卒業論文（2024 年度）

ジョブの投入順序の変更量を考慮した
再スケジューリング手法の提案

早稲田大学　創造理工学部

経営システム工学科

1X21C034-4 鬼頭拓海

指導教員　谷水義隆

要旨

製造業の生産現場では，作業の遅延や機械の故障，材料の供給遅延など，さまざまな予期せぬ外乱が頻繁に発生し，生産計画の遂行を大きく阻害することがある．これらの外乱は，生産性の低下や納期遅延，コスト増大といった企業競争力への悪影響を直接的にもたらすため，迅速かつ柔軟に対応できるスケジューリング手法の確立が急務となっている．外乱に応じて再スケジューリングを行うリアクティブスケジューリングは有効なスケジューリング手法であるが，頻繁に再スケジューリングを行うことによって，現場の混乱やリソース再配置コスト，原材料の再注文コストなどのスケジュール変更コストが増加することが知られている．本研究では，スケジュール変更コストを抑制するために元のスケジュールからの変更の度合いを抑制する，すなわちスケジュールの安定性を確保した再スケジューリングの手法を提案する．

ほとんどの既存研究では，元スケジュールと修正されたスケジュールの差を評価する指標として，作業開始時刻の変更量が用いられているが，実際の現場ではジョブの投入順序の変更も同様に重要な要因である．そこで本研究ではジョブの投入順序を順位として評価し，順位の変更量に着目した順位偏差という関数を提案する．さらに順位偏差に再スケジューリング時刻からの順位的距離に応じたペナルティ関数を乗算することによって順位ごとのそれぞれのジョブのスケジュール変更コストの変化量を表現する．

本研究では遺伝的アルゴリズム（GA）の目的関数に総所要時間と提案する安定性関数を組み込み，これらを重みパラメータ法で合成することによって二目的最適化を行う．

10 ジョブ 10 リソースのジョブショップスケジューリング問題で，作業遅延を想定した数値実験を行った結果，総所要時間をほとんど悪化させずに安定性関数値を大幅に抑制できることを確認した．さらに本研究で定義した安定性が確保された再スケジューリングが行われていることを確認した．

目次

1. 緒言

1.1 研究背景

製造業の生産現場では作業の遅延や機械の故障，材料の供給遅延など様々な予期せぬ外乱が頻繁に発生し，生産計画の遂行に重大な影響を与えることも多い[1]．これらの外乱は，生産効率の低下や納期遅延，コスト増加など，企業の競争力に直接的な悪影響を及ぼすため，迅速かつ柔軟に対応するスケジューリング手法の開発が求められている．

外乱に対応するためのスケジューリングのアプローチとしては大きく二つに分類される．一つは初期スケジュールを外乱に対して耐性の高い，すなわちロバスト性の高いものにするという静的アプローチである． 静的アプローチでは再スケジューリングすることは考慮せず，あらかじめ時間的余裕をもたせるなどの手法によって外乱が発生した際にスケジュール全体への影響ができる限り小さくなるような初期スケジュールをあらかじめ作成する[2]． しかし，外乱への耐久性を高めるほど，スケジュールの生産効率は下がってしまう． また機械の停止などの影響の大きな外乱が発生した際には初期スケジュール通りに生産を進めるのは困難であるという限界が存在する．

第二のアプローチは，外乱の発生に応じて生産の途中で再スケジューリングを行う動的アプローチである． 動的アプローチは外乱に合わせてリアルタイムにスケジュールを更新するため，柔軟に生産活動を継続することが可能である．動的アプローチは計画期間ごとに再スケジューリングを行うローリングスケジューリング，事前に生産スケジュールを持たずに逐次生産対象を割り当てるリアルタイムスケジューリング，外乱に応じて再スケジューリングを行うリアクティブスケジューリング等に分類される[3]． 谷水ら[4]は作業の遅延やオーダの不規則な追加に対して，動的に再スケジューリングを行うリアクティブスケジューリングの手法を提案した． しかし再スケジューリングを頻繁に行ってスケジュールを変更することによって現場の混乱やリソースの再配置コスト，原材料の再注文コスト等の多様なコストが発生するリスクがあることが知られている[1]． 本研究ではこれらの再スケジューリングをすることによって生じるコストやデメリットを総称してスケジュール変更コストと呼称する．スケジュール変更コストは製造現場にとっても企業にとっても望ましくないものである． スケジュール変更コストを小さくするためには元のスケジュールから大幅に変更しすぎないスケジュールを作成することが求められる．

再スケジューリングにおける修正したスケジュールと元のスケジュールの変更の度合いは普遍的に定義された用語は存在しないが Zhang ら[5]やいくつかの再スケジューリングに関する研究では Stability と呼称されており，本研究では安定性と呼称する．安定性に関する既存の研究はいくつか存在するが，ほとんどの研究では安定性の評価指標には作業の開始時刻の絶対値の差の総和を基本としたものが用いられている．この指標を本研究では開始時刻偏差と呼称する．

1.2 研究目的

本研究では，ジョブショップスケジューリング問題を問題状況として外乱が発生した際に安定性を確保しつつ再スケジューリングを行う手法を提案する．安定性の評価指標として従来の開始時刻偏差ではなく，リソースごとのジョブの投入順序に着目した順位偏差という指標を基本指標として用いる．この順位偏差を遺伝的アルゴリズム（GA）の目的関数に組み込むことで，安定性を維持しながらスケジュールの効率性を向上させる再スケジューリングを実現し，外乱に対して柔軟に対応可能なスケジューリング手法の確立を目指す．

1.3 本論文の構成

本論文の構成を以下に示す．第 2 章では既存の研究と課題点について説明し，本研究での安定性の評価について定義する． 第 3 章では本手法で用いる遺伝的アルゴリズムや再スケジューリング手法について記述する．第 4 章では計算機実験によって再スケジューリングを行い，その有効性を検証する．最後に第 5 章で結論と今後の課題について述べる．

2. 再スケジューリングにおける安定性

2.1 用語の定義

既存の研究を説明する前に，ジョブショップスケジューリング問題および再スケジューリングにおいて共通して使用される用語をここで定義する．

(1) ジョブ（Job）

ジョブとは，部品や製品などの処理が施される対象を指す．

(2) リソース（Resource）

リソースとは，作業を実行するために必要な加工設備や作業者等の資源を指す．リソースは同時に 1 つの作業のみを行うことができる．

(3) 作業（Operation）

作業とは，加工等のジョブに施される処理を指す． 特定のリソースを用いて，決められた処理時間で実行される． 作業は一度開始したら中断することができない．リソース i のジョブ j の作業は，作業の開始時刻$st_{i,j}$，作業の終了時刻$ft_{i,j}$，および処理時間$pt_{i,j}$の情報を持つ．これらは以下の関係式(2.1)が成り立つ．

$$ft_{i,j} = st_{i,j} + pt_{i,j} \tag{2.1}$$

(4) ガントチャート（Gantt Chart）

ガントチャートとは，リソースが行う作業の順序を視覚的に表した生産スケジュールを指す． 横軸が時間，縦軸がリソースを表す．

(5) 総所要時間（Makespan）

総所要時間とは，すべてのジョブを完了するために必要な時間を指す． スケジューリング問題において主要な効率性の指標の 1 つである． 総所要時間 MS は以下の式(2.2)で求めることができる．

$$MS = \max\left(ft_{i,j}\right) - \min\left(st_{i,j}\right) \tag{2.2}$$

(6) ジョブショップスケジューリング問題（Job Shop Scheduling Problem）

ジョブショップスケジューリング問題とは，製造業やサービス業等で見られる典型的なスケジューリング問題の 1 つで，目的関数を最小化または最大化するために，各リソースにおける各ジョブの加工順序を決定する問題である． 目的関数としては総所要時間や納期遅れ和等がよく用いられる． ジョブショップスケジューリング問題は NP 困難問題として知られており，規模の大きな場合は厳密な最適解を求めるのではなく近似解を求めることも多い． 10 ジョブ 10 リソースのジョブショップスケジューリング問題のガントチャートの例を図 2.1 に示す．

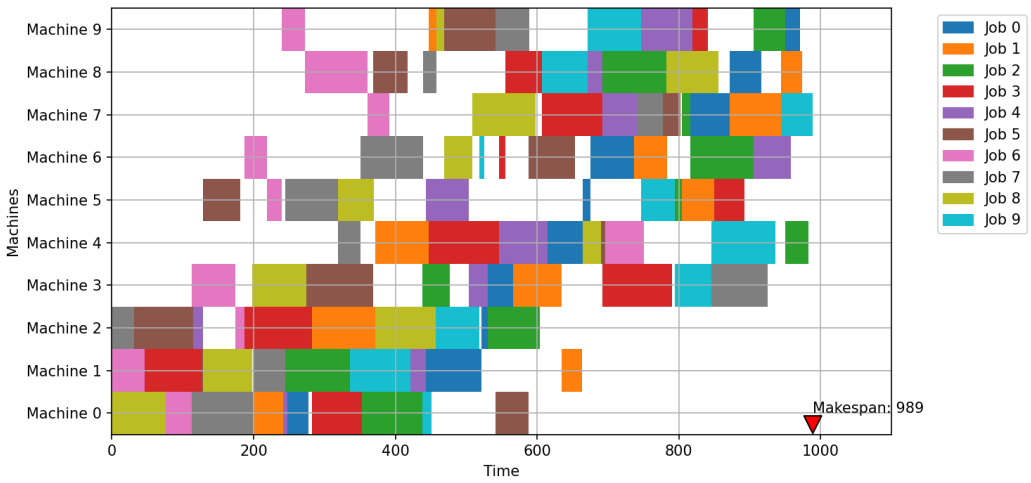

図 2.1 ジョブショップスケジューリング問題のガントチャートの例

(7) 効率性（Efficiency）

効率性とは，スケジューリング問題の評価指標として用いられる総所要時間や納期遅れ和等のスケジュールの効率を評価する指標を指す．

(8) 安定性（Stability）

安定性とは，再スケジューリングにおいて修正したスケジュールと元のスケジュールの変更の度合いを指す． 変更度合いについては普遍的に定義された用語は存在しないが，いくつかの既存研究では stability と呼ばれており，本研究ではこれ以降，安定性と呼称する． 安定性という用語はロバスト性を意味する場合にも使用されるが，本研究では変更度合いの意味で用いる．

2.2 安定性に関する既存の研究

スケジューリングの分野において，総所要時間や納期遅れ和等の効率性の指標は多くの先行研究が存在する．しかし再スケジューリングにおける安定性についての研究は少ない． スケジュール変更コストはスケジュールの変更度合いが大きいほど大きくなる． そのため，再スケジューリングにおいては安定性を高めることは効率性と同様に重要なことである[6]． いくつかの既存研究では安定性を評価する関数が提案されている． Alagöz ら[7]は安定性の評価関数として異なるリソースが割り当てられた作業の数を用いた．Calhoun ら[8]は変更する作業の個数を評価関数として用いている．

Wu ら [9] は 1 台のリソースにおける再スケジューリング問題において安定性の定義として式(2.3)の開始時刻偏差を提案した．

$$\mathrm{D}(S_p, S_q) = \sum_{j \in J} \left| st_j^p - st_j^q \right| \tag{2.3}$$

・S_p：初期スケジュール
・S_q：再スケジューリング後のスケジュール
・J: 再スケジューリング対象のジョブの集合
・st_j^p：S_pにおけるジョブ jの作業開始時刻
・st_j^q：S_qにおけるジョブ jの作業開始時刻

開始時刻偏差は元のスケジュールと修正されたスケジュールの各作業の開始時刻の絶対値の差の総和であり，これまでのほとんどの研究において安定性の指標の基本として用いられている．

Rangsaritratsamee ら [10] はジョブショップスケジューリング問題を対象として式(2.4)を安定性の評価関数として提案した．

$$\mathrm{D}(S_p, S_q) = \sum_{i \in M} \sum_{j \in J} \left[\left| st_{i,j}^p - st_{i,j}^q \right| + PF_{i,j} \right] \tag{2.4}$$

$$PF_{i,j} = \begin{cases} \dfrac{\alpha}{\left(st_{i,j}^p + st_{i,j}^q - 2t \right)^{\beta}}, & \mathrm{st}_{i,j}^{\mathrm{q}} < \mathrm{st}_{i,j}^{p} \\ 0, & \mathrm{st}_{i,j}^{\mathrm{q}} \geq \mathrm{st}_{i,j}^{p} \end{cases} \tag{2.5}$$

・M：再スケジューリング対象のリソースの集合
・$st_{i,j}^{p}$：S_pにおけるリソース i のジョブ j の作業開始時刻
・$st_{i,j}^{q}$：S_qにおけるリソース i のジョブ j の作業開始時刻
・$PF_{i,j}$：リソース i のジョブ j に対するペナルティ関数
・α：ペナルティ関数のスケールを調整する係数
・β：ペナルティの傾斜を調整する係数
・t：再スケジューリング時刻

Rangsaritratsamee らは再スケジューリング時刻からの時間的距離が近い作業ほどスケジュール変更コストが大きいという事実を指摘し，開始時刻が早まったジョブに対して開始時刻からの時間的距離に応じたペナルティを開始時刻偏差に加算した関数を安定性の指標として提案した． Rangsaritratsamee らは開始時刻が早まったジョブだけではなく，開始時刻が遅くなったジョブもスケジュール変更コストが生じることに言及はしていながらも，前者のほうがスケジュール変更コストが大きいため，開始時刻が早まったジョブのみにペナルティを与えている．

Pfeiffer ら[11]はジョブショップスケジューリング問題を対象として式(2.6)を安定性の評価関数として提案した．

$$\mathrm{D}(S_p, S_q) = \sum_{i \in M} \sum_{j \in J} \left[\left| \mathrm{st}_{i,j}^{p} - \mathrm{st}_{i,j}^{q} \right| + PF_{i,j} \right] \tag{2.6}$$

$$PF_{i,j} = \begin{cases} \dfrac{\alpha}{\left(st_{i,j}^{p} - t \right)^{\beta}}, & \mathrm{st}_{i,j}^{\mathrm{q}} \neq \mathrm{st}_{i,j}^{p} \\ 0, & \mathrm{st}_{i,j}^{\mathrm{q}} = \mathrm{st}_{i,j}^{p} \end{cases} \tag{2.7}$$

Pfeiffer らは時間的距離に応じたペナルティを，開始時刻が早まったジョブに対しても開始時刻が遅くなったジョブに対しても等しく，元のスケジュールの作業の開始時刻に対して与えている．

Ayoub ら [1] は各作業の重要度によって重み付けした係数を開始時刻偏差に乗算した式(2.8)を安定性の指標として提案した．

$$\mathrm{D}(S_p, S_q) = \sum_{i \in M} \sum_{j \in J} \omega_{i,j} \left| \mathrm{st}_{i,j}^{p} - \mathrm{st}_{i,j}^{q} \right| \tag{2.8}$$

・$\omega_{i,j}$：リソース i のジョブ j の重要度に対する重み値

2.3 既存研究の課題

ほとんどの既存の研究は開始時刻偏差を基準にした関数を安定性の指標として用いている．しかし実際の作業現場ではジョブの開始時刻の変化だけではなく，ジョブの投入順序の違いも同様にスケジュール変更コストの重要な要因となる．開始時刻偏差は時間的な影響の大きい外乱が発生した場合，例えば機械の故障などが発生した際に，投入順序を正しく評価できない場合があることを実験によって確認した．

そこで本研究では安定性の評価指標として開始時刻偏差ではなくジョブの投入順序に着目した，順位偏差と呼称する新たな評価関数を提案する．また Rangsaritratsamee らが指摘した再スケジューリング時刻からの時間的距離が近い作業ほどスケジュール変更コストが大きいという点も表現した，新たな安定性の評価指標を作成する．

3. 安定性を考慮したジョブショップスケジューリング問題の再スケジューリング手法

3.1 本研究における安定性の評価関数の定義

本研究では安定性の評価関数として順位偏差を基本とした関数を提案する．順位偏差ではジョブの投入順序を順位として評価する．再スケジューリングにおいては再スケジューリング対象のジョブのみを対象として順位付けを行う．順位付けの例を図3.1に示す．

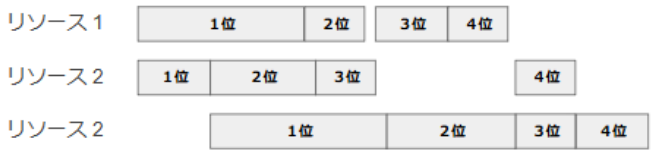

図 3.1 ジョブの順位付けの例

リソースごとにジョブの投入順序を順位で表している．順位付けは各リソースごとに投入順序が早い順に降順で行う．順位偏差は式(3.1)として定式化される．

$$\mathrm{D}(S_p, S_q) = \sum_{i \in M} \sum_{j \in J} \left| r_{i,j}^{p} - r_{i,j}^{q} \right| \tag{3.1}$$

・$r_{i,j}^{p}$：S_pにおける機械 i のジョブ j の順位
・$r_{i,j}^{q}$：S_qにおける機械 i のジョブ j の順位

式(3.1)は元のスケジュール S_p と再スケジューリング後のスケジュール S_q の各ジョブの順位の変更距離の総和を表している．例えばリソース i のジョブ j の順位が 1 位から 5 位になった場合，順位の変更距離は 4 となる．

本研究では，Rangsaritratsamee らが指摘した再スケジューリング時刻からの時間的距離が近い作業ほどスケジュール変更コストが大きいという点を考慮して，順位偏差にペナルティ関数$\omega_{i,j}$を導入した関数である式(3.2)を安定性の評価指標として用いる．

$$\mathrm{D}(S_p, S_q) = \sum_{i \in M} \sum_{j \in J} \omega_{i,j} \left| r_{i,j}^{p} - r_{i,j}^{q} \right| \tag{3.2}$$

$$\omega_{i,j} = \frac{1}{\left(r_{i,j}^{q}\right)^{\beta}} \tag{3.3}$$

・$\omega_{i,j}$：ジョブの順位に対するペナルティ関数

・β：ペナルティの傾斜を調整する係数

本関数の目的は順位変更量の総和の抑制と，その中でも特に投入順序の早いジョブの順位変更量の抑制の２つである．すなわちこの２点の抑制が本研究における安定性の定義と同義である．前者は順位偏差，後者はペナルティ関数$\omega_{i,j}$によって表現されている．

ペナルティ関数は投入順序が早いジョブほど大きなペナルティを与える関数である．１０ジョブのリソースにおける各順位のジョブのペナルティ値の例を図 3.2 に示す．$\beta = 1.25$とする．

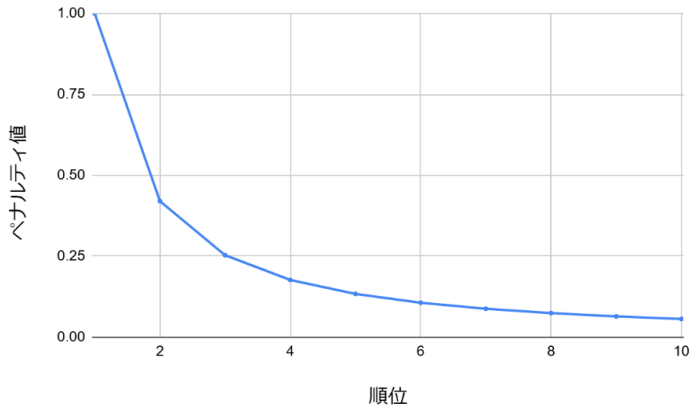

図 3.2 順位ごとのペナルティ値の例

スケジュール変更コストをジョブ単位でみると，再スケジューリング時刻からの時間的距離が近いほど，大きくなる．既存研究ではこの点を表現するためにペナルティ関数としてべき乗関数を用いている．本研究でもペナルティ関数はべき乗関数として表現する．開始時刻偏差を用いている既存研究ではジョブの作業開始時刻に対してペナルティを与えていたが，本研究ではジョブの順位に対してペナルティを与える．

本研究のペナルティ関数が既存研究と比較して特徴としている点は大きく２つある．

１つ目は再スケジューリングによってジョブが早くなるか遅くなるかによってペナルティの大きさに差を付けている点である．ジョブの開始時間が遅くなることによって生じるスケジュール変更コストには在庫や材料や部品の保管コストがあり，ジョブの開始時間が早くなることによって生じるスケジュール変更コストにはリソースの輸送コストや原材料の不足のリスクなどがある． Rangsaritratsameeらはジョブの開始時刻が遅くなることに比べてジョブの開始時刻が早くなる方がスケジュール変更コストが大きいという点に言及し，開始時刻が早まったジョブのみにペナルティを与えている．またPfeifferらは両者に等しくペナルティを与えており，両者に差をつけてペナルティを与えている既存研究はない．これはスケジュール変更コストが様々な要因が絡んだものであり，なおかつそれぞれの作業現場によって異なるため，正確に定式化できないことが原因であると考えられる．

本研究では，ペナルティを元のスケジュールにおける順位ではなく再スケジューリング後のスケジュールの順位に与えることによって両者の差を表現している．比較として，Pfeiffer らのペナルティ関数である式(3.3)はペナルティを元のスケジュールの作業開始時刻に与えている．

$$PF_{i,j} = \begin{cases} \dfrac{\alpha}{\left(st^{p}_{i,j} - t\right)^{\beta}}, & \mathrm{st}^{\mathrm{q}}_{i,j} \neq \mathrm{st}^{p}_{i,j} \\ 0, & \mathrm{st}^{\mathrm{q}}_{i,j} = \mathrm{st}^{p}_{i,j} \end{cases} \tag{3.3}$$

この場合，各ジョブのペナルティの値は再スケジューリングによって作業開始時刻が早くなるか遅くなるかに関わらず等しくなる．

一方、本研究では再スケジューリング後のジョブの順位に対してペナルティを与えている．それによって開始時刻が早まったジョブのほうが開始時刻が遅くなったジョブよりもペナルティを大きくすることができる．具体例として再スケジューリングによってリソース i のジョブ j の投入順序が３位分早くなる，もしくは遅くなる場合のペナルティ関数$PF_{i,j}$の値を表 3.1 に示す．$\beta = 1.25$とする．

表 3.1 ジョブ j の順位が 3 位分移動する場合の $PF_{i,j}$ の値

	$PF_{i,j}$ の値
投入順序が早くなる	0.420
投入順序が遅くなる	0.074

このようにジョブの投入順序が早くなる方が遅くなる方に比べて 6 倍ほど大きなペナルティを課されている．同様にジョブ j の順位が 3 位分移動した場合のリソース i 全体の安定性関数 $\mathrm{D}(S_p, S_q)$ の値を表 3.2 に示す．$\beta = 1.25$ とする．

表 3.2 ジョブ j の順位が 3 位分移動する場合の $\mathrm{D}(S_p, S_q)$ の値

	$\mathrm{D}(S_p, S_q)$ の値
投入順序が早くなる	1.825
投入順序が遅くなる	0.551

リソース全体の関数値で見るとジョブ j が移動したことによって他のジョブも 1 つずつ移動する．リソース全体の関数値も投入順序が早くなる場合の方が 3 倍以上の値となっている．安定性関数値は個体ごとの安定性を相対的に評価する尺度であるため，指標の絶対的な値自体は意味を持たない．

2 つ目のペナルティ関数の特徴は順位偏差に加算するのではなく，乗算している点だ．Rangsaritratsamee らや Pfeiffer らの研究では開始時刻偏差にペナルティ関数を加算している．しかし加算ではジョブが移動する時間的距離とペナルティ値の両者を同時に考慮することができない．各ジョブに与えられるペナルティ値は開始時刻の変更度合いの大きさに関わらず一定となる．また開始時刻偏差とペナルティ関数のスケールをあわせるための変数 α を導入することが求められる．本研究では順位偏差とペナルティ関数を乗算することによってこれらの問題を解消している．

3.2 GA によるジョブショップスケジューリング問題の再スケジューリング手法

本研究ではジョブショップスケジューリング問題を対象として，GA によって再スケジューリングを行う．外乱の中でも作業の終了時刻の遅延に焦点を当てる． 遺伝的アルゴリズム(GA)とは生物の進化の仕組みを模倣した近似解の探索手法である．GA で

は解候補を遺伝子として表現したものを個体と呼ぶ． 個体群を遺伝的操作によって交配させて，適応度の低い個体を淘汰しながら世代交代を行っていくことによって適応度の高い個体，すなわち目的関数値のよい解を探索する．GA は広範囲の解空間を探索できるため，ジョブショップスケジューリング問題のような膨大な解候補がある問題に適している．

本研究では植村ら[12]が使用した手法を利用して再スケジューリングを行う．GA を用いて再スケジューリングを行うプロセスを図 3.3 に示す．

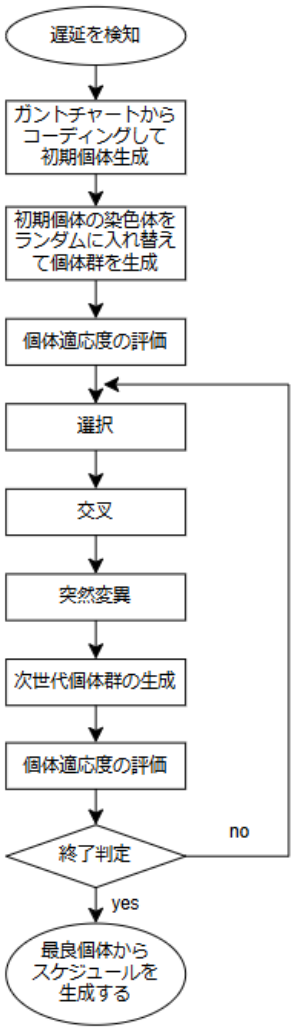

図 3.3 再スケジューリングのプロセス

詳細なステップを以下に示す．

Step1 再スケジューリング対象の作業の決定

遅延したジョブの終了時刻x分とし，計算時間をΔt分とって$x+\Delta t$分を再スケジューリング時刻とする．その時点でまだ始まっていない作業を再スケジューリング対象とする．再スケジューリング対象の作業の条件式は式(3.4)となる．

$$\mathrm{st}_{\mathrm{i,j}}^{\mathrm{p'}} \geq x + \Delta t \tag{3.4}$$

・$st_{i,j}^{p}$：遅延が発生したスケジュール$\mathrm{S_p}'$におけるリソース i のジョブ j の開始時刻

Step2 初期個体群の生成

Step1 で決定した作業群から初期個体を生成する．生産スケジュールを遺伝子として表現して個体に変換することをコーディングという． ジョブショップスケジューリング問題のコーディング手法はいくつか研究されているが，本研究では Giffler-Thompson Algorithm（GT 法）を用いる．コーディングの例を図 3.4 に示す．

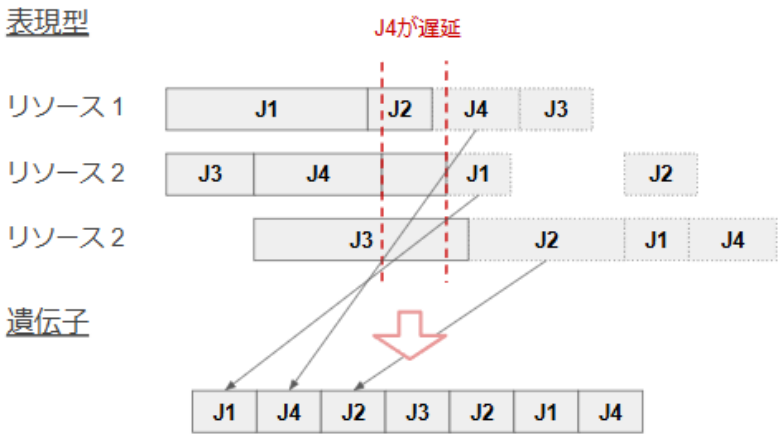

図 3.4 再スケジューリングにおけるコーディングの例

GT 法では生産スケジュールのガントチャートから，機械に関係なく開始時刻が早いものから順に一列にジョブを並べたものを遺伝子として扱う． 開始時刻が同刻の作業が存在する場合は，機械番号が小さい順に並べる．GT 法は遺伝的操作を行っても致死遺伝子が発生しないという長所がある一方，遺伝子型と表現型が一対一の関係ではないという短所がある．

初期個体の染色体をランダムに並び替えて M 個の個体を生成し，初期個体群とする．

Step3 初期個体群の適応度の評価

初期個体群の適応度を評価するためにそれぞれの個体から生産スケジュールを作成する．コーディングされた遺伝子型を表現型である生産スケジュールに変換することをデコーディングという．デコーディングの例を図 3.5 に示す．

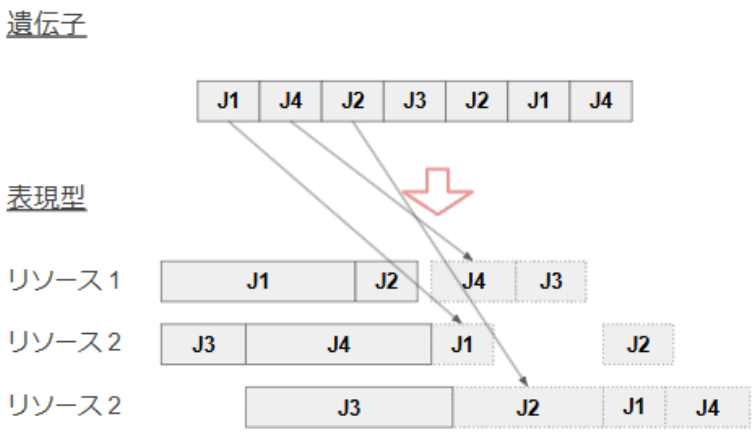

図 3.5 再スケジューリングにおけるデコーディングの例

再スケジューリングにおけるデコーディングでは既に完了している作業のガントチャートに遺伝子の先頭のジョブから再割り付けを行う．生成されたガントチャートから目的関数値を取得し，それぞれの個体に情報を保存する．

Step4 次世代の個体群の生成

初期個体群に対して遺伝的操作を施し，次世代個体群を生成する． 本研究では遺伝的操作として，選択手法にはトーナメント選択，交差手法には Partially Mapped Crossover（PMX），突然変異手法には逆位を用いている．

トーナメント選択とは個体群から任意の数の個体を選択してその中で目的関数値のよいものを交叉の対象として選択する手法である．個体群サイズ M 個の個体を選択する．

PMX とは致死遺伝子が発生しない 2 点交叉法である． 遺伝子の中で交叉域をランダムに指定する． その域中で対の親遺伝子内で自身と同じ位置にある染色体と同じジョブの染色体を自身の中から選択して交換することを繰り返す． PMX の例を図 3.6 に示

す.

図 3.6 PMX の例

逆位とは交叉域をランダムに指定して，範囲内の遺伝子の順序を逆転させる突然変異手法である． 突然変異は個体群が局所解に収束し，探索範囲が限定されることを防ぐ目的で行われる． 逆位の例を図 3.7 に示す.

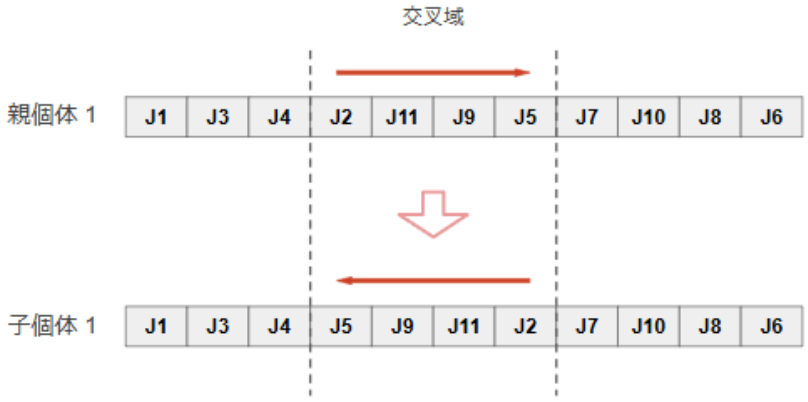

図 3.7 逆位の例

交叉と突然変異は任意の確率に従って行われる． また本研究ではエリート選択を行っている． エリート選択とは前の世代の個体群の中で最も目的関数値がよい個体を必ず次の世代に残す手法である.

Step5 個体適応度の評価

生成された次世代の個体群のそれぞれの個体の適応度を求める．

Step6 終了条件の判定

GA の終了条件を満たしていなければ Step4 に戻り，さらに次世代の個体群を生成する．終了条件を満たしていれば GA を終了し，個体群の中で最も適応度のよい個体をデコーディングしたものを生産スケジュールとする．本研究では終了条件として世代数を指定している．

3.3 目的関数の定義

安定性を担保しながら効率性を改善する再スケジューリングを行うために，安定性の評価関数と効率性の評価関数の二目的最適化を行う．効率性関数には最も一般的である総所要時間を，安定性関数には式(3.2)を用いる．安定性関数と総所要時間を式(3.4)の重みパラメータ法によって合成した関数を GA の目的関数として最小化を行う．

$$\lambda \times \mathrm{D}\left(S_{\mathrm{p}}, S_{\mathrm{q}}\right) + (1-\lambda) \times MS \tag{3.4}$$

・λ：安定性関数の重み

重みパラメータ法とは複数の関数に重み係数を付けて加重平均を取って合成関数を構築する手法である． その際に各目的関数のスケールが異なるとスケールの大きい目的関数の影響力が大きくなってしまうため，式(3.5)を用いて min-max 正規化を行う．

$$O_i' = \frac{O_i - O_{i,min}}{O_{i,max} - O_{i,min}} \tag{3.5}$$

・O_i：目的関数 i の値
・$O_{i,min}$：目的関数 i の最小値
・$O_{i,max}$ ：目的関数 i の最大値
・O_i'：正規化後の目的関数 i の値

最大値と最小値は GA で各世代の適応度を評価する際に更新していく．

4. 計算機実験

4.1 実験条件

本研究では 10 ジョブ，10 リソースのジョブショップスケジューリング問題を扱う．初期スケジュールとして総所要時間が 1019 分のスケジュールを用意し，その中で Machine 1 のジョブ 8 の作業終了時刻を 318 分から 378 分に 60 分遅延させた状況で再スケジューリングを行う． Δt=1，β=1.25 とする．初期スケジュールと再スケジューリングが行われる際のガントチャートを以下に示す．

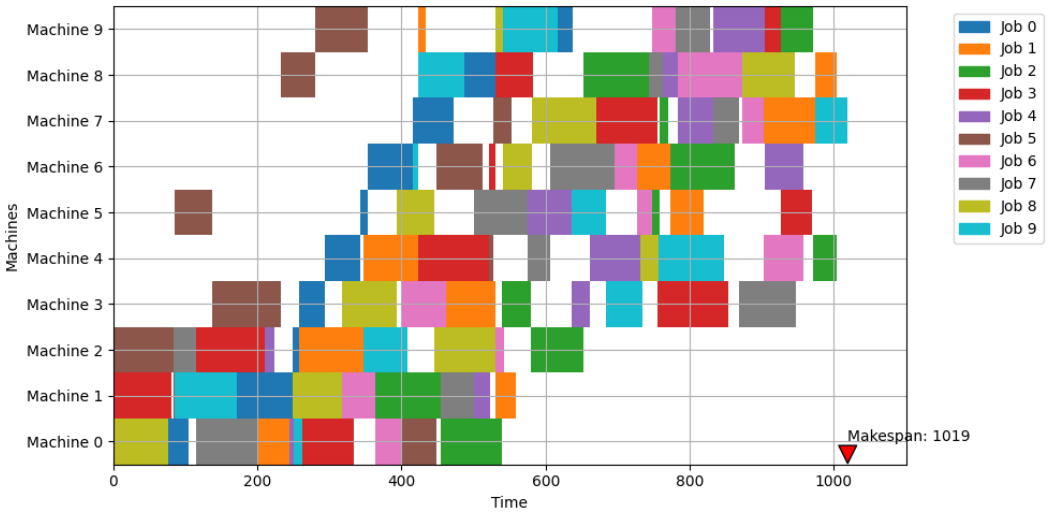

図 4.1 初期スケジュールのガントチャート

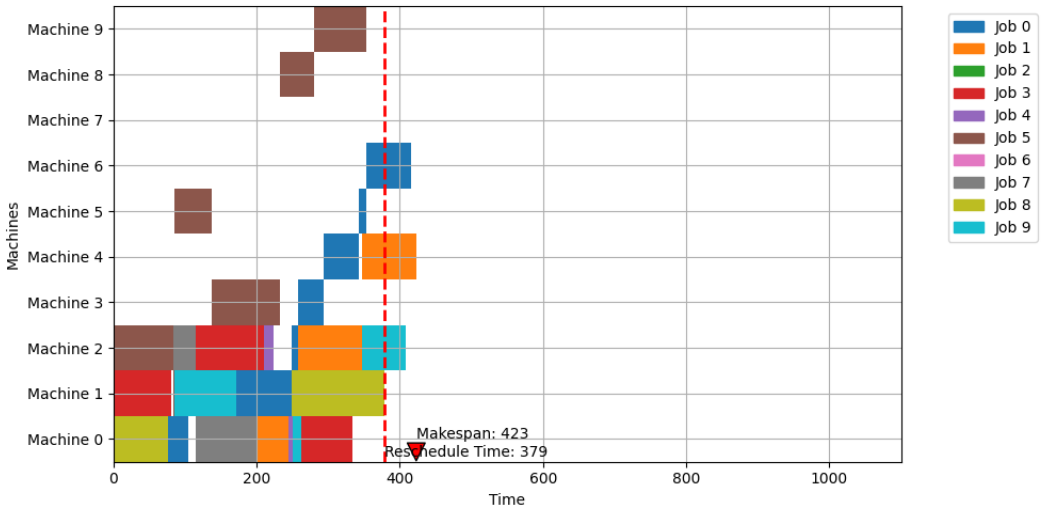

図 4.2 遅延が発生した瞬間のガントチャート

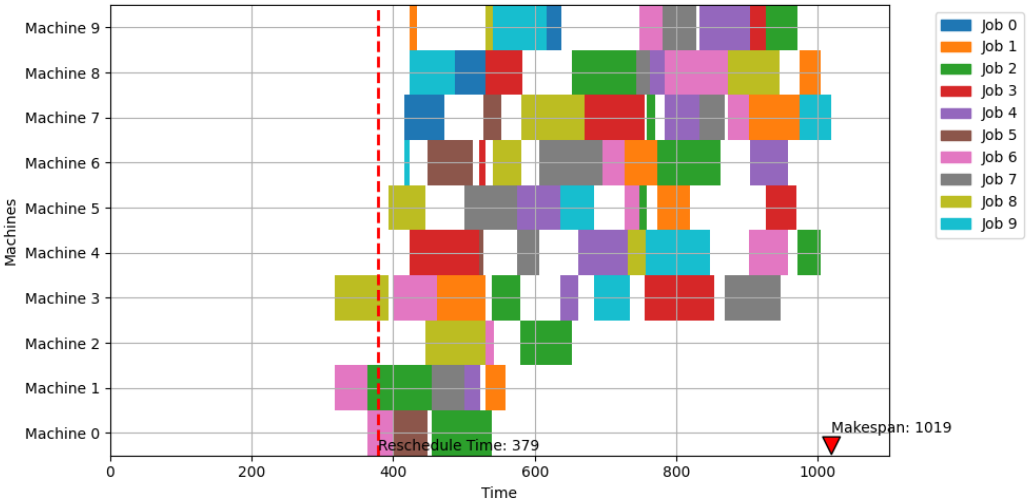

図 4.3 再スケジューリング対象のジョブのガントチャート

再スケジューリング時刻の 379 分の時点でまだ作業が始まっていないジョブをリスケジューリングの対象として，コーディングを行い，再スケジューリングする．

GA の手法とパラメータは表 4.1 にまとめている．

表 4.1 GA の手法とパラメータ

	世代数	集団数	選択	交叉	突然変異
GA の手法	-	-	トーナメント	PMX	逆位
			4 個体を抽出		
パラメータ	100	50	したうち 2 個	0.85	0.1
			体を選択		

本研究では 2 つの実験を行う． 実験 1 では再スケジューリングを行った際に，重み λ によって総所要時間と安定性関数値がどのように推移するのかを検証する． 実験 2 では実際に安定性が確保できているのかを検証する． そのためにジョブの順位変更量の総和と順位の高いジョブの順位変更量の総和が小さくなっているのかを目的関数ごとの比較実験によって確認する．

実験環境は以下に示す．

CPU：AMD Ryzen 5 7530U with Radeon Graphics 2.00 GHz

メモリ：16.00GB

OS：Windows 11 Home

モデリング言語：Python

4.2 実験1：λによる総所要時間と安定性関数値の推移

実験 1 では安定性関数と総所要時間の値が重みλを変化させた際にどのように推移するかを検証する．結果を表 4.2 と図 4.4 に示す．

表 4.2 総所要時間と安定性関数値の平均値

λ	総所要時間（分）	安定性関数値	解が初期解と同じスケジュールになる確率
0	1058.62	10.4425	0.11
0.0125	1061.99	7.4934	0.26
0.025	1063.37	5.8951	0.33
0.05	1064.59	5.5646	0.37
0.1	1066.02	4.3258	0.48
0.15	1070.42	2.3782	0.68
0.2	1076.09	0.81191	0.88
0.25	1078.27	0.3315	0.94
0.3	1079	0	1

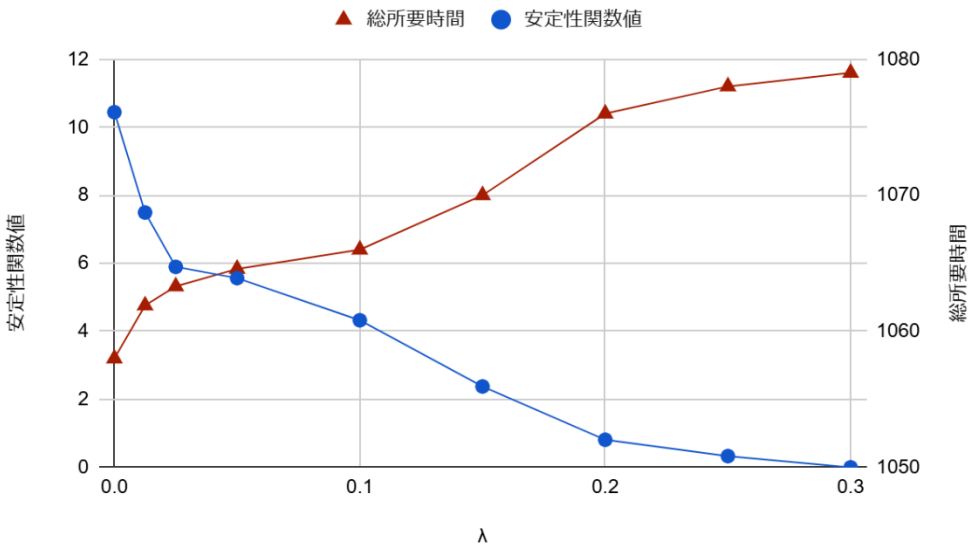

図 4.4 総所要時間と安定性関数値の平均値

各λの値は 100 回 GA を行った平均値である． この結果を見るとλを大きくするほど，総所要時間は長くなり，安定性関数値は小さくなっていくことがわかる．

表 4.2 の解が初期解と同じスケジュールになる確率というのは GA を行った結果，最適解がジョブの投入順序を変更しないという解が得られる確率のことである． この解が求まること自体は，初期スケジュールがそもそも総所要時間最小化を目的関数として作成されていることから考えると自然な結果である． 解が初期解になる確率はλを大きくするごとに高くなり，λが 3 以上になると 1 となる． 上記のデータは平均値であるため，初期解と同じスケジュールになる解の個数に大きく影響を受けた結果となってしまっている．そのため解が初期解と同じスケジュールを除いたデータを表 4.3 と図 4.5 に示す．

表 4.3 解が初期解と同じスケジュールの場合を除いた
総所要時間と安定性関数値の平均値

λ	総所要時間（分）	安定性関数値	解の個数（個）
0	1055.84	11.7188	89
0.0125	1055.97	10.0911	74
0.025	1055.68	8.756	67
0.05	1056.25	8.7816	63
0.1	1055.46	8.2435	52
0.15	1053.62	7.2787	32
0.2	1056.5	6.3079	12
0.25	1066.3	4.784	6

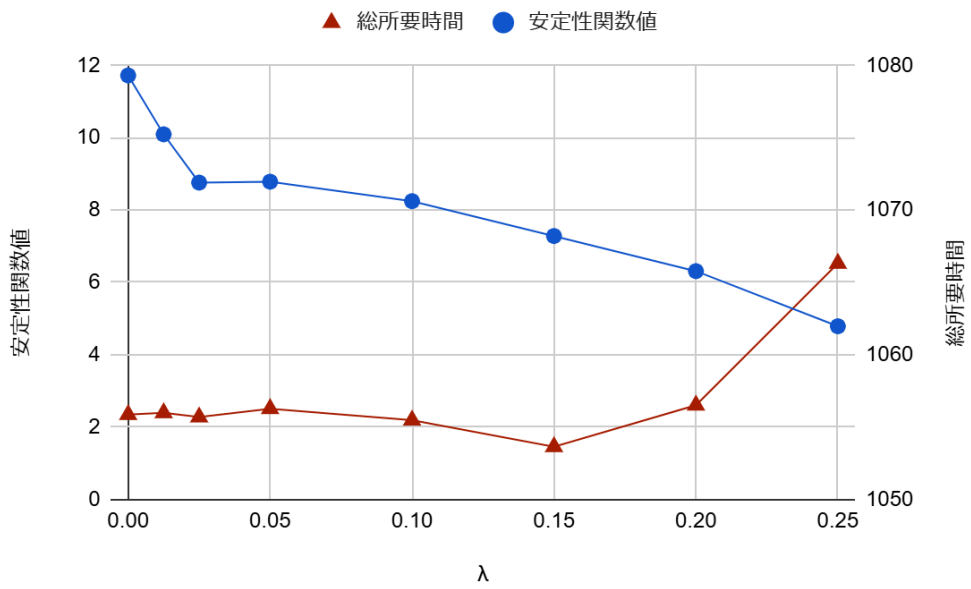

図 4.5 解が初期解と同じスケジュールの場合を除いた
総所要時間と安定性関数値の平均値

こちらの結果でも λ を大きくするほど安定性関数値の値が小さくなっていることが確認できる． 総所要時間に関しては λ=0.25 の場合を除くと， λ を大きくしても値がほとんど変化していないことがわかる．

4.3 実験2:ジョブの順位変更量についての検証

実験１では本手法によって安定性関数値が改善されていることを確認したが，実験２ではジョブの順位変更量の総和と投入順序の早いジョブの順位変更量の総和が実際に小さくなっているのかを検証するためにいくつかの指標を目的関数ごとに比較する．結果を表 4.4，表 4.5，表 4.6 と図 4.5 に示す．

表 4.4 目的関数ごとの各指標の値

目的関数	総所要時間（分）	安定性関数値	解の個数（個）	順位変更量の総和	1ジョブあたりの順位変更量の平均
総所要時間のみ	1055.69	10.0970	83	38.9879	1.3277
総所要時間+順位偏差	1053.51	7.2733	27	21.4814	1.3216
総所要時間+安定性関数	1056.00	6.7692	15	20.3636	1.3297

表 4.5 目的関数ごとの各順位における 1 ジョブあたりの順位変更量の平均

目的関数	順位ごとの１ジョブあたりの順位変更量の平均				
	１位	２位	３位	４位	５位
総所要時間のみ	0.2662	0.6096	0.7433	1.0421	0.6129
総所要時間+順位偏差	0.2	0.5555	0.6592	0.8611	0.0185
総所要時間+安定性関数	0.1818	0.5181	0.5818	0.875	0.0227

表 4.6 目的関数ごとの各順位における 1 ジョブあたりの順位変更量の平均

目的関数	順位ごとの１ジョブあたりの順位変更量の平均				
	６位	７位	８位	９位	１０位
総所要時間のみ	0.506	0.5111	0.1858	0.2138	0.2771
総所要時間+順位偏差	0.0317	0.0105	0	0	0
総所要時間+安定性関数	0.0389	0.0129	0	0	0

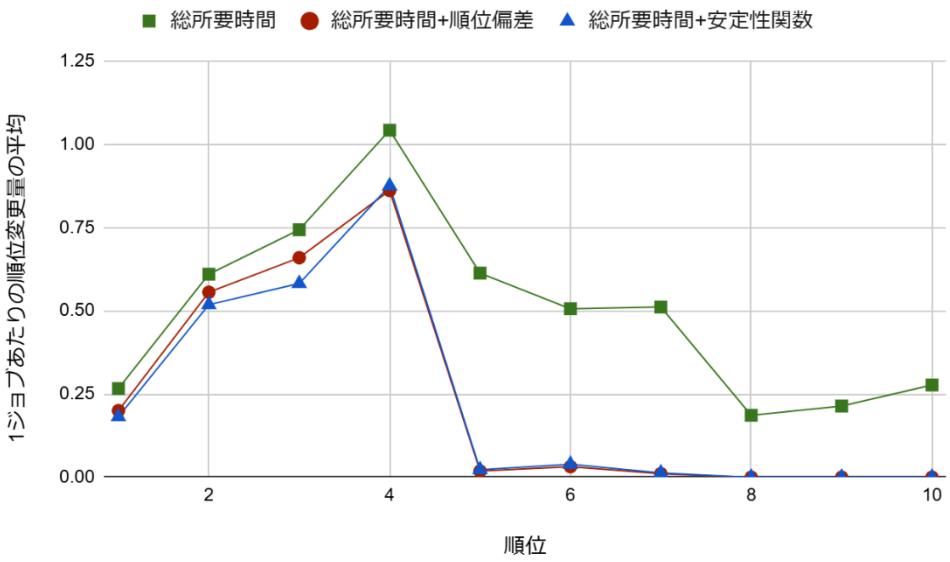

図 4.5 目的関数ごとの各順位における 1 ジョブあたりの順位変更量の平均の比較

値は実験 1 と同様に 100 回 GA を行ったうち，解が初期解と同じスケジュールの場合を除いた結果の平均値である． $\lambda = 0.2$ としている．

表 4.4 を見ると，目的関数に安定性関数，もしくは順位偏差を入れることによっての順位変更量の総和を約半分に抑えることができていることがわかる． また 1 ジョブあたりの順位変更量の平均に関しては安定性関数を入れる前後でほとんど変化がないことがわかる． 1 ジョブあたりの順位変更量は目的関数が総所要時間のみの場合でも 1.3277 と小さい値になっており，総所要時間最小化によって作成された初期スケジュールは既にジョブの投入順序が整っているので再スケジューリングをしたとしても大きく作業順序を入れ替える必要がないことが要因であると推測できる．

次に表 4.5，表 4.6 と図 4.5 の目的関数ごとの各順位における 1 ジョブあたりの順位変更量の平均について見ていく． このデータを見ると順位ごとに大きく変更量の差があることがわかるが，これは交叉方法の PMX の特性上，端の染色体が選択されづらい，つまり投入順序が早いジョブと投入順序が遅いジョブは変更されづらいという点が反映されていると考えられるため，順位ごとの比較はあまり意味をなさない． 順位ごとに各目的関数の値を比較する．

全順位において目的関数が総所要時間のみのときに比べて目的関数に順位偏差か安定性関数を組み込んでいる場合のほうが順位変更量が小さくなっている． その中でも投入順序が早いジョブは変更量の小さくなる幅が大きい． 目的関数に安定性関数と順

位偏差を入れている場合は，５位以降はほとんど０となっている． これは初期スケジュールがそもそも総所要時間最適化を目的関数として作成しているため，外乱が発生した箇所から遠い，順位の低いジョブは既にジョブの投入順序が整っていて無駄に入れ替えても総所要時間は大きく短くはならないためであるということが推測される．

投入順序が早いジョブに関しては，目的関数が総所要時間+順位偏差と総所要時間+安定性関数の場合を比べると若干量ではあるが総所要時間＋安定性関数のほうが順位変更量を抑えられていることがわかる． これは安定性関数内の順位ペナルティによって投入順序が早いジョブの変更量が抑えられているということである． 効果が若干量となってしまっている原因としては，外乱が発生した際には外乱箇所の近辺，すなわち投入順序が早いジョブを入れ替えないと総所要時間が大きく改善しないことが考えられる． 投入順序が早いジョブを入れ替えない場合は，安定性関数値を悪化させる数値的メリットが有るほど総所要時間が改善しないため初期解と同じスケジュールが最適解が求まっていると考えられる．

4.4 実験結果のまとめ

実験１では再スケジューリングを行った際に，重みλの値を変化させた際の総所要時間と安定性関数値の推移を示した．実験２では実際に安定性が確保できているのかを検証した． 実験１ではλを大きくすることによって安定性関数値を改善することができ，またその際に総所要時間はあまり悪化しないということが確認できた． 実験２では安定性関数を用いることによって順位変更量の総和の抑制と，その中でも投入順序の早いジョブの順位変更量の抑制が達成できている，つまり安定性を確保できているということが確認できた．

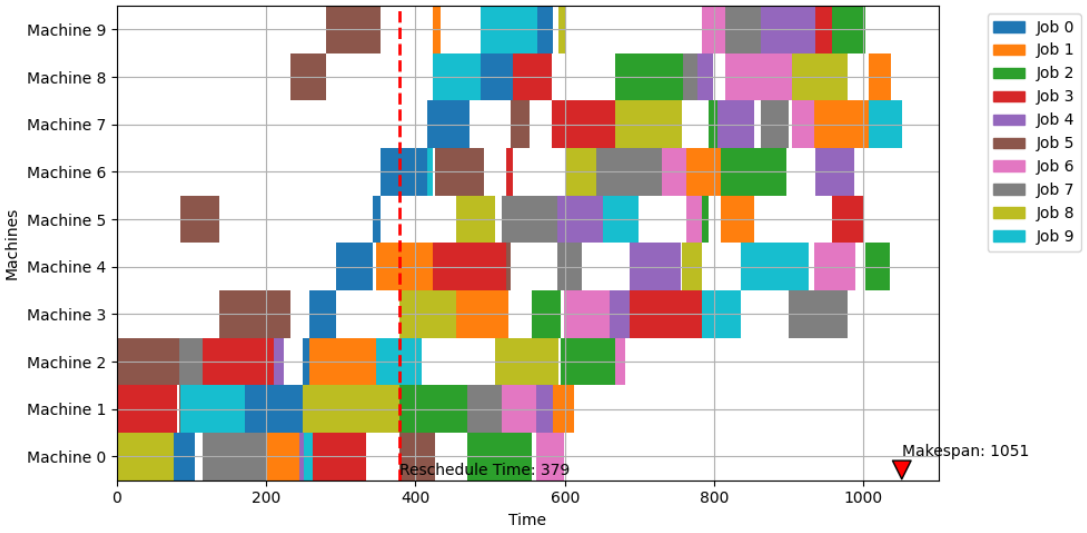

図 4.6 再スケジューリング後のガントチャート

5. 結言

本研究ではジョブショップスケジューリング問題の生産現場を問題状況として，作業終了時刻の遅延が発生した際に今回新たに提案した安定性関数と総所要時間の二目的最適化を GA によって求解することによって，安定性を担保しながら再スケジューリングを行う手法を提案した． 実験結果より，安定性を担保できていること，また安定性のと効率性のトレードオフが制御可能であることを示した．

実験によって明らかになった課題として初期解と同じスケジュールの解が求まる確率が高いということが挙げられる． 実際の現場では再スケジューリングの結果，ジョブの投入順序を変更しないという結果は問題ないが，再スケジューリングの手法を提案するという本研究の趣旨を満たさない結果であるため，初期解と同じスケジュールが求まる確率を下げ，よりよい解を探索できるようにする必要がある． 今回の実験では遺伝的操作として PMX や逆位など汎用的な手法を用いていたがより効果的に解の探索をするためには本手法にあった遺伝的操作を開発する必要があると考えられる． 効果的な解の探索を実現することによって今回の結果よりも良い解が得られるだけでなく、初期解と同じスケジュールが求まる確率が下がる可能性がある．

本実験では 10×10 のジョブショップスケジューリング問題のある一箇所で遅延を発生させた場合のみに対して実験を行ったが，さらに大規模なジョブショップスケジューリング問題への対応や，作業の終了時刻の遅延以外の外乱での再スケジューリング性能の検証も行う必要がある． また確率的に外乱の発生する作業現場に対して，再スケジューリングシステムがどのように挙動するのかをシミュレーションによって検証することも必要である。

参考文献

[1] Tighazoui, A., Sauvey, C., & Sauer, N. (2021). Predictive-reactive strategy for identical parallel machine rescheduling. Computers & Operations Research, 134, 105372.

[2] 森澤和子, 平林直樹. (2021). 不確実な生産環境下における遅 延リスク回避のためのロバストスケジューリング法の開発. https://kaken.nii.ac.jp/grant/KAKENHI-PROJECT-18K04614/

[3] 貝原俊也, 谷水義隆, 西 竜志, 企業間の戦略的提携. (2011). マルチエージェント交渉による次世代 SCM. サプライチェーンマネジメント講座 4, 朝倉書店.

[4] 谷水義隆, 阪口龍彦, 杉村延広. (2003). 遺伝的アルゴリズムを用いたリアクティブスケジューリング, 日本機械学会論文集 (C 編), 69 (685), 2458-2463.

[5] Zhang, L., Gao, L., & Li, X. (2013). A hybrid genetic algorithm and tabu search for a multi-objective dynamic job shop scheduling problem. International Journal of Production Research, 51(12), 3516-3531.

[6] Khodke, P. M., & Bhongade, A. S. (2013). Real-time scheduling in manufacturing system with machining and assembly operations: a state of art. *International Journal of Production Research, 51*(16), 4966-4978.

[7] Alagöz, O., & Azizoglu, M. (2003). Rescheduling of identical parallel machines under machine eligibility constraints. *European Journal of Operational Research,* 149(3), 523-632.

[8] Calhoun, K. M., Deckro, R. F., Moore, J. T., Chrissis, J. W., & Hove, J. C. V. (2002). Planning and re-planning in project and production scheduling. *Omega, 30*, 155-170.

[9] Wu, S. D., Storer, R. H., & Chang, P. C. (1993). One-machine rescheduling heuristics with efficiency and stability as criteria. Computers & Operations Research, 20(1), 1-14.

[10] Rangsaritratsamee, R., Ferrell Jr., W. G., & Kurz, M. B. (2004). Dynamic rescheduling that simultaneously considers efficiency and stability. Computers & Industrial Engineering, 46(1), 1-15.

[11] Pfeiffer, A., Kádár, B., & Monostori, L. (2007). Stability-oriented evaluation of rescheduling strategies, by using simulation. *Computers in Industry, 58*(7), 630-643.

[12] 植村夏帆, 谷水義隆. (2024). 作業の外乱発生を考慮したリアクティブスケジューリングシステムの開発, 2023 年度早稲田大学卒業論文.

謝辞

本論文の作成にあたり，谷水義隆教授，渡邉るりこ講師，松野思迪講師から深いご指導と御助言を頂いたことに対して御礼申し上げます．また，谷水研究室のすべての方々に心から感謝致します.

鬼頭拓海