「フレキシブル・オートメーション」総合特集号

論　文

2 機械順列フローショップ問題に対する dynasearch の拡張*

田中　俊二†

Extension of the Dynasearch to the Two-Machine Permutation Flowshop Scheduling Problem*

Shunji Tanaka†

The purpose of this study is to construct a solution algorithm for the two-machine permutation flowshop problem based on the dynasearch. The dynasearch is an efficient local search algorithm that employs a special neighborhood structure called dynasearch swap neighborhood. Its primary advantage is that the neighborhood of a solution can be explored in polynomial time although it is composed of an exponential number of solutions. The dynasearch for machine scheduling was originally developed for the single-machine total weighted tardiness problem. Then, it was extended to the problem with idle time and setup times. This study further extends the dynasearch to the two-machine permutation flowshop problem and its effectiveness is examined by numerical experiments for both total weighted tardiness and total weighted earliness-tardiness objectives.

1. はじめに

1 機械スケジューリング問題に対する効率的な局所探索アルゴリズムとして，dynasearch[1] が知られている．dynasearch の特長は，dynasearch 近傍とよばれる近傍構造を用いることによって，指数オーダの近傍サイズでありながら，近傍内の最良解を多項式時間で求めることができる点にある．dynasearch は，仕事の完了コスト和を目的関数とし，遊休時間が発生しない 1 機械スケジューリング問題なら，コスト関数の形状によらず適用可能であるが，もともとは，1 機械重み付き納期遅れ和問題に対して提案された解法である．とくに，反復局所探索の枠組に dynasearch を組み込んだ iterated dynasearch は，OR-Library[2] の 1 機械重み付き納期遅れ和問題のベンチマーク問題集に対する最良解を与えた近似解法としてもよく知られている．dynasearch はその後，近傍サイズの拡張による探索性能の向上 [3] や近傍内探索の計算量低減 [4]，また，遊休時間や段取り時間が存在する問題への拡張 [5] などがはかられている．

本論文の目的は，この dynasearch をさらに拡張し，2 機械順列フローショップ問題へ適用することである．順列フローショップ問題とは，各機械上での仕事の処理順序が同じでなければならないという制約がついたフローショップ問題であり，滞留時間和（完了時刻和）最小化の場合，2 機械でも NP 困難であることが知られている [6]．従来，この滞留時間和，あるいは最大完了時刻最小化問題などに対する近似解法の研究が多く行われてきたが，本研究では，これまであまり扱われてこなかった重み付き納期遅れ和，重み付き納期ずれ和最小化問題（いずれも NP 困難）に対して数値実験を行い，その有効性を示す．

本論文の構成は以下の通りである．まず **2.** では，本論文で対象とする 2 機械順列フローショップの概要を述べる．続いて **3.** では，遊休時間のない 1 機械問題に対して提案されている dynasearch を，また **4.** では，遊休時間が存在する問題への拡張を述べる．それらをふまえ，**5.** で，2 機械順列フローショップ問題に対する dynasearch の拡張を提案する．さらに **6.** で，dynasearch を反復局所探索の枠組で適用する iterated dynasearch を説明する．そして **7.** で，数値実験を行って提案解法の有効性を確認する．最後に **8.** で，本研究の成果をまとめるとともに，今後の課題を述べる．

* 原稿受付　2010 年 6 月 14 日

† 京都大学 大学院 工学研究科　Graduate School of Engineering, Kyoto University; Kyotodaigaku-Katsura, Nishikyo-ku, Kyoto City, Kyoto 615-8510, JAPAN

Key Words: scheduling, two-machine permutation flowshop problem, dynasearch, dynamic programming.

2. 2機械順列フローショップ問題

本節では，本研究で対象とする2機械順列フローショップ問題の説明を行う．

2台の機械（機械1，機械2）上で n 個の仕事（仕事1，…，仕事 n）を処理することを考える．各仕事 i は，機械1上での作業 O_{i1}（処理時間 p_{i1}），機械2上での作業 O_{i2}（処理時間 p_{i2}）の二つの作業から構成される．O_{i1} は時刻0から開始可能であるが，O_{i2} は O_{i1} が完了しないと開始することができない．すなわち，O_{i1}，O_{i2} の完了時刻をそれぞれ C_{i1}，C_{i2} とすると，

$$C_{i2} \geq C_{i1} + p_{i2} \tag{1}$$

の関係を満たす．各機械は同時に一つの作業しか処理することができず，また，一度作業の処理を開始すると，処理が完了するまで中断できないものとする．さらに，仕事の処理順序は機械1，2上で共通であるものとする．したがって，たとえば機械1上で O_{11}, O_{21}, O_{31} の順に作業の処理が行われるとき，機械2上での作業の処理順序は O_{12}, O_{22}, O_{32} となる．

各仕事 i にはコスト関数 $f_i(t)$ が与えられており，O_{i2} の完了時刻 C_{i2} に応じてコスト $f_i(C_{i2})$ が発生する．本研究で対象とする2機械順列フローショップ問題は，完了コスト和 $\sum_{i=1}^{n} f_i(C_{i2})$ が最小となる仕事の処理順序を求める問題である．

以下，簡単のため，作業の処理時間 p_{i1}, p_{i2} および完了時刻 C_{i1}, C_{i2} $(1 \leq i \leq n)$ はすべて整数であるものとする．

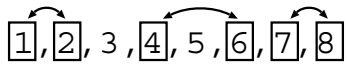
3. 1機械スケジューリング問題に対するdynasearch

本節では，遊休時間のない1機械スケジューリング問題に対して提案されている dynasearch[1] と，その拡張である enhanced dynasearch[3] を紹介する．そして次節で，遊休時間が存在する問題に対する dynasearch の拡張を説明する．そのために，本節と次節では，仕事数 n，仕事 i の処理時間が p_i，コスト関数が $f_i(t)$ である1機械スケジューリング問題を考えるものとする．

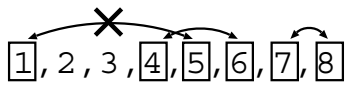
dynasearch では，dynasearch 近傍とよばれる近傍構造を利用する．dynasearch 近傍は，

> 現在の解（仕事の順列）において仕事対を一つまたは複数選び，その処理順序を入れ替えて得られる解集合

と定義される．ただし，互いに交差する処理順序の入れ替えは禁止される．たとえば，Fig. 1 において，(a) のように交差しない入れ替えは何度行ってもよいが，(b) のような交差する入れ替えは禁止される．この dynasearch 近傍のサイズは $2^{n-1}-1$ となり（元の解を含めると 2^{n-1}），仕事数の指数オーダであるが，動的計画法を用いることで，近傍内の最良解を多項式時間で求めることができる．



(a) Independent pairwise interchanges



(b) Dependent pairwise interchanges are forbidden

Fig. 1 Job interchanges in the dynasearch swap neighborhood

現在の解（仕事の順列）を $\Pi_n = \{\pi_1,\ldots,\pi_n\}$，$k$ 番目までの仕事 $\{\pi_1,\ldots,\pi_k\}$ からなる部分列を Π_k $(0 \leq k \leq n)$ と表すものとする．また，Π_k の処理時間の総和 $\sum_{i=1}^{k} p_{\pi_i}$ を $P(\Pi_k)$，部分列 Π_k の完了コスト和 $\sum_{i=1}^{k} f_{\pi_i}(P(\Pi_i))$ を $F(\Pi_k)$ と表す．さらに，Π_k の dynasearch 近傍内の最良解を $\Gamma_k = \{\gamma_1,\ldots,\gamma_k\}$ とする．このとき，Π_n の dynasearch 近傍内の最良解 Γ_n は，以下の再帰方程式により求めることができる．

$$F(\Gamma_0) = 0 \tag{2}$$

$$F(\Gamma_k) = \min\Bigg\{ F(\Gamma_{k-1}) + f_{\pi_k}(P(\Pi_k)), \min_{1 \leq i \leq k-1} F(\Lambda_{ik}^{\mathrm{PI}}) \Bigg\}, \quad k = 1,2,\ldots,n \tag{3}$$

ただし，$\Lambda_{ik}^{\mathrm{PI}}$ $(1 \leq i < k \leq n)$ は

$$\begin{aligned} \Lambda_{ik}^{\mathrm{PI}} &= \{\lambda_{i1},\ldots,\lambda_{ik}\} \\ &= \{\gamma_1,\ldots,\gamma_{i-1},\pi_k,\pi_{i+1},\ldots,\pi_{k-1},\pi_i\} \end{aligned} \tag{4}$$

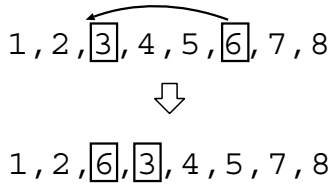
で定義される部分列である．すなわち，Π_k において，$i-1$ 番目までの部分列 Π_{i-1} を，その dynasearch 近傍内での最良解 Γ_{i-1} で置き換え，さらに，i 番目と k 番目の仕事の処理順序を入れ替えた部分列である．

$\Lambda_{ik}^{\mathrm{PI}}$ の先頭からの長さ j の部分列を $\Lambda_{ikj}^{\mathrm{PI}}$ $(i \leq j \leq k)$ と表すものとすると，$F(\Lambda_{ik}^{\mathrm{PI}})$ は

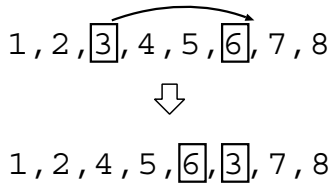
$$F(\Lambda_{ik}^{\mathrm{PI}}) = F(\Gamma_{i-1}) + \sum_{j=i}^{k} f_{\lambda_{ij}}(P(\Lambda_{ikj}^{\mathrm{PI}})) \tag{5}$$

で計算できるため，ある i, k の組に対する $F(\Lambda_{ik}^{\mathrm{PI}})$ の計算量は $O(n)$ である．よって，各 k に対する $F(\Gamma_k)$ の計算量は $O(n^2)$ となり，結局 Γ_n は $O(n^3)$ で求めることができる．しかし，重み付き納期遅れ和最小化問題の場合，Γ_n の計算量を $O(n^2)$ に削減できることが示されている [4]．

一方，Grosso ら [3] は，dynasearch 近傍を拡張した enhanced dynasearch 近傍を提案している．この近傍では，2仕事の入れ替え操作 (Pairwise Interchange; PI) だけでなく，Fig. 2 のような仕事の挿入操作 (Extraction



(a) Extraction and backward-shifted reinsertion



(b) Extraction and forward-shifted reinsertion

Fig. 2 Operations in the enhanced dynasearch neighborhood

and Backward-Shifted Reinsertion; EBSR, Extraction and Forward-Shifted Reinsertion; EFSR) も考慮することにより，近傍サイズを広げ，探索能力の向上をはかっている．これらの操作を組み込んだ場合の動的計画法の再帰方程式は，

$$F(\Gamma_0)=0 \tag{6}$$

$$F(\Gamma_k)=\min\Bigg\{F(\Gamma_{k-1})+f_{\pi_k}(P(\Pi_k)),\ \min_{1\le i\le k-1}F(\Lambda_{ik}^{\mathrm{PI}}),\ \min_{1\le i\le k-2}F(\Lambda_{ik}^{\mathrm{EBSR}}),\ \min_{1\le i\le k-2}F(\Lambda_{ik}^{\mathrm{EFSR}})\Bigg\},\quad k=1,2,\ldots,n \tag{7}$$

となる．ただし，$\Lambda_{ik}^{\mathrm{EBSR}}$，$\Lambda_{ik}^{\mathrm{EFSR}}$ は

$$\Lambda_{ik}^{\mathrm{EBSR}}=\{\gamma_1,\ldots,\gamma_{i-1},\pi_k,\pi_i,\ldots,\pi_{k-1}\} \tag{8}$$

$$\Lambda_{ik}^{\mathrm{EFSR}}=\{\gamma_1,\ldots,\gamma_{i-1},\pi_{i+1},\ldots,\pi_k,\pi_i\} \tag{9}$$

である．

4. 遊休時間が存在する 1 機械スケジューリング問題に対する dynasearch の拡張

前節で述べた (enhanced) dynasearch は，遊休時間の発生しない 1 機械スケジューリング問題に対して提案されたものであった．本節では，コスト関数 $f_i(t)$ が単調非減少ではないため，あるいは，仕事に処理開始可能時刻の制約が与えられているために遊休時間が発生する問題への拡張 [5] を説明する．以下ではまず，一般的な拡張方法を説明し，その後，その枠組で処理開始可能時刻を取り扱う方法を述べる．ここでは簡単のため，PI のみを考慮した dynasearch について説明するが，EBSR，EFSR を考慮した enhanced dynasearch の場合も同様である．

前節の遊休時間の発生しない問題に対する dynasearch では，ある 2 仕事の処理順序を入れ替えても，後続の仕事の完了時刻が変化しないことを利用していた．一方，遊休時間が存在する問題の場合には，処理順序の入れ替えにより最適な遊休時間の長さが変化し，後続の仕事の完了時刻に影響が出る可能性がある．このため，dynasearch 近傍内の最良解は，最後の仕事の完了時刻にも依存することになる．そこで，時刻 t までに部分列 Π_k の仕事すべてを処理し終わるという条件のもとでの，Π_k の dynasearch 近傍内の最良解を $\Gamma_k(t)$ と表すものとする．また，計画期間が $[0,T]$ であるとする．このとき，$\Gamma_n(T)$ を求めることが目的となる．

$\Gamma_n(T)$ は，以下の再帰方程式で求めることができる．

$$F_{\mathrm{I}}(\Gamma_0(t))=\begin{cases}\infty, & t<0\\ 0, & 0\le t\le T\end{cases} \tag{10}$$

$$F_{\mathrm{I}}(\Gamma_k(t))=\begin{cases}\infty, & t<P(\Pi_k)\\ \min\Big\{F_{\mathrm{I}}(\Gamma_k(t-1)),\ F_{\mathrm{I}}(\Gamma_{k-1}(t-p_{\pi_k}))+f_{\pi_k}(t),\ \min\limits_{1\le i\le k-1}F_{\mathrm{I}}(\Lambda_{ik}^{\mathrm{PI}}(t))\Big\}, & P(\Pi_k)\le t\le T\end{cases}\quad k=1,2,\ldots,n \tag{11}$$

ただし，$F_{\mathrm{I}}(\Gamma_k(t))$ は，(遊休時間を最適に挿入した際の) $\Gamma_k(t)$ の完了コスト和である．また，$\Lambda_{ik}^{\mathrm{PI}}(t)=\{\lambda_{i1},\ldots,\lambda_{ik}\}$ は，以下を満たす仕事列である．

(1) 含まれる仕事は時刻 t 以前にすべて完了する
(2) 前半の $(i-1)$ 仕事は $\Gamma_{i-1}(t')$
(3) 後半の $(k-i+1)$ 仕事は π_k, π_{i+1}, …, π_{k-1}, π_i
(4) t' は $F_{\mathrm{I}}(\Lambda_{ik}^{\mathrm{PI}}(t))$ が最小になるよう決定

この $F_{\mathrm{I}}(\Lambda_{ik}^{\mathrm{PI}}(t))=F_{\mathrm{I}}(\Lambda_{ikk}^{\mathrm{PI}}(t))$ は，以下の再帰方程式で与えられる．

$$F_{\mathrm{I}}(\Lambda_{iki}^{\mathrm{PI}}(t))=\begin{cases}\infty, & t<P(\Lambda_{iki}^{\mathrm{PI}}(\cdot))\\ \min\Big\{F_{\mathrm{I}}(\Lambda_{iki}^{\mathrm{PI}}(t-1)),\ F_{\mathrm{I}}(\Gamma_{i-1}(t-p_{\lambda_{ii}}))+f_{\lambda_{ii}}(t)\Big\}, & P(\Lambda_{iki}^{\mathrm{PI}}(\cdot))\le t\le T\end{cases} \tag{12}$$

$$F_{\mathrm{I}}(\Lambda_{ikj}^{\mathrm{PI}}(t))=\begin{cases}\infty, & t<P(\Lambda_{ikj}^{\mathrm{PI}}(\cdot))\\ \min\Big\{F_{\mathrm{I}}(\Lambda_{ikj}^{\mathrm{PI}}(t-1)),\ F_{\mathrm{I}}(\Lambda_{ik,j-1}^{\mathrm{PI}}(t-p_{\lambda_{ij}}))+f_{\lambda_{ij}}(t)\Big\}, & P(\Lambda_{ikj}^{\mathrm{PI}}(\cdot))\le t\le T\end{cases}\quad j=i+1,\ldots,k \tag{13}$$

ある i, k $(1\le i\le k\le n)$ の組に対し，$F_{\mathrm{I}}(\Lambda_{ik}^{\mathrm{PI}}(t))$ を

任意の t $(0 \le t \le T)$ について求めるには，(12), (13) 式より $O(nT)$ の計算量が必要となる．したがって，ある k に対し $F_{\mathrm{I}}(\Gamma_k(t))$ を任意の t について求めるには，$O(n^2T)$ の計算量が必要となる．よって，$\Gamma_n(T)$，あるいは $F_{\mathrm{I}}(\Gamma_n(T))$ は，計算量 $O(n^3T)$ で求めることができる．しかし，T が大きくなると計算量が増大してしまうため，文献 [5] では，$f_i(t)$ が t に関して区分線形であるものとして計算を効率化する方法 [7] を適用している．これは，$F_{\mathrm{I}}(\Gamma_k(t))$ を t に関する区分線形関数として扱い，各線形区間の端点でのみ値を評価するという方法である．関数同士の加算や区間最小値の計算などの処理が必要になるが，とくに $f_i(t)$ の区間数が少ない場合，計算時間を大幅に削減することができる．

なお，各仕事 i に処理開始可能時刻 $r_i \ge 0$ が与えられている問題も，この拡張により容易に扱うことができる．すなわち，コスト関数 $f_i(t)$ を

$$f_i(t) = \begin{cases} \infty, & t < r_i + p_i \\ f_i(t), & t \ge r_i + p_i \end{cases} \tag{14}$$

と置き直して上述の方法を適用すればよい．このことは，次節で述べる 2 機械順列フローショップ問題へ dynasearch を拡張する際に重要となる．

5. 2 機械順列フローショップ問題に対する dynasearch の拡張

本節では，前節の dynasearch を，さらに 2 機械順列フローショップ問題へ拡張する方法を提案する．

2 機械順列フローショップ問題では，仕事の処理順序は各機械上で共通なので，解は仕事の順列により表すことができる．そこで，**3.**, **4.** と同様，現在の解における仕事の順列を $\Pi_n = \{\pi_1, \ldots, \pi_n\}$ とし，dynasearch 近傍内の最良解（仕事の順列）$\Gamma_n(T)$ を求めることを考える．機械 1 上ではコスト関数 $f_i(t)$ の形状によらず前詰めスケジュールのみを考えればよいので，各仕事 π_k $(1 \le k \le n)$ の機械 1 上での完了時刻 $C_{\pi_k 1}$ は，

$$C_{\pi_k 1} = \sum_{i=1}^{k} p_{\pi_i 1} \tag{15}$$

と表すことができる．一方，機械 2 上での作業 $O_{\pi_k 2}$ の処理は時刻 $C_{\pi_k 1}$ より前に開始できないため，作業 $O_{\pi_k 2}$ の処理開始可能時刻が $C_{\pi_k 1}$ である，とみなすことができる．この機械 2 上での処理開始可能時刻は，1 機械スケジューリング問題における通常の意味での処理開始可能時刻とは異なり，仕事の処理順序に依存して変化する．しかし，(15) 式よりわかるように，$C_{\pi_k 1}$ は部分列 $\Pi_{k-1} = \{\pi_1, \ldots, \pi_{k-1}\}$ 内での処理順序の入れ替えの影響を受けないので，前節の最後での議論により，コスト関数 $f_{\pi_k}(t)$ をうまく置き直すことで考慮することができる．

いま，部分列 Π_k に含まれる仕事の機械 2 上での総処理時間を $P_2(\Pi_k) = \sum_{i=1}^{k} p_{\pi_i 2}$ と表すものとすると，2 機械順列フローショップ問題に対する dynasearch の再帰方程式は以下のようになる．

$$F_{\mathrm{F}}(\Gamma_0(t)) = \begin{cases} \infty, & t < 0 \\ 0, & t \ge 0 \end{cases} \tag{16}$$

$$F_{\mathrm{F}}(\Gamma_k(t)) = \begin{cases} \infty, & t < P_2(\Pi_k) \\ \min\Big\{ F_{\mathrm{F}}(\Gamma_k(t-1)), & \\ \quad F_{\mathrm{F}}(\Gamma_{k-1}(t-p_{\pi_k 2})) & \\ \quad\quad + f'_{\pi_k}(\Gamma_{k-1}(\cdot), t), & \\ \quad\quad \min_{1 \le i \le k-1} F_{\mathrm{F}}(\Lambda^{\mathrm{PI}}_{ik}(t)) \Big\}, & \\ P_2(\Pi_k) \le t \le T & \\ k = 1,2,\ldots,n \end{cases} \tag{17}$$

ただし，$F_{\mathrm{F}}(\Gamma_k(t))$ は，（機械 2 上で遊休時間を最適に挿入した際の）$\Gamma_k(t)$ の仕事の完了コスト和，$f'_j(\Gamma_k(\cdot), t)$ は，

$$f'_j(\Gamma_k(\cdot), t) = \begin{cases} \infty, & t < P_1(\Gamma_k(\cdot)) + p_{j1} + p_{j2} \\ f_j(t), & t \ge P_1(\Gamma_k(\cdot)) + p_{j1} + p_{j2} \end{cases} \tag{18}$$

である．ここで，$P_1(\Gamma_k(\cdot))$ は，部分列 $\Gamma_k(\cdot)$ に含まれる仕事の機械 1 上での総処理時間を表す．また，$F_{\mathrm{F}}(\Lambda^{\mathrm{PI}}_{ik}(t)) = F_{\mathrm{F}}(\Lambda^{\mathrm{PI}}_{ikk}(t))$ は，

$$F_{\mathrm{F}}(\Lambda^{\mathrm{PI}}_{iki}(t)) = \begin{cases} \infty, & t < P_2(\Lambda^{\mathrm{PI}}_{iki}(\cdot)) \\ \min\Big\{ F_{\mathrm{F}}(\Lambda^{\mathrm{PI}}_{iki}(t-1)), & \\ \quad F_{\mathrm{F}}(\Gamma_{i-1}(t-p_{\lambda_{ii} 2})) & \\ \quad\quad + f'_{\lambda_{ii}}(\Gamma_{i-1}(\cdot), t) \Big\}, & \\ P_2(\Lambda^{\mathrm{PI}}_{iki}(\cdot)) \le t \le T & \end{cases} \tag{19}$$

$$F_{\mathrm{F}}(\Lambda^{\mathrm{PI}}_{ikj}(t)) = \begin{cases} \infty, & t < P_2(\Lambda^{\mathrm{PI}}_{ikj}(\cdot)) \\ \min\Big\{ F_{\mathrm{F}}(\Lambda^{\mathrm{PI}}_{ikj}(t-1)), & \\ \quad F_{\mathrm{F}}(\Lambda^{\mathrm{PI}}_{ik,j-1}(t-p_{\lambda_{ij} 2})) & \\ \quad\quad + f'_{\lambda_{ij}}(\Lambda^{\mathrm{PI}}_{ik,j-1}(\cdot), t) \Big\}, & \\ P_2(\Lambda^{\mathrm{PI}}_{ikj}(\cdot)) \le t \le T & \\ j = i+1,\ldots,k \end{cases} \tag{20}$$

で計算される．

計算量は，遊休時間が存在する 1 機械スケジューリング問題に対する dynasearch と同様であり，また，コスト関数の区分線形性による計算の効率化も同様に適用可能である．さらに，EBSR，EFSR も考慮した enhanced dynasearch も同様に構成できる．

6. iterated dynasearch

dynasearch は局所探索法であるため，そのまま単独で適用しても，局所最適解に陥ってしまい効率的な解探索を

行えない可能性がある．そこで本節では，dynasearch を反復局所探索法の枠組で用いる iterated dynasearch[1] について説明する．

基本的な (enhanced) dynasearch では，適当な初期解からスタートして，(enhanced) dynasearch 近傍内の最良解に遷移していき，解が改善しなくなると終了する．iterated dynasearch では，局所最適解に到達し解が改善しなくなった際に，解に摂動を加える（kick とよぶ）ことで探索を継続する．ただし，一定回数 kick および dynasearch を繰り返した後は，現在までに求まっている最良解に遷移し（backtrack とよぶ），そこから再び kick および dynasearch を繰り返す．

kick には，文献 [1] と同様，random swap（2 仕事をランダムに選んでその処理順序を入れ替える）を適用するものとし，その回数 α をパラメータとして与える．α を大きくすると摂動幅が大きくなるため，局所解の情報が失われる可能性があるが，逆に小さくしすぎると，局所解から抜け出せなくなる．また，backtrack の適用頻度を決定するパラメータ β も与える．kick，dynasearch の適用 β 回ごとに一回 backtrack を適用する．さらに，総反復回数 γ もパラメータとして与える．

まとめると，iterated dynasearch のアルゴリズムは以下のようになる．

手順 0（初期化） 初期解を生成し，最良解 $\Pi_n^{\rm best}$ および現在解 $\Pi_n^{\rm cur}$ を初期解にセット．反復回数 $i:=1$.

手順 1（探索） 現在解 $\Pi_n^{\rm cur}$ を (enhanced) dynasearch で改善する．

手順 2（最良解の更新） もし $\Pi_n^{\rm cur}$ が $\Pi_n^{\rm best}$ よりよい解なら，$\Pi_n^{\rm best}:=\Pi_n^{\rm cur}$ とする．

手順 3（終了判定） もし $i=\gamma$ なら，$\Pi_n^{\rm best}$ を得られた解として出力し，終了．

手順 4 (backtrack) もし $i \bmod \beta=0$ なら，$\Pi_n^{\rm cur}:=\Pi_n^{\rm best}$ とする．

手順 5 (kick) $\Pi_n^{\rm cur}$ に random swap を α 回適用する．

手順 6（次の反復） $i:=i+1$ とし，手順 1 へ．

7. 数値実験

前節までで述べた解法を重み付き納期遅れ和 (total weighted tardiness) 最小化，重み付き納期ずれ和 (total weighted earliness-tardiness) 最小化の例題に適用し，その有効性を検討する．コスト関数 $f_i(t)$ は，重み付き納期遅れ和最小化問題では

$$f_i(t)=\max(w_i(t-d_i),0) \tag{21}$$

に，重み付き納期ずれ和最小化問題では

$$f_i(t)=\max(e_i(d_i-t),w_i(t-d_i)) \tag{22}$$

になる．ただし，w_i，e_i は，それぞれ納期遅れ，納期進みに対する重みである．

重み付き納期遅れ和最小化問題の例題作成方法は以下の通りである．まず，処理時間 p_{i1}, p_{i2} をそれぞれ [1,100] の一様乱数（整数）で生成する．続いて，Johnson 法 [8] を用いて，最大完了時刻 $\max_{1\le i\le n}C_{i2}$ の最小値 $C^*_{\max}$ を求める．そして，納期 d_i を，τ, R をパラメータとして $[\max(C^*_{\max}(1-\tau-0.5R),0),C^*_{\max}(1-\tau+0.5R)]$ の一様乱数（整数）で生成する．さらに，納期遅れの重み w_i を [1,10] の一様乱数（整数）で生成する．

τ，R をそれぞれ 0.2, 0.4, 0.6, 0.8, 1.0 と変化させ，一組の n，τ，R に対し，乱数系列を変えて 5 題の問題を作成する．すなわち，各 n について 125 題の例題を作成することになる．仕事数 n は $n=10$, 20, 50, 100 と選ぶ．

重み付き納期ずれ和最小化問題の例題は，重み付き納期遅れ和最小化問題と同様の方法で生成するが，納期進みの重み e_i も [1,10] の一様乱数（整数）で生成する点，R を $R=0.4$, 0.7, 1.0, 1.3, 1.6 と変化させる点が異なる．

計画期間 T は，重み付き納期遅れ和最小化の場合は

$$T=\sum_{i=1}^{n}p_{i1}+\sum_{i=1}^{n}p_{i2} \tag{23}$$

重み付き納期ずれ和最小化の場合は

$$T=\max\left(\max_{1\le i\le n}d_i,\ \sum_{i=1}^{n}p_{i1}\right)+\sum_{i=1}^{n}p_{i2} \tag{24}$$

とした．これらの条件を満たす最適スケジュールが必ず存在することは，容易に確かめられる．

これらの例題に iterated dynasearch（以下 ID と略），iterated enhanced dynasearch（以下 IED と略）を適用する．ただし，dynasearch，enhanced dynasearch においては，$f_i(t)$ の区分線形性を利用した高速化（**4.** 参照）を行うものとする．また，初期解は，納期の非減少順，および $(p_{i1}+p_{i2})/w_i$ の非減少順に並べたスケジュールの評価のよい方を用いた．そして，総反復回数 γ は $\gamma=50$ とした．計算には Pentium4 3.4GHz のデスクトップコンピュータを用いた．

まず，パラメータ (α,β) を決定するため，α は 1 から 15，β は 1 から 8 まで，それぞれ変化させて，平均目的関数値が最も小さくなる組合せを求めた．結果を Table 1 に示す．表よりわかるように，仕事数が増えるにしたがい，パラメータ依存性が強まっていく．また，backtrack のパラメータは，文献 [1] の「$\beta=5$ が最良」という結果と異なり，ほとんどの例題について $\beta=1$ が最良という結果になった．この理由として，本研究における総反復回数 γ は 50 と少ないこととが挙げられる．このため，解空間を広範囲に探索するよりも，$\beta=1$ として毎回 backtrack を適用し，最良解の周辺を集中的に探索する方が，よりよい結果が得られたのではないかと考えられる．また，文献 [1] の 1 機械重み付き納期遅れ和最小化問題と比較して，本研究の 2 機械順列フローショップ問題の方が，dynasearch にとって難しい問題であるこ

Table 1 Choice of parameters (α, β)

n	tardiness		earliness-tardiness	
	ID	IED	ID	IED
10	103 pairs	112 pairs	103 pairs	all pairs
20	(7,1)	32 pairs	(4, 8), (8,1), (9,2)	90 pairs
50	(9,1)	(12,1)	(11,1)	(10,2)
100	(10,1)	(6,1)	(12,1)	(11,1)

Table 2 Comparison of computation time and solution quality

(a) Total weighted tardiness

n	ID			IED		
	time	best	quality	time	best	quality
10	0.01	125	0.00	0.02	125	0.00
20	0.11	119	0.08	0.26	125	0.00
50	1.91	62	0.22	6.54	99	0.05
100	16.17	42	0.75	34.27	59	0.09

(b) Total weighted earliness-tardiness

n	ID			IED		
	time	best	quality	time	best	quality
10	0.02	125	0.00	0.03	125	0.00
20	0.21	125	0.00	0.51	125	0.00
50	9.32	76	0.15	24.69	109	0.01
100	127.02	30	0.40	313.27	74	0.05

とも考えられる．

つぎに，(α, β) を Table 1 のように選んだ際の結果を Table 2 に示す．表中,「time」は平均計算時間 [s] を,「best」は，すべてのパラメータ設定を通じて得られた最良解が，Table 1 のパラメータ設定で 125 問中何問に対して求まったかを，さらに「quality」は，最良解からの誤差（1 –（最良解の目的関数値）/（得られた解の目的関数値））の平均 [%] を，それぞれ表している．ただし，仕事数 10 の場合のみ，混合整数計画問題として定式化した問題（付録参照）を商用のソルバである ILOG CPLEX9.1 により解いて得られた最適解を用いて評価している．なお，最適解を求めるのに，重み付き納期遅れ和最小化の場合は平均 1722 s，重み付き納期ずれ和最小化の場合は平均 790 s かかった．

表より，10 仕事の問題については，ID，IED のいずれを用いても全問最適解が求まっていることが確認できる．また，ID よりも IED の方が計算時間はかかるものの，よりよい解が得られている．ただし，いずれの方法でも，仕事数が増えると最良解が得られる割合が低下する．納期ずれ和最小化問題の方が納期遅れ和最小化問題よりも計算時間がかかっているのは，コスト関数の形状がより複雑であるため，区分線形関数の演算を行う際に考慮すべき端点数が増えるのが原因と考えられる．

Table 3 Comparison of computation time and solution quality (Single run without iteration)

(a) Total weighted tardiness

n	dynasearch			enhanced dynasearch		
	time	best	quality	time	best	quality
10	0.00	71	5.40	0.00	102	0.61
20	0.00	38	4.00	0.00	84	0.99
50	0.02	23	4.34	0.05	37	1.63
100	0.19	24	4.39	0.60	31	2.17

(b) Total weighted earliness-tardiness

n	dynasearch			enhanced dynasearch		
	time	best	quality	time	best	quality
10	0.00	92	1.68	0.00	118	0.37
20	0.00	45	3.33	0.00	82	0.68
50	0.05	7	4.54	0.14	40	1.43
100	0.88	3	4.95	2.35	18	1.71

つぎに，反復局所探索の効果を調べるため，反復を行わず dynasearch および enhanced dynasearch を 1 回のみ適用した（すなわち，総反復回数を $\gamma = 1$ とした）結果を Table 3 に示す．表中の値の意味は Table 2 と同様である．表より，仕事数が少ないときにはある程度よい解が求まるものの，仕事数が多くなるにつれて解が悪化していくことがわかる．Table 2 の結果と比較すると，たとえば IED では，精度が 0.1% 未満であるのに対し，反復を行わない enhanced dynasearch では 2% を超える場合もある．このように，反復局所探索の枠組により，単体の (enhanced) dynasearch に比べ，解の精度を大幅に向上させることができる．

つづいて，最良解が得られる問題をより詳細に検討するため，仕事数 100 の問題 125 問について，IED により最良解が得られた問題数と，問題パラメータ (τ, R) との関係を Table 4 に示す．表より，τ が大きい，すなわち納期が厳しい問題ほど最良解数が減少しており,（少なくとも IED にとって）難しい問題であることがわかる．

最後に，ID と IED の性能を公平に比較するため，総反復回数 γ を 150 に増やした ID を適用した．これは，同じ総反復回数の場合，IED は ID の約 3 倍計算時間がかかるためである（Table 2 参照）．このことは，dynasearch では PI 操作のみを，enhanced dynasearch では PI，EBSR，EFSR 操作を適用していることからも予想できる結果である．

総反復回数 $\gamma = 150$ の ID のパラメータ (α, β) は，Table 1 と同様の方法で得られた Table 5 にしたがって再調整し，総反復回数 $\gamma = 50$ の IED と比較した結果が Table 6 である．表中，$>$，$=$，$<$ はそれぞれ，ID で得られた解の方が IED より悪かった，同じだった，よかった問題数を表している．明らかに，IED の方が ID よりもよい結果が得られていることがわかる．これは，1 機

Table 4 Relation between the instance parameters (τ, R) and the number of best solutions obtained (IED, $n=100$)

(a) Total weighted tardiness

$\tau\backslash R$	0.2	0.4	0.6	0.8	1.0
0.2	5	5	5	5	5
0.4	3	4	5	5	5
0.6	2	0	4	3	2
0.8	0	0	1	0	0
1.0	0	0	0	0	0

(b) Total weighted earliness-tardiness

$\tau\backslash R$	0.4	0.7	1.0	1.3	1.6
0.2	5	4	5	5	5
0.4	4	4	5	5	5
0.6	2	1	4	4	5
0.8	1	0	2	2	3
1.0	0	0	1	1	1

Table 5 Choice of parameters (α, β) for ID with $\gamma=150$

n	tardiness	ealiness-tardiness
10	112 pairs	112 pairs
20	(5,4), (6,2), (8,1)	34 pairs
50	(11,1)	(10,2)
100	(6,1)	(8,1)

Table 6 Comparison of ID ($\gamma=150$) and IED ($\gamma=50$)

(a) Total weighted tardiness

	time (ID)	time (IED)	$>$	$=$	$<$
10	0.03	0.02	0	125	0
20	0.28	0.26	0	125	0
50	6.26	6.54	41	76	8
100	32.69	34.27	58	42	25

(b) Total weighted earliness-tardiness

n	time (ID)	time (IED)	$>$	$=$	$<$
10	0.05	0.03	0	125	0
20	0.63	0.51	0	125	0
50	26.74	24.69	30	90	5
100	283.99	313.27	65	45	15

械重み付き納期遅れ和最小化問題に対する文献[3]の結果と同様であり，近傍サイズを広げることで探索性能が向上していることが確かめられた．

なお，Table 6では，$n=100$の場合にIDの方が計算時間が多少短いため，比較の公平性に少々難があるが，IDの総反復回数をさらに増やして$\gamma=200$とし，$\gamma=50$のIEDより計算時間を長く取った実験も行ったものの（詳細は省略），やはり$\gamma=50$のIEDには及ばなかった．

8. おわりに

本研究では，1機械スケジューリング問題に対する局所探索法の一種であるdynasearchを，2機械順列フローショップ問題へ拡張した．そして，反復局所探索の枠組でdynasearchを用いるiterated (enhanced) dynasearchを，納期遅れ和最小化問題，納期ずれ和最小化問題に適用する数値実験を行った．その結果，とくにiterated enhanced dynasearchは有効であることが示された．しかし，仕事数が多くなると，探索性能が反復局所探索のパラメータ設定に左右されることもわかった．

現在のアルゴリズムでは計算時間がかかるため，総反復回数をあまり増やすことができない．この総反復回数の少なさが，パラメータ依存性につながっていると考えられる．そこで，今後は計算時間の短縮を行っていく必要がある．また，パラメータの自動調整や，dynasearchにおける近傍操作の追加による探索性能の向上なども必要であろう．

dynasearchの枠組は，1機械スケジューリング問題や，今回対象とした2機械順列フローショップ問題だけでなく，より広範なスケジューリング問題に対しても有効であると期待される．したがって，今後，2機械順列フローショップ問題において処理開始可能時刻に制限のある問題や，3機械以上の問題などへ本解法を拡張していくことが，理論・実用の両面において重要であると考えられる．

参考文献

[1] R. K. Congram, C. N. Potts and S. L. van de Velde: An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem; *INFORMS Journal on Computing*, Vol. 14, pp. 52–67 (2002)

[2] OR-Library: http://people.brunel.ac.uk/˜mastjjb/jeb/info.html

[3] A. Grosso, F. Della Croce and R. Tadei: An enhanced dynasearch neighborhood for the single-machine total weighted tardiness scheduling problem; *Operations Research Letters*, Vol. 32, pp. 68–72 (2004)

[4] Ö. Ergun and J. B. Orlin: Fast neighborhood search for the single machine total weighted tardiness problem; *Operations Research Letters*, Vol. 34, pp. 41–45 (2006)

[5] F. Sourd: Dynasearch for the earliness-tardiness scheduling problem with release dates and setup constraints; *Operations Research Letters*, Vol. 34, pp. 591–598 (2006)

[6] M. R. Garey, D. S. Johnson and R. Sethi: The complexity of flowshop and jobshop scheduling; *Mathematics of Operations Research*, Vol. 1, pp. 117–130 (1976)

[7] F. Sourd: Optimal timing of a sequence of tasks with general completion costs; *European Journal of Oper-*

ational Research, Vol. 165, pp. 82–96 (2005)
[8] S. M. Johnson: Optimal two and three stage production schedules with setup times included; *Naval Research Logistics Quartry*, Vol. 1, pp. 61–68 (1954)

付　　録

混合整数計画問題としての定式化

重み付き納期ずれ和を最小化する 2 機械順列フローショップ問題は，以下の混合整数計画問題として定式化できる．なお，重み付き納期遅れ和の場合も同様である．

$$\min \sum_{i=1}^{n}(e_iE_i+w_iT_i) \tag{A1}$$

$$\text{s.t. } C_{i2}-T_i+E_i=d_i, \quad 1\le i\le n \tag{A2}$$

$$C_{j1}-p_{j1}x_{ij}\ge C_{i1}-M(1-x_{ij}), \quad 1\le i,j\le n,\ i\ne j \tag{A3}$$

$$C_{j2}-p_{j2}x_{ij}\ge C_{i2}-M(1-x_{ij}), \quad 1\le i,j\le n,\ i\ne j \tag{A4}$$

$$C_{i1}\ge p_{i1}, \quad 1\le i\le n \tag{A5}$$

$$C_{i2}\ge C_{i1}+p_{i2}, \quad 1\le i\le n \tag{A6}$$

$$\sum_{i=0}^{n}x_{ij}=1, \quad 1\le j\le n+1 \tag{A7}$$

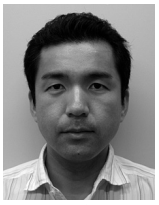
$$\sum_{j=1}^{n+1}x_{ij}=1, \quad 0\le i\le n \tag{A8}$$

$$x_{ij}\in\{0,1\}, \quad 0\le i\le n,\ 1\le j\le n+1 \tag{A9}$$

この定式化において，x_{ij} は，仕事 i の直後に仕事 j を処理するとき 1 となる決定変数，M は十分大きい定数を表す．

著 者 略 歴

田中（たなか）　俊二（しゅんじ） (正会員)



1993 年 3 月京都大学工学部電気工学第二学科卒業，2000 年 3 月同大学院工学研究科電気工学専攻博士後期課程修了．同年 4 月同助手，2006 年 4 月同助教となり現在に至る．生産スケジューリング問題，エレベータ群管理問題など最適化問題のモデル化および解法の研究に従事．博士（工学）．IEEE，計測自動制御学会，電気学会，日本オペレーションズ・リサーチ学会，スケジューリング学会の各会員．