工場のピーク電力抑制を目的とした生産設備群の運転計画最適化

小熊　祐司

製造業において工場の電力料金の低減・抑制は大きな課題である．一般に電力料金は，最大需要電力（ピーク電力）に比例する基本料金と，電力量に比例する電力量料金から構成されており，前者に関しては同生産設備で同量の生産を行う場合でも，設備群の運転計画を最適化しピーク電力を抑えることで低減できる可能性がある．このような運転計画を得る問題は整数計画問題として定式化できるが，厳密解法である分枝限定法が苦手とする「最大値最小化」の構造をとる．そこで本稿では，メタヒューリスティクスの利用や探索空間の制限，目的関数の工夫など，同問題の良質な近似解を求めるための方法をいくつか紹介する．また掲題の技術開発を題材として，数理最適化ソルバやデモソフトウェアの利活用による開発プロセスの効率化についても述べる．

キーワード：最大値最小化，メタヒューリスティクス，エネルギーマネジメント

1. はじめに

製造業において，自社工場のエネルギーコストの低減・抑制や CO_2 排出量削減は大きな課題であり，IHIではそれに貢献するための技術として，工場内のエネルギー設備の最適な運用や制御を担うエネルギーマネジメントシステム (EMS) の開発・展開を推進している．

コスト改善を目的とした工場のエネルギーマネジメントで重要となるのは，生産量の維持・向上とコスト低減・抑制の両立である．エネルギーコストのうち代表的なものである電力料金は，おもに最大需要電力（ピーク電力）に比例する基本料金と，使用した電力量に比例する電力量料金から構成され [1]，前者に関しては，同生産設備で同量の生産を行う場合でも，ピーク電力の抑制により低減できる可能性がある．その方法としては太陽光発電システムや蓄電池の導入・運用があるほか，設備投資を必要としないものとして，消費電力の大きい生産設備の運転計画を調整することで電力需要のピークを抑える，ピークシフトとよばれる操業方式もあげられる．しかしながら，ピークシフトの実践には生産とエネルギーの双方を考慮した運転計画が必要であり，これを人力で作成するのはかならずしも容易ではない．そこで著者らは，生産とエネルギーの協奏したエネルギーマネジメント技術の一つとして，生産量を維持しつつピーク電力を抑制する設備運転計画を自動作成する技術の開発に取り組んできた [2]．

こぐま　ゆうじ
（株）IHI 技術開発本部技術基盤センター数理工学グループ
〒 235–8501　横浜市磯子区新杉田新中原 1
koguma2654@ihi-g.com

本稿では，数理最適化の視点から同技術開発の取り組みを紹介する．上述のような運転計画を得る問題は「最大値最小化」の構造をとる整数計画問題として定式化できるが，この種の問題は厳密解法である分枝限定法と相性が悪く，同法を用いて実際に想定される規模の問題を解くことは困難である．そこで本稿では，メタヒューリスティクス [3, 4] の利用や，探索空間の制限，目的関数の工夫など，同問題の良質な近似解を求める方法をいくつか紹介する．また掲題の技術開発を題材として，「解くべき問題を早期に明確にする」という観点から，数理最適化ソルバやデモソフトウェアの利活用による開発プロセスの効率化についても述べる．

2. 問題の概要

電力を加熱源とするバッチ式の真空熱処理炉を例として，本稿で議論する運転計画問題について説明する．真空熱処理炉は，鋼部品を高温（500 ℃～1,000 ℃以上）で加熱処理する設備であり，図 1 に示すように，稼働開始後，炉内温度を所定の温度まで上昇させる際に消費電力ピークを形成する．バッチ式炉では，いったん処理（ジョブ）を開始すると，完了するまで部品の

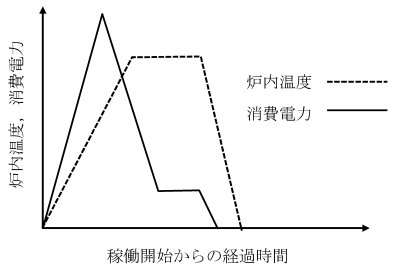


図 1　真空熱処理炉における稼働中消費電力推移の概形

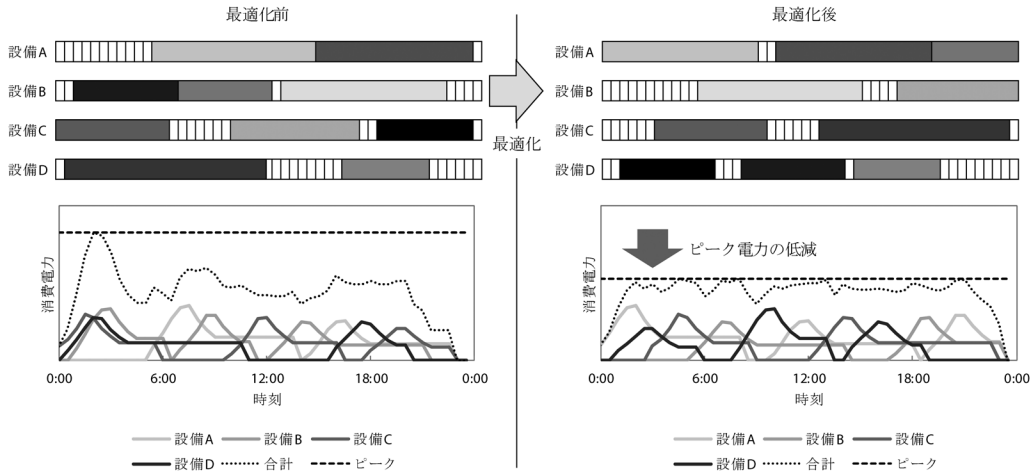


図 2　ピーク電力を最小化する生産設備群の運転計画最適化の概念図

取入れや取出しはできない．複数のバッチ式真空熱処理炉を有する工場においてピーク電力を抑制するためには，各炉のピークの重なりを回避した運転計画を組む必要がある．本稿で扱う問題は，真空熱処理炉のようにピークの鋭い消費電力推移を特徴とする生産設備の複数台の運用において，一定の計画期間を考え，各種制約条件を遵守したうえで所定のジョブをすべて完了させつつ，全生産設備の合計消費電力ピークが最小となる運転計画を求めるものである．

図 2 に，本最適化問題の概念図を示す．図 2 では 4 台の真空熱処理炉設備の 1 日の運転計画を例に，最適化前後のガントチャートを，消費電力推移とあわせて示している．ガントチャート中の 1 コマは一定の時間幅を示しており，塗りつぶされたコマは当該時間帯にジョブが実行中であることを意味する．各ジョブの所要時間や消費電力推移は処理内容に依存する．最適化前の消費電力推移をみると，計画対象日の未明に設備 B, C, D が稼働開始する運転計画となっており，これらのピークが重なることで全設備合計で大きなピークを形成している．一方で最適化後の消費電力推移は，各設備の稼働タイミング，ピークが重なることなく，全体としてピーク電力が抑制されたものとなっている．

3.　定式化

2 節で述べた運転計画問題を整数計画問題として定式化する．定式化に用いる各種記号を表 1 のとおり定義する．

表 1 および以降において，$\boldsymbol{x}$ は $x_{jmk}(j \in \mathcal{J}, m \in \mathcal{M}_j, k \in \mathcal{K}_j)$ をまとめた表記とする．また関数 $\lambda_{mk}(\boldsymbol{x})$, $p_k(\boldsymbol{x})$ は，それぞれ下式で計算される：

表 1　記号の定義

(a) 集合

記号	説明
$\mathcal{J}$	ジョブの集合 ($\mathcal{J} = \{1, \ldots, \lvert\mathcal{J}\rvert\}$)
$\mathcal{M}$	設備の集合 ($\mathcal{M} = \{1, \ldots, \lvert\mathcal{M}\rvert\}$)
$\mathcal{M}_j$	ジョブ j を実行可能な設備の集合 ($\mathcal{M}_j \subseteq \mathcal{M}$)
$\mathcal{K}$	時刻（ステップ）の集合 ($\mathcal{K} = \{1, \ldots, \lvert\mathcal{K}\rvert\}$)
$\mathcal{K}_j$	ジョブ j を実行開始可能な時刻の集合 ($\mathcal{K}_j \subseteq \mathcal{K}$)

(b) パラメータ

記号	説明
$L_j \in \mathbb{N}$	ジョブ j の所要時間（ステップ数）
$\delta_{j\kappa} \in \{0, 1\}$	ジョブ j の κ ステップ目の実行状況（1：実行中，0：実行中でない）
$q_{j\kappa} \in \mathbb{Z}$	ジョブ j の κ ステップ目の消費電力

(c) 決定変数

記号	説明
$x_{jmk} \in \{0, 1\}$	ジョブ j を設備 m で時刻 k に実行開始するとき 1，そうでないとき 0
$y \geq 0\,(y \in \mathbb{Z})$	ピーク電力

(d) 関数（決定変数に依存する量）

記号	説明
$\lambda_{mk}(\boldsymbol{x}) \in \mathbb{Z}$:	設備 m の時刻 k における稼働状況（≥ 1：実行中，0：実行中でない）
$p_k(\boldsymbol{x}) \in \mathbb{R}$:	時刻 k における全設備合計消費電力

$$\lambda_{mk}(\boldsymbol{x}) = \sum_{j \in \mathcal{J}} \sum_{\kappa=1}^{L_j} \delta_{j\kappa} x_{jm(k-\kappa+1)} \quad (m \in \mathcal{M},\, k \in \mathcal{K}),$$

$$p_k(\boldsymbol{x}) = \sum_{j \in \mathcal{J}} \sum_{m \in \mathcal{M}} \sum_{\kappa=1}^{L_j} q_{j\kappa} x_{jm(k-\kappa+1)} \quad (k \in \mathcal{K}).$$

以上の定義のもとで，運転計画問題 (P) を，

$$
\begin{aligned}
(\mathrm{P}):\ &\underset{\boldsymbol{x},y}{\text{minimize}}\ \ y && (1)\\
&\text{subject to}\ \sum_{m\in\mathcal{M}_j}\sum_{k\in\mathcal{K}_j} x_{jmk}=1\ \ (j\in\mathcal{J}), && (2)\\
&\qquad \lambda_{mk}(\boldsymbol{x})\le 1\ \ (m\in\mathcal{M},k\in\mathcal{K}), && (3)\\
&\qquad p_k(\boldsymbol{x})\le y\ \ (k\in\mathcal{K}), && (4)\\
&\qquad x_{jmk}\in\{0,1\}\\
&\qquad (j\in\mathcal{J},\ m\in\mathcal{M}_j,\ k\in\mathcal{K}_j), && (5)\\
&\qquad y\ge 0 && (6)
\end{aligned}
$$

として定式化する．制約条件 (2)–(4) の意味は以下のとおりである．

式 (2)： 各ジョブは 1 設備で 1 回だけ実行される．

式 (3)： 各設備は複数ジョブを同時実行できない．

式 (4)： いずれの時刻においても，全設備合計の消費電力は y 以下である．

実際には上述の制約条件に加え，ジョブの順序関係や優先度，設備のメンテナンス計画，運転担当者の勤務シフトに関する制約など多数の制約条件を考慮する必要がある．これらの具体的な定式化については本稿では割愛するが，上で挙げた例も含め，多くの制約条件が決定変数 $\boldsymbol{x}$ に関する線形式で記述できる．問題規模としてはジョブ数 200，設備数 8，計画期間 1 週間程度を想定している．1 ステップの時間をピーク電力算定基準である 30 分とすると，1 週間に相当するステップ数は $|\mathcal{K}|=336$ であり，仮にすべてのジョブが全設備，全時刻で実行開始可能である場合，0-1 変数の数は $|\mathcal{J}|\times|\mathcal{M}|\times|\mathcal{K}|=537,600$ となる．

問題 (P) は，厳密解法である分枝限定法との相性が悪い「最大値最小化」型の構造 [5] をもつ整数計画問題であり，同法を用いて上述規模の問題に対して現実的な時間で最適解を得ることは困難である．問題 (P) の実行可能解において，ピーク電力に直接寄与しているのはピーク電力発生時刻に実行中のジョブのみであることに注意する．ほかのジョブについては一定の計画自由度が残るため，その結果として，優劣のない複数の実行可能解が存在しうることとなる．最適解の見込みがない領域を探索範囲から除外することで列挙を効率化する分枝限定法にとって，優劣のない複数の解の存在は探索効率低下の要因となりうる．図 3 に，ランダムに生成した小規模な問題インスタンス（ジョブ数 5〜10，設備数 4，ステップ数 48）に対して分枝限定法を適用し，最適解を得るまでに要した時間を示す[1]．

[1] 求解には Windows10 計算機 (CPU: Intel(R) Xeon(R) W-2125, RAM: 64GB) と NTT データ数理システム Numerical Optimizer 22.1.0 を使用した．

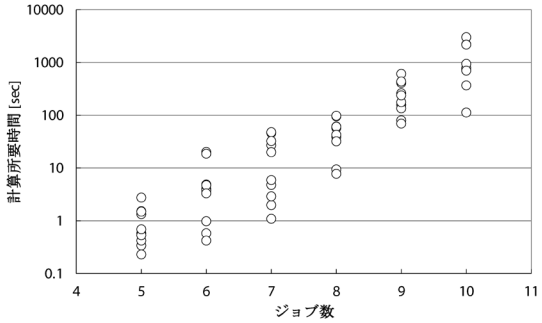


図 3 小規模な問題 (P) のインスタンスに対して分枝限定法を適用した場合の求解時間（ジョブ数ごとにランダムに生成した 10 題に対する結果を「○」で示している）

この規模においても 1,000 秒以上の計算時間を要する場合もあり，想定規模の問題に対する分枝限定法適用は現実的でないことがわかる．

4. 近似解計算のためのアプローチ

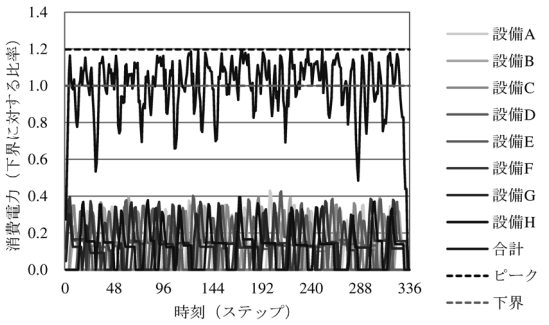
実務において問題 (P) を解くにあたり，かならずしも真の最適解は必要ではなく，良質な実行可能解を短時間で求めることが重要となる．そこで，メタヒューリスティクスに基づく近似解法を考えることとする．NTT データ数理システム Numerical Optimizer [6] は，タブーサーチに基づくアルゴリズム（以下では Numerical Optimizer のオプション名より「WCSP」とよぶこととする） [7, 8] を具備しており，想定規模の問題 (P) に対しても，定式化を変更することなく数分程度で良質な実行可能解を得ることができる．

さて，アルゴリズムとして WCSP を用いる場合でも，問題構造を利用して探索効率化を図ることができる．戦略的な「多様化と集中化」を通してその真価を発揮するメタヒューリスティクスにおいて，ありうる計画すべてを探索対象とするのは，集中化の観点から得策ではない．探索空間の制限により探索効率化を狙う方法として，実行可能解のうち一部の変数を固定しつつ反復的に最適化を行う方法が知られており [9]，問題 (P) においては以下のような手法が考えられる．

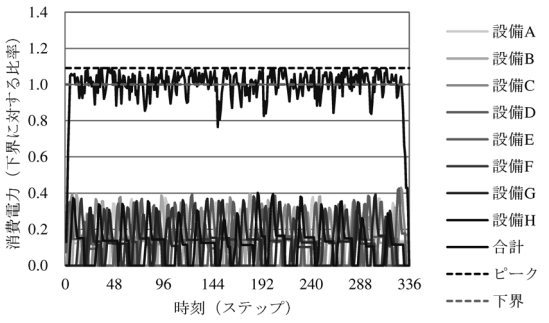
手法 1 各ジョブの実行設備を既得の実行可能解のものに固定し，かつ実行開始時間を実行可能解のものから $\pm N$ $(N\in\mathbb{N})$ に制限する．

手法 2 既得の実行可能解のピーク電力発生時刻から $\pm M$ $(M\in\mathbb{N})$ ステップに含まれるジョブ以外を固定する．

このうち手法 2 については，ピーク電力に直接寄与するのは一部のジョブのみに限られるという問題構造に着目したものである．この構造により，ピーク電力発



(a) WCSP 単純適用



(b) 探索空間制限に基づく段階的解法

図 4 メタヒューリスティクスにより得られた運転計画の消費電力推移（ランダムに生成したジョブ数 200，設備数 8，ステップ数 336 の問題インスタンスに対する結果）

生時刻付近のジョブのみを最適化対象としても一定の改善効果を見込むことができる．ただし，手法 2 は計画期間全体のピーク電力を抑制するものではなく，探索空間を制限した問題を解いた結果，別の時間帯にピークが残る可能性がある．したがって，手法 2 を採用する場合は反復的な計算が必要となる．

著者の検討では，上述の考え方に基づき，

Step 1. 問題 (P) の実行可能解を求める．

Step 2. 手法 1 により探索空間を制限して計画期間全体のピーク電力抑制を図る．

Step 3. 手法 2 により探索空間を制限して局所的なピークを順次抑制する．

という手順による段階的解法を構築し，想定規模の問題 (P) に対して，WCSP の単純適用で得られた解よりも良い解を得ることに成功している [10]． 図 4 に，WCSP の単純適用および探索空間制限に基づく段階的解法で得られたスケジュールの消費電力推移を，(P) の下界値（問題 (P) の連続緩和問題の最適値）に対する比率として示す．各手法で得られた計画のピーク電力と下界値とのギャップは，WCSP 単純適用によるものが約 20%であるのに対し，段階的解法によるものは約 8%であり，後者の解が優れていることがわかる．

WCSP を用いた検討を経て，最終的に著者が採用したのは，図 5 に示す近傍定義に基づく，WCSP と同様のタブーサーチの C++ による実装である[2]．また目的関数の設計に関しても独自の工夫を講じている．問題 (P) をタブーサーチで解く場合，一部のジョブのみがピーク電力に寄与するという問題構造が影響し，計画全体を見渡した解の改善が生じづらい．そこで，計画期間ピーク電力を目的関数とするのではなく，適当な目標値 p_{target} に対する逸脱量

$$v_k(\boldsymbol{x}) = \max(p_k(\boldsymbol{x}) - p_{\mathrm{target}}, 0) \ \ (k \in \mathcal{K})$$

を考え，$v_k(\boldsymbol{x})$ の全時間帯における和と最大値の加重和

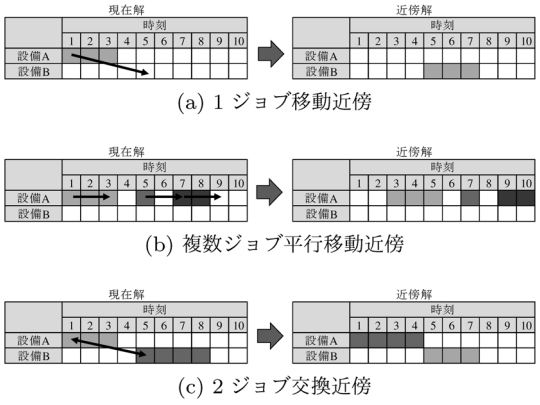
$$f'(\boldsymbol{x}) = \alpha \max_k v_k(\boldsymbol{x}) + \frac{\beta}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} v_k(\boldsymbol{x})$$

を目的関数として採用した [11]．本目的関数の採用により，目的関数第 2 項の意味で計画期間全体としての電力の平準化を図りつつ，その結果として第 1 項の最小化を誘導することができる．ここで，$\alpha \geq 0$, $\beta \geq 0$ は各項の重み係数であり，$\beta = 0$ の場合，$f'(\boldsymbol{x})$ はピーク電力のみを考慮したものとなる．図 6 に，誘導項（目的関数第 2 項）の有無に応じた最適化性能の比較として，ピーク電力のみを考慮した場合，誘導項を付与した場合のピーク電力上界値の推移を示している．計算開始直後のデータがない期間は実行可能解が未取得であることを意味している．各ケースにおけるピーク電力上界値は，問題 (P) の連続緩和問題[3]を解いて得た下界値とのギャップとして示している．なお，ケース (b) における p_{target} もこの下界値を設定している．図 6 より，誘導項を加えることで，ピーク電力最小化の意味でも継続的に解の更新が行われていることが確認できる．

ところで，タブーサーチは毎回の反復で多数の近傍解評価を行うため，目的関数や制約条件を高速計算する差分評価アルゴリズム設計が必要となり，実装には相応の工数を要する．最適化の効果の見込みが得られていない検討初期段階での実装着手はかならずしも有効ではない．著者のタブーサーチ実装の前提として，

[2] 本実装ではアルゴリズムの単純さを重視し，本稿で述べた探索空間制限に基づく段階的探索は行っていない．

[3] 正確には，ここでは問題 (P) から制約条件 (3) を除去したものの緩和問題を採用した．いくつかの数値実験から，本制約条件は緩和問題最適値への影響は僅少ながら，計算時間に大きく影響することが観察されていた．



(a) 1 ジョブ移動近傍

(b) 複数ジョブ平行移動近傍

(c) 2 ジョブ交換近傍

図 5　独自実装タブーサーチにおける近傍の定義

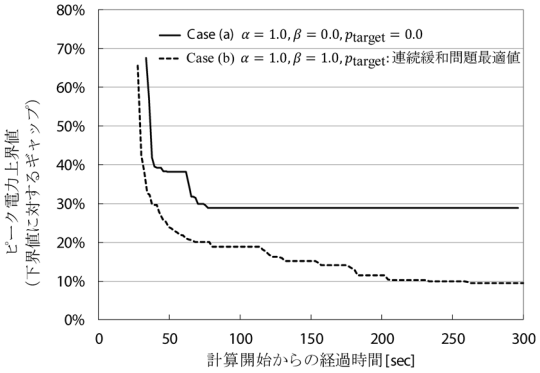


図 6　誘導項の有無による最適化性能の比較（ジョブ数 200，設備数 8，ステップ数 336 のランダムに生成した問題インスタンスに対する結果）

Numerical Optimizer を用いたアルゴリズムのプロトタイピング・試計算があったことを強調しておく．

5.　開発プロセス

前節までは問題 (P) の解法に主眼をおいていたが，本節では視点を変え，本技術開発を例に数理最適化の応用開発プロセスについて私見を述べる．図 7 は著者の考える数理最適化の応用開発プロセスを示したものである．数理最適化の応用先は多岐にわたるが，ここではスケジューリングや各種計画など，従来人が担ってきた業務の最適化・効率化を企図したものを想定する．プロセスは大雑把に (1) 問題分析，(2) 定式化，(3) アルゴリズム選定・開発，(4) ユーザからのフィードバック，(5) システム設計・運用，からなり，研究開発段階ではおもに (1)〜(4) が検討範囲となる．最初から決定変数，制約条件，目的関数などが与えられていることは稀であり，まずは一連のサイクルを高速にまわし，解くべき問題を明確にすることが重要となる．以

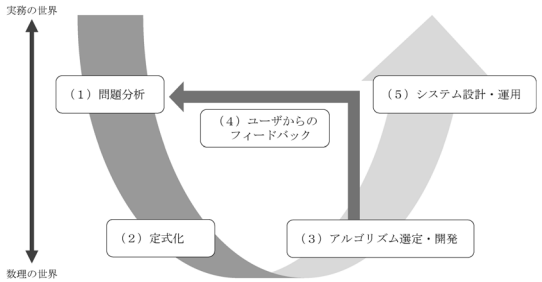


図 7　数理最適化の応用開発プロセス

下では各プロセスについて個別にみていく．

5.1　問題分析

問題分析では，ユーザからのヒアリングを行いつつ，想定効果，計算に必要なデータの入手性，解の実現性などの観点から，制約条件を整理しつつ，最適化の範囲や対象を特定する．この段階で適当な緩和問題を定式化し解くことで，求まった下界値（最小化問題の場合）から最適化のポテンシャルを見積もることもできる．得られた下界値が最適化適用前と大きく変わらない場合，最適化範囲を見直すか，あるいは最適化の適用自体再考することとなる．本稿で述べた事例においては，計画期間の全消費電力量を計画時間で割ることでピーク電力の下界値を得ることができ，実設備の運転データから十分な効果を見込んでいた．

5.2　定式化およびアルゴリズム選定・開発

定式化では問題分析で得た知見をもとに，決定変数，制約条件および目的関数を定義する．定式化とアルゴリズムは不可分であり，両者あわせた検討が必要となるが，早い段階で問題に特化したアルゴリズムを作りこむことは避ける．まずは数理最適化ソルバを利用してある程度の質の実行可能解を工数をかけずに求め，ユーザからのフィードバックを受けることを優先する．問題構造を効果的に利用したアルゴリズム開発は最適化エンジニアにとって心躍る工程であるが，その問題構造自体がユーザからのフィードバックの結果，容易に破綻しうるためである．たとえばスケジューリング系の案件における「平準化」は，最適化に望まれる機能として頻出の部類であるものの，初期段階のニーズとしては出てこない場合も多く，しかもアルゴリズムによって好ましい問題構造を大きく破壊しがちである．まずは実行可能解の計算とユーザからのフィードバックのサイクルを素早くまわし，平準化も含め，隠れたニーズの抽出を優先することを勧めたい．本工程において，問題によっては分枝限定法を用いた厳密解計算がむずかしい場合もあるが，整数計画問題として定式化

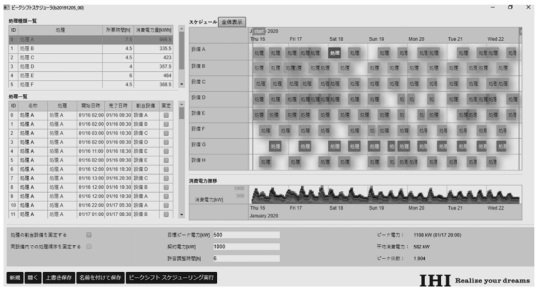


図 8　デモソフトウェアの画面構成例

された問題であれば，Numerical Optimizer の WCSP を利用することで良質な実行可能解を求められる場合がある．またその際，3 節でも述べたような問題構造を利用した工夫を講じることで，さらなる解の改善を図ることも可能である．

5.3　ユーザからのフィードバック

ユーザからのフィードバックを受ける場合はデモソフトウェアを試作・活用すると効果的である．最適化のユースケースや入力・出力の情報が明確になるほか，最適化で得られた解を手修正できるインターフェースとしておくことで，ユーザの企図する解の性質をユーザ・最適化エンジニア間で共有できるという利点がある．デモソフトウェアは，むやみに高機能化せず，当初の最適化コンセプトに特化したものとしたほうがフィードバックを受けやすい．経験的な目安としては，図 8 に例示するように，主要な入出力（表・グラフなど）や各種ボタンが一画面に入る程度のコンパクトなものがよいようである．デモソフトウェア開発は，たとえば Electron [12] といったデスクトップアプリケーションフレームワークを用いることで比較的少ない工数で実現できる．ファイル入出力など，多くの案件で流用可能な機能も多いため，一度テンプレートを作っておけば，さまざまな案件に対して効率的なデモソフトウェアの開発が可能となる．

6.　おわりに

本稿では，著者にて実施した掲題の技術開発事例を数理最適化の視点から紹介し，とくに本最適化で扱う「最大値最小化」の構造の問題に対して，メタヒューリスティクスの利用や，探索空間の制限や目的関数の工夫など，同問題の良質な近似解を求めるための方法をいくつかとりあげた．また本稿では，数理最適化の応用プロセスの観点から，数理最適化ソルバやデモソフトウェアの利活用による応用開発プロセスの効率化についても言及した．実務では「最大値最小化」や「平準化」など，分枝限定法が苦手とする問題構造は多数あり，最終的には専用のアルゴリズムの構築が求められる場合も多い．しかしながら，考慮すべき制約条件が検討の過程で変遷・追加となることも多いため，検討初期からの専用アルゴリズムの作りこみはリスクが高い．そこで，ユーザからのフィードバックを受けつつ解くべき問題を明確にすることを第一に行う．このフェーズにおいては Numerical Optimizer のようなメタヒューリスティクスを具備した汎用ソルバは有用であり，その単純適用，あるいは問題構造の知見を反映した使い方の工夫で，この目的に叶う品質の解が得られることも多い．

いずれについても著者の限られた経験・見識に基づくものであり恐縮ではあるが，本稿で紹介した知見・アイデアが少しでも実務の最適化に携わる諸兄のお役に立てば幸甚である．

謝辞　本稿執筆に際して，（株）NTT データ数理システムの藤井浩一氏，多田明功氏，原田耕平氏より数多くの貴重なご意見を賜りました．この場を借りて御礼申し上げます．

参考文献

[1] 東京電力エナジーパートナー，「電気料金の計算方法」，https://www.tepco.co.jp/ep/corporate/charge_c2/decision02.html（2020 年 11 月 24 日閲覧）
[2] 株式会社 IHI，“「生産」に踏み込んだ工場エネルギー最適化，”IHI 技報，**58**, pp. 12–15, 2018.
[3] F. Glover, “Future paths for integer programming and links to artificial intelligence,” *Computers & Operations Research*, **13**, pp. 533–549, 1986.
[4] 柳浦睦憲，茨木俊秀，『組合せ最適化—メタ戦略を中心として—』，朝倉書店，2001.
[5] 宮代隆平，松井知己，“ここまで解ける整数計画，”システム／制御／情報，**50**, pp. 363–368, 2006.
[6] NTT データ数理システム，Numerical Optimizer, https://www.msi.co.jp/nuopt/（2020 年 7 月 30 日閲覧）
[7] K. Nonobe and T. Ibaraki, “A tabu search approach to the constraint satisfaction problem as a general problem solver,” *European Journal of Operational Research*, **106**, pp. 599–623, 1998.
[8] K. Nonobe and T. Ibaraki, “An improved tabu search method for the weighted constraint satisfaction problem,” *INFOR*, **39**, pp. 131–151, 2001.
[9] 久保幹夫，“数理計画ソルバーを用いたメタ解法，”システム／制御／情報，**50**, pp. 357–362, 2006.
[10] 小熊祐司，“工場のピーク電力抑制を目的とした生産設備群の運転スケジュール最適化，”NTT データ数理システムユーザーコンファレンス 2017 講演論文集，2017.
[11] 小熊祐司，大野正夫，“電力ピークシフトを目的とした生産設備スケジューリング問題とその近似解法，”電気学会研究会資料，ST2018(140-150), pp. 17–24, 2018.
[12] GitHub, Inc., Electron, https://www.electronjs.org（2020 年 7 月 30 日閲覧）