SCM における小売業の機会損失に関する研究

生　島　　義　英

SCM における小売業の機会損失に関する研究

目　次

Study on Opportunity Loss of the Retail Trade in SCM

Yoshihide　Ikushima

This doctoral dissertation examined the Opportunity Loss of Retail Trade in supply chain management. The study dealt with the department store industry, the backyard of a retail store and an automated warehouse. This paper extracted opportunity losses for each one. This paper calculated the economic effect that would be provided by planning and thus ameliorating these problems.

This doctoral dissertation consists of seven chapters.

Chapter 1 provides the introduction and outlines the dissertation, its motive, its purpose and the result of research generally clarifies the whole dissertation.

Chapter 2 is a study on the opportunity loss in SCM. It defines opportunity loss and SCM. In this chapter, this paper considered opportunity losses by extracting the opportunity loss at SCM from merchandising and from within the manufacturing industry.

Chapter 3 is a case study on opportunity losses of SCMs in the department store industry. In this chapter, this paper studied the supply chains of the department store industry by focusing on the "efficiency of the merchandise procurement use" as a means of restructuring a department store. This paper clarifies what kind of impacts the opportunity losses in SCMs bring to the department store industry.

Chapter 4 is a "case study on the opportunity losses of backyards of large retailers." This chapter focuses on the backyard of a large retail store and calculates the economic effect obtained by decreasing the backyard area.

Chapter 5 is a "case study on opportunity loss in automated warehouses."

In this chapter, this paper focus on automated warehouses uses as retail and e-commerce logistics centers. This paper clarifies the opportunity losses required from the initial installation cost and the average occupancy rate in buildings designed to be automated warehouses.

Chapter 6 is titled "Consideration of opportunity loss with SCMs in the retail industry."

Chapter 7 is the "conclusion." This chapter refers to the results of the research and suggests future-related tasks. In conclusion, information sharing in the manufacturing, distribution and sales stages of the retail industry has hardly been planned. The decrease in operational efficiency caused by this fact leads to opportunity losses.

第 1 章　序論

1. はじめに

本研究では,「小売業視点からの SCM における機会損失」をテーマに掲げ, 研究を進めることとする。

サプライチェーンとは, 消費者に商品を供給するために様々な業務・企業のつながりを鎖に例えたものである。これらの業務は 1 社ではなく多数の独立した企業によって分担されている。すなわち, 最終消費者が商品を手にするためには, 様々な企業が参画・連鎖し, 商品の製造段階から小売段階を経て, 最終消費者に渡ることになる。

サプライチェーンマネジメント（以下 SCM とする）とは, 個々の企業の業務を接続させることであり, 製造段階から販売段階, 最終消費者まで, それぞれの業務を繋ぐことでサプライチェーン全体を最適にすることである。すなわち自社のサプライチェーン前後の取引先・顧客の業務を鑑み, サプライチェーン全体で最適化することが望まれる姿であると考える。

しかしながら, サプライチェーンを構成する各企業の日々の活動は, 売上を上げる, 利益を伸ばすために様々な努力をし, 基本的には企業内の最適化を図ることで利益を出すことに注力を注いでいる。すなわち, ある企業で業務改革になる活動を自社のみで捉え, それ以外のことは取引先に押し付けている。これらの活動は SCM とは別の方向を目指すものである。

SCM による合理化に関する研究は進んでいるが機会損失を対象とする研究論文は皆無に近い。一方, SCM には多数の機会損失対象分野が存在する。SCM における機会損失の研究はコスト削減ないしは利益拡大に関する研究と同様に重要である。機会損失を最小にする事は, 被顕在的効果を顕在化することである。サプライチェーン内で発生している機会損失を顕在化させ, サプライチェーンを構成する各企業がその機会損失を認識し, 機会損失を排除することにより, 全体最適化目指す是正を行い, 全体利益の向上を図ることが重要であると考える。よって, SCM における機会損失の研究は速やかに行う必要があると共に新たな研究対象として重要な意味を持っていると考える。

2. 研究目的

本研究の目的は, 小売業における SCM の機会損失を明らかにし, その対策を研究する事を目的とし, 本論文は次の 4 つの研究目標を設定し, 具体的には小売業に関する機会損失, その解明を試みたものである。

(1)SCM における機会損失領域を抽出する。

(2)百貨店業における取引先との情報共有化向上による機会損失の経済効果を測定と改善策を提言する。

(3)大型小売店における店舗バックヤード改善による機会損失の経済効果の測定と解決案を提言する。
(4)自動倉庫の最適設計モデルと入出庫分布特性に基づく機会損失の検証と解決案を提言する。

3. 論文の構成

本論文は全 7 章から構成されている。

第 1 章は，「序論」である。

本研究全体の構成，研究動機，研究目的並びに研究成果に就いて概要を述べ，本研究の全体を明らかにしたものである。合わせて，小売業の現状，百貨店，GMS が低迷している現状の把握と専門店の台頭，e コマースの急成長，コンビニエンスストアの成長など流通業の現状について記す。

第 2 章は，「SCM における機会損失に関する研究」である。

SCM の定義，ならびに機会損失の定義を明らかにするともに，SCM における機会損失を流通業，製造業の領域で抽出するともに機会損失の分析を行う。

本研究では，機会損失の定義を「方策の選択により得ることができなかった利益」として定義し，議論を進める。

また，SCM の定義は，「個々の企業の業務を接続させることであり，製造段階から販売段階，最終消費者まで，それぞれの業務を繋ぐことでサプライチェーン全体を最適にすることである。」として定義し，議論を進める。

第 2 章では，SCM おける機会損失を流通業，製造業の領域で抽出し，機会損失と対応策を取りまとめた。その結果，小売業における 3 つの機会損失の課題を抽出した。その 3 つの課題は，第 3 章で研究する「百貨店業界における SCM の取組みと機会損失」，第 4 章で研究する「大型小売店におけるバックヤードの機会損失」，第 5 章で研究する「自動倉庫における機会損失」である。

第 3 章は，「百貨店業界における SCM の取組みと機会損失に関する研究」である。

百貨店業界に関与する取引先を含めたサプライチェーンを研究対象とし，百貨店の構造改革として，「運用業務の効率化」に焦点を絞り，SCM がどのような効果を百貨店業界にもたらすのかを明らかにすることを試みる。

まずは，百貨店が取引先に発注した商品を百貨店が仕入計上する取引形態をビジネスフローに落とし込み，作業内容を分析する。このビジネスフローをベースに基に，取引先との情報共有化を基礎とする「買取型 EDI ビジネスフロー」を構築する。

このビジネスフローにおいて，４つの運用効率化が検討できる。その項目は，次の通りである。

①商品単品マスターの取引先からの配信
②納品提案，発注から納品・仕入計上までのシステム化
③売上情報の共有化
④買掛金支払案内の電子化
　これら，4つの効率化から具体的に運用業務の効率化施策として以下の5つの施策が検討される。
①商品マスター登録自動化
②注文伝票・仕入伝票削減化
③検品削減化
④値札削減化
⑤支払案内電子化
である。
　①商品マスター登録自動化は，取引先との情報共有化を推進するためのEDIの採用には，百貨店と取引先で共通に利用する商品マスターの共有化が前提条件となる。しかし，従来百貨店は単品管理を一部の商品のみしか実施しておらず，大半の商品は自社商品コード体系に基づく，品番管理である。単品管理を行うためには，共通単品商品コードのマスターを百貨店，取引先で登録する必要があり，それが百貨店，取引先と同期する必要がある。大半の百貨店は前述のとおり単品管理を実施していないため，共通単品商品コードを商品マスターとして管理するコストは発生していない。よって，検討の結果，商品マスター登録自動化によってもたらされる現状と比較して費用削減効果はないということがいえる。
　②注文伝票・仕入伝票削減化による効果は，仕入伝票を商品勘定システムに入力するパンチ入力費用と注文伝票を作成する費用がEDI化することにより，取引先と百貨店相互に費用が削減されることである。
　③検品削減化による効果は，百貨店に商品を納品する際に実施される検品作業，すなわち仕入商品の値札を1点ずつ仕入伝票と取引先と照合する作業を無くすことにより，取引先と百貨店相互に費用が削減されることである。
　④値札削減化による効果は，取引先で付与する商品単品コード，すなわちソースマーキング・JANコードで代替し，百貨店独自の値札を廃することにより，取引先と百貨店相互に費用が削減されることである。
　⑤支払案内電子化による効果は，百貨店が，従来月締め後に各取引先に当月の支払案内通知書を印刷し，郵送により発送している。取引先は，送られてきた支払案内通知書と取引先の売掛金とを照合している。「EDI」を採用することにより，支払案内通知書が標準フォーマットにより送信することにより，相互の照合作業がＩＴ化され，取引先と百貨店相互に業務効率化が図れるともに，

百貨店は，支払案内通知の印刷・封入作業削減化・郵送経費・用度品経費の削減が期待されることである。

これらの効果を算出する。百貨店業界は，「EDI」導入しないことにより，この経済効果を失っている。また，取引先も経済効果を失っている。ここに機会損失が発生していることを明らかにする。

第 4 章は，「大型小売店におけるバックヤードの機会損失に関する研究」である

百貨店・ＧＭＳ・ＳＭをはじめとする大型小売店は，店舗毎にバックヤードを有している。バックヤードには，商品・用度・什器備品ストック，厨房，事務所，休憩室，通路，階段（避難階段含む），搬入口，ごみ処理室，及び機械室等から構成されている。バックヤードを売場に転用すれば，新たに大きな投資することなく，既存の施設を利用することで収益を上げる事が可能である。すなわち，バックヤードは店頭業務をサポートする機能利用されており，収益を得られていない。すなわち，バックヤードはコストセンターであり，ここに機会損失が発生しているものと考える。

百貨店・GMS・スーパーマーケットなどの大型小売店の「店舗バックヤードの機会損失」に焦点を絞り，取引先との情報交換による最適な在庫計画に基づく売場の適正在庫予算を策定し，予算遵守することにより在庫過多が発生しない仕組みを構築し，バックヤードの商品在庫を排除し，バックヤードを縮小化することにより得られる経済効果，すなわちバックヤードを売場転用しないことによって生じる機会損失を明らかにすることを試みた。

本研究では，まずはバックヤードを解析するために必要な基礎数値を調査する。具体的には，業態別大型小売店へのフィールドスタディを実施する。次に，フィールドスタディで得られない数値データは，有価証券報告書，業界団体調査数値，官庁発表数値など公表されているデータを調査，抽出･分析し，解析に必要な基礎数値を把握する。これらの数値データを取りまとめ，解析に必要な基礎数値データとして確立する。次に，基礎数値データを基に，解析を実施し，解析結果から期待される経済効果を算出する。

それにより,バックヤードを活用しないことによって，機会損失が発生していることを明らかにする。

第 5 章は，「自動倉庫における機会損失に関する研究」である。

主に卸売業や小売業，e コマースの物流センターで活用している自動倉庫に焦点を絞り，設定した条件及び要求性能を満たす中でコスト最少の立体自動倉庫を設計する汎用型の最適設計モデルシミュレーションを構築する。

立体自動倉庫のラック及び入出庫データを前提として，クレーン台数，ラック段数，ラック連数を様々なパターンで計算してラックコスト，レールコスト，クレーンコスト，ビルディングコスト，およびスプリンクラーコストからなる，

初期導入に要する総費用を算出し，当該総費用が最少となる最適設計モデルをより現実的な制約条件下で構築する。

その結果，入出荷処理のアルゴリズム・モデルをどのように設定するかにより，クレーンの稼働率が大きく変化し，要求性能を維持しつつ導入費用を低減可能であることを論じる。そして，構築するシミュレーションを基に，入出荷命令の分布と初期導入コスト，稼働率，機会損失額と入出荷命令の分布特性との関係性について明確化することで，投資効率のモデル検証を行う。

第 6 章は，「小売業における SCM の機会損失の考察」である。

第 2 章で定義した機会損失を踏まえ，第 3 章「百貨店における SCM の取り組みと機会損失に関する研究」，第 4 章「小売業におけるバックヤードの機会損失に関する研究」，第 5 章「自動倉庫における機会損失に関する研究」の結果を踏まえた結果，小売業において機会損失が存在することが明らかになった。

本章では，SCM における小売業の機会損失について考察し，企業経営において機会損失の位置づけを明らかにする。また，機会損失を企業発展原動力の側面から解析し，これからの企業経営の方向性を示す。

第 7 章は，結論である。

本研究全体の研究成果並びに今後の課題において言及したものである。

4.小売業の現状

4.1 小売業の体系

小売業とは，日本商工会議所の定義では，「小売業とは，メーカーや卸売業から商品を仕入れ，流通機構の最終段階に位置する消費者に直接，商品などを販売する事業者のことである。(商工会議所)」である。

小売業の分類については，過去日本においては生産体系からくる「業種」と言う概念が主流であった。八百屋，魚屋，肉屋等はそれぞれ農業・漁業・畜産業と言う生産体系からの分類であり，電気店，薬局/薬店，化粧品店等はメーカー等の生産体系で分類されてきた。そしてそれら業種は８０年代まで日本政府の保護・規制の政策，許認可，メーカーの系列化等の中で，小売業の大半を占めてきた。その為日本の小売業界は「家業」，「零細過多」，「参入障壁」等の言葉に代表されるような特色を持ったドメステイックなものとして，成長してきた。

「業種」に対し「業態」は消費者の視点から，より多くの商品を，より便利に，より安くという小売業の競争原理の中から次々と生みだされてきた体系である。

規制の無い自由競争こそが消費者の利益につながると考える米国では，小売業界への自由な参入と撤退を促し，次々と新しい革新的な業態の登場を促している。

日本においては９０年以降，日米構造協議で政府の政策も「中小小売業の保護」から「消費者利益の保護」に軸足を移さざるを得なくなり，国際化・グローバル化の波の中で業態開発が活発となり，それに伴って「業種」と言われる店舗は急速に淘汰され始めたのである。

小売業の業態の分類は非常に難しくなってきている。即ち消費者の購買行動・ニーズの変化，マーケットの変化，競争の変化，法規制の変化等で絶えず新しい業態が出現し，既存の業態も変容したり，消滅したりを繰り返し，分類は難しい。しかし，本研究では，以下のとおり分類し，議論を進めることとする。

「業態」による小売業の分類は，まずは大きく店舗販売と無店舗販売に分け，店舗販売は店舗という目に見える形態から，扱い商品，規模，販売形態等で分類される。一方，無店舗販売は通信販売，訪問販売に分類され，通信販売は使用するメディアによってさらに分類される。この考え方に基づき，小売業を代表的な業態別にまとめると下記「表 1.1」のとおりである。

最近では，アマゾン・楽天などに代表とされるインターネットを利用したｅコマースが急成長している。それを受け，店舗を有する大型小売業は，例えば，セブンアイグループやＪＦＲ(大丸松坂屋パルコ)などは，コマースと店舗販売を合体させた「オムニチャネル」といった，融合形態の業態を積極的に取り組んでいる。

4.2　小売業の現状

4.2.1 小売業全体の推移

経済産業省「商業販売統計」によると，小売業（自動車販売や家電販売を含む）の 2015 年年間販売額は約 140 兆 7 千万円である。

販売額の推移を「図 1.1」に示す。小売業は 1997 年の 148 兆円をピークに減少に転じ, 2002 年に 132 兆円でピーク時の 89%となり底を打った。その後徐々に回復したが, 2008 年リーマンショックの影響で落ち込み, 2010 年から再び復調したが, 2011 年大震災の影響で再び低迷した。2014 年は消費税引上げ駆け込み需要により増加したが, 2015 年は消費税の反動により低迷傾向にある。

主な小売業の業態別販売額推移は，「表 1.2」「図 1.2」に示す通りである。

百貨店業界は，バブル経済が崩壊した後，衰退傾向を示し，日本百貨店協会が発表した 2016 年の全国百貨店売上高は, 2015 年比 2.9%減（既存店ベース）の 5 兆 9780 億円であった。年間売上高のピークは 1991 年の約 9 兆 7130 億円であり，ピーク時の 6 割までに低下している。

スーパー業界は，総合スーパー（GMS）は，下落傾向にあるが，食品スーパーがそれを補完し，横這いに推移している。

一方，コンビニエンスストア業界は，順調に販売額を伸ばしており, 2009 年には百貨店の販売額を超え，スーパー業界の販売額に近づきつつあるのが現状である。

近年の傾向をまとめると，ワンストップショッピングが可能な大規模な商業施設である百貨店やGMSが下落傾向にあり，コンビニエンスストア・食品スーパーなどが上昇傾向である。アマゾンなどに代表されるeコマースは大幅に上昇している。

表 1.1　小売業業態一覧表

分類 1	分類 2	分類 3
有店舗販売	百貨店（デパート）	
	スーパーマーケット	総合スーパー(GMS)
		食品スーパー(SM)
	コンビニエンスストア	
	ディスカウントストア	
	チェーン専門店	家電量販店
		衣料量販店
		ドラックストア
		その他量販店
	売店（キヨスクなど）	
	専門店(従来型)	食料品店（八百屋，酒屋など）
		衣料品店
		電器店
		書店
		薬品・薬局
		その他
無店舗販売	通信販売	ｅコマース(アマゾンなど)
		テレビショッピング
		カタログ通販
	自動販売機	
	訪問販売	

（単位：10 億円）

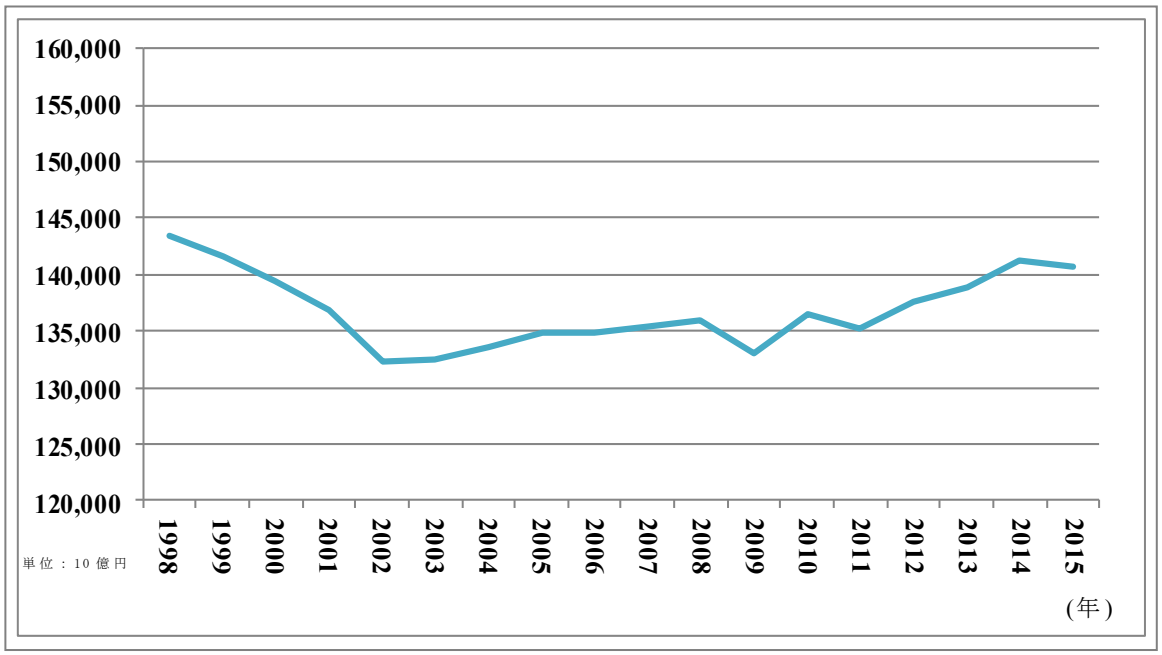

図 1.1　小売業計販売額推移[3]

(単位：10 億円)

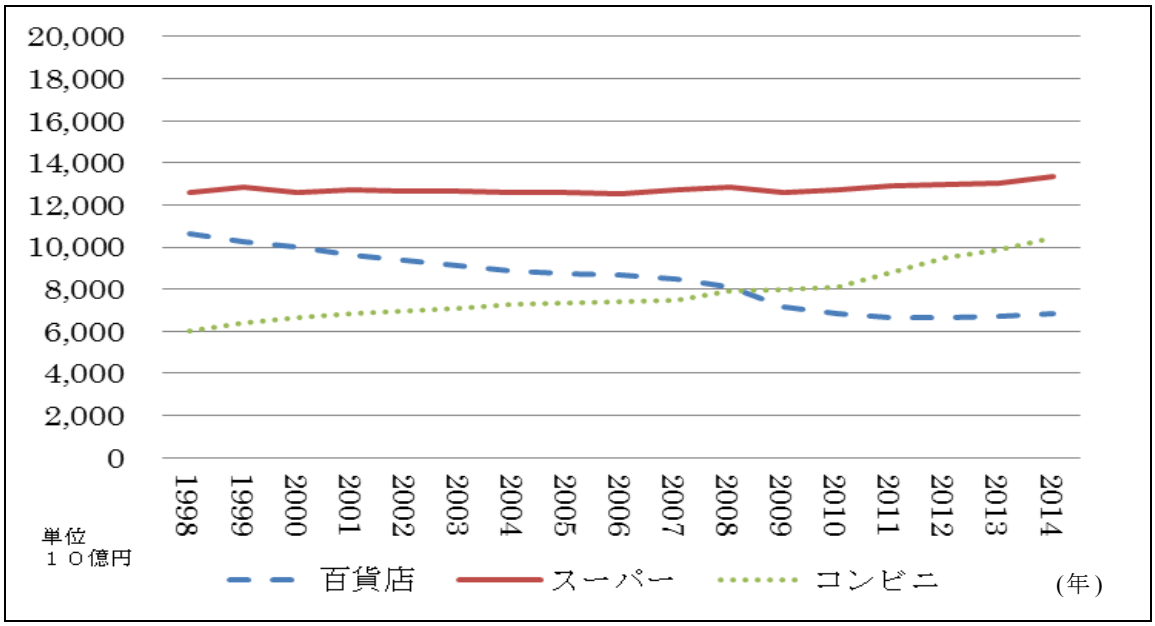

図 1.2　主な小売業の販売推移[3]

表 1.2　業態別小売業販売額推移表[3]　　（単位:10 億円）

西暦	百貨店	スーパー	コンビニエンスストア	小売合計
1998	10,657	12,591	6,049	143,494
1999	10,285	12,839	6,383	141,528
2000	10,011	12,622	6,680	139,435
2001	9,626	12,715	6,846	136,808
2002	9,365	12,668	6,980	132,280
2003	9,107	12,653	7,096	132,446
2004	8,854	12,614	7,289	133,649
2005	8,763	12,565	7,360	134,828
2006	8,644	12,501	7,399	134,911
2007	8,465	12,734	7,490	135,417
2008	8,079	12,872	7,943	136,019
2009	7,177	12,599	7,981	132,961
2010	6,842	12,737	8,114	136,479
2011	6,661	12,933	8,775	135,157
2012	6,639	12,953	9,477	137,585
2013	6,720	13,058	9,872	138,897
2014	6,827	13,370	10,423	141,219
2015	6,826	13,223	10,996	140,666

4.2.2 業態別販売額の推移

経済産業省「我が国流通業の現状と 取組・課題について」によると，業態別では，百貨店・総合スーパー(GMS)の売上が減少傾向にある。

一方，コンビニエンスストア，ドラックストアやアマゾン・楽天などに代表される e コマース，インターネット取引・通信販売等が大きく成長し，増加している。 (「表 1.3」参照)

今後は，オムニチャンネル・バーチャルストアなどＩＴとインターネット技術を活用したフリマアプリの「メルカリ」などの新しい業態，創業・発展していくとともに，「アマゾン」や「楽天市場」などのｅコマースが更なる発展していくことが予想されている。

また，今後予想される人口減少に伴い，国内の消費財需要は確実に影響を受ける見込みであり，例えば，家計食料品消費は 2050 年には 2000 年比で約 3 割

の縮小が予測されている。(経済産業省　第１回 産業構造審議会 流通部会 審議用参考資料より)

これらのデータが示すとおり，百貨店・ＧＭＳに代表される総合型大型小売業を取り巻く環境はとても厳しい状況であると言える。この環境下でいかに利益を上げていくためには，業務効率を向上させ，機会損失を排除する方策を検討する必要があると考える。

表 1.3　主要業態別売上高推移[4]　（単位：兆円）

業態	2001 年	2011 年	傾向
百貨店	8.6	6.2	↓
総合ｽｰﾊﾟｰ	15.9	12.7	↓
ｺﾝﾋﾞﾆｴﾝｽｽﾄｱ	6.7	8.7	↑
ﾄﾞﾗｯｸﾞｽﾄｱ	2.7	5.6	↑
ﾎｰﾑｾﾝﾀｰ	3.8	2.7	↓
通信販売	2.5	4.7	↑
小売業全体	136.8	134.0	↓

5. おわりに

第１章では，研究目的並びに研究成果に就いて概要を述べ，本研究全体の構成，本研究の全体像を明らかにした。合わせて，小売業の現状について述べ，百貨店，GMS が低迷している現状の把握とチェーン型専門店の台頭，ｅコマースの急成長，コンビニエンスストアの成長など流通業の現状について論じることができた。

参考文献

[1] 生島義英, 佐藤哲也, 唐澤豊, 若林敬造, 小売業におけるバックヤードの機会損失に関する基本的研究, 日本ロジスティクスシステム学会誌, 2016 年 3 月
[2] 生島義英, 小売業におけるバックヤードの機会損失に関する基本的研究, 日本ロジスティクスシステム学会全国大会予稿集, 2015 年 7 月
[3] 経済産業省,商業動態統計年報, 1998 年～2015 年
[4] 経済産業省,我が国流通業の現状と 取組・課題について, 第 1 回 産業構造審議会 流通部会 審議用参考資料, 平成２４年４月
[5] 小学館,デジタル大辞泉, URL:http://www.daijisen.jp/digital/index.html, (2016 年 1 月確認)
[6] 日本百貨店協会, URL:http://www.depart.or.jp ホームページ, 百貨店売上高データ,(2017 年 10 月確認)

第２章　SCM における機会損失に関する研究

1. はじめに

第２章は，機会損失の定義ならびに SCM の定義を明らかにするとともに，SCM おける機会損失を流通業，製造業の領域で抽出するとともに機会損失の分析を行う。

2. 機会損失の定義

ここでは本論文における「機会損失」について定義する。機会損失の定義を文献で調査する。その調査結果をまとめると下記のとおりである。

①Merriam-WEBSTER の定義[9]
Definition of OPPORTUNITY COST
「the added cost of using resources (as for production or speculative investment) that is the difference between the actual value resulting from such use and that of an alternative (as another use of the same resources or an investment of equal risk but greater return)」

直訳すると以下のとおりである。

「機会損失とは，生産または投機的な投資のための資源を，実際の価値とその代替のものとの間の差異（同じ資源の別の使用または等しいリスクの投資として）」と記載されている。

②日本オペレーションズ・リサーチ学会辞典の定義[2]
③大辞泉の定義[3]
④統計・0R 活用事典の定義[4]
⑤マネー辞典の定義[11]

上記の文献で機会損失の定義を調査した。

機会損失定義調査の結論としては，本研究では，機会損失の定義を下記のとおりとする。

機会損失とは，「方策の選択により得ることができなかった利益」であるとする。以下，本研究ではこの機会損失の定義に従い，議論を進めることとする。

3. SCM と機会損失の展望

3.1 サプライチェーンとは

サプライチェーンとは，消費者に商品を供給するために様々な業務・企業のつながりを鎖に例えたものである。これらの業務は 1 社ではなく多数の独立した企業によって分担されている。

たとえばアパレル産業を例としてあげれば，繊維・糸など素材メーカーである製糸産業，織物・編み物などのテキスタイル産業，アパレル商品を企画・生産・卸売するアパレル産業，消費者に販売するアパレル小売業とそれらの産業を繋ぐ物流サービスを包含する一連の供給連鎖のことである。

下記「図 2.1」に示すとおりである。

このように，最終消費者が商品を手にするためには，様々な企業が参画・連鎖し，商品の製造段階から小売段階を経て，最終消費者に渡ることになる。

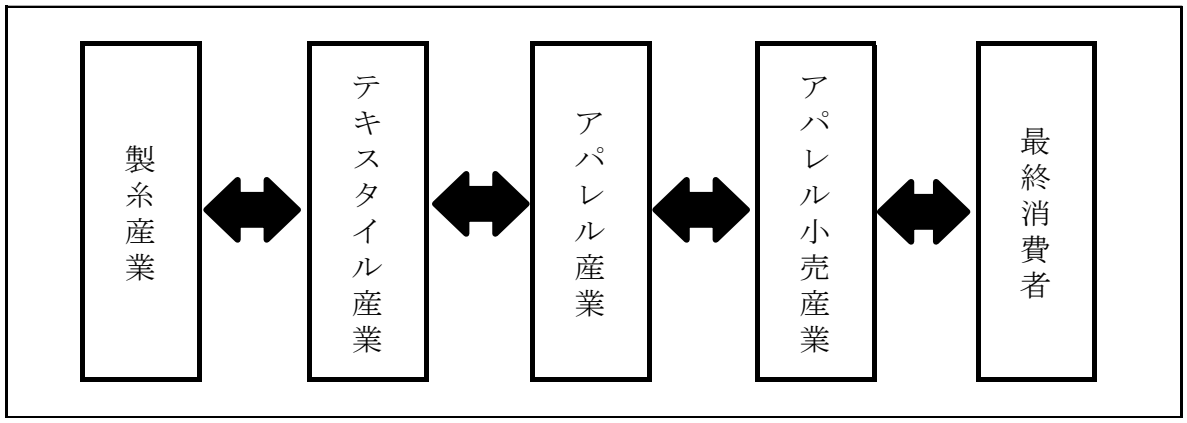

図 2.1　アパレル産業のサプライチェーンイメージ図（供給連鎖）

3.2 SCM の定義と戦略

ここでは本論文における「SCM」について定義する。

SCM とは，個々の企業の業務を接続させることであり，製造段階から販売段階，最終消費者まで，それぞれの業務を繋ぐことでサプライチェーン全体を最適にすることである。

サプライチェーンを構成する各企業の日々の活動は，売上を上げる，利益を伸ばすために様々な努力をしている。基本的には企業内の最適化を図ることで利益を出すことに注力を注いでいる。ある企業で業務改革になる活動を自社のみで捉え，それ以外のことは取引先に押し付けるというのでは SCM とは言えない。

SCM は自社のサプライチェーン前後の取引先・顧客の業務を鑑み，サプライチェーン全体で最適化することが望まれる姿であるといえる。

SCM の目指す方向性は，サプライチェーン全体の最適化を目指すものであり，企業間の垣根を越えて複数の企業がビジネスネットワークを組んで最適化を目指すことである。

これらの動きの中において，サプライチェーン内で発生している機会損失を顕在化させ，サプライチェーンを構成する各企業が機会損失を認識し，機会損失を排除することにより，全体最適化目指す是正を行い，全体利益の向上を図

ることが重要であると考える。

SCM 戦略の要素は，スピード・コストダウン・品質などを同時にまたは個別に満足することを前提に，チャネル・共同化・標準化・開発・人事教育・ネットワーク・システム・最適化などを戦略の核として実現することである。

主要パラメーターはコストミニマム・最少在庫・最少リードタイム・アウトソーシング・ジャストインタイム・最大能力・時間待ち最少・ミックスなどを挙げることができる。SCM のリードタイム最少や生産のリードタイム最小のようにリードタイム最少を目標関数として取り上げ，結果としてコスト最小を期待する方法がポイントである。このコスト最小を実現するために，機会損失の側面からアプローチし，主要パラメーターの最適化を戦略の核として位置づける。

3.3 製造業における機会損失

製造業（メーカー）は，メーカー・卸・小売のサプライチェーンにおいて上流に位置し，卸・小売業からの販売実績やサプライチェーン各段階での流通在庫状況をタイムリーにかつ正確に把握し，適切な生産計画に基づいて製品を生産すれば，「ムダ・ムラ・ムリ」を排除し，サプライチェーン全体の在庫を縮小することや納品の積載効率高めることが可能となる。

しかし，自動車業界など垂直統合が図られている業界の一部を除き，現状では卸・小売とメーカーとの間で情報共有，サプライマネジメントの構築が図られず，店頭の販売情報を川上に伝達し，店頭の販売動向と連動した生産は実現されていない。メーカーは展示会の概算受注や社会情勢などを鑑み，予測で原材料を調達し，商品を製造しているのが実態である。この動きによりサプライチェーンの各段階に在庫が過多や在庫不足・欠品が発生し，「在庫回転率・在庫維持の機会損失」が発生している。

また，メーカーは卸・小売業からの販売実績・在庫実績が把握できないため，卸・小売からの変動する発注が輸送計画の平準化を妨げ，輸送効率の低下となることとなる。ここに「積載効率の機会損失」が発生していると考えられる。

一方，メーカーは，原材料を調達して，生産計画を立てロット単位で製品を生産している。小売店頭である商品がヒットし，その商品が欠品しても，補充生産が間に合わないことが多々あり，商品供給ができず売り逃しが発生している。これが「調達スピードの機会損失」である。

サプライチェーンの上流に位置するメーカーにおいては，調達する原材料などはすでに「ジャストインタイム」や「かんばん方式」など様々施策を用いて，生産効率の向上に努力し成果を得ている。更なる生産効率向上のためには，現状発生している様々な機会損失を排除する必要がある。そのためには，小売業や卸売業から販売実績，流通在庫の情報を正確かつタイムリーに入手し，消費者のニーズを把握し，分析してこれに基づく商品供給の業務設計や情報活用す

ることにより，適切な生産計画を実行することにより「ムダ・ムラ・ムリ」のない生産を実現すること，すなわちネットワーク型需要充足システム，ネットワーク連鎖システムが求められている。

製造業で発生している機会損失は，サプライチェーンにおける川下の情報が共有できていないことが原因の一つであり，サプライチェーン全体での情報共有化を推進することが機会損失を解消する大きなカギを握っていると考える。

3.4 流通業における機会損失

機会損失をサプライチェーンの視点から商品開発，調達市場から流通市場を鳥瞰すると，下記「図 2.2」に示すとおりにまとめられる。

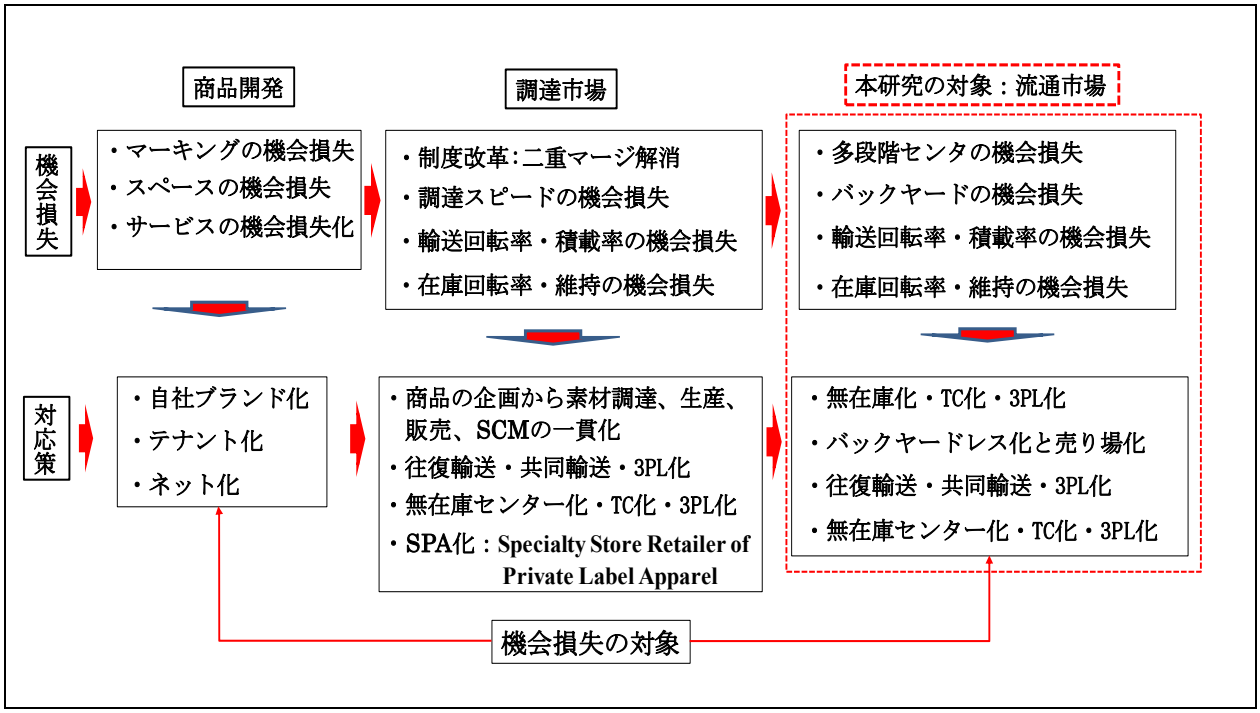

図 2.2　機会損失の対象概念図

流通市場における機会損失は，メーカー・卸・小売のサプライチェーンで発生する多段階の商品在庫・センター運用などに起因する「多段階センターの機会損失」，「在庫回転率の機会損失」，小売業での「バックヤードの機会損失」，サプライチェーンの各段階で発生する「輸送回転率・積載率の機会損失」が考えられる。

「多段階センターの機会損失」とは，「図 2.3」に示すとおり，サプライチェーンを構成するメーカー，卸売業，小売業の各段階に商品在庫を有していることに起因する在庫過多である。

小売業は店頭商品欠品による売り逃しを防ぐため，商品を確保し店舗バックヤードや小売業の物流センターに保管する。卸売業は小売業からの発注に欠品

することなく迅速に対応するため，ＤＣ（ディストリビュートセンター）に商品在庫を持つ。メーカーは卸の発注に対応するため商品在庫を持つ。このようにサプライチェーンを構成する各段階での部分最適を追求することで，サプライチェーン全体の在庫が過多となり，多くの「ムダ・ムラ・ムリ」を発生させることとなり，ここにサプライチェーン全体としての機会損失が発生していると考える。

流通業における「在庫回転率の機会損失」とは，店頭商品欠品による売り逃しを重視することから，商品確保に重点を置くことにより売り上げに対し在庫が過多となり，商品回転率が低くなり経営効率が低下することである。また，在庫が増えることは，更なる商品保管場所を確保する必要が発生し，店舗バックヤード商品ストックの拡大，物流センター商品保管庫の確保と多くの経費が発生することとなり，ますます経営効率が低下することとなる。すなわち，ここに機会損失が発生している。

小売業の「バックヤードの機会損失」は，商品在庫確保により店舗バックヤードの商品ストックを確保することにより売場面積が縮小されること，もしくは売場として活用できないスペースが増大し，売上を上げるチャンスを失うことにより発生する機会損失である。

在庫過多は「商品の値下げや商品廃棄」，「商品の陳腐化や品質劣化」，「商品保管スペースの増大」，「商品整理など付帯作業」，「保管場所からの輸送経費」が発生し，売買差益の減少，売り上げの減少，人件費の増大，無駄なコストの増大が生じ，最終的に企業収益の悪化に繋がる。また，百貨店などの仕入形態である「返品条件付き買取仕入（委託仕入）」は，販売シーズン終了後売れ残った商品を取引先へ返品することできる。返品により取引先は，「返品商品の返品物流経費の負担」「返品商品の商品整理など付帯作業」，「返品商品の値下げ」，「商品廃棄」，「次期販売シーズンまでの保管」など多大なコストが発生し，卸・メーカーなどの取引先の収益低下につながることとなる。

「輸送回転率・積載率の機会損失」は，納品の効率にかかわる問題である。日本の商慣習として小売業へ納品する取引先は，店舗検品所まで取引先の費用負担で納品することが慣習となっている。したがって，小売業は納品車の積載効率とは関係なく，店頭の売り上げに応じた商品を発注している。納品する取引先は，変動する発注に対応するために，過去の発注データに基づき配車計画を立て運用しているのが実態であり，イレギュラーな事態に対応が難しく，多少の増減を意識した余裕のある計画を立てることとなる。そこには，自ずと「ムダ・ムラ・ムリ」が発生し，積載率が低下し，ここにも機会損失が発生していると考える。

いままでで述べてきたように，流通業において様々な「ムダ・ムラ・ムリ」が発生し，機会損失が発生していることが明らかになった。ここで発生している機会損失は，コストとして最終的には最終消費者が負担することとなり，こ

の機会損失を無くすことが消費者に高い価値を提供することに繋がると考える。

しかしながら，サプライチェーンを構成する多くの企業は，企業ごとに部分最適を追求しており，サプライチェーン構成する前後の取引先に自社運用・システムを押し付けるなど最適化とは逆の方向に進んでいる事例も未だに散見される。

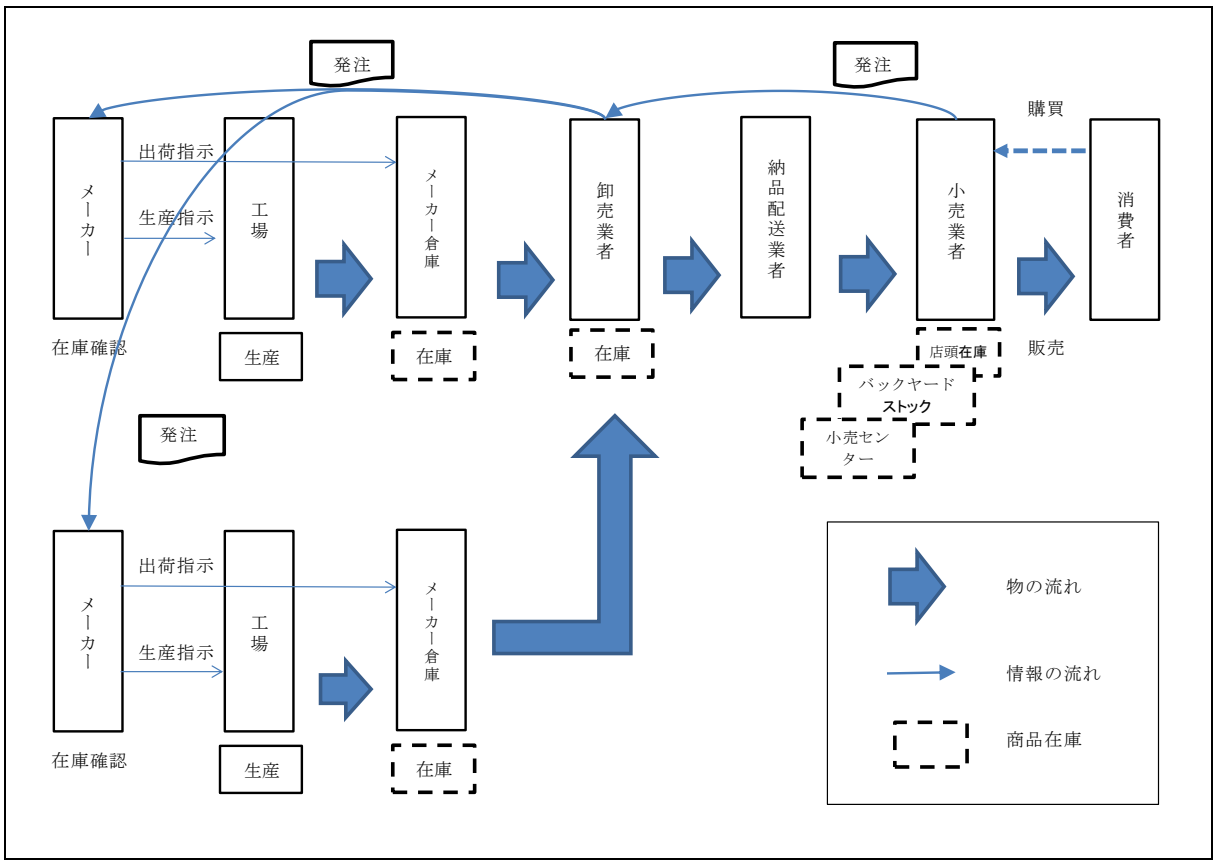

図 2.3　従来型メーカー・卸・小売の物の流れ

4. 機会損失の分析と低減の方法

4.1 SCM と情報共有化

いままで述べてきたように「流通業の機会損失」および「製造業の機会損失」は，サプライチェーン全体で仕組みが効率的に運用され，消費者に高い価値で提供できるようにマネジメントが機能していないため発生していると考えられる。すなわち，個々の企業の業務を接続させ，製造段階から販売段階，最終消費者まで，それぞれの業務を同期化することができていないことと考えられる。

機会損失を無くすためには，SCM を機能させることである。機能させるためには，基本的には取引に連なる各企業がパートナーとして手を握り合い，十分な情報交換と共有化と小ロット短納期生産・物流を基礎に，売れる商品を，余さず，切らさず小売店頭に供給することである。

各企業はサプライチェーン全体の情報共有化を図り，最も効率よくなるよう

に行動することにより，欠品を防ぎ，消費者に的確な商品を供給するとともに，かつ在庫過多を回避することにより，サプライチェーンが供給する商品の価値下落を防ぎ，商品の価値を維持することによって，各企業の利益を確保することである。

結果的には，各企業はサプライチェーンへの関わりを高め，その中での「統合」はますます強められる。この好循環の成果を継続することによって，SCM は機能し続けることができると考える。

このように SCM の導入にあたり，情報共有化をスムーズにすすめるかどうかが根本的な要素と考えられる。更に SCM を維持し，発展させていくために，既存の制度，商慣習，組織体制や取引形態を見直し，ビジネスネットワーク化に適切な制度・組織体制づくりを検討する必要がある。

しかし，SCM の現状は，メーカー・卸・小売の業態間，商品分野の違いによる業種間で情報連携が図れていない。消費財流通全体で多くの異種システムが並立して，統一化されていない課題が生じている。この現状を鑑み，経済産業省は 2006 年度から「流通システム標準化事業」を実施し，データ交換に関する規約を取り決め，標準的な仕組みが着実に実運用に移行されるべき検討を継続し，現在に至っている。この標準 EDI が「流通ビジネスメッセージ標準（流通 BMS）」である。

なお，標準規約とは，①EDI 取引業務手順，②EDI メッセージ，③商品コード・事業者コードなど使用コード，④通信手順（通信プロトコイル）などである。この標準機約を取り決めることにより，EDI システムの検討・開発・導入に要する時間とコストが削減されるとともに，多くのシステム会社がこの標準規約に則ったパッケージソフトを開発・販売・提供することにより，リーズナブルなコストで EDI システムが導入できることとなり，メーカー・卸・小売の情報共有化が加速されることが期待される。この情報共有化が SCM を機能させる原動力となる。

4.2 機会損失の低減の方法

サプライチェーンにおけるビジネスネットワーク化を前提として，発生している機会損失を低減するための具体的な方策を以下に検討する。

「図 2.2」に示したとおり，機会損失ごとにその対応策を一覧にまとめると「表 2.1」のとおりである。

「多段階センターの機会損失」「在庫回転率の機会損失」の対応策は，前提として川上から川下に至る情報共有化を支える EDI を構築するとともに，サプライチェーンの各段階にある商品在庫を上流側に集約保管し，小売側の販売実績情報に基づき，ジャストインタイムで商品を店頭に供給することである。それにより，小売側の物流センターは TC 化し，在庫を持たないこととなる。また，商品在庫の保管を 3PL 業者に委ね，複数の取引先と複数の小売業者を集約

して業務を遂行することにより，保管と輸送の効率向上を図ることが可能となる。

「バックヤードの機会損失」の対応策は，販売実績情報によりジャストインタイムで店頭へ商品補充を共有する体制を構築することにより，店舗バックヤードの商品在庫を排除する。不要となったバックヤードを売場へ転用することにより，売上増を狙うものである。

「輸送回転率・積載率の機会損失」および「在庫回転率・在庫維持の機会損失」の対応策は，サプライチェーンの各段階に存在する商品在庫を 3PL 業者に集約し，小売業への納品業務を委託することにより，複数のメーカーと複数の小売業者を集約して，一括した業務を遂行することにより，保管と輸送の効率向上を図ることが可能となる。

「調達スピードの機会損失」の対応策は，SPA 化や SCM の一貫化により，店頭への商品リードタイムを短くし，販売実績の把握に基づく追加生産や生産調整が可能となる。それにより，商品の欠品や在庫過多によるロスを防ぎことができる。また，シーズン中の販売実績情報に基づき，新商品を投入することにより店頭の品揃えの幅を広げることが可能となる。このように，スピード感がある商品供給体制と販売実績の同期が，機会損失をなくすことが可能となる。

以上述べてきたとおり，これら対応策は，サプライチェーンにおける情報共有化を前提としたサプライチェーン全体の最適化により，すなわち企業間の垣根を越えて複数の企業がビジネスネットワークを組んで最適化を目指すことにより，サプライチェーンで発生している様々な「ムダ・ムラ・ムリ」を排除し，機会損失ゼロ化を実現するものである。

表 2.1　機会損失と対応策

	機会損失	対応策	内容
流通市場	多段階センターの機会損失 在庫回転率の機会損失	無在庫化	サプライチェーンどこかに在庫を集約して保管する。迅速な店頭への商品供給体制を構築する。
		TC 化	
		3PL 化	
	バックヤードの機会損失	バックヤード削減化	バックヤードの商品ストックを売場へ転用する。 販売状況に応じた迅速な店頭への商品供給体制を構築する。
		バックヤードの売場化	
	輸送回転率・積載率の機会損失	往復輸送	共同化，3PL 化を推進し，複数の取引先の納品を集約して配送し，積載効率を高める。 店舗への納品の集約化進めるための情報共有化を推進する。
		共同配送	
		3PL 化	
調達市場	制度改革	SCM の一貫化 川上から川下までの情報共有化	消費者の嗜好の移り変わりを迅速に製品に反映させ在庫のコントロールが行いやすい。
	調達スピードの機会損失	SPA 化 情報の共有化	
	輸送回転率の機会損失	往復輸送	共同化，3PL 化を推進し，複数の取引先の納品を集約して配送し，積載効率を高める。
		共同配送	
		3PL 化	
	在庫回転率維持の機会損失	無在庫化	共同化，3PL 化を推進し，複数の取引先の納品を集約して配送し，積載効率を高める。 店舗への納品の集約化進めるための情報共有化を推進する。
		TC 化	
		3PL 化	

5. 小売業における機会損失

5.1 百貨店おける機会損失

バブル経済の崩壊，リーマンショックを経て，百貨店は売上高が低迷し，取り巻く環境は大変厳しい状況にある。年間売上高のピークは 1991 年の約 9 兆 7130 億円であり，ピーク時の 6 割までに低下している。百貨店の構造改革しなければ，売り上げの減少，利益減少を止めるのは難しい状況にあり，ビジネスとして成り立たなく可能性が指摘されている。この状況を改善するためには，百貨店と取引先との間に SCM を構築し，情報共有化により機会損失の改善を図ることが喫緊の課題であると考える。

百貨店のビジネスモデルの特徴は，その仕入形態にある。百貨店の仕入形態は，大きく２つの形態に分類され，買取仕入（買取仕入）と消化仕入（売上仕入)である。買取仕入は更に完全買取仕入と返品条件付買取仕入に分類される。これらの仕入形態をまとめると下記「表 2.2」のとおりとなる。

買取仕入は，取引先から百貨店が商品を仕入れるため，納品の際に検品を実施し，仕入伝票を計上し，買掛金に計上し所有権の移動を明確に行う。一方，消化仕入は店頭で商品が売上計上された時点で仕入が立つので，売上計上されるまで取引先の商品である。よって，納品時に仕入検品などは実施せず，取引先の都合で自由に商品を店頭に出し入れし，他店に移動することができる。

店頭販売体制は，買取仕入は，百貨店の社員と取引先の派遣社員による体制である。一方，消化仕入は，ほぼ取引先の派遣社員によって構成されている。現状商品の在庫管理，商品業務は取引先の仕組みで運用されている。

表 2.2 百貨店仕入形態一覧

	仕入形態	在庫商品 所有権	販売･在庫 管理責任	仕入計上	返品可否
1	買取 (本)仕入 完全買取	百貨店	百貨店	納品時仕入 伝票計上	返品不可
2	買取 (本)仕入 返品条件付	百貨店	百貨店	納品時仕入 伝票計上	取引先了承要 返品伝票で返品
3	消化 (売上)仕入	取引先	取引先	売上計上時 POS 自動仕入計上	取引先の指示 により返品

買取仕入・消化仕入以外に，店舗内のスペースを賃貸契約する形態が「テナント（賃貸契約)」である。場所貸しとなるので，百貨店内に営業しているが，販売責任は一切負わない形態である。また，テナントの売上は，原則百貨店の売上として計上しない。特にレストランフロアや一部独立店舗，ユニクロや書籍などのテナントで運用されている。

近年，アパレルを中心に仕入形態が消化仕入にシフトしている。特にファッション性の高い婦人服や子ども服は消化仕入の割合が急増し，売上の 7 割以上が消化仕入となりつつあり，また 9 割が消化仕入の百貨店もある。消化仕入割合が増大することは，百貨店の売買差益率を減少させ，自らの販売力低下，業務改善能力の低下に繋がっている。

消化仕入にシフトしている要因は，取引先の商品管理レベルが IT 化により向上し，取引先が商品の販売動向が把握することができるようになり，売れていない店舗の商品在庫を売れている店舗に商品移動が簡単にできる消化仕入に取引形態を転換し，それにより取引先の商品消化率を高め，商品の不良在庫化を防ぐためである。一方，百貨店側は消化仕入化により自社販売員が必要無くなり人件費が削減できるとともに商品の在庫リスクを回避できるなど両者の目先の利害が一致した結果からである。

一方，百貨店の買取仕入の運営形態は，未だに伝票を中心とした紙ベースの管理が主流を占めている。詳細の業務フローを「図 2.4」に示すとおりである。百貨店の商品管理は，商品の品番(商品群・売場場所管理)による管理が主で，コンビニエンス業界やチェーンアパレルなどで当然として実施されている単品管理はほとんど実施されていないのが実情である。これは，取引先との情報共有化が行われていないことを示している。すなわち，SCM が構築されていないことを示している。

一部の大手百貨店では，取引先との EDI 化が 2000 年ごろから進められてきているが大きく進展し，変化してきているとは言えない状況を呈している。それは，百貨店業界で EDI 手法が統一されず，かつ運用の徹底が図れず，物流･商品管理運用コスト削減や販売機会損失削減などの効果を得ることができず普及していないことが挙げられる。

特にアパレルを中心としたファッション業界では，QR，SCM 戦略の影響を受け，取引先との情報共有を基本に，取引先と百貨店とが協働していく考え方が進められてきているが，未だその成果を得ていないのが実情であると考える。

ここに，百貨店と百貨店取引先の低迷の原因があると考える。すなわち百貨店と百貨店取引先との間に SCM が構築できないこと，ここに百貨店と百貨店取引先の機会損失が存在していると考える。

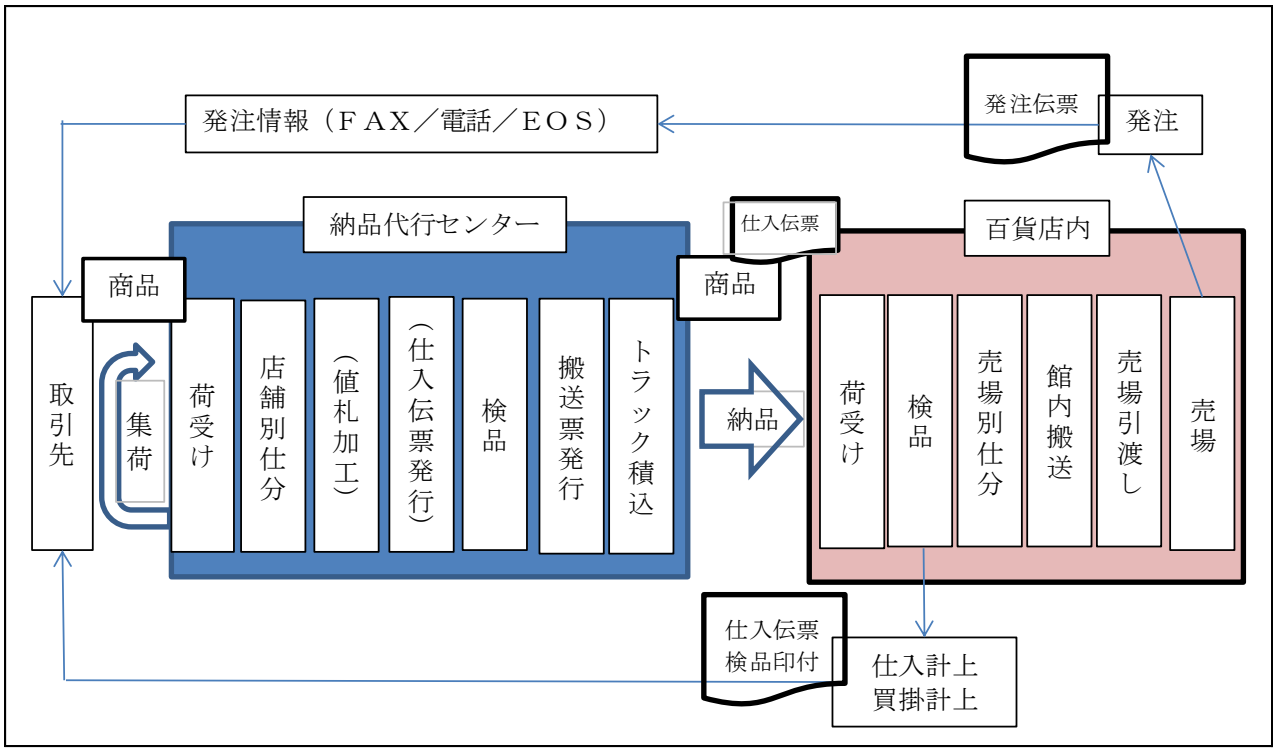

図 2.4 百貨店商品納品フロー

5.2 大型小売店におけるバックヤードの機会損失

リーマンショック以降，百貨店・総合スーパー（ＧＭＳ）をはじめとする大型小売業は販売額が低迷し，取り巻く環境は大変厳しい状況にある。この環境下で収益を向上させる方法として，すでに投資してある既存店舗を最大限に活かす方策を検討する必要があると考える。

百貨店・ＧＭＳ・ＳＭをはじめとする大型小売業は，店舗毎にバックヤードを有している。バックヤードには，商品・用度・什器備品ストック，厨房，事務所，休憩室，通路，階段（避難階段含む），搬入口，ごみ処理室，及び機械室等から構成されている。

バックヤードを売場に転用すれば収益を上げる事が可能であるが，現在は店頭業務をサポートする機能利用されており，収益を得られてとは言えない。すなわち，ここに機会損失が発生しているものと考えられる。

商品ストックバックヤードを有することは，在庫過多・過剰による「値下げや商品廃棄」，「商品の陳腐化や品質劣化」，「商品保管スペースの増大」，「商品整理など付帯作業の増加」が発生し，売買差益の減少，売り上げの減少，人件費の増大，無駄なコストの発生などが生じ，最終的に企業収益の悪化に繋がることが懸念される。

よって，バックヤードの機会損失を無くすことはサプライチェーン全体に多大なメリットを得ることとなる。

5.3 自動倉庫における機会損失

コンビニエンスストア，GMS や SM などのスーパー業界では，IT 化の進展や取引先との EDI など様々な情報システムが導入され，小売りと取引先との EDI による情報データ交換により，多品種少量発注をベースにした多頻度即納型物流システムをもたらすこととなった。この傾向はオーダーピッキングを主とする流通センターにも反映し，倉庫や流通センターの自動化と統合管理システムの実現にすることとなる。流通センターの自動化の中心システムは，立体自動倉庫であり，仕分け機である。

立体自動倉庫を導入する卸売業や運用を受託する物流業においては，業務の支障が発生しないように稼働率及び機会損失・投資効率の検証が十分に行われておらず過剰投資傾向といえる。

ここに，過剰投資という機会損失が発生していると考える。

6. おわりに

第２章では，機会損失の各定義を調査し，本研究での定義として「方策の選択により得ることができなかった利益」とした。

SCM の定義は，「個々の企業の業務を接続させることであり，製造段階から販売段階，最終消費者まで，それぞれの業務を繋ぐことでサプライチェーン全体を最適にすることである。」とした。

また，SCM おける機会損失を流通業，製造業の領域で抽出し，機会損失と対応策を一覧にまとめるとともに，小売業における３つの機会損失の課題を抽出した。この３つの課題に対し，次章以降で更に研究を進める。

参考文献

[1] 生島義英, 佐藤哲也, 唐澤豊, 若林敬造, 小売業におけるバックヤードの機会損失に関する基本的研究, 日本ロジスティクスシステム学会誌 Vol.15, No.1,2016 年 3 月
[2] 日本オペレーションズ・リサーチ学会辞典, URL:http://www.orsj.or.jp/~wiki/wiki/index.php/（2016 年 1 月確認）
[3] 小学館,デジタル大辞泉,URL: http://www.daijisen.jp/digital/index.html
[4] 森村英典, 牧野都治, 真壁肇, 杉山高一,統計・OR 活用事典,東京書籍, 1984 年 9 月
[5] 日本百貨店協会, 2006 年版百貨店ＩＴ白書, 日本百貨店協会, 平成 18 年 1 月
[6] 財団法人流通システム開発センター, 流通ＢＭＳ導入の手引き, 流通ＢＭＳ協議会, 平成 23 年 9 月
[7] 唐澤豊, 現代ロジスティクス概論,ＮＴＴ出版, 2000 年
[8] 陳玉燕, 唐澤豊, 若林敬造, 井上敬介, 生島義英, 豊谷純, ＳＣＭ戦略論の研究と戦略フレームワークの提案, 日本ロジスティクスシステム学会誌 Vol14No1,2014 年 12 月
[9] URL:https://www.merriam-webster.com/dictionary/opportunity%20cost merriam-webster.com (2016 年 1 月確認)
[10] 菊池康也, ＳＣＭの理論と戦略, 税務経理協会, 平成 18 年 4 月
[11] 株式会社ゴーガ編, URL:http://m-words.jp/w/(2016 年 1 月確認)

第３章　日本の百貨店における SCM の取組みと機会損失に関する基本的研究

1. はじめに

1.1 研究目的

バブル経済の崩壊，リーマンショックを経て，百貨店は売上高が低迷し，取り巻く環境は大変厳しい状況にある。日本百貨店協会が発表した 2016 年の全国百貨店売上高は, 2015 年比 2.9%減（既存店ベース）の 5 兆 9780 億円であった。年間売上高のピークは 1991 年の約 9 兆 7130 億円であり，ピーク時の６割までに低下している。百貨店の構造改革しなければ, 売り上げの減少, 利益減少を止めるのは難しい状況にあり，ビジネスとして成り立たなく可能性もある。

構造改革の一つとして，製・配・販三層を対象とし，全体最適化を図る仕組みとして「サプライチェーンマネジメント（SCM)」がある。SCM とは，「メーカー・卸売業など取引先との流通サプライチェーンの仕組みが効率的に運用され，消費者に最も高い価値で提供できるようにコントロールすることである。

流通業である百貨店，卸，生産の各企業それぞれの業務を協働という視点から一つのつながりとして，生産から販売に至る一連の流れの全体最適化を図る SCM への取組みが動きはじめている。

本研究の目的は，百貨店業界に関与する卸・メーカーを含めたサプライチェーンを研究対象とし, SCM がどのような効果を百貨店業界に齎すのかを明らかにする。

本研究では，次の３つの内容について論じる。①百貨店サプライチェーン改革の内容と経緯，現状を整理する。②取引先との情報共有化を推進する EDI モデルを理解し，業務効率化の具体的な内容を解析する。③取引先との SCM の情報共有化により，サプライチェーンマネジマントに期待される効果を算出し，その経済効果を測定する。

1.2 先行研究調査

先行研究として, 参考文献[15][16][17][18][19][20][21][22][23][24][25]がある。いずれの論文は百貨店における SCM もしくは EDI, 配送の取り組み内容の事例紹介や SCM の概念，情報技術などに関わる内容であった。先行研究では，百貨店業界で取引先との SCM の情報共有化により，SCM に期待される効果を解析し，その経済効果を測定した研究論文は皆無に近い。

百貨店業界において, SCM に期待される効果をビジネスプロセスに基づき分析・解析し，その経済効果を策定することは, SCM を導入するうえで経営判断上重要な情報であり，このシステムの導入可否の一つの判断材料であると言える。よって，新たな研究対象として重要な意味を持っていると考える。

2. 百貨店業界の SCM への取組みの経緯

2.1 百貨店業界の情報システム化への流れ

百貨店業界における SCM への取組みは, 業務の情報システム化から起点が発している。POS システム導入することにより, 業務ごとに積み重ねられてきたシステムが IT（インフォメーション・テクノロジー）の進展ともにトータルシステムとして再構築され, SCM に対応にまで発展することとなる。ここでは, 百貨店協会の資料, 2001 年版・2006 年版百貨店 IT 白書, 流通システム開発センターの資料を基に百貨店に業界における SCM に至る取組みの経緯を時系列に整理する。

百貨店の情報システム化・IT 化の進展はその変化の過程の特徴から概ね次の三つの段階を経て今日に至っている。第 1 段階は, POS システムを中心としたシステム導入時期（1980 年頃まで）から普及期（1985 年頃まで）, そして拡大期（1992 年前後まで）である。第 2 段階はこれまで, 一つ一つの業務ごとに構築されてきたシステムが情報活用（すなわち情報化）といった視点からトータルなシステムとして再構築されると同時に, バブル崩壊後の状況を反映して, BPR や QR・SCM への対応の時代（今日まで）である。そして第 3 段階として, パソコン, インターネット等の普及やモバイル時代に適合した IT（インフォメーション・テクノロジー）活用の時代（今日まで）である。

2.2 システム導入期

初期の百貨店における情報システム化の最大の課題は POS システムの導入であった。いうまでもなく POS システムはポイント・オブ・セールスの略で, 販売時点情報管理システムである。導入の目的は, 商品情報, 価格情報, 売上情報, さらに納入先情報を値札から販売時点で読み取り, その情報を加工・分析して, 一つにマーチャンダイジング（商品仕入れ, 品揃え等）, 二つに決済処理（支払情報）と一元管理していこうとするものである。しかしながら, レジまわりの省力効果や売上高の早期把握と言った効果が結果として主力を占め, POS 本来の目的であるマーチャンダイジングへの効果は期待されなかったのが実態である。

百貨店における POS システムの導入時期については, 少なくとも 1972 年頃から POS 研究が始まった。導入目的や効果, 課題についてみると, 「売上・商品情報の把握」, 「省力化」, 「クレジット対応」が主であり, POS システムの本来の目的であるマーチャンダイジングとはかけ離れたものであった。

2.3 システム普及期～拡大期

百貨店の情報システム化の中心は POS システムである。もっとも, 情報システム化以前の EDP 化の時代においては, 経理財務業務が早くから EDP 化されて

いるが，まさに業務処理中心で，情報システム的考え方はなかった。ちなみに百貨店における EDP 化は全産業的にみても 1960 年代後半以降であり，百貨店もその例外ではなく決して遅れていたとは言えない。

1990 年代には，完成期を迎えたシステムとしては「経理財務システム」と「クレジットシステム」である。経理財務システムは前述のとおりであるが，クレジットシステムについてみると，クレジットカードが
1985 年以降次第に提携化や自社カード化の流れから普及し，それに伴い与信業務（オーソリゼーション）の迅速化が要求された結果，POS システムの普及促進的役割を果たしながら完成期を迎えたと言える。それ以外の多くの個別業務システムは発展期にあると言える。そのほかで注目すべきシステムでは，「部門別利益管理システム」および「通販システム」が導入期であり，「受発注システム」および「調達物流システム」は実験期にあるという事実である．こうした状況も都市と地方とではかなりの差異があることも事実で，都市においては殆どのシステムが完成期にあり，「受発注」，「部門別利益管理」「調達物流」のシステムが導入ないし実験期にあるにすぎない．また地方では殆どのシステムが発展期にあるが，「配送管理システム」が導入期のほか「通販システム」，「受発注」，「調達物流」，「部門別利益管理」がまだ実験期段階にあるというのが実態である．

なお，「部門別利益管理システム」や「調達物流システム」の導入が進展しない背景には，商品政策の面で取引先依存度が高かったことがあげられる．当時は殆どが取引先の営業が中心になって商品管理をしていたために，発注業務も取引先にまかせ，百貨店は発注伝票に判を押すだけといったことが多く，百貨店の管理は支払管理中心だった．加えて，商品管理面から言えば，売上情報も単品ベースではなく，品番レベルであったという背景も無視できない．

百貨店の情報システム化が POS システムの導入を契機として百貨店のさまざまな業務分野におよび，1991 年頃にはかなりの導入が進展してきた。

その後，システム統合やデータベース化によって情報の活用がはかられ，1994 年頃までには取引先とのオンラインシステム化の導入期を迎えた。しかし，取引先とのオンライン化にしても，百貨店ごとにコードや伝票のフォーマットが異なり，伝送手順やメッセージもそれぞれバラバラでは，システム作りの面や効率の面からも高コストになる。また，値札からの情報入力ひとつとっても正確性や迅速化のために，値札の標準化が求められた。

しかしながら，1995 年頃から QR(または QRS）が登場し，しだいに百貨店業界においても対応に迫られた。この QR は，わが国においては，経済産業省繊維製品課が中心となり，繊維産業構造審議会における今後の繊維産業のあり方を示す「繊維ビジョン」の中で示されたものである。その基本的考えは，これまでの「プロダクトアウト」から「マーケットイン」の生産体制の構築にあり，生産から流通・小売を含む繊維産業全般を，情報技術を駆使して実需型の産業に

再構築するものである。すなわち, SCM システムを構築するものである。

その基本要件に JAN コードの利用がある。その当時 JAN コードの利用は百貨店業界やアパレル業界においても殆ど使用されておらず, したがって, QR への対応にはそれらの基盤を早急に整備する必要があった。

百貨店業界は 1995 年度の事業として（財）流通システム開発センターより, QR 基盤整備事業の一環として「値札標準化」の検討を開始した。値札標準化の目的は, 「①コスト削減を図ること」「②QR への対応基盤を整えること」「③取引先とのシステムの相互互換性を確保すること」であった。

値札は各百貨店によってサイズや表示の仕方がまちまちで, 納品の際に納入業者が作成し取り付ける。そのために, 各納入業者も各百貨店の値札を在庫として抱え, その保管コストや作成コストも負担している。また, 百貨店においても取引先ごとに値札引渡しなどの管理コストもかかっており, 当時の百貨店協会の調査では大手百貨店で売上高の 0.01%のコストが発生していた。このために, 各社各様の形状や種類を集約し, 百貨店各社のロゴは発行時に値札発行機で印字するなど, コスト削減を図ることが目的である。次に, JAN ソースマーキングの利用を促進し, 読み取りシンボルとしての JAN コードへの切り替えをスムーズにし, QR への対応基盤を整えることであった。また, 値札の標準化によって取引先とのシステムの相互互換性を確保し, さらに EDI による情報の迅速性や正確性を確保しようとすることであった。こうして, 1995 年 9 月に, 百貨店「標準値札」として制定した。現在では殆どの百貨店が採用している。(「図 3.1　百貨店標準値札例」参照)

一方で, ソースマーキングの普及とともに, 百貨店値札は必要ないとする意見もアパレル業者中心にあり, 百貨店値札廃止への移行が進捗している。

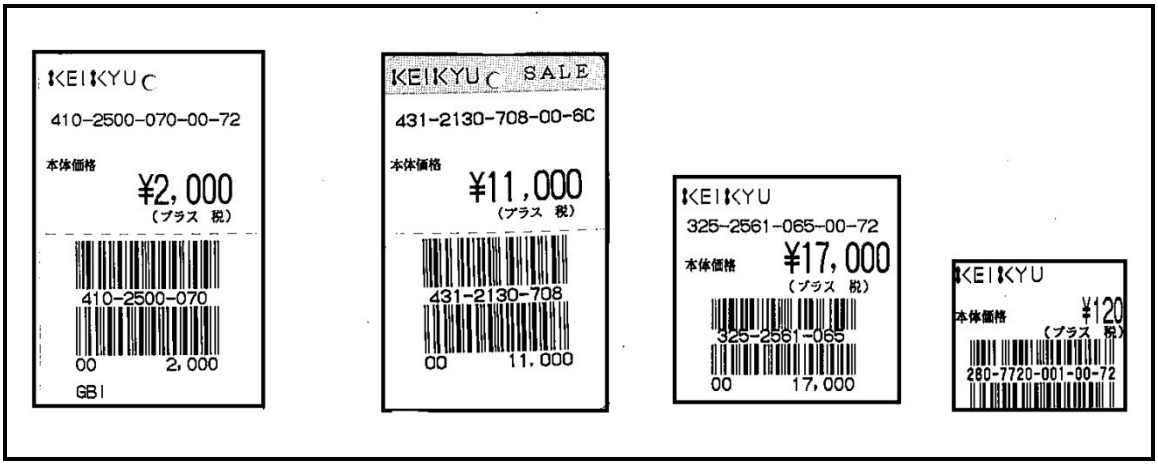

図 3.1　百貨店標準値札例

QR への対応は EDI の標準化の第一歩である。百貨店業界が流通システム開発センターから, 1996 年度の受託研究事業の形で作業を進められた。当時百貨

店にかかわる標準 EDI としては，一つに流通システム開発センターの EDIFACT 基準による『流通 EDI』（1996 年 3 月制定）のものと，QR の推進機関である繊維構造改善事業協会が中心となってアメリカのモデルをもとに作った『繊維 EDI』（1996 年 8 月制定）の二つが存在していた。どちらも百貨店の現行取引実態からみてそのままでは対応が出来ないといったことに加え，標準 EDI を運用していく上での問題点等も検証していくためにも，百貨店バージョンの標準化を検討した。

主要な流れは百貨店の商品企画から発注，納品，精算であるが，その間にメーカー主催の展示会における仮発注，マスター登録，仕入伝票の送付（発注伝票），補充発注，検品，返品，支払データ送付など等さまざまな業務がからんでいる。

標準化の内容は大きく商品マスター，発注データ，売上データ，在庫情報の四つにかかわるメッセージの標準化である。商品マスターとは，商品のカタログ情報，商品属性情報（基本属性，スタイル，SKU），取引条件などから構成され，各百貨店にとって極めて重要な情報である。しかしながら，このマスターの情報が各社バラバラであり，百貨店によっては，いわゆる商品関連情報以外も登録してあり，標準化にあたっては，必要不可欠な情報を絞り込むのに労力が必要であったその理由はオンライン受発注を実施している百貨店が少なかったことがその背景にある。

こうして，1996 年に標準案をとりまとめ，流通システム開発センターに提示するとともにアパレル業界にも案を提示し，調整作業に入った。この結果『流通 EDI』には本案が標準 EDI として取り入れられたが，アパレル，ボディファッション協会との調整過程では，百貨店案にはオプション項目が多すぎるとの観点から，かなりの項目が削除・整理され（1998 年 4 月），最終的には『繊維 EDI』に取り込まれた。

1999 年においては，QR 基盤整備の受託研究事業として，SCM（shipping carton または container marking：出荷梱包識別ラベル）および ASN（advanced Shipping notice：事前出荷明細案内）のメッセージ標準化を策定した。

これらの標準化の大きな目的は，納品時における検品作業の効率化にある。百貨店の検品は，納品時における全数検品が原則的に実施されている。こうした SCM や ASN の実施には単品管理が前提であるが，1996 年の第 7 回百貨店協会実態調査によれば，単品管理を実施している百貨店は全国で 47 社 63.5％に及んでいるが，いずれの企業もグロサリー・日配品・ステショナリー・ストッキング靴下など一部の商品分野での採用に過ぎない。まして，アパレル商品の単品情報の EDI は 12 社となっている状態である。こうした実態のなかで，QR 基盤の整備として EDI メッセージの標準化を行った。

1995 年から 1997 年において，QR の基盤整備としての EDI 標準化作業は終了したが，作業開始頃の QR への認識について第 7 回百貨店協会実態調査の結

果からみると，まず QR に対する認知度は「よく知っている」が 66.2%となっているが，一方では「聞いたことがあるが内容はよく知らない」も 30%以上もある。さらに，QR の必要性についてみると，「絶対に必要」が 45.9%，「なんとなく必要」が 44.6%を占め，必要性は認識していた。また今後の対応についてみると，「積極的に導入」は 3.4%にすぎず，「導入前提で検討研究」が 48.3%，「今後検討」が 34.5%となっている。これらの結果からも分かるように，当時は QR といった言葉がようやく認知されつつある段階で，それはもっぱら情報システム担当者が中心となっており，トップレベルの戦略的対応にはほど遠い状況であったと言える。

こうした状況下における QR 対応の基盤整備としての EDI メッセージの標準化であり，ネットワーク時代への準備は整備されることとなった。

2.4 情報活用期

システムを活用する時代との認識から，百貨店協会では情報システム推進委員会は廃止され，それにかわって百貨店協会 BPR 委員会が立ち上がった。

しかし，百貨店業界 EDI の現状は旧来からの複数の EDI 標準メッセージが存在する。具体的には「繊維産業 EDI 標準（1997 年)」，「ＪＡＩＣ標準（2000 年)」，「流通標準 EDI（JEDICOS：1997 年)」などである。このようにメッセージ標準が複数あり，標準メッセージの中で利用しているメッセージが企業により異なり，結果として各社により対応が異なるため，取引先は百貨店各社に対応するため，非効率な投資が必要となっている。特に取引額が小さい中小取引先は，データ交換をするうえでメッセージルールなどの様々なルールに対応するための投資や IT スキルの問題があり，且つ自社システムとの接続が難しいなど EDI を導入することの大きな障壁となっている。

取引先システムと比較的に簡単に接続できる仕組みとして，インターネットを介した EDI 情報共有サービス（WEB-EDI）がある。この WEB-EDI も複数のサービスが存在する。2001 年にスタートした伊勢丹系「IQRS.Net」と 2002 年 9 月には高島屋・三越が「百貨店 eMP サービス」である。

このように多くの異種システムが並立して，統一化されていない課題が生じている。

この現状を鑑み，経済産業省は 2006 年度から「流通システム標準化事業」を実施し，データ交換に関する規約を取り決め，標準的な仕組みが着実に実運用に移行されるべき検討を継続し，現在に至っている。この標準 EDI が「流通ビジネスメッセージ標準（流通 BMS)」である。

2010 年 10 月には経済産業省が主導した，流通ビジネスメッセージ標準（流通 BMS）が百貨店業界の取引に必要な標準メッセージ「百貨店版 Ver2.1」を公開し，取引先と百貨店を結ぶ標準化した EDI の基盤が整備された。標準規約とは，①EDI 取引業務手順，②EDI メッセージ，③商品コード・事業者コードなど

使用コード，④通信手順（通信プロトコイル）などである。この標準規約を取り決めることにより，EDI システムの検討・開発・導入に要する時間とコストが削減されるとともに，多くのシステム会社がこの標準規約に則ったパッケージソフトを開発・販売・提供することにより，リーズナブルなコストで EDI システムが導入できることとなり，メーカー・卸・小売の情報共有化が加速されることが期待されている。

流通ビジネスメッセージ標準（流通 BMS）の導入により，「インターネット技術を活用し，廉価に導入できる EDI の仕組みの構築」，「取引高に応じた費用負担で導入可能な料金体系のサービスの提供」，「百貨店業界に留まらず流通業界全体で EDI の統合を図り，標準化したシステムの構築」が期待され，富士通 FIP など各システムベンダーから流通 BMS に対応するパッケージシステムがリリースされている。

SCM への対応もようやくトップレベルにおける認識が高まり，システムレベルから企業レベルとなり，現場での導入段階に入って行った。特に SCM は政府の政策的支援もあり，徐々に普及しつつある。

一方，百貨店協会 BPR 推進委員会は百貨店の課題である「高コスト低収益体質」からの脱却を目指し，「ローコストオペレーション」の実践を取引先とのコラボレーションにより推進していった。この委員会の前身は 1996 年に「業務改善プロジェクト」として発足し，その後「業務改善委員会」に改組，そして 1998 年に現在の BPR 推進委員会となった。この間に，ハンガー納品に係る業務改善の検討や，検品削減・伝票削減体制の構築に向けた活動，さらに玩具流通に係る問題などについて活動してきた。特にハンガー納品については，納品から店頭陳列，返品といった過程のなかで 3 回もハンガーの架け替え作業がバックヤード等で行われ，そのハンガーの返却業務やストックスペースの確保など，高コストにつながっていたのが実態であった。それを改善するために統一ハンガーとして標準化し，1997 年 10 月に業界として導入して，架け替え作業の減少と同時に導入の結果としてリサイクル，リユースされ省資源といった環境問題にも貢献した。その後，取引改革の検討を続け，2001 年には「コラボレーション取引」と称する取引改革を実行した。この新しい取引は百貨店の欠品率や返品率を下げ，百貨店における売り逃しを減少させ売上を伸ばそうとするもので，百貨店は消化率を取引先に約束し，取引先は納品率を約束して取引するものである。当然そのためには，需要予測や EDI といった IT 技術を利用したシステム対応が前提となっている。先にも見たように，百貨店は 1990 年代前半まではいわゆる「情報システム化」が中心に行われ，その後システムの統合，システム間連動といった課題を経て情報活用が重要視された。そして 1990 年代の後半には，BPR から次第に IT の活用の時代へと移行して今日に至っている。

2.5 IT 活用期

IT 時代と言われる背景には，コンピュータ技術はもちろんであるがネットワーク技術の進展があり，携帯電話やモバイル端末器の著しい発展がある。百貨店においても，ホームページの開設やギフトのネット受注，さらに携帯電話による情報サイトの開設など新しい分野への進出が活発になってきている。そして百貨店業界においては，2001 年 1 月に百貨店の IT に関する総合調査の結果をとりまとめた「百貨店 IT 白書」を刊行され，百貨店における B to B や B to C への取り組みなどの実態を明らかされた。

そこで，同書から今回の調査研究と関連の深い事項をピックアップして最近の実態を鑑みる。まず，B to B 関連から EDI 取引の実態をみると，EDI 取引を実施している百貨店は４２社である。EDI 交換情報としては 41 社のうち 65.9%の企業が「発注情報」であり，50.0%が「支払情報」，68.2%が「商品マスター（属性）情報」，54.5%が「販売情報」となっている。また，今後導入したい交換情報のうち，その比率が 40%以上の高い項目を順に列挙すると，「商品マスター情報」「発注情報」「販売情報」「事前出荷明細情報」「支払情報」「検品結果情報」の順になっている。

次に，物流関連について，利用値札の形態について見ると，百貨店標準値札の利用が 51 社（60.6%）となっている反面，値札廃止（ソースマーキングのみ）が 30 社（35.7%）となっている。さらに，検品等の状況について見ると，殆どが手作業（90.5%）で，SCM や ASN はそれぞれ 10.7%，2.4%と極めて低率で依然普及していないのが実態である。今後の導入予定についても，48.8%の企業は「3 年以上先あるいは導入予定がない」と答えている。

一方，B to C への取組みは，ホームページ開設は多いものの，インターネット通販は特にギフト期におけるインターネット受注は，全体的な売上規模はまだまだ小さいが，年々増加している百貨店が多い。しかし，楽天やアマゾンなどの e コマースは，この 10 年で急成長しており，百貨店はこの急成長の恩恵を得ることができていないのが現状である。また，一部の百貨店ではオムニチャネルに進出しているが，いまだ道半ばといった状況である。

3. SCM 普及推進と阻害要因

3.1 SCM の阻害要因

SCM における情報共有化とは，サプライチェーンを構成する企業間で，お互いに保有している情報を EDI で相互にやり取りすることによって，お互いのビジネスに活用することである。これにより，個々の企業の最適化への活動をサプライチェーン全体に広げ，取引先と百貨店相互にメリットを得ることである。すなわち，製造段階・販売段階の業務を接続することによりサプライチェーン全体の最適化を目指していくことにある。具体的な施策としては，①マーチャ

ンダイズの最適化と②運用業務の効率化が検討できる。

①マーチャンダイズの最適化は,「欠品による販売機会損失の削減」「百貨店顧客情報から得られる顧客ニーズによる実需の創造」「過剰在庫による値下げの抑制と不良在庫の低減」「返品による物流コストの削減」を図ることができる。

一方, ②運用業務の効率化は,「伝票処理の一元化」「伝票削減・検品削減による物流業務の効率化」「値札廃止による値札発行・取付作業の削減」「商品マスターの登録効率化」を図ることができる。SCM を普及させ, サプライチェーン全体の最適化を図るためには, 百貨店業界標準の策定による投資の抑制や業界全体のシステム化水準の底上げ, 並びにサプライチェーン全体の業務効率化を図ることが喫緊の課題であることを認識することができた。

２００６年版百貨店 IT 白書による調査結果では,「表 3.1」に示すとおり, SCM を進めるうえで重要な百貨店におけるＥＤＩでの取引規模は, 全体平均で２０％前後の取引規模に留まっており, 現時点(2017 年)においても大きな変化はなく, 普及道半ばの状況である。

表 3.1　ＥＤＩ実施状況[4]

商品群	平均
衣料品	20.9%
雑貨	21.7%
リビング	19.0%
食料品	22.9%

現在の百貨店業界を取り巻く環境としては, 以下の阻害要因が挙げられる。

①前述したとおり百貨店業界には, 複数の EDI 標準が存在し, それに伴う複数の運用が存在している。そのため, 取引先が新たに取引する際に導入コストがかかり, 阻害要因なっている。一方, GMS 業界では平成１５年度から流通 BMS の規格に統一し, EDI の標準化を実施しており, 合わせて WEB-EDI の整備がなされ, どの取引先でも利用できる体制が構築されている。

②従来の EDI は買取仕入型の取引を前提として組み立てられているため, 消化仕入型の取引に十分に対応できていない。

③特に中小規模の地方百貨店や中小規模の取引先において, 単品管理・商品マスターの装備の遅れ, システム投資の割に効果が小さいなど, 投資対効果が不鮮明なため取組みが遅れている。

よって, このような背景のもとにインターネットやクラウドシステムなどの技術革新を利用した経済性・利便性の高い EDI の標準を策定することが求められており, 百貨店業界として SCM の鍵を握る EDI を「流通ビジネスメッセージ

標準（流通 BMS)」に一本化する必要があると考える。この考えに応えるものとして百貨店業界流通システム標準化委員会が平成１８年に経済産業省委託事業としてスタートした。

しかしながら現時点では,「流通 BMS」普及は進んでなく, 既存の IQRS や eMP, など複数の方式が並行稼働している状況にあり,「流通 BMS」への一本化は未だできていないのが実情である。

3.2 買取型ビジネスプロセスと取引先との情報共有化

買取型ビジネスプロセスとは, 百貨店が取引先に発注した商品を百貨店が仕入計上する取引形態, すなわち買取仕入(本仕入)のことを示す。「図 3.2」に, 買取型の従来型のビジネスプロセスプロセスを示す。

商品企画段階で取引先の展示会に百貨店バイヤーが出向き, 取引先へ商品の仮発注を行う。単品管理は実施しておらず, ダラー管理レベルでの仮発注を基に取引先は商品を生産する。商品できあがり時点で百貨店が連絡を受け, 百貨店が発注伝票を起票し発注する。発注した商品に対して, 取引先（卸・メーカー)が商品を手配し, 百貨店の値札を発行･取り付けし, 出荷する。百貨店は, 発注伝票と仕入伝票, 値札, 商品を相対検品して, 仕入計上を行う。仕入計上を行った時点で商品所有権は, 百貨店に移動する。それにより, 百貨店在庫商品となり, 管理責任は百貨店が負うこととなる。一方, 返品は百貨店が仕入・販売し, 売れ残った商品について, 一定の条件のもとで返品が認められる取引形態であり, 仕入と同様に買掛金に計上し, 月締め時点で仕入･返品を精算して, 取引先へ買掛金を支払う。このようなビジネスプロセスが, 伝票ベース・電話・口頭ベースで運用されていた。この一連のプロセスが買取型ビジネスプロセスの具体的な内容である。

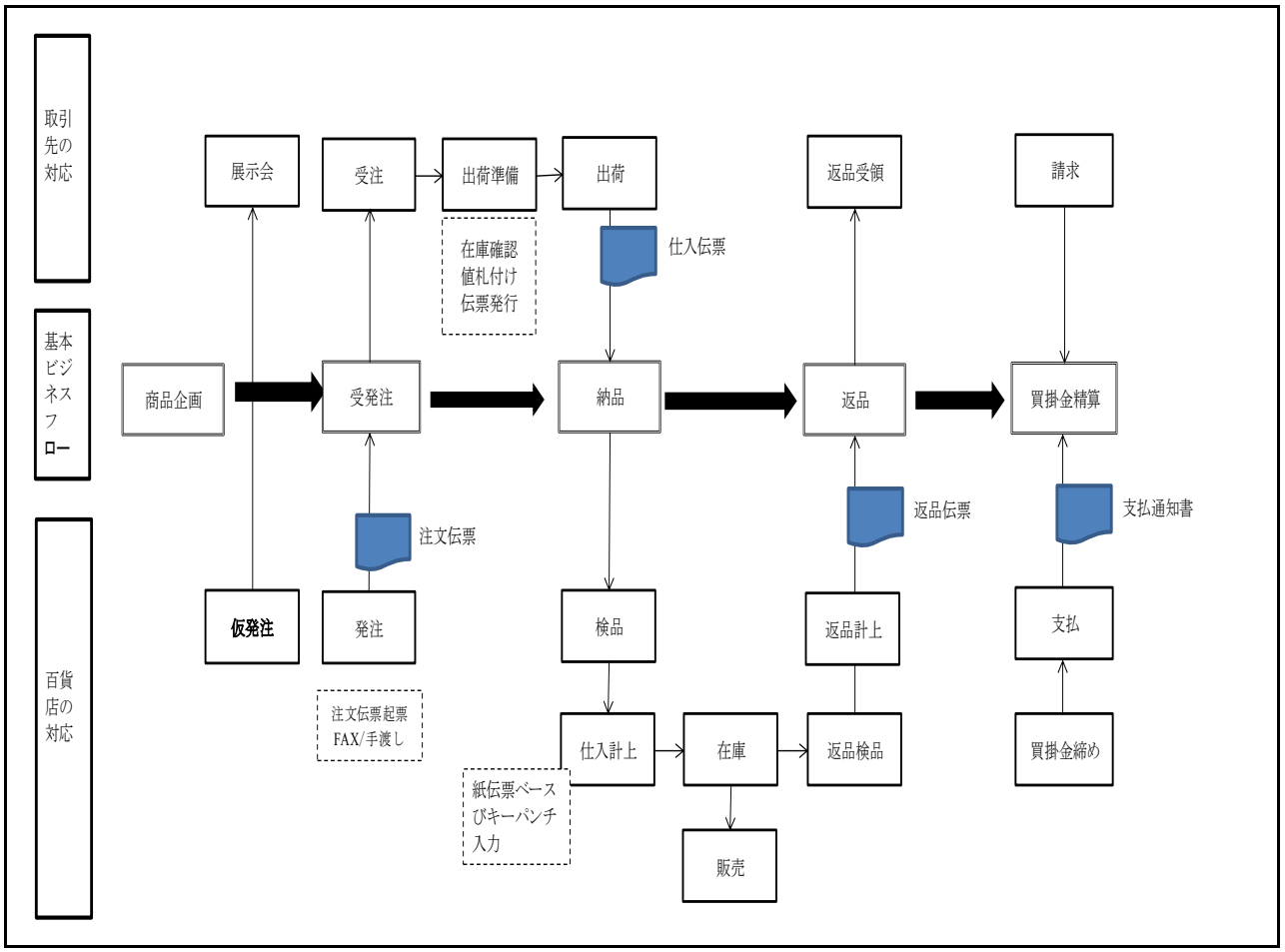

図 3.2　従来型買取型ビジネスプロセス

この運用をベースに基に，SCM への具体的な取組みとして，取引先との情報共有化を基礎として，「図 3.2」に示すとおり「標準 EDI 買取型ビジネスプロセス」を構築する。

このビジネスプロセスでは，４つの情報交換・情報共有化から構成される。以下に 4 つの情報交換・情報共有化を示す。

①商品マスターの取引先からの配信
②納品提案から納品・仕入計上までのシステム化
③売上情報・在庫情報の共有化
④買掛金支払案内の電子化

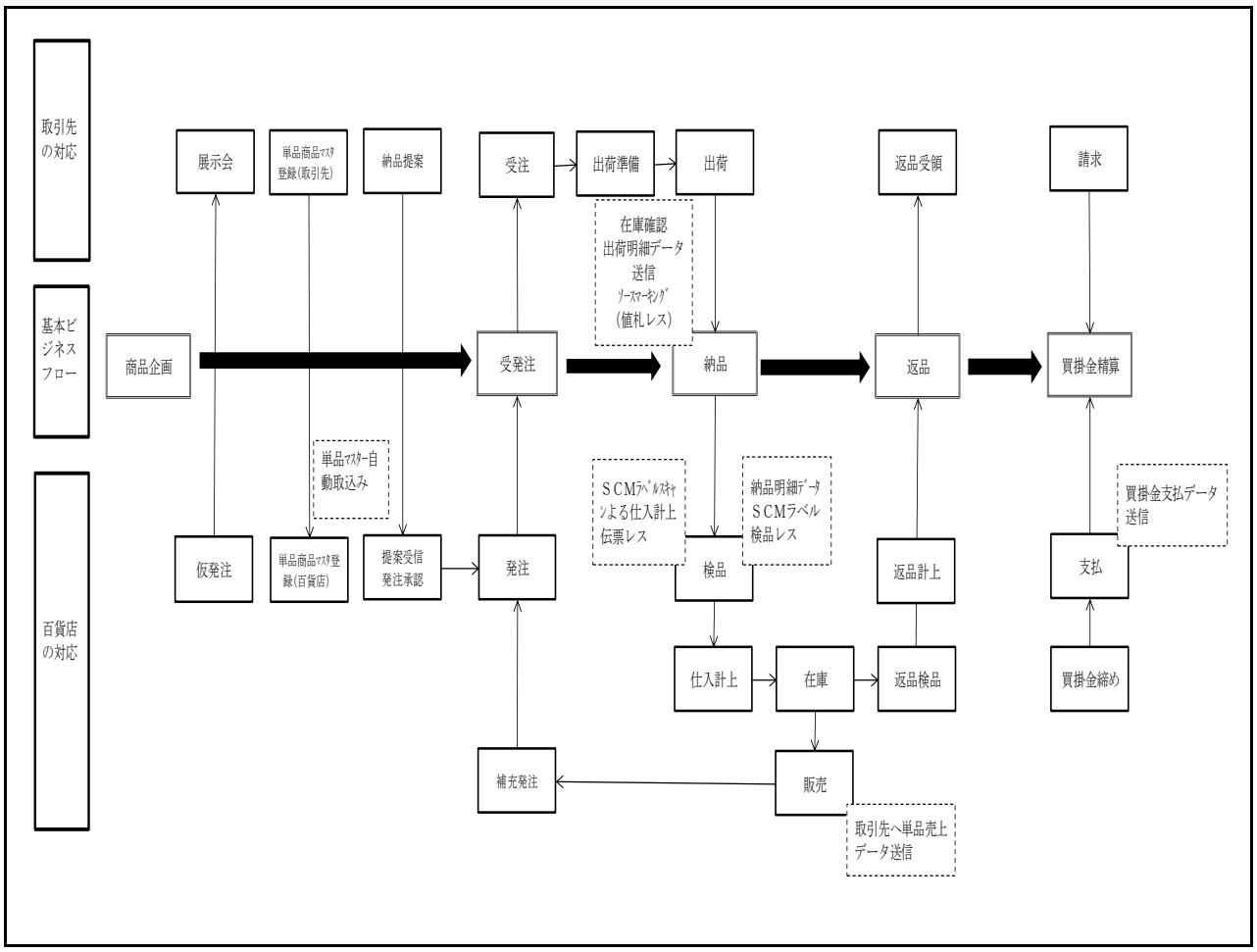

図 3.3　標準 EDI 買取型ビジネスプロセス

従来百貨店は単品管理をほとんどの商品で行っていない。

一部の商品で単品管理を行っているが，商品マスター管理は取引先と情報共有化されていないため，取引先から商品マスターをエクセルや登録票を受領し，単品マスター管理システムへ手入力で入力・登録作業を行っている。

一方，標準 EDI 買取型ビジネスプロセスでは，商品マスターの取引先からの配信は，取引先から単品商品コードを自動配信し，百貨店単品マスター管理システムへ自動登録する仕組みであり，単品管理・ＰＬＵを実施するうえで一番のネックとなる商品マスターコードの登録の作業を大幅に軽減する効果が期待できる。

商品仕入管理は，従来，展示会の仮発注後，取引先から商品が出来上がりの連絡を受け，売場が発注伝票を起票，取引先に送付し，その発注伝票をもとに取引先が商品を準備し，値札を発行・取り付け，仕入伝票の起票を行い，梱包のうえ発送する。百貨店は，納品された商品を発注伝票・仕入伝票・値札・商品を検品する。検品で間違いがないことを確認後，検品印を伝票に捺印する。検

品後の仕入伝票は，整理しバッチ入力票を添付して，仕入伝票の内容をパンチ入力する。入力後，エラーがある場合は伝票内容を修正し，再入力をする。入力後，商品勘定システムに登録され，在庫計上されるとともに，取引先への買掛金に計上される。

一方，標準 EDI 買取型ビジネスプロセスでは，取引先から商品が出来上がり・納品の連絡を，取引先から納品提案で百貨店へ送信される。納品提案承認後，取引先へ発注として送信される。取引先は発注データをもとに取引先が商品を出荷検品し，梱包，SCM ラベル貼付のうえ発送する。並行して出荷データを百貨店へ送信する。値札は商品にソースマーキングされているため不要である。百貨店は，SCM ラベルをスキャンすることで，簡易検品とし，受領データとして取引先に送信する。並行して，仕入計上し，在庫・買掛金データに反映する。

この情報共有化より，注文伝票起票・送付廃止，仕入伝票削減化，検品削減化，伝票パンチ入力削減化が実現可能となる。

標準 EDI 買取型ビジネスプロセスでの売上情報・在庫情報の共有化は，百貨店が POS を通して得ている単品売上データ，在庫データを取引先と共有することであり，取引先は日々の商品売上情報・在庫情報を得ることが可能となり，マーチャンダイズの最適化を百貨店・取引先双方で実現できる。

標準 EDI 買取型ビジネスプロセスでの買掛金支払案内の電子化により，従来の月締め処理後，支払通知書を印刷し，取引先に郵送するとともに，支払日に買掛金を支払う一連の作業の自動化が実現可能となる。

以上述べてきたとおり標準 EDI 買取型ビジネスプロセスでの情報共有化により，①マーチャンダイズの最適化と②運用業務の効率化が実現可能となり，ひいては上流工程の効率化にもつながり，サプライチェーン全体の効率化に大いに貢献することができる。

平成 21 年に百貨店業界流通システム化標準化委員会で買取型 EDI 業務モデルが構築された。このモデルは，「図 3.4」に「買取型 EDI 業務モデル」を示すとおりである。

「百貨店業界流通システム化標準化委員会」買取型 EDI 業務モデルは，百貨店の業務プロセスに沿ったモデルであり，WEB-EDI として先行して開発された「IQRS.Net」や「百貨店 eMP サービス」の機能を踏襲した内容ではある。

しかし，違いは，経済産業省が推進する「流通システム標準化事業」において，国として唯一の標準 EDI である「流通ビジネスメッセージ標準（流通 BMS）」を利用することである。すなわち，今まで様々な規格の EDI が存在し，各々の対応でなかなか EDI 普及しなかったが，国として標準化することにより，流通業に関わる製配販で共通に利用できる標準 EDI を整備したことである。それにより，取引先・百貨店は EDI を採用すれば，すべての取引を同じプロセスで完結することができるようになる。

それにより，サプライチェーンマネジマントを推進し，①マーチャンダイズの最適化と②運用業務の効率化を図り，各プロセスで発生する「むだ・むり・むら」を低減させ，効率化の向上図ることが可能となる。

「百貨店業界流通システム化標準化委員会」買取型 EDI 業務モデルに沿ったパッケージソフトウエアが富士通 FIP，日立システムなどシステムベンダーからリリースされつつあり，システムの開発コスト負担ができない中小事業者などは，パッケージソフトを利用することにより，リーズナブルにシステム利用をすることが可能となりつつある。

3.3 消化型ビジネスプロセスと取引先との情報共有化

消化型ビジネスプロセスとは，取引先の商品を百貨店が販売し，売上データに基づいて仕入・支払を行う取引形態，すなわち消化仕入（売上仕入）のことを示す。一般的には，売上発生時に初めて商品の所有権や管理責任が百貨店に移行する。

マーチャンダイズはすべて取引先主導で進められることとなる。また，取引先の販売員により商品管理・販売管理が行われ，いわば，場所貸し的な商売が展開されることとなる。

平成 21 年に百貨店業界流通システム化標準化委員会において消化型 EDI 業務モデルが構築した。これは，「図 3.5」である。

このモデルは，大きく３つの情報共有化から構成される。

①商品マスターの取引先からの配信

②売上情報の共有化

③買掛金支払案内の電子化

オプションとして，物流情報として，搬入･到着情報の標準化効率化，情報共有として在庫情報の共有化が設定されている。

商品マスター管理は，従来は取引先と情報共有化されていないため，取引先から商品マスターをエクセルや登録票を受領し，単品マスター管理システムへ手入力で入力・登録作業を行っている。一方，商品マスターの取引先からの配信は，取引先から単品商品コードを自動配信し，百貨店単品マスター管理システムへ自動登録する仕組みであり，単品管理・PLU を実施するうえで一番のネックとなる商品マスターコードの登録の作業を大幅に軽減する効果が期待できる。

売上情報の共有化は，百貨店が POS を通して得ている単品売上データを取引先と共有することであり，取引先は日々の商品売上情報・在庫情報を得ることが可能となり，マーチャンダイズの最適化を百貨店･取引先双方で実現できる。

買掛金支払案内の電子化により，従来の月締め処理後，支払通知書を印刷し，取引先に郵送するとともに，支払日に買掛金を支払う一連の作業が自動化が実現可能となる。

EDI 業務モデルの導入により百貨店･取引先のメリットをまとめると「表 3.2」のとおりである。これにより，百貨店，取引先ともメリットを得ることができ，SCM の全体最適化が実現可能となる。このメリットを百貨店，取引先ともに得られるようにオペレーションを確立することが重要なポイントとなる。

表 3.2　EDI 業務モデルで齎される百貨店・取引先のメリット

内容	百貨店メリット	取引先メリット
商品マスターの取引先からの配信	● 売場で商品マスターを登録する作業が解消 ● 百貨店 POS での PLU の利用	● 商品マスターの同一化 ● 百貨店 POS での PLU の利用
納品提案から納品・仕入計上までの標準化効率化	● 売場で発注伝票を起票解消 ● EOS もしくは納品提案への承認のみ ● 検品削減化・検品簡素化 ● 出荷情報受信・納品確定による，仕入伝票パンチ入力の解消 ● 値札廃止（ｿｰｽﾏｰｷﾝｸﾞの利用）	● 発注情報の電子化 ● 納品の迅速化 ● 値札廃止：値札発行・取付不要 ● 伝票削減化：伝票経費の削減 ● 納品検品削減化：立会検品費用の削減
売上情報・在庫情報の共有化	● 単品別売上・在庫情報を取引先と共有することにより，取引先から納品提案，VMI などが可能 ● 売上・在庫情報の共有化による販売機会損失への対応 ● 取引先との協業体制を構築し，双方の業務効率化の推進	● マーチャンダイズの最適化 ● 繊維産業全般を実需型の産業に再構築 ● 百貨店顧客情報から得られる顧客ニーズによる実需の創造 ● 過剰在庫による値下げの抑制と不良在庫の低減 ● 返品による物流コストの削減
買掛金支払案内の電子化	● 買掛金支払案内の電子化	● 出荷情報と支払案内の照合作業のシステム化：業務効率の向上

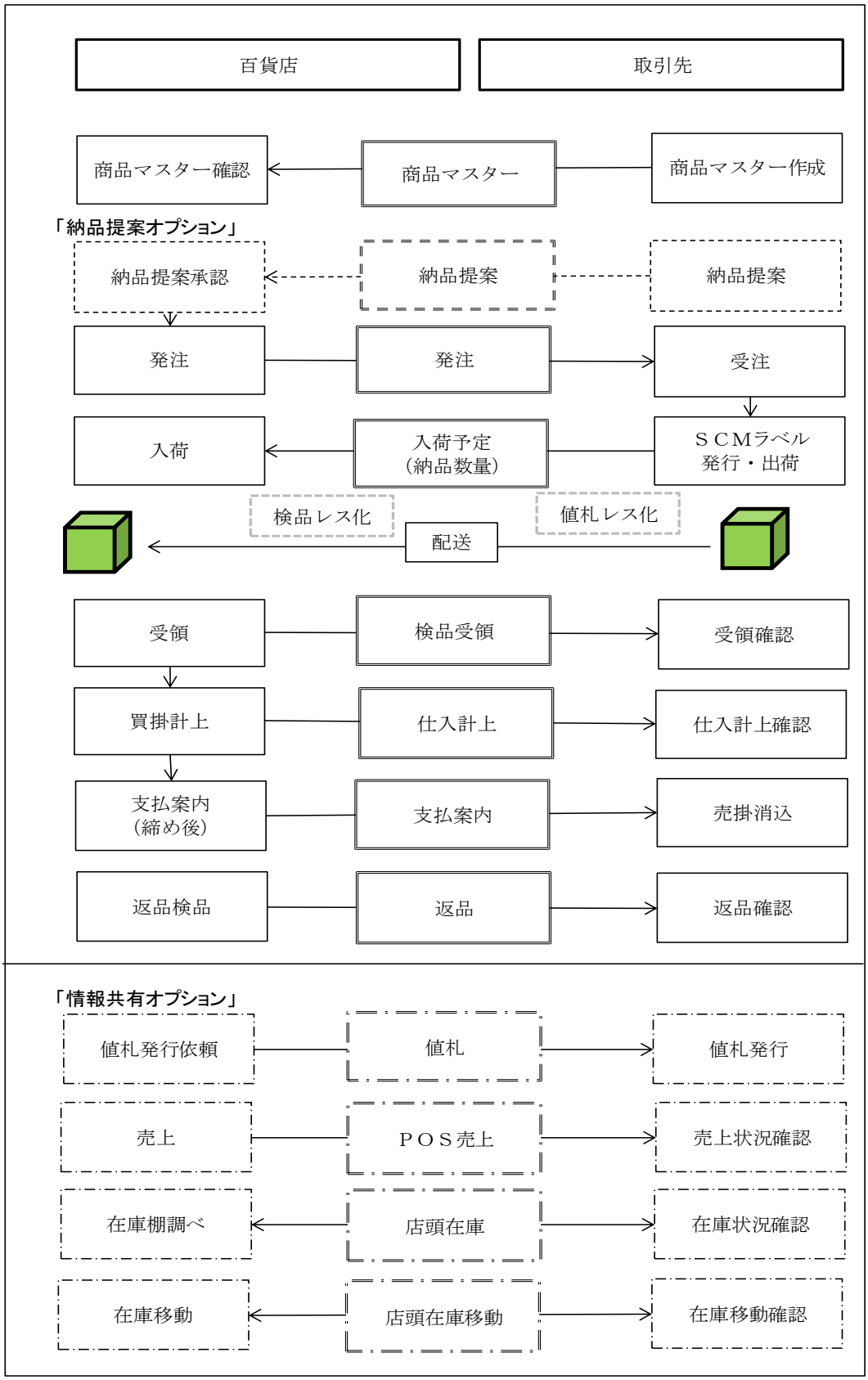

図 3.4　買取型 EDI 業務モデル

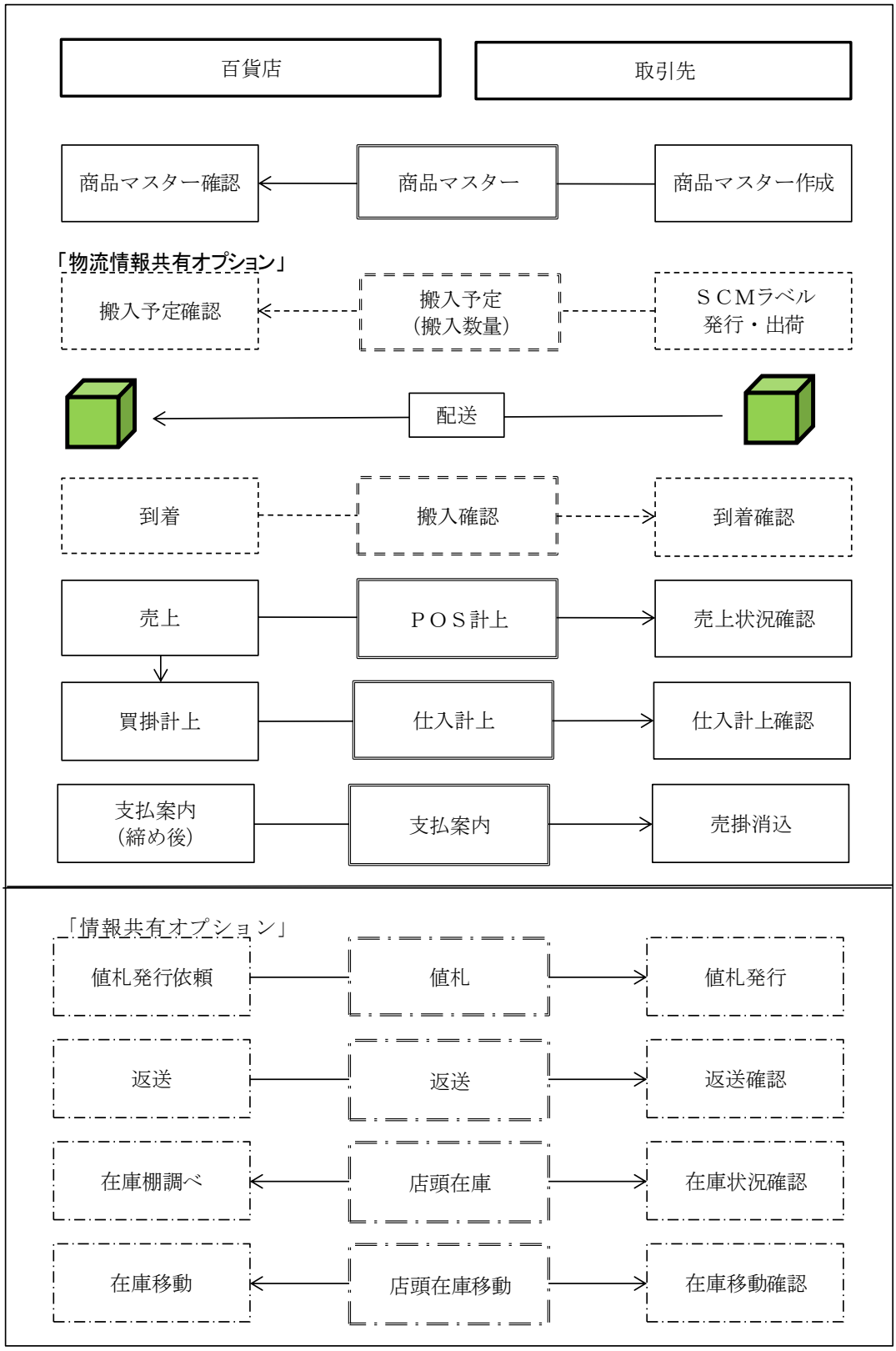

図 3.5　消化型ＥＤＩ業務モデル

4. 取引先との SCM の取組みにより経済効果の測定

4.1 本研究における SCM の経済効果測定範囲の明確化

SCM の具体的な施策としては，①マーチャンダイズの最適化と②運用業務の効率化である。ここで，SCM の取組みにより，百貨店業界でどれくらいの経済効果が得られるのかを検討する。

「マーチャンダイズの最適化」は，「欠品による販売機会損失の削減」「百貨店顧客情報から得られる顧客ニーズによる実需の創造」「過剰在庫による値下げの抑制と不良在庫の低減」「返品による物流コストの削減」の効果が得られると一般的に言われているが，いずれの項目を数値分析するためには，分析に必要な各種データが，公に公開されていない。例えば，単品管理によりどれくらい欠品を防止し，売上増の効果を生み出しているかを業界全体で測定した研究論文は皆無である。 実際に効果を測定するためには，環境与件が類似している実店舗 2 店舗以上を用意し，各対策の実施可否を比較し，その効果を測定するなど大掛かりな実験が必要である。今回の研究では，費用面，体制面などを鑑み，このような実験を実施することは難しい。よって，「マーチャンダイズの最適化」については，今回の経済効果の測定範囲から外すこととする。

一方，「運用業務効率化」は，「商品マスターの登録効率化」「伝票削減・検品削減による物流業務の効率化」「値札廃止による値札発行・取付作業の削減」「伝票処理の一元化」などが挙げられる。これらの項目については，公開されている経済産業省の実証実験データや統計資料などから効果を類推することが可能なため，今回の研究では，「運用業務の効率化」を対象として捉え，百貨店の標準ビジネスプロセスから導き出された「買取型 EDI モデル」ならびに「消化型 EDI モデル」を利用して，運用効率化の各項目の分析を行い，百貨店における経済効果を測定するものとする。「図 3.6」に対象範囲を示す。

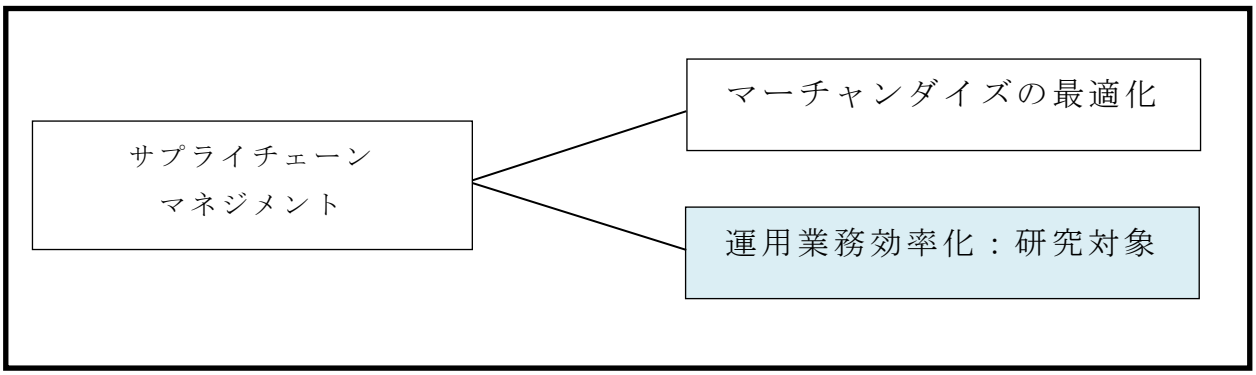

図 3.6　SCM 施策の本研究の対象

4.2 EDI モデル導入による業務プロセスの変化

下記に「従来型業務プロセス」,「ＥＤＩ導入業務プロセス」を「図 3.7」,「図 3.8」に示す。「図 3.8」に示すとおり, EDI モデルを導入により, 取引先と百貨店との情報共有化が実現しすることにより,「商品マスター登録自動化」,「注文伝票削減化」,「検品削減化」,「仕入伝票削減化」,「支払案内電子化」の業務効率化が実現され, 業務プロセスの違いを理解できることができる。

なお, 消化仕入については, 売上計上時に自動的に仕入が計上する仕組みなので,「注文伝票」,「検品」,「仕入伝票」はもともとの業務プロセスに存在しないため, 業務プロセスの変化はない。

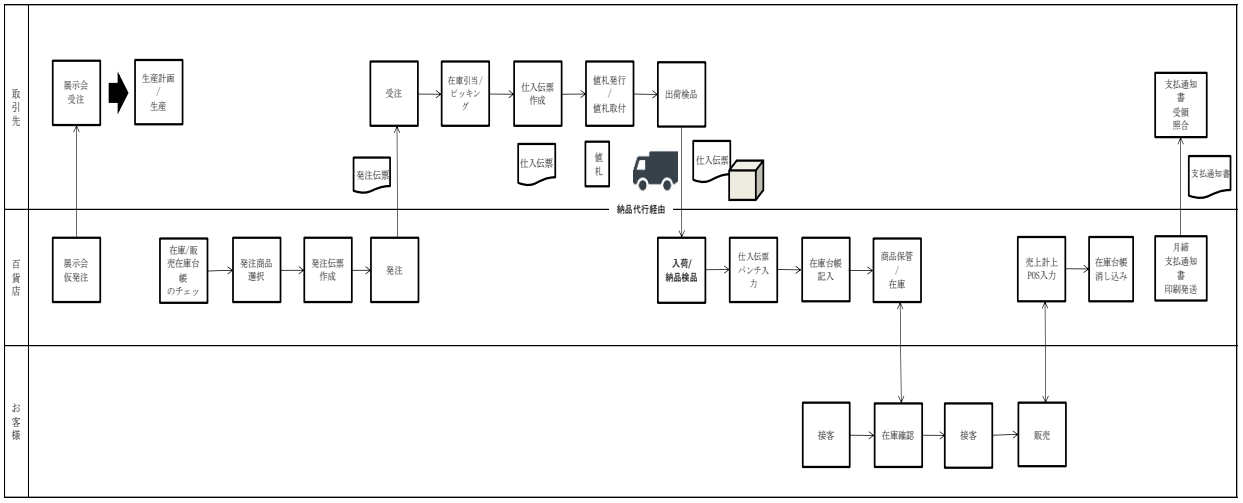

図 3.7　従来型業務プロセス

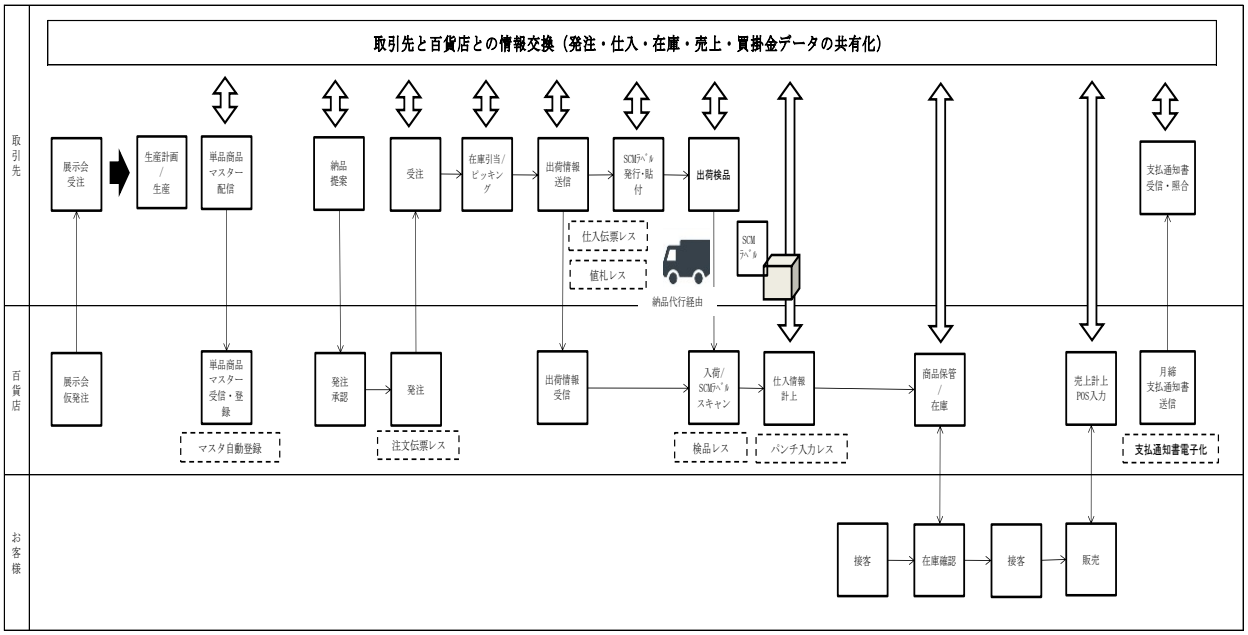

図 3.8　ＥＤＩ導入業務プロセス

4.3 EDI モデル導入による経済効果

「図 3.4」および,「図 3.8」に示すとおり,「買取型 EDI モデル」では,取引先との EDI を利用して,「商品マスターデータのやり取り」,「納品提案～発注～商品出荷～受領～仕入計上のやり取り」,「支払案内のやり取り」がシステム化され,「①商品マスター登録自動化」,「②注文・仕入伝票削減化」,「③検品削減化」,「④百貨店値札廃止化」,「⑤支払案内電子化」が実現可能となり業務が効率化される。

一方,「消化型 EDI モデル」では,取引先との EDI を利用して,「商品マスターデータのやり取り」,「支払案内のやり取り」がシステム化され,「商品マスター登録自動化」,「支払案内電子化」が実現可能となり,業務効率が図られることとなる。

これらの従来運用で発生しているコスト内容を分析,試算することにより,具体的なコストを算出する。このコストが,SCM の情報共有化導入によってもたらされる経済効果と考える。モデルごとに業務効率が得られるかをまとめると,下記「表 3.3」のとおりとなる。

表 3.3　モデル別業務効率化可否一覧

項目	買取型 EDI	消化型 EDI
①商品マスター登録自動化	△	△
②注文・仕入伝票削減化	○	×
③検品削減化	○	×
④百貨店値札廃止化	○	×
⑤支払案内電子化	○	○

備考：「○」：効果あり,「×」：効果なし,「△」：効果不明

次に,本研究で対象である百貨店業界の売上高など基礎数値は,日本百貨店協会が公開している 2015 年の数値を採用した。日本全国百貨店の商品別売上高は「表 3.4」に示すとおりである。2015 年は,年間売上高は 6 兆 1743 億円である。年間売上高のピークは 1991 年の約 9 兆 7130 億円であり,ピーク時の 6 割までに低下している。今回の数値分析は,この 2015 年売上高の数値を基礎数値として分析を進める。

表 3.4　2015 年百貨店商品別売上高一覧[26]

商　品　別	売上高（千円）	構成比（%）	対前年増減率（%）	
総　　額	**6,174,278,636**	**100.0**	**△0.2**	**(△0.6)**
紳士服・洋品	423,987,947	6.9	△2.1	(△2.4)
婦人服・洋品	1,298,079,816	21.0	△3.9	(△4.3)
子供服・洋品	150,767,929	2.4	△0.2	(△0.4)
その他衣料品	144,256,432	2.3	△5.2	(△5.7)
衣　料　品	**2,017,092,124**	**32.7**	**△3.4**	**(△3.7)**
身のまわり品	**811,230,186**	**13.1**	**0.7**	**(0.4)**
化 粧 品	401,550,558	6.5	12.5	(12.2)
美術・宝飾・貴金属	358,825,178	5.8	8.0	(7.7)
その他雑貨	239,794,268	3.9	△0.3	(△1.2)
雑　　貨	**1,000,170,004**	**16.2**	**7.6**	**(7.1)**
家　　具	72,588,615	1.2	△7.2	(△7.6)
家　　電	17,580,680	0.3	△5.0	(△5.3)
その他家庭用品	196,654,157	3.2	△1.5	(△1.9)
家 庭 用 品	**286,823,452**	**4.6**	**△3.3**	**(△3.6)**
生鮮食品	344,190,134	5.6	△2.7	(△3.3)
菓　　子	466,636,817	7.6	0.4	(0.1)
惣　　菜	371,200,828	6.0	0.7	(0.3)
その他食料品	519,660,424	8.4	△0.1	(△0.7)
食　料　品	**1,701,688,203**	**27.6**	**△0.3**	**(△0.8)**
食 堂 喫 茶	**172,133,905**	**2.8**	**△0.8**	**(△1.0)**
サ ー ビ ス	**65,877,854**	**1.1**	**△3.6**	**(△3.8)**
そ　の　他	**119,262,908**	**1.9**	**△0.1**	**(△0.7)**

4.4 商品マスター登録自動化による効果

取引先との情報共有化を推進するための EDI モデルの採用には，百貨店と取引先で共通に利用する商品マスターの共有化が前提条件となる。具体的には，商品についているメーカータグやソースマーキングなどの共通単品商品コード，JAN コードなどを利用して，発注・仕入・在庫・売上・返品・棚卸を行うことによって単品での商品動向を把握する。すなわち，共通単品商品コードを利用して，単品レベルの商品管理を行うことである。

しかし，従来百貨店は単品管理を一部の商品のみしか実施しておらず，大半の商品は自社商品コード体系に基づく，品番管理である。品番管理とは，どこの売場（品番）で，どんな商品（商品分類・クラス）をどこの取引先のどんな条件（取引条件・ブランド）を自社コードに置き換えて管理することであり，すなわち商品の群管理である。この自社商品コード単位に商品勘定を計算し，売上・利益を確定している。一方，取引先で単品での売り上げ動向を把握したい会社は，販売した商品のタグコードを派遣社員が閉店後に，取引先端末に入力や手書きで報告するなどしている。

単品管理を行うためには，共通単品商品コードのマスターを百貨店，取引先で登録する必要があり，それが百貨店，取引先と同期する必要がある。取引先で登録された共通単品商品コードは EDI を介してデータを送信し，百貨店の単品商品マスターに同期することとなる。

そこで，商品マスター登録自動化による効果であるが，大半の百貨店は前述のとおり単品管理を実施していないため，共通単品商品コードを商品マスターとして管理するコストは発生していない。したがって，商品マスター登録自動化によってもたらされる効果はないということがいえる。もしくは，一部売場で部分的に実施している PLU 機能や EOS 機能を利用している部分的な売場のみ手作業で行っている作業が，システム化され自動化される。

ただし，取引先との情報共有化を推進するためには，単品管理が前提条件であり，この機能がなければ成立しない。また，百貨店の取り扱い単品数は，数百万単品とも言われており，この機能の効果は絶大であるが従来業務になく，新たに発生する業務であるため，今回の効果測定では除外するものとする。

4.5 注文伝票・仕入伝票削減化による効果測定

4.5.1 効果測定式

EDI モデルを採用することによって得られる「注文伝票・仕入伝票削減化」による効果測定を以下に検討する。

SCM により，取引先との情報交換により，注文伝票・仕入伝票削減化が実現できる。「図 3.3」および「図 3.4」に示すとおり，注文伝票，仕入伝票は買取仕入の場合に発生する。伝票削減は，百貨店と取引先との間で伝票ベースの仕入から，EDI による情報交換により仕入データを電子的に計上することである。伝票削減により，百貨店側は発注伝票の起票が解消されるとともに，伝票計上のパンチ入力作業がなくなり，それらのコストが削減されるとともに，取引先側は伝票発行コスト，支払通知書の照合作業が削減されることが期待できる。伝票削減により期待されるコスト削減効果を以下に解析し試算する。

百貨店の伝票削減化によるコスト削減効果は，仕入伝票を商品勘定システムに入力するパンチ入力費用と注文伝票を作成する費用である。なお，注文伝票は，仕入伝票と相対して作成されるため，計算上仕入伝票と同数の発注伝票が発生すると仮定する。

次に，注文伝票・仕入伝票削減化算出式を構築し，効果を算出する。

①買取仕入売上金額は，総売上金額から買取仕入比率を乗じて算出する。

$$SVps = SV \times Rps \qquad \cdots\cdots\cdots (3.1)$$

②買取仕入仕入数量は，買取仕入売上金額から平均単価を除して算出する。

$$SQps = \frac{SVps}{Pav} \qquad \cdots\cdots\cdots(3.2)$$

③仕入れた商品は，すべて販売されるわけではなく，売れ残った商品は取引先に返品されるため，商品を売り切る率を消化率として定義し，買取仕入売上数量に消化率を除して買取仕入仕入数量を算出する。

$$PQps = \frac{SQps}{Dig} \qquad \cdots\cdots\cdots(3.3)$$

④ ③で求めた買取仕入仕入数量を伝票 1 枚当たりの商品数を除して，取扱仕入伝票枚数を算出する。

$$SPQps = \frac{PQPS}{MQper} \qquad \cdots\cdots\cdots(3.4)$$

⑤ ④で求めた取扱仕入伝票枚数にパンチ入力の伝票入力単価を乗することにより，パンチ入力コストを算出する。百貨店商慣習では，伝票用紙費用は取引先の負担なため除外する。

$$Pec = SPQps \times Peper \qquad \cdots\cdots\cdots(3.5)$$

⑥ 注文伝票コストは，注文伝票と仕入伝票は相対しているため，同数と仮定する。取扱注文伝票枚数に注文伝票単価を乗じて注文伝票コストを算出する。

$$OSc = OSQps \times OSper \qquad \cdots\cdots\cdots(3.6)$$

⑦ ⑤で求めたパンチ入力コスト，⑤で求めた注文伝票コストを和して伝票削減化により得られる効果を算出する。

$$SLc = Pec + OSc \qquad \cdots\cdots\cdots(3.7)$$

使用記号

SVps：買取仕入売上金額（円）
SV：総売上金額（円）
Rps：買取仕入比率（%）
SQps：買取仕入売上数量（個）
Pav：平均単価（円）
PQps：買取仕入仕入数量（個）
Dig：消化率（%）
SPQps：取扱仕入伝票枚数（枚）
Mqper：伝票 1 枚あたりの商品数（個）
Pec：パンチ入力コスト（円）
Peper：伝票パンチ入力単価（円）
OSc：注文伝票コスト（円）
OSQps：取扱注文伝票枚数（枚）
OSper：注文伝票単価（円）
SLc：伝票削減効果（円）

以上の計算式で伝票削減化の効果を算出する。

4.5.2 集計結果

まずは，日本百貨店協会の 2015 年売上金額から商品群別買取仕入の売上金額を算出する。2006 年版百貨店 IT 白書[4]による調査結果では，全体の取引形態で買取仕入：消化仕入の比率は 50%：50%であり，衣料品は 42%：58%，雑貨は 52%：48%，リビングは，60%：40%，食品は 42%：58%であった。食堂，サービス，その他は，ほとんどが消化仕入もしくは，賃貸テナントのため，0%：100%として算出するものとする。

次に商品群別平均単価，消化率，伝票パンチ入力単価，伝票単価，伝票 1 枚あたりの商品数の設定する。

商品群別の平均単価および消化率，伝票パンチ入力単価，伝票単価，伝票 1 枚あたりの商品数は，代表的な百貨店数社へのヒアリング調査を実施し，得られて数値から平均単価を算出し，使用する。ただし，回答の匿名が条件であるため，ヒアリングした百貨店企業名は伏せることとする。商品群別に平均単価をまとめた表が「表 3.5」である。

表 3.5　商品群別平均単価一覧表　　（金額：円）

商品群	平均単価	消化率	伝票パンチ入力単価	注文伝票単価	伝票１枚あたりの商品数
衣料品	10,000	90%	72	10	9
身の回り品	5,000	80%	72	10	17
雑貨	1,100	90%	72	10	20
家庭用品	1,500	80%	72	10	20
食料品	1,100	100%	72	10	61

調査によって求めた各係数を 2015 年百貨店売上高の売上金額に基づき，効果を算出した。その結果は「表 3.6」のとおりである。

算出の結果，パンチ入力コストが 4,374 百万円，注文伝票コストが 608 百万円と算出され，合計で 4,982 百万円のコストが注文伝票・仕入伝票削減化により削減されることが算出された。

表 3.6　注文伝票・仕入伝票削減化による効果一覧表

商品群	売上金額（千円）	買取仕入比率	買取仕入売上金額（千円）	商品平均単価（円）	買取仕入売上数量	消化率	買取仕入仕入数量
記号	*SV*	*Rps*	*SVps*	*Pav*	*SQps*	*Dig*	*PQps*
衣　料　品	2,017,092,124	42%	847,178,692	10,000	84,717,869	90%	94,130,966
身のまわり品	811,230,186	52%	421,839,697	5,000	84,367,939	80%	105,459,924
雑　　貨	1,000,170,004	52%	520,088,402	1,100	472,807,638	90%	525,341,820
家 庭 用 品	286,823,452	60%	172,094,071	1,500	114,729,381	80%	143,411,726
食　料　品	1,701,688,203	42%	714,709,045	1,100	649,735,496	100%	649,735,496
食 堂 喫 茶	172,133,905	0%	0				
サ ー ビ ス	65,877,854	0%	0				
そ　の　他	119,262,908	0%	0				
合計	6,174,278,636	-	2,675,909,907	-	1,406,358,323	-	1,518,079,932

商品群	伝票１枚当りの商品数	取扱仕入伝票枚数	パンチ入力単価	パンチ入力コスト（円）	注文伝票単価（円）	注文伝票コスト（円）
記号	*Mqper*	*SPQps*	*PEper*	*PEc*	*OSper*	*OSc*
衣　料　品	9	10,458,996	72	753,047,712	10	104,589,960
身のまわり品	17	6,203,525	72	446,653,800	10	62,035,250
雑　　貨	20	26,267,091	72	1,891,230,552	10	262,670,910
家 庭 用 品	20	7,170,586	72	516,282,192	10	71,705,860
食　料　品	61	10,651,402	72	766,900,944	10	106,514,020
食 堂 喫 茶						
サ ー ビ ス						
そ　の　他						
合計	-	60,751,600	-	4,374,115,200	-	607,516,000

4.6 検品削減化による効果測定

4.6.1 効果測定式

EDI モデルを採用することによって得られる「検品削減化」による効果測定を以下に検討する。

SCM による取引先との情報交換により，取引先から送信される出荷データ，および出荷時に SCM ラベルを商品梱包に添付することにより，百貨店入荷時に SCM ラベルをスキャンすることにより，受領データを取引先に送信する。それにより，検品削減化を実現する。検品削減化により，立ち合い検品を無くすことができ，取引先の立ち合いコストが削減できるとともに，百貨店の受領検品コストを無くすことが可能となる。

百貨店の検品削減化によるコスト削減効果は，納品時における，商品・値札・注文伝票・仕入伝票との照合作業を無くすことある。ここでは，検品に関わるコストを分析し，コスト計算を実施する。

検品のコストは，買取仕入商品の値札を 1 点ずつ仕入伝票と取引先と照合する作業である。コスト内容を分析すると買取仕入商品数量に商品 1 点あたりの検品作業所要時間を乗じ，作業所要時間を割り出し，それに人件費を乗じれば，検品コストを算出することできる。

① 検品所要時間は，買取仕入仕入数量に商品 1 点あたりの作業時間を乗じることにより算出する。

$$Iv = PQps \times Iper \qquad \cdots\cdots\cdots(3.8)$$

② 検品作業コストは，検品所要時間に人件費単価を乗じることにより算出する。

$$PI = Iv \times Pper \qquad \cdots\cdots\cdots(3.9)$$

③ 検品削減の効果は，検品作業コストである。

$$ILC = PI \qquad \cdots\cdots\cdots(3.10)$$

使用記号

Iv：検品所要時間(時間)

$PQps$：買取仕入仕入数量(個)

$Iper$：商品 1 点あたりの検品時間(時間)

PI：検品作業コスト(円)

Iv：検品所要時間（時間）
Pper：人件費単価（円）
ILC：検品削減化効果（円）
PI：検品作業コスト(円)

以上，計算式で検品削減化の効果を算出する。

4.6.2 集計結果

商品 1 点あたりの検品作業時間，人件費は，百貨店・アパレル電子タグ推進委員会によりまとめられた「百貨店業界・アパレル業における電子タグ実証実験報告書　第８章　経済効果推計」[12]の調査結果数値を利用して算出する。なお，衣料品においては，パッキン･ハンガーの納品形態があり，その比率はハンガー：パッキン＝35%：65%として算出する。詳細は，「表 3.7」に示すとおりである。また，人件費は契約社員を仮定し，時給単価を 1,500 円とし，福利厚生費など付帯人件費を人件費の 25%として付与して，1,875 円を人件費時給単価とした。

表 3.7　衣料品ハンガー・パッキン割合表

商品群	荷姿	比率	仕入数量
衣料品	パッキン	35%	32,945,838
	ハンガー	65%	61,185,128
衣料品合計	-	100%	94,130,966

調査によって求めた各係数を 2015 年百貨店売上高の売上金額に基づき，効果を算出した。その結果は「表 3.8」のとおりである。算出の結果，検品削減化の効果は，8,362 百万円と算出された。

表 3.8　検品削減化による効果一覧表

商品群	荷姿	買取仕入 仕入数量	1点あたり 検品時間 (秒)	検品時間 (時間)	人件費 単価(円)	検品作業 コスト（円）
記号		*PQps*	*Iper*	*Iv*	*Pper*	*PI*
衣　料　品	パッキン	32,945,838	5.43	99,386	1,875	186,349,323
衣　料　品	ハンガー	61,185,128	1.92	65,264	1,875	122,370,128
身のまわり品	パッキン	105,459,924	5.43	318,138	1,875	596,508,222
雑　　貨	パッキン	525,341,820	5.43	1,584,782	1,875	2,971,465,460
家 庭 用 品	パッキン	143,411,726	5.43	432,626	1,875	811,173,163
食　料　品	パッキン	649,735,496	5.43	1,960,036	1,875	3,675,066,949
合計		1,518,079,932	-	4,460,231	-	8,362,933,245

4.7 百貨店値札廃止による効果測定

4.7.1 効果測定式

百貨店における値札は，お客様に提供する価格を表示するともに，商品コードを表示し売上・利益・在庫といった商品勘定を算出するうえで重要な役割を有している。値札廃止化とは，取引先で付与する商品単品コード，すなわちソースマーキング，JAN コードで代替し，百貨店独自の値札を廃することである。

従来値札は，各百貨店仕様値札を取引先が作成し，商品に取り付けして納品するのが商慣習であり，値札の台紙費用を百貨店が負担している。また，消化仕入においては，百貨店値札は省略されており，対象は買取仕入商品の値札となる。

値札廃止化により，百貨店は値札台紙コストが無くなり，取引先は値札作成・商品取り付け作業が無くなる。

値札コストは，コスト内容を分析すると買取仕入商品数量に値札台紙単価を乗じて，算出する。なお，値札台紙単価は，代表的な百貨店へのヒアリング調査を実施し，得られて数値から平均単価を算出し，使用する。また，商品群により使用するもっとも代表的な値札サイズの単価を使用する。

値札コストは，買取仕入仕入数量に値札台紙単価を乗じることにより算出する。

$$TLc = PQps \times Tper \qquad \cdots\cdots\cdots (3.11)$$

使用記号

TLc：値札コスト(円)

PQps：買取仕入仕入数量（個）

Tper：値札台紙単価（円）

以上，計算式で値札廃止化の効果を算出する。

4.7.2 集計結果

調査によって求めた各係数を 2015 年百貨店売上高の売上金額に基づき，効果を算出した。その結果は「表 3.9」のとおりである。

算出の結果，値札廃止化の効果は，637 百万円と算出された。

表 3.9　値札廃止化による効果一覧表

（金額：円）

商品群	買取仕入 仕入数量	値札単価	値札コスト	備考
記号	*PQps*	*Tper*	*TLc*	
衣　料　品	94,130,966	0.6	56,478,580	主に 1 号値札使用
身の回り品	105,459,924	0.5	52,729,962	主に 2 号値札使用
雑　　貨	525,341,820	0.4	210,136,728	主に 7 号値札使用
家 庭 用 品	143,411,726	0.4	57,364,690	主に 7 号値札使用
食　料　品	649,735,496	0.4	259,894,198	主に 7 号値札使用
合計	1,518,079,932	－	636,604,158	

4.8 支払案内電子化による効果測定

4.8.1 効果測定式

支払案内電子化は，百貨店が，従来月締め後に各取引先に当月の支払案内通知書を印刷し，郵送により発送している。取引先は，送られてきた支払案内通知書と取引先の売掛金とを照合している。

EDI モデルを採用することにより，支払案内通知書が標準フォーマットにより送信されることにより，照合作業が IT 化され，業務効率化が図れる。支払案

内電子化の効果は，支払案内通知書印刷・封入作業の解消と郵送費の削減である。この２つについて作業内容を分析し，コストを算出する。

① ひと月の１店舗あたり支払案内通知書印刷コストは，支払案内通知書印刷単価に取引先数を乗じて算出する。

$$PC1 = Vq \times PPper \qquad \cdots\cdots\cdots(3.12)$$

② 封入作業時間は，印刷物を取引先別に仕分けし，封筒に封入する作業であり，取引先数に１社あたり封入作業時間を乗じて封入作業時間を算出する。

$$OC1 = Vq \times OCv1 \qquad \cdots\cdots\cdots(3.13)$$

③ ひと月の１店舗あたり封入作業人件費は，１店舗あたり封入作業時間に人件費単価を乗じて算出する。

$$OCP1 = OC1 \times Pper \qquad \cdots\cdots\cdots(3.14)$$

④ ひと月の店舗あたり郵送コストは，取引先数に郵送料を乗じて算出する。

$$PST1 = Vq \times PSper \qquad \cdots\cdots\cdots(3.15)$$

⑤　店舗年間支払通知書総コストは，「①１店舗あたり支払通知書印刷コスト」，「③１店舗あたり封入作業人件費」と「④郵送コスト」に１２か月を乗じて算出する。

$$APC1 = （PC1+OCP1+PST1）\times M \qquad \cdots\cdots\cdots(3.16)$$

⑥ 百貨店業界全体では，１店舗年間支払通知書総コスト（APC1）に店舗数を乗じて算出するものとする。

$$APC = APC1 \times DSq \qquad \cdots\cdots\cdots(3.17)$$

使用記号

$PC1$：１店舗あたり支払案内通知書印刷コスト（円）

Vq：取引先数（社）

$PPper$：印刷単価（円）

OC1：１店舗あたり封入作業時間（時間）
OCv1：１社あたり封入作業時間（時間）
OCP1：１店舗あたり封入作業人件費（円）
Pper：人件費単価(円)
PST1：１店舗あたり郵送コスト（円）
PSper：郵送単価（円）
APC1：１店舗年間支払通知書総コスト（円）
PC1：１店舗あたり支払通知書印刷コスト(円)
PST1：郵送コスト（円）
M：年間月数(月)
APC：支払案内通知書コスト（円）
DSq：店舗数（店）

以上，計算式で支払案内電子化による効果を算出する。

4.8.2 集計結果

支払案内電子化を測定するため，百貨店数社へのヒアリング調査を実施し，得られて数値から係数を仮定し算出する。

① 取引先数は，催事有無，季節変動，店舗の規模により多少の上下はあるが，今回の算出では，取引先数を 1,500 社と仮定して算出する。
② 支払案内に封入する書類は，「支払通知書」「仕入明細書」「自動切り替え伝票」「経費総裁明細書」「売場ＰＯＳ過不足金明細書」などから構成され，これらの書類を印刷し，取引先別に封入する。1 社当たり平均 6 枚の書類が印刷され，書類発行コストを 1 社あたり 100 円とし，封入作業にかかる作業時間は 1 社当たり 2 分と仮定する。また，郵送料は，郵便定形 50 グラムとし，92 円とする。
③ 人件費単価は，4.7.3 検品削減で求めた人件費単価 1,875 円を使用する。
④ 百貨店店舗数は，日本百貨店協会の統計資料を使用し，273 店舗とする。

以上，各係数を定めて，計算式に挿入しコストを求めた。

表 3.10　支払案内電子化による効果測定結果一覧

項目	数値
PC1	150,000
OC1	50
OCP1	93,750
PST1	138,000
APC1	4,581,000
APC	1,250,613,000

単位：金額：(円), 時間：(時間)

算出の結果,支払案内電子化による効果は, 1,251 百万円と算出された。

4.9 EDI モデル導入による経済効果測定の結論

以上述べてきたように, EDI モデル導入による効果を測定した結果をまとめると「表 3.10」のとおりとなる。

伝票削減化の効果は 4,982 百万円, 検品削減による効果は 8,363 百万円, 値札廃止による効果は 637 百万円, 支払案内電子可能効果は 1,251 百万円, 合計で 15,232 百万円の経費削減効果が得られる結果が算出された。

表 3.11EDI モデル導入による経済効果測定結果

（単位：千円）

項目	経済効果金額
伝票削減化の効果	4,981,631
検品削減化の効果	8,362,933
値札廃止化の効果	636,604
支払案内電子化の効果	1,250,613
合計	15,231,782

5. 結論

EDI モデル導入による運用業務効率化を百貨店業界で解析した結果は，金額で 15,232 百万円の効果があることが明らかとなった。この金額は，買取仕入売上金額 2,675,910 百万円に対し，売上比 0.57%である。

百貨店の一般的な商品売買差益率を約 25%とすると，このコスト削減額を売上で稼ぐとなると，609 億円の売り上げが必要となる。この売上高は，業界 29 位の大丸札幌店 627 億円，30 位の阪神梅田本店 589 億円に匹敵する数字である。

現在，売上が厳しい状況下にある百貨店業界では，構造改革が求められている，製・配・販三層を対象とし，全体最適化を図る仕組みとして「SCM」の一環として，この EDI モデル導入による運用業務効率化施策，すなわち「商品マスター登録自動化」，「注文伝票削減化」，「検品削減化」，「伝票削減化」，「支払案内電子化」を実行することはとても重要である。今回の解析結果から約 152 億円の直接的効果が得られることを明らかにしたことは，この EDI モデルを実行する必要性を示していると考えられる。更に，「マーチャンダイズの最適化」による効果期待することができる。

この結果が示すとおり，百貨店業界は EDI モデル導入しないことにより，チェンスロスが発生している。すなわち，ここに機会損失が発生している。今後，百貨店が小売業として事業を継続するならば，百貨店が百貨店業界はこれらの構造改革を速やかに実施し，機会損失を最小にする取組みを実行することが求められていると考える。

6. おわりに

前述したとおり百貨店業界には，複数の EDI 標準が存在し，それに伴う複数の運用が存在している。そのため，取引先が新たに取引する際に導入コストがかかり，阻害要因なっている。この阻害要因を排除するため，経済産業省が推進する「流通システム標準化事業」において「百貨店業界流通システム化標準化委員会」を経て，EDI 業務モデルを構築し，国として唯一の標準 EDI である「流通ビジネスメッセージ標準（流通 BMS）」が公開された。

しかし，百貨店業界では普及がなかなか進めないのが現状である。その理由は，中小規模の百貨店・取引先で単品管理の装備が遅れている，システム投資の割にメリットが少ないと考えている，効果をよく理解できていないなどである。また，百貨店の取引形態が買取仕入中心から，場所貸しに近い消化仕入が増加しており，SCM を推進する力が低下する懸念もある。

この研究が示すとおり，SCM の導入によりもたらされる経済効果はあり，その経済効果を企業ごとに明らかにし，システム投資に対する効果を明確化することが，SCM を推進するうえで必要であると考える。

本研究では，導入による運用業務効率化の側面からの解析を実施したが，もう一つの側面である「マーチャンダイズの最適化」については対象外とした。単品管理の実施と取引先との SCM の実施により，「マーチャンダイズの最適化」を図ることにより「欠品による販売機会損失の削減」「百貨店顧客情報から得られる顧客ニーズによる実需の創造」「過剰在庫による値下げの抑制と不良在庫の低減」「返品による物流コストの削減」の効果が得られる。運用業務効率化による効果とマーチャンダイズの最適化による効果により，更なる大きな効果を百貨店と取引先は得ることが期待できる。

今後の研究課題として「マーチャンダイズの最適化」の側面を研究し，その効果を分析，効果を測定するともに，総合的に SCM の効果を明らかにすることが必要であると考える。

参考文献

[1] 生島義英, 唐澤豊, 若林敬造, 日本の百貨店における SCM の取組みと機会損失に関する研究, 日本ロジスティクスシステム学会第 20 回全国大会予稿集, 2017 年 7 月
[2] Yoshihide Ikushima, Yutaka Karasawa, Keizo Wakabayashi, A Study on SCM and Opportunity Loss in department Store in Japan., The 12th ICLS2017 Proceeding, August 2017
[3] 日本百貨店協会, 百貨店 IT 白書, 2001 年 1 月
[4] 日本百貨店協会, 2006 年版百貨店 IT 白書, 2006 年 1 月
[5] 百貨店業界流通システム標準化委員会, 百貨店業界における流通システム標準化事業活動報告, 平成 21 年 2 月
[6] 財団法人流通システム開発センター, 概説流通ＳＣＭ, 平成 19 年 3 月
[7] ㈱富士通総研,野村昌弘, 百貨店-アパレル間の情報共有基盤の現状
[8] 百貨店業界流通システム標準化委員会, 百貨店業界における流通システム標準化事業調査研究報告書, 平成 21 年 2 月
[9] 経済産業省, 流通システム標準化事業普及説明会テキスト東京会場, 平成 20 年
[10] 流通ＢＭＳ協議会会報流通, 流通ＢＭＳニュース№8,2010 年 7 月
[11] 百貨店ｅＭＰ事務局, 百貨店ｅＭＰ5 年史, 2007 年 11 月
[12] 百貨店・アパレル電子タグ導入推進委員会, 百貨店業界・アパレル業界における電子タグ実証実験報告書, 平成 17 年 3 月
[13] 百貨店ｅＭＰホームページ, URL:http://www.dept-emp.com/
[14] イクルスホームページ, URL:https://www.iqrs.net/
[15] 海上幸宏,百貨店における事例 小田急百貨店での流通 BMS の取組みについて : 業界唯一の標準を利用した EDI システムの効率化 (流通 BMS の現状と普及拡大に向けた新たな動き)流通とシステム (152), 16-21, 2012, 流通システム開発センター
[16] 星太一,百貨店業界のコラボレーション取引取り組みと EDI/SCM 導入 (ユビキタスネット社会の自動認識システム) , 自動認識 19(4), 41-45, 2006-03, 日本工業出版
[17] 西田雅一, 百貨店におけるコラボレーション型サプライチェーン (特別企画 EDI と新技術による流通情報システムの躍進), 流通とシステム (106), 5-10, 2000, 流通システム開発センター
[18] 原田保, 統合デ-タベ-スシステムによる経営戦略の革新--百貨店における競争力復権への新機軸 (特別企画/本格運用始まる流通 EDI), 流通とシステム (93), 38-46, 1997-09 , 流通システム開発センター

[19] 周嵩, 百貨店の SCM 導入に関する考察, 神戸学院大学経営学論集 1(1), 63-76, 2004-09-20
[20] 藤野直明, 統合オペレーション戦略のケーススタディ : 百貨店チャネルのアパレル流通における取引改革の分析(<特集>統合オペレーションの戦略・マネジメント), オペレーションズ・リサーチ : 経営の科学 48(12), 892-898, 2003-12-01
[21] 下村博史, 知識共有がもたらすサプライチェーンの革新:百貨店とアパレルとの協業的取引事例より, 日本物流学会誌 2004(12), 81-88, 2004
[22] 藤野直明,統合オペレーション戦略のケーススタディ : 百貨店チャネルのアパレル流通における取引改革の分析(<特集>統合オペレーションの戦略・マネジメント), オペレーションズ・リサーチ : 経営の科学 48(12), 892-898, 2003-12-01
[23] 池田暁治, 池田宗平, トータル物流システム導入事例/百貨店 配送受付管理システムと WMS の連携でギフト物流を刷新--(株)日立製作所の Giftmaster/HITLOMANS 利用事例 (特集 SCM を支えるロジスティクス技術と IT),マテリアルフロ- 44(11), 64-67, 2003-11
[24] 日経情報ストラテジー, SCM が迫る"最後の改革" 顧客起点で取引制度を刷新, 真のパートナー関係築け, 日経情報ストラテジー,10(10), 62-67, 2001-11
[25] 流通設計, CASE STUDY(1)SCM(岩田屋)百貨店主導で流通再編 物流は4社に一貫委託,流通設計 29(6), 89-91, 1998-06
[26] 日本百貨店協会,URL:http://www.depart.or.jp ホームページ,百貨店売上高データ

第 4 章　小売業におけるバックヤードの機会損失に関する基本的研究

1. はじめに

1.1 研究目的

百貨店・GMS・SM をはじめとする大型小売業は，店舗毎にバックヤードを有している。バックヤードには，商品・用度・什器備品ストック，厨房，事務所，休憩室，通路，階段（避難階段含む），搬入口，ごみ処理室，及び機械室等から構成されている。

バックヤードを売場に転用すれば収益を上げる事が可能であるが，現在は店頭業務をサポートする機能に利用されており，収益を得られていない。すなわち，ここに機会損失が発生しているものと考えられる。

本研究の目的は，百貨店及びスーパーマーケットに代表される大型小売店のバックヤードにおける機会損失額と売場転用収益効果において提案し，小売業の競争力強化並びに個別企業の経営革新の一助とする事にある。

本研究では，バックヤードの利用実態を調査し，バックヤードの縮小を検討し，売場転用可能なバックヤードを売場化することの効果を明らかにすることである。また，実務経験に基づいた機会損失の実感を分析・解析することを明らかにする。

最後にバックヤードの最小化を実現する解決策を検討し，マクロ的な効果を提言する。

1.3 先行研究調査

先行研究として，参考文献[17][18][19]がある。いずれの論文は，スーパーマーケットの BY から店頭への補充費用と店舗内在庫の確認費用の両面から BY と店頭への在庫の最適配備法やバックヤードにおける作業事故などの職場の安全衛生に関わる内容であった。先行研究では，大型小売業のバックヤードの利用実態を調査し，バックヤードの縮小を検討し，売場転用可能なバックヤードを売場化することの効果，その経済効果を測定した研究論文は皆無に近い。

バックヤードを縮小し売場化することにより得られる経済効果は経営判断上重要な情報である。よって，新たな研究対象として重要な意味を持っていると考える。

2. 店舗バックヤードの機会損失

2.1 店舗バックヤード

小売業の店舗は売場とバックヤードに区分される。バックヤードは，大きく商品ストックなどの部分，事務所・休憩所などの付帯施設の部分と設備機器・施設の部分に分類される。

商品ストックは，「図 4.1」のとおり，商品・用度品・什器等を保管している。商品は，店頭で販売する商品の在庫であり，店頭販売状況に応じて店頭に補充している。また，用度品は包装紙や手付き袋であり，店頭に補充する用度品がバックヤードに保管されている。

付帯施設は，事務所や検品所，社員食堂，休憩室，厨房などである。従業員の福利厚生施設を含む施設である。

図 4.1　店舗バックヤード例

設備・機器施設は，建物に付随し，建築基準法・消防法などにより設定が義務づけられている避難階段・避難器具や，電気空調設備や建物に付随する施設である。大型小売店は，消防法の規制により，常用階段の他に緊急時の避難階段などが設けられている。

これらを表にまとめると，「表 4.1」のとおりとなる。

表 4.1　小売業店舗機能区分

場所		内容
売場(店頭)		客用施設
バックヤード	商品ストック	商品・備品・用度品保管
	付帯施設	事務所・検品所・社員食堂・休憩室・厨房・作業場など
	設備・機器施設	機械室・電気室・空調機室，避難通路・避難階段など

2.2 店舗バックヤードの機会損失

本研究では，百貨店，スーパーマーケットなどの大型小売店の「店舗バックヤードの機会損失」に焦点を絞り，売場の適正在庫予算を策定し，予算遵守することにより在庫過多が発生しない仕組みと SCM により取引先との EDI を構築し，小売側のデータを取引先に提供することによりジャストインタイムの納品体制を構築し，バックヤードの不要不急な商品ストック在庫を排除し，バックヤードを縮小化すること，すなわちバックヤード削減化により得られる経済効果，すなわちバックヤードを売場転用しないことによって生じる機会損失を明らかにすることである。

店舗における在庫過多・過剰は「値下げや商品廃棄」，「商品の陳腐化や品質劣化」，「商品保管スペースの増大」，「商品整理など付帯作業の増加」が発生し，売買差益の減少，売り上げの減少，人件費の増大，無駄なコストの発生などが生じ，最終的に企業収益の悪化に繋がることとなる。

また返品条件付き買取仕入の場合，過剰在庫は販売シーズン終了後売れ残った商品として取引先へ返品することとなり，取引先の収益悪化にもつながることとなる。

よって，バックヤードの機会損失を無くすことはサプライチェーン全体に多大なメリットを得ることとなる。

2.3 バックヤードの機会損失の研究プロセス

2.3.1 研究プロセス

バックヤード機会損失の研究にあたり，次の手順ですすめる。

① バックヤード機会損失の経済効果を算出するための計算式を検討する。

② 業態別大型小売店へのフィールドスタディを実施する。バックヤードに関する調査票に基づき，調査結果をまとめ，業態別・バックヤード機能別の面積を推定する。

③ 把握した機能別バックヤード面積から，売場転用可能なバックヤードの面積を推定する。

④ 有価証券報告書，業界団体調査数値，官庁発表数値など公表されているデータを調査，抽出し，分析し，経済効果を算出するうえで必要な数値を把握する。これらの数値データを取りまとめ,必要な数値データとして確立する。

⑤ 数値データから解析し，縮小可能なバックヤード面積をもとめ，機会損失額を推定する。数値データを基に，計算を実施し，期待される経済効果を示す。

⑥ バックヤード削減化実現の方策を検討する。サプライチェーンマネジメントの視点から，どのようなモデルを組み立てれば，バックヤード削減化が実現の可能となるかを考察する。

2.3.2 バックヤード削減化の計算式の検討

バックヤードを売場に転用するバックヤード削減化によりもたらされるバックヤード解析に必要な項目を求める為，項目ごとに内容を検討し，計算式を提案する。

① 総バックヤード面積

$$Bs : Ss = bsr : Ssr \qquad \cdots\cdots\cdots(4.1)$$

$$Bs = \frac{Ss \times bsr}{Ssr} \qquad \cdots\cdots\cdots(4.2)$$

② 売場転用可能バックヤード面積

$$Brs = Bs \times Br \qquad \cdots\cdots\cdots(4.3)$$

③ １平方メートルあたりの売上高

$$Sm = \frac{Si}{Ss} \qquad \cdots\cdots\cdots(4.4)$$

④ 商品回転率

$$It = \frac{Si}{Is} \qquad \cdots\cdots\cdots(4.5)$$

⑤ 在庫経費

$$Ict = Is \times Icr \qquad \cdots\cdots\cdots(4.6)$$

⑥ １㎡あたりバックヤードスペースコスト

$$Sc = Bc \times M \qquad \cdots\cdots\cdots(4.7)$$

⑦ 補充配送コスト

$$Rc = Si \times Rcr \qquad \cdots\cdots\cdots(4.8)$$

⑧ 税引き前純利益率

$$Pir = \frac{Pi}{Si} \qquad \cdots\cdots\cdots(4.9)$$

⑨ 売上増に伴う雇用創出

$$\triangle Li = \frac{\triangle Si}{Ms} \qquad \cdots\cdots\cdots(4.10)$$

使用記号

Bs：総バックヤード面積(㎡)
Ss：総売場面積(㎡)
Bsr：総バックヤード面積比率(%)
Ssr：総売場面積比率(%)
Brs：売場転用可能バックヤード面積(㎡)
Br：バックヤード売場転用可能比率(%)
Sm：１平方メートルあたりの売上高(円)
Si：総売上高(円)
It：商品回転率(回)
Is：平均在庫金額(円)
Ict：在庫経費(円)
Icr：在庫経費率(%)
Sc：１㎡あたりバックヤードスペースコスト(円)
Bc：１㎡あたりバックヤード月額賃料(円)
Rc：補充配送コスト(円)
Rcr：補充配送コスト率(%)
Pir：税引き前純利益率(%)
Pi：税引前純利益(円)
△Li：売上増に伴う雇用創出(人)
△Si：売上増加額(円)
Ms：従業員１人当り売上(円)

2.3.3 バックヤード削減化の経済効果の計算式の検討

バックヤードを売場に転用するバックヤード削減化によりもたらされる経済効果の計算式を検討し，以下に示す。

なお，売場転用可能バックヤード売上は，現行売場と同等の売上が得られることを仮定として算出する。

① 売上増加額

$$\triangle Si = Brs \times Sm \qquad \cdots\cdots\cdots(4.11)$$

② 売上増に伴う在庫金額

$$Ia = \frac{\triangle Si}{It} \qquad \cdots\cdots\cdots(4.12)$$

③ 増加在庫コスト

$$\triangle Ic = Ia \times Icr \qquad \cdots\cdots\cdots(4.13)$$

④ バックヤード削減化により削減されるスペースコスト

$$Lc = Bs \times Sc \qquad \cdots\cdots\cdots(4.14)$$

⑤ 想定される税引き前利益増加見込み

$$\triangle Pi = \triangle Si \times Pir \qquad \cdots\cdots\cdots(4.15)$$

⑤　利益見込想定される経済効果

$$Pe = \triangle Ic + Rc + Lc + \triangle Pi \qquad \cdots\cdots\cdots(4.16)$$

⑦ 新規雇用機会

$$\triangle Li = \frac{\triangle Si}{Ms} \qquad \cdots\cdots\cdots(4.17)$$

⑧ 想定される総売上高

$$S'i = Si + \triangle Si \qquad \cdots\cdots\cdots(4.18)$$

使用記号

$\triangle Si$：売上増加額(円)
Brs：売場転用可能バックヤード面積(㎡)
Sm：1㎡あたり売上高(円)
Ia：売上増に伴う増加在庫金額(円)
It：商品回転率(回転)
$\triangle Ic$：増加在庫コスト(円)
Icr：在庫経費率(%)
Lc：バックヤード削減化により削減されるスペースコスト(㎡)
Bs：バックヤード削減面積(㎡)
Sc：1㎡あたりバックヤードスペースコスト(円)
$\triangle Pi$：想定される税引き前利益増加見込み(円)
Pir：税引前純利益率(%)
Pe：利益見込想定される経済効果（円）
$\triangle Li$：新規雇用機会(人)
Ms：従業員一人当り売上高(円)
$S'i$：想定される総売上高(円)

3 大型小売店への調査票による調査

3.1 調査概要

業態別大型小売店のバックヤード面積の現状を把握するために，代表的な小売業へ調査票を郵送，調査およびフィールドスタディを実施し，バックヤードの機能別面積を把握するための基礎データを入手する。

百貨店，ＧＭＳ・ＳＭの代表的な企業にバックヤードに関する調査票に基づく調査を2015年6月から9月にかけて実施し，バックヤードの機能別面積を推定するための基礎数値を把握する。

調査票による具体的な調査項目は下記のとおりである。

① 面積割合の把握および商品ストック・事務所等と設備関連面積割合の把握
② バックヤード商品ストック保管商品・物品などの内容把握
③ 年間売上高の把握
④ 仕入形態別売上構成の把握（本仕入・売上仕入・賃貸）
⑤ 物流センター有無と機能，面積の把握
⑥ 現在のバックヤード縮小への取り組み有無と取り組み内容の把握

⑦ 取引先とのＥＤＩの取り組み内容の把握

郵送による調査票調査と合わせて，個別企業の担当者と直接面接し，調査票に基づくヒアリング調査も並行して実施する。直接会うことにより，書面では得られにくい現状の問題点や課題，ならびに回答内容の細かいニュアンスなどを調査することにより，調査結果をより現実的なものにブラッシュアップする。

3.2 調査結果

百貨店売上上位 10 社・スーパー売上上位 16 社に郵送によるアンケート調査票を送達した。6 社より返送され，そのうち 4 社は回答不能であり，芳しい調査結果を得ることができなかった。回答のない各社へ追跡調査を実施し，回答できない理由を調査したところ，「物流業務をアウトソーシングしており，実態が把握できていない。」「官公庁の調査以外回答しない。」，「繁忙なため回答できない。」，「守秘義務により回答できない。」，　「コンプライアンス上回答不能」などであった。

アンケート調査の回答が十分に得られなかったため，個別企業に直接アプローチを行い，ヒアリング調査を実施した。匿名を条件に百貨店 2 社，スーパー2 社にヒアリング調査を実施し，調査データを得ることができた。

アンケート調査および公的調査結果により得られたバックヤード比率は「表 4.2」に示すとおりである。

アンケート・ヒアリング調査結果より百貨店・スーパーのそれぞれのバックヤード比率の平均値は，百貨店バックヤード比率 18.9%，スーパーバックヤード比率 30.0%であった。

また，日本スーパーマーケット協会・オール日本スーパーマーケット協会・一般社団法人 新日本スーパーマーケット協会による「平成 26 年スーパーマーケット年次統計調査報告書」調査結果では，「バックヤード比率は平均 29.8%」である。

著者が調査した 1987 年（約 30 年前）のスーパーマーケットのバックヤード比率調査結果は，22.4%であった。当時の調査は，大手 5 社中 4 社から回答があり，合計 7 社から回答を得られ，その平均値をバックヤード比率とした。今回の調査では，約 30 年前と比較し，バックヤード比率が増加傾向にあることが理解できる。

そこで，本研究では，バックヤード比率を「百貨店：20%」，「スーパーマーケット：30%」と仮定して計算するものとする。

表 4.2　バックヤード比率調査結果一覧[9]

	A 社	B 社	百貨店平均	C 社	D 社	E 社	日本ｽｰﾊﾟｰ協会*1	平均
業態	百貨店	百貨店	百貨店	ｽｰﾊﾟｰ	ｽｰﾊﾟｰ	ｽｰﾊﾟｰ	ｽｰﾊﾟｰ	ｽｰﾊﾟｰ
売場	74.8	87.3	81.1	70.0	60.0	80.0	70.2	70.0
ﾊﾞｯｸﾔｰﾄﾞ	25.2	12.7	18.9	30.0	40.0	20.0	29.8	30.0

3.3 売場転用可能なバックヤード面積の推定

アンケートならびにヒアリング調査から判明したバックヤードの利用内訳は「表 4.2」に示すとおりである。

設備機器施設は，避難階段，電気室，消防ポンプ室，防災センターなど建物に付随し，建築基準法・消防法などにより設定が義務づけられている設備や建物に付随する施設であるため，売場転用は難しい。

また，付帯施設は，福利厚生施設である社員食堂･休憩室は，立っての接客業務がほとんどを占める小売業の従事形態ではそれらを廃止することは不可能である｡厨房は惣菜などできたて商品提供するうえでなくてはならいものである。売場事務所は，最小限の機能であり，売場対応を鑑みるとなくすことは難しい。また，検品所は納返品の店舗起点であるため，検品方法の簡略化により縮小することは可能であるが，廃止することはできない。

したがって，売場転用可能な機能は，商品ストックのみと言える。最大売場転用可能面積は，バックヤードのうち商品等ストック面積とし，調査結果では，平均値 42.4%であったため，ここでは数値を「40%」と仮定して計算するものとする。

よって，「図 4.3」に示すとおり，バックヤードを「40%」縮小し，売場化するとして，売上増加額を算出するものとする。一方，現在のバックヤードの 60%は，付帯施設・設備機器施設としてそのまま残存させる。

表 4.2　バックヤード機能内訳一覧

機能	A 社	B 社	C 社	平均
商品ストック	37.2%	40.0%	50.0%	42.4%
付帯施設	25.4%	20.0%	25.0%	23.5%
設備機器施設	37.4%	40.0%	25.0%	34.1%

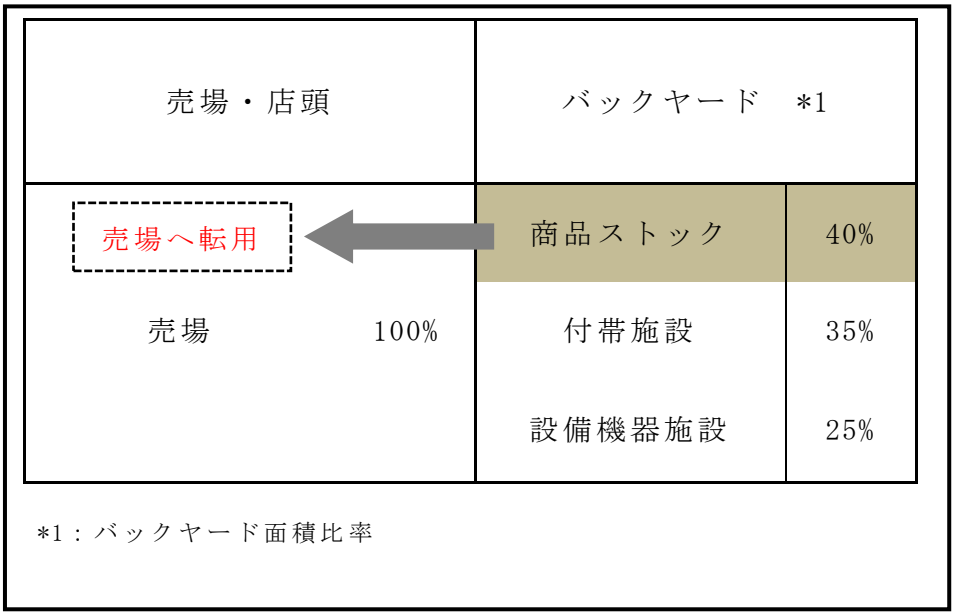

図 4.3　バックヤードの売場転用可能比率の見積

4 バックヤード削減化の効果測定のための基礎数値

4.1 経済産業省「商業動態統計」調査結果

小売業のマクロデータは，経済産業省から公表されている「商業動態統計」から引用し，全国，産業経済局単位，都道府県単位の販売額，売場面積，雇用人数などの基礎データを調査し，基礎数値として把握する。

事業者数，販売金額，従業員数，売場面積は，経済産業省が発表している「平成 25 年商業動態年報」の数値を採用し，基礎数値とした。これらの数値データをまとめたのが「表 4.4」である。

表 4.4　大型小売業データ[8] (金額:百万円,人数:人,面積:1,000 ㎡)

項目	百貨店	スーパー	合計
事業所数	254	5,057	5,311
販売額	6,719,526	13,057,880	19,777,407
従業員数	87,280	528,062	615,342
一人当たり販売額	77.0	24.7	32.1
売場面積	6,659	25,571	32,229
１㎡あたり販売額	1.01	0.51	0.61

4.2 「有価証券報告書」調査結果

百貨店，スーパーの上場企業の有価証券報告書を分析することにより，売上高，期末在庫高，原価率，経常利益，税引き前利益などの決算数値などを把握し，バックヤードを解析するための基礎数値を把握する。

商品回転率，税引き前利益率などは，上場企業百貨店９社，スーパー２９社の２０１４年度決算有価証券報告書から各指標を引用して算出する。各指標をまとめたのが「4.5」,「表 4.6」である。

ただし，業界最大手であるセブン&イレブン（イトーヨーカドー・西武そごう），三越伊勢丹，ＪＦＲ（大丸松坂屋），イオン，ユニー，ダイエーなどは，持ち株会社制に移行し，H2O（阪急・阪神百貨店）などは，電鉄子会社のため，それらの企業は，セグメント情報として売上高などのデータは記載されているが，小売り本体の経営指標，すなわち売上原価，期末在庫原価，経常利益，税引き前利益などが有価証券報告書に記載されていない。

よって，今回の分析に必要なデータが得られないため，持ち株会社制企業の数値は採用していない。有価証券報告書には期末在庫原価のみが記載されてい

るため，商品原価率を除して期末在庫売価金額を求めた。計算式は，「期末在庫売価金額＝期末在庫原価　÷　商品原価率」である。

また，期末は商品在庫が最も少ない時期であることから，平均在庫売価金額を算出するために期末在庫売価に調整在庫係数 105％と仮定し，乗じて算出するものとする。

商品回転率は，売上高に平均在庫金額を除して算出する。計算式は，「商品回転率＝売上高÷平均在庫売価金額」である。

これらの数値結果から解析に必要な商品回転率，売上原価，期末在庫金額，税引き前利益率などの基礎数値を「表 4.7」に取りまとめた。

表 4.5　有価証券報告書決算数値一覧（百貨店）[7]　　(金額:百万円)

会社名	高島屋	近鉄	松屋	井筒屋	大和	丸栄	山陽百貨店	ながの東急	金沢名鉄丸越百貨店	合計
売上高	689,779	259,861	70,881	62,341	46,743	20,997	19,454	19,166	14,924	1,204,145
売上原価	517,414	196,226	54,434	46,985	36,691	16,796	15,299	14,791	11,956	910,593
商品原価率	75.0%	75.5%	76.8%	75.4%	78.5%	80.0%	78.64%	77.17%	80.12%	75.6%
商品利益	172,365	63635	16,447	15,356	10,052	4201	4,155	4,375	2,967	293,553
商品利益率	25.0%	24.5%	23.2%	24.6%	21.5%	20.0%	21.4%	22.8%	19.9%	24.4%
期末商品在庫原価	32,368	9,666	1,854	2,576	2,492	393	1,053	936	747	52,084
期末商品在庫売価換算	43,151	12,801	2,414	3,418	3,174	491	1,339	1,213	932	68,933
平均在庫売価	45,308	13,441	2,535	3,589	3,333	516	1406	1,273	979	72,379
商品回転率	15.2	19.3	28.0	17.4	14.0	40.7	13.8	15.1	15.2	16.64
調整在庫係数	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05
経常利益額	14,503	2,440	2,011	1,272	651	-75	220	72	68	21,161
経常利益率	2.1%	0.9%	2.8%	2.0%	1.4%	-0.4%	1.1%	0.4%	0.5%	1.76%
税引前純利益額	18,577	-479	1,906	-4,471	1,195	-270	186	59	48	16,752
税引前純利益率	2.69%	−0.18%	2.69%	−7.17%	2.56%	−1.29%	0.96%	0.31%	0.32%	1.39%
備考		アベノハルカス特損		抱合せ株式消滅差損	固定資産権利変換益				特損：282 特収：262	

表 4.6　有価証券報告書決算数値一覧（スーパー）[7]　　(金額：百万円)

会社名	ライフコーポレーション	イズミ	マルエツ	平和堂	バロー	フジ	ヤオコー	オーケー	マックスバリュ西日本	オークワ
売上高	568,717	530,507	339,681	328,477	316,199	291,710	282,449	282,241	263,041	260,391
売上原価	413,667	426,219	246,455	239,110	236,152	237,546	214,975	222,595	201,513	196,345
商品原価率	72.7%	80.3%	72.6%	72.8%	74.7%	81.4%	76.1%	78.9%	76.6%	75.4%
商品利益	155,050	104,288	93,226	89,367	80,047	54,164	67,474	59,646	61,528	64,046
商品利益率	27.3%	19.7%	27.4%	27.2%	25.3%	18.6%	23.9%	21.1%	23.4%	24.6%
期末商品在庫原価	21,686	21,977	9,145	14,774	15,118	9,350	5,465	4,220	9,608	8,822
期末商品在庫売価換算	29,814	27,354	12,604	20,296	20,242	11,482	7,180	5,351	12,542	11,700
平均在庫売価	31,305	28,722	13,234	21,311	21,255	12,056	7,539	5,618	13,169	12,285
商品回転率	18.2	18.5	25.7	15.4	14.9	24.2	37.5	50.2	20.0	21.2
調整在庫係数	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05
経常利益額	10,928	25,058	3,870	13,589	7,885	4,259	12,599	14,868	4,700	2,787
経常利益率	1.9%	4.7%	1.1%	4.1%	2.5%	1.5%	4.5%	5.3%	1.8%	1.1%
税引前純利益額	8,838	24,318	777	13,439	6,221	3,723	13,112	14,103	3,589	633
税引前純利益率	1.55%	4.58%	0.23%	4.09%	1.97%	1.28%	4.64%	5.00%	1.36%	0.24%
備考										

会社名	カスミ	イオン九州	マックスバリュ東海	いなげや	マックスバリュ中部	イオン北海道	サンエー	マックスバリュ九州	関西スーパーマーケット	マックスバリュ東北
売上高	240,375	226,405	202,583	185,000	158,831	156,182	155,097	141,936	113,077	109,011
売上原価	179,254	165,473	152,758	131,976	119,270	114,274	106,942	108,879	86,181	84,849
商品原価率	74.6%	73.1%	75.4%	71.3%	75.1%	73.2%	69.0%	76.7%	76.2%	77.8%
商品利益	61,121	60,932	49,825	53,024	39,561	41,908	48,155	33,057	26,896	24,161
商品利益率	25.4%	26.9%	24.6%	28.7%	24.9%	26.8%	31.0%	23.3%	23.8%	22.2%
期末商品在庫原価	5,383	26,355	5,102	4,215	3,685	12,773	10,144	3,662	2,772	3,051
期末商品在庫売価換算	7,218	36,060	6,766	5,908	4,908	17,457	14,712	4,774	3,637	3,920
平均在庫売価	7,579	37,863	7,104	6,204	5,153	18,330	15,447	5,013	3,819	4,116
商品回転率	31.7	6.0	28.5	29.8	30.8	8.5	10.0	28.3	29.6	26.5
調整在庫係数	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05
経常利益額	7,006	-1,106	3,840	2,182	1,071	7,765	12,425	1,448	51	231
経常利益率	2.9%	-0.5%	1.9%	1.2%	0.7%	5.0%	8.0%	1.0%	0.0%	0.2%
税引前純利益額	6,200	-3,252	2,960	1,490	542	6,552	12,068	795	-1,050	-871
税引前純利益率	2.58%	-1.44%	1.46%	0.81%	0.34%	4.20%	7.78%	0.56%	-0.93%	-0.80%
備考		特損： 3,460		抱合せ株式消滅差益： 5,243						

会社名	アオキスーパー	マックスバリュ北海道	ハローズ	ヤマザワ	マルキョウ	東武ストア	マキヤ	天満屋ストア	北雄ラッキー	合計
売上高	97,738	95,239	93,016	85,378	84,350	80,941	56,446	49,384	42,669	5,837,069
売上原価	81,785	74,404	70,192	64,584	66,318	58,904	44,280	38,668	31,960	4,415,529
商品原価率	83.7%	78.1%	75.5%	75.6%	78.6%	72.8%	78.4%	78.3%	74.9%	75.6%
商品利益	15,953	20,835	22,824	20,793	18,032	22,037	12,166	10,716	10,709	1,421,540
商品利益率	16.3%	21.9%	24.5%	24.4%	21.4%	27.2%	21.6%	21.7%	25.1%	24.4%
期末商品在庫原価	1,406	2,240	2,407	2,783	3,191	2,039	5,070	2,934	1,888	221,265
期末商品在庫売価換算	1,680	2,867	3,190	3,679	4,059	2,802	6,463	3,748	2,520	292,500
平均在庫売価	1,764	3,010	3,349	3,863	4,262	2,942	6,786	3,935	2,646	307,125
商品回転率	55.4	31.6	27.8	22.1	19.8	27.5	8.3	12.6	16.1	19.0
調整在庫係数	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05	1.05
経常利益額	3,138	1,596	3,354	917	1,933	1,192	856	1,113	223	149,778
経常利益率	3.2%	1.7%	3.6%	1.1%	2.3%	1.5%	1.5%	2.3%	0.5%	2.6%
税引前純利益額	2,740	950	3,389	876	1,859	430	815	575	177	125,998
税引前純利益率	2.80%	1.00%	3.64%	1.03%	2.20%	0.53%	1.44%	1.16%	0.41%	2.16%
備考							業務スーパー			

表 4.7　業態別有価証券報告書経営指標一覧[7] (金額：百万円)

項目	百貨店 9 社	スーパー29
売上高	1,204,145	5,837,069
売上原価	910,593	4,415,529
商品原価率	75.6%	75.6%
商品利益	293,553	1,421,540
商品利益率	24.4%	24.4%
期末商品在庫原価	52,084	221,265
期末商品在庫売価換算	68,933	292,500
平均在庫売価	72,379	307,125
商品回転率	16.64	19.01
調整在庫係数	1.05	1.05
経常利益額	21,161	149,778
経常利益率	1.76%	2.57%
税引前純利益額	16,752	125,998
税引前純利益率	1.39%	2.16%

4.3 物流センターコスト調査結果

かつて大型小売店は商品を納品・検品・各店舗への集約納品する物流センターを自社運営で有していた。しかし，近年コスト削減のため自社運営センターを廃止し，納品代行や第 3 者が運営する形態（3PL，主幹問屋制）が大半を占めている。

現在，百貨店業界は自社物流センター有しているが，現在の使用目的は中元歳暮に代表されるギフト配送に利用されており，調達系物流は納品代行業者による検品代行・集約納品の形態をとっており，自社センターを利用した納品はほとんど行われていない。よって，物流センターによるコスト削減は期待できない。

一方，スーパー業界は小売物流センターに納品にあたりセンターフィ，一般的には納品上代の 2.5%を支払っている。近年センターフィに関しては，公正取引委員会の指導もあり，3PL 業者もしくは，菱食や国分などの卸売業（主幹取引先）が受領する形態をとっており，物流センター廃止による自社コスト削減は期待できない。よって，本研究では，コスト効果から物流センターコストを除外して，算出することとする。

4.4 その他の基礎数値の調査結果

在庫経費率の基礎数値は，日本ロジスティクスシステム協会の「ＪＩＬＳ 2014 年度物流コスト調査報告書」から在庫経費率の数値は，ヘスケット方式マクロ物流コスト推計値(在庫経費率)の数値を採用し，経費比率は 22%とする。

その他必要な基礎数値や参考数値は，百貨店協会，チェーンストア協会，日本スーパーマーケット協会・オール日本スーパーマーケット協会・一般社団法人 新日本スーパーマーケット協会など各業界団体の調査統計資料を調査し，基礎数値として把握する。

5. バックヤード機会損失額の計算

5.1 機会損失計算基礎数値のまとめ

大型小売店への調査，および各文献や調査報告書から得られた基礎数値をまとめたものが，「表 4.8」である。

表 4.8　バックヤード削減化基礎数値一覧　(単位：金額：百万円　面積：1,000 ㎡)

記号	項目名	全国計		
		百貨店	スーパー	合計
Si	総売上高	6,719,526	13,057,880	19,777,406
Ss	総売場面積	6,659	25,571	32,230
Ls	従業員数	87,280	528,062	615,342
Ms	従業員一人あたり売上高	76.99	24.7	102
Bsr	総バックヤード面積比率	20%	30%	-
Ssr	総売場面積比率	80%	70%	-
Bm	バックヤード割合(売場対)	25.0%	42.9%	-
Bs	総バックヤード面積	1,665	10,959	12,624
Br	バックヤード売場転用可能比率	40.0%	40.0%	-
Brs	バックヤード売場転用可能面積	666	4,384	5,050
Sm	1 ㎡あたり売上高	1.01	0.51	0.61
Is	商品在庫金額(平均)	403,818	687,057	-
It	商品回転率	16.64	19.01	-
Ict	在庫経費	88,840	151,152	239,992
Icr	在庫経費率	22%	22%	-
Sc	１㎡あたりバックヤードコスト	0.036	0.036	-
Rc	店舗への商品補充配送コスト	167,988	326,447	494,435
Pr	税引前純利益率	1.39%	2.16%	-

5.2 バックヤード機会損失額の計算

「表 4.8」の基礎数値を 2.3.4 項で述べたバックヤード削減化経済効果計算式に代入し，経済効果を算出した結果が「表 4.9」である。

表 4.9　バックヤード削減化によりもたらされる効果解析結果一覧

(単位：金額：百万円　面積：1,000 ㎡)

記号	項目名	全国計		
		百貨店	スーパー	合計
ΔSi	売場面積増による売上増加	671,953	2,238,494	2,910,446
ΔSir	売上増加比率	10.0%	17.1%	14.7%
Es	従業員雇用増加数	8,728	90,525	99,253
Sh	雇用対売上割合	76.99	24.73	-
Ia	在庫金額	40,382	117,781	158,163
Ic	在庫コスト	8,884	25,912	34,796
Lc	削減されるスペースコスト	23,972	157,810	181,782
ΔPi	想定される税引き前利益増加額	9,340	48,320	57,660
Pe	想定される経済効果	42,197	232,041	274,238
ΔLi	新規雇用機会	8,728	90,525	99,253
$S'i$	想定される総売上高	7,391,479	15,296,374	22,687,852

バックヤードを売場に転用することによりもたらされる売上増加額, すなわち売上の機会損失が発生している金額は次のとおりである。

百貨店においては, 671,953 百万円の売上増となり, 総売上の 10.0%増加させることとなる。税引き前利益では, 9,340 百万円増加させる。

一方, スーパーは, 2,238,494 百万円の売上増となり, 総売上の 17.1%増加させることとなる。税引き前利益では, 48,320 百万円増加させる。

更に合計で 2,910,446 百万円の売上増となり, 14.7%%増加させることとなる。

税引き前利益では, 57,660 百万円増加させる。

さらに, 百貨店は, 42,197 百万円の経済効果を得られることとなる。一方, スーパーは 232,041 百万円の経済効果得られるとなる。更に合計で 274,238 百万円の経済効果を得られることとなる。

バックヤード削減化により齎される新規雇用数は次の通りとなる。

百貨店は 8,728 人の雇用増となる。一方, スーパーは 90,525 人の雇用増となる。更に合計で 99,253 人の雇用増となる。

5.3 バックヤード削減化によりもたらされる効果についての考察

バックヤード削減化によりもたらされる効果解析結果から, 百貨店においては, 6,720 億円の売上増が期待される。この売上額は, 2015 年度決算数値を鑑みた場合,業界大手である高島屋本体営業収益 6,898 億円に匹敵する額である。税引き前利益では, 93 億円の増益が期待される。これらの金額は, 低迷する百貨店業界では無視できる金額ではないと思われる。

一方, スーパー業界においては, 2 兆 2,385 億円の売り上げ増が期待される。この売上額は, 2015 年度決算数値を鑑みた場合, ＡＥＯＮグループ GMS の収益が, 3 兆 3,555 億, SM・DS（ディスカウントストア）・小型店事業の収益が 2 兆 1,613 億円, イトーヨーカドーの収益が 1 兆 2,537 億円である。これらの金額と比較した場合, その効果金額の大きさを実感することができる。更に税引前利益では, 483 億円の増益が期待できる。

百貨店, スーパー合わせて, 2 兆 9,104 億円の売上増が期待される。大型小売業の売上を 14.7%引き上げる金額である。これだけの売上増を既存店舗施設で運営の仕組みを変えること, すなわちバックヤード削減化によって実現可能となるのである。表現を換えれば, バックヤード削減化を実施しないことにより, チャンスロスすなわち機会損失が発生していると考えられる。

一方, バックヤード削減化には EDI をはじめとするサプライチェーンマネジメントシステムを取引先と小売側で構築する必要がある。一般的に, 小売業におけるシステム投資可能額は売上の 0.5%と言われている。この数値から年間のシステム投資額を試算すると, 売上増により年間 146 億円を新たにシステム投資に回すことが可能であると考えられる。システムの償却を 5 年とした場合, バックヤード削減化のサプライチェーンシステムに 730 億円のシステム投資が追加可能であると考えられる。小売業としては, これを原資としてシステム開発を行うことが可能であり, この投資によりバックヤード削減化実現可能となる。

また, 物流業務を受託する企業は, 売上増加額の 2.5%が, 配送コストとして新たに発生するため, 年 728 億円の増収が期待できる。

これらのように, 小売業の売上増加は, 小売業に関わる様々な産業に波及し, 相乗効果により, より大きな経済効果を得ることができると考える。

さらに,バックヤード削減化によりもたらされる売上増加に伴い，新規雇用を得ることが可能となる。

百貨店は 8,728 人，スーパーは 90,525 人，合計で 99,253 人の雇用増が期待される。小売業は，販売業務に従事する者が大半を占めることから，高齢者や女性の活躍の場を新たに提供することが可能となる。

これらの売上増，利益増大，雇用増加，周辺産業への波及などの効果が示すとおり，バックヤード削減化を実施しないことにより，大型小売店は大きな機会損失が発生していると考えられる。

6.バックヤード削減化実現の提案

6.1 従来型サプライチェーンモデル

「図 4.5」に従来型サプライチェーンのモデルを示す。従来型モデルでは，メーカー倉庫・卸売業者・小売業者各々が商品在庫を有しており，多段階在庫の状況を呈している。このような状況は，商品の過多・過剰を生み出す。それぞれの段階でそれぞれの思惑により在庫を抱え，それが積み重なって在庫が膨らむことになる。

従来型サプライチェーンモデルでは在庫が膨らむ問題が発生している。特に百貨店など大型小売業は，売上を確保するために，少しでも多くの商品を確保するため，商品の取り合い状態となる。それにより小売側の在庫は多くなり，それを保管するために店舗バックヤード，店外倉庫を拡大して，多くの商品在庫を抱えることとなる。

しかし，百貨店は販売シーズン終了後に返品条件付き買取仕入のため，残った商品を取引先に返品する。取引先はシーズン終了後に商品を返品され，販売のタイミングを失った商品，すなわち格下げ品，不良在庫・商品ロスが発生する。サプライチェーン全体でみれば大きな損失が発生する。

取引先は自社の利益を優先するため，返品による商品ロスを少しでも減らすために，売れている店，販売力がある店舗に優先的に商品を供給する。一方，売れていない店，販売力がない店には商品供給の優先度が低くなる。

近年，ファッションアパレルの取引先は，返品条件付き買取仕入から，売上仕入（消化仕入）に契約条件を変更している。売上仕入にすることにより，商品の所有権は取引先となり，小売側の決裁なく，自由に商品を売れている店舗へ商品移動をかけることができるようになる。この方策により，取引先は不良在庫を削減している。

一方，売れていない店舗は，欲しい商品確保のため返品条件なし買取仕入でも，バックヤードや倉庫に商品を貯めこむ傾向が強くなる。そのため，セール期は自社の資金で売変するなど売買差益は低下し，更にシーズン終了後，返品できず不良在庫となる。それにより，小売り側の利益額は悪化する。

地方百貨店や郊外店などは，商品確保できないことによる売上低迷，商品ロ

スによる収益の悪化が店舗の赤字化につながり，事業そのものが継続できなくなり，業態変更や閉店に追いやられているのが昨今の現状である。この悪循環を SCM により改革する必要性が小売業全体に求められていると考える。

6.2 バックヤード削減化と SCM

「図 4.3」に今回提案する小売業のバックヤード削減化のモデルを示す。バックヤード削減化を実現させるためには，店舗バックヤードストックに商品在庫を持たないで営業が成り立つ仕組みが必要となる。すなわち川下である店舗に欠品することなく，必要な時に必要な量の商品が恒常的に供給されることがポイントである。

バックヤードを縮小するために，「単品管理」と「データ交換」などにより小売業と取引先，メーカーとの間に情報共有化するある SCM を構築する必要がある。

SCM は収益を改善する経営管理手法として，産業界に広く認知され，高度な情報技術を適用し，需要予測・在庫・生産計画などの諸情報を企業間で共有することが，在庫極小化や物流効率化に繋がるとされている。

具体的には，商品にＪＡＮコードを付けるという単品管理レベルから始まって，自動補充発注もしくは取引先に在庫情報を提供し取引先主導の在庫管理(VMI)を実施する。自動補充発注や VMI は品切れをゼロにするとともに，過剰在庫を減らす。

さらに，店別・単品別の売上と数年間分の在庫情報を全部ネットワーク上にオープンにし，取引先がその中身をいつでも見に行けるようにする。さらには，小売業が自社店頭の自動補充発注をするのではなく，卸やメーカーが小売の店頭の自動補充発注を代りに行う，パートナーシップ自動補充発注を実施する。あわせて，SCM 検品など納品手続きの簡略化・迅速化を実現し，検品などに係る時間と手間を削減し，納品リードタイムを短縮する。

それらの施策により，店頭在庫のみで売場を運営し，バックヤードの商品在庫を排除することにより，バックヤード削減化が実現可能となる。すなわち，製・配・販三層間の情報共有をベースに，SCM の仕組みを構築することにより，バックヤード削減化が実現されると考える。

唐澤豊が提唱している在庫川上論の考え方では，流通の各段階にある在庫を圧縮し，末端の小売り販売情報を迅速に生産計画に反映させるためには，小売業は在庫を極力減らして卸売業の在庫を利用すればよい，また卸売業は自社に在庫を持たずにメーカー在庫を活用するシステムを実現させることであると提唱されている。このシステムは，メーカー・卸売業の在庫を小売業の自社在庫のように利用できるしくみである。すなわち商品在庫を川上に集約・保管し，川下の必要に応じて供給することである。

具体的には，小売業者の売上データ，在庫データは EDI を介して，メーカ

一・卸売業に連動することにより，売上・在庫実績データを基に，メーカーが小売り側へ適正な商品供給をマネジメントする SCM システムを導入する。

それにより，卸売業者などを排除し，メーカーと小売業を短絡化することにより，多段階在庫をなくすことが可能となる。これにより，中間在庫を排除する。一方，卸売業者をなくすことにより，卸の手間やコストが削減される。

販売実績に基づき，自動的に商品供給することにより小売業の無駄な在庫を排除する。無駄な在庫を排除することにより，店頭のみの商品在庫量で店舗を運営することができるようになり，最終的にはバックヤードストックを排除することが可能となる。

さらに SCM の進展により，今まで卸売業が担っていた機能を SCM システムが担うことになる。メーカーは，SCM システムと生産管理システムの連携・高度活用により，需要に応じた生産体制を構築し，サプライチェーン全体で発生する無駄を排除することが可能となる。そのイメージを「図 4.4」に示す。

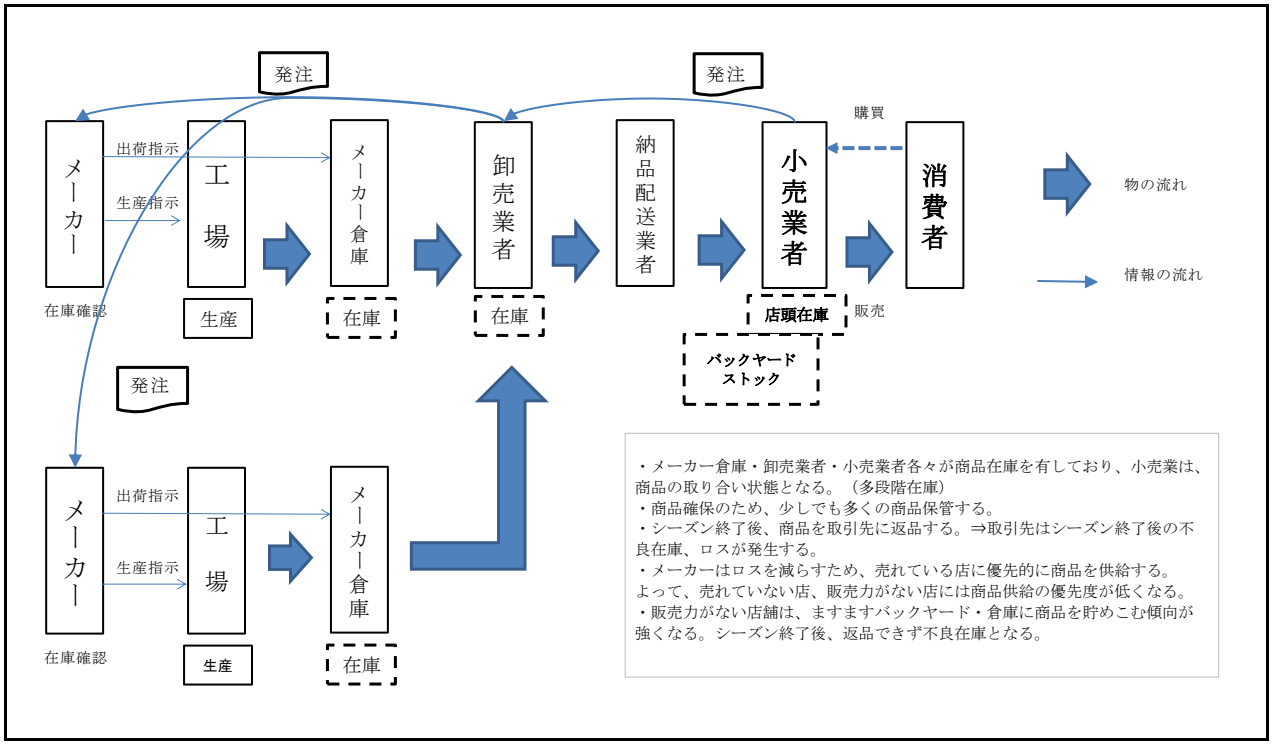

図 4.3　従来型のサプライチャーンイメージ

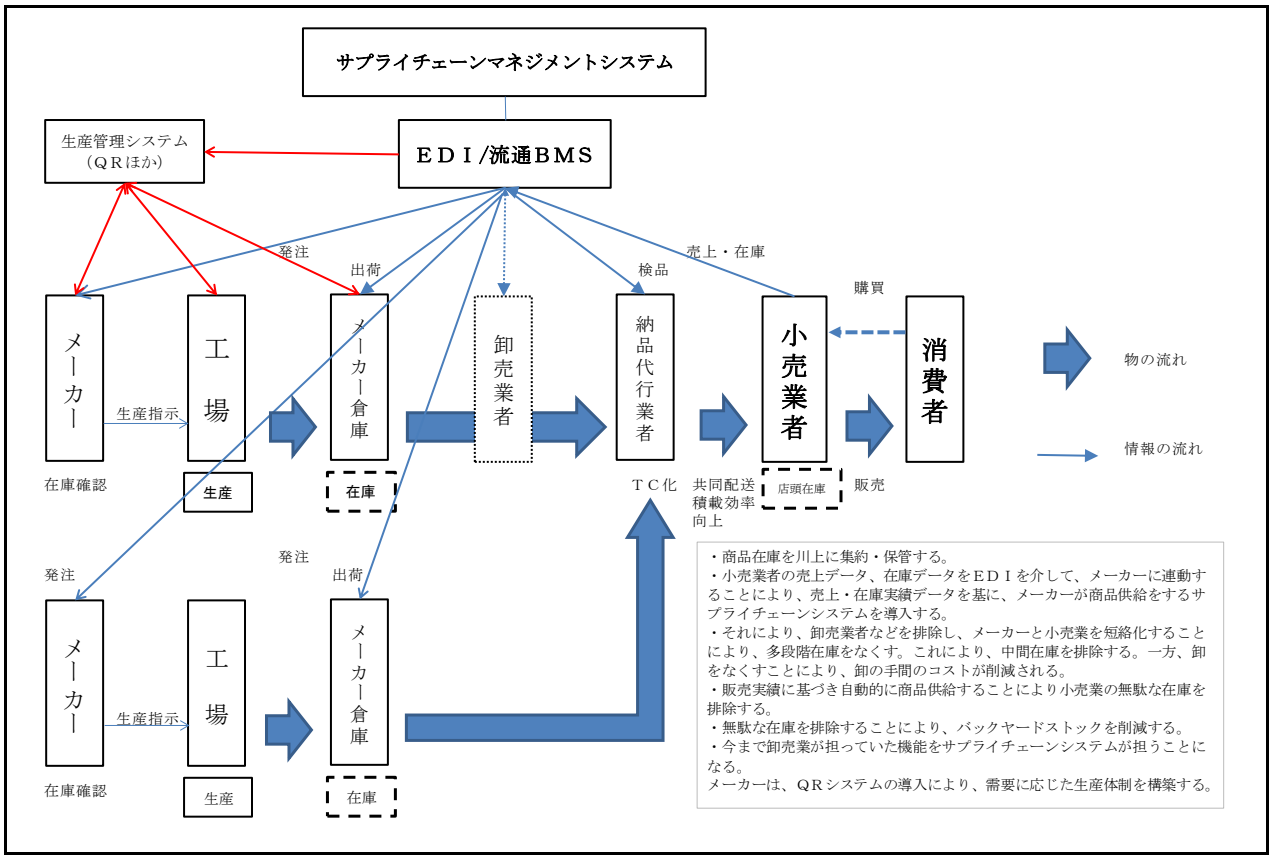

図 4.4　バックヤード削減化サプライチェーンイメージ

6.3 情報共有化の課題

バックヤード削減化を推進するために必須な仕組み, SCM において企業間の情報共有がなかなか進展していない現状がある。

企業間の情報共有がなかなか進まない状況は，インビジブル抑止志向型経営の視点に立って，情報共有化を推進していないのではないかと考える。すなわち「目に見えない」課題，潜在化・内在化する課題，機会損失，チャンスロスにより失われている課題に対し，意識の払われ方が弱いのではないかと考える。バックヤード削減化を推進するために必要な SCM は，機会損失の側面からアプローチする必要があると考える。

経済産業省が主導して，製・配・販三層間の情報共有を円滑にするため，「流通 BMS “Business Message Standards ”」の普及を推進している。流通 BMS によって，小売-サプライヤー間の受発注用基本情報フォーマットを活用して，発注・納品・出荷等の様々な情報を迅速にやりとりすることができる。このように，基盤が整備されつつある。

店頭在庫のみで売場を運営し，バックヤードの在庫を排除すること，すなわち，バックヤード削減化を推し進める為には，製・配・販三層間の情報共有化

が大きなテーマであり，製・配・販三層間の情報共有を推進するためには何が求められているか，このような情報基盤の整備とともに，商慣習や契約などを新しい仕組みに即したものに変化させていく必要があると考える。([図 4.5　小売業と取引先との情報共有化イメージ図]参照)

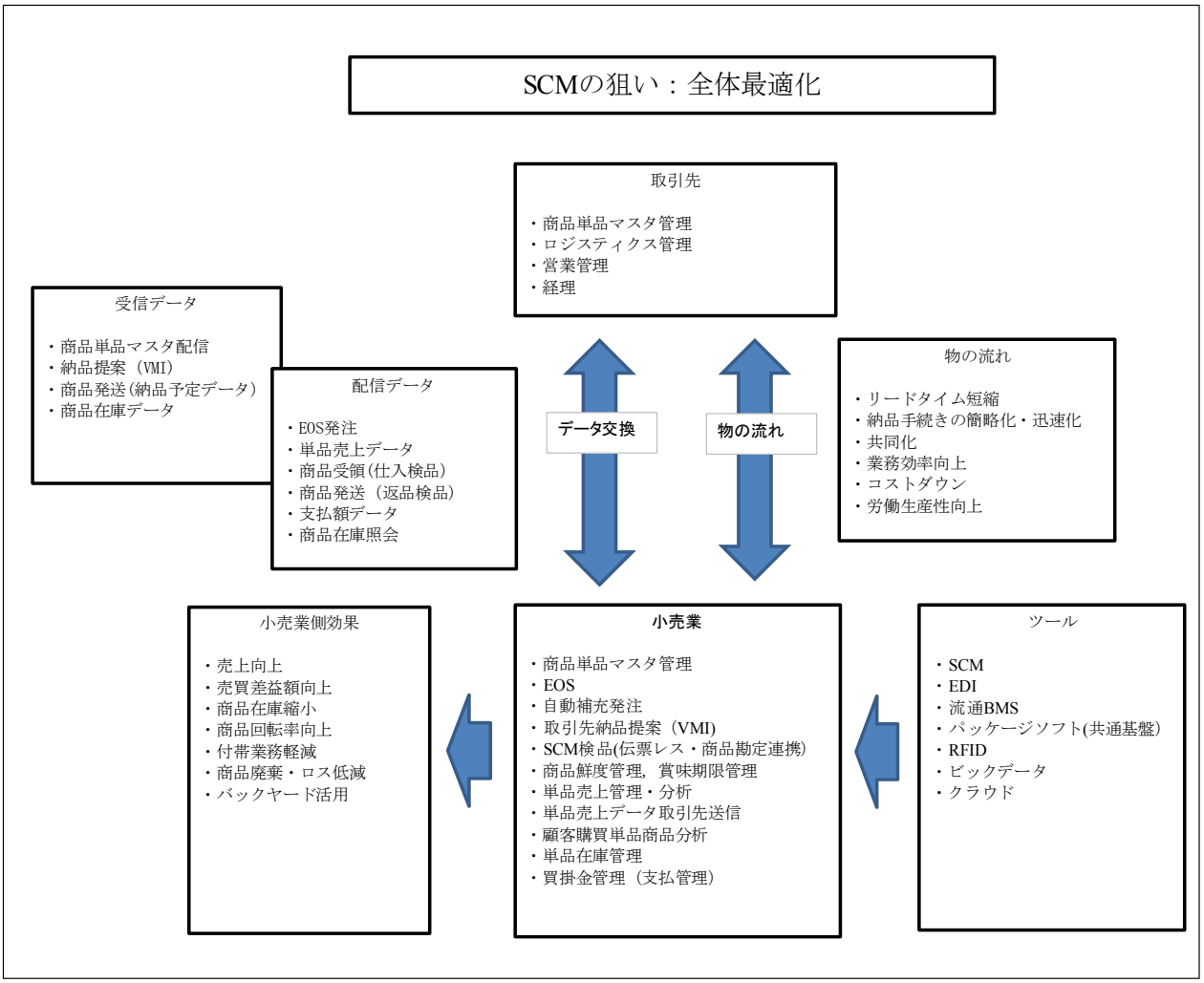

図 4.5　小売業と取引先との情報共有化イメージ図

7. 結論

小売業におけるバックヤードの機会損失に関する基本的研究の結果，すでに投資してある既存店舗を最大限に活かす方策として，利益を生み出していないバックヤード商品ストックを売場に転用する仕組み，バックヤード削減化について論じてきた。

実地調査に基づきバックヤードの機会損失の実態を明らかにするとともに，バックヤードを収益貢献の原資産として積極的な経営資源として活用すること，すなわちバックヤードを売場に転用することにより店舗を新たに設けることや増床することなく，現有店舗面積で売場面積を拡大することが可能となり，それに伴う売上増収額を試算した。

今回の調査結果では，バックヤードストックを売場に転用しないことにより，大型小売店販売額の約 15%，2 兆 9104 億円の売上げが機会損失として失われていることが判明した。また，失われた販売額により約１０万人もの雇用のチャンスが失われていることが判明した。一方，経済効果として 5,496 億円を得られることが判明した。

また，バックヤード削減化により，サプライチェーン全体で発生する無駄を排除することは，環境負荷軽減にもつながる。よって，バックヤード削減化は，企業経営にとって，すでに投資してある既存店舗を最大限に活かす方策であることが明らかになった。

8.おわりに

第 4 章では，大型小売店のバックヤードに焦点をあて，バックヤードを縮小することにより，そのスペースを売場化し，売上を向上させる方策を提案した。

研究の結果，多大な売り上げ増が期待できることが明らかになった。すなわち，ここに機会損失があると言える。今までの考え方にとらわれず潜在する機会損失をなくすことができることが明らかになった。

本研究の課題としては，本研究における基礎数値データの調査標本数が少ない点にある。近年，組織防衛，守秘義務，コンプライランスなどの動きと同調し，社内情報を外部から得ることが非常に難しくなっている。また，有価証券報告書から得られる情報についても，ホールディング化，持株会社化することにより，セグメント情報だけでは，従来得られていた企業情報が得られにくくなったこと研究を通じて理解できた。

今回の研究では，商品群別，企業規模別，売場面積・規模別，都市部と郊外，地方別などの各要素をブレークダウンした分析まで行うことができなかった。今後，要素ごとにブレークダウンした研究を継続することにより，要素ごとの実践的なバックヤード削減化の処方箋を提案することが可能と考える。

参考文献

[1] 生島義英, 佐藤哲也, 唐澤豊, 若林敬造, 小売業におけるバックヤードの機会損失に関する基本的研究ＳＣＭ, 日本ロジスティクスシステム学会誌, 2016 年 3 月
[2] Yoshihide Ikushima, Tetsuya Sato, Yutaka Karasawa, Keizo Wakabayashi, A Basic Research on Opportunity Loss of Back yard,The 11th ICLS2016 Proceeding,July 2016
[3] 生島義英, 佐藤哲也, 唐澤豊, 若林敬造, 小売業におけるバックヤードの機会損失に関する基本的研究ＳＣＭ, 日本ロジスティクスシステム学会第 18 回全国大会予稿集, 2015 年 7 月
[4] 日本オペレーションズ・リサーチ学会辞典,URL: http://www.orsj.or.jp/,(2016 年 1 月確認)
[5] 小学館,デジタル大辞泉, URL: http://www.daijisen.jp/digital/index.html,(2016 年 1 月確認)
[6] 森村英典, 牧野都治, 真壁肇, 杉山高一, 統計・OR 活用事典,東京書籍, 1984 年 9 月
[7] 金融庁, 有価証券報告書, ＥＤＩＮＥＴ,2014 年度決算資料
[8] 経済産業省, 商業動態統計年報, 平成 25 年
[9] 日本スーパーマーケット協会, 平成 26 年スーパーマーケット年次統計調査報告書, 平成 26 年 10 月
[10] 日本ロジスティクス協会, ＪＩＬＳ　2014 年度物流コスト調査報告書, 2015 年
[11] 経済産業省, 我が国流通業の現状と 取組・課題について, 第 1 回産業構造審議会流通部会審議用参考資料, 平成 24 年 4 月
[12] 中川義英,バックヤードレス化システムの経済効果測定と考察, 産業能率大学卒業論文, 1988 年
[13] Yutaka Karasawa, Economical estimation of effects and design model for A backroomless system, Society of logistics engineers,The 26th Annual International Logistics Symposium,1991
[14] 唐澤豊, 現代ロジスティクス概論, ＮＴＴ出版, 2000 年
[15] 下村博史, 知識共有がもたらすサプライチェーンの革新～ 百貨店とアパレルとの協業的取引事例より～, 日本物流学会誌第 12 号, 平成 16 年 5 月
[16] 一般財団法人流通システム開発センター, 流通 BMS.com, http://www.mj-bms.com/, (2016 年 1 月確認)
[17] 燕春栄, 小売業における在庫とその制御に関する研究 , 流通経済大学大学院物流情報学研究科論集 / 大学院物流情報学研究科論集刊行会 編, 流通経済大学大学院, 2003 巻号(6), 2008.9

[18] 燕春栄, 増田悦夫, 小売店舗におけるバックヤードの活用法に関する基礎検討, 日本物流学会誌 , 日本物流学会誌 2008(16), 193-200, 2008
[19] 労働安全衛生広報,業種別 実例に学ぶリスクアセスメント 流通・小売業(バックヤードでの作業) ,労働安全衛生広報 39(919), 22-27, 2007-07-15

第５章 自動倉庫における機会損失の低減に関する基本的研究

1. はじめに

1.1 研究目的

価値の多様化は消費者行動にインパクトを与え，その結果，コンビニエンスストアが台頭するとともに，ＧＭＳやＳＭなどのスーパー業界でも，様々な情報システムが導入され，小売りと取引先とのＥＤＩによる情報データ交換などにより，多品種少量発注をベースにした多頻度即納型流通システムをもたらすこととなった。この傾向はオーダーピッキングを主とする流通センターにも反映し，倉庫や流通センターの自動化と統合管理システムの実現を見るに至ることになる。流通センターの自動化のコアシステムは，立体自動倉庫であり，仕分け機であるが，特に本研究で対象としている立体自動倉庫の汎用型モデルは極めて少なく，かつ法規制を含むコスト機能を現実にビルトインしたモデルは皆無の状況である。一方，シミュレーションについては，依然として G. P. S. S. (General Purpose Simulation System) が主力であり，コスト面の最適化という視点の要素がなく，必ずしも満足のいくものとはいえない。

そのため，立体自動倉庫のラック及び入出庫データを前提として，クレーン台数，ラック段数，ラック連数を様々なパターンで計算してラックコスト，レールコスト，クレーンコスト，ビルディングコスト，およびスプリンクラーコストからなる，初期導入に要する総費用を算出し，当該総費用が最少となる最適設計モデルをより現実的な制約条件下で構築することが求められていると考える。

また，物流業界においては現状，稼働率及び機会損失・投資効率の検証が十分に行われておらず過剰投資傾向にある。これらの課題解決のための検証手法確立が求められていると考える。

本研究では，設定する条件及び要求性能を満たす中でコスト最少の立体自動倉庫を設計する汎用型の最適設計モデルシミュレーションを構築する。その結果，入出荷処理のアルゴリズム・モデルをどのように設定するかにより，クレーンの稼働率が大きく変化し，要求性能を維持しつつ導入費用を低減可能であることを論じる。そして，構築するシミュレーションを基に，入出荷命令の分布と初期導入コスト，稼働率，機会損失額と入出荷命令の分布特性との関係性について明確化することで，投資効率のモデル検証を行うものである。

1.2　先行研究調査

先行研究として，参考文献[2]がある。昭和 63 年に「最適設計モデル」を汎用化した「汎用最適設計モデル」を構築し，このモデルをある 2 社の実データに適用し，40 %近いコストダウンをシミュレーションにより確認した，前回モ

デルに対してファジィ理論を適用し，よりフレキシブルな対応が可能なようにした」といった内容であった。

本研究では，設定した条件及び要求性能を満たす中でコスト最少の立体自動倉庫を設計する汎用型の最適設計モデルシミュレーションを構築するものであり，コスト最少化の視点より構築したシミュレーションを基に，入出荷命令の分布と初期導入コスト，稼働率，機会損失額と入出荷命令の分布特性との関係性について明確化することで，投資効率のモデル検証を行うものである。
よって，新たな研究対象として重要な意味を持っていると考える。

2. 立体自動倉庫の変遷と課題

2.1 物流と自動倉庫

物流は我々の生活面，経済活動面のあらゆる分野の物の流れを対象としたものであり，経済活動の基幹をなす要素である。物流の近代化思想の構築と発展の過程なって，各物流の整合性や合理化の必要性が叫ばれ，その実現化の一つの方策として登場したのが自動倉庫である。それ以前の姿は倉庫内でフォークリフト，コンベアを駆使していたが，こうした平面から縦走方式へ進み，人力から機械へと変化した。

倉庫荷役作業は，人力による荷役から，人間尊重の精神から機械荷役に変化し，パレットを用いたフォークリフト作業が処理能力の向上を図った。

平倉庫の時代から，次に倉庫内に天井クレーンを用いる時代を経て，より効率を高めるためにＸＹＺ軸方向にレールを配置した積み付け用の天井クレーンが開発された。これが自動倉庫のスタッカークレーンの基礎となるものである。

日本初の立体自動倉庫は，昭和 41 年に開発された。立体自動倉庫は，①土地の有効活用，②保管効率の向上，③倉庫作業の省人・省力化，④管理レベル向上による業務の軽減及びコストの削減を目的としており，1970 年頃から建設が進んだ。自動倉庫は，全体の機構とアイデアは明治時代に特許が出願されており，周知の技術であった。当時，アイデアは実用化されず，昭和 40 年代から実用化されはじめた。メーカーのうち数社は先進国と技術提携し，展開するとともに，後発メーカーは研究を続け，結果として特に高速で走行するスタッカークレーンを開発し，参入した。

昭和 40 年代の高度経済成長期には，経済活動が飛躍的に増大する中，人で不足となり，3K 職場と揶揄される産業には人が集まらず，第 3 次産業へと労働力が移行した。労働力対策と企業経営の近代化，合理化のシステムとして，自動倉庫の建設・導入は昭和 40 年代の中ごろから急激に増加し，低成長時代になって個性化・多様化に対応する小型で安価な簡易自動倉庫システムが急増し，平成 25 年度でも年間約 2, 000 台以上の自動倉庫が国内向けに出荷されている。

自動倉庫は，保管機能合理化の代表的なものであり，従来，人が行っていた保管管理作業の省力化・効率化を図り，かつ生産設備の進歩に対応することに

よって倉庫業務の改善と向上を図りことが可能となった。倉庫部門は，企業の中でも重視される機能ではなく，正確に迅速に必要な時に，必要なものを出荷させれば事が足りていたとし，進んで改革を行う部署としては他動的であった。

物流は客先ニーズの側から逆流させて，何が問題であるかを論じる場においても，販売機能の充実，物の輸送の合理化，生産システムとの整合性，品質管理などは研究されても，保管する倉庫については，必要最小限の自家倉庫を持ち，オーバーフロー分は工場の片隅に置かれるのが企業の中で主流であった。

しかし，昭和 40 年代から自動倉庫を設置して，企業経営の主要な合理化の柱として，製品コストの競争力を得るための先行投資として考えられるようになってきた，特に著しい事例としては，自動車業界が世界一の座を目指して生産と販売体制を充実してゆく発展途上にあって，納入後の保守部品を他企業より少しでも早く供給するために，製品の品質管理と同程度に重視し，ＩＴを利用して倉庫機能のレベルアップと在庫管理を強化し，本社部門からの指令を事前にデータインプットし，休みなくスタッカークレーンを働かせ，必要な部品を出庫させる体制を設け，合理化効率化した在庫管理データを基にメンテナンス部品のみならず，全社の体質強化に寄与させた。

また，流通業界では従来の倉庫機能は改革させ，その結果，低成長時代にあっても多品種少量の客先要求に対して，対応できる体質を構築している。

2.2 立体自動倉庫の現状

立体自動倉庫が我が国に現れたのは昭和 40 年代初頭であり，以来 40 年以上が経過した。当初はその構造から

① 格納効率のの向上
② 正確な在庫管理
③ 迅速な入出庫
④ 肉体労働の軽減化
⑤ 敷地の有効活用
⑥ 作業の安全化

などが挙げられる。その後の我が国の経済発展に伴い，年間 1 千億規模の市場に成長した。この要因は多くあるが，以下にあげられる要因により進展してきたと考える。

ＦＭＳ：フレキシブル生産システム，ＦＡ：ファクトリーオートメーション，ＣＩＭ：コンピューター統合生産システムおよびＣＡＬＳ（Continuos Acquisition and Lifecycle Support）などの企業競争力強化の問題は常にシステム化であった。計算機システムを標準装備している立体自動倉庫はそのシステム化投資の一貫として位置づけられ，積極的に導入されてきた。

我が国の経済特徴として一人当たりの GNP の飛躍的増大，労働人口の第 3 次産業への移転，高齢化社会への移行に伴い，省力化省人化のニーズは高く，と

りわけ３K 職場の多い物流現場では，自動化の要求が大きい。立体自動倉庫は，ピッキング・搬送・仕分など他の物流設備と融合したシステムの中で中心設備としてこれに応えてきた。

立体自動倉庫の設置形態としては，建物内に設置する小型のユニット式と，建物として自立するビル式の 2 種に大きく分けられる。また運用形態としては，パレット積みされユニット化された荷を扱うパレット用と，通箱・バケット・カートンを単位とするバケット用に分けることが可能である。高度経済成長を経て，1980 年代にはパソコンとパッケージソフトを使用した比較的廉価な簡易自動倉庫システムが普及し，1990 年代には年間 1,000 台，そして 2013 年にはついに年間 2,000 台のシステムが納入・設置・建設されるに至った。「表 5.1」及び「図 5.1」，「図 5.2」，「図 5.3」に，年間の自動倉庫システム売上高，台数，パレット数の推移を示す。

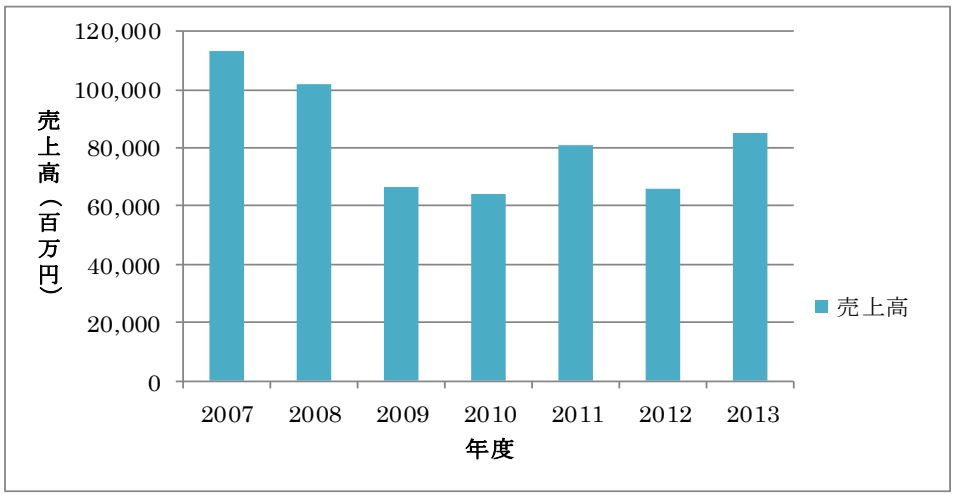

図 5.1　自動倉庫売上高の推移[7][8][9][10][11][12]

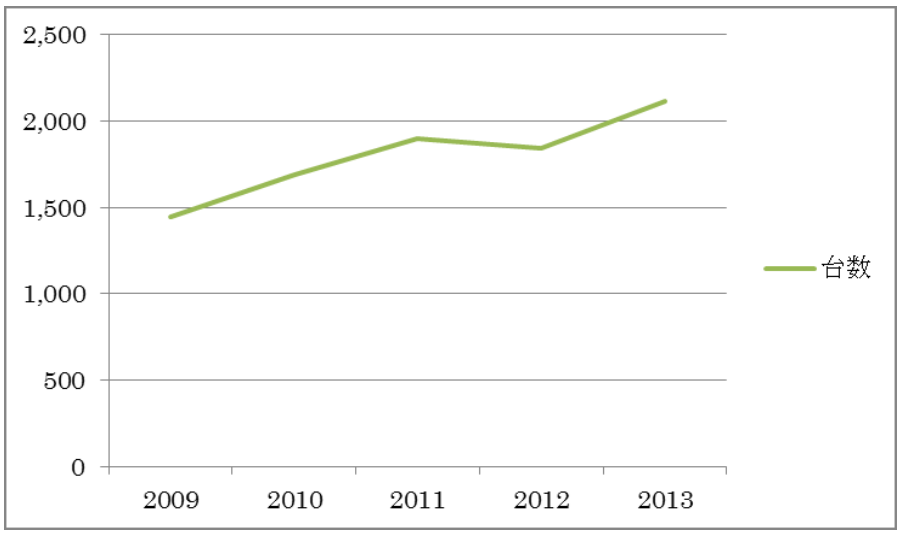

図 5.2　自動倉庫システム台数[7][8][9][10][11][12]

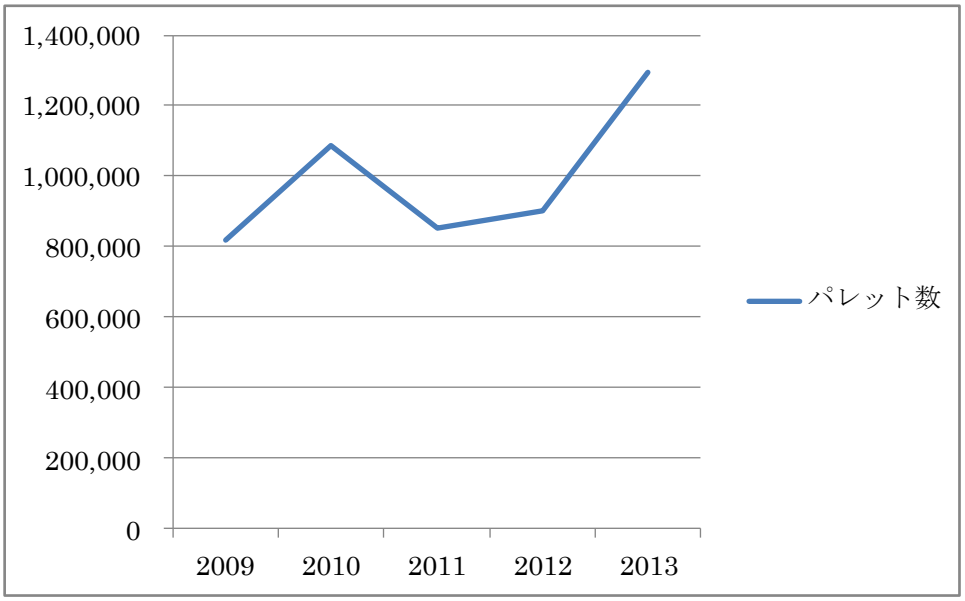

図 5.3　自動倉庫パレット数の推移[7][8][9][10][11][12]

表 5.1　自動倉庫売上高・台数・パレット数の推移 [6][7][8][9][10][11][12]

（単位：百万円）

年度	2007	2008	2009	2010	2011	2012	2013
売上高	113,251	101,851	66,329	63,982	80,623	66,144	85,305
台数	-	-	1,445	1,691	1,900	1,842	2,118
ﾊﾟﾚｯﾄ数	-	-	818,143	1,087,956	852,969	901,401	1,294,601

このように現代において，立体自動倉庫システムは物流の高効率化，経営の合理化のために必要不可欠なものとなっている。しかし，上述したように立体自動倉庫のラック及び入出庫データを前提とした，初期導入費用最少となる最適設計モデルの構築，稼働率及び機会損失・投資効率の検証は十分に行われておらず，過剰投資傾向にあるといえる。

立体自動倉庫は，パソコンの普及及びその他各種技術革新により，登場初期と比較すれば廉価なシステムになった。規模にもよるが 1 台あたりの導入に数千万円から数十億円を要するシステムであり，高額な初期投資が必要であるにも関わらず，その投資効率を向上させるための施策，そしてその効果に関する検討は，分野においてほぼ皆無といっても過言ではない。

このシステムでは，いかにして命令分布をコントロールするかが，効率的に機能させる上で重要な課題である。しかし，現実の倉庫では，入出荷の命令分布にいくつもの波が存在する。1 日の中であっても入出荷は朝夕などに集中し，さらに商慣習を理由とする月間波動，そして年間波動をも考慮しなければならず，ピークの要求処理能力を基にして立体自動倉庫の設計を行った場合には，当然の結果であるが稼働率が下がる。すなわち非稼働率が上がり，投資額に対する機会損失も大幅に増大することとなる。

3. 立体自動倉庫のコスト最少化

3.1 コスト最少化計算条件

本研究で計算機シミュレーションモデルの構築を行う対象は，T11 規格のパレットを使用したビル式の立体自動倉庫とした。ただし，パラメーターの設定を変更することで，ユニット式，バケット式の立体自動倉庫設計にも応用可能である。本モデルでは，ラック 2 列毎にクレーンが昇降・走行するスペースを設ける。計算モデルにおけるラック配置図を「図 5.4」に示す。そして，入荷と出荷の処理を 1 度のクレーンの動作で行う“デュアルサイクル動作”を原則とする。シミュレーション計算に必要な各種条件は，自動倉庫メーカー (株)ダイフ

クより提供いただいた概算コスト・仕様データをもとに設定する。「表 5.2」に示すとおりである。

商品は ABC 分析に基づいて事前にグループ分けすることを想定し，予め初期在庫，入荷，出荷についてリストを乱数により生成する。まず出荷リストの順序に従い，該当する商品のうち最もロット番号の若い商品を出荷するとする。次に入荷先ラックをクレーンのアクセスタイムが最小となるように決定，入出荷を一度のクレーン動作で行う，デュアルサイクル動作のシステムとしてモデルを構築する。「図 5.5」に示すとおりである。

立体自動倉庫のコストは以下の5つのコストから構成されている。

① ラックコスト
② クレーンコスト
③ レールコスト
④ スプリンクラーコスト
⑤ 耐震床コスト

以下に各コストの計算式を示す。

① ラックコストC_O

$$C_O = n \times K_O = rd \times (2 \times cn) \times rl \times K_O \quad \cdots\cdots\cdots (5.1)$$

② クレーンコストCR

$$CR = cn \times Kr \quad \cdots\cdots\cdots (5.2)$$

③ レールコストC_Z

$$C_Z = cn \times (rl \times RL) \times K_Z \quad \cdots\cdots\cdots (5.3)$$

④ スプリンクラーコストC_S

$$C_S = (cn + 1) \times (rd / 2) \times rl \times K_S \quad \cdots\cdots\cdots (5.4)$$

⑤ 耐震床コストC_F

$$C_F = ((3 \times cn \times RW) \times ((rl + 1) \times RL)) \times K_F \quad \cdots\cdots\cdots (5.5)$$

使用記号

Co：ラックコスト(円)

n：ラック数(個)

Ko：ラック1個あたりのコスト(円)

Rd：ラック段数(段)

Cn：クレーン台数(台)

Rl：ラック連数(連)

Kr：クレーン1台あたりのコスト(円)

Kz：レール1 mあたりのコスト(円)

RL：ラックの長さ(m), rl×RLは小数点以下切り上げ

KS：スプリンクラー1基あたりのコスト(円)

ただし，倉庫面積≧1, 400 m^2かつラック全体の高さ ≧ 10 mの場合のみ, rd / 2は小数点以下切り上げ

RW：ラック幅(m)

KF：耐震床1m2あたりのコスト(円)

3×cn×RW：倉庫幅(クレーン含む)(円)

(rl + 1)×RL：倉庫長さ(入出荷口含む)(円)

((3×cn×RW)×((rl + 1)×RL))：面積(面積は小数点以下切り上げ)(m)

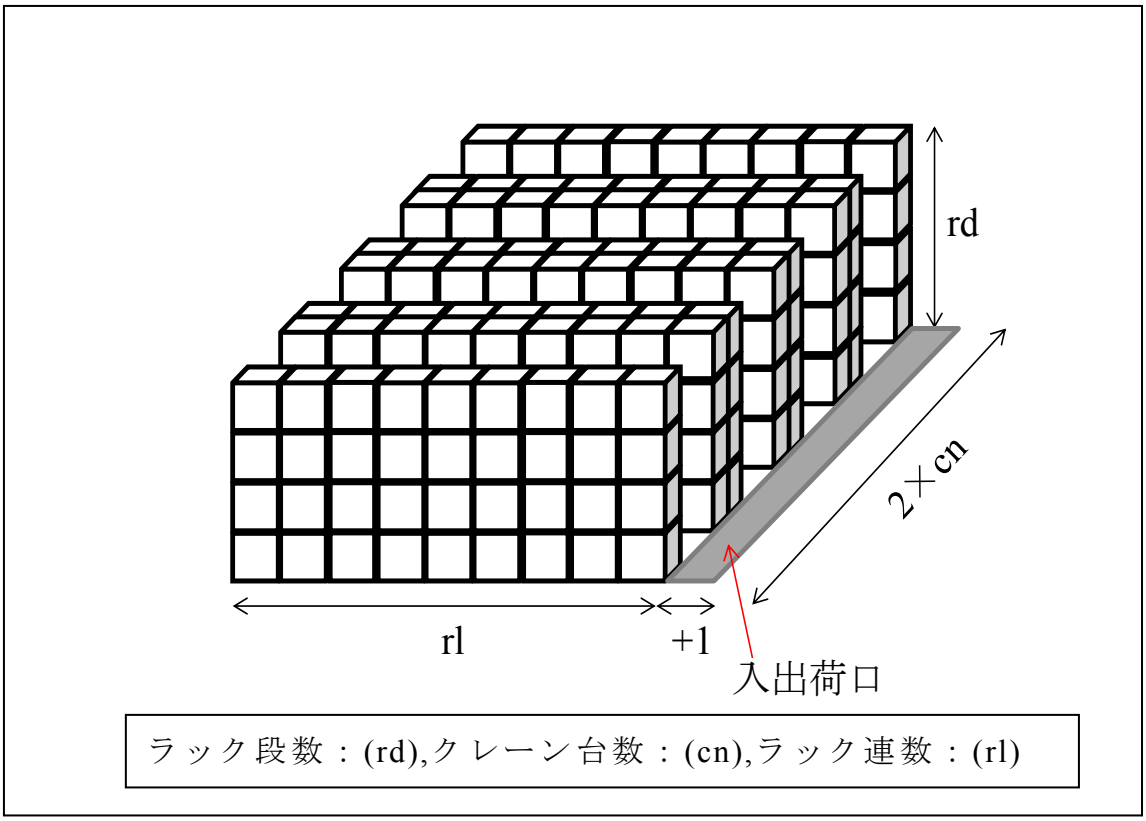

図 5.4　計算モデルにおけるラック配置図

表 5.2　シミュレーション条件 (速度・寸法・条件・概算コスト)

項目	条件	コスト〔万円〕	規格	備考
ラック	高さ10m以下	4	荷姿　1100×1100×1200mm	1ラックあたり
	高さ20m以下	4.5	重量パレットこみ800kg	
	高さ30m以下	5		
クレーン	高さ10m以下	1000	10m　走行120m/分　昇降　30m/分　フォーク　20m/分	1台あたり　本体制御付き
	高さ20m以下	2000	20m　走行160m/分　昇降　40m/分　フォーク　20m/分	
	高さ30m以下	3000	30m　走行200m/分　昇降　60m/分　フォーク　20m/分	
レール	高さ10m以下	10		mあたり
	高さ20m以下	15		
	高さ30m以下	20		
スプリンクラー			スプリンクラーは　2段に1箇所	面積1400㎡以下　いらない
	高さ3m、平面2ラックに1台	8		高さ10m以下いらない
耐震床	1000㎡以上	8		㎡あたり　高さ20m以下に利用
	1000㎡以下	10		これ以上高いと建物側で対応

注）荷姿　1100×1100×1200mm　高さ　重量パレットこみ800kg
　　建物は、別途とする。
　　ダイフクへの調査結果

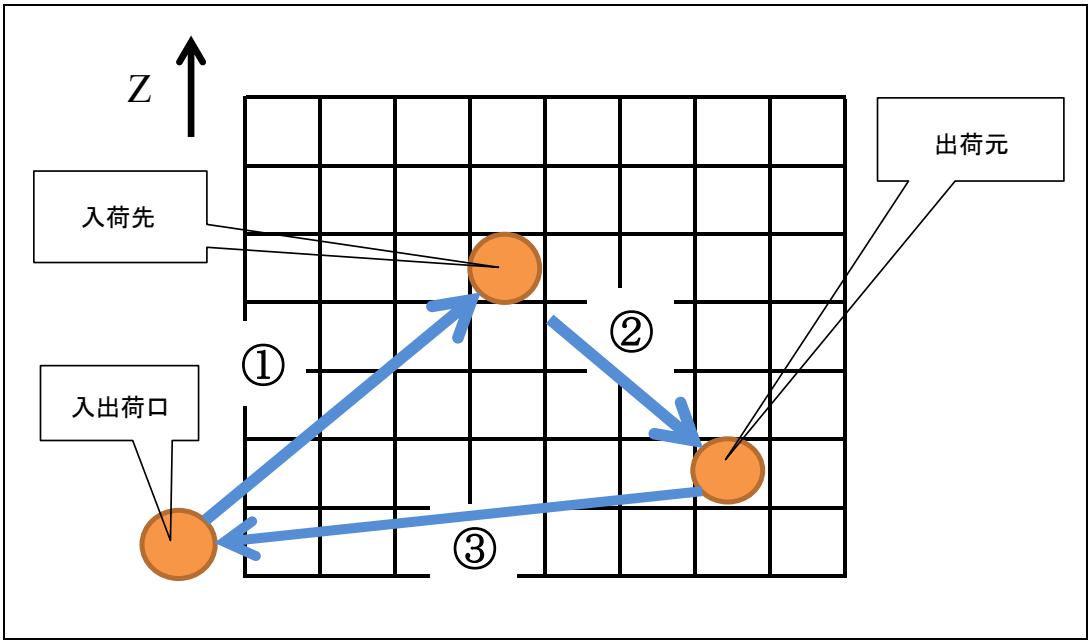

図 5.5　デュアルサイクル動作の概念図

ここで，ラックは[rd][2×cn][rl]の 3 次元配列とし，rd ラック段数，cn: クレーン台数，rl: ラック連数，RH: ラック高さ，RW: ラック幅，RL: ラック長さとする。

「図 5.6」にシュミレーションプログラムの基本フローチャートを示す。また，シミュレーションプログラムは，Microsoft Visual Studio 2012 Update 5 にて

構築を行った (Visual C++ / OpenMP)。また，計算環境は, CPU: AMD Phenom II X6-1090T (6 コア 6 スレッド, 3.2 GHz), メインメモリ: DDR2 PC2-6400 SDRAM 16GB, OS: Microsoft Windows 7 x64 (SP1) とする。

本項では，自動倉庫設計のための要求として，ラック総数 4, 000 個，使用面積 1, 800 m^2 以内，ラック積み上げ高さ 30 m 以内 (法令による制約) を設定しており，初期在庫商品数は 1, 000 (在庫充填率 25 %)とする。また，入出荷リストにおける命令数はそれぞれ 2, 000 とし，これら入出荷処理が 1 時間で完了することを，要求される処理性能と設定する。

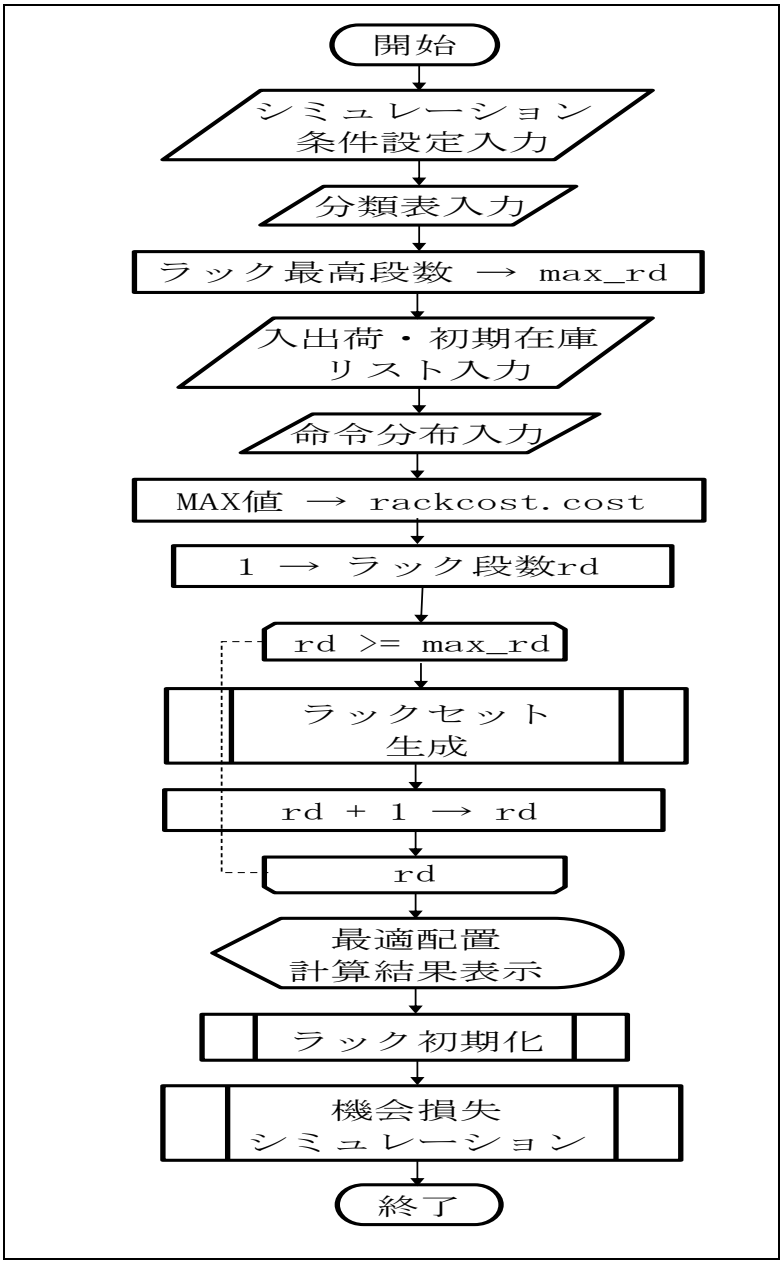

図 5.6　シュミレーションプログラム基本フローチャート

3.2 固定ロケーション型自動倉庫

はじめに，固定ロケーション型の立体自動倉庫モデルについて検討を行った。「図 5.7」に，固定ロケーション型立体自動倉庫におけるラックへのグループ割付の概念図を示す。

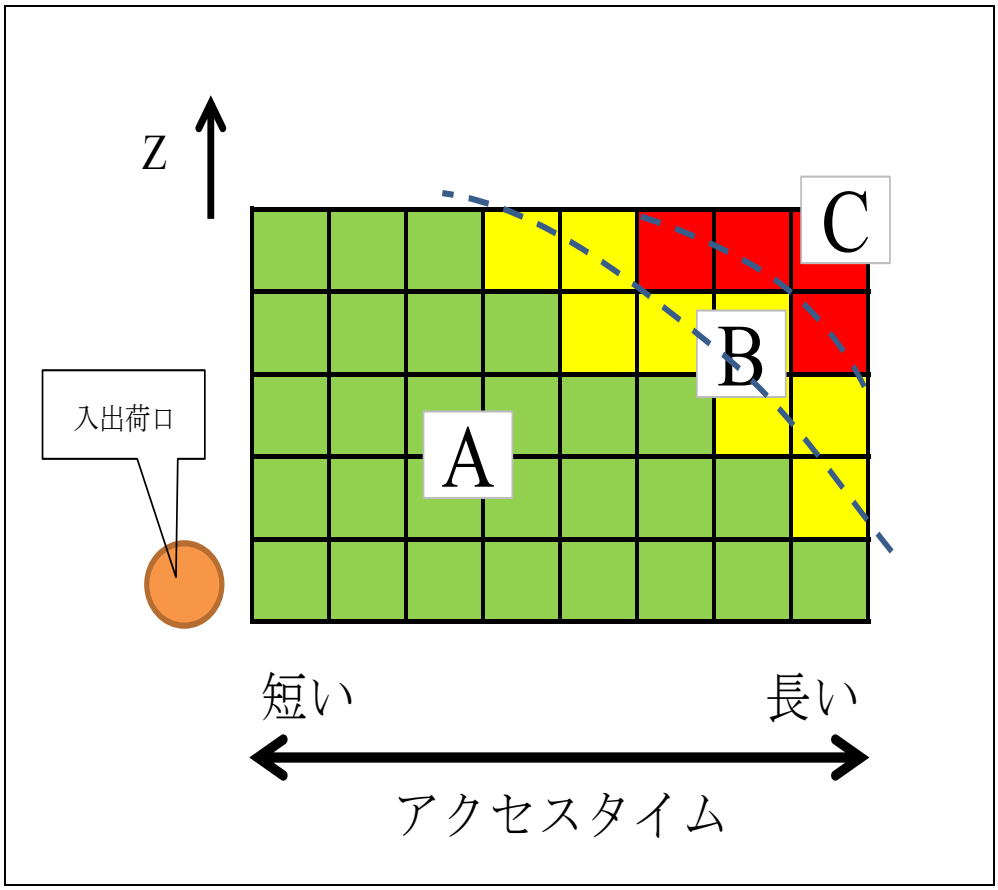

図 5.7　固定ロケーション型倉庫の概念図

クレーンが全て独立に動作可能な理想的状態では，入出荷オペレーションの完了に要する時間 S_T は各クレーンの処理時間 C_T[i]を用い，以下のように表すことができる。

$$S\_T = \frac{1}{n}\sum_{i=1}^{n} C\_T[i] \qquad \cdots\cdots\cdots(5.6)$$

使用記号

S_T：入出荷オペレーションの完了に要する時間(min)

C_T[i]：各クレーンの処理時間(min)

このとき，ラックの配置が[6][52][13]，クレーン台数 26 台，使用面積 1751.43 m^2 の条件において，要求処理能力を満たす中でのコスト最少となった (60, 920 万円)。なお，各 2, 000 件の入出荷オペレーションを完了するまでに要する時間は 54.32 min.であった (シミュレーション計算時間: 60, 839.53 ms)。

しかし，本結果は複数台のクレーンの並列動作性について一切考慮しておらず，全クレーンの処理時間の平均をオペレーション完了までの時間とする場合である。そこで，先入れ先出し (FIFO: First In, First Out) の原則のもと，入出荷オペレーションを各リストの順序通りに実行することを前提とし (In-Order 処理)，クレーンの並列処理性を考慮した処理時間算定を行うこととする。入荷先ラックは，入出荷処理のセット毎に倉庫内のラック全てを端からラスタスキャンし，アクセスタイム最短のラックを選択している。「図 5.8」に示すとおりである。

このとき，要求を満たしつつコスト最少となる条件は，ラックの配置が[7][148][4]，クレーン台数 74 台，使用面積 1,780.30 m^2 であった (197, 156 万円，処理時間 55.35 min.，シミュレーション計算時間 56,653.93 ms)。

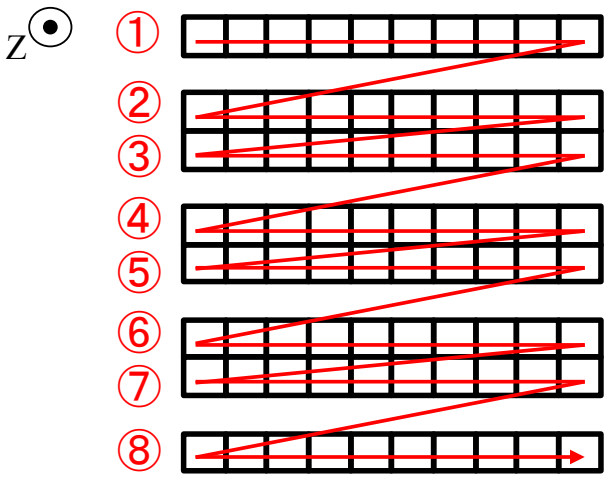

図 5.8　ラスタスキャンによるアクセスタイム最短の入荷可能ラックの探索

3.3 クレーン並列処理

本計算モデルでは，出荷元ラックは商品種別とロット番号によって一意に決まるため，そのとき使用するクレーンの担当範囲ラックにて ABC のグループが割付された空きラックのうちアクセスタイム最短の場所を入荷先ラックとしている。該当範囲に空きラックがない場合には，入出荷処理を別のクレーンで行うものとしている。つまりこのモデルにおいては，出荷元ラックの決定がクレーンの並列動作性に最も影響する。そこでまず，入荷先ラックの決定アルゴリズムについて改良を検討する。

入出荷口からのアクセスタイムが等しいラックは，倉庫全体で 2×cn 個存在するが，ロット番号が連続する同一商品を同じクレーンが担当するラックに格納する場合，入出荷処理においてクレーンの並列動作に支障が生じ，待機が必要となると考えられるためである。

つまり，アクセスタイムが等しいラックも，全てを等価として扱うべきではないといえる。そこで，倉庫内のラックを 2×rl×rd = 1 単位として扱い，「図 5.9」のように走査して入荷先ラックを決定する改良を行った。これにより，ロット番号の連続する同一商品が同一クレーンの担当するラックに格納される確率が低減され，クレーンの並列動作性が向上すると期待できる。

この時に要求を満たしつつコスト最少となる条件は，ラックの配置が [10][80][5]，クレーン台数 40 台，使用面積 1,154.79 m^2 であった (111, 440 万円，処理時間 59.49 min.，シミュレーション計算時間 57,086.56 ms)。「図 5.10」に，前述の①理想的完全並列処理，②並列性を考慮しない入荷処理と③並列性を考慮する入荷処理における初期導入コスト，クレーン台数，要求を満たすラック配列パターン数，シミュレーション計算時間の推移を示す。このときの「図 5.10」中③における初期導入コストは，前述のラスタスキャンによる入荷可能ラック探索アルゴリズム (図 5.10 中②) と比較し，約 43%の低減が見られた。

このように，入荷先ラックの決定方法を変更するだけで，同じ要求性能を満たしつつ，自動倉庫全体の導入コストが大きく変化することが明らかとなった。しかし，各クレーンが完全に並列動作可能な理想状態 (図 5.10 中①) と比較すると，依然として 2 倍近いコストを要していることがわかる。そこで次に，入出荷処理の順序最適化について検討を行った。

ここでは，出荷リストについては順序を変えず，上述の通りにロット番号の若い商品から順に出荷するものとする。そして，入荷については，クレーンの台数 cn を上限とし，リスト内の順序を入れ替えることを検討する (Out-of-Order 処理)。これは，入出荷口のベルトコンベアにて入荷商品をクレーンまで運ぶ際，その先を調整することに相当する。ここで，入荷リストのうち入れ替えを検討する単位を n とすると，シミュレーションにおける計算量は nPn の順列 (Permutation) となるため，厳密には n!回の計算が必要となる。しか

し, ABC 分析に基づく分類が 3 種類しかないため, 実質的には 3×n 回の計算回数で概ね等しい結果を得ることができる。

上記条件において, 入荷リスト内の順序入れ替えを検討する範囲 n を変化させたときの, 各入荷先ラック決定アルゴリズムにおける自動倉庫の初期導入コストの推移を「図 5.11　」に, シミュレーション計算時間の推移を「図 5.12」に, 要求を満たすパターン数の推移を「図 5.13」に示す。

結果より, クレーンの並列処理性を考慮していない入荷処理アルゴリズムにおいては効果が大きく見られるものの, Out-of-Order 処理範囲を拡大するに従って計算時間が指数関数的に著しく増大することが確認できる。入出荷処理でクレーンが動作している間にリアルタイムでこのような入出荷順序最適化のシミュレーション計算を行うことを考えると, 「図 5.12」より 6 命令程度の最適化が限界であり, 計算機の性能への要求に対し効果が少ないことから, 現実的な手法ではないことが言える。

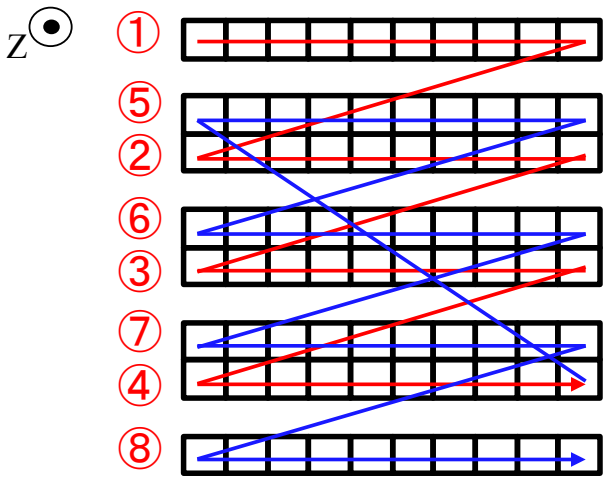

図 5.9　ラックを 2×rl×rd = 1 単位として扱った
アクセスタイム最短の入荷可能ラックの探索

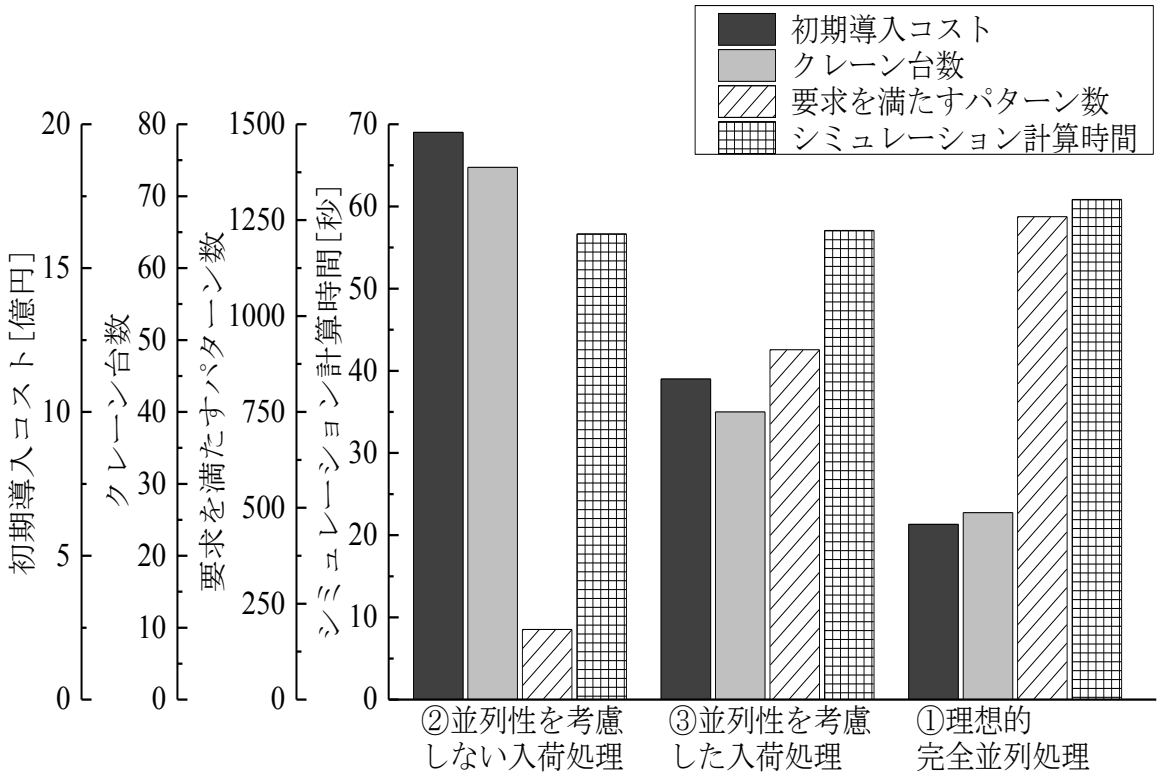

図 5.10　入荷先ラック決定法による初期導入コスト，クレーン台数 (倉庫規模)，要求を満たすラック配列パターン数，シミュレーション計算時間の差 (固定ロケーション型)

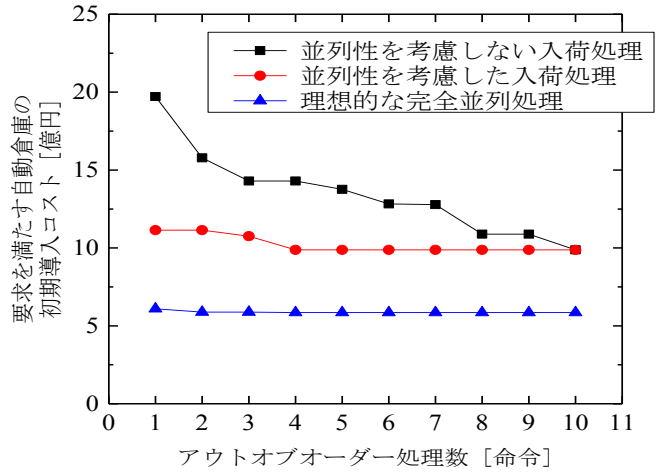

図 5.11 立体自動倉庫の初期導入コストの Out-of-Order 処理範囲依存性

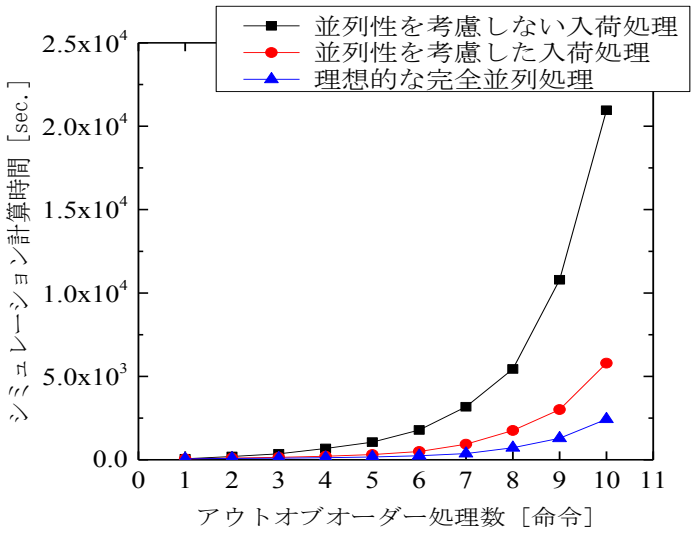

図 5.12 立体自動倉庫設計シミュレーション計算の Out-of-Order 処理範囲依存性

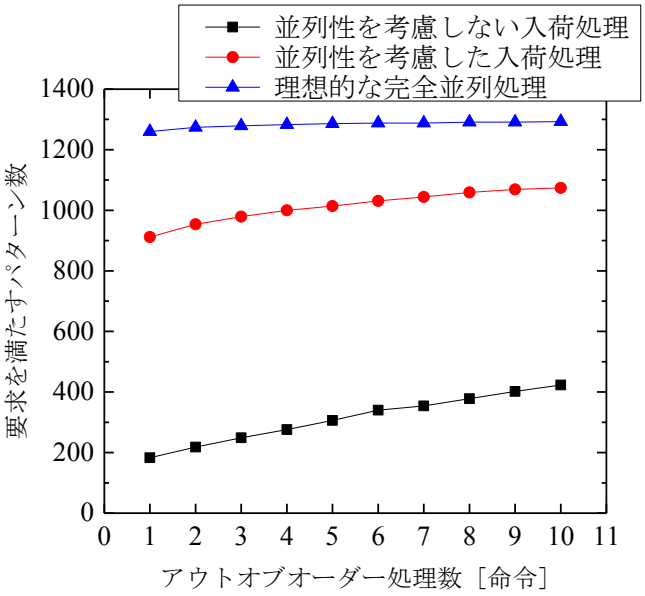

図 5.13 立体自動倉庫設計シミュレーション計算において要求を満たすパターン数の Out-of-Order 処理範囲依存性

3.4 フリーロケーション型自動倉庫

ここまで，ABC分析に基づくラックへのグループ割付を行う固定ロケーション型の自動倉庫を対象に，入出荷アルゴリズム改良に関するモデル検討をシミュレーション計算により行ってきた。ここでは，ラックへのグループ割付を行わないフリーロケーション型の立体自動倉庫について検討を行う。

フリーロケーション型の自動倉庫は，どの棚にどの品目を保管しても良い運用であり，入荷商品は空いている場所に保管していく。この方式は大手通信販売事業者である Amazon などの，バラピッキング (ピースピッキング) 型の物流センターで採用されている方式である。

ここでは，前節の検討と同様，①理想的な完全並列処理，②並列性を考慮しない入荷処理，③並列性を考慮する入荷処理について検討を行う。なお，入荷リストのOut-of-Order処理については，効果が理論上ほぼ見込めないため実施しなかった。

①の理想的な完全並列処理において要求性能を満たしつつコスト最少となる条件は，ラックの配置が[6][38][18]，クレーン台数 19 台，使用面積 1,737.00 m^2 であった (54,062 万円，処理時間 56.92 min.，シミュレーション計算時間 28,164.09 ms)。

②の並列性を考慮しない入荷処理条件において要求性能を満たしつつコスト最少となる条件は，ラックの配置が[12][86][4]，クレーン台数 43 台，使用面積 1,034.50 m^2 であった (116,726 万円，処理時間 58.99 min.，シミュレーション計算時間 27, 637.66 ms)。

そして，③の並列性を考慮する入荷処理条件にて要求性能を満たしつつコスト最少となる条件は，ラックの配置が[6][68][10]，クレーン台数 34 台，使用面積 1799.55 m^2 であった (69,480.0 万円，処理時間 54.57 min.，シミュレーション計算時間 27,056.86 ms)。

これら結果をまとめたものを，「図 5.14」に示す。結果から，本設定条件では，並列性を考慮する入荷処理 (図 5.14 中③) を行うことで，理想的な完全並列処理(図 5.14 中①) と比べても 28 %の初期導入コスト増で済むことが明らかとなった。これは，固定ロケーション型自動倉庫における入荷順序の最適化 (Out-of- Order 処理) と比較しても約 30%のコスト減であり，効果が高いものであるといえる。

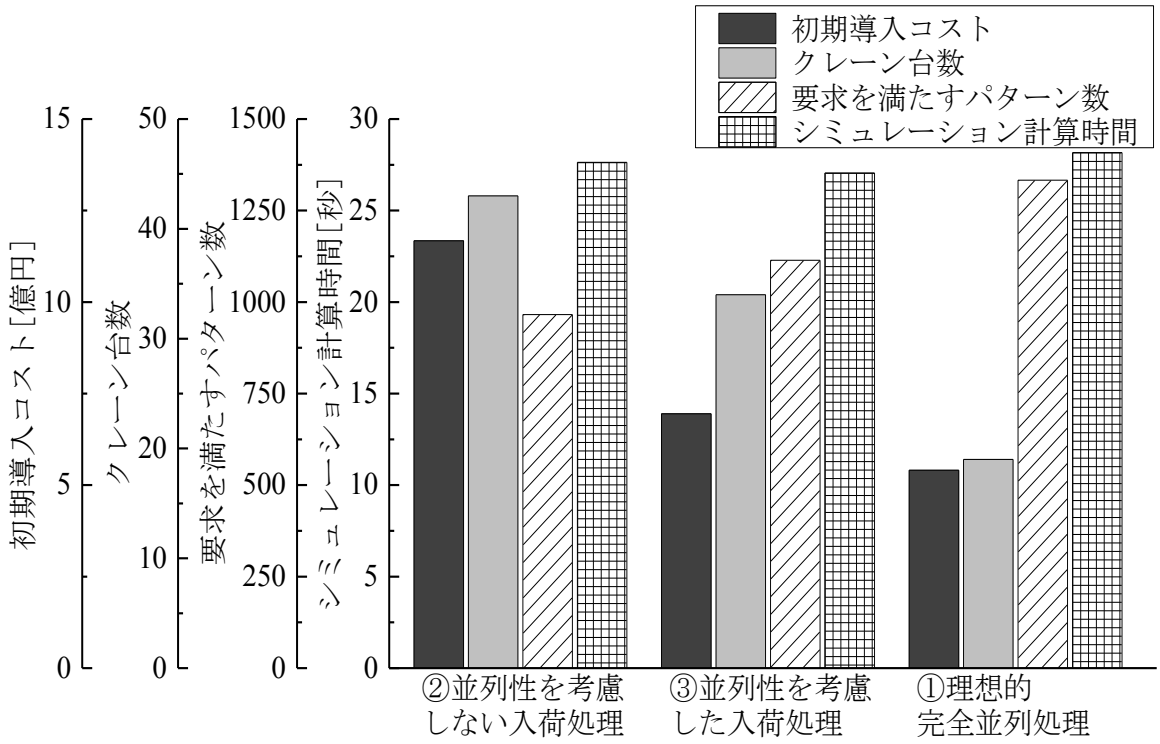

図 5.14 入荷先ラック決定法による初期導入コスト，クレーン台数 (倉庫規模)，要求を満たすラック配列パターン数，シミュレーション計算時間の差 (フリーロケーション型)

しかし，一般的には固定ロケーション型の方が出荷時の動作効率が高いとされており，初期在庫条件 (充填率)，入出荷リストの条件，要求性能の設定によって，今回の結果は変化すると考えられる。本項で使用するテストデータは，在庫充填率が 25 %と低く，多頻度入出荷グループ以外に割り付けられたラックまでクレーンが移動するよりも，割り付けを行わずにアクセスタイムの短いラックへ入荷する方がよい条件であり，普遍的な結果とは言えない。

そこで，要求性能を満たす自動倉庫の初期導入コストについて，初期在庫の格納数 (充填率) 依存性を検討する。在庫充填率を 25%～95%の範囲で変化させ，固定ロケーション型，フリーロケーション型の双方について，ラックを 2×rl×rd = 1 単位として扱ったアクセスタイム最短の入荷可能ラックの探索アルゴリズムにて計算検討を行った。

結果を「図 5.15」に示す。計算結果では，それぞれ在庫リストを乱数生成したため，在庫充填率の変化に対し前後での増減が大きくなっているが，いずれの場合も固定ロケーション型とフリーロケーション型では同じ在庫リストを使用している。

結果より，在庫充填率 70 %以下では，差の大小はあるが，固定ロケーション

型よりもフリーロケーション型の方がコスト的に有利となる結果を得ることが出来た。また, 在庫の充填率に対し固定ロケーション型ではコスト的に差がほとんど見られず安定しているのに対し, フリーロケーション型ではコストに大きな変化が現れている。自動倉庫の設計時には, 将来の需要拡大を想定して要求能力・処理性能・格納用ラックを高め・多めに設定することが一般的であるが, 運用初期において在庫充填率が低い場合にはフリーロケーション型で運用し, 需要量の増大とともに固定ロケーション型への転換を行うことも, 稼働率向上の観点では有効であると, この結果は示している。

なお, 本項で使用する在庫・入出荷リスト及び分類表は, A : B : C = 7 : 2 : 1 の比とし, それぞれ 2 品種の合計 6 品種しか設定していない。固定ロケーション型が有意性を示すのは取扱品種が多い場合であり, パレート特性に従った品種の分類を行った場合であることを, 併せて記しておく。

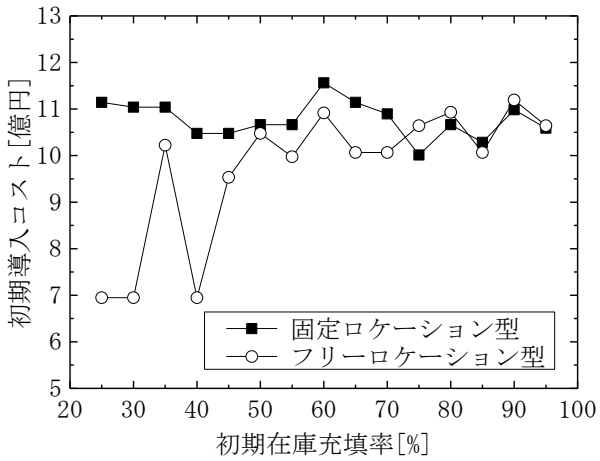

図 5.15 固定ロケーション型とフリーロケーション型の立体自動倉庫における初期導入コスト (最少) の初期在庫充填率依存性

3.5 運用期間の推移による稼働率低下の検証

ここまで, 初期在庫を格納した直後の入出荷処理から, コスト最少となる立体自動倉庫の自動設計モデルについて述べてきた。しかし, 現実の運用においては, 倉庫のラック内に格納されている在庫の状態が逐次変化する。そのため, 自動倉庫設置後の初期在庫格納状態から, 運用期間が推移することにより処理性能の低下がどの程度引き起こされるのか, 検証を行う必要がある。1 日の稼働時間を 8 時間とし, 1 週間 (7 日間) に相当する 56 時間分, 要求される入出荷命令数の負荷をかけ続けた場合の, クレーンの稼働率について検証を行った。

シミュレーション条件は, ここまでと同じくラック総数 4,000 個, 使用面積

1,800 m^2 以内, ラック積み上げ高さ 30 m 以内 (法令による制約) と設定する。初期在庫商品数は 1,000 (在庫充填率 25 %) とする。また, 入出荷リストにおける命令数はそれぞれ 2,000 とし, これら入出荷処理が 1 時間で完了することを, 要求される処理性能と設定する。

結果を「図 5.16」に示す。自動倉庫の導入時点において稼働率が 100 %以内に収まることを条件として設計したモデルの場合, 固定ロケーション型, フリーロケーション型を問わず, ラック内の商品の格納状態がランダムに近づいていくにつれ, 徐々に稼働率が増加する。すなわち, 1 時間以内に完了すべき入出荷命令を処理しきれない状態となることをこの結果は示している。そしていずれの場合にも, 運用時間 10 時間程度でほぼ安定した運用が可能となることがわかる。なお, フリーロケーション型と比べ固定ロケーション型の方が, 稼働率の上昇 (処理性能の低下) が少ない。つまり, 入出荷頻度の大きく異なる商品群を対象とする場合には, 固定ロケーション型の方が安定した運用が可能である。

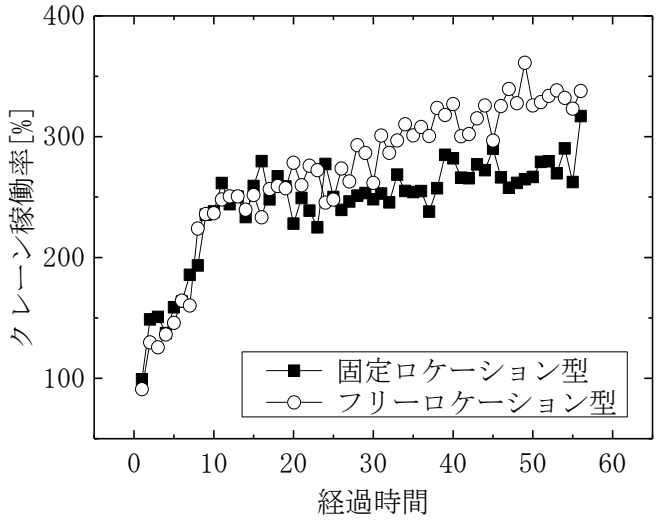

図 5.16 固定ロケーション型とフリーロケーション型の立体自動倉庫におけるクレーン稼働率の運用時間依存性

4. 立体自動倉庫における機会損失

4.1 機会損失の検討

3 項では, 設定した要求処理性能を満たしつつ初期導入コスト最少となる立体自動倉庫をモデル設計するためのシミュレーションを構築し, 入荷先ラック決定アルゴリズムの差異による初期導入コストの変化について検討を行った。本項では, 上記手法を用いてモデル設計する立体自動倉庫における稼働率, 非稼働率, そして機会損失について検討を行う。

前項3.5における結果 (図5.16) からも明らかなように, 最良の条件における運用開始直後の処理性能をもって倉庫の設計モデルにおける処理性能とすべきではない。そこで, 上記検証と同様, 8時間 (1日の稼働時間) ×1週間 (7日) = 56時間分, 入力する入出荷命令分布のピーク値となる命令数を処理し続けることが可能であることを要件として設定する。

命令分布特性を生成するにあたり, 現実に稼働している立体自動倉庫を対象に, 倉庫規模と入出荷頻度の対応について調査を行った。「表 5.3」に, ラック数と入出荷頻度 (1日換算の在庫回転率) の対応を示す0。なお, 本研究の対象としている立体自動倉庫のモデルはデュアルサイクル動作であり, 入荷と出荷の件数は等しいことが前提となっている。そのため, 入荷と出荷の件数が等しくない立体自動倉庫の例 (主に製造業者のものが該当する) においては, 頻度の高いものにあわせることとする。「表 5.3」より, 自動倉庫の規模のみならず入出荷頻度も大小があり, 在庫回転率は最低の事例で3 %, 最高の事例で53 %, 平均24 %であった。このことから, 立体自動倉庫の入出荷頻度は, 一般的には1日あたり25 %前後であると考えられる。

命令分布特性については, 1日の中での入出荷を対象と, 全入出荷数を固定し, 平均分布, 指数分布, 正規分布, 混合分布の関数を用いて以下のように生成する。

表 5.3　自動倉庫売上高・台数・パレット数の推移 [14][15]

倉庫名	ラック数	入出庫頻度（日）	在庫回転率	入庫頻度（日）	出庫頻度（日）	備考
三菱電機㈱福岡製作所中間自動倉庫	1,800	960	53%	960	960	*1
富士重工㈱ブヒンセンター	9,720	640	7%	640	560	*1
日野自動車販売㈱部品流通センター	2,520	900	36%	150	900	*1
日産石油化学㈱千葉工場立体倉庫	3,600	240	7%	240	200	*1
サラリー社冷凍自動倉庫	5,310	530	10%	530	530	*1
チバ社自動倉庫	20,358	680	3%	680	680	*1
ピジョン㈱立体倉庫	2,105	1,053	50%	1,053	1,053	*2

注) *1:[14], *2:[15]

4.2 入出庫命令分布特性の違いによる機会損失

4.2.1 平均分布による入出庫命令分布特性

1 日の中での入出荷分布が均一である場合を想定し, 以下の式にて生成する。ここで, f: 命令数/時間, c: 1 日の全命令数, T: 合計稼働時間 (8 時間) とする。

$$f = \frac{c}{T} \quad (\text{一定}) \qquad \cdots\cdots\cdots(5.7)$$

使用記号

f : 命令数/時間

C : 1 日の全命令数

T : 合計稼働時間

「図 5.17」に, 1 日の入出荷命令数 c = 1,000 とするときの, 1 日の命令数の分布を例として示す。

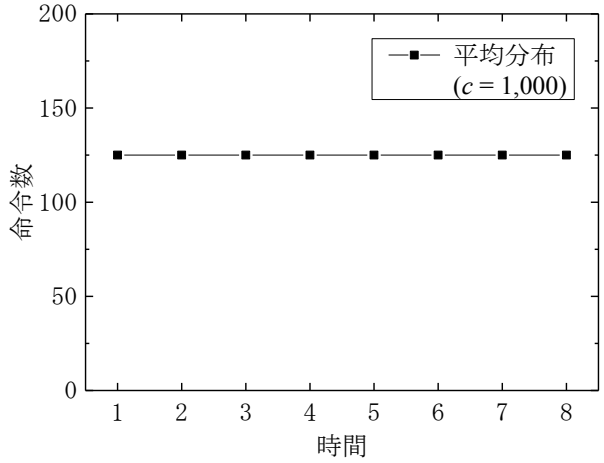

図 5.17 平均分布における入出荷命令分布

4.2.2 指数分布による命令分布特性の検討

1 日の中で夕刻に入出荷が集中する場合を想定し, 以下の式にて生成する。ここで, $f(t)$: 命令数/時間, c: 1 日の全命令数, A = 0.5, T: 合計稼働時間 (8 時間), t: 稼働時間 ($1 \leqq t \leqq 8$) とする。

$$f(t) = c \cdot A^{T-t+1} \qquad \cdots\cdots\cdots(5.8)$$

使用記号

$f(t)$：命令数/時間：)

c：1 日の全命令数：(c)

T：合計稼働時間：()

t：稼働時間 (1≦t≦8)：()

「図 5.18」に, 1 日の入出荷命令数 c = 1,000 とするときの, 1 日の命令数の分布を例として示す。

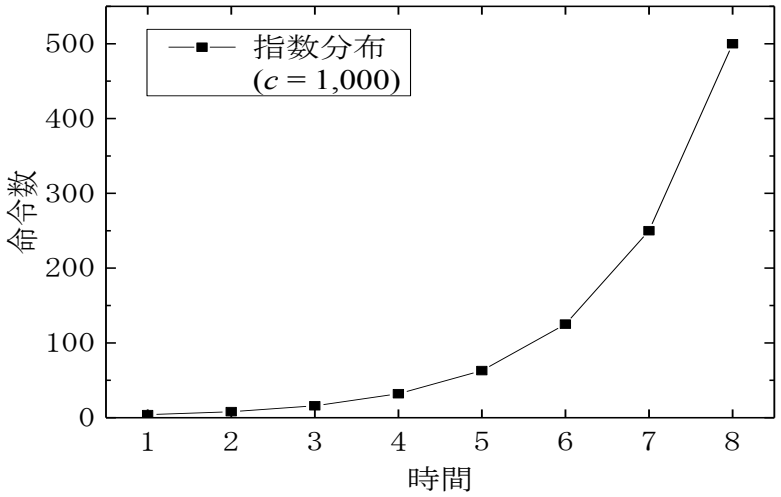

図 5.18 指数分布における入出荷命令分布

4.2.3 正規分布による命令分布特性の検討

1 日の中で昼に入出荷が集中する場合を想定し, 以下の式にて生成する。ここで, $f(t)$: 命令数/時間, c: 1 日の全命令数, t: 稼働時間 (1≦t≦8), 標準偏差σ = 1, 平均μ = 4 とする。

$$f(t) = c \cdot \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{(t-\mu)^2}{2\sigma^2} \right\} \qquad \cdots\cdots\cdots(5.9)$$

使用記号

$f(t)$：命令数/時間

c：1 日の全命令数

t：稼働時間

σ：標準偏差

μ：平均

「図 5.19」に, 1 日の入出荷命令数を 1,000 とするときの, 1 日の命令数の分布を例として示す。

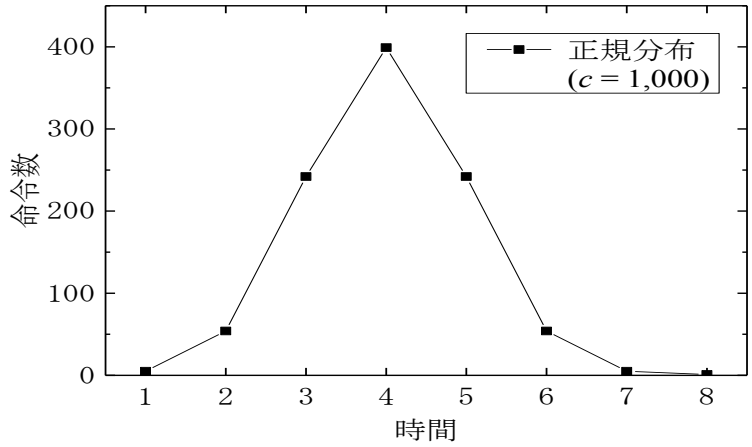

図 5.19 正規分布における入出荷命令分布

4.2.4 混合分布による命令分布特性の検討

1 日の中で午前と午後に入出荷が集中する場合を想定し, 朝と夕それぞれの分布が正規分布であるとして, 以下のように生成する。

午前のピークは, $f(t_1)$: 命令数/時間, c_1: 午前の全命令数 (=1 日の全命令数の半分, $c / 2$), t_1: 稼働時間 ($1 \leqq t_1 \leqq 5$), 標準偏差 $\sigma_1 = 0.8$, 平均 $\mu_1 = 3$ とする。

$$f(t_1) = c_1 \cdot \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left\{ -\frac{(t_1 - \mu_1)^2}{2\sigma_1^{\,2}} \right\} \qquad \cdots\cdots\cdots(5.10)$$

使用記号

$f(t1)$：命令数/時間

$c / 2$：午前の全命令数：

t_1：稼動時間

σ_1：標準偏差

μ_1：平均

同様に午後のピークは, $f(t_2)$: 命令数/時間, c_2: 午後の全命令数 ($= c / 2$), t_2: 稼働時間 ($4 \leqq t_2 \leqq 8$), 標準偏差 $\sigma_2 = 0.8$, 平均 $\mu_2 = 6$ とする。

$$f(t_2) = c_2 \cdot \frac{1}{\sqrt{2\pi}\sigma_2} \exp\left\{ -\frac{(t_2 - \mu_2)^2}{2{\sigma_2}^2} \right\} \quad \cdots\cdots\cdots(5.11)$$

使用記号

$f(t_2)$：命令数/時間

$(c / 2)$：午前の全命令数

t_2：稼働時間

σ_2：標準偏差

μ_2：平均

そして，以下のように午前のピーク $f(t_1)$ と午後のピーク $f(t_2)$ を足しあわせたものを，1 日の命令分布 $f(t)$とする。

$$f(t) = f(t_1) + f(t_2) \quad \cdots\cdots\cdots(5.12)$$

「図 5.20」に，1 日の入出荷命令数を 1,000 とするときの，1 日の命令数の分布を例として示す。

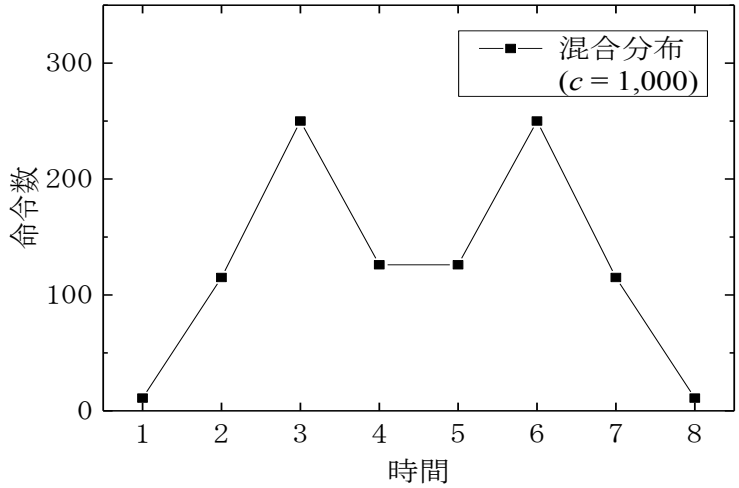

図 5.20 混合分布における入出荷命令分布

4.3 各命令分布の固定ロケーション型とフリーロケーション型の検討

このようにして生成した各命令分布を用いて立体自動倉庫の入出荷シミュレーションを行うことで，クレーンの稼働率を求めることとする。立体自動倉庫は，前項の検討を踏まえ，固定ロケーション型とフリーロケーション型それぞれにつき，検討を行った。なお，初期在庫充填率は 80 %及び 50 %とする。また，在庫回転率 25 % (入出荷命令数 1,000 / 日) ～50 % (入出荷命令数 2,000 / 日) の範囲にて，上記のように生成した各命令分布特性における 1 時間あたりの入出荷命令数のピークを，1 週間 (56 時間) 連続して実行可能な性能を要件として設定し，前項で示した手法によって自動設計を行っている。平均稼働率は，入出荷シミュレーション計算を行った 56 時間の平均値とする。なお，機会損失額は，平均非稼働率 (100 %－平均稼働率)×初期導入コスト (投資額) と定義しており，投資額に占めるムダともいえる。

まず，固定ロケーション型立体自動倉庫における，初期在庫充填率 80 %での，各命令分布における初期導入コスト (投資額)，平均稼働率，機会損失額の命令数 (在庫回転率) 依存性について計算検討を行った。初期導入コストの推移を「図 5.21」，平均稼働率の推移を「図 5.22」，機会損失額の推移を「図 5.23」に示す。

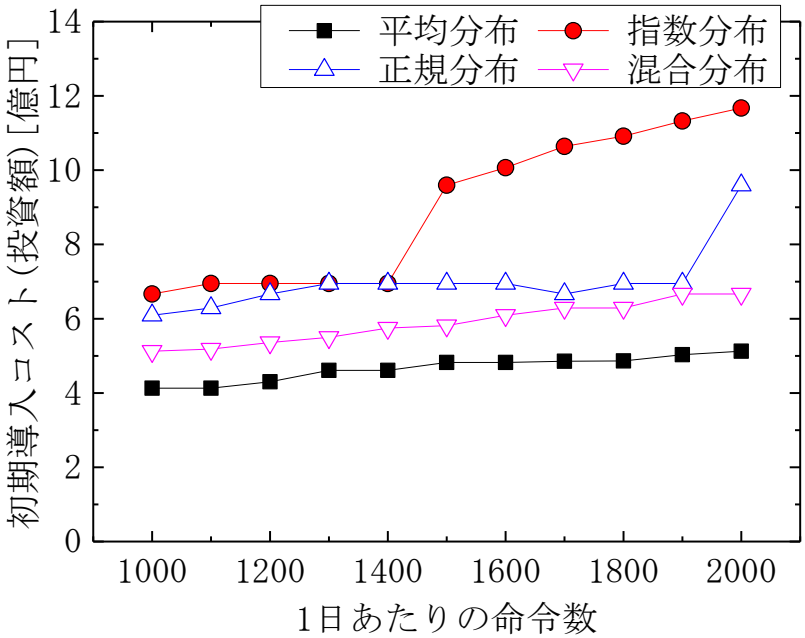

図 5.21 固定ロケーション型 (在庫充填率 80 %) における初期導入コスト (投資額) の入出荷命令数 (在庫回転率 25～50 %) 依存性

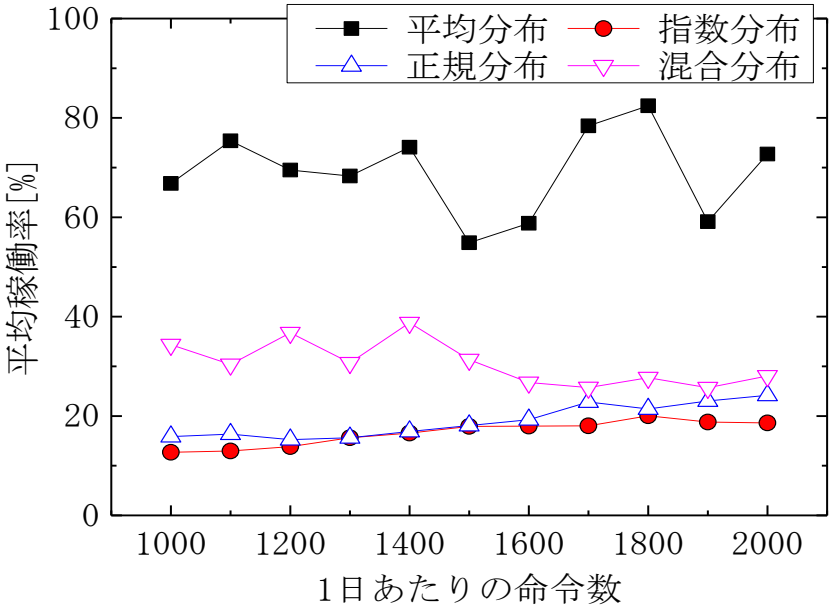

図 5.22 固定ロケーション型 (在庫充填率 80 %) における平均稼働率の入出荷命令数 (在庫回転率 25～50 %) 依存性

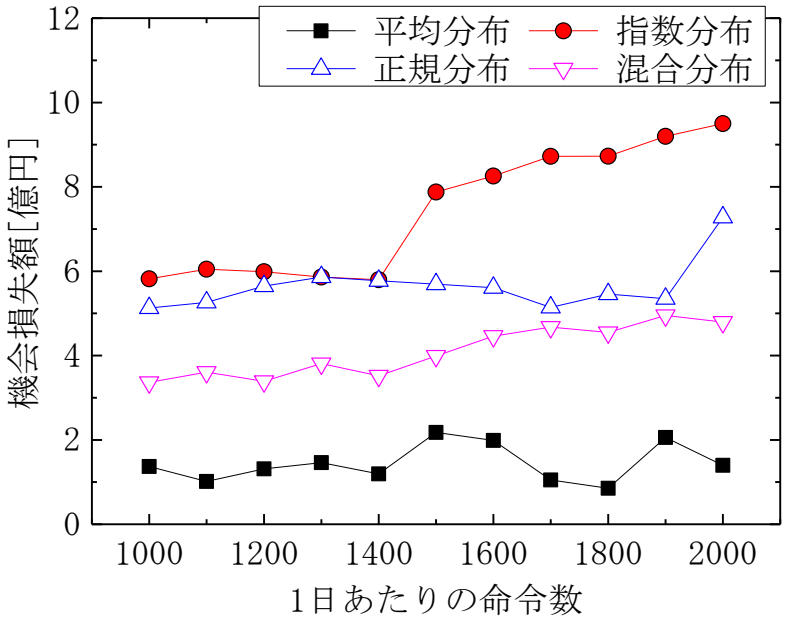

図 5.23 固定ロケーション型 (在庫充填率 80 %) における機会損失額の入出荷命令数 (在庫回転率 25～50 %) 依存性

結果より，在庫回転率 (命令数) の増加に伴い，いずれの命令分布においても投資額が増大することがわかる。平均分布のときが最もコストが低く，混合分布，正規分布と続き，指数分布が最大となる。これは，1 時間あたりの命令数のピーク数と対応する。

平均稼働率に着目をすると，平均分布では 60 %～80 %で推移しているが，混合分布では 30 %～40 %，正規分布で 20 %前後，指数分布では 10 %～20 %程度となっている。つまり，平均非稼働率は平均分布では 20 %～40 %，混合分布では 60 %～70 %，正規分布では 80 %前後，指数分布では 80 %～90%である。非稼働率とはすなわち，入出荷命令を処理しておらずクレーンが稼動していない状態の時間比率である。

同様にして計算検討を行った，固定ロケーション型立体自動倉庫における初期在庫充填率 50 %での結果を「図 5.24，図 5.25，図 5.26」に，フリーロケーション型立体自動倉庫における初期在庫充填率 80 %の結果を「図 5.27，図 5.28，図 5.29」に，フリーロケーション型立体自動倉庫における初期在庫充填率 50 %の結果を「図 5.30，図 5.31，図 5.32」に，それぞれ示すが，同様の傾向を示している。

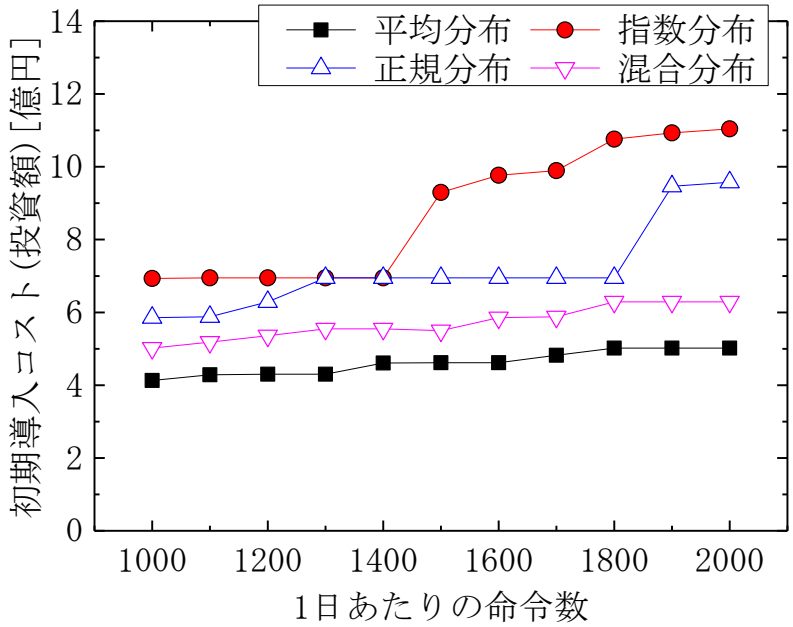

図 5.24 固定ロケーション型 (在庫充填率 50 %) における初期導入コスト (投資額) の入出荷命令数 (在庫回転率 25～50 %) 依存性

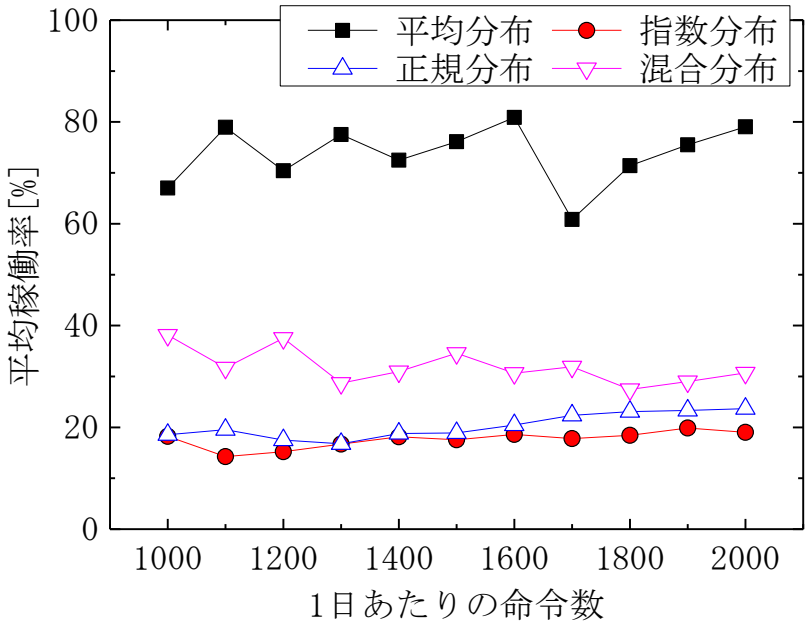

図 5.25 固定ロケーション型 (在庫充填率 50 %) における平均稼働率の入出荷命令数 (在庫回転率 25～50 %) 依存性

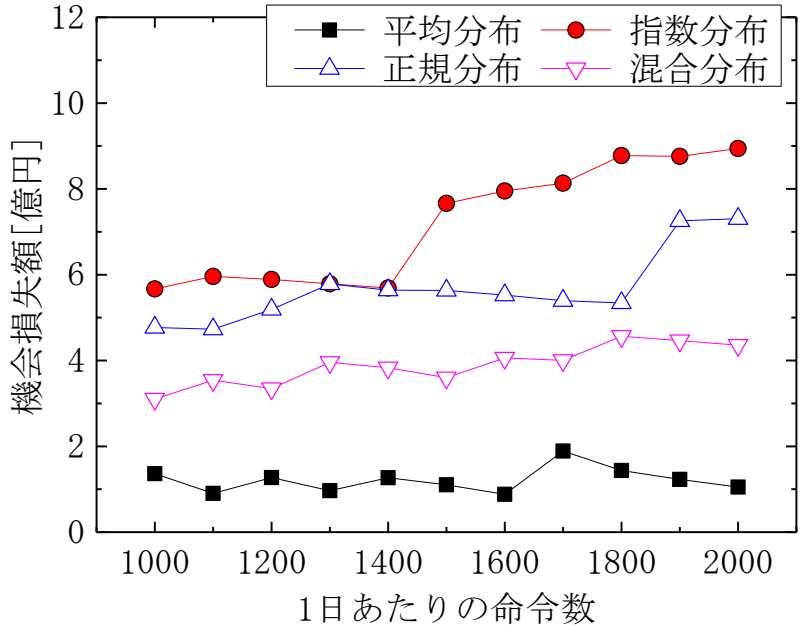

図 5.26 固定ロケーション型 (在庫充填率 50 %) における機会損失額の入出荷命令数 (在庫回転率 25～50 %) 依存性

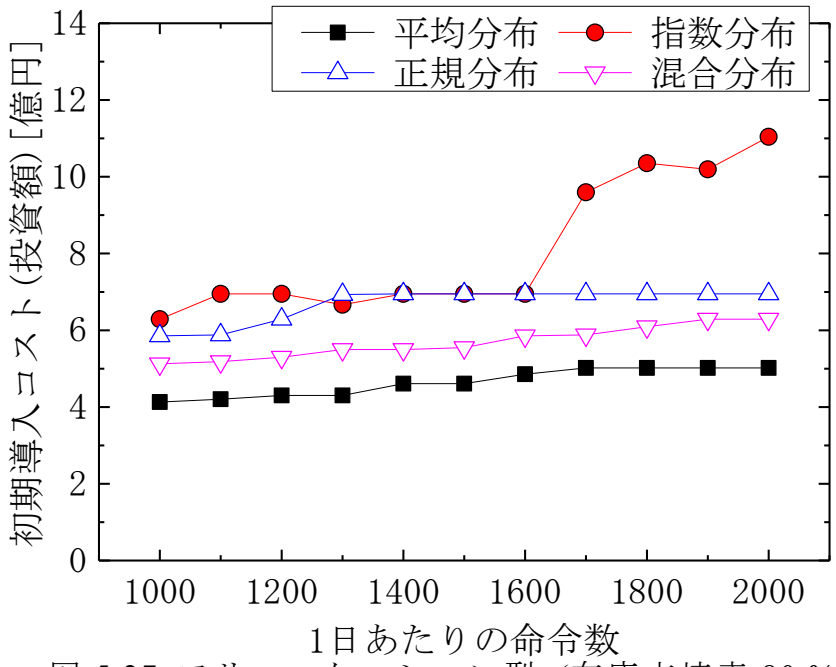

図 5.27 フリーロケーション型 (在庫充填率 80 %) における初期導入コスト (投資額) の入出荷命令数 (在庫回転率 25～50 %) 依存性

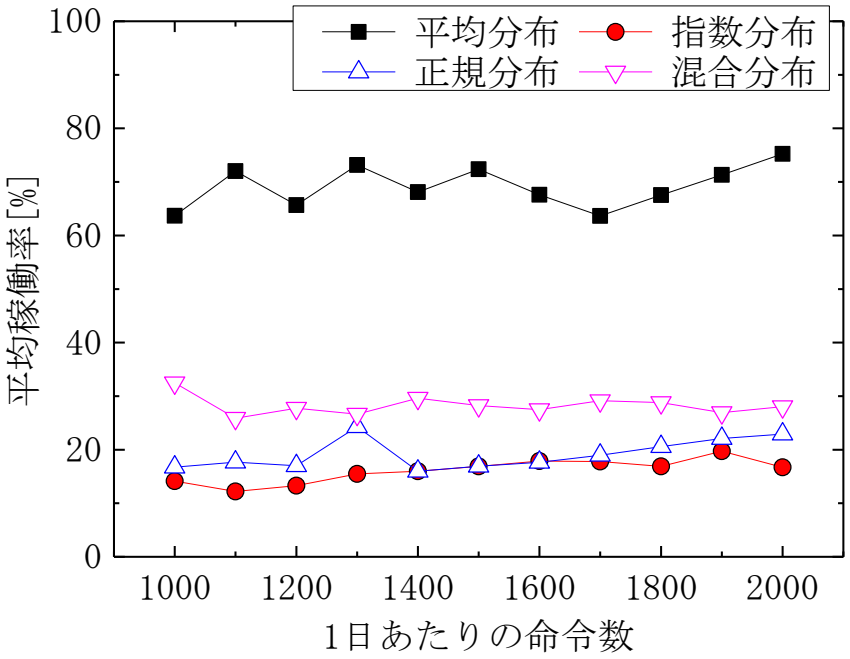

図 5.28 フリーロケーション型 (在庫充填率 80 %) における平均稼働率の入出荷命令数 (在庫回転率 25～50 %) 依存性

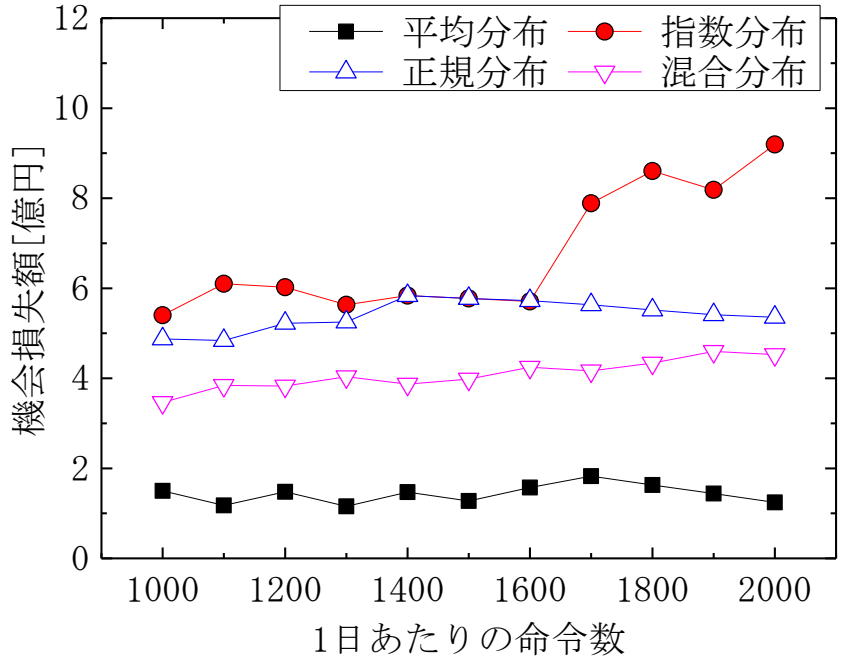

図 5.29 フリーロケーション型 (在庫充填率 80 %) における機会損失額の入出荷命令数 (在庫回転率 25～50 %) 依存性

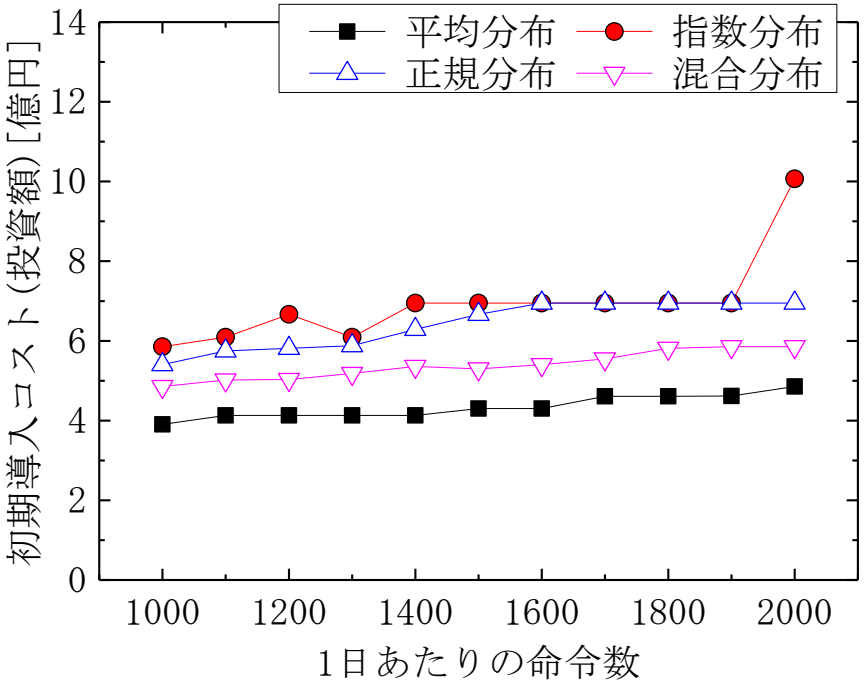

図 5.30 フリーロケーション型 (在庫充填率 50 %) における初期導入コスト (投資額) の入出荷命令数 (在庫回転率 25～50 %) 依存性

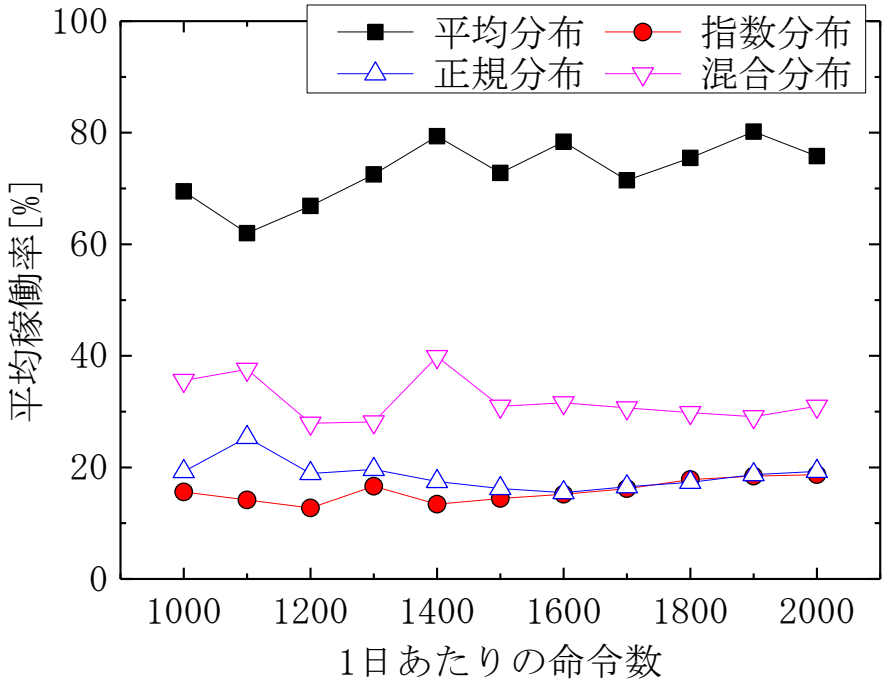

図 5.31 フリーロケーション型 (在庫充填率 50 %) における平均稼働率の入出荷命令数 (在庫回転率 25～50 %) 依存性

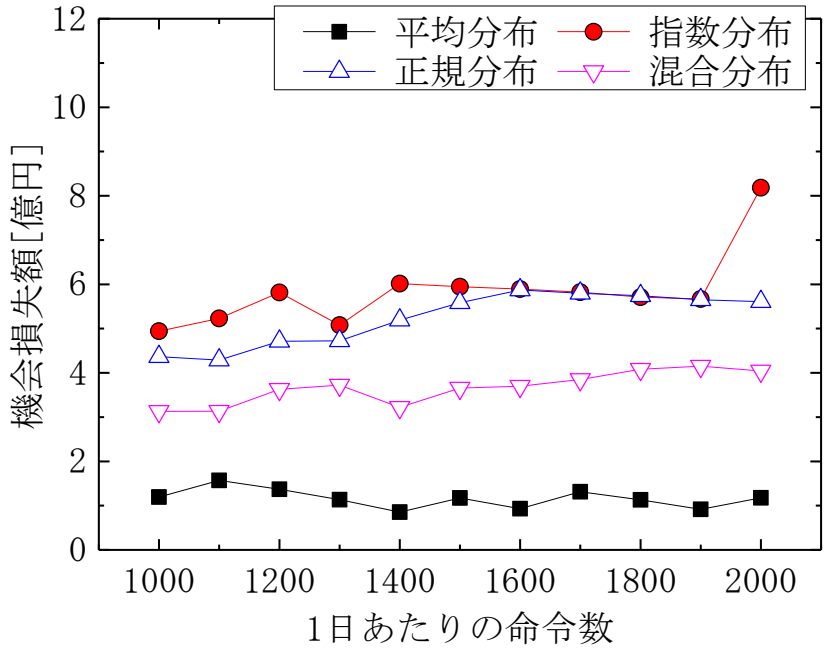

図 5.32 フリーロケーション型 (在庫充填率 50 %) における機会損失額の入出荷命令数 (在庫回転率 25～50 %) 依存性

これら結果はいずれも，自動倉庫の運用をマネジメントレベルで入出荷の分布を平均分布に近づけることが可能であれば，稼働率が向上し，機会損失額を大幅に低減可能であることを意味している。
しかし，1 日の中の入出荷分布を平滑化することは，特に小売業態においては，店舗の営業時間や開店前の商品品出し時間との関係で入出荷が集中しやすく，現実的には困難といえる。これに対して，月間波動は消費者の購買活動によるため，給与支給日や年金支給日以降売上が上昇する傾向にあり，月末に向けて商品発注量が増加する傾向にある。
一方，締め日，支払日と金利の関係といった商慣習に起因することも考えられる。これは，例えば 20 日締め翌月 20 日払いであった場合，20 日に仕入れたときと 21 日に仕入れたときとでは，1 日の違いであっても支払日が 1 ヶ月伸びる。つまり 1 ヶ月分の金利を稼いだと考えられ，仕入れは締め日後に行う。
これらの消費者行動や商慣習に起因した波動は，立体自動倉庫の平均稼働率を大きく引き下げ，機会損失を増大させることに繋がっているといえる。すなわち，サプライチェーン全体を鑑みると締め日を意識した仕入れによる利益確保を，倉庫規模拡大による過剰投資で相殺していることも考えられる。ここに改善の余地が大いにあるといえる。
年間波動は季節による需要の増減と対応する。季節商品，ボーナス時期，お中元・お歳暮やクリスマスなどといったものがこれに当たるが，これら需要量自体の変動を商業的努力によって平滑化することはむずかしい。これら全てに対し倉庫規模の拡大で対応を行うことは非効率的であり，需要量の増加する季節に限定し別の対応をとることが有効と考える。
しかし，このような変動する小売業の売上情報・在庫情報・納品情報などをメーカー・取引先・小売業が情報共有化することにより，ピーク値を抑え，平準化を図り 「ムリ・ムダ・ムラ」を縮小することが可能である。すなわち，このような変動に合わせた運用計画を策定し，実行するためには，生産から小売にいたるサプライチェーンにおいて，情報共有化が求められていると考える。

5. 結論

本研究では，デュアルサイクル動作の自動倉庫システムを対象に，パレットの入出荷に用いるクレーンの並列動作性を考慮したコンピュータモデルシミュレーションを構築する。そして，設定した要求性能下における最適ラック配置を，入荷先・出荷元ラック決定に関し複数のアルゴリズムを適用し，検討を行った。
その結果，並列動作性と導入コストは，等しい要求性能下であっても入出荷のアルゴリズムや在庫の格納状況によって大きく変化することを明らかとした。また，初期導入コストと機会損失額は，日毎の入出荷命令の負荷を平準化する

ことで低減可能であることも，シミュレーション計算結果によって数値的に示した。

また，小売業の売上情報・在庫情報・納品情報などをメーカー・取引先・小売業が情報共有化することにより，ピーク値を抑え，平準化を図り 「ムリ・ムダ・ムラ」を縮小することが可能である。

6. おわりに

第 5 章では，自動倉庫のシステムの機会損失にて論じた。シュミレーションの結果により，機会損失が発生し，日毎の入出荷命令の負荷を平準化することで低減可能であることが明らかになった。

初期導入コストと日毎の入出庫を平準化するためには，川下である小売の売上情報・在庫情報を川上の卸売業，メーカーと共有するために SCM を構築し，情報共有化を図り、サプライチェーン全体で商品量を平準化することが必要であることが明らかになった。

参考文献

[1] 生島義英, 佐藤哲也, 唐澤豊,若林敬造, 自動倉庫における機会損失の低減に関する基本的研究, 日本ロジスティクスシステム学会誌, 2016 年 3 月
[2] ファジィ理論を応用した自動倉庫の最適設計モデルとその評価, 内田智史・相浦宣徳・唐澤豊, 日本経営工学会誌, 日本経営工学会誌 43(6), 453, 1993-02-15 , 社団法人日本経営工学会
[3] 株式会社ダイフク, Daifuku News No. 169(2003)
[4] 宮城渉, 物流拠点における自動倉庫の構造と役割, 日本生産管理学会論文誌, Vol. 11, No. 2, 139-144 (2005)
[5] 鈴木準, 世界に見る物流センター40 年, 第 21 回ロジスティクス懇話会, 東京海洋大学 (2009).
[6] 日本産業機械工業会, 自動倉庫納入実績表.
[7] 公益社団法人日本ロジスティクスシステム協会・一般社団法人日本物流システム機器協会, 2013 年度 物流システム機器生産出荷統計【概要版】(2014)
[8] 公益社団法人日本ロジスティクスシステム協会, 2008 年度 物流システム機器生産出荷統計の概要(2009)
[9] 公益社団法人日本ロジスティクスシステム協会, 2009 年度 物流システム機器生産出荷統計【概要版】(2010)
[10] 公益社団法人日本ロジスティクスシステム協会・一般社団法人日本物流システム機器協会, 2010 年度 物流システム機器生産出荷統計【概要版】(2011)
[11] 公益社団法人日本ロジスティクスシステム協会・一般社団法人日本物流システム機器協会, 2011 年度 物流システム機器生産出荷統計【概要版】(2012)
[12] 公益社団法人日本ロジスティクスシステム協会・一般社団法人日本物流システム機器協会, 2012 年度 物流システム機器生産出荷統計【概要版】(2013)
[13] 塩見敏弘, 自動倉庫の最適設計に関する基本的研究, 神奈川大学大学院工学研究科 博士前期課程 経営工学専攻 修士論文 (2000)
[14] 物流システム研究所, 倉庫・配送センター実例集, 株式会社テクノ (1975).
[15] 唐澤豊 監修,実例 現代の倉庫・配送センターI, PDS 物流システム研究所 (1972)

第 6 章　小売業における機会損失と SCM の役割

1. はじめに

第 2 章で定義した機会損失を踏まえ，第 3 章「百貨店における SCM の取り組みと機会損失に関する研究」，第 4 章「小売業におけるバックヤードの機会損失に関する研究」，第 5 章「自動倉庫における機会損失に関する研究」の結果を踏まえ，小売業において様々な機会損失が存在することが明らかになった。

機会損失の原因は，サプライチェーンの非効率化である，すなわち SCM が全体最適になっていないことに起因すると考える。

しかし，機会損失は「目に見えない」課題であり潜在化・内在化する課題といえる。既存の手法を継続する方向性では気が付かない経営課題である。

2.経営における二つの側面の統合

企業経営において，企業発展の原動力として二つの側面があると考える。ひとつの側面はビジブル志向型経営であり，もう一つの側面はインビジブル抑止志向型経営である。すなわち，経営の光の部分と影の部分，陰陽である。

これらの概念をまとめると「図 6.1」に示すとおりである。

ビジブル志向型経営の定義は，「目に見える」課題に対する経営であり，戦略思考型，需要創造型の経営方針である。マーケティングの技法を活用し，有効需要を創造するとともに，潜在需要を掘り起こし，新製品の開発に取り組んできた。また，近年の IT 技術，インターネットのネットワークの発展に伴い，ネットビジネスやヴァーチャルビジネスなど新しい販売経路の創造に取組み，現在ではリアル店舗に匹敵する販売経路として成長しつつある。

これらの様々な「カイゼン」により，新たな需要を創造し，新製品を開発し，新しい販売経路で市場に投入することにより，継続的に企業成長を図る経営に取り組んでいる。多くの日本企業は「カイゼン」のために積極的な投資を行い，経営合理化に取り組み，「日のあたる，目に見える側」に多くの注意を払ってきた。

インビジブル抑止志向型経営の定義は，「目に見えない」課題，潜在化・内在化する課題に対する経営である。機会損失，チャンスロスにより失われている課題を解決し，企業の発展を図る経営方針である。すなわち，既存の枠組みの中で発生している「既存のルール」，「当たりまえ」，「慣習」という思考停止状態に陥っている課題を掘り起し，顕在化させ，機会損失として捉え，この機会損失を抑止することにより経営改善する取組みである。

機会損失として捉える内容は，「スピード機会損失」，「ムリ・ムダ・ムラの三無機会損失」，「顧客リピート・ロイヤルティ機会損失」があげられる。

「スピード機会損失」は，タイムリーの対応できないため発生する機会損失

である。その範囲は，商品開発・生産・ロジスティクス・販売に至るサプライチェーンのいたるところで発生している。生産ではジャストインタイムやカンバン方式の資材調達がスムーズに機能しない。ロジスティクスでは積載効率や輸送回転率，在庫過多・在庫不足などが発生し，コストアップ，業務効率の低下を引き起こしている。販売では，欠品・不良在庫などが発生し，欠品は小売での販売機会損失を招き，不良在庫は値下げや廃棄などで収益の悪化することとなる。これらは，生産・ロジスティクス・販売に至るサプライチェーンの対応スピードの遅さやサプライチェーンの滞りが原因で発生している。これらのサプライチェーンのスムーズさ，スピードを高めることにより，サプライチェーンで発生している「ムリ・ムダ・ムダ」の機会損失を改善することが可能となる。これらの機会損失の改善により，最終消費者の顧客ロイヤリティ向上に繋がることとなる。これらサプライチェーンで発生している課題を捉え，それらを改善すること，すなわちSCMを導入し，SCM戦略を構築していくことが経営に求められている。

今後更なる企業を発展させるためには，ビジブル志向型経営とインビジブル抑止志向型経営を統合して，「目のあたる，目に見える側」と「目に見えない側」の経営改善をどちらかに偏ることなく総合的に推し進めていくことが重要である。

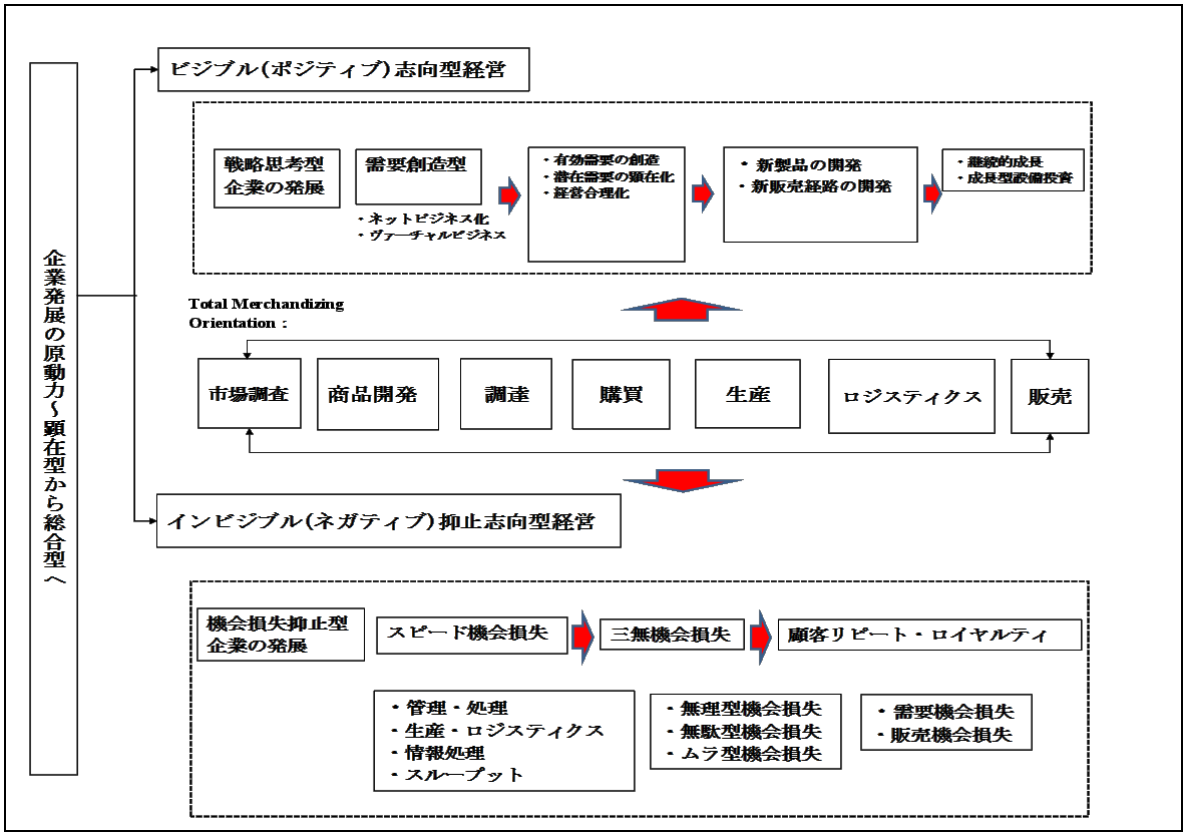

図 6.1　ビジブル志向型経営とインビジブル抑止志向型経営の概念図

3. 協調指向型 SCM の必要性

3.1 協調指向型 SCM とは

大辞林によると，協調とは①力を合わせて事をなすこと，②利害の対立するものが，力を合わせて事にあたることである。

SCM は，サプライチェーンを構成する個々の企業の業務を接続・統合させることであり，資源の調達購買，製造，流通，販売の諸機能をサプライチェーンの中で単一実態として捉え，統合することを図ったものである。

SCM の成果は，①コスト低減，②顧客価値と顧客満足の向上，③競争の優位性の確保である。サプライチェーンを構成する個々の企業が成果を得られなければ SCM が成り立たないのは自明である。よって協調志向型 SCM とは，SCM の成果を得るためにサプライチェーンを構成する個々の企業が力を合わせて，業務を統合させることである。

3.2 SCM 以前の協調取組み

SCM 以前の取組み事例として，食品業界では ECR（効率的な消費者対応），アパレル業界では QR（クイックレスポンス）があり，EDI（電子データ交換）で情報交換を行う情報システムとそれに依拠する標準規格を利用した受発注処理システムと物流を連動させたシステムである。

QR とは，クイック・レスポンス（Quick Response）の頭文字である。輸入衣料品に圧されたアメリカ繊維産業が 1984 年に輸入対抗・国内産業生存策として打ち出した業界運動に端を発している。

QR は，取引企業間の対等な「パートナーシップ」の確立を基礎に，「適切な商品を，適切な時期に，適正な価格で，適切な場において供給するシステム」を，「最短のリードタイムと最小のリスクで，しかも最大の競争力を持つように構築する」ことを目指すものある。

QR とは繊維素材や織編物などの原材料段階から縫製段階をへて小売段階に至るまで，取引関係でつながった各々の企業が「パートナーシップ」に基づいて協働して信頼・協力関係を築き上げ，情報交換を行って，売れる商品を過不足なく素早く生産し，供給していくものである。アメリカの場合，QR は大きな成功を収めた。

その成功の要因は，以下のとおりである。

① 「敵対的」な関係にあるとされてきた売り手企業と買い手企業（テキスタイル企業とアパレル企業，アパレル企業と小売企業など）が互いにパートナーとして手を握り合ってともに利益を得る方向で業務改革を行った。

② 当時めざましい発展を遂げつつあった情報技術（IT）の成果を十分に活用した。

③ 「業界標準」を採用し，IT を活用した。

④ メーカー側は小ロット･短納期生産を実現するためのさまざまな工夫を実施した。
⑤ 小売側は売場で発生する販売データ（POS データ）をいち早く的確にメーカー側に伝えることで情報の共有化を推進した。

QR や ECR では，小売業の販売実績，すなわち最終需要発生を起点とする各段階での受注に迅速に対応することによって，欠品を回避し，販売を確保することを目指した。これらのシステムでは，伝票のペーパー削減化や受発注作業の削減が中心であったほか，販売や物流，在庫などのサプライチェーンの一部最適化にとどまるものであり，資源の調達購買，製造，流通，販売の諸機能をサプライチェーンの中で単一実態として捉え，統合するまでには至らなった。よって，QR や ECR によって，サプライチェーンを構成する各企業ならびに顧客に生産性，効率性，在庫の低減，販売増がもたらされたかは疑問が残る。

3.3 協調指向型 SCM の必要性

SCM は，個々の企業の業務を接続・統合させることであり，接続する要素は以下のとおりである。その要素とは，①情報の共有化・②リスクと報酬の共有化・③協働化・④目標の共通化・⑤キープロセスの統合・⑥長期的安定的関係の維持・⑦各機能の調整である。これらの要素を同期化していくことが求められる。

また，サプライチェーン間での協働計画，生産・販売・在庫などの戦略的情報を交換することが必要である。この戦略的情報交換が共有されない，または各企業の計画と企業間での協働計画がなされていないと最終消費者が変化するとついていけないこととなる。これは需要と供給の同期が行われていない，サプライチェーン間の協働が行われていないこととなる。よって，SCM ではサプライチェーン間での同期と協働が求められ，サプライチェーン間の協調が必須の条件となる。協調する具体的な内容は，①サプライチェーンの全体最適志向と責任の明確化・②情報共有化・③コミュニケーションツールの明確化・④ボトルネックの発見と改善・⑤需要管理・⑥協働である。

よって，個々の企業を統合する SCM においては，各企業の協調が必要不可欠な条件となると考える。

4. 共同指向型 SCM の展望

4.1 SCM 共同化の基本

SCM 共同化の基本は，生産供給ネットワークと小売需要ネットワーク共同化の二つに分類されると考える。生産供給ネットワーク共同化は生産者主体型 SCM の共同化であり，小売需要ネットワーク共同化は小売業主体型 SCM の共同化である。ロジスティクスの側面から考えると，3PL 主導型共同化と運輸業主体型共同化が考えられる。このように，戦略主体によって戦略となる軸足の視

点が異なっている。

4.2 共同化推進主体と基本要素

共同化戦略の他の側面として検討しなければならない点は，共同化推進の主体が荷主主体であるか，3PLあるいは4PL主体であるかという点であり，アウトソーシングに関わる戦略事項である。

共同化戦略の基本要素は，①チャネル・②推進・③荷主・④業種・⑤システムなどを累計の軸として，ミクロ特性を勘案して事象にマッチした戦略展開要素である。

共同化の基本要素は，①機能主体・②チャネル主体・③内容主体・④業種主体を基本として，その戦略的な展開を図ることである。図に示すと「図 6.2」に示す通りとなる。これらの基本要素をベースとして，共同化推進主体が荷主の場合，更に細分化し戦略展開するための要素は，①委託類型（計画提案型，管理型，業務型）・②機能類型（輸送，配送，保管，情報，流通加工）・③チャネル類型（水平型，垂直型，グリット型）・④推進類型（単一型，共同型）・⑤産業類型（メーカー型，卸売型，小売業型）・⑥業種類型（同業種型，異業種型）を軸として戦略的な展開を図ることである。ＳＣＭ戦略の展開が荷主主体の物であれば，これらの諸要素を如何に組み合わせるかが望ましいか，あるいは実現可能かにおいて検討することが重要である。一方，３ＰＬ主体の場合の要素は，①受託類型（コンサル型，計画提案型，管理型，業務型）・②流通類型（水平型，垂直型）・③発展類型（現状拡大型，共同拡大型）・④推進類型（自社単独型，他社共同型）・⑤産業類型（メーカー型，卸売型，小売業型）・⑥業態類型（同業種型，異業種型）となり，基本要素は同一ではあるが，荷主と業者の視点の相違から若干要素が異なることとなる。

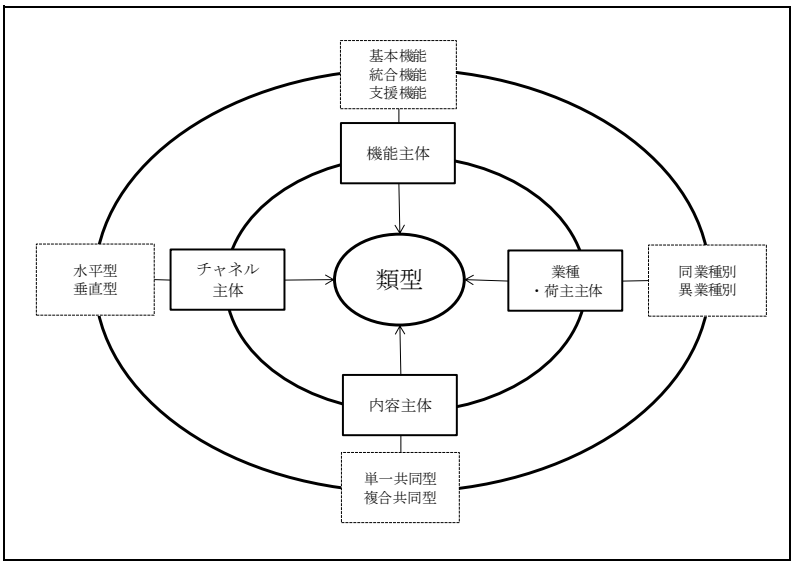

図 6.2　ＳＣＭ共同化の主要基本要素

4.3 共同指向型 SCM の展望

単一企業の経営改新への挑戦は限りないものであるが，規模・効率・人材・ノウハウ・情報・管理あるいは社会性・公共性の制約など早晩限界に突き当たると考える。このような単一企業の限界を克服し，ブレークスルーする大きな手段として，共同化が重要である。これからの共同化の展望は，ネットワークをベースとする情報技術（IT）と管理理論の現実的展開をベースとして，共同指向型 SCM を推進する必要があると考える。

5. SCM の新しいミッション

SCM の新しいミッションを考えるうえで，最近の注目される取組みとして「オムニチャネル」があげられる。アメリカでオムニチャネルが注目されるようになったのは，全米小売業協会のレポートで取り上げられ，米百貨店メーシーズの CEO がオムニチャネル化宣言をした 2011 年からである。この背景には EC やスマートフォンの利用が進んだことがあげられる。 2013 年のアメリカにおける EC 化率（商取引に占める EC の利用率）は 5.8%に達し，その後成長を続け，2018 年には 8.9%にまで達する見込みとされ，実店舗の売上に影響が出てきている。一方，日本国内では，2015 年度の B to C における日本国内の EC 市場規模は 13 兆 7746 億円で，対前年比 7.6%の伸び率であった。また，物販系分野の EC 化率は 4.75%で，対前年比 0.38 ポイント増になった。

さらにスマートフォンの普及と共に，小売店頭で現物を確認した上で価格の安い EC サイトで購入する「ショールーミング」の利用が広がり，インターネットの影響が小売企業にとって見逃せないものとなっている。この対策として，単に EC サイトに力を入れるマルチチャネルのアプローチではなく，複数のチャネルを持つ強みを活かすオムニチャネルに取り組み始めたのである。日本国内では，セブン＆アイが提供するオムニチャネルサービス「オムニ７」などが稼働し始めている。

経済産業省「平成２６年度電子商取引に関する市場調査」によると，“オムニ（omni）”とは日本語で“すべて，あまねく”という意味を持つ。消費者が商品の購入に至る過程において，実店舗，PC サイト，モバイルサイト（スマートフォン），ソーシャルメディア，従来型メディア（新聞・雑誌・TV），カタログ，DM 等，あらゆる販売チャネル・情報流通チャネルを経由する時代となっている。

オムニチャネルとは，消費者がこれらの複数のチャネルを縦横どのように経由してもスムーズに情報を入手でき購買へと至ることができるための，小売事業者によるチャネル横断型の戦略やその概念，および実現のための仕組みを指す。すなわち，顧客との接点になっている全てのチャネルを融合させることである。

具体的には，実店舗，PC サイト，モバイルサイト（スマートフォン），ソー

シャルメディア，従来型メディア（新聞・雑誌・TV），カタログ，DM 等など，あらゆる顧客接点から同質の利便性で商品を注文・購入できるという点，および，ウェブ上で注文し，店舗での受取りや自宅への配送，また店舗で在庫がなかった商品を即座にオンラインでの問い合わせで商品補充できるようする点，といった要素が含まれる。すなわち，オムニチャネルはサービス内容だけでなく裏側のオペレーションやデータ管理までチャネルをまたがって融合している。これは，まさにサプライチェーンであり，発展と業務効率化を図るためには SCM 戦略が求められている。

オムニチャネルというと，お客様との接触点（WEB や店舗）ばかりが注目されるが，オムニチャネルを裏で支えるのはロジスティクスである。オムニチャネルを実現するためには，裏側のオペレーションとして①顧客からの受注処理，②商品在庫およびと在庫場所の確認処理，③配送伝票の発行処理，④商品のピッキングと包装処理，⑤商品の出荷処理，⑥お客様への商品の引き渡し，⑦商品の補充発注処理，⑧商品の仕入れと在庫処理などのプロセスが計画的かつ効率的に実施されている。このプロセスがロジスティクスであり，サプライチェーンである。このロジスティクスのサービスレベルの違い，すなわち承りからお客様にお渡しできるリードタイムを短くすることが，競争優位を確立することとなり，顧客から支持されるオムニチャネルを決める重要な要素となる。スピード・コストダウン・品質などを同時にまたは個別に満足することであり，それに適した SCM 戦略を構築することが求められている。

今後，オムニチャネルを発展させるためには，ビジブル志向型経営とインビジブル抑止志向型経営を統合して，「日のあたる，目に見える側」であるお客様との接触点と「目に見えない側」であるロジスティクス・SCM の経営改善をどちらかに偏ることなく総合的に推し進めていくことが求められていると考える。

6. おわりに

第 6 章では，更なる企業を発展させるためには，ビジブル志向型経営とインビジブル抑止志向型経営を統合して，「日のあたる，目に見える側」と「目に見えない側」の経営改善をどちらかに偏ることなく総合的に推し進めていくことが重要であることが明らかになった。

ビジブル志向型経営とインビジブル抑止志向型経営を統合することは，機会損失，チャンスロスにより失われている課題を解決し，企業の発展を図る経営方針である。すなわち，既存の枠組みの中で発生している「既存のルール」，「当たりまえ」，「慣習」という思考停止状態に陥っている課題を掘り起し，顕在化させ，機会損失として捉え，この機会損失を抑止することにより経営改善すると考える。

機会損失は，「目に見えない」課題，潜在化・内在化する課題を改善しない

ことにより発生するものである。今回の研究では，サプライチェーンにおける小売業の非効率化が明らかになった。すなわち SCM が全体最適になっていないことに起因すると考える。

そして，SCM の全体最適化を図るためには，単一企業の限界を克服し，ブレークスルーする大きな手段として共同化が重要である。これからの共同化の展望は，ネットワークをベースとする情報技術（IT）と管理理論の現実的展開をベースとして，共同指向型 SCM を推進する必要があると考える。

参考文献

[1] 生島義英,佐藤哲也,唐澤豊,若林敬造,小売業におけるバックヤードの機会損失に関する基本的研究,日本ロジスティクスシステム学会誌 Vol.15,No.1,2016 年 3 月
[2] 生島義英,佐藤哲也,唐澤豊,若林敬造,小売業におけるバックヤードの機会損失に関する基本的研究ＳＣＭ, 日本ロジスティクスシステム学会第 18 回全国大会予稿集,2015 年 7 月
[3] 唐澤豊,現代ロジスティクス概論,ＮＴＴ出版, 2000 年
[4] 松村明,大辞林, 第三版, 三省堂刊行,2006 年
[5] 陳玉燕,唐澤豊,若林敬造,井上敬介,生島義英,豊谷純,ＳＣＭ戦略論の研究と戦略フレームワークの提案, 日本ロジスティクスシステム学会誌 Vol.14,No1, 2014 年 12 月
[6] 角井亮一,オムニチャネル戦略,日本経済新聞社出版部,2015 年 10 月
[7] 菊池康也,ＳＣＭの理論と戦略,税務経理協会,平成 18 年 4 月
[8] 経済産業省, 商務情報政策局 情報経済課,平成 27 年度我が国経済社会の情報化・サービス化に係る基盤整備（電子商取引に関する市場調査）, 平成 28 年 6 月
[9] 総務省,平成 27 年版情報通信白書,平成 27 年 7 月

第 7 章　結論

1. 結論

本研究では，第 2 章で機会損失の定義，ならびに SCM の定義を明らかにするとともに，SCM おける機会損失を流通業，製造業の領域で抽出するとともに機会損失の分析を述べた。

本研究での機会損失の定義は，「方策の選択により得ることができなかった利益」とした。次に SCM の定義は，SCM は自社のサプライチェーン前後の取引先・顧客の業務を鑑み，サプライチェーン全体で最適化することが望まれる姿であるとした。そして，SCM の目指す方向性は，サプライチェーン全体の最適化を目指すものであり，企業間の垣根を越えて複数の企業がビジネスネットワークを組んで最適化を目指すこととした。

次に SCM おける機会損失を川上である製造業から川下である流通業を俯瞰して抽出した。本研究の対象である流通市場における機会損失は，メーカー・卸・小売のサプライチェーンで発生する多段階の商品在庫・センター運用などに起因する「多段階センターの機会損失」，「在庫回転率の機会損失」，小売業での「バックヤードの機会損失」，サプライチェーンの各段階で発生する「輸送回転率・積載率の機会損失」が考えられた。

小売業において様々な「ムダ・ムラ・ムリ」が発生し，機会損失が発生していることが明らかとなり，具体的な小売業における 3 つの機会損失の課題を抽出し，その課題対する研究を第 3 章から第 5 章で深耕することとした。

第 3 章では，「日本の百貨店における SCM の取組みと機会損失に関する基本的研究」をテーマに掲げ，百貨店業界に関与する取引先を含めたサプライチェーンを研究対象とし，百貨店の構造改革として，「運用業務の効率化」に焦点を絞り，SCM がどのような効果を百貨店業界にもたらすのかを明らかにすることを試みた。その結果，運用業務の効率化施策に焦点を絞り，①商品マスター登録自動化，②注文伝票・仕入伝票削減化，③検品削減化，④値札削減化，⑤支払案内電子化を EDI による取引先との情報共有化により SCM を構築することにより，年間 15,232 百万円の経済効果があることが明らかとなった。

百貨店業界は，SCM を導入しないことにより，この経済効果を失っている。また，取引先も経済効果を失っている。ここに機会損失が発生していると考える。百貨店業界はこれらの構造改革を速やかに実施し，機会損失を最小にする取組みを実行することが求められているが明らかになった。

第 4 章では，「大型小売店におけるバックヤードの機会損失に関する研究」をテーマに掲げ，バックヤードを売場に転用すれば，新たに大きな投資することなく，既存の施設を利用することで収益を上げる事が可能であることが明らかになった。

百貨店・GMS・スーパーマーケットなどの大型小売店の「店舗バックヤード

の機会損失」に焦点を絞り，取引先との情報交換による最適な在庫計画に基づく売場の適正在庫予算を策定し，予算遵守することにより在庫過多が発生しない仕組みを構築し，バックヤードの商品在庫を排除し，バックヤードを縮小化することにより得られる経済効果，すなわちバックヤードを売場転用しないことによって生じる機会損失を明らかにすることを試みた。

算出した結果，百貨店業界においては，671,953 百万円の売上増となり，総売上を 10.0%増加させることとなり，税引き前利益では，9,340 百万円増加させるが明らかになった。

スーパー業界においては，2,238,494 百万円の売上増となり，総売上を 17.1%増加させることとなり，税引前利益では，48,320 百万円増加させることが明らかになった。

百貨店・スーパー合計では，2,910,446 百万円の売上増となり，総売上を 14.7%%増加させることとなる。税引き前利益では，57660 百万円増加させる。

さらに，百貨店業界では，42,197 百万円の経済効果を得られることとなる。スーパー業界では 232,041 百万円の経済効果得られるとなる。合計では，274,238 百万円の経済効果を得られることとなる。

また，バックヤード削減化によりもたらされる売り上げ増により新規雇用数は百貨店業界で 8,728 人の雇用増となり，スーパー業界で 90,525 人の雇用増となり，合計では，99,253 人の雇用増となることが明らかになった。

既存店舗のバックヤードを活用することにより，これだけの経済効果を得ることが明らかになった。バックヤードを活用しないことによって，これだけの機会損失が発生していることが明らかになった。

第５章では，「自動倉庫における機会損失に関する研究」をテーマに掲げ，主に卸売業や小売業，e コマースの物流センターで活用している自動倉庫に焦点を絞り，設定した条件及び要求性能を満たす中でコスト最少の立体自動倉庫を設計する汎用型の最適設計モデルシミュレーションを構築した。

その結果，並列動作性と導入コストは，等しい要求性能下であっても入出荷のアルゴリズムや在庫の格納状況によって大きく変化することを明らかとなった。また，業務オペレーションをマネジメントレベルで入出荷の分布を平均分布に近づけることが可能であれば，稼働率が向上し，機会損失額を大幅に低減可能であることが理解できた。また，初期導入コストと機会損失額は，日毎の入出荷命令の負荷を平準化することで低減可能であることが明らかになった。

第６章では，「小売業における SCM の機会損失の考察」をテーマに掲げ，第２章で定義した機会損失を踏まえ，第３章「百貨店における SCM の取り組みと機会損失に関する研究」，第４章「小売業におけるバックヤードの機会損失に関する研究」，第５章「自動倉庫における機会損失に関する研究」の結果を踏まえた結果，機会損失が存在することが明らかになったこと受けて，機会損失は，「目に見えない」課題，潜在化・内在化する課題を改善しないことにより

発生するものであると考察する。

企業経営において，企業発展の原動力として二つの側面があると考える。ひとつの側面はビジブル志向型経営(目に見える)であり，もう一つの側面はインビジブル抑止志向型経営（目に見えない）である。

ビジブル志向型経営の定義は，「目に見える」課題に対する経営であり，戦略思考型，需要創造型の経営方針である。一方，インビジブル抑止志向型経営の定義は，「目に見えない」課題，潜在化・内在化する課題に対する経営である。機会損失，チャンスロスにより失われている課題を解決し，企業の発展を図る経営方針である。すなわち，既存の枠組みの中で発生している「既存のルール」，「当たりまえ」，「慣習」という思考停止状態に陥っている課題を掘り起し，顕在化させ，機会損失として捉え，この機会損失を抑止することにより経営改善すると考察する。

本研究の結論としては，小売業における製造・流通・販売段階での情報共有化が図られていないことから生じる業務効率の低下が機会損失をもたらせていると考える。取引先と小売業との SCM による業務効率の向上がいまだに道半ばであり，経済産業省が推進している流通 BMS をはじめとする製配販の情報共有化がなかなか進まず，業務効率が向上していない。ここに大きな機会損失が発生していると考える。

これは，SCM の目的である，サプライチェーンを構成する個々の企業の業務を接続･統合させることができていない，すなわち資源の調達購買，製造，流通，販売の諸機能をサプライチェーンの中で単一実態として捉え，統合することが図られていないことを示している。今後 SCM を発展させていくためには，SCM の成果を得るためにサプライチェーンを構成する個々の企業が力を合わせて，業務を統合させ協調志向型 SCM を確立する。そして，次のステップとして個々の企業の限界を克服し，ブレークスルーする大きな手段として，共同化の方向性を模索することが重要である。これからの共同化の展望は，ネットワークをベースとする情報技術（IT）と管理理論の現実的展開をベースとして，共同指向型 SCM を推進する必要がある。

日本の小売業は非効率であり，利益率も低いとの指摘がある。これらは SCM が十分に機能していないことが原因の一つである。特に SCM への対応が遅れている百貨店などは，小売業としてその業態の存続が危ない状況になりつつある。今回の研究では，サプライチェーンにおける小売業の非効率化が明らかになった。すなわち SCM が全体最適になっていないことに起因すると考える。

以上，本論文は新たな知見としてサプライチェーン内で発生している機会損失を顕在化させ，サプライチェーンを構成する各企業がその機会損失を認識し，機会損失を排除することによる全体最適化を目指す是正を図ることによって，全体利益の向上を図ることが重要であることを立証した。

2. 今後の研究課題

本研究では，小売業からの視点から SCM における機会損失について論じてきた。

今後更なる研究を進めていくうえで，川上の製造から川下の流通・小売，最終的には顧客まで含めた SCM 全体の効果を得る方策，すなわち全体最適化を課題として，機会損失の切り口で研究を進めていく必要がある。

本研究では，運用業務の効率化を中心に議論を進めてきたが，顧客の消費動向など効果を数値化しにくいマーチャンダイズの側面についても機会損失の研究を進め，SCM の全体最適化を目指す方策を検討する必要があると考える。

付録1　バックヤード調査票

調　　査　　票

２０１５年6月

一般財団法人　日本ロジスティクスシステム学会

SCM 研究委員会

委員長　若林　敬造

委　員　生島　義英

1.　　調査目的

本調査では、大型小売店の店舗バックヤードを調査し、その結果を分析・考察することを目的とする。

2.　　調査対象

全国の上位大型小売店（百貨店・GMS・スーパーの代表的な企業）

3.　　調査時期

２０１５年6月２５日配布

4.　　調査結果

ご回答いただきましたデータは、研究論文の基礎データとして使用させて頂きますので個別企業名での利用は致しません。

5.　　ご返送締切日

２０１５年7月２０日

２０１５年7月２０日までに添付されている封筒に調査票を封入し、ご返信ください。

6.　　調査担当者

日本大学院生産工学研究科　博士後期課程　若林研究室　ＳCM研究委員会委員　生島　義英

大学所在地　〒275-8575 千葉県習志野市泉町 1-2-1　TEL.047-474-2201

担当者問合先　携帯電話　090-7202-0073　　e-mail: ikushima1197@outlook.com

7.　　調査責任者

日本大学生産工学部マネジメント工学科教授・博士(工学)　SCM 研究委員会委員長　若林　敬造

1．会社概要についてお伺いします。

① ２０１４年度の売上高はどのくらいですか。　　（　　百万円円）

② 仕入形態別売上割合はどのくらいですか。

（　本仕（買取仕入）　　％・売仕（消化仕入）　　％・賃貸　　％）

③ ２０１４年度の売買差益率はどのくらいですか。　　（　　％）

④ ２０１４年度末での従業員数はどれくらいですか。

（正社員　　人・契約社員　　人・パート社員　　人）

⑤ 店舗数はどのくらいですか。　　（　　店舗）

⑥ 総売場面積はどのくらいですか。　　（　　㎡）

⑦ ㎡当り年平均売上高はどのくらいですか。　　（　　円）

⑧ 商品別売上構成と商品回転率はどれくらいですか。

商品分類	売上割合（%）	本仕売上割合*1	売仕売上割合*1	本仕商品回転率（年間）	備考
衣料品（婦人・紳士・子供）	%	%	%		
身の回り品	%	%	%		
食料品	%	%	%		
家庭用品（家具・家電含む）	%	%	%		
その他（　　）	%	%	%		
食堂喫茶	%	%	%	—	

*1：本仕：売仕の売上割合をご記入ください。　例：30%：70%

2．在庫管理について

① 全社の平均在庫金額はどのくらいですか。　　（　　千円）

② 全社の年間商品回転率はどのくらいですか。　　（　　）

③ 全社場所別在庫金額の割合はどのくらいですか。

（店頭：店舗バックヤード：物流センター　＝　　％：　　％：　　％）

3．自社物流センターについてお伺いします。

① 自社物流センターはありますか。　（　有　・　無　）

② 自社物流センターの数はいくつありますか。　（　　箇所）

③ 自社物流センターの総面積はどのくらいですか。　（　　㎡）

④ 自社物流センターの所有形態の自社所有：賃貸：外部委託比率はどれくらいですか。

（ 自社所有　：　賃貸　：　外部委託＝　　：　　：　　）

⑤ 自社物流センター在庫の保有形態はどのような形態ですか。

（ 自社保有　・　ベンダー（問屋）保有　・　メーカー保有　・　共同保有（　　））

4．店舗バックヤードについてお伺いします。

① 売場面積とバックヤード面積の比率はどのくらいですか。（建物総面積を100%とした場合）

（ 売場面積　：　バックヤード面積　＝　　%：　　%）

② 店舗バックヤード商品ストックは十分に確保されていますか。　（ はい　・　いいえ ）

③ 後方通路などに商品・什器・備品などが溢れていますか。　（ はい　・　いいえ ）

④ 繁忙期など一時的に商品在庫が膨らむ時期はどこに商品を退避させていますか。

（ 自社物流センター・後方通路など・取引先倉庫・外部倉庫・その他（　　））

⑤ 今後の方向性として、バックヤードの縮小を検討していますか。　（ はい　・　いいえ ）

⑥ バックヤード縮小によって得られた面積を何に活用したいと考えていますか。

（ 売場化（売場面積の拡大）・テナント貸・賃貸主への返却・その他（　　））

⑦ 店舗バックヤードの機能別面積割合はどのくらいですか。（建物総面積を100%とした場合）

機能	面積割合（%）	備考
商品ストック	%	
事務所	%	
設備・後方施設	%	階段・通路・機械室・電気室など

*1：店舗により数値が異なる場合は、代表的な店舗の数値をご記入いただければ幸いです。

⑧ バックヤード縮小を図るために具体的に検討していることがあればお教えください。

5．在庫合理化についてお伺いします。

① 在庫合理化案に優先順位をつけてください。

内容	優先順位	備考
店頭商品在庫の合理化		
店舗バックヤード商品在庫の合理化		
物流センター商品在庫の合理化		
その他（　　）		

② 在庫合理化の手段として、下記表のうち何を選択しますか、既に導入している手段は「導入済」欄に、将来導入を検討している手段は「導入予定」欄に「○、×」を記入してください。在庫合理化手段の優先順位をつけてください。

項目	導入済	導入予定	優先順位
単品管理による売れ筋、死筋の把握			
売れ行きに応じてフェイス数を割当の徹底			
ＥＯＳ導入			
自動補充発注システム導入			
納品リードタイムの短縮化			
ＳＣＭ検品（伝票レス・検品レス・商品勘定連携）			
多頻度納品による在庫圧縮			
商品鮮度管理・賞味期限管理システム導入			
店舗バックヤードストックの縮小			
取引先へＰＯＳ売上データを開示し、取引先納品提案（ＶＭＩ）			
取引先に在庫データを開示し、効率的な納品体制の構築			
取引先とのＥＤＩ推進（流通ＢＭＳ・ｅＭＰ・IQRS など）			
自社物流センターの整理統合もしくは廃止			
ＩＣタグの導入（ＲＦＩＤ）			

5-②項掲げた以外の在庫合理化策で御社が検討されている合理化策があればお教えください。

6．ご回答者様のプロフィールについてお伺いします。

① 御社名、ご所属部署、ご芳名、ご連絡先などをお答えください。

御社名	
ご所属部署	
お役職名	
お名前	
ご連絡先電話	
メールアドレス	

ご協力ありがとうございました

付録2.　自動倉庫シュミレーションプログラムフローチャート

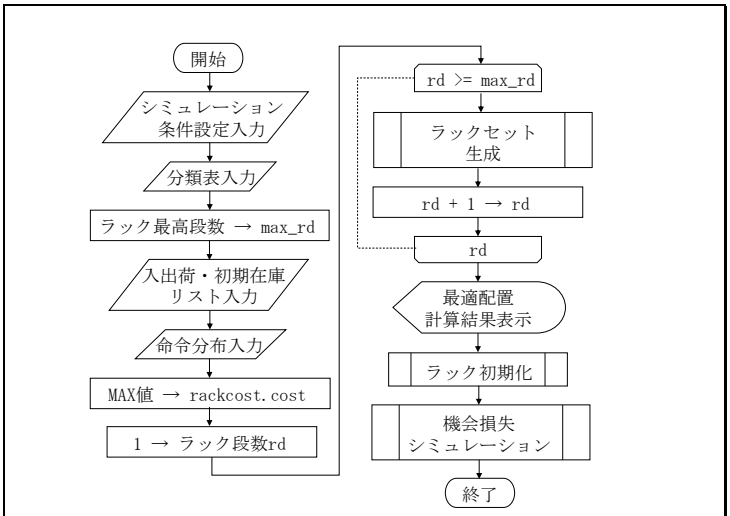

図1 フローチャート（メインルーチン）

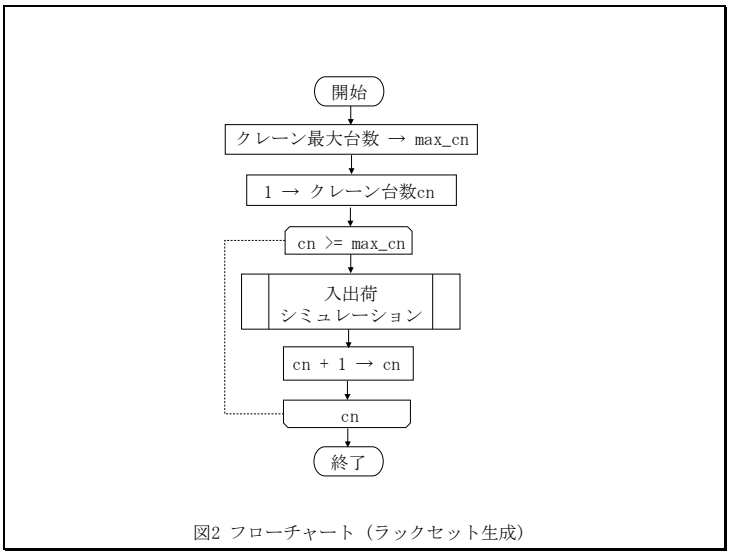

図2 フローチャート（ラックセット生成）

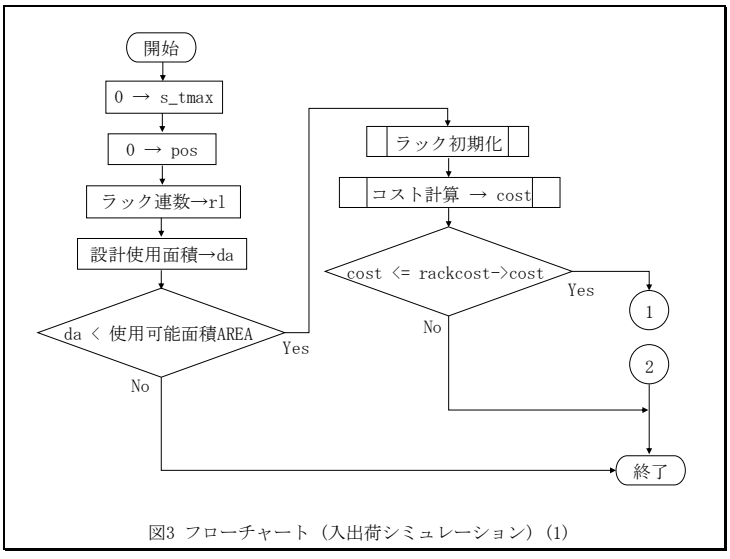

図3 フローチャート（入出荷シミュレーション）(1)

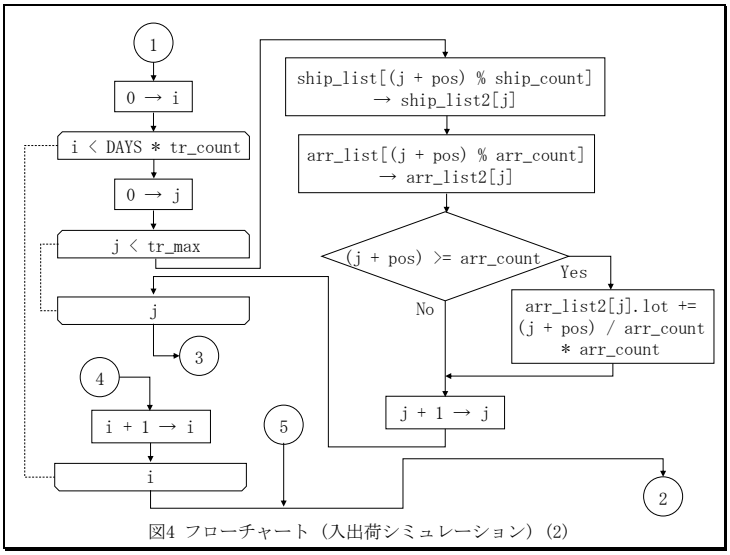

図4 フローチャート（入出荷シミュレーション）(2)

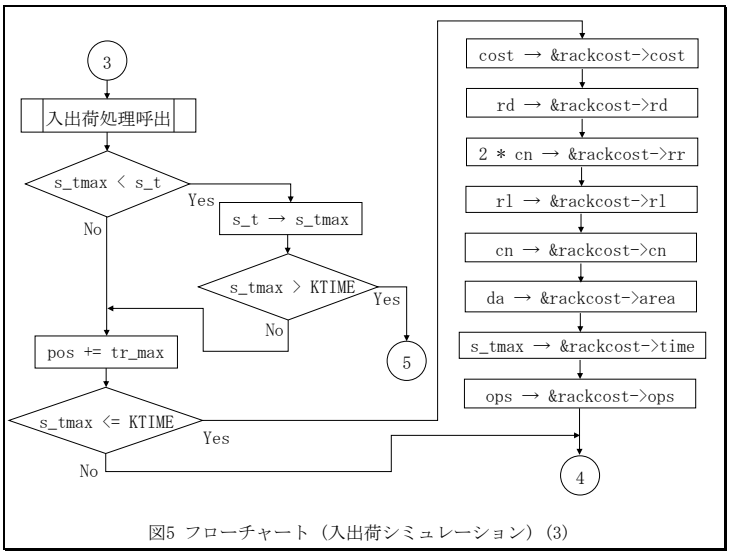

図5 フローチャート（入出荷シミュレーション）(3)

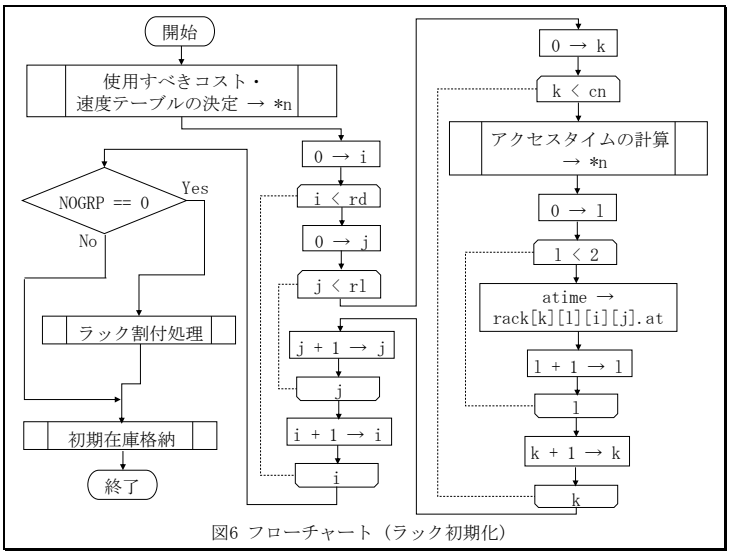

図6 フローチャート（ラック初期化）

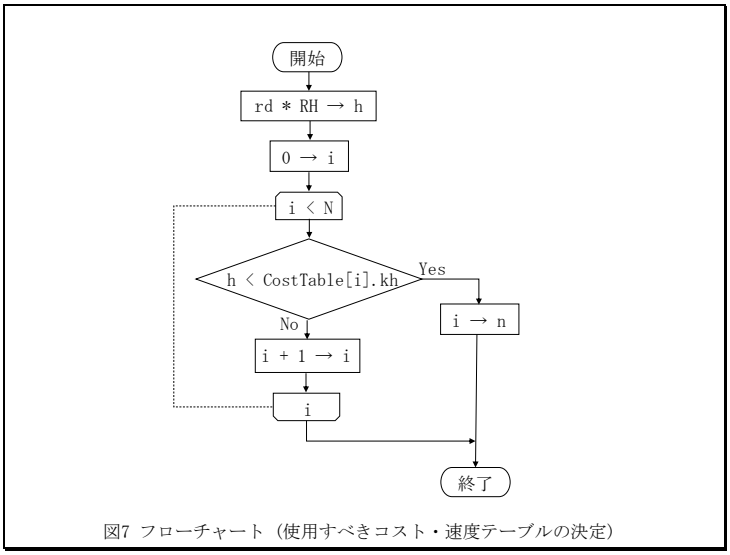

図7 フローチャート（使用すべきコスト・速度テーブルの決定）

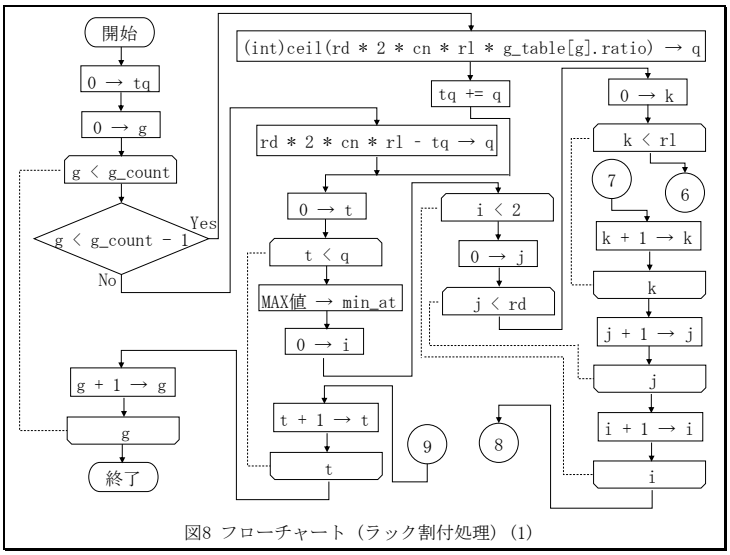

図8 フローチャート（ラック割付処理）(1)

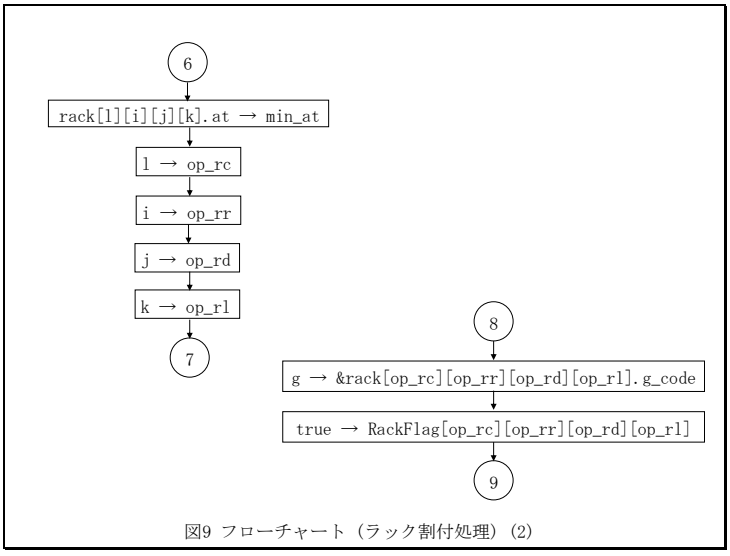

図9 フローチャート（ラック割付処理）(2)

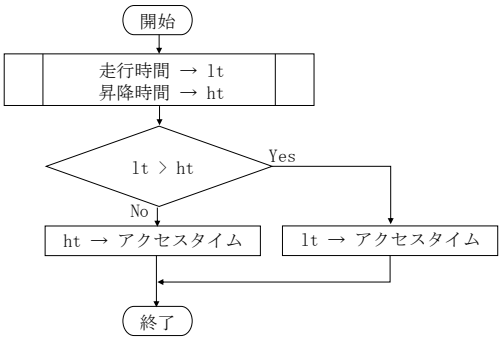

図10 フローチャート（アクセスタイムの計算）

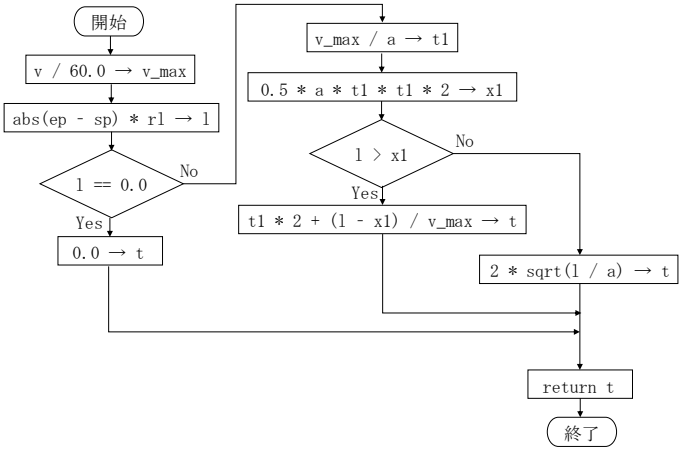

図11 フローチャート（走行・昇降時間の計算）

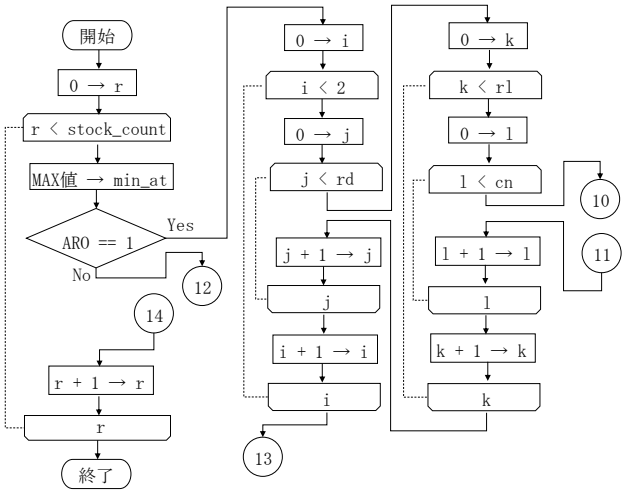

図12 フローチャート（初期在庫格納）(1)

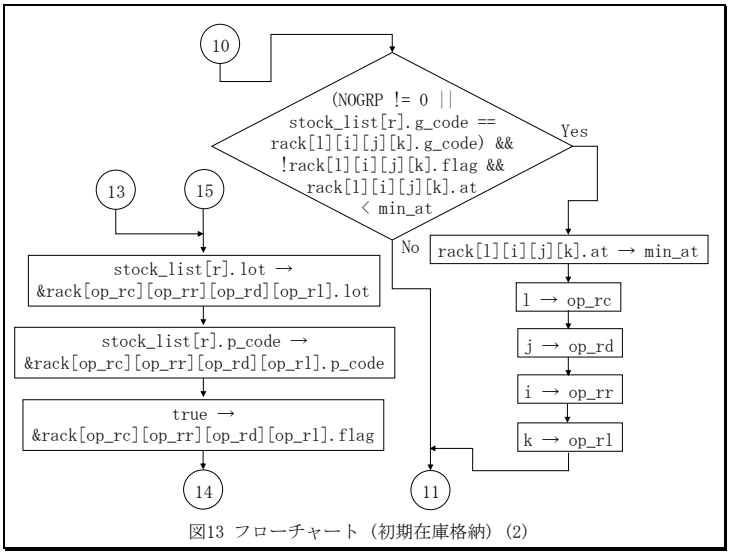

図13 フローチャート（初期在庫格納）(2)

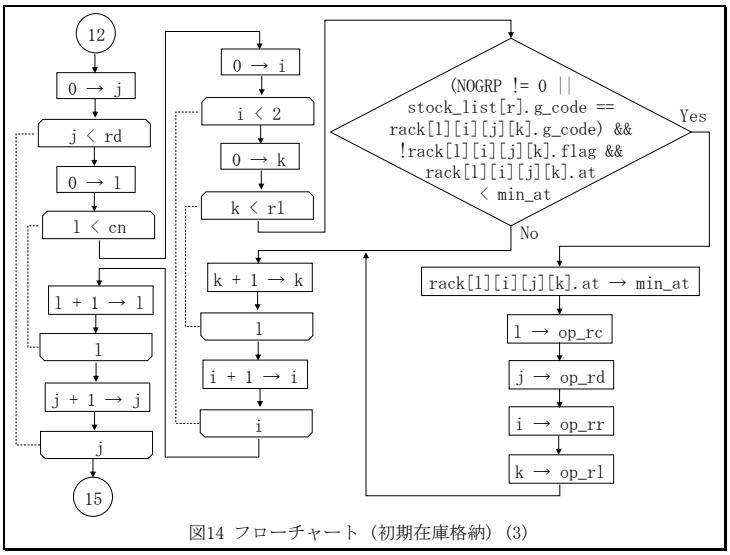

図14 フローチャート（初期在庫格納）(3)

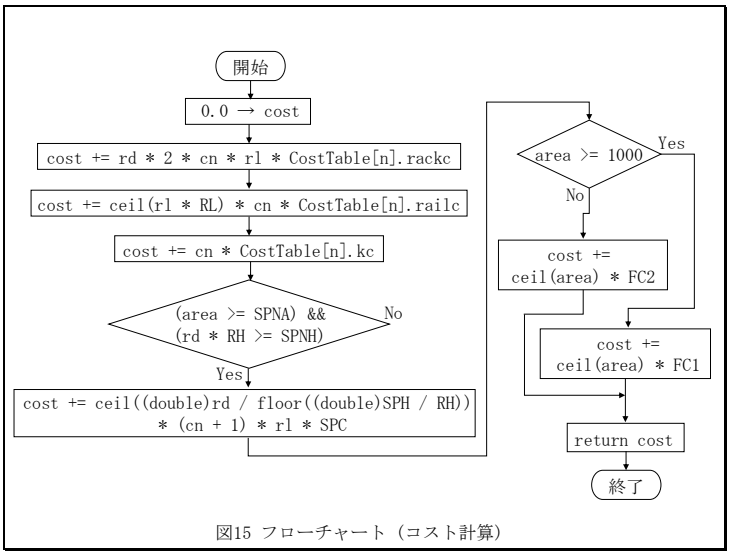

図15 フローチャート（コスト計算）

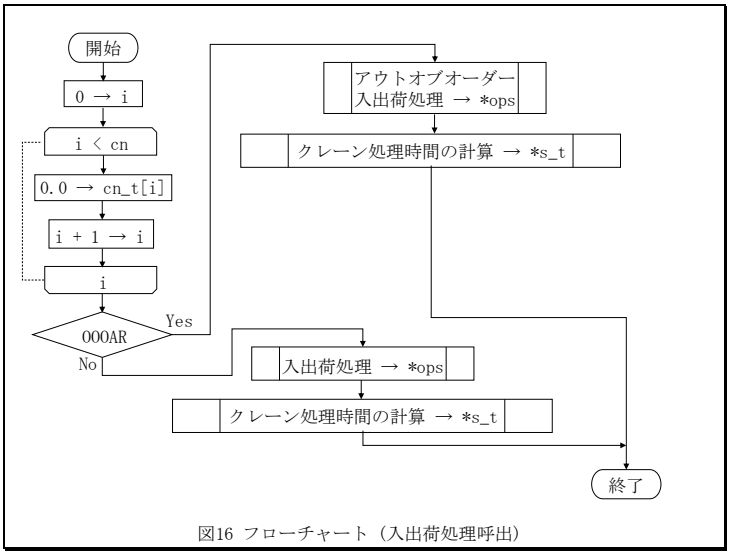

図16 フローチャート（入出荷処理呼出）

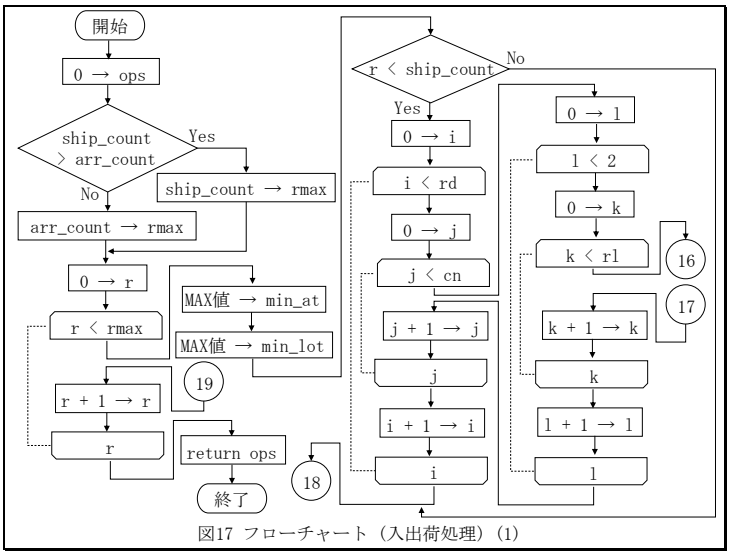

図17 フローチャート（入出荷処理）(1)

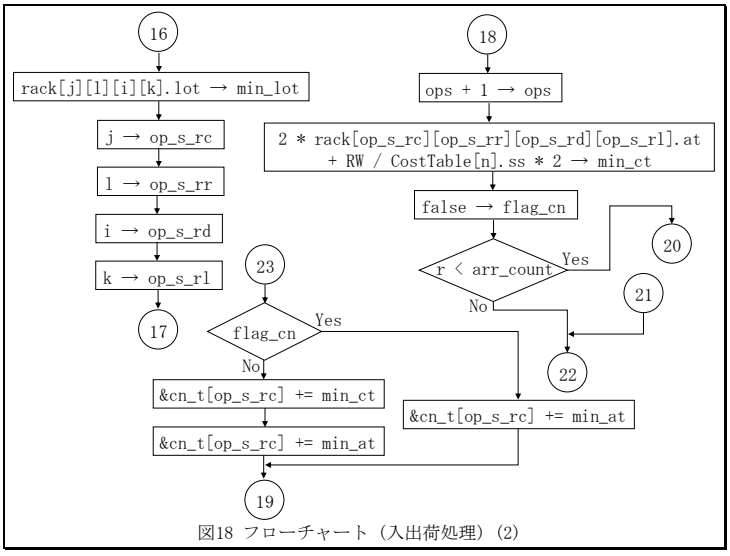

図18 フローチャート（入出荷処理）(2)

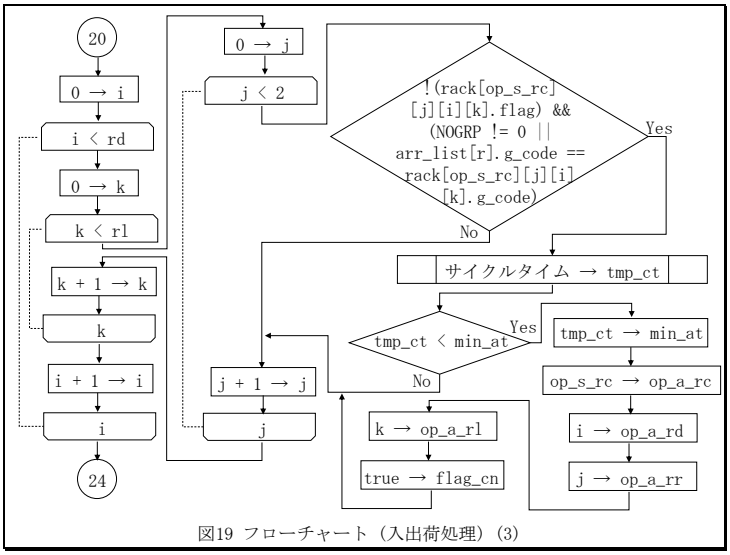

図19 フローチャート（入出荷処理）(3)

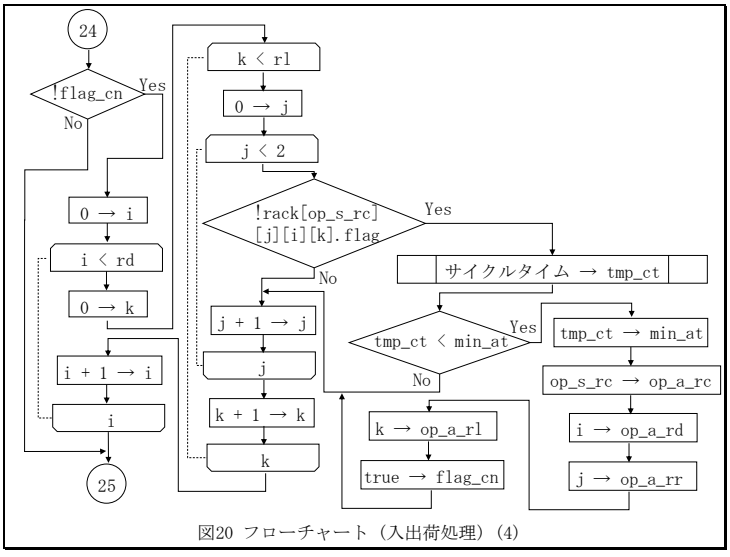

図20 フローチャート（入出荷処理）(4)

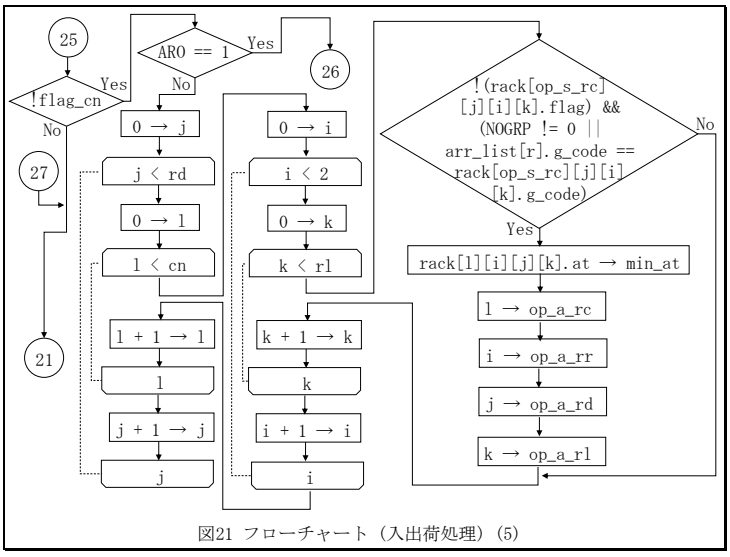

図21 フローチャート（入出荷処理）(5)

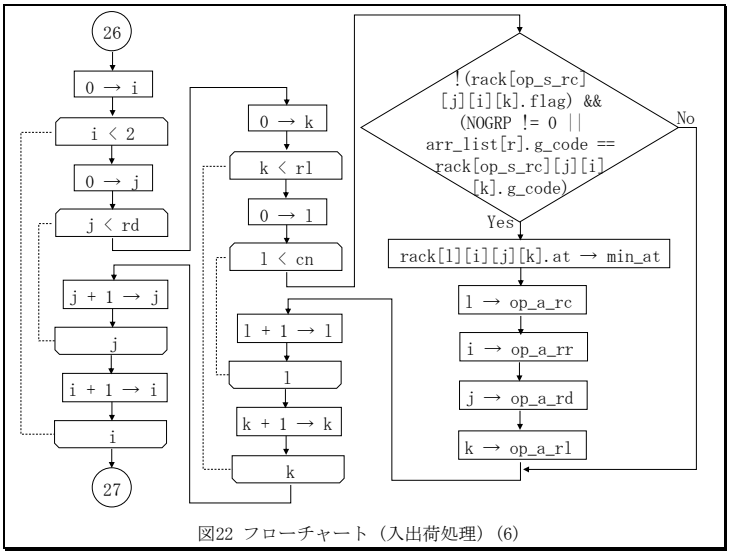

図22 フローチャート（入出荷処理）(6)

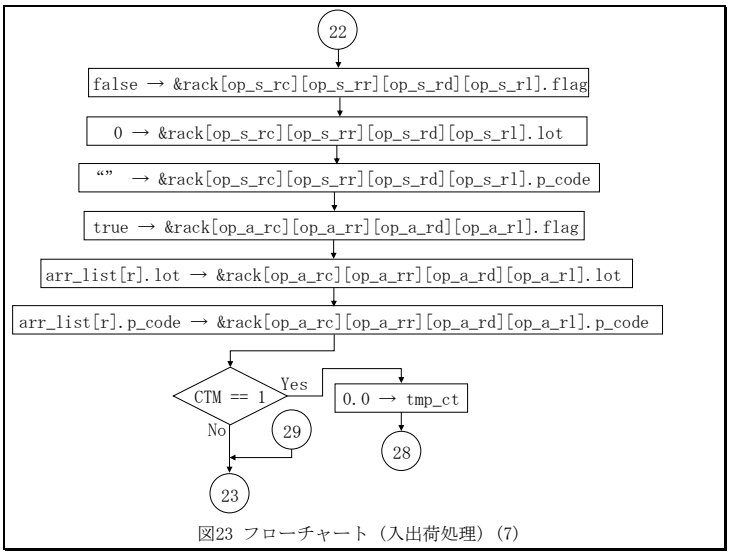

図23 フローチャート（入出荷処理）(7)

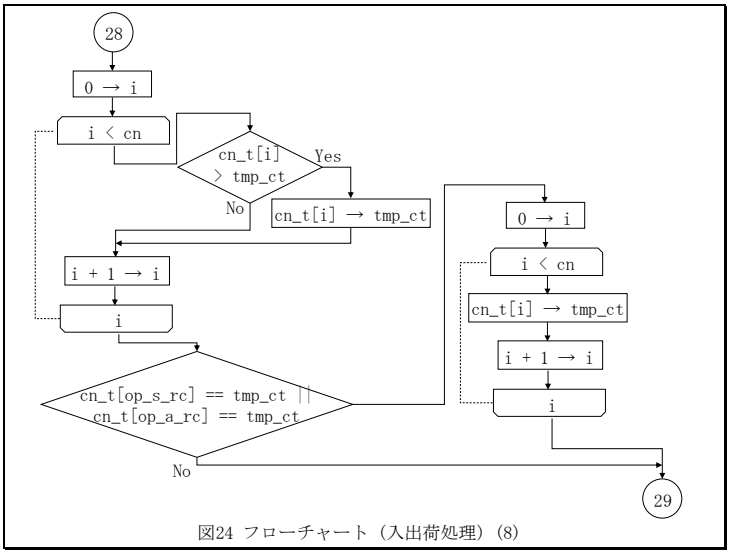

図24 フローチャート（入出荷処理）(8)

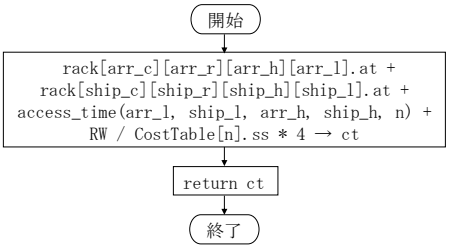

図25 フローチャート（サイクルタイム計算）

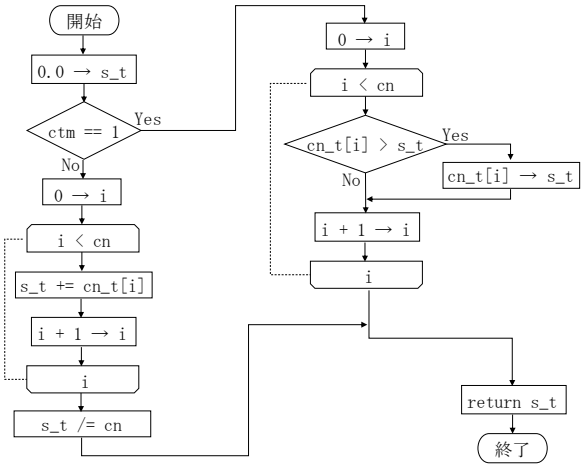

図26 フローチャート（クレーン処理時間の計算）

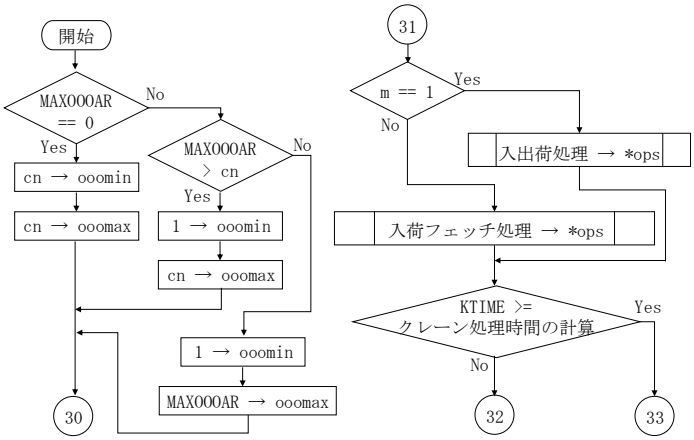

図27 フローチャート（アウトオブオーダー入出荷処理）(1)

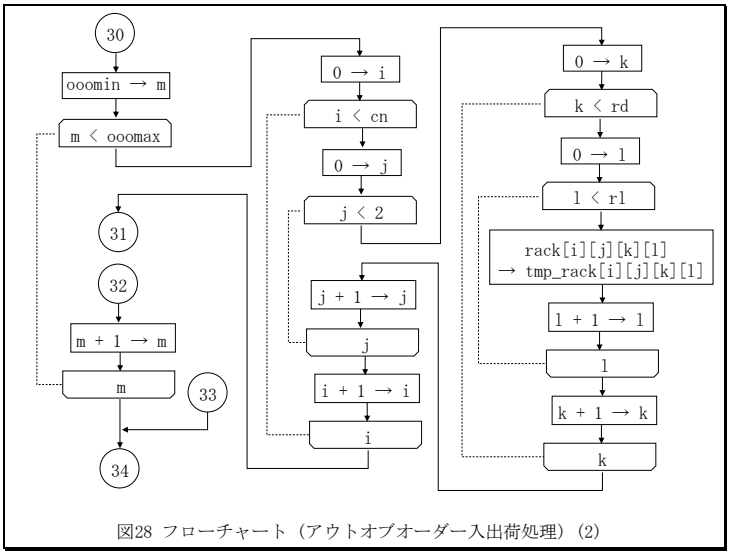

図28 フローチャート（アウトオブオーダー入出荷処理）(2)

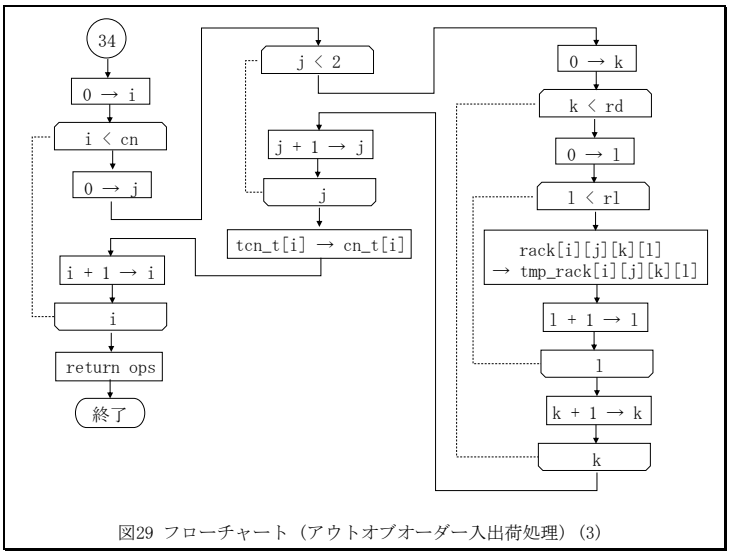

図29 フローチャート（アウトオブオーダー入出荷処理）(3)

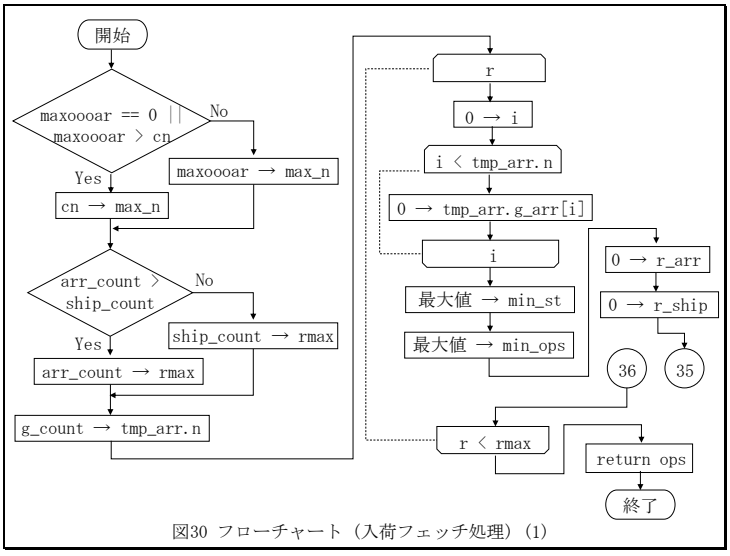

図30 フローチャート（入荷フェッチ処理）(1)

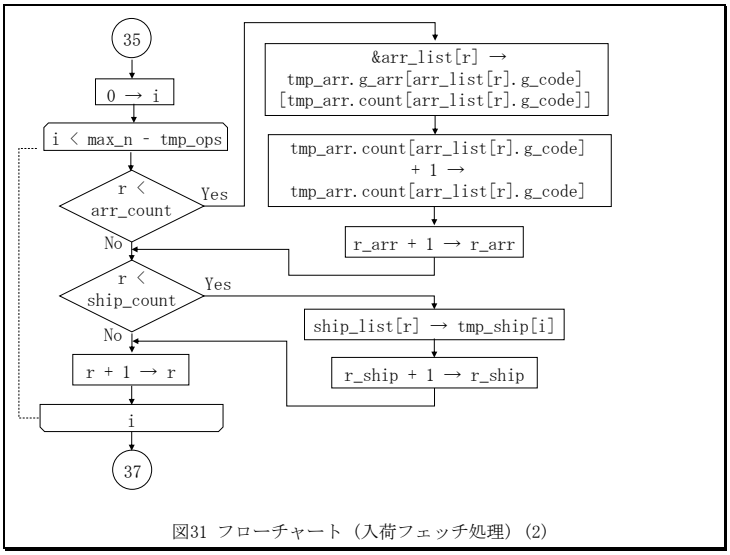

図31 フローチャート（入荷フェッチ処理）(2)

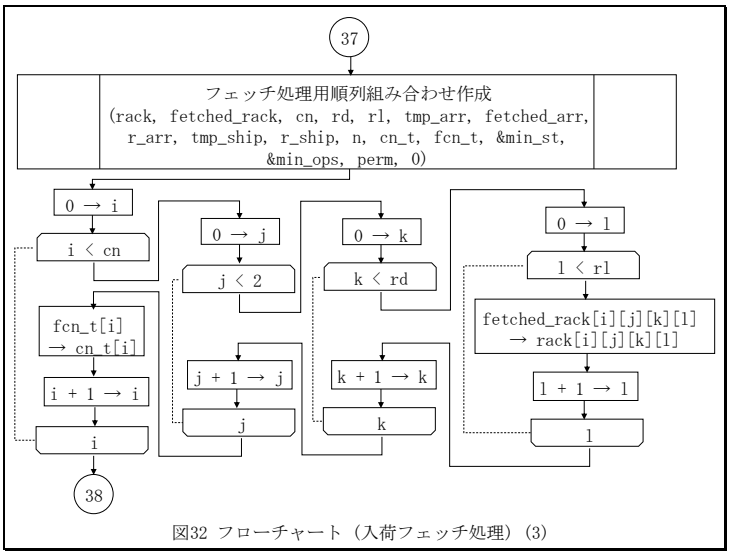

図32 フローチャート（入荷フェッチ処理）(3)

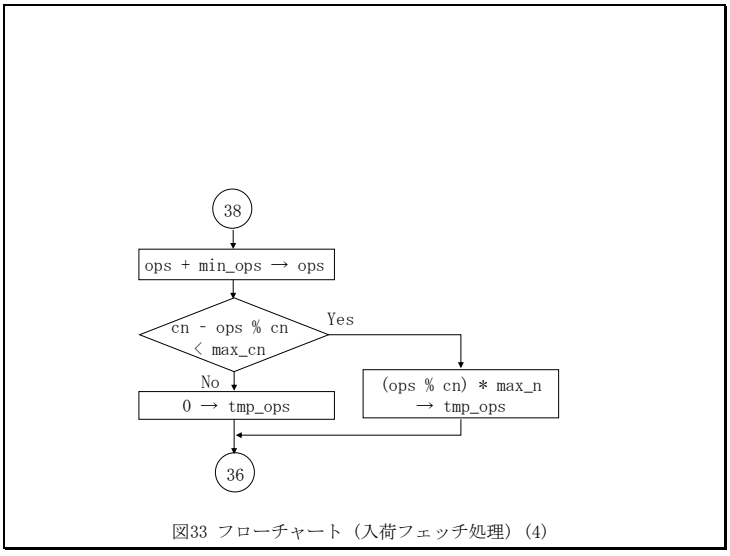

図33 フローチャート（入荷フェッチ処理）(4)

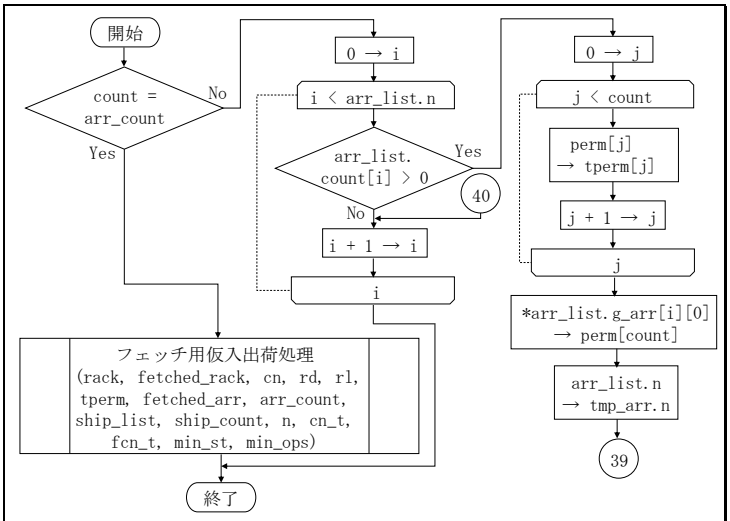

図34 フローチャート（フェッチ処理用順列組み合わせ作成）(1)

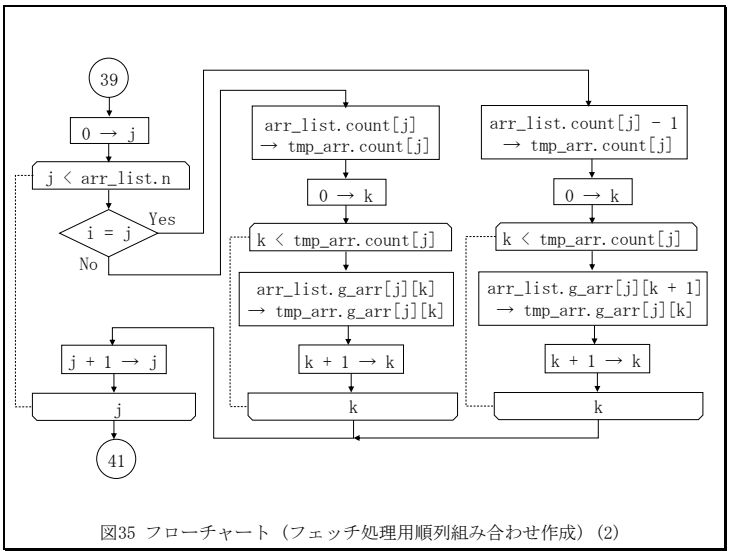

図35 フローチャート（フェッチ処理用順列組み合わせ作成）(2)

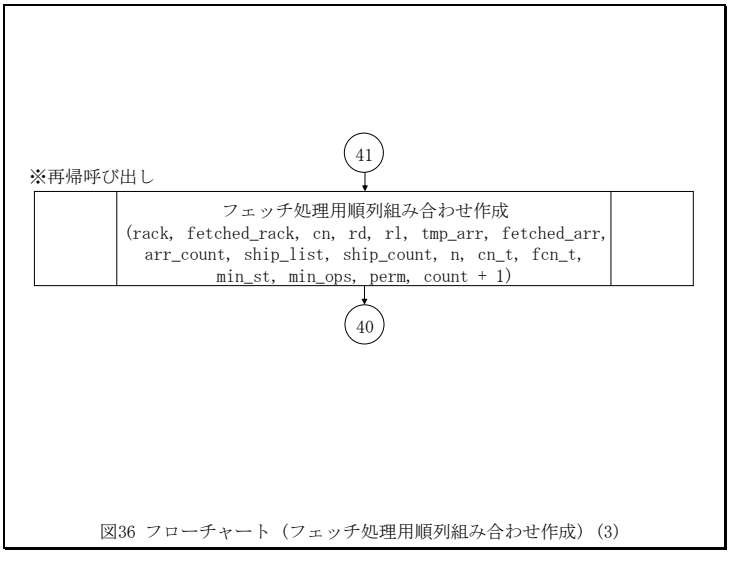

図36 フローチャート（フェッチ処理用順列組み合わせ作成）(3)

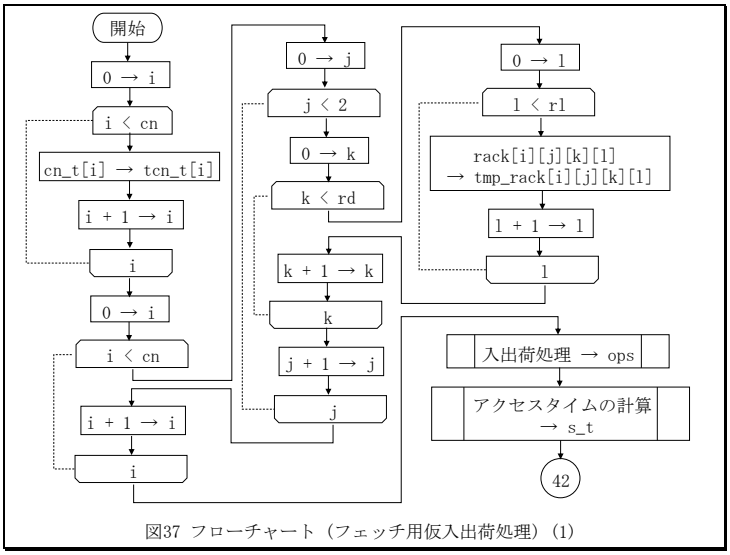

図37 フローチャート（フェッチ用仮入出荷処理）(1)

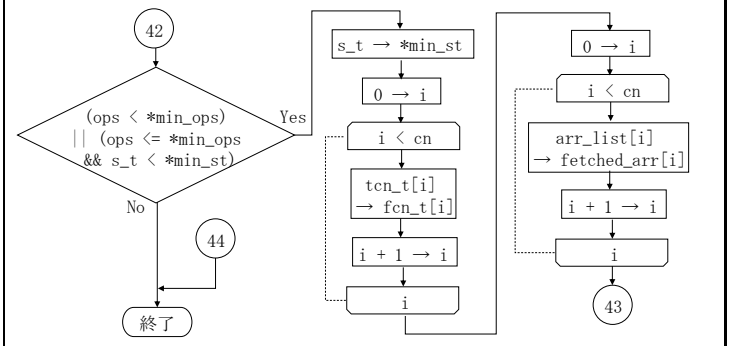

図38 フローチャート（フェッチ用仮入出荷処理）(2)

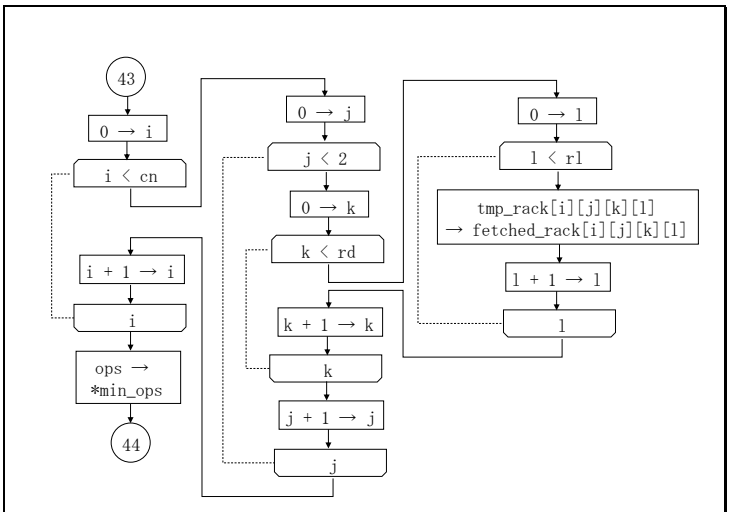

図39 フローチャート（フェッチ用仮入出荷処理）(3)

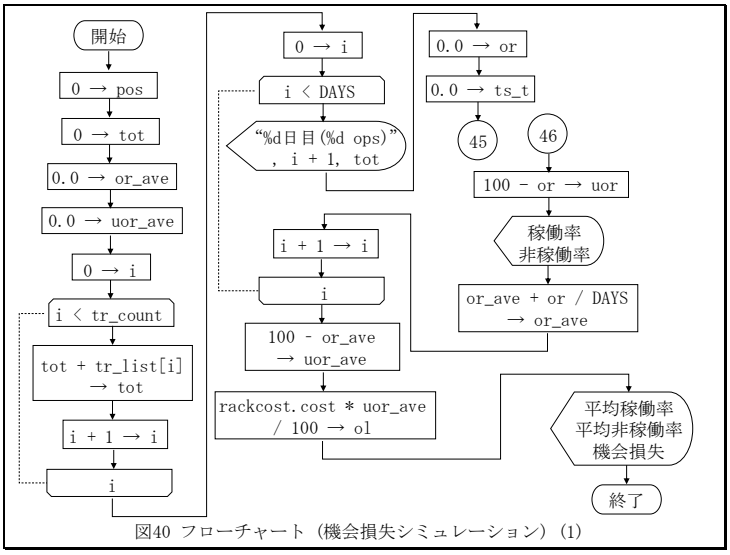

図40 フローチャート（機会損失シミュレーション）(1)

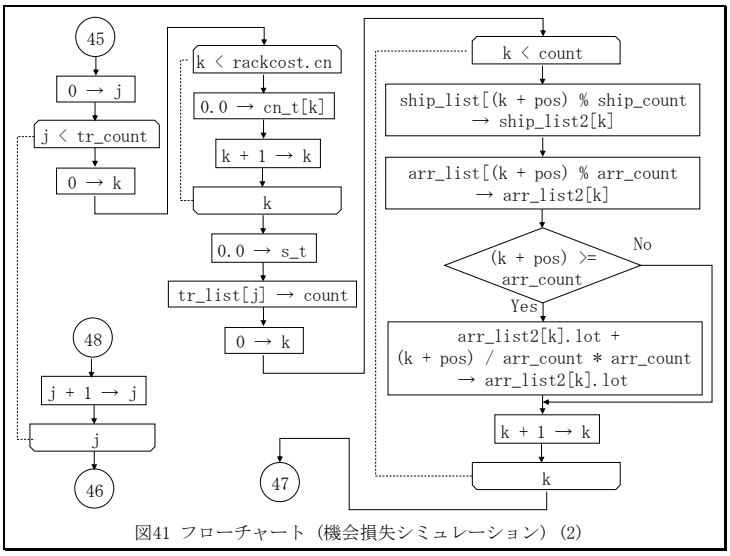

図41 フローチャート（機会損失シミュレーション）(2)

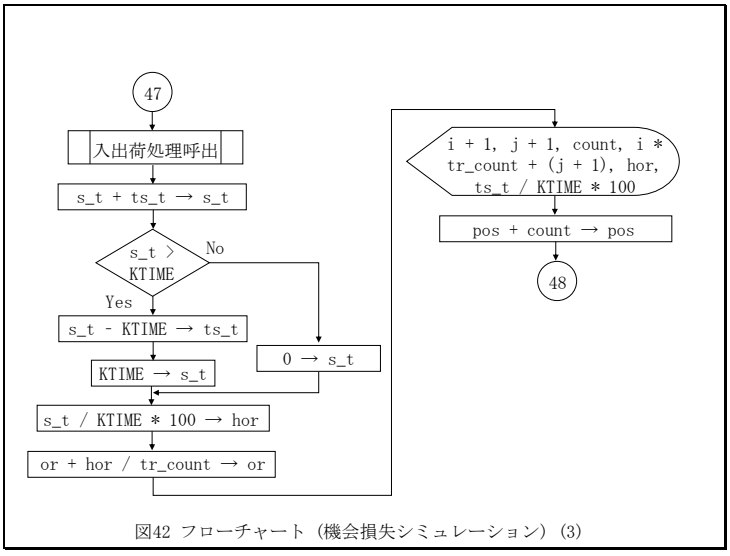

図42 フローチャート（機会損失シミュレーション）(3)

付録3　自動倉庫シュミレーションプログラムソースリスト

```
//*************************************************************************//
//  自動倉庫シミュレーションプログラム
// 2015.12.05 Y.Ikushima　T. Sato
// <AutomaticWarehouse.cpp>
//*************************************************************************//

#include "stdio.h"
#include "stdlib.h"
#include "string.h"
#include "math.h"

#ifdef _MSC_VER
#include "windows.h" //Windows 用: Query Performance Counter 使用のため
#pragma warning(disable:4996)
#pragma warning(disable:4703)
#else
#include "time.h" //Linux 用: clock_gettime / Mac OSX 用: gettimeofday 使用のため
#endif

#define CONFIG     "config.csv"     //初期設定ファイル (config.csv)
#define GROUP      "group.csv"      //分類リスト          (group.csv)
#define ARRLIST    "arrival.csv"    //入荷リスト          (arrival.csv)
#define SHIPLIST "shipment.csv" //出荷リスト          (shipment.csv)
#define STOCK      "stock.csv"      //初期在庫リスト      (stock.csv)
#define TRLIST     "trlist.csv"     //命令数分布表        (trlist.csv)

//ラックデータ
typedef struct {
        char p_code[20]; //商品コード
        int g_code; //分類コード番号
        int lot; //ロット番号
        bool flag; //在庫の有無
        double at; //入出荷口からのアクセスタイム
} rack_t;

//分類リスト
typedef struct {
        char g_code[20]; //分類コード
        double ratio; //割合(100% = 1)
} group_t;
```

```
//入荷リスト
typedef struct {
	char p_code[20]; //商品コード
	int g_code; //分類コード番号
	int lot; //ロット番号
} arr_t;

//アウトオブオーダー入荷処理用分類毎入荷リスト
typedef struct {
	int n; //分類数
	int *count; //分類毎の入荷数
	arr_t ***g_arr; //分類毎の入荷リストポインタ
} arrlist_t;

//出荷リスト
typedef struct {
	char p_code[20]; //商品コード
	int g_code; //分類コード番号
} ship_t;

//クレーンデータ
typedef struct {
	double kh; //高さ条件[m 以下]
	double kc; //クレーンコスト[万円/台]
	double ls; //走行速度[m/min.]
	double hs; //昇降速度[m/min.]
	double ss; //シャトル速度[m/min.]
	double al; //走行加速度[m/sec^2]
	double ah; //昇降加速度[m/sec^2]
	double railc; //レールコスト[万円/m]
	double rackc; //ラックコスト[万円/個]
} cost_t;

cost_t *CostTable;

//倉庫コスト
typedef struct {
	double cost; //コスト[万円]
	int rd; //ラック段数
	int rr; //ラック列数
	int rl; //ラック連数
	int cn; //クレーン台数
	double area; //面積[m^2]
```

```
        double time; //クレーン処理時間[sec.]
        int ops; //クレーンの処理回数
} rackcost_t;

//ラック個数
int RS;

//ラックサイズ
double RH; //ラック高さ[m]
double RL; //ラック長さ[m]
double RW; //ラック幅[m]

//倉庫高さ制約[m]
double H;

//与件の使用面積[m^2]
double AREA;

//クレーン処理時間の上限[sec.]
double KTIME;

//スプリンクラー条件
double SPC; //スプリンクラーコスト[万円/個]
double SPH; //設置高さ間隔[m]
double SPNH; //設置不要高さ[m]
double SPNA; //設置不要面積[m^2]

//耐震床コスト[万円/m^2]
double FC1; //1000[m^2]以上
double FC2; //1000[m^2]未満

//クレーン・ラック・レールの高さによる条件数
int N = 0;

//ラック入荷順序モード
int ARO = 0; //(0:端からラスタスキャン、1:同一クレーンの担当するラックが連続しないようにスキャン)

//クレーン処理時間算出モード
int CTM = 0; //(0: 各クレーンの処理時間の平均、1：並列処理を考慮した待機を含む各クレーン処理時間のうち最長のもの)

//アウトオブオーダー入荷モード
```

```
int OOOAR = 0; //(0：入荷リスト通りに入荷処理を実行、2：MAXOOOAR 分の範囲の入荷トランザクションをアクセスタイム最小になるよう並べ替えて実行)
int MAXOOOAR = 0; //アウトオブオーダー処理の先読み上限 (0：クレーン本数を上限)

//ラックの割付モード
int NOGRP = 0; //(0：ABC 分類に基づくラックのグループ割付、1：グループの割付をしない)

//設定したシミュレーション条件下におけるラックの組み方のパターン数
int count1 = 0; //count1:  面積≦AREA かつ処理時間≦KTIME のパターン数
int count2 = 0; //count2:  面積≦AREA のパターン数

int DAYS = 1; //機会損失の推移を計算する日数

static char seps[] = ","; // [,]文字による区切り-->CSV ファイル

void makeRackSet(int rd, rackcost_t* rackcost, group_t* g_table, int g_count, arr_t* arr_list, int arr_count, ship_t* ship_list, int ship_count, arr_t* stock_list, int stock_count, int tr_max, int tr_count);

void funcSimulation(int rd, int cn, rackcost_t *rackcost, group_t *g_table, int g_count, arr_t *arr_list, int arr_count, ship_t *ship_list, int ship_count, arr_t *stock_list, int stock_count, int tr_max, int tr_count);

void setGroupToRack(rack_t**** rack, int cn, int rd, int rl, group_t* g_table, int g_count);
void setStockToRack(rack_t**** rack, int cn, int rd, int rl, arr_t* stock_list, int stock_count);

int funcFetchArr(rack_t ****rack, int cn, int rd, int rl, arr_t *arr_list, int arr_count, ship_t *ship_list, int ship_count, int n, double *cn_t, int g_count);
int funcFetchArrOrder(rack_t ****rack, int cn, int rd, int rl, arr_t *arr_list, int arr_count, ship_t *ship_list, int ship_count, int n, double *cn_t, int maxoooar, int g_count);
void PermutationFetchArr(rack_t ****rack, rack_t ****fetched_rack, int cn, int rd, int rl, arrlist_t arr_list, arr_t *fetched_arr, int arr_count, ship_t *ship_list, int ship_count, int n, double *cn_t, double *fcn_t, double *min_st, int *min_ops, arr_t *tperm, int count);
void funcFetchArrShip(rack_t ****rack, rack_t ****fetched_rack, int cn, int rd, int rl, arr_t *arr_list, arr_t *fetched_arr, int arr_count, ship_t *ship_list, int ship_count, int n, double *cn_t, double *fcn_t, double *min_st, int *min_ops);

int funcArrShip(rack_t**** rack, int cn, int rd, int rl, arr_t* arr_list, int arr_count, ship_t* ship_list, int ship_count, int n, double* cn_t);
double funcCost(int cn, int rd, int rl, double area, int n);
double funcProcessingTime(int cn, double *cn_t, int ctm);

double access_time(int sl, int el, int sh, int eh, int n);
double v_time(int sr, int er, double rl, double v, double a);
int condition(int rd);
```

```
double cycle_time(rack_t**** rack, int arr_c, int arr_r, int arr_h, int arr_l, int ship_c, int ship_r, int ship_h, int ship_l, int n);

void readInitialFile(char* filename);
void readargc(int argc, char **argv);
void printInitialValues();

void InitRack(int rd, int rl, int cn, rack_t ****rack, int *n, group_t *g_table, int g_count, arr_t *stock_list, int stock_count);
void ArrShip(double *cn_t, rack_t ****rack, int rd, int rl, int cn, int n, arr_t *arr_list, int arr_count, ship_t *ship_list, int ship_count, int g_count, int *ops, double *s_t);
void OpportunityLoss(rackcost_t rackcost, rack_t ****rack, int n, arr_t *arr_list, int arr_count, ship_t *ship_list, int ship_count, int g_count, int* tr_list, int tr_count);

group_t* readGroupFile(int* g_count, char* filename);
arr_t* readArrList(group_t* g_table, int g_count, int* arr_count, char* filename);
ship_t* readShipList(group_t* g_table, int g_count, int* ship_count, char* filename);
int* readTransactions(int* tr_count, char* filename);

FILE* freadopen(char* filename);
int readFileRow(FILE* fp);
void chop_crlf(char* buff);

double timer();

rack_t**** allocRackSet(int rd, int cn, int rl);
group_t* allocGroupTable(int row);
bool**** allocRackFlag(int cn, int rd, int rl);
double* allocCraneTime(int cn);

void freeRackFlag(bool**** flag, int cn, int rd);
void freeRackSet(rack_t**** rack, int cn, int rd);

//メイン関数
int main(int argc, char *argv[])
{
        //*************************************************************************//
        group_t *g_table; //分類表
        arr_t *arr_list; //入荷リスト
        ship_t *ship_list; //出荷リスト
        arr_t *stock_list; //初期在庫リスト
        rackcost_t rackcost; //最小コスト
```

```
int max_rd; //ラック最高段数
int rd; //設計時のラック段数
int g_count; //分類数
int arr_count; //入荷命令数
int ship_count; //出荷命令数
int stock_count; //初期在庫数
int i;
double tstart, tend;
rack_t ****rack;
int n; //コストテーブル番号
int *tr_list; //命令分布
int tr_count;
int tr_max = 0;
//*************************************************************************//

//初期化時間計測スタート
tstart = timer();

//シミュレーション条件設定ファイルの読み込み
readInitialFile(CONFIG);

//コマンドライン引数によるシミュレーション条件設定の変更
readargc(argc, argv);

//シミュレーション条件設定の表示(デバッグ用)
//printInitialValues();

//分類表の読み込み
g_table = readGroupFile(&g_count, GROUP);

//ラック最高段数
max_rd = (int)floor(H / RH); //小数点以下切り捨て

//入荷リスト読み込み
arr_list = readArrList(g_table, g_count, &arr_count, ARRLIST);

//出荷リスト読み込み
ship_list = readShipList(g_table, g_count, &ship_count, SHIPLIST);

//初期在庫リスト読み込み
stock_list = readArrList(g_table, g_count, &stock_count, STOCK);

//命令分布表読み込み
```

```
	tr_list = readTransactions(&tr_count, TRLIST);

	for (i = 0; i < tr_count; i++) {
		if (tr_max < tr_list[i]) {
			tr_max = tr_list[i];
		}
	}

	//ラックコストの初期化
	rackcost.cost = 1e100;
	rackcost.cn = 0;
	rackcost.rd = 0;
	rackcost.rr = 0;
	rackcost.rl = 0;
	rackcost.area = 0.0;
	rackcost.time = 0.0;
	rackcost.ops = 0;

	//自動倉庫の設計

	//初期化時間表示
	tend = timer();
	printf("** Initializing Time: %.2lf [ms]¥n", (tend - tstart) * 1000);

	//計算時間計測スタート
	tstart = timer();

	//ラック 1 段～max_rd 段
	for (rd = 1; rd <= max_rd; rd++) {
		makeRackSet(rd, &rackcost, g_table, g_count, arr_list, arr_count, ship_list, ship_count,
stock_list, stock_count, tr_max, tr_count);
	}

	//出力
	printf("** Minimum Cost: %.1lf[万円]¥n(Rack[%d][%d][%d], cn=%d[台], area=%.2lf[m^2],
time=%.2lf[min.], %d ops)¥n", rackcost.cost, rackcost.rd, rackcost.rr, rackcost.rl, rackcost.cn, rackcost.area,
rackcost.time / 60, rackcost.ops);
	printf("処理時間≦%.2lf[sec.]¥n", KTIME);
	//printf("処理時間≦%.2lf[sec.]: %d / %d パターン¥n", KTIME, count1, count2);

	//計算時間表示
	tend = timer();
	printf("** Calculation Time: %.2lf [ms]¥n", (tend - tstart) * 1000);
```

```
	//計算時間計測スタート
	tstart = timer();

	//機会損失シミュレーション
	rack = allocRackSet(rackcost.rd, rackcost.cn, rackcost.rl);
	InitRack(rackcost.rd, rackcost.rl, rackcost.cn, rack, &n, g_table, g_count, stock_list, stock_count);
	OpportunityLoss(rackcost, rack, n, arr_list, arr_count, ship_list, ship_count, g_count, tr_list, tr_count);

	//計算時間表示
	tend = timer();
	printf("** Calculation Time: %.2lf [ms]¥n", (tend - tstart) * 1000);

	//動的配列の解放
	free(g_table);
	free(CostTable);

	freeRackSet(rack, rackcost.cn, rackcost.rd);
	free(ship_list);
	free(arr_list);
	free(stock_list);

	return 0;
}

void OpportunityLoss(rackcost_t rackcost, rack_t ****rack, int n, arr_t *arr_list, int arr_count, ship_t *ship_list,
int ship_count, int g_count, int* tr_list, int tr_count)
{
	//*************************************************************************//
	int i, j, k;
	double *cn_t;
	int ops;
	double s_t, ts_t;
	ship_t *ship_list2;
	arr_t *arr_list2;
	int pos = 0, count, tot = 0;
	double hor, or, uor, or_ave = 0.0, uor_ave = 0.0, ol; //時間あたり稼働率, 稼働率, 非稼働率, 平均
稼働率, 平均非稼働率, 機会損失額
	//*************************************************************************//

	printf("** 稼働率・非稼働率・機会損失 計算¥n");
	printf("日, 時間, 命令数, 累積時間, 稼働率, 処理未完¥n");
```

```
#pragma omp parallel
{
	#pragma omp for schedule(dynamic, 1) nowait
	for (i = 0; i < tr_count; i++) {
		tot += tr_list[i];
	}
}

for (i = 0; i < DAYS; i++) { //日

	printf("** %d 日目(%d ops)¥n", i + 1, tot);

	or = 0.0;
	ts_t = 0.0;

	for (j = 0; j < tr_count; j++) { //単位時間

		//クレーン処理時間配列のメモリ確保
		cn_t = allocCraneTime(rackcost.cn);

		#pragma omp parallel
		{
			#pragma omp for schedule(dynamic, 1) nowait
			for (k = 0; k < rackcost.cn; k++) {
				cn_t[k] = 0.0;
			}
		}

		s_t = 0.0;

		//単位時間あたりの処理命令数の読み込み
		count = tr_list[j];

		ship_list2 = (ship_t*)calloc(count, sizeof(ship_t));
		arr_list2 = (arr_t*)calloc(count, sizeof(arr_t));

		#pragma omp parallel
		{
			#pragma omp for schedule(dynamic, 1) nowait
			for (k = 0; k < count; k++) {
				ship_list2[k] = ship_list[(k + pos) % ship_count];
				arr_list2[k] = arr_list[(k + pos) % arr_count];
```

```
                                        if ((k + pos) >= arr_count) {
                                                arr_list2[k].lot += (k + pos) / arr_count * arr_count;
                                        }
                                }
                        }

                        //入出荷処理
                        ArrShip(cn_t, rack, rackcost.rd, rackcost.rl, rackcost.cn, n, arr_list2, count,
ship_list2, count, g_count, &ops, &s_t);

                        //前の 1 単位時間で処理完了していない場合の処理時間分を追加
                        s_t += ts_t;

                        //1 単位時間で処理完了しなかった分を次の 1 単位時間で処理
                        if (s_t >  KTIME) {
                                ts_t = s_t - KTIME;
                                s_t = KTIME;
                        } else {
                                ts_t = 0;
                        }

                        //単位時間あたりの稼働率(%)
                        hor = s_t / KTIME * 100;

                        //1 日の稼働率(%)
                        or += hor / tr_count;

                        //出力
                        printf("%2d,%2d,%4d,%3d,%6.2lf%%, %6.2lf%%¥n", i + 1, j + 1, count, i *
tr_count + (j + 1), hor, ts_t / KTIME * 100);

                        pos += count;

                        //クレーン処理時間配列のメモリ解放
                        free(cn_t);

                        //性能確認用入出荷リストのメモリ解放
                        free(ship_list2);
                        free(arr_list2);

                }

                //1 日の非稼働率(%)
```

```
		uor = 100 - or;

		printf("稼働率, %5.2lf%%,  非稼働率, %5.2lf%%¥n", or, uor);

		//平均稼働率(%)
		or_ave += or / DAYS;

	}

	//平均非稼働率(%)
	uor_ave = 100 - or_ave;

	//機会損失=コスト×非稼働率(万円)
	ol = rackcost.cost * uor_ave / 100;

	printf("平均稼働率,  %5.2lf%%,  平均非稼働率,  %5.2lf%%,  機会損失,  %.1lf[万円]¥n",  or_ave,
uor_ave, ol);

}

void makeRackSet(int rd, rackcost_t *rackcost, group_t *g_table, int g_count, arr_t *arr_list, int arr_count,
ship_t *ship_list, int ship_count, arr_t *stock_list, int stock_count, int tr_max, int tr_count)
{
	//*********************************************************************//
	int max_cn; //クレーン最大台数
	int cn;   //設計時のラック連数, クレーン台数
	//*********************************************************************//

	//クレーン最大台数
	max_cn = (int)ceil((double)RS / (rd * 2)); //rl=1 のときが最大台数 小数点以下切り上げ

	//クレーン台数 1 台～max_cn 台
	#pragma omp parallel
	{
		#pragma omp for schedule(dynamic, 1) nowait
		for (cn = 1; cn <= max_cn; cn++) {
			funcSimulation(rd, cn, rackcost, g_table, g_count, arr_list, arr_count, ship_list,
ship_count, stock_list, stock_count, tr_max, tr_count);
		}
	}
}

//ラックの初期化
```

```
void InitRack(int rd, int rl, int cn, rack_t ****rack, int *n, group_t *g_table, int g_count, arr_t *stock_list, int stock_count)
{
	//*************************************************************************//
	int i, j, k, l;
	double atime; //ラックのアクセスタイム
	//*************************************************************************//

	//使用するコスト・速度テーブルの決定
	*n = condition(rd);

	//各ラックのアクセスタイムを計算
	for (i = 0; i < rd; i++) {
		for (j = 0; j < rl; j++) {
			atime = access_time(-1, j, 0, i, *n);
			for (k = 0; k < cn; k++) {
				for (l = 0; l < 2; l++) {
					rack[k][l][i][j].at = atime;
				}
			}
		}
	}

	//ラックの割り付け処理
	if(NOGRP == 0) {
		setGroupToRack(rack, cn, rd, rl, g_table, g_count);
	}

	//初期在庫の格納
	setStockToRack(rack, cn, rd, rl, stock_list, stock_count);

}

//入出荷処理
void ArrShip(double *cn_t, rack_t ****rack, int rd, int rl, int cn, int n, arr_t *arr_list, int arr_count, ship_t *ship_list, int ship_count, int g_count, int *ops, double *s_t)
{
	//*************************************************************************//
	int i;
	//*************************************************************************//

	//クレーン処理時間配列の初期化
	for (i = 0; i < cn; i++) {
```

```
			cn_t[i] = 0.0;
		}

		//入出荷処理
		if (OOOAR) {
			//アウトオブオーダー入出荷処理
			*ops = funcFetchArr(rack, cn, rd, rl, arr_list, arr_count, ship_list, ship_count, n, cn_t,
g_count);

			//クレーン処理時間の算出
			*s_t = funcProcessingTime(cn, cn_t, CTM);

		} else {
			//インオーダー入出荷処理
			*ops = funcArrShip(rack, cn, rd, rl, arr_list, arr_count, ship_list, ship_count, n, cn_t);

			//クレーン処理時間の算出
			*s_t = funcProcessingTime(cn, cn_t, CTM);

		}
}

//入出荷シミュレーション
void funcSimulation(int rd, int cn, rackcost_t *rackcost, group_t *g_table, int g_count, arr_t *arr_list, int
arr_count, ship_t *ship_list, int ship_count, arr_t *stock_list, int stock_count, int tr_max, int tr_count)
{
		//*************************************************************************//
		double da, cost, s_t, s_tmax = 0; //設計時の使用面積, コスト, 処理時間
		int rl;  //設計時のラック連数, クレーン台数
		rack_t ****rack; //ラック配列
		int n; //コスト・速度テーブル
		double *cn_t; //クレーン処理時間配列
		int ops; //クレーンの処理回数
		int *ip;
		double *dp;
		int pos = 0, i, j;
		arr_t *arr_list2; //入荷リスト
		ship_t *ship_list2; //出荷リスト
		//*************************************************************************//

		//ラック連数
		rl = (int)ceil((double)RS / (rd * cn * 2)); //小数点以下切り上げ
```

```
		//使用面積
		da = (3 * RW * cn) * (RL * (rl + 1)); //(rl + 1)は入出荷口分

		//倉庫面積が与件の使用可能面積以下の場合のみ以後の計算を実施
		if (da <= AREA) {

			//動的配列の初期化
			rack = allocRackSet(rd, cn, rl); //rack[クレーン番号][列][段][連]

			//ラックの初期化
			InitRack(rd, rl, cn, rack, &n, g_table, g_count, stock_list, stock_count);

			//コスト計算
			cost = funcCost(cn, rd, rl, da, n);

			//コストの表示
			//printf("Cost: %.1lf[万円] (Rack[%d][%d][%d], cn=%d[台], area=%.2lf[m^2]) [%d]¥n",
cost, rd, 2 * cn, rl, cn, da, ++count2);
			//++count2;

			if (cost <= rackcost->cost) {

				for (i = 0; i < (DAYS * tr_count); i++) {
					//クレーン処理時間配列のメモリ確保
					cn_t = allocCraneTime(cn);

					//性能確認用入出荷リスト作成
					ship_list2 = (ship_t*)calloc(tr_max, sizeof(ship_t));
					arr_list2 = (arr_t*)calloc(tr_max, sizeof(arr_t));

					for (j = 0; j < tr_max; j++) {
						ship_list2[j] = ship_list[(j + pos) % ship_count];
						arr_list2[j] = arr_list[(j + pos) % arr_count];

						if ((j + pos) >= arr_count) {
							arr_list2[j].lot += (j + pos) / arr_count * arr_count;
						}
					}

					//入出荷処理
					ArrShip(cn_t, rack, rd, rl, cn, n, arr_list2, tr_max, ship_list2, tr_max,
g_count, &ops, &s_t);
```

```
                        //printf("i=%d, ops=%d, lot=%d¥n", i, ops, arr_list2[0].lot);

                        if (s_tmax < s_t) {
                                s_tmax = s_t;

                                if (s_tmax > KTIME){
                                        break;
                                }
                        }

                        pos += tr_max;

                        //クレーン処理時間配列のメモリ解放
                        free(cn_t);

                        //性能確認用入出荷リストのメモリ解放
                        free(ship_list2);
                        free(arr_list2);

                }

                if (s_tmax <= KTIME) {

                        //クレーン処理時間(合計)の表示 (デバッグ用)
                        //printf("s_t(Rack[%d][%d][%d]) = %.2lf[min.] (%d ops) [%d]¥n", rd, 2
* cn, rl, s_t / 60, ops, ++count1);
                        //++count1;

                        #pragma omp critical
                        {
                                dp = &rackcost->cost;
                                *dp = cost;
                                ip = &rackcost->rd;
                                *ip = rd;
                                ip = &rackcost->rr;
                                *ip = 2 * cn;
                                ip = &rackcost->rl;
                                *ip = rl;
                                ip = &rackcost->cn;
                                *ip = cn;
                                dp = &rackcost->area;
                                *dp = da;
                                dp = &rackcost->time;
```

```
                                        *dp = s_tmax;
                                        ip = &rackcost->ops;
                                        *ip = ops;
                                }
                        }

                }

                //動的配列のメモリ解放
                freeRackSet(rack, cn, rd);
        }
}

//クレーン処理時間の計算
double funcProcessingTime(int cn, double *cn_t, int ctm)
{
        //*************************************************************************//
        int i;
        double s_t = 0.0; //クレーンの処理時間
        //*************************************************************************//

        if (ctm == 1) {
                //各クレーンの処理時間のうち最長のものを処理時間とする(CTM=1)
                for (i = 0; i < cn; i++) {
                        if (cn_t[i] > s_t) {
                                s_t = cn_t[i];
                        }
                }
        } else {
                //各クレーンの処理時間の総和
                for (i = 0; i < cn; i++) {
                        s_t += cn_t[i];
                }

                //各クレーンの処理時間の平均を処理時間とする(CTM=0)
                s_t /= cn;
        }

        return s_t;
}

//コスト計算
double funcCost(int cn, int rd, int rl, double area, int n)
```

```
{
	//*********************************************************************//
	double cost = 0.0;
	//*********************************************************************//

	//ラックコスト
	cost += rd * 2 * cn * rl * CostTable[n].rackc;

	//レールコスト (長さは1本ずつ小数点以下切り上げ)
	cost += ceil(rl * RL) * cn * CostTable[n].railc;

	//クレーンコスト
	cost += cn * CostTable[n].kc;

	//スプリンクラーコスト
	if ((area >= SPNA) && (rd * RH >= SPNH)) {
		cost += ceil((double)rd / floor((double)SPH / RH)) * (cn + 1) * rl * SPC;
	}

	//耐震床コスト (床面積は小数点以下切り上げ)
	if (area >= 1000) {
		cost += ceil(area) * FC1;
	} else {
		cost += ceil(area) * FC2;
	}

	return cost;
}

//入荷処理のフェッチ操作
int funcFetchArr(rack_t ****rack, int cn, int rd, int rl, arr_t *arr_list, int arr_count, ship_t *ship_list, int ship_count, int n, double *cn_t, int g_count)
{
	//*********************************************************************//
	int i, j, k, l, m;
	int ooomax, ooomin;
	int ops;
	rack_t ****tmp_rack;
	double *tcn_t;
	//*********************************************************************//

	tcn_t = (double*)calloc(cn, sizeof(double));
	tmp_rack = allocRackSet(rd, cn, rl);
```

```
		if (MAXOOOAR == 0) {
			ooomin = cn;
			ooomax = cn;
		} else if (MAXOOOAR > cn) {
			ooomin = 1;
			ooomax = cn;
		} else {
			ooomin = 1;
			ooomax = MAXOOOAR;
		}

		//アウトオブオーダー処理命令数の決定
		for (m = ooomin; m <= ooomax; m++) {
			for (i = 0; i < cn; i++) {
				for (j = 0; j < 2; j++) {
					for (k = 0; k < rd; k++) {
						for (l = 0; l < rl; l++) {
							tmp_rack[i][j][k][l] = rack[i][j][k][l];
						}
					}
				}
				tcn_t[i] = cn_t[i];
			}

			if (m == 1) {
				ops = funcArrShip(tmp_rack, cn, rd, rl, arr_list, arr_count, ship_list, ship_count, n,
tcn_t);
			} else {
				ops = funcFetchArrOrder(tmp_rack, cn, rd, rl, arr_list, arr_count, ship_list,
ship_count, n, tcn_t, m, g_count);
			}

			if (KTIME >= funcProcessingTime(cn, tcn_t, CTM)) break;

		}

		for (i = 0; i < cn; i++) {
			for (j = 0; j < 2; j++) {
				for (k = 0; k < rd; k++) {
					for (l = 0; l < rl; l++) {
						rack[i][j][k][l] = tmp_rack[i][j][k][l];
					}
```

```
                        }
                }

                cn_t[i] = tcn_t[i];

        }

        free(tcn_t);
        freeRackSet(tmp_rack, cn, rd);

        return ops;
}

//入荷処理のフェッチ処理実行(maxoooar 分のトランザクションをフェッチ、ただしクレーン本数分を上限とする)
int funcFetchArrOrder(rack_t ****rack, int cn, int rd, int rl, arr_t *arr_list, int arr_count, ship_t *ship_list, int ship_count, int n, double *cn_t, int maxoooar, int g_count)
{
        //*************************************************************************//
        int i, j, k, l;
        int r = 0, rmax, r_arr, r_ship, ops = 0;
        int max_n;
        double s_t = 0.0, min_st;
        double *fcn_t;
        arr_t *perm, *fetched_arr;
        ship_t *tmp_ship;
        int min_ops, tmp_ops = 0;
        rack_t ****fetched_rack;
        arrlist_t tmp_arr;
        //*************************************************************************//

        fcn_t = (double*)calloc(cn, sizeof(double));
        fetched_rack = allocRackSet(rd, cn, rl);

        if (maxoooar == 0 || maxoooar > cn) {
                max_n = cn;
        } else {
                max_n = maxoooar;
        }

        if (arr_count > ship_count) {
                rmax = arr_count;
        } else {
```

```
            rmax = ship_count;
        }

        tmp_arr.n = g_count;

        do {
            fetched_arr = (arr_t*)calloc(max_n - tmp_ops, sizeof(arr_t));
            perm = (arr_t*)calloc(max_n - tmp_ops, sizeof(arr_t));
            tmp_arr.g_arr = (arr_t***)calloc(g_count, sizeof(arr_t**));
            tmp_arr.count = (int*)calloc(g_count, sizeof(int));
            for (i = 0; i < tmp_arr.n; i++) {
                tmp_arr.g_arr[i] = 0;
            }
            tmp_ship = (ship_t*)calloc(max_n - tmp_ops, sizeof(ship_t));

            min_st = 1e100;
            min_ops = 2147483647;
            r_arr = 0;
            r_ship = 0;

            for (i = 0; i < max_n - tmp_ops; i++) {

                if (r < arr_count) {
                    tmp_arr.g_arr[arr_list[r].g_code] =
(arr_t**)realloc(tmp_arr.g_arr[arr_list[r].g_code], (tmp_arr.count[arr_list[r].g_code] + 1) * sizeof(arr_t*));
                    tmp_arr.g_arr[arr_list[r].g_code][tmp_arr.count[arr_list[r].g_code]] =
&arr_list[r];
                    tmp_arr.count[arr_list[r].g_code]++;
                    r_arr++;
                }

                if (r < ship_count) {
                    tmp_ship[i] = ship_list[r];
                    r_ship++;
                }

                r++;

            }

            PermutationFetchArr(rack, fetched_rack, cn, rd, rl, tmp_arr, fetched_arr, r_arr, tmp_ship,
r_ship, n, cn_t, fcn_t, &min_st, &min_ops, perm, 0);
```

```
                for (i = 0; i < cn; i++) {
                        for (j = 0; j < 2; j++) {
                                for (k = 0; k < rd; k++) {
                                        for (l = 0; l < rl; l++) {
                                                rack[i][j][k][l] = fetched_rack[i][j][k][l];
                                        }
                                }
                        }

                        cn_t[i] = fcn_t[i];

                }

                ops += min_ops;

                //1 回のクレーン並列入出荷処理を最適化の上限とする
                if ((cn - ops % cn) < max_n) {
                        tmp_ops = (ops % cn) % max_n;
                } else {
                        tmp_ops = 0;
                }

                free(fetched_arr);
                for (i = 0; i < g_count; i++) {
                        free(tmp_arr.g_arr[i]);
                }
                free(tmp_arr.g_arr);
                free(tmp_arr.count);
                free(tmp_ship);
                free(perm);

        } while (r < rmax);

        freeRackSet(fetched_rack, cn, rd);
        free(fcn_t);

        return ops;

}

//フェッチ処理用順列組み合わせ作成(再帰呼び出し)
void PermutationFetchArr(rack_t ****rack, rack_t ****fetched_rack, int cn, int rd, int rl, arrlist_t arr_list, arr_t
*fetched_arr, int arr_count, ship_t *ship_list, int ship_count, int n, double *cn_t, double *fcn_t, double *min_st,
```

```
int *min_ops, arr_t *tperm, int count)
{
	//*************************************************************************//
	int i, j, k;
	arr_t *perm;
	arrlist_t tmp_arr;
	//*************************************************************************//

	if (count == arr_count) {
		funcFetchArrShip(rack, fetched_rack, cn, rd, rl, tperm, fetched_arr, arr_count, ship_list,
ship_count, n, cn_t, fcn_t, min_st, min_ops);
	} else {
		for (i = 0; i < arr_list.n; i++) {
			if (arr_list.count[i] > 0) {
				perm = (arr_t*)calloc(count + 1, sizeof(arr_t));
				for (j = 0; j < count; j++) {
					perm[j] = tperm[j];
				}
				perm[count] = *arr_list.g_arr[i][0];

				tmp_arr.n = arr_list.n;
				tmp_arr.count = (int*)calloc(arr_list.n, sizeof(int));
				tmp_arr.g_arr = (arr_t***)calloc(arr_list.n, sizeof(arr_t**));
				for (j = 0; j < arr_list.n; j++) {
					if (i == j) {
						tmp_arr.count[j] = arr_list.count[j] - 1;
						tmp_arr.g_arr[j] = (arr_t**)calloc(tmp_arr.count[j],
sizeof(arr_t*));
						for (k = 0; k < tmp_arr.count[j]; k++) {
							tmp_arr.g_arr[j][k] = arr_list.g_arr[j][k +
1];
						}
					} else {
						tmp_arr.count[j] = arr_list.count[j];
						tmp_arr.g_arr[j] = (arr_t**)calloc(tmp_arr.count[j],
sizeof(arr_t*));
						for (k = 0; k < tmp_arr.count[j]; k++) {
							tmp_arr.g_arr[j][k] = arr_list.g_arr[j][k];
						}
					}
				}

				PermutationFetchArr(rack, fetched_rack, cn, rd, rl, tmp_arr, fetched_arr,
```

```
arr_count, ship_list, ship_count, n, cn_t, fcn_t, min_st, min_ops, perm, count + 1);

                                        for (j = 0; j < tmp_arr.n; j++) {
                                                free(tmp_arr.g_arr[j]);
                                        }
                                        free(tmp_arr.g_arr);
                                        free(tmp_arr.count);
                                        free(perm);
                                }
                        }
                }

}

//フェッチ用仮入出荷処理
void funcFetchArrShip(rack_t ****rack, rack_t ****fetched_rack, int cn, int rd, int rl, arr_t *arr_list, arr_t *fetched_arr, int arr_count, ship_t *ship_list, int ship_count, int n, double *cn_t, double *fcn_t, double *min_st, int *min_ops)
{
        //*************************************************************************//
        int i, j, k, l;
        int ops; //クレーンの処理回数
        double s_t; //クレーン処理時間
        rack_t ****tmp_rack, *rp;
        arr_t *ap;
        int *ip;
        double *dp;
        double *tcn_t;
        //*************************************************************************//

        tcn_t = allocCraneTime(cn);
        tmp_rack = allocRackSet(rd, cn, rl);

        //処理前の各クレーン処理時間を反映
        for (i = 0; i < cn; i++) {
                tcn_t[i] = cn_t[i];
        }

        //処理前のラック状態の反映
        for (i = 0; i < cn; i++) {
                for (j = 0; j < 2; j++) {
                        for (k = 0; k < rd; k++) {
                                for (l = 0; l < rl; l++) {
```

```
                                        tmp_rack[i][j][k][l] = rack[i][j][k][l];
                                }
                        }
                }
        }

//仮入出荷処理の実行
ops = funcArrShip(tmp_rack, cn, rd, rl, arr_list, arr_count, ship_list, ship_count, n, tcn_t);

//アクセスタイムの計算
s_t = funcProcessingTime(cn, tcn_t, 0);

//命令数最小かつアクセスタイム最小のとき、入出荷処理結果を戻す
if ((ops < *min_ops) || (ops <= *min_ops && s_t < *min_st)) {
        dp = min_st;
        *dp = s_t;

        for (i = 0; i < cn; i++) {
                fcn_t[i] = tcn_t[i];
        }

        for (i = 0; i < arr_count; i++) {
                ap = &fetched_arr[i];
                *ap = arr_list[i];
        }

        for (i = 0; i < cn; i++) {
                for (j = 0; j < 2; j++) {
                        for (k = 0; k < rd; k++) {
                                for (l = 0; l < rl; l++) {
                                        rp = &fetched_rack[i][j][k][l];
                                        *rp = tmp_rack[i][j][k][l];
                                }
                        }
                }
        }

        ip = min_ops;
        *ip = ops;
}

freeRackSet(tmp_rack, cn, rd);
free(tcn_t);
```

```
}


//入出荷処理
int funcArrShip(rack_t ****rack, int cn, int rd, int rl, arr_t *arr_list, int arr_count, ship_t *ship_list, int ship_count, int n, double *cn_t)
{
	//*************************************************************************//
	int r, i, j, k, l, rmax;
	double min_ct, min_at, tmp_ct; //サイクルタイム
	int min_lot; //最小ロット番号
	int op_s_rl, op_s_rd, op_s_rr, op_s_rc; //出荷ラック，クレーン番号
	int op_a_rl, op_a_rd, op_a_rr, op_a_rc; //入荷ラック，クレーン番号
	bool flag_cn; //クレーンの担当ラックに空きがあるかの判定
	int ops = 0; //クレーンの処理回数
	bool *bp;
	int *ip;
	double *dp;
	char *cp;
	//*************************************************************************//

	//入出荷トランザクション数が異なる場合用
	if (ship_count > arr_count) {
		rmax = ship_count;
	} else {
		rmax = arr_count;
	}

	for (r = 0; r < rmax; r++) {
		min_lot = 2147483647;
		min_at = 1e100;

		//出荷ラックの決定
		if (r < ship_count) {
			for (i = 0; i < rd; i++) {
				for (j = 0; j < cn; j++) {
					for (l = 0; l < 2; l++) {
						for (k = 0; k < rl; k++) {
							if (rack[j][l][i][k].flag && strcmp(ship_list[r].p_code, rack[j][l][i][k].p_code) == 0 && rack[j][l][i][k].lot < min_lot) {
								min_lot = rack[j][l][i][k].lot;
								op_s_rc = j;
```

```
                                                op_s_rr = l;
                                                op_s_rd = i;
                                                op_s_rl = k;
                                        }
                                }
                        }
                }
        }

        ops++;
        min_ct = 2 * rack[op_s_rc][op_s_rr][op_s_rd][op_s_rl].at + RW / CostTable[n].ss * 2;

        //入荷ラックの決定
        flag_cn = false;
        if (r < arr_count) {
                for (i = 0; i < rd; i++) {
                        for (k = 0; k < rl; k++) {
                                for (j = 0; j < 2; j++) {
                                        if (!(rack[op_s_rc][j][i][k].flag) && (NOGRP != 0 ||
arr_list[r].g_code == rack[op_s_rc][j][i][k].g_code)) {
                                                tmp_ct = cycle_time(rack, op_s_rc, j, i, k,
op_s_rc, op_s_rr, op_s_rd, op_s_rl, n);
                                                if (tmp_ct < min_at) {
                                                        min_at = tmp_ct;
                                                        op_a_rc = op_s_rc;
                                                        op_a_rr = j;
                                                        op_a_rd = i;
                                                        op_a_rl = k;
                                                        flag_cn = true;
                                                }
                                        }
                                }
                        }
                }

                //出荷対象ラックと同じクレーンの担当するラックに分類コードの一致
する空きラックがなく、他の分類コードのラックに入れる処理
                if (!flag_cn) {
                        for (i = 0; i < rd; i++) {
                                for (k = 0; k < rl; k++) {
                                        for (j = 0; j < 2; j++) {
                                                if (!rack[op_s_rc][j][i][k].flag) {
```

```
                                                tmp_ct = cycle_time(rack, op_s_rc, j, i, k, op_s_rc, op_s_rr, op_s_rd, op_s_rl, n);
                                                if (tmp_ct < min_at) {
                                                    min_at = tmp_ct;
                                                    op_a_rc = op_s_rc;
                                                    op_a_rr = j;
                                                    op_a_rd = i;
                                                    op_a_rl = k;
                                                    flag_cn = true;
                                                }
                                            }
                                        }
                                    }
                                }
                            }

                            //出荷対象ラックと同じクレーンの担当するラックに分類コードの一致する空きラックがなく、他の分類コードのラックにも空きがない場合(デュアルサイクル処理をしない)
                            if (!flag_cn) {
                                if (ARO == 1) {
                                    //クレーンの並列動作性を考慮した入荷ラックの決定順序
                                    for (i = 0; i < 2; i++) {
                                        for (j = 0; j < rd; j++) {
                                            for (k = 0; k < rl; k++) {
                                                for (l = 0; l < cn; l++) {
                                                    if (!(rack[l][i][j][k].flag) && (NOGRP != 0 || arr_list[r].g_code == rack[l][i][j][k].g_code) && (rack[l][i][j][k].at < min_at)) {
                                                        min_at = rack[l][i][j][k].at;
                                                        op_a_rc = l;
                                                        op_a_rr = i;
                                                        op_a_rd = j;
                                                        op_a_rl = k;
                                                    }
                                                }
                                            }
                                        }
                                    }
                                } else {
                                    //クレーンの並列動作性を考慮しない入荷ラックの決定
```

順序(端からのラスタスキャン)

```
                                        for (j = 0; j < rd; j++) {
                                            for (l = 0; l < cn; l++) {
                                                for (i = 0; i < 2; i++) {
                                                    for (k = 0; k < rl; k++) {
                                                        if
(!(rack[l][i][j][k].flag) && (NOGRP != 0 || arr_list[r].g_code == rack[l][i][j][k].g_code) && (rack[l][i][j][k].at <
min_at)) {
                                                            min_at      =
rack[l][i][j][k].at;
                                                            op_a_rc = l;
                                                            op_a_rr = i;
                                                            op_a_rd = j;
                                                            op_a_rl = k;
                                                        }
                                                    }
                                                }
                                            }
                                        }
                                    }
                                    ops++;
                                    min_at = 2 * min_at + RW / CostTable[n].ss * 2;
                                }
                            }

                            //入出荷実行
                            bp = &rack[op_s_rc][op_s_rr][op_s_rd][op_s_rl].flag;
                            *bp = false;
                            ip = &rack[op_s_rc][op_s_rr][op_s_rd][op_s_rl].lot;
                            *ip = 0;
                            cp = rack[op_s_rc][op_s_rr][op_s_rd][op_s_rl].p_code;
                            strcpy(cp, "");
                            bp = &rack[op_a_rc][op_a_rr][op_a_rd][op_a_rl].flag;
                            *bp = true;
                            ip = &rack[op_a_rc][op_a_rr][op_a_rd][op_a_rl].lot;
                            *ip = arr_list[r].lot;
                            cp = rack[op_a_rc][op_a_rr][op_a_rd][op_a_rl].p_code;
                            strcpy(cp, arr_list[r].p_code);

                            //既に処理が実行されているクレーンと並列処理可能かの確認
                            if (CTM == 1) {
                                tmp_ct = 0.0;
                                for (i = 0; i < cn; i++) {
```

```
                        if (cn_t[i] > tmp_ct) {
                            tmp_ct = cn_t[i];
                        }
                    }
                    if (cn_t[op_s_rc] == tmp_ct || cn_t[op_a_rc] == tmp_ct) {
                        for (i = 0; i < cn; i++) {
                            cn_t[i] = tmp_ct;
                        }
                    }
                }

                if (flag_cn) {
                    dp = &cn_t[op_s_rc];
                    *dp += min_at;
                } else {
                    dp = &cn_t[op_s_rc];
                    *dp += min_ct;
                    dp = &cn_t[op_a_rc];
                    *dp += min_at;
                }
            }

            return ops;

}

//初期在庫の格納
void setStockToRack(rack_t ****rack, int cn, int rd, int rl, arr_t *stock_list, int stock_count)
{
            //*************************************************************************//
            int r, i, j, k, l;
            double min_at; //最短移動時間
            int op_rl, op_rd, op_rr, op_rc; //最小コスト時のラック連数, 段数, 列数, クレーン番号
            int *ip;
            bool *bp;
            char *cp;
            //*************************************************************************//

            for (r = 0; r < stock_count; r++) {
                min_at = 1e100;

                if (ARO == 1) {
```

```
                    //クレーンの並列動作性を考慮した入荷ラックの決定順序
                    for (i = 0; i < 2; i++) {
                        for (j = 0; j < rd; j++) {
                            for (k = 0; k < rl; k++) {
                                for (l = 0; l < cn; l++) {
                                    if ((NOGRP != 0 || stock_list[r].g_code ==
rack[l][i][j][k].g_code) && !rack[l][i][j][k].flag && rack[l][i][j][k].at < min_at) {
                                        min_at = rack[l][i][j][k].at;
                                        op_rc = l;
                                        op_rd = j;
                                        op_rr = i;
                                        op_rl = k;
                                    }
                                }
                            }
                        }
                    }

                } else {

                    //クレーンの並列動作性を考慮しない入荷ラックの決定順序
                    for (j = 0; j < rd; j++) {
                        for (l = 0; l < cn; l++) {
                            for (i = 0; i < 2; i++) {
                                for (k = 0; k < rl; k++) {
                                    if ((NOGRP != 0 || stock_list[r].g_code ==
rack[l][i][j][k].g_code) && !rack[l][i][j][k].flag && rack[l][i][j][k].at < min_at) {
                                        min_at = rack[l][i][j][k].at;
                                        op_rc = l;
                                        op_rd = j;
                                        op_rr = i;
                                        op_rl = k;
                                    }
                                }
                            }
                        }
                    }

                }

                ip = &rack[op_rc][op_rr][op_rd][op_rl].lot;
                *ip = stock_list[r].lot;
                cp = rack[op_rc][op_rr][op_rd][op_rl].p_code;
```

```
                strcpy(cp, stock_list[r].p_code);
                bp = &rack[op_rc][op_rr][op_rd][op_rl].flag;
                *bp = true;
        }
}

//サイクルタイムの計算
double cycle_time(rack_t ****rack, int arr_c, int arr_r, int arr_h, int arr_l, int ship_c, int ship_r, int ship_h, int ship_l, int n)
{
        //*************************************************************************//
        double ct;
        //*************************************************************************//

        ct = rack[arr_c][arr_r][arr_h][arr_l].at + rack[ship_c][ship_r][ship_h][ship_l].at + access_time(arr_l, ship_l, arr_h, ship_h, n) + RW / CostTable[n].ss * 4;
        return ct;
}

//割り付け処理
void setGroupToRack(rack_t ****rack, int cn, int rd, int rl, group_t *g_table, int g_count)
{
        //*************************************************************************//
        int q, tq = 0; //分類毎に割り付けるラック数
        int t, g, i, j, k, l;
        double min_at; //最短移動時間
        int op_rl, op_rd, op_rr, op_rc; //最小コスト時のラック連数, 段数, 列数, クレーン番号
        int *ip;
        bool ****RackFlag;
        //*************************************************************************//

        RackFlag = allocRackFlag(cn, rd, rl); //割付済みフラグ

        for (g = 0; g < g_count; g++) {
                if (g < g_count - 1) {
                        q = (int)ceil(rd * 2 * cn * rl * g_table[g].ratio);
                        tq += q;
                } else {
                        q = rd * 2 * cn * rl - tq;
                }

                //分類毎のラック数の表示 (デバッグ用)
                //printf("%s: %lf -> %d¥n", g_table[g].g_code, g_table[g].ratio, q);
```

```
                for (t = 0; t < q; t++) {
                        min_at = 1e100;
                        for (i = 0; i < 2; i++) {
                                for (j = 0; j < rd; j++) {
                                        for (k = 0; k < rl; k++) {
                                                for (l = 0; l < cn; l++) {
                                                        if (!(RackFlag[l][i][j][k]) &&
(rack[l][i][j][k].at < min_at)) {
                                                                min_at = rack[l][i][j][k].at;
                                                                op_rc = l;
                                                                op_rr = i;
                                                                op_rd = j;
                                                                op_rl = k;
                                                        }
                                                }
                                        }
                                }
                        }
                        ip = &rack[op_rc][op_rr][op_rd][op_rl].g_code;
                        *ip = g;
                        RackFlag[op_rc][op_rr][op_rd][op_rl] = true;
                }
        }

        freeRackFlag(RackFlag, cn, rd);

}

//アクセスタイムの計算
double access_time(int sl, int el, int sh, int eh, int n)
{
        //*************************************************************************//
        double lt, ht; //走行時間, 昇降時間
        //*************************************************************************//

        lt = v_time(sl, el, RL, CostTable[n].ls, CostTable[n].al);
        ht = v_time(sh, eh, RH, CostTable[n].hs, CostTable[n].ah);

        if (lt > ht) {
                return lt;
        } else {
                return ht;
```

```
	}

}

//走行・昇降時間
double v_time(int sp, int ep, double rl, double v, double a) //初期位置, 到達位置, パレットの長さ or 高さ, 速度[m/min], 加速度[m/s^2]
{
	//*************************************************************************//
	double t, l; //走行時間合計[s], 走行距離[m]
	double v_max; //最高走行速度[m/s]
	double t1, x1;
	//*************************************************************************//

	v_max = v / 60.0; //走行最高速度[m/s]
	l = abs(ep - sp) * rl; //走行距離[m]

	if (l == 0.0) { //移動しないとき
		t = 0.0;
	} else {
		t1 = v_max / a; //加速・減速にそれぞれ必要な時間[s]
		x1 = 0.5 * a * t1 * t1 * 2; //加速・減速中にそれぞれ走行する距離の合計[m]
		if (l > x1) { //最高速度に到達するとき
			t = t1 * 2 + (l - x1) / v_max;
		} else { //最高速度に到達しないとき
			t = 2 * sqrt(l / a);
		}
	}

	return t; //[sec.]
}

//初期設定ファイルの読み込み
void readInitialFile(char *filename)
{
	//*************************************************************************//
	char buf[256], *token;
	int col;
	unsigned int flag = 0x0;
	unsigned int *flag2;
	unsigned int flag3 = 0xffff, flag4 = 0x1ff;
	char parameter[20];
	int n;
```

```
bool error = false;
FILE *fp;
//*****************************************************************************//

fp = freadopen(filename);

rewind(fp);
while (fgets(buf, 256, fp)) {

        chop_crlf(buf);
        if (buf[0] == '#' || strlen(buf) == 0) continue;
        col = 0;
        token = (char*)strtok(buf, seps);

        while (token != NULL) {
                if (col == 0) {
                        sscanf(token, "%s", &parameter);
                } else if (col == 1) {
                        if (strcmp(parameter, "RS") == 0) { //ラック個数[個]
                                sscanf(token, "%d", &RS);
                                flag = flag | 0x00000001;
                        } else if (strcmp(parameter, "AREA") == 0) { //与件の使用面積[m^2]
                                sscanf(token, "%lf", &AREA);
                                flag = flag | 0x00000002;
                        } else if (strcmp(parameter, "H") == 0) { //倉庫高さ制約[m]
                                sscanf(token, "%lf", &H);
                                flag = flag | 0x00000004;
                        } else if (strcmp(parameter, "RH") == 0) { //ラック高さ[m]
                                sscanf(token, "%lf", &RH);
                                flag = flag | 0x00000008;
                        } else if (strcmp(parameter, "RL") == 0) { //ラック長さ[m]
                                sscanf(token, "%lf", &RL);
                                flag = flag | 0x00000010;
                        } else if (strcmp(parameter, "RW") == 0) { //ラック幅[m]
                                sscanf(token, "%lf", &RW);
                                flag = flag | 0x00000020;
                        } else if (strcmp(parameter, "SPC") == 0) { //スプリンクラーコスト
[万円/個]
                                sscanf(token, "%lf", &SPC);
                                flag = flag | 0x00000040;
                        } else if (strcmp(parameter, "FC1") == 0) { //耐震床コスト(1000m^2
以上)[万円/m^2]
                                sscanf(token, "%lf", &FC1);
```

```
                flag = flag | 0x00000080;
            } else if (strcmp(parameter, "FC2") == 0) { //耐震床コスト(1000m^2
未満)[万円/m^2]
                sscanf(token, "%lf", &FC2);
                flag = flag | 0x00000100;
            } else if (strcmp(parameter, "KTIME") == 0) { //クレーン処理時間の
上限[sec.]
                sscanf(token, "%lf", &KTIME);
                flag = flag | 0x00000200;
            } else if (strcmp(parameter, "N") == 0) { //クレーン・ラック・レール
条件数
                sscanf(token, "%d", &N);
                flag = flag | 0x00000400;
                CostTable = (cost_t*)calloc(N, sizeof(cost_t));
                flag2 = (unsigned int*)calloc(N, sizeof(unsigned int));
                for (n = 0; n < N; n++) {
                        flag2[n] = 0x0;
                }
            } else if (strcmp(parameter, "SPH") == 0) { //スプリンクラー設置高
さ間隔[m]
                sscanf(token, "%lf", &SPH);
                flag = flag | 0x00000800;
            } else if (strcmp(parameter, "SPNH") == 0) { //スプリンクラー設置高
さ間隔[m]
                sscanf(token, "%lf", &SPNH);
                flag = flag | 0x00001000;
            } else if (strcmp(parameter, "SPNA") == 0) { //スプリンクラー設置高
さ間隔[m]
                sscanf(token, "%lf", &SPNA);
                flag = flag | 0x00002000;
            } else if (strcmp(parameter, "ARO") == 0) { //入荷先ラック決定順序
モード
                sscanf(token, "%d", &ARO);
                flag = flag | 0x00004000;
            } else if (strcmp(parameter, "CTM") == 0) { //クレーン処理時間モー
ド
                sscanf(token, "%d", &CTM);
                flag = flag | 0x00008000;
            } else if (strcmp(parameter, "OOOAR") == 0) {//アウトオブオーダー
入荷モード
                sscanf(token, "%d", &OOOAR);
                flag = flag | 0x00010000;
            } else if (strcmp(parameter, "MAXOOOAR") == 0) {//アウトオブオ
```

```
ーダー入荷モード
                    sscanf(token, "%d", &MAXOOOAR);
                    flag = flag | 0x00020000;
                } else if (strcmp(parameter, "NOGRP") == 0) { //ラックのグループ割
付モード
                    sscanf(token, "%d", &NOGRP);
                    flag = flag | 0x00040000;
                } else if (strcmp(parameter, "DAYS") == 0) { //機会損失の推移を計
算する日数
                    sscanf(token, "%d", &DAYS);
                    flag = flag | 0x00080000;
                } else if (strncmp(parameter, "KH", 2) == 0) { //クレーン許容高さ範
囲[m]
                    n = atoi(&parameter[2]) - 1;
                    if (n < N) {
                        sscanf(token, "%lf", &CostTable[n].kh);
                        flag2[n]= flag2[n] | 0x00000001;
                    }
                } else if (strncmp(parameter, "KC", 2) == 0) { //クレーンコスト[万円]
                    n = atoi(&parameter[2]) - 1;
                    if (n < N) {
                        sscanf(token, "%lf", &CostTable[n].kc);
                        flag2[n]= flag2[n] | 0x00000002;
                    }
                } else if (strncmp(parameter, "LS", 2) == 0) { //クレーン走行速度
[m/min.]
                    n = atoi(&parameter[2]) - 1;
                    if (n < N) {
                        sscanf(token, "%lf", &CostTable[n].ls);
                        flag2[n]= flag2[n] | 0x00000004;
                    }
                } else if (strncmp(parameter, "HS", 2) == 0) { //クレーン昇降速度
[m/min.]
                    n = atoi(&parameter[2]) - 1;
                    if (n < N) {
                        sscanf(token, "%lf", &CostTable[n].hs);
                        flag2[n]= flag2[n] | 0x00000008;
                    }
                } else if (strncmp(parameter, "SS", 2) == 0) { //クレーンシャトル速度
[m/min.]
                    n = atoi(&parameter[2]) - 1;
                    if (n < N) {
                        sscanf(token, "%lf", &CostTable[n].ss);
```

```
                                        flag2[n]= flag2[n] | 0x00000010;
                                }
                        } else if (strncmp(parameter, "AL", 2) == 0) { //クレーン走行加速度
[m/sec^2]
                                n = atoi(&parameter[2]) - 1;
                                if (n < N) {
                                        sscanf(token, "%lf", &CostTable[n].al);
                                        flag2[n]= flag2[n] | 0x00000020;
                                }
                        } else if (strncmp(parameter, "AH", 2) == 0) { //クレーン昇降加速度
[m/sec^2]
                                n = atoi(&parameter[2]) - 1;
                                if (n < N) {
                                        sscanf(token, "%lf", &CostTable[n].ah);
                                        flag2[n]= flag2[n] | 0x00000040;
                                }
                        } else if (strncmp(parameter, "RKC", 3) == 0) { //ラックコスト[万円/
個]
                                n = atoi(&parameter[3]) - 1;
                                if (n < N) {
                                        sscanf(token, "%lf", &CostTable[n].rackc);
                                        flag2[n]= flag2[n] | 0x00000080;
                                }
                        } else if (strncmp(parameter, "RLC", 3) == 0) { //レールコスト[万円
/m]
                                n = atoi(&parameter[3]) - 1;
                                if (n < N) {
                                        sscanf(token, "%lf", &CostTable[n].railc);
                                        flag2[n]= flag2[n] | 0x00000100;
                                }
                        }

                }
                token = (char*)strtok(NULL, seps);
                col++;
        }
    }

    //初期設定ファイルのパラメータ不足(エラー)時に True を返す
    if ((flag & flag3) == flag3) {
        for (n = 0; n < N; n++) {
            if ((flag2[n] & flag4) != flag4) {
                error = true; //パラメータ不足
```

```
                                        break;
                                }
                        }
                } else {
                        error = true; //パラメータ不足
                }
                if (N > 0) {
                        free(flag2);
                }

                if (error) {
                        fprintf(stderr, "Parameter is NOT enough.");
                        exit(1);
                }

                fclose(fp);

}

//コマンドライン引数の確認
void readargc(int argc, char **argv)
{
        //*************************************************************************//
        int i, j, n;
        char *token;
        char parameter[20];
        //*************************************************************************//

        for (i = 0; i < argc; i++) {
                chop_crlf(argv[i]);
                j = 0;
                token = (char*)strtok(argv[i], "=");

                while (token != NULL) {
                        if (j == 0) {
                                sscanf(token, "%s", &parameter);
                        } else if (j == 1) {
                                if (strcmp(parameter, "RS") == 0) { //ラック個数[個]
                                        sscanf(token, "%d", &RS);
                                } else if (strcmp(parameter, "AREA") == 0) { //与件の使用面積[m^2]
                                        sscanf(token, "%lf", &AREA);
                                } else if (strcmp(parameter, "H") == 0) { //倉庫高さ制約[m]
                                        sscanf(token, "%lf", &H);
```

```
} else if (strcmp(parameter, "RH") == 0) { //ラック高さ[m]
        sscanf(token, "%lf", &RH);
} else if (strcmp(parameter, "RL") == 0) { //ラック長さ[m]
        sscanf(token, "%lf", &RL);
} else if (strcmp(parameter, "RW") == 0) { //ラック幅[m]
        sscanf(token, "%lf", &RW);
} else if (strcmp(parameter, "SPC") == 0) { //スプリンクラーコスト
[万円/個]
        sscanf(token, "%lf", &SPC);
} else if (strcmp(parameter, "FC1") == 0) { //耐震床コスト(1000m^2
以上)[万円/m^2]
        sscanf(token, "%lf", &FC1);
} else if (strcmp(parameter, "FC2") == 0) { //耐震床コスト(1000m^2
未満)[万円/m^2]
        sscanf(token, "%lf", &FC2);
} else if (strcmp(parameter, "KTIME") == 0) { //クレーン処理時間の
上限[sec.]
        sscanf(token, "%lf", &KTIME);
} else if (strcmp(parameter, "N") == 0) { //クレーン・ラック・レール
条件数
        sscanf(token, "%d", &N);
        CostTable = (cost_t*)realloc(&CostTable, N * sizeof(cost_t));
} else if (strcmp(parameter, "SPH") == 0) { //スプリンクラー設置高
さ間隔[m]
        sscanf(token, "%lf", &SPH);
} else if (strcmp(parameter, "SPNH") == 0) { //スプリンクラー設置高
さ間隔[m]
        sscanf(token, "%lf", &SPNH);
} else if (strcmp(parameter, "SPNA") == 0) { //スプリンクラー設置高
さ間隔[m]
        sscanf(token, "%lf", &SPNA);
} else if (strcmp(parameter, "ARO") == 0) { //入荷先ラック決定順序
モード
        sscanf(token, "%d", &ARO);
} else if (strcmp(parameter, "CTM") == 0) { //クレーン処理時間モー
ド
        sscanf(token, "%d", &CTM);
} else if (strcmp(parameter, "OOOAR") == 0) {//アウトオブオーダー
入荷モード
        sscanf(token, "%d", &OOOAR);
} else if (strcmp(parameter, "MAXOOOAR") == 0) {//アウトオブオ
ーダー入荷モード
        sscanf(token, "%d", &MAXOOOAR);
```

```
		} else if (strcmp(parameter, "NOGRP") == 0) { //ラックのグループ割付モード
			sscanf(token, "%d", &NOGRP);
		} else if (strncmp(parameter, "KH", 2) == 0) { //クレーン許容高さ範囲[m]
			n = atoi(&parameter[2]) - 1;
			if (n < N) {
				sscanf(token, "%lf", &CostTable[n].kh);
			}
		} else if (strncmp(parameter, "KC", 2) == 0) { //クレーンコスト[万円]
			n = atoi(&parameter[2]) - 1;
			if (n < N) {
				sscanf(token, "%lf", &CostTable[n].kc);
			}
		} else if (strncmp(parameter, "LS", 2) == 0) { //クレーン走行速度[m/min.]
			n = atoi(&parameter[2]) - 1;
			if (n < N) {
				sscanf(token, "%lf", &CostTable[n].ls);
			}
		} else if (strncmp(parameter, "HS", 2) == 0) { //クレーン昇降速度[m/min.]
			n = atoi(&parameter[2]) - 1;
			if (n < N) {
				sscanf(token, "%lf", &CostTable[n].hs);
			}
		} else if (strncmp(parameter, "SS", 2) == 0) { //クレーンシャトル速度[m/min.]
			n = atoi(&parameter[2]) - 1;
			if (n < N) {
				sscanf(token, "%lf", &CostTable[n].ss);
			}
		} else if (strncmp(parameter, "AL", 2) == 0) { //クレーン走行加速度[m/sec^2]
			n = atoi(&parameter[2]) - 1;
			if (n < N) {
				sscanf(token, "%lf", &CostTable[n].al);
			}
		} else if (strncmp(parameter, "AH", 2) == 0) { //クレーン昇降加速度[m/sec^2]
			n = atoi(&parameter[2]) - 1;
			if (n < N) {
				sscanf(token, "%lf", &CostTable[n].ah);
```

```
                                        }
                                } else if (strncmp(parameter, "RKC", 3) == 0) { //ラックコスト[万円/
個]
                                        n = atoi(&parameter[3]) - 1;
                                        if (n < N) {
                                                sscanf(token, "%lf", &CostTable[n].rackc);
                                        }
                                } else if (strncmp(parameter, "RLC", 3) == 0) { //レールコスト[万円
/m]
                                        n = atoi(&parameter[3]) - 1;
                                        if (n < N) {
                                                sscanf(token, "%lf", &CostTable[n].railc);
                                        }
                                } else if (strcmp(parameter, "DAYS") == 0) { //機会損失の推移を計
算する日数
                                        sscanf(token, "%d", &DAYS);
                                }

                        }
                        token = (char*)strtok(NULL, "=");
                        j++;
                }
        }

}

void printInitialValues()
{
        //*************************************************************************//
        int n;
        //*************************************************************************//

        //初期設定ファイルから読み込んだパラメータの表示(デバッグ用)
        printf("**Initial File: ¥n");
        printf("RS      = %d[個]¥n", RS);
        printf("AREA    = %lf[m^2]¥n", AREA);
        printf("H       = %lf[m]¥n", H);
        printf("RH      = %lf[m]¥n", RH);
        printf("RL      = %lf[m]¥n", RL);
        printf("RW      = %lf[m]¥n", RW);
        printf("KTIME = %lf[sec.]¥n", KTIME);
        printf("SPC     = %lf[万円/台]¥n", SPC);
        printf("SPH     = %lf[m]¥n", SPH);
```

```
		printf("SPNH    = %lf[m]¥n", SPNH);
		printf("SPNA    = %lf[m^2]¥n", SPNA);
		printf("FC1     = %lf[万円/m^2]¥n", FC1);
		printf("FC2     = %lf[万円/m^2]¥n", FC2);
		printf("N       = %d¥n", N);
		printf("ARO     = %d¥n", ARO);
		printf("CTM     = %d¥n", CTM);
		printf("OOOAR = %d¥n", OOOAR);
		printf("MAXOOOAR= %d¥n", MAXOOOAR);
		printf("NOGRP = %d¥n", NOGRP);
		printf("DAYS    = %d¥n", DAYS);
		for (n = 0; n < N; n++) {
			printf("*KH%d    = %lf[m 以下]¥n", n + 1, CostTable[n].kh);
			printf(" KC%d    = %lf[万円/台]¥n", n + 1, CostTable[n].kc);
			printf(" LS%d    = %lf[m/min.]¥n", n + 1, CostTable[n].ls);
			printf(" HS%d    = %lf[m/min.]¥n", n + 1, CostTable[n].hs);
			printf(" SS%d    = %lf[m/min.]¥n", n + 1, CostTable[n].ss);
			printf(" AL%d    = %lf[m/sec.^2]¥n", n + 1, CostTable[n].al);
			printf(" AH%d    = %lf[m/sec.^2]¥n", n + 1, CostTable[n].ah);
			printf(" RKC%d = %lf[万円/台]¥n", n + 1, CostTable[n].rackc);
			printf(" RLC%d = %lf[万円/m]¥n", n + 1, CostTable[n].railc);
		}
}

//入出庫頻度(命令数)分布表の読み込み
int* readTransactions(int* tr_count, char* filename)
{
		//*************************************************************************//
		char buf[256], *token;
		int row = 0;
		FILE *fp;
		//*************************************************************************//

		fp = freadopen(filename);

		int* tr_list = (int*)calloc(readFileRow(fp), sizeof(int));

		rewind(fp);

		while (fgets(buf, 256, fp)) {

			chop_crlf(buf);
			if (buf[0] == '#' || strlen(buf) == 0) continue;
```

```
		token = (char*)strtok(buf, seps);

		while (token != NULL) {
			sscanf(token, "%d", &tr_list[row]);
			token = (char*)strtok(NULL, seps);
		}

		row++;
	}

	*tr_count = row;

	fclose(fp);

	return tr_list;

}

//入荷リスト読み込み
arr_t* readArrList(group_t *g_table, int g_count, int *arr_count, char *filename)
{
	//*********************************************************************//
	char buf[256], *token;
	int i, col, row = 0;
	char g_code[20];
	FILE *fp;
	//*********************************************************************//

	fp = freadopen(filename);

	arr_t* arr_list = (arr_t*)calloc(readFileRow(fp), sizeof(arr_t));

	rewind(fp);

	while (fgets(buf, 256, fp)) {

		chop_crlf(buf);
		if (buf[0] == '#' || strlen(buf) == 0) continue;
		col = 0;
		token = (char*)strtok(buf, seps);

		while (token != NULL) {
			if (col == 0) {
```

```
                        sscanf(token, "%s", &arr_list[row].p_code);
                    } else if (col == 1) {
                        sscanf(token, "%s", &g_code);
                        for (i = 0; i < g_count; i++) {
                            if (strcmp(g_table[i].g_code, g_code) == 0) {
                                arr_list[row].g_code = i;
                                break;
                            }
                        }
                    } else if (col == 2) {
                        sscanf(token, "%d", &arr_list[row].lot);
                    }
                    token = (char*)strtok(NULL, seps);
                    col++;
                }

                row++;
            }

            *arr_count = row;

            fclose(fp);

            return arr_list;
}

//出荷リスト読み込み
ship_t* readShipList(group_t *g_table, int g_count, int *ship_count, char *filename)
{
            //*********************************************************************//
            char buf[256], *token;
            int i, col, row = 0;
            char g_code[20];
            FILE *fp;
            //*********************************************************************//

            fp = freadopen(filename);

            ship_t *ship_list = (ship_t*)calloc(readFileRow(fp), sizeof(ship_t));

            rewind(fp);

            while (fgets(buf, 256, fp)) {
```

```
		chop_crlf(buf);
		if (buf[0] == '#' || strlen(buf) == 0) continue;
		col = 0;
		token = (char*)strtok(buf, seps);

		while (token != NULL) {
			if (col == 0) {
				sscanf(token, "%s", &ship_list[row].p_code);
			} else if (col == 1) {
				sscanf(token, "%s", &g_code);
				for (i = 0; i < g_count; i++) {
					if (strcmp(g_table[i].g_code, g_code) == 0) {
						ship_list[row].g_code = i;
						break;
					}
				}
			}
			token = (char*)strtok(NULL, seps);
			col++;
		}

		row++;
	}

	*ship_count = row;

	fclose(fp);

	return ship_list;
}

//分類表読み込み
group_t* readGroupFile(int *g_count, char *filename)
{
	//*********************************************************************//
	char buf[256], *token;
	int col;
	int row = 0;
	FILE *fp;
	//*********************************************************************//

	fp = freadopen(filename);
```

```
	group_t *g_table = allocGroupTable(readFileRow(fp));

	rewind(fp);
	while (fgets(buf, 256, fp)) {

		chop_crlf(buf);
		if (buf[0] == '#' || strlen(buf) == 0) continue;
		col = 0;
		token = (char*)strtok(buf, seps);

		while (token != NULL) {
			if (col == 0) {
				sscanf(token, "%s", &g_table[row].g_code);
			} else if (col == 1) {
				sscanf(token, "%lf", &g_table[row].ratio);
			}
			token = (char*)strtok(NULL, seps);
			col++;
		}

		row++;
	}

	*g_count = row;

	fclose(fp);

	return g_table;
}

//ファイルの行数判定
int readFileRow(FILE *fp)
{
	//*********************************************************************//
	char buf[256];
	int row;
	//*********************************************************************//

	rewind(fp);
	row = 0;

	while (fgets(buf, 256, fp)) {
```

```
			chop_crlf(buf);
			if(buf[0] == '#' || strlen(buf) == 0) continue;
			row += 1;
		}
		return row;
}

//使用すべきコスト・速度テーブルの決定
int condition(int rd)
{
		//*************************************************************************//
		int n;
		int i;
		double h;
		//*************************************************************************//

		h = rd * RH;

		for (i = 0; i < N; i++) {
			if (h <= CostTable[i].kh) {
				n = i;
				break;
			}
		}

		return n;
}

//動的配列のメモリ確保
rack_t**** allocRackSet(int rd, int cn, int rl)
{
		//*************************************************************************//
		int i, j, k, l;
		bool *bp;
		int *ip;
		//*************************************************************************//

		rack_t ****rack = (rack_t****)calloc(cn, sizeof(rack_t***));

		#pragma omp parallel for
		for (i = 0; i < cn; i++) {
			rack[i] = (rack_t***)calloc(2, sizeof(rack_t**));
			for (j = 0; j < 2; j++) {
```

```
                rack[i][j] = (rack_t**)calloc(rd, sizeof(rack_t*));
                for (k = 0; k < rd; k++) {
                        rack[i][j][k] = (rack_t*)calloc(rl, sizeof(rack_t));
                        for (l = 0; l < rl; l++) {
                                bp = &rack[i][j][k][l].flag;
                                *bp = false;
                                ip = &rack[i][j][k][l].lot;
                                ip = 0;
                        }
                }
        }
    }

    return rack;
}

double* allocCraneTime(int cn)
{
    double *cn_t = (double*)calloc(cn, sizeof(double));
    return cn_t;
}

bool**** allocRackFlag(int cn, int rd, int rl)
{
    //*************************************************************************//
    int i, j, k, l;
    bool *bp;
    //*************************************************************************//

    bool ****flag = (bool****)calloc(cn, sizeof(bool***));

    #pragma omp parallel for
    for (i = 0; i < cn; i++) {
        flag[i] = (bool***)calloc(2, sizeof(bool**));
        for (j = 0; j < 2; j++) {
            flag[i][j] = (bool**)calloc(rd, sizeof(bool*));
            for (k = 0; k < rd; k++) {
                flag[i][j][k] = (bool*)calloc(rl, sizeof(bool));
                for (l = 0; l < rl; l++) {
                    bp = &flag[i][j][k][l];
                    *bp = false;
                }
            }
```

```
                }
        }

        return flag;
}

group_t *allocGroupTable(int row)
{
        group_t *g_table = (group_t*)calloc(row,sizeof(group_t));
        return g_table;
}

//動的配列のメモリ解放
void freeRackSet(rack_t ****rack, int cn, int rd)
{
        //*************************************************************************//
        int i, j, k;
        //*************************************************************************//

        #pragma omp parallel for
        for (i = 0; i < cn; i++) {
                for (j = 0; j < 2; j++) {
                        for (k = 0; k < rd; k++) {
                                free(rack[i][j][k]);
                        }
                        free(rack[i][j]);
                }
                free(rack[i]);
        }

        free(rack);
}

void freeRackFlag(bool ****flag, int cn, int rd)
{
        //*************************************************************************//
        int i, j, k;
        //*************************************************************************//

        #pragma omp parallel for
        for (i = 0; i < cn; i++) {
                for (j = 0; j < 2; j++) {
                        for (k = 0; k < rd; k++) {
```

```
                                        free(flag[i][j][k]);
                                }
                                free(flag[i][j]);
                        }
                        free(flag[i]);
                }

                free(flag);
}

//改行コードの削除
void chop_crlf(char *buff)
{
                //*************************************************************************//
                int i;
                //*************************************************************************//

                #pragma omp parallel for
                for (i = 0; i < (int)strlen(buff); i++) {
                        if (buff[i] == '¥r' || buff[i] == '¥n') buff[i] = 0;
                }

}

//ファイルオープン(読み込み用、ファイル有無チェック)
FILE* freadopen(char *filename)
{
                //*************************************************************************//
                FILE *fp;
                //*************************************************************************//

                fp = fopen(filename, "r");

                if (fp == NULL) {
                        fprintf(stderr, "%s: No such file¥n", filename);
                        exit(1);
                }

                return fp;
}

//高分解能タイマー関数[マイクロ秒]
double timer()
```

```
{
#ifdef _MSC_VER
        //Windows 用: QueryPerformanceCounter 使用(精度 ns)
        //************************************************************************//
        static LARGE_INTEGER frequency;
        LARGE_INTEGER now;
        //************************************************************************//
        if (frequency.QuadPart == 0) {
                ::QueryPerformanceFrequency(&frequency);
        }
        ::QueryPerformanceCounter(&now);
        return now.QuadPart / double(frequency.QuadPart);
#elif __APPLE__
        //Mac OSX 用: gettimeofday 使用(精度 us)
        //************************************************************************//
        struct timeval now;
        //************************************************************************//
        gettimeofday(&timeval, NULL);
        return now.tv_sec + now.tv_usec / 1000000.0;
#else
        //Linux 用: clock_gettime 使用(精度 ns)
        //************************************************************************//
        struct timespec now;
        //************************************************************************//
        clock_gettime(CLOCK_MONOTONIC, &now);
        return now.tv_sec + now.tv_nsec / 1000000000.0;
#endif
}
```

謝　辞

本論文は，日本大学大学院生産工学研究科マネジメント工学専攻教授・博士(工学)若林敬造先生の懇切丁寧なるご指導とお力添えによる成果であり，ここに謹んで感謝の意を表します。先生の温かいご指導・ご鞭撻なくしては本論文の完成はなかったものと，心から厚く御礼申し上げます。

本論文の作成に当たりましては，学術的な視点から多大なるご指導を賜りました日本大学大学院生産工学研究科マネジメント工学専攻教授・博士(工学)鈴木邦成先生，日本大学大学院生産工学研究科マネジメント工学専攻教授・博士(農学)五十部誠一郎先生，日本大学大学院生産工学研究科マネジメント工学専攻教授・博士(工学)豊谷純先生に心から厚く御礼申し上げます。

本研究の遂行ならびに論文作成のあらゆる面でご指導，ご助言をいただきました神奈川大学名誉教授・工学博士唐澤豊先生の献身的な御努力とご熱意に感謝申し上げます。

本研究においてプログラム開発などにご指導，ご協力，ご助言いただきました早稲田大学理工学術院創造理工学部経営システム工学科助手・博士（工学）佐藤哲也先生に謝意を表します。

最後に，様々な面におきましてご協力いただきました日本大学大学院生産工学研究科マネジメント工学専攻の先生ならびに事務局の皆様方，および私の勤務先である株式会社京急百貨店ならび株式会社京急自動車学校の皆様方，および多大なご協力いただきました百貨店業界の皆様方に感謝を申し上げます。