RPA におけるスケジューリング問題

M2021SS007 吉田水紀
指導教員：鈴木敦夫

1　はじめに

技術の進歩により IT を活用すれば様々な作業の効率化が可能となっている．そこで多くの企業では IT を活用し手作業で行う業務の時間を短縮することを目指している．近年 DX 推進が社会の大きなテーマとなっているが，紙で行っていたことが電子化されることで，IT で業務自体を自動化することも可能となった．このような業務を自動化するための IT の利用にオペレーションズ・リサーチ（OR）を組み合わせることができれば，より効率的な業務改善が可能となり，さらなる DX 推進につながると考える．OR とは科学的に現実の様々な問題の最適解を見つける手法あり，近年様々なシステムにこの手法が用いられ最適化が行われている．

我々は企業で利用されている RPA という技術を，より効率的に利用できるよう OR の手法を用いて研究する．RPA とは，Robotic Process Automation の略であり，近年多くの企業で導入されている業務を効率的に行うための技術である．コンピュータ上で繰り返し行われる人間の判断が介在しない定型業務を一度だけ記録し，繰り返し作業自体はロボットがその記録に基づいて行うことで，業務が容易に自動化されるという技術である．RPA を用いれば，人間が手作業で繰り返し行っていた作業を自動化できることから，少ない労働人口でも生産性を維持することが可能となる．そのため，RPA は近年の働き方改革において業務効率化の有効な手段の一つとして注目されている．実際，すでに多くの企業で業務効率化のために RPA を導入しているが，対象となる定型業務の手順については，その最適化が十分に行われていないのが現状である．RPA を適用する作業そのものが効率的でなければ，業務効率が全体的に最適化されたとは言えないであろう．作業の手順最適化を行うことで RPA の効率が向上したら，業務全体の最適化をすることができる．本研究では，既存の手順のまま RPA により作業を自動化するのではなく，自動化する前に作業の手順の最適化も行うことで，RPA の効率の向上を目指す．

RPA に関連したスケジューリング問題における過去の研究 [3] では，金融業界での作業を例として，RPA を適用する際に二段階の最適化を行っている．作業の順番については従来のまま，第一段階で全てのトランザクションを完了するのに必要なロボット数の最適化を行い，第二段階ではトランザクションを第一段階で出したロボット数の制限の中で割り当てている．本研究ではそれとは異なるモデルで作業手順の最適化を行う．RPA の手順の最適化問題を解くことによって RPA の効率を向上させることが期待できる．手順の最適化も行う RPA を用いることで，作業時間がさらに減少すれば， それは生産性向上に貢献することになるので，その開発を目的とする．

2　企業における業務効率化問題

2.1　現場の RPA の利用ついて

現在企業では手作業で行っていたいくつもの定型業務を RPA により自動化することで業務効率化を図る取り組みを進めている．しかし RPA により自動化したい業務そのものの手順の最適化は一般には行われておらず，業務を RPA 化する際には従来のまま逐次的に自動化している．また企業で用意することができるロボットの数は企業ごとに制限がある．以上のことから，作業の流れを可視化し整理することで，どの作業を並行して行うことができるか確認しロボット数が許す限り業務を並行して行わせることでどれほどの時間を短縮できるか検証する．作業の前後関係を見て並行してできる作業は並行して行うことで，RPA の効率を向上させ所要時間を短縮できると考える．

2.2　問題解決の方法

手順の最適化はスケジューリングの手法である PERT/CPM を用いるのが適切であると考える．PERT/CPM とは，Program Evaluation and Review Technique / Critical Path Method の略で，大規模なプロジェクトを管理するために開発された手法である [2]．PERT/CPM では，複雑な作業工程の流れをアローダイアグラム呼ばれる作業を矢印で結んだ図によりを明確に表示する．また各作業の所要時間をアローダイアグラムに割り当てた上で，作業工程を表す各経路の最短時間や最長時間を見つける．特に，所要時間が最大となる経路のことをクリティカルパスと呼ぶ．全工程を進める上でクリティカルパス内の作業が遅れると作業終了時刻も遅れるため，クリティカルパスが表す作業工程は，スケジュールに致命的な影響を及ぼす．また，本研究が対象とする工程では，各作業でアクセスするデータベースのようなリソースが存在するので，既存の PERT/CPM の手法にリソースについての制約を加えなくてはならない．

3　RPA におけるスケジューリング問題の定式化

定式化では，RPA の所要時間の短縮のために全作業の終了時刻を最小化することを目的とする．ロボット 1 つが処理できる作業は同時に 1 つまでである．また各作業には複数の先行作業があり，それらの処理が終了しないとその作業を開始することはできない．さらに今回は通常の PERT の手法に加えて，作業が重複しているときは同じリソースに同時にアクセスすることができないというリソースの制約を課す．これらを定式化した整数計画問題をプログラムで解くことで，制約を満たす作業手順を求める．定式化では，全作業を先行作業とするダミー作業 $m+1$ を導入し，この作業の完了が全作業終了となる．

3.1 記号の定義

定数

I: 作業の集合 $I = \{1, \cdots, m+1\}$
d_i: 作業 i の所要時間
P_i: 作業 i の直前の作業の集合
R: リソースの集合 $(j \in R)$
n ：ロボットの数

$$r_{ij} = \begin{cases} 1: & \text{作業 } i \text{ がリソース } j \text{ を利用するとき} \\ 0: & \text{その他} \end{cases}$$

変数

x_i ：作業 i の開始時刻

$$s^1_{ii'} = \begin{cases} 1: & \text{作業 } i \text{ の開始時刻が,} \\ & \text{作業 } i' \text{ の終了時刻より早いとき} \\ 0: & \text{その他} \end{cases}$$

$$s^2_{ii'} = \begin{cases} 1: & \text{作業 } i' \text{ の開始時刻が,} \\ & \text{作業 } i \text{ の終了時刻より早いとき} \\ 0: & \text{その他} \end{cases}$$

$$t^1_{ii'} = \begin{cases} 1: & \text{作業 } i' \text{ の終了時刻が,} \\ & \text{作業 } i \text{ の開始時刻より早いとき} \\ 0: & \text{その他} \end{cases}$$

$$t^2_{ii'} = \begin{cases} 1: & \text{作業 } i \text{ の終了時刻が,} \\ & \text{作業 } i' \text{ の開始時刻より早いとき} \\ 0: & \text{その他} \end{cases}$$

$$u_{ii'} = \begin{cases} 1: & \text{作業 } i \text{ と作業 } i' \text{ の作業時間が} \\ & \text{重なっているとき} \\ 0: & \text{その他} \end{cases}$$

3.2 定式化

目的関数

$$\min. \quad x_{m+1} \tag{1}$$

制約条件

$$x_i \geq \max_{i' \in P_i} \{x_{i'} + d_{i'}\}, \ (i \in I \backslash \{1\}) \tag{2}$$
$$x_i - x_{i'} - d_{i'} + M s^1_{ii'} \geq 0, \ (i, i' \in I, i \neq i') \tag{3}$$
$$x_{i'} - x_i - d_i + M s^2_{ii'} \geq 0, \ (i, i' \in I, i \neq i') \tag{4}$$
$$x_{i'} + d_{i'} - x_i + M t^1_{ii'} \geq 0, \ (i, i' \in I, i \neq i') \tag{5}$$
$$x_i + d_i - x_{i'} + M t^2_{ii'} \geq 0, \ (i, i' \in I, i \neq i') \tag{6}$$
$$s^1_{ii'} + t^1_{ii'} = 1, \ (i, i' \in I, i \neq i') \tag{7}$$
$$s^2_{ii'} + t^2_{ii'} = 1, \ (i, i' \in I, i \neq i') \tag{8}$$
$$u_{ii'} = (s^1_{ii'} + s^2_{ii'}) - 1, \ (i, i' \in I, i \neq i') \tag{9}$$
$$\sum_{i' \in I, j \in R, i \neq i', r_{ij} = r_{i'j} = 1} u_{ii'} \leq 0, \ (i \in I) \tag{10}$$
$$\sum_{i' \in I, i \neq i'} u_{ii'} \leq n - 1, \ (i \in I) \tag{11}$$
$$x_i \geq 0, \ (i \in I) \tag{12}$$
$$s^1_{ii'}, s^2_{ii'}, t^1_{ii'}, t^2_{ii'}, u_{ii'} \in \{0, 1\}, \ (i, i' \in I) \tag{13}$$

ここでは M は大きな数である．

各式の説明

(1) 終了時刻を最小にすることを目的とした目的関数
(2)作業 i の開始時刻が作業 i のいずれの先行作業の終了時刻より遅い
(3)作業 i の開始時刻が作業 i'の終了時刻より早いかを判別
(4)作業 i'の開始時刻が作業 i の終了時刻より早いかを判別
(5)作業 i'の終了時刻が作業 i の開始時刻より早いかを判別
(6)作業 i の終了時刻が作業 i'の開始時刻より早いかを判別
(7)$s^1_{ii'}$と $t^1_{ii'}$は片方のみ 1 となる制約
(8)$s^2_{ii'}$と $t^2_{ii'}$は片方のみ 1 となる制約
(9)作業が重なっているかを確認する制約
(10) 作業が重なっているときに同じリソースにアクセスすることができない制約
(11) ロボット数の制約
(12)x の範囲
(13) バイナリ制約

図 1 は制約式の $s^1_{ii'}, s^2_{ii'}, t^1_{ii'}, t^2_{ii'}$ についての図解である．作業が同時に開始または終了する部分ははいずれも $s^1_{ii'} = 0, s^2_{ii'} = 0, t^1_{ii'} = 0, t^2_{ii'} = 0$ となる．作業同士が重複しているときは $s^1_{ii'} = 1, s^2_{ii'} = 1, t^1_{ii'} = 0, t^2_{ii'} = 0$ となる．

4 小規模な例についての分析

定式化した問題を評価するために，簡単な人工的な例を多数作成し，作業数や先行作業数，リソース数を変えながら問題を解き，どのような所要時間になるのか検証する．人工的なモデルを作成するために，作業の数，作業の所要時間，各作業の先行作業，各作業が必要とするリソースを出力するプログラムを Python で作成した．このとき，作業数や作業の所要時間の範囲，先行作業の数の範囲，リソースの種類などは任意の数に設定できる．それぞれの値は設定した範囲で乱数によりランダムに決定する．各作業が必要とするリソースとは，例えば作業がアクセスするデータベースのことである．作成したプログラムにより人工的な作業のモデルを自動で多数作ることができる．作成した例について，手順の最適化を行わない場合と，行なった場合で，総作業時間の短縮度合いがどの程度であるかを数値的に比較・検証する．

今回のモデルでは作業の所要時間 (1〜10)，各作業の先行作業，各作業が必要とするリソース (1〜3) をランダムに選び結果を出力した．先行作業は 1〜3 個の間でランダムに選ぶ．作業 1 は一番初めの作業のため先行作業は存在しない．また作業 11 は終了のためのダミーの作業であ

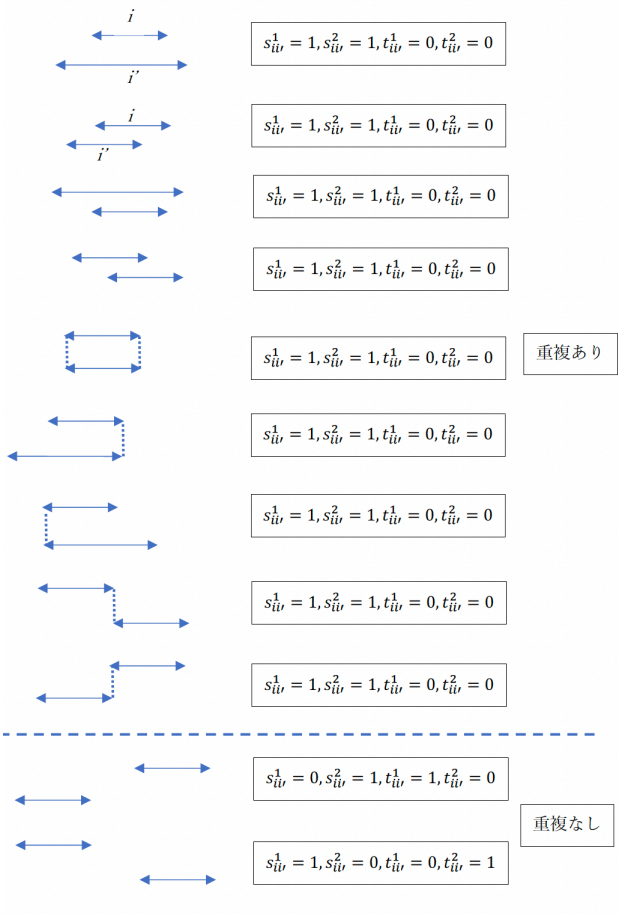

図 1　作業の重なりについて

り，作業 11 の先行作業の欄は他の作業の先行作業になっていない作業を抜き出している．

上記の表のトイモデルで逐次的に作業を行った場合にかかる時間は $\sum_{d=1}^{10} d_i$ より 62 である．ここでリソースの制約がなかった場合に PERT/CPM の手法を用いたときにかかる時間は図 2 より 46 である．図 2 は上記のトイモデルの作業の流れをリソースの制約を除いてにアローダイアグラムに表したものである．総所要時間を変えない範囲で開始時刻を選べる作業があるが，図 2 では S の括弧内の左の数字が最早開始時刻（その作業を最も早く開始できる時刻）で右の数字が最遅開始時刻（その作業を最も遅く開始できる時刻）であり，F の括弧内の左の数字が最早終了時刻（その作業を最も早く終了できる時刻）で右の数字が最遅終了時刻（その作業を最も遅く終了できる時刻）を表している．今回の例だと 1 → 2 → 3 → 4 → 7 → 8 → 9 → 11 のパスと，1 → 3 → 4 → 7 → 8 → 9 → 11 のパスと，1 → 2 → 9 → 11 のパスがクリティカルパスとなっており，このパス上の作業に遅れが生じると全体の終了時間が遅れる．

5　計算結果と考察

表 1 のトイモデルの例で 3 章で述べた最適化問題を解く．リソースの制約については，例えば表 1 のトイモデルだと作業 2 は作業 6 の先行作業ではないが，同じリソース 1 にアクセスするため並行して作業を実行することはできない．

表 1　トイモデル例

作業	所要時間	先行作業	リソース
1	10	[]	2
2	1	[1]	1
3	7	[1,2]	2
4	8	[3]	3
5	10	[1,2]	2
6	1	[5]	1
7	4	[4,6]	3
8	8	[6,7]	1
9	8	[2,8]	1
10	5	[2,5,6]	1
11		[9,10]	

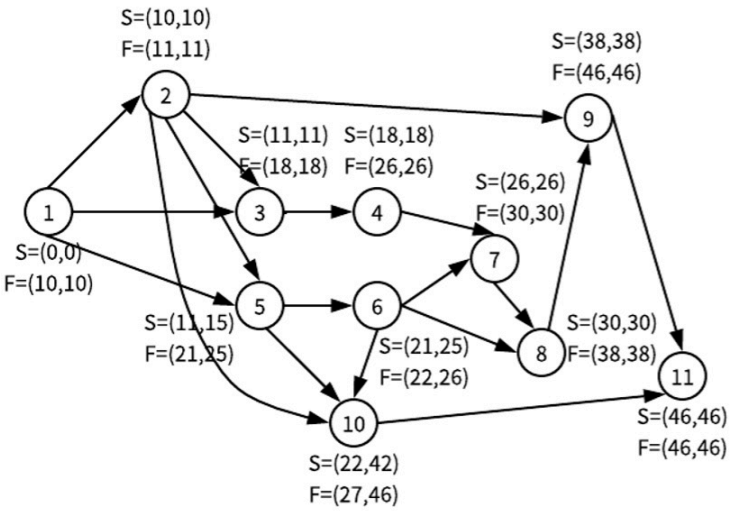

図 2　アローダイアグラム

今回の例の場合でロボットがいくつのときに時間がどれだけ短縮されるか計算する．ロボット数を増やしていき短縮時間が増えなくなる部分を探したところ，以下の表 2 のようになった．このとき $M = 100$ としており，計算時間の上限を 600 秒としている．作業数の 10 と同じ数のロボットを用意しても 19.35%の短縮であるから，これ以上の短縮は望めないと考えられる．今回のモデルではロボットが 2 つ以上あれば最大の短縮率となる計算結果となった．ロボット数が 2 つ以上のときを図に表したものが図 3 である．利用するリソースごとに色分けしており，オレンジがリソース 1 を利用する作業，青がリソース 2 を使用する作業，緑がリソース 3 を利用する作業である．同じリソースを利用する作業同士は重なっておらず，正しい結果となったと考えられる．図からも重なっている作業数は最大 2 つであるため，必要なロボット数の最大が 2 であることもわかる．

今回のトイモデルでは，リソースの制約を考えない場

表 2　実行結果

時間 \ ロボット数	1	2	3	4	5	6	7	8	9	10
PERT を用いた時間	62	50	50	50	50	50	50	50	50	50
逐次的に行った時間	62	62	62	62	62	62	62	62	62	62
短縮された時間	0.00%	19.35%	19.35%	19.35%	19.35%	19.35%	19.35%	19.35%	19.35%	19.35%

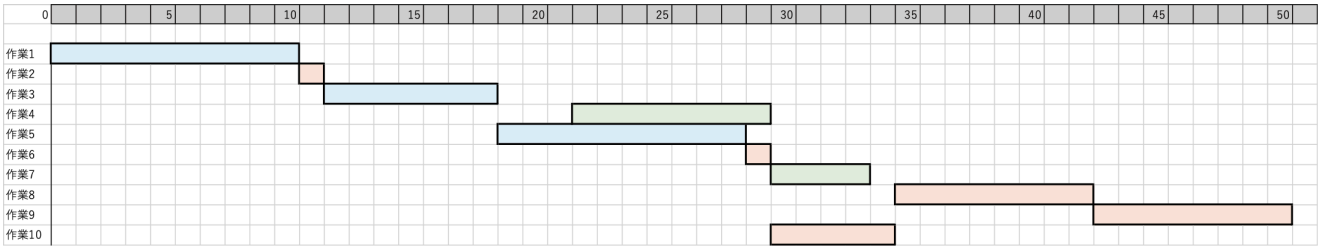

図 3　計算結果の作業の流れ

合に PERT の手法を用いることで作業時間が 46 となった．リソースの制約があることで最小でも 50 となったため，リソースの制約があると短縮できる時間により厳しい限界があることがわかる（表 3）．しかし逐次的に作業を行うよりも，手順を最適化した方が所要時間は短くなることがわかった．

次に簡単のため仮想的な例を乱数を用いて多数作成し，先行作業の数，ロボット数などの条件により所要時間がどのように変化するかを調べる．これらの例について PERT の手法を用いた場合と従来通り逐次的の場合を比較して，RPA の総作業時間の短縮率がどの程度であるかを数値的に比較・検証する．結果の一例でロボット数が 100・先行作業数が 1〜3・リソース数が 3 の時に，各作業数で 10 回ずつ計算し，標準偏差を求めたものが図 4 である．今回の例では作業数 100 が一番多い作業数のため，ロボット数も 100 で計算している．ロボット数が作業数より多いということは，作業に前後関係がなければ各作業に 1 つずつロボットを使用することができる．作業数が多いほど並行して行うことができる作業も多いため短縮された時間の割合も大きくなっていると考えられ，最大で約 60 %短縮できる結果となった．ここでは計算時間の上限を全て 200 秒としている．計算機環境は，CPU：Apple M1，RAM：16.00GB，OS：Mac OS Big Sur，最適化ソフトウェア：Gurobi9.5.2 である．

表 3　総作業時間比較

逐次的	リソースなし PERT	リソースあり PERT
62	46	50

6　おわりに

RPA を利用するにあたって，自動化したい作業手順の最適化を行うことでより効率的な RPA の利用を目指すことは，労働人口が減少している中で今後必要になってくると考えられる．計算結果から，RPA によって自動化したい作業はリソースの制約がある場合でも，逐次的に行

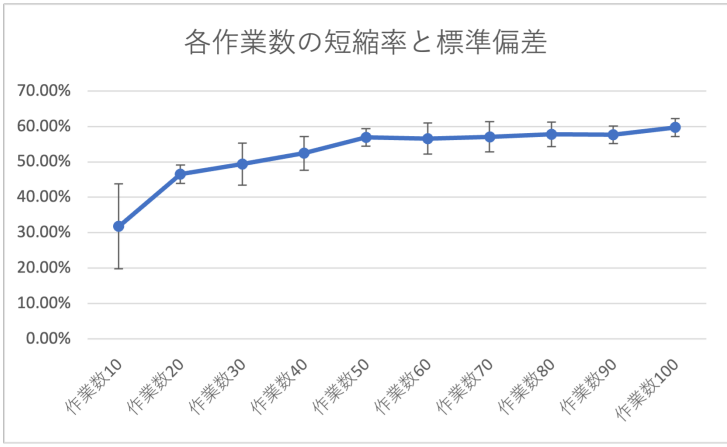

図 4　各作業数で 10 回ずつ計算した時の各作業数の標準偏差

うより PERT の手法を応用し，最適化することで時間を短縮することができるとわかった．企業によって用意できるロボットの数は変わってくるため，それに合わせて RPA 化前に手順を最適化することができたら，制限のある中でより作業時間を短縮することができ，より効率的な業務につながると考えられる．

参考文献

[1] 大野勝久，玉置光司，石垣智徳，伊藤崇博：『EXCEL による経営科学』．コロナ社，東京，2005.

[2] 関根智明：『PERT・CPM』．日科技連出版社，東京，1965.

[3] Sara Seguin and Imene Benkalai：Robotic Process Automation (RPA) Using an Integer Liner Programming Formulation - CYBERNETICS AND SYSTEMS: AN INTERNATIONAL JOURNAL, VOL51, NO.4, pp357-369, 2020.

[4] Christian Langmann and Julia Kokina：Robotic Process Automation (RPA) in accounting - 2021