博　士　論　文

強化学習を用いたコンテナ船物流向け
多品目在庫管理に関する研究

指導教員　　森川 博之 教授

東京大学大学院工学系研究科
先端学際工学専攻

氏　名　　37-187129　末次 浩詩

提　出　日　2020年12月1日

目次

図目次

表目次

第１章

序論

1.1　本論文の背景と目的

本論文は、コンテナ船輸送および賃貸型物流倉庫を利用するサプライチェーンにおける多品目在庫システムの在庫管理を対象とする。

図 1.1 に本論文で考えるサプライチェーンと在庫管理計画を示す。本論文では、単層のサプライチェーンを対象とする。倉庫に注文（需要）を満たす在庫がなければ機会損失費用が発生し、倉庫の在庫に対しては在庫保管費用が発生し、補充に対しては発注費用が発生する。この 3 つの総費用が最小になるような在庫管理計画を考える。在庫管理計画は、補充計画と倉庫容量計画の異なる 2 つの時間軸の計画から構成されるものとする。補充計画とは商品ごとの補充量を決定する週次サイクルの計画であり、倉庫容量計画は在庫倉庫の容量を決定する年次サイクルの計画である。

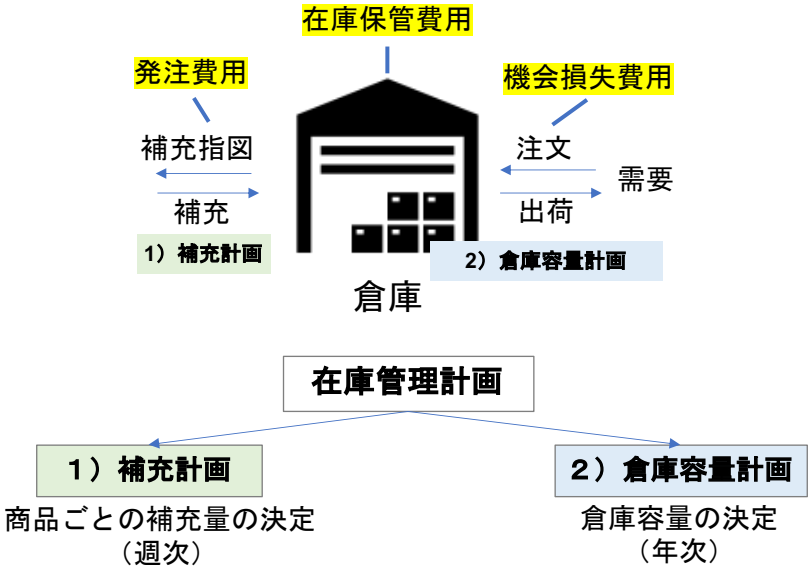

図 1.1　サプライチェーンと在庫管理計画

本論文は、プライベート（PB）商品への着目を動機としている。近年多くの小売業にとって PB 商品は利益の源泉となっている [10]。さらなる利益率の向上に向け、生産国に倉庫をもち、自社で日本の倉庫に輸送する形態が増えている。これにより、生産国側の在庫費用や海上輸送費用が仕入価格から差し引かれ、仕入価格の低減につながるが、海外調達費用の削減が求められる。

海外調達 PB 商品のサプライチェーン上の特徴は大きく 2 つある。FCL（Full Container

Load）によるコンテナ船輸送と、賃貸型倉庫の活用である。

FCL とは、コンテナ船による海上輸送において、コンテナ 1 本単位で輸送する形態のことであり、コンテナ内の荷物の積載率に関わらずコンテナ本数に対して費用が課される。そのほかの形態としては LCL（Less Than Container Load）がある。LCL では、1 つのコンテナ内に複数の企業の荷物が混在する形態であり、物量に応じて費用が課される。LCL は小ロットでの輸送が可能である代わりに、高積載率での FCL に比べると割高な費用となる。一般に、PB 商品は低価格が売りであるため、海上輸送には FCL が用いられることが多い。

賃貸型倉庫は、近年日本国内で一般的な倉庫の利用形態である。以前は在庫を所有する企業が自社倉庫を資産として保有する形態が一般的だったが、現在では東京都市圏で賃貸型の比率は 70% を超える [11]。賃貸型倉庫の場合、初期投資が不要であることに加え、保有倉庫と比べて倉庫容量の変更がより柔軟に可能である。PB 商品は一般に需要の成長率が高く、将来の需要の不確実性が大きい。そのため、国内在庫倉庫として賃貸倉庫が利用される。

次に、FCL によるコンテナ船輸送と賃貸倉庫利用の在庫管理における意味合いを説明する。

FCL によるコンテナ船輸送では、輸送費用は発注量に対して階段状の費用となり、商品単位で輸送費用を計算することはできない。このように商品単位には計算できない発注費用は「共通発注費用」と呼ばれ、共通発注費用が存在する在庫システムは「多品目在庫システム」と呼ばれる。

賃貸倉庫利用においては、在庫管理において 2 つの特徴が挙げられる。1 つは、在庫保管費用が在庫量に対して非線形な関数となることである。一般に、在庫量が契約容量以下の場合には所定の固定的な費用が課され、契約容量を超えた部分についてはペナルティーとして割高な変動費用が課される。これにより、在庫保管費用は在庫量に対して非線形な費用関数となる。もう 1 つの特徴として、賃貸倉庫は保有倉庫と異なり倉庫容量の変更が可能である。物量の増加など外部環境の変化に合わせ、途中で倉庫容量の変更ができることから、倉庫容量計画は動的な計画となる。

以上をまとめると、コンテナ船輸送、及び、賃貸倉庫利用を想定した多品目在庫システムにおいて、補充計画と動的倉庫容量計画を求めることが本論文の全体像である。図 1.2 に全体像を示す。特に多品目在庫システムにおける補充計画は一般に同時補充問題と呼ばれ、多くの既存研究が存在する。

最後に、本論文全体の問題設定を説明する。補充計画では在庫保管費用、輸送費用、および機会損失費用の総費用最小化を目的とする。動的倉庫容量計画では、割引現在価値の

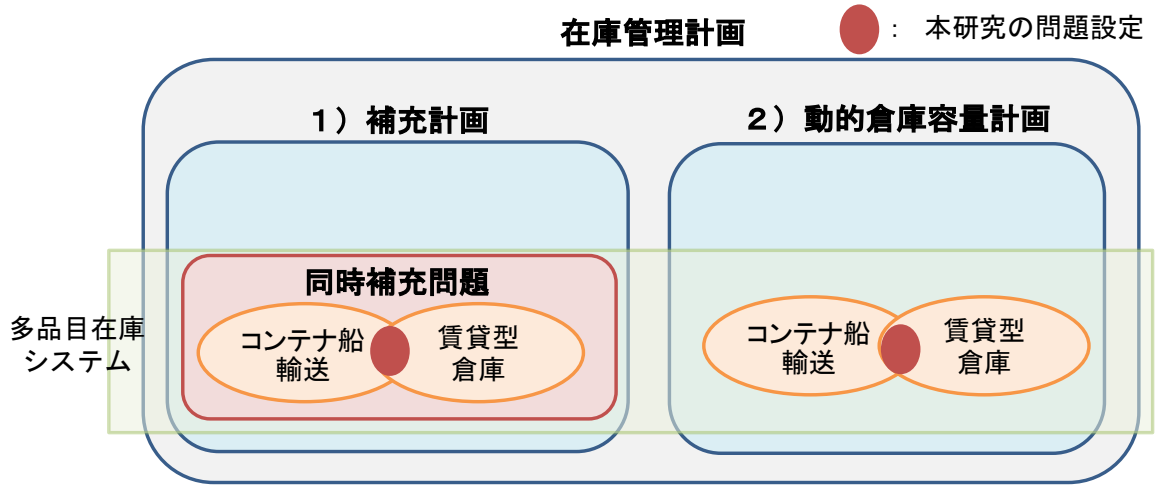

図 1.2　本論文の全体像

最大化を目的とする。割引現在価値、もしくは NPV（Net Present Value）は、各年の利益の累積割引和として表される。ここで、各年の利益とは売上から総費用を差し引いたものとする。

需要に関しては、同一年内は多次元正規分布に従う定常な需要を想定し、その平均ベクトルとして表される需要水準は毎年確率的に変動するものとする。

1.2　本論文の構成

本論文の具体的な構成は以下の通りである。本論文の構成を図 1.3 に示す。

第 1 章　序論
第 2 章　多品目在庫システムの補充計画と倉庫容量計画
第 3 章　マルチエージェント強化学習を用いた補充計画
第 4 章　費用推定モデルを用いた動的倉庫容量計画
第 5 章　結論

まず、第 2 章では多品目在庫システムにおける補充計画と倉庫容量計画それぞれについて先行研究を概説し、本論文の目的からみた従来研究の課題を述べる。

第 3 章では、多品目在庫システムの補充計画について、第 2 章にて示した課題を解決するため、マルチエージェント強化学習を用いた解法を示す。商品ごとの補充数の相互作用が大きく、高次元離散行動空間の問題である同時補充問題に対して、自商品以外の合計補

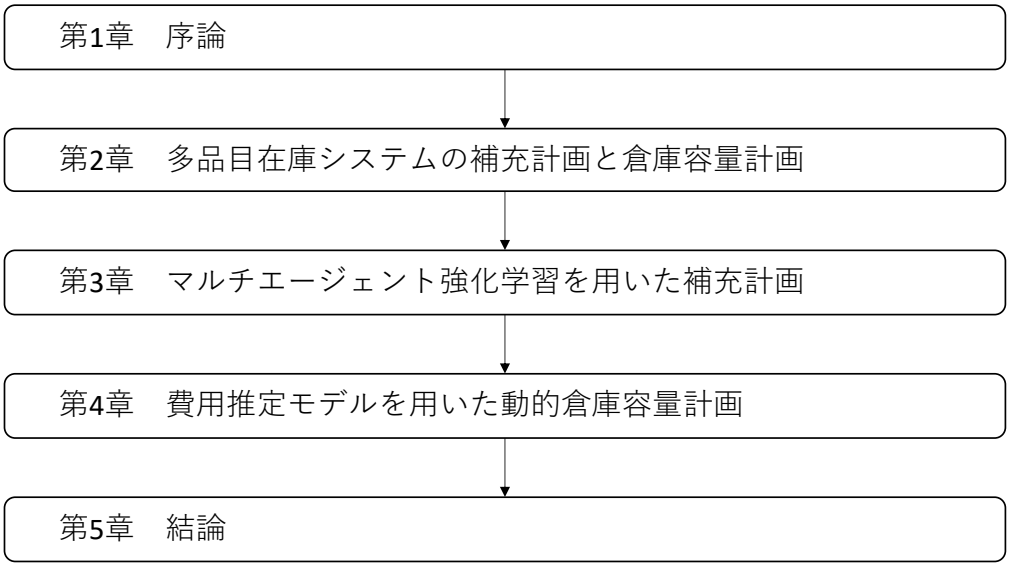

図 1.3　本論文の構成

充量で条件つけた行動価値を学習する C-IQL（Conditional Independent Q-Learner）を提案し、数値実験を通じて複数の取引条件下で既存の近似方策同等以上のパフォーマンスが得られることを示す。

第 4 章では、多品目在庫システムにおける動的倉庫容量計画について示す。これまで従来研究では、補充方策と倉庫容量を同時決定する手法は確立されていない。本論文では、倉庫容量のとりうる値が離散的であるという仮定のもと、ガウス過程回帰を用いて費用関数を推定し、その費用推定モデルを用いて動的倉庫容量計画を立案する手法を示す。

最後に第 5 章でまとめとする。本論文の主たる成果について述べ、今後の課題について言及する。

第２章

多品目在庫システムの補充計画と倉庫容量計画

2.1　はじめに

本章では、まず商品間で共通発注費用がある多品目在庫システムにおける補充計画の立案問題、すなわち同時補充問題に関して、決定的な需要と確率的な需要それぞれでの先行研究を説明する。次に、多品目在庫システムにおける倉庫容量計画について説明する。

2.2　補充計画

本節では、一般的な同時補充問題について説明し、需要及び取引条件の 2 つの観点から先行研究を概説する。

図 2.1 に同時補充問題のイメージを示す。

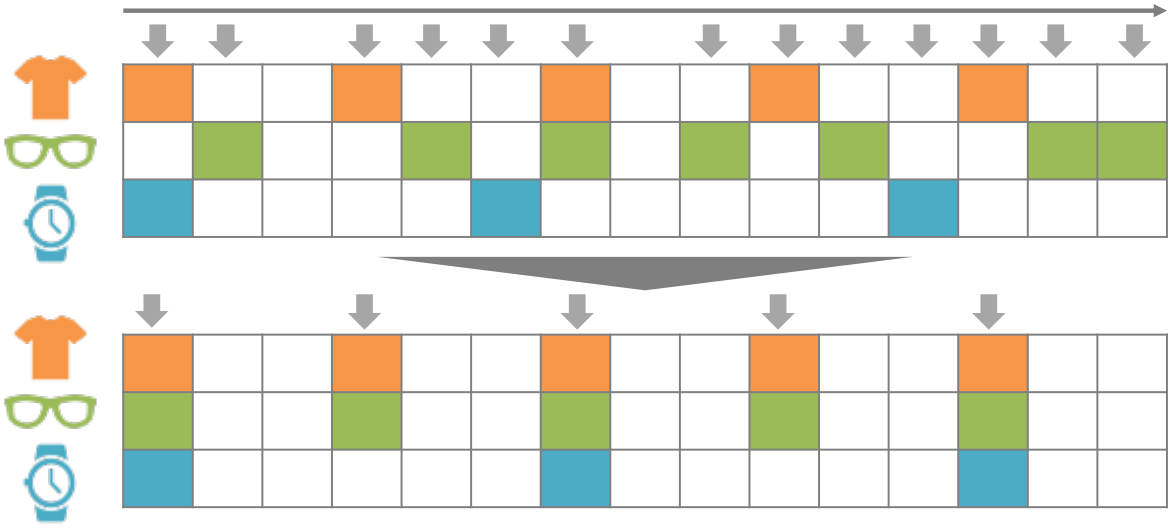

図 2.1　同時補充問題

共通発注費用が存在する場合には、図 2.1 上段のように商品ごとに独立に補充タイミングを決定するのではなく、下段のように商品間で補充タイミングを合わせることで、共通発注費用の削減が可能である。その際に、在庫保管費用や在庫切れに伴う機会損失費用とのトレードオフの考慮が重要となる。

同時補充問題に該当する実際の取引条件は数多く、例として以下が挙げられる。

- コンテナ船/トラック輸送費用 (1 本, 台あたり)
- 港使用料・通関料 (1 回あたり)

- 輸送費用のボリュームディスカウント (ton, m^3 ごと)

同時補充問題の複雑さは、需要に関する想定によって大きく異なる。決定的な需要、すなわち将来の需要が既知で固定的な場合には、同時補充問題は商品別の発注間隔を求める問題となる。一方で、確率的な需要を考えると、各商品の在庫量などを状態とする状態空間から、各商品の補充数を表す決定空間への写像として補充方策を求める問題となる。確率的な需要においては、商品数が 2 の場合ですら最適解が複雑な形式であることが知られている [22, 23] 。

同時補充問題とは、一般に共通発注費用 S の存在の元で、発注費用と在庫保管費用の合計を最小とする補充方策を求める問題であり、合計費用 (TC) は以下のように表される。ここでは、簡単のため決定的な需要を想定した定式化を示している。

$$TC(T, \mathbf{K}) = \frac{T}{2} \sum_{i=1}^{n} k_i D_i h_i + \frac{S + \sum_{i=1}^{n} \frac{s_i}{k_i}}{T} \tag{2.1}$$

ここで、n は商品数、D_i は商品 i の需要, $K = \{k_i\}_{i=0}^{n}$ は基本発注間隔 T の整数倍として表される商品 i の発注間隔、h_i は商品 i の在庫保管単価、S は共通発注費用、s_i は商品に紐づく発注費用である。ここでは以下の費用が想定されており、これを本論文では基本取引条件と呼ぶ。

- 固定的な発注費用
- 在庫量に対し線形な在庫保管費用
- 在庫量や発注量に関して無制約

図 2.2 に、基本取引条件と、本論文で想定する取引条件を示す。従来研究で想定される基本取引条件に対し、本論文では、輸送容量制約もしくは輸送量に対して階段状の輸送費用と、非線形な在庫保管費用を考える。

同時補充問題の従来研究は、需要が決定的か確率的か、および、基本取引条件を考えるか追加的な条件を考慮するか、という 2 軸により整理される。2006-2015 年の 10 年間に特定のジャーナル紙に掲載された同時補充問題の論文 128 本のうち確率的需要のものは 19 本であり、大半は決定的需要が想定されている [12]。図 2.3 にここ十数年での同時補充問題の研究動向を示す。確率的需要を想定した研究では、大半は基本取引条件を想定している。確率的需要において在庫や輸送制約といった追加的条件を考慮した研究は少数のみ存在する。本論文の対象はこの分類に該当し、コンテナ輸送における階段状の輸送費用及び非線形な在庫保管費用を考慮する必要がある。

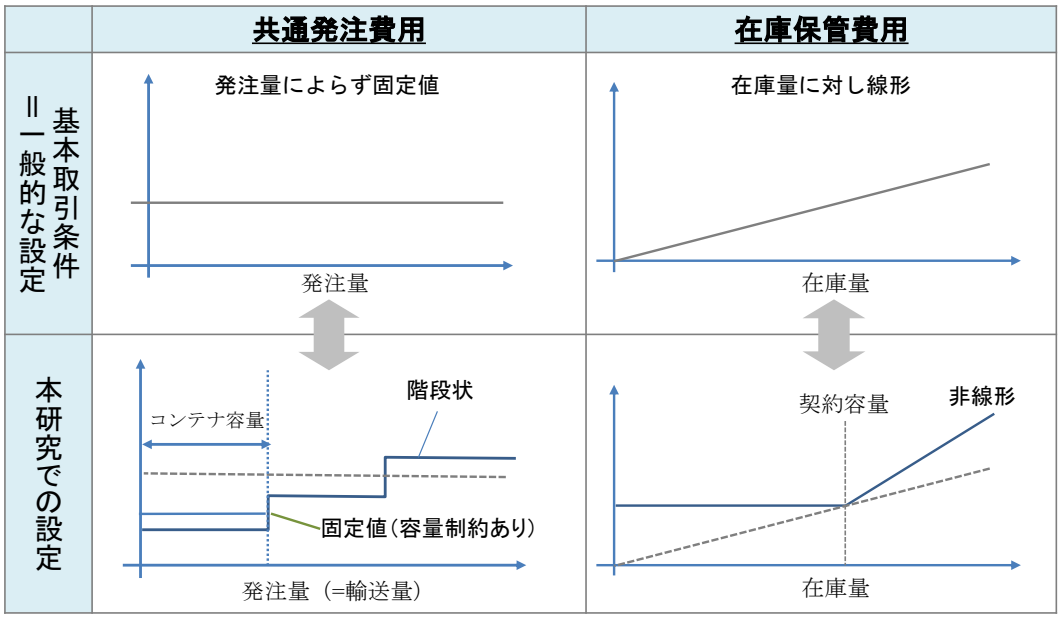

図 2.2　本論文と従来研究における同時補充問題における取引条件

需要の想定 （求めるもの）	基本取引条件における 代表的な解法	追加的条件の考慮
決定的需要 $(\boldsymbol{K})$ $\boldsymbol{K}$:商品別発注間隔	RANDアルゴリズム	制約条件の考慮 - 倉庫容量・輸送容量・予算制約 - 最小補充量制約 その他 - 多層サプライチェーンの考慮 - 陳腐化する在庫の考慮
確率的需要 $(\pi: S \rightarrow A)$ π：補充方策	近似方策 ・発注可能点方策 ・定期発注方策	在庫・輸送制約を考慮したものが 少数存在 **本研究の対象**

図 2.3　同時補充問題の研究動向

2.2.1 決定的需要の同時補充問題

(a) 代表的な解法: RAND

決定的需要においては、効率の良い計算方法として RAND [13] が提案され、改良版が数多く提案されてきた。決定的需要では、式 2.1 における基本発注間隔 T および、商品別の発注間隔を表す $\mathbf{K}$ を求める問題となる。$\mathbf{K}$ を固定した場合、式 2.1 の T による偏微分は、

$$\frac{\partial TC}{\partial T} = \frac{\sum k_i D_i h_i}{2} - \frac{S + \sum \frac{s_i}{k_i}}{T^2} \tag{2.2}$$

であるため、1 階の条件より TC を最小とする T_0 は、以下で表される。

$$T_0 = \sqrt{\frac{2(S + \sum \frac{s_i}{k_i})}{\sum k_i D_i h_i}} \tag{2.3}$$

また、Goyal [14] により、T の上限と下限は以下のように与えられている。

$$T_{\max} = \left[2 \left(S + \sum_{i=1}^{n} s_i \right) / \sum_{i=1}^{n} D_i h_i \right]^{1/2}, \tag{2.4}$$

$$T_{\min} = \min_{1 \leqslant i \leqslant n} \sqrt{\frac{s_i}{h_i D_i}} \tag{2.5}$$

RAND は、以下の手順で最も TC が低い T および $\mathbf{K}$ を見つける。

(1) $T_{\min}$ と $T_{\max}$ の範囲を m 個に等間隔に分割

(2) 各 T に対し、商品 i ごとに以下の $k_{i,r}$ を計算

$$k_{i,r}^2 = 2 s_i T_j^2 / D_i h_i \tag{2.6}$$

(3) 以下を満たす $k_{i,r}^*$ を取得

$$k_{i,r}^* = L \quad \text{if } L(L-1) < k_{i,r}^2 \leqslant L(L+1) \tag{2.7}$$

(4) $\mathbf{K}$ を固定し、式.2.3 に従い T を更新

(5) TC が最も低い $(T, \mathbf{K})$ を出力

RAND は最適解との差異が 0.002% であることが報告されている [15]。また、実務的な制約を踏まえ、発注間隔について 2 の冪乗という制約を加えることで、発注間隔の制約のない設定での費用との差異を 6% 以内に抑えた上で、さらなる計算効率の向上が達成されている [16]。

(b)　追加的な取引条件の考慮

近年の研究では、基本取引条件に対し、より実際の物流条件に即した設定を目指し、様々な拡張が提案されてきた [17–20]。例えば、予算についての上限制約 B が課される場合、以下の制約のもとで式 2.1 が最小となる $(T, \mathbf{K})$ を見つける問題となる。

$$\sum_{i=1}^{n} D_i k_i T b_i \leqslant B \tag{2.8}$$

ここで、b_i は商品 i の単価を表す。

Moon ら [20] は、RAND を拡張した Constraint-RAND（C-RAND）と呼ばれるラグランジュ未定乗数法を用いて制約条件を考慮した手法と、遺伝的アルゴリズムを用いた手法の 2 つの手法を提案している。数値実験の結果、C-RAND の方が総費用の観点で良い結果が得られているものの、遺伝的アルゴリズムによる手法の利点として、他の制約条件を容易に組み込めることが強調されている。

2.2.2　確率的需要の同時補充問題

まず、決定的需要ではなく確率的需要を想定することで、1 商品の場合においても総費用の観点で最適補充方策が変化することを示す。式 2.1 は同時補充問題における合計費用を表したものであるが、これは決定的需要を想定したものである。式 2.1 において在庫量はサイクル在庫のみで表現されているが、確率的需要を想定した場合には、欠品を抑えるために安全在庫が必要となる。1 商品で需要の予測誤差に正規分布を仮定した場合、発注間隔 T での平均費用は以下のように近似的に表される [21]。

$$TC = \frac{TDh}{2} + z\sigma h\sqrt{T+L} + \frac{S}{T} \tag{2.9}$$

ここで、L は納品までのリードタイム、z は許容欠品水準を規定する安全係数、σ は予測誤差の標準偏差を示す。ここで、右辺の第 1 項がサイクル在庫、第 2 項が安全在庫を表す。1 階の条件は、

$$\frac{\partial TC}{\partial T} = \frac{Dh}{2} + \frac{z\sigma h}{2\sqrt{T+L}} - \frac{a}{T^2} = 0 \tag{2.10}$$

となる。この解は閉形式では求めることはできないが、$\frac{\partial z\sigma h\sqrt{T+L}}{\partial T}$ の項が決定的需要の場合との差であるので、上式の解を T^* とすると、1 商品の場合の決定的需要の解 $T_0^d = \sqrt{(2S/hD)}$ よりも小さい、すなわち $T^* < T_0^d$ であり、決定的需要の場合と比較して発注間隔は短くなる。

(a)　代表的な近似方策

確率的需要の場合、同時補充問題はマルコフ決定過程で定式化される。マルコフ決定過程においては、状態遷移確率や報酬関数が既知の場合、方策反復法もしくは価値反復法により最適方策を得ることが可能である。同時補充問題においては、需要分布に関する何らかの仮定のもとで、状態遷移確率や報酬関数は既知となるため、方策反復法や価値反復法により最適方策が求まる。

マルコフ決定過程における最適方策の導出については次の章で説明するが、方策反復法の場合は $\mathcal{O}(|S|^2|A|)$、価値反復法の場合には $\mathcal{O}(|S|^3)$ の計算量が必要である。一方、同時補充問題では、状態空間、決定空間が共に商品数に対して指数的に増加してしまうことが問題である。確率的需要の同時補充問題に対し、探索空間を限定して計算量を改善した修正方策反復法による解法が提案されている [22, 23] が、解が得られているのは高々商品数が 4 までの場合である。そのため、より商品数が多い場合については、少数のパラメータで表現した近似方策が提案されてきた。

ここで、確率的需要の同時補充問題において、既存研究で想定されてきた在庫観測・発注機会に関する 2 つの設定を導入する。1 つは在庫量の観測及び発注機会が連続であり、任意のタイミングで発注が可能な想定であり、もう 1 つは、在庫量の観測もしくは発注機会が定期的に一定間隔で訪れるという想定である。本論文では、前者を連続的在庫観測システム (Continuous-Review Inventory System)、後者を定期的在庫観測システム (Periodic-Review Inventory System) と呼ぶ。

既存研究の多くは連続的在庫観測システムを想定しているが、本論文では定期的在庫観測システムを想定する。近年では IT 技術の浸透により実務においても多くのケースでほぼリアルタイムで在庫量の観測は可能である一方、発注機会については、海外調達を想定すると、一般に港業務の制約等の観点から週 1 から 2 回程度に限定されるためである。

定期的在庫観測システムにおける代表的な近似方策に発注可能点方策と定期発注方策がある。発注可能点方策は、Balintfy により連続的在庫観測システムの仮定のもとで提案された [24]。効率的なパラメータ推定方法が提案され [25–27]、その後定期的在庫観測システム向けに拡張された [28]。発注可能点方策は、商品ごとに 3 つのパラメータ (s_i, c_i, S_i) を持つ。在庫位置が発注点 s_i 以下になると S_i になる数量が発注される。一方、在庫位置が発注点 s_i より大きく発注可能点 c_i 以下の場合には、同じタイミングで発注点以下となる商品が存在する場合には、この商品も発注に含まれ、S_i になる数量が発注される。

定期発注方策 [29] は Atkins により提案され、その後 Viswanathan [30] により MP（Modified Periodic）方策として改良された。MP 方策は、発注間隔を表すグローバルな

パラメータ T と、商品別のパラメータ (s_i, S_i) を持つ。一定間隔 T ごとに発注機会が訪れ、そのタイミングにおいて商品 i の在庫が s_i 以下であれば発注対象となり S_i まで発注される。

図 (a) に発注可能点方策、図 (a) に MP 方策を図示する。

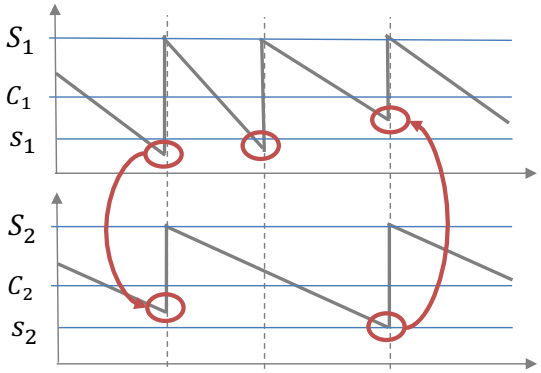

図 2.4　発注可能点方策

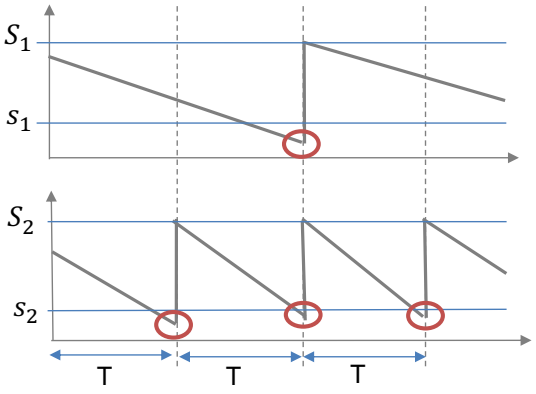

図 2.5　MP 方策

発注可能点方策や MP 方策では、方策の型を明示的に限定しているため、需要の特性に応じて適切な近似方策が異なるという課題がある。発注可能点方策は、商品間の需要に相関がある場合に最適解との差異が大きくなることが指摘されている [31–33]。一方で、需要のばらつきが大きい場合には発注可能点方策が MP 方策よりも優れていると報告されている [28]。

次に、近似方策のパラメータ導出方法について述べる。近似方策のパラメータ数は、発注可能点方策であれば $3n$、MP 方策であれば $2n + 1$ である。先行研究では商品ごとの子問題に分割して各商品のパラメータを反復的に更新する分割法が一般的に用いられてきた。同時補充問題においては、他の商品の補充機会に相乗りすることで、共通コストの削減が可能である。この相乗りする機会を割引発注機会と呼ぶ。分割法では、割引発注機会を近似表現することで、商品ごとのパラメータを反復的に独立に更新する。そのため、先行研究の多くが、基本取引条件のもとで欠品は考慮せず、需要に複合ポアソン過程を仮定するという近似表現が容易な問題設定としている。

しかし、近似方策におけるパラメータの推定において、分割法における近似誤差により、分割法を用いて求められた方策と最適方策との間に大きな差異があることが指摘されている。研究 [34] によると、発注可能点方策において、特に共通発注コストが大きい場合

には、分割法により得られた補充方策の元での総費用が、最適なパラメータでの発注可能点方策下での総費用に比べて大きいことが指摘されている。

(b)　追加的な取引条件の考慮

ここでは、確率的需要の同時補充問題に対して、基本取引条件以外の追加的な取引条件に関する先行研究について述べる。確率的需要において、基本取引条件以外の制約を考慮した研究はいくつか存在するが、現時点ではいずれも連続的在庫観測システムを想定したものに限られる。

Minner [32,33] は、連続的在庫観測システムにおいて倉庫容量制約を考慮した同時補充方策を提案している。倉庫容量制約のもとで、需要に複合ポアソン過程を仮定し、セミマルコフ決定過程で定式化した上で、ヒューリスティック解法を提案しているが、補充リードタイムは 0 を仮定しており、倉庫容量制約において正のリードタイムのもとで考えるべき将来需要の不確実性に対する安全在庫は考慮されていない。

同じく連続的在庫観測システムのもとで、QS 方策 [35] を前提とした、トラック積載量を考慮した同時補充方策も提案されている [36]。QS 方策は、全商品の合計在庫推移に従い発注タイミングを決める方策であり、発注点を表すグローバルなパラメータ s と商品ごとのパラメータ S_i を持つ。合計在庫数が s 以下になると、商品 i は S_i まで発注される。需要の到来タイミングがポアソン分布に従うとして、かつ 1 回あたりの需要量を 1 に限定すると、以下を満たすようにパラメータ $(s, \boldsymbol{S})$ を決定することで常に満載が達成できる。

$$\sum_{i=1}^{N} S_i - s = Q \tag{2.11}$$

ここで、Q はトラックの積載量である。しかし、定期的在庫観測システムの場合には発注機会が任意ではないため、発注機会が到来した際に s を大きく下回ることもあり、上記の QS 方策を用いた満載制約を満たす手法は適用できない。

以上、確率的需要のもとで基本取引条件以外の要素として倉庫容量および輸送容量を考慮した先行研究を説明した。これらは需要分布もしくは補充リードタイムに関して強い制約が置かれており、実務における一般的な設定に即したものにはなっていない。また、本論文が対象とする定期的在庫観測システムにおいては基本条件以外の条件を考慮した研究は存在していない。さらに、物流実務における取引条件は多様である。確率的需要における同時補充問題の複雑さから、先行研究では分割法を用いた近似方策の解法が提案されてきたが、実務における多様な取引条件に対し、先行研究で考慮されてきた取引条件は限定的である。

(c)　既存の近似方策の課題

以上を踏まえて確率的需要における同時補充問題の従来解法の課題をまとめる。

1 点目は、基本取引条件においても適切な近似方策が需要特性に応じて異なることである。これは、そもそも補充方策を少数のパラメータを持つ形に限定していること、及び、分割法を用いた方策パラメータの推定において最適なパラメータが必ずしも得られないことに起因する。

2 点目は、基本取引条件以外の要素を考慮した先行研究が限定的なことである。これまで輸送制約や倉庫容量制約など、特定の条件における解法が提案されてはいるものの数少ない。

本論文では、上記 2 点の課題に対して強化学習による手法を提案している。

2.3　倉庫容量計画

本節では、倉庫容量計画について説明する。

まず、倉庫容量計画は、補充の想定により 2 つに分類される [37]。図 2.6 にその分類を示す。

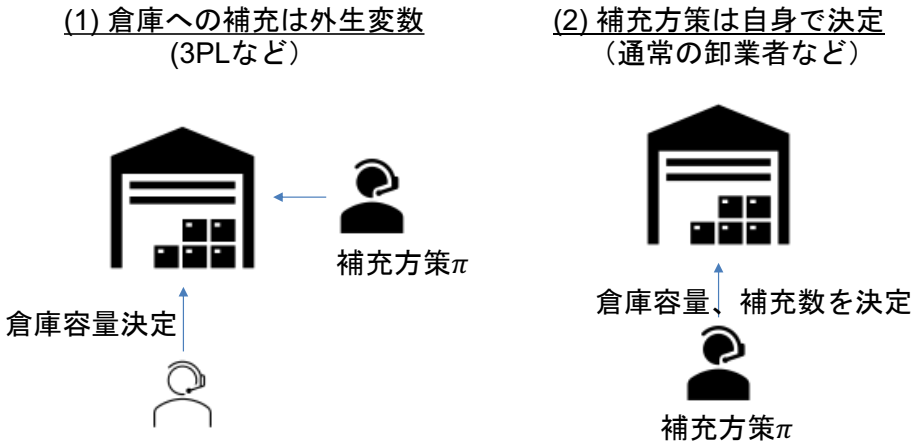

図 2.6　倉庫容量計画を考える上での分類

1 つは、倉庫への補充は倉庫容量の意思決定者がコントロールできない外生変数として与えられる、すなわち補充方策を所与とする設定で、例としては 3PL（サードパーティロジスティックス）の業者が該当する。この場合は、倉庫容量のみを決定する問題となるた

め、非定常な需要のもとで、動的倉庫容量計画が考えられてきた [37]。もう 1 つは補充方策と倉庫容量の意思決定者が同一のケースで、これは通常の卸業者などが該当する。本論文は後者に該当する。後者の場合、倉庫容量は在庫水準に依存し、そして在庫水準は補充方策により決まる、という関係性から、倉庫容量と補充方策は相互に依存しており、同時決定が求められる。

まず、前者の補充方策を所与とした場合の動的倉庫容量計画について説明する。動的倉庫容量計画は、一般に図 2.7 に表されるようなシナリオツリーが用いられる。ある時点の状態は、意思決定者による倉庫の縮小や拡張といった意思決定と、意思決定者がコントロールできない外部環境によって定まる需要変動という二つのノードにより次の状態に遷移する。そのため、状態は需要水準と倉庫容量から構成されるが、シナリオツリーの各ノードに対応する状態において、ある補充方策下での費用は定まるため、動的計画法を用いることで、累積期待割引利益、すなわち NPV が最大となる動的倉庫容量計画を求めることが可能である。

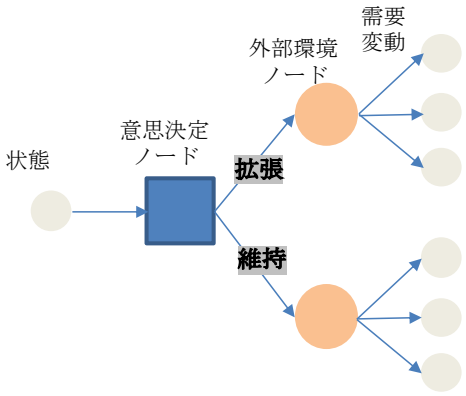

図 2.7　動的倉庫容量計画おけるシナリオツリー

次に、後者の補充方策と倉庫容量を同時決定する場合の研究動向を図 2.8 に示す。

倉庫容量計画についても、同時補充問題と同様に需要が決定的か確率的かで大別される。倉庫容量計画の既存研究の大半は、共通発注費用を存在がない設定であり、一方で、本論文で対象とする共通発注費用を考慮した研究は決定的需要を想定したものに限られる。決定的需要においては、固定的な共通発注費用、及び、在庫量に対して区分線形関数で表現される在庫保管費用の設定のもとで、Goh らにより既存の RAND を拡張した解法が提案されている [38]。また、Yao [39] らは同様の費用設定のもとで、より効率的に補充間隔及び倉庫容量が求まるアルゴリズムを提案している。

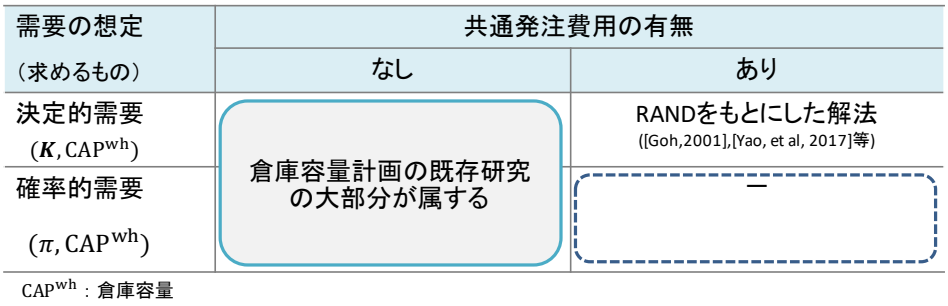

図 2.8　倉庫容量計画の分類

共通発注費用が存在し、かつ、確率的な需要のもとでは、補充方策と倉庫容量を同時に決定する解法は確立されていない。そこで本論文では、倉庫容量のとりうる値は離散的であるという仮定のもとで、倉庫容量を所与とした補充方策を求めるという同時補充問題を解いた上で、倉庫容量計画を求めるという 2 段階のアプローチをとる。すなわち、補充方策と倉庫容量を同時決定するのではなく、倉庫容量に関する離散空間においてそれぞれ補充方策を求めることで、総費用が最小となる倉庫容量を選択する。動的倉庫容量計画に拡張する場合においては、シナリオツリーにおける全てのノードにおいて補充方策を求めることで、動的倉庫容量計画を導くことが可能である。

2.4　おわりに

以上、多品目在庫システムにおける補充計画と倉庫容量計画について述べた。

確率的需要の同時補充問題は、商品数が大きい場合に最適方策は求まらないため、少数のパラメータを持つ近似方策が提案されてきた。しかし、近似方策は需要特性による得意不得意があり、適切な近似方策が需要特性によって異なるという課題がある。また、確率的需要においてはほとんどの既存研究が基本取引条件を前提としており、輸送制約や倉庫容量制約など、特定の条件における解法が提案されてはいるものの、需要分布や物流条件に強い制約が置かれている。特に本論文の対象である定期的在庫観測システムにおいては、基本取引条件以外を考慮した先行研究は存在していない。実務での多様な取引条件に対し、既存研究がカバーできている取引条件は限定的と言える。

倉庫容量計画については、倉庫容量は補充方策と相互に依存するため、補充方策と倉庫容量を同時に決定する必要がある。しかし、多品目在庫システムにおいて、確率的な需要のもとで補充方策と倉庫容量を同時に決定する解法は確立されていない。そこで本論文で

は、倉庫容量を所与とした補充方策を求めるという同時補充問題を解いた上で、倉庫容量計画を求めるという 2 段階のアプローチをとる。

第 3 章

マルチエージェント強化学習を用いた同時補充計画

3.1　はじめに

本章では、確率的需要における同時補充問題に対して、強化学習を用いた解法を提示する。

第 2 章で説明した通り、確率的需要においては、代表的な近似方策として発注可能点方策と MP 方策が提案されてきた。これらのルールベースの方策は、需要の特性によって、適切な方策が異なるという課題がある。また、基本取引条件以外の条件を考慮した研究は少ない。輸送制約や在庫制約など、追加的な条件を考慮しようとすると、方策の型自体、もしくはパラメータを求めるアルゴリズムを新たに考案しなければならず、実務における多様な取引条件に対して、研究が網羅する取引条件が追いついていない状況である。

そこで、既存研究における需要分布や取引条件によって新たな解法が必要となること自体が、同時補充問題の実務での広範な条件への適用の障壁となっていると考え、本論文では、状態遷移確率と報酬関数を用いない強化学習を用いた解法を提案する。ニューラルネットワークによる関数近似を用いた強化学習を用いることで、方策の型を限定することなく、かつ簡易な修正のみで基本取引条件以外の条件を考慮した場合への拡張が可能な解法の確立を目的とする。しかし、同時補充問題に対して強化学習を適用する場合、高次元離散行動空間が課題となる。商品数に対して、決定空間が指数的に増加してしまうためである。

本論文では、高次元離散行動空間に対応しつつ、協調的な補充方策を学習できる強化学習手法を提案する。具体的には、商品ごとに独立したエージェントとしたマルチエージェント強化学習を用い、他商品の合計補充量を状態の一部とした Q 学習手法を提案する。商品数や需要の分散といった需要特性と、輸送容量及び在庫保管容量を考慮した取引条件のもとでの数値実験により、総費用の観点で既存の近似方策との比較評価を実施した。

3.2　強化学習

本節では、マルコフ過程、マルコフ報酬過程を導入したのち、マルコフ決定過程を説明する。その後、マルコフ決定過程における解法として、状態遷移確率及び報酬関数が既知の場合に用いる政策反復法と価値反復法、及び状態遷移確率及び報酬関数を用いない強化学習手法である Q 学習について説明する。次に、行動空間に関する特徴の観点から強化学習手法を説明する。なお、以下の整理において、[1] 及び [40] を参考にしている。

3.2.1 マルコフ決定過程

(a) マルコフ過程

離散時刻 $t \in 0, 1, 2, ...$ における状態を S_t とするとき、確率過程は $\{S_0, S_1, S_2, ...\}$ と表現され、とり得る状態の集合 $\mathcal{S}$ を状態空間と呼ぶ。確率過程 $\{S_t; t \in \mathcal{N}\}$ が次のマルコフ性を満たす時、マルコフ過程と呼ぶ。

$$p(S_{t+1}|S_t) = p(S_{t+1}|S_1, S_2, ..., S_t) \tag{3.1}$$

これは、時点 $t+1$ における状態 S_{t+1} は時刻 t での状態 S_t が与えられたもとでそれ以前の状態とは独立であることを意味する。マルコフ過程 $\{S_t; t \in \mathcal{N}\}$ において、状態 s から状態 s' への状態遷移確率は、次のように表される。

$$p_{ss'} = p(S_{t+1} = s'|S_t = s) \tag{3.2}$$

全ての状態 $s \in \mathcal{S} = \{1, 2, ...n\}$ から次の状態への遷移確率を状態遷移行列として以下のように表される。マルコフ過程は、状態空間と状態遷移行列のタプル $< \mathcal{S}, \mathcal{P} >$ として表される。

$$\mathcal{P} = \begin{bmatrix} \mathcal{P}_{11} & \dots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \dots & \mathcal{P}_{nn} \end{bmatrix} \tag{3.3}$$

(b) マルコフ報酬過程

マルコフ報酬過程は、マルコフ過程に報酬関数と割引率が加わったもので、$< \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma >$ と表される。ここで、$\mathcal{R}$ は報酬関数であり、時刻に依存しない状態のみで定まる報酬を受け取るとする。すなわち、$\mathcal{R}_\mathcal{S} = \mathbb{E}[R_{t+1}|\mathcal{S}_t = s]$ である。$\gamma \in [0, 1]$ は割引率である。

ここで、収益と状態価値を導入する。収益 G_t は、時刻 t から無限期間にわたって得られる割引報酬和で次のように表される。

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3.4}$$

マルコフ報酬過程における状態 s における価値は、現在の状態 s における期待収益として次のように表される。

$$v(s) = \mathbb{E}\left[G_t \mid S_t = s\right] \tag{3.5}$$

状態価値は、即時報酬 R_{t+1} と、割引された遷移先の状態 S_{t+1} における状態価値として次のように分解される。これを状態価値に関するベルマン方程式と呼ばれる。

$$\begin{aligned} v(s) &= \mathbb{E}\left[G_t \mid S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma\left(R_{t+2} + \gamma R_{t+3} + \ldots\right) \mid S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma v\left(S_{t+1}\right) \mid S_t = s\right] \end{aligned} \tag{3.6}$$

ベルマン方程式は、全ての状態に対して、次のように行列表現される。ここで、$\boldsymbol{v}$ は状態を表す n 次元列ベクトルである。

$$\boldsymbol{v} = \mathcal{R} + \gamma \mathcal{P} \boldsymbol{v} \tag{3.7}$$

要素を表現すると次の通りである。

$$\begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ \vdots \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & \ldots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{11} & \ldots & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ \vdots \\ v(n) \end{bmatrix} \tag{3.8}$$

これは未知数 n の線形方程式であり、各状態の状態価値は次のように求まる。

$$\boldsymbol{v} = (I - \gamma \mathcal{P})^{-1} \mathcal{R} \tag{3.9}$$

(c) マルコフ決定過程

マルコフ決定過程は状態遷移がマルコフ性を持つ環境を表現し、タプル $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ として表される。マルコフ報酬過程に対して、$\mathcal{A}$ が加わっており、$\mathcal{A}$ は有限の決定空間である。ここで、$\mathcal{P}$ は状態遷移確率行列であり、状態 s で行動 a をとった時に状態 s' に遷移する確率は次のように定義される。

$$p^a_{ss'} = p(S_{t+1} = s' | S_t = s, A_t = a) \tag{3.10}$$

また、報酬関数 $\mathcal{R}$ は状態 s で行動 a をとった時に受け取る即時報酬 R_{t+1} の期待値であり、次のように定義される。

$$\mathcal{R}^a_{\mathcal{S}} = \mathbb{E}[R_{t+1} | S_t = s, A_t = a] \tag{3.11}$$

ここで、政策 π を導入する。Π を、時刻 t における履歴 $h_t = (s_0, a_0, s_1, a_1, ..., s_{t-1}, a_{t-1}, s_t) \in H_t$ に基づいて決定をする政策の集合であるとし、政策空間と呼ぶ。時刻 t における政策

π_t は次のように表される。

$$\pi_t(a|h_t) = \mathbb{P}\left[A_t = a \mid H_t = h\right] \tag{3.12}$$

このうち、各時刻 t でとる決定は状態 s_t のみにより定まり、s_t 以外の履歴 H_t によらない政策をマルコフ政策と呼び、次のように表現される。

$$\pi_t(a|s_t) = \mathbb{P}\left[A_t = a \mid S_t = s\right] \tag{3.13}$$

さらに、政策が時刻に依存しない場合、定常政策と呼ぶ。

$$\pi(a|s) = \mathbb{P}\left[A_t = a \mid S_t = s\right] \tag{3.14}$$

ここで、マルコフ決定過程 $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$、及びマルコフ定常政策 π の下で、状態と報酬の系列 $S_1, R_2, S_2, ...$ はマルコフ報酬過程 $< \mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma >$ となる。ここで、$\mathcal{P}^\pi, \mathcal{R}^\pi$ は次の通りである。

$$\mathcal{P}^\pi_{s,s'} = \sum_{a \in \mathcal{A}} \pi(a \mid s) \mathcal{P}^a_{ss'}, \tag{3.15}$$

$$\mathcal{R}^\pi_s = \sum_{a \in \mathcal{A}} \pi(a \mid s) \mathcal{R}^a_s \tag{3.16}$$

次に、状態価値関数 $v_\pi(s)$ と行動状態価値関数 $q_\pi(s,a)$ を次のように定義する。$v_\pi(s)$ とは、状態 s から始めて方策 π に従った場合の期待収益であり、$q_\pi(s,a)$ は、状態 s で行動 a をとったのちに方策 π に従った場合の期待収益を表す。

$$v_\pi(s) = \mathbb{E}_\pi\left[G_t \mid S_t = s\right], \tag{3.17}$$

$$q_\pi(s,a) = \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right] \tag{3.18}$$

ここで、同様に状態価値関数 $v_\pi(s)$ は即時報酬及び次の状態における割引価値の和として次のように分解できる。これは状態価値についてのベルマン期待方程式と呼ばれる。

$$v_\pi(s) = \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi\left(S_{t+1}\right) \mid S_t = s\right] \tag{3.19}$$

行動状態価値関数 $q_\pi(s,a)$ についても同様に以下のように分解でき、状態行動価値についてのベルマン期待方程式と呼ばれる。

$$q_\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} + \gamma q_\pi\left(S_{t+1}, A_{t+1}\right) \mid S_t = s, A_t = a\right] \tag{3.20}$$

以下 2 式より、

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) q_\pi(s,a), \tag{3.21}$$

$$q_\pi(s,a) = \mathcal{R}^a_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} v_\pi\left(s'\right) \tag{3.22}$$

式 3.19 は以下のように表される。

$$v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a \mid s) \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_\pi\left(s'\right) \right) \tag{3.23}$$

よって、$\boldsymbol{v}_\pi$ を各状態の状態価値を表す列ベクトルとすると、式 3.9 と同様に、以下のように直接求まる。

$$\boldsymbol{v}_\pi = \left(I - \gamma \mathcal{P}^\pi\right)^{-1} \mathcal{R}^\pi \tag{3.24}$$

状態行動価値についてのベルマン期待方程式 (式 3.20) は、次のように表される。

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a'\in\mathcal{A}} \pi\left(a' \mid s'\right) q_\pi\left(s', a'\right) \tag{3.25}$$

最適価値関数は、全ての政策における状態価値関数 $v_\pi(s)$ の最大値として、次のように定義される。

$$v_*(s) = \max_\pi v_\pi(s) \tag{3.26}$$

また、最適状態行動価値は、同様に全ての政策における状態行動価値関数 $q_\pi(s,a)$ の最大値として定義される。

$$q_*(s,a) = \max_\pi q_\pi(s,a) \tag{3.27}$$

ここで、政策の良さについて定義する。ある二つの政策 π, π' について、以下が成り立つ場合に政策 π は政策 π' よりも優れているとする。

$$v_\pi(s) \geq v_{\pi'}(s), \forall s, \tag{3.28}$$

$$v_\pi(s) > v_{\pi'}(s), \exists s \tag{3.29}$$

以下 2 式より、

$$v_*(s) = \max_a q_*(s,a), \tag{3.30}$$

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_*\left(s'\right) \tag{3.31}$$

次の最適値方程式を得る。

$$v_*(s) = \max_a \{\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a v_*\left(s'\right)\} \tag{3.32}$$

上記最適値方程式は閉形式で解くことはできないため、次以降で示す反復的な解法を用いる。

3.2.2 価値反復法と政策反復法

ここでは、報酬関数 $\mathcal{R}_s^a$ 及び状態遷移確率 $\mathcal{P}_{\mathrm{ss}'}^a$ が既知の場合の解法について説明する。以下、2 つの状態価値関数 u と v の無限大ノルムを次のように定義する。

$$\|u - v\|_\infty = \max_{s \in \mathcal{S}} |u(s) - v(s)| \tag{3.33}$$

(a) 価値反復法

まず、ある状態 s' に関して $v_*(s')$ が分かっているとすると、状態 s の $v_*(s)$ は次のように計算できる。

$$v_*(s) \leftarrow \max_{a \in \mathcal{A}} \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_* \left(s'\right) \tag{3.34}$$

上記の更新を全ての状態に対して反復的に計算するのが価値反復法である。価値反復法の手順を以下に示す。

(1) 関数 $v^0(s)$ $(s \in S)$ を選び、$\epsilon(> 0)$ を設定する。$k = 0$ とする。

(2) 各 $s \in S$ に対し、以下で $v^{k+1}(s)$ を更新する。

$$v^{k+1}(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^k \left(s'\right) \right) \tag{3.35}$$

(3) 次式を満たす場合は (4) に進み、そうでなければ $k = k + 1$ として (2) に戻る

$$\|v^{k+1} - v^k\|_\infty = \max_{s \in \mathcal{S}} |v^{k+1}(s) - v^k(s)| \leq \epsilon(1 - \gamma)/2\gamma \tag{3.36}$$

(4) 各 $s \in S$ に対し、次の $\pi_*(s)$ を出力して終了する。

$$\pi_*(s) = \arg\max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^{k+1} \left(s'\right) \right) \tag{3.37}$$

上記の手順において、価値関数の更新を作用素 T

$$T^*(v) = \max_{a \in \mathcal{A}} \mathcal{R}^a + \gamma \mathcal{P}^a v \tag{3.38}$$

とすると、作用素 T は γ 縮小写像であることが示されている。すなわち、以下が成り立つ。

$$\|T^*(u) - T^*(v)\|_\infty \leq \gamma \|u - v\|_\infty \tag{3.39}$$

そのため、上記のように $v^k = Tv^{k-1}$ と更新していくと $k \to \infty$ の時に $v^k(s)$ は $v_*(s)$ に収束する。また、終了条件を式 3.36 のように設定した場合には、ここで求められた政策 $\pi_*(s)$ は ϵ 最適政策であり、以下が成り立つ。

$$\|v_{\pi *}(s) - v_*(s)\|_\infty \leq \epsilon \tag{3.40}$$

(b)　政策反復法

価値反復法は、$v^k(s)$ を更新していく手法だったのに対し、政策反復法は決定を更新することにより最適政策に近づく方法である。定常な決定性マルコフ政策 π のもとでは、$a = \pi(s)$ であり、以下が成り立つ。

$$v_\pi(s) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi\left(s'\right), s \in \mathcal{S} \tag{3.41}$$

また、以下のような貪欲な行動選択により、政策を改善することができる。

$$\pi'(s) = \underset{a \in \mathcal{A}}{\text{argmax}} q_\pi(s, a) \tag{3.42}$$

この上記の政策評価と政策改善を反復的に計算するのが政策反復法である。政策反復法の手順を以下に示す。

(1) 決定性政策 $\pi^0(s)$ を選択し、$k = 0$ とする。

(2) $a = \pi^k(s)$ として、次の連立方程式を解き、$\{v_k(s); s \in \mathcal{S}\}$ を求める。

$$v^k(s) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^k\left(s'\right), s \in \mathcal{S} \tag{3.43}$$

(3) 次のように政策を更新する

$$\pi^{k+1}(s) = \arg\max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v^k\left(s'\right) \right), s \in \mathcal{S} \tag{3.44}$$

(4) 全ての $s \in S$ に対して $\pi^{k+1}(s) = \pi^k(s)$ であれば $\pi_*(s) = \pi^k(s)$ を出力して終了する。そうでなければ $k = k + 1$ として (2) に戻る。

作用素 T^π を以下のように定義すると、作用素 T^π は γ 縮小写像であることが示されている。

$$T^\pi(v) = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v \tag{3.45}$$

縮小写像の原理より、T^π は唯一の不動点に収束する。

3.2.3 モンテカルロ学習と TD 学習

前述のマルコフ決定過程における価値反復法及び方策反復法は、状態遷移確率及び報酬関数が既知であることを前提としていた。ここからは、状態遷移確率及び報酬関数を用いず、シミュレーションを通して環境から実際に得られたデータに基づいて学習していく手法を説明する。以降、1 回のシミュレーションは有限期間で終えるものとし、1 回のシミュレーションをエピソードと呼ぶ。

(a) モンテカルロ学習

政策 π のもとでの状態 s における状態価値関数は次のように定義される。

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right] \tag{3.46}$$

モンテカルロ学習では、期待値計算を経験平均で置き換えて計算する。すなわち、状態 s から方策 π に従って行動した時、$(R_2, R_3, R_4, ...)$ の報酬を得たとすると、その時の収益は式 3.4 により計算される。これを複数回施行し、単純平均を計算して状態価値関数 $v_\pi(s)$ を推定する手法である。オンラインで更新する場合には、実際に得られた G_t を用いて以下のように状態価値の推定値を更新する。ここで、α は学習率を表す。

$$V\left(S_t\right) \leftarrow V\left(S_t\right) + \alpha \left(G_t - V\left(S_t\right) \right) \tag{3.47}$$

(b) TD 学習

モンテカルロ学習では、毎回エピソードの終わりまで計算しなければ更新ができない手法であり、環境から得られた実際の収益 G_t に近づけるように価値関数を更新していた。一方で TD 学習では、TD 誤差を用いてエピソードの終わりを待たずに学習する手法であり、R_{t+1} を環境から取得した場合に、推定される収益 $R_{t+1} + \gamma V\left(S_{t+1}\right)$ に近づくように価値関数を更新する。

$$V\left(S_t\right) \leftarrow V\left(S_t\right) + \alpha \left(R_{t+1} + \gamma V\left(S_{t+1}\right) - V\left(S_t\right) \right) \tag{3.48}$$

ここで、$R_{t+1} + \gamma V\left(S_{t+1}\right)$ を TD ターゲットと呼び、現在の価値関数の推定値との差分 $\delta_t = R_{t+1} + \gamma V\left(S_{t+1}\right) - V\left(S_t\right)$ を TD 誤差と呼ぶ。

モンテカルロ学習で用いる収益 G_t は真の $v_\pi(S_t)$ の不偏推定量であるが、TD 学習で用いる TD ターゲット $R_{t+1} + \gamma V\left(S_{t+1}\right)$ は $v_\pi(S_t)$ の不偏推定量ではない。一方で、収益

G_t は TD ターゲットに比べると高分散である。以上から、モンテカルロ学習は不偏だが高分散、TD 学習は低分散だがバイアスが大きいという特徴がある。

(c)　これまでのまとめ

図 3.1 に動的計画法を用いた価値反復法・政策反復法、モンテカルロ学習、及び、TD 学習による価値関数の推定手法を図示している。またそれぞれの価値関数の更新式を以下にまとめて記す。

- 動的計画法

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1}) \right] \tag{3.49}$$

- モンテカルロ学習

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t) \right) \tag{3.50}$$

- TD 学習

$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right) \tag{3.51}$$

(d)　n ステップ TD

ここで、モンテカルロ学習と TD 学習の中間的な学習法を導入する。TD 学習では、環境から得られた R_{t+1} を用いて期待される収益 $R_{t+1} + \gamma V(S_{t+1})$ に近づけて学習していた。n 期先まで実際にある政策に従った結果 $R_{t+1}, R_{t+2}, ..., R_{t+n}$ が得られたとして、n ステップ収益を次のように定義する。

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n}) \tag{3.52}$$

n ステップ収益を用いる n ステップ TD 学習は次のように表らわされる。

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right) \tag{3.53}$$

ここで n をエピソードの最終時点と設定した場合がモンテカルロ学習となる。

3.2.4　Q 学習

モンテカルロ学習も TD 学習も、ある政策 π を固定し、その政策のもとでの状態価値関数を環境から得られるデータを元に推定する手法であった。ここでは、より良い政策を求める手法について説明する。

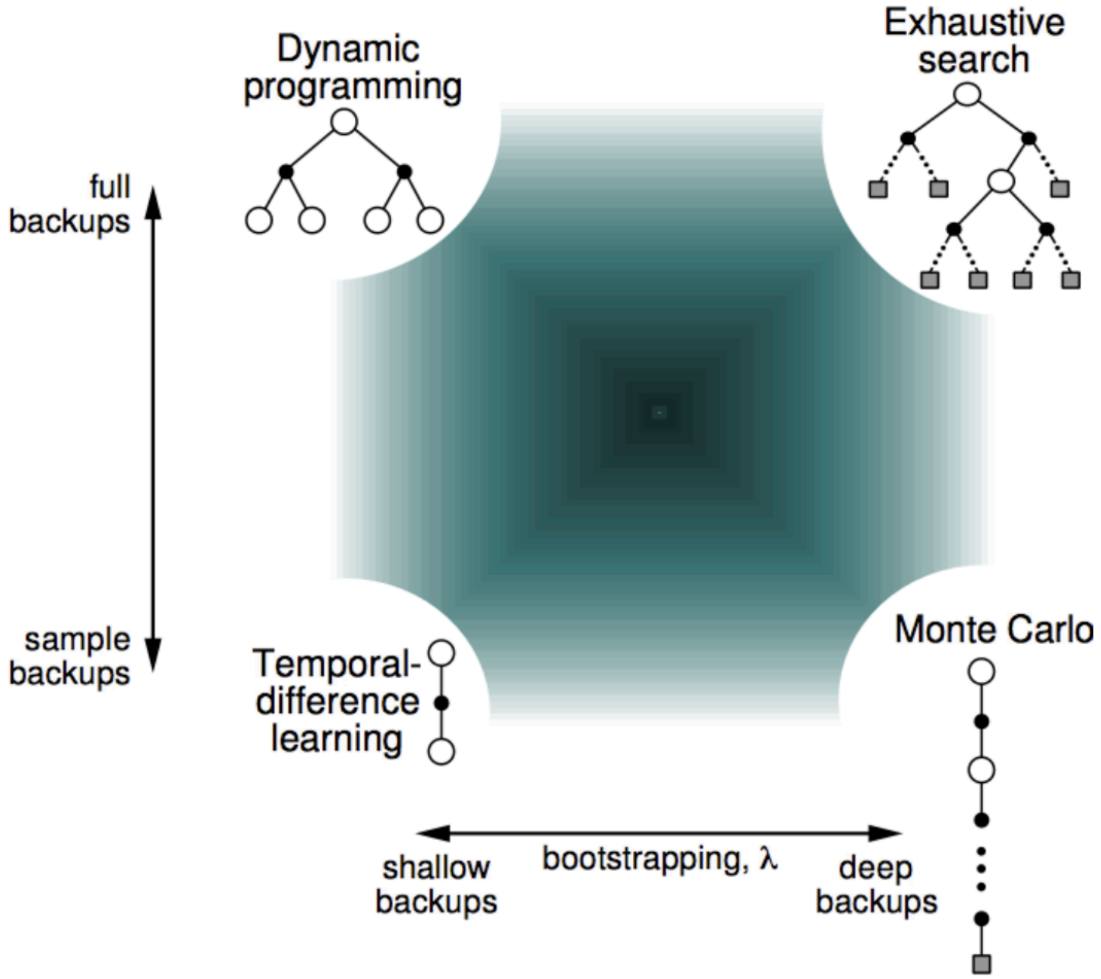

図 3.1　価値関数推定手法の整理（文献 [1] より抜粋）

政策反復法で行われていたように、状態価値に基づく貪欲な政策改善では、環境のモデル、すなわち状態遷移確率と報酬関数を必要としている。

$$\pi'(s) = \underset{a \in \mathcal{A}}{\mathrm{argmax}} \mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s') \tag{3.54}$$

一方で、状態行動価値 $Q(s,a)$ に基づく貪欲な政策改善では環境のモデルは必要としない。

$$\pi'(s) = \underset{a \in \mathcal{A}}{\mathrm{argmax}} Q(s,a) \tag{3.55}$$

Q 学習 [41, 42] は、状態行動価値に関する最適値方程式に基づき、最適状態行動価値を学習する方策オフ型の手法である。方策オフ型とは、シミュレーションにおいて次の行動

を選択する政策（これをビヘイビア政策 μ とする）と、学習対象となる政策（これをターゲット政策 π とする）が異なることを意味する。

ビヘイビア政策 μ の例として、ϵ 貪欲探索を説明する。ϵ 貪欲探索は探索と利用のトレードオフに対するシンプルな政策であり、次のように確率 $1-\epsilon$ で貪欲な行動選択、確率 ϵ でランダムな行動選択をとる。

$$\pi(a \mid s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a^* = \underset{a \in \mathcal{A}}{\text{argmax}} Q(s,a) \\ \epsilon/m & \text{otherwise} \end{cases} \tag{3.56}$$

ϵ 貪欲探索以外によく使われる手法として、ボルツマン選択がある。ボルツマン選択では、現在の行動価値に基づき、決定 $a \in \mathcal{A}$ を選択する確率を次のように定める。

$$\pi(a \mid s) = \frac{\exp(Q(s,a)/\tau)}{\sum_{a' \in \mathcal{A}} \exp(Q(s,a)/\tau)} \tag{3.57}$$

τ は温度パラメータであり、τ が高いほどランダムな決定の選択に近づき、τ が小さいほど行動価値の大きな決定が選択されるようになる。学習の初期では τ を高くしておいて、徐々に下げていくことが多い。

Q 学習では、次のように状態行動価値を更新する。

$$Q\left(S_t, A_t\right) \leftarrow Q\left(S_t, A_t\right) + \alpha_t \left(R_{t+1} + \gamma Q\left(S_{t+1}, A'\right) - Q\left(S_t, A_t\right)\right) \tag{3.58}$$

ここで、A' は環境において実際にとった行動ではなく、状態行動価値に関して貪欲なターゲット政策、すなわち $\pi\left(S_{t+1}\right) = \underset{a'}{\text{argmax}} Q\left(S_{t+1}, a'\right)$ により決定される。よって、次式で表される式展開より、

$$\begin{aligned} & R_{t+1} + \gamma Q\left(S_{t+1}, A'\right) \\ =& R_{t+1} + \gamma Q\left(S_{t+1}, \underset{a'}{\text{argmax}} Q\left(S_{t+1}, a'\right)\right) \\ =& R_{t+1} + \max_{a'} \gamma Q\left(S_{t+1}, a'\right) \end{aligned} \tag{3.59}$$

Q 学習は次のように表される。

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha_t \left(R_{t+1} + \gamma \max_{a'} Q\left(S_{t+1}, a'\right) - Q(S_t, A_t)\right) \tag{3.60}$$

α_t は学習率であり、現在の推定値とこれまでの推定値の加重平均のパラメータを意味する。Q 学習が最適状態行動価値に収束するには、この学習率は十分にゆっくりと小さくしていくことが必要である [42–45]。

ここで、$R_{t+1} + \max_{a'} \gamma Q\left(S_{t+1}, a'\right)$ は状態行動価値についての TD 誤差であるが、これを状態価値についての n ステップ TD と同様に、n ステップ収益を用いることが可能である。すなわち、政策 μ に基づき環境から $(R_{t+1}, R_{t+2}, ..., R_{t+n})$ を得たとして、以下を TD ターゲットとする Q 学習を n ステップ Q 学習と呼ぶ。

$$R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma \max_{a'} Q\left(S_{t+n}, a'\right) \tag{3.61}$$

以降、TD ターゲットとして $R_{t+1} + \max_{a'} \gamma Q\left(S_{t+1}, a'\right)$ をとる場合を、TD(0) と表記する。

(a)　関数近似 Q 学習

これまでの手法は、状態価値関数 $v(s)$ 及び状態行動価値関数 $q(s, a)$ は、表形式で表現されていた。一方で、実際には状態空間、決定空間双方、離散的だとしても全て表現しようとすると極めて大きい、もしくはそもそも連続値もあり得る。そこで、価値関数を関数近似する手法が提案されてきた。ここでは、離散的な行動空間において、各行動 $a \in \mathcal{A}$ について 状態行動価値関数 $q_\pi(s, a)$ を以下のようにパラメータ $\boldsymbol{w}$ を持つ関数により近似する関数近似 Q 学習について説明する。

$$\hat{q}(s, a, \boldsymbol{w}) \approx q_\pi(s, a) \tag{3.62}$$

ここで、微分可能な関数 $J(\boldsymbol{w})$ を考え、$J(\boldsymbol{w})$ の勾配を以下のように表記する。

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \begin{pmatrix} \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_1} \\ \vdots \\ \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_n} \end{pmatrix} \tag{3.63}$$

勾配法では α を学習率として、以下のように $J(w)$ が小さくなる方向にパラメータ $\boldsymbol{w}$ を更新する。

$$\Delta \mathbf{w} = -\frac{1}{2} \alpha \nabla_{\mathbf{w}} J(\mathbf{w}) \tag{3.64}$$

状態行動価値関数 $q_\pi(s, a)$ をパラメータ $\boldsymbol{w}$ をもつ微分可能な関数 $\hat{q}(s, a, \boldsymbol{w})$ で近似するとき、関数近似二乗誤差は次のように表される。

$$J(\mathbf{w}) = \mathbb{E}_\pi \left[\left(q_\pi(S, A) - \hat{q}(S, A, \mathbf{w}) \right)^2 \right] \tag{3.65}$$

上の二乗誤差を小さくする確率的勾配法は以下のように表される。

$$\begin{aligned} -\frac{1}{2} \nabla_{\mathbf{w}} J(\mathbf{w}) &= \left(q_\pi(S, A) - \hat{q}(S, A, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{q}(S, A, \mathbf{w}) \\ \Delta \mathbf{w} &= \alpha \left(q_\pi(S, A) - \hat{q}(S, A, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{q}(S, A, \mathbf{w}) \end{aligned} \tag{3.66}$$

真の状態行動価値関数 $q_\pi(s,a)$ は未知であるので、今までと同様に TD ターゲットにより代替する。TD(0) では、以下に従ってパラメータ $\boldsymbol{w}$ を更新する。

$$\Delta \mathbf{w} = \alpha \left(R_{t+1} + \gamma \hat{q} \left(S_{t+1}, A_{t+1}, \mathbf{w} \right) - \hat{q} \left(S_t, A_t, \mathbf{w} \right) \right) \nabla_{\mathbf{w}} \hat{q} \left(S_t, A_t, \mathbf{w} \right) \tag{3.67}$$

上記方法では、サンプル効率が低いため、バッチ型の学習手法が提案されてきた。バッチ型の関数近似 Q 学習では、経験メモリにこれまでの状態、行動、報酬、次の状態を保持しておき、経験メモリ上のデータを用いて二乗誤差が小さくなるように学習する。

$$\mathcal{D} = \{(s_1, a_1, r_2, s_2), (s_2, a_2, r_3, s_3), ...\} \tag{3.68}$$

関数近似にニューラルネットを用いた経験メモリによるバッチ型の Q 学習は、Riedmiller [46] に提案され、Neural Fitted Q と呼ばれている。

しかし、経験メモリ上のデータは時系列の相関が強く、独立同分布からのサンプルとみなせないこと、及び TD ターゲットが推定対象である状態行動価値関数 $\hat{q}(s,a,\boldsymbol{w})$ に依存してしまうために学習が安定しないという問題があった。この問題に対し、経験メモリからのランダムサンプリングと、TD ターゲットを求める状態価値関数と更新対象の状態価値関数を分離するという工夫が加えられた手法が提案され、Deep Q-Network [47] と呼ばれている。

$$\mathcal{L}\left(w\right) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\left(r + \gamma \max_{a'} Q\left(s', a'; \boldsymbol{w}^-\right) - Q\left(s, a; \boldsymbol{w}\right) \right)^2 \right], \tag{3.69}$$

ここで、$\boldsymbol{w}^-$ は TD ターゲットを求める状態価値関数（ターゲット Q）のパラメータである。ターゲット Q のパラメータ $\boldsymbol{w}^-$ が一定間隔で $\boldsymbol{w}$ と同期される。

本手法の登場により、状態空間が大きいコンピュータゲーム [47–50] やロボット制御 [51, 52] の問題に対しても強化学習が良い成績を上げるようになった。

3.2.5 強化学習における行動空間の分類

本項では、行動空間の観点から問題設定を 3 つに分類し、それぞれの分類における強化学習手法を説明する。

1 つ目は 1 次元の離散行動空間である。代表的な例としては Atari 2600 のテレビゲームが挙げられる。ALE(Arcade Learning Environment) [2] は Atari 2600 を操作するエージェントを評価するためのプラットフォームとして開発され、多くの強化学習アルゴリズムが ALE を用いて評価されてきた。人がゲームするときと同様に、ゲームの画像をインプットとする設定において状態空間は多次元である一方、行動空間は最大で 18（ジョ

イスティックの操作とボタンの押下に対応）に限定される。そのため、前項で説明した関数近似 Q 学習が適用可能であり、関数近似 Q 学習の成功事例の多くはこの分類に該当する。

2 つ目は高次元連続行動空間である。代表的な例としては 2 次元もしくは 3 次元上の運動タスクが挙げられる。ベンチマーク環境として、物理演算エンジンソフトウェアである MuJoCo が用いられることが多い。連続行動空間においては、離散行動空間を前提とする Q 学習は適用できず、REINFORCE [54] に代表される方策勾配法、もしくは Q 学習と方策勾配を組み合わせた Actor-Critic [55, 56] による手法が提案されてきた [57–59]。

3 つ目は高次元離散行動空間であり、同時補充問題はこの分類に該当する。高次元離散行動空間に対しても近年いくつかの解法が提案されており、次項で説明する。

3.2.6　高次元離散行動空間に対する強化学習

本項では、高次元離散行動空間に対する強化学習手法を説明する。

Durac-Arnold ら [2] は、DDPG（Deep Deterministic Policy Gradient）[57] をベースとした手法を提案している。連続行動空間上で学習し、本来の離散行動空間から、得られた連続値行動ベクトルに近いものを選択するという手法である。図 3.2 に概要を示す。

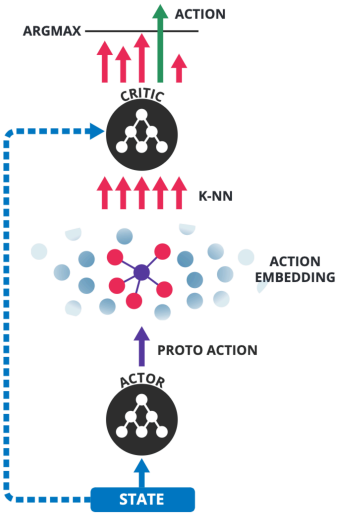

Figure 1. Wolpertinger Architecture

図 3.2　DDPG をベースとした離散行動空間での学習手法 (文献 [2]) より抜粋）

DDPG と同様に、 Actor はパラメータ θ^{π} を持つニューラルネットにより表現され、状

態 $\boldsymbol{s}$ を入力として、以下のプロトアクション $\hat{\mathbf{a}}$ を出力する。

$$\hat{\mathbf{a}} = f_{\theta^\pi}(\mathbf{s}) \tag{3.70}$$

ここで、プロトアクション $\hat{\mathbf{a}}$ は連続値ベクトルであるので、本来の離散行動空間から $\hat{\mathbf{a}}$ 近傍の k 個を選択し（この集合を $\mathcal{A}_k$ とする）、$\mathcal{A}_k$ の中から最も Q 値の大きい行動ベクトルを選択する。

$$\mathbf{a} = \arg\max_{\mathbf{a}_j \in \mathcal{A}_k} Q_{\theta^Q}(\mathbf{s}, \mathbf{a}_j) \tag{3.71}$$

ここで、Q_{θ^Q} はパラメータ θ^Q を持つニューラルネットワークにより表現された Critic である。しかし、連続行動空間上で表現される行動ベクトルから上記の方法で得られる離散行動空間での行動ベクトルは、最適な行動との差異が大きくなることが指摘されている [60]。

Metz ら [3] は、N 次元行動空間の MDP を、行動を 1 次元ずつ状態に加えて複数の環境に分割した Sequential DQN（SDQN）というモデルにおいて、1 次元ずつ逐次的に行動を決めていく手法を提案している。図 3.3 に 3 次元行動空間に対する SDQN のイメージを示す。時点 t において、k 番目の子 MDP は、$k-1$ 番目までに決定された一連の行動 $\boldsymbol{a}_t^{1:k-1}$ を状態の一部として k 番目の行動を出力する。これにより、各子 MDP は 1 次元の行動のみを扱うこととなり、本来 N 次元行動空間の問題を N 個の 1 次元行動空間の問題に変換している。

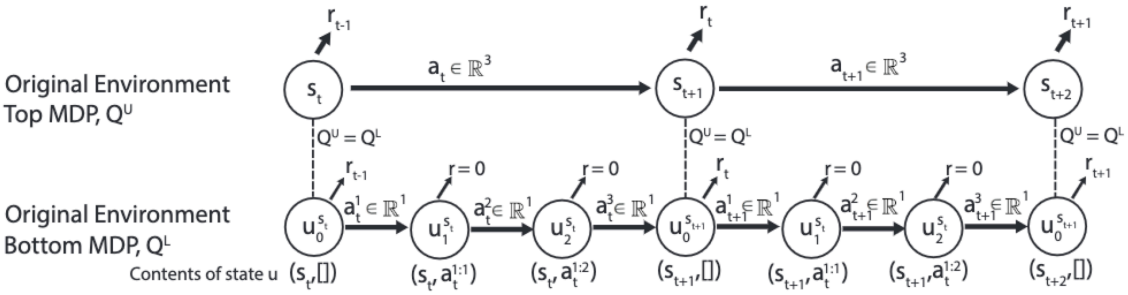

図 3.3　Sequential DQN (文献 [3]) より抜粋）

本手法はその特性から、逐次的、もしくは階層的な行動決定問題に対して有効であると想定される。階層的な行動決定問題である StarCraft ii [61] に適用され、他の様々な手法と組み合わされることで、人間のトップパフォーマーを超える性能を達成している [62]。

Tavakoli ら [7] は、行動の次元ごとに独立したエージェントとする、マルチエージェントの問題としての解法を提案している。エージェントごとに独立して行動決定させつつ、

協調的な方策を学習させるために、ニューラルネットワークで表現した行動価値の下層部分のパラメータをエージェント間で共有し、同一の状態価値を与えている。本論文では、高次元離散行動空間に対処する強化学習手法として、マルチエージェントでの定式化を採用しており、本手法を含めた詳細な説明は次節以降で行う。

3.3　マルチエージェント強化学習

ここから、環境に複数のエージェントが存在する、マルチエージェントの設定を考える。マルチエージェントの設定では、マルコフゲームとして環境が表現される。n エージェント下のマルコフゲームは、$< \mathcal{S}, \mathcal{A}_1, ..., \mathcal{A}_n, \mathcal{P}, \mathcal{R}_1, ..., \mathcal{R}_n, \gamma >$ で表現される。ここで、

- $\mathcal{A}_i$: エージェント i の決定空間
- $\mathcal{P}$: $\mathcal{S} \times \mathcal{A}_1 \times ... \times \mathcal{A}_n \to \Pi(\mathcal{S})$ は状態遷移確率
- $\mathcal{R}_i$: $\mathcal{S} \times \mathcal{A}_1 \times ... \times \mathcal{A}_n \to \mathbb{R}$ はエージェント i の報酬関数

である。

一般にマルチエージェント強化学習においては、全体としての期待割引報酬和の最大化を目的とする協調ゲームか、エージェントそれぞれが独自の期待割引報酬和の最大化を目的とする敵対ゲームかによって分類される。以降、協調ゲームを前提とする。

協調ゲームにおけるマルチエージェント強化学習は、大きく Team Learning と Concurrent Learning に大別される [63]。Team Learning では、全エージェントの行動を $\boldsymbol{a} = (a_1, a_2, ..., a_n)$ とすると、状態が観測可能である場合にはシングルエージェントの場合と全く同一の手法により $Q(\boldsymbol{s}, \boldsymbol{a})$ を学習することが可能である。しかし、決定空間がエージェントの数に対して指数的に増加するという問題がある。

一方で、複数のエージェントが同時並行的に学習していく手法が Concurrent Learning である。Concurrent Learning では、一般に Centralized Learning with Decentralized Execution、すなわち学習時は全ての情報が利用可能だが、実行時（行動決定時）には各エージェントは個々の観測のみ利用可能で独立に行動を決定する、という前提が置かれる。そのため、Concurrent Learning では、独立に行動を決定することから、どのようにエージェント間で協調的な方策を学習できるのかが課題とされてきた [63]。

Concurrent Learning の中で古くからあり、シンプルかつ成果を上げてきた手法に Independent Q-learning（IQL） [64] が挙げられる。IQL では、各エージェントは他のエージェントの存在を無視して環境の一部として取り扱い、次のようにエージェントごと

に独立した状態行動価値を保持する。

$$Q_i\left(s, a_i\right) = \left(1-\alpha\right) Q_i\left(s, a_i\right) + \alpha\left(r_i + \gamma \max_{a_i} Q\left(s', a_i\right)\right) \tag{3.72}$$

i はエージェントを表すインデックスであり、添字 i の存在のみがシングルエージェントの Q 学習との差異である。IQL では、他のエージェントを環境の一部として取り扱うことにより、他エージェントの行動による影響が把握できず、各エージェントがそれぞれ同時に学習する中で、各エージェントの環境が非定常になることが指摘されている。

これまで、Team Learning と IQL は複数の問題設定において比較されてきた。IQL には前述の問題点があるものの、他のエージェントの存在を考慮しない IQL が常に不利というわけではないことが知られている。Jansen ら [65] は、子問題への分割が可能である場合には IQL が望ましいと指摘している。もしエージェントごとに独立した問題への分割が可能な場合には、Team Learning の場合と比べて探索しなければいけない決定空間が圧倒的に減るからである。

しかし、IQL、及び、一般に Concurrent Learning では、同時並行で学習することによりいくつかの問題が発生する。以降、Concurrent Learning の問題を報酬分配 [66]、環境の非定常性、経験メモリ利用、の 3 つに分類して、それぞれに対する先行研究を説明する。

(a)　報酬分配

各エージェントがそれぞれ独立して学習するマルチエージェント環境においては、得られた報酬をどのように各エージェントに分配するか、という報酬分配を決める必要がある。

最もシンプルな報酬分配としてはグローバル報酬分配があり、これは得られた報酬を均等に各エージェントに分配する方法である [63]。一般に、グローバル報酬分配では、各エージェントは自分の行動に対して十分なフィードバックが得られないため、難しい問題に対してはうまく学習できないと報告されている [67]。

もう一つの分配方針がローカル報酬分配である。ローカル報酬分配では、各エージェントの行動に基づいて得られた報酬を分配するやり方である。この場合、直接報酬に紐づく行動以外については評価されないため、他のエージェントを助けるような行動に対してインセンティブを与えない構造となる。そのため、ローカル報酬分配の場合は比較的早く学習は進むが、必ずしも収束した結果がグローバル報酬分配よりも良いとは限らない [63, 68, 69]。

報酬分配に対する手法として、Difference Rewards [67, 70] が挙げられる。ある状態 s でジョイントアクション $\boldsymbol{u}$ をとった際に、グローバルな報酬 r を受け取ったとする。

Difference Rewards とは、その時のエージェント k への報酬分配 $r_k(s, \boldsymbol{u})$ を決定する際に、他のエージェントの行動は固定して、自分があるデフォルトの行動 c^k をとった時に得られたであろう報酬との差分として次のように定義される。

$$r_k(s, \boldsymbol{u}) = r(s, \boldsymbol{u}) - r\left(s, \left(\mathbf{u}^{-k}, c^k\right)\right) \tag{3.73}$$

ここで、$\mathbf{u}^{-k}$ はエージェント k 以外の行動である。しかし、この手法では、デフォルトの行動 c^k を設計者が決める必要があり、その設定は一般に簡単ではないことが指摘されている [4]。

COMA（Counterfactual Multi-Agent Policy Gradients）[4] は関数近似を用いた Actor-Critic ベースの手法であり、Difference Rewards のコンセプトを継承しつつも前述のデフォルトの行動設定の問題を回避している。図 3.4 に COMA の概要を図示する。

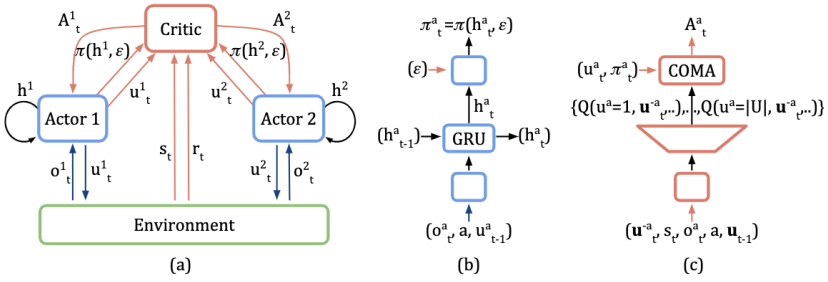

Figure 1: In (a), information flow between the decentralised actors, the environment and the centralised critic in COMA; red arrows and components are only required during centralised learning. In (b) and (c), architectures of the actor and critic.

図 3.4　COMA（文献 [4] より抜粋）

COMA では、全エージェントのジョイントアクション $\mathbf{u}$ で条件つけられた中央集権的な Critic を学習し、エージェントごとのローカルな観測のみを入力とする Actor により行動が決定されることが特徴である。報酬分配については、明示的にデフォルトの行動をとった場合との差異を計算するのではなく、次のように他のエージェントの行動 $\mathbf{u}^{-k}$ を固定した上で、自分の行動で周辺化した Q 値との差分をアドバンテージとして学習に用いている。

$$A^a(s, \mathbf{u}) = Q(s, \mathbf{u}) - \sum_{u'^a} \pi^a\left(u^a \mid \tau^a\right) Q\left(s, \left(\mathbf{u}^{-a}, u'^a\right)\right) \tag{3.74}$$

一方で、グローバルな報酬のみを用い、エージェントごとの Q 値の分解を内部的に計算する手法も提案されている [5, 6]。

VDN（Value Decomposition Network）[5] は、グローバルな Q 値 Q_{tot} を各エージェントのローカルな観察 h と決定 a のみで条件つけられるエージェントごとの Q_i の足しあ

わせにより近似表現している。

$$Q_{tot}\left(\left(h^1, h^2, \ldots, h^d\right), \left(a^1, a^2, \ldots, a^d\right)\right) \approx \sum_{i=1}^{d} \tilde{Q}_i\left(h^i, a^i\right) \tag{3.75}$$

図 3.5 に VDN の概要を図示する。

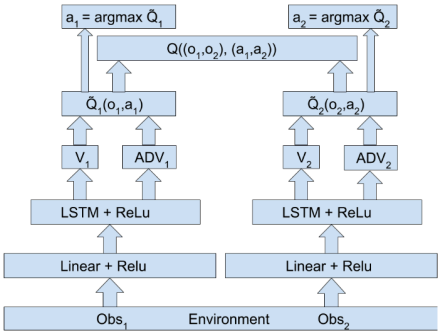

Figure 15: Value-Decomposition Individual Architecture

図 3.5　VDN（文献 [5] より抜粋）

各エージェントは、それぞれ $\arg\max Q_i$ により独立に行動を決定し、環境から得られるグローバルな報酬をもとにニューラルネットワークのパラメータを更新する。

VDN では、単純な足しあわせにより Q_{tot} を表現するために、その表現力には制約が多く、また行動決定時には利用できないが学習時に利用できる情報を活用し切れていないという課題がある。それに対し、エージェントごとの Q 値に対して非線形な処理を通じて Q_{tot} を表現する QMIX [6,8] が提案されている。

QMIX では、Q_{tot} 各エージェントのローカルな Q を最大にする行動が Q_{tot} の最大化になることを保証するために以下の制約をおいている。

$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \forall a \tag{3.76}$$

図 3.6 に QMIX の概要を図示する。上記制約を満たすために、Q_{tot} を生成する Mixing Network のパラメータが全て正になるように hypernetwork により生成されている。

(b)　環境の非定常性

IQL では、他のエージェントの選択した行動を考慮せずに学習するために、通常の行動価値の更新ルールでは最適なジョイントアクションが学習されないことが起きうる。例として次のような設定を考える。

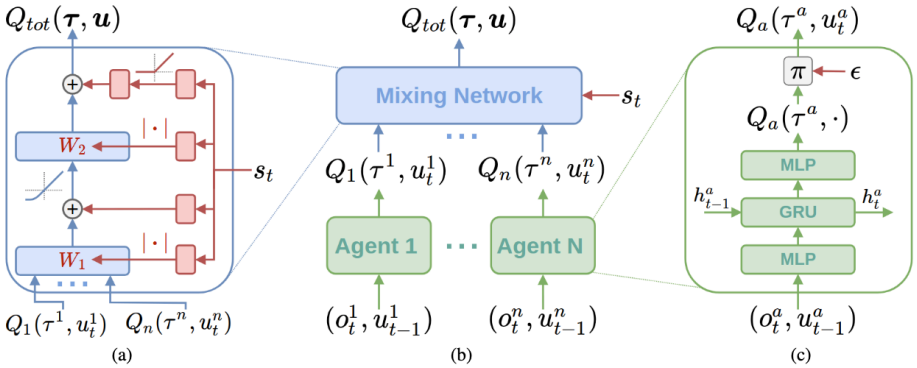

Figure 2. (a) Mixing network structure. In red are the hypernetworks that produce the weights and biases for mixing network layers shown in blue. (b) The overall QMIX architecture. (c) Agent network structure. Best viewed in colour.

図 3.6　QMIX（文献 [6] より抜粋）

ある状態に k 回訪問し、あるエージェントが毎回 a_1 の行動を選択したとする。その時に得られた報酬が、$r_1, ..., r_k$ だったとした場合に、r_i の変動は、報酬関数の内在的な変動だけでなく、他のエージェントがどのような行動をとったかにも左右される。この状況で Q 学習すると、状態 s で行動 a_1 をとった場合に得られる即時報酬の推定値としては、得られた報酬の平均値が用いられることになる。他のエージェントがある特定の行動をとった時のみ大きな報酬が得られるような問題設定においては、このように平均値を用いて行動価値を更新していると、最適なジョイントアクションを求めることは困難である。

これに対する手法として、Hysteretic Q [71] と Lenient Q [72] が提案された。

Hysteretic Q では、以下のように TD 誤差の正負に応じて異なる学習率を適用する。数値実験 [71] では、$\alpha = 0.1, \beta = 0.01$ が適用されており、TD 誤差が負の場合には学習率を相対的に低く設定することで、楽観的に行動価値を更新する。

$$\begin{aligned} \delta &\leftarrow r + \gamma \max_{a'} Q_i(s', a') - Q_i(s, a_i) \\ Q_i(s, a_i) &\leftarrow \begin{cases} Q_i(s, a_i) + \alpha\delta \text{ if } \delta \geq 0 \\ Q_i(s, a_i) + \beta\delta \text{ else} \end{cases} \end{aligned} \tag{3.77}$$

一方で、Lenient-Q 学習では、平均ではなく最大値を用いて行動価値を更新する。環境から取得した即時報酬に基づく行動価値が現在の行動価値よりも大きい場合、すなわち $Q(s, a)$ が増加する場合には常に更新し、一方で減少する場合には温度パラメータで表現される条件を満たす場合のみ更新する。これにより、ある状態 s において適切なジョイントアクションが選ばれた場合をより重視して行動価値が更新されることになる。

(c)　経験メモリの利用

IQL においても、シングルエージェントにおける関数近似 Q 学習と同様に行動価値関数の関数近似が可能である。しかし、関数近似で一般的に用いられる経験メモリによるバッチ型の学習では、IQL における非定常な環境の問題はより深刻となる。各エージェントの経験メモリに蓄積されたデータはその時点での他のエージェントの行動価値に依存しており、すなわち現在の環境のダイナミクスを表すものではなくなるためである。

Lenient-Q、及び、Hysteretic-Q 共に、関数近似を用いた場合においても拡張されている。

Lenient-DQN (LDQN) [73] では、Lenient-Q のコンセプトを深層ニューラルネットで関数近似した場合に拡張しているが、経験メモリを活用するために、各経験に対して学習に使うかどうかを判定する条件とする leniency 値を保持している。

Hysteretic-Q を RNN(Recurrent Neural Network) を用いた関数近似 Q 学習に拡張した Hysteretic Deep Recurrent Q-Networks (HDRQNs) [74] では、Hysteretic-Q と同様に TD 誤差の正負に応じた二つの学習率が導入されており、数値実験では、α/β が 0.1 から 0.6 の範囲では協調的な方策の学習に成功しており、特に 0.4 から 0.5 の設定が最も良い結果が得られたと報告されている。

3.3.1　高次元離散行動空間に対するマルチエージェント強化学習

本来シングルエージェントの問題を決定空間の問題に対処するためにマルチエージェントとして扱う研究結果は複数報告されている [7, 75]。

Branching Dueling Q-Network（BDQ） [7] は、複数可動部を持つロボット制御を想定して提案された。図 3.7 に BDQ のアーキテクチャを示す。

可動部が複数ある場合、それぞれの可動部の決定空間が $\mathcal{A}_i$ だとした時に、全体としてのジョイントアクションの決定空間の大きさは、$\prod_i |\mathcal{A}_i|$ となり、可動部の数に応じて指数的に増加する。決定空間の問題に対処しながらも可動部間で協調的な方策を学習することを意図し、ニューラルネットワークを用いた関数近似 Q 学習エージェントを、下層部分に共通層に続いて n 次元の枝分かれした形で表現することで、全体として行動数はロボットの可動部の数 n に対して線形に増加するアーキテクチャが提案されている。これは、下層のニューラルネットワークのパラメータがエージェント間で共有される構造を持つ特殊な IQL とみなすことができる。MuJoCo のベンチマークタスクにおける数値実験では、共通部分を持たない Independent Dueling Q-Network（IDQ）では可動部の

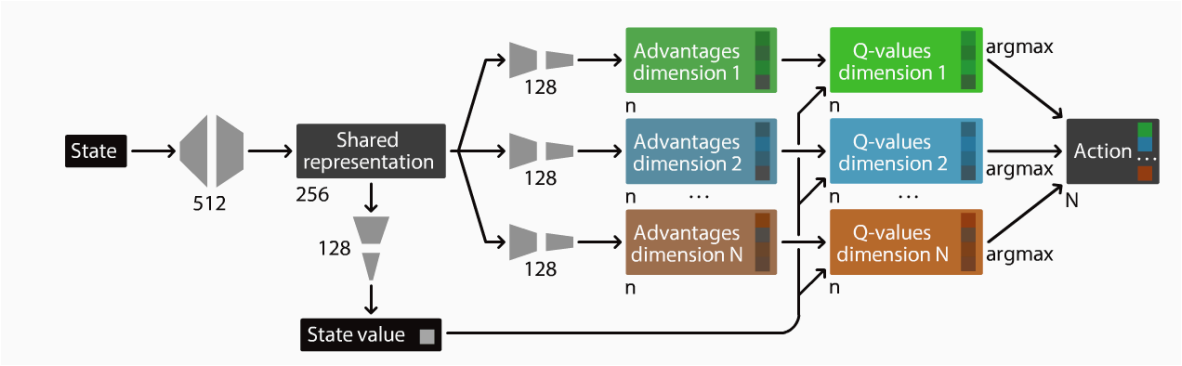

図 3.7　BDQ (文献 [7]) より抜粋)

数が増えるにつれ協調的な行動が学習できないのに対し、BDQ では学習に成功しており、DDPG [57] などの強化学習手法より良いパフォーマンスが得られたことが報告されている。

BDQ では、グローバルな報酬のみが得られることが想定されており、TD ターゲットおよび損失関数は次のように定義されている。

$$y_d = r + \gamma \frac{1}{N} \sum_d Q_d^- \left(s', \arg\max_{a'_d \in \mathcal{A}_d} Q_d\left(s', a'_d\right) \right), \tag{3.78}$$

$$L = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[\frac{1}{N} \sum_d \left(y_d - Q_d\left(s, a_d\right) \right)^2 \right] \tag{3.79}$$

ここで、$d \in \{1, ..., N\}$ は可動部（枝）のインデックス、$|\mathcal{A}_d| = n$ は各枝の決定空間、y_d は枝 d の TD ターゲット、$a_d \in \mathcal{A}_d$ は枝 d の行動、Q_d は状態 s で行動 a_d をとった場合の枝 d の行動価値、Q_d^- はターゲットネットワーク、$\mathcal{D}$ は経験メモリ、a はジョイントアクション $(a_1, a_2, ..., a_N)$ である。数値実験において、TD ターゲットの設定として、式 3.78 の方が以下で表されるナイーブな設定よりも良い結果が得られている。

$$y_d = r + \gamma Q_d^- \left(s', \arg\max_{a'_d \in \mathcal{A}_d} Q_d\left(s', a'_d\right) \right) \tag{3.80}$$

また、以下のように Dueling Network [76] を導入しており、ある状態 s の状態価値について、全ての枝が同一の評価を持つような設定となっている。

$$Q_d\left(s, a_d\right) = V(s) + \left(A_d\left(s, a_d\right) - \frac{1}{n} \sum_{a'_d \in \mathcal{A}_d} A_d\left(s, a'_d\right) \right) \tag{3.81}$$

3.3.2　サプライチェーン関連分野における強化学習の適用例

最後に、サプライチェーンにおける強化学習の適用例について説明する。これまで同時補充問題に対して強化学習による既存研究はおそらく存在していないが、サプライチェーンマネジメントという広い枠組みにおいてはいくつも例がある [75,77–81]。このうち、関数近似を用いた強化学習による研究についていくつか説明する。

Gabel ら [75] は、ジョブショップスケジューリングと呼ばれる工場内の工程計画の作成にマルチエージェント強化学習を適用している。工場における各リソースを独立したエージェントとして、Neural Fitted Q による関数近似 Q 学習が用いられている。

Kemmer ら [81] は、1 工場、複数倉庫のサプライチェーンにおいて、工場での生産数量と各倉庫への輸送数量の決定を強化学習により求めている。ルールベースでの方策をベンチマークとして、関数近似強化学習による方策の方が総費用の観点で良いパフォーマンスが得られている。

また、多層サプライチェーンの典型的な問題設定であるビールゲームを題材とした研究も複数存在する。Oroojlooyjadid ら [79] は、サプライチェーンにおけるエンティティの一つを学習対象として、他のエンティティの方策を基在庫方策など一般的な方策に固定して学習させている。一方で、Fuji ら [80] は、ビールゲームにおいて登場する全てのエンティティ（工場や卸、小売）を学習対象として、それぞれの補充方策をマルチエージェント強化学習により求めている。ここでは、マルチエージェント強化学習の非定常な環境に対処するために、学習プロセスにおいてあるエージェント以外の方策を固定する、という工夫がされている。

3.4 問題設定

本論文での問題設定を説明する。確率的な定常需要のもと、海外倉庫から国内倉庫への補充を想定した多品目在庫システムを考え、国内倉庫における在庫保管費用、欠品費用、輸送費用の合計の最小化を目的とする。以下の表記を用いる。

i：商品インデックス, $i = 1, ..., N$
t：時点, $t = 1, ..., T$
LT：補充リードタイム (週)
l_i：商品 i の補充ロットサイズ (パレット)
$d_{i,t}$：商品 i の時点 t での需要数 (パレット)
$x_{i,t}$:商品 i の時点 t での補充数 (パレット)
$Out_{i,t}$：国内倉庫から顧客への商品 i 時点 t での出荷数 (パレット)
$In_{i,t}$：海外倉庫からの商品 i 時点 t の補充入荷数 (パレット)
$I_{i,t}$：国内倉庫の商品 i 時点 t での手元在庫数 (パレット)
$I^p_{i,t}$：国内倉庫の商品 i 時点 t での在庫位置 (パレット)
$u_{i,t}$：商品 i 時点 t での国内倉庫での欠品数 (パレット)

ここで、欠品時は機会損失とする. 時刻 t に補充された商品は、時刻 $t + 1$ 以降に利用可能であるとする。本論文では、上流倉庫での在庫欠品や、納品遅れはないものと想定する。そのため、手元在庫量、補充量、出荷量、需要量、欠品数量は以下の関係が成り立つ。ここで、在庫位置とは手元在庫量に既に補充をかけたが未入荷の数量を加えた数量を表す。

$$In_{i,t+LT_i} = x_{i,t} \tag{3.82}$$
$$Out_{i,t} = \min(d_{i,t}, I_{i,t}) \tag{3.83}$$
$$I_{i,t+1} = I_{i,t} - Out_{i,t} + In_{i,t} \tag{3.84}$$
$$I^p_{i,t+1} = I^p_{i,t} - Out_{i,t} + x_{i,t} \tag{3.85}$$
$$u_{i,t} = d_{i,t} - Out_{i,t} \tag{3.86}$$

補充数はパレット単位を想定し、補充ロットサイズの整数倍とする。最終顧客からの需要はパレット単位ではなく、バラ単位を想定するため小数を想定する。本論文では、補充

リードタイム（補充指図から入荷までに必要な期間）は、日本と東南アジア諸国との一般的なコンテナ船輸送の現状を踏まえ、3 週間とする。

3.4.1 取引条件

C^{trans}, C^{hold}, C^{pel} をそれぞれ輸送、在庫保管、欠品費用とし、それぞれの単価を U^{trans}, U^{hold}, U^{pel} とする。欠品費用は、欠品数に比例する形で以下のように定義される。

$$C_{i,t}^{\text{pel}} = U^{\text{pel}} \times u_{i,t} \tag{3.87}$$

輸送および在庫保管費用については、いくつかの取引条件を考える。1 つ目は先行研究でもよく想定されている基本取引条件であり、共同発注費用である輸送費用は数量に依らず固定値、在庫保管費用は手元在庫数に比例するものとした。ここで、輸送数についても在庫数についても制約はないものとする。

$$C_t^{\text{trans}} = U^{\text{trans}} \ (\text{if} \sum_i x_{i,t} > 0) \tag{3.88}$$

$$C_{i,t}^{\text{hold}} = U^{\text{hold}} \times I_{i,t} \tag{3.89}$$

2 つ目は輸送制約を導入する。すなわち、輸送容量 $\text{CAP}_{\text{trans}}$ を設定し、1 回の輸送量はこの輸送容量を超えてはいけない ($\sum_i x_{i,t} \leq \text{CAP}_{\text{trans}}$) という制約を設ける。

3 つ目は階段状の輸送費用、すなわち輸送費用は、積載率に関わらず必要なコンテナ本数に比例するとした。これはコンテナ船輸送やトラックなど本数・台数単位で輸送費用が決定されるケースに該当する。

$$C_t^{\text{trans}} = U^{\text{trans}} \times \lceil \frac{\sum_i x_{i,t}}{\text{CAP}_{\text{trans}}} \rceil \tag{3.90}$$

4 つ目は非線型な在庫保管費用の導入である。ここでは、ある一定の契約容量 (CAP_{wh}) までは在庫量に関わらず固定的な費用が発生し、その容量を超えた分については、割高な単価で容量に応じた在庫保管費用が発生する設定を考える。

$$C_t^{\text{hold}} = U^{\text{hold}_{\text{f}}} + U^{\text{hold}_{\text{v}}} \times \max(0, \sum_i I_{i,t} - \text{CAP}_{\text{wh}}) \tag{3.91}$$

ここで、$U^{\text{hold}_{\text{f}}} = U^{\text{hold}} \times \text{CAP}_{\text{wh}}$、及び、$U^{\text{hold}_{\text{v}}} > U^{\text{hold}}$ とする。

5 つ目として、3 つ目と 4 つ目の組み合わせ、すなわち、階段状の輸送費用、かつ、非線形な在庫保管費用の設定を考える。

表 3.1 が上述の取引条件の設定を図示したものであり、図 3.8 はその視覚的な表現である。

表 3.1　本論文で検証する取引条件

取引条件	輸送費用	在庫保管費用
1) 基本	固定	線形
2) 輸送制約	固定（制約あり）	線形
3) 階段状の輸送費用	階段状	線形
4) 非線型在庫保管費用	固定	非線形
5) 階段状輸送 + 非線型在庫	階段状	非線形

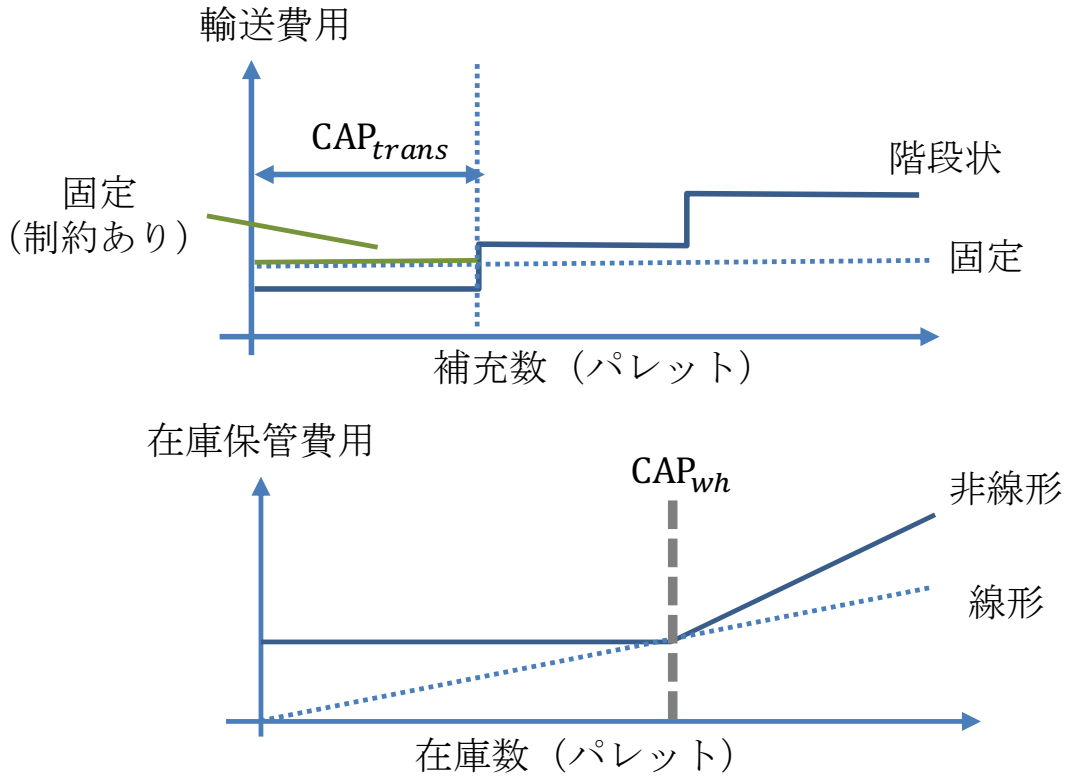

図 3.8　輸送費用および在庫保管費用の設定

3.5　提案手法

3.5.1　従来の方法と強化学習手法の比較

マルコフ決定過程における解法は、状態遷移関数 $p(s'|s,a)$ や報酬関数 $r(s,a)$ を用いる価値反復法や方策反復法と、状態遷移関数や報酬関数を必要としない強化学習に大別され

る。状態遷移関数 $p(s'|s,a)$ と報酬関数 $r(s,a)$ が既知だとしても、状態空間が大きい場合には次元の呪い、状態遷移が複雑な場合にはモデリングの呪いにより、実際には適用が難しいケースも存在する [82, 83]。

同時補充問題においては、需要分布に関する何らかの仮定のもとで、状態遷移関数や報酬関数は既知として扱うことができる。しかし、状態遷移関数や報酬関数が既知だとしても、取引条件によっては、状態遷移及び期待報酬の表現はより複雑になる。

例えば、既存研究では多くの場合、受注した時点で在庫が足りない場合でも、機会損失ではなく受注残となる取引を想定している。欠品を受注残として想定すると、マルコフ決定過程における状態は在庫位置のみで表現でき、状態遷移が単純になるからであるが、実際には欠品時には顧客に待ってもらえずに機会損失となるケースも多々発生する。また、倉庫容量や輸送容量等に何らかの制約を置く場合にも、状態遷移の複雑性は増加する。需要についても、これまで従来研究では連続時間の元で複合ポアソン過程とする場合が多いが、本論文のように離散時間の元で正規分布を仮定した場合には複雑な状態遷移となる。以上のように、基本取引条件に加えて実務における追加的な取引条件の要素の追加や、異なる需要分布を考慮しようとすると、前述のモデリングの呪いの問題は一層深刻になる。

一方で、強化学習である Q 学習では、状態遷移関数 $p(s'|s,a)$ や報酬関数 $r(s,a)$ を用いずに、ある状態で行動を選択したことで環境から得られるスカラー値としての報酬を用い、サンプル平均をとる形で行動価値を学習していく。同時補充問題においては、ある補充方策のもとで時系列での在庫の受払を計算し、補充・在庫・欠品数量に基づいて計算される総費用を報酬として環境から受け取ることで学習を進める。ある補充方策に基づく在庫受払の計算は、取引条件や需要分布が複雑であっても容易にシミュレーションが可能である。

第 2 章で見た通り、発注可能点方策や MP 方策など、少数のパラメータを持つ近似方策に限定したとしても、割引発注機会の近似表現を利用した分割法によるパラメータ導出が一般的であり、需要分布や取引条件に応じた解法が必要となる。一方で、強化学習を用いる場合には、学習に必要なものは実際に行動して得られる報酬のデータのみであり、取引条件が複雑になったとしても、取引条件に従って費用を計算するシミュレータを更新するだけで学習が可能である。

一方で、本提案手法は、近似方策を用いる既存手法に比べて 2 点欠点がある。1 つは、方策の型を限定せず、ニューラルネットワークを用いた関数近似により補充方策が表現されるため、既存の発注可能点方策や MP 方策とは異なり、得られた方策を人間が解釈することは困難であること。もう 1 つは、多様な取引条件に対する適用を目的とする中で、既存手法に対して計算効率では劣ることである。これらについては、前者は在庫管理業務

の自動化が一層進むことで、方策の解釈の容易さの重要性は相対的に低下すると想定できること、後者については自動化コンピュータの計算能力の増加や分散型強化学習手法の進展 [58, 84–87] に鑑み、将来的に大きな問題にはならないと考えた。

3.5.2　強化学習適用の課題

同時補充問題に強化学習を用いる上では、高次元離散行動空間への対処が課題となる。一般に高次元離散行動空間でのタスク設定はいくつかあるが、図 3.9 のように分類することができる。

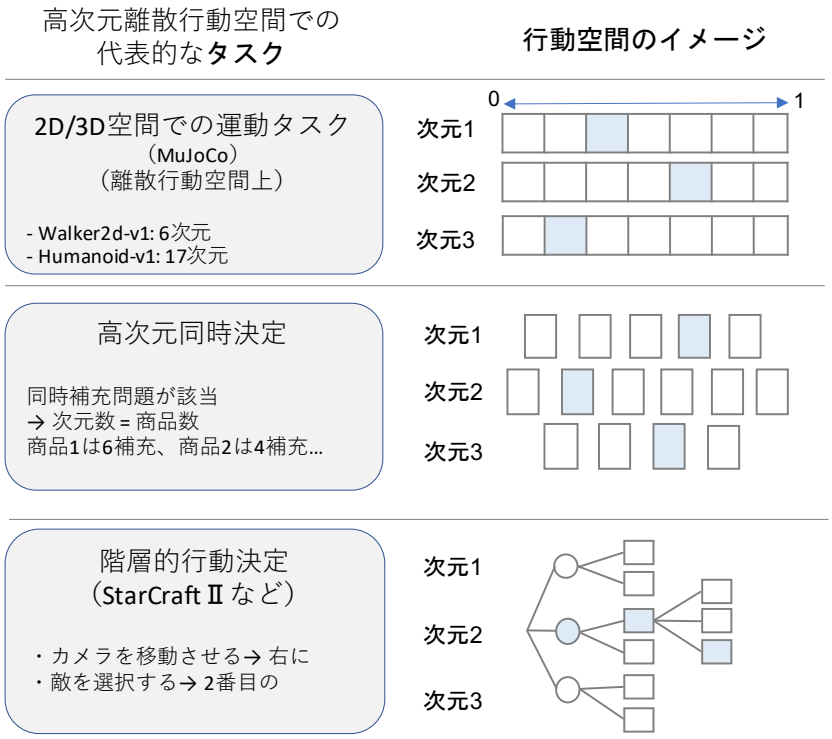

図 3.9　高次元離散行動空間の代表的なタスク

1 つ目は、本来連続空間で定義される行動空間を細かく離散化して離散行動空間の問題に変換する場合である。例えば、運動タスクにおいて、それぞれ連続空間で定義される複数の可動部がある場合に、それぞれの可動部の行動空間を離散化した問題設定が該当する。2 つ目は、複数の離散的な選択肢からなる決定項目が複数存在する場合であり、ここでは高次元の同時決定が求められる。同時補充問題はこの分類に該当する。3 つ目は、逐次的もしくは階層的に行動を決定する場合で、StarCraft ii [61] のように上位の階層で選

択した行動によって詳細の行動の選択肢が定まる場合や、複数時点先までの行動を決定する必要がある場合が該当する。

これらはいずれも次元数に対して指数的に行動空間が増大するという点では同じであるが、その構造は異なることがわかる。3.2.6 項では、高次元離散行動空間に対する強化学習手法を説明した。DDPG をベースにした Durac-Arnold ら [2] の手法は、前述の 1 つ目の分類においては良い結果が報告されているものの、学習された連続空間での行動ベクトルと実際の離散行動空間で得られる行動ベクトルとの不整合が指摘されている [7]。また、高次元の行動について 1 つずつ逐次的に行動を決定する Metz ら [3] の手法では、行動を決定する順番を設計者が固定的に定める必要がある。3 つ目の分類である階層的行動決定の問題では行動空間の特性から問題とならないが、商品間の補充数の相互作用が大きい同時補充問題においては、ある固定的な順序で行動を常に決定していくことで、局所解に陥ることが懸念される。

以上から、本論文では高次元離散行動空間を持つ同時補充問題の特徴を踏まえ、各商品を独立したエージェントとするマルチエージェント強化学習手法を提案する。

3.5.3　マルチエージェント強化学習の適用

同時補充問題に対してマルチエージェント強化学習を適用する場合、以下 2 点の両立が求められる。

(1) 商品間での協調的な補充の実現

(2) 商品数に応じて指数的に大きくなる決定空間への対処

まず、Team Learning の適用を考える。本問題設定においては、各商品の在庫位置と手元在庫を状態とすると、将来の費用は過去の状態には依存せず現在の状態にのみ依存して決定される。各時刻において、エージェントは全商品の手元在庫と在庫位置を観測、補充量を決定し、輸送、在庫保管、欠品の合計費用のマイナス 1 倍を即時報酬 r_{t+1} として受け取る。

この時、前述の通り Q 学習は以下のように定義される。

$$Q\left(s_t, \boldsymbol{a}_t\right) = \left(1-\alpha_t\right) Q\left(s_t, a_t\right) + \alpha_t \left(r_{t+1} + \gamma \max_a Q\left(s_{t+1}, a\right)\right) \tag{3.92}$$

ここで、本問題設定においては、$\boldsymbol{a} = (a_1, a_2, ..., a_n) \in \mathcal{A}$ であり、これをジョイントアクションと呼ぶ。ここで、商品 i の行動を $a_i \in \mathcal{A}_i$ とすると、$|\mathcal{A}| = \prod |\mathcal{A}_i|$ となる。仮に、商品ごとに 6 つの離散的な行動の選択肢があった場合、商品数が 10 の場合 $|\mathcal{A}| = 6^{10} > 60,000,000$ となる。Team Learning では、決定空間が商品数に対して指数

的に増加するため学習の収束は期待できない。

次に、Concurrent Learning の中で最もシンプルなアプローチである IQL の適用を考える。IQL では、各エージェントは他のエージェントの行動を環境の一部として次のように学習する。

$$Q_i\left(s_{i,t}, a_{i,t}\right) = \left(1-\alpha_t\right) Q_i\left(s_{i,t}, a_{i,t}\right) + \alpha_t \left(r_{i,t+1} + \gamma \max_{a_i} Q\left(s_{i,t+1}, a_i\right) \right) \tag{3.93}$$

ここで、式 3.92 との差異は添字 i の存在のみであり、ここで $r_{i,t+1}$ は何らかの報酬分配に従って商品 i に分配された報酬である。しかし、IQL では、一般に他のエージェントの行動を考慮せずに独立して行動を決定するため、協調的な学習が困難であることが報告されている。

「商品間での協調的な補充の実現」と「商品数に応じて指数的に大きくなる決定空間への対処」の 2 点の両立に向け、報酬分配に着目したマルチエージェント強化学習手法を提案する。

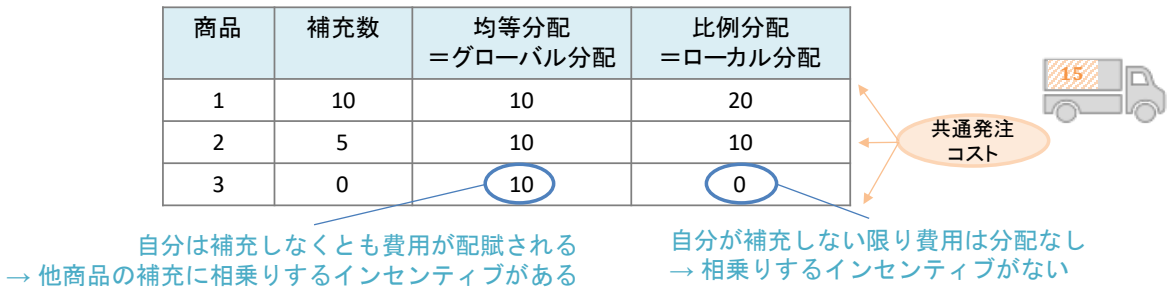

図 3.10　報酬分配の例

図 3.10 は、合計 15 の補充で 30 の共通発注費用がかかる例における報酬分配の様子を示している。グローバル分配では、補充数に依らず均等に分配されるため、30 の費用が各商品に 10 ずつ分配されている。一方で、ローカル分配では、補充数に比例する形で分配されるため、補充数が 0 である商品 3 に対して共通発注費用は分配されない。

グローバル分配では、補充数によらず費用が分配されるため、まだ輸送容量に空きがある場合、商品 3 からすると、商品 1,2 の補充を固定した場合に追加費用なく補充が可能であるため、在庫量から考えてすぐの補充が必要でなくともこのタイミングで補充をかけるインセンティブが存在する。一方で、ローカル分配では、自分が補充しない限り費用の分配はないため、どんなに輸送容量に空きがあったとしてもその補充に「相乗り」をするインセンティブが存在しないことがわかる。

このように、報酬分配は、マルチエージェント強化学習において、各エージェントが学習する補充方策に大きな影響を与える。同時補充問題では、商品間の補充数（行動）の相互作用が大きい、すなわち、商品ごとの適切な補充数は他商品の補充数に大きく依存することが、マルチエージェント強化学習で協調的な方策を学習することをより困難にする。

報酬分配に関しては、3.3 項で先行研究を説明した。これまで提案されてきた VDN [5] や QMIX [6,8] では、内部的に分解するエージェントごとの Q 値の表現に制約があり、商品間での補充数に強い相互作用のある同時補充問題には適さないと考えられる。

実際に、商品数が 2 で輸送容量が 20 パレットである階段上の輸送費用を持つ同時補充問題を考える。輸送費用はコンテナ数に比例するため、各商品の補充数に対する行動価値は、コンテナの積載率が高くなる行動において高くなる、図 3.5.3 のような形状になることが想定される。しかし、単調性を制約とする QMIX で表現できる関数は図 3.5.3 のような形状であり、図 3.5.3 のような価値関数を表現することはできない [88]。なお、VDN は QMIX により強い制約を置いたものであるため、QMIX で表現できない関数は VDN でも表現することはできない。

商品1の補充数 \ 商品2の補充数	0	10
0	5	-5
10	-5	10

図 3.11　階段上の輸送費用の同時補充問題での行動価値

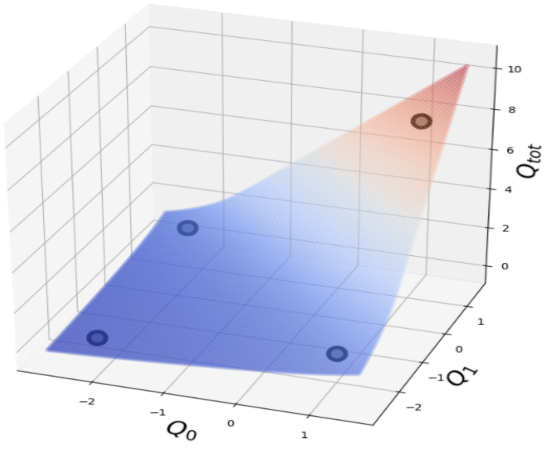

図 3.12　QMIX で表現可能な関数例（文献 [8] より抜粋

また COMA [4] では、Critic はジョイントアクションで条件つけられているものの、行動を決定する Actor はローカルな観測のみに基づいて行動を決定する。COMA では、この Actor を用いて報酬分配が内部的に計算されているため、同様に商品間での補充数に強い相互作用のある本問題設定には適さない。

一方で、本論文では、マルチエージェント強化学習で一般的な Centralized Learning with Decentralized Execution の設定ではなく、あくまで高次元離散行動空間に対処するためのマルチエージェントの設定であるため Centralized Learning with Centralized Execution の設定である。すなわち、行動を決定する際に、グローバルな状態や各エージェントの方策は全て観測可能であり、かつ中央集権的な行動決定が可能である。

そこで、グローバルな状態観測が可能であることを活かした BQL（Branching Q-Learner）と、中央集権的な行動を決定する C-IQL（Conditional Independent Q-Learner）の 2 つの手法を提案する。

3.5.4　提案手法 1: BQL

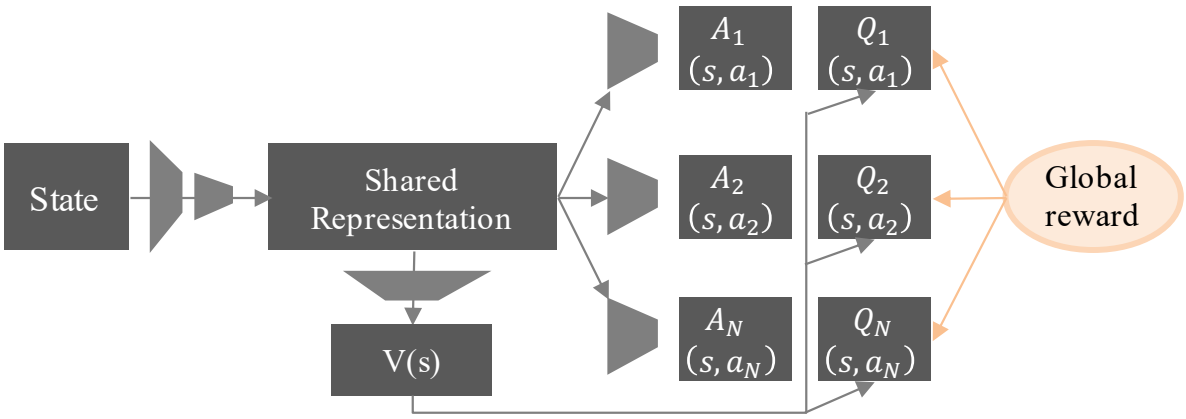

図 3.13　BQL（Branching Q-Learner）

ここでは、協調的な行動の学習を目的として、IQL に対し、Q 値のニューラルネットワークによる関数近似においてネットワークの下層をエージェント間で共有する BQL（Branching Q-Learner）を説明する。図 3.13 に提案手法の概要を示す。

BQL は、複数の可動部を持つロボット制御向けに提案された BDQN [7] をもとにしている。BDQN では、各可動部を独立した意思決定者としているが、本提案手法では、各商品を独立した意思決定者とした定式化を試みた。しかし、グローバルなスカラー報酬のみが得られるロボットの制御と異なり、補充方策の場合には、線形な在庫保管費用を想定した場合の在庫保管費や欠品費用については商品ごとに分解することが可能であり、また輸送費用に関しても、各商品の補充量の情報をもとに、何からしらの分配ロジックにより商品ごとに報酬を分配することが可能である。そこで、BDQN に対し報酬分配関数を導

入した。

(a)　報酬分配によるエージェントの振る舞いの違い

本手法では、結果的に既存の近似方策には劣った（数値実験結果は後述）が、次に示す手法（C-IQL）を考案する上で重要な示唆が得られた。報酬分配の設定によるエージェントの振る舞いの違いである。同時補充問題では、前述の通り、他の商品と強調して補充タイミングを合わせることで共通発注費用の低減が可能である。ここで、協調的な行動として以下の 2 つの行動が必要である。

- 協調的な補充行動：他商品が補充するなら自分も合わせて補充
- 協調的な延期行動：自分しか補充しないなら次回に見送り

このうち、報酬分配としてグローバルな分配とローカルな分配を双方設定したが、どちらの報酬分配においても、協調的な補充行動もしくは延期行動のいずれかが学習できないことがわかった。

表 3.2、及び、図 3.14 に、10 商品数における BQL での学習における設定と、学習した補充方策の結果を示す。

表 3.2　商品別の設定と補充結果

商品	1	2	3	4	5	6	7	8	9	10
週平均需要	0.3	0.4	0.5	0.5	0.7	0.9	1.0	1.0	1.2	1.2
補充ロット	1.0	1.0	1.0	1.0	2.0	2.0	3.0	3.0	3.0	3.0
平均補充数（グローバル報酬分配）	1.0	1.0	1.0	2.7	2.0	2.0	3.0	3.0	3.0	3.0
平均補充数（ローカル報酬分配）	1.8	1.8	2.0	1.4	2.3	4.7	3.0	3.1	5.1	3.5

ここでは、輸送容量が 20 パレットの階段状の輸送費用を設定しており、各週の需要は全商品合計で輸送容量よりも小さい設定としている。そのため、商品間で補充タイミングを合わせて 1 回の補充が 20 パレットになるように調整するのが輸送費用の観点からは望ましい。

図 3.14 に示す全商品合計の補充数と在庫数の時系列推移から見て分かる通り、グローバルな報酬分配では、毎週補充が発生しており、20 パレットの輸送容量に対して 5 パレット未満の非常に積載率の低い補充が行われていることがわかる。対してローカルな報酬分配では、全商品の合計補充量が 0 である週が多く存在している。すなわち、「協調的な延期行動」はグローバルな報酬分配では学習できていない。

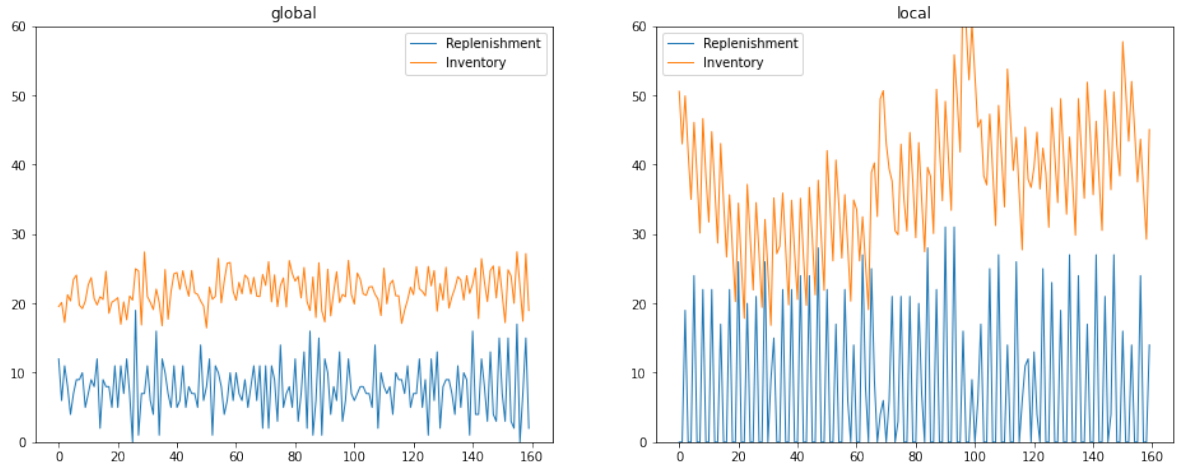

図 3.14　報酬分配の設定により学習される補充方策の特徴（時系列推移例）

左: グローバル報酬分配、右: ローカル報酬分配

一方で、グローバルな報酬分配の方がローカルな報酬分配に比べ、在庫水準が低く抑えられていることがわかる。この在庫水準の違いは、商品別の発注数に表れている。表 3.2 に示すとおり、グローバルな報酬分配では、各商品の平均補充数は補充ロットに一致しているが、ローカルな報酬分配では、補充ロットよりも大きな値になっている。これは、ローカルな報酬分配においては、商品間で協調して 20 パレットにおける積載率を高めるのではなく、各商品の補充数を多くすることで積載率を高めていることを意味し、「協調的な補充行動」が学習できていない。

上記の報酬分配によるエージェントの振る舞いの違いは、報酬分配の設定から図 3.15 で示すように説明される。

以上から、ローカル報酬分配もグローバル報酬分配も一長一短であり、エージェント間でパラメータを共通にしても、報酬分配設定のみによりマルチエージェント強化学習で協調的な補充方策を学習することは困難であることを示した。

3.5.5　提案手法 2: C-IQL

数値実験結果に示す通り、BQL では商品数が少ない場合には総費用の観点で近似方策同等であったが、商品数が増えるに従い近似方策に劣る結果となった。そこで、他商品の補充量を明に考慮する必要があると考え、IQL のように商品ごとに独立したエージェントとしつつ、自商品以外の合計補充量を状態の一部とした Q 関数を推定するように定式化

報酬分配	協調的な補充行動 （増やす方向）	協調的な延期行動 （減らす方向）
グローバル （均等分配）	○ 自分以外の商品で補充がかかる場合コストが配賦されるため， 相乗りするインセンティブがある	× 発注量に関わらずコストが配賦されるため， 補充を止めるインセンティブがない
ローカル （比例分配）	× 発注しなければコストは0なので， 自分がすぐに必要でなければ他商品に合わせて補充する必要がない	○ 最後に残った商品は全てのコストが課されるため，延期するインセンティブがある

図 3.15　報酬分配による学習される補充方策の違い

した。本手法を C-IQL（Conditional Independent Q-Learner）と呼ぶ。

図 3.16 に提案手法を示す。各エージェントが 1 商品を構成しており、商品 i のエージェントの状態行動価値関数はパラメータ θ_i を持つニューラルネットにより表現される。提案手法 1 との一番の差異は、各エージェントの状態行動価値関数において、他商品の合計補充量を状態の一部としていることである。これにより、提案手法 1 では各商品の状態価値は他商品の補充量に依存せず独立に表現されていたのに対して、本手法では合計補充量という形で条件つけられた Q 値を学習する。また、全商品の補充量を決定する中央行動決定機構を設けた。アルゴリズム 1 は本提案手法の概要を疑似コードで示している。大きな流れは通常の IQL と変わらないが、ジョイントアクションの選択、共同 ϵ 貪欲探索、報酬分配が本手法の特徴であり、これらは以下で説明する。

在庫保管費用が在庫量に比例する場合 (取引条件 1,2,3 の場合)、他商品の合計補充量以外では、各エージェント i の状態は、商品 i の在庫と在庫の位置のみとなる。したがって、状態は、 $s_{i,t} = (s^-_{i,t}, \sum_{j\neq i} x_{j,t})$ で定義される。ここで、 $s^-_{i,t} = (I_{i,t}, I^p_{i,t})$ であり、$x_{j,t} = l_j a_{j,t}$ は、エージェント j が行動 a_j を選択したときの補充量である。非線形な在庫保管費用（取引条件 4,5 の場合）の場合には、合計在庫量 $\sum_k I_{k,t}$ も状態として加える。

各時点で、状態 $s_{i,t}$ にある各エージェントは、行動 $a_{i,t}$ (ジョイントアクション選択機構で決定される) をとり、分配された即時報酬 $r_{i,t+1}$ (報酬分配の設定に従う) を受け取り、次の状態 $s_{i,t+1}$ に遷移する。無限の行動空間を考えることは非現実的であり、また、需要に対してあまりに大きな数量を補充することはサプライチェーンの観点から非現実的であるため、商品 i の補充可能量を $X_i = \{l_i a_i \mid a_i \in \mathcal{A}_i = \{0,1,2,3,4,5\}\}$ に限定した。

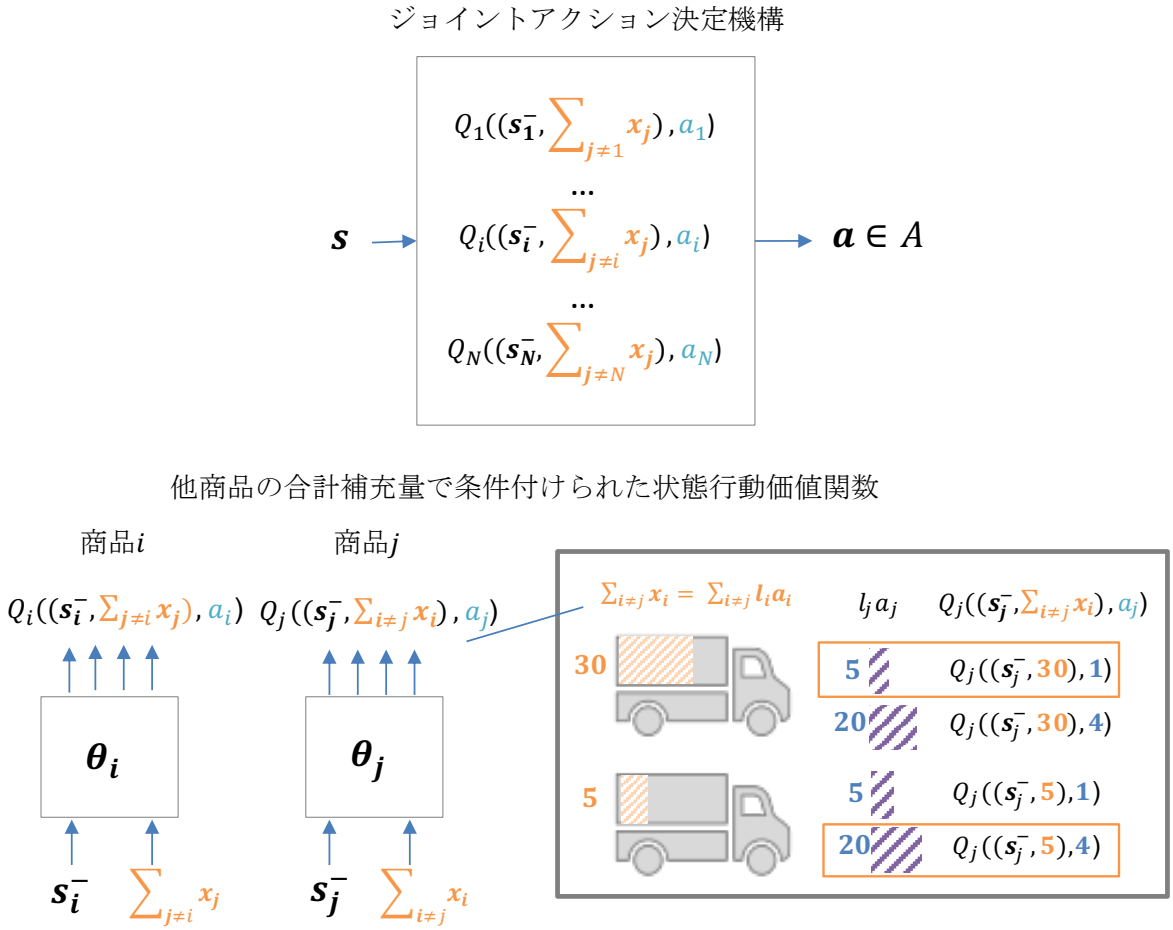

図 3.16　C-IQL（Conditional Independent Q-Learner）

ここで、$\mathcal{A}_i$ は、商品 i の決定空間を表す。この場合、すべての商品のジョイントアクションの決定空間の大きさは、6^n となる。

関数近似を用いた各エージェントの状態行動価値は、以下の損失に応じて更新される。

$$L_i = \mathbb{E}_{\left(s_i, a_i, r_i, s_i'\right) \sim \mathcal{D}_i} \left[L_\delta \left(y_i, Q_i \left(s_i, a_i \right) \right) \right] \tag{3.94}$$

ここで、

$$y_i = r_i + \gamma Q_i^- (s_i', \arg\max_{a_i} Q_i(s_i', a_i)) \tag{3.95}$$

L_δ はフーバー損失関数であり、Q^- はターゲットネットワークである。

Algorithm 1 Learning procedure

Require: K_e, K_s, Freq_{target}

Initialize:

$Q_i, \hat{Q}_i$ for all i

for episode $= 1, \cdots, K_e$ **do**

Initialize:

$\boldsymbol{s}_1, \boldsymbol{a}_1$

for $t = 1, \cdots, K_s$ **do**

if random $\in [0, 1] < \epsilon$ **then**: ▷ ϵ-greedy exploration

$\boldsymbol{a}_t \leftarrow$ select action at random

$\boldsymbol{s}_t \leftarrow$ update states based on the randomly selected actions

$r_{t+1} \leftarrow$ simulate inventory movement based on (states, actions)

$[r_{i,t+1}]_{i=1}^{N} \leftarrow$ allocate reward r_{t+1} ▷ credit assignment

$\boldsymbol{s}_{t+1}, \boldsymbol{a}_t \leftarrow$ action selection in central unit ▷ joint action selection

Memory $\mathcal{D}_i \leftarrow (s_{i,t}, a_{i,t}, s_{i,t+1}, r_{i,t+1})$ for all i

update parameters of Q_i for all i

if episode $= 0 \pmod{\mathrm{Freq}_{target}}$ **then**:

$\hat{Q}_i \leftarrow Q_i$ for all i

(a) 報酬分配

環境には複数のエージェントが存在するため、時間 t における負の輸送、在庫保管、機会損失費用の和が $\sum_i r_{i,t}$ を満たすような報酬分配を決定する必要がある。在庫保管費用が式 3.89 のように在庫量に線形な設定（取引条件 1,2,3）においては、以下のように、グローバル報酬配分により輸送費用 C_t^{trans} を商品ごとに均等に配分し、在庫保管費用は商品ごとに独立して計算した。

$$r_{i,t+1} = -(C_t^{\mathrm{trans}}/N + C_{i,t}^{\mathrm{hold}} + C_{i,t}^{\mathrm{pel}}) \tag{3.96}$$

一方で、在庫保管費用が式 3.90 のように在庫量に非線形な設定（取引条件 4,5）の場合には、次のように、在庫保管費用の固定費用の部分はグローバルな報酬配分、変動費用の

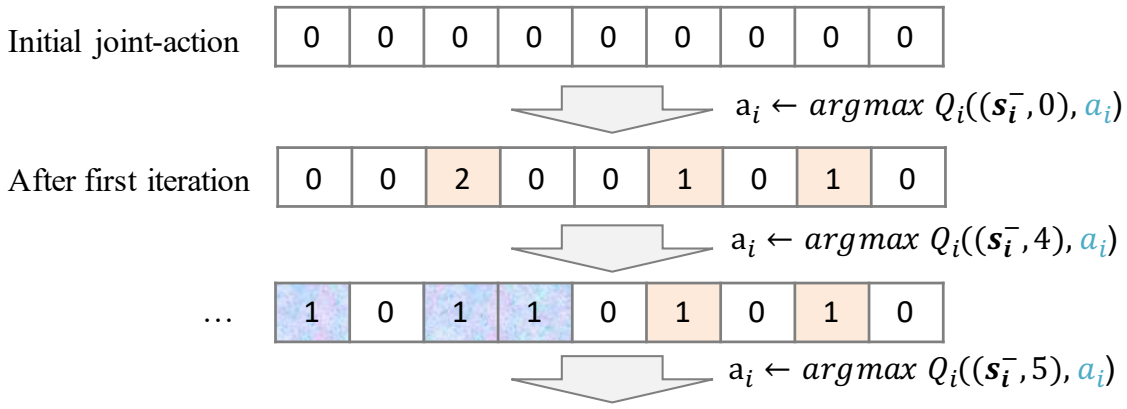

図 3.17　ジョイントアクション選択手順

部分はローカルな報酬配分とした。

$$r_{i,t+1} = -(C_t^{\text{trans}}/N + U^{\text{hold}_\text{f}}/N + C_t^{\text{hold}_\text{f}} \times \frac{I_{i,t}}{\sum_i I_{i,t}} + C_{i,t}^{\text{pel}}) \tag{3.97}$$

ここで、$C_t^{\text{hold}_\text{f}} = C_t^{\text{hold}} - U^{\text{hold}_\text{f}}$ である。

(b)　共同 ϵ 貪欲探索

探索は基本的には ϵ 貪欲法を用いる。ただし、共同補充を促すため、確率 ϵ で全てのエージェントが同一タイミングでランダムに補充量を決定する共同 ϵ 貪欲探索を用いた。また、各エージェントはそれぞれ独立に、確率 ϵ/N でランダムな意思決定をとることとした。

(c)　ジョイントアクションの選択

各エージェントに対して分配した報酬の合計値はその時点での全体の報酬に一致するように分配しているため、ある時点での最適な意思決定とは、$\sum_i Q_i$ が最大になるようなジョイントアクションを選択することであり、以下のように表される。

$$\boldsymbol{a} = (a_0, a_1, ...a_n) = \arg\max_{\boldsymbol{a} \in A} \sum_i Q_i((s_i^-, \sum_{j \neq i} x_j), a_i) \tag{3.98}$$

しかし、このままでは依然として広い行動空間からジョイントアクションを選択するという行動空間の大きさの問題が解決できていない。学習過程では、この最適化問題をステップごとに解く必要があり、式 3.98 の近似解を効率的に求める必要がある。

ここで、各エージェントの行動価値関数が利用可能であるため、他のエージェントの行動を固定して扱うことで、各 Q_i が最大になるように各エージェントの行動を修正できる。このようにして各エージェントの行動を変更することで、より高い行動価値が得られるように行動空間を効率的に探索することが可能である。このジョイントアクションの決定ヒューリスティックをアルゴリズム 2 に示す。

図 3.17 に概要を示す。初期解 $(0, 0, ..., 0)$ から探索を開始する。各反復において、各エージェントは、他のエージェントの行動が現在の解であると仮定して決定を更新する。現在の解が $(a_0, ..., a_i, ..., a_N)$ であれば、エージェント i にとって他のエージェントの補充量の合計は、$\sum_{j \neq i} l_j a_j$ となる。したがって、エージェント i は、Q_i が最大になるように、状態 $\boldsymbol{s}_i^-$ が与えられたときに、行動を更新する。すなわち、$a_i \leftarrow \arg\max Q_i((\boldsymbol{s}_i^-, \sum_{j \neq i} l_j a_j), a)$ のように行動を更新する。

ここで、報酬分配に応じて学習される方策の特徴を利用している。3.5.4 項に示したとおり、補充量によらず均等の共通発注費用を分配するグローバル報酬配分を用いた場合、学習エージェントは協調的な遅延行動の学習ができなかった。しかし、初期解を全商品について補充なし、すなわち $\boldsymbol{a} = (0, 0, \cdots, 0)$ から始めることで、協調的な遅延行動の機会は常に評価されることとなり、そこを開始点として増やす方向の可能性を探索することで、グローバル報酬分配の弱点を補っている。

グローバル報酬配分では、補充をかけないエージェントに対しても、共通発注費用の負担分は $1/N$ である。そのため、ジョイントアクション探索手順では、一旦、1 つ以上のエージェントが補充を決定したとき、在庫量がそれほど低くないエージェントについても自分の補充数に関係なく共通発注費用が配分されるため、いますぐの補充が必要でない場合にも、他とタイミングとあわせて補充をかける、すなわち協調的な補充行動行動が生まれることにつながる。

ここで、上述のヒューリスティック手順に対して既存方策の観点からの解釈を与える。既存方策の一つである発注可能点方策では、商品ごとに (s, c, S) のパラメータを持つが、そのうち、s, c により以下を決定している。

- (1) 在庫がなく、すぐに補充しなければいけない商品: 在庫 $\leq s$
- (2) 今すぐの補充は必要ないがもうすぐなくなりそうなので便乗したい商品: $s <$ 在庫 $\leq c$

一方で、本提案手法におけるヒューリスティック手順は、以下のような探索であると解釈できる。

Algorithm 2 Joint-action selection

Require: $[s_i]_{i=1}^N$, $[Q_i]_{i=1}^N$, K

function EVALUATE($\boldsymbol{a}$)

$x_i \leftarrow l_i a_i$ for all i

$q_{this} \leftarrow \sum_i Q_i(s_i, \sum_{j \neq i}(x_j), a_i)$

if $(q_{this} > q_{best})$ **then**

$q_{best} \leftarrow q_{this}$

$\boldsymbol{a}_{best} \leftarrow \boldsymbol{a}$

return q_{this}

function SEARCH($\boldsymbol{x}$, sequential:bool, start:integer=1)

for $i = 1, \cdots, N$ (starting from start) **do**

$a_i \leftarrow \arg\max Q_i(s_i, \sum_{j \neq i}(x_j))$

if sequantial **then**:

$x_i \leftarrow l_i a_i$

$\boldsymbol{x} \leftarrow \{l_i a_i\}_{i=1}^N$

return $\boldsymbol{a}$, $\boldsymbol{x}$

Initialize:

$\boldsymbol{a}_{best} \leftarrow (0, 0, \cdots, 0)$

$q_{best} \leftarrow$ evaluate($\boldsymbol{a}_{best}$)

for $k = 1, \cdots, K$ **do**

$\boldsymbol{a}, \boldsymbol{x} \leftarrow$ search($\boldsymbol{x}$, sequential=False) ▷ batch search

$q_{best}^k \leftarrow$ evaluate($\boldsymbol{a}$)

for $i = 1, \cdots, N$ **do**

$\boldsymbol{a}, \boldsymbol{x}^{seq} \leftarrow$ search($\boldsymbol{x}$, sequential=True, start=i) ▷ sequential search

$q^{seq} \leftarrow$ evaluate($\boldsymbol{a}$)

if $(q^{sec} > q_{best}^k)$ **then**:

$q_{best}^k \leftarrow q^{sec}$

$\boldsymbol{x} \leftarrow \boldsymbol{x}^{seq}$

return $\boldsymbol{a}_{best}$

- 上記 (1) に該当する商品を初回の反復で探索
- その条件のもとで、(2) のように「相乗り」する商品を探索

ただし、そのルールは (S, c, s) のように固定的なものではなく、合計の Q 値がより大きくなるような組み合わせを反復的に探索する。

3.6　数値実験

3.6.1　実験設定

本提案手法の本来の目的は、様々な需要特性に対して単一のモデルで良い結果を達成し、そして実務でありうる取引条件に対し、簡易な修正で適用可能な解法を確立することであった。これを確認するために、以下の 2 点を検証項目とする数値実験を行った。

(1) 提案手法は、基本取引条件のもと、様々な需要特性に対して、既存の近似方策同等以上のパフォーマンスが達成できるか？

(2) 提案手法は、追加的な取引条件のもと、既存の近似方策同等以上のパフォーマンスが達成できるか？

上記を確認するため、1) 商品数、2) 取引条件、3) 需要特性（需要の分散および商品間での相関）を要素とした数値実験を行った。商品数については、確率的需要における同時補充問題の数値実験の一般的な設定が先行研究では 2 から 12 までの範囲であることに鑑み、2, 5, 10 の 3 つのケースを検証した。需要特性については、需要の分散の程度を表す c_v と、商品間での相関の度合いを表す ρ の 2 つのパラメータを変化させた。これら 2 つのパラメータは全商品共通である。

需要は平均ベクトル μ、分散共分散行列 Σ とする多次元正規分布に従うとし、次のように時刻 t における需要ベクトル d_t を発生させた。

$$d_t = [d_{i,t}]_{i=1}^{N} \sim N(\mu, \Sigma) \tag{3.99}$$

$$\Sigma_{i,j} = \sigma_i \times \sigma_j \times \rho^{|i-j|} \tag{3.100}$$

ここで、商品 i の需要の分散は $\sigma_i = c_v \times \mu_i$ である。

数値実験における単位期間当たり平均需要ベクトル μ, ロットサイズ、輸送容量および倉庫容量の設定を図 3.3 に示す。階段状輸送費用を想定する取引条件 3 及び 5 については、単位期間当たりの需要量が輸送キャパシティ $\mathrm{CAP}_{\mathrm{trans}}$ よりも大きくするため、他の取引条件に比べて大きな需要を設定している。以上をまとめると、数値実験は以下の要素より構成される。

(1) 商品数 n: 2, 5, 10

(2) 取引条件: 1, 2, 3, 4, 5

(3) 需要特性 c_v: 0.4, 0.6、ρ: -0.5, 0, 0.5

表 3.3 数字実験の設定（商品数、平均需要、ロットサイズ、及び、輸送/倉庫容量）

取引条件	商品数	μ	ロットサイズ	輸送容量	倉庫容量
1	2	[2, 2]	[4, 4]	-	-
2		[2, 2]	[4, 4]	20	-
3		[15, 15]	[10, 10]	20	-
4		[2, 2]	[4, 4]	-	20
5		[15, 15]	[10, 10]	20	30
1	5	[0.3, 0.4, 0.5, 0.5, 0.7]	[1, 1, 1, 1, 2]	-	-
2		[0.3, 0.4, 0.5, 0.5, 0.7]	[1, 1, 1, 1, 2]	20	-
3		[3, 4, 5, 5, 7]	[5, 5, 5, 5, 5]	20	-
4		[0.3, 0.4, 0.5, 0.5, 0.7]	[1, 1, 1, 1, 2]	-	20
5		[3, 4, 5, 5, 7]	[5, 5, 5, 5, 5]	20	30
1	10	[.3, .4, .5, .5, .7, .9, 1., 1., 1.2, 1.2]	[1, 1, 1, 1, 2, 2, 3, 3, 3, 3]	-	-
2		[.3, .4, .5, .5, .7, .9, 1., 1., 1.2, 1.2]	[1, 1, 1, 1, 2, 2, 3, 3, 3, 3]	20	-
3		[1.5, 2, 2.5, 2.5, 3.5, 4.5, 5, 5, 6, 6]	[3, 3, 3, 3, 5, 5, 5, 7, 10, 10]	20	-
4		[.3, .4, .5, .5, .7, .9, 1., 1., 1.2, 1.2]	[1, 1, 1, 1, 2, 2, 3, 3, 3, 3]	-	20
5		[1.5, 2, 2.5, 2.5, 3.5, 4.5, 5, 5, 6, 6]	[3, 3, 3, 3, 5, 5, 5, 7, 10, 10]	20	30

また、費用の単価として、U^{hold}, U^{pel}, U^{trans} は、コンテナ船輸送および日本国内での在庫保管費用を踏まえ、それぞれ 0.02、1.0、1 とした。

3.6.2 ベンチマーク方策

定期的在庫観測システムの同時補充方策として代表的な発注可能点方策および MP 方策を比較対象手法とした。発注可能点方策は、商品ごとに 3 つのパラメータ s、c、S があり、それぞれ発注点、発注可能点、基準在庫点を表し、MP 方策は、発注タイミングの頻度を表すグローバルパラメータ 1 つと、商品ごとの 2 つのパラメータ、発注点 s、基準在庫点 S を持つ。

遺伝的アルゴリズム (GA) を用いて各方策のパラメータを導出した。GA は進化コンピューティングの一部である。GA は、母集団と呼ばれる解の集合（染色体で表される）から始まる。ある母集団から解を取り出し、新しい母集団を形成するために使用し、新しい母集団が古い母集団よりも優れたものになるようにする。新しい解（子孫）を形成する

ために選択された解は、その適合度 (fitness) に応じて選択される。

それぞれの解、すなわちベンチマーク方策のパラメータセットについて 12 回のシミュレーションを行い、その 12 回のシミュレーションの平均的な総費用を適合度関数とした。交叉と突然変異は GA の基本的な 2 つの演算子である。ここで 2 点交叉を用いた。交叉が行われた後、突然変異が行われる。母集団数、交叉確率、突然変異確率、世代数は、それぞれ 50 回 N、0.5 回、0.2 回、100 回である。s, c (発注可能点方策の場合) と S はそれぞれ実数値であり、MP 方策のグローバルパラメータは整数である。バイナリコーディングを用いた実装には、ライブラリ deap [90] を使用した。

発注可能点方策および MP 方策は、どちらも輸送容量を考慮していない。そのため輸送容量の設定のある取引条件 2,3,5 については、輸送容量を考慮するための追加的なロジックを実装した。確率的な需要での同時補充問題で輸送容量考慮した研究 [89] と同様に、輸送容量が設定されている場合には積載率をできるだけ高めるような調整を加えている。

研究 [89] と同様に輸送容量の残容量を考慮し、補充数量の合計が輸送容量を超えていたり、積載率が低い場合に、補充数を調整する手順を実装した（積載率は $\sum_i x_{i,t}/(\lceil \frac{\sum_i x_{i,t}}{\mathrm{CAP}_{\mathrm{trans}}} \rceil \times \mathrm{CAP}_{\mathrm{trans}})$ で定義される)。なお、容量制約のある取引条件 2 の場合には、利用できるコンテナが 1 本のみという制約が加わる。

アルゴリズム 3 に積載率調整手順の擬似コードを示す。発注可能点方策および MP 方策はどちらも、商品 i の基在庫点を表すパラメータ S_i があり、このアルゴリズムは、調整在庫位置 $I_i^p + \hat{x}_i$ が基在庫点 S_i に最も近い商品を調整対象商品として選択する。ここで、$\hat{x}_i$ は商品 i の調整後の補充数を表す。補充数を減らすか増やすかは、積載率に基づいて決定する。補充数を減らす場合には在庫位置が s^1 以下、増やす場合には s^2 以上である商品の中から、上述した基準に基づいて商品を選択する。ここで、(s^1, s^2) は、発注可能点方策の場合は (s, c)、MP 方策の場合は (s, s) である。この手順を追加することで、ベンチマーク方策の性能が向上することを確認した。積載率調整手順の評価は 3.6.4 に記載する。

3.6.3 強化学習による解法

強化学習による手法としては、BQL と C-IQL に加え、通常の IQL も比較対象として加えた。それぞれの手法における詳細な設定をここで示す。BQL では、ネットワークの共通部分は 512,256 の隠れ層を持ち、枝分かれした部分については枝ごとに 128 の隠れ層の設定とし、オプティマイザーには Adam を用いた。すべての隠れ層は ReLu による活性関数で、出力層は線形である。C-IQL および IQL についてはどちらも各エージェント

Algorithm 3 積載率調整手順

Require: $\mathrm{CAP}_{\mathrm{trans}}, \boldsymbol{x}, \boldsymbol{I}^p, \boldsymbol{s}_i^1, \boldsymbol{s}_i^2$

$n \leftarrow \lceil \frac{\sum_i x_{i,t}}{\mathrm{CAP}_{\mathrm{trans}}} \rceil$

mode ← decrease or increase depending on loading ratio

if mode = decrease **then** ▷ emptying one vehicle

  while $\sum_i x_i > (n-1) \times \mathrm{CAP}_{\mathrm{trans}}$ **do**

    $j \leftarrow \arg\min_{i \in j | I_j^p < s_j^1} (|S_i - (I_i^p + x_i - l_i)|$

    $x_j \leftarrow x_j - l_j$

else ▷ filling the "free" remaining capacity

  while $\sum_i x_i < n \times \mathrm{CAP}_{\mathrm{trans}}$ **do**

    j ← $\arg\min_{i \in j | I_j^p > s_j^2} (|S_i - (I_i^p + x_i + l_i)|$

    $x_j \leftarrow x_j + l_j$

return $\boldsymbol{x}$

64, 32, 32 の隠れ層を持ち、オプティマイザーおよび活性化関数の設定は BQL と同様である。

1 エピソードは 200 ステップ、学習エピソード数は 3,000、経験メモリ容量は 100,000、学習におけるミニバッチサイズは 32 とした。ターゲットネットワークの更新頻度は 10 エピソードに 1 回とした。学習率はエピソードに対して指数減衰させ、C-IQL では、TD 誤差が負の場合にはその時点の学習率に 0.4 を乗じる形で TD 誤差の正負に応じて異なる学習率を適用した [71, 74]。

3.6.4 数値実験結果

(a) 総費用評価

表 3.5 は、実験結果の概要として総費用を示す。表 3.7 は、学習した方策下での在庫受払時系列の統計量として平均在庫量と積載率を示しており、学習した補充方策の特徴を輸送費用と在庫保有費用のトレードオフの観点から検討できる。また、学習した方策に従った在庫受払のシミュレーション結果例として、全商品合計の補充量及び在庫量の時系列推移を図 3.7 以降に示す。なお、商品間の需要相関の有無が性能に有意な影響を与えなかったため、表 3.5 では $\rho = 0$ の場合の結果のみを示している。参考までに、需要相関がゼロ

ではない 2 商品の場合の結果を表 3.8 に示した。

発注可能点方策と MP 方策の比較では、商品数が少ない場合、特に需要偏差が大きい問題では MP 方策の性能が低く、従来研究での報告と一致する。一方で、商品数が 10 の場合は、すべての取引条件のもとで、MP 方策の性能は発注可能点方策と同等かそれ以上の性能を示している。

固定的な共通発注費用を想定した基本取引条件（取引条件 1）では、提案手法の性能は、需要偏差や商品数に関わらず、ベンチマーク方策の優れた方とほぼ同等であった。ただし、表 3.7 から、総費用は同程度であるものの、学習した方策の特徴は異なることがわかる。提案手法における在庫量はベンチマーク方策よりも低い一方で、特に商品数が多くなると、ベンチマーク方策の積載率は提案手法よりも高くなっている。C-IQL 以外の強化学習手法との比較は表 3.6 に示す。IQL は、商品数が 2 の単純な場合においてもベンチマーク方策に大きく劣る結果となっている。BQL は商品数が少ない場合にはベンチマーク方策同様の結果が得られているが、商品数の増加と共にベンチマーク方策に対するパフォーマンスの低下が見られた。

輸送容量制約がある場合と階段状輸送費用の場合（取引条件 2,3,5）には、ベンチマーク方策に比べて C-IQL の性能が良く、商品数の増加に伴って性能の差は拡大している。この性能差は、ベンチマーク方策が輸送容量を適切に考慮できていないことを示唆している。先に示したとおり、輸送容量を考慮するために、追加の積載率調整ヒューリスティック手順を導入した。しかし、GA によるパラメータ最適化と積載率調整ヒューリスティックという 2 段階の手順は、それぞれ総費用の削減に寄与しているものの、必ずしも良い解が得られるわけではない（2 段階手順の詳細な評価については後述）。表 3.7 を見ると、基本取引条件の場合と同様に、提案手法の在庫量が発注可能点方策よりも低いのに対し、発注可能点方策の積載率は提案手法よりも高くなっている。総費用としては提案手法の方が小さいことから、C-IQL の方が、ベンチマーク方策を用いた 2 段階手順よりも在庫保管費用と輸送費用のトレードオフにより適切に対処できていることを示している。

一方、需要のばらつきが大きい場合 ($c_v = 0.6$)、ベンチマーク方策に対するパフォーマンスの向上度合いが限定的であることがわかる。需要の分散は強化学習において報酬の分散に直接的につながっており、その結果学習が安定していない可能性が考えられる。もう一つの可能性としては、ジョイントアクション選択のヒューリスティックにおける逐次探索手順が考えられる。アルゴリズム 2 の逐次探索では、2 番目以降の検索結果がランダムに選択されるため、入力状態と行動値関数のパラメータが同じであっても、選択されたジョイントアクションが変化する可能性があり、すなわち、選択されたジョイントアクションが最適ではない可能性がある。この逐次探索におけるランダム探索の影響は、需要

偏差が大きいほど大きくなると考えられる。

非線形在庫保管費用の場合（取引条件 4）では、ベンチマーク方策に対して有意に良好な結果は達成できなかった（平均値としては提案手法の方が若干性能が劣るが、6 回の実行における標準偏差を考慮した場合には、その差は有意なものではない）。実験結果から、GA を用いて方策パラメータを最適化した場合、今回のベンチマーク方策は追加のロジックを必要とせずとも非線形在庫保管費用を考慮できているといえ、非線形な在庫保管費用のためのみに新たな解法を考案する必要性は比較的低いと結論づけられる。

最後に、環境問題の観点から積載率について述べる。本実験ではあくまでも輸送費用、在庫保管費用、機会損失費用の合計費用のみを考慮しており、その合計費用の観点からは必ずしも積載率の最大化が費用最小化につながらないことを示した。一方で、サプライチェーンにおける温室効果ガス、特に CO2 排出量の大きさは従来より問題視されている。コンテナ船による海上輸送における CO2 排出量は、これまでは貨物の重量と運送する距離のみで定まるトンキロベースで簡易的に計算されることが一般的であったが、積載率が低い場合にトンキロ当たりの CO2 排出量が増大することがわかってきており、コンテナ船での業界標準を提供する Clean Cargo Working Group（CCWG）も 2018 年のレポート [91] において、将来的に CO2 排出量の計算式に積載率を導入することを示している。本提案手法においては、CO2 排出量を金額換算した費用に変換して導入することで、CO2 排出量も考慮した補充方策の学習が可能である。

上海–東京間を 40ft コンテナ船で輸送した場合を想定した CO2 排出に関わる環境価値の試算結果を図 3.4 に示す。ここで、CO2 取引価格とは、2020 年 9 月における再エネクレジットの価格 [92] を用いており、コンテナ船によるトンキロあたり CO2 排出量は 56.4g とした [93]。上海–東京間での 40ft コンテナ輸送費用は近年変動が大きいが、JETRO 調査レポート [94] をもとに 30,000 円とすると、再エネクレジットの価格で換算した CO2 排出の環境価値は輸送費の 2 割弱を占めることがわかる。また、再エネクレジットは近年上昇傾向であり、今後輸送費に対する割合はさらに上昇する可能性も考えられることから、RE100 加盟企業など再生可能エネルギーの積極的利用を宣言する企業にとっては、海上輸送における CO2 排出量を考慮した補充方策の重要性は年々増加していくことが想定される。

(b)　商品数及び決定空間の大きさを増やした場合の評価

前述の実験では、最大で商品数は 10 であり、各エージェントの行動空間 $|\mathcal{A}_i|$ の大きさは 6 に制限していた。商品数及び各エージェントの行動空間の設定はどちらもジョイントアクションの行動空間に影響を与える。そこで、これらを増やした場合の性能評価を実施

表 3.4　CO2 排出に伴う環境価値

		数値	単位
再エネクレジット	CO2 取引価格	1,887	円/ton
40ft コンテナ	CO2 排出量単位	56.4	g/ton-km
	重量	30,000	kg
	距離（上海–東京）	1,764	km
	運賃（上海–東京）	30,000	円
	輸送 ton-km	52,920	ton-km
	CO2 排出量	2.98	ton
	CO2 価格換算	5,632	円

した。

まず、商品数を 50 にまで増やした場合について述べる。ここでは基本取引条件で商品数が 10 の場合をもとに、需要やロットサイズの条件が同様の商品がそれぞれ 5 つずつ存在する商品数 50 の状況を設定し、ベンチマーク方策との比較評価を行なった。表 3.9 に結果を示す。ここで、$(c_v, \rho) = (0.2, 0)$ である。C-IQL での数値実験の設定は他の商品数との条件と同一であるが、ベンチマーク方策では世代数は 100 では収束しなかったため 200 とした。

表 3.9 の通り、C-IQL で学習した方策が最もパフォーマンスが良い結果となり、商品数が増えた場合に計算時間はエージェント数に比例して長くなるものの、GA よりも総費用の観点で良い解に収束している。

発注可能点方策が MP 方策と比べて高い費用になっているのは、方策のパラメータ数が発注可能点方策では商品ごとに 3 つ存在するため 50 商品では 150 のパラメータが存在し、結果として良い解に収束できていないことが考えられる。

次に、行動空間についての評価について述べる。行動空間が小さすぎると、最適解が含まれていない可能性があり、一方で行動空間が大きすぎると、収束しなかったり、収束するまでに長いエピソードが必要になったりする可能性が想定される。そこで、$c_v = 0.2$ の基本取引条件について、行動空間を大きくした場合の性能への影響を評価するための追加実験を行った。なお、$|\mathcal{A}_i| = 6$ のシミュレーションでは、商品数が 2, 5, 10 の基本取引条件において、$\{0, 1, 2, 3, 4, 5\}$ の中で選択された最大の行動（ロットサイズの乗数）は、それぞれ 3, 4, 2 であり、全ての商品に対して $|\mathcal{A}_i| = 6$ は十分に大きい設定であると言える。

表 3.10 から、同じ実験環境下では、商品数 5 の場合までは、$|\mathcal{A}_i| = 10$ と $|\mathcal{A}_i| = 20$ の場合は、$|\mathcal{A}_i| = 6$ の場合と同等の性能が得られることを確認した。一方で、商品数 10 の場合には、$|\mathcal{A}_i|$ の増加に対し、若干のパフォーマンス低下が見られた。本手法を適用する場合には、最適解が含まれると想定される範囲でなるべく小さな決定空間を設定することが望ましい。

(c)　ベンチマーク手法における 2 段階手順の評価

輸送容量制約及び階段状の輸送費用の取引条件については、輸送能力を考慮するために、アルゴリズム 3 で示される追加的なヒューリステスティック手順を導入した。ここでは、1) GA を用いてベンチマーク方策のより良いパラメータを求め、2) 積載率調整手順を適用する、という 2 段階で構成されている。この 2 段階の手順の有効性を個別に評価するために、5 商品の取引条件 3（階段状輸送費用）を対象として、パラメータ固定（GA 最適化なし）and/or 積載率調整なしの追加実験を実施した。

固定パラメータとして、各商品の予め設定された発注点 s_i は、0.1% 欠品水準を想定した安全在庫に 3 期間分の平均需要を加えたもの、すなわち $3\mu_i + 3.1\sigma_i\sqrt{\mathrm{LT}}$ とした。各商品の発注可能点 c_i は、s_i に 1 期間分の平均需要を加えたものとし、各商品の基在庫量 S_i は、s_i に 2 期間分の平均需要を加えたものとした。MP 方策の補充頻度を表すグローバルパラメータは 1 とした。

表 3.11 に結果を示す。GA によるパラメータ最適化の有効性としては、GA を適用しない場合の方が GA を適用した場合よりも総費用が高くなっており、GA で導出されたパラメータを適用することで改善されていることがわかる（表 3.11 で積載率調整しない場合の行ごとの比較）。また、積載率の調整手順については、GA による最適化の有無に関わらず、全てのケースで総費用が減少している（表 3.11 の列ごとの比較）。

しかし、$c_v = 0.2$ の発注可能点方策では、GA と積載率調整の 2 段階手順を踏んだ方策の総費用（460.0）が、GA を使わずに積載率調整手順を用いた方策の総費用（448.7）よりも高くなっていることがわかる。これは、2 段階の手順で最適解に収束できなかったことを意味しており、発注可能点方策や MP 方策を含む既存の方策を、基本取引条件以外に拡張することの難しさを反映していると考えられる。

(d)　計算時間の評価

提案手法とベンチマーク手法の計算時間について述べる。ここで、実際の計算時間は、GA では世代数と各母集団の個体数の設定に依存し、強化学習を用いた提案手法ではエピソード数の設定に依存するため、絶対的な計算時間の比較ではなく、商品数に対する計算

時間の変化を分析することに重点を置く。

表 3.12 に平均的な計算時間を示す。ここで、提案手法は商品数に関係なく同じエピソード数を設定した一方、GA の設定では、母集団の数を商品数に応じて変更している。そのため、公平な比較のため、実際の時間を $N/2$ で割った仮想的な計算時間を GA の場合に $(\cdot)$ で示している。

実験は同一コンピュータ上で行い、GPU は一切使用していない。なお、行動値関数のニューラルネットワークのアーキテクチャは単純であるため、GPU を使用しても計算時間の改善にはつながらない。

GA では母集団と世代数、提案手法ではエピソード数を商品数によらずに設定した場合、どちらも商品数に対して計算時間がほぼ線形となっている。ここで、本数値実験においては、並列計算処理は採用していない。

提案手法における学習時間の大部分は各エージェントの行動価値関数のニューラルネットワークのフォワードとバックワード処理に費やされており、これらの処理をエージェントごとに並列計算することで、提案手法の計算時間の短縮は可能である。提案手法では、他の商品の情報は、合計補充量という形で各エージェントのネットワークにスカラーとして入力されるため、各エージェントの行動価値のパラメータ数は商品数に対して変化しない。したがって、十分な数の CPU が利用可能であれば、計算時間は商品数に対して亜線形にしか増加しない。また Ape-X [85] や IMAPLA [87] などシングルエージェントに向けに提案された分散型強化学習手法はマルチエージェントの設定にも適用可能であり [95]、さらなる計算時間短縮が可能であると想定される。

一方で、GA のパラメータ最適化の計算時間についても、各解の評価での並列処理、もしくは本論文で採用したバイナリコーディングの代わりにグレイコーディング [96, 97] を用いることで短縮が可能である。

3.7 おわりに

同時補充問題に対し、高次元離散行動空間への対処と、協調的な補充方策の学習の両立を目的として、報酬分配に着目したマルチエージェント強化学習手法の BQL と C-IQL を提案した。

数値実験の結果、C-IQL では、共通発注費用が発注量に関わらず固定的に課され、かつ在庫保管費用が在庫量に対して線形であるという基本取引条件のもとでは、いずれの需要特性・商品数においても、既存近似方策のうち優れた方と同等の結果が得られた。さらに、輸送容量を考慮した場合には、既存近似方策（発注可能点方策及び MP 方策のいずれ

か良い方）と比較して、総費用の観点で最大 16 %のパフォーマンスの向上を確認した。

ただし、ジョイントアクションの決定にはヒューリスティック手法を用いており、本数値実験で示した取引条件以外の条件下でも本手法が有効かどうかは検証が必要である。また、本提案手法は需要の分散が大きい場合にはパフォーマンスの改善度合いが限定的であった。これは、強化学習における報酬の高分散、もしくは、ジョイントアクションのヒューリスティック探索における逐次探索手順における探索順の設計の観点でさらなる調査が求められる。

さらに、本手法を実務に適用する場合にはシミュレーションと実世界との差異が課題となる。同時補充問題では、需要分布や補充リードタイムがシミュレーションと実世界との差異の要因となりうる。シミュレーションを通じたオフラインでの学習において想定していなかった状況に対しても適切な補充数を決定するためには、実行時に局所的な探索を加えることが考えられる。本問題は連続状態空間、高次元離散行動空間の問題であるため、実行時の限られた計算時間における効率的な探索手法が求められ、実務適用に向けた重要な研究課題である。

表 3.5 数値実験結果

取引条件	商品数	(c_v, ρ)	発注可能点方策	MP 方策	C-IQL	改善率
1)	2	(0.2,0)	95.0 ± 0.9	99.2 ± 0.8	95.3 ± 0.7	-0.4%
		(0.6,0)	116.5 ± 2.1	125.3 ± 2.4	115.3 ± 2.3	1.0%
	5	(0.2,0)	76.6 ± 0.6	76.6 ± 0.5	73.6 ± 1.2	3.9%
		(0.6,0)	87.8 ± 0.7	89.1 ± 1.2	87.0 ± 0.8	0.9%
	10	(0.2,0)	173.8 ± 1.7	169.0 ± 1.5	171.6 ± 0.2	-1.6%
		(0.6,0)	210.7 ± 2.7	209.4 ± 2.0	210.5 ± 1.7	-0.5%
2)	2	(0.2,0)	98.4 ± 1.3	102.5 ± 2.5	97.1 ± 0.6	1.4%
		(0.6,0)	119.0 ± 2.0	129.4 ± 5.7	118.1 ± 1.2	0.7%
	5	(0.2,0)	78.1 ± 4.7	77.0 ± 0.3	72.9 ± 0.7	5.3%
		(0.6,0)	87.5 ± 0.7	89.0 ± 1.3	87.1 ± 0.5	0.5%
	10	(0.2,0)	192.5 ± 8.3	189.3 ± 0.6	178.5 ± 1.2	5.7%
		(0.6,0)	229.3 ± 6.0	225.5 ± 1.3	219.1 ± 1.3	2.9%
3)	2	(0.2,0)	507.5 ± 7.1	503.3 ± 4.3	506.5 ± 2.4	-0.6%
		(0.6,0)	728.8 ± 16.2	718.6 ± 7.6	715.8 ± 12.8	0.4%
	5	(0.2,0)	460.0 ± 4.6	479.2 ± 25.6	444.0 ± 3.6	3.5%
		(0.6,0)	611.1 ± 8.5	618.3 ± 12.7	607.7 ± 4.4	0.5%
	10	(0.2,0)	918.6 ± 23.4	893.6 ± 3.7	751.7 ± 3.1	15.9%
		(0.6,0)	1126.4 ± 28	1108.9 ± 24.9	1014.2 ± 12.8	8.5%
4)	2	(0.2,0)	90.2 ± 1.6	91.8 ± 0.4	90.1 ± 0.5	0.1%
		(0.6,0)	103.5 ± 0.8	110.6 ± 2.0	104.0 ± 1.2	-0.5%
	5	(0.2,0)	75.0 ± 0.4	74.3 ± 0.3	75.4 ± 0.3	-1.5%
		(0.6,0)	81.5 ± 0.5	83.0 ± 1.3	82.2 ± 0.8	-0.9%
	10	(0.2,0)	152.5 ± 2.8	148.6 ± 1.3	149.6 ± 2.4	-0.6%
		(0.6,0)	191.1 ± 2.5	188.4 ± 2.1	190.1 ± 1.1	-0.9%
5)	2	(0.2,0)	477.5 ± 9.3	470.8 ± 4.6	474.4 ± 4.6	-0.8%
		(0.6,0)	705.0 ± 13.7	683.4 ± 9.6	680.8 ± 15.6	0.4%
	5	(0.2,0)	442.3 ± 38.8	449.8 ± 16.8	425.9 ± 2.3	3.7%
		(0.6,0)	569.0 ± 7.5	583.3 ± 7.9	569.0 ± 5.2	0.0%
	10	(0.2,0)	752.3 ± 14.8	763.9 ± 2.1	722.1 ± 3.2	4.0%
		(0.6,0)	960.6 ± 11.2	999.5 ± 17.0	949.1 ± 5.4	1.2%

ベンチマーク手法と提案手法それぞれ、異なる初期化シードにより 6 回の学習を行った。各学習では、需要生成のシードを変えた 12 回のシミュレーションの総費用の平均値を計算し、結果を導出した。(±·) の値は、6 回の学習における標準偏差を表す。改善率は、ベンチマーク手法の良い方に対して、提案手法による改善率を表す。

表 3.6 数値実験結果（強化学習手法: 基本取引条件）

商品数	(c_v, ρ)	発注可能点方策	MP 方策	IQL	BQL	C-IQL
2	(0.2,0)	95.0	99.2	115.2	98.2	95.3
	(0.6,0)	116.5	125.3	133.6	120.7	115.3
5	(0.2,0)	76.6	76.6	156.4	95.6	73.6
	(0.6,0)	87.8	89.1	154.5	111.2	87.0
10	(0.2,0)	173.8	169.0	255.5	245.7	171.6
	(0.6,0)	210.7	209.4	296.4	309.1	210.5

BQL, IQL についても C-IQL と同様の条件で実験を行った。

表 3.7 学習方策によるシミュレーション結果での平均在庫量・補充量・積載率

			発注可能点方策			MP 方策			C-IQL		
Cost scenario	Products	(c_v, ρ)	Inv.	Load	Repl.	Inv.	Load	Repl.	Inv.	Load	Repl.
1)	2	(0.2,0)	16.5	-	23.1	16.8	-	23.6	14.6	-	21.1
		(0.6,0)	19.5	-	22.6	20.8	-	21.6	17.0	-	17.9
	5	(0.2,0)	14.4	-	20.2	14.6	-	20.5	10.7	-	13.7
		(0.6,0)	16.4	-	18.3	15.7	-	18.3	13.7	-	13.8
	10	(0.2,0)	35.6	-	39.8	33.8	-	37.9	23.2	-	18.8
		(0.6,0)	41.2	-	34.0	38.7	-	30.5	30.7	-	18.6
2)	2	(0.2,0)	15.2	93%	18.6	13.7	85%	17.0	12.7	81%	16.2
		(0.6,0)	19.0	94%	18.8	20.1	77%	15.4	17.1	84%	16.7
	5	(0.2,0)	13.5	91%	18.2	13.3	83%	16.7	10.5	69%	13.8
		(0.6,0)	15.4	83%	16.6	15.0	81%	16.2	13.4	65%	13.1
	10	(0.2,0)	33.1	91%	18.4	26.0	77%	15.3	23.5	79%	16.7
		(0.6,0)	37.5	86%	17.6	32.5	77%	15.4	32.0	76%	16.3
3)	2	(0.2,0)	59.0	97%	30.1	60.4	99%	29.9	54.5	98%	30.2
		(0.6,0)	92.2	93%	37.2	93.9	97%	38.3	68.1	97%	35.4
	5	(0.2,0)	51.2	91%	23.8	65.6	94%	51.6	48.8	86%	24.1
		(0.6,0)	81.1	85%	24.0	93.7	92%	49.8	74.0	84%	22.5
	10	(0.2,0)	213.3	90%	93.2	127.2	89%	114.8	85.2	82%	39.5
		(0.6,0)	163.1	87%	59.3	165.6	90%	105.3	110.8	83%	44.4
4)	2	(0.2,0)	18.2	-	23.7	18.9	-	23.8	18.0	-	24.9
		(0.6,0)	21.7	-	22.0	23.1	-	23.6	20.3	-	23.0
	5	(0.2,0)	15.9	-	20.5	16.1	-	21.6	16.7	-	23.6
		(0.6,0)	18.3	-	20.0	18.2	-	19.6	18.1	-	19.5
	10	(0.2,0)	32.9	-	32.1	33.6	-	37.9	28.2	-	23.0
		(0.6,0)	41.1	-	31.2	39.8	-	31.8	34.8	-	22.4
5)	2	(0.2,0)	60.2	97%	30.0	61.4	99%	29.9	53.3	98%	30.0
		(0.6,0)	91.9	92%	36.0	92.1	97%	35.0	80.9	96%	30.1
	5	(0.2,0)	51.3	94%	23.6	65.4	93%	55.6	52.2	84%	23.9
		(0.6,0)	80.9	85%	23.9	92.7	93%	48.0	75.0	82%	23.3
	10	(0.2,0)	84.9	80%	38.2	104.4	86%	76.2	81.8	81%	38.2
		(0.6,0)	127.4	81%	38.7	151.7	84%	70.2	121.1	80%	38.1

学習した方策下でシミュレーションした時系列変動例から得られる統計量を示す。それぞれの条件に対して 6 回学習しているため、各学習結果に対して、1 回のシミュレーションを実施し、6 回の学習結果に対する平均的な在庫量、積載率、1 回あたりの補充数量を示す。

表 3.8 数値実験結果（2 商品、需要相関あり）

取引条件	(c_v, ρ)	発注可能点方策	MP 方策	C-IQL	改善率
1)	(0.2,-0.5)	95.8 ± 1.6	99.0 ± 1.0	95.5 ± 0.6	0.3%
	(0.2,0)	95.0 ± 0.9	99.2 ± 0.8	95.3 ± 0.7	-0.4%
	(0.2,0.5)	95.8 ± 1.9	98.7 ± 1.3	94.5 ± 0.6	1.4%
	(0.6,-0.5)	116.4 ± 0.9	123.7 ± 2.8	117.3 ± 2.0	-0.7%
	(0.6,0)	116.5 ± 2.1	125.3 ± 2.4	115.3 ± 2.3	1.0%
	(0.6,0.5)	115.2 ± 1.4	123.1 ± 1.7	114.0 ± 1.8	1.0%
2)	(0.2,-0.5)	98.7 ± 1.3	105.2 ± 6.9	97.6 ± 0.4	1.1%
	(0.2,0)	98.4 ± 1.3	102.5 ± 2.5	97.1 ± 0.6	1.4%
	(0.2,0.5)	98.4 ± 0.9	101.4 ± 2.0	96.5 ± 0.8	1.9%
	(0.6,-0.5)	120.0 ± 2.0	125.3 ± 4.8	118.3 ± 1.3	1.5%
	(0.6,0)	119.0 ± 2.0	129.4 ± 5.7	118.1 ± 1.2	0.7%
	(0.6,0.5)	117.0 ± 2.2	131.9 ± 5.0	116.2 ± 1.1	0.6%
3)	(0.2,-0.5)	516.6 ± 9.1	506.9 ± 5.5	506.5 ± 2.7	0.1%
	(0.2,0)	507.5 ± 7.1	503.3 ± 4.3	506.5 ± 2.4	-0.6%
	(0.2,0.5)	500.8 ± 3.2	500.3 ± 4.5	500.2 ± 2.2	0.0%
	(0.6,-0.5)	733.0 ± 5.5	718.8 ± 8.0	716.1 ± 12.2	0.4%
	(0.6,0)	728.8 ± 16.2	718.6 ± 7.6	715.8 ± 12.8	0.4%
	(0.6,0.5)	729.8 ± 14.8	715.4 ± 23.4	708.5 ± 3.6	1.0%
4)	(0.2,-0.5)	90.1 ± 0.5	91.8 ± 1.1	90.0 ± 0.4	0.1%
	(0.2,0)	90.2 ± 1.6	91.8 ± 0.4	90.1 ± 0.5	0.1%
	(0.2,0.5)	89.3 ± 0.3	91.9 ± 0.7	90.0 ± 0.6	-0.8%
	(0.6,-0.5)	103.7 ± 1.1	109.3 ± 2.0	104.0 ± 0.9	-0.3%
	(0.6,0)	103.5 ± 0.8	110.6 ± 2.0	104.0 ± 1.2	-0.5%
	(0.6,0.5)	102.6 ± 1.8	110.2 ± 1.6	102.6 ± 1.2	0.1%
5)	(0.2,-0.5)	489.7 ± 12.9	475.1 ± 4.8	478.8 ± 5.3	-0.8%
	(0.2,0)	477.5 ± 9.3	470.8 ± 4.6	474.4 ± 4.6	-0.8%
	(0.2,0.5)	481.6 ± 10.0	470.6 ± 2.9	469.6 ± 3.9	0.2%
	(0.6,-0.5)	710.6 ± 12.4	682.1 ± 8.8	673.8 ± 21.5	1.2%
	(0.6,0)	705.0 ± 13.7	683.4 ± 9.6	680.8 ± 15.6	0.4%
	(0.6,0.5)	691.1 ± 13.9	672.2 ± 16.9	672.9 ± 18.1	-0.1%

表 3.9　商品数 50 での数値実験結果

商品数	発注可能点方策	MP 方策	C-IQL
50	732.3 ± 7.5	678.2 ± 2.5	575.3 ± 0.5

基本取引条件 ($(c_v, \rho) = (0.2, 0)$) における平均総費用を示す。C-IQL での数値実験の設定は他の商品数との条件と同一。ベンチマーク方策では世代数は 100 では収束しなかったため 200 とした。

表 3.10　行動空間の設定に関する数値実験結果

商品数	$\lvert\mathcal{A}_i\rvert = 6$	$\lvert\mathcal{A}_i\rvert = 10$	$\lvert\mathcal{A}_i\rvert = 20$
2	95.3	94.9	95.6
5	73.6	72.9	72.9
10	171.6	173.0	175.6

基本取引条件 ($c_v = 0.2$) における平均総費用を示す。決定空間を大きくした場合の数値実験も $|\mathcal{A}_i| = 6$ と同様の設定のもとで実施した。

表 3.11　ベンチマークの 2 段階手順の有効性の評価

			積載率調整の有無	
方策	c_v	GA	No	Yes
発注可能点	0.2	No	493.2	448.7
		Yes	468.3	460.0
	0.6	No	668.3	629.0
		Yes	617.0	611.1
MP	0.2	No	536.6	508.0
		Yes	489.3	479.2
	0.6	No	692.1	663.7
		Yes	626.8	618.3

5 商品の取引条件 3（階段状輸送費用）に対する平均総費用を示す。

表 3.12　計算時間

商品数	GA(発注可能点方策)	C-IQL
2	23 (23)	85
5	143 (57)	235
10	540 (108)	520

計算時間は分単位で記載している。公平な比較のため、GA における (·) は、商品数に対する世代数の違いを調整した仮想的な時間を表している。

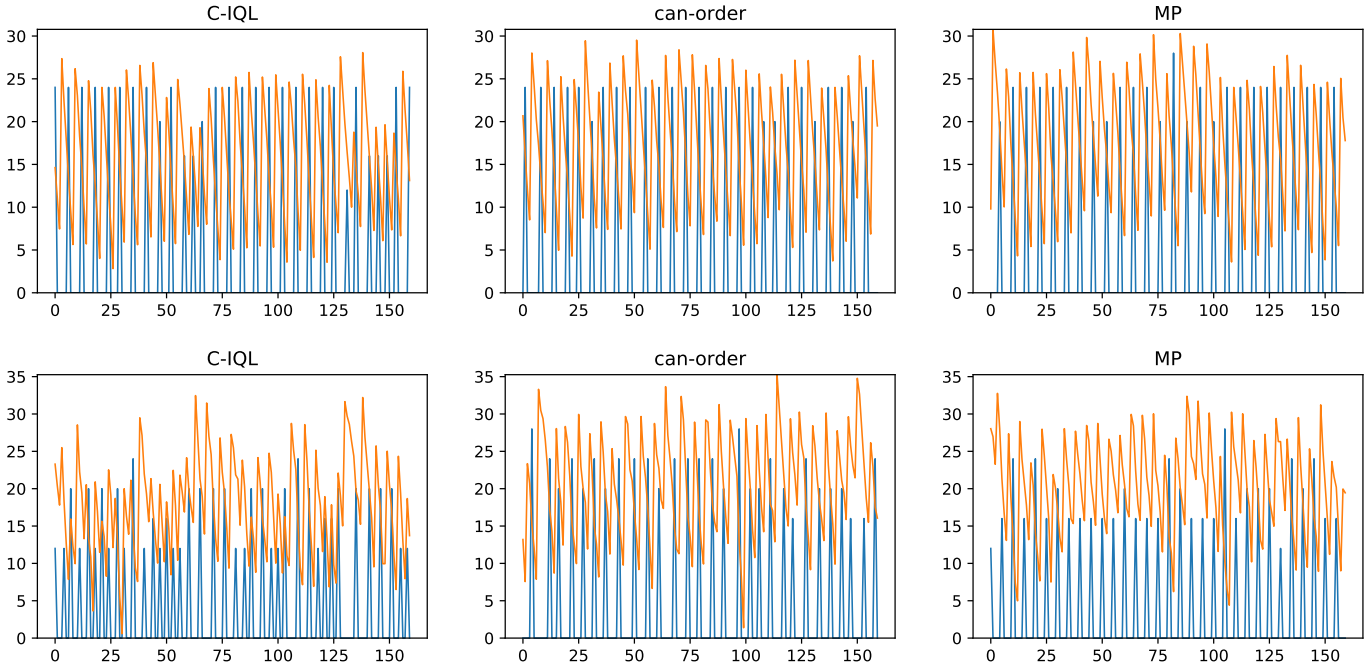

図 3.18　時系列推移例: 取引条件 1 商品数 2

オレンジ線が平均在庫量、青線が補充量を表す。左から、C-IQL（提案手法)、発注可能点方策、MP 方策。上から、$(c_v, \rho) = (0.2, 0), (0.6, 0)$。

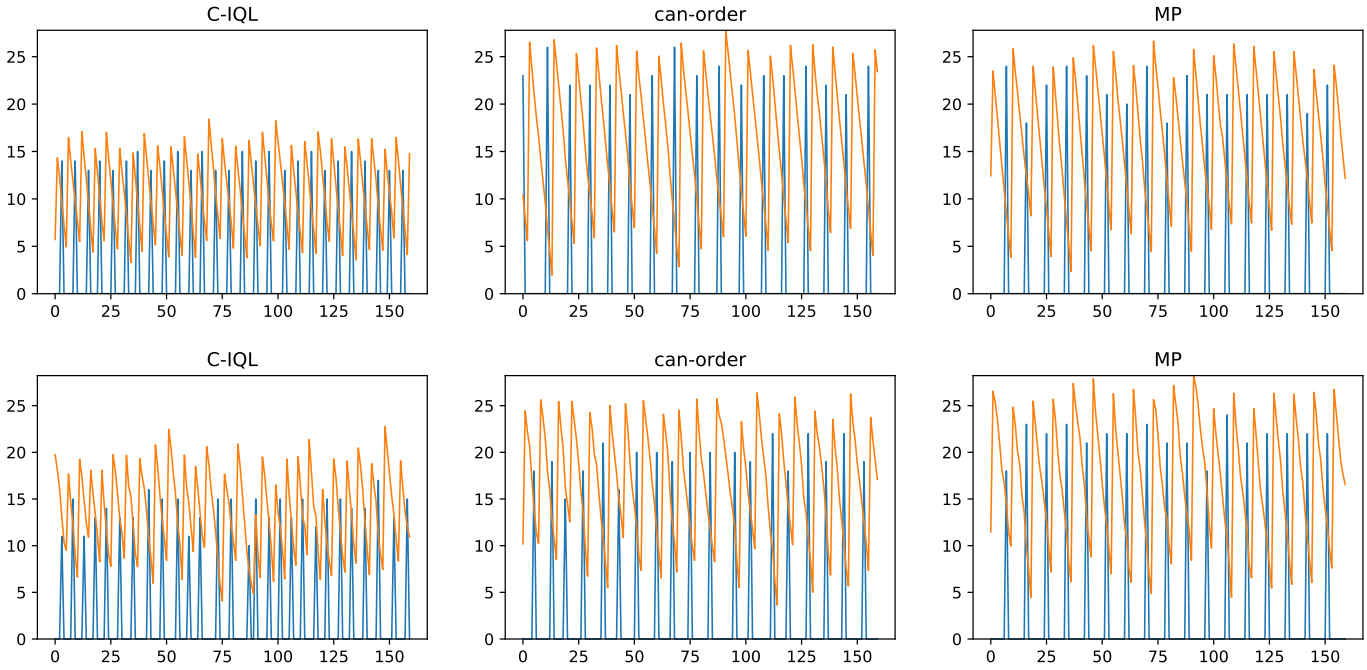

図 3.19　時系列推移例: 取引条件 1 商品数 5

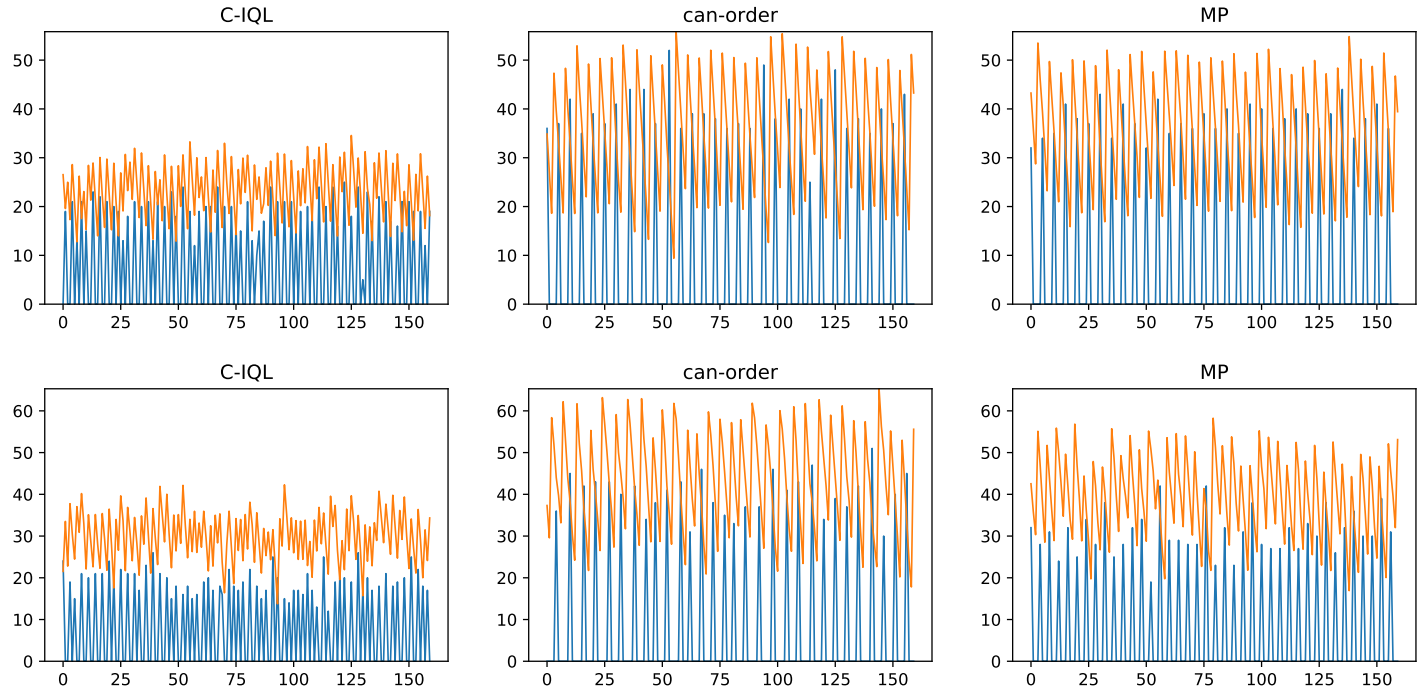

図 3.20　時系列推移例: 取引条件 1 商品数 10

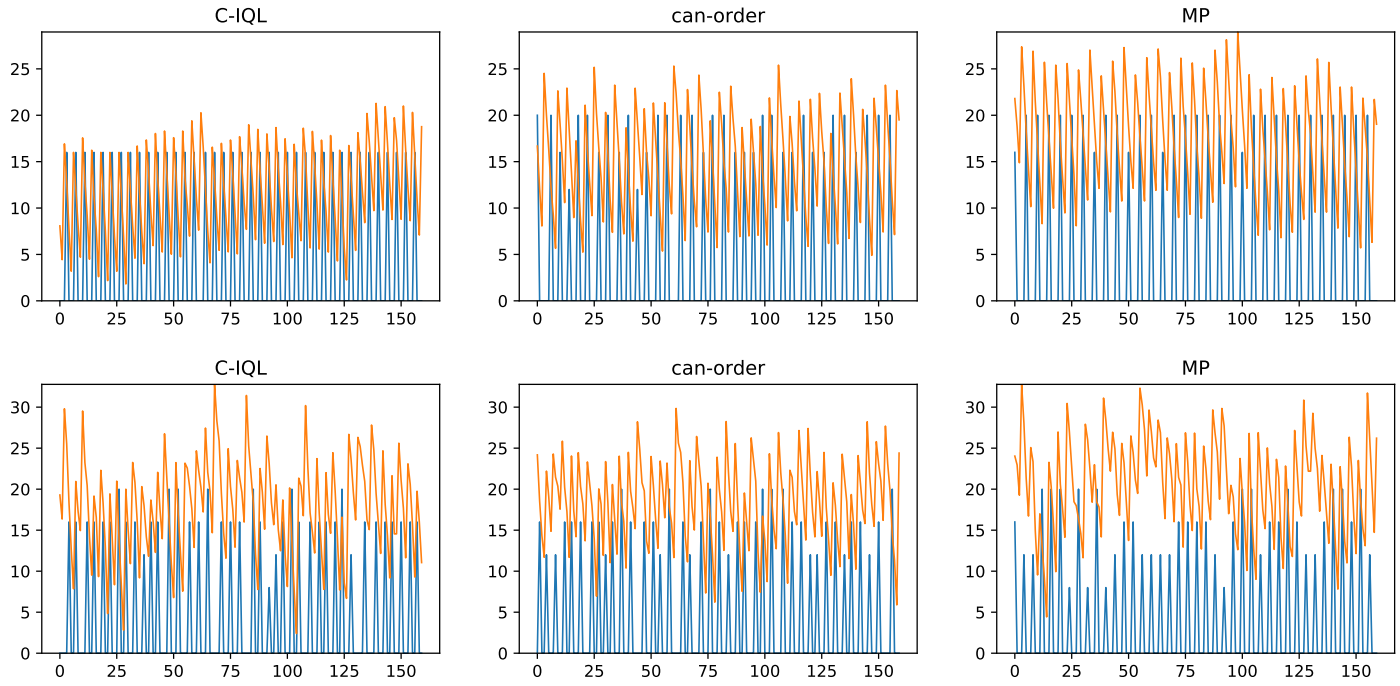

図 3.21　時系列推移例: 取引条件 2 商品数 2

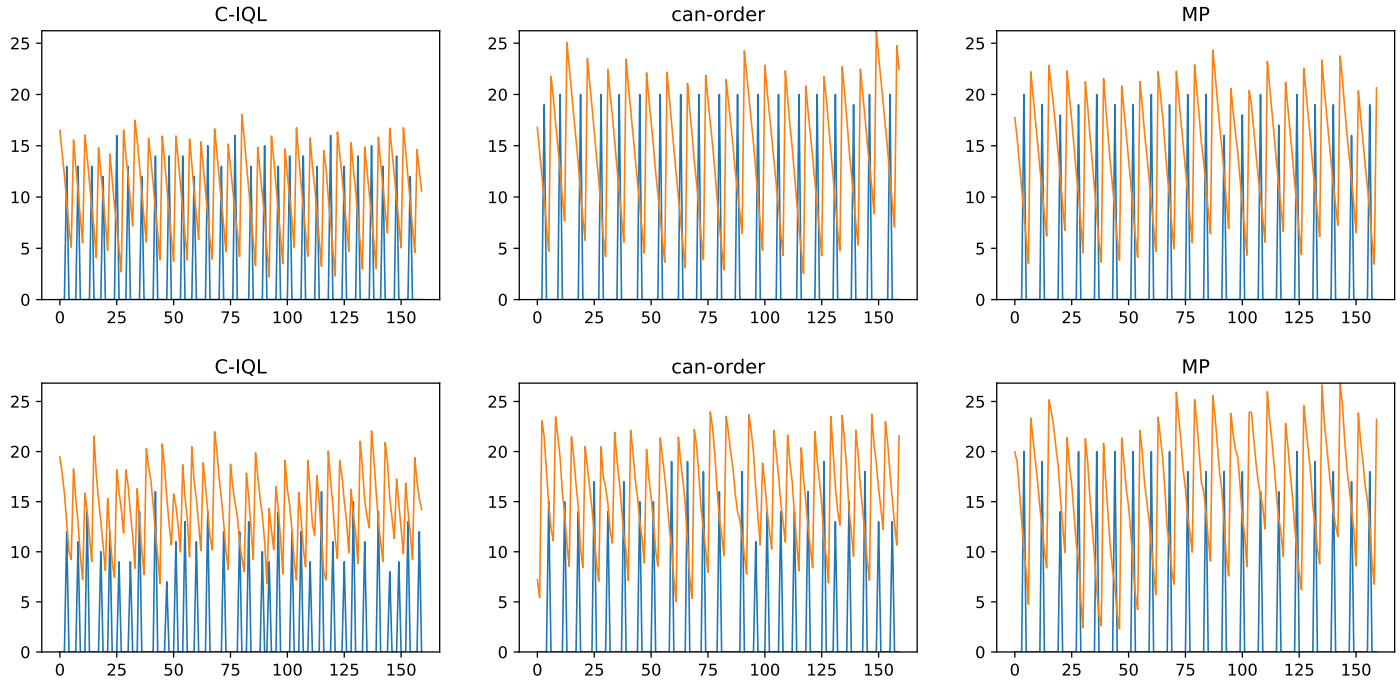

図 3.22　時系列推移例: 取引条件 2 商品数 5

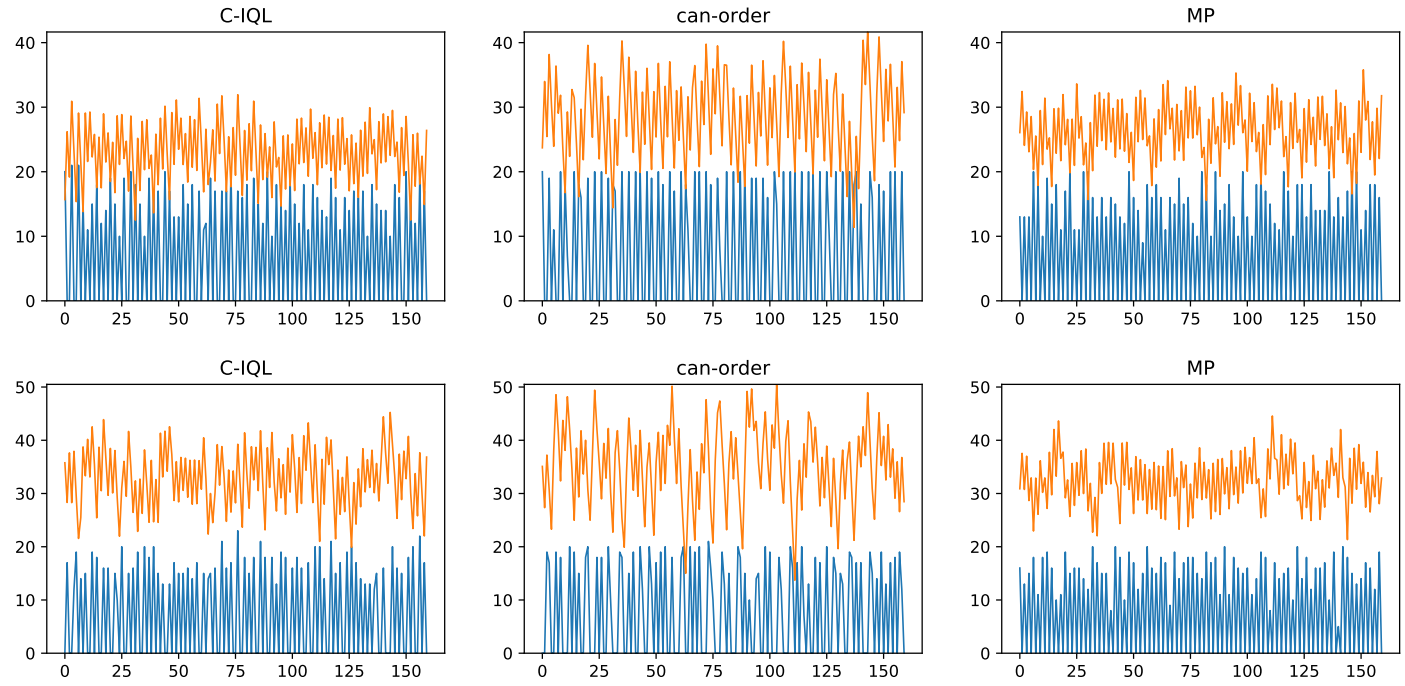

図 3.23　時系列推移例: 取引条件 2 商品数 10

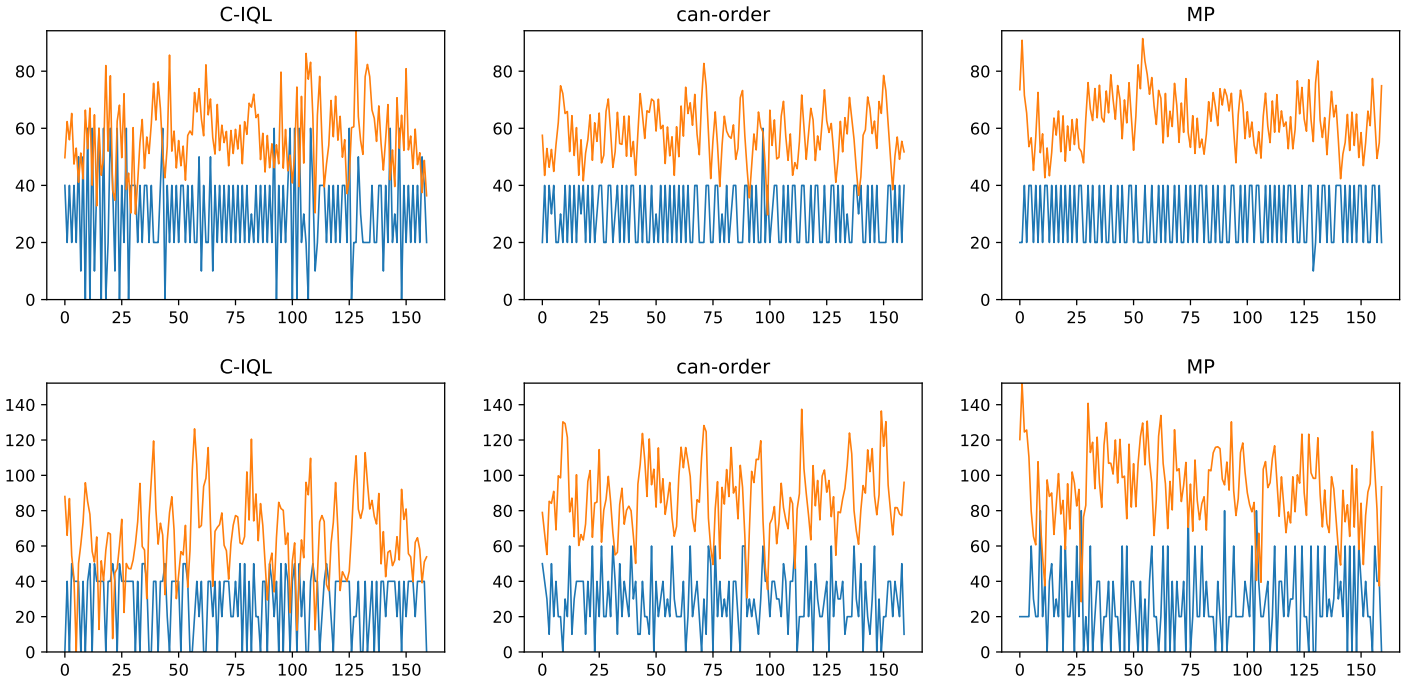

図 3.24　時系列推移例: 取引条件 3 商品数 2

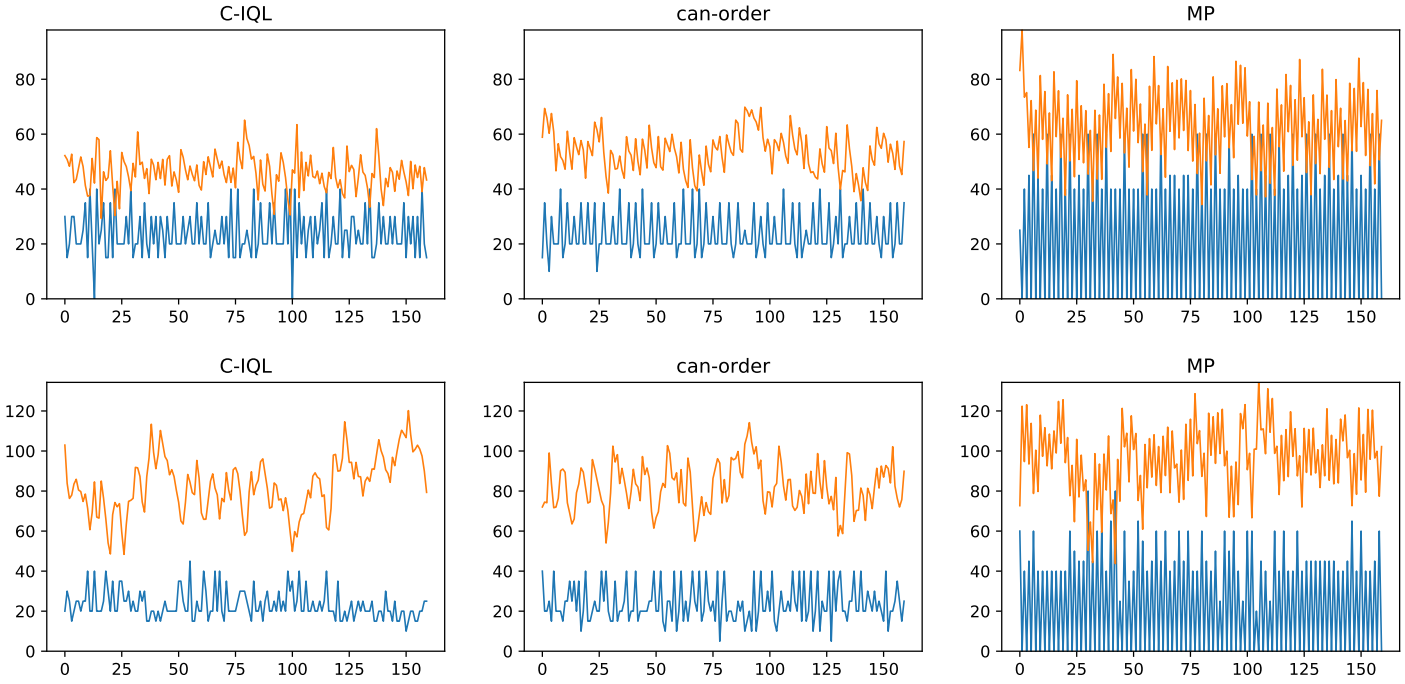

図 3.25　時系列推移例: 取引条件 3 商品数 5

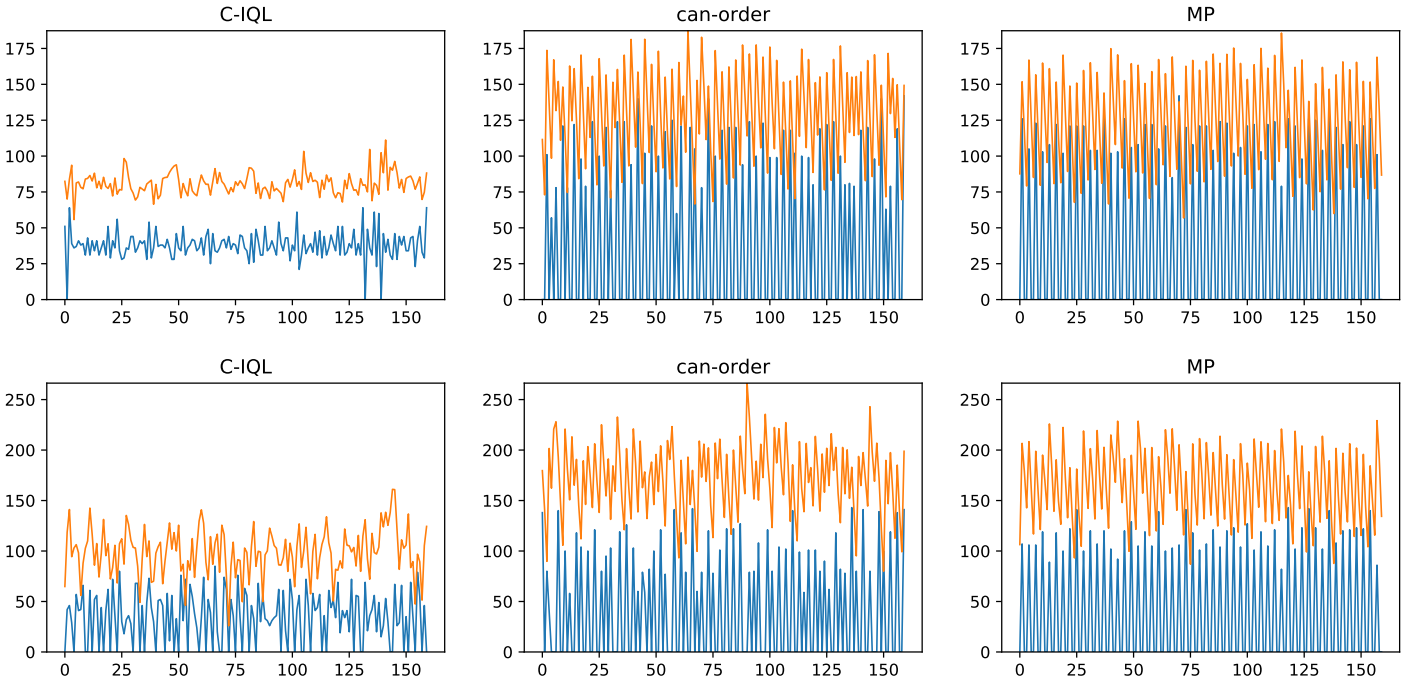

図 3.26　時系列推移例: 取引条件 3 商品数 10

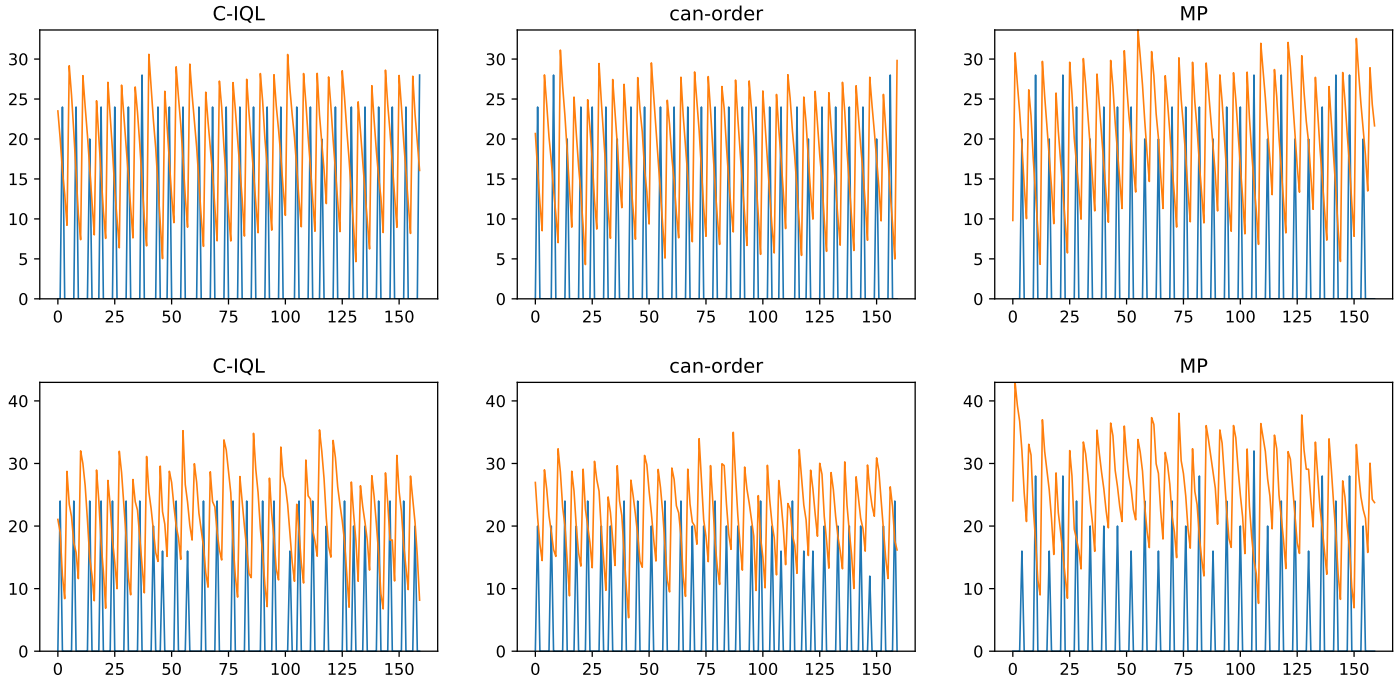

図 3.27　時系列推移例: 取引条件 4 商品数 2

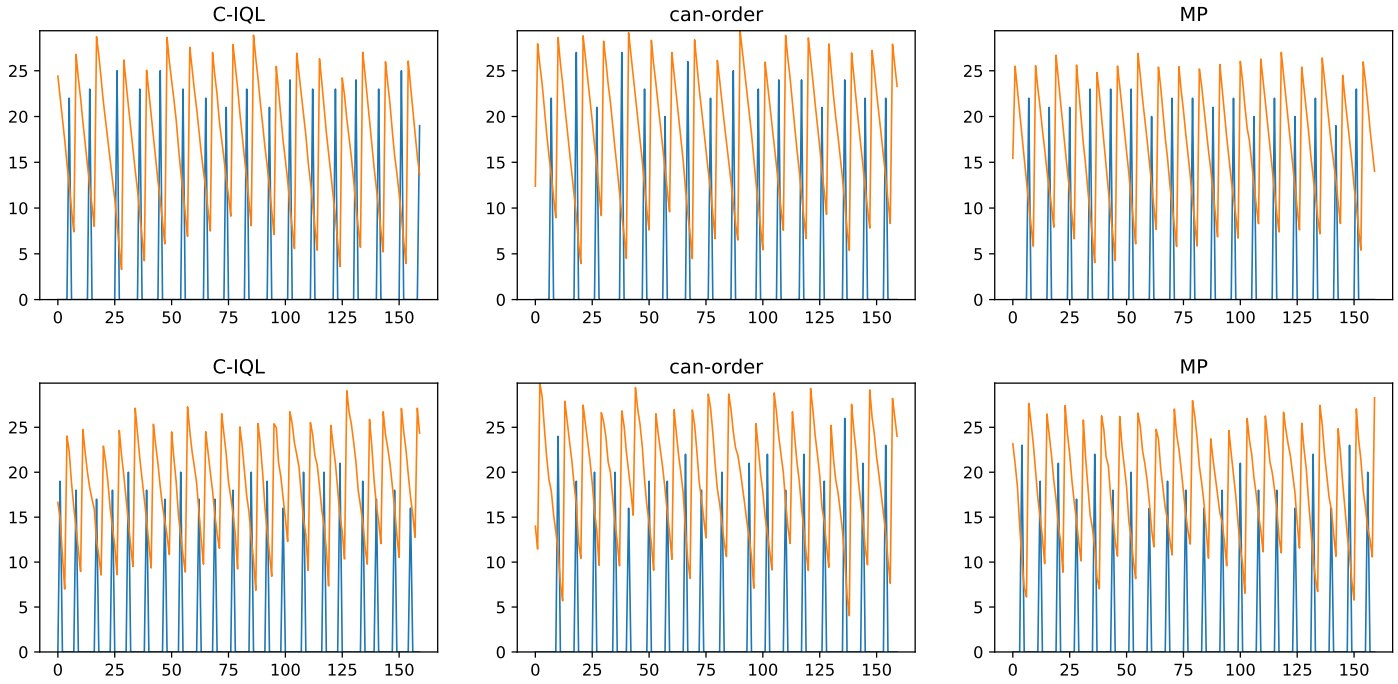

図 3.28　時系列推移例: 取引条件 4 商品数 5

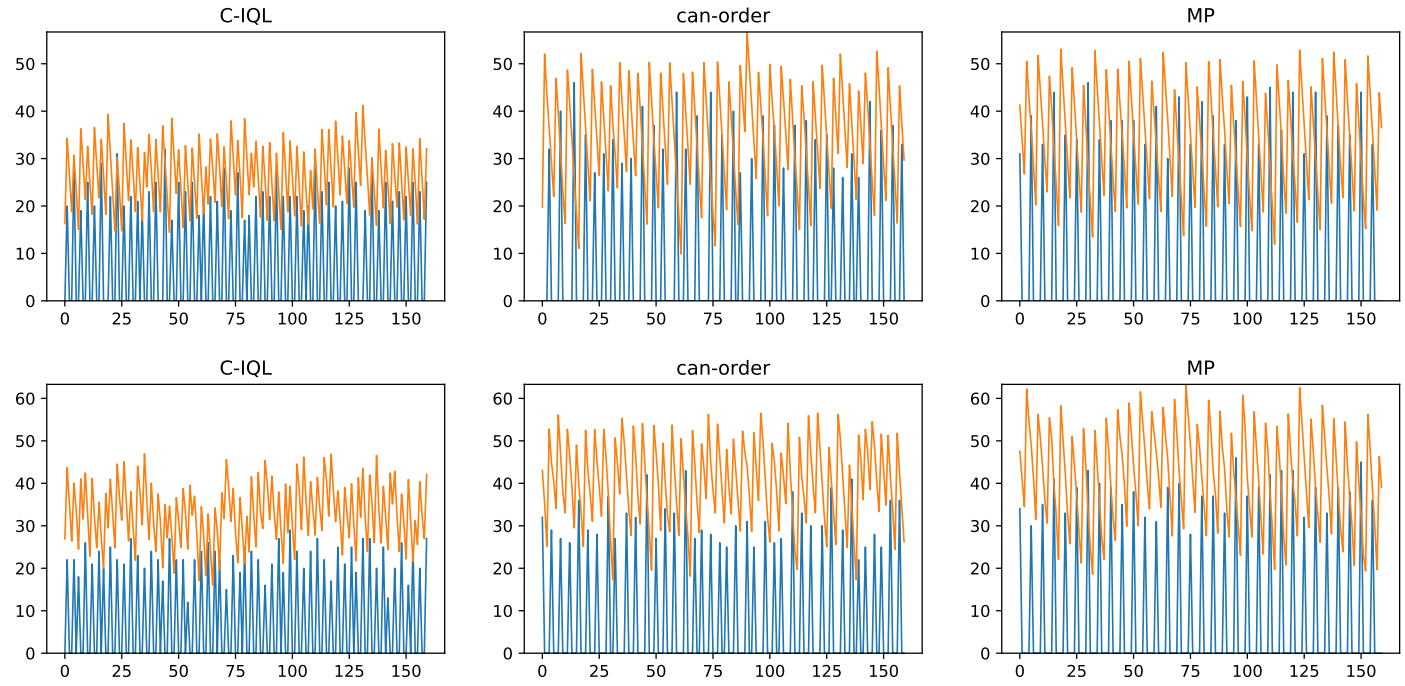

図 3.29　時系列推移例: 取引条件 4 商品数 10

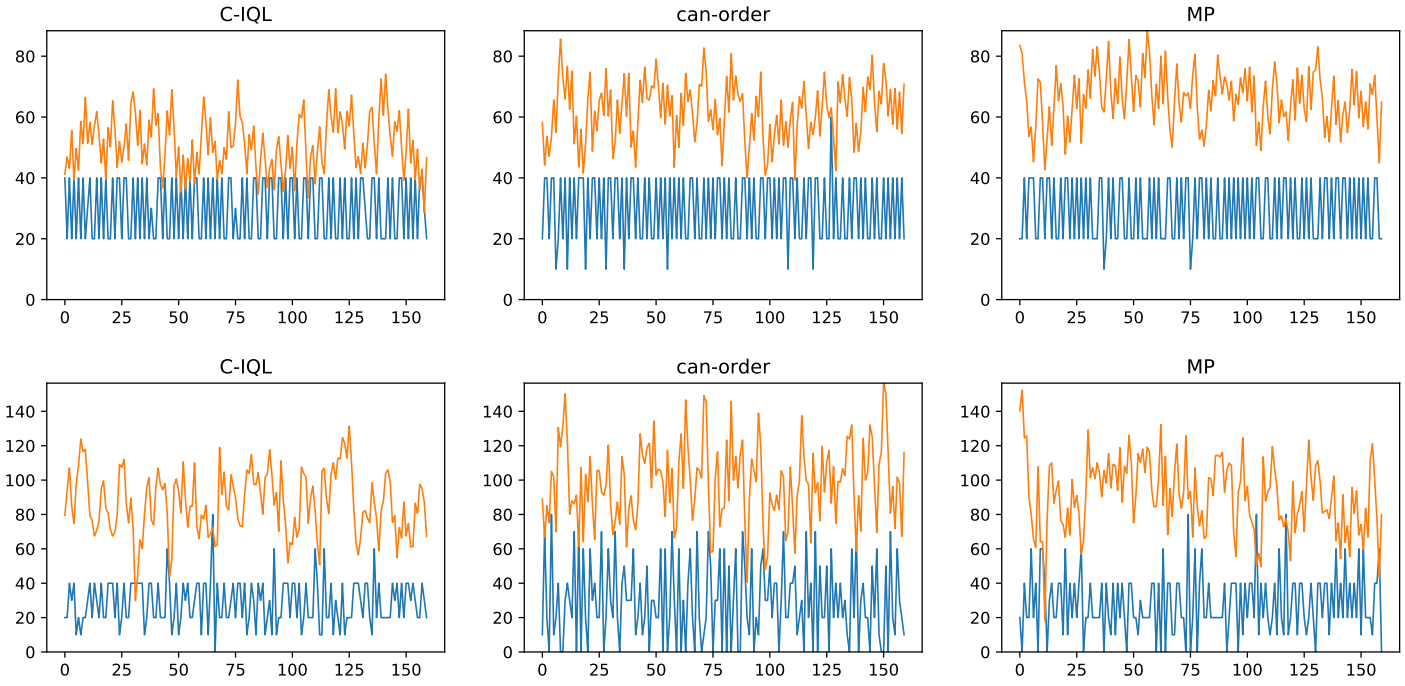

図 3.30　時系列推移例: 取引条件 5 商品数 2

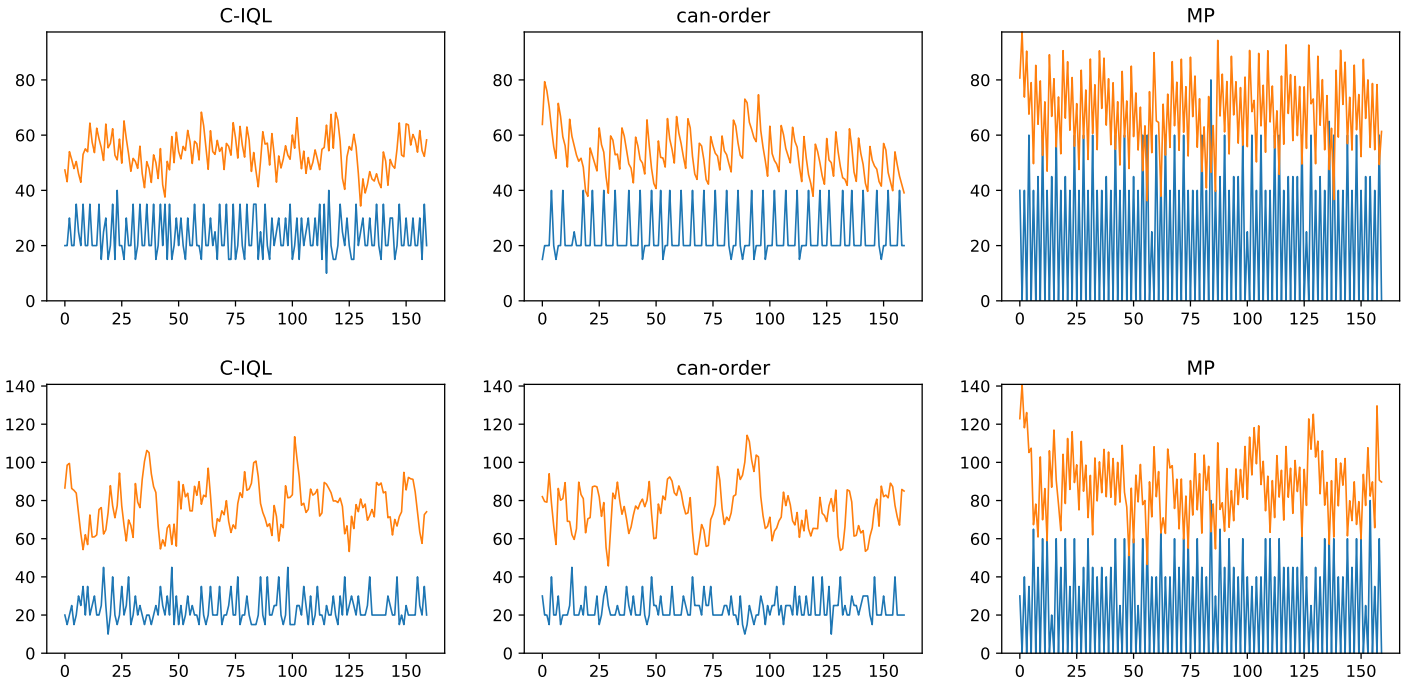

図 3.31　時系列推移例: 取引条件 5 商品数 5

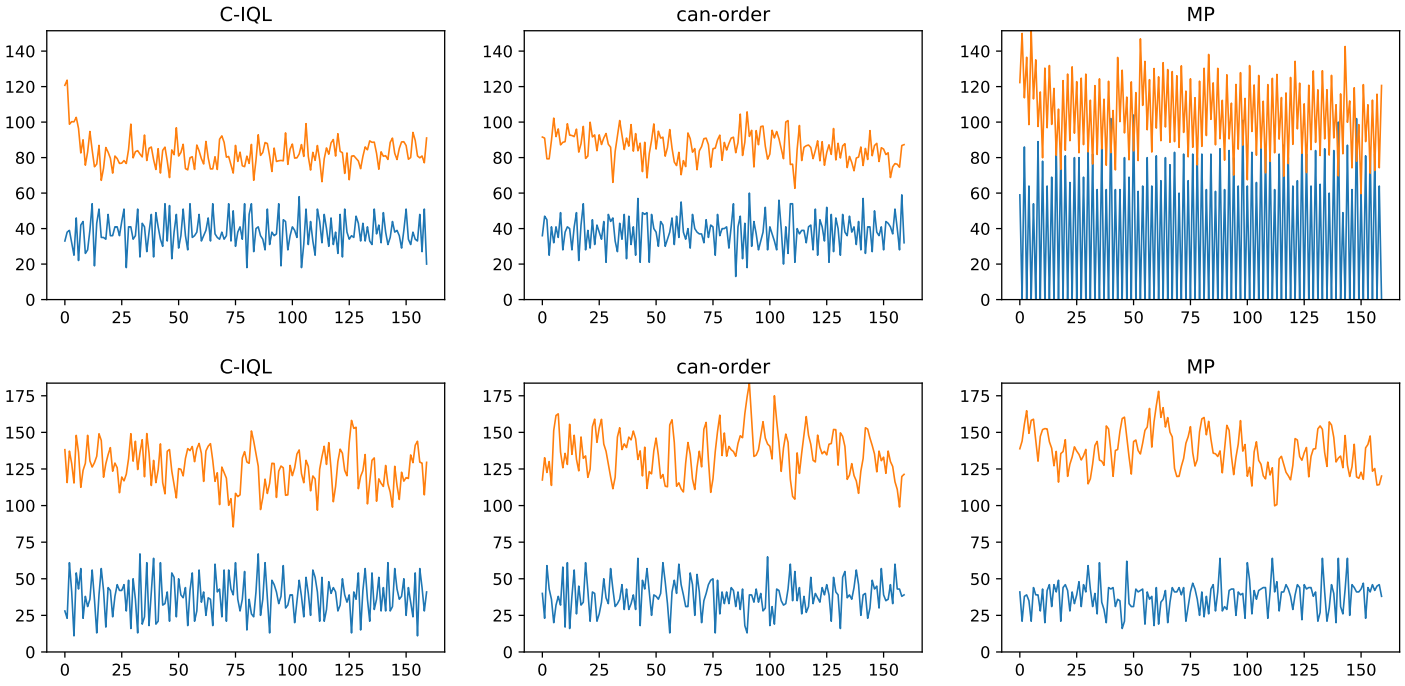

図 3.32　時系列推移例: 取引条件 5 商品数 10

第 4 章

費用推定モデルを用いた動的倉庫容量計画

4.1　はじめに

本章では、多品目在庫システムで確率的な需要のもとでの動的倉庫容量計画を考える。ここで、在庫倉庫には賃貸型倉庫を利用し、賃貸型倉庫では契約容量に対し固定的な費用が課され、その容量は契約からある一定期間経過後に変更可能であり、契約容量を超過した分については在庫量に比例する変動的な費用が課されるものとする。

2.3 節で述べた通り、共通発注費用のもと、確率的需要において補充方策と倉庫容量を同時に求める手法は提案されていない。ただし、3 章で示した手法により、倉庫容量を所与とした場合の補充方策を学習することは可能である。一般に、契約できる賃貸倉庫容量は離散的であるため、現実的にありうる全ての倉庫容量に対して補充方策を学習し、その中で費用が最小となる倉庫容量を選ぶことは可能である。しかし、本論文では動的倉庫容量計画、すなわち倉庫容量が変更可能である状況を考えている。本章では、3 章で説明した補充方策学習器を前提として、動的倉庫容量計画を求める手法を提案する。

動的倉庫容量計画においては、非定常な需要が想定される。本論文では、同一年内では定常な需要だが、年単位では需要水準が確率的に変動する状況を考える。この時、動的倉庫容量計画は逐次意思決定問題であり、マルコフ決定過程で定式化され、状態は需要水準と倉庫容量のみで定められる。需要水準は全商品共通や商品カテゴリ単位などで与えられるが、長期の需要変動であるため、商品単位など細かい粒度での設定は現実的ではない。そのため、ここでのマルコフ決定過程における状態空間は小さく、一般的な価値反復法を用いることで容易に最適な倉庫容量拡張意思決定を導くことが可能である。

しかし、この価値反復法において、全ての状態について即時報酬の情報が必要である。ここで、1 時点は 1 年であり、即時報酬とは 1 年間における利益、すなわち、売上からサプライチェーン費用を差し引いたものとして表される。ここでのサプライチェーン費用として、3 章の手法を用いて、需要水準および倉庫容量を所与とした場合に低費用となる補充方策に基づいて運用した結果としての数値が必要である。価値反復法で解く上で状態空間は大きくないが、全ての状態に対して 3 章の手法で補充方策を学習することは現実的ではない。

そこで、全ての状態に対して補充方策を学習するのではなく、いくつかの需要水準および倉庫容量のもとで学習した補充方策下でのサプライチェーン費用をもとに、他の条件における費用を推定し、その推定値を用いて価値反復法で動的倉庫容量計画を求めることとした。そのため、なるべく少ない学習結果をもとに精度高く費用を推定することが重要である。本論文では、予測分布の分散を明示的に表現できるガウス過程回帰を用いた。

4.2　ガウス過程回帰とサンプリング戦略

本章で用いるガウス過程回帰とサンプリング戦略について説明する。

4.2.1　ガウス過程回帰

(a)　ガウス過程

ガウス過程 [98] は、ノンパラメトリックなカーネルベースの確率モデルであり、未知関数を推定するのに広く用いられている。例としては、三次元空間で鉱物資源量を推定するクリギングや機械学習におけるハイパーパラメータ最適化などがあげられる [99–101]。

ガウス過程とは、平均関数 $m(x)$、共分散関数 $v(x,x')$ によって定義される確率過程で、任意の n 点 $X_n = (x_1, ..., x_n)^T \in \mathcal{V}$ の関数値 $\mathbf{f}(\mathbf{X_n})$ の分布が平均 $\boldsymbol{m}(X_n)$、分散共分散行列 $V(X_n, X_n)$ の多次元ガウス分布に従うものである [9]。ここで、以下の記法を用いた。

$$\boldsymbol{f}\left(X_n\right) = \left(f\left(x_1\right), \ldots, f\left(x_n\right)\right)^{\mathrm{T}} \tag{4.1}$$

$$\boldsymbol{m}\left(X_n\right) = \left(m\left(x_1\right), \ldots, m\left(x_n\right)\right)^{\mathrm{T}} \tag{4.2}$$

$$V\left(X_n, X_n\right) = \begin{pmatrix} v\left(x_1, x_1\right) & \cdots & v\left(x_1, x_n\right) \\ \vdots & \ddots & \vdots \\ v\left(x_n, x_1\right) & \cdots & v\left(x_n, x_n\right) \end{pmatrix} \tag{4.3}$$

ガウス過程の事前分布として、平均関数 $m(x) = m_0(x)$, 共分散関数 $v(x,x') = k(x,x')$ とおく。カーネル関数 $k(x,x')$ の典型的なものに、ガウスカーネル

$$k\left(x, x'\right) = \exp\left(-\beta \left\|x - x'\right\|^2\right) \tag{4.4}$$

がある。図 4.1 にカーネル関数にガウスカーネルを用いて生成した結果を示す。

$\boldsymbol{y} = [y_1, ..., y_N]^T$ が観測されたとき、独立な観測ノイズ $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 I_n)$ を考慮すると、出力データ y の確率密度関数は次のガウス分布で定義される。

$$p(\mathbf{y}) = \mathcal{N}\left(\boldsymbol{m}_0(X_n), \mathbf{K} + \sigma^2 \mathbf{I_n}\right) \tag{4.5}$$

ここで、$\boldsymbol{m}_0(X_n)$ は $m_0(x_i)$ を並べたベクトルであり、K は $k(x_i, x_j)$ を i, j 成分とする行列である。

次に、x^* における $f(x^*)$ については、

$$f(x^*) \sim \mathcal{N}\left[m_0(x^*), k(x^*, x^*)\right] \tag{4.6}$$

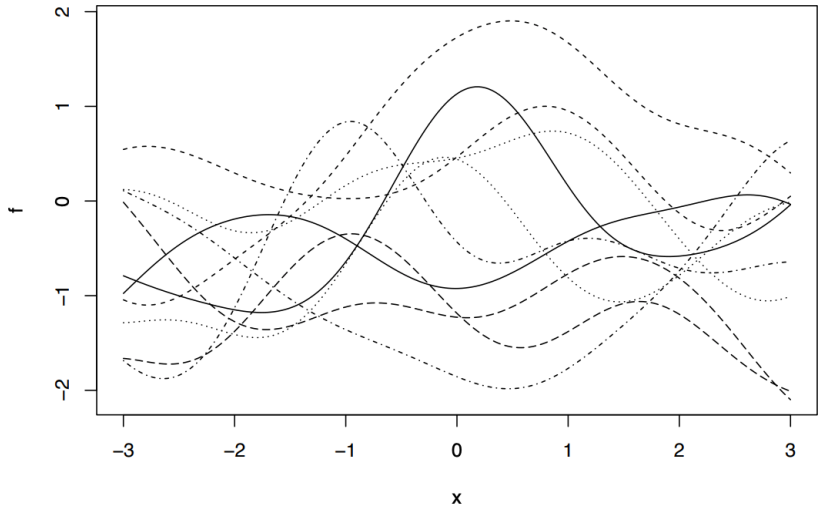

図 4.1　ガウスカーネルを共分散関数として生成したランダムな関数（文献 [9] より抜粋）

であるので、$\boldsymbol{y}, f(x^*)$ の同時分布は、

$$\begin{pmatrix} \boldsymbol{y} \\ f(x^*) \end{pmatrix} \sim \mathcal{N}\left[\begin{pmatrix} \boldsymbol{m}_0(X_n) \\ m_0(x^*) \end{pmatrix}, \begin{pmatrix} K_{n,n} + \sigma^2 I_n & \boldsymbol{k}_n(x^*) \\ \boldsymbol{k}_n(x^*)^{\mathrm{T}} & k(x^*, x^*) \end{pmatrix} \right] \tag{4.7}$$

と表せる。ここで、以下の表記を用いた。

$$K_{n,n} = K\left(X_n, X_n\right) \tag{4.8}$$

$$\boldsymbol{k}_n(x^*) = \left(k\left(x_1, x^*\right), \ldots, k\left(x_n, x^*\right)\right)^{\mathrm{T}} \tag{4.9}$$

ここで、観測データを $\mathcal{D} = \left\{\left(\mathbf{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N} (\mathbf{x} \in \mathcal{V})$、観測済みの入力の集合 $\{\mathbf{x}^{(n)}\}_{n=1}^{N}$ を $\mathcal{S}$ とする。

観測データ $\mathcal{D}$ の元で、未観測の入力 $x^* \in \mathcal{U} = \mathcal{V} \backslash \mathcal{S}$ における y^* の確率 $p(y^*|x^*, \mathcal{D})$ は平均 $\mu_{x^*|\mathcal{D}}$、分散 $\sigma^2_{x^*|\mathcal{D}}$ のガウス分布に従う。ここで、$\mu_{x^*|\mathcal{D}}$、$\sigma^2_{x^*|\mathcal{D}}$ は条件付き多次元ガウス分布の定理により次のように表される。

$$\mu_{x^*|\mathcal{D}} = m_0(x^*) + \Sigma_{x^*\mathcal{S}}\Sigma_{\mathcal{S}\mathcal{S}}^{-1}\mathbf{y} \tag{4.10}$$

$$\sigma^2_{x^*|\mathcal{D}} = k(x^*, x^*) - \Sigma_{x^*\mathcal{S}}\Sigma_{\mathcal{S}\mathcal{S}}^{-1}\Sigma_{\mathcal{S}x^*} \tag{4.11}$$

$\Sigma_{\mathcal{S}\mathcal{S}} = K_{n,n} + \sigma^2 I_n$ は共分散行列、$\Sigma_{x^*\mathcal{S}}$ は要素を $k(x^*, u)$ $(u \in \mathcal{S})$ とする共分散ベクトルであり、$\Sigma_{\mathcal{S}x^*} = \Sigma^T_{x^*\mathcal{S}}$ である。

(b)　ガウス過程回帰におけるパラメータ推定

次に、ガウス過程回帰におけるノイズの分散 σ^2 やカーネル関数に含まれるハイパーパラメータの推定方法について述べる。例えば、平均関数として定数関数 $m_0(x) = \alpha$、カーネル関数にガウスカーネル $k(x, x') = \exp\left(-\beta \|x - x'\|^2\right)$ を考えると、観測ノイズのパラメータ含めて、以下のハイパーパラメータ $\boldsymbol{\theta}$ の推定が必要となる。

$$\boldsymbol{\theta} = \begin{bmatrix} \alpha \\ \beta \\ \sigma^2 \end{bmatrix} \tag{4.12}$$

ガウス過程回帰のハイパーパラメータは、以下の周辺対数尤度を最大になるように推定することができる。

$$L = -\frac{1}{2}(\boldsymbol{y} - \boldsymbol{m}_0(X_n))^{\mathrm{T}} \left(K_{nn} + \sigma^2 I_n\right)^{-1} (\boldsymbol{y} - \boldsymbol{m}_0(X_n)) - \frac{1}{2}\log\det\left(K_{nn} + \sigma^2 I_n\right) - \frac{n}{2}\log 2\pi \tag{4.13}$$

(c)　二乗誤差の推定

任意の点 x^* について、予測分布の分散は式 4.11 で表される通りである。ハイパーパラメータ θ が既知であれば、二乗誤差の下限は予測分布の分散に一致し、

$$\mathrm{MSE}(\hat{\mathrm{f}}(\mathrm{x}^*)) \geq \sigma^2_{\mathrm{x}^*|\mathcal{D}}(\boldsymbol{\theta}) \tag{4.14}$$

である。

しかし、通常はハイパーパラメータ $\boldsymbol{\theta}$ は未知であり、観測したデータから上述の方法などで推定することが多い。その場合には、ハイパーパラメータ θ についての不確かさを考慮する必要がある。観測データから θ の不偏推定量を得た場合には、点 x^* の二乗誤差の下限は HCRB(Hybrid Cramer-Rao Bound) [102–104] を用いて次のように表されることが示されている [105]。

$$\mathrm{MSE}(\hat{\mathrm{f}}(\mathrm{x}^*)) \geq \sigma^2_{\mathrm{x}^*|\mathcal{D}}(\boldsymbol{\theta}) + \mathbf{g}^{\mathrm{T}}\mathbf{M}^{-1}\mathbf{g} \tag{4.15}$$

ここで、$\mathbf{g}, \mathbf{M}$ は以下の通りであり、平均関数、共分散関数における $\boldsymbol{\alpha}, \boldsymbol{\beta}$ はベクトルとして表記している。

$$\mathbf{g} = \frac{\partial}{\partial \boldsymbol{\alpha}}\left(m_\star - \mathbf{m}^\top \mathbf{w}\right) \tag{4.16}$$

$$\mathbf{M} = \frac{\partial \mathbf{m}^\top}{\partial \boldsymbol{\alpha}}\left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1}\frac{\partial \mathbf{m}}{\partial \boldsymbol{\alpha}^\top} \tag{4.17}$$

$$\mathbf{w} = \left(\mathbf{K} + \sigma^2\mathbf{I}\right)^{-1}\mathbf{k}_n(x^*) \tag{4.18}$$

式 4.15 の右辺第二項 $\mathbf{g}^T\mathbf{M}^{-1}\mathbf{g}$ は非負であり、θ の不確かさによる追加的な誤差を表す。また、これは定数関数 $m(x) = \alpha$ を平均関数とした場合でも非ゼロであることが示されている [105]。

4.2.2　ガウス過程を用いた能動サンプリングと受動サンプリング

回帰問題でも分類問題でも、データのアノテーションコストが高い場合、すなわち、ある入力 $\boldsymbol{x}$ に対して出力 y を得るコストが高い場合には、学習モデルの損失だけでなく、観測数についても最小化することを考える必要がある。つまり、なるべく少ないデータからより良いモデルを得ることが重要である。ここで、どの x について観測データを得るかを決める戦略として、受動サンプリング戦略と能動サンプリング戦略がある。受動サンプリング戦略は、学習したモデルと観測データに関係なく、次にどのサンプルを取るかを決定する戦略であり、能動サンプリングは、観測されたデータと学習モデルが与えられた場合に、最も有益であろう次のサンプルを選択する戦略である [106]。

前述の通り、ガウス過程を用いると、新しい入力値 x^* に応じた事後分布の分散を明示的に表現できる。図 4.2 に事後分散の分散に基づく能動サンプリングの様子を図示している。黒丸が観測データであり、太線が事後分布の平均、点線が事後分布の平均 $\pm\sqrt{}$ 分散を示している。左図において事後分布の分散最大の次の観測地点として、得られた値からさらに事後分布を更新した結果が右図であり、分散が大きかった直前の観測地点付近の事後分布の分散が減少していることが分かる。

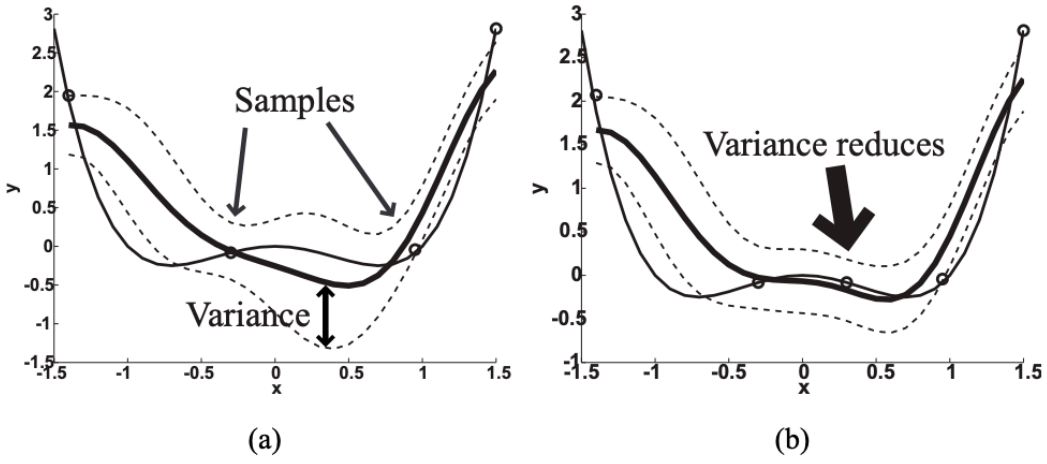

図 4.2　事後分布を用いた能動サンプリング

ガウス過程における観測選択において、観測地点を事前に選択するケースはこれまで多く研究されている [107, 108]。ガウス過程のパラメータが既知の場合には、予測分布の分散は観測したデータに依存しないため、事前に決定できる。しかし、ガウス過程のパラメータが未知の場合には、予測分布の分散はそれまでに観測したデータに依存するため、事前に観測点を決めるよりも、それまでに観測した結果に基づいてその後の観測地点を決める方が望ましい。

本論文は後者に該当するため、観測点をそれまでに観測したデータを用いて逐次的に決める能動サンプリングが有効であると想定される。

以降、能動サンプリング戦略として、エントロピーに基づく戦略と相互情報量に基づく戦略、及び、受動サンプリング戦略について説明する。

(a)　情報量

ここでは、以下の能動サンプリング戦略で用いるエントロピーと相互情報量という二つの情報量について説明する。ある事象 x が起きる確率が $p(x)$ の時、事象 x の自己情報量は次のように表される。

$$i(x) = -\log p(x) \tag{4.19}$$

ここで離散事象系 $A = a_1, a_2, ..., a_n$ を考え、A のいずれか 1 つの事象が実現した時に得られるであろう情報量として平均情報量（エントロピー）は次のように定義される。

$$H(A) = -\sum_{a \in A} p(a) \log p(a) \tag{4.20}$$

なお、連続事象 X の場合には、X のエントロピーは確率密度関数を用いて次のように表される。

$$H(X) = -\int_{-\infty}^{\infty} P_X(x) \log P_X(x) dx \tag{4.21}$$

次に、2 つの離散事象系 $A = a_1, a_2, ..., a_n$ と $B = b_1, b_2, ..., b_m$ を考える。ここで B についての情報が与えられた元で、A のいずれか 1 つの事象が実現した時に得られるであろう情報量を考える。これを条件付き情報量と呼び、次のように定義される。

$$H(A|\ B) = -\sum_{a \in A} \sum_{b \in B} p(a, b) \log p(a|b) \tag{4.22}$$

一般に、B を知るだけでは A の全てを知ることはできないため、次の不等式が成り立つ。

$$H(A) \geq H(A|\ B) \tag{4.23}$$

ここで、$H(A)$ と $H(A|\ B)$ の差分、すなわち、$H(A) - H(A|\ B)$ を相互情報量と呼ぶ。これは、B を知ることによって本来 A を知ることで得られる情報がどれだけ減少するのかを示す。

次に、ガウス分布におけるエントロピーおよび相互情報量について説明する。X が平均 0, 分散 σ^2 の正規分布に従う時、エントロピーは式 4.21 に代入することで以下のように表される。

$$\begin{aligned} H(X) &= -\int_{-\infty}^{\infty} dx P_X(x) \log\left\{\frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-\frac{x^2}{2\sigma^2}}\right\} \\ &= -\int_{-\infty}^{\infty} dx P_X(x) \left\{-\frac{x^2}{2\sigma^2}\right\} + \frac{1}{2}\log\left(2\pi\sigma^2\right) \int_{-\infty}^{\infty} dx P_X(x) \\ &= \frac{1}{2\sigma^2}\int_{-\infty}^{\infty} dx x^2 P_X(x) + \frac{1}{2}\log\left(2\pi\sigma^2\right) \int_{-\infty}^{\infty} dx P_X(x) \\ &= \frac{1}{2}\log\left(2\pi\sigma^2\right) \end{aligned} \tag{4.24}$$

(b)　エントロピーに基づく能動サンプリング戦略

能動サンプリング戦略における最もナイーブものとして、最も不確実な点を選択する、という戦略が考えられる。最も不確実な点というのは、観測データに条件付けされた条件付きエントロピーが最大となる点として表現できる。すなわち、$\mathcal{S}$ についてのデータを観測した時に、未観測地点 $\mathcal{U} = \mathcal{V}\backslash\mathcal{S}$ についてのエントロピー、$H\left(\mathcal{X}_{\mathcal{V}\backslash\mathcal{S}} \mid \mathcal{X}_{\mathcal{S}}\right)$ として表される。この条件付きエントロピーを用いると、例えば、k 回の観測後に条件付きエントロピーを最小化するような観測地点として、次のように選ぶことが考えられる。

$$\mathcal{S}^* = \mathrm{argmin}_{\mathcal{S}\subset\mathcal{V}:|\mathcal{S}|=k} H\left(\mathcal{X}_{\mathcal{V}\backslash\mathcal{S}} \mid \mathcal{X}_{\mathcal{S}}\right) \tag{4.25}$$

ここで、$H\left(\mathcal{X}_{\mathcal{V}\backslash\mathcal{S}} \mid \mathcal{X}_{\mathcal{S}}\right) = H(\mathcal{X}_{\mathcal{V}}) - H(\mathcal{X}_{\mathcal{S}})$ であるので、以下を得る。

$$\mathcal{S}^* = \mathrm{argmin}_{\mathcal{S}\subset\mathcal{V}:|\mathcal{S}|=k} H\left(\mathcal{X}_{\mathcal{V}\backslash\mathcal{S}} \mid \mathcal{X}_{\mathcal{S}}\right) = \mathrm{argmin}_{\mathcal{S}\subset\mathcal{V}:|\mathcal{S}|=k} H(\mathcal{X}_{\mathcal{S}}) \tag{4.26}$$

しかし、式 4.26 は NP 困難であることが知られており [109]、貪欲解法が提案されている [110, 111]。貪欲解法では、$\mathcal{S}_0 = \emptyset$ からはじめて、1 つずつ以下のように最もエントロピーが高い点 x_H^* を次の観測地点として選択する。

$$x_H^* = \arg\max_{x_* \in U}(H(x_*|\mathcal{S})) \tag{4.27}$$

よって、エントロピーに基づくサンプリング戦略は以下のように表される。

$$x_H^* = \frac{1}{2}\log\sigma^2_{x^*|\mathcal{D}} \tag{4.28}$$

この手法の計算コストは低いが、選択された点は探索領域の境界に位置する傾向があり、そのため、必ずしも領域全体のエントロピーの低減につながるとは限らないという課題が知られている [112]。

(c)　相互情報量に基づくサンプリング戦略

上記の問題点を克服するために、相互情報量に基づくサンプリング戦略が提案された。

エントロピーに基づくサンプリング戦略で探索領域の境界に位置する問題は、未観測地点全体の不確かさではなく、その観測地点 1 点のみの不確かさだけを考えていることに由来する。そうではなく、未観測地点の不確かさを最も減らすことのできる地点を次の観測地点に選ぶアプローチが相互情報量に基づくサンプリング戦略である。すなわち、k 個の観測地点を選ぶ際に、未観測地点の情報量を最も減らすように選ぶことを考える。

$$\mathcal{S}^* = \mathrm{argmin}_{\mathcal{S}\subset\mathcal{V}:|\mathcal{S}|=k} H(\mathcal{X}_{\mathcal{V}\backslash\mathcal{S}}) - H\left(\mathcal{X}_{\mathcal{V}\backslash\mathcal{S}} \mid \mathcal{X}_{\mathcal{S}}\right) \tag{4.29}$$

ここで、$H(\mathcal{X}_{\mathcal{V}\backslash\mathcal{S}}) - H\left(\mathcal{X}_{\mathcal{V}\backslash\mathcal{S}} \mid \mathcal{X}_{\mathcal{S}}\right)$ は $\mathcal{S}$ と $\mathcal{V}\backslash\mathcal{S}$ の間の相互情報量 $I\left(\mathcal{S}; \mathcal{V}\backslash\mathcal{S}\right)$ である。

エントロピーに基づくサンプリング戦略と同様に、この最適化問題は NP 困難であることが示されており [101]、貪欲な解法が提案されている。

相互情報量を $MI(\mathcal{S}) = I\left(\mathcal{S}; \mathcal{V}\backslash\mathcal{S}\right)$ とすると、相互情報量に基づいて次の点 x^*_M を決めるサンプリング戦略は次のように表される。

$$x^*_M = \arg\max_{x_*\in\mathcal{U}} MI(\mathcal{S}\cup\{x*\}) - MI(\mathcal{S}) \tag{4.30}$$

$\arg\max[\cdot]$ の中身は、$\mathcal{S}$ と $\overline{\mathcal{S}}$ の条件付きエントロピーの差異であり、次のように表される。

$$MI(\mathcal{S}\cup\{x*\}) - MI(\mathcal{S}) = H(x*|\mathcal{S}) - H(x*|\overline{\mathcal{S}}) \tag{4.31}$$

ここで、$\overline{\mathcal{S}}$ は $\mathcal{V}\backslash(\mathcal{S}\cup\{x*\})$ であり、$-H(x*|\overline{\mathcal{S}})$ の項がエントロピーベースのサンプリング戦略との違いである。条件付きエントロピーの定義とガウス過程の予測分布より、次を得る。

$$H(x*|\mathcal{S}) - H(x*|\overline{\mathcal{S}}) = \Sigma_{x^*\mathcal{S}}\Sigma^{-1}_{\mathcal{S}\mathcal{S}}\Sigma_{\mathcal{S}x^*} - \Sigma_{x^*\overline{\mathcal{S}}}.\Sigma^{-1}_{\overline{\mathcal{S}}\overline{\mathcal{S}}}\Sigma_{\overline{\mathcal{S}}x^*} \tag{4.32}$$

(d)　受動サンプリング戦略

受動サンプリングは、現在の学習モデルに依存せずに、サンプル間の距離が最大になるように次のサンプルを選択する手法である [113]。受動サンプリングにおいては、任意の点の不確かさ $\boldsymbol{x}$ を、次式に表される、観測されたサンプルの中で最も近い点からの距離として考える。

$$u(\boldsymbol{x}, \mathcal{S}) = \min_{\forall \boldsymbol{x}_i\in\mathcal{S}\backslash\boldsymbol{x}} dist\left(\boldsymbol{x}, \boldsymbol{x}_i\right) \tag{4.33}$$

この距離の合計が最小となるように入力の集合を決定すると、

$$\mathcal{S} = \arg\max_{\forall \mathcal{S} \subset \mathcal{D}} \left(\sum_{\forall \boldsymbol{x} \in \mathcal{S}} u(\boldsymbol{x}, \mathcal{S}) \right) \tag{4.34}$$

この方法では、$|S|$ に比例して可能な組み合わせが指数関数的に増加するため、次のように各時間ステップで最も不確実な次のサンプルを順次選択する貪欲なアルゴリズムが提案されている。

$$\arg\max_{\boldsymbol{s}(\in \mathcal{S}_i)} \left(\min_{\forall \boldsymbol{x} \in X} \mathrm{dist}(\boldsymbol{x}, \boldsymbol{s}) \right) \tag{4.35}$$

$\mathcal{S}_i$ は時点 i での観測済みの集合である。

数値実験により、特に観測ノイズが大きい場合には、受動サンプリングの方が能動サンプリングよりも優れる場合があると報告されている。

4.3 問題設定

倉庫容量計画は年次の意思決定サイクルである。概要は 1 章にて示した通りであるが、改めて本論文における倉庫容量計画の問題設定を示す。

同一年内は多次元正規分布 $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ に従う定常需要を想定し、年単位では各商品の需要平均は、3 項モデルに従う変動率により確率的に変動するものとする。この需要変動率の確率質量関数を $\phi_X(d) = p(X = d), d \in (d^1, d^2, d^3)$ と表す。この変動率は商品カテゴリなど一定のグループごとに設定される。本章では、需要水準を $\boldsymbol{l} = (l_1, l_2, ..., l_J) \in \mathcal{L}$、$l_j \in \mathcal{L}_j$ と表す。開始時点における需要水準を $\boldsymbol{l}^1 = (l_1^1, l_2^1, ..., l_J^1)$ として、あるカテゴリ j の t 期までの需要変動率の実現値が $\tilde{d}_1, \tilde{d}_2, ...\tilde{d}_t$ だったとすると、$t+1$ 期でのカテゴリ j の需要水準は、$l_j^1 \prod_{t'}^{t} \tilde{d}_{t'}$ となる。

意思決定者は毎年末に倉庫の拡張判断を行う。倉庫容量の変更単位を δW として、倉庫の拡張判断 $a^{\mathrm{wh}} \in \mathcal{A}^{\mathrm{wh}}$ が決定されると、$a^{\mathrm{wh}}\delta W$ の倉庫容量拡張が即時実行され、翌年の年初から拡張された倉庫容量での運用が始まるものとする。また初期倉庫容量は $a_0^{\mathrm{wh}}\delta W_0, a_0^{\mathrm{wh}} \in \mathcal{A}_0^{\mathrm{wh}}$ であり、a_0^{wh} も動的倉庫容量計画で決定される。

年内はある補充方策に従って運用し、結果としての総費用（輸送費用 + 在庫保管費用 + 機会損失費用）を売上から差し引いたものをその年の利益とする。

倉庫容量計画の目的は、以下の有限期間 T 期までの期待総割引利益の最大化である。この期待総割引利益を現在割引価値（NPV: Net Present Value）と呼ぶ。

$$G = \mathbb{E}(\sum_{t=0}^{T} \gamma^t r_t^*) \tag{4.36}$$

ここで、γ は割引率であり、r_t^* は総費用が最小となる最適な補充方策 π^* に従って運用した場合に得られる利益であり、以下のように表される。

$$r_t^* = \mathrm{Sales}(\boldsymbol{l}_t) - \mathrm{Cost}(\pi^*, \boldsymbol{l}_t, w_t) \tag{4.37}$$

ここで、Sales は売上であり、販売量に販売単価を乗じて決定される金額であるため、需要水準ベクトル $\boldsymbol{l}$ により決定される。各年の補充方策として、ここでは前章で説明した C-IQL による解法により導かれる補充方策を用いることとする。これを $\pi^\star(\cdot|\boldsymbol{l}, w)$ として、以下のように表す。

$$r_t^\star = \mathrm{Sales}(\boldsymbol{l}_t) - \mathrm{Cost}(\pi^\star(\cdot|\boldsymbol{l}_t, w_t)) \tag{4.38}$$

以上より、本章では以下の最大化を目的とした動的倉庫容量計画を導出することを考える。

$$G = \mathbb{E}(\sum_{t=0}^{T} \gamma^t r_t^\star) = \mathbb{E}(\sum_{t=0}^{T} \gamma^t (\mathrm{Sales}(\boldsymbol{l}_t) - \mathrm{Cost}(\pi^\star(\cdot|\boldsymbol{l}_t, w_t)))) \tag{4.39}$$

補充計画の設定としては、コンテナ船輸送、賃貸型倉庫の利用を考えるため、前章の取引条件 5、すなわち階段状の輸送費用及び非線形な在庫保管費用を想定する。

4.4 提案手法

4.3 節の問題設定を踏まえると、年単位での状態遷移は図 4.3 で表されるシナリオツリーとして表現される。ここで、外部環境ノードによる年単位での需要変動は、3 項モデルで表される確率質量関数 $\phi_X(d)$ に従い、倉庫容量の変更は、前年の容量に対して、$a^{\mathrm{wh}}\delta W, a^{\mathrm{wh}} \in \mathcal{A}^{\mathrm{wh}}$ だけ増加する。

ここで、期待総割引利益を最大にする動的倉庫容量計画は、動的計画法を用いて導かれる。ある状態の価値は、その時点での単年の利益に、次の状態の割引価値を加えたものであるので、最適価値関数は次のように表せる。

$$V^*(\boldsymbol{l_t}, w_t) = \max_{a^{\mathrm{wh}} \in \mathcal{A}^{\mathrm{wh}}} \{r_t^\star + \gamma \sum_{\boldsymbol{l}_{t+1}} p(\boldsymbol{l}_{t+1}|\boldsymbol{l}_t) V(\boldsymbol{l}_{t+1}, w_t + a^{\mathrm{wh}}\delta W)\} \tag{4.40}$$

$p(\boldsymbol{l}_{t+1}|\boldsymbol{l}_t)$ は需要水準 $\boldsymbol{l}_t$ から $\boldsymbol{l}_{t+1}$ へ遷移する確率である。通常の投資決定の枠組みでは投資費用が設定されるが、ここでは容量拡張において追加費用のない賃貸契約を想定している。

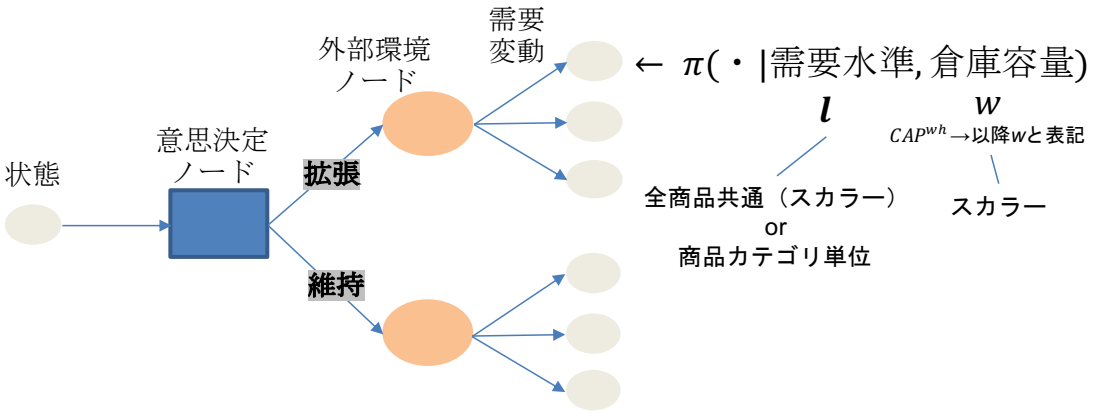

図 4.3　シナリオツリー

上記の最適価値関数から、期待総割引利益を最大化するための各状態における容量拡張決定は、次のように表される。

$$a^{\mathrm{wh}} \leftarrow \mathrm{argmax}_{a^{\mathrm{wh}} \in \mathcal{A}^{\mathrm{wh}}} \{\gamma \sum_{\boldsymbol{l}_{t+1}} p(\boldsymbol{l}_{t+1}|\boldsymbol{l}_t) V(\boldsymbol{l}_{t+1}, w_t + a^{\mathrm{wh}} \delta W)\} \tag{4.41}$$

しかし、シナリオツリーにおける全ての $(\boldsymbol{l}, w)$ について、$\mathrm{Cost}(\pi^\star(\cdot|\boldsymbol{l}, w))$ の情報が必要となるが、全ての条件に対して C-IQL による学習を実行するのは計算量の観点から現実的ではない。

そこで、一部の条件に対してのみ $\pi^\star(\cdot|\boldsymbol{l}, w)$ を学習し、その時の総費用を教師データとして他の条件における $\pi^\star(\cdot|\boldsymbol{l}, w)$ の元での総費用を推定する回帰モデル $\hat{f}(\boldsymbol{l}, w)$ を構築する（図 4.4 に図示する）。回帰モデルの構築においては、得られた推定モデルの精度を評価するための予測誤差の正しい評価と、次にどの条件についての補充方策を学習させるかというサンプリング戦略が重要となる。

費用推定モデルが学習されれば、そのモデル $\hat{f}(\boldsymbol{l}, w)$ を用いてシナリオツリーにおける全ての $(\boldsymbol{l}, w)$ における費用を取得し、動的計画法により動的容量計画を求めれば良い。

以上から、本手法の全体像は図 4.5 のように表される。本手法は、$\mathrm{Cost}(\pi^\star(\cdot|\boldsymbol{l}, w))$ の費用推定モデル $\hat{f}(\boldsymbol{l}, w)$ と、その推定モデルを用いた動的倉庫容量決定モデルから構成される。

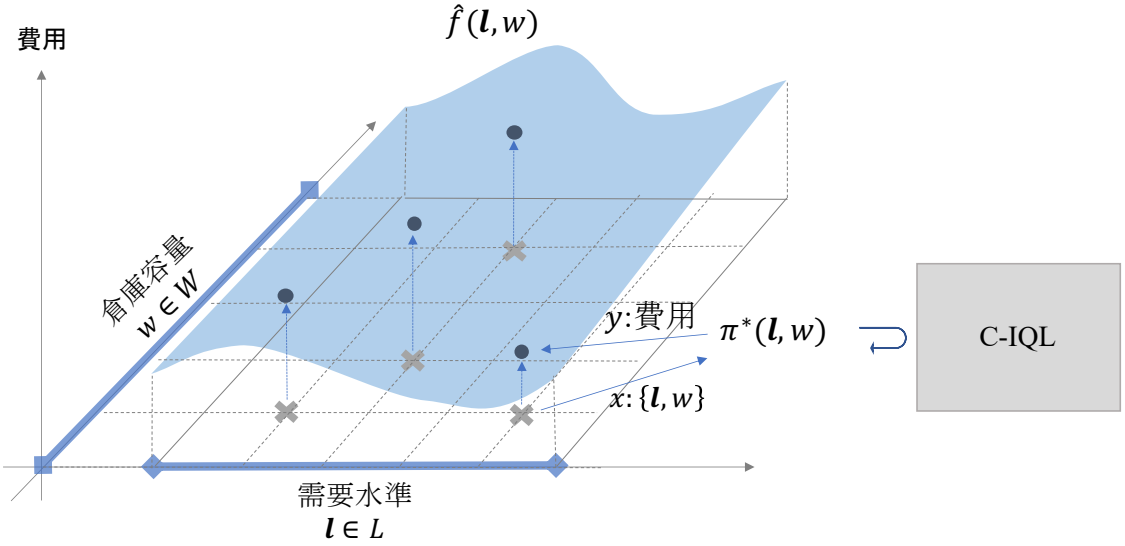

図 4.4　ガウス過程回帰による費用推定

4.4.1　費用推定モデル

費用推定モデルの関心は、なるべく少ないデータで精度の高い費用推定モデルをどのように構築するか、である。そのためには、C-IQL で学習する条件（需要水準と倉庫容量）$\boldsymbol{x} = (\boldsymbol{l}, w)$ をどのように決定するかというサンプリング戦略と、得られた推定モデルによる予測誤差の評価が重要となる。

推定モデルとしてガウス過程回帰を用いる。平均関数としては定数関数 $m(x) = \alpha$、共分散関数はガウスカーネル $k(x, x') = \exp\left(-\beta \|x - x'\|^2\right)$ とし、ハイパーパラメータは周辺尤度最大化により求める。

費用推定モデルの手順を以下に示す。

(1) $\mathcal{W}$、$\mathcal{L}$ をそれぞれ倉庫容量、需要水準の探索空間として、初期条件 $\boldsymbol{x}$ を $\mathcal{V} = \mathcal{W} \bigotimes \mathcal{L}$ からランダムにサンプル

(2) $\boldsymbol{x}$ について C-IQL により補充方策を学習し、得られた費用 $y = \mathrm{Cost}(\pi^\star(\cdot|\boldsymbol{x}))$ を観測データに加え（$\mathcal{D} \leftarrow (\boldsymbol{x}, y)$）、観測済み条件 $\mathcal{S}$ と未観測条件 $\mathcal{U}$ を更新

(3) ガウス過程回帰のハイパーパラメータを更新し、未観測条件 $\mathcal{U}$ についての予測誤差を評価

(4) 終了条件を満たせば終了し、そうでなければ次に学習する条件 $\boldsymbol{x}$ をサンプリング

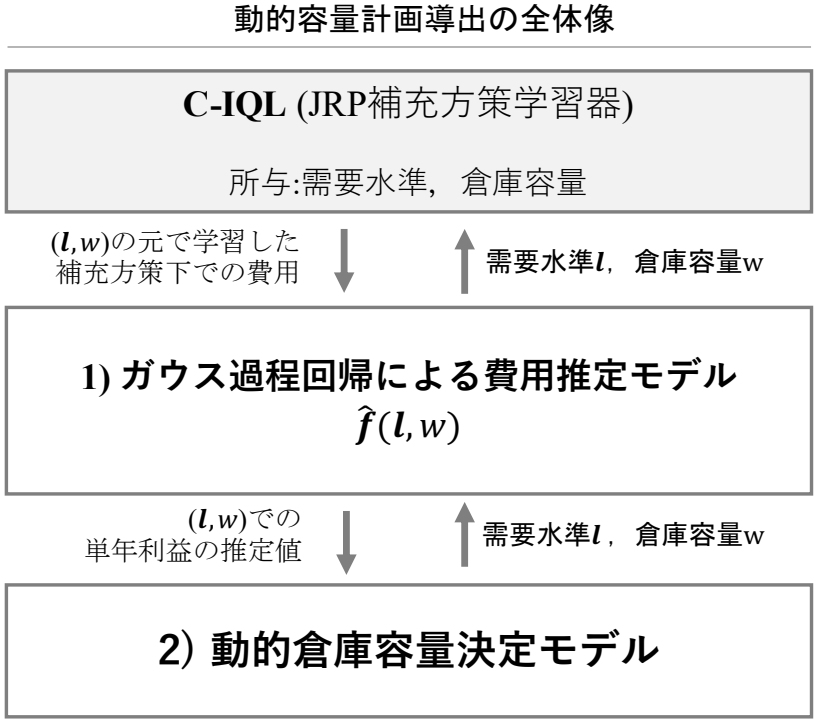

図 4.5 動的容量計画の全体像

戦略により決定し、(2) に戻る

ここで、(2) での C-IQL での補充方策の学習では、強化学習の特性を踏まえ、各 $\boldsymbol{x}$ に対し、3 章と同様に初期乱数シードを変えて 6 回学習し、その平均費用を計算する。

次に、(3) の未観測条件の予測誤差について説明する。需要水準と倉庫容量について、観測済み、すなわち C-IQL による補充方策学習済みの集合を $\mathcal{S} \subset \mathcal{V}$、そうでない集合を $\mathcal{U} = \mathcal{V} \backslash \mathcal{S}$ とする。 この時、未観測の集合についての二乗誤差は、

$$\mathrm{MSE}_{\mathrm{actual}} = \frac{1}{|\mathcal{U}|} \sum_{i=1}^{|\mathcal{U}|} |\hat{f}(x_i) - y_i|^2 \tag{4.42}$$

と表せる。費用推定モデルの目的は、上記 MSE が十分に小さいモデルを得ることである。

しかし、実際の状況では真の値 y は未知である。そのため得られたモデルについての予測誤差を何らかの形で推定する必要がある。一般にガウス過程回帰では、ある点 x_* における予測分布の分散は式 4.11 のように表される。これはガウス過程回帰のハイパーパラメータが既知である場合には二乗誤差期待値の下限に一致する。ハイパーパラメータは本設定では未知であり、式 4.13 で表される対数周辺尤度が最大になるようにハイパーパラ

メータを推定する。ここで、ハイパーパラメータの不確実性を踏まえた二乗誤差の推定値として、式 4.15 で表される HCRB を用いる。

予測誤差の推定精度の指標として、以下に示す $\mathrm{diff}_{\mathrm{RMSE}}$ を導入する。

$$\mathrm{diff}_{\mathrm{RMSE}} = \mathrm{RMSE}_{\mathrm{pred}} - \mathrm{RMSE}_{\mathrm{actual}} \tag{4.43}$$

ここで、$\mathrm{RMSE}_{\mathrm{pred}}, \mathrm{RMSE}_{\mathrm{actual}}$ はそれぞれ実際の RMSE(root mean squared error) と推定 RMSE であり、次のように表される。

$$\mathrm{RMSE}_{\mathrm{actual}} = \sqrt{\frac{1}{|\mathcal{U}|}\sum_{i=1}^{|\mathcal{U}|}|\hat{f}(x_i) - y_i|^2}, \tag{4.44}$$

$$\mathrm{RMSE}_{\mathrm{pred}}^{\mathrm{non-HCRB}} = \sqrt{\frac{1}{|\mathcal{U}|}\sum_{i=1}^{|\mathcal{U}|}\sigma^2_{x_i|\mathcal{D}}}, \tag{4.45}$$

$$\mathrm{RMSE}_{\mathrm{pred}}^{\mathrm{HCRB}} = \sqrt{\frac{1}{|\mathcal{U}|}\sum_{i=1}^{|\mathcal{U}|}(\sigma^2_{x_i|\mathcal{D}} + \boldsymbol{g}^T M^{-1}\boldsymbol{g})}, \tag{4.46}$$

$$\tag{4.47}$$

ここで、$\mathrm{RMSE}_{\mathrm{pred}}^{\mathrm{non-HCRB}}$ は予測分布の分散を二乗誤差推定値とした場合であり、$\mathrm{RMSE}_{\mathrm{pred}}^{\mathrm{HCRB}}$ はハイパーパラメータ θ の不確かさを考慮した HCRB に基づく二乗誤差推定値である。

理想的には、$\mathrm{diff}_{\mathrm{RMSE}}$ は 0 が望ましく、その場合モデルが予測結果に対する予測誤差を正しく評価していることを意味し、その時点での未観測地点における予測誤差を表す $\mathrm{RMSE}_{\mathrm{pred}}$ が十分に小さければ、十分な精度の推定モデルが構築できたと判断できる。

(4) におけるサンプリング戦略、すなわち、どの条件 $\boldsymbol{x} = (\boldsymbol{l}, w)$ を次に学習するかについては、エントロピー及び相互情報量に基づく能動学習と、条件間の距離のみで定まる受動学習の 3 つの戦略を評価対象とする。

4.4.2 動的倉庫容量決定モデル

動的倉庫容量決定モデルでは、倉庫容量計画は将来の需要変動に応じて変化するものであり、ここでの関心は計画そのものではなく、倉庫容量を変更できる柔軟性の経済価値にある。特に共通発注費用が存在する多品目在庫システムにおいては、確率的需要のもとで動的容量計画を求める従来研究はなく、賃貸型倉庫の容量変更の柔軟性が持つ経済価値は明らかになっていない。そこで、倉庫容量変更の柔軟性についていくつかの倉庫契約シナリオを想定し、倉庫容量変更の柔軟性の経済価値を定量的に評価する。

倉庫契約シナリオとしては、以下の 3 つを考える。

(1) 固定的シナリオ: 容量拡張は初期のみ可能

(2) 定期的シナリオ: 容量拡張は定期的 (N 期に 1 回) なタイミングのみで可能

(3) 適応的シナリオ: 容量拡張は前回の変更から N 期経過後の任意のタイミングで可能

ここで、定期的シナリオと適応的なシナリオの差異は、容量変更の延期オプションの有無である。適応的シナリオでは、N 期経過後任意のタイミングで容量変更の実行を行使できるため、延期オプションを保持していることになり、より高い事業価値が期待できる。

動的倉庫容量決定モデルでは、3 項格子モデルで表現した需要変動の元で、期待割引価値の最大化を目指す。その際に、需要水準と倉庫容量で定まる各状態における $\mathrm{Cost}(\pi^\star(\cdot|\boldsymbol{l}_t, w_t))$ として、ガウス過程回帰による推定値を用いる。

動的計画法により最適な倉庫容量計画が得られたら、NPV の分布をモンテカルロシミュレーションにより求める。具体的には、3 項格子モデルで表される需要変動に従い、需要変動を 10,000 サンプル発生させ、それぞれの場合の割引現在価値を評価することで NPV の期待値、及び、事業リスクを表す VaR（Value At Risk）を評価する。

4.5 数値実験

4.5.1 検証項目

図 4.6 に、ガウス過程回帰による費用推定モデル、及び、動的倉庫容量決定モデルそれぞれについての検証項目と検証シナリオの概略を示す。

まず、費用推定モデルの検証項目は以下の通りである。

(1) 予測誤差の正しい評価

(2) 適切なサンプリング戦略

これらを検証するためには、予測対象である $\mathrm{Cost}(\pi^\star(\cdot|\boldsymbol{l}, w))$ が必要であり、探索空間のすべての条件について、C-IQL による補充方策の学習が必要となる。そのため、数値実験においては、需要水準 $\boldsymbol{l}$ を 1 次元のスカラーに限定、すなわち、年単位での需要変動は全商品で共通として、倉庫容量、需要水準の探索空間はそれぞれ、$\mathcal{W} = \{\Delta W \times z | z \in \{0, 1, 2, ...z_{max}\}\}$、$\mathcal{L} = \{0.5n | n \in \mathbb{N}, 2L_{min} \leq n \leq 2L_{max}\}$ とした。

次に、動的倉庫容量決定モデルの検証項目を示す。動的倉庫容量決定モデルにおいては、賃貸型倉庫の容量拡張の柔軟性を定量的に評価することが目的である。そこで、容量拡張に関する 3 つの契約シナリオ、容量拡張の柔軟性が存在する場合の契約固定期間、契

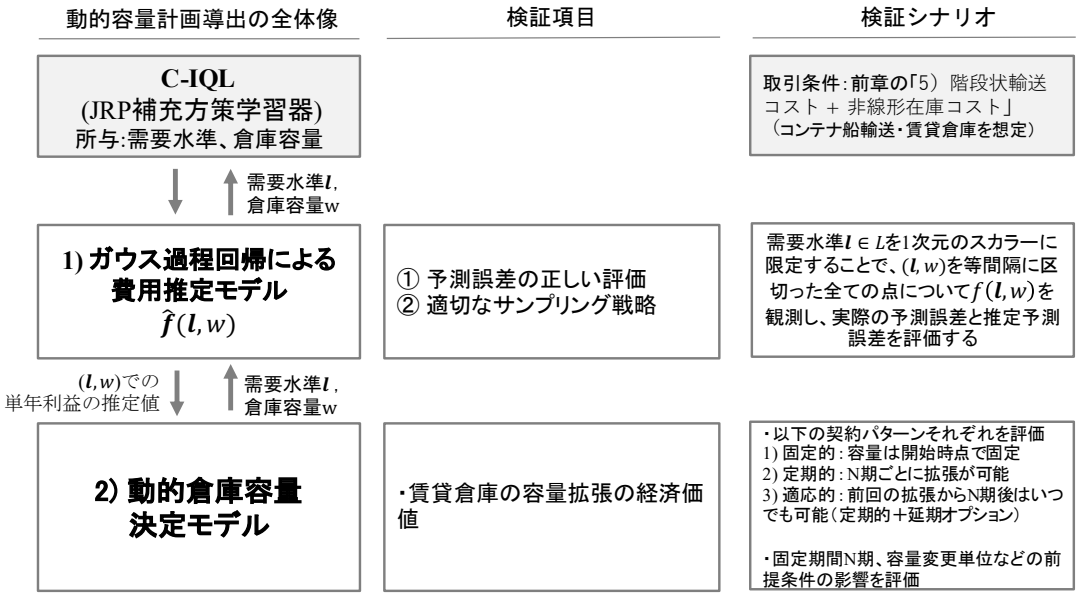

図 4.6　動的容量計画における検証項目と検証シナリオ

約容量変更単位を構成要素として、それぞれの場合の NPV、及び、VaR（Value At Risk）を評価した。検証における構成要素を以下に示す。

（1）契約パターン: 固定的、定期的、適応的

（2）契約固定期間（年）: 2, 3, 4, 5

（3）契約容量変更単位（パレット）δW: 10, 20

4.5.2　実験設定

上記を検証する上で、数値実験を行った。前章における表 3.3 における商品数 5、取引条件 5 の設定を用いている。需要については、需要水準 l=1 における設定を表 3.3 の通りとして、$(c_v, \rho) = (0.2, 0)$ とした。また、輸送費用、在庫保管費用、機会損失費用の単価設定も前章と同一である。

(a)　費用推定モデル

費用推定モデルにおいては、章 4.4 で説明した手順において、初期点をランダムに 2 点選び、20 回のサンプリング、という処理をサンプリング戦略ごとに 10 回試行し、10 回の平均値により評価した。表 4.1 に実験で用いたパラメータ設定を示す。

表 4.1　費用推定モデルにおけるパラメータ設定

ΔW	10
$z_{\max}$	12
$L_{\min}$	0.5
$L_{\max}$	6

(b)　動的倉庫容量決定モデル

表 4.2 に実験で用いたパラメータ設定を示す。

表 4.2　動的倉庫容量決定モデルにおけるパラメータ設定

$\mathcal{A}^{\mathrm{wh}}$	容量変更選択肢	(0, 1, 2)
$\mathcal{A}_0^{\mathrm{wh}}$	初期容量選択肢	(0, 1, 2, 3, 4)
δW_0	初期容量設定単位	10
γ	割引率	1 / 1.05
T	評価期間（年）	10
d	年単位での需要変動率	[0.9,1,1.2]
$\phi(d)$	需要変動率の確率質量関数	[0.2, 0.5, 0.3]
S^0	初期売上	230

4.5.3　数値実験結果

(a)　費用関数推定

図 4.7 に費用関数推定モデルの実験結果を示す。予測誤差の評価については、HCRB を導入することで、ガウス過程回帰の予測分布分散を用いた場合に生じる予測誤差の過小評価を抑えられることを確認した。特に学習初期のデータが十分でない時点では、ガウス過程回帰におけるハイパーパラメータな不確かさが大きいと想定されるが、HCRB において $\boldsymbol{g}^T M^{-1} \boldsymbol{g}$ で表される追加項としてこの不確かさが定量化されていることがわかる。

サンプリング戦略については、本実験では受動学習に対し、能動学習の優位性は観察できなかった。これは実際の予測誤差を観察するために、$(\boldsymbol{l}, w)$ を 2 次元空間に限定したことに起因していると考えられる。能動学習では、観測したデータにより更新されたモデル

から得られる予測分布の分散の情報を用いるが、その推定においてはデータから推定したハイパーパラメータが必要であり、ハイパーパラメータの不確かさが大きい学習初期においてはこの予測分布の分散の推定精度が低いために受動学習に対して良好な結果が得られていない。2 次元空間に限定したことで、少ないサンプル数である程度の精度の推定モデルが得られるため、全体として受動学習に対する優位性が観察されないと考える。

一方で、実務的には商品カテゴリ別など多次元で需要水準を設定する状況も十分に考えられる。その場合、探索空間は高次元になり、能動学習を用いることで、受動学習に比べてより効率的に探索空間全体の推定精度が高められると想定される。学習初期にはモデルの不確かさが大きいことから、HCRB に対する $\boldsymbol{g}^T M^{-1}\boldsymbol{g}$ が一定程度小さくなった時点で能動学習に切り替えるサンプリング戦略が有効であると想定される。

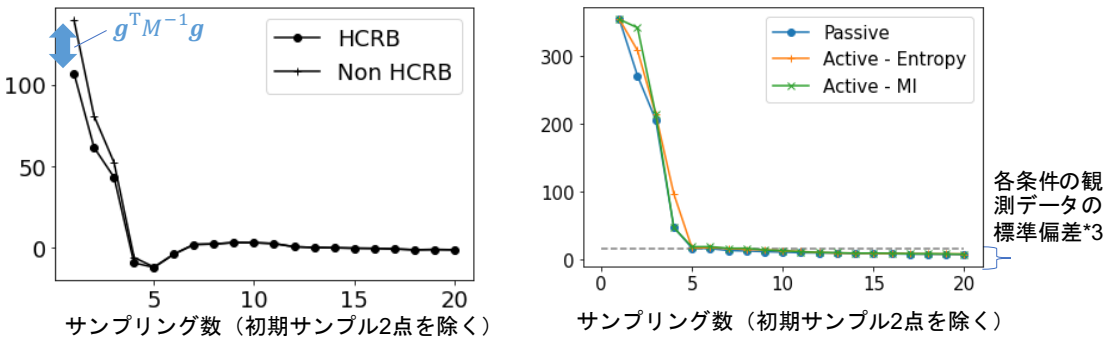

図 4.7　費用推定モデルの評価結果

左：予測誤差の評価（$\mathrm{diff}_{\mathrm{RMSE}}$）（エントロピーベースサンプリング結果）、
右：サンプリング戦略ごとのサンプリング数に対する予測誤差（$\mathrm{RMSE}_{\mathrm{actual}}$）の推移

(b)　動的倉庫容量計画

図 4.8、及び、表 4.3 に結果を示す。図 4.8 は、固定的シナリオに対して、NPV の増加率を示したものであり、表 4.3 はそれぞれのシナリオの NPV の期待値を示す。容量変更単位が小さく、固定期間が短い場合には、定期的な契約と適応的な契約で大きな差異はない一方で、容量変更単位が大きく、固定期間が長い場合、すなわち容量変更の柔軟性が小さくなるに比例して、定期的な契約と適応的な契約での NPV 増加率の差が増大する。これは、容量変更の柔軟性が小さい場合には、適応的な契約の延期オプションの価値が増大することを示している。なお、初期倉庫容量はそれぞれのケースにおいて NPV が最大となる容量を選択している。

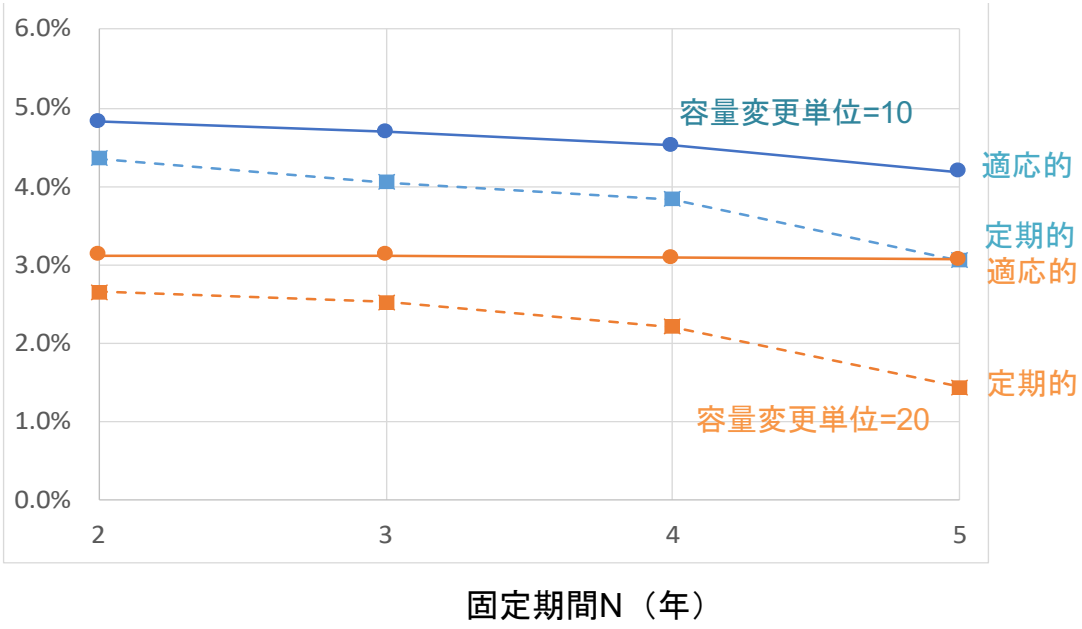

図 4.8　固定的シナリオに対する NPV の増加率

表 4.3　各シナリオの NPV

容量変更単位	契約固定期間	固定的	定期的	適応的
-	-	117.5		
10	2		122.7	123.2
	3		122.3	123.0
	4		122.0	122.8
	5		121.1	122.4
20	2		120.7	121.2
	3		120.5	121.2
	4		120.1	121.2
	5		119.2	121.1

表 4.4、4.5 に容量変更単位がそれぞれ 10、20 の場合の計算結果の詳細を示す。NPV が最大になる初期容量を表中に太字で表す。初期容量の設定は容量拡張が可能である契約シナリオにおいても重要であるが、固定的な契約と比べるとその影響は大きくない。容量が固定される契約では、初期容量を長期の観点から適切に設計しなければ、需要変動に

よっては大きな損失を被る可能性があることが、5%, 10%VaR の値から分かる。一方で定期的もしくは適応的に容量拡張が可能である場合には、仮に初期容量の設定が適切でなかったとしても、その後容量変更を実施することで損失を抑えることが可能である。

4.6　おわりに

共通発注費用の存在する多品目在庫システムにおいては、動的倉庫容量計画の従来研究はなく、倉庫容量の変更の柔軟性が持つ経済価値は明らかではなかった。本章では、倉庫容量の選択肢は離散的であるという仮定のもと、ガウス過程回帰による費用推定モデルを用いた動的倉庫容量決定モデルを構築した。

数値実験の結果、将来の容量拡張オプションを考慮することの経済価値を定量的に示した。費用推定にはガウス過程回帰を用いた。ガウス過程のカーネル関数などのハイパーパラメータが未知である場合に、予測分布の分散では実際の予測誤差を過小評価するが、HRCB を用いることで、特に観測データが十分にない（ハイパーパラメータの不確かさが大きい）場合に、予測誤差の過小評価を抑えることが可能であることを示した。

表 4.4 動的倉庫容量計画の数値実験結果詳細（容量変更単位 10）

シナリオ	固定期間	初期容量	NPV	VaR（5%）	VaR（10%）
固定的	-	0	-106.3	-233.7	-212.6
		10	0.5	-121.0	-100.4
		20	83.5	-53.1	-28.5
		30	**117.5**	**-48.0**	**-18.6**
		40	87.5	-118.1	-82.0
定期的	2	0	66.9	-96.7	-68.6
		10	99.7	-69.1	-39.7
		20	118.1	-51.4	-22.6
		30	**122.7**	**-48.3**	**-19.3**
		40	88.2	-118.7	-80.8
	3	0	37.8	-118.3	-90.7
		10	87.4	-79.0	-49.8
		20	114.9	-52.6	-23.3
		30	**122.3**	**-49.5**	**-19.3**
		40	88.1	-118.7	-81.4
	4	0	7.7	-136.3	-111.5
		10	74.0	-86.2	-58.7
		20	111.5	-52.8	-24.8
		30	**122.0**	**-48.3**	**-18.7**
		40	88.1	-118.6	-82.4
	5	0	-22.7	-154.2	-131.8
		10	59.0	-92.5	-65.9
		20	107.0	-52.2	-25.0
		30	**121.1**	**-48.0**	**-18.1**
		40	87.7	-116.3	-80.3
適応的	2	0	67.3	-97.5	-69.4
		10	100.2	-69.2	-40.4
		20	118.7	-52.6	-23.0
		30	**123.2**	**-47.2**	**-18.5**
		40	88.3	-117.6	-81.5
	3	0	37.8	-117.4	-91.5
		10	87.7	-78.4	-50.5
		20	115.3	-52.4	-24.1
		30	**123.0**	**-48.9**	**-19.3**
		40	88.3	-118.8	-82.4
	4	0	7.8	-137.0	-112.4
		10	74.1	-85.0	-57.3
		20	111.8	-52.4	-24.9
		30	**122.8**	**-48.3**	**-18.9**
		40	88.2	-118.0	-81.7
	5	0	-22.7	-154.6	-132.0
		10	59.0	-92.4	-65.8
		20	107.1	-52.4	-25.2
		30	**122.4**	**-48.4**	**-18.5**
		40	88.2	-118.0	-82.1

表 4.5 動的倉庫容量計画の数値実験結果詳細（容量変更単位 20）

シナリオ	固定期間	初期容量	NPV	VaR（5%）	VaR（10%）
固定的	-	0	-106.3	-233.7	-212.6
		10	0.5	-121.0	-100.4
		20	83.5	-53.1	-28.5
		30	**117.5**	**-48.0**	**-18.6**
		40	87.5	-118.1	-82.0
定期的	2	0	61.4	-99.9	-77.9
		10	97.7	-70.0	-40.1
		20	106.6	-55.8	-33.2
		30	**120.7**	**-48.0**	**-18.0**
		40	87.9	-117.8	-80.8
	3	0	39.2	-117.6	-95.8
		10	85.8	-77.9	-49.4
		20	105.7	-53.4	-30.5
		30	**120.5**	**-48.0**	**-18.7**
		40	87.9	-117.8	-81.9
	4	0	17.5	-140.6	-119.8
		10	73.5	-86.1	-58.0
		20	104.3	-55.8	-34.2
		30	**120.1**	**-48.1**	**-18.9**
		40	87.9	-118.0	-81.2
	5	0	-4.1	-155.5	-135.1
		10	60.7	-91.3	-65.1
		20	102.0	-52.6	-30.9
		30	**119.2**	**-47.9**	**-18.0**
		40	87.5	-117.4	-81.5
適応的	2	0	61.8	-101.0	-77.9
		10	98.2	-69.5	-40.6
		20	107.2	-51.6	-28.8
		30	**121.2**	**-47.9**	**-17.9**
		40	88.1	-117.7	-81.3
	3	0	39.7	-118.1	-95.3
		10	86.3	-77.8	-50.0
		20	107.2	-51.7	-28.9
		30	**121.2**	**-48.3**	**-19.1**
		40	88.0	-117.3	-81.4
	4	0	17.6	-140.5	-119.2
		10	73.7	-85.3	-57.6
		20	106.1	-52.0	-29.5
		30	**121.2**	**-48.7**	**-19.6**
		40	88.0	-119.5	-82.4
	5	0	-4.1	-155.4	-134.5
		10	60.7	-91.9	-65.0
		20	103.6	-52.3	-28.9
		30	**121.1**	**-47.6**	**-18.8**
		40	88.0	-116.5	-81.2

第 5 章

結論

5.1　本研究の主たる成果

本論文では、近年小売業にとって重要性が増している海外調達 PB 商品におけるサプライチェーン費用の低減を動機として、コンテナ輸送及び賃貸倉庫を想定した補充計画、及び、動的倉庫容量計画の解法を提案し、実験的に検証した。補充計画では、同時補充問題に対し、報酬分配に着目したマルチエージェント強化学習手法を提案した。提案手法である C-IQL では、ジョイントアクションの大きな決定空間から効率的に各商品の補充数を探索するヒューリスティック手順を導入し、複数の需要特性、及び、取引条件において、既存の近似方策同等以上のパフォーマンスが達成できることを示した。動的倉庫容量計画に対しては、倉庫容量の選択肢が離散的であるという仮定のもとで、費用関数の推定モデルを用いた解法を提案した。ガウス過程回帰による費用関数推定モデルにおいて、HCRB により二乗誤差を評価することで、より適切に学習の終了判定が可能であることを示した。また、推定した費用関数を用いて、賃貸倉庫の容量拡張の柔軟性が持つ経済価値を定量的に示した。

具体的には、第 1 章では海外調達 PB 商品を想定したサプライチェーンにおける特徴として、FCL によるコンテナ船輸送および賃貸型倉庫利用について説明し、これらが在庫管理においてどのような問題設定につながるのかについて述べた。第 2 章では、本論文が対象とする同時補充問題の一般的な問題設定における従来研究と、倉庫容量計画に関する従来研究について述べた。第 3 章では、同時補充問題に対するマルチエージェント強化学習による解法として、報酬分配に着目し、同時補充問題の特徴である高次元離散行動空間への対処と、協調的な補充方策の学習の両立が達成可能な C-IQL を提案した。数値実験により、在庫保管費用・輸送費用・機会損失費用の総費用の観点で既存のルールベースでの補充方策同等以上の性能を示した。第 4 章では、倉庫容量計画において、倉庫容量が離散的な値を取りうるという仮定のもとで、ガウス過程回帰を用いた費用推定モデルと、推定した費用を用いた動的容量計画決定モデルを提案した。費用推定モデルでは、ガウス過程回帰のハイパーパラメータの不確かさを考慮に入れた HCRB に基づいて予測誤差を評価することにより、より適切に学習の終了判定が可能であることを示した。また、得られた費用推定モデルを用いた動的容量計画決定モデルでは、従来研究では示されてこなかった多品目在庫システムにおいて賃貸型倉庫が持つ柔軟な容量変更による経済価値を示した。

5.2　今後の課題

今後の課題としては、補充計画については、取引条件の観点でより汎用的な解法の確立が挙げられる。本論文では、確率的需要の同時補充問題においてこれまで考慮されてこなかった、階段状の輸送費用は非線形な在庫保管費用を考慮することのできる解法を提案した。しかし、実務での多様な取引条件に対し、本論文で実証できた取引条件は限定的であり、汎用的な解法とまでは言えない。今後は、本論文での提案手法 C-IQL が、ボリュームディスカウントなどの他の取引条件に対しても有効かどうかの検証が求められる。ジョイントアクションの決定にヒューリスティック手順を適用している以上、他の取引条件で有効かどうかは評価が必要である。

また、実務で本提案手法の方策を用いようとしたときの企業の受容性や実装上の課題も整理が必要と考える。特に、方策の解釈容易性が既存のルールベースの方策に比べて劣ることは確かであり、実務での検証を通して具体的に発生しうる課題の整理が必要である。

さらに、本手法を実務に適用する場合にはシミュレーションと実世界との差異が課題となる。シミュレーションを通じたオフラインでの学習において想定していなかった状況に対しても適切な補充数を決定するためには、実行時に局所的な探索を加えることが考えられる。本問題は連続状態空間、高次元離散行動空間の問題であるため、実行時の限られた計算時間における効率的な探索手法が求められ、実務適用に向けた重要な研究課題である。

動的容量計画については、数値実験では評価の観点から、将来需要の推移はすべての商品で同一としたが、実際にはカテゴリ別水準など多次元での評価が必要である。多次元空間を対象とすることで、予測誤差の推定における HCRB の活用や能動サンプリング戦略がより有効になると想定される。

謝辞

本研究を進めるに当たり、懇切なるご指導とご鞭撻を賜りました森川博之教授に深く感謝いたします。研究に関する鋭いご指摘はもちろんのこと、文章の書き方からプレゼンテーション技法まで様々な知見を頂きました。

成末義哲助教には、基本的な論文の作法も含め、多くのことをご教示いただきました。研究者としての視点から、論理的な矛盾について指摘していただいたことで、論文の品質を高められました。

また、本研究の審査にあたり、副査をお引き受け頂くとともに、大変貴重なご助言を頂きました、鶴岡慶雅教授、西成活裕教授、矢入健久教授、柳澤大地准教授に厚く御礼申し上げます。いただいたご指摘により、複数の視点から新たな考察を追加できました。

その他、ここで名前を挙げることはいたしませんが、森川研究室の皆様方、印象的な講義を通じて新たな分野の知見を与えてくださった先生方に感謝いたします。

最後に、これまでの研究生活を支えて頂いた家族に感謝いたします。ありがとうございました。

2020 年 12 月 1 日

参考文献

[1] David Silver. Lecture 2: Markov decision processes. https://www.davidsilver.uk/wp-content/uploads/2020/03/MDP.pdf.

[2] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, Vol. 47, pp. 253–279, 2013.

[3] Luke Metz, Julian Ibarz, Navdeep Jaitly, and James Davidson. Discrete sequential prediction of continuous actions for deep rl. arXiv preprint arXiv:1705.05035, 2017.

[4] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. arXiv preprint arXiv:1705.08926, 2017.

[5] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In AAMAS, pp. 2085–2087, 2018.

[6] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. arXiv preprint arXiv:1803.11485, 2018.

[7] Arash Tavakoli, Fabio Pardo, and Petar Kormushev. Action branching architectures for deep reinforcement learning. In Thirty-Second AAAI Conference on Artificial Intelligence, pp. 4131–4138, 2018.

[8] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. arXiv preprint

arXiv:2003.08839, 2020.

[9] 赤穂昭太郎. ガウス過程回帰の基礎. システム/制御/情報, Vol. 62, No. 10, pp. 390–395, 2018.

[10] 菊池宏之. 小売業におけるPB商品取扱の現状と成果並びに課題. http://www.fmric.or.jp/management/zaimu22/73.pdf.

[11] 国土交通省. 物流をめぐる状況について. https://www.mlit.go.jp/common/001089684.pdf.

[12] Leonardo dos Bastos, Matheus Mendes, Denilson de Nunes, André Melo, Mariana Carneiro, Brasil do de Janeiro, Brasil Vargas, and Brasil do do Pará. A systematic literature review on the joint replenishment problem solutions: 2006-2015. Prod, Vol. 27, No. 0, 2017.

[13] Moshe Kaspi and Meir J Rosenblatt. On the economic ordering quantity for jointly replenished items. The International Journal of Production Research, Vol. 29, No. 1, pp. 107–114, 1991.

[14] Sureh K Goyal. Determination of optimum packaging frequency of items jointly replenished. Management Science, Vol. 21, No. 4, pp. 436–443, 1974.

[15] Moutaz Khouja and Suresh Goyal. A review of the joint replenishment problem literature: 1989–2005. European Journal of Operational Research, Vol. 186, No. 1, pp. 1–16, 2008.

[16] Peter Jackson, William Maxwell, and John Muckstadt. The joint replenishment problem with a powers-of-two restriction. IIE Transactions, Vol. 17, No. 1, pp. 25–32, 1985.

[17] M.A. Hoque. An optimal solution technique for the joint replenishment problem with storage and transport capacities and budget constraints. European Journal of Operational Research, Vol. 175, No. 2, pp. 1033 – 1042, 2006.

[18] Eric Porras and Rommert Dekker. An efficient optimal solution method for the joint replenishment problem with minimum order quantities. European Journal of Operational Research, Vol. 174, No. 3, pp. 1595 – 1615, 2006.

[19] C. Y. Li, J. Gao, T. W. Zhang, and X. T. Wang. Differential evolution algorithm for constraint joint replenishment problem. In 2014 8th International Conference on Future Generation Communication and Networking, pp. 64–67, 2014.

[20] I.K. Moon and B.C. Cha. The joint replenishment problem with resource restriction. European Journal of Operational Research, Vol. 173, No. 1, pp. 190

– 198, 2006.

[21] AMIT EYNAN and DEAN H KROPP. Periodic review and joint replenishment in stochastic demand environments. IIE Transactions, Vol. 30, No. 11, pp. 1025–1033, 1998.

[22] Katsuhisa Ohno, Tomonori Ishigaki, and Toshiaki Yoshii. A new algorithm for a multi-item periodic review inventory system. Zeitschrift Für Operations Res, Vol. 39, No. 3, pp. 349–364, 1994.

[23] K Ohno and T Ishigaki. A multi-item continuous review inventory system with compound poisson demands. Math Method Oper Res, Vol. 53, No. 1, pp. 147–165, 2001.

[24] Joseph L. Balintfy. On a basic class of multi-item inventory problems. Management Science, Vol. 10, No. 2, pp. 287–297, 1964.

[25] Edward A Silver. A control system for coordinated inventory replenishment. International Journal of Production Research, Vol. 12, No. 6, pp. 647–671, 1974.

[26] Edward A Silver. Establishing reorder points in the (s, c, s) coordinated control system under compound poisson demand. The International Journal of Production Research, Vol. 19, No. 6, pp. 743–750, 1981.

[27] Awi Federgruen, H Groenevelt, and Hendrik Cornelis Tijms. Coordinated replenishments in a multi-item inventory system with compound poisson demands. Management Science, Vol. 30, No. 3, pp. 344–357, 1984.

[28] SG Johansen and P Melchiors. Can-order policy for the periodic-review joint replenishment problem. Vol. 54, No. 3, pp. 283–290, 2003.

[29] Derek R. Atkins and Paul O. Iyogun. Periodic versus “can-order” policies for coordinated multi-item inventory systems. Management Science, Vol. 34, No. 6, pp. 791–796, 1988.

[30] S Viswanathan. Note. Periodic Review (s , S) Policies for Joint Replenishment Inventory Systems. Management Science, Vol. 43, No. 10, pp. 1447–1454, 1997.

[31] Haolin Feng, Qi Wu, Kumar Muthuraman, and Vinayak Deshpande. Replenishment policies for Multi - Product stochastic inventory systems with correlated demand and Joint - Replenishment costs. Vol. 24, No. 4, pp. 647–664, 2015.

[32] Stefan Minner and Edward A Silver. Multi-product batch replenishment strategies under stochastic demand and a joint capacity constraint. Vol. 37, No. 5, pp. 469–479, 2005.

[33] Stefan Minner and Edward A Silver. Replenishment policies for multiple products with compound-Poisson demand that share a common warehouse. Vol. 108, No. 1-2, pp. 388–398, 2007.

[34] Marc JG van Eijs. On the determination of the control parameters of the optimal can-order policy. Zeitschrift für Operations Research, Vol. 39, No. 3, pp. 289–304, 1994.

[35] Pricha Pantumsinchai. A comparison of three joint ordering inventory policies*. Decision Sciences, Vol. 23, pp. 111 – 127, 06 2007.

[36] G.P. Kiesmüller. Multi-item inventory control with full truckloads: A comparison of aggregate and individual order triggering. European Journal of Operational Research, Vol. 200, No. 1, pp. 54–62, 2010.

[37] Gilles Cormier and Eldon A. Gunn. A review of warehouse models. European Journal of Operational Research, Vol. 58, No. 1, pp. 3–13, 1992.

[38] Mark Goh, Ou Jihong, and Teo Chung-Piaw. Warehouse sizing to minimize inventory and storage costs. Naval Research Logistics (NRL), Vol. 48, No. 4, pp. 299–312, 2001.

[39] Ming-Jong Yao and Jia-Yen Huang. Optimal lot-sizing and joint replenishment strategy under a piecewise linear warehousing cost structure. Journal of Intelligent Manufacturing, Vol. 28, No. 3, pp. 791–803, 2017.

[40] 中出康一. マルコフ決定過程:理論とアルゴリズム. シリーズ:情報科学における確率モデル 4. コロナ社, 2019.

[41] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

[42] Christopher JCH Watkins and Peter Dayan. Q-learning. Machine learning, Vol. 8, No. 3-4, pp. 279–292, 1992.

[43] Tommi Jaakkola, Michael I Jordan, and Satinder P Singh. Convergence of stochastic iterative dynamic programming algorithms. In Advances in neural information processing systems, pp. 703–710, 1994.

[44] Csaba Szepesvári and Michael L Littman. A unified analysis of value-function-based reinforcement-learning algorithms. Neural computation, Vol. 11, No. 8, pp. 2017–2060, 1999.

[45] John N Tsitsiklis. Asynchronous stochastic approximation and q-learning. Machine learning, Vol. 16, No. 3, pp. 185–202, 1994.

[46] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In European Conference on Machine Learning, pp. 317–328. Springer, 2005.

[47] V Mnih, K Kavukcuoglu, D Silver, AA Rusu, and Veness J Nature. Human-level control through deep reinforcement learning. Nature, 2015.

[48] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. arXiv preprint arXiv:1609.05521, 2016.

[49] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. arXiv preprint arXiv:1511.05952, 2015.

[50] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. arXiv preprint arXiv:1509.06461, 2015.

[51] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In 2017 IEEE international conference on robotics and automation (ICRA), pp. 3389–3396. IEEE, 2017.

[52] Tim De Bruin, Jens Kober, Karl Tuyls, and Robert Babuška. The importance of experience replay database composition in deep reinforcement learning. In Deep reinforcement learning workshop, NIPS, 2015.

[53] Open AI. https://gym.openai.com/envs.

[54] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning, Vol. 8, No. 3-4, pp. 229–256, 1992.

[55] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In Advances in neural information processing systems, pp. 1008–1014, 2000.

[56] Vijaymohan R Konda and Vivek S Borkar. Actor-critic–type learning algorithms for markov decision processes. SIAM Journal on control and Optimization, Vol. 38, No. 1, pp. 94–123, 1999.

[57] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971, 2015.

[58] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. arXiv preprint arXiv:1611.01224, 2016.

[59] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290, 2018.

[60] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. Large-scale interactive recommendation with tree-structured policy gradient. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, pp. 3312–3320, 2019.

[61] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. arXiv preprint arXiv:1708.04782, 2017.

[62] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature, Vol. 575, No. 7782, pp. 350–354, 2019.

[63] Lucian Buşoniu, Robert BabuŠka, and Bart Schutter. A Comprehensive Survey of Multiagent Reinforcement Learning. IEEE Transactions on Systems Man and Cybernetics-Part C: Applications and Reviews, Vol. 38, No. 2, pp. 156–172, 2008.

[64] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In Proceedings of the tenth international conference on machine learning, pp. 330–337, 1993.

[65] Thomas Jansen and R Paul Wiegand. Exploring the explorative advantage of the cooperative coevolutionary (1+ 1) ea. In Genetic and Evolutionary Computation Conference, pp. 310–321. Springer, 2003.

[66] Yu-Han Chang, Tracey Ho, and Leslie P Kaelbling. All learning is local: Multi-agent learning in global reward games. In Advances in neural information processing systems, pp. 807–814, 2004.

[67] David H Wolpert and Kagan Tumer. Optimal payoff functions for members of collectives. In Modeling complexity in economic and social systems, pp. 355–369. World Scientific, 2002.

[68] Tucker Balch. Behavioral diversity in learning robot teams. Technical report, Georgia Institute of Technology, 1998.

[69] Tucker Balch. Reward and diversity in multirobot foraging. 1999.

[70] Kagan Tumer and Adrian Agogino. Distributed agent-based air traffic flow management. In Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, pp. 1–8, 2007.

[71] L. Matignon, G. J. Laurent, and N. Le Fort-Piat. Hysteretic q-learning : an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 64–69, 2007.

[72] Liviu Panait, Keith Sullivan, and Sean Luke. Lenient learners in cooperative multiagent systems. In Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, pp. 801–803, 2006.

[73] Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani. Lenient multi-agent deep reinforcement learning. arXiv preprint arXiv:1707.04402, 2017.

[74] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. JMLR. org, pp. 2681–2690, 2017.

[75] Thomas Gabel and Martin Riedmiller. On a Successful Application of Multi-Agent Reinforcement Learning to Operations Research Benchmarks. 2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, pp. 68–75, 2007.

[76] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In International conference on machine learning, pp. 1995–2003, 2016.

[77] Zheng Sui, Abhijit Gosavi, and Li Lin. A Reinforcement Learning Approach for Inventory Replenishment in Vendor-Managed Inventory Systems With Consignment Inventory. Engineering Management Journal, Vol. 22, No. 4, pp. 44–53, 2015.

[78] P Pontrandolfo, A Gosavi, Okogbaa OGof ⋯. Global supply chain management: a reinforcement learning approach. 2002.

[79] Afshin Oroojlooyjadid, MohammadReza Nazari, Lawrence Snyder, and Martin Takáč. A deep q-network for the beer game: A deep reinforcement learning algorithm to solve inventory optimization problems. arXiv preprint arXiv:1708.05924, 2017.

[80] Taiki Fuji, Kiyoto Ito, Kohsei Matsumoto, and Kazuo Yano. Deep multi-agent reinforcement learning using dnn-weight evolution to optimize supply chain performance. 01 2018.

[81] Lukas Kemmer, Henrik Kleist, Diego Rochebouet, Nikolaos Tziortziotis, and Jesse Read. Reinforcement learning for supply chain optimization. 10 2018.

[82] RS Sutton and AG Barto. Reinforcement learning: An introduction. IEEE Transactions on Neural Networks, Vol. 9, No. 5, pp. 1054–1054, 1998.

[83] Richard Bellman. Dynamic Programming. Dover Publications, 1957.

[84] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In International conference on machine learning, pp. 1928–1937, 2016.

[85] Dan Horgan, John Quan, David Budden, Gabriel Barth-Maron, Matteo Hessel, Hado Van Hasselt, and David Silver. Distributed prioritized experience replay. arXiv preprint arXiv:1803.00933, 2018.

[86] Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In International conference on learning representations, 2018.

[87] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. arXiv preprint arXiv:1802.01561, 2018.

[88] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. In Advances in Neural Information Processing Systems, pp. 7613–7624, 2019.

[89] Nejib Ben-Khedher and Candace A. Yano. The multi-item joint replenishment problem with transportation and container effects. Transportation Science, Vol. 28, No. 1, pp. 37–54, 1994.

[90] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. Journal of Machine Learning Research, Vol. 13, pp. 2171–2175, jul 2012.

[91] BSR. Clean cargo emissions factors 2018 report. https://www.bsr.org/en/our-insights/report-view/clean-cargo-emissions-factors-2018-report.

[92] Ｊ－クレジット制度事務局. Ｊークレジット制度について（データ集）.

[93] 赤倉康寛, 鈴木武, 松尾智征. 我が国貨物の国際国内海上輸送による CO2 排出量の推計. 国土技術政策総合研究所, 2009.

[94] JETRO. 投資関連コスト比較調査.

[95] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In International Conference on Machine Learning, pp. 3053–3062, 2018.

[96] Jerome Andre, Patrick Siarry, and Thomas Dognon. An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization. Advances in engineering software, Vol. 32, No. 1, pp. 49–60, 2001.

[97] Gray Frank. Pulse code communication, March 17 1953. US Patent 2,632,058.

[98] Christopher KI Williams and Carl Edward Rasmussen. Gaussian processes for regression. In Advances in neural information processing systems, pp. 514–520, 1996.

[99] Mark S Handcock and Michael L Stein. A bayesian analysis of kriging. Technometrics, Vol. 35, No. 4, pp. 403–410, 1993.

[100] Carl Edward Rasmussen. Gaussian processes in machine learning. In Summer School on Machine Learning, pp. 63–71. Springer, 2003.

[101] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. Journal of Machine Learning Research, Vol. 9, No. Feb, pp. 235–284, 2008.

[102] Trees Van, K Bell, Z Tiany, et al. Detection estimation and modulation theory. In Detection, Estimation, and Filtering Theory. Wiley & Sons, Inc., 2013.

[103] Yosef Rockah and P Schultheiss. Array shape calibration using sources in unknown locations–part i: Far-field sources. IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 35, No. 3, pp. 286–299, 1987.

[104] H. Messer. The hybrid cramer-rao lower bound - from practice to theory. In Fourth IEEE Workshop on Sensor Array and Multichannel Processing, 2006., pp. 304–307, 2006.

[105] Johan Wagberg, Dave Zachariah, Thomas Schon, and Petre Stoica. Prediction performance after learning in gaussian process regression. In Artificial Intelligence and Statistics, pp. 1264–1272, 2017.

[106] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[107] Sambu Seo, Marko Wallat, Thore Graepel, and Klaus Obermayer. Gaussian process regression: Active data selection and test point rejection. In Mustererkennung 2000, pp. 27–34. Springer, 2000.

[108] Michael C Shewry and Henry P Wynn. Maximum entropy sampling. Journal of applied statistics, Vol. 14, No. 2, pp. 165–170, 1987.

[109] Chun-Wa Ko, Jon Lee, and Maurice Queyranne. An exact algorithm for maximum entropy sampling. Operations Research, Vol. 43, No. 4, pp. 684–691, 1995.

[110] David JC MacKay. Information-based objective functions for active data selection. Neural computation, Vol. 4, No. 4, pp. 590–604, 1992.

[111] Noel Cressie. Statistics for spatial data. John Wiley & Sons, 2015.

[112] Andreas Krause and Carlos Guestrin. Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In Proceedings of the 24th international conference on Machine learning, pp. 449–456. ACM, 2007.

[113] Hwanjo Yu and Sungchul Kim. Passive sampling for regression. In 2010 IEEE International Conference on Data Mining, pp. 1151–1156. IEEE, 2010.

発表文献

論文誌 (査読あり)

[1] H.Suetsugu, Y.Narusue, and H.Morikawa, “Branching Deep Q-Network Agent with Reward Allocation Mechanism for Joint Replenishment Policy,” IPSJ Transactions on Mathematical Modeling and Its Applications, Vol.13, No.2, pp.36-49, August 2020.

[2] H.Suetsugu, Y.Narusue, and H.Morikawa, “Multi-Agent Reinforcement Learning Based Approach for Periodic-Review Joint Replenishment Problem under Practical Cost Structures,” IPSJ Transactions on Mathematical Modeling and Its Applications, to appear.

[3] H.Suetsugu, Y.Narusue, and H.Morikawa, “Two-stage Framework for Long-term Warehouse Capacity Planning for Joint Replenishment Problem,” Journal of Japan Industrial Management Association, submitted.

国際会議における発表

[4] H.Suetsugu, Y.Narusue, and H.Morikawa, “Joint Replenishment Policy in Multi-Product Inventory System Using Branching Deep Q-Network with Reward Allocation,” International Conference on Parallel and Distributed Processing Techniques and Applications, pp.115-121, Las Vegas, USA, July 2019.

[5] H.Suetsugu, Y.Narusue, and H.Morikawa, “A Reinforcement Learning Approach for Joint Replenishment Policy in Multi-Product Inventory System,” Reinforcement Learning for Real Life (RL4RealLife) Workshop in the 36th International Conference on Machine Learning, Long Beach, USA, June 2019.

研究会/全国大会

[6] 末次浩詩，成末義哲，森川博之，“同時補充問題における倉庫キャパシティプラニング 〜 ガウス過程回帰における能動サンプリングのサンプル効率分析 〜,” 電子情報通信学会人工知能と知識処理研究会, AI2019-18, July 2019.

[7] H.Suetsugu, Y.Narusue, and H.Morikawa, “Branching Deep Q-Network Agent for Joint Replenishment Policy,” 情報処理学会数理モデル化と問題解決研究会, 2019-MPS-124(2), July 2019.

[8] 末次浩詩，成末義哲，森川博之，“同時補充問題における需要の不確実性を考慮した倉庫キャパシティ計画 〜ガウス過程回帰によるコスト推定とリアルオプションによる将来の意思決定の柔軟性評価〜,” 日本経営工学会秋季大会, C05, July 2019.

[9] 末次浩詩，成末義哲，森川博之，“同時補充問題における倉庫キャパシティプラニング 〜ガウス過程回帰における能動サンプリングのサンプル効率分析〜,” 電子情報通信学会ソサイエティ大会, A-10-9, July 2019.

[10] 末次浩詩，成末義哲，森川博之，“同時補充問題における長期の倉庫キャパシティ計画,” 日本リアルオプション学会研究発表大会, December 2019.

[11] H.Suetsugu, Y.Narusue, and H.Morikawa, “Periodic-Review Joint Replenishment Policy using Multi-Agent Reinforcement Learning,” 情報処理学会数理モデル化と問題解決研究会, 2020-MPS-130(1), September 2020.