実験計画法の並列分散処理による ブルウィップ効果における 要因の部分効用の解明

情報基盤工学講座　　横井　稜

December 19, 2019

1.1. 本研究の背景

サプライチェーンの課題の一つに、取引する人の心的要因やサプライチェーンの上流から下流への情報伝達の遅れなどの要因が重なり、サプライチェーンの上流に行くほど需要量のばらつきが増大する現象がある。その現象をブルウィップ効果という。
変動に対応するために上流のサプライヤーほど在庫を多く保有する傾向にあり、余剰在庫が経営状態を悪化させる。先行研究では、ブルウィップ効果に影響する要因を限定して考察している。

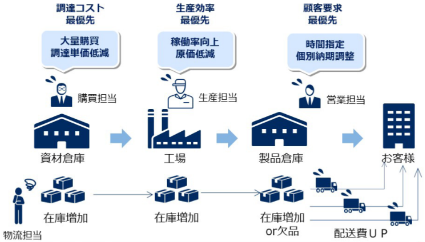


図１　サプライチェーンの問題

ブルウィップ効果には多くの複合的な要因が影響している。そこで、複合的要因を考慮した効果を明らかにする。また、要因が沢山あるので主効果だけではなくて交互作用がある。組み合わせ爆発が起こるので並列分散処理が必要である。
本研究では、シミュレータを開発し複合要因の影響度合いを明確化する。そして、ブルウィップ効果の低減方法を提案する。

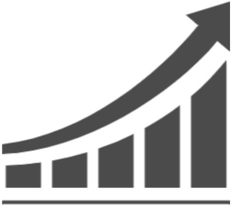
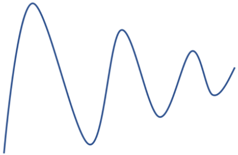


図 2　組み合わせ爆発

図 3　並列コンピュータ

2.1 サプライチェーンにおけるブルウィップ効果

$$B = \frac{\mathrm{Var}[q_t]}{\mathrm{Var}[D_t]}$$



- 自動車業界
- パソコン製造業
- 食品製造業
- 電気部品製造業
- 他にも多数

需要に鞭のようなバラツキが発生

図 4　サプライチェーンとブルウィップ効果

B: ブルウィップ効果、q_t: 発注量、D_t: 需要量
$\mathrm{Var}[q_t]$: 発注量の分散、$\mathrm{Var}[D_t]$: 需要量の分散

D. Simchi-Levi, P.Kaminsky and E. Simchi-Levi:Designing and Managing the Supply Chain Concepts,Strategyies,and Case Studies,McGraw Hill(2000)

小売業におけるブルウィップ効果

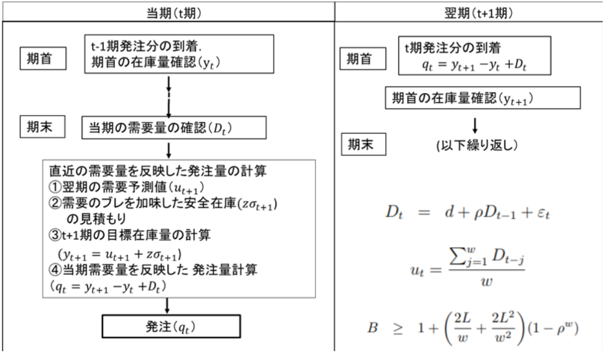


図 5　小売業の発注方法

ρ: 前期の需要量との相関を表すパラメータ，L: リードタイム
w: 移動平均における過去のデータの採取数，ε_t: 需要予測値の誤差
z: 安全在庫係数，σ_t: t 期における需要予測誤差の標準偏差の推定量
d: 需要量の平均

内示生産システムにおけるブルウィップ効果

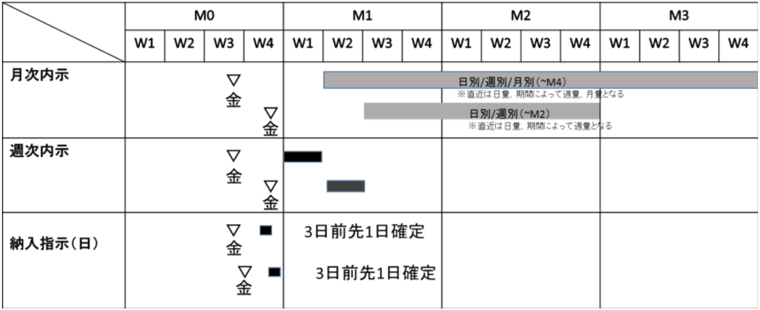


図 6　内示生産システムの発注方法

$$B_{\mathrm{w}} = \frac{M + z^2 \mathrm{Var}[(\widehat{\sigma}_{t+1} - \widehat{\sigma}_t)])}{M} \quad B_{\mathrm{m}} = \frac{2M + N + z^2 \mathrm{Var}[(\overline{\sigma}_{t+2} - \widehat{\sigma}_{t+1} - \widehat{\sigma}_t)]}{M}$$

M: t 期における週次内示と確定注文のブレの分散,
N: t 期における週次内示と月次内示のブレの分散,
$\widehat{\sigma}_t$: t 期における週次内示と確定注文のブレの標準偏差の推定量,
$\overline{\sigma}_t$: t 期における週次内示と月次内示のブレの標準偏差の推定量

2.2. ブルウィップ効果に影響する要因

実際の週次内示 $\hat{D}_t$ と確定注文 D_t との差

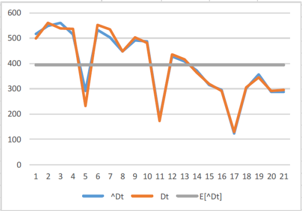


図 7　$\hat{D}_t, D_t, E[\hat{D}_t]$ のグラフ

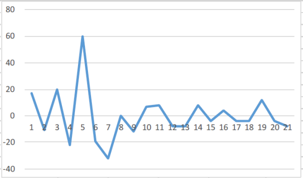


図 8　$\hat{D}_t - D_t$ のグラフ

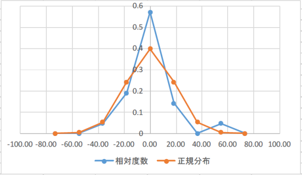


図 9　$\hat{D}_t - D_t$ の相対度数分布

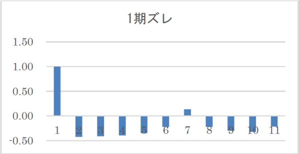


図 10　$\hat{D}_t - D_t$ の自己相関係数

単位根過程

実際のデータの検定する

TOYAMA Prefectural University

ブルウィップ効果の実際的な要因

＜合理的な意思決定による場合＞

- ●需要サイド
  - 需要プロセス（AR,MA,ARMA,非定常過程） と予測手法
  - 季節性
    - 需要パターン
    - 発注間隔
- ●供給サイド
  - 潜在的な供給不足(supply shortage)
  - 有限能力
    - 生産能力
      - 能力上限制約
        - 平準化
        - 生産計画(作成サイクル、生産制約)
    - オーダー順でない出荷順序
    - 倉庫能力
  - バッチ単位の発注(batch ordering)
- ●費用項目 在庫補充法
  - 固定費（積載率,段取替え）
  - 販促用のディスカウント(購買集中・前倒し施策）
  - 変動に対するペナルティ
  - 週1回発注、週2回発注、毎日発注方式
  - 調達リードタイム
- ●データ集約 (aggregation)
  - 時間(日、週、月、4半期、半期、年）
  - 製品集約（品種、グループ）
  - 場所集約
  - 連結構造
- ●内示特性
  - 週次内示/月次内示,トレンドとばらつき
- ●サプライヤー間の情報共有

＜非合理的な人間判断による場合＞

- ●合理性追求欠如
- ●恣意的な合理的結果の変更　需要ピークを外した発注
- ●需要の変化に対するオーバーアクション、品切れに対する過剰な挽回策

図 11　ブルウィップ効果の実際的な要因

数理モデルとシミュレータの違い

本研究で考慮する要因

1. 内示のトレンドの変動 **(水準数:27)** ←理論でも考慮
2. 内示のばらつきの変動 **(水準数:5)** ←理論でも考慮
3. 発注量のバッチサイズ **(水準数:5+10)** ←理論でも考慮
4. 能力上限制約 **(水準数:6)** ←理論でも考慮
5. 発注から納入までのリードタイム **(水準数:7)**
6. 需要のばらつき **(水準数:5)**
7. 安全な在庫目標の決め方 **(水準数:5)**

1. 2 要因 (i,j) の組み合わせが **6C2=15**
2. 2 要因以外の特定の組み合わせに対して **1000** 回
3. 期間は **400** 日間 $y_{ij1} \sim y_{ij1000}$

表 1　要因による週次内示のブルウィップ効果

ケース		考慮する項目			内示生産システムのブルウィップ効果の式（週次内示の場合）	ブルウィップ効果への影響
		需要プロセス	設備能力制約	バッチオーダー		
①	・$y_{t+1}-y_t=0$ ・設備能力制約なし ・バッチオーダーなし	○	×	×	1	ブルウィップ効果なし
②	・$y_{t+1}-y_t\neq 0$ ・設備能力制約なし ・バッチオーダーなし	○	×	×	$\frac{M+\mathrm{Var}[y_{t+1}-y_t]}{M}$	在庫目標の変動により増幅
③	・$y_{t+1}-y_t\neq 0$ ・設備能力制約あり ・バッチオーダーなし	○	○	×	$\frac{M+\mathrm{Var}[y_{t+1}-y_t]}{M}\cdot\frac{\mathrm{Var}[\hat{Q}_t]}{\mathrm{Var}[Q_t]}$	在庫目標の変動により増幅、設備能力制約により減幅の複合化
④	・$y_{t+1}-y_t\neq 0$ ・設備能力制約なし ・バッチオーダーあり	○	×	○	$\frac{M+\mathrm{Var}[y_{t+1}-y_t]}{M}\cdot\left(1+\frac{E[X_t(OS-X_t)]}{Var[\hat{Q}_t]}\right)$	在庫目標の変動により増幅 バッチサイズにより増幅の複合化
⑤	・$y_{t+1}-y_t\neq 0$ ・設備能力制約あり ・バッチオーダーあり	○	○	○	$\frac{M+\mathrm{Var}[y_{t+1}-y_t]}{M}\cdot\frac{\mathrm{Var}[\hat{Q}_t]}{\mathrm{Var}[Q_t]}\cdot\left(1+\frac{E[X_t(OS-X_t)]}{Var[\hat{Q}_t]}\right)$	在庫目標の変動により増幅 設備能力制約により減幅 バッチサイズにより増幅の複合化

y_t: 在庫目標，Q_t: 発注量
$\hat{Q}_t$: 設備能力制約ありの時の発注量，OS: バッチサイズ
X_t: Q_t を OS で割った余り

ブルウィップ効果への交互作用
(在庫目標の変動・バッチサイズ)

④	・$y_{t+1}-y_t \neq 0$ ・設備能力制約なし ・バッチオーダーあり	○	×	○	$\frac{M+\mathrm{Var}[y_{t+1}-y_t]}{M}\cdot\left(1+\frac{E[X_t(OS-X_t)]}{Var[\hat{Q}_t]}\right)$	在庫目標の変動により増幅 バッチサイズにより増幅の複合化

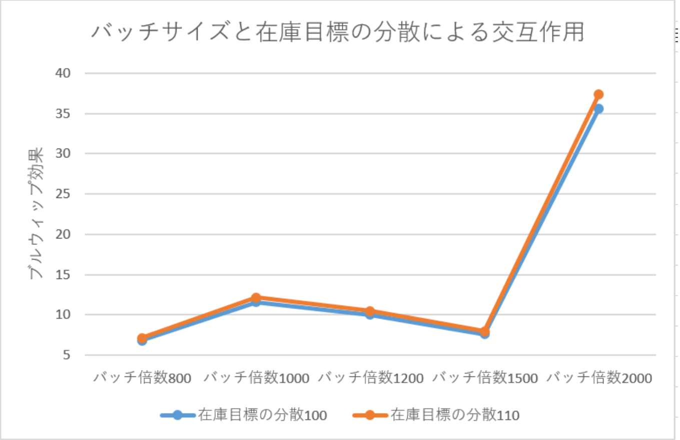


図 12　数理モデルによる交互作用

3.2. シミュレータによる交互作用の把握

主効果と交互作用

主効果とは、それぞれの独立変数がそれぞれ「独自」に従属変数へ与える単純効果のこと。交互作用とは、独立変数を組み合わせた場合の複合効果のこと。

表 2　二元配置の実験表

	B_1	B_2	$\cdots$	B_b
A_1	y_{111}	y_{121}	$\cdots$	y_{1b1}
	$\vdots$	$\vdots$	$\cdots$	$\vdots$
	y_{11k}	y_{12k}	$\cdots$	y_{1bk}
A_2	y_{211}	y_{221}	$\cdots$	y_{2b1}
	$\vdots$	$\vdots$	$\cdots$	$\vdots$
	y_{21k}	y_{22k}	$\cdots$	y_{2bk}

A_1, A_2, B_1, B_2, B_3:要因 A, B の水準
$y_{111}, \cdots, y_{2bk}$:結果のデータ (自分の研究ではブルウィップ効果)

実験計画法の活用 (1)

シミュレータにより複数の要因から交互作用を導出するために実験計画法を活用する。

多元配置法　因子が $r \geq 3$ の場合は，(6), (7) を必要なだけ繰り返す．

(1)　二つの因子の組合わせ全てについて，二元配置実験表を作成する．

(2)　修正項 CT，総二乗和 S_{TE} を求める．自由度 f_{TE} は総データ数 -1 である．

$$CT = \frac{(\text{総計})^2}{\text{総データ数}}, \qquad S_{TE} = \sum(\text{個々のデータ値})^2 - CT$$

(3)　1 因子の主効果 (例えば A) の二乗和 S_A を求める．自由度 f_A は 水準数 -1 である．

$$S_A = \sum_{i=1}^{a} \frac{(A_i\text{で測定されたデータの合計})^2}{A_i\text{で測定されたデータ数}} - CT$$

(4)　2 因子の組合わせ効果 (例えば AB) の二乗和 S_{AB} を求める．

$$S_{AB} = \sum_{i=1}^{a}\sum_{j=1}^{b} \frac{\{(A_i, B_j)\text{ で測定されたデータの合計}\}^2}{(A_i, B_j)\text{ で測定されたデータ数}} - CT$$

実験計画法の活用 (2)

(5) 2因子の交互作用 (例えば $A \times B$) の二乗和 $S_{A \times B}$ を求める．自由度 $f_{A \times B}$ は $f_A \times f_B$ である．

$$S_{A \times B} = S_{AB} - S_A - S_B$$

(6) 3因子の組合わせ効果 (例えば ABC) の二乗和 S_{ABC} を求める．

$$S_{ABC} = \sum_{i=1}^{a} \sum_{j=1}^{b} \sum_{k=1}^{c} \frac{\{(A_i, B_j, C_k)\text{ で測定されたデータの合計 }\}^2}{(A_i, B_j, C_k)\text{ で測定されたデータ数}} - CT$$

(7) 3因子の交互作用 (例えば $A \times B \times C$) の二乗和 $S_{A \times B \times C}$ を求める．自由度 $f_{A \times B \times C}$ は $f_A \times f_B \times f_C$ である．

$$S_{A \times B \times C} = S_{ABC} - S_A - S_B - S_C - S_{A \times B} - S_{B \times C} - S_{C \times A}$$

(8) 残差二乗和 S_E を S_{TE} から各要因の変動を全て取り去ることで求める．自由度 f_E は f_{TE} から各要因の自由度を全て取り去ることで求める．

MPI

MPI とは，並列コンピューティング利用するための標準化された規格である．MPI はプログラミング言語とは独立の通信プロトコルで，並列計算機上で動くプログラムに使用される．

実験環境

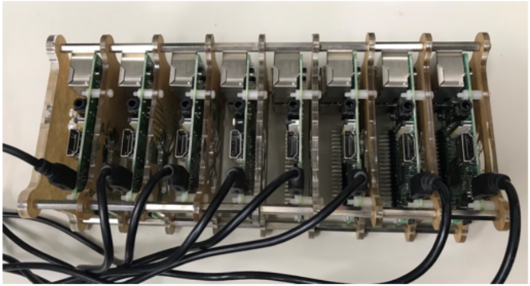


図 13　本研究で用いる並列計算機

並列分散処理

- 3つの概念から構成
  - 分散ファイルシステム(Hadoop Distributed File System; HDFS)
    - 複数のマシン上でデータを共有し冗長化するためのファイルシステム。ネットワーク上で構成されたRAIDのようなもの。FUSEを使ってmountすると一般のファイルシステムとほぼ同等に扱える。
  - 分散処理(Map/Reduce)
    - 複数のマシン上で処理を分散し、分散処理された結果を最後に集めて集計する。
  - 分散データベース(hBase/HyperTable)
    - Google BigTableのようにカラム型データベースを実現する。

- 上記のどれかにだけ注目して利用することも可能
  - RAIDの代わりにHDFSのみ利用してデータの冗長化をしたい
  - 分散処理だけ利用したい
    (※ Hadoopで分散処理するにはHDFS上にファイルを置いた方が分散性があがる)
  - HDFSを使わずカラム型データベースだけ利用したい

結局Hadoopで何ができるの？という問いには…

データの冗長化ができる

Google BigTableのようなものを自社環境で利用できる(改造もOK)

分散処理を少ない手間で記述・実行できる
(分散性の高い処理に限る)

汎用的なPC数台で構築できる

※ 構築・運用コストは安くないので自前で構築する場合は要注意

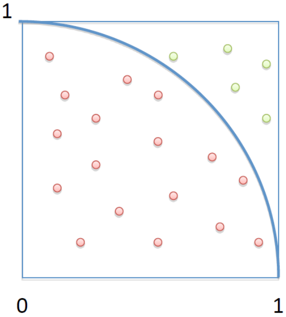
並列分散処理

- 特徴
  - Write Once/Read Manyのアクセスモデルに最適化
  - アクセスパーミッション (POSIXライク)
    - ownerとgroupにread/write/execを付与
    - sticky bit (/tmpのように誰でも書きこめて所有者や管理者のみ削除可能な機能)はなし
    - setuid,setgid (実行時に所有者の権限で実行する機能)はなし
  - クォータを設定可能
    - Name Quotas ... ファイルとディレクトリの最大数
    - Space Quotas ... 1ファイルで扱える最大サイズ
- 利点
  - 超巨大ファイルに対応
    - 数GB～数PBのファイルでも格納可能
  - 安価なハードウェアで高可用性を実現
    - 複数のハードウェアで運用することを前提に作られているため1台くらい壊れても問題なし
    - ファイルのコピーは複数のシステムに分散されている
- 欠点
  - Read/Writeが遅い
    - 特にWriteはかなり遅い
  - 大量の小さいファイルは苦手
    - NameNodeのヒープメモリが枯渇する
  - ディスクの使用効率が悪い
    - RAIDのようにパリティのみ格納というモデルではなくそのまま別ハードウェアに保存
  - NameNodeとして利用するサーバは高可用性が必要

MapReduceの特徴

- データ通信
  - 各Map処理、Reduce処理は完全に並列に実行可能
  - マシンを増やせばその分処理能力が増える
- 耐故障性
  - 失敗したMap, Reduce処理は他のノードで再実行される
  - 遅いMap, Reduce処理についても同じ
- ローカリティ
  - データのある場所で計算を始めれば、ネットワークを使う必要がなくなる
  - Moving Computation is Cheaper Than Moving Data

並列分散処理



モンテカルロ計算による円周率の求め方
x: 0から1までの乱数
y: 0から1までの乱数
として、点(x,y)をランダムに生成する。

・(x,y)が円弧の中に含まれた場合の数をinside
・(x,y)が円弧の外に含まれた場合の数をoutside
とすると扇形の面積はπ/4、正方形の面積は1なので、
扇形の中の点数/全体の点数=π/4
が成立する。

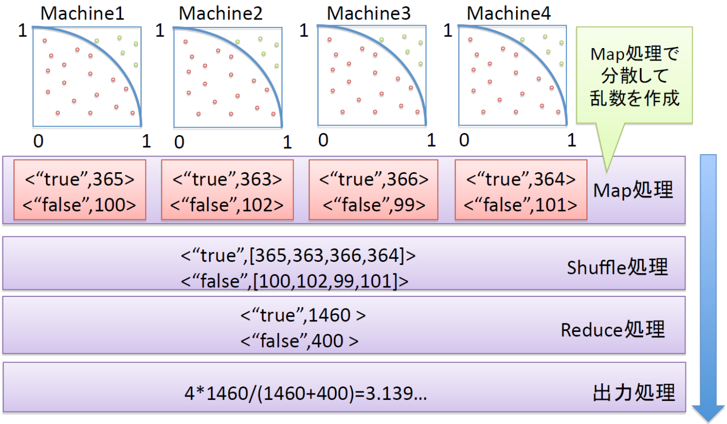
すなわちinside/(inside+outside)=π/4となるので、πを求めるには4*inside/(inside+outside)を計算すればよい。

Hadoopを使って円周率を求めるには、乱数を発生させMap処理によって

<“true”, inside>　<“false”, outside>

という<キー,値>のペアを作成する。このマップは分散された個数だけ生成されるので、
Reduce処理によって集計する必要がある。
”true”, ”false”というキーに対して集計すると、処理全体でのinside, outsideが求まる。
最後に終了処理として4*inside/(inside+outside)を出力する。

並列分散処理



Machine1
Machine2
Machine3
Machine4
1
0
1
Map処理で分散して乱数を作成
<“true”,365>
<“false”,100>
<“true”,363>
<“false”,102>
<“true”,366>
<“false”,99>
<“true”,364>
<“false”,101>
Map処理
<“true”,[365,363,366,364]>
<“false”,[100,102,99,101]>
Shuffle処理
<“true”,1460 >
<“false”,400 >
Reduce処理
4*1460/(1460+400)=3.139…
出力処理

実験条件

実験

1 企業の取引を **1000** 日間シミュレーションしたものを **1** 回として、それぞれの実験ケースについて **100** 回実験を行う。

4.1. 提案手法による主効果の把握

	需要のばらつき10	需要のばらつき50	需要のばらつき100	需要のばらつき200	需要のばらつき300
バッチ最小1500	21.062299	11.356332	7.4979775	8.636711	12.5857345
バッチ最小2000	24.33940125	11.9295705	7.145912	7.5258665	10.1071345
バッチ最小2500	27.28434825	12.933844	8.5568315	8.2711685	10.198321
バッチ最小3000	29.875385	13.116196	11.172752	16.064309	4.597033
バッチ最小3500	33.09853	14.278241	16.130483	6.609861	5.111856
バッチ最小4000	35.781746	15.6365635	10.585587	7.149569	5.608439
バッチ倍数800	16.396506	10.058824	4.844139	5.737173	6.314352
バッチ倍数1000	16.321897	15.4303105	11.625644	16.22241	5.513199
バッチ倍数1200	16.660924	7.167245	9.315898	9.315505	11.278782
バッチ倍数1500	16.370468	9.154422	4.878246	7.38767	6.733644
バッチ倍数2000	16.269289	12.496615	16.233276	3.358085	7.051571

図 14　バッチサイズと需要のばらつきによるブルウィップ効果

提案手法による主効果の把握

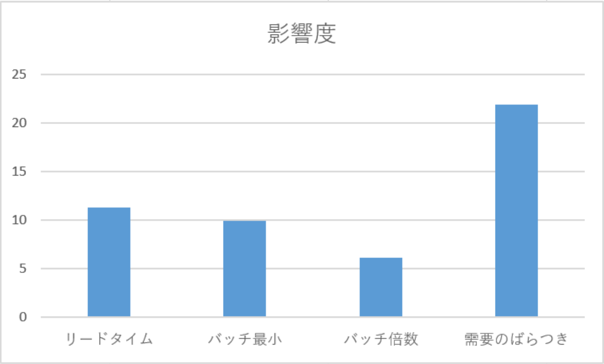


図 **15**　ブルウィップ効果に対する影響度

影響度は主効果の絶対値の和である。

4.2. 提案手法による交互作用の把握

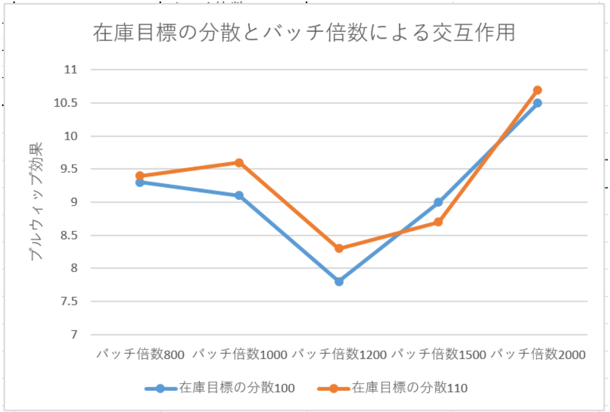


図 16　シミュレータによる交互作用

TOYAMA Prefectural University

提案手法による交互作用の把握

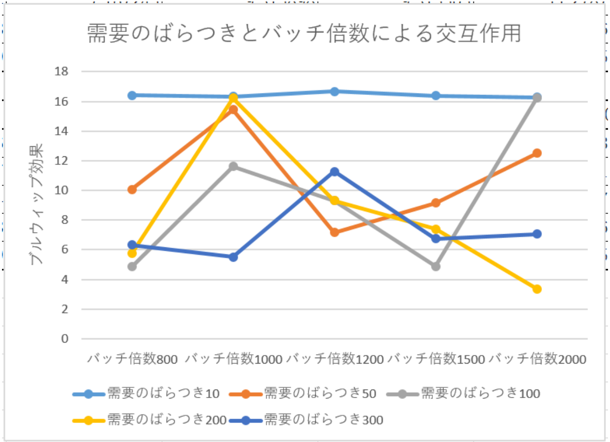


図 17　シミュレータによる数理モデルで考慮されていない要因の交互作用

4.3. 並列分散処理の効果

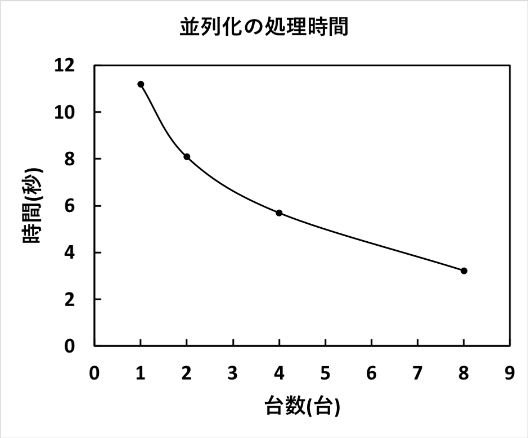


図 18　並列分散処理の結果

5. まとめと今後の課題

まとめ

1. シミュレータを作成し各要因ごとの影響度を導出した。
2. 数理モデルで考慮されていない要因を含む交互作用を導出した。
3. 並列分散処理により処理時間を短縮した。

今後の課題

1. 直交表を用いてさらに実験回数を減らす。
2. さらに要因数を増やしブルウィップ効果の低減方法を示す。