船舶スケジューリング数理モデル作成の具体的手順

小林　和博

本稿では，船舶スケジューリングと呼ばれる海上物流の実務問題を扱う．この問題を整数計画問題にモデル化する具体的な方法を解説し，さらに実務的な制約条件を扱う際の注目点と，それを実際に数理モデルに反映する手順を順を追って説明する．対象としては，最短路問題を学び，C 言語などのプログラミング言語を用いてプログラムを開発した経験のある大学学部生程度を想定している．

キーワード：船舶スケジューリング，列生成法，集合分割問題

1. 船舶スケジューリング

製造業や流通業など，業務のなかで貨物の輸送が必要な企業がある．これらの企業は, いつも自分で貨物を輸送するわけではなく，多くの場合，海運企業に運賃を払って輸送を依頼する．これらの依頼を受け，船舶で輸送を行う企業を，海運企業と呼ぶ．

海運企業は，複数の企業から，複数の輸送依頼を引き受ける．これらを，自らが運航する船舶で処理する．輸送依頼は，積み港，降ろし港，積み日，降ろし日，製品，量により指定される．表 1 に，輸送依頼データ 2 件分の例を挙げた．

表 1　輸送依頼のデータ例

積み港	降ろし港	積み日	降ろし日	製品	量
東京	大阪	10 日	12 日	A	600
大阪	名古屋	9 日	11 日	B	300

1.1 船舶スケジューリングと整数計画

海運企業が，輸送依頼を処理するための船舶の運航スケジュールを決めることを，船舶スケジューリングと呼ぶ．このとき，コストがなるべく小さいスケジュールを見つけるために，さまざまな数理的アプローチが試みられてきた [1].

船舶スケジューリングでは，整数計画によるモデルが有効である．船舶スケジューリングの数理的構造は，配送計画問題（Vehicle Routing Problem, VRP）と同じである．配送計画問題では, 陸上輸送での事例が多く報告されている．では，海上輸送での特徴は何だろうか．陸上輸送では，数百台, 数千地点の超大規模な問題例を解く必要がある．これに対して海上輸送では，数十台, 数十地点の問題例が解ければ十分である．また，陸上輸送では，運搬車や運用条件が等質であることが多いのに対して, 海上輸送では，企業や港ごとに異なる運用条件を扱う必要がある．

では, 船舶スケジューリングにはどのような整数計画モデルが有効だろうか．

1.2 集合分割問題による定式化

船舶スケジューリングには, 集合分割問題によるモデルが有効である．このモデルの利点は何だろう．

まず，数理計画ソルバーをサブルーチンとして用いるため，実装の手間が少なくすむ．また，ソルバーの性能が上がると, 解ける問題のサイズの拡大と，計算時間の短縮が期待できる．

加えて，実際の複雑な制約条件を扱うことができる．船舶や港湾の運用には，さまざまな制約が課される．例えば，港湾で貨物を積む/降ろす作業（荷役とよぶ）は，港湾の営業時間内にしかできないし，船舶の積載上限量以上に貨物を積むことはできない．他に, 港湾と船舶との関係で決まる制約もある．例えば, 水深の浅い港には大きな船は入れない．

さらに，問題のサイズ（貨物の数，船舶の数）が数倍になっても，計算時間の増加は数倍で済むことが期待できる．船舶スケジューリングでは，問題のサイズが計画のたびに数倍異なることは珍しくない．したがって, 問題のサイズが数倍大きくなると計算時間が急激に大きくなるような定式化は，具合が悪い．

こばやし　かずひろ
（独）海上技術全研究所 運航・物流系
〒 181–0004 東京都三鷹市新川 6–38–1

1.3 枝定式化およびメタヒューリスティックとの比較

配送計画問題の整数計画モデルとして，パス定式化と枝定式化が知られている．集合分割モデルはパス定式化に対応する．では，他方の枝定式化の特徴は何だろうか．

枝定式化は，(混合) 整数計画問題としての理論的研究が進んでいるという美点がある．時間枠制約や運搬車の容量制約など典型的な制約については，数理的構造が解明されている．一方，運搬車の数や貨物の数が増えると，制約式や変数の数が急激に増えるという欠点がある．また，複雑な制約を取り込むのが難しい．

配送計画問題の解法として，メタヒューリスティックが多用される．メタヒューリスティックの美点は，超大規模な問題が扱えることである．数百台，数千地点の問題例の近似解が，数分程度で得られる．ただし，比較的実装が大変で，制約条件を変えるためにはアルゴリズムの多くの箇所の変更が必要になる，という難点がある．船舶スケジューリングでは，超大規模な問題例を扱わないかわり，制約の追加，削除を頻繁に行う．その点，メタヒューリスティックよりも集合分割モデルのほうが扱いやすい．

集合分割モデルでは，変数の数は理論的には指数サイズになる．ただし，変数のなかで必要なものだけを見つけ出す工夫ができ，それにより実際に用いる変数の数は小さく抑えることができる．

2. 集合分割モデルの実現に必要なこと

集合分割問題は，ある集合をその部分集合に分割する方法を扱う．例えば，10 個の要素からなる集合があるとし，その要素に 1 から 10 まで番号をつける．この集合は，2 つの部分集合 $\{1,2,3,4,5,6\}, \{7,8,9,10\}$ にも，3 つの部分集合 $\{1,2,5\}, \{3,4,7,8\}, \{6,9,10\}$ にも分割できる．それ以外にも，多数の分割の仕方がある．いま，集合と，その部分集合の族が与えられたとする．さらに，部分集合のそれぞれにコストが与えられたとする．集合の各要素が，ちょうど 1 つの部分集合に含まれる部分集合の選び方で，コストの総和が最小になるものを見つけるのが集合分割問題である．

集合分割問題は，0-1 整数計画問題として定式化できる．集合の要素を $M = \{1, 2, \ldots, m\}$，与えられた部分集合族を $N = \{1, 2, \ldots, n\}$ と表す．また，N の要素 j のコストを c_j，集合 M の要素 i が部分集合 $j \in N$ に含まれるときに 1，それ以外のときに 0 をとる定数を a_{ij} とする．最後に，部分集合 j を選ぶときに 1，それ以外のとき 0 を取る変数を x_j とする．これらを用いると，集合分割問題は次のように定式化される．

$$
\begin{aligned}
&\text{最小化} && \sum_{j \in N} c_j x_j \\
&\text{制約条件} && \sum_{j \in N} a_{ij} x_j = 1 \ \ (i \in M), \\
& && x_j \in \{0,1\} \ \ (j \in N).
\end{aligned}
$$

ここで，M を輸送依頼の集合と，N の 1 要素を運航スケジュールの 1 候補と対応づけると，船舶スケジューリングは集合分割問題として定式化できる．

では，船舶スケジューリングの集合分割モデルを実現するには，どのような点に注意すればよいだろうか．

2.1 膨大な数の実行可能スケジュール

船舶の運航スケジュールの候補は，膨大な数にのぼる．1 つの運航スケジュール候補に対して，集合分割問題の制約行列の 1 つの列が定義されることに注意すると，運航スケジュール候補の数と，集合分割問題の制約行列の列の数が等しいことがわかる．まずいことに，スケジュールの候補数は数十万にもなることがある．この場合，制約行列の列の数が数十万にもなってしまう．列の数が数十万の集合分割問題は，2012 年現在のソルバーでも簡単には解けない．

2.2 効率の良いスケジュールだけを扱う

制約行列の列の数が膨大になるのは，すべての実行可能なスケジュール候補を挙げるからだ．ところで，実行可能なスケジュール候補のなかには，効率の悪いものも含まれる．これらは運航スケジュールとして採用されることはないだろうから，候補から外しても問題ない．そうすれば，制約行列の列の数を小さく保つことができる．このアイデアの実現には，列に含むべき効率の良いスケジュールを見つける仕組みが必要だ．この仕組みとして，列生成法を用いるとよい．列生成法の原理や適用範囲の詳細は，本特集号の宮本氏の記事や文献 [2] を参照されたい．

3. 数理モデルの具体的開発手順

前章までで，モデル開発の方針を述べた．本章では，それを具体的に実現するための手順を述べる．

3.1 集合分割問題を列生成法で近似的に解く

集合分割問題の表現を，船舶スケジューリングにあった形に書き直す．船舶の集合を V，その要素を v と表す．船舶 v が実行可能なスケジュール候補の集合を R_v と表す．輸送依頼の集合を P，その要素を p と表す．変数は x_{rv} と y_p であり，ともに 0-1 変数である．変数 x_{rv} は，船舶 v によるスケジュール r を採用するとき

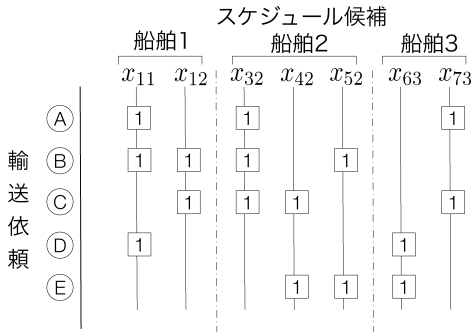

図 1 船舶スケジューリングの集合分割モデル．例えばスケジュール候補 x_{11} は，輸送依頼 A, B 及び D を処理する．

に 1，それ以外のときに 0 をとる（図 1）．変数 y_p は，輸送依頼 p がどのスケジュールでも処理できないときに 1，それ以外のときに 0 をとる．これらを用いて，船舶スケジューリングに対する集合分割問題（Ship-SP）は次のように表される．

$$
\begin{aligned}
\text{最小化} \quad & \sum_{v\in V}\sum_{r\in R_v} C_{rv}x_{rv} + \sum_{p\in P} F_p y_p \\
\text{制約条件} \quad & \sum_{v\in V}\sum_{r\in R_v} a_{p,rv}x_{rv} + y_p = 1 \ (\forall p \in P), \\
& \sum_{r\in R_v} x_{rv} \le 1 \ (\forall v \in V), \\
& x_{rv} \in \{0,1\} \ \ (\forall r \in R_v,\ \forall v \in V), \\
& y_p \in \{0,1\} \ \ (\forall p \in P).
\end{aligned}
$$

変数 y_p を導入すると，この問題は常に実行可能になる．ここで，係数 F_p は，輸送依頼 p が処理できないペナルティとみなせる．すべての輸送依頼を処理する解を得るには，この F_p を充分大きな値に設定するとよい．

問題 (Ship-SP) の制約行列の列の数は膨大になり，現実的な時間では解けない．そこで，以下のヒューリスティックによって近似的に解く．まず，問題（Ship-SP）の線形緩和を，列生成法を用いて解く．こうして得られた線形緩和の最適解 x_{rv}, y_p は，0 か 1 とは限らない．つまり，問題 (Ship-SP) の解になるとは限らない．そこで，線形緩和が最適に解かれた時点での制約行列をそのままにして，変数の 0-1 条件を付加して 0-1 整数計画問題として解く．その結果得られた最適解を，本来解きたい問題 (Ship-SP) の近似解とする．

このヒューリスティックは，線形緩和を最適に解くことで，0-1 整数計画問題で用いるための効率的なスケジュール候補を生成していると言える．ここでは，問題 (Ship-SP) の線形緩和を最適に解いているに過ぎず，もとの問題 (Ship-SP) の最適解は得られない．問題 (Ship-SP) の厳密な最適解を得るには分枝価格法の実行が必要だが，それには計算時間が長くかかる可能性がある．それを避けるために，このヒューリスティックが有効である．この方法により，経験的には十分に良い解が得られる．

3.2 列生成法の主問題と部分問題

列生成では，主問題と部分問題と呼ばれる数理計画問題を定義する．これらを交互に反復的に解くことで，主問題の制約行列の列を必要なだけ生成する．その反復手順は次のとおりである．

列生成法の反復手順

Step A 実行可能解が得られる程度の変数を用意する．

Step B Step A で用意した変数のみで主問題（線形計画問題）を解いて，最適解を得る．

Step C その最適解の双対情報を使って部分問題を定義し，解く．

Step C-1 最適値の値が負なら最適解から変数（=列）を生成し，主問題に加え，Step B へ移る

Step C-2 最適値の値が非負なら手続きを終了する．

では，問題 (Ship-SP) に対する主問題と部分問題は，どう定めればよいだろうか．まず主問題は，集合 R_v を，その部分集合 $\tilde{R}_v$ に置き換え，さらに変数を線形緩和したものである．これは線形計画問題になる．そして部分問題は、制約付き最短路問題である．この制約付き最短路問題において，船舶のスケジュールは s-t パスとして表される．

3.3 部分問題の定義

部分問題を定義するためには，ネットワークを定義する必要がある．

まず，1 つの輸送依頼に対して，1 つのノードを定義する．ノード i を訪問することは，対応する輸送依頼

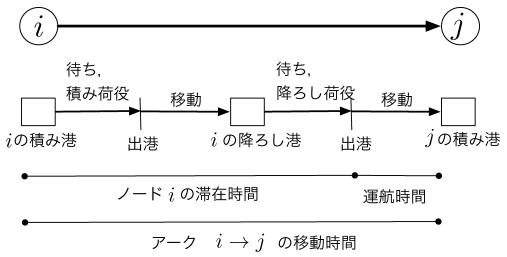

図 2 輸送依頼を処理する一連の作業とアークによる表現．

i を処理することを表す．“輸送依頼 i を処理する”とは，i の積み港で積み荷役を行ったあと，降ろし港に移動して降ろし荷役を行い，出港する，という一連の作業のまとまりのことを指す（図 2）．したがって，ノード i 上には，これらの作業を行うために一定時間滞在する必要がある．輸送依頼を表すこれらのノードに加えて，ソース s とシンク t を定義する．ソース s は，船舶の初期状態を，シンク t は船舶の最終状態を表している．

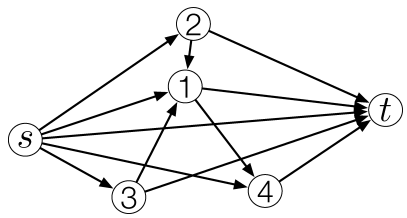

図 3　部分問題を定義するネットワークの例．

これらのノード間にアークを定義する．ノード i からノード j へのアークは，輸送依頼 i を処理した直後，輸送依頼 j の積み港に向かうことを表す．このアークには，移動時間とコストの 2 つの値をつける．移動時間は，i の積み港に到着してから，j の積み港に到着するまでの経過時間で定義する．いいかえると，ノード i での滞在時間と，i の降ろし港から j の積み港までの運航時間の和で定義する（図 2）．コストは，i の降ろし港から，j の積み港への運航距離で定義する．輸送依頼を表すノード間のアークに加えて，ソース s から，s 以外のすべてのノードへ，また，t から，t 以外のすべてのノードへアークを定義する．これでネットワークが定義できた．4 つの輸送依頼に対するネットワークの例を，図 3 に示した．例えば，パス $s \to 1 \to 4 \to t$ は，輸送依頼 1 と 4 を処理する運航スケジュールを表す．ここで，船舶ごとに別々のネットワークを定義する必要がある．それは，船舶の性能が互いに異なるためである．

船舶スケジューリングではさまざまな制約を扱う必要がある．ここでは制約の例として，時間枠制約と時刻依存の移動時間を取り上げる．

輸送依頼には，積み日と降ろし日が指定されていた（表 1）．これは，貨物の引き取りを積み日までに行い，降ろし港への運搬を降ろし日までに行わなければならない，という制約である．これは，ノードを訪問する時間枠制約として表現される．

実は，アーク $i \to j$ 上の移動時間は，i の積み港への到着時刻に依存する．アーク $i \to j$ 上の移動時間には，船舶が港で滞在する時間が含まれる（図 2）．いま，i の積み港の営業時間が，午前 9 時から午後 5 時だとしよう．この港に午後 11 時に船舶が到着すると，荷役を始めるには翌日の午前 9 時まで，10 時間待つ必要がある．一方，翌日の午前 9 時に到着すると，すぐに荷役を始めることができるため，待ち時間は必要ない．すなわち，ある日の午後 11 時に着いた場合は，翌日の午前 9 時に着いた場合よりも，待ち時間 10 時間分だけ滞在時間が長くなる．この待ち時間は，アークの移動時間にカウントされる．このように，港に到着した時刻に依存して，ネットワーク上のアークの移動時間が異なる．

これらの条件をモデル化すると，部分問題は，「時刻依存の移動時間をもつ，時間枠付き最短路問題」になる．これは困ったことだ．なぜなら，この問題は解きにくいからである．たしかに，ラベル修正法の一般的な枠組みはある [3]．しかしこの枠組みでは，指数的に増えるラベルを処理する必要がある．したがって，大変な計算時間がかかる可能性がある．また，アルゴリズムの実装と制約条件が緊密に結びついており，制約条件の追加と削除が簡単にはできない．

3.4　部分問題を解く

難しいとはいえ，この制約付き最短路問題を解かなければならない．そのために，この制約付き最短路問題を時空間ネットワーク上の最短路問題に変換する．この変換により，解くべき最短路問題は，制約条件のない最短路問題になる．そのため，1 ノードに 1 ラベルを保持する単純なラベル修正法で解くことができる．

では，時間枠制約と時刻依存の移動時間はどうなるのだろうか．それは，時空間ネットワーク上のアークの定義の仕方によって表現される．以下で，時間枠制約と時刻依存移動時間を持つネットワークを，時空間ネットワークに変換する例を見てみよう．変換前のネットワークが，図 4 のとおりだとする．ここでは，見やすさのため，アーク $1 \to 2$ 上の時刻依存移動時間のみ図示したが，実際にはすべてのアークが時刻依存の移動時間を持つことに注意してほしい．

時空間ネットワークを定義するには，まず時間を離散化する．ここでは，離散時刻を $\tau = 1, 2, 3, 4, 5, 6$ と設定する．時空間ネットワーク上のノード集合は，もとのネットワーク（図 4）のノード i と，離散時刻 τ のペア (i, τ) として定義される．したがって，時空間ネットワークのノードは 24 個定義する（図 5）．ここで，ノード (i, τ) は，ノード i に時刻 τ に到着するこ

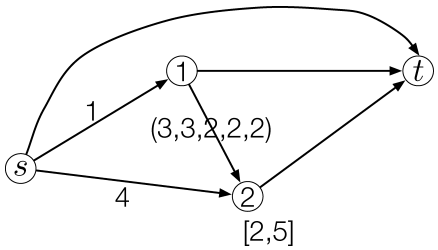

図 4 時間枠と時刻依存移動時間をもつネットワーク．[ ] はノードの時間枠，( ) はアーク 1 → 2 上の各時刻の移動時間を表す．

とを表す．また，最小の離散時刻を $\tau_{\min}$，最大の離散時刻を $\tau_{\max}$ と表す．なお，“時刻を離散化” するとは，一定間隔の時刻のみを取り扱うことを意味する．例えば，間隔を 1 時間とする離散時刻では，時刻として 1 時，2 時，3 時などのみを考える．それ以上細かい，例えば 1 時 30 分や 2 時 45 分などは考えない．

次に，これらのノード間にアークを定義する．時空間ネットワーク上のアークは，$(i,\tau_1) \to (j,\tau_2)$ と表現される．これは，輸送依頼 i の積み港に時刻 τ_1 に到着して輸送依頼 i を処理し，その直後に，輸送依頼 j の積み港に移動して時刻 τ_2 に到着することを表す．ここでは例として，輸送依頼 1 に関わるノード $(1,*)$ を出て，輸送依頼 2 に関わるノード $(2,*)$ に入るアークを定義する．もとのネットワーク上でノード 1 に時刻 1 に到着し，ノード 2 に向かう場合，かかる移動時間は 3 である（図 4）．したがって，時空間ネットワーク上のノード $(1,1)$ からノード $(2,4)$ にアークを定義する（図 5）．一方で，ノード 1 に時刻 3 に到着し，ノード 2 に向かう場合は，移動時間は 2 かかる．したがって，時空間ネットワーク上のノード $(1,3)$ からはノード $(2,5)$ にアークが定義される．すなわち，時刻に依存した移動時間の違いが，どのノードに向かってアークを定義するか，によって表される．

もとのネットワーク（図 4）のノード 2 は，時間枠 [2, 5] をもつ．これは，ノード 2 には時刻 1 以前，および時刻 6 以降には到着できないことを課す．この時間枠制約は，時空間ネットワークにおいて，$1 \geq \tau$ または $6 \leq \tau$ を満たすノード $(1,\tau)$ に入るアークを定義しないことで表現できる（図 5）．このように，時空間ネットワークでは，時間枠制約と時刻依存の移動時間を，アークの定義の仕方により表現できる．こうして定義した時空間ネットワーク上の $(s,\tau_{\min})$-$(t,\tau_{\max})$ パスは，すべて実行可能パスになる．したがって，最小コストの $(s,\tau_{\min})$-$(t,\tau_{\max})$ パスを見つけるアルゴリズムでは制約条件を考える必要がない．

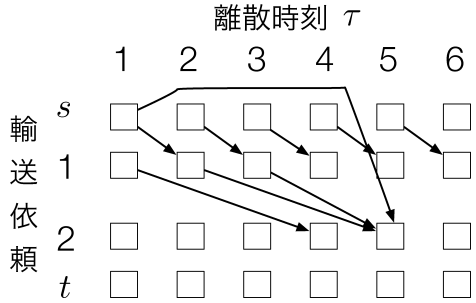

図 5 時空間ネットワーク．シンク t へのアークの表示は省略している．

3.5 列生成法の反復の具体的手順

これで，列生成法の主問題と部分問題が定義できた．これらを用いて列生成法を実行する．実行開始時点の主問題は，問題 (Ship-SP) において，スケジュール候補の集合 R_v を空集合とし，変数を線形緩和した線形計画問題である（Step A）．まず，この線形計画問題を解く (Step B)．その最適解が得られると，最初の制約式に対する双対変数 λ_p と，2 番目の制約式に対する双対変数 π_v の値も得られる．これらを用いて部分問題を定義する（Step C）．具体的には，部分問題のネットワークにおいて，輸送依頼に対応するノードを始点とするアーク $(i,\tau_1) \to (j,\tau_2)$ のコスト $c_{(i,\tau_1)\to(j,\tau_2)}$ は，双対変数 λ_i を使って更新する．このコストは，i の降ろし港から，j の積み港 j までの移動距離 d_{ij} から，i に関する双対変数 λ_i の値を引いた値として定義する．

$$c_{(i,\tau_1)\to(j,\tau_2)} := d_{ij} - \lambda_i. \tag{1}$$

双対変数 π_v の値は，ソース s に関するノード $(s,*)$ から出るアークのコストの定義に用いる．ノード (s,τ_1) から，輸送依頼 k に関するノード (k,τ_2) へのアーク $(s,\tau_1) \to (k,\tau_2)$ のコストは，初期位置から輸送依頼 k の積み港までの移動距離 d_{sk} から，π_v の値を引いた値として定義する．

$$c_{(s,\tau_1)\to(k,\tau_2)} := d_{sk} - \pi_v. \tag{2}$$

こうして定義したコストを用いて，時空間ネットワーク上の最小コストの $(s,\tau_{\min})$-$(t,\tau_{\max})$ パスを求める．求めた最適パスから制約行列に加える列を作り出し，追加する（Step C-2）．これは，実行可能スケジュールの集合 $\tilde{R}_v$ に追加する要素を作り出し，追加していることになる．

制約行列に列を追加することは，変数 x_{rv} を追加することと同じである．変数 x_{rv} の追加には，その目的関数での係数 c_{rv} と，制約行列での行の要素 $a_{p,rv}$ を

指定する必要がある．今，最適パスは船舶の運航スケジュールを表すので，その実行に必要な航行距離が計算できる．この航行距離を係数 c_{rv} の値とする．また，スケジュールで処理される輸送依頼の集合がわかる．それらに対応する行の要素を 1 とする．例えば，輸送依頼 1, 3, 7 が処理されるとしたら，$a_{1,rv}, a_{3,rv}, a_{7,rv}$ を 1 とする．それ以外の行は 0 とする．

部分問題は船舶ごとに定義して解くので, 最適パスは 1 つの船舶に対して 1 つずつ求まる．例えば，7 隻の船舶があれば 7 本のパスが求まり，そこから 7 本の列が得られる．これらすべてを制約行列の列として加える．

これで, 一回の反復が終わりである．次に，再び主問題を解く（Step B）．最適解が得られたときの双対変数 λ_p, π_v の値は，前回主問題を解いたときとは異なっている．そこで，(1) および (2) に従って部分問題のアークのコストを更新する（Step C）．その後, 再度最短路問題を解く．そうして得られた最適パスから制約行列の列を作り出し, 主問題に加える．この手順を繰り返す.

この反復はいつやめればよいだろうか．それは，最適パスのコストがすべて非負になったときである．このとき，もう主問題に加えるべき効率の良いスケジュールはない（Step C-1）．すなわち，それまでの反復で追加してきた列を用いれば十分であることを示している．こうして，問題 (Ship-SP) の線形緩和を解くことができた．あとは，3.1 節で述べたように, 制約行列をそのままにして変数の 0-1 条件を付加し, 0-1 整数計画問題として解くことで近似解を得る.

4. 実装方法

これで, 船舶スケジューリングの集合分割モデルを定式化し, 解くための方法が開発できた．この手順をパソコン上で実行できるソフトウェアとして実装する.

4.1 パソコン上で実装する

このソフトウェアに必要な機能を整理しよう．列生成の主問題である線形計画問題を繰り返し解く必要がある．また，部分問題のネットワークを表現するデータ構造を，保持・操作する必要がある．さらに，最短路問題を解く必要がある．加えて，運航に関わる船舶や港湾の機能, 運用条件を処理するためのプログラミング機能が必要である．最後には, 0-1 整数計画問題を解く必要がある．これらの機能を実現するためには，どのような技術を用いればよいだろうか.

4.2 汎用的なプログラミング言語を用いる

C 言語などの汎用的なプログラミング言語を用いる方法がある．その豊富なプログラミング機能を用いると，ネットワークを表現するデータ構造や，運用上の制約の処理を容易に実現できる．また，最短路問題を解くために既存のライブラリを用いることができ，必要ならば自ら性能のいいアルゴリズムを実装することも容易である．線形計画問題および整数計画問題を解くには, API を通じてソルバーをよびだせばよい.

汎用的なモデリング言語を用いる以外の方法として，モデリング言語を用いる方法がある．モデリング言語は，入力データを準備するためのプログラミング機能を持っている．これらの機能を使ってネットワークを表現し，最短路問題を解くアルゴリズムを実現すればよい．ただし，汎用的なプログラミング言語のような豊富な機能は期待できないし，外部のライブラリを用いることもできない．したがって，実装に多少苦労する場面があるかもしれない.

5. まとめ

本稿では，船舶スケジューリングの集合分割モデルの開発手順を述べた．このモデルで必要とするのは，基本的な最短路問題を解く技術と，線形計画ソルバーおよび整数計画ソルバーを用いる技術である．いずれも，大学学部の学生にも問題なく扱えるものであり，整数計画問題を用いた実務的モデル開発の，格好の入門素材となるだろう．本稿で述べた方法によって国内の船舶スケジューリングの実務問題を解いた結果については，文献 [4] を参照してほしい.

参考文献

[1] M. Christiansen, K. Fagerholt and D.Ronen, “Ship Routing and Scheduling: Status and Perspectives,” *Transportation Science*, **38** (2004), 1–18.

[2] J. Desrosiers and M. E. Lübbecke, “A Primer in Column Generation,” in *Column Generation*, G. Desaulniers, J. Desrosiers and M. M. Solomon, eds., Springer, 2005.

[3] S. Irnich and G. Desaulniers, “Shortest Path Problems with Resource Constraints,” in *Column Generation*, G. Desaulniers, J. Desrosiers and M. M. Solomon, eds., Springer, 2005.

[4] K. Kobayashi and M. Kubo, “Optimization of Oil Tanker Schedules by Decomposition, Column Generation, and Time-Space Network Techniques,” *Japan Journal of Industrial and Applied Mathematics*, **27** (2010), 161–173.

