修 士 論 文

機械の種類を考慮した
調理手順最適化の数理モデル

令和２年度 修了
三重大学大学院　工学研究科
博士前期課程　情報工学専攻
コンピュータソフトウェア研究室

石野　ちあき

概 要

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調理する施設では，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．例えば病院食は，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．そのため，効率的な作業計画が重要である．しかし現実の調理では，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，効率の良い作業計画を立てるのは困難である．

より効率的な調理業務の支援のために，調理手順最適化の研究がされている．調理のような現実問題の最適化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．効率的に解きにくい問題を解く場合は，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．つまり，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の支援を目標として，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得られる．汎用モデルでは解ける問題が制限的すぎるため，現実の調理の工夫を考慮し拡張した数理モデルを提案する．特に，機械の種類を考慮するという特性は単純に定式化すると非線形制約になるが，big-M 法を用いて線形制約で表したモデルを開発する．さらに、big-M 法で表現し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し，制約と決定変数を削減する．

目 次

第1章　はじめに

1.1　研究背景

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調理する施設では，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．例えば病院食は，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．そのため，効率的な作業計画が重要である．しかし現実の調理では，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，効率の良い作業計画を立てるのは困難である．

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同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．効率的に解きにくい問題を解く場合は，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．つまり，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．

1.2 研究目的

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の支援を目標として，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得られる．汎用モデルでは解ける問題が制限的すぎるため，現実の調理の工夫を考慮し拡張した数理モデルを提案する．特に，機械の種類を考慮するという特性は単純に定式化すると非線形制約になるが，big-M 法を用いて線形制約で表したモデルを開発する．さらに、big-M 法で表現し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し，制約と決定変数を削減する．これらの二つのモデルを実装し，計算機実験によって比較する．

1.3 論文の構成

本論文の構成は次の通りである．2 章で調理手順最適化問題に関連するジョブショップスケジューリング問題と，定式化で用いる big-M 法という手法について説明する．3 章で本研究で扱う調理手順最適化問題の汎用モデルから拡張する点について説明し，混合整数計画問題として定式化した数理モデルを示す．またその際に，big-M 法で表現した機械の種類を考慮する特性を決定変数の値範囲として再度定式化する．4 章で提案した二つのモデルを計算機実験によって比較し，解ける問題の規模の違いについて考察する．5 章で結論と課題を述べる．

第2章　関連問題

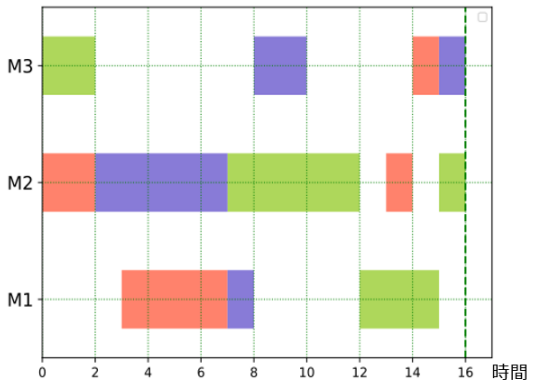
2.1　スケジューリング問題

調理手順最適化問題と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 [8]．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 2.1 に示す．図 2.1(b) ガントチャートは，縦軸を機械，横軸を時間として作業の流れを示した図である．太い水平線で作業の開始と終了を示しており，図 2.1(a) 仕事と同じ色で対応している，

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 2.1(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．

仕事	使用機械, 処理時間			
	工程 1	工程 2	工程 3	工程 4
J1	M2, 2	M1, 4	M2, 1	M3, 1
J2	M3, 2	M2, 5	M1, 3	M2, 1
J3	M2, 5	M1, 1	M3, 2	M3, 1

(a) 仕事



(b) ガントチャート

図 2.1: JSSP の例

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 (Disjunctive Constraint) と呼ぶ [4]. JSSP の制約の一つである同時処理禁止制約や, 本研究で扱う複数種類から処理機械を選択する制約は, 離接制約で表現できる. 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, Manne [5] が開発した big-M 法を用いる. Manne は, 非常に大きな正の定数である big-M と, 0-1 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, $x_j \in \{0, 1, \cdots T\}$ 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 (2.1) と式 (2.2) のように, 一方の処理を終えてからでないと他方の処理を開始できないという制約を課す.

$$\text{either} \qquad x_j - x_k \geq a_k \tag{2.1}$$

$$\text{or else} \qquad x_k - x_j \geq a_j \tag{2.2}$$

このとき, 式 (2.1) の仕事 j が仕事 k より前に処理される制約と式 (2.2) の仕事 k が仕事 j より前に処理される制約の二つの順序制約は相反しており，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.

$$(T + a_k)y_{jk} + (x_j - x_k) \geq a_k \tag{2.3}$$

$$(T + a_k)(1 - y_{jk}) + (x_k - x_j) \geq a_j \tag{2.4}$$

$y_{jk} = 1$ のとき, 式 (2.4) は仕事 j が仕事 k より前に処理される制約を満たし, 式 (2.3) 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.

第3章　提案手法

3.1　モデルの拡張

調理手順最適化問題を FJSSP として捉えるとき，FJSSP では解ける問題が制限的すぎるため現実の調理に特有の制約を考慮できない問題がある．例えば，工程の処理に複数の機械を用いることや，あるいは機械が不要な工程が存在することなど，特有の制約を許すようにモデルを拡張する必要がある．特に，本研究は大規模な調理施設の効率的な業務支援を目標としていることから，効率的に作業を進めるための工夫を取り入れた調理手順を求める必要がある．現実の調理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．それぞれの拡張した点について説明する．

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, 同じ機器を用いる工程をまとめて処理する場合がある. 例えば, オーブ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. 具体的には, 次のような二つの工程はまとめて調理できる.

例 1

- 工程 1：ほうれん草を 100 度で 3 分スチーム
- 工程 2：かぶを 100 度で 5 分スチーム

例 2

- 工程 1：くるみを 160 度で 10 分ロースト
- 工程 2：カシューナッツを 160 度で 7 分ロースト

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 3.1(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 3.1(a) のレシピは (b) のように DAG で表せる. 各頂点は工程を，各有向辺は順序関係を表す．隣接する頂点は，先行する工程が完了した後でなければもう一方の工程が開始できないことを示す．この DAG による表現は，PERT(Program Evaluation and Review Technique) などで用いられるアローダイアグラムによるプロジェクトのモデル化 [7] と本質的に同じものである.

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

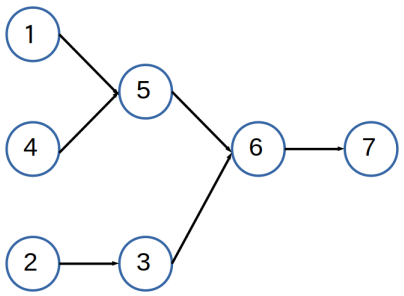
機械の種類を考慮するとき, 仕事によって機械の分類方法が異なる場合がある. 例えば, ある仕事を処理するときはコンロとオーブンを同じ種類の機械として扱っても問題ないが, 別の仕事ではコンロとオーブンを区別したい, という場合がある. どの仕事を処理するときも, 同じ種類に属する機械の区別はつかないような種類分けをするために, 次の手順で機械の種類を考える.

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」という制約を許せば, 仕事によって使える機械の種類が異なっていても, 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場合がある.

	いか大根
1	いかを切る
2	大根を切る
3	大根を下茹でする
4	煮汁と生姜を煮立てる
5	いかをさっと煮て取り出す
6	大根を煮る
7	いかを戻して煮る

(a) レシピの例



(b) DAG の例

図 3.1: レシピと DAG の例

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 3.1 に示す.

表 3.1: 制約の比較

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		○
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		○

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.

記号の説明：

表 3.2 に記号の説明を示す. $\mathbb{Z}_0^+$ 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, 混合整数線形計画法として定式化するために, 機械の種類分けを式 (3.1) のように整数範囲の直和分割で表現する. 各 M_k の最小の要素を $\min(M_k)$ と表し, 最大の要素を $\max(M_k)$ と表す.

$$
\begin{aligned}
&\forall k \in K \\
&\quad M_1 &&= \{1, 2, \cdots, |M_1|\} \\
&\quad M_2 &&= \{|M_1| + 1, \cdots, |M_1| + |M_2|\} \\
&\quad \cdots \\
&\quad M_{|K|} &&= \left\{ \sum_{k=1}^{|K|-1} |M_k| + 1, \cdots, |M| \right\}
\end{aligned}
\tag{3.1}
$$

表 3.2: 記号の説明

	J	仕事の集合
	V	工程の集合
$\forall j \in J$	$V_j \subseteq V$	仕事 j の工程の集合
		ただし, $\forall j \in J \quad j \neq j' \quad \Rightarrow$
		$V_j \cap V_{j'} = \emptyset \quad$ かつ $\quad \bigcup_{j \in J} V_j = V$
	$E_{\mathrm{o}} \subseteq V \times V$	工程間の順序関係
	$M = \{1, 2, \cdots, \lvert M \rvert\}$	機械の集合
	$K = \{1, 2, \cdots, \lvert K \rvert\}$	機械の種類の集合
$\forall k \in K$	$M_k \subseteq M$	種類 k の機械の集合
		ただし, $\forall k, k' \in K \quad k \neq k' \quad \Rightarrow$
		$M_k \cap M_{k'} = \emptyset \quad$ かつ $\quad \bigcup_{k \in K} M_k = M$
$\forall v \in V$	$K_v \subseteq K$	工程 v の処理機械の種類の集合
$\forall v \in V$	$t_v \in \mathbb{Z}_0^+$	工程 v の処理時間
	$E_{\mathrm{p}} \subseteq V \times V$	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

決定変数：

決定変数を次に示す. $C_{\max}$ はメイクスパン, s_v は工程 v の開始時刻, m_v は工程 v の処理機械を表す. $x_{v,k}$ は複数種類から処理機械を選ぶ制約を big-M 法で表すために用いる二値変数であり, $a_{v,v'}, b_{v,v'}, c_{v,v'}, d_{v,v'}$ 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.

$$
\begin{aligned}
& & & C_{\max} & \in \{0, 1, \cdots, T\} \\
& \forall v \in V & & s_v & \in \mathbb{Z}_0^+ \\
& \forall v \in V & & m_v & \in M \\
\forall v \in V \quad & \forall k \in K_v & & x_{v,k} & \in \{0, 1\} \\
\forall v \in V \quad & \forall v' \in V & & & \\
a_{v,v'}, \quad & b_{v,v'}, \quad c_{v,v'}, & & d_{v,v'} & \in \{0, 1\}
\end{aligned}
$$

目的関数：

$$\min \quad C_{\max} \tag{3.2}$$

制約：

$$\forall v \in V_j \qquad C_{\max} \geq s_v + t_v \tag{3.3}$$

$$\forall (v, v') \in E_{\mathrm{o}} \qquad s_{v'} \geq s_v + t_v \tag{3.4}$$

$$\forall v \in V \qquad \textstyle\sum_{k \in K_v} x_{v,k} = 1 \tag{3.5}$$

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

$$\begin{cases} m_v \geq \min(M_k) - w(1 - x_{v,k}) & (3.6) \\ \max(M_k) \geq m_v - w(1 - x_{v,k}) & (3.7) \end{cases}$$

$$\forall j, j' \in J \quad \forall v \in V_j \quad \forall v' \in V_{j'}$$

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

$$\begin{cases} a_{v,v'} + b_{v,v'} + c_{v,v'} + d_{v,v'} = 1 & (3.8) \\ m_v > m_{v'} - w(1 - a_{v,v'}) & (3.9) \\ m_{v'} > m_v - w(1 - b_{v,v'}) & (3.10) \\ s_v \geq s_{v'} + t_{v'} - w(1 - c_{v,v'}) & (3.11) \\ s_{v'} \geq s_v + t_v - w(1 - d_{v,v'}) & (3.12) \end{cases}$$

目的関数と各制約の意味について説明する．式 (3.2) は, メイクスパンの最小化が目的であることを表す．式 (3.3) 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 (3.4) 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と, 他方の処理を始められないという制約を課す．この制約は 3.1.2 で述べた拡張を表したものであり, 工程処理順序制約を DAG の辺で表すことで同じ料理内であっても複数工程の並行作業を可能にしている．式 (3.5) から式 (3.7) は, 3.1.3 で述べた拡張である複数種類から処理機械を選ぶ制約を表している．式 (3.8) から式 (3.12) は, 機械による工程の同時処理の禁止を表す．全ての工程のペアのうち，処理機械の種類が互いに素でなく，かつ同時処理禁止関係を持つペアにのみ制約を与える．同時処理禁止関係を表す E_{p} を前提条件とし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．この前提条件は 3.1.1 で述べた拡張を表している．

3.2.2 同時処理禁止制約の線形化

式 (3.8) から式 (3.12) 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 (3.13) のように書ける．

$$\begin{aligned}(v,v') \in E_p \quad \wedge \quad m_v = m_{v'} \quad \Rightarrow \\ s_v \geq s_{v'} + t_{v'} \quad \vee \quad s_{v'} \geq s_v + t_v\end{aligned} \tag{3.13}$$

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と他方の処理を始められないという制約を課す．

しかし, 式 (3.13) の同時処理禁止制約は論理和を用いた制約であるので, 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．まず，決定変数が制約の前提条件に含まれている問題を解決する．式 (3.13) の二つの不等式の論理和を R と置き, $A \Rightarrow B \Leftrightarrow \neg A \vee B$ を用いると式 (3.14) から式 (3.18) のように同値変形できる．

$$(v,v') \in E_p \quad \wedge \quad m_v = m_{v'} \quad \Rightarrow \quad R \tag{3.14}$$

$$\neg((v,v') \in E_p \quad \wedge \quad m_v = m_{v'}) \quad \vee \quad R \tag{3.15}$$

$$\neg((v,v') \in E_p) \quad \vee \quad (m_v \neq m_{v'} \quad \vee \quad R) \tag{3.16}$$

$$(v,v') \in E_p \quad \Rightarrow \quad m_v \neq m_{v'} \quad \vee \quad R \tag{3.17}$$

$$\begin{aligned}(v,v') \in E_p \quad \Rightarrow \\ (m_v > m_{v'} \quad \vee \quad m_{v'} > m_v) \quad \vee \\ (s_v \geq s_{v'} + t_{v'} \quad \vee \quad s_{v'} \geq s_v + t_v)\end{aligned} \tag{3.18}$$

つまり, 式 (3.18) の四つの不等式のうち少なくとも一つが成立するということである．これは big-M 法を用いて式 (3.8) から式 (3.12) のように線形制約で表せる．$a_{v,v'}$ 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十分大きな正の実数値 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, 成り立たないはずの制約式を決定変数の値によらず成り立たせている．

3.2.3 複数種類から機械を選ぶ制約の big-M 法による表現

式 (3.5) から式 (3.7) の複数種類から機械を選ぶ制約について説明する．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 の処理機械の種類の集合が $K_v = \{k_1, k_2\}$ であるとすると, 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, 工程 v の処理機械を表す決定変数 m_v は $m_v \in M_{k_1} \cup M_{k_2}$ を満たす．このことから，複数種類から処理機械を選ぶ制約は式 (3.19) のように書ける．

$$\forall j \in J \quad \forall v \in V_j \qquad \bigvee_{k \in K_v} m_v \in M_k \tag{3.19}$$

式 (3.19) は, big-M 法を用いて式 (3.5) から式 (3.7) 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．

3.2.4 複数種類から機械を選ぶ制約の値範囲による表現

3.2.3 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 (3.19) より, 式 (3.20) のように表せる．

$$\forall j \in J \quad \forall v \in V_j \qquad m_v \in \bigcup_{k \in K_v} M_k \tag{3.20}$$

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 3.2 で 工程「fry」の処理機械の種類の集合は $K_{\mathrm{fry}} = \{1, 2\}$ であるので, 処理機械の値範囲は $m_{\mathrm{fry}} = \{1, 2, 3, 4\}$ となる．

特に大規模な調理施設の場合，複数の機能を持つ万能調理機器を導入することが多く，値範囲に変換できる場合が多い．例として，加熱調理を 8 つの作業に分類し，十河ら [10] が示す加熱調理方法別ガス調理機器の分類表を用いて業務用調理機器を分類したものを表 3.3 に示す．スチームコンベクションオーブンや回転窯のように複数の調理方法が可能な機械が多い場合，整数値の割り当てを柔軟に変更できるため，値範囲に変換できる場合が多い．

3.2.3 での離接制約は, 式 (3.5) から式 (3.7) 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 $|K_v|$ 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．このように，二値変数を用いた制約ではなく決定変数の値範囲として離接制約を表す方法は，制約プログラミング (Constraint Programming, CP) におけるドメイン変数の機能と同様である．ドメイン変数はある集合の要素をとり得る変数 [11] であり，言い換えれば $x \in [a, b] \cup \cdots \cup [c, d]$ のように複数範囲の和集合の要素をとり得る変数である．本手法は複数範囲が隣接するように整数値を割り当てることでドメイン変数の機能を実現している．

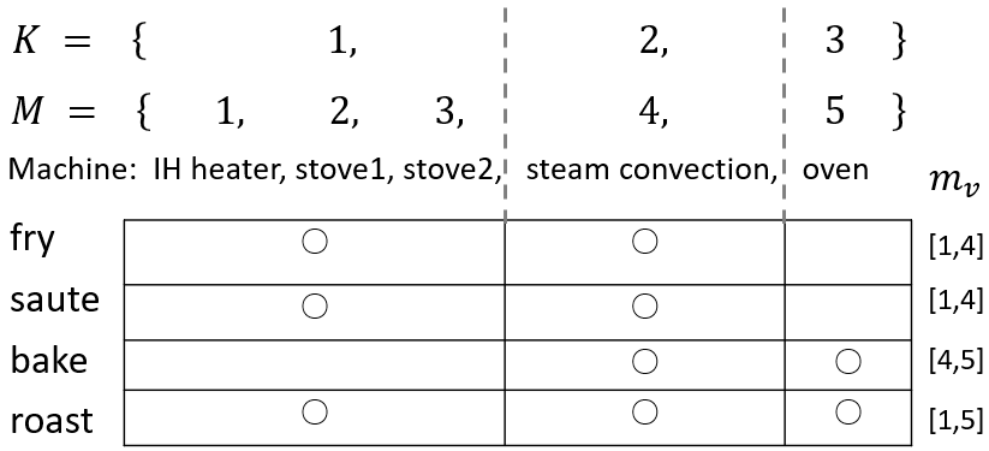


図 3.2: 値範囲の例

表 3.3: 調理方法別の給食センターの調理機器の分類例

		蒸し器	オーブン	スチコン	コンロ	回転窯	フライヤー
温式加熱	茹でる			○	○	○	
	煮る			○	○	○	
	蒸す	○	○	○	○	○	
乾式加熱	直火焼き				○		
	間接焼き			○	○		
	熱風焼き		○	○			
	炒める			○	○	○	
	揚げる			○	○	○	○

第4章　実験と評価

4.1　実験

本研究で提案したモデル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 [12] を用いてモデルを作成し, 汎用ソルバー SCIP [13] に与え問題を解く．SCIP は非商用ソルバーの中で最も高速なものの一つである [14]．Web 上のレシピ [15] 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 段階の規模で各 20 題用意する．本研究では日々作業計画を考え直す場面を想定していることと，事前実験で計算時間が 5 分以内または 60 分以上と二極化したことから，制限時間を 3600 秒に設定して各問題を解く． 機械は，レシピ内で用いている機械の合計 5 種 8 台とする．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．実験に用いた計算機の仕様を表 4.1 に示す．

表 4.1: 計算機の仕様

OS	Ubuntu 20.04.1 LTS
CPU	Intel(R) Core(TM) i7-8700 3.20GHz
RAM	8GB

4.2 big-M の値の設定

big-M 法は, 大きな値を用いるため線形緩和が悪くなり, 計算時間が増加すると言われている [9]. そのため, 実験ではできる限り小さく, かつ十分な大きさの big-M を用いる必要がある. big-M の値の設定について, 以下で説明する.

4.2.1 同時処理禁止制約の場合

式 (3.9) では, $m_v < m_{v'}$ であるときに 式 (3.9) が満たされるように w の値を設定しなければならない. すなわち, 式 (3.9) において大きな正の整数 w が満たすべき条件は,

$$\begin{aligned} m_v &> m_{v'} - w \\ w &> m_{v'} - m_v \end{aligned} \tag{4.1}$$

である. $m_{v'}, m_v \in M$ と $M = \{1, 2, \cdots, |M|\}$ より, $|M| > |m_{v'} - m_v|$ であるので, $|M|$ は w として用いるのに十分大きな値である. 同様に, 式 (3.10) における w の値にも $|M|$ を設定できる.

式 (3.11) では, $s_v \leq s_{v'} + t_{v'}$ であるときに, 式 (3.11) を満たすように w の値を設定しなければならない. すなわち, 式 (3.11) において w が満たすべき条件は,

$$\begin{aligned} s_v &\geq s_{v'} + t_{v'} - w \\ w &\geq s_{v'} - s_v + t_{v'} \end{aligned} \tag{4.2}$$

である. ここで, 変数 s_v の上限を定めるために, 計画期間 T を考える. 計画期間は, スケジュールを計画する期間を表す. 本研究では, 目的関数がメイクスパンの最小化であることと, 機械を使わない工程を許していることから, 最大計画期間は単純に全工程の処理時間の総和で考えられる. 計画期間 T を式 (4.3) のように定める.

$$T = \sum_{v \in V} t_v \tag{4.3}$$

すると, s_v の範囲は式 (4.4) のように制限できる.

$$\forall j \in J, \quad \forall v \in V_j \qquad s_v \in \{1, 2, \cdots, T\} \tag{4.4}$$

よって, $T \geq |s_{v'} - s_v|$ であるので, 式 (4.2) から $T + t_{v'}$ が w として用いるのに十分大きな値だとわかる. 同様に, 式 (3.10) における w の値にも $T + t_v$ を設定できる.

4.2.2 複数種類から機械を選ぶ制約の場合

式 (3.6) では, $m_v \leq \min(M_k)$ であるときに, 式 (3.6) が満たされるように w の値を設定しなければならない. すなわち,w が満たすべき条件は

$$\begin{aligned} m_v &\geq \min(M_k) - w \\ w &\geq \min(M_k) - m_v \end{aligned} \tag{4.5}$$

である. $m_v \in M$, $M_k \subseteq M$ と $M = \{1, 2, \cdots, |M|\}$ より, $|M| > |\min(M_k) - m_v|$ であるので, $|M|$ は w として用いるのに十分大きな値だとわかる. 同様に, 式 (3.7) における w の値にも $|M|$ を設定できる.

4.3 実験結果と考察

表 4.2 に決定変数と制約の平均数を示す．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る．図 4.1 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 4.2 に仕事数 6 の場合, 図 4.3 に仕事数 8 の場合, 図 4.4 に仕事数 10 の場合, 図 4.5 に仕事数 12 の場合を示す．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 4.2: 決定変数と制約の平均数

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

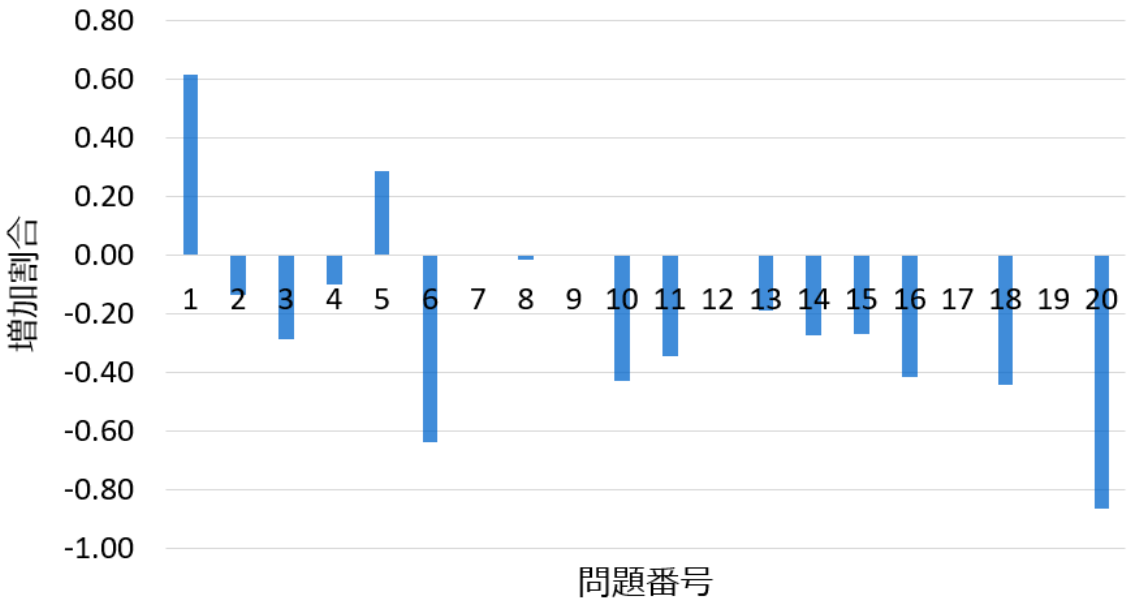


図 4.1: 計算時間の増加割合 (仕事数 4 の場合)

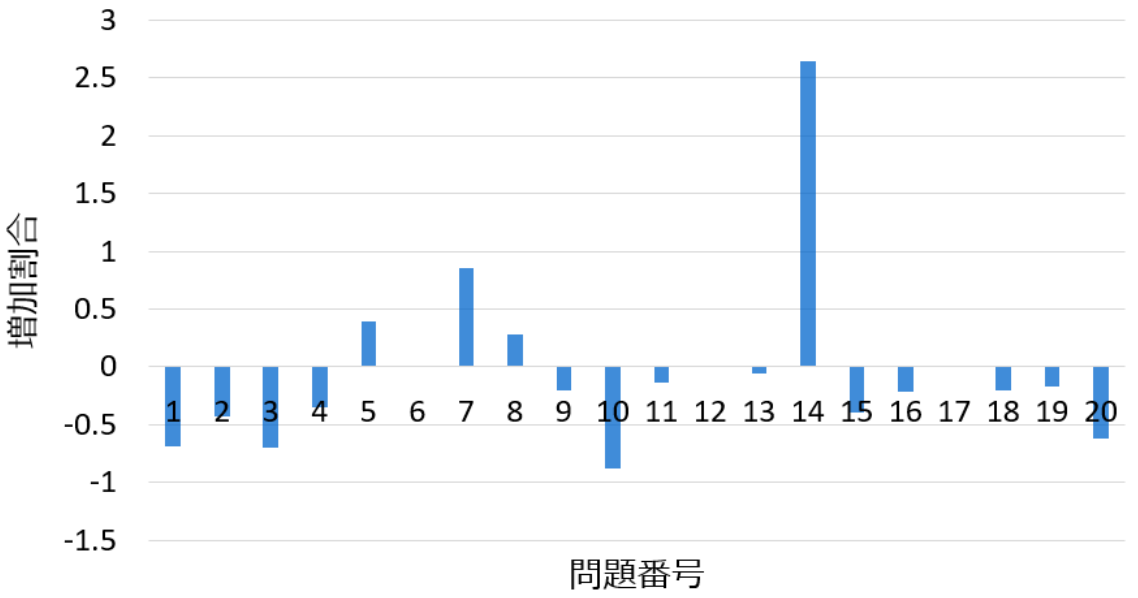


図 4.2: 計算時間の増加割合 (仕事数 6 の場合)

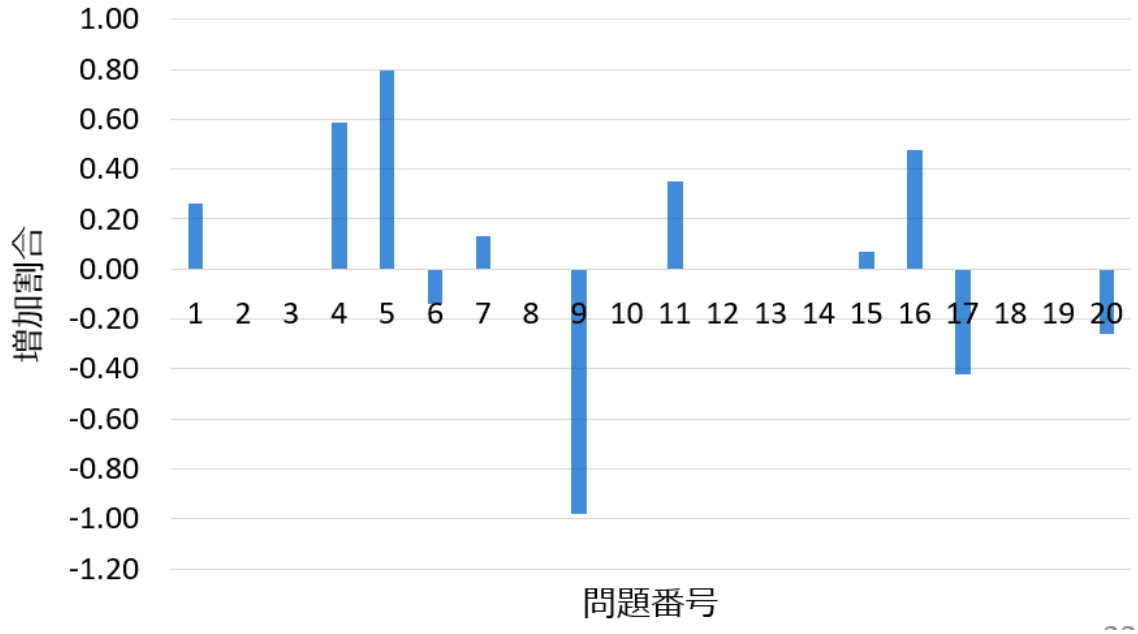


図 4.3: 計算時間の増加割合 (仕事数 8 の場合)

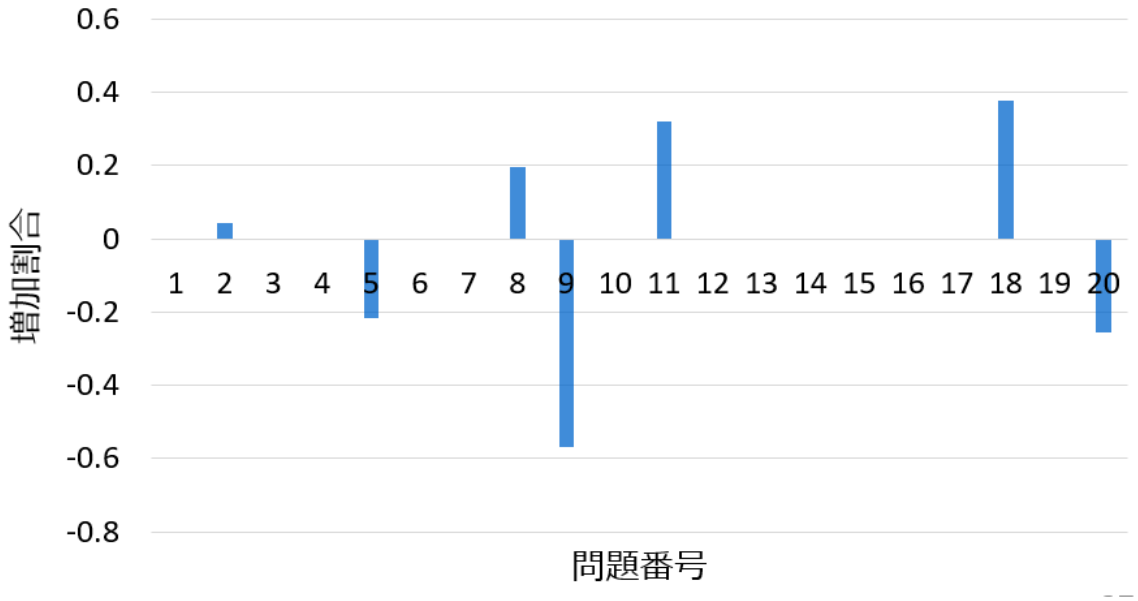


図 4.4: 計算時間の増加割合 (仕事数 10 の場合)

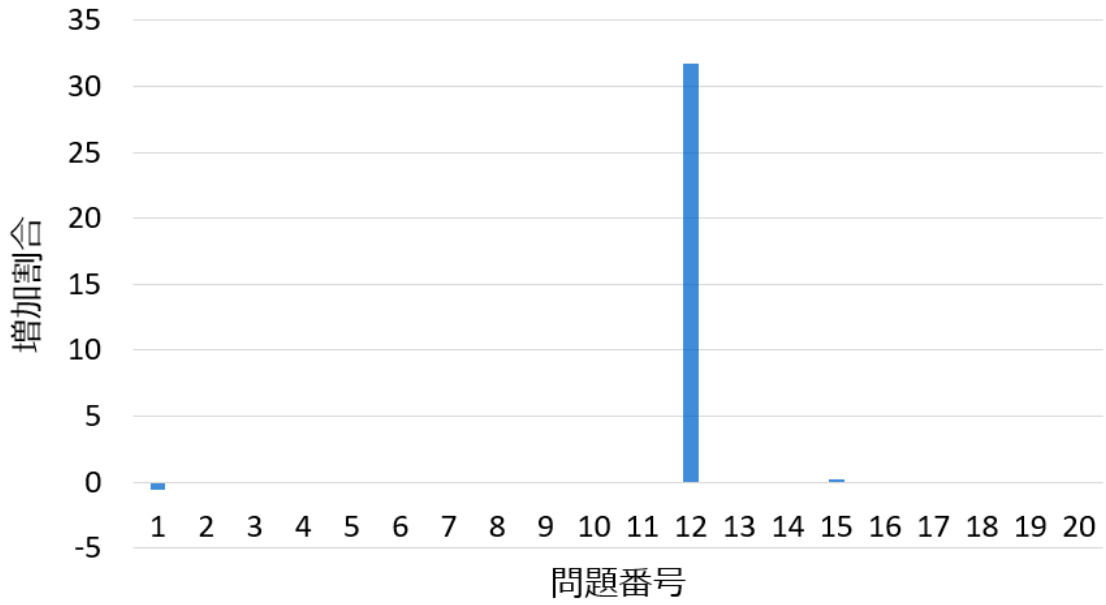


図 4.5: 計算時間の増加割合 (仕事数 12 の場合)

表 4.3 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 (3.6) から式 (3.7) の複数種類から処理機械を選ぶ制約よりも, 式 (3.9) から式 (3.12) 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．

表 4.3: 制約ごとの二値変数の割合

仕事数	二値変数の平均数	式 (3.9)-(3.12)	割合	式 (3.6)-(3.7)	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%

一方, 表 4.4 に示すように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．

表 4.4: 時間内に解けた問題数

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

表 4.5 に制限時間内に解けなかった問題の二つのモデルから得られた暫定解のギャップの比較を示す．表中の $gap_{\text{int-range}}$ は値範囲のモデルから得られた暫定解のギャップを表し，$gap_{\text{big-M}}$ は big-M 法のモデルから得られた暫定解のギャップを表す．ソルバー SCIP における相対ギャップの定義は，双対問題の最適値を $dual$，主問題の実行可能解の目的関数値を $primal$ とすると，式 (4.6) のように表される [13]．

$$relative\,gap = \frac{|primal - dual|}{\min(|dual|, |primal|)} \tag{4.6}$$

したがって相対ギャップとは，暫定解の目的関数値と最適値のギャップの上界を示している．表 4.5 より時間内に解けなかった問題の暫定解の精度を二つのモデル間で比較すると，ほとんどの問題で差は現れなかったが，仕事数 8 以下の問題では値範囲のモデルで得られた解のほうがギャップの上界が小さく，仕事数 12 になると big-M 法のモデルで得られた解のほうがギャップの上界が小さい傾向が見られた．

表 4.5: 二つのモデルから得られた暫定解のギャップの比較

	仕事数				
	4	6	8	10	12
時間内に解けなかった問題数	0	3	9	13	18
$gap_{\text{int-range}} = gap_{\text{big-M}}$ の問題数	-	2	6	9	15
$gap_{\text{int-range}} < gap_{\text{big-M}}$ の問題数	-	1	3	2	1
$gap_{\text{int-range}} > gap_{\text{big-M}}$ の問題数	-	0	0	2	2
$gap_{\text{int-range}} - gap_{\text{big-M}}$ の最大値 [%]	-	0.00	0.00	7.58	3.56
$gap_{\text{int-range}} - gap_{\text{big-M}}$ の最小値 [%]	-	−3.02	−11.11	−6.70	−2.41
$gap_{\text{int-range}} - gap_{\text{big-M}}$ の平均値 [%]	-	−1.00	−2.13	−0.14	1.30

第5章　結論と今後の課題

5.1　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．提案したモデルは並行調理や代替機械の使用など，効率的に作業を進めるための工夫が考慮されている．特に機械を考慮する特性を big-M 法を用いて線形制約で表し，さらに big-M 法で表した制約を決定変数の値範囲を用いて再度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ソルバーで容易に解ける．二つのモデルを実験的に評価し，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．

5.2　今後の課題

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．仕事数が 8 以上の問題では big-M 法のほうが高速であることを示したが，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 [9]．さらに大規模な問題を解く場合には，Günlük ら [16] 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 [17]．また，3.2.4 の複数種類から機械を選ぶ制約を決定変数の値範囲として表現できない場合，変換できない制約のみ 3.2.3 の big-M 法で表すことで定式化できるが，できるだけ多くの制約を決定変数の値範囲へ変換できるように機械へ整数を割り当てることが求められる．本研究では設備が固定された調理場における調理手順最適化を考えているが，特に調理設備が頻繁に変更されるような場合は，機械へ整数を割り当てる方法に考察の余地がある．割り当て方法の改善は今後の課題である．

謝辞

本研究を進めるにあたり，日頃よりご指導くださった山田俊行講師，河内亮周教授，森本尚之講師に深く感謝申し上げます．また，多くの意見をくださったコンピュータソフトウェア研究室の皆様，研究活動を様々な場面で支えてくださった落合美子事務員に心よりお礼申し上げます．

参考文献

[1] 松島由紀子，舩曳信生，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] Malcolm, D. G., Roseboom, J. H., Clark, C. E. and Fazar, W. , “Application of a Technique for Research and Development Program Evaluation”, Operations Research, 7, issue 5, pp. 646–669(1959).

[8] 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).

[9] 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)．

[10] 十河桜子，相墨智，伊藤あすか，西川向一,「ガス調理機器の燃焼およびその周辺技術」，日本燃焼学会誌，Vol. 52，No. 162，pp. 267–274(2010)．

[11] Apt, K. R. ,『*Principles of Constraint Programming*』, Cambridge University Press(2003).

[12] COIN-OR, PuLP, `https://pypi.org/project/PuLP/`, (2020 年 11 月 10 日参照).

[13] Zuse Institute Berlin (ZIB)，SCIP (Solving Constraint Integer Programs), `https://www.scipopt.org/`, (2020 年 11 月 10 日参照).

[14] 宮代隆平,「整数計画ソルバー入門」，オペレーションズ・リサーチ，Vol.57，pp.183–189(2012)

[15] 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/`.

[16] Günlük, O. and Linderoth, J. , “Perspective Reformulations of Mixed Integer Nonlinear Programs with Indicator Variables”, Mathematical Programming, Series B, 124, pp.183–205(2010).

[17] 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)