

機械の種類を考慮した調理手順最適化の 値範囲と big-M 法を用いた離接制約の比較

石野 ちあき^{1,a)} 森本 尚之^{2,b)} 山田 俊行^{1,c)}

概要：給食や病院食など、日々変わる献立で大量の調理をする施設では、時間内に業務を全て終えるために、効率的な作業計画を立てることが重要である。しかし、現実の調理では様々な要素を考慮しなければならず、最適な作業計画を立てることは困難である。本研究では、調理業務効率化の支援を目標とし、調理手順最適化問題を FJSSP を応用した混合整数線形計画問題として定式化する。開発した数理モデルを利用すれば、専用ソルバーを開発することなく汎用 MILP ソルバーで容易に問題を解ける。さらに、制約と決定変数を削減するために、決定変数の値範囲を用いて離接制約を再定式化し、big-M 法と比較する。

Disjunctive Constraints Using Integer Range for Food Preparation Scheduling Considering Machine Type

1. はじめに

学校給食や介護食、病院食のような、日々変わる献立で大量の食事を作る施設では、時間内に調理を終えるために作業の順序や調理者の配置を毎日考える必要がある。特に病院食は、患者個々の身体状態や病状を考慮した食事を提供するため食事の種類が多く、調理業務は複雑になる。しかし、調理施設の設備条件、調理者の数や能力、作業配分による調理者への負担など、考慮すべき要素が多く、効率の良い調理手順を決定するのは困難である。

より効率的な調理業務を支援するために、調理手順最適化の研究がされている。調理手順最適化は、各料理の調理手順を「切る」などの工程に分解し、それらの処理順序を異なる料理間で適切に並び替えることで、最適な調理手順を探索する問題である [1]。松島ら [6] は、手作り料理を対象とした調理ガイダンスシステムを開発している。Web 上のレシピを入力として、SA(Simulated Annealing) を用いたアルゴリズムで調理手順を最適化し、求めた調理手順を元に調理のガイダンスを行う。食材や調理方法に応じた作業

時間の算出、調理者の習熟度に応じたガイダンスなど、実用化に向けた実装がされている。山吹ら [2] は、飲食店を対象とした同時卓提供のための調理支援システム開発を目指したモデルを提案している。調理に固有の制約を考慮するにはモデル化能力に限界があるとし、単純な SA を用いた手法で実験的評価をしている。

調理のような現実問題の最適化をするには二つのアプローチがある。一つ目は、汎用の数理計画ソルバーを利用する方法である。実問題を数理モデルで表現できれば、専用アルゴリズムを開発することなく汎用ソルバーで容易に問題を解ける。しかし、汎用アルゴリズムでは解ける問題の規模には限界があるうえ、現実の複雑な構造や特性を数学的構造に落とし込むのは難しい。二つ目は、専用の最適化アルゴリズムを開発する方法である。問題の構造や特性に適したアルゴリズムを開発できれば、大規模で複雑な問題でも効率的に解ける。しかし、効率的なアルゴリズムを開発するには十分な知識と技術が必要なうえ、手間もかかる。

特に、調理手順最適化問題と類似しているジョブショップスケジューリング問題は、NP 困難な問題として知られている。効率的に解くことが困難な実問題を解く場合には、まずは汎用モデルを元に問題を扱い、汎用数理計画ソルバーを用いて検討した後に専用アルゴリズムの開発を検討することで、アルゴリズムの開発、修正に要する手間を削減できる。松島ら [6] は手作り料理を対象として、山吹ら [2] は飲

¹ 三重大学大学院工学研究科
Tsu, Mie 514-8507, Japan

² 三重大学地域人材教育開発機構
Tsu, Mie 514-8507, Japan

a) ishino@cs.info.mie-u.ac.jp

b) morimoto@cs.info.mie-u.ac.jp

c) toshi@cs.info.mie-u.ac.jp

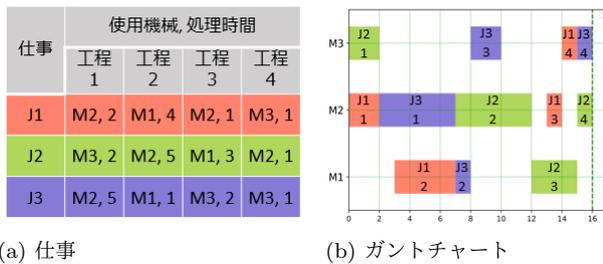


図 1 JSP の例
Fig. 1 An example of JSP

食店を対象としてそれぞれ SA を用いたアルゴリズムを開発しているが、汎用ソルバーで解けるような調理手順最適化問題の数理モデルを開発すれば、専用アルゴリズムを開発する前に容易に問題の検討ができる。

本研究では、病院食のような大量の食事を提供する調理施設の効率的な業務を支援することを目標として、調理手順最適化問題を混合整数線形計画問題 (Mixed Integer Linear Programming, MILP) として定式化する。提案モデルを汎用 MILP ソルバーで解くことで、総調理時間を最小化した調理スケジュールが求められる。特に、機械の種類を考慮するという特性を big-M 法を用いて数学的に表したモデルを開発する。さらに、制約と決定変数を削減するために、big-M 法を用いた制約を決定変数の値範囲として再定式化したモデルを開発する。

2. 関連問題

2.1 スケジューリング問題

類似した問題としてジョブショップスケジューリング問題 (Job Shop Scheduling Problem, JSSP) [3] がある。JSSP は、複数の仕事を限られた機械で処理する場合に、最大完了時刻を最小にする処理スケジュールを決定する問題である。最大完了時刻とは、全ての仕事の処理が完了するスケジュールの長さのことであり、メイクスパンとも呼ぶ。各仕事は順序付けられた工程で構成されており、全ての工程に処理機械と処理時間が与えられている。JSSP には二つの制約がある [7]。一つ目は、仕事の工程処理順序の遵守である。定められた順に機械を使い、途中で機械の使用順序を変更してはならない。二つ目は、機械による仕事の同時処理の禁止である。機械は、同時に最大で一つしか仕事を処理できず、処理の一時的な中断と再開を認めない。JSSP の例を図 1 に示す。

JSSP を拡張し機械の種類を考慮した問題が、FJSSP (Flexible JSSP) である。FJSSP では、「各工程は特定の集合のうちどの機械で処理しても良い」という制約を許す。例えば、図 1(a) では仕事 J1 の工程 1 の処理機械が M2 に指定されているが、FJSSP では処理機械を {M1, M2} のように集合で指定し、M1 または M2 のどちらの機械で処理しても良い、とすることができる。つまり、指定された機械の

集合を機械の種類と見なせば、FJSSP では機械の種類が考慮されており、処理機械を種類で指定できると言える。本研究では、仕事に料理の品目を、機械に調理機器や調理人を対応させ、FJSSP を応用して調理手順最適化問題を定式化する。

2.2 big-M 法

複数の制約のうち少なくとも一方の制約が満たされるとき、それらの制約を離接制約と呼ぶ [4]。JSSP の制約の一つである工程処理順序の順守や、本研究で扱う複数種類から処理機械を選択する制約は、離接制約で表現される。離接制約を線形制約として定式化するために、本研究では、Manné [5] が開発した big-M 法を用いる。Manné は、非常に大きな正の定数である big-M と、二値変数 y_{jk} を用いて同時処理禁止制約を表している。 y_{jk} は、仕事 j が仕事 k より前に処理されるときに 1 を、それ以外の場合に 0 をとる。 T はスケジュールを計画する期間、 $x_j \in \{0, 1, \dots, T\}$ は仕事 j の開始日、 a_j は仕事 j の処理に必要な日数を表す。仕事 j と k が同じ機械を使うとき、式 (1) と式 (2) のように、一方の処理を終えてからでないで他方の処理を開始できないという制約を課す。

$$\text{either } x_j - x_k \geq a_k \tag{1}$$

$$\text{or else } x_k - x_j \geq a_j \tag{2}$$

このとき、式 (1) と式 (2) の二つの順序制約が同時に成立することは有り得ない。つまり、いずれか一方の制約のみが成立する離接制約を表している。この制約は、二値変数 y_{jk} を用いて次のように表せる。

$$(T + a_k)y_{jk} + (x_j - x_k) \geq a_k \tag{3}$$

$$(T + a_k)(1 - y_{jk}) + (x_k - x_j) \geq a_j \tag{4}$$

$y_{jk} = 1$ のとき、式 (4) は仕事 j が仕事 k より前に処理される制約を満たし、式 (3) は左辺に非常に大きな値を加えることで、 x_j, x_k の値によらず式を成り立たせている。 $y_{jk} = 0$ の場合も同様に、二値変数と大きな正の定数 $(T + a_k)$ をかけ合わせることで、相反する順序制約を取り除いている。big-M 法は、離接制約を線形制約として実現するための手法である。

3. 提案手法

3.1 モデルの拡張

並行作業などの工夫を取り入れたより効率的な調理手順を求めるために、本研究では、二種類の並行作業と複数種類から機械を選ぶことを考慮した調理手順最適化問題を考える。

3.1.1 調理機器による複数工程の並行作業

JSSP では機械による仕事の同時処理禁止が定められているが、本研究では機械による複数工程の同時処理を可能にした調理モデルを提案する。調理の際、同じ機器を用い

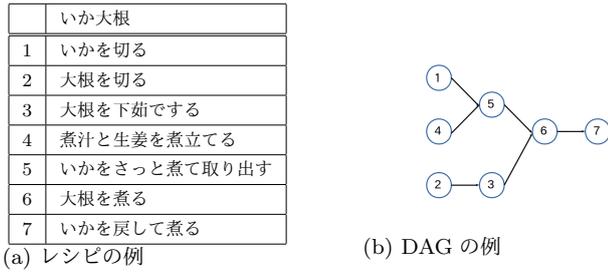


図 2 レシピと DAG の例

Fig. 2 An example of recipe and DAG

る工程をまとめて処理する場合がある。例えば、オープンやスチームコンベクションのような調理機器は、複数の段にそれぞれ別の食材を入れられるため、温度等の条件が同じ工程であればまとめて調理し、時間差で取り出すことも可能である。まとめて処理可能な複数の工程を同時処理可能にすることで、より効率的な調理手順が得られる場合がある。

3.1.2 同じ料理内の複数工程の並行作業

料理のレシピは一列に、すなわち全順序で書かれているが、工程処理順序とは関係ない、不要な順序が含まれている場合がある。例えば、図 2(a) のレシピでは、工程 (1),(2) で扱う食材は異なっており、それぞれ別で処理が進められるにもかかわらず、二つの工程間には順序制約が含まれている。このような不要な制約を除き、各料理の調理手順を有向非巡回グラフ (Directed Acyclic Graph, 以下 DAG という) で表す。すると、図 2(a) のレシピは (b) のように DAG で表せる。半順序で表すことで、同じ料理内の調理工程であっても並行作業が可能になることがあるため、より効率的な調理手順が得られる場合がある。

3.1.3 複数種類から処理機械を選ぶ

機械の種類を考慮するとき、仕事によって機械の分類方法が異なる場合がある。例えば、ある仕事を処理するときにはコンロとオープンと同じ種類の機械として扱っても問題ないが、別の仕事ではコンロとオープンを区別したい、という場合がある。どの仕事を処理するときも、同じ種類に属する機械の区別はつかないような種類分けをするために、次の手順で機械の種類を考える。

- (1) 作業を分類する
- (2) 各作業にとって区別のつかない機械の集合を考える
- (3) 全ての作業にとって区別のつかない最小単位で機械を分類する
- (4) 最小単位をそれぞれ「種類」とし、その和集合から処理機械を選ぶ

つまり、「複数の種類の機械のうちどれで処理してもよい」という制約が実現できれば、仕事によって使える機械の種類が異なっても、各仕事に合った処理機械を選べる。これにより、機械を別の仕事で使用中でも代替機械で処理できる可能性があるため、より効率的な調理手順が得られる

表 1 制約の比較

Table 1 Comparison of constraints

制約	FJSSP	提案モデル
仕事の工程処理順序の遵守	○	○
特定の種類から処理機械が選べる	○	○
特定の複数の種類から処理機械が選べる		○
機械による複数工程の同時処理を許す		○
同じ仕事内の複数工程の並行作業		○
一つの工程による複数機械の使用		○
機械を使わない工程の存在を許す		○
前後の工程で同じ機械を使うことを許す		○

表 2 インスタンス

Table 2 Instances

J	仕事の集合
V	工程の集合
$\forall j \in J \quad V_j \subseteq V$	仕事 j の工程の集合 ただし、 $\forall j \in J \quad j \neq j' \Rightarrow V_j \cap V_{j'} = \emptyset$ かつ $\bigcup_{j \in J} V_j = V$
$E_o \subseteq V \times V$	工程間の順序関係
$M = \{1, 2, \dots, M \}$	機械の集合
$K = \{1, 2, \dots, K \}$	機械の種類集合
$\forall k \in K \quad M_k \subseteq M$	種類 k の機械の集合 ただし、 $\forall k, k' \in K \quad k \neq k' \Rightarrow M_k \cap M_{k'} = \emptyset$ かつ $\bigcup_{k \in K} M_k = M$
$\forall v \in V \quad K_v \subseteq K$	工程 v の処理機械の種類集合
$\forall v \in V \quad t_v \in \mathbb{Z}_0^+$	工程 v の処理時間
$E_p \subseteq V \times V$	工程間の同時処理禁止関係
$w \in \mathbb{Z}^+$	非常に大きな正の定数
$T \in \mathbb{Z}^+$	計画期間

場合がある。

3.1.4 制約の比較

その他の拡張点も含め、提案モデルの制約と、FJSSP の制約を比較し、表 1 に示す。

3.2 定式化

3.2.1 数理モデル

調理手順最適化問題を次の混合整数線形計画問題として定式化する。

インスタンス：

表 2 にインスタンスを示す。 \mathbb{Z}_0^+ は 0 か正の整数集合、 \mathbb{Z}^+ は 0 を含まない正の整数集合を表す。

本研究における機械の種類は、3.1.3 で述べたように機械を最小単位で分類するため、全ての種類は互いに素であり、機械全体の集合の直和分割で表される。さらに、混合整数線形計画問題として定式化するために、機械の種類を整数範囲の直和分割で表現する。各 M_k の最小の要素を $M_k.first$ と表し、最大の要素を $M_k.last$ と表す。

決定変数：

決定変数を表 3 に示す。 $x_{v,k}$ は複数種類から処理機械を選ぶ制約を、 $a_{v,v'}, b_{v,v'}, c_{v,v'}, d_{v,v'}$ は同時処理禁止制約を big-M 法で表すために用いる二値変数である。

表 3 決定変数

Table 3 Decision variables

	$C_{\max} \in \{0, 1, \dots, T\}$	メイクスパン
$\forall v \in V$	$s_v \in \mathbb{Z}_0^+$	工程 v の開始時刻
$\forall v \in V$	$m_v \in M$	工程 v の処理機械
$\forall v \in V \quad \forall k \in K_v$	$x_{v,k} \in \{0, 1\}$	二値変数
$\forall v \in V \quad \forall v' \in V$	$a_{v,v'}, b_{v,v'}, c_{v,v'}, d_{v,v'} \in \{0, 1\}$	二値変数

目的関数：

$$\min C_{\max} \quad (5)$$

制約：

$$\forall v \in V_j \quad C_{\max} \geq s_v + t_v \quad (6)$$

$$\forall (v, v') \in E_o \quad s_{v'} \geq s_v + t_v \quad (7)$$

$$\forall v \in V \quad \sum_{k \in K_v} x_{v,k} = 1 \quad (8)$$

$$\forall v \in V \quad \forall k \in K_v$$

$$\begin{cases} m_v \geq M_k \cdot \text{first} - w(1 - x_{v,k}) \\ M_k \cdot \text{last} \geq m_v - w(1 - x_{v,k}) \end{cases} \quad (9) \quad (10)$$

$$\forall v \in V \quad \forall v' \in V$$

$$K_v \cap K_{v'} = \emptyset \quad \wedge \quad (v, v') \in E_p \Rightarrow$$

$$\begin{cases} a_{v,v'} + b_{v,v'} + c_{v,v'} + d_{v,v'} = 1 & (11) \\ m_v > m_{v'} - w(1 - a_{v,v'}) & (12) \\ m_{v'} > m_v - w(1 - b_{v,v'}) & (13) \\ s_v \geq s_{v'} + t_{v'} - w(1 - c_{v,v'}) & (14) \\ s_{v'} \geq s_v + t_v - w(1 - d_{v,v'}) & (15) \end{cases}$$

目的関数と各制約の意味について説明する。式 (5) は、メイクスパンの最小化が目的であることを表す。式 (6) は、メイクスパンが全ての料理の完成時刻のうち最も遅い時刻を表すことを示す。式 (7) は、工程処理順序の遵守を表す。二つの工程に順序制約があるとき、一方の処理を終えてからでない、他方の処理を始められないという制約を課す。工程処理順序を DAG で表すことで、同じ料理内の複数工程の並行作業を可能にしている。式 (8) から式 (10) は、複数種類から処理機械を選ぶ制約を表している。式 (11) から式 (15) は、機械による工程の同時処理の禁止を表す。同時処理禁止関係を表す E_p を用いて制約を追加することで、調理機器による複数工程の並行作業を可能にしている。

3.3 同時処理禁止制約の線形化

式 (11) から式 (15) の同時処理禁止制約について説明する。機械の種類を考慮するとき、処理機械 $m_{j,v}$, $m_{j',v'}$ を決定変数とすれば、同時処理禁止制約は式 (16) のように書ける。

$$(v, v') \in E_p \quad \wedge \quad m_v = m_{v'} \Rightarrow$$

$$s_v \geq s_{v'} + t_{v'} \quad \vee \quad s_{v'} \geq s_v + t_v \quad (16)$$

同じ機械を使う二つの工程が同時処理不可能であるとき、一方の処理を終えてからでない、他方の処理を始められないという制約を課す。

しかし、式 (16) の同時処理禁止制約は論理和を用いた制約になっているため、混合整数線形計画問題として定式化するために線形制約に変形することを考える。式 (16) の二つの不等式の論理和を R と置き、同値変形すると

$$(v, v') \in E_p \quad \wedge \quad m_v = m_{v'} \Rightarrow R \quad (17)$$

$$(v, v') \in E_p \Rightarrow m_v \neq m_{v'} \vee R \quad (18)$$

$$(v, v') \in E_p \Rightarrow$$

$$m_v > m_{v'} \quad \vee \quad m_{v'} > m_v \quad \vee$$

$$s_v \geq s_{v'} + t_{v'} \quad \vee \quad s_{v'} \geq s_v + t_v \quad (19)$$

となる。すなわち、式 (19) の四つの不等式のうち一つが成立するというのである。これは big-M 法を用いて式 (11) から式 (15) のように表せる。 $a_{v,v'}$ または $b_{v,v'}$ のいずれかが 1 であるとき、二つの工程が異なる機械を使用することを表す。 $c_{v,v'} = 1$ のとき、工程 v' が工程 v よりも先に処理されることを表す。 $d_{v,v'} = 1$ のとき、工程 v が工程 v' よりも先に処理されることを表す。これらの二値変数に w をかけ合わせ、右辺から非常に大きな値を差し引くことにより、三つの制約式を決定変数の値によらず成り立たせている。

3.4 複数種類から機械を選ぶ制約の big-M 法による表現

本研究で提案するモデルでは、3.1.3 で述べたように、複数種類から処理機械を選べるようにモデルを拡張している。例えば、工程 v の処理機械の種類が $K_v = \{k_1, k_2\}$ であるとする、工程 v は種類 k_1 または k_2 の機械で処理できることを表し、 $m_v \in M_{k_1} \cup M_{k_2}$ を満たす。複数種類から処理機械を選ぶ制約は式 (20) のように書ける。

$$\forall j \in J \quad \forall v \in V_j \quad \bigvee_{k \in K_v} m_v \in M_k \quad (20)$$

式 (20) は、big-M 法を用いて式 (8) から式 (10) のように表せる。 $x_{v,k} = 1$ のとき、工程 v が種類 k の機械で処理されることを表す。

3.5 複数種類から機械を選ぶ制約の値範囲による表現

3.4 の離接制約を、決定変数の値範囲を用いて表すことで、決定変数と制約を削減する手法を提案する。決定変数 m_v のとり得る値は、式 (20) より、式 (21) のように表せる。

$$\forall j \in J \quad \forall v \in V_j \quad m_v \in \bigcup_{k \in K_v} M_k \quad (21)$$

つまり、 K_v の全ての要素が連続した整数になるように機械に整数値を割り当てられる場合、決定変数 m_v の値範囲は連続した整数範囲に変換できる。例えば、図 3 で工程「fry」

$K = \{$	1,	2,	3 }		
$M = \{$	1,	2,	3,	4,	5 }
Machine: LH heater, stove1, stove2, steam convection, oven m_v					
fry	○		○		[1,4]
saute	○		○		[1,4]
bake					[4,5]
roast	○		○		[1,5]

図 3 値範囲の例

Fig. 3 An example of integer range

表 4 時間内に解けた問題数

Table 4 The number of problems solved in time

仕事数	big-M	値範囲
4	20	20
6	17	17
8	11	12
10	7	7
12	3	2

の処理機械の種類集合は $K_{fry} = \{1, 2\}$ であるので、処理機械の値範囲は $m_{fry} \in \{1, 2, 3, 4\}$ となる。

3.4での離接制約は、式(8)から式(10)で表されるように、一つの決定変数 m_v に対して $2|K_v| + 1$ 個の制約と $2|K_v|$ 個の決定変数が必要であった。一方、値範囲で表現した場合は、これらの制約と決定変数は不要となる。

4. 実験

本研究で提案した制約によって得られるスケジュールの妥当性を評価するために計算機実験を行う。モデラー PuLP [9] を用いてモデルを作成し、汎用 MILP ソルバー SCIP [10] に与え問題を解く。Web 上のレシピ [11] から 36 種類の料理のデータを準備し、その中からランダムにいくつかの料理を選択することで献立、すなわち問題を作成する。評価するモデルは、複数種類から機械を選ぶ制約を big-M 法で定式化したモデルと、値範囲で定式化したモデルの二種類である。仕事数(料理数)が 4, 6, 8, 10, 12 の 5 段階の規模の問題を用意し、各問題規模ごとに 20 種類の問題を制限時間 3600 秒で解く。評価指標は、制限時間内に解けた問題数、big-M 法モデルと比べた値範囲モデルの計算時間の増加割合、制約数、決定変数の数、二値変数の数、整数変数の数である。

4.1 実験結果

表 4 に時間内に解けた問題数を示す。表 5 に決定変数と制約の数を示す。図 4 に仕事数 4 の場合に big-M モデルと比較した値範囲モデルの計算時間の増加割合を示す。同様に、図 5 仕事数 6 の場合、図 6 に仕事数 8 の場合、図 7 に仕事数 10 の場合、図 8 に仕事数 12 の場合を示す。

表 5 から、big-M 法のモデルに比べ、値範囲のモデルは制約数と決定変数の数がたしかに削減できていることがわか

表 5 決定変数と制約の平均数

Table 5 Average number of decision variables and constraints

仕事数	決定変数		二値変数		整数変数		制約	
	big-M	値範囲	big-M	値範囲	big-M	値範囲	big-M	値範囲
4	704	655	643	594	61	61	1716	1621
6	1536	1464	1448	1376	88	88	3787	3643
8	2775	2676	2653	2554	122	122	6819	6620
10	4289	4166	4138	4016	150	150	10513	10268
12	5784	5640	5605	5461	179	179	14067	13780

表 6 制約ごとの二値変数の割合

Table 6 Rate of binary variables per constraint

仕事数	二値変数の平均数	式(12)-(15)	割合	式(9)-(10)	割合
4	643	594	92.4%	49	7.6%
6	1448	1376	95.0%	72	5.0%
8	2653	2554	96.3%	99	3.7%
10	4138	4016	97.1%	122	2.9%
12	5605	5461	97.4%	144	2.6%

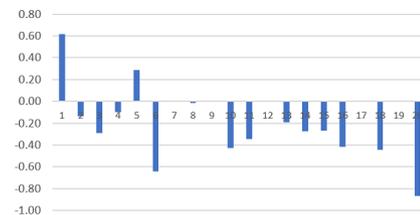


図 4 計算時間の増加割合 (仕事数 4 の場合)

Fig. 4 Increase rate of calculation time (for 4 jobs)

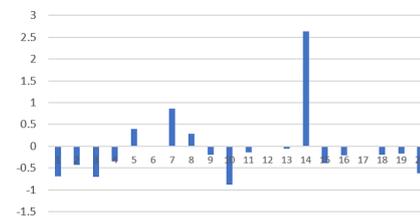


図 5 計算時間の増加割合 (仕事数 6 の場合)

Fig. 5 Increase rate of calculation time (for 6 jobs)

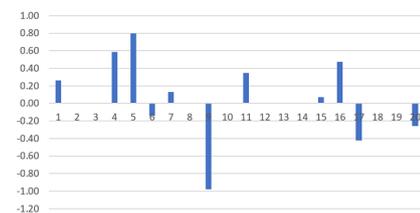


図 6 計算時間の増加割合 (仕事数 8 の場合)

Fig. 6 Increase rate of calculation time (for 8 jobs)

る。仕事数 4,6 の規模の小さい問題では、値範囲のモデルのほうが計算時間を削減できているが、仕事数 8 以上になると、値範囲のモデルのほうが計算時間が増加した問題が多かった。これは、問題の規模が大きくなるにつれ、値範囲により削減できる二値変数の割合が減少したためではないかと考える。表 6 に制約ごとの二値変数の割合を示す。問題の規模の拡大に対して、式(9)から式(10)の複数種類から

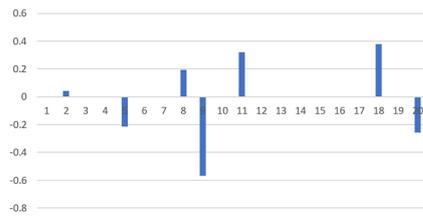


図 7 計算時間の増加割合 (仕事数 10 の場合)

Fig. 7 Increase rate of calculation time (for 10 jobs)

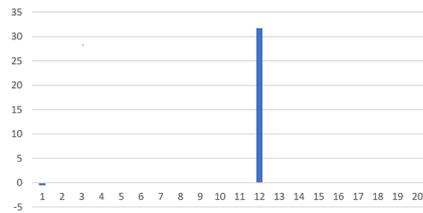


図 8 計算時間の増加割合 (仕事数 12 の場合)

Fig. 8 Increase rate of calculation time (for 12 jobs)

処理機械を選ぶ制約よりも、式 (12) から式 (15) の同時処理禁止制約に含まれる二値変数のほうが、二値変数の増加率が大きい。仕事数が 8 以上になると、削減できた二値変数の割合は二値変数全体の 5% を下回っており、計算時間削減の効果は小さくなったと考えられる。一方、時間内に解けた問題数は問題の規模によらず大差はなく、仕事数 12 の規模では全体の 1 割程度の問題が解けた。

5. 結論

機械の種類を考慮した調理手順最適化問題を、FJSSP を応用することで混合整数線形計画問題として定式化した。さらに、決定変数の値範囲を用いて離接制約を再定式化することにより、制約と決定変数を削減したモデルを提案した。開発した数理モデルは汎用 MILP ソルバーで容易に解くことができる。小さな規模の問題に対しては値範囲を用いて再定式化したモデルのほうが高速に解ける場合が多く、仕事数が 8 以上の問題では big-M 法を用いたモデルのほうが高速に解ける場合が多いことを示した。

しかし、big-M 法は大きな値を用いるため、多用すると計算時間が増加することが知られている [8]。より大規模な問題を解く場合には、Günlük ら [12] が提案する Perspective Reformulation など、big-M 法の代替手法が必要となる可能性がある [13]。

今後、提案手法により現実の調理手順最適化問題に対して有用な解を得られるのか、実データを用いて検証する必要がある。提案したモデルには、調理者の負担度合の評価など追加で考慮すべき要素があり、モデルを改善の余地がある。モデルを拡張することで、より現実に適した有用な解が得られる可能性がある。

参考文献

- [1] 松島由紀子, 船曳信生, 中西透: 多種料理の調理順最適化アルゴリズムの提案, 情報処理学会研究報告, Vol. 2009-MPS-76, No. 13, pp. 1-6(2009).
- [2] 山吹卓矢, 小野典彦, 永田裕一: 同卓スケジューリング問題のモデル化と動的スケジューリング, 計測自動制御学会論文集, Vol. 54, No. 3, pp. 346-356(2018).
- [3] Garey, M. R. and Johnson, D. S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*(1990).
- [4] 梅谷俊治: 組合せ最適化入門: 線形計画から整数計画まで, 自然言語処理, Vol. 21, No. 5, pp. 1059-1090(2014).
- [5] Manne, S. A.: On the Job Shop Scheduling Problem, *Operations Research, INFORMS*, Vol.8, No.2, pp.219-223(1960).
- [6] 松島由紀子: 手作り料理の調理手順の最適化に関する研究, 岡山大学博士論文 (2015).
- [7] 樋野励: ジョブショップスケジューリング問題の数理表現, システム/制御/情報, Vol.61, No.1, pp.14-19(2017).
- [8] 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, Vol. 50, No. 9, pp. 363-368(2006).
- [9] COIN-OR, PuLP, <https://pypi.org/project/PuLP/>, (2020 年 11 月 10 日参照).
- [10] Zuse Institute Berlin (ZIB): SCIP (Solving Constraint Integer Programs), <https://www.scipopt.org/>, (2020 年 11 月 10 日参照).
- [11] キッコーマン株式会社: ホームクッキング, (2019 年 2 月 8 日参照) <https://www.kikkoman.co.jp/homecook/>.
- [12] Günlük, O. and Linderoth, J.: Perspective Reformulations of Mixed Integer Nonlinear Programs with Indicator Variables, *Mathematical Programming, Series B*, 124, pp.183-205(2010).
- [13] 宮代隆平, 整数計画法メモ, <http://web.tuat.ac.jp/~miya/ipmemo.html>, (2020 年 11 月 19 日参照)