修士卒業論文

遺伝的アルゴリズムと深層強化学習に基づく ジョブスケジュール最適化問題 ―PCB加工プロセスをモデルとして―

Job Scheduling Optimization Based on Genetic Algorithms and Deep Reinforcement Learning
―A Model of the PCB Manufacturing Process―

関西学院大学大学院
総合政策研究科
汪 永豪
2026年1月

要旨

近年、市場グローバル化が進む中、製品に対する市場ニーズは多様化しており、顧客ごとに仕様が異なる多品種少量生産方式への対応が重要となっている。本研究では、プリント基板 (PCB) 製造工場を対象に、同一加工機械を複数回使用する生産プロセスをモデル化した上で、スケジューリング最適化に関する検討を行った。静的な環境においては、遺伝的アルゴリズム (GA) および深層強化学習に基づく Deep Q-Network(DQN) を用いて最適化を行い，両手法の性能比較を実施した。さらに、機械故障等による環境変動が発生する動的な環境で高速にスケジュールを生成可能な DQN により, 適応性能の向上を図った。これらから静的および動的な生産環境におけるスケジューリングを検討する。

キーワード：インダストリーエンジニアリング、オペレーション・リサーチ、ジョブスケジューリング最適化問題、遺伝的アルゴリズム、深層強化学習

Abstract

In recent years, as market globalization has advanced, market demands for products have become increasingly diversified, making it essential to respond to high-mix, low-volume production systems in which specifications differ for each customer. This study focuses on a printed circuit board (PCB) manufacturing factory and models a production process in which identical processing machines are used multiple times. Based on this model, scheduling optimization is investigated. Under a static production environment, this study applies a genetic algorithm (GA) and a Deep Q-Network (DQN) based on deep reinforcement learning to perform scheduling optimization, and conducts a comparative evaluation of the performance of these two methods. Furthermore, under a dynamic production environment where environmental changes such as machine failures occur, this study aims to improve adaptability by utilizing DQN, which can generate schedules rapidly in response to such changes. Through these approaches, this study examines scheduling optimization for both static and dynamic production environments.

Keywords : Industry Engineering, Operations Research, Job Shop Scheduling Optimization Problem, Genetic Algorithm, Deep Reinforcement Learning

目 次

遺伝的アルゴリズムと深層強化学習に基づく ジョブスケジュール最適化問題 ―PCB加工プロセスをモデルとして―

Job Scheduling Optimization Based on Genetic Algorithms and Deep Reinforcement Learning ―A Model of the PCB Manufacturing Process―

汪 永豪 *1　　山田 孝子 *2

Wang Yonghao　　Takako Yamada

*1関西学院大学大学院総合政策研究科

Graduate School of Policy Studies, Kwansei Gakuin University

*2関西学院大学総合政策学部メディア情報学科

Department of Applied Informatics, School of Policy Studies, Kwansei Gakuin University

1. はじめに

世界経済のグローバル化に伴い、製造業企業は他社より安い価格で注文を受ける必要がある. いかにして低コストな生産システムと高品質な品質管理を両立させ，安定的に維持するかは、製造業が必ず直面すると言って過言ではない難しい問題である. 昨今ではこの両立にさらに安全管理を含むワークライフバランスを考えた労務管理が加わり，生産現場では達成するべき課題が複雑化している．コスト削減のため、多くな企業は人間作業員の代わりに、製造工程のロボットを採用し，工場内物流には複積載搬送ロボット (AGV) の導入をすすめている．AGV の導入により生産機械間の半製品や仕掛品の運搬は自動化された. 生産計画には, スケジューリングだけではなく,AGV の搬送計画も含まれ, 経路や運搬すべき半製品の決定までを生産計画に含むことになった．本研究では，以下のシステムを順次作成し，生産管理計画システムとして実装し，最適解を求める計算機実験を実施，システムの性能を検証した.

本研究でこれまで行ったことをまとめる

1. 実験用ジョブセットとして，製品種別ごとの加工機械の利用順番，利用時間の組み合わせデータの自動生成システムの作成
2. 生成されたデータを用いた生産機械のスケジューリングを総当たり法で厳密解を求める最短生産スケジュール算出システムの作成
3. 生産スケジュールの可視化システムの作成

連絡先: 関西学院大学総合政策研究科, 兵庫県三田市学園上ヶ原, ili53007@kwansei.ac.jp

4. 遺伝的アルゴリズムをベースとする，緩和手法による許容解算出と解の改善システムの作成
5. 生産機械に加え複数台 AGV を導入した生産スケジュールの最適化システムの作成
6. 求められた生産スケジュールに基づく生産機械の遊休時間活用する余剰生産品提案システムの作成

以上を行ったうえで、本研究では機械と AGV 両方同時スケジューリングを求め、現有な仕事を最短時間で終了する許容解を導出し, その許容解のもとで, 機械の遊休時間を活用する余剰製品を提案するシステムの開発に取り組む．このような問題は、JSSP(Job Shop Scheduling Problem) 問題と言います。

2. 先行研究

グローバル経済の進展や企業のコスト削減要求に伴い、生産現場では柔軟で効率的な生産体制の構築が求められている。その中で、需要の多様化に対応するために、少量多品種生産が一般的となり、このような生産形態に適したスケジューリングの最適化が重要な課題となっている。特に、ジョブショップスケジューリング問題 (Job Shop Scheduling Problem: JSSP) は、生産システムにおける代表的な最適化課題として広く研究されている。JSSP は、複数のジョブを限られた機械に割り当て、全体の処理時間や納期遅延を最小化することを目的とする問題である。また、各ジョブには処理順序が定められており、この順序を厳密に守る必要がある点が特徴である。この問題は NP 困難であり、最適解を求めることは計算量的に非常に困難である。そのため、これまで多くの研究では、遺伝的アルゴリズム (Genetic Algorithm: GA) や焼きなまし法 (Simulated Annealing: SA) などのヒューリスティック手法を用いて、効率的に近似解を求めるアプローチが提案されてきた。しかし、実際の製造現場では、機械の故障や新規ジョブの追加、処理時間の変動など、動的な要素が頻繁に発生する。そのような動的環境下では、あらかじめ定められたルールに基づくヒューリスティック手法では十分に対応できない場合がある。この問題を解決するため、近年では強化学習 (Reinforcement Learning: RL) を用いて、環境変化に応じて自律的にスケジューリング方策を学習する手法が注目されている。静的な環境においては GA や SA などの伝統的アルゴリズムが一定の成果を上げている一方で、動的な生産環境では強化学習に基づくアプローチのほうが、より高い適応性と柔軟性を示すことが報告されている。このように、JSSP におけるスケジューリング最適化の研究は、従来の探索型手法から学習型手法へと発展しつつある。

実際に、近年の研究ではこの流れを反映した多様なアプローチが報告されている。K.T. Chung ら [1] は現在工業 5.0 の時代に向けて、機械学習、特に強化学習を用いて、CO(combinatorial optimization)

問題を解決しようについて、現在の研究進展を述べた。M. Xu ら [2] は遺伝的プログラミングと強化学習を統合的に整理・比較しは JSSP スケジュール最適化問題ににおいて果たす役割と近年の研究動向を明らかにした。C. Ngwu ら [3] は動的な JSSP 問題は強化学習や進化的ヒューリスティック、機械学習を用いた手法でリアルタイムな意思決定は可能であることを示しました。X.R. Shao ら [4] は小規模かつ短時間の JSSP 問題に対して、多層畳み込みニューラルネットワーク (ML-CNN) と反復局所探索 (ILS) を組み合わせ、ML-CNN で全体経路を学習し、ILS で最適な局所経路を探索する手法が提案した。L.B. Wang ら [5] は機械故障や再作業などの動的な環境での JSSP 問題につい深層強化学習 (DRL) を用いた動的スケジューリング手法が提案され、PPO により最適な方策を学習することで、リアルタイムな生産スケジューリングが可能であることが示された。S. Lee ら [6] は射出成形金型製造プロセスにおいて、Deep Q-Network を活用することで従来より総加重遅延を低減できる金型生産スケジューリングを最適化した。C. Pickardt ら [7] は半導体製造プロセスにおいて、遺伝的プログラミング (GP) と確率的離散事象シミュレーションを組み合わせることで自動的に従来のルールより性能優れたなルールを生成した。村山ら [8] は JSSP 問題に基づき、加工機械と複数積載型 AGV の同時スケジューリングを対象とした遺伝的アルゴリズム (GA) 手法が提案した。

これらの研究から、JSSP およびその拡張問題に対して、静的環境ではヒューリスティック手法、動的環境では強化学習や深層学習手法が有効であることが確認されている。

3. 生産管理問題のモデル

本研究では受注生産を行う工場の製造工程, 受注型ジョブショップスケジュール問題 (Job Shop Scheduler prob-lem:JSSP) を最適化問題として解く. したがってモデルとしてクライアント側から提供される注文書が最適化の対象となり，このようなデータは不可欠である.

ここでは実際のプリント基板 (PCB) 製造工場をモデルの対象とする製造工程としてとりあげる. 本来は使用する注文書データが実生産に基づくものであることが望ましい. しかし,PCB 製造に関する実データは, 企業の営業機密や顧客情報に直結するデータのためこうしたデータは入手困難で, 共同研究であっても外部に具体的な数値の利用は難しい.

そのため本研究では, 先行研究 [22] に記載されている PCB 製造工程を参考にして, 各加工機械に対して加工役割を割り当てた人工的なデータをまず生成する.

最適化問題として解くのは, このデータを用いた PCB 製造プロセスの生産スケジューリング問題である. PCB 製造工程では, 注文種類によって必要とされる加工時間、加工工程数、あるいは加

工順序が異なる. そこでこれらの差異を反映したデータを生成する.

以下で本論文の PCB 製造工程におけるモデル設定と符号の定義を紹介する。

表 1: 本研究に用いる記号の定義

記号	説明
O	工場が受領した注文書の集合
O_i	第 i 番に工場が受領した注文書
J_s	第 s 製品種別の製品 J
O_iJ_s	第 i 注文書に記載された種別番号 s のジョブ J
$lotJ_{i,s}$	第 i 注文書に記載された種別製品 s の注文ロット数
n_s	対象ジョブセットに記載された種別製品 s の合計注文個数
n_s	$10\sum_{i=1} lotJ_{i,s}$(個数＝ロット数 $\times 10$)
J_sF_q	第 s 種別製品の第 q 番目の加工工程
$m_{s,q}$	第 s 種別製品の第 q 工程に割り当てられる機械番号
M	使用可能な加工機械の集合
M_m	第 m 番加工機械
N_{ops}	ジョブセット内全種別製品の全加工工程の数
$t_{s,q}$	第 s 種別製品の第 q 工程に要する加工時間
$ft_{s,q}$	第 s 種別製品の第 q 工程の完了時刻
$ft_{m_{s,q}}$	機械 $m_{s,q}$ 上で直前に処理された工程の完了時刻
TJ_s	第 s 種別製品の加工終了時間
makespan	全製品種別における最終工程完了時刻の最大値
OM	各加工工程の機械割当を表す行列
$m_{s,q} = OM_{s,q}$	機械割当行列の要素定義
OT	各加工工程の加工時間を表す行列
$t_{s,q} = OT_{s,q}$	加工時間行列の要素定義
C	制約条件の集合

3.1 モデル設定

1. 注文票は様々な注文主からくる。
2. 各注文書には製品 J の種別が 1 種類以上が記載されている。
3. 注文書ごとに製品ごとロット数が記載されている。
4. 工場は、一定期間に様々な注文主からくる注文書 O_i をまとめ、これをジョブセットと呼ぶ。
5. 製品種別に応じて、使用する加工機械、加工時間とその加工機械の使用順は予め決められている. ただし加工時間はロット数に依存する.
6. 工場はジョブセットごとに同一出荷日を設定したい. この同一出荷日は早いほどよいので, 受注した注文書の加工機械の利用順番をスケジューリング最適化し最短総加工生産時間を求める.
7. 各機械はそれぞれ待機状態と稼働状態がある。

待機状態：機械は製品の加工作業していない、いつでも新たな加工工程を加工させる。

稼働状態：機械は製品の加工作業している、加工している作業が終わるまで、新たな加工工程を加工させない。

8. 機械は注文書に記載された製品種別で指定されたロット数ごとに加工する．加工にとりかかったロットの途中で中断や割り込みはない.
9. 機械のセットアップタイムは考慮しない。
10. 製品の加工間の搬送時間はこのモデルでは考慮しない。
11. 機械は特定の加工工程のみを担当できる．別加工工程の転用は不可能とする.
12. モデルはイベント駆動型である。

    イベント：加工工程の処理完了や機械の待機状態の発生など、スケジューリング状態が更新される契機となる事象を指す。
13. 製品ごとに全部の加工工程が終了したら、その製品の加工スケジューリングが終了する。
14. 全部の製品の加工スケジューリングが終了したら、ジョブセットのスケジューリングが終了する

3.2 問題設定

本研究は、JSSP を制約付き最適化問題として定式化し、目標関数としては:

$$\text{opt. } JSSP = \min\left\{\max_{s} TJ_s\right\} \tag{1}$$

ここで，TJ_s は第 s 種別製品の最終加工工程の完了時刻を表す。したがって, 本最適化問題は, ジョブセットに記載された全製品種別の最終加工完了時刻の最大値を求める. 様々なスケジューリング案の最終加工完了時刻の中で最小のものを本論文では $makespan$ と呼ぶ, すなわち, 最適化とは, この $makespan$ の最小化である.

制約条件としては：

$$\text{s.t.} \quad \text{C1:} \quad ft_{s,q} \geq ft_{s,q-1} + t_{s,q} \tag{2}$$

$$\text{s.t.} \quad \text{C2:} \quad ft_{s,q} \geq ft_{s',q'} + t_{s,q} \tag{3}$$

1. 制約条件 $C1$ は、同一製品種別 J_s における隣接する加工工程 F_{q-1} と F_q の加工順序制約を表しており、前工程が完了した後でなければ次工程を開始できないことを意味する。

2. 制約条件 C2 における $ft_{s',q'}$ は、加工工程 J_sF_q と同じ機械 $m_{s,q}$ を使用するが，製品種別が異なる別製品 $J_{s'}$ の加工工程 $J_{s'}F_{q'}$ の完了時刻を表す。すなわち，$ft_{s',q'}$ は，同一機械上で J_sF_q の直前に処理された工程の完了時刻に対応する。

3. 制約条件 $C2$ は、各加工機械は同時刻に一つの加工工程しか処理できないという機械リソース制約を表しており，ある工程が機械 M_m 上で処理中の場合，次の工程はその加工工程が終了するまで開始できない。

4. JSSP においては、加工工程と加工機械の対応関係を表す機械割当行列 OM と，各加工工程に要する加工時間を表す加工時間行列 OT が既知である。OM と OT に関することは次章に述べる。

各製品種別の加工順序が与えられた場合，各加工工程 J_sF_q の完了時刻は次の式に基づいて算出することができる。

$$ft_{s,q} = \max\left(ft_{s,q-1},\, ft_{m_{s,q}}\right) + t_{s,q} \tag{4}$$

ここで，$ft_{s,q}$ は第 s 種別製品の第 q 加工工程の完了時刻を表す。本式は，各加工工程の完了時刻が，同一製品における直前工程の完了時刻 $ft_{s,q-1}$ と，同一加工機械 $m_{s,q}$ 上で直前に処理された別製品工程の完了時刻 $ft_{m_{s,q}}$ のいずれか遅い方に，加工時間 $t_{s,q}$ を加えた値として決定されることを示している。

ジョブショップスケジューリングの目的は、各加工機械をできるだけ連続的に稼働させるとともに、各製品の加工工程を待ち時間なく加工することにより、総加工時間を最小化することである。したがって、JSSP は待機している加工機械に対して加工対象となるジョブを割り当てる逐次意思決定問題とみることができる. スケジューリングは、加工機械上で各加工工程の処理完了イベントにごとに次に行う加工工程をジョブセットから選択することで逐次的に決まる。同一製品内における加工工程の順序制約および、各加工工程が指定された加工機械で加工されなければならないという制約を考慮すると、待機中の加工機械に製品の加工工程を割り当てるには、現時点で処理可能な加工工程が存在するかどうかを判断する必要がある。処理可能な加工工程がジョブセットに複数存在する場合には、その中から一つを選択して次の加工を開始する。一方，現時点で処理可能な加

工工程が存在しない場合には，次の加工工程完了イベントが発生するまで待機する。この意思決定過程は次の図 1 に示す。

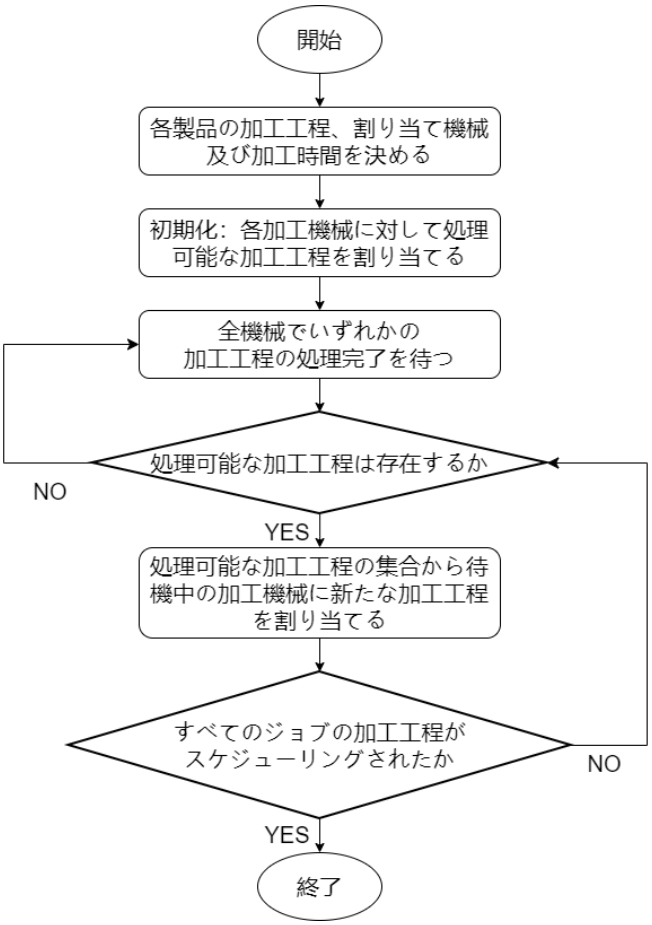


図 1: ジョブスケジューリング問題の逐次解を求める手続き

4. 実験用データ自動生成

本来であれば、行列 OM および OT に用いる機械加工順序データや加工時間データは、実工場から取得した実績データに基づくことが望ましい。しかしながら、個々の製品の加工順序や加工時間、注文書などの生産情報は企業の機密および顧客情報として保護される．これらは共同研究等であっても外部公表は困難である。そこで本研究では、文献 [22] に記載された多層 PCB の製造工程を基づく各製品種別ごとの加工工程と使用機械をもとに，注文書のばらつきやロット数を確率的にデータとして生成するし，ジョブセットを得る．本節ではその実験データの自動生成について述べる．

4.1 多層 PCB の製造工程

多層 PCB の製造工程は以下の通りとする．

1. 材料準備
2. 材料表面の汚れや油分を除去し
3. パターン転写（フォトリソグラフィ方式かダイレクト印刷方式のいずれか一方とする）
4. エッチングとレジスト除去（複数回加工する場合ある）
5. 積層（複数回加工する場合ある）
6. ドリル加工
7. スルーホールめっき
8. ソルダーレジスト塗布
9. 表面処理
10. 電気検査
11. 包装

ここまでの加工ができたら加工作業は終了し出荷可能となる. この加工工程に対して, それぞれの工程に用いる加工機械の種類と役割を表 2 に示す.

表 2: 加工機械および工程内容の定義

機械番号	工程内容
M_1	銅箔付きの絶縁板 1 を取る
M_2	銅箔付きの絶縁板 2 を取る
M_3	表面の汚れや油分を除去する
M_4	パターン転写（フォトリソグラフィ方式）
M_5	パターン転写（ダイレクト印刷方式）
M_6	エッチングおよびレジスト除去
M_7	積層
M_8	ドリル加工
M_9	スルーホールめっき
M_{10}	ソルダーレジスト塗布
M_{11}	表面処理
M_{12}	電気検査
M_{13}	包装

本研究の生成データのため、６種類の製品が生産可能な工場を仮定する。この６種類の製品の加工工程順序は以下のように設定する。

4.2 ジョブセットのデータ構成

加工工場は6種類の製品種別に生産が可能とする。6種類の製品種別ごとの加工工程順序を行列 OM の形式で表す．行は製品種別，列は各製品の加工工程順番に対応する．

$$OM = \begin{bmatrix} 1 & 3 & 4 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 3 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3 & 4 & 6 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 0 & 0 & 0 & 0 \\ 2 & 3 & 5 & 6 & 4 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 0 & 0 & 0 & 0 \\ 1 & 3 & 5 & 6 & 4 & 6 & 7 & 8 & 9 & 4 & 6 & 10 & 11 & 12 & 13 & 0 & 0 \\ 2 & 3 & 5 & 6 & 4 & 6 & 7 & 8 & 9 & 4 & 6 & 5 & 6 & 10 & 11 & 12 & 13 \end{bmatrix}$$

ここで $OM_{s,q} = \{1, 2, \ldots, 12, 13\}$ は製品 s の第 q 番目加工工程の加工機械は $OM_{s,q}$ の要素を意味する。$OM_{s,q} = 0$ の場合、製品 s の第 q 番目加工工程自体がないことを意味する。

エッチングとレジスト除去、積層、スルーホールめっき、この三つの加工操作は PCB をロット単位で一度に加工できる工程となる。したがってこの3つの工程の加工時間は定値とする．これ以外の加工工程ではその工程の加工時間は $n_s \times$ 単位時間 の積となる。次は各機械の単位加工時間もしくは定値加工時間を設定する。

表 3: 加工機械別加工時間パラメータの設定

加工時間パラメータ	設定値
MT_1	0.5（単位加工時間）
MT_2	0.5（単位加工時間）
MT_3	2（単位加工時間）
MT_4	8（単位加工時間）
MT_5	7（単位加工時間）
MT_6	250（基準加工時間）
MT_7	600（基準加工時間）
MT_8	3（単位加工時間）
MT_9	750（基準加工時間）
MT_{10}	5（単位加工時間）
MT_{11}	4（単位加工時間）
MT_{12}	3（単位加工時間）
MT_{13}	1（単位加工時間）

受注数およびロット数のばらつきを表現するために、受注数は平均 $\mu_{odder} = 10$、分散 $\sigma^2_{odder} = 2$ の正規分布に従う乱数で生成し、ロット数は平均 $\mu_{lot} = 3$、分散 $\sigma^2_{lot} = 1$ の正規分布に従う乱数と

して生成する。ただし負の値および 0 は現実的でないため除外し, 正の値のみを採用する。以上の手順で受注量とロットサイズにランダムネスを持つジョブセットを作る。

表 4: オーダー数およびロット数に関する確率分布パラメータ

パラメータ	説明
μ_{order}	総オーダー数の平均値
σ^2_{order}	総オーダー数の分散
μ_{lot}	各オーダーに注文された製品ロット数の平均値
σ^2_{lot}	各オーダーに注文された製品ロット数の分散

これにより、実際の生産現場に見られる受注量とロットサイズのランダムネスでデータとして生成する。

ジョブセットに含まれる各注文票には、最大 6 種類の製品種別が含むことができる. 製品種別は確率 P = 0.7 の二項分布に従い注文票に記載される．一つの注文票の製品種別に重複はない．注文票別の製品種別行列 $odder$ を構成する. この注文票ごとに製品種別ごとの総注文件数 n_s をロット数とする. 1 ロットあたりの基板数は 10 枚とし，加工時間は定数で決まる工程以外はロット単位で決まる．各加工工程の処理時間は、n_s とあらかじめ機械別に与えた単位加工時間または基準加工時間パラメータで決定する。例えば、ドリル加工や電気検査など加工時間が枚数に比例すると考えられる工程では、$n_s \times MT_m, m = \{1, 2, 3, 4, 5, 8, 10, 11, 12, 13\}$ で工程時間を設定し、エッチングや積層などバッチ処理的な工程では、製品枚数に依存しない基準時間 $MT_m, m = \{6, 7, 9\}$ を用いる。こうして得られた処理時間行列 OT を得る。

$$OT_{s,q} = \begin{cases} n_s * MT_m & m \in \{1, 2, 3, 4, 5, 8, 10, 11, 12, 13\} \\ MT_m & m \in \{6, 7, 9\} \end{cases} \tag{5}$$

まとめると、本研究の自動生成データは、

1. 多層プリント基板の代表的な加工工程列を反映した機械割当行列 OM
2. 受注数・ロット構成のばらつきと機械特性を考慮して決定される処理時間行列 OT

から構成される。

4.3 データ生成手続き

具体的なデータ生成手続きは Python で実装した. その手順は以下のとおりである.

手順 1: 乱数 seed $seed_{value}$ を設定し，受注数およびロット数の正規分布パラメータ，ならびに機械別単位加工時間・基準加工時間パラメータを初期化する.

手順 2: 生成するジョブセット数を $gen = 10000$ とし，$gen = 0, \ldots, gen - 1$ について以下の処理を繰り返す.

手順 3: 受注数の候補として，平均 μ_{odder}，分散 σ^2_{odder} の正規分布からサンプルサイズ $odder_{size}$ 個の乱数系列を生成する．生成された乱数のうち 0 以下の値を除外し，残った候補の中から 1 つをランダムに選択して，実際の受注数 $odder_{number}$ とする.

手順 4: 同様に，平均 μ_{lot}，分散 σ^2_{lot} の正規分布からロット数候補を生成し，0 以下を除外した上でロット数の候補集合を得る.

手順 5: 受注数 $odder_{number}$ と製品種類数 6 に基づき，オーダー原始データ行列 $odder$ （サイズ $odder_{number} \times count_{type}$）を初期化する．各注文 O_i と製品種別 J_s について，確率 $P = 0.7$ で当該製品が注文に含まれるとし，含まれる場合にはロット数候補集合から 1 つをランダムに選択して $odder_{i,s}$ に代入する.

手順 6: 行列 $odder$ の列方向の合計に，基板 1 ロットあたり 10 枚を乗じることで，各製品種別ごとの総基板枚数ベクトル $product_{count}$ を算出する.

手順 7: 各製品種別 J_s および工程番号 q について，機械割当行列 OM の要素 $OM_{s,q}$ を参照し，処理時間行列 $OT_{s,q}$ を決定する．枚数に比例する工程では $product_{count}[s]$ と単位加工時間を乗じ，バッチ処理工程では基準加工時間をそのまま用いる.

手順 8: 行列 OM および OT を用いて，行列 $JOBSET$ を作成する．$JOBSET$ は行方向に製品種別 (製品 1 製品 6), 列方向に工程番号 (加工工程 1 加工工程 17) を取り，各要素に「(機械番号, 加工時間)」の組を文字列として格納する．これにより，$JOBSET$ の機械番号部分が OM, 加工時間部分が OT に対応する.

手順 9: 生成条件（分布種別，受注数分布パラメータ，ロット数分布パラメータ，使用機械数，シード値）に基づいて出力ディレクトリおよびファイル名を構成し，$JOBSET$ を `.csv` 形式で保存する.

そして同じジョブの作業について、同じ機械で連続的な二つの加工工程を加工しないように、作業の前作業と後作業のどれかの作業機械は一致しないと設定する。

$$\begin{cases} \text{before}\{m_{s,q}\} \neq m_{s,q} \\ \text{after}\{m_{s,q}\} \neq m_{s,q} \end{cases} \tag{6}$$

4.4 データ生成結果

先節で述べた通りに 10000 個のデータを実際に生成した。この中から 121 個を選択し、一つづつジョブセットの中にすべての加工工程の加工時間の分散を計算し、分散の大きさを基準として、三分位で注文書を注文 (小)、注文 (並) と注文 (大) に分けてラベルつける。選択されたデータを次章の実験で利用する。

id	分散	タイプ	40	210131.65	一般注文	81	162248.8	小さい注文
0	291969.24	一般注文	41	411175.81	大きい注文	82	380666.39	大きい注文
1	599314.24	大きい注文	42	138922.01	小さい注文	83	78159.49	小さい注文
2	319267.09	一般注文	43	182554.24	小さい注文	84	229245.53	一般注文
3	188772.75	一般注文	44	201636.06	一般注文	85	299946.27	一般注文
4	294717.66	一般注文	45	63946.44	小さい注文	86	344938.76	大きい注文
5	110509.24	小さい注文	46	597108.72	大きい注文	87	370693.73	大きい注文
6	247450.44	一般注文	47	301417.72	一般注文	88	364219.22	大きい注文
7	260110.28	一般注文	48	327616.44	大きい注文	89	130341.63	小さい注文
8	324851.89	一般注文	49	251280.34	一般注文	90	388925.06	大きい注文
9	188229.49	小さい注文	50	150776.23	小さい注文	91	258825.81	一般注文
10	188604.81	小さい注文	51	245016.77	一般注文	92	257119.68	一般注文
11	152457.34	小さい注文	52	343285.68	大きい注文	93	392275.28	大きい注文
12	672143.73	大きい注文	53	190369.68	一般注文	94	176454.68	小さい注文
13	102620.49	小さい注文	54	185743.92	小さい注文	95	432331.39	大きい注文
14	131368.97	小さい注文	55	155210.91	小さい注文	96	283358.84	一般注文
15	171182.26	小さい注文	56	130460.12	小さい注文	97	113377.07	小さい注文
16	272638.45	一般注文	57	339203.95	大きい注文	98	144882.4	小さい注文
17	292861.52	一般注文	58	365846.14	大きい注文	99	188579.98	小さい注文
18	236118.57	一般注文	59	270086.07	一般注文	100	379744.05	大きい注文
19	182211.52	小さい注文	60	343895.57	大きい注文	101	292252.14	一般注文
20	391714.39	大きい注文	61	409258.35	大きい注文	102	305206.57	一般注文
21	209852.97	一般注文	62	312287.02	一般注文	103	626685.43	大きい注文
22	205242.88	一般注文	63	370204.37	大きい注文	104	159087.5	小さい注文
23	366097.62	大きい注文	64	139458.61	小さい注文	105	356260.74	大きい注文
24	386418.16	大きい注文	65	148883.48	小さい注文	106	380604.03	大きい注文
25	405648.95	大きい注文	66	265197.71	一般注文	107	100954.11	小さい注文
26	89952.37	小さい注文	67	569503.32	大きい注文	108	397146.19	大きい注文
27	331673.7	大きい注文	68	301312.64	一般注文	109	109580.38	小さい注文
28	224600.76	一般注文	69	106459.81	小さい注文	110	182487.72	小さい注文
29	129098.16	小さい注文	70	297568.34	一般注文	111	376685.34	大きい注文
30	394861.39	大きい注文	71	442687.81	大きい注文	112	137131.08	小さい注文
31	592042.46	大きい注文	72	325593.35	一般注文	113	119496.01	小さい注文
32	168737.83	小さい注文	73	612936.46	大きい注文	114	163610.74	小さい注文
33	253819.68	一般注文	74	387869.48	大きい注文	115	295326.65	一般注文
34	299788.45	一般注文	75	120853.66	小さい注文	116	115524.36	小さい注文
35	106247.96	小さい注文	76	331364.62	大きい注文	117	347402.52	大きい注文
36	326180.19	大きい注文	77	484482.24	大きい注文	118	352087.02	大きい注文
37	199461.7	一般注文	78	324833.29	一般注文	119	420986.01	大きい注文
38	244235.5	一般注文	79	183299.36	小さい注文	120	314926.07	一般注文
39	210356.64	一般注文	80	170159.77	小さい注文			

図 2: データ分類

5. 遺伝的アルゴリズムによる解の改善

本研究では、JSSP に対して、GA を用いたスケジューリングの最適解探索手法を採用する。加工順序が厳密に定まる問題なので、PPS(Precedence Preservative Crossover) という交差手法を採用する。

5.1 遺伝的アルゴリズムの考え方

本研究では、JSSP に対して、ジョブごとの工程順序を厳密に保持しつつ、複数の製品に属する加工工程の加工順序を染色体として表現する遺伝的アルゴリズム (GA) を用い、解の改善を行う。

まず、注文書であるジョブセットの各製品ジョブ J_s の加工工程列は、機械割当行列 OM および加工時間行列 OT に基づいて保存する。1 つのジョブは、「使用機械 $m_{s,q}$ と加工時間 $t_{s,q}$ の組から

なる工程」として表現される。染色体は、すべてのジョブに属する各工程を一度ずつ含む遺伝子列

$$\text{genes} = \left[J_s F_q, J_{s'} F_{q'}, \ldots \right]$$

として定義され、各遺伝子は「ジョブ J_s の第 q 工程」を表す。スケジュールの手順について、遺伝子列に従って工程を順に配置し、同一製品ジョブ内の工程順序制約および各機械が同時に 1 工程しか処理できないという資源制約のもとで処理開始時刻と終了時刻を更新することにより、スケジュール全体の総加工時間を算出する。

初期解は、各製品ジョブ内の加工工程順序を保つままランダムで遺伝子列に導入する方法によって生成する。具体的には、各ジョブについて「次に処理すべき工程インデックス」を管理し、未処理工程を有するジョブの集合からランダムに 1 ジョブを選択し、そのジョブの次の加工工程を遺伝子列の末尾に追加する操作を繰り返す。この手続きを全ジョブの全加工工程が配置されるまで継続することで、すべての工程を一度ずつ含み、かつ製品ジョブ内の加工工程順序が必ず守られた初期解が得られる。

GA の適応度評価は、各染色体が表現するスケジュールの品質を定量的に評価するために行われる。本研究では、JSSP における代表的な性能指標である総加工時間を評価基準として採用する。各染色体が示す遺伝子列をスケジュールへとデコードし、製品ジョブ内の工程順序制約および機械資源制約を満たす実行計画を構成した上で、対応する総加工時間を算出することで、解の良さを比較する。

適応度評価では、各染色体に対してスケジュールを配置し、対応する総加工時間を適応度として計算する。本研究では総加工時間の最小化を目的とするため、適応度は値が小さいほど良好な解であることを意味する。計算された総加工時間は染色体オブジェクト内に保存され、同一染色体に対する重複総加工時間計算を避けることで、計算効率の向上を図っている。

親個体の選択にはトーナメント選択を採用する。許容解集団からランダムに複数個体（本研究では 5 個体）を抽出し、その中で総加工時間が最小の個体を勝者として選択する操作を 2 回行うことで、二つの親染色体を決定する。この選択方式により、適応度の高い個体が次世代に残りやすくなる一方で、一定のランダム性が維持され、探索の多様性が確保される。

GA で二つの親から特徴を継承し子個体を生成するためには、交叉操作が必要です。交叉操作には、PPS(Precedence Preservative Crossover) と呼ばれる順序保持型の交叉手法を用いる。この交差手法の紹介は次章で述べます。生成後は染色体修復処理を行う、加工工程の欠落や重複が生じないように解の有効性を保証する。

また、本研究の GA ではエリート保存戦略を採用している。各世代において総加工時間が最小となる個体から、あらかじめ定めた個数（本研究では 2 個体）を次世代集団へそのまま引き継ぐことで、進化の過程で得られた最良解が失われることを防止する。残りの個体は、選択・交叉操作によって生成された子個体で構成され、所定の最大ジェネレーション世代数（本実装では 200 世代）に達するまで進化計算を繰り返す。最終的に、全ジェネレーション世代を通じて得られた最良染色体を GA の解として採用し、対応するスケジュールについて甘特図と各ジェネレーショ世代にもっとも短い総加工時間の折れ線グラフを描画することで、視覚的な評価も行う。

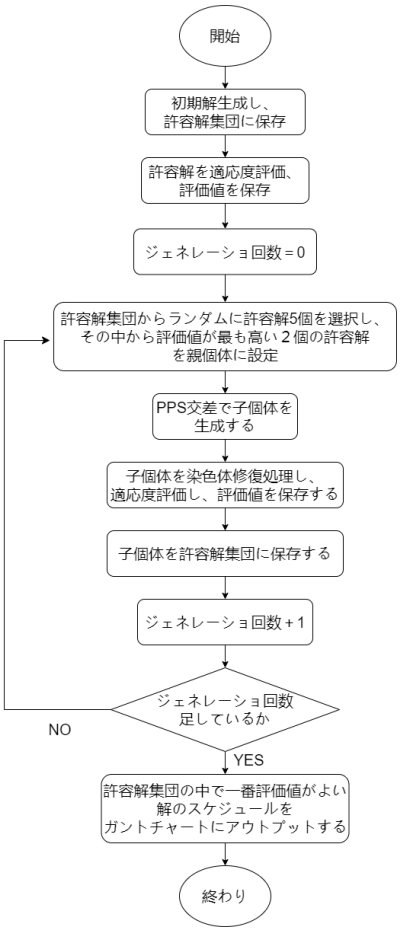


図 3: 遺伝的アルゴリズムフローチャート

5.2 PPS 交差法の考え方

PPS 交差法は、JSSP 問題などの順序制約を伴う組合せ最適化問題において有効な交差手法の一つである。本手法は、親個体の遺伝子列情報を交差する際に、工程間の優先順序を保持しつつ、新たな子個体を生成することを目的としている。

これから、PPS の交差手順を紹介します。

1. 2 つの親染色体 (Parent1, Parent2) を許容解集団から選択する。

2. Parent1 と Parent2 から操作配列を主に利用する。

3. 子個体の染色体を初期化する。

4. Parent1 と Parent2 をランダムに選択する。選択された親個体の頭から、選択されない操作を子個体操作配列の最後に挿入する。

5. すべての操作が配置されるまで繰り返す。

数式：

- $J = \{J_1, J_2, \ldots, J_s\}$：製品種類の集合
- 各製品 J_s は q 個の加工操作を持ち、$F_q = \{J_sF_1, J_sF_2, \ldots, J_sF_q\}$ とする。
- 親個体は操作列として表現される：$P_1 = [o_1^1, o_2^1, \ldots, o_\theta^1]$, $P_2 = [o_1^2, o_2^2, \ldots, o_\theta^2]$
- 親個体の頭からまだ子個体に配置していない操作：$P_1o \in [o_1^1, o_2^1, \ldots, o_\theta^1], P_1o \notin C$ 、 $P_2o \in [o_1^2, o_2^2, \ldots, o_\theta^2], P_2o \notin C$
- 子個体 C を空列として初期化：$C \leftarrow [\ ]$

1. $i \leftarrow 1$
2. **while** $|C| < \theta$ **do**:
   (a) $Po = random\{P_1o, P_2o\}$
   (b) $C \leftarrow Po$
   (c) $i \leftarrow i + 1$
3. **return** C

5.3 GA の実験結果

以上の手順で 121 個データをスケジュールして、最後の許容解総加工時間とガントチャートを結果としてアウトプットした。

本研究を背景としての問題設定およびジョブセットの規模でしたら、ジェネレーショ終えて許容解を出せるまで概ね 17 秒左右がかかる。以下は注文タイプにより、三つ種類の中でランダムに選択された注文のジェネレーショの最短総加工時間折れ線グラフとガントチャート。

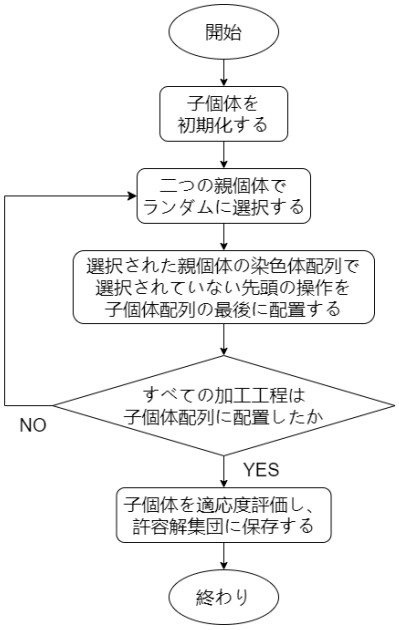
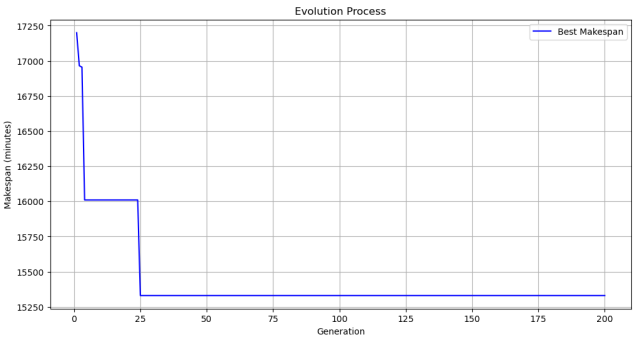


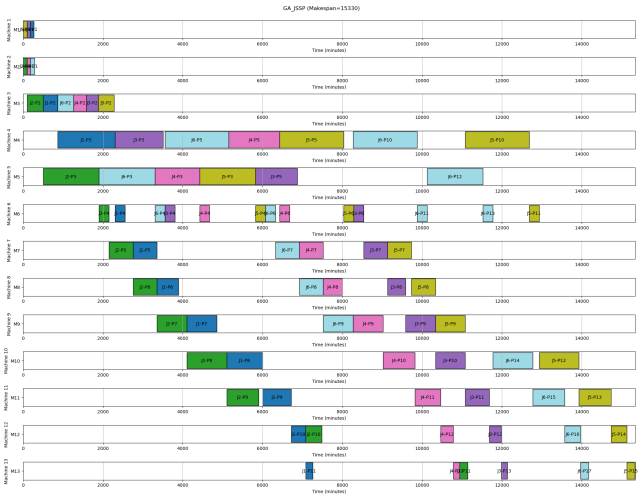
図 4: PPS フローチャート

id	タイプ	GA総加工時間	40	一般注文	16565	81	小さい注文	13035
0	一般注文	18665	41	大きい注文	21210	82	大きい注文	19930
1	大きい注文	24955	42	小さい注文	12575	83	小さい注文	9945
2	一般注文	19085	43	小さい注文	14135	84	一般注文	16290
3	一般注文	15300	44	一般注文	16365	85	一般注文	17830
4	一般注文	18510	45	小さい注文	8585	86	大きい注文	20235
5	小さい注文	12740	46	大きい注文	24050	87	大きい注文	19940
6	一般注文	16660	47	一般注文	18410	88	大きい注文	20230
7	一般注文	18270	48	大きい注文	18945	89	小さい注文	13080
8	一般注文	19485	49	一般注文	17385	90	大きい注文	20515
9	小さい注文	14860	50	小さい注文	13395	91	一般注文	16700
10	小さい注文	14385	51	一般注文	15380	92	一般注文	17335
11	小さい注文	14030	52	大きい注文	18820	93	大きい注文	20995
12	大きい注文	26620	53	一般注文	14660	94	小さい注文	14500
13	小さい注文	11310	54	小さい注文	14160	95	大きい注文	21030
14	小さい注文	12840	55	小さい注文	14340	96	一般注文	17845
15	小さい注文	14375	56	小さい注文	13630	97	小さい注文	12330
16	一般注文	15950	57	大きい注文	18840	98	小さい注文	12825
17	一般注文	17505	58	大きい注文	20120	99	小さい注文	14000
18	一般注文	16400	59	一般注文	16180	100	大きい注文	20535
19	小さい注文	15100	60	大きい注文	19310	101	一般注文	16120
20	大きい注文	20435	61	大きい注文	21265	102	一般注文	18625
21	一般注文	15310	62	一般注文	18925	103	大きい注文	25225
22	一般注文	15410	63	大きい注文	21065	104	小さい注文	14430
23	大きい注文	19655	64	小さい注文	13080	105	大きい注文	19000
24	大きい注文	19955	65	小さい注文	13655	106	大きい注文	20255
25	大きい注文	20820	66	一般注文	17690	107	小さい注文	10575
26	小さい注文	9865	67	大きい注文	24375	108	大きい注文	20685
27	大きい注文	18330	68	一般注文	18795	109	小さい注文	11805
28	一般注文	15410	69	小さい注文	11475	110	小さい注文	15000
29	小さい注文	12755	70	一般注文	19055	111	大きい注文	20095
30	大きい注文	21430	71	大きい注文	21900	112	小さい注文	13415
31	大きい注文	24855	72	一般注文	19990	113	小さい注文	12455
32	小さい注文	13660	73	大きい注文	25270	114	小さい注文	14520
33	一般注文	17900	74	大きい注文	20270	115	一般注文	19000
34	一般注文	18780	75	小さい注文	13005	116	小さい注文	12460
35	小さい注文	11995	76	大きい注文	18955	117	大きい注文	18450
36	大きい注文	19640	77	大きい注文	21975	118	大きい注文	19650
37	一般注文	15320	78	一般注文	18940	119	大きい注文	21220
38	一般注文	16950	79	小さい注文	14875	120	一般注文	18490
39	一般注文	15685	80	小さい注文	14170			

図 5: GA により許容解の総加工時間

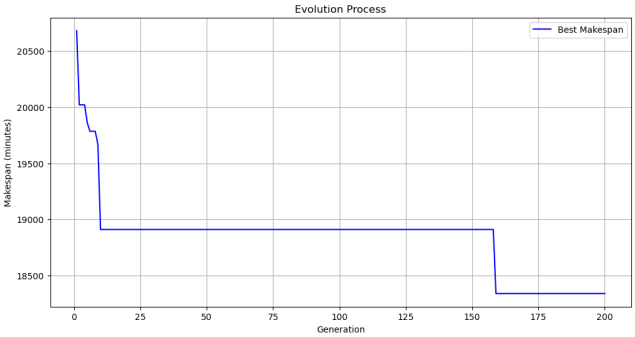


(a) 注文書 19(小さい注文) のジェネレーショ折れ線グラフ

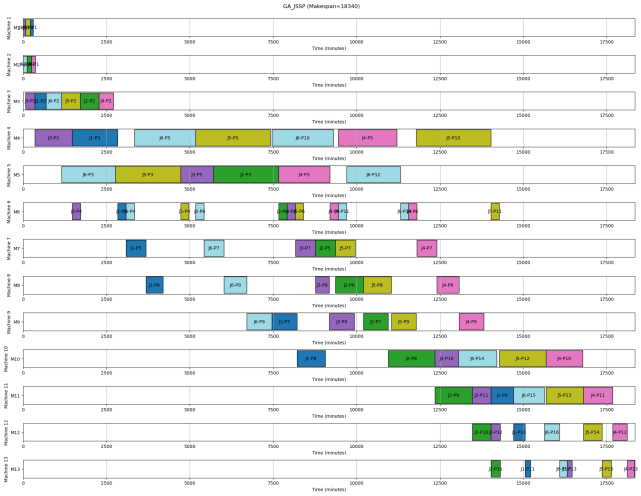


(b) 注文書 19(小さい注文) の許容解ガントチャート

図 6: 注文書 19(小さい注文) の実験結果

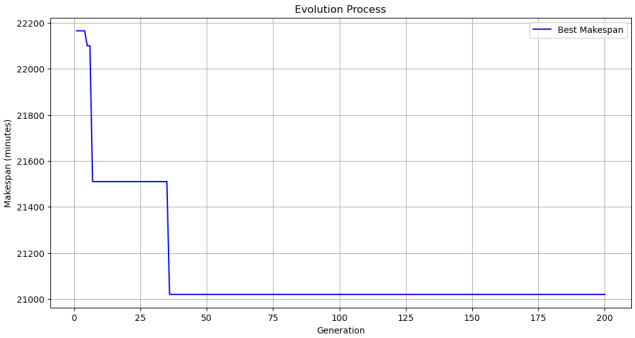


(a) 注文書 68(一般注文) のジェネレーショ折れ線グラフ

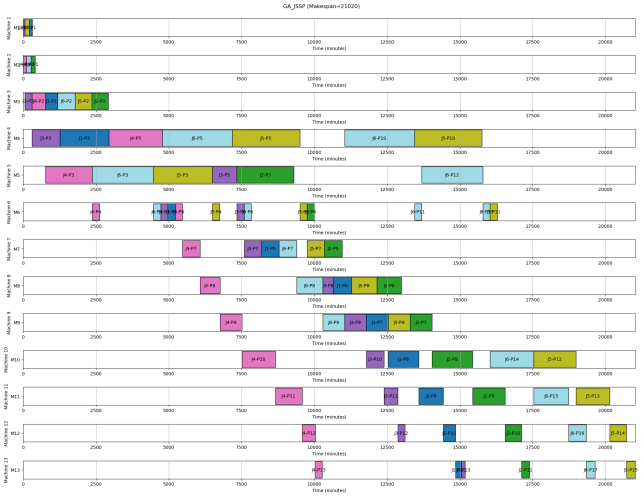


(b) 注文書 68(一般注文) の許容解ガントチャート

図 7: 注文書 68(一般注文) の実験結果



(a) 注文書 93(大きい注文) のジェネレーショ折れ線グラフ



(b) 注文書 93(大きい注文) の許容解ガントチャート

図 8: 注文書 93(大きい注文) の実験結果

GA の実験結果を見て、ジェネレーショ回数は 50 回以下でも、場合によって許容解を出せる。さいあくでも 200 回以下でいい許容解を出せる。ガントチャートから見ると、加工工程のスケジューリングも詰まられて仕事した。JSSP は典型的な NP 問題であることで、問題により厳密解を出せるためには総当たり法が必要となります。こうやって、GA により JSSP のスケジュール最適化は高速に受けれる解を出せるのが見られます。

6. 深層強化学習による解の改善

深層強化学習 (Deep Reinforcement Learning,Deep RL) とは、強化学習 (Reinforcement Learning,RL) と深層学習を組み合わせた機械学習の一分野である。RL は意思決定問題を扱う手法として広く用いられており、一般にマルコフ決定過程 (Markov Decision Process、MDP) として定式化される。RL はエージェントと環境から構成され、エージェントは環境との相互作用を通じて、全体として最適なポリシーを探索する。

エージェントは、まず環境の状態を観測し、現在保持している方策に基づいて行動候補の中から行動を選択する。その行動により環境の状態が変化し、エージェントは新たな環境において再び行動を選択する。この過程において、環境から報酬が与えられ、エージェントは将来にわたる報酬を定期的に評価することで、ポリシーの更新および改善を行う。

RL と Deep RL の主な違いは、行動選択の際にニューラルネットワークを用いる点にある。Deep RL では、ニューラルネットワークによって状態と行動の関係を近似することで、行動候補の中から適切な行動を選択する。これにより、複雑なモデルを比較的簡潔な形式で表現できるとともに、意思決定に関与する多様な要因を柔軟に考慮することが可能となる。

さらに、Deep RL においては、エージェントが試行錯誤を通じて方策を学習・改良することで最適ポリシーを獲得し、動的に変化する環境においても、その最適ポリシーに基づいた意思決定が可能となる。

6.1 深層強化学習の考え方

強化学習は、環境との相互作用を通じて逐次的に意思決定を行う枠組みであり、一般にマルコフ決定過程 (MDP) として表現される。MDP は、状態集合 (環境) S、行動集合 A、報酬関数 R、および割引率 γ から構成される。

本研究では、価値関数に基づく強化学習手法である Deep Q-Network(DQN) を用いる。DQN では、各状態 s_t における行動 a_t の価値を表す行動価値関数 $Q(s_t, a_t)$ をニューラルネットワークにより近似し、各状態における行動選択を Q 値に基づいて行う。

エージェント、時刻 t において環境の状態 $s_t \in S$ を観測し、$Q(s_t, a)$ が最大となる行動を選択することで、将来にわたる累積報酬の最大化を目指す。学習過程においては、探索とすでに学習した特徴を活用のバランスを取るため、ε-greedy 法を用いて確率的にランダム行動を選択する。

時刻 t において，$[0, 1]$ の一様分布から乱数 u を生成し，行動 a_t は次式により決定される：

$$a_t = \begin{cases} \text{random}(A_t) & \text{if } u < \varepsilon, \\ \arg\max_{a_t \in A_t} Q(s_t, a_t; \theta) & \text{if } u \geq \varepsilon. \end{cases} \tag{7}$$

なお，本研究では学習の進行に伴い ε を段階的に減少させることで、初期段階では探索を重視し、学習が進むにつれて活用を優先する設計としている。

本研究で扱う JSSP 環境は、加工工程の開始および完了といったイベントによって状態が更新されるイベント駆動型の環境として構成されている。

DQN における 1 step とは、時刻 t においてエージェントが状態 s_t を観測し、行動 a_t を選択した後、環境が次状態 s_{t+1} と報酬 r_t を返すまでの連続的な意思決定単位を指す。

行動 a_t には、

- 実行可能な作業工程を選択してスケジューリングを行う行動
- 次のイベント発生まで待機する行動

の両方が含まれており、各 step において環境状態が更新される。

そして、episode とは、一つのジョブセットに対して、初期状態から開始し、すべての作業工程が完了するまでのことを指す。すなわち、1 episode は一つのジョブセットに対する完全なスケジューリング過程に対応する。

本実装では、メインネットワークとターゲットネットワーク、二つのネットワークを用いる。メインネットワークは、状態 s_t を入力として各行動の Q 値を出力し、ε-greedy 法により実行行動 a_t を決定するために用いられる。また、経験再生でサンプルされた (s_t, a_t, r_t, s_{t+1}) に対して、メインネットワーク出力から $Q(s_t, a_t)$ を算出し、誤差逆伝播によりメインネットワークのパラメータ更新を行う。

経験再生は DQN の学習においては、連続した経験データ間の相関を減り、学習の安定性とデータ効率を向上させるため導入する。

各 step において得られる経験 (s_t, a_t, r_t, s_{t+1}) は、リプレイバッファ(replay buffer) に保存される。学習時には、このバッファからランダムにミニバッチを抽出し、損失関数に基づいてメインネット

ワークの更新を行う。この手法により、

- 学習データの相関を低減できる
- 過去の重要な経験を繰り返し利用できる
- 学習の発散を抑制できる

といった効果が得られる。

また本研究では，一定数以上の経験が蓄積された後に経験再生による学習を開始することで，初期段階における不安定な更新を回避している。

ターゲットネットワークは、将来報酬に相当する非リアルタイムな価値成分の推定に用いられる。具体的には、次状態 s_{t+1} に対してターゲットネットワークから $\max_{a'} Q(s_{t+1}, a'; \theta^-)$ を計算し、

$$y_t = r_t + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-) \tag{8}$$

として目標値 y_t を構成する。これにより、即時に得られる報酬 r_t（リアルタイムな報酬成分）と、将来の累積報酬に対応する価値（非リアルタイムな価値成分）を分離して扱うことができる。

さらに、本実装では学習の安定化のため、一定ステップ間隔ごとにメインネットワークのパラメータをターゲットネットワークへコピーするハード更新を行う。具体的には、現時点ステップ数が設定していた更新ステップの倍数となるたびに $\theta^- \leftarrow \theta$ として同期する。また、固定テストデータ集合に対する定期評価では、評価値のばらつきを抑える目的からターゲットネットワークを用いて計算を行う。

メインネットワークのパラメータ θ の更新には，確率的勾配降下法に基づく最適化手法である Adam オプティマイザを用いる。学習率を α とすると，各学習ステップにおけるパラメータ更新は，損失関数 $L_t(\theta)$ の勾配に基づいて行われる。以上の処理により，各ステップにおいてメインネットワークのパラメータ更新が行われる。

DQN における学習は，メインネットワークが出力する行動価値 $Q(s_t, a_t; \theta)$ と，ターゲットネットワークを用いて構成される目標値 y_t との差を最小化することを目的として行われる。

損失関数として平均二乗誤差 (Mean Squared Error, MSE) を採用し、時刻 t における損失 $L_t(\theta)$ を次式で定義する：

$$L_t(\theta) = \mathbb{E}\left[(y_t - Q(s_t, a_t; \theta))^2\right]. \tag{9}$$

ここで，目標値 y_t はターゲットネットワークを用いて式 8 として計算される。

表 5: 強化学習および DQN に用いる記号の定義

記号	説明
S	状態集合（環境が取り得る全状態の集合）
A	行動集合（エージェントが選択可能な行動の集合）
s_t	時刻 t における環境の状態
a_t	時刻 t において選択される行動
A_t	状態 s_t における実行可能行動集合
r_t	時刻 t に得られる即時報酬
R	報酬関数
α	学習率
γ	割引率（将来報酬の重み付け係数）
$Q(s,a)$	状態 s において行動 a を選択した際の行動価値
$Q(s,a;\theta)$	メインネットワークにより近似される行動価値関数
$Q(s,a;\theta^-)$	ターゲットネットワークにより近似される行動価値関数
θ	メインネットワークのパラメータ
θ^-	ターゲットネットワークのパラメータ
y_t	学習における目標値
ε	ε-greedy 法における探索率
a'	次状態における行動候補
$L_t(\theta)$	時刻 t における損失

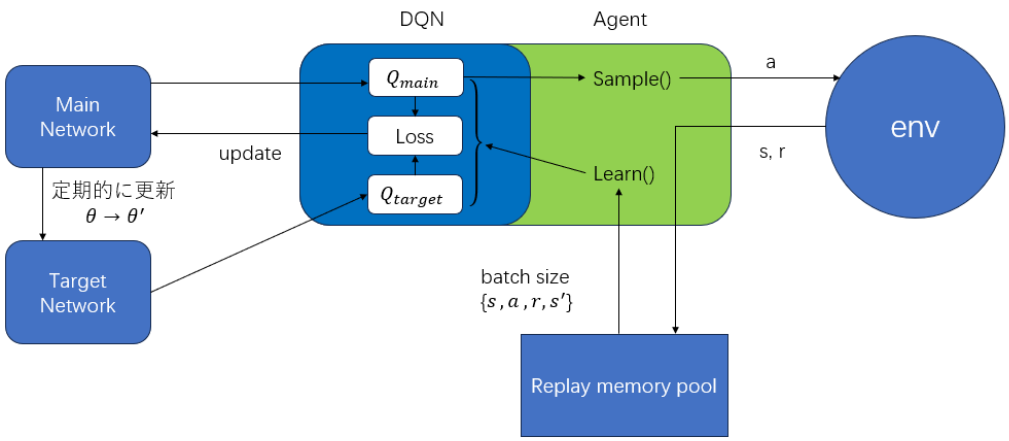


図 9: DQN の仕組み

6.2 DQN を JSSP に適用する理由

前章では、GA を用いて JSSP に対するスケジューリング最適化を行い、比較的高品質な許容解を得られることを示した。GA は問題構造に依存せず適用可能であり、事前学習を必要としない点で汎用性の高い手法である。一方で、各ジョブセットに対して毎回多数の世代にわたる進化計算を実行する必要があり、1 回のスケジューリングに要する計算時間は問題規模に応じて増大する。

これに対し、本研究で採用する DQN は、環境との相互作用を通じてスケジューリングポリシーを Q-value によって学習する手法である。DQN では、学習段階において多数のジョブセットを用いてモデルを訓練する必要があるものの、一度学習が完了した後は、各ジョブセットに対するスケジューリングを逐次的な推論処理として高速に実行できる。そのため、同一規模のジョブセットに対する 1 回の完全スケジューリングに要する時間は、GA と比較して大幅に短縮される。

特に、生産現場においては、機械故障や新規受注の追加などによりスケジュールを頻繁に再計算する必要が生じる。このような状況では、毎回最初から探索を行う GA よりも、学習済みモデルに基づいて即時に意思決定を行える DQN の方が適していると考えられる。

以上より、GA は静的環境における高精度なスケジューリング最適化に有効である一方、DQN は学習コストを伴うものの、単一ジョブセットに対する計算時間が短く、動的環境やリアルタイム性が要求される状況において優位性を有する。本研究では、これらの特性を踏まえ、GA と DQN の両手法を比較対象として位置付ける。

6.3 報酬関数の構成

本研究では DQN によりジョブショップスケジューリングを逐次意思決定として解くため, 各 step における即時報酬 r_t を設計する. 報酬設計は学習の収束性、アクション選択方策の偏りまたは訓練済みモデルの性能に強く影響するため,

- 複数指標を組み合わせた複合型報酬関数
- $makespan$ を直接最適化する単純型報酬関数

の 2 種類を実装し, 比較した.

6.3.1 複合型報酬関数

複合型報酬では, スケジューリングの良さを単一指標に限定せず, 「$makespan$ の短縮」「各ジョブの加工プロセスの推進」「機械負荷の同一化」「待ち時間の抑制」など，複数の観点を同時に評価することで学習を安定化またはモデル性能の向上させることを狙う．時刻 t における報酬 r_t は, 以下の重み付き和として定義する：

時刻 t における報酬 r_t は，以下の重み付き和として定義する：

$$r_t = r_t^{\text{time}} + r_t^{\text{prog}} + r_t^{\text{bal}} + r_t^{\text{comp}} + r_t^{\text{opp}} + r_t^{\text{util}} + r_t^{\text{cp}} + r_t^{\text{bn}} + r_t^{\text{final}}. \tag{10}$$

各項は以下の通りである（実装では係数はハイパーパラメータとして設定する）：

- **加工時間ペナルティ**: 選択した第 s 種別製品の第 q 番目の加工工程の加工時間を $t_{s,q}$ とすると，

$$r_t^{\text{time}} = -\alpha t_{s,q}. \tag{11}$$

  長い工程の選択を抑制し，局所的な時間増加を抑える目的で導入する．

- **加工工程推進報酬**: 全工程数を N_{ops}, 当 step の完了工程数を N_{step}^f とし，進捗率を $g_{step} = N_{step}^f / N_{ops}$ と定義する．

$$r_t^{prog} = \beta(g_{step+1} - g_{step}). \tag{12}$$

  1 step の意思決定がイベント駆動により「工程完了」という推進することを明示するための報酬である．

- **機械負荷平準化**: 機械 M_m の累積使用時間を t_m^{sum} とし，

$$r_t^{bal} = -\gamma std\left(\{t_m^{sum}\}\right). \tag{13}$$

  特定機械へのボトルネック化を抑える目的で導入する．

- **ジョブスケジュール完了報酬**: あるジョブの最終工程が完了した場合に定数報酬を与える：

$$r_t^{\text{comp}} = \begin{cases} \delta & (\text{ジョブ } s \text{ が完了}) \\ 0 & (\text{それ以外}) \end{cases}. \tag{14}$$

- **機会損失ペナルティ（最短工程の未選択）**：実行可能行動集合を A_t とし，その中で最短加工時間を t_{min}^A とする．選択した第 s 種別製品の第 q 番目の加工工程の加工時間 $t_{s,q}$ が t_{min}^A より大きい場合，

$$r_t^{\text{opp}} = -\eta(t_{s,q} - t_{min}^A). \tag{15}$$

  「すぐ終わる作業」の優先を促す報酬である．

- **アイドル抑制（空き機械の即時利用）**：空き機械が存在するにもかかわらず開始が遅れる状況

を抑えるため，機械が隙間がないように稼働させた場合は小さな正な報酬，逆に機械加工待ちが増える場合は軽微な負報酬を与える．

- **残作業最大ジョブ優先**: 残り作業量の最も大きいジョブを優先した場合に小さな正な報酬を与える．

- **ボトルネック回避**: 現時点で機械が残した加工工程が一番多いのをボトルネックとみなし，その機械をさらに悪化させる選択には軽微な負な報酬を与える一方，非ボトルネック機械を活用する選択には正報酬を与える．

- **エピソード終端の効率報酬**: すべてのジョブの加工工程が完了し，エピソードが終了した時点において，当該スケジューリング案の総加工時間 $makespan$ を $C_{\max}$ とする．また，全加工工程の加工時間総和を

$$T^{sum} = \sum_{s} \sum_{q} t_{s,q} \tag{16}$$

と定義し，加工機械台数を $|M|$ とすると，理論的な下界として

$$C^{LB} = \frac{T^{sum}}{|M|} \tag{17}$$

を用いる．

エピソード終端時の報酬は，実際の $makespan$ が理論下界にどの程度近いかを評価する指標として，以下のように定義する：

$$r_t^{\text{final}} = \begin{cases} \kappa \dfrac{C^{LB}}{C_{\max}} & \text{(エピソード終端時)} \\ 0 & \text{(それ以外)} \end{cases}. \tag{18}$$

この終端報酬により，エージェントは各 step における局所的な判断だけでなく，エピソード全体として $makespan$ を短縮する方策を学習することが期待される．

一方で，複合型報酬は各項が相互に干渉しやすく，重みの調整によっては「局所的に見栄えの良い行動」を過度に促進してしまい，目的である $C_{\max}$ 最小化と整合しない方策に収束する可能性がある．本研究の実験でも，複合型報酬は学習の安定性は得られる場合がある一方，最終的な $makespan$ 改善が限定的となる傾向が確認された．

6.3.2 $makespan$ 中心型報酬関数

次に，目的関数と報酬の整合性を高めるため，$makespan$ の増分に基づく単純型報酬を設計する．step における $makespan$(推測完了時刻の最大値) を $C_{\max}(t)$ とすると，

$$r_t^{\text{main}} = -\Delta C_{\max}(t) = -\left(C_{\max}(t+1) - C_{\max}(t)\right). \tag{19}$$

すなわち，意思決定によって $C_{\max}$ が増えるほど負の報酬となり，$C_{\max}$ の増加を抑える行動を学習する．

さらに，実装上はプロセス推進の停滞（無意味な待機や不適切な選択可能な加工プロセススキップ）を抑制するため，以下の補助項を加える：

$$r_t = r_t^{\text{main}} + r_t^{\text{imm}} + r_t^{\text{pen}} \tag{20}$$

- **即時着手ボーナス** r_t^{imm}: 「最も早く待機している機械」に対して直ちに工程を開始できた場合に小さな正報酬を与える．これは $C_{\max}$ だけでは区別しにくい局所的な停滞を抑えるためである．
- **待機・プロセススキップに関連ペナルティ** r_t^{pen}: 実行可能工程が存在するにもかかわらず待機する，または不適切なスキップを繰り返すなど，スケジュール生成に推進しない行動に負報酬を与える．

この単純型報酬関数は目的関数 $C_{\max}$ と直接かかわっているため，報酬設計によるバイアスが小さく，本研究の実験では複合型より $makespan$ を安定的に改善できる傾向が確認された．

参考文献

[1] K.T. Chung, C.K.M.Lee and Y.P. Tsang, "Neural combinatorial optimization with reinforcement learning in industrial engineering: a survey" *Artif Intell Rev.* vol. 58, 130(2025).

[2] M. Xu, Y. Mei, F.F. Zhang and M.J. Zhang, "Learn to optimise for job shop scheduling: a survey with comparison between genetic programming and reinforcement learning" *Artif Intell Rev.* vol. 58, 160(2025).

[3] C. Ngwu, Y. Liu and R. Wu, "Reinforcement learning in dynamic job shop scheduling: a comprehensive review of AI-driven approaches in modern manufacturing" *J Intell Manuf.*(2025).

[4] X.R. Shao and C.S. Kim, "An Adaptive Job Shop Scheduler Using Multilevel Convolutional Neural Network and Iterative Local Search" *IEEE Access.* vol. 10, pp. 88079-88092(2022).

[5] L.B. Wang, X. Hu, Y. Wang, S. Xu, S. Ma, K. Yang, Z. Liu and W. Wang, "Dynamic job-shop scheduling in smart manufacturing using deep reinforcement learning" *Computer Networks.* vol. 190, 107937(2021).

[6] S. Lee, Y. Cho and Y.H. Lee, "Injection Mold Production Sustainable Scheduling Using Deep Reinforcement Learning" *Sustainability.* vol. 12, 8718(2020).

[7] C. Pickardt, J. Branke, T. Hildebrandt, J. Heger and B. Scholz-Reiter, "Generating dispatching rules for semiconductor manufacturing to minimize weighted tardiness" *Proceedings of the 2010 Winter Simulation Conference.*Baltimore, MD, USA, pp. 2504-2515(2010).

[8] 村山 昇, 川田 誠一, "遺伝的アルゴリズムを用いた加工機械と複積載 AGV の同時スケジューリング" 日本機械学会論文集 *(C編).* vol. 12, 712, pp. 3638-3643(2005-12).

[9] B. Yu, Z. Hui, Z. Xin, C. Zheng, J. Riku, Y. Kun, "Toward Autonomous Multi-UAV Wireless Network: A Survey of Reinforcement Learning-Based Approaches" *IEEE COMMUNICATIONS SURVEYS AND TUTORIALS.* vol. 25, No. 4, FOURTH QUARTER 2023, pp. 3038-3067(2023).

[10] 小野 啓介, 森川 克己, 長沢 敬祐, 高橋 勝彦, "フレキシブルジョブショップ環境の受託製造企業におけるエネルギー消費量配分問題" *J Jpn Ind Manage Assoc.* vol. 72, pp. 179-187(2022).

[11] 貝原 俊也, 國領 大介, 藤井 信忠, 村上 亘, 梅田豊裕, ”フレキシブルジョブショップを対象とした受注生産における機械稼働計画立案もための基礎検討” 第 *63*回自動制御連合講演会. (2020).

[12] 貝原 俊也, 國領 大介, 藤井 信忠, 西村 翔平, ”CPS 型ファクトリセキュリティ実現に向けた生産スケジューリング手法に関する研究-マスカスタム生産対応フレキシブルオープンショップを対象とした検討-” 第 *61* 回自動制御連合講演会. (2018).

[13] 平沼 智之, 安田 翔也, 藤堂 健世, 谷口 茉帆, 山村 雅幸, ”組合せ最適化ぬおけるベイジアン最適化アルゴリズムを組み込んだ遺伝的アルゴリズムの提案” *The 35th Annual Conference of the Japanese Society for Artificial Intelligence*. (2021).

[14] 三神 賢雅, 伊原 滉也, 佐久間 拓人 , 加藤 昇平, ”混合整数最適化に基づく生産ライン作業スケジュール生成システムの開発” *The 36th Annual Conference of the Japanese Society for Artificial Intelligence*. (2022).

[15] 蘭 嘉, 田中 瑛理, 佐々木 優 , 森江 翔, 有馬 澄佳, ”複数種のリソースを共用する多品種生産システムの分散協調スケジューリング-自動車部品後補充生産への適用-” 日本経営工学会論文誌 . vol. 72, No.1, pp. 75-87(2021).

[16] G.Y. Shi, H. IIMA, N. Sannomiya, ”A New Encodnig Scheme for Solving Job Shop Problems by Genetic Algorithm” *Conference on Decision and Control*. (1996).

[17] UMIT BILGE, GUNDUZ ULUSOY, ”A TIME WINDOW APPROACH TO SIMULTANEOUS SCHEDULING OF MACHINES AND MATERIAL HANDLING SYSTEM IN AN FMS” *Operations Research*.vol. 43, No.6, (1995).

[18] 花田 良子, 廣安 知之, 三木 光範, ”遺伝的アルゴリズムによる工場の生産スケジュールの自動生成” *THE SCIENCE AND ENGINEERING REVIEW OF DOSHISHA UNIVERSITY*.vol. 48, No.4, pp.241-248, (2008).

[19] 平中 雄一朗, 西 竜志, 乾口 雅弘, ”ラグランジュ緩和とカット生成による生産工程と複数台搬送車の同時スケジューリング問題に対する分解法” システム制御情報学会論文誌.vol. 20, No.12, pp.465-474, (2007).

[20] Raghda B. Taha, Amin K. El-Kharbotly, Yomna M. Sadek, Nahid H. Afia, ”A Genetic Algorithm for solving two-sided assembly line balancing problems” *Ain Shams Engineering Journal*.vol. 2, pp.227-240, (2011).

[21] Kazi Shah Nawaz Ripon, N. H. Siddique, Jim Torresen, ”Improved precedence preservation crossover for multi-objective job shop scheduling problem” *Evolving Systems*.vol. 2, pp.119-129, (2011).

[22] Andrzej Kiernich, Jerzy Kalenik, Wojciech Steplewski, Marek Koscielski and Aneta Chołaj, ”Impact of Particular Stages of the Manufacturing Process on the Reliability of Flexible Printed Circuits” *Sensors*.25, 140, (2025).

[23] 斎藤 康毅, 『ゼロから作る Deep Learning - Python で学ぶディープラーニングの理論と実装』, オライリー・ジャパン,2022.

[24] 野寺 隆志, 『楽々LATEX』, 共立出版株式会社,1996.

[25] 飯塚 修平, 『ウェブ最適化ではじめる機械学習』, オライリー・ジャパン,2020.

[26] 矢沢 久雄, 『基本情報技術者 らくらく突破 Python』, 技術評論社,2021.

[27] 伊藤 多一, 今津 儀充, 須藤 広大, など 『現場で使える! Python 深層強化学習入門 - 強化学習と深層学習による探索と制御』, 株式会社翔泳社,2025.

[28] Kirill Bobrov, 『なっとく！並列処理プログラミング』, 株式会社翔泳社,2025.

[29] 平井 有三, 『はじめてのパターン認識』, 森北出版株式会社,2023.

[30] 大用 倉智, 山田 孝子, 『作りながら丁寧に学ぶ Python プログラミング入門』, 関西学院大学出版会,2022.