ロット編成問題を有するフレキシブル・フローショップを対象とした
分散協調型スケジューリング手法の一提案*貝原俊也^{*1}, 藤井信忠^{*1}, 西林 聡^{*2}
藤井 進^{*3}, 梅田豊裕^{*4}A Proposal of Distributed Cooperative Scheduling Approach
for Flexible Flowshop with Lot Formation ProblemToshiya KAIHARA^{*5}, Nobutada FUJII, Satoshi NISHIBAYASHI,
Susumu FUJII and Toyohiro UMEDA^{*5} Graduate School of Engineering, Kobe University,
1-1 Rokkodai-cho, Nada-ku, Kobe-shi, Hyogo, 657-8501 Japan

In this paper, we propose a distributed cooperative scheduling approach for flexible flowshop manufacturing system. This method breaks up the system into each process, and they are integrated as a single system. Since each process needs only local information, it is effective to apply the method to the large production system. Our approach focuses on lotsizing problem as well as manufacturing scheduling concurrently with Lagrangian decomposition and coordination method. We try to compare our method with conventional distributed cooperative method and centralized method, and show the effectiveness of the proposed concept.

Key Words: Manufacturing Scheduling, Lagrangian Decomposition and Coordination, Lotsizing Problem, Flexible Flowshop

1. は じ め に

近年の製造業において企業は多様な顧客のニーズに迅速に対応する必要があり, そのため生産システムは大規模化かつ複雑化している. このような状況下において, 生産スケジューリング問題を考える際, すべての情報を一元管理して最適化を行うのは非常に困難かつ非効率であり, そこで有効となるが分散型スケジューリング手法である. ここで本研究では, 最適化問題に対する分散型解法として, ラグランジュ分解・調整法⁽¹⁾に着目する. これは, 原問題を複数の部分問題に分解し, それらの解の導出と調整を行うことによって, 精度保証を行いながら準最適な実行可能解を高速に導出する方法であり, 従来より生産スケジューリング問題へ適用されている⁽²⁾⁽³⁾. 本研究はこのラグランジュ分解・調整法をバッチ処理によるロット編成を有するフレキシブル・フローショップ問題に適用し, 部分問題の求解ごとにロット編成を含めて最適化する手法を提案する. また他の従来手法との比較実験においてそ

の有効性を検証する.

2. 対象モデルと定式化

本研究で対象とするモデルは, 図1に示すようなバッチ処理を有するフレキシブル・フローショップ型の生産システムであり, 工程 $k \{k = 1, \dots, K\}$ においては機械が M_k 台存在する. また各工程では, ジョブは複数個纏められてロットとして処理され, 機械ごとに処理できるロットの編成ジョブ数の上限が異なるため, 各工程では機械への配分を考慮しながらロット編成を行う必要がある. 本モデルの特徴を以下に示す.

Fig. 1 Flexible flowshop model

- 各工程間には容量無限のバッファが存在する
- 各工程前のバッファでロット編成が行われる

* 原稿受付 2009年8月10日.

^{*1} 正員, 神戸大学大学院工学研究科情報知能学専攻(〒657-8501 神戸市灘区六甲台町1-1).^{*2} 神戸大学大学院工学研究科情報知能学専攻.^{*3} 正員, フェロー, 上智大学理工学部機械工学科(〒102-8554 東京都千代田区紀尾井町7-1).^{*4} (株)神戸製鋼所生産システム研究所(〒651-2271 神戸市西區高塚台1-5-5).

E-mail: kaihara@kobe-u.ac.jp

- 同じ品種を1つのロットに編成する
- 各工程には複数の機械が存在する
- 各機械が同時刻に処理可能なロットは高々1つである
- 各機械は1つのロットに編成できるジョブ数の上限が異なる
- 各機械における処理時間はジョブの品種と工程により決定され、ロットを編成するジョブ数によらず一定である

2.1 原問題

本研究では前節で説明したモデルを対象に、総納期遅れ最小化問題として原問題を定式化する。なお処理時間については、より汎用的な問題への適用を視野に入れ、ここではロットサイズにより処理時間が変化するものとして定式化を行う。用いる記号は以下の通りであり、あらかじめ既知のものには(*)と記載している。

i : 品種番号 * ($i = 1 \dots I$)

j : ジョブ番号 * ($j = 1 \dots J$)

k : 工程番号 * ($k = 1 \dots K$)

m_k : 工程 k における機械番号 * ($m_k = 1 \dots M_k$)

D_j : ジョブ j の納期 *

p_i^k : 工程 k における品種 i のジョブのロットサイズ 1 における処理時間 *

$N_{m_k}^k$: 工程 k における機械 m_k のロット纏め上限個数 *

$I_{i,j}$: ジョブと品種の関係 *, ジョブ j が品種 i である時 1, そうでなければ 0

l_k : 工程 k におけるロット番号

$lp_{l_k}^k$: ロット l_k の処理時間

PT_j^k : 工程 k におけるジョブ j の処理時間

λ_j^k : ジョブ j の工程 k におけるラグランジュ乗数

$\mu_{l_k,i}$: ロットと品種の関係, ロット l_k が品種 i であるとき 1, そうでなければ 0

s_j^k : 工程 k における j の処理開始時刻

π_{l_k,m_k}^k : 工程 k においてロット l_k が機械 m_k で処理される時 1, そうでなければ 0

δ_{j,l_k}^k : 工程 k においてジョブ j がロット l_k に属するとき 1, そうでなければ 0

$Ls_{l_k}^k$: 工程 k におけるロット l_k の処理開始時刻

また原問題 P0 を納期遅れ最小化問題として以下のように定式化する。

$$\text{P0 : } \min_{\{Ls_{l_k}^k, \delta_{j,l_k}^k, \pi_{l_k,m_k}^k\}} \sum_j T_j \quad (1)$$

$$\text{where : } T_j = \max\{0, (s_j^K + PT_j^K - D_j)\} \quad (2)$$

$$s_j^k = \sum_{l_k} Ls_{l_k}^k \delta_{j,l_k}^k \quad (3)$$

$$PT_j^k = \sum_{l_k} \delta_{j,l_k}^k lp_{l_k}^k \quad (4)$$

$$lp_{l_k}^k = \sum_i \mu_{l_k,i} p_i^k \quad (5)$$

$$\text{s.t. } s_j^k \geq 0 \quad (\forall j, \forall k) \quad (6)$$

$$s_j^{k+1} \geq s_j^k + PT_j^k \quad (\forall j, \forall k) \quad (7)$$

$$\sum_{l_k} \delta_{j,l_k}^k = 1 \quad (\forall j, \forall k) \quad (8)$$

$$\sum_i \frac{\sum_j \delta_{j,l_k}^k I_{i,j}}{\sum_j \delta_{j,l_k}^k I_{i,j} + 1} < 1 \quad (\forall k, \forall l_k) \quad (9)$$

$$\sum_{m_k} \pi_{l_k,m_k}^k = 1 \quad (\forall k, \forall l_k) \quad (10)$$

$$\sum_{m_k} \sum_j \delta_{j,l_k}^k \pi_{l_k,m_k}^k \leq \sum_{m_k} N_{m_k}^k \pi_{l_k,m_k}^k \quad (\forall k, \forall l_k) \quad (11)$$

$$Ls_{l_k}^k \pi_{l_k,m_k}^k \pi_{l'_k,m_k}^k - Ls_{l'_k}^k \pi_{l_k,m_k}^k \pi_{l'_k,m_k}^k \geq \pi_{l'_k,m_k}^k \pi_{l_k,m_k}^k lp_{l'_k}^k \quad (\forall k, \forall m_k, \forall l_k, \forall l'_k \text{ where } l_k \neq l'_k) \quad (12)$$

(1) 式は目的関数である総納期遅れを示し, (2)~(5) 式は順に納期, ジョブの処理開始時間, ジョブの処理時間, ロットの処理時間の関係式である. (2) 式は, ジョブの納期遅れが各ジョブの納期とジョブの最終工程の完了時間との差から求まる関係式, (3) 式はジョブの処理開始時間がロットの処理開始時間に基づいて決まることを示す関係式, (4) 式はジョブの処理時間が編成ロットの処理時間によって決定されることを示す関係式, (5) 式はロットの処理時間はロットの品種によって決定されることを示す関係式である. また (6)~(12) 式は制約式を示している. (6), (7) 式は処理開始時間に関する制約で, (6) 式は処理開始時間非負制約, (7) 式は後続する作業の処理開始時間は先行する処理の終了時刻以降であることを示す先行関係制約である. (8)~(12) 式はロットに関する制約であり, (8) 式はジョブはどれか 1 つのロットに必ず属することを, (9) 式はロットに属するジョブの品種は 1 つであることを, (10) 式はロットがどれかの機械で必ず処理されることを, (11) 式はロット内ジョブ数が割り当て機械の処理可能容量を超えないことをそれぞれ示し, (12) 式は割り当て機械におけるロットの同時処理禁止制約である. また決定変数は π_{l_k,m_k}^k , δ_{j,l_k}^k , $Ls_{l_k}^k$ であり, 制約を満たす形でロット編成と機械割り当てを行い, ロットの処理時間を決定することでジョブの処理時間を決定し, 納期遅れを最小化する問題となる.

2.2 部分問題への分解

問題 P0 を工程ごとの部分問題に分割するため, ジョブ j に対し, 工程ごとに, 残りの作業を空き時間なく実行した時にちょうど

納期を満たす最遅開始時刻 d_j^k を次のように定義する.

$$\begin{aligned} d_j^k &= D_j - \sum_{k'=k+1}^K PT_j^{k'} \quad (k=1, \dots, K-1) \\ &= D_j \quad (k=K) \end{aligned} \quad (13)$$

さらに最遅開始時間 d_j^k からの各工程における開始時間の遅れを納期遅れペナルティ T_j^k として以下のよう設定する.

$$T_j^k = \max\{0, (s_j^k - d_j^k)\} \quad (14)$$

この納期遅れペナルティを用いて, 問題 P0 を以下の問題 P1 で近似する.

$$\begin{aligned} \text{P1 : } & \min_{\{Ls_{l_k}^k, \delta_{j,l_k}^k, \pi_{l_k,m_k}^k\}} \sum_k \sum_j T_j^k \quad (15) \\ \text{s.t. } & \text{Eqs. (6) } \sim (12) \end{aligned}$$

これにより, 目的関数は工程ごとに加法的になる. また (7) 式を非負のラグランジュ乗数 λ_j^k を用いてラグランジュ緩和することで近似問題に対する以下の緩和問題 RP1 が形成される.

$$\begin{aligned} \text{RP1 : } & \min_{\{Ls_{l_k}^k, \delta_{j,l_k}^k, \pi_{l_k,m_k}^k\}} L \quad (16) \\ \text{where : } & L = \sum_k \sum_j T_j^k + \\ & \sum_{k=1}^{K-1} \sum_j \{\lambda_j^k (s_j^k + PT_j^k - s_j^{k+1})\} \quad (17) \\ \text{s.t. } & \text{Eqs. (6), (8) } \sim (12) \\ & \lambda_j^k \geq 0 \quad (\forall k, \forall l_k) \quad (18) \end{aligned}$$

また上記の緩和問題 RP1 において, $s_j^0 = 0$, $PT_j^0 = 0$, $s_j^{K+1} = \infty$ とし, $s_j^0 + PT_j^0 \leq s_j^1$, $s_j^K + PT_j^K \leq s_j^{K+1}$ を緩和すると, 問題 RP1 は以下のように変形が可能である.

$$\begin{aligned} \text{RP2 : } & \min_{\{Ls_{l_k}^k, \delta_{j,l_k}^k, \pi_{l_k,m_k}^k\}} L \quad (19) \\ \text{where : } & L = \sum_k \sum_j T_j^k + \\ & \sum_k \sum_j \{(\lambda_j^k - \lambda_j^{k-1}) s_j^k + \lambda_j^k PT_j^k\} \quad (20) \\ \text{s.t. } & \text{Eqs. (6), (8) } \sim (12), (18) \\ & \lambda_j^k = 0 \quad (\forall j, k=0, K) \quad (21) \end{aligned}$$

従って (20) 式は工程 k ごとの部分問題へと分解可能なので以下の部分問題 RP2- k が導出される.

$$\text{RP2-}k : \min_{\{Ls_{l_k}^k, \delta_{j,l_k}^k, \pi_{l_k,m_k}^k\}} L_k \quad (22)$$

$$\begin{aligned} \text{where : } & L_k = \sum_j T_j^k + \\ & \sum_j \{(\lambda_j^k - \lambda_j^{k-1}) s_j^k + \lambda_j^k PT_j^k\} \quad (23) \\ \text{s.t. } & \text{Eqs. (6), (8) } \sim (12), (18), (21) \end{aligned}$$

また各工程には複数機械が存在するので, 工程ごとの納期遅れは工程における機械の納期遅れの和として以下のように定義される.

$$\sum_j T_j^k = \sum_{m_k} \sum_j T_{j,m_k}^k \quad (24)$$

$$\begin{aligned} & \sum_{m_k} \sum_j T_{j,m_k}^k = \\ & \max \left\{ 0, s_j^k \sum_{l_k} \delta_{j,l_k}^k \pi_{l_k,m_k}^k - d_j^k \right\} \quad (25) \end{aligned}$$

従って工程ごとの部分問題 RP2- k は以下の通りである.

$$\begin{aligned} \text{RP2-}k : & \min_{\{Ls_{l_k}^k, \delta_{j,l_k}^k, \pi_{l_k,m_k}^k\}} L_k \quad (26) \\ \text{where : } & L_k = \sum_{m_k} \sum_j T_{j,m_k}^k + \\ & \sum_j \{(\lambda_j^k - \lambda_j^{k-1}) s_j^k + \lambda_j^k PT_j^k\} \quad (27) \\ \text{s.t. } & \text{Eqs. (6), (8) } \sim (12), (18), (21) \end{aligned}$$

また以上の式変形より問題 P1 の双対問題 DP1 は以下のように定式化される.

$$\begin{aligned} \text{DP1 : } & \max_{\lambda_j^k} \left\{ \min_{\{Ls_{l_k}^k, \delta_{j,l_k}^k, \pi_{l_k,m_k}^k\}} \sum_k L_k \right\} \quad (28) \\ \text{s.t. } & \text{Eqs. (6), (8) } \sim (12), (18), (21) \end{aligned}$$

従って L_k の最小化と $Ls_{l_k}^k$, δ_{j,l_k}^k , π_{l_k,m_k}^k を固定した時のラグランジュ乗数の調整を繰り返すことが必要であることがわかる. 部分問題の最適化手法と, ラグランジュ乗数の更新手法については後に述べる.

2.3 解の振動への対策 先行関係を単純に緩和した L_k を最適化した場合, 工程 k におけるラグランジュ関数 L_k を最小とする $Ls_{l_k}^k$ の値は, $(\lambda_j^k - \lambda_j^{k-1})$ の値に大きく依存し, 解の振動及び等式制約から大きくはずれた解の導出につながる. そこで本研究では解の収束性を向上させるために提案されている近似ラグランジュ緩和法 (ALR 法) を適用する⁽⁴⁾. ALR 法は前後工程の処理終了時間と処理開始時間からのずれに 1 次のペナルティを目的関数に課することが上記の問題への対策であり, ロット編成を含む問題に対しても同様に有効と考えられる⁽⁵⁾. 問題 RP2- k にペナルティを加えた問題 RP2'- k は以下の通りとなる.

$$\text{RP2}' - k : \min_{\{L_{i_k}^k, \delta_{j,i_k}^k, \pi_{i_k, m_k}^k\}} L_k' \quad (29)$$

$$\begin{aligned} \text{where} : L_k' &= \sum_{m_k} \sum_j T_{j, m_k}^k + \\ &\sum_j \left\{ (\bar{\lambda}_j^k - \bar{\lambda}_j^{k-1}) s_j^k + \bar{\lambda}_j^k P T_j^k \right\} + \\ &\max \left\{ 0, \bar{\rho}_j^k (s_j^k + P T_j^k - \bar{s}_j^{k+1}) \right\} + \\ \max \{ 0, &\bar{\rho}_j^{k-1} (\bar{s}_j^{k-1} + P T_j^{k-1} - s_j^k) \} \quad (30) \\ \text{s.t.} & \text{Eqs. (6), (8) } \sim \text{(12), (18), (21)} \end{aligned}$$

ここで、 $\bar{s}_j^{k-1}$, $\bar{s}_j^{k+1}$ は、ラグランジュ乗数の更新における前回の繰返しで求めた部分問題 $\text{RP2}' - k$ を解くことによって得られた解で、 $\bar{\lambda}_j^{k-1}$, $\bar{\lambda}_j^k$, $\bar{\rho}_j^{k-1}$, $\bar{\rho}_j^k$ は前回の繰返しで更新された値である。また ρ_j^k はペナルティ関数の重みであり、更新法などは後で説明する。ただし、この手法を用いた場合、ペナルティ関数により問題を近似しているため $\text{RP2}' - k$ の解は $\text{RP2} - k$ の最適解の近似となり、問題 $\text{RP2}' - k$ の最適解が問題 P1 の下界となる保証はない。

3. 最適化アルゴリズム

提案する手法は各部分問題の最適化の過程において、ロット編成の組合せも探索する手法である（以下同時最適化手法とする）。同時最適化手法では、ラグランジュ乗数の更新により、工程間で相互の制約違反を協調的に解消する毎にロット編成を行うため探索範囲が広がり、十分な探索を行うためには計算時間を要する。一方で各工程で準最適なロット編成を考慮しているため、システム全体として解の最適性は強まる。アルゴリズムの流れを図2に示す。また以下にその流れを説明する。

Step 1 オーダを読み込みラグランジュ乗数 $\lambda_j^k = 0$, ペナルティ係数 $\rho_j^k = 0$ とする

Step 2 各部分問題を SA(Simulated Annealing) 法と DP(Dynamic Programming) 法により部分問題の最適化 (a) を行う

Step 3 各部分問題の解から先行関係制約違反を確認し違反がなければ終了、そうでなければ **Step 4** へ

Step 4 規定回数に達している場合は **Step 6** へそうでなければ **Step 5** へ

Step 5 ラグランジュ乗数 λ_j^k , ペナルティ係数 ρ_j^k の更新 (c) を行い、**Step 2** へ

Step 6 実行可能解の生成 (d) を行い終了

Fig. 2 Concurrent optimization algorithm

ここで、**Step 2** の部分問題最適化手法 (a) およびそれに用いるロット編成・再編成アルゴリズム (b)、**Step 5** の各パラメータの更新法 (c)、**Step 6** の実行可能解の生成 (d) について、以降で詳しく説明する。

部分問題最適化手法 (a)

部分問題 $\text{RP2}' - k$ を最小化するにあたり、以下のように、先行研究において提案されている SA(simulated Annealing) 法を用いた手法⁽³⁾を適用している。本研究ではさらに、部分問題最適化手法へ DP(Dynamic Programming) 法を新たに適用している。ロット編成・再編成アルゴリズム (b) をロット編成手法として用いたアルゴリズムを以下に示す。

Step 1 オーダを納期順にソートし、その後品種順にソートすることで初期ジョブ順序列を形成

Step 2 初期ジョブ順序列に基づいてロット編成・再編成アルゴリズム (b) を適用し初期ロット・機械割り当てを決定

Step 3 DP 法により機械ごとに割り当てられたロットの処理開始時間 $L_{i_k}^k$ を決定し初期解 x を導出。また暫定解 $x' = x$ とする。また SA における温度パラメータ T を初期温度 T_0 にセットし、温度更新回数 $n = 1$, 反復回数 $r = 1$ とする

Step 4 現在の温度 T における反復回数 r が反復終了回数 R 未満なら **Step 5** へ、そうでなければ

Step 8 へ

Step 5 暫定解 x' に対して近傍構成法 1 あるいは 2(後述) を適用し, DP 法に基づいて近傍解 $y = N(x')$ を構成. また評価値 $f(y)$ を計算

Step 6 評価値を T に基づく確率により判定し, 暫定解 x' を決定

Step 7 $r = r + 1$ として **Step 4** へ

Step 8 温度更新回数 n が更新終了回数 N 未満なら $n = n + 1$ として **Step 9** へ, そうでなければ探索終了

Step 9 $T = T\zeta$, $r = 1$ として **Step 4** へ

なお, 初期温度 T_0 , 同一温度での探索回数 R , 温度の減衰回数 N , 減衰係数 ζ については実験により適宜設定する. **Step 3**, **Step 5** の DP 法の適用においてはロット編成・再編成アルゴリズム (b) に基づいた機械割り当て, ロット順序に基づいて機械ごとに行う. 従ってその条件下における最適解が導出される. ただし第 1 工程最適化においては処理順序が確定したロットを前詰めにする事で最適解が求まるので, 計算コストを要する DP 法は適用しない. また **Step 5** の近傍構成法は以下の 2 つをランダムで選択する.

- 近傍構成法 1

現在のジョブの順序列に対して挿入近傍により新たな順序列を構成し, それに対してロット編成・再編成アルゴリズム (b) を適用する.

- 近傍構成法 2

ランダムで機械を選択し, 機械内のロットの順序列に対して挿入近傍により新たな順序列を構成する.

また **Step 6** の解の判定方法は以下の式を用いた⁽⁶⁾.

$$\begin{aligned} & \text{if } \epsilon \leq 0 \text{ then } x' := y \\ & \text{else if } \exp\left(\frac{-\epsilon}{T}\right) \geq \text{random}[0, 1) \text{ then } x' := y \\ & \text{else } x' := x' \\ & \text{where } \epsilon = f(y) - f(x') \end{aligned} \quad (31)$$

ここで $f(x')$, $f(y)$ は (30) 式により計算されるラグランジュ関数値である. また ϵ はその差を示しており, (31) 式によって計算される確率によって暫定解の更新が行われる.

ロット編成・再編成アルゴリズム (b)

ジョブの順序列からロット編成および機械割り当てを行うアルゴリズムである. まずロットを割り当てるべき機械を選択する. このとき処理ロットの処理時間の和が最小の機械を選択するが, 処理時間が等しい場合はロット編成上限値 N_{mk}^k の大きい機械を選択する. 続いてワークの順序列に対し

て, 同じ種類のジョブが連続するか, 上限値 N_{mk}^k に達するまで同じロットに編成する. ジョブの種類が異なるか, 上限値 N_{mk}^k を超えた場合, そのジョブからは新たなロットとする. この手続きをすべてのジョブがロットに編成されるまで繰り返す.

ラグランジュ乗数およびペナルティ係数の更新法 (c)

双対問題 DP1 を解くために λ_j^k の更新ルールには (33) 式の劣勾配法を適用する. λ_j^k に対する劣勾配 g_j^k は (32) 式であり, 更新式は (33) 式である.

$$g_j^k = s_j^k + PT_j^k - s_j^{k+1} \quad (32)$$

$$\lambda_j^k = \bar{\lambda}_j^k + \alpha g_j^k \quad (33)$$

また ρ_j^k の更新ルールは単調非減少とする (34) 式のルールを適用する.

$$\rho_j^k = \bar{\rho}_j^k + \gamma \max(0, g_j^k) \quad (34)$$

上記の式において $\bar{\lambda}_j^k$, $\bar{\rho}_j^k$ は 1 つ前の繰り返しにおける各パラメータ値であり, また α , γ は実験において適宜設定するステップ幅である.

実行可能解の生成 (d)

ラグランジュ分解・調整法においては, 必ず実行可能解が得られるとは限らない. そこで双対問題 DP1 の解の構造を利用して実行可能解を生成する手続きによって, 実行可能解の生成を保証する. まず違反を起こしている解のロットの処理順序を基に各工程ごとにリストを作り, 第 1 工程からリストの順に前工程の処理終了時間あるいは同じ機械で先行して処理されるロットの処理終了時間を越えない範囲で前詰めにする事で実行可能解が得られる.

4. 実験と考察

本章では実験結果と考察を述べる. ここで, 基本となる実験条件を表 1 に示す. まず実験 1 では, 同時最適化手法と本研究の先行研究において提案したロットを固定する階層型的手法⁽⁵⁾⁽⁷⁾(以下階層型最適化手法)との比較を示す. 実験 2 では同時最適化手法のロット編成手法にランダム性を持たせた拡張手法を用いて他の手法との比較を示す. また実験 3 では同時最適化手法と集中型的手法における比較実験を行う. これらの実験において, 試行回数を 10 回, 計画対象期間を 2 日とし, 使用計算機は Intel Core 2 Duo CPU 3.00 Ghz である.

4.1 実験 1 同時最適化手法と階層型最適化手法の比較を行った. 階層型最適化手法のアルゴリズムを簡単に示す.

Step 1 オーダを読み込みラグランジュ乗数 $\lambda_j^k = 0$, ペナルティ係数 $\rho_j^k = 0$ とする

Table 1 Experimental Parameter

I	4	R	100
J	25	N	200
K	3	ζ	0.97
M_1	3	α	0.005
M_2	3	ρ	0.005
M_3	3		

Step 2 ロット編成・再編成アルゴリズム (b) に基づいてロット編成・機械割り当て

Step 3 各部分問題を機械ごとのロット順序の全探索と DP 法により最適化を行う

Step 4 各部分問題の解から先行関係制約違反を確認し違反がなければ終了、そうでなければ **Step 5** へ

Step 5 規定回数に達している場合は **Step 7** へそうでなければ **Step 6** へ

Step 6 ラグランジュ乗数 λ_j^k , ペナルティ係数 ρ_j^k の更新 (c) を行い, **Step 3** へ

Step 7 実行可能解の生成 (d) を行い終了

Step 2, Step 6, Step 7 の各アルゴリズムは同時最適化手法と同様である。また両手法において劣勾配法を適用する場合、実行可能解がでた後にも最良解の更新が行われる可能性がある。従って実行可能解導出の可否にかかわらず協調終了回数は 50 回とした。実験結果を以下に示す。

Fig. 3 Transition of the violation by hierarchical method

Fig. 4 Transition of the violation by concurrent method

Table 2 Value of objective function

	Hierarchical Method	Concurrent Method
Ave.	4011	2956
S.D.	1551	407
Min.	2537	2367
Max.	6620	3579
CPU Time (sec.)	82	13815

図 3, 図 4 に同じデータに対して同時最適化手法及び階層型最適化手法を適用した場合の先行関係制約 (7) 式の違反数の推移を示す。階層型手法ではロットが探索毎に固定であるため、同じステップサイズにもかかわらず実行可能解への収束は早い。一方で同時最適化手法のほうも実行可能解への収束に要する繰り返し数はかかっているものの、協調が進むことによるペナルティの増加により、実行可能解へ収束している。また表 2 は原問題の目的関数値 (1) 式の値と計算時間の平均値を示している。この結果から提案手法が階層型最適化手法よりも目的関数値で良い結果を示していることがわかる。計算時間に関して提案手法が階層型最適化手法に比べて大きいのは、部分問題最適化において繰り返し毎にロット編成の探索と DP 法を行っているため探索空間が広まり、その結果、DP 法と SA 法の計算コストがかかるためであることを確認した。

4.2 実験 2 提案手法および提案手法の一部に変更を加えた手法との比較を行った。この変更点は、ロット編成アルゴリズム (b) において、ジョブ順序列から最初にロットを構成する際、割り当て機械の決定にランダム性を付加したことである。この場合、たとえ繰り返し毎にジョブ順序列を納期ごと、種類ごとにソートしたとしても作られる初期ロットが異なり、その結果、ロット編成の探索空間が広まることが期待できる。協調回数は実験 1 と同様 50 回とした。以下に実験結果を示す。

Fig. 5 Transition of the violation by extended method

図 5 は図 4 と同じデータに対して拡張手法を適用

Fig. 6 Transition of the value of the objective function by extended optimization method

Table 3 Value of objective function

	Concurrent Method	Extended Method
Ave.	2956	2796
S.D.	407	381
Min.	2367	2417
Max.	3579	3404
CPU Time (sec.)	13815	13678

した時の先行関係制約 (7) 式の違反数の推移であり、図 6 はそのときの原問題の目的関数 (1) 式の値の推移である。図 4 は協調が進むにつれて実行可能解に収束しているが、拡張手法においては協調の終盤においても解の振動が見られる。これは繰り返して毎に初期編成ロットが異なるため、前回のラグランジュ乗数を基に探索を進めた場合最終的に前回と異なる局所解が求まり、それが協調および解の収束を阻害していると考えられる。また表 3 には提案手法と拡張手法における (1) 式の値と計算時間の平均値を示す。表 3 の結果から最小値では提案手法が勝っているものの、平均値では拡張手法のほうが勝っている。これは、探索における初期ロットを毎回ランダムで変更させることで探索範囲が広がり、結果として良い解を導出できているためである。

4.3 実験 3 提案手法と集中型のアルゴリズムとの比較を行った。集中型の手法は問題の分割は行わず、常に実行可能解を導出しながら解の探索を行う手法である。集中型の最適化手法としては提案手法と同じ SA 法を用いた。また **Step 2** におけるロット編成アルゴリズム (b) は提案手法と同じアルゴリズムを用いた。以下にその流れを示す。

Step 1 各工程においてジョブを納期順にソートし、その後品種順にソートすることで初期ジョブ順序列を形成

Step 2 工程ごとに初期ジョブ順序列に基づいてロット編成・再編成アルゴリズム (b) を適用し初期ロッ

ト・機械割り当て・ロット順序を決定

Step 3 前工程から現在のロット編成の基でロットの処理開始時間 $Ls_{l_k}^k$ を確定していく。また暫定解 $x' = x$ とし、 $T = T_0$, $n = 1$, $r = 1$ とする

Step 4 $r < R$ なら、**Step 5** へ、そうでなければ **Step 8** へ

Step 5 暫定解 x' に対して近傍構成法を適用する。また **STEP 3** と同様に処理開始時間を確定し近傍解 $y = N(x)$ を構成。また評価値 $f(y)$ を計算

Step 6 評価値を T に基づく確率により判定し、暫定解 x' を決定

Step 7 $r = r + 1$ として **Step 4** へ

Step 8 $n < N$ なら終了、そうでなければ $n = n + 1$ として **Step 9** へ

Step 9 $T = T\zeta$, $r = 1$ として **Step 4** へ

なお **Step 3**, **Step 5** の処理時間確定法は第 1 工程は前詰めで処理時間を確定する。第 2 工程以降は、先行関係制約を違反しないかつ、機械において順序が先の割り当てロットの終了時間を比較し、その中で早い時間とする。また近傍構成法は工程をランダムで選択し、その工程に対して、提案手法における部分問題最適化手法 (a) の近傍構成法 1 あるいは 2 をランダムで適用した。**Step 6** の解の受理判定も (31) 式と同じであるが、評価関数 f は原問題 P0 の (1) 式であることが同時最適化手法と異なる。集中型手法の終了条件は提案手法が実行可能解導出に必要な計算時間の平均値を終了条件とし、ジョブ数 $J = 50$, $J = 75$ について実験を行った。以下に実験結果を示す。表 4 は、実

Table 4 Value of objective function

	Concurrent Method		Centralized Method		CPU Time (sec.)
J	Ave.	S.D.	Ave.	S.D.	Ave.
50	18578	1048	16390	956	3527
75	35678	2179	37678	2307	5245

験 1 と同じ入力データ (10 試行) を用いて同時最適化型手法と集中型手法において得られた目的関数値と計算時間の平均値を示している。 $J = 50$ 時では集中型手法が目的関数値において勝っており、 $J = 75$ 時には同時最適化手法のほうが良い値を示している。この原因としては集中型では解く問題のサイズが同時最適化手法に比べて大きいために、ジョブ数が多くなるにつれ解を十分に探索できなくなるためである。一方でジョブ数が比較的少ない問題では、同時最適化手法では部分問題に分割するために近似問題を構成し、それ

を解く事で目的関数である (1) 式の値を求めているため、原問題とのギャップが生じ集中型手法のほうが良い結果を示していることが分かった。

5. ま と め

本論文では分散協調型スケジューリング手法としてラグランジュ分解・調整法を適用し、ロット編成を有する問題を対象に部分問題を解く過程でロット編成とスケジュールを同時に最適化する手法を提案した。また3種類の計算機実験を通して解の精度や求解速度についての評価を行い、以下の点を明らかにした。

- 実験1: 階層型最適化手法との比較実験において、計算時間はかかるものの解の精度については勝っていることが確認され、提案手法の基本的な優位性を示すことができた。
- 実験2: 提案手法の一部に変更を加え解の探索空間を広げることを試みた。その結果、解の精度は改善するが、その一方で探索空間の広がり解の振動を引き起こし収束性が悪化することが明らかになった。
- 実験3: 集中型スケジューリング手法と提案法による比較を行った。その結果、ジョブ数が少ない場合は目的関数をそのまま取り扱うことができる集中型手法が、ジョブ数が多い場合は提案手法がそれぞれ有効であることが確認できた。スケジューリング手法としては、取り扱えるジョブの多い場合が実用的であることは自明であり、その意味では提案手法の優位性を明らかにできたと考えている。

今後の課題としては、実規模モデルを対象とした提案手法の解精度および計算時間に関する有効性検証が挙げられる。

文 献

- (1) Imaizumi, J., A survey of production scheduling by Lagrangian decomposition and coordination method, *Journal of business administration*, (2001), Vol. 54, pp. 1-20
- (2) Muramatsu, K., Lagrangean decomposition coordination method for multi-item multi-process dynamic lot size scheduling, *Proceedings of International Symposium on Scheduling*, (2004), pp. 224-229
- (3) Fukayama, T., Nishi T. and Konishi, M., Optimization of lotsizing and scheduling by an autonomous decentralized scheduling method for Aluminum production line, *IEEE Transactions on Electronics, Information and Systems*, Vol. 126, No. 6, (2006), pp. 771-779
- (4) Nishi, T., Hasebe, S. and Hashimoto I., Machine-oriented decomposition of flowshop scheduling problems using approximate Lagrangian relaxation, *Proceedings of the Scheduling Symposium '99*, (1999), pp. 29-34
- (5) Kaihara, T., Fujii, N., Nishibayashi, S., Fujii, S. and Umeda, T., A distributed cooperative approach for flowshop scheduling problem using approximate Lagrangian method, *SICE Symposium on Systems and Information* 2008, (2008), pp. 187-192,
- (6) Aiyoshi, E. and Yasuda K., *Metaheuristics and their applications*, (2007), Ohmsha, Tokyo
- (7) Kaihara, T. Sashio, S., Nishibayashi, S., Fujii S. and Umeda, T., A proposal of distributed cooperative scheduling to solve flow shop scheduling problems, *2008 International Symposium on Flexible Automation*, (2008), Atlanta, GA, USA