食品工場におけるスケジューリング問題に対する RCPSP ソルバーを用いた解法

2015SS024 加納沙綾　2015SS046 中村琢美
指導教員：佐々木美裕

1　はじめに

本研究では, コンビニエンスストアで販売する弁当等の生産を行う工場における 2 つのスケジューリング問題について考える.

対象とする工場では, 受注した弁当やおにぎりの生産, 配送を行っている. この工場には, 加熱, 炊飯, 仕込み, 盛付の 4 つの部門があり, 部門ごとにタイムテーブルを作成しているが, 手作業で行なっているため, 膨大な時間がかかるうえに, 最適な生産スケジュールを作成できているかわからないことが問題となっている.

江川・中西 [1] は, 短時間で最適な生産スケジュールとタイムテーブルを作成するために, 資源制約付きスケジューリング問題としてモデル化を行った. 工場に勤務する作業者にはそれぞれ可能な作業, 不可能な作業があり, 能力に違いがある. 作業者を能力別のグループに分け, スケジュールを作成することにより, 作業者の能力を考慮したスケジュール作成が可能となった. しかしながら, グループ内の作業者 1 人 1 人を区別していないため, 作業者ごとのスケジュールを出力することができない.

本研究では, 作業者 1 人 1 人に作業を割り当て, 作業者別スケジューリングモデルを作成する. また 4 つの部門の中で, 複雑な制約が多い炊飯部門における炊飯の最適スケジューリングモデルを作成する. これによって労務管理の簡易化 (残業時間の削減・組織の柔軟性向上) と生産性の向上が期待できる.

2　RCPSP としてのモデル化

2.1　RCPSP

資源制約付きプロジェクトスケジューリング問題のことであり, 英語表記 (resource constrained project scheduling problem) の頭文字をとり, RCPSP[2] と呼ぶ. これは, プロジェクトを構成する各作業を実施する際に必要となる資源 (設備や人員など) の利用可能量等に関する条件がある中で, 各作業の開始時刻および終了時刻を決定する問題である.

2.2　OptSeq

OptSeq は, メタヒューリスティックを基とし作成されたスケジューリング最適化ソルバーであり, Python のモジュールとして提供されている. 様々なタイプの資源制約の記述が可能である. 実務における様々な条件を含むスケジューリング問題に対して, 短時間で良好な解を探索することができる [2].

2.3　モード

OptSeq では, 各作業の処理方法を複数設定することが可能である. この作業の処理方法をモードと呼ぶ. 各作業に対して複数のモードを設定すると, 条件を満たす最適なモードを OptSeq が選択する. 作業時間や資源の使用量はモードごとに決めることができる [2].

図 1 は, 作業 1 に対して作業時間や資源の使用量の異なるモードを 3 つ設定した例である. 資源の使用量（図 1 では作業者の数) の最小化を目的にするか, 作業時間の最短化を目的にするかによって、選択されるモードは変わる.

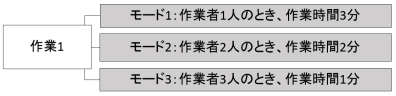

図 1　資源の使用量や作業時間の異なるモード

2.4　モデル化の方法

作業者別スケジューリングモデル, 炊飯の最適スケジューリングモデルの両方を RCPSP としてモデル化をする. スケジューリングをする上で, 各作業にモードを設定し, OptSeq を用いて最適なモードを選択することで, 最適なスケジュールを作成する.

3　作業者別スケジューリングモデル

3.1　現状

江川・中西 [1] は, 作業者の能力を考慮した生産スケジューリングモデルを提案した. これにより, 実際に勤務中の作業者のみで実行可能なスケジュールを求めたが, 同じ能力を持つ複数の作業者が勤務中の場合, 具体的に誰に作業を割り当てるのかについては決定していない. そのため, 個々の作業者の作業者別スケジュールを求めることはできない. 実際には, 勤務中の作業員は, 個々に与えられた詳細スケジュールに従って作業に従事するため, 作業者別のスケジュールの作成は重要である. そこで, 個々の作業者を区別し, 各作業者の詳細スケジュールを求める作業者別スケジューリングモデルを提案する.

また, 使用するデータに合わせて作業者のモードを手動で定義しているため, データが変わると新たにモードを定義し直す必要がある. そのため, 作業者のモード設定を自動で行うようにプログラムを改良する.

3.2 作業者別のモード設定

江川・中西 [1] では, 同じ能力を持つ複数の作業者をグループ化し, 作業のモード設定をグループをベースに行う. 図 2 は, 加熱作業のみを可能とするグループ (以下, 加熱グループ) の作業者 2 人を α_1, α_2 とし, 炊飯作業のみを可能とするグループ (以下, 炊飯グループ) の作業者 2 人を β_1, β_2 とし, 両方の作業を可能とするグループ (以下, 万能グループ) の作業者は γ とした場合の作業者グループの例を示す.

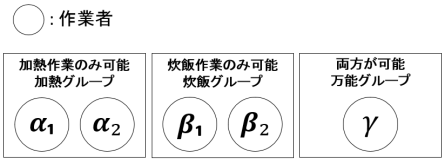

図 2 作業者の能力別グループの例

加熱作業を 2 人で行う場合に, グループをベースにしたモードの設定では, 図 3 のように, グループへ作業を割り当てるため, 具体的にグループの誰に作業を割り当てるのかについては決定できない. 一方, 本研究で提案する作業者をベースにしたモードの設定では, 図 4 のように, 作業者 1 人 1 人に作業を割り当てるため, モードが選択されると実際に誰が作業を行うのかを決定できる.

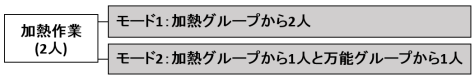

図 3 グループをベースにしたモードの設定

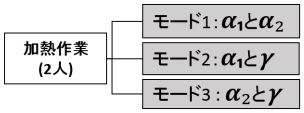

図 4 作業者をベースにしたモードの設定

3.3 モード設定の自動化

3.3.1 itertools モジュールの combinations 関数

itertools はイテレータを構築するためのモジュールである. イテレータ（iterator）とは, 要素を反復して取り出すことのできる仕組みのことである. combinations 関数は itertools モジュールの関数であり, combinations(s,r): s 内の要素の長さ r の組み合わせを全て生成する [2].

文字列 s ='ABC' に対して, combinations(s,2) を実行すると,

$$\text{combinations}(s, 2) = \{(\text{A}, \text{B}), (\text{A}, \text{C}), (\text{B}, \text{C})\}$$

を得る.

3.3.2 itertools によるモード設定の自動化

例えば図 4 のように, 3 つのモードを設定する際に, プログラム上で combinations 関数を用いることで, 加熱作業を行う作業者のモードを全て自動で求めることができる.

作業者を新たに作業可能な部門ごとにグループ分けをする（図 5）. 加熱作業を 2 人で行う場合, 加熱可能グループから 2 人選択する全ての組み合わせを combinations によって求める.

$$t = (\alpha_1, \alpha_2, \gamma)$$
$$\text{combinations}(t, 2) = (\alpha_1, \alpha_2), (\alpha_1, \gamma), (\alpha_2, \gamma) \quad (1)$$

(1) で導き出した組み合わせをすべてモードとして設定する. このようにモード設定を自動化することにより, 使用するデータが変わり, 作業者の人数が変わったり行う作業が変わっても, スケジューリングが可能となる.

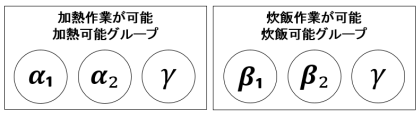

図 5 作業者の部門別グループの例

3.4 作業者の勤務時間を考慮したスケジューリング

作業者にはそれぞれ勤務時間があり, その時間を考慮することでより現実的なモデルに近づく. プログラム上で作業者を定義する際に, 勤務時間を制約条件として追加をすることで, 勤務時間を考慮したスケジューリングが可能となる. 過去の研究では, 松崎 [3] が, 出勤ローテーションスケジューリング問題を 0-1 整数計画問題として定式化をしている. 本研究では, 各作業者の勤務時間は所与とし, 作業者の能力を考慮した具体的な作業割り当てスケジューリングを行う.

3.5 出力結果

図 2 の 5 人の作業者で, 加熱作業 5 時間 (2 人), 炊飯作業 7 時間 (2 人) を行う. ただし, α_1 は作業開始から 2 時間後に出勤する. 最大完了時刻最小化を目的として求めた. スケジューリングの結果を図 6 に示す. 結果は, OptSeq が出力するものである. Act[1] は加熱作業, Act[2] は炊飯作業を表す. 図右上部のタイムテーブルは, 各作業を行う時間帯を示す. 作業者にはそれぞれ 2 行ずつタイムテーブルがある. 下の行は勤務時間を示し, 出勤していれば 1, 出勤していなければ 0 とする. 上の行は作業時間を示し, 作業していれば 1, 作業していなければ 0 とする.

4 炊飯のスケジューリングモデル

4.1 基本情報

炊飯部門では, 炊飯用の特殊な生産ラインを用いて炊飯作業を行っている. 図 7 は生産ラインを図示したものである.

activity	mode	duration	1	2	3	4	5	6	7
Act[1]	Mode[0_0]	5			=	=	=	=	=
Act[2]	Mode[1_0]	7	=	=	=	=	=	=	=
resource usage/capacity									
$\alpha 1$			0	0	1	1	1	1	1
			0	0	1	1	1	1	1
$\alpha 2$			0	0	1	1	1	1	1
			1	1	1	1	1	1	1
$\beta 1$			1	1	1	1	1	1	1
			1	1	1	1	1	1	1
$\beta 2$			1	1	1	1	1	1	1
			1	1	1	1	1	1	1
γ			0	0	1	1	1	1	1
			1	1	1	1	1	1	1

図 6 出力結果

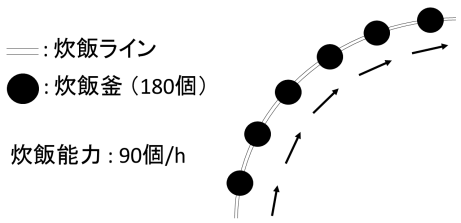

図 7 炊飯部門の生産ライン

炊飯ラインは円形状であり, ライン上には 180 個の炊飯釜 (以下, 釜) が並べられ釜がライン上を動きながら炊飯を行う. ライン全体で 1 時間に炊飯できるのは 90 個である.

1 つの釜で炊飯できるご飯の最大量は 6.5kg であり, 需要量に応じて, 4kg, 3kg でも炊飯できるが, これ以外の量では炊飯できない. 以下では, この 3 種類の炊き方をそれぞれ, 6.5 釜, 4 釜, 3 釜と呼ぶことにする.

炊飯部門では毎日の弁当やおにぎりの受注状況によって, その日に炊飯するご飯の種類, 需要量, 炊飯を完了する時刻などが決められる. 状況によっては, ご飯の種類が同じでも, 需要量や炊飯を完了しなければならない時刻が異なる作業が存在する. これらの条件を満たした上で最も時間の無駄のない最適な炊飯スケジュールを求める.

4.2 調整釜とサニテーション

工場では, 白米だけでなく, わかめご飯や菜飯など, 複数の種類のご飯を炊飯する. 炊飯するご飯の種類を変えるときには, 水だけを入れた釜 (調整釜) をライン上に流すサニテーションという作業が必要となる. 調整釜の数はご飯の種類によって変わる. 図 8 に例を示す. 白飯を炊いた後には調整釜を 3 個入れてから次の種類のご飯を炊飯する. また, 混ぜご飯を炊いた後には調整釜を 6 個入れてから次の種類のご飯を炊飯する.

炊飯するご飯の種類によってはベースとなるご飯が同じものがある. 例として, わかめご飯は白飯にわかめを混ぜて作るため, ベースは白飯である. 通常, 白飯を炊いた後は調整釜を 3 個入れる必要があるが, 白飯, わかめご飯の順

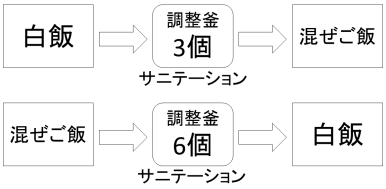

図 8 サニテーションの例

番で炊飯を行う場合は, 白飯を炊いた後に調整釜を 2 個入れ, わかめご飯を炊飯することができる. このようにベースが同じご飯を連続して炊くことにより, サニテーションに必要な調整釜の個数を減らすことができる. ただし, わかめご飯, 白飯の順番で炊飯を行う場合は, 調整釜を減らすことはできない.

また, 通常炊きあがったご飯を番重に回収する作業は機械が行っているが, この作業を作業員が行うことで調整釜を 3 個にすることができる. このことを直受けを呼ぶ. 直受け作業を担当できる作業員がいる時間は直受けを行うことによってさらに調整釜を減らすことが可能である.

このように, 炊飯する順番および直受けを行うかどうかによって調整釜の数が変わる. 調整釜数が多いと全体の作業時間が長くなるため, 調整釜数がなるべく少なくなるような炊飯スケジュールを求めることが重要である.

4.3 使用する釜の組み合わせ

炊飯するのに必要な釜の数を求めるため, 2 つの条件を満たした釜の組み合わせを導き出す.

1 つ目の条件は, 余りを最小にすることである. 廃棄量を減らすために, 3 種類の炊き方を組み合わせて, ご飯の需要量に対して余りが最も少なくなるようにしなければならない. 2 つ目の条件は, 使用する釜の数を最小にすることである. 余りを最小にした上で, 炊飯時間を短くするためにできるだけ使用する釜の個数を少なくしなければならない.

問題を定式化するにあたり, 以下の記号を定義する.

a: 必要なご飯の量
x_1: 6.5 釜の数
x_2: 4 釜の数
x_3: 3 釜の数

はじめに, 炊飯量の余りを最小化する釜の組み合わせを求める. 余りを最小化する釜の組み合わせを求める問題は以下のように定式化できる.

$$\text{min.} \quad 6.5x_1 + 4x_2 + 3x_3 \tag{1}$$

$$\text{s.t.} \quad 6.5x_1 + 4x_2 + 3x_3 \geq a \tag{2}$$

$$x_1 \in Z_+, x_2 \in Z_+, x_3 \in Z_+$$

$$Z_+ = \{0, 1, 2, 3 \cdots\} \tag{3}$$

(1) は炊飯する量を最小にする目的を表している. (2) は必要とするご飯の量以上炊飯する制約を表している. (3) は x_i の非負整数制約である.

次に, 余りが最小になる釜の組み合わせの中から, 釜の

数が最小になるものを求める．

$$\text{min.} \quad x_1 + x_2 + x_3 \tag{4}$$

$$\text{s.t.} \quad 6.5x_1 + 4x_2 + 3x_3 = y \tag{5}$$

$$x_1 \in Z_+, x_2 \in Z_+, x_3 \in Z_+$$

$$Z_+ = \{0, 1, 2, 3 \cdots\} \tag{6}$$

ここで，y は (1) の最適値である．(4) は使用する釜の数を最小にする目的を表している．(5) は余りが最も少なくなる量を炊飯することを表している．(6) は x_i の非負整数制約である．

これらの定式化によって求めた最適な釜の組み合わせから必要な釜の個数が分かる．最適解は Gurobi Optimizer8.0 を用いて求めた．

4.4 RCPSP としてのモデル化

本研究では炊飯ラインと直受けを行う作業員を資源とする．各ご飯を炊く作業と，作業と作業の間に行うサニテーションを定義する．白飯，酢飯，おに白はベースのご飯であり，ベースが同じご飯を連続して炊く場合は間の調整釜の個数は 2 個になる．また，作業員を使用して直受けを行う場合は調整釜の個数は 3 個になる．図 9 のようにサニテーションに，使用資源と調整釜の個数が異なる複数のモードを設定することで，調整釜を減らす方法を実現している．また，炊飯を完了しなければならない時刻が異なる同じ種

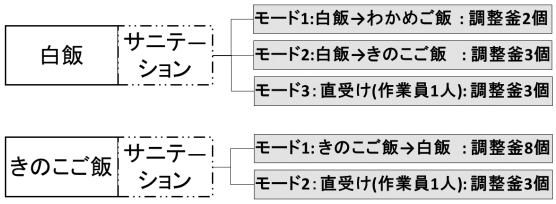

図 9　サニテーションのモード設定

類のご飯を連続して炊く場合は調整釜が 0 個になるように定義している．この複数のモードの中から目的に合った最適なモードが自動的に選択される．スケジューリングの目的として，最大完了時刻最小化または納期遅れ最小化を選択することができる．すべての作業が納期内に完了できる場合は最短で完了できるように，どうしても納期遅れが出てしまう場合は納期遅れの合計時間を最小にするように，というように状況に応じて最適なスケジュールを求めることができる．

プログラムでは手動で作業に関するデータを入力する．csv ファイルに作業番号，作業名，需要量，納期，サニテーションに必要な調整釜数，先行作業とベースのご飯が同じ作業の一覧を入力する．次に，作業全体の開始時刻を入力するとプログラムが実行される．

4.5 結果の出力

使用したデータは，工場で使用している実データに基づいて作成した 25 個の作業データである．プログラムを実行すると，1 分程度で各作業の開始時刻と終了時刻が csv ファイルに出力される．また，作業全体のガントチャート，サニテーションに設定されたモードの中から選択されたモード，各時間における未使用資源数も別の csv ファイルに出力される．簡易的なガントチャートも出力されるが，可視化モジュールの matplotlib を用いることで，より見やすいガントチャートを出力することができる (図 10)．

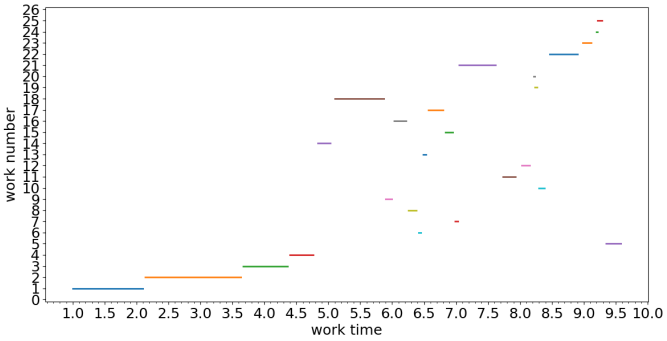

図 10　matplotlib を用いて作成したガントチャート

5　おわりに

作業者別スケジューリングモデルでは，作業者それぞれの能力を考慮した上でより短い生産時間で作業を行う生産スケジュールを求めるプログラムを作成した．作業者それぞれの能力を考慮することで，作業者 1 人 1 人のスケジュールを求めることができる．また，モード設定を自動化することにより，使用するデータが変わってもスケジュールを求めることができる．加えて，作業者の勤務時間を考慮した上でのスケジューリングが可能となった．

炊飯スケジューリングモデルでは，各ご飯の需要量，納期を考慮した上でより少ない釜数で炊飯を行う生産スケジュールを求めるプログラムを作成した．最適な釜数，スケジュールを求めることで，廃棄量を減らし，効率の良いスケジュールで生産することができる．このモデルでは作業員は直受けを行う場合でしか使用していない．直受け以外にも作業員が行う作業を追加し，実際の作業員の勤務時間を考慮することでより現実的なモデルに近づけることができると考えられる．

参考文献

[1] 江川奈那・中西加奈:『食品工場における生産スケジュールの最適化』．南山大学 理工学部 2017 年度 卒業論文, 2018.

[2] 久保幹雄・小林和博・斉藤努・並木誠・橋本英樹:『Python 言語によるビジネスアナリティクス実務家のための最適化・統計解析・機械学習』．近代科学社, 2016.

[3] 松崎佳人:『スケジューリングシステム開発についての実際的研究』．南山大学大学院 理工学研究科 2017 年度 修士論文, 2018.