パスが異なる編成 h の集合を $\mathbf{diffH}(A_0, A)$ とし，さらに $h \in \mathbf{diffH}(A_0, A)$ かつ $\mathbf{NG}_A(h) > 0$ である編成 h の集合を $\mathbf{diffngH}(A_0, A)$ とする．

定義 3.4. 割当 A において，$\mathbf{h}_A : V \to \mathcal{H}$ を v から $v \in P_h$ となる編成 h への写像とし，$\mathbf{first}_A : \mathcal{H} \to V^+$ を編成 h から編成 h の初日ノードへの写像とする．また編成 h の運用パスにおいてノード v がノード u より後にあることを $u \prec_{A,h} v$ と書くこととする．

3.4. 割当に対する交換の定義と性質

割当 A に対する**交換** $\sigma(u, v)$ を以下のように定義する．

定義 3.5. 割当 A において，4 個の異なるノード $u, v, u'(\equiv \mathbf{next}_A(u)), v'(\equiv \mathbf{next}_A(v)) \in V$ 間に $(u, v') \in E, (v, u') \in E$ および $\mathbf{h}_A(u) \neq \mathbf{h}_A(v)$ なる関係がある場合に，このつなぎを入れ換え，$\mathbf{next}_{A'}(u) = v'$, $\mathbf{next}_{A'}(v) = u'$ となる割当 A' に変換する操作を**交換** $\sigma(u, v)$ と呼ぶ．また，このとき元の割当 A と交換 $\sigma(u, v)$ により得られる割当 A' との関係を $A' = \sigma(u, v) \cdot A$ または $A' = \sigma \cdot A$ と表すものとする．

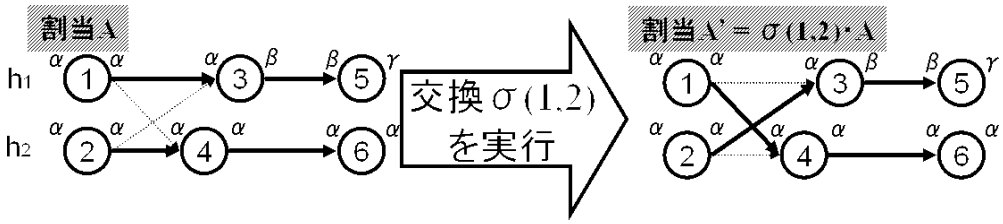

図 5: 交換 σ の例

交換は例えば図 5 のようなつなぎ換えを表し，$\sigma(u,v)$ は，

$$A = \begin{pmatrix} \cdots & u & \cdots & v & \cdots \\ \cdots & u' & \cdots & v' & \cdots \end{pmatrix} \tag{3.3}$$

を，

$$A' = \sigma(u,v) \cdot A = \begin{pmatrix} \cdots & u & \cdots & v & \cdots \\ \cdots & v' & \cdots & u' & \cdots \end{pmatrix} \tag{3.4}$$

に変換する．

この交換 $\sigma(u,v)$ は，割当 A 上の任意の $u,v \in V^+$ に対し定義できるわけではなく，$(u, \mathbf{next}_A(v)) \in E$,$(v, \mathbf{next}_A(u)) \in E$ および $\mathbf{h}_A(u) \neq \mathbf{h}_A(v)$ を満たす必要があるが，逆にこの条件を満たせば，$A' = \sigma(u,v) \cdot A$ は必ず割当となる．なお次の補題に示す通り，$\mathbf{h}_A(u) \neq \mathbf{h}_A(v)$ の条件は不要である．

補題 3.1. 割当 A の 2 個の異なるノード $u,v \in V$ に対し，$(u, \mathbf{next}_A(v)) \in E$ および $(v, \mathbf{next}_A(u)) \in E$ が共に成り立つならば，$\mathbf{h}_A(u) \neq \mathbf{h}_A(v)$ となり，割当 A に対し交換 $\sigma(u,v)$ を定義できる．

この補題 3.1 により，次の定理 3.1 を示せる．

定理 3.1. A_0, A_g を任意の割当とするとき，割当 A_g は割当 A_0 に有限回の交換を適用することにより得られる．

定理 3.1 から，任意の割当に対し，割当 A からの最小交換回数による距離を考えることができる．割当 A から K 回交換以内で得られる割当全体の集合を $\mathbf{Reach}_A(K)$ とおくと，図 6 のように，$A \in \mathbf{Reach}_A(1) \subset \mathbf{Reach}_A(2) \subset \ldots$ なる関係が成り立つ．

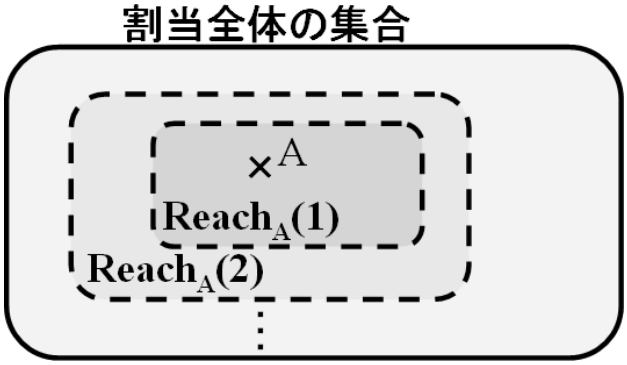

図 6: 割当全体の集合と $\mathbf{Reach}_A(K)$ との関係

また補題 3.1 により，4 節で用いる次の定理 3.2 を示せる．

定理 3.2. $i,j,k,l \in V$ が全て異なるとき，割当 A に対する交換 $\sigma(i,j)$ および，割当 $\sigma(i,j)\cdot A$ に対する交換 $\sigma(k,l)$ が共に定義できるならば，割当 A に対する交換 $\sigma(k,l)$ および，割当 $\sigma(k,l)\cdot A$ に対する交換 $\sigma(i,j)$ も共に定義でき，$\sigma(k,l)\cdot\sigma(i,j)\cdot A = \sigma(i,j)\cdot\sigma(k,l)\cdot A$ が成り立つ．

なおこの補題 3.1，定理 3.1 および定理 3.2 の証明は A 節で与える．

4. 車両運用計画問題の解法

本節では，車両運用計画問題の解法の詳細について説明する．

定理 3.1 では，任意の割当から有限回の交換により，任意の割当に到達可能であることを示した．ただし，ある割当 A において交換 $\sigma(i,j)$ を定義できるノード $i,j \in V^+$ の組は一般には膨大となるため，交換回数を K 回以内に限定した場合であっても，割当 A から K 回交換以内で到達可能な割当全体の集合 $\mathbf{Reach}_A(K)$ のサイズも膨大となる．そこで $\mathbf{NG}_A(h) > 0$ となる NG 編成 h に着目し，$\mathbf{NG}_A(h) = 0$ を目標として必要な割当のみを探索することで，探索空間を限定する方針を採る．

4.1. 提案解法の概要

まず提案解法の概要について説明する．

提案解法は，初期解となる割当を作成するグリーディ割当部 (図 7 左) と，初期解に交換を繰り返し適用し，より $\mathbf{NGtot}(A)$ が小さい割当を探索するバックトラック探索部 (図 7 右) から構成される．

まずグリーディ割当では，各運用ノード $v \in V_{\mathcal{O}}$ を出庫時刻 $\mathbf{dept}(v)$ が小さい順に，割当可能な編成のいずれかに割当てる．

次にグリーディ割当により得られた割当 A_0 を初期解とし，$\mathbf{NG}_{A_0}(h) > 0$ となる編成 h がある場合は，A_0 を初期割当，**NG** 編成 h を起点編成としてバックトラック探索を実行する．このバックトラック探索は，交換閾値 K を 1 から **K_MAX** まで順次増やしながら**成功**となるまで繰り返し実行する．

この初期割当 A_0 の起点編成 h_0 に対し，交換閾値 K 以内で得られる割当 A で $\mathbf{NGtot}(A) < \mathbf{NGtot}(A_0)$ となる割当を得るアルゴリズムを**SMB(Swap-path Multiple Backtracking) アルゴリズム**と呼ぶ．

この交換閾値 K の SMB アルゴリズムの探索範囲を $\mathbf{SMBset}_A(K)$ とおくと，交換閾値に関する繰り返し処理は $\mathbf{SMBset}_A(1), \mathbf{SMBset}_A(2), \ldots$ と順次探索範囲を広げて探索する処理である．この繰り返し処理は，反復深化 [10] と呼ばれる手法であり，少ない記憶領域により，初期割当 A_0 から近い範囲を優先的に探索できることが知られている．

SMB アルゴリズムは，$\mathbf{NGtot}(A) < \mathbf{NGtot}(A_0)$ となる割当 A を発見する場合に**成功**となるため，SMB アルゴリズムの実行前後で，$\mathbf{NGtot}(A)$ の値が狭義単調減少する．よって提案解法は山登り法の一種である．

4.2. グリーディ割当の処理

グリーディ割当では，まず全ての編成の最終ノード $\mathbf{last}(h)$ を v_h^+ に初期化する．

次に各運用ノード $v \in V_{\mathcal{O}}$ につき，出庫時刻 $\mathbf{dept}(v)$ が小さい順に以下の処理を実行する．

運用ノード v に対し，$(\mathbf{last}(h), v) \in E$ を満たす編成 h があれば，編成 h に運用ノード v を割当て $\mathbf{next}(\mathbf{last}(h)) \leftarrow v$ とする．その後に h の最終ノード $\mathbf{last}(h)$ を v に更新する．

この処理を全ての運用ノード $v \in V_{\mathcal{O}}$ について繰り返す．

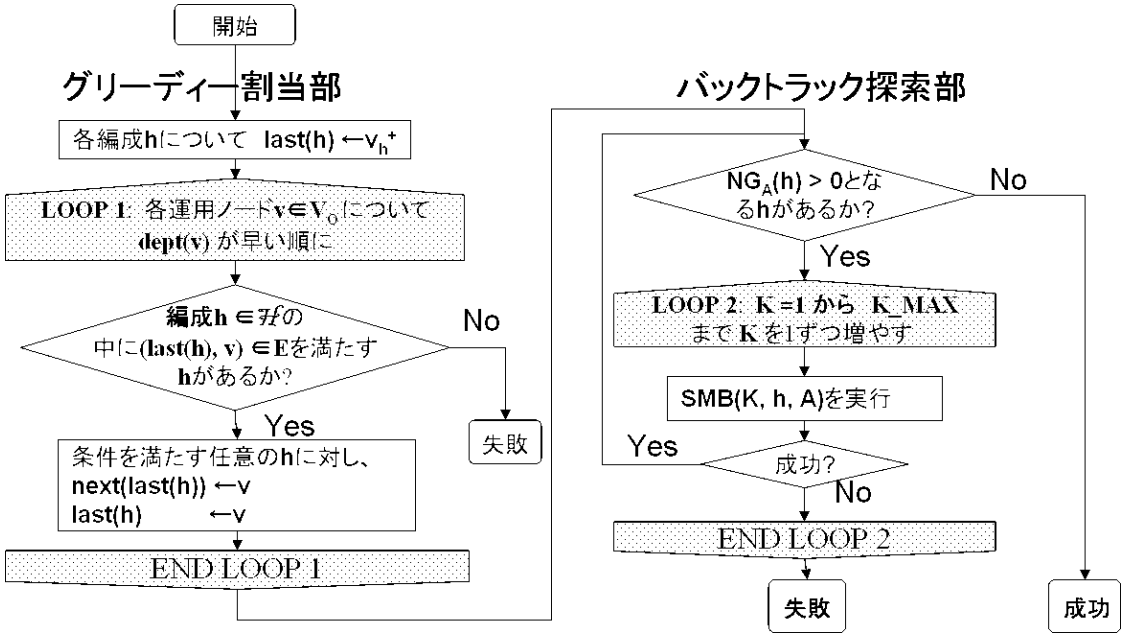

図 7: 解法全体のフローチャート

定理 4.1 により，つなぎ制約を満たす割当が存在するならば，グリーディ割当は必ず割当の発見に成功する．

定理 4.1. 運用ノード $\{u_1, u_2, \ldots\} \in V_{\mathcal{O}}$ を編成 $\{h_1, h_2, \ldots\} \in \mathcal{H}$ に漏れなく重複なく割当てる割当が存在するならば，グリーディ割当は必ず成功する．

4.3. SMB(K, h, A) によるバックトラック探索の処理

本節では，バックトラック探索の処理を図 8 のフローチャートおよび図 9 の適用例を用いて説明する．

まず図 7 のフローに従って，グリーディ割当の結果図 9 左上のような割当 A_0 が得られたものとする．この割当では，編成 h_1 が最終日制約を満たさないため，起点編成を h_1 として，バックトラック探索関数 $\mathbf{SMB}(K, h_1, A_0)$ を $K = 1$ から解を発見するまで順次 K を増やしながら繰り返し実行する．なお以下の例では，交換閾値 $K = 2$ の場合を説明する．

このバックトラック探索では，起点編成 h の初日ノード v_h^+ を始点とする $G = (V, E)$ 上のパス p の内，交換回数 K 回以下のもの (**交換パス**と呼ぶ) を順に探索する．なお交換パス $p = v_1 \rightarrow v_2 \rightarrow \ldots$ 上の隣接するノードの内，$\mathbf{h}_A(v_k) \neq \mathbf{h}_A(v_{k+1})$ かつ $(\mathbf{next}_A^{-1}(v_{k+1}), \mathbf{next}_A(v_k)) \in E$ を満たす $(v_k, v_{k+1}) \in E$ を**交換アーク**と呼び，交換パス p 上の交換アークの数を交換パス p の**交換回数** $S(p)$ と呼ぶ．

図 9 左上の割当 A の例では，起点編成 $h = h_1$，交換閾値 $K = 2$ である場合は，起点編成 h_1 の 初日ノードであるノード 1 を始点とする交換回数 2 回以下の交換パスを探索する．ノード 1 を始点とするパスには，$p_1 = 1 \rightarrow 5 \rightarrow 9$ や，$p_2 = 1 \rightarrow 5 \rightarrow 7 \rightarrow 10$ などがあり，p_1 は $1 \rightarrow 5$ が交換アークであるため $S(p_1) = 1$，p_2 は $1 \rightarrow 5$ および $5 \rightarrow 7$ が交換アークであるため $S(p_2) = 2$ の交換パスとなる．

この交換パス p 上の全ての交換アーク $(u_k, v_k) \in E$ $(k = 1, 2, \ldots, K'(\leq K))$ に関し，$(\mathbf{next}_A^{-1}(v_k), \mathbf{next}_A(u_k)) \in E$ となる場合，補題 3.1 より $\mathbf{h}_A(u_k) \neq \mathbf{h}_A(v_k)$ となり，交換 $\sigma_k(u_k, \mathbf{next}_A^{-1}(v_k))$ を定義できる (図 10) ため，A_0 に σ_k を順に適用した割当 $A' = \sigma_{K'} \cdots \sigma_2 \cdot$

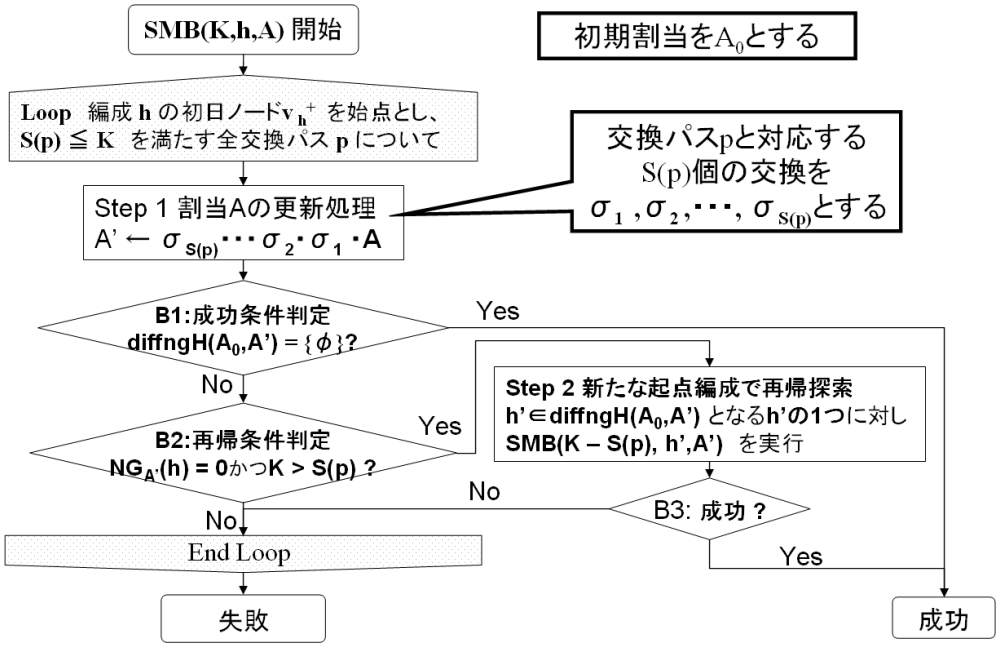

図 8: バックトラック探索のフローチャート

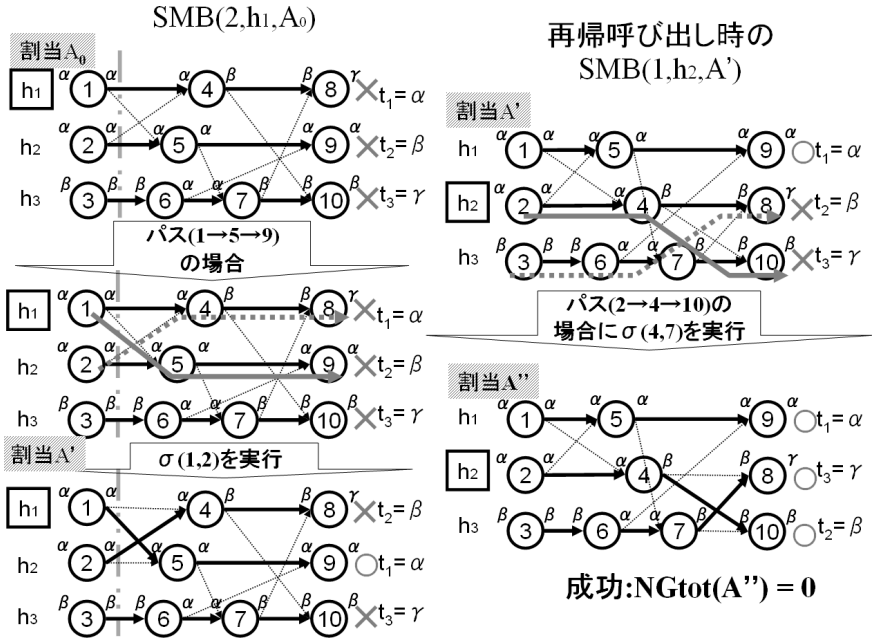

図 9: バックトラック探索の適用例

$\sigma_1 \cdot A_0$ を作成する．

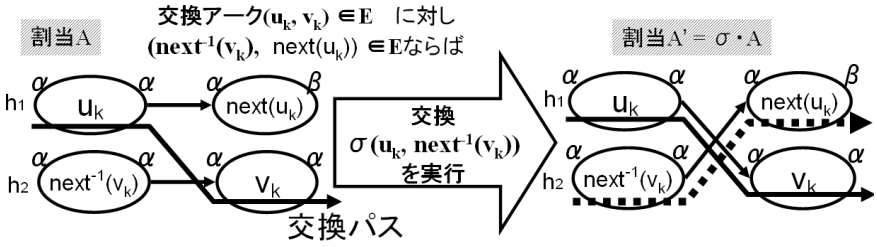

図 10: 交換パスと交換の対応の例

例えば，交換パス p_1 では交換アーク $1 \to 5$ に対応して $\sigma_1 = \sigma(1,2)$ が得られ，この交換 σ_1 により割当 $A' = \sigma_1 \cdot A$ が得られる (図 9 左下)．

この A' と A_0 において運用パス P_h が異なる全ての編成集合 $h' \in \mathbf{diffH}(A_0, A')$ に対し，**成功条件**

$$\text{成功条件}: \quad \sum_{h' \in \mathbf{diffH}(A_0,A')} \mathbf{NG}_{A'}(h') = 0 \quad (\Leftrightarrow \mathbf{diffngH}(A_0, A') = \phi) \tag{4.1}$$

を満たす場合は**成功**となり終了する．ここで $\mathbf{NG}_A(h)$ の値は，運用パス P_h 上のノード列のみによって決定されるため，A_0 と A' とで運用パス P_h が同一である h については，必ず $\mathbf{NG}_{A_0}(h) = \mathbf{NG}_{A'}(h)$ となる．また起点編成 $h \in \mathbf{diffH}(A_0, A')$ に対し $\mathbf{NG}_{A_0}(h) > 0$ であるため，初期割当 A_0 と成功解 A' とを比較すると，必ず $\mathbf{NGtot}(A') < \mathbf{NGtot}(A_0)$ となることに注意する．

一方割当 A' において，$\mathbf{diffngH}(A_0, A') \neq \phi$ であり成功条件を満たさない場合であっても，起点編成 h が以下の**再帰条件**

$$\text{再帰条件}: \quad \mathbf{NG}_{A'}(h) = 0 \ \text{ かつ } \ K > S(p) \tag{4.2}$$

を満たす場合は，新たに $h' \in \mathbf{diffngH}(A_0, A')$ を満たす **NG** 編成の 1 つを起点編成 h' とし，かつ交換閾値を $K - S(p)$ として，関数 **SMB** を再帰呼び出しし，交換閾値を $K - S(p)$，起点編成を h' としたバックトラック探索を実行する．

図 9 左下の例では，新たに得られた割当 A' において，$\mathbf{diffH}(A_0, A') = \{h_1, h_2\}$ の内，$\mathbf{NG}_{A'}(h_1) = 0$ ではあるが $\mathbf{NG}_{A'}(h_2) > 0$ であるため**成功**とはならないが，$K > S(p) (\Leftrightarrow 2 > 1)$ であるため図 8 の **Step 2** に進み，今度は編成 $h' (\equiv h_2)$ の交換閾値を $K - S(p) (\Leftrightarrow 2 - 1)$ として関数 $\mathbf{SMB}(K - S(p), h', A')$ を再帰呼び出しする．

この再帰呼び出し時の割当 A' からは，起点編成を h_2，交換閾値を $K'' = 1$ としてバックトラック探索を実行する (図 9 右上)．ここで交換パス $p_3 = 2 \to 4 \to 10 (S(p_3) = 1)$ があり，p_3 に基づく交換を実行すると，図 9 右下のような割当 A'' が得られる．この A'' では $\mathbf{diffH}(A_0, A'') = \{h_1, h_2, h_3\}$ に属する全ての h' に対し $\mathbf{NG}_{A''}(h') = 0$ となるため**成功**となる．

一方，再帰呼び出しされた関数 **SMB** が**失敗**となる場合や元々 $\mathbf{NG}_{A'}(h) = 0$ とならない場合は，次の交換パス候補へと進む．

このようにバックトラック探索では，$G = (V, E)$ 上の起点編成 h の初日ノード v_h^+ を始点とする交換パス p を $S(p)$ 回の交換に対応付けられることに着目し，交換候補の探索をパス探索に帰着する．また **NG** 編成を起点編成とする交換を優先的に探索することで，効率的な探索を行う．

4.4. $\mathbf{SMBset}_A(K)$ と $\mathbf{Reach}_A(K)$ との関係

本節では，$\mathbf{SMB}(K,h,A)$ の探索範囲である $\mathbf{SMBset}_A(K)$ と割当 A から K 回交換以内で得られる割当全体の集合 $\mathbf{Reach}_A(K)$ との関係について述べる．

まず図 8 のフローチャートの **Step 2** において，$\mathbf{SMB}(K-S(p),h',A')$ を再帰呼び出しする場合，割当 A から割当 A' を得るために $S(p)$ 回の交換を実行しているのに対し，交換閾値は丁度 $S(p)$ 減少する．よって初期割当 A_0 から交換閾値 K の $\mathbf{SMB}(K,h_0,A_0)$ による探索を行った場合，探索範囲に含まれる割当は A_0 から K 回以内の交換で得られる割当に限られる．

したがって

$$\mathbf{SMBset}_{A_0}(K) \subseteq \mathbf{Reach}_{A_0}(K) \tag{4.3}$$

が成り立つ．

実際，成功条件を満たす割当 A_g を得るまでに起点編成となった編成を順に $h_0,h_1,\ldots,h_n$ とし，それぞれに対応する交換パスを $p_0,p_1,\ldots,p_n$，各交換パス p_k に対応する交換を順に $\sigma_{k,1},\sigma_{k,2},\ldots,\sigma_{k,S(p_k)}$ とするとき，

$$A_g = \overbrace{\sigma_{n,S(p_n)}\cdots\sigma_{n,1}}^{p_n\text{と対応する交換}}\cdots\overbrace{\sigma_{1,S(p_1)}\cdots\sigma_{1,1}}^{p_1\text{と対応する交換}}\cdot\overbrace{\sigma_{0,S(p_0)}\cdots\sigma_{0,1}}^{p_0\text{と対応する交換}}\cdot A_0 \tag{4.4}$$

$$S(p_0)+S(p_1)+\ldots+S(p_n) \le K \tag{4.5}$$

が成り立つ．

逆に，

$$A_g = \sigma_K(u_K,v_K)\cdots\sigma_2(u_2,v_2)\cdot\sigma_1(u_1,v_1)\cdot A_0 \tag{4.6}$$

$$(u_1,u_2,\ldots,u_K,v_1,v_2,\ldots,v_K\text{は全て異なる}) \tag{4.7}$$

が成り立ち，かつある編成 h_0 につき $\mathbf{NG}_{A_0}(h_0)>0$ である場合を考える．

このとき，定理 3.2 により任意の 2 つの σ_k は交換可能であるため，A_g における h_0 の運用パス上のノードを含む交換を先頭に移動できる．この交換を $\sigma_{0,1},\sigma_{0,2},\cdots,\sigma_{0,K'}$ とすると，割当 A_0 においてこの交換と対応する交換パス p_0 が存在するため，この交換により得られる割当 $A'=\sigma_{0,K'}\cdots\sigma_{0,1}\cdot A_0$ は，$A'\in\mathbf{SMBset}_{A_0}(K')$ を満たし，A' は $\mathbf{SMB}(K',h_0,A_0)$ の探索範囲に含まれることが分かる．

以下同様の議論を繰り返すと，次の定理 4.2 を示すことができる．証明は A.5 節に示す．

定理 4.2. 割当 A_0 において，$\mathbf{NG}_{A_0}(h_0)>0$ とする．このとき $\mathbf{NGtot}(A_g)=0$ を満たす割当 A_g の内，A_0 からの交換回数が最小となるものの交換回数が K 回であり，

$$A_g = \sigma_K(u_K,v_K)\cdots\sigma_2(u_2,v_2)\cdot\sigma_1(u_1,v_1)\cdot A_0 \tag{4.8}$$

とする．このとき $u_1,u_2,\ldots,u_K$ および $v_1,v_2,\ldots,v_K$ が全て異なるならば，$\mathbf{SMB}(K,h_0,A_0)$ は**成功**となる．

4.5. 成功解を発見できない場合の例

一方，割当 A_0 と A_g との間に (4.6) 式の関係があっても，(4.7) 式の条件を満たさない場合には，$A_g\in\mathbf{Reach}_{A_0}(K)$ であっても，$A_g\notin\mathbf{SMBset}_{A_0}(K)$ となる場合がある．この場合の $\mathbf{Reach}_A(K)$ と $\mathbf{SMBset}_A(K)$ の関係の模式図を図 11 に示す．

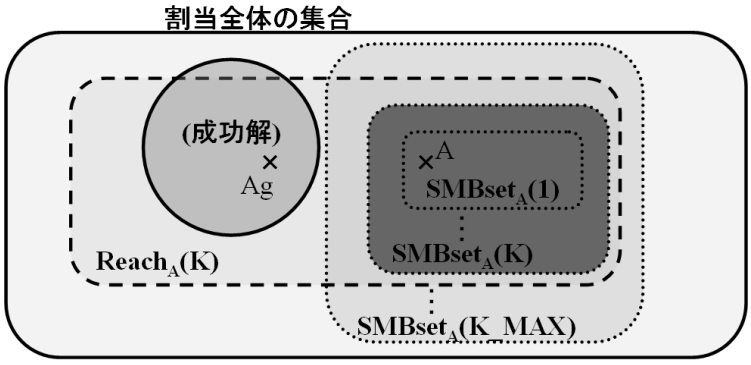

図 11: $\mathbf{SMBset}_A(K)$ が $\mathbf{Reach}_A(K)$ にある成功解の探索に失敗するケースの模式図

実際図 12 の例では，右上図のような $\mathbf{NGtot}(A_g) = 0$ となる割当 A_g が，

$$A_g = \sigma_2(1,2) \cdot \sigma_1(1,3) \cdot A_0 \tag{4.9}$$

を満たすため $A_g \in \mathbf{Reach}_{A_0}(2)$ が成り立つが，左上の初期割当 A_0 から h_1 を起点編成とする $\mathbf{SMB}(2, h_1, A_0)$ による探索では，右上の割当 A_g を得られない．なぜならこの場合，割当 A' を得るために交換アーク $(1,5) \in E$ を考えたいが，$(2,4) \notin E$ となるため，(1,5) と対応する交換を定義できないからである．

一方，図 13 の例の場合も，右上図のような $\mathbf{NGtot}(A_g) = 0$ となる割当 A_g が，

$$A_g = \sigma_2(1,3) \cdot \sigma_1(1,2) \cdot A_0 \tag{4.10}$$

を満たすため $A_g \in \mathbf{Reach}_{A_0}(2)$ が成り立つが，左上の初期割当 A_0 から h_1 を起点編成とする $\mathbf{SMB}(2, h_1, A_0)$ による探索では，右上図の割当 A_g を得られない．なぜならこの場合，割当 A' を得たいために交換アーク $(1,4) \in E$ を考えたいが，$\mathbf{h}_{A_0}(1) = \mathbf{h}_{A_0}(4) = h_1$ であるため，アーク (1,4) と対応する交換が定義できないからである．

以上のように $\mathbf{SMB}(K, h, A)$ は，$\mathbf{Reach}_A(K)$ の範囲に成功解があっても，必ずしも成功解を発見できない場合がある．この不完全性を補うため，実際の求解の際には 5.1.3 節で述べる多点探索手法により，$\mathbf{SMB}(K, h, A)$ を補完した．

ただし本節で紹介した失敗事例は，編成数が少なく，かつ **NGpairs** により可能な交換パスが極端に限定される作為的な事例である．$\mathbf{SMB}(K, h, A_0)$ が成功解を発見できないのは，成功条件を満たす A_g 全てに対し，いかなる交換パスによっても (4.4) 式の形式の交換演算列を発見できない場合に限られる．実際の事例では，複数の編成間には膨大な交換パスの組合せがあるため，解を発見できないケースは稀である．

実際の問題において，解の発見に失敗する頻度と，多点探索手法の効果については，5 節の評価実験において改めて述べる．

4.6. SMB アルゴリズムの高速化

前節までの説明では，簡単のため $\mathbf{SMB}(K, h, A)$ は $\mathbf{SMBset}_A(K)$ に属する割当全てを探索範囲としたが，集合 $\mathbf{SMBset}_A(K)$ のサイズは通常 $O(C^K)$(C は定数) であり，計算量も $O(C^K)$ と膨大になる．実際，以下に述べる高速化処理を省いた場合，5.3 節の実験設定と同様の予備実験では，求解に数 10 分以上要する事例や現実的な時間内に求解できない事例が多く生じることを確認した．そこで，次に述べるような高速化を実施した．

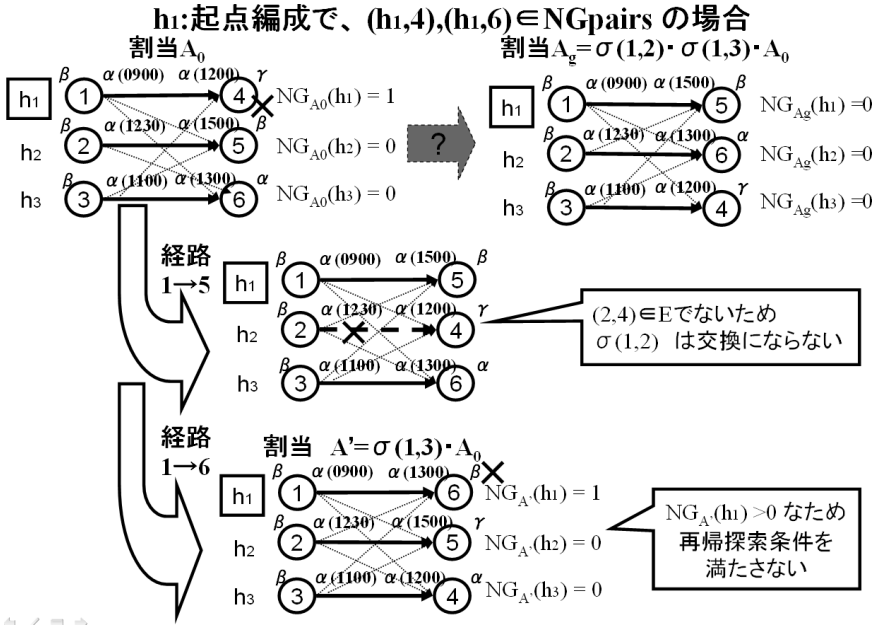

図 12: SMB アルゴリズムでは割当 A_g に到達できない場合 (1)

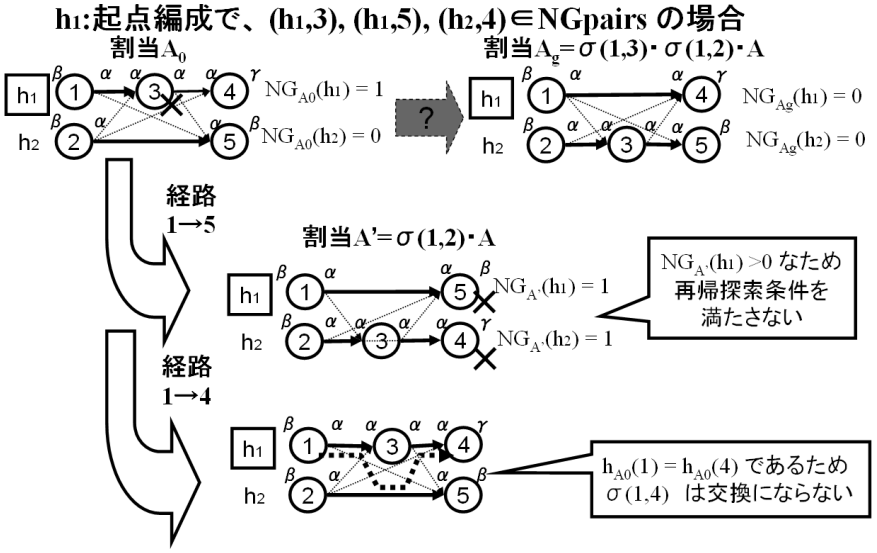

図 13: SMB アルゴリズムでは割当 A_g に到達できない場合 (2)

4.6.1. 成功判定条件の緩和

4.3 節では，**成功**判定条件を $\mathbf{diffngH}(A_0, A_g) = \phi$ とすることで，成功解 A_g が $\mathbf{NGtot}(A_g) < \mathbf{NGtot}(A_0)$ を満たし，SMB アルゴリズムの実行前後で $\mathbf{NGtot}(A)$ の値が狭義単調減少することを保証した．

一方この狭義単調減少性を保証するためには，成功条件を以下のように，狭義単調減少条件そのものとしても良い．

$$\text{成功条件：}\quad \mathbf{NGtot}(A) < \mathbf{NGtot}(A_0) \tag{4.11}$$

成功条件を (4.11) 式に変える場合，(4.1) 式よりも条件が緩和されるため，**成功**に至るまでに必要な交換回数 K' が減少することが期待できる．

実際，成功条件である (4.11) 式を満たすために必要な交換回数が K から $K'(< K)$ に減少すると，$\mathbf{SMB}(K, h, A)$ の計算量が $O(C^K)$ から $O(C^{K'})$ に減少することが期待できる．

4.6.2. 枝刈りの導入

$\mathbf{SMB}(K_0, h, A_0)$ の探索範囲 $\mathbf{SMBset}_{A_0}(K_0)$ は，割当 A_0 から K_0 回以内の交換により到達できる割当全体の集合である．よって $K_0 - K$ 回の交換を実行し，残り交換回数が K 回となった割当 A において，残り K 回の交換により**成功**となることが期待できない場合に枝刈りすることは有効である．

例えば交換回数 1 回あたりの $\mathbf{NGtot}(A)$ の変化量の期待値を $E(\delta\mathbf{NG})$ とする場合に，初期割当 A_0 から $K_0 - K$ 回の交換を実行した割当 $A = \sigma_{K_0-K} \cdots \sigma_2 \cdot \sigma_1 \cdot A_0$ が，

$$\text{枝刈り条件：}\quad \mathbf{NGtot}(A) - \mathbf{NGtot}(A_0) > K \cdot E(\delta\mathbf{NG}) \tag{4.12}$$

を満たす場合は，割当 A に残り K 回の交換を実行しても，成功条件 $\mathbf{NGtot}(A) < \mathbf{NGtot}(A_0)$ を満たすことは困難と考えられるため，枝刈りすることが有効であると考えられる．なお $E(\delta\mathbf{NG})$ の値は，問題の性質に基づき事前に与えられるものとする．

この枝刈り条件を導入するためには，図 8 のフローチャートの **Step 1** の A の更新処理の際に，交換 1 回ごとに枝刈り条件判定を行い，k 番目までの交換を実行して得られた割当 $A = \sigma_k \cdots \sigma_2 \cdot \sigma_1 \cdot A_0$ が枝刈り条件を満たした場合は，当該交換パスを候補とせず枝刈りする処理を追加すれば良い．なおこの割当 A が枝刈り条件を満たす場合，最初の k 回の交換が $\sigma_1, \sigma_2, \ldots, \sigma_k$ と一致する全ての交換パスを枝刈りできる．

この結果，始点から k 番目の交換アークまでを部分パスとして含む全ての交換パスの探索が不要となることで，$\mathbf{SMB}(K, h, A)$ の計算量 $O(C^K)$ の指数の底 C が $C'(< C)$ に減少し，探索時間が $O(C^K)$ から $O(C'^K)$ に減少することが期待できる．

以上 2 種類の高速化処理を実施した $\mathbf{SMB}(k_0, h_0, A_0)$ の擬似コードを A.6 節に示した．これらの高速化によっても計算量は交換閾値 K に関する指数オーダのままではあるが，5 節で述べるように現実的な時間での求解が可能となる．

5. 評価実験

本節では，実問題に基づく問題設定において，提案解法と CPLEX による解法の求解時間の比較を行い，提案解法のスケーラビリティーを評価する．

5.1. 提案解法の実装上の工夫

本節では SMB アルゴリズムを用いた提案解法の実装上の工夫について述べる．

5.1.1. 予備ノードの導入とアークの制限

まず予備ノードの導入とアークの制限について述べる.

3.1 節の運用ネットワークの定義では，つなぎ制約を満たす全てのノード u, v 間にアーク $(u, v) \in E$ を張ることとしたが，このままではアークの総数が膨大である．そこで，運用ノード v の出庫時刻 $\mathbf{arvt}(v)$ と入庫時刻 $\mathbf{dept}(v)$ が必ず同一日 (v の**実施日**と呼ぶ) の時刻であることを利用し，グラフ $G = (V, E)$ の代わりに次に定義するグラフ $G' = (V', E')$ を考えることとした.

まず d 日目場所 r の予備に相当する予備ノードを v_{dr}，予備ノード集合を V_y とし，V に V_y を加えたノード集合を V' とした．ここで予備ノード v_{dr} の入出庫場所は $\mathbf{DEP}(v_{dr}) = \mathbf{ARV}(v_{dr}) = r$ とした．また v_{dr} の入出庫時刻は，d 日目の別のノードと接続不要なことから，$\mathbf{dept}(v_{dr}) = (d$ 日目の十分早い時刻) および $\mathbf{arvt}(v_{dr}) = (d$ 日目の十分遅い時刻) と定めた.

またノード $u, v(\in V')$ 間のアーク $(u, v) \in E'$ は，$\mathbf{arvt}(u) < \mathbf{dept}(v)$ および $\mathbf{DEP}(u) = \mathbf{ARV}(v)$ を満たすものの内，さらに**同日つなぎ** (ノード u と v の実施日が同一) または**翌日つなぎ** (ノード u の実施日がノード v の実施日の前日) であるものに制限した.

5.1.2. 1 日ごとの計画作成による工夫

計画初日を d_0，計画最終日を d_f とする場合に，予備実験により，仮の計画最終日 d' を d_0 から d_f まで 1 日ずつ増やし，運用を都度加える方針が有効であることが分かった．具体的には，各日 d' 日目ごとに，d' 日目の運用ノードを加えるグリーディ割当と，d' 日目までの制約を全て満たす解を発見するためのバックトラック探索とを行い，この処理を d' を 1 日ずつ増やしながら繰り返すこととした.

5.1.3. 多点探索手法による工夫

4.5 節で述べた通り，SMB アルゴリズムは全ての制約を満たす成功解がある場合に，必ずしも成功解を発見できるとは限らないため，多点探索を行うこととした．具体的には，バックトラック探索による探索パス数が閾値 X を超えた場合に**失敗**とみなし，改めて計画初日から再試行する.

なおグリーディ割当においてある運用ノード v を割当可能な候補編成が複数ある場合には，擬似乱数により割当編成を決定することとしたため，再試行の際は異なる探索結果が得られる.

5.2. 多品種フロー定式化による解法

2.2 節で定義した車両運用計画問題は，多品種フロー問題の一種であり，(5.1) 式から (5.6) 式のような 0-1 整数計画問題として定式化できる.

変数は $x_u^h \in \mathbf{x}$, $f_{uv}^h \in \mathbf{f}$ の 2 種類で，x_u^h は編成 h にノード u が割当てられるか否かを表す 0-1 変数, f_{uv}^h は編成 h にノード u の次にノード v が割当てられるか否かを表す 0-1 変数とする.

(5.1) 式は目的関数であり，各編成における予備となる日数の総和の最大化を表す．(5.2) 式はつなぎ制約，最終日制約を，(5.3) 式は運用限定制約を，(5.4) 式は各運用を漏れなく重複なくいずれかの編成に割当てる制約をそれぞれ表す.

$$\max \quad \sum_{h\in\mathcal{H}}\sum_{u\in V_{\mathcal{Y}}} x_u^h \tag{5.1}$$

$$\text{s.t.} \quad \sum_{v:(v,u)\in\mathbf{E}'} f_{vu}^h - \sum_{v:(u,v)\in\mathbf{E}'} f_{uv}^h = \begin{cases} -1 & \text{if } u = v_h^+ \\ +1 & \text{if } u = v_h^- \\ 0 & \text{otherwise} \end{cases} \quad \forall u \in V,\ \forall h \in \mathcal{H}, \tag{5.2}$$

$$x_u^h = 0 \qquad \forall (h,u) \in \mathbf{NGpairs}, \tag{5.3}$$

$$\sum_{h\in\mathcal{H}} x_u^h = 1 \qquad \forall u \in V_{\mathcal{O}}, \tag{5.4}$$

$$x_u^h = \sum_{v:(u,v)\in\mathbf{E}'} f_{uv}^h \qquad \forall u \in V^+, \forall h \in \mathcal{H}, \tag{5.5}$$

$$f_{uv}^h \in \{0,1\} \qquad \forall u,v \in V, \forall h \in \mathcal{H}. \tag{5.6}$$

なお上記の定式化のままでは変数の数が膨大となり求解が困難になる．例えば M 日目の運用の内，入庫場所が場所 r である運用の数を K とおき，$M+1$ 日目の運用の内，出庫場所が場所 r である運用の数も K とすると，場所 r に関する M 日目から $M+1$ 日目に至る翌日つなぎとなるアークの総数は K^2 となる．

そこで，まず各日 d 各場所 $r \in \mathcal{R}$ ごとに定義した仮想的な中間日ノード $w_{dr} \in V_M$ を追加し，各編成のフローは d 日目の最後には必ず最終場所 r に相当する中間日ノード w_{dr} を通ることとした．また，元の翌日つなぎに相当するアーク全てを消去し，代わりに中間日ノードを経由するアークを導入することとした (図 14)．

この工夫により，M 日目から $M+1$ 日目に至る翌日アークの総数を K^2 から $2K$ に削減することができる．この結果 5.3 節の実験において運用数が最大の b 路線においても，変数の総数 $|\mathbf{f}|$ を 25 万程度まで抑えられる効果がある (表 1)．

制約充足解を得る自然な定式化としては，(5.3) 式の制約を除き，代わりに運用限定制約の違反箇所数の最小化を目的関数とするものが考えられる．しかし，5.3 節の実験設定と同様の予備実験では，求解時間が 10 分を超える事例や現実的な時間内に求解できない事例が多く発生した．そのため本節では，求解時間が最小となった最も有効なものとして，予備日数の総和の最大化を目的関数とした定式化をとりあげた．

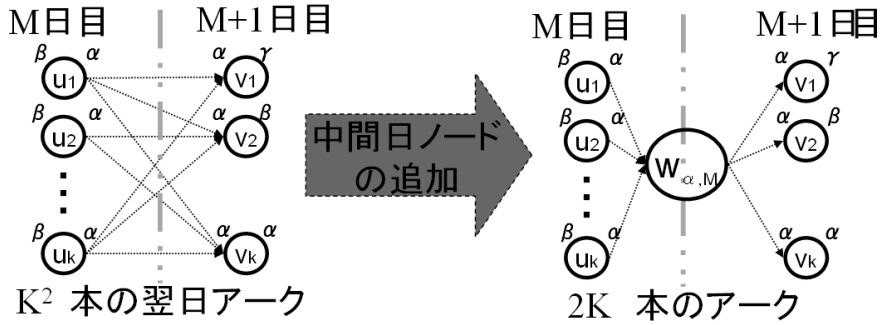

図 14: 中間日ノードの導入の例

5.3. 実験方法

本節では実験方法について説明する．

まず，できるだけ実問題に近いテストセットを作成するため，国内の複数鉄道事業者に属する計 9 路線の実際の 14 日分の実計画データを元に次の手順で実験用のテストセットを作成した．

まず各路線の運用パターンとしては，平日/ 土休日ごとに定義された実際の運用データを使用した．次に計画初日 d_0 と計画最終日 d_f を適当に定め (実計画データが 1 から 14 日目までなので，$d_0 \leq d_f$ となる (d_0, d_f) の組合せは計 $\binom{14+1}{2} = 105$ 種類ある)，実計画データの d_0 日目の出庫場所を初日出庫場所とし，実計画データの d_f 日目の入庫場所を最終日入庫場所とした．

なおテストセット中の d 日目編成 h において，検査・清掃・工場入場など（以下，**作業** w と呼ぶ）が設定されている場合には，d 日目編成 h に割当可能な運用は運用限定制約により限定した．具体的には，各路線の実計画データ全体における当該作業 w の実施日 (実計画データ中の他の日や他の編成も含む) に割当てられた実績がある運用 (または予備) のみに限定し，それ以外の運用は **NGpairs** に追加した．

以上により定まるテストセットに対しては実際の運用データが解の 1 つとなるため，つなぎ制約，最終日制約，運用限定制約を満たす解の存在が保証される．よってこの 105 種類 ×9 路線のテストセットに対し，

- SMB アルゴリズムに 5.1 節の工夫を加えたもの (以下 SMB と略す)
- 5.2 節の定式化を行い，CPLEX を用いて求解したもの (以下 CPLEX と略す)

の 2 種類による求解を行い，成功回数と成功の場合の求解時間を測定した．

SMB において 1 日ごとのバックトラック探索における最大パス探索数 X の値は 1 日ごとに定めるものとし，計画最終日 d_f 日目を除き 32 万，計画最終日 d_f 日目に限り 320 万とした．計画最終日の最大パス探索数 X の値を 10 倍の値としたのは，計画最終日に限り最終日制約があり，それ以外の日よりも求解が困難であると想定されるためである．最大パス探索数の値をこのように定めた場合，SMB アルゴリズムは毎秒数 100 万オーダのパス探索が可能であるため，1 回あたりの試行時間は，計画期間が 14 日間の場合であってもほぼ数秒程度である．5.1.3 節で述べた通り，この各回の試行が**失敗**した場合は，計画初日からの計画を全て消去し，異なる擬似乱数による再試行を行う．

一方，本実験は制約充足解求解時間の比較を目的としているため，CPLEX においても最初の許容解発見までの時間を計測すべきである．ここでは予備実験の実行ログにより，全ての場合につき最初の許容解発見までの時間が総実行時間となることが分かったため，総実行時間を CPLEX の求解時間として採用した．

なお実験に使用した PC は Intel Xeon CPU 3.20GHz，2.00GB RAM であり，CPLEX は CPLEX 10.1.0 である．

5.4. 実験データ

実験に使用した路線 a から路線 i の実際の運用データの内容を表 1 に示す．

$|\mathcal{H}|$ は編成数，$|O_W|$ は平日運用数，$|O_H|$ は休日運用数，$|\mathcal{R}|$ は留置される駅または車両基地の総数，$|\mathbf{NGpairs}|$ は計画期間が 14 日間の場合の **NGpairs** に含まれる (h, o) ペアの総数，$|\mathbf{f}|$ は 5.2 節の CPLEX による定式化において計画期間が 14 日間の場合の変数の総数を表す．

5.5. 実験結果

実験結果を表 2，表 3 および図 15 に示す．

まず表 2 の失敗個数を見ると，SMB の成功率が 100%であるのに対し，CPLEX では失敗する事例があることが分かった．CPLEX の失敗原因は路線 b において，変数の総数が 15 万程度を越えた場合にメモリオーバーフローとなるためである．本実験では計画期間を最大

表 1: 実験に使用した各路線データの内容

路線名	$\|\mathcal{H}\|$	$\|O_W\|$	$\|O_H\|$	$\|\mathcal{R}\|$	$\|\mathbf{NGpairs}\|$	$\|\mathbf{f}\|$
a	約 40	約 50	約 30	約 20	15,098	75,924
b	約 60	約 70	約 30	約 15	29,101	239,658
c	約 40	約 50	約 20	約 10	14,166	137,080
d	約 50	約 60	約 30	約 15	15,578	127,680
e	約 40	約 40	約 30	約 20	18,185	97,656
f	約 40	約 60	約 40	約 10	15,862	123,144
g	約 30	約 20	約 10	約 10	5,598	16,956
h	約 20	約 30	約 20	約 15	6,685	34,125
i	約 20	約 20	約 20	約 5	1,753	8,490

14 日間としたが，期間長に比例して変数の数は増大するため，路線 b 等では，これ以上長い期間の計画作成は困難であると考えられる．

表 2: SMB と CPLEX による求解時間の内訳

求解時間 (sec)	～0.1	～1.0	～10.0	～100.0	100.0～	失敗	合計
SMB	800	119	25	1	0	0	945
CPLEX	282	216	212	171	46	18	945

表 3: SMB の再試行回数の内訳

	0	1	2	3	4	5～	合計
再試行回数 (回)	898	36	5	4	2	0	945

また時間レンジごとの求解時間の内訳を見ると，SMB では 90%以上のケースで 1.0(sec) 以内で求解できる一方，CPLEX では 100 秒以上要するケースも少なくない．以上の結果は，SMB の性能が平均的には CPLEX の性能を上回ることを示している．

次に SMB の再試行回数の内訳 (表 3) を見ると，90%以上のケースで再試行なしで**成功**に至ることが分かった．また 1 回の試行で**成功**しない場合についても，最大 4 回の再試行で**成功**に至ることが分かった．このことは SMB アルゴリズムが探索に失敗するケースは多くなく，仮に失敗する場合にも多点探索手法と組合せることで，提案解法が制約充足解を発見できる可能性が高いことを示唆している．

最後に表 2 を詳細化した，路線ごとの求解時間の散布図を図 15 に示す．各散布図は，縦軸を SMB による求解時間，横軸を CPLEX による求解時間とし，各軸を 0.01(sec) から 100.0(sec) の範囲とした両対数グラフである．グラフの右下側の点は SMB が CPLEX よりも早く求解できたことを表し，左上側の点は CPLEX が SMB よりも早く求解できたことを表す．個々の事例の中には，CPLEX の求解時間が SMB よりも短いケースがあるが，これは SMB では，成功解を得るための最小交換回数が大きい場合に，長時間を要するためである．

一方 SMB と CPLEX の平均的な求解時間を比較すると，全ての路線において SMB は CPLEX よりも概ね 10～100 倍高速に求解できることが分かった．

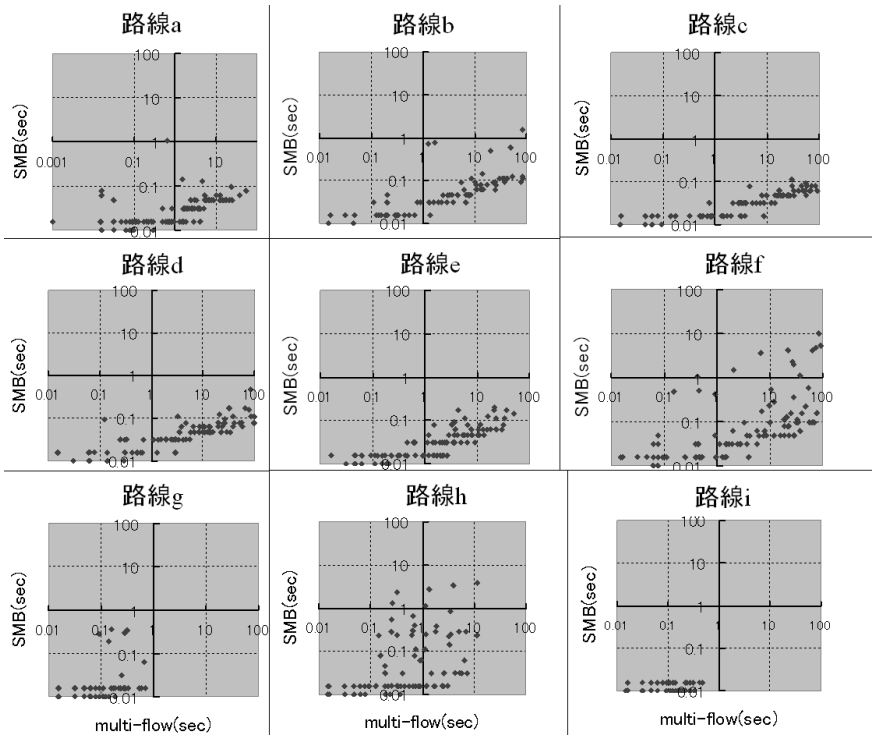

図 15: 路線 a から路線 i に関する SMB と CPLEX の実験結果

6. NGpairs により表現可能な制約条件と運用限定制約の拡張に関して

前節までの説明では，制約違反箇所数 $\mathbf{NG}_A(h)$ を編成 h の運用パス P_h 上のノード $p_{h,k}$ の内 $(h, p_{h,k}) \in \mathbf{NGpairs}$ となるものの数とした．この編成と運用の組 $(h, o) \in \mathbf{NGpairs}$ による単純な制約表現を用いた場合でも，例えば次のような実問題の制約条件を表現できる．

- 編成数の違い，装置の有無などに伴う特定の編成 h に対する割当可能な運用 o の限定
- 特定日特定編成に割当てる運用の決め打ち
- 工場入場や新車の導入，廃車の実施などに伴う計画期間内の総編成数の増減
- 予め入出庫時刻/場所が決まっている回送や臨時運用の特定の編成に対する割当て

一方提案解法では，運用限定制約を $\mathbf{NG}_A(h)$ を通してのみ利用しているため，運用限定制約を，運用パス P_h 上のノード $p_{h,1}, p_{h,2}, \ldots, p_{h,N_h}$ と h の組合せで表現できる任意の制約条件に，容易に拡張できる．

この場合バックトラック探索の探索効率が低下する可能性があるが，P_h 上の途中のノードで条件違反を判定できる場合には途中ノードで探索を打ち切るなどの，個々の制約条件に応じた高速化が可能である．

この拡張により，例えば次のような制約条件も表現できる．

- 同一編成に対する n 日以上の予備の連続の禁止
- 同一編成に対する 1 日 n 個以上の運用割当の禁止
- 編成ごとの走行距離の上下限

このように $p_{h,1}, p_{h,2}, \ldots, p_{h,N_h}$ の組合せによる制約表現は，大きな表現力を有するため，提案解法の評価関数は設計の自由度が大きいと考えられる．

7. おわりに

本稿では最初に，車両運用計画を運用ネットワーク上のパス群で表現し，このパス群に対し定義した交換演算とパス探索との対応関係に着目した探索解法である SMB アルゴリズムを提案した．提案解法は，実問題に基づくテストセットによる評価では，汎用ソルバー CPLEX による解法と比べ，10〜100 倍程度高速に求解できることを確認した．また提案解法は制約条件の表現力が高く，多くの実問題の制約を表現可能であることを確認した．

一方，SMB アルゴリズムは理論上は成功解の発見に失敗する可能性があるため，多点探索手法により補完した．この結果，本稿の実験では成功率が 100%となり，複数回の試行が必要な場合でも最大 4 回以下の再試行により解を得られることが分かった．必ず成功解を発見できるように SMB アルゴリズムを拡張することは今後の課題である．

提案解法は非常に高速であり，また評価関数の設計自由度が大きいことから，多くの鉄道事業者に対し適用可能な汎用解法となり得ると考えられる．

なお，ここでは SMB アルゴリズムを制約充足問題の解法としたが，アルゴリズムの終了条件を緩和することで，目的関数を有する最適化問題の解法に拡張できる可能性がある．また本報告では，検査・清掃などの構内作業計画は所与のものとして扱った．これらの計画には回帰周期等の一定のルールがあり，制約条件として扱うことが可能である．今後 SMB アルゴリズムの高速性を生かし，構内作業と運用の同時計画への拡張を検討したい．

参考文献

[1] E. Abbink, V.D. Berg, L. Kroon, and M. Salomon: Allocation of railway rolling stock for passenger trains. *Transportation Science*, **38-1** (2004), 33–41.

[2] A. Alfieri, R. Groot, L. Kroon, and A. Schrijver: Efficient circulation of railway rolling stock. *Transportation Science*, **40-3** (2006), 378–391.

[3] J. Cordeau, F. Soumis, and J. Desrosiers: A benders decomposition appoach for the locomotive and car assignment problem. *Transportation Science*, **34-2** (2000), 133–149.

[4] J. Cordeau, P. Toth, and D. Vigo: A survey of optimization models for train routing and scheduling. *Transportation Science*, **32-4** (1998), 380–404.

[5] 福村直登, 中村達也, 西森進矢, 坂口隆: 鉄道の車両運用計画作成アルゴリズム. スケジューリングシンポジウム 2008 予稿集 (2008), 167–172.

[6] 加藤怜, 今泉淳: ダイヤ乱れ時における運用計画修正問題に対する列生成アプローチ. スケジューリングシンポジウム 2008 予稿集 (2008), 145–150.

[7] 北古賀圭祐, 今泉淳, 重田英貴, 森戸晋: 機関車の基地内留置計画に対する整数計画アプローチ. スケジューリングシンポジウム 2008 予稿集 (2008), 161–166.

[8] 小林和博, 久保幹雄: 船舶スケジューリング. RAMP シンポジウム論文集 (2008), 61–75.

[9] L. Lettovsky, E.L. Johnson, and G.L. Nemhauser: Airline crew recovery. *Transportation Science*, **34-4** (2000), 337–348.

[10] S. Russell and P. Norvig: Artificial Intelligence: A Modern Approach (2nd edition)(Prentice Hall, 2003).

[11] 佐藤圭介, 福村直登: 輸送障害時の車両スケジュール変更問題. 日本オペレーションズ・リサーチ学会 2008 年秋季研究発表会予稿集 (2008), 192–193.

[12] 高橋理, 片岡健司, 小島央士, 浅見雅之: ダイヤ乱れ時における列車乗務員運用整理案の自動作成. 電気学会論文誌 D, **128-11** (2008), 1291–1297.

[13] 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 (株式会社エヌ・ティー・エス, 2005).

A. 付録

A.1. 補題 3.1 の証明

補題 3.1: 割当 A の異なる 2 ノード $u, v \in V$ に対し，$(u, \mathbf{next}_A(v)) \in E$ および $(v, \mathbf{next}_A(u)) \in E$ が共に成り立つならば，$\mathbf{h}_A(u) \neq \mathbf{h}_A(v)$ となり，割当 A に対し $\sigma(u, v)$ が定義できる．

Proof. まず $\mathbf{h}_A(u) = \mathbf{h}_A(v)$ と仮定して矛盾を導く．

一般性を失わず $u \prec_{A,h} v$ とする．このとき $\mathbf{next}_A(u) = v$ または $\mathbf{next}_A(u) \prec_{A,h} v$ であるが，いずれの場合も $(v, \mathbf{next}_A(u)) \in E$ と矛盾する．よって $\mathbf{h}_A(u) \neq \mathbf{h}_A(v)$ である．

いま $(u, \mathbf{next}_A(v)) \in E$, $(v, \mathbf{next}_A(u)) \in E$, $\mathbf{h}_A(u) \neq \mathbf{h}_A(v)$ の 3 条件が成り立つため，割当 A に対し交換 $\sigma(u, v)$ が定義できる． □

A.2. 定理 3.1 の証明

定理 3.1: A_0, A_g を任意の割当とするとき，割当 A_g は割当 A_0 に有限回の交換を適用することにより得られる．

Proof. $n \in V^+$ の内，$\mathbf{next}_{A_0}(n) \neq \mathbf{next}_{A_g}(n)$ を満たす n の集合を $V_{A_0}^{\mathbf{diff}}$ とし，この集合 $V_{A_0}^{\mathbf{diff}}$ のサイズ $K \equiv |V_{A_0}^{\mathbf{diff}}|$ に関する帰納法により証明する．

まず $K = 0$ の場合は $A_0 = A_g$ なので明らかに題意を満たす.

次に $K = k - 1$ 以下の場合に題意を満たすとし $K = k$ の場合を考える．

いま $n \in V_{A_0}^{\mathbf{diff}}$ の内，$\mathbf{arvt}(n)$ 最大のものの 1 つを u とおく．また簡単のため $u' \equiv \mathbf{next}_{A_0}(u)$, $v' \equiv \mathbf{next}_{A_g}(u)$, および $v \equiv \mathbf{next}_{A_0}^{-1}(v')$ とおく．明らかに u, u', v, v' は互いに異なる．

最初に，割当 A_g において $v' = \mathbf{next}_{A_g}(u)$ であることから $(u, v') \in E$ と分かる．

次にノード v, u' 間のつなぎ制約を考える．いま割当 A_0 では $\mathbf{next}_{A_0}(v) = v'$ であるが，割当 A_g では $v' = \mathbf{next}_{A_g}(u) (\neq \mathbf{next}_{A_g}(v))$ であり，$\mathbf{next}_{A_0}(v) \neq \mathbf{next}_{A_g}(v)$ となるため，$v \in V_{A_0}^{\mathbf{diff}}$ である．よって $\mathbf{arvt}(u)$ の最大性から，$\mathbf{arvt}(v) \leq \mathbf{arvt}(u)$ である．一方 $(u, u') \in E$ より $\mathbf{arvt}(u) < \mathbf{dept}(u')$ なので，合わせて $\mathbf{arvt}(v) < \mathbf{dept}(u')$ が言える．

また $(v, v'), (u, v'), (u, u') \in E$ より，$\mathbf{ARV}(v) = \mathbf{DEP}(v') = \mathbf{ARV}(u) = \mathbf{DEP}(u')$ となり，$\mathbf{ARV}(v) = \mathbf{DEP}(u')$ も成立するため，$(v, u') \in E$ である．

以上 $(u, v'), (v, u') \in E$, および補題 3.1 より $\sigma(u, v)$ は交換となる．

ここで A_0 では $u, v \in V_{A_0}^{\mathbf{diff}}$ であったのに対し，$A' = \sigma(u, v) \cdot A_0$ では $\mathbf{next}_{A'}(u) = \mathbf{next}_{A_g}(u)$ であり $u \notin V_{A'}^{\mathbf{diff}}$ であるため，$|V_{A'}^{\mathbf{diff}}| \leq k - 1$ となる．よって帰納法の仮定により，この A' に対しては題意を満たす交換がある．

よって，$K = k$ の場合も題意は示された． □

A.3. 定理 3.2 の証明

定理 3.2: $i,j,k,l \in V$ が全て異なるとき，割当 A に対する交換 $\sigma(i,j)$ および，割当 $\sigma(i,j)\cdot A$ に対する交換 $\sigma(k,l)$ が共に定義できるならば，割当 A に対する交換 $\sigma(k,l)$ および，割当 $\sigma(k,l)\cdot A$ に対する交換 $\sigma(i,j)$ も共に定義でき，$\sigma(k,l)\cdot\sigma(i,j)\cdot A = \sigma(i,j)\cdot\sigma(k,l)\cdot A$ が成り立つ．

Proof. いま $i' \equiv \mathbf{next}_A(i)$, $j' \equiv \mathbf{next}_A(j)$, $k' \equiv \mathbf{next}_A(k)$, および $l' \equiv \mathbf{next}_A(l)$ とおく．

このとき，A および $A'' = \sigma(k,l)\cdot\sigma(i,j)\cdot A$ は次のように書ける

$$A = \begin{pmatrix} \cdots & i & \cdots & j & \cdots & k & \cdots & l & \cdots \\ \cdots & i' & \cdots & j' & \cdots & k' & \cdots & l' & \cdots \end{pmatrix} \tag{A.1}$$

$$A'' = \sigma(k,l)\cdot\sigma(i,j)\cdot A = \begin{pmatrix} \cdots & i & \cdots & j & \cdots & k & \cdots & l & \cdots \\ \cdots & j' & \cdots & i' & \cdots & l' & \cdots & k' & \cdots \end{pmatrix} \tag{A.2}$$

いま，A'' が割当であることから $(k,l') \in E$, $(l,k') \in E$ が成り立つため，補題 3.1 より A に対し，交換 $\sigma(k,l)$ を定義できる．次に $A' = \sigma(k,l)\cdot A$ とおくと A' は次のように書ける．

$$A' = \sigma(k,l)\cdot A = \begin{pmatrix} \cdots & i & \cdots & j & \cdots & k & \cdots & l & \cdots \\ \cdots & i' & \cdots & j' & \cdots & l' & \cdots & k' & \cdots \end{pmatrix} \tag{A.3}$$

同様に A'' が割当であることから $(i,j') \in E$, $(j,i') \in E$ が成り立つため，再び補題 3.1 より A' に対し，交換 $\sigma(i,j)$ を定義できる．この結果 $A'' = \sigma(i,j)\cdot A' = \sigma(i,j)\cdot\sigma(k,l)\cdot A$ が割当となるため，題意は示された． □

A.4. 定理 4.1 の証明

定理 4.1: 運用ノード $\{u_1,u_2,\ldots\} \in V_{\mathcal{O}}$ を編成 $\{h_1,h_2,\ldots\} \in \mathcal{H}$ に漏れなく重複なく割当てる割当が存在するならば，グリーディ割当は必ず成功する．

Proof. いまグリーディ割当において運用 v_0 の割当に失敗したとする．この失敗時点までに決定された写像 **next** を **next**： $V'' \to V'_{\mathcal{O}} - \{v_0\}$ とする．なお $V'', V'_{\mathcal{O}} - \{v_0\}$ はそれぞれ **next** の定義域および値域とし，**next** の決め方から $V'_{\mathcal{O}}$ は $\{v' \in V_{\mathcal{O}} | \mathbf{dept}(v') \leq \mathbf{dept}(v_0)\}$ となる．この場合に $v \in V'_{\mathcal{O}}$ を漏れなく重複なく割当てる割当が存在する，すなわちある $V'^{+}(\subseteq (V^{+}_{\mathcal{H}} \cup V'_{\mathcal{O}}))$ に対し $(v, \mathbf{next_0}(v)) \in E$ となる全単射 $\mathbf{next_0}: V'^{+} \to V'_{\mathcal{O}}$ が存在すると仮定し矛盾を導く．

いま $\mathbf{PreSET}(v)$ を $\mathbf{PreSET}(v) = \{u | (u,v) \in E\}$ と定義し，$\mathbf{PreSET}(v_0)$ に属するノードを $u_1,u_2,\ldots,u_k$ とする．このときグリーディ割当が v_0 の割当に失敗したことから，任意の $u_l \in \mathbf{PreSET}(v_0)$ は **next** の定義域 V'' に含まれる．なぜならそうでなければ，$\mathbf{next}(u_l) \leftarrow v_0$ としてグリーディ割当は成功するからである．以下では $v_l \equiv \mathbf{next}(u_l)$ とおく．

このときグリーディ割当の手続きから，v_0 は $v \in V'_{\mathcal{O}}$ の中で $\mathbf{dept}(v)$ が最大であるため，任意の $v_l (l = 0,1,\ldots,k)$ に対し $\mathbf{PreSET}(v_l) \subseteq \mathbf{PreSET}(v_0)$ が成り立つ．

ここで $\mathbf{next}_0$ は全単射であり，また $l = 0,1,2,\ldots,k$ に対し $\mathbf{next}_0^{-1}(v_l) \in \mathbf{PreSET}(v_l)$ $(\subseteq \mathbf{PreSET}(v_0))$ であるため，写像 $\mathbf{next}_0^{-1}$ は，$k+1$ 個の異なる要素である $v_l \in V'_{\mathcal{O}}$ を $\mathbf{PreSET}(v_0)$ の異なる要素に写す必要がある．ところが集合 $\mathbf{PreSET}(v_0)$ の要素数は k であることからこれは不可能である．よって矛盾し題意は示された． □

A.5. 定理 4.2 の証明

定理 4.2: 割当 A_0 において，$\mathbf{NG}_{A_0}(h_0) > 0$ とする．このとき $\mathbf{NGtot}(A_g) = 0$ を満たす割当 A_g の内，A_0 からの交換回数が最小となるものの交換回数が K 回であり，

$$A_g = \sigma_K(u_K, v_K) \cdots \sigma_2(u_2, v_2) \cdot \sigma_1(u_1, v_1) \cdot A_0 \tag{A.4}$$

とする．このとき $u_1, u_2, \ldots, u_K$ および $v_1, v_2, \ldots, v_K$ が全て異なるならば，$\mathbf{SMB}(K, h_0, A_0)$ は**成功**となる．

Proof. K に関する帰納法により証明する．

まず $K = 0$ の時は $A_0 = A_g$ なので明らかに題意を満たす．

次に $K = k - 1$ 以下の時に題意を満たすとし，$K = k$ の場合を考える．

$\sigma_l(u_l, v_l)(l = 1, 2, \ldots, k)$ の内，$\mathbf{h}_{A_g}(u_l) = h_0$ または $\mathbf{h}_{A_g}(v_l) = h_0$ を満たす σ_l が l_0 個ある (以下では一般性を失わず u_l が $\mathbf{h}_{A_g}(u_l) = h_0$ を満たすとする) とき，この l_0 個の σ_l を $\mathbf{dept}(u_l)$ の昇順に並べ替え，残りの $k - l_0$ 個の σ_l をこれらの後ろに並べた交換の列を，改めて $\sigma'_1(u'_1, v'_1), \sigma'_2(u'_2, v'_2), \ldots, \sigma'_k(u'_k, v'_k)$ とおくと，$u_1, u_2, \ldots, u_k$ および $v_1, v_2, \ldots, v_k$ が全て異なることと定理 3.2 とにより任意の 2 つの交換演算 σ_l の交換が可能であり，

$$\begin{aligned} A_g &= \sigma_k(u_k, v_k) \ldots \sigma_{l_0}(u_{l_0}, v_{l_0}) \cdots \sigma_1(u_1, v_1) \cdot A_0 \\ &= \sigma'_k(u'_k, v'_k) \cdots \sigma'_{l_0}(u'_{l_0}, v'_{l_0}) \cdots \sigma'_1(u'_1, v'_1) \cdot A_0 \end{aligned} \tag{A.5}$$

と並べ替えられる．

このとき $\sigma'_1, \sigma'_2, \ldots, \sigma'_{l_0}$ の作り方から，割当 A_0 において $v^+_{h_0}$ を始点として $(u'_l, \mathbf{next}(v'_l))(l = 1, 2, \ldots, l_0)$ を交換アークとする交換パス p が存在する．ここで $\mathbf{NG}_{A_0}(h_0) > 0$ であるため，A_0 から A_g を得るためには少なくとも $\mathbf{h}_{A_0}(u) = h_0$ となるノード u を含む交換 $\sigma(u, v)$ が 1 回以上必要であり $l_0 > 0$ であることに注意する．

$\mathbf{SMB}(l_0, h_0, A_0)$ では $v^+_{h_0}$ を始点とする交換回数 l_0 回以内の全ての交換パスを探索空間に含むため，候補となる前に**成功**となる場合を除き，必ず交換パス p が候補となる．

いま交換パス p により得られる割当を $A' = \sigma'_{l_0} \cdots \sigma'_1 \cdot A_0$ とすると，$k - l_0 = 0$ の場合は $A' = A_g$ であり，A' は成功条件 $\mathbf{NGtot}(A') = 0$ を満たす．

一方 $k - l_0 > 0$ の場合は，$\mathbf{NG}_{A'}(h_0) = 0$ かつ $\mathbf{diffngH} \notin \phi$ となるため，$\mathbf{NG}_{A'}(h') > 0$ となる新しい起点編成 h' があり，次は $\mathbf{SMB}(k - l_0, h', A')$ を再帰呼び出しする．

ここで割当 A' から割当 A_g を得るための最小交換回数は $k - l_0 (< k)$ 回であり，$A_g = \sigma'_k \cdots \sigma'_{l_0+2} \cdot \sigma'_{l_0+1} \cdot A'$ が成り立つため，帰納法の仮定より $\mathbf{SMB}(k - l_0, h', A')$ は**成功**となる．

よって $\mathbf{SMB}(K, h_0, A_0)$ は必ず**成功**となり，$K = k$ のときも題意は成り立つ． □

A.6. SMB アルゴリズムの擬似コード

本節では SMB アルゴリズムの擬似コードを示す．擬似コードは **leaf**，**SMBsearch**，および **SMB** の 3 関数からなる．初期割当を A_0，起点編成を h_0，交換閾値を K_0 とする場合，最初に $\mathbf{SMB}(K_0, h_0, A_0)$ を呼び出し，**成功**の場合は A_g に最終結果が得られるものとする．**SMBsearch** 関数は深さ優先順に交換パスを探索する関数であり，**leaf** 関数は成功条件の判定と必要なら起点編成を変更して **SMB** 関数を再帰呼び出しする関数である．

なお **pop**(**SET**) を集合 **SET** の中の要素 1 個を取り出す関数とする．

Algorithm 1 **leaf**(K, A)

Require: K:残り交換回数，A:現割当，A_0:初期割当

if $(\mathbf{NGtot}(A) < \mathbf{NGtot}(A_0))$ **then**
  $A_g \leftarrow A$
  return **成功**
end if
$h_{new} \leftarrow \mathbf{pop}(\mathbf{diffngH}(A_0, A))$
return $\mathbf{SMBsearch}(K, \mathbf{first}_A(h_{new}), A)$

Algorithm 2 **SMBsearch**(K, v, A)

Require: K:残り交換回数，v:現ノード，A:現割当，A_0:初期割当

if $(\mathbf{NG}_A(\mathbf{h}_A(v)) == 0$ **and** $\mathbf{leaf}(K, A) ==$ **成功**$)$ **then**
  return **成功**
end if
if $(K \leq 0$ **or** $\mathbf{NGtot}(A) - \mathbf{NGtot}(A_0) > K \cdot E(\delta\mathbf{NG}))$ **then**
  return **失敗**
end if
if $(\mathbf{SMBsearch}(K, \mathbf{next}_A(v), A) ==$ **成功**$)$ **then**
  return **成功**
end if
for all $w \in \{w' | (v, w')$ が交換アーク$\}$ **do**
  if $(\mathbf{SMBsearch}(K-1, w, \sigma(v, w) \cdot A) ==$ **成功**$)$ **then**
    return **成功**
  end if
end for
return **失敗**

Algorithm 3 **SMB**(K_0, h_0, A_0)

Require: A_0:初期割当 $(\mathbf{NG}_{A_0}(h_0) > 0)$，$K_0$:交換閾値，$h_0$:起点編成

Ensure: 返り値は **成功** or **失敗** (**成功**の場合は $\mathbf{NGtot}(A_g) < \mathbf{NGtot}(A_0)$ を満たす)

return $\mathbf{SMBsearch}(K_0, \mathbf{first}_{A_0}(h_0), A_0)$

大槻 知史
(株) 東芝 研究開発センター
〒 212-8582 川崎市幸区小向東芝町 1
E-mail: tomoshi1.otsuki@toshiba.co.jp

ABSTRACT

THE SOLUTION FOR CONSTRAINT SATISFACTION PROBLEMS OF RAILWAY ROLLING STOCK ALLOCATION

Tomoshi Otsuki Hideyuki Aisu Toshiaki Tanaka
Toshiba Corporation

Recently in many railway companies, experts have spent a lot of time and effort on developing plans of rolling stock allocation for certain periods, and on modifying these plans whenever railroad services are disrupted. In this paper, we propose a search-based constraint satisfaction method for railway rolling stock allocation, to solve the problem based on practical examples faster than CPLEX. Our algorithm is efficient both for developing and modifying plans, and its evaluation function is very flexible, so that it can be generally applicable for many railway companies.