実務的なシフトスケジュール作成問題への

メタヒューリスティクスアルゴリズムの応用

新田 利博　　嶋田 佳明　　田辺 隆人
株式会社 数理システム

〒160-0022 東京都新宿区新宿 2 丁目 4 番 3 号 フォーシーズンビル 10 階
nitta@msi.co.jp, shimada@msi.co.jp, tanabe@msi.co.jp

1 はじめに

我々はある公共サービス会社の各拠点におけるシフトスケジューリングを自動的に行うプログラムの開発を業務として行う機会を得た．このプロジェクトは 2006 年頭から始まり，サービスインは 2007 年 10 月，2008 年 4 月と段階的に行うという,まさに現在進行形のプロジェクトである. 2007 年 8 月現在では,このシフトスケジューリング問題についての実装は完了しており，最終的なテストを実際のシフトを手作業で行っている社員の方にしていただいているという段階である.

本稿では，このプロジェクトを通して実務で使用したメタヒューリスティクスアルゴリズムの紹介をし，その際に出てきた課題を提示していく.

2 シフトスケジューリング問題の紹介

本節では，本プロジェクトが対象としているスケジューリング問題の概要を整理する.

スケジューリング対象箇所は受注元会社の各拠点であり, 合計 1500 箇所にのぼる．1500 箇所の各々では人数およびシフトの数が違い,「数人，10 シフト」の規模から「200 人，80 シフト」の規模まで様々である．また，スケジューリング期間は 1 ヶ月である.

また，シフトスケジューリング問題を定式化するにあたり，シフトを決める際に考慮している項目(要件)について調査をした．下記にその調査例を列挙する((*)印がついているものは必ず満たさなければならない項目を意味しており，(*)印がついていないものはできるだけ満たさなければならない項目を意味している).

■シフト毎の必要社員数の確保(*)

■社員とシフトの間の相性(相性がよい/普通/必ず割り当てない(*))
■シフト並びの規則(必ず並べる(*)/優先的に並べる/並べることができる/必ず並べない(*)))
■連続勤務日数上限(*)
■3 連続以上の泊まり勤務の禁止(*)
■労働時間数上限(*)
■シフトの固定(必ず固定(*)/できれば固定)
■休日，勤務，休日の並びの禁止
■社員毎の休日数平準化
■社員毎の泊まり勤務数の平準化
■休前日の休日数平準化

必ず満たさなければならない要件以外に，できるだけ満たせればよいという要件も多数ある点が重要である．これは，必ず満たさなければならない要件とは法律などで既定されているものであるが， できるだけ満たせればよいという要件は人間が勤務表を作成する際に考慮しているもので，割り当て結果の質と関係するためである．

3 メタヒューリスティクスアルゴリズム WCSP

我々の最適化パッケージ NUOPT には Version7 からメタヒューリスティクスアルゴリズムのひとつである WCSP[1][2]を搭載している. これは, 京都大学「問題解決エンジン」開発グループによるタブサーチ法によるアルゴリズムである. NUOPT ではこの WCSP をモデリング言語 SIMPLE[3]から呼べるようなインタフェースを提供することにより，容易に WCSP を使用できる環境となっている．例えば，Bin Packing 問題を例にする．Bin Packing 問題は次のような定式化で表すことができる.

$$
\begin{aligned}
\text{[BP]}\quad \min \quad & \sum_k z_k & & \\
\text{s.t.} \quad & \sum_k y_{kl} = 1 & & \forall l = 1,..,L \\
& \sum_l D_l y_{kl} \le V z_k & & \forall k = 1,..,K \\
& y_{kl}, z_k = 0,1 & & \forall k = 1,..,K\ ;\ \forall l = 1,..,L
\end{aligned}
$$

この問題を WCSP として SIMPLE で記述すると次のようになる.

図 1 : BinPacking 問題の SIMPLE 記述例

```
Set K(name="K");
Element k(set=K);
Set L(name="L");
Element l(set=L);

Parameter D(name="D", index=l);
Parameter V(name="V");
IntegerVariable y(name="y", index=(k,l), type=binary);
IntegerVariable z(name="z", index=k, type=binary);

selection(y[k,l], k);
sum(D[l] * y[k,l], l) <= V * z[k];

Objective obj(type=minimize, target=0);
obj = sum(z[k], k);

options.method = "wcsp";

solve();
```

このように定数を表す D_l，V，変数を表す y_{kl}，z_k を宣言している箇所を除くと一般の定式化と SIMPLE の記述では容易に対応がつくのがわかる．このように対象となる問題を定式化することができれば, SIMPLE を通して WCSP を使用することができる.

今回のシフトスケジューリング問題では, この WCSP を使用することとした. これは，次の理由によるものである.

a. SIMPLE で記述できるため保守性がよい

b. WCSP が得意とするパターンの問題である

c. WCSP にはハード制約・ソフト制約という概念がある

項目 a は，Bin Packing の例からも明らかだが，問題の記述はアルゴリズムの実装とは完全に分離しているため保守の対象は SIMPLE 中の記述に限られる.

項目 b は, WCSP は割り当て系の問題に対して短時間で非常によい解を求めてくれるという経験上の知見がある．特に割り当てられる対象が多く，割り当てる種類が少ない場合に顕著である．シフトスケジューリング問題は，人×日付のマス目にシフトを割り当てる問題であるため，WCSP が得意とする問題であると考えられる．最後の項目 c は，制約式に 2 種類の意味をつけることができる． 1 つがハード制約といい，他方がソフト制約という．簡単に説明すると，ハード制約とは必ず守らなければならない制約，ソフト制約は可能な限り守らなければならない制約のことである．これにより，シフトスケジューリングを考える上で労働基準法で決められたような規則(例えば，月の労働時間に対する制約など)はハード制約で表し，人の要望(可能ならこの日はこのシフトに入りたいなど)はソフト制約で表すことで簡潔に記述することが可能となる．このように制約式に 2 種類の意味があることで柔軟な表現が可能である．

以上より，我々は対象となるシフトスケジューリング問題専用のアルゴリズムを新たに開発するよりも，汎用的であるが強力な武器を備えた WCSP を使用することとした．

4 シフトスケジューリング問題のモデル化

対象となるシフトスケジューリング問題のモデル化は 2 節で述べたような要件を制約式という形で表すことである．本シフトスケジューリング問題の基本的な部分の定式化は次のようになっている．

$$\text{s.t.} \quad \sum_{s} x_{m,d,s} = 1 \qquad \forall m = 1,..,M\ ,\forall d = 1,..,D$$

$$\sum_{m} x_{m,d,s} = request_{d,s} \qquad \forall d = 1,..,D\ ,\forall s = 1,..,S$$

$$x_{m,d,s} = 0,1 \qquad \forall k = 1,..,K\ ;\ \forall l = 1,..,L$$

$x_{m,d,s}$ ： 決定変数．社員 m にシフト s を割り当てる際に 1

Man={1,..,M}：社員集合
Day={1,..,D}：日付集合
Shift={1,..,S}：シフト集合

$request_{d,s}$ ： シフト毎の必要人員数

これに追加して，第 2 節で紹介した要件に対する制約式を順次記述していくことで，シフトスケジューリング問題に対する定式化が完了する．

5　問題点

全てのハード制約を満たす解がない場合，または，WCSP の探索によりハード制約を満たしている解を見つけることができなかった場合は，どのハード制約を破っている解が得られるかはあらかじめ知ることはできない．これは実務的には非常に困る状況である．例えば，連続勤務が 5 連続までしか許されない状況で 6 連続勤務が得られた場合を考える．これはたまたま 5 連続勤務のハード制約が破られただけで，他のハード制約を破ることで 5 連続勤務の制約が守られているのとペナルティ的には同等の解は存在している．このような場合，実務家はこのハード制約が破られた割り当てを調整して，扱いやすい割り当てにする必要がある．しかし，どのハード制約が違反となるかは不明なため，ハード制約の破り方により様々な考察が必要となるため，これではとても使いやすいシステムとは言えない．そこで，例えば，ハード制約違反が生じた場合にある特定の制約にしわ寄せが行くような仕組みがあれば，話は別である．実行可能解がない状況では指定されたハード制約を優先して破ることにより，実務家はハード制約を破っている解を得たとしても，毎回質的に同様の割り当てを得ることができる．

以上の考察より，このしわ寄せが行く制約式を重みを大きくしたソフト制約として実現すればよいと考えるかもしれない．しかし，WCSP の実装上ソフト制約の違反量の総計には上限があるため，ソフト制約の重みの上限はそれよりも小さい．そのため，この方法の場合，ソフト制約の重みの調整が問題によって必要となる．

そこで，我々はこの実務的な観点での問題を解決するために，制約式にハード制約とソフト制約の中間を意味するものを追加し，これをセミハード制約と命名した．セミハード制約は，ハード制約を破る際にまず最初にセミハード制約を破るというどのハード制約よりも優先度が低いハード制約という見方と，どのソフト制約よりも重みが重いソフト制約という見方ができるものである．

今回のプロジェクトでは，指定されたシフト毎の必要人員数を満たさなければならないという制約式をセミハード制約とした．これにより，実行可能解がない場合においては，必要人員数を違反するような解が得られることとなる．これは実務家の直感にあったものであり，さらに，実行可能解がない場合は必要人員数を少なくすることで必ず実行可能解があるという事実とも整合している．

６　終わりに

今回シフトスケジューリング問題を WCSP(+NUOPT)で実現した．その中でセミハード制約のような新しい概念の追加を行った．同様に WCSP に追加するとより柔軟な定式化が可能となる概念として「～の平準化」がある．平準化とは，例えば，社員毎の休日の平準化がある．これは各社員の休日数の分散を小さくするという意味であるが，当初は次のような(ソフト)制約式で実現をしていた．

$$hNum_i = average$$

($hNum_i$ は社員 i の 1 ヶ月の休日数，$average$ は全体の 1 ヶ月の休日数平均)

この制約式では，確かに各社員の休日数が休日数平均と同じになるように見えるが，実はならない．社員 1,2 が休日数平均よりも 1 ずつ小さな値をとっている場合を想定する．この場合の制約式違反のペナルティの総計は 2 である．しかし，社員 1 の休日を休日数平均と同じ値とし，社員 2 の休日数が休日数平均よりも 2 小さな値をとっているとすると，この場合も同じく総ペナルティは 2 である．つまりこの状況では，平準化は達成できていないこととなる．これを正しく実現するためには，次のような定式化を考えることができる．

$$hNum_i = average$$
$$averate - 1 \le hNum_i \le average + 1$$

こうすることにより，休日数平均から社員の休日数が離れれば離れるほど総ペナルティ値は増えるため，「ペナルティを小さくする=平準化」となる．平準化はこのように定式化側での対処も可能であるが，これを WCSP のソルバーの中で行うことでより性能は上がると考えられる．

また，今回のプロジェクトでは WCSP はよい結果を示してくれている．シフトスケジューリング問題以外にも，人員配置問題，ルート割り当て問題，設備割り当て問題に対しても適用実績があり，いずれも実運用に耐えうる結果を残している．このように WCSP のソルバーとしての性能は非常に強力であることは明白であるが，実務で必要なのは強力なソルバーだけであろうか．我々はそうではないと考える．強力なソルバーがあることは確かに必要であるが，それ以前にどのように適用するのかという点が非常に重要である．例えば，次のような点である．

■具体的な定式化はどのようにするのか.

■実務で求められていることはなにか.

前者は，対象の定式化を考える際に，例えば WCSP が得意なパターンに持ち込めるのかということである．WCSP が得意ではないパターンしか出てこないのであれば，WCSP を使用する意味がない．後者は，運用方法とも絡む．例え

ば, ユーザがトライ&エラーを繰り返しながら答えを見つけていくという運用が必要であれば，システム自体をそのように作成する必要があり，さらに定式化もそれにあわせる必要がある．このような点を無視してしまうと，ユーザにとっては使いやすいシステムとは言えず， 結局，ソルバーの性能ほどの結果を残すことは不可能となってしまうだろう．

参考文献

[1] K.Nonobe, T.Ibaraki: A tabu search approach for the constraint satisfaction problem as a general problem solver, *European Journal of Operational Research*, Vol.106, pp.599-623(1998).
[2] K.Nonobe, T.Ibaraki: An improved tabu search method for the weighted constraint satisfaction problem, *INFOR*, Vol.39, pp.131-151(2001).
[3] (株)数理システム:NUOPT マニュアル(2007).