社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－AL－108 (8)
2006／9／27

衝突確率を考慮したバッファ配置問題に対する計算機シミュレーションを利用した手法

千葉英史† 浅野哲夫†† 茨木俊秀†

液晶や半導体の製造装置には，衝突回避のために，製造ライン上に通常十分にバッファが装備されているが，実際には余分なバッファも少なくない．一方，コスト面から見ると，できるだけバッファ数を減らすことが求められている．本論文では，ジョブの投入間隔が一定の直列システムにおいて，各機械のバッファ数を任意に設定できるとき，計算機シミュレーションにより衝突確率を求める手法を提案する．この手法を利用することで，与えられた衝突確率を満たすようなバッファ配置の中で，総バッファ数ができるだけ少ない配置を実時間内で求めることができる．また，代表的な数値結果も与える．

A Method for the Buffer Allocation Problem with Collision Probability Using Computer Simulation

EISHI CHIBA,† TETSUO ASANO†† and TOSHIHIDE IBARAKI†

Most manufacturing systems for liquid crystal panels or integrated circuits are usually equipped with enough buffer numbers to avoid collision between workpieces. However, they often contain redundant buffers which are not actually used. In order to reduce the production cost, the number of buffers should be decreased as possible. In this paper, assuming a constant inter-arrival time of workpieces, we present a computer simulation method for computing the collision probability on an in-line system in which each machine has an arbitrary size of buffer. Then, we apply the method to obtain a buffer allocation that achieves the smallest total number of buffers under an arbitrarily specified collision probability. We also present some simulation results.

1. はじめに

液晶や半導体の製造工程の最適制御は，製造の担当者が抱える大きな問題の一つである．半導体や液晶を形成するための基板（ガラス基板）の大型化や技術進歩に伴い，生産ラインも日々変化し，新たな課題も生じている．最近，Crystal Flow のような高度なシステムが開発されているが[5]，これは既存ラインだけでなく，次世代生産工程におけるより高度なライン制御を目指して，進化していくものである．

液晶や半導体の製造工程を大まかに捉えると，直列につながった処理装置によって，ガラス基板が順番に処理されていく流れと見なすことができる．それぞれの処理装置は，洗浄，成膜，レジストコーディング，露光，現像，エッチングなどの専用装置である．これらすべての処理装置の処理時間が一定値 T であるならば，最適なライン制御は容易である．なぜなら，毎回時間 T の間隔で，ガラス基板をシステムへ投入すれば十分だからである．しかし，装置の処理時間は，様々な原因（例えば，熱処理，薬品処理などの影響のため）で一定とならず，流れてくるガラス基板毎に異なってしまう．

システム稼動中に，ガラス基板がある処理装置に到着したとき，装置が別のガラス基板を処理している最中であれば，装置のバッファへと一時的に退避する．装置のバッファもすべて別のガラス基板で満たされていれば，運ばれてきたガラス基板の行き場所が無い．この現象を衝突と呼ぶ．スケジューリング理論では，従来からこのような現象をブロッキングと呼んでいる[3]．ガラス基板は，高価でもろいので，できるだけ衝突を避けることが求められている．

† 関西学院大学理工学研究科情報科学専攻
Department of Informatics, School of Science and Technology, Kwansei Gakuin University
E-mail: {e-chiba, ibaraki}@ksc.kwansei.ac.jp
†† 北陸先端科学技術大学院大学情報科学研究科
School of Information Science, Japan Advanced Institute of Science and Technology
E-mail: t-asano@jaist.ac.jp

それゆえ，実際にシステムを稼動する前に，予め衝突確率を見積もっておくことは大事である．得られた衝突確率を考慮して，ガラス基板のシステムへの投入間隔や，各装置のバッファ数などの修正も出来る．しかし，これらを理論的に解析するのは，非常に難しいといえる．そのため現場では，衝突を考慮した様々なシステム制御を過去の経験をもとに行っており，著者らの知る限り理論的な研究はあまりなされていない．

しかし近年，文献 1)，2) ではガラス基板の投入間隔が一定の直列システムにおいて，バッファサイズが 0 であるとして，衝突確率を求める公式を与えた．処理時間が指数分布のときはその確率を与える閉じた形が得られ，正規分布などの他の一般分布でも数値的に計算可能である，というところが特徴である．指数分布を仮定すると，解析結果は閉じた形で表されるが，実際の処理時間を必ずしも正確に表現していないと考えられる．文献 6) では，指数分布をアーラン分布に一般化し，やはり閉じた形で衝突確率を表せることを示した．いずれの場合も，バッファがある場合の理論的な解析は困難であるため，文献 1) では，各機械のバッファサイズが 1 の場合に，計算機シミュレーションにより衝突確率を求めた．

衝突確率を求めることが出来るようになると，より実用的な問題を考えることができるようになる．例えば，文献 1) では，衝突確率を考慮したタクトタイム最小化問題を定式化し，この問題に対する効率の良い擬似二分探索法を提案した．他には，各装置にどの程度バッファを装備させるか，という問題が考えられる．製造装置には，通常十分にバッファが装備されているが，実際には余分なバッファも少なくない．一方，コスト面から見ると，できるだけバッファ数を減らすことが求められている．一般に衝突確率とバッファ数の間にはトレードオフの関係があるが，本論文では，両方を考慮した問題を提案する．すなわち，与えられた衝突確率を満たすようなバッファ配置の中で，総バッファ数が最も少ない配置を求める問題である．あらかじめタクトタイムが決まっているときに，この問題は特に重要になってくる．

本論文では，ガラス基板の投入間隔が一定の直列システムにおいて，各機械のバッファサイズを任意に入力として与えるとき，計算機シミュレーションにより衝突確率を求める手法を提案する．本手法は，処理時間の分布によらず，適用することができる．また，衝突確率を考慮しつつ最適バッファ配置を求めるために，計算機シミュレーションを利用した近似手法を提案して，計算機実験によりその効果を確かめる．

第 2 章では，モデルの説明と問題の定式化を行う．第 3 章では，衝突確率を求めるための計算機シミュレーションの手続きを示す．また，その手続きを利用して，衝突確率付きバッファ配置問題を高速に解く近似手法を述べる．第 4 章では，代表的な数値結果を示し，考察する．第 5 章では，結論と今後の課題を述べる．

2. 問題の定式化

本論文で扱う問題を定義するために以下の変数を準備する．

- $M_1, M_2, \ldots, M_m$: m 台の直列に並んだ異なる機械．
- $J_1, J_2, \ldots, J_n$: n 個のジョブ．
- $t_i^{(j)}$ (> 0): 機械 M_j におけるジョブ J_i の処理時間．
- T_{tact} (> 0): タクトタイム，すなわち，2 つの連続するジョブの入口からの投入間隔．

典型的なバッファ付きの直列システムを図 1 に示す．各ジョブは，入口から T_{tact} 間隔でシステムへ投入されて，まずは機械 M_1 で処理される．M_1 で処理が終了すると，順次 $M_2, \ldots, M_m$ へと直列に並んだ m 台の機械で処理されていき，最後に出口へと到着する．ジョブがある機械へと到着した際に，その機械が他のジョブを処理している最中であれば，一時的にバッファへと退避する．これは，FIFO 規律の待ち行列が機械の前にあるとみなせばよい．機械が空になると，自動的にバッファから機械へと運ばれて，処理が開始される．このとき，待ち行列の長さは，その機械が装備しているバッファの個数以内でなくてはならない．もし，待ち行列の長さがバッファの個数と等しいとき，すなわち，バッファが満杯のときに，新たなジョブが到着すると，ジョブの行き場所がなくなってしまう．この現象が衝突である．

各機械上のジョブの処理時間に関して，前もって正確な値を把握することはできず，処理が終わってはじめて知ることができる．しかし，過去の経験から，処理時間に関する分布を手に入れることが出来るので，本研究では，処理時間を確率変数として扱う．

まず，後述の問題 2 および問題 3 を解くのに必要なので，各機械が無限にバッファを装備している場合のスケジューリング問題を説明する．ジョブ J_i のスケジュールとは写像 $S : J_i \mapsto (st_i^{(1)}, ft_i^{(1)}, st_i^{(2)}, ft_i^{(2)}, \ldots, st_i^{(m)}, ft_i^{(m)})$ である．ただし，$st_i^{(j)}$ と $ft_i^{(j)}$ はそれぞれジョブ J_i が機械 M_j で処理を開始する時刻，および処理を完了する時刻である．与えられた $t_i^{(j)}$

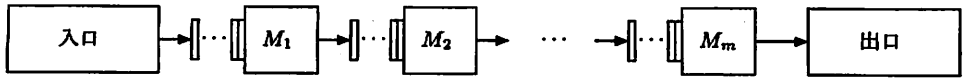

図 1 バッファ付きの m 台の機械から成る直列システム

に対して，以下の条件が満たされる時，スケジュール S は実行可能であると言う．

(i) $st_i^{(j)} + t_i^{(j)} = ft_i^{(j)}$ $(i \in I_n, j \in I_m)$,
(ii) $ft_i^{(j)} \leq st_i^{(j+1)}$ $(i \in I_n, j \in I_{m-1})$,
(iii) $ft_i^{(j)} \leq st_{i+1}^{(j)}$ $(i \in I_{n-1}, j \in I_m)$,
(iv) $st_i^{(j)}, ft_i^{(j)} \geq 0$ $(i \in I_n, j \in I_m)$,
(v) $st_{i+1}^{(1)} - st_i^{(1)} = T_{tact}$ $(i \in I_{n-1})$.

ただし，I_k は集合 $\{1, 2, \ldots, k\}$ のことである．

次のようにスケジューリング問題を記述できる．

問題 1 メイクスパン最小化スケジューリング問題

入力: ジョブ数 n，機械数 m，処理時間 $t_i^{(j)}$ $(i \in I_n, j \in I_m)$，タクトタイム T_{tact}．

目的関数: $ft_n^{(m)} \rightarrow$ 最小．

制約条件: スケジュール S は実行可能である．

この問題は動的計画法により解くことができる．具体的には，st と ft 用の 2 次元配列を準備して，次の関係式

$$st_i^{(j)} = \max\{ft_i^{(j-1)}, ft_{i-1}^{(j)}\} \quad (i \in I_n, j \in I_m),$$

$$ft_i^{(j)} = \begin{cases} 0 & (i = 0, j \in I_m), \\ (i-1)T_{tact} & (i \in I_n, j = 0), \\ st_i^{(j)} + t_i^{(j)} & (i \in I_n, j \in I_m) \end{cases}$$

から，st と ft の表を埋めると解くことができる．従って，計算時間は $O(mn)$ である．

また，求まった表を利用して，各機械で使用されたバッファ数をやはり $O(mn)$ で求めることができる．J_i が M_j に到着したとき，J_h $(h < i)$ が M_j で処理中であるための必要十分条件は，$st_h^{(j)} < ft_i^{(j-1)} < ft_h^{(j)}$ を満たす h が存在することである．また，この条件式が成立するとき，M_j で待っているジョブは $J_{h+1}, J_{h+2}, \ldots, J_i$ であるから，$i - h$ 個のバッファが使われている．ある機械 M_j において，各ジョブ $J_1, J_2, \ldots, J_n$ を順に調べていくとき，上の条件式を h が小さい順にチェックする．その際，一つのジョブに対して，上の条件式を複数回チェックすることがあるが，一度チェックからはずれた J_h は，二度とチェックする必要がない．それゆえ，M_j で使われたバッファ数を求める全体としての計算時間は，$O(n)$ でしかない．この操作を各機械で行うのである．

次に，バッファ数に制限のあるシステムを考える．機械 M_j のバッファ配置数 z_j $(\in \mathbb{Z}_+)$ とは M_j に装備されているバッファの個数である．また，総バッファ数を $Z := \sum_{j=1}^{m} z_j$ と記す．

このシステム上で衝突確率を求める問題を下に記す．

問題 2 衝突確率問題

入力: ジョブ数 n，機械数 m，タクトタイム T_{tact}，処理時間 $t_i^{(j)}$ $(i \in I_n, j \in I_m)$ の分布，バッファ配置 z_j $(j \in I_m)$，正整数 c_1．

出力: 衝突確率．

次に衝突確率を考慮して，各機械にどの程度バッファを装備させるか，という問題を定式化する．すなわち，衝突確率に関する制約条件もとで，総バッファ数 $Z = \sum_{j \in I_m} z_j$ を最小化する問題である．

問題 3 衝突確率を考慮したバッファ配置問題

入力: ジョブ数 n，機械数 m，タクトタイム T_{tact}，処理時間 $t_i^{(j)}$ $(i \in I_n, j \in I_m)$ の分布，α $(0 \leq \alpha \leq 1)$，正整数 c_0, c_1．

目的関数: $Z \rightarrow$ 最小．

制約条件: バッファ配置 z_j $(j \in I_m)$ による衝突確率は α 以下である．

上の制約条件を満たすバッファ配置は実行可能であるという．実行可能なバッファ配置の中で，Z を最小にするバッファ配置を最適バッファ配置と呼び，その時の $Z = Z_{opt}$ を最適総バッファ数と呼ぶ．

3. 計算機シミュレーション

本章では，前章で述べた問題 2 と問題 3 を計算機シミュレーションを利用して解く手法を述べる．まず，問題 2 を解く手続きを下に示す．

Procedure Collision_Probability

Step 1: $loop := 1$.

Step 2: 処理時間 $t_i^{(j)}$ $(i \in I_n, j \in I_m)$ の擬似乱数を生成する．各機械がバッファを無限に持つと仮定して，問題 1 を解くことにより，各機械で使用されたバッファ数 z'_j $(j \in I_m)$ を求める．

Step 3: Step 2 で求めた z'_j $(j \in I_m)$ と，入力で与えられたバッファ数 z_j $(j \in I_m)$ を比較して，衝

突が生じるかどうかを判定する．$loop := loop+1$．$loop \leq c_1$ ならば，Step 2 へ．そうでないなら，Step 4 へ．

Step 4: 衝突確率（すなわち，Step 3 で衝突が生じた回数 $/c_1$）を出力

Step 3 での衝突の有無の判定は，$z_j < z'_j$ を満たす j $(\in I_m)$ が存在するならば，衝突有りと判定し，そうでないならば，衝突無しと判定する．Step 2 と Step 3 は c_1 回繰返し実行されるが，c_1 の値が大きいほど，Step 4 で信頼できる衝突確率を得ることができる．計算時間は，$O(c_1(mn+m)) = O(c_1mn)$ であり，c_1 を定数扱いすれば，$O(mn)$ となる．

次に問題 3（衝突確率を考慮したバッファ配置問題）に対する近似手法を下に示す．この手続きは，手続き Collision_Probability を含んでいる．また，問題 3 を厳密に解いているわけではないが，局所的に最適バッファ配置を求める．基本的にはどの機械もバッファを一つも持たない状態から始めて，衝突確率に関して最も効果のある機械に効率良くバッファを配置していく．

Procedure Buffer_Allocation

Step 1: 衝突がほとんど生じないようなバッファ配置より始める．そのときの M_j のバッファ数を z_j^* $(j \in I_m)$ とする．また，$z_j := 0$, $left_j := 0$, $right_j := z_j^*$ $(j \in I_m)$, $h := 0$.

Step 2: z_j $(j \in I_m)$ を入力として，手続き Collision_Probability を実行して，得られた衝突確率を α' とする．その際，機械 M_j で衝突が生じた回数 a_j も同時に記録しておき，$a_k = \max_{j \in I_m} a_j$ を満たす k を求める．

Step 3: $\alpha < \alpha'$ ならば，$left_k := z_k$，バッファ増加操作 $(z_k := \lceil (left_k + right_k)/2 \rceil)$，$h := k$，Step 2 へ．$\alpha \geq \alpha'$ ならば，Step 4 へ．

Step 4: $h = 0$ のとき，終了．そうでないとき，M_h のバッファ数 (z_h) から減らしていく．$k := h$ として，Step 5 へ．

Step 5: $right_k := z_k$.

Step 6: $z_k := \lfloor (left_k + right_k)/2 \rfloor$.

Step 7: z_k $(k \in I_m)$ を入力として，手続き Collision_Probability を実行して，得られた衝突確率 α' とする．

Step 8: $\alpha < \alpha'$ ならば，$left_k := z_k + 1$．そうでないなら，$right_k := z_k - 1$.

Step 9: $left_k \leq right_k$ ならば，Step 6 へ．そうでないなら，Step 10 へ．

Step 10: $z_k := left_k$.

Step 11: すべての $k \in I_m \setminus \{h\}$ に対して，Steps 5–10 を実行する．

Step 1 で z_j^* $(j \in I_m)$ の計算は次のように行う．各機械がバッファを無限に持つと仮定して，処理時間 $t_i^{(j)}$ を分布にしたがってランダムに発生しては，全てのジョブのスケジューリングを行い，各機械で実際に使われたバッファ数を計算する．これは，問題 1 を解くことに対応する．この計算を c_0 回繰返し，各 M_j で使用されたバッファ数の中で最大のバッファ数を z_j^* とする．すなわち，シミュレーション結果では，z_j^* $(j \in I_m)$ による配置の衝突確率は 0 である．

また，z_j は，$left_j$ 以上 $right_j$ 以下の範囲内をとる．どの機械もバッファを一つも持っていない $z_j = 0$ $(j \in I_n)$ から始め，衝突確率に最も影響するところに $left_j \leq z_j \leq right_j$ の範囲内でバッファを配置していくのである．Step 2 と Step 3 は，衝突確率が α 以下である，という制約条件を満たすまで，バッファ数を増やす局面である．また，どの機械のバッファを増やしたかをそのつど h に記録しておく．

衝突確率に関する制約条件を満たしたとき，Step 4 へと進む．このとき，直前にバッファを増やした場所 M_h で，必要以上に多く配置した可能性がある．そこで，$left_h$ と $right_h$ の範囲間で二分探索を行い，適切な z_h の値を求める．すなわち，この値よりも少なくすると，衝突確率に関する制約条件を満たさなくなってしまう．他のバッファ配置 z_k（ただし，$k \in I_m \setminus \{h\}$）に関しても同様の処理を施す．このような局面が残りの Steps 5–11 に対応する．

手続き Buffer_Allocation の実行後，得られたバッファ配置 z_j $(j \in I_m)$ は明らかに制約条件を満たしている．さらに，どの z_j を減らしても制約条件を満たさなくなる．それゆえ，得られた z_j $(j \in I_m)$ は局所的最適バッファ配置である．

Step 1 の計算時間は，問題 1 を解く繰り返し回数 c_0 を用いると，$O(c_0mn)$ である．Step 2 と Step 7 の計算時間は，先に述べたように $O(c_1mn)$ である．残りの各 Step は定数時間である．また，Steps 2–3 の繰返し回数と Steps 5–10 の全体としての繰返し回数は，高々 $\log z_1^* + \cdots + \log z_m^* = \log \prod_{j=1}^m z_j^*$ でしかない．よって，全体の計算時間は $O(c_0mn + c_1mn \log \prod_{j=1}^m z_j^*)$ であり，c_0 と c_1 を定数扱いすれば，$O(mn \log \prod_{j=1}^m z_j^*) = O(m^2n \max_{j \in I_m} \log z_j^*)$ となる．

バッファの増やし方，あるいは減らし方には上の手続きで示したやり方以外にも，様々なバリエーションが考えられる．しかし，計算時間の観点から適当な

表 1　手続き Collision_Probability による衝突確率と計算時間

バッファ配置	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	Z	衝突確率 (%)	計算時間 (sec.)
1	1	1	1	1	1	1	1	1	8	100	14.093
2	2	2	1	1	1	1	1	1	10	99.95	14.094
3	2	4	2	1	1	1	1	1	13	99.94	14.078
4	3	2	2	1	1	1	1	1	12	52.45	14.094
5	3	2	2	2	1	1	1	1	13	52.34	14.094
6	3	2	2	2	2	1	1	1	14	52.32	14.078
7	3	2	2	2	2	2	1	1	15	52.3	14.094
8	4	2	2	1	1	1	1	1	13	0.24	14.078
9	4	3	2	1	1	1	1	1	14	0.24	14.078
10	5	2	2	2	2	2	1	1	17	0	14.078

増やし方を選ばなくてはならない．少しずつバッファを増やしていくと，良いバッファ配置を得やすいが，その分計算時間が余計にかかる．逆に，大雑把にバッファを増やしていくと，計算時間の面で有利だが，良いバッファ配置という面で不利になる．

4. 数値結果

実験環境を以下に示す．

Machine: FMV ESPRIMO
CPU: Intel Pentium4 3.00GHz
OS: Microsoft Windows XP
Memory: 1GB
Compiler: Visual C++ .NET 2003

当面は各種パラメータは，$n = 1{,}000$, $m = 8$, $T_{tact} = 0.997$, $c_1 = 10{,}000$, $c_2 = 10{,}000$ と設定して，処理時間は正規分布 $N(1, 0.01^2)$ に従うと仮定する．処理時間の乱数を生成するため，本研究では LEDA[4] で用意されている関数を利用した．また，処理時間の乱数が負となった場合には，特例として 0 を割り当てた．

手続き Collision_Probability を実装して，衝突確率を計算した結果を表 1 に示す．表 1 は，各機械のバッファ数をいろいろ変えながら，それぞれに対して衝突確率と計算時間を示したものである．10 種類の異なるバッファ配置で実験を行った．

配置 3，配置 5，配置 8 を比較すると，どちらの配置も総バッファ数が 13 で同じになるが，入口に近い方の機械にバッファを多く装備する方が，衝突確率を下げるのに効果があると見てとれる．配置 8 と配置 9 を比較すると，どちらも衝突確率は 0.24 である．それゆえ，配置 9 には，M_2 に無駄なバッファがあることが分かる．配置 10 では，一度も衝突が生じなかった．配置 4,5,6,7 は，総バッファ数が 1 つずつ増えている．しかし，衝突確率はほぼ同じような値を示した．入口から遠い機械にバッファを多く装備しても，衝突確率にほとんど影響を与えないという様子を示している．むしろ，配置 8 のように，配置 4 の M_1 に一つバッファを増やした方が，衝突確率への効果は非常に大きい．

次に，手続き Buffer_Allocation を実装して，計算機シミュレーションを行った結果を表 2 に示す．表 2 は，与えられた衝突確率に対して，得られたバッファ配置，および計算時間を示している．衝突確率が増えるに従って，必要なバッファ数が減少する傾向を見ることができる．また，計算時間も減少するが，その理由は，Steps 2-3 の繰返し回数が少ないからである．

衝突確率 5%と 10%の結果を見ると，バッファ配置の違いは，M_2 においてのみである．ここでは，M_2 のバッファ数を増加させるのが最も衝突確率を減らすのに効果がある．実際に，このバッファ配置 (4,2,1,1,1,1,1,1) を入力として，手続き Collision_Probability を実行すると，衝突確率は 0.91%であった．一方，M_1 のバッファ数を増加させたバッファ配置 (5,1,1,1,1,1,1,1) を入力として，手続き Collision_Probability を実行すると，衝突確率は 5.25%となってしまう．このように，手続き Buffer_Allocation は，反復ごとに衝突確率に関して，最も効果のある場所にバッファを追加していることがわかる．

次に，機械 M_3 と M_6 の処理時間の平均値を 10%増やし，正規分布 $N(1.1, 0.01^2)$ に従うときに（残りのパラメータ設定は，これまでと同じ），手続き Buffer_Allocation を実行した結果を表 3 に示す．表 3 から分かるように，M_3 のバッファ数が他と比べて非常に多く，M_3 以降では，あまりバッファを必要としないことが分かった．M_3 と M_6 は同じ処理時間の分布を持つにもかかわらず，後の方の機械のバッファ数は少なくて済む．さらに，M_3 のバッファ数は表 3 全体を通して，91 あるいは 92 であり，それよりも少なくすると，衝突確率が急激に 100%になってしまうのが興味深い．このように，本シミュレーションは機

表 2　手続き Buffer_Allocation によるバッファ配置と計算時間

衝突確率 α (%)	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	Z	計算時間 (sec.)
0	5	2	2	2	2	2	1	1	17	519.734
5	4	2	1	1	1	1	1	1	12	441.078
10	4	1	1	1	1	1	1	1	11	425.422
20	4	1	1	1	1	1	1	1	11	425.234
30	4	1	1	1	1	1	1	1	11	429.015
40	4	1	1	1	1	1	1	1	11	425.031
50	4	1	1	1	1	1	1	1	11	425
60	3	1	1	1	1	1	1	1	10	425.031
70	3	1	1	1	1	1	1	1	10	425.218
80	3	1	1	1	1	1	1	1	10	425.062
90	3	1	1	1	1	1	1	1	10	425.046
100	0	0	0	0	0	0	0	0	0	30.343

表 3　手続き Buffer_Allocation によるバッファ配置と計算時間 (M_3 と M_6 の処理時間が他に比べて時間がかかる場合)

衝突確率 α (%)	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	Z	計算時間 (sec.)
0	5	2	92	0	1	2	0	1	103	491.953
5	4	2	92	0	0	1	0	0	99	399.468
10	4	1	92	0	0	1	0	0	98	384
20	4	1	91	0	0	1	0	0	97	368.656
30	4	1	91	0	0	1	0	0	97	371.765
40	4	1	91	0	0	1	0	0	97	369
50	4	1	91	0	0	1	0	0	97	368.64
60	4	1	91	0	0	1	0	0	97	368.625
70	3	1	91	0	0	1	0	0	96	368.641
80	3	1	91	0	0	1	0	0	96	368.64
90	3	1	91	0	0	1	0	0	96	368.859
100	0	0	0	0	0	0	0	0	0	29.5

表 4　手続き Buffer_Allocation によるバッファ配置 ($\alpha = 0.05$ のとき)

	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	Z
Step 4 のとき	1	1	66	11	1	43	11	1	135
全 Steps 終了時	1	1	64	11	1	34	11	1	124

表 5　手続き Buffer_Allocation によるバッファ配置 ($\alpha = 0.1$ のとき)

	z_1	z_2	z_3	z_4	z_5	z_6	z_7	z_8	Z
Step 4 のとき	1	1	66	11	1	29	11	1	121
全 Steps 終了時	1	1	61	11	1	29	11	1	116

械毎に異なる処理時間分布を持つ場合にも，対応している．

次に，表 2 と表 3 を得るのに行った実験の際，Step 4 でのバッファ配置と全ステップ終了時のバッファ配置を比較した．前章で述べたように，Step 4 でのバッファ配置は実行可能であり，残りのステップでバッファ数をできるだけ減らしていく．しかし，表 2 と表 3 の全ての場合で，どちらのバッファ配置も同じであり，すなわち，Step 4 で既に局所的最適バッファ配置を求めていることが分かった．

最後に，各種パラメータを $n = 100$, $m = 8$, $T_{tact} = 1.0$, $c_1 = 200,000$, $c_2 = 200,000$ と設定し，M_3 と M_6 の処理時間は正規分布 $N(1.1, 3^2)$，残りの機械での処理時間は正規分布 $N(1, 0.01^2)$ に従うとして実験を行った．手続き Buffer_Allocation の入力で，それぞれ $\alpha = 0.05$, $\alpha = 0.1$ としたときの結果を表 4，表 5 に示す．計算時間は，それぞれ 1240.19 秒，1235.06 秒であった．これらの結果から分かるように，一般には Step 4 でのバッファ配置と全ステップ終了時のバッファ配置は異なり得る．また，これらの結果が確かに局所的最適バッファ配置であることも確かめた．すなわち，得られたバッファ配置からどの

機械のバッファを減らしても，制約条件を満たさなくなることを，手続き Collision_Probability を使って確認した．ただし，このとき衝突確率の精度をあげるために $c_1 = 200,000$ と設定した．

前章において，手続き Buffer_Allocation は局所最適バッファ配置を求めると述べたが，(大域) 最適バッファ配置を求めることを保障していない．しかし，我々のシミュレーションの経験では，そのような例を探すのは非常に難しい．特に，実用上重要である α の値が 0 に近いときには，バッファ数の増え方も小さくなり，多くの場合一つずつ増えるので，(大域) 最適バッファ配置を計算することが期待できる．

5. おわりに

本論文では，衝突確率を考慮したバッファ配置問題を提案した．現実には，液晶や半導体の製造装置は，複数の直列システムがつながって，全体のシステムを構成している．そのため，前後のシステムとの関連から，あらかじめタクトタイムが決められている状況が出てくる．そのような場面では，バッファ配置を設計することで，衝突確率を調節することができる．

本論文で示した手法は，シミュレーション結果に基づいており，実時間で良いバッファ配置を計算するが，理論的にバッファ配置の良さを保障しているわけではない．衝突確率が z_j に関して凸関数であり，シミュレーションの精度が十分高ければ，我々の手法は最適バッファ配置を求めていることになり，今後の課題とする．さらに，実際の現場のデータを基に我々の手法の効果を確かめたい．

謝辞 関西学院大学 博士研究員 藤原洋志氏には，日頃から様々な貴重な御助言を賜りました．株式会社 FEBACS 三塚郁夫氏には，実用的な見地から様々な御意見を頂きました．深く感謝致します．

参考文献

1) E. Chiba, T. Asano, T. Miura, N. Katoh and I. Mitsuka, "Analysis of Collision Probability in Automatic Transportation Systems", submitted.
2) E. Chiba, T. Asano, T. Miura, N. Katoh and I. Mitsuka, "Modeling of Transportation Systems Using Crash Probability", Proc. 8th Korea-Japan Workshop on Algorithms and Computation, pp.142–149, Seoul, Korea, Aug. 2005.
3) M. Pinedo, Scheduling: Theory, Algorithms, and Systems, Prentice Hall, 1995.
4) http://www.algorithmic-solutions.com/enleda.htm
5) http://www.febacs.co.jp/solution/crystal.html
6) 千葉英史, 藤原洋志, 関口良行, 茨木俊秀, "生産ラインにおける衝突確率: 処理時間がアーラン分布に従う場合", 日本オペレーションズ・リサーチ学会, 2006 年秋季研究発表会．