

修士論文 概要書

遺伝的アルゴリズムと深層強化学習に基づく ジョブスケジューリング最適化問題

—PCB加工プロセスをモデルとして—

1 はじめに

本研究では、プリント基板 (PCB) 製造工程を対象に、同一の加工機械を複数回使用する生産プロセスのスケジューリング最適化に取り組んだ。遺伝的アルゴリズム (GA) と深層強化学習に基づく Deep Q-Network (DQN) を用い、各手法の性能を比較した。

2 生産管理問題のモデル

本研究では受注生産する工場の製造工程として受注型ジョブショップスケジューリング問題 (Job Shop Scheduler problem: JSSP) を扱う。この JSSP を制約付き最適化問題として定式化する。目標関数は:

$$\text{opt. } JSSP = \min \left\{ \max_s TJ_s \right\} \quad (1)$$

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。

最適化問題ではジョブセット (同一の締め切りを持つ注文書集合) に対し、セットに含まれる全製品種類の加工完了時刻の最大値を当該スケジューリング案の終了時刻とする。そして代替スケジューリング案の終了時刻から、最小終了時刻 (*makespan*) をもつスケジューリング案を最適案とする。

本論文の JSSP は、待機中の加工機械に対し、加工対象ジョブを割り当てる逐次意思決定問題とみなし、その意思決定過程を図 1 に示す。

3 実験用データ自動生成

JSSP の検討にはクライアント側から提供される注文書が必要だが、実データの入手は困難なため本

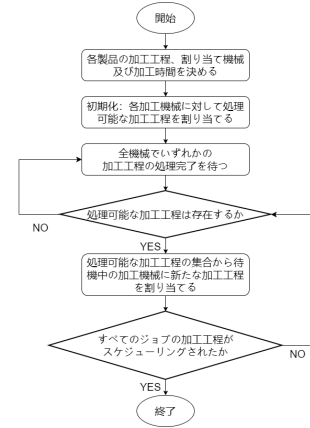


図 1: ジョブスケジューリング問題フローチャート

研究では先行研究 [?] の PCB 製造工程を参考に、製品種別ごとに加工機械、加工役割と生産手順を予め与え、ジョブセットに含まれる注文書数、注文書に記載される製品種別やロット数は確率的に変動させて生成した。

4 遺伝的アルゴリズムによる解の改善

ここでは JSSP に対し GA を用いた最適解探索手法を述べる。

各染色体は、全ジョブに属する加工工程を一度ずつ含む遺伝子列として表現する。各遺伝子は「ジョブ J_s の第 q 工程」とし、制約を満たすようスケジュールを構成する。各ジョブは処理可能なジョブからランダムに工程を選択配置して初期染色体を生成する。各染色体デコードで得られた総加工時間 *makespan* を適応度として評価する。トーナメント選択により最良個体を選択し、親染色体を決定して制約を保持するように PPS(Precedence Preservative Crossover) という交差手法で子個体を生成する。以上の進化計算を繰り返し、十分に収束した後に得られる染色体を最適解とする。最終解に対応するカントチャート

と世代ごとの *makespan* の折れ線グラフで視覚的に評価する.

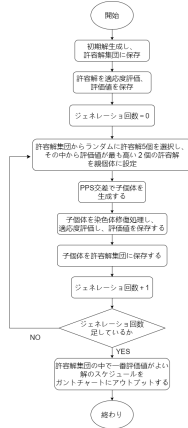


図 2: 遺伝的アルゴリズムフローチャート

5 深層強化学習による解の改善

ここでは価値関数に基づく深層強化学習手法 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, 各状態の行動選択を Q 値に基づいて選択する.

DQN にはメインネットとターゲットネットがあり, メインネットでは, 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 を決定に用いる. 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のパラメータ更新を行う. 一定ステップ間隔ごとにメインネットのパラメータをターゲットネットへコピーするハード更新を行う. 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を用いて計算を行う.

6 実験結果

DQN 報酬関数の構造を複合型報酬関数と *makespan* 中心型報酬関数にそれぞれ性能について GA と比較した. 実験結果により複合型報酬に含まれる要素が多岐にわたることで, 各要素間の影響が相互に干渉し, 学習信号にノイズが多く含まれる結果となった. そのため, 学習過程では一定の収束傾向が観測されるものの, 最終的に得られた方策は実際の JSSP に対して十分に有効な解を導出できていない場合が多かった. すなわち, 複合型報酬関数

は「学習しているように見えるが, 本質的な方策改善に結びつきにくい」という特性を有していると考えられる. *makespan* 中心型報酬関数を用いた場合, 報酬が C_{\max} の変化に直接対応するため, エージェントはスケジューリング全体を意識した意思決定戦略を比較的明確に学習できる. その結果, 多くの問題例において, GA と同等水準の解, あるいはそれを上回る解が得られた. 図 3 に 120 のジョブセットに対し GA と *makespan* 中心型報酬関数 DQN で得られた最適スケジューリングの *makespan* を示す. 多くのケースで DQN の *makespan* は GA と同程度 (比率 1.0 前後) で, 一部では GA を上回る結果も見られるが, ジョブセットにより性能が低下する場合も存在する.

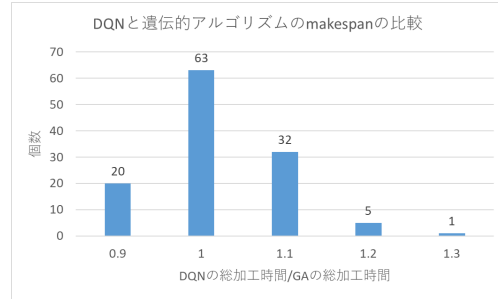


図 3: DQN と GA の *makespan* の比較

図 4 には *makespan* を横軸, 縦軸はジョブセット番号順で並べた. ここではジョブセットに含まれる注文の処理時間の分散を 3 段階で色分けし, GA と DQN の *makespan* 差を示す.

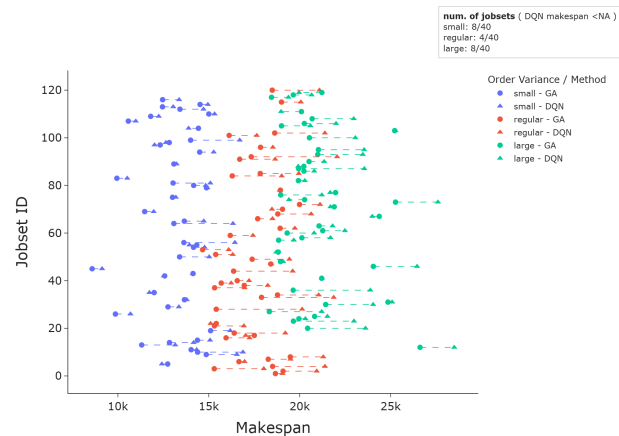


図 4: 同じ jobset により両手法の *makespan* 比較

GA と *makespan* 中心型報酬関数のほうが安定してスケジュール案を作成している.