情報処理学会論文誌 Vol.60 No.10 1603–1616 (Oct. 2019)

最短経路探索問題のための動的計画法へのコスト平準化の指標の適用

松井 俊浩[1,a)] マリウス シラギ[2,b)] 平山 勝敏[3,c)] 横尾 真[4,d)] 松尾 啓志[1,e)]

受付日 2019年1月25日, 採録日 2019年7月3日

概要：最短経路探索問題はエージェントの行動決定における基礎的な問題である．従来の問題では，最適経路は部分経路の合計コスト値が最小となる経路として定義され，解法として動的計画法が用いられる．本研究では，経路全体の最大コスト値を改善し，かつコストを平準化することを目的とする経路最適化問題に注目する．このような問題は，各部分経路の近隣住民の不満や施設の消耗の程度を考慮する状況と関係し，実際的な要求を考慮する解のある種の下界を与える点で一定の重要性があると考えられる．このために，多目的最適化問題において公平性を重視する leximin に類する指標を，最短経路探索問題のための動的計画法に適用する．まず，この指標に基づき A*探索アルゴリズムを拡張した解法を提案し，その効果を実験的に評価する．また，未知の環境に適用可能な，探査と学習に基づくインクリメンタルな解法の可能性について考察し，下界に基づく探査を行うオンライン型探索アルゴリズムを直接的に拡張することが困難であることを示す．そして代替案として，探査と学習の段階を繰り返すエピソードに基づく学習に発見的な手法を適用し，完全な知識に基づく A*探索アルゴリズムと同等の解に収束しうることを実験的に示す．

キーワード：公平性，動的計画法，A*，最短経路問題

Applying Criterion of Leveling of Costs to Dynamic Programming for Shortest Path Finding Problems

Toshihiro Matsui[1,a)] Marius C. Silaghi[2,b)] Katsutoshi Hirayama[3,c)] Makoto Yokoo[4,d)] Hiroshi Matsuo[1,e)]

Received: January 25, 2019, Accepted: July 3, 2019

Abstract: Shortest path finding problems are fundamental to behavior-decision processes of agents. In traditional problems, the optimal path is defined as having the minimum total cost value among partial paths. To solve such problems, dynamic programming methods are employed. We focus on the cases where the objective of path optimization is the minimization of the worst-case cost and the leveling of costs among partial paths. Issues considered include the penalty from the complaint of each resident and the lifetime of each facility on partial paths, whose lower bound will be important information in practical problems. To this end, we apply an egalitarian criterion that resembles leximin to dynamic programming methods for shortest path finding problems. We first address the path optimization problem with the new criterion employing a variant of the A* algorithm and empirically evaluate the effect of the algorithm. Then, for unknown environments, the opportunity to develop incremental search methods based on exploration and learning is also investigated and we show that it is hard to directly generalize an existing online search algorithm with the new criterion. Instead of that, we employ an episode-based learning algorithm with heuristic approaches and experimentally show that the proposed solution method is able to achieve the same quality of the solution which is obtained by the extended A* algorithm.

Keywords: fairness, dynamic programming, A*, shortest path problem

[1] 名古屋工業大学
Nagoya Institute of Technology, Nagoya, Aichi 466–8555, Japan
[2] フロリダ工科大学
Florida Institute of Technology, Melbourne FL 32901, United States of America
[3] 神戸大学
Kobe University, Kobe, Hyogo 658–0022, Japan
[4] 九州大学
Kyushu University, Fukuoka 819–0395, Japan
[a)] matsui.t@nitech.ac.jp
[b)] msilaghi@fit.edu
[c)] hirayama@maritime.kobe-u.ac.jp
[d)] yokoo@inf.kyushu-u.ac.jp
[e)] matsuo@nitech.ac.jp

1. はじめに

エージェントの行動経路の最適化は基礎的な問題である．ルート案内，配送サービス，移動ロボットのプランニングなどの多様なタスクが経路の最適化に基づく．一般的な経路最適化問題では，経路上の区間の合計コストを最小化することを目的とする．基本的な経路最適化手法に，最良優先探索と動的計画法に基づく A*アルゴリズムがある [5], [6].

その一方で，実際的な問題においては，各区間のコストやリスクについての最悪ケースや不公平さを平準化することが求められる場合があると考えられる．たとえば，各区間に関わる個々の設備に電池などの寿命があり，それらの消耗の程度を抑制しつつ経路を決定する場合が考えられる．また，区間を通過する際の不満の程度がその近隣の住民ごとに異なり，補償のような財の移動無しにそれらの程度を抑制する際に，先鋭的な不満が生じうる区間を回避する場合が考えられる．このような問題は各区間のコストを目的とする多目的最適化問題ととらえられる．

多目的最適化問題の解を選択する指標として，様々な社会厚生やスカラ化関数が知られている [7], [13]．個々の効用を最大化する多目的最適化問題において，最悪ケースと公平性を考慮する最適化の指標に lexmin がある [2], [4]．leximin は効用値を昇順に整列した目的ベクトルについての辞書順序に基づいて定義される．これは，多目的問題の目的ベクトルをスカラ化して比較する社会厚生の指標に相当する．leximin についての最大化は，最小の利得を最大化し，かつ，公平性をある程度改善する．この leximin についての最適化問題は，動的計画法に基づいて部分問題に分解できる [8], [10]．A*探索アルゴリズムのような経路最適化手法は動的計画法に基づくため，leximin と同様の指標を経路の評価に導入できると考えられる．そこで本研究では，このような指標に基づき，各区間の最悪ケースと公平性を改善しつつコストを最適化する経路探索手法を提案する[*1]．

本研究の貢献は大別して次の 2 点である．最初の貢献は，部分経路コストの平準化を考慮する指標に基づく A*アルゴリズムの拡張である．ここでは，経路最適化問題を 2 次元平面上の重み付き無向グラフで表現するものとする．本研究で対象とする問題は，経路に使用する辺に正数のコストが生じ，未使用の辺のコストが 0 であるとしたときに，leximin に類する指標である *leximax* に基づいてコストを削減することにより，すべての辺のコストについてのコストの平準化を優先し，かつそのときの最短経路を求める問題である．ただし，従来研究で対象としている無制約の組合せ最適化問題とは異なり，ある経路に対応する解を求める．

従来研究で示されているように，leximin に基づく組合せ探索問題に動的計画法の適用が可能である [8], [10] ことから，leximax の指標に基づく経路最適化問題を解くために，動的計画法に基づく最短経路問題の解法を拡張して適用する．そのためには，部分問題への分割に対応した部分的なコストを表現する目的ベクトルが必要である．

問題についての完全な知識がある場合は，経路に含まれない未使用の辺のコストを 0 として，すべての辺に対する固定長の目的ベクトルに含めることにより，経路を leximax により比較できる．また，動的計画法における部分問題すなわち部分経路のコストにも，未使用の辺のコストを 0 としたベクトルをつねに用いることはできる．しかし，部分経路ごとに使用または未使用の辺の数は異なりうるため，それらの部分的なコストのベクトルを結合する際に，未使用の辺の数を毎回考慮してベクトル長を調整しなければならず，冗長な操作が必要となる．さらに，後述する探索と学習に基づく解法では，初期状態では経路の総数が未知であるために未使用の辺の数の予測を目的ベクトルに用いない．

そこで，未使用の辺のコストを省略して表現する，より一般化された可変長の目的ベクトルを導入する．経路に含まれない辺のコストを目的ベクトルから除外する表現は，経路最適化問題のための動的計画法に適用する表現として，直観的であり自然な表現と考えられる．一般に辺の総数は多いため，コストのベクトルに含まれる値を比較的小さい値域の離散値に制限する仮定と，ヒストグラム（もしくはランレングス圧縮）による表現を用いてベクトルのデータ表現の規模を抑制する．このような可変長の目的ベクトルと leximax に基づく指標を用いる A*アルゴリズムの拡張を提案し，その効果を実験的に評価する．

もう 1 つの貢献は，同様のコストの指標を探査と学習に基づく解法に適用する手法についての考察と，初期の発見的な解法の提案である．ここでは，より挑戦的な課題として，動的計画法に基づく探査と学習により段階的に経路を改善するインクリメンタルな解法への同様の指標の適用を試みる．探査と学習に基づく手法は，完全な知識に基づく A*アルゴリズムなどと比較して探索の効率が優れることはないが，エージェントにとって未知の環境において動作し（準）最適解に収束することおよび，その解法は強化学習 [14] の基礎である点でも重要である．コストの下界を優先する探査と学習を同時に行うオンライン型探索の解法として，LRTA*（Learning Real Time A*）アルゴリズム [1] が知られている．しかし，このような解法では，経路のコストの上界は最適性の原理に従って収束する一方で下界は閉路に陥るという，この問題の特殊な性質のために直接的な拡張が困難であることを指摘する．

そして，インクリメンタルな学習への leximax に基づく

[*1] 本研究は文献 [9] をもとに記述を改善し，適用する問題と比較手法および実験結果を補強した．

指標の導入の初期の検討として，探査と学習の段階を反復するエピソードに基づく学習への適用を試み，A*アルゴリズムの拡張と同等の解への収束を目指す．ここでは（準）最適解への収束を意図する発見的な学習則と探査戦略を導入し，実験的に A*アルゴリズムと同等の解への収束が可能であることを示す．

本稿の以降の構成は次のようである．次章では，本研究の背景である，経路探索問題と関連する解法および，部分経路のコストの平準化について述べる．このコストの平準化のために公平性/不平等性の指標を用いる準備として，多目的最適化問題および効用/コストの最悪ケースや不平等性に関する概念について述べる．そして，3 章では拡張された leximax を A*探索アルゴリズムに適用する方法を提案する．4 章では，環境を探査するエージェントのためのアプローチを述べる．5 章では実験により提案手法の基礎的な評価をする．また，議論と結言を 6 章と 7 章に示す．

2. 背景

2.1 経路最適化問題

経路最適化問題は一般に最短経路問題に基づく．本研究では，無向グラフ $G = \langle V, E\rangle$ 上で定義される基礎的な問題を対象とする．

V は頂点の集合，E は無向辺の集合である．$s_i, s_j \in V$ を接続する各辺 $e_{i,j} \in E$ について，関数 $\omega : e_{i,j} \to \mathbb{Z}+$ により，正の整数のコスト値 $w_{i,j} = \omega(e_{i,j})$ が定義される．

パス P は頂点の系列 $(s_1, s_2, \cdots, s_n) \in V^n$ として定義される．ただし，辺 $e_{i,i+1} \in E$ が頂点のペア $s_i, s_{i+1} \in V$ すべてについて存在する．パス P のコストは $\sum_1^{n-1} w_{i,j}$ のように評価される．ある始点 $s_s \in V$ から，ある終点 $s_g \in V$ までのパスのうち，パスのコストが最小となるものが最短経路である．

一般には，辺のコスト値の結合は加算であり，パスに含まれるコスト値の合計が経路のコスト値となる．経路コストが最小となる最適経路を求めることが目的である．

本研究では，コスト値は，たとえば $\{1, 2\}$ or $\{1, \cdots, 10\}$ などの，数個から十個などの十分に少ない数の正数の離散値で表現されるものとする．コスト値は，たとえば各辺に対応する道路を通過する際の付近の住民の不満や，設備の消耗の深刻さなどの程度を表すものととらえられる．

本稿では「頂点」と「ノード」および，「パス」と「経路」を区別せずに用いる．

また，次節の A*アルゴリズムにおける発見的距離関数の議論を簡単にするため，および初期の検討として，**図 1** に示されるような正方形の格子グラフから議論を始める．図の頂点の数字はその名前を表し，辺の数字はそのコスト値を表す．

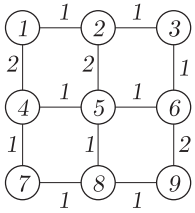


図 1 格子グラフ
Fig. 1 Lattice graph.

2.2 動的計画法に基づく経路探索アルゴリズム

動的計画法に基づく経路探索アルゴリズムには，ダイクストラ法や A*アルゴリズムなどがある．ここではダイクストラ法を含む基礎的な解法として A*アルゴリズムに注目する．A*アルゴリズム [5], [6], [11] は最良優先探索と動的計画法に基づく経路探索アルゴリズムである．まず，経路を始点から終点まで探索しつつ，各頂点についての経路コストの推定値を動的計画法により更新し，終点が発見されればそれまでの情報から経路コストが最小となる最適経路を決定する．探索木においては，各頂点についての経路コストの推定値をもとに，展開する頂点を選択する．各頂点 $s_i \in V$ について，経路コストの推定値 $f(s_i)$ は，始点から頂点 s_i までの経路コスト $g(s_i)$，発見的距離関数に基づく頂点 s_i から終点までの経路コスト $h(s_i)$ の合計として計算される．

$$f(s_i) = g(s_i) + h(s_i). \tag{1}$$

$f(s_i)$（および $g(s_i)$）は動的計画法により更新され，$h(s_i)$ は発見的な値として事前に与えられる．$h(s_i)$ は正しい下界値であれば「許容的」であり最適経路が得られる．格子グラフの場合は，辺のコストの下限値を考慮しつつ，マンハッタン距離やユークリッド距離に基づいて発見的距離関数を定義できる．

アルゴリズムの詳細については文献 [5], [6], [11] などを参照されたい．

2.3 実時間探索アルゴリズム

実時間探索アルゴリズムも，探索と動的計画法に基づく解法である．この解法は，エージェントが経路探索において探査と利用を併用するように設計される．まず，エージェントはたとえば，$s_i \in V$ について $h(s_i) = 0$ のように初期化された最適コストの推定値とともに，始点ノードに置かれる．エージェントは環境中を探査し，各ノードの最適コストの推定値を学習する．これらの推定コスト値に基づき，エージェントはゴールノードを探査する．エージェントがゴールに到着したとき，次の経路探査の試行が開始される．この過程では，各ノード $s_i \in V$ の推定値 $h(s_i)$ が随時更新され，利用される．

このアプローチの重要な点は，動的計画法が，未知の環境におけるエージェントの学習の過程と解釈されることに

ある．適切な探査戦略と学習の規則により，各ノードにおける最適コストの推定値は，試行の繰り返しにおいて改善される．Learning Real-Time A*（LRTA*）[1] は基礎的な実時間探索アルゴリズムである．このアルゴリズムでは，エージェントは各試行の各ノード s_i において次の簡単なステップを実行する．

(1) ノード s_i に隣接するすべてのノード s_j について，エージェントは推定コスト値 $f(s_j) = w_{i,j} + h(s_j)$ を評価する．
(2) エージェントは推定コスト値を $h(s_i) \leftarrow \min_{s_j} f(s_j)$ のように更新する．
(3) エージェントは $s_i \leftarrow \arg\min_{s_j} f(s_j)$ のように次のノードに移動する．

上記の規則では，$h(s_i)$ の値は最適化される推定値であり，閉路から脱出するためのペナルティとしても機能する．アルゴリズムの詳細は，文献 [1] を参照されたい．

2.4 部分経路間の最大コストと公平性

各区間のコストの最大コストと公平性を改善することを考える．図 1 の例において，始点 s_s を 1 の頂点，終点 s_g を 9 の頂点とするとき，従来の合計距離の最小化では，最適経路の 1 つは頂点 1，2，3，6，9 の順であり，その合計コストは 5 である．これに対し，最大コストを改善しつつ経路コストを削減する場合は，最適経路は頂点 1，2，3，6，5，8，9 の順であり，その合計コストは 6 であるが，この経路にはコスト値 2 が含まれない．ここでは，単に 1 つの区間の最大コストを改善するのではなく，全体として各区間のコストを平準化しつつ削減することを目的とする．このような経路は，たとえば各区間に対応する周辺住民や設備の不満や消耗がたびたび尖鋭化する状況を抑制する場合に考慮されうる．

2.5 多目的最適化問題

多目的最適化問題は複数の目的関数の値を同時に最適化することを目的とする．前述のような効用やコストの最悪ケースと公平性をともなう経路最適化問題は多目的最適化問題としてもとらえられる．ここでは次のような多目的最適化問題 MOP を考える．

Definition 1（MOP）． *MOP* は $\langle X, D, F\rangle$ により定義される．X は変数の集合，D は変数の値域の集合，F は目的関数の集合である．変数 $x_i \in X$ は有限離散集合 $D_i \in D$ の要素である変数値をとる．変数の集合 $X_i \subseteq X$ について，関数 $f_i \in F$ は $f_i(x_{i,1}, \cdots, x_{i,k}) : D_{i,1} \times \cdots \times D_{i,k} \to \mathbb{N}$ のように定義される．ただし $x_{i,1}, \cdots, x_{i,k} \in X_i$ である．$f_i(x_{i,1}, \cdots, x_{i,k})$ を簡単に $f_i(X_i)$ と表す．各目的関数の値をある指標のもとで最適化することが目的である．

各目的関数の値の組合せは目的ベクトルにより表される．

Definition 2（目的ベクトル）． 目的ベクトル $\mathbf{v}$ は $[v_1, \cdots, v_K]$ のように定義される．ただしある変数の集合 X_j に対する割り当て $\mathcal{A}$ について，v_j は $v_j = f_j(\mathcal{A}_{\downarrow X_j})$ である．

理想的にはすべての目的関数値についての最大化を目的とする．しかし，一般に目的の間にはトレードオフがあるため，同時には最大化できない．そのため，目的ベクトル間のパレート従属性に基づくパレート最適解を選択する [7], [13].

2.6 Leximin

一般にパレート最適解は複数あり，それらの選択の指標として社会厚生の指標やスカラ化関数が用いられる [7], [13]. 目的ベクトルに含まれる値の合計や不平等性などの，社会厚生の指標やスカラ化関数を用いる場合は，目的ベクトルの値を区別せずに並べ替えることができる．従来の社会厚生である合計 $\sum_{j=1}^{K} f_j(X_j)$ では効率性が考慮される．合計の最大化はパレート最適であるが，公平性は考慮されない．*Maximin* は最小の目的の値 $\min_{j=1}^{K} f_j(X_j)$ を最大化する．これにより最悪の場合の目的を改善するが，パレート最適性は保証されない．パレート最適性を保証するためには合計などによるタイブレークが必要である．また最悪値の改善のみが考慮され，そのほかの目的の値は区別されない．

Leximin は昇順に整列された目的ベクトルについての辞書順に基づく順序関係である [2], [8], [10].

Definition 3（整列された目的ベクトル）． 整列された目的ベクトル $\mathbf{v}$ は値が昇順に整列された目的ベクトルである．

Definition 4（Leximin）． $\mathbf{v} = [v_1, \cdots, v_K]$ と $\mathbf{v}' = [v'_1, \cdots, v'_K]$ を長さ K の整列された目的ベクトルとするとき，$\prec_{leximin}$ による順序は次のように定義される．$\exists t, \forall t' < t, v_{t'} = v'_{t'} \wedge v_t < v'_t$ であり，かつそのときのみ $\mathbf{v} \prec_{leximin} \mathbf{v}'$ である．

すなわち lexmin は最小値による比較を順に繰り返す指標である．Leximin についての最大化は maximin のサブセットであるため最悪の場合を改善する．さらに，ある程度の公平性が考慮され，パレート最適である．

整列された目的ベクトルの加算はベクトルの連結と再整列により定義される．

Definition 5（整列された目的ベクトルの加算）． $\mathbf{v}$ と $\mathbf{v}'$ をベクトル $[v_1, \cdots, v_K]$ と $[v'_1, \cdots, v'_{K'}]$ とするとき，2 つのベクトルの合計 $\mathbf{v} \oplus \mathbf{v}'$ は $\mathbf{v}'' = [v''_1, \cdots, v''_{K+K'}]$ である．ただし $\mathbf{v}''$ の値は $\mathbf{v}$ と $\mathbf{v}'$ に含まれるすべての値からなる．$\mathbf{v}''$ の値は昇順に整列される．

整列された目的ベクトルの加算について次の不変性が成り立つ [8].

Proposition 1（leximin の不変性）． $\mathbf{v}$ と $\mathbf{v}'$ を同じ長さの整列された目的ベクトルとする．また，$\mathbf{v}''$ を別の整列された目的ベクトルとする．$\mathbf{v} \prec_{leximin} \mathbf{v}'$ であるとき，$\mathbf{v} \oplus \mathbf{v}'' \prec_{leximin} \mathbf{v}' \oplus \mathbf{v}''$ である．

この不変性により，最適化問題のための動的計画法を構成できる [8], [10]．ただし，組合せ最適化問題を同じ長さのベクトルについての部分問題に分解することが前提である．

また，整列された目的ベクトルは，目的の値とその数のペアを整列したベクトルとして表現できる [8]．これは，連長圧縮あるいは整列されたヒストグラムに相当する．この表現において leximin の比較およびベクトルの加算を直接的に行うことは容易である．

2.7 タイル尺度

上述の合計や leximin などのいくつかの指標はパレート最適性を満足する．その一方で，不平等性の指標も重要である．後述の節でパスを評価するために，不平等性の指標であるタイル尺度 [3], [12] を用いる．

Definition 6（タイル尺度）． n 個の目的について，タイル尺度 T は次のように定義される．

$$T = \frac{1}{n}\sum_i \frac{v_i}{\bar{v}} \log \frac{v_i}{\bar{v}} \tag{2}$$

ただし，v_i はある目的の効用またはコストであり，$\bar{v}$ はすべての目的についての平均である．

タイル尺度はエントロピーに基づいて定義され，$[0, \log n]$ の値を取る．すべての効用またはコスト値が等しいとき，タイル尺度の値は 0 となる．また，不平等性の尺度の要件を満たし，異なる数のメンバからなる集団を比較するために用いることができる．lexmin の最大化は基本的には最悪値の改善を繰り返して適用するものであるため，タイル尺度をつねに減少させるものではないことに注意されたい．

3. 最大コストと公平性を考慮する経路最適化

区間全体の最大コストと公平性を考慮する最適化問題の解法について検討する．前述の lexmin に基づく最適化問題を動的計画法に基づいて分解できることから，ダイクストラ法や A*アルゴリズムにおけるコスト値の結合を，合計から整列ベクトルの結合に置き換えることを試みる．この最適化のための新たな指標を仮定し，問題は次のように再定義される．

- 「最短」経路問題は重み付き，無向グラフ $G = \langle V, E \rangle$ 上で定義される．集合 V，E，辺の重み，およびパス P は元の問題と同様である．
- パス P のコストは合計ではなく，目的ベクトルとして結合される．コストは leximin に類似する指標により比較される．ただし，この指標は可変長のベクトルにおける最小化問題についての指標である．この指標に基づく最小コストのパスが，始点ノード $s_s \in V$ から終点ノード $s_g \in V$ への最短経路であり，最適経路とする．最適経路が，同一の辺を複数回通ることはない．

組合せ最適化の最大化問題のための手法を，最短経路問題に適用するために，

(1) leximin についての最大化を，同様の指標である *leximax* についての最小化に置き換える，
(2) 異なる長さのベクトルを比較する，

の 2 点が必要である．

3.1 Leximax についての最小化

leximin についての最大化は，最小化問題においては，大小関係が逆である leximax についての最小化に置き換えられる．leximax は leximin と同様に定義されるが，目的ベクトルの値が降順に整列されることが異なる．

Definition 7（降順に整列された目的ベクトル）． 降順に整列された目的ベクトル $\mathbf{v}$ は値が降順に整列された目的ベクトルである．

Definition 8（Leximax）． $\mathbf{v} = [v_1, \cdots, v_K]$ と $\mathbf{v}' = [v'_1, \cdots, v'_K]$ を長さ K の降順に整列された目的ベクトルとするとき，$\prec_{leximax}$ による順序は次のように定義される．$\exists t, \forall t' < t, v_{t'} = v'_{t'} \wedge v_t < v'_t$ であり，かつそのときのみ $\mathbf{v} \prec_{leximax} \mathbf{v}'$ である．

これにより，各目的の最悪の場合の値が最大コスト値となり，最適化問題の目的は利得の最大化からコストの最小化になる．降順に整列された目的ベクトルの加算は leximin と同様であるが，その値が降順に整列される．

3.2 可変長ベクトルの比較

経路最適化問題では，経路長が異なれば目的ベクトル長も異なる．そこで，leximax の定義を経路長の比較に拡張した可変長ベクトル長の leximax（vleximax）を用いる．

Definition 9（Vleximax）． $\mathbf{v} = [v_1, \cdots, v_K]$ と $\mathbf{v}' = [v'_1, \cdots, v'_{K'}]$ を長さ K と K' の降順に整列された目的ベクトルとする．ただし，K と K' は異なる場合がある．$K'' = \min(K, K')$ について，$[v_1, \cdots, v_{K''}] \prec_{leximax} [v'_1, \cdots, v'_{K''}]$ の場合は，$\prec_{vleximax}$ は $\prec_{leximax}$ と同様である．その他の場合は，$K < K' \wedge [v_1, \cdots, v_K] = [v'_1, \cdots, v'_K]$ であり，かつそのときのみ $\mathbf{v} \prec_{vleximax} \mathbf{v}'$ である．さらに，空の目的ベクトル $[\,]$ について一般化し，空ではない目的ベクトル $\mathbf{v}$ に対して $[\,] \prec_{leximax} \mathbf{v}$ である．

たとえば，$[2,1,1,1] \prec_{vleximax} [5,1,1]$，$[2,1,1] \prec_{vleximax} [5,1,1,1]$，$[2,1,1] \prec_{vleximax} [2,1,1,1]$ である．直感的には，この比較は，双方のベクトルを十分な長さとし，不足分をゼロで埋めることに相当する．系に含まれない未使用の経路のコストをゼロとし，無限に長いベクトルを用いることを考える．このとき，leximax による比較はベクトルの先頭からのタイブレークによるため，末尾のゼロの部分列を省略して比較できる．

Vleximax に基づき，最小および最大値関数を次のように定義する．

Definition 10（可変長ベクトルの最小，最大値関数）．
2つの可変長ベクトル $\mathbf{v}$ と $\mathbf{v}'$ に関する最小値関数 $\min(\mathbf{v},\mathbf{v}')$ は，$\mathbf{v} \prec_{vleximax} \mathbf{v}'$ であれば，小さい可変長ベクトル $\mathbf{v}$ を返し，そうでなければ $\mathbf{v}'$ を返す．可変長ベクトルの集合 $\mathbf{V}$ についての最小値関数を，簡単に $\min_{\mathbf{v}\in\mathbf{V}}(\mathbf{v})$ と表す．$\min_{\mathbf{v}\in\mathbf{V}}(\mathbf{v})$ は，$\exists\mathbf{v}'' \in \mathbf{V}, \forall\mathbf{v}' \in \mathbf{V}\setminus\{\mathbf{v}''\}, \mathbf{v}'' \prec_{vleximax} \mathbf{v}'$ であれば $\mathbf{v}''$ を返し，そのような $\mathbf{v}''$ がなければ $\mathbf{V}$ のいずれかの要素を返す．最大値関数 $\max(\mathbf{v},\mathbf{v}')$，$\max_{\mathbf{v}\in\mathbf{V}}(\mathbf{v})$ も，関係 $\mathbf{v} \prec_{vleximax} \mathbf{v}'$ における大きい可変長ベクトル $\mathbf{v}'$ に基づく最大値であることを除いて，同様に定義される．また，$\mathrm{secondmin}_{\mathbf{v}\in\mathbf{V}}(\mathbf{v})$ により，$\mathbf{V}$ の要素で2番目に小さいものを表す．$\min_{\mathbf{v}\in\mathbf{V}}(\mathbf{v})$，$\max_{\mathbf{v}\in\mathbf{V}}(\mathbf{v})$，$\mathrm{secondmin}_{\mathbf{v}\in\mathbf{V}}(\mathbf{v})$ の $\mathbf{v}_{\in}\mathbf{V}$ の代わりに，複数のベクトルの列挙を表す記述を必要に応じて用いる．

また，可変長ベクトルの加算は次のように一般化される．
Definition 11（可変長ベクトルの加算）． 2つの可変長ベクトル $\mathbf{v}$ と $\mathbf{v}'$ の合計 $\mathbf{v}\oplus\mathbf{v}'$ は，それぞれのベクトルの要素数 K と K' が異なりうることを除いて，定義5と同様に定義される．

3.3 発見的距離関数と辺のコストベクトル

A*アルゴリズムにおける発見的距離関数が返す経路コストの推定値 $h(s_i)$ は，真の経路コスト以下の下界値でなければならない．格子グラフについては，辺のコストの下限値とマンハッタン距離に基づいて発見的距離関数を定義できる．vleximax についての最小化の場合は，マンハッタン距離の個数分の，辺のコストの下限値からなるベクトルを，経路コストの下界値とすることができる．一般のグラフに適用する場合には，直接的な発見的距離関数の定義はなく，距離ゼロに相当する，空のベクトルとなる．

簡単な道路網などの表現に用いられる平面直線グラフ，すなわち，2次元平面上で辺が交差せずかつ辺が線分で表現されるグラフの場合を考える．平面直線グラフでは，辺の長さとコスト値を関連付けることが自然と考えられる．ここでは例として，頂点の座標に基づくユークリッド距離として辺の長さが定義される場合に，その辺の長さを切り捨てて丸めた整数の個数のコスト値を連結したベクトルを辺のコストと定義する．たとえば，長さ3でコスト値2の辺のコストを $[2,2,2]$ とする．これは，辺が複数の単位長の部分に近似的に等分され，それぞれに同一のコスト値が与えられることと等価である．より一般化すれば，各部分に異なるコスト値が与えられたベクトルを辺のコストと定義できる．この場合の発見的距離関数はユークリッド距離を切り捨てて丸めた整数値の個数分の，辺のコストの下限値からなるベクトルとすることができる．

3.4 A*アルゴリズムへの可変長目的ベクトルと vleximax の適用

A*アルゴリズムに基づく提案手法では，スカラ値のコストと演算を，可変長の目的ベクトルにより表されるコストとそれらの演算により置換する．他の計算は元の A*アルゴリズムと同様である．

スカラ値のコスト0および辺の重み w に対応するコストは，$[\,]$ および $[w]$ である．コストのベクトルの比較と加算は，定義9および11に従って置換される．

3.5 解法の正しさと複雑さ

A*アルゴリズムによる経路が最適であるためには，発見的距離関数が返す経路コストの推定値 $h(s_i)$ が真の経路コスト以下であればよい．格子グラフにおける vleximax に基づく経路の最小化では，ある頂点から終点までのマンハッタン距離は残りの可能な最小ベクトル長であるから，その数の目的の下限値からなるベクトルは，残りの経路についての下界値である．vleximin による異なる長さのベクトル比較は，真の経路コストのベクトルの比較において，同じ長さの部分までが同一の場合に，ベクトル長すなわち区間数が小さいものを選ぶタイブレークの効果をもたらす．経路コストの推定値は下界値であるから，ベクトルの長さが異なっても誤った経路は最適経路とはならない．したがって，解法は vleximax についての最適な経路を返す．また，合計コストの最小化の場合に推定値 $h(s_i)$ を0とすることができるように，vleximax についての最小化でも単に空のベクトルを $h(s_i)$ として用いても最適経路は得られる．3.3節のユークリッド距離を考慮する発見的距離関数の定義にも同様の議論が成り立つ．

vleximax を用いる場合の計算コストのオーバヘッドは，単にスカラ値の合計と比較に基づく場合よりも顕著に大きいが，多項式的である．実装上は整列された目的ベクトルを連長圧縮表現し，その表現のまま演算できる [8]．特に，コストの離散値の種類が定まっている場合はヒストグラムとしてより簡易的に表現でき，各ベクトルの比較および格納のための時空間複雑度は，コストの離散値の種類が n 個であれば $O(n)$ である．この表現のために配列を用いれば，2つのベクトルの合計や比較はそれぞれのコスト値の数の加算や比較となり，その時間複雑度は $O(n)$ である．その一方で，ギャップが小さい下界としての発見的距離関数の設計は一般には容易ではなく，探索における近傍の展開が，非効率的となる．しかしながら，最悪の場合でもダイクストラ法に上記のオーバヘッドが課される程度である．

3.6 他の指標に基づく最適化

動的計画法により分解できる指標 [8] であれば，経路探索の最適化の指標とすることができる．たとえば，最大コストの最小化を合計コストの最小化によりタイブレークす

る，拡大チェビシェフ関数を適用できる．ただし，この関数の最適化では一般には leximax ほどには公平性の改善は得られない．後述の評価においては，拡大チェビシェフ関数を最適化する場合と比較する．

4. インクリメンタルな最適化

未知の環境に適用可能な，探査と学習に基づくインクリメンタルな最適化手法に注目する．まず，オンライン型探索である実時間探索アルゴリズムが vleximax についても一般化されるかに注目する．しかし，これは閉路のコストの非単調性のために不可能である．この例として，エージェントが乗り越えられない「壁」を含む格子グラフを**図 2** に示す．

図 2 のノード 1 からエージェントが移動を始める場合について例示する．ノード 1 に隣接するノードについて，$h(2) \oplus [w_{1,2}] = [\,] \oplus [1] = [1]$ および $h(4) \oplus [w_{1,4}] = [\,] \oplus [2] = [2]$ である．vleximax と 2.3 節の LRTA*の規則により，エージェントはノード 2 に移動し，$h(1)$ を [1] に更新する．次のステップでは，ノード 2 に隣接するノードについて，$h(1) \oplus [w_{1,2}] = [1] \oplus [1] = [1,1]$，$h(3) \oplus [w_{2,3}] = [\,] \oplus [2] = [2]$，および $h(5) \oplus [w_{2,5}] = [\,] \oplus [2] = [2]$ である．したがって，エージェントはノード 1 に戻り，$h(2)$ を $[1,1]$ に更新する．さらに次のステップでは，ノード 1 に隣接するノードについて，$h(2) \oplus [w_{1,2}] = [1,1] \oplus [1] = [1,1,1]$ および $h(4) \oplus [w_{1,4}] = [\,] \oplus [2] = [2]$ である．したがって，エージェントは再びノード 2 に戻り，コスト値 1 を $h(1)$ と $h(2)$ に加えつつ，この往復を永久に繰り返す．

上記の例は，可変長の整列ベクトルの場合には，異なる探査の方法が必要であることを示している．このようなサイクルがあるパスはベクトル長の閾値の超過により検出することができる．そして，そのような誤ったベクトルを，閉路上の移動を解消するような，何らかのベクトルに置き換えることができる．しかし，そのようなアプローチには，vleximin についての不変性が維持されず，動的計画法の正しさに影響するかもしれないという問題点がある．

4.1 エピソードに基づく方法

ここでは，初期の検討として，どちらかといえば従来の探索アルゴリズムの拡張である，簡単な方法について検討する．そこで，現在得られている情報のみによる動的計画法を反復する他の方法として，エピソードに基づく学習を用いる．エピソードはある探査の試行により得られたパスに相当する．すなわち，学習の段階を探査の段階から分離してオフライン学習の方法をとる．エピソードに基づく学習を用いる提案手法の手順は次のようである．

(1) 初期状態では，始点ノードと始点ノードに接続する辺を除く，他のノードと辺は未知であり，エージェントは探査した範囲のノードとそれらに接続する辺およびそのコストのみを知る．各ノード s_k から終点までの経路の推定コストを $h(s_k)$ とし，その初期値は [ ] である．初期状態から (2) と (3) の手順を反復する．
(2) エージェントを始点に置く．適当な探査戦略によりエージェントは終点までを移動し，エピソードすなわちパスを得る．
(3) 得られたパスに基づいて学習を行い，各ノードの推定コストを更新する．
(4) 収束または試行回数などの適当な終了条件により試行と学習の反復を終了する．

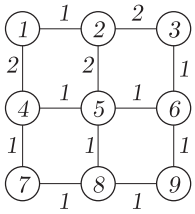


図 2 「壁」がある格子グラフ
Fig. 2 Lattice graph with *walls*.

エピソードとして，完全なパス，すなわち始点から終点までのパスが得られたと仮定する．閉路があるパスは学習の機会を増やすために許容される．

そのパスは終点ノードから最初の始点ノードまで順に，終点ノード以外の対応する推定値 $h(s_k)$ を更新しつつ，走査される．ここで s_k は，あるパス上の最初の始点ノードから k 番目のノードである．

(1) エージェントは $f(s_{k+1}) = [w_{i,j}] \oplus h(s_{k+1})$ を評価する．ただし，$w_{i,j}$ は s_k と s_{k+1} の間の辺 $e_{i,j}$ の重みである．
(2) もしも $h(s_k)$ が更新されていなければ，$f(s_{k+1})$ により更新する．そうでなければ，$\min(f(s_{k+1}), h(s_k))$ により更新する．

上記の更新は終点ノードから順に行われるため，$h(s_k)$ はノード s_k から終点ノードまでの最適コスト値の上界である．アルゴリズムは正確な動的計画法を部分的に実行しているため，エージェントの探査が十分であれば，$h(s_k)$ は最適値に収束する．

4.2 経路コストの境界

上記のエピソードに基づく方法は終点ノードへの完全なパスを必要とするが，適切な探査戦略があれば収束する．他の問題点として，パス上にない近傍ノードの情報を用いていないことがあげられる．また，アルゴリズムは最良優先探索に利用できる下界の状況を把握する仕組みを持たない．

そこで，最適コストの下界 $\underline{h}(s_i)$ と上界 $h(s_i)$ を同時に用いる．境界 $\underline{h}(s_i)$ と $h(s_i)$ は次のように初期化される．

(1) 終点ノードを除いて，$\underline{h}(s_i)$ と $h(s_i)$ はそれぞれ [ ] と

$[\top \cdots \top]$ に初期化される．ただし，$\top$ は系において最大のコスト値である．$h(s_i)$ の初期値は，vleximin において他の目的ベクトルより大きくなるように，十分な数の最大値から構成されなければならない．

(2) 終点ノードでは，$\underline{h}(s_i)$ と $h(s_i)$ の初期値は $[\,]$ である．

終点ノードを除いて，ノード s_i は，その $\underline{h}(s_i)$ と $h(s_i)$ を以下のように更新する．

$$\underline{h}(s_i) \leftarrow \max(\underline{h}(s_i), \min_{s_j}([w_{i,j}] \oplus \underline{h}(s_j))) \quad (3)$$

$$h(s_i) \leftarrow \min(h(s_i), \min_{s_j}([w_{i,j}] \oplus h(s_i))) \quad (4)$$

ただし，s_j は s_i のすべての隣接ノードを表す．ここで，$\underline{h}(s_i)$ と $h(s_i)$ は同時に更新される．

あるパスが探査により得られたとき，次の操作が行われる．

(1) パスは終点ノードから最初の始点ノードまで順に走査される．

(2) 終点ノードを除いて，パス上の各ノード s_i について，対応する $\underline{h}(s_i)$ と $h(s_i)$ を更新する．

上記の境界を用いて，探査と学習の段階は動的計画法の一部を実行する．ここで，$h(s_i) \preceq_{vleximax} \underline{h}(s_i)$ となるとき，s_i のコストが収束したと見なす．従来の合計コストの最小化では，境界はいずれ収束する．しかし，vleximax の場合は収束は保証されない．図 2 における一例を示す．$\underline{h}(3)$，$\underline{h}(4)$ および $\underline{h}(5)$ が $[\,]$，すなわちコストを合計する場合のゼロに相当すると仮定するとき，初期状態から $\underline{h}(1)$ と $\underline{h}(2)$ を交互に更新すると，それらは共に $[\,], [1], [1,1], [1,1,1], \cdots, [1, \cdots, 1]$ のように増加する．これらの 1 を合計する場合は，いずれ他の近傍への辺のコストである 2 を超える．しかし，これらのベクトルは永久にベクトル $[2]$ を超えることはない．同様の状況は，実際の伝搬においても起こりうる．

この状況は前述の LRTA*アルゴリズムを vleximax とともに容易に一般化できないことと同様である．この問題ではコストの上界は従来の最適性の原理に従う一方で，下界は単調に増加するものの，永久に上界に到達せずに増加しつづける場合がありうる．このような下界の振舞いは負閉路に類似する性質ともとらえられる．A*アルゴリズムは距離ゼロのノードからのコストの伝搬に基づき閉路を仮定しないため，この影響を受けない．しかし，LRTA*は純粋な下界駆動によるため，この性質の影響を受け，永久に閉路を移動し続ける場合がある．

そこで，ここでは閉路を含む経路を最適経路としないという仮定に基づき，このような「壊れた」下界を次のように補正する．

(1) $\underline{h}(s_i)$ の長さがある閾値を超えたとき，$\underline{h}(s_i)$ は 2 番目に小さい値 $\underline{f}(s_j)$ に置換される．ここで，$\underline{f}(s_j) = \mathrm{secondmin}_{s_k}([w_{i,k}] \oplus \underline{h}(s_k))$ である．そのような次点の値が「壊れて」いる場合は，境界の情報を修正する情報はただちに得られない．そのときは，何もせずに将来のコスト値の伝搬を待つ．

(2) この変更と伝搬の結果，$h(s_i) \prec_{vleximax} \underline{h}(s_i)$ となる状況が起こりうる．そのときは，$\underline{h}(s_i)$ は $h(s_i)$ に置換される．すなわち，強制的に境界を閉じる．

格子グラフであれば，辺の総数をもとに補正のためのベクトルの長さの閾値を設定できる．平面直線グラフにおいて，辺の長さに応じた数のコスト値からなるベクトルを辺のコストとする場合は，すべての辺の長さの合計をもとに，補正のための長さの閾値を設定できる．

この方法は，正確ではなく，上界が下界よりも速やかに収束することを期待している．ここでは，オフライン学習により距離 0 ノードからの伝搬において上界と下界が同時に更新されるため，一般には上界の収束は下界よりも早いと考えられる．また，下界の補正は，ベクトル長が閾値を超過してから生じるため，さらに収束のペースが落ちると考えられる．この仮定が十分に満足されれば，A*と同様の経路が得られるが，不十分な状況では，誤った解が得られると考えられる．

4.3 発見的な探査

探査戦略は境界とともにすべての解を網羅すべきである．任意の探査戦略が利用できるが，ここでは次のような発見的な探査戦略を用いる．このために，次の情報を各ノード s_i に加える．

- $vstcnt_i$：各探査における，ノードへの訪問回数．各探査が開始されるときにゼロに初期化される．
- $lasttime_i$：ノードに最後に訪問した「時刻」．この情報は最適化の過程を通して維持される．このために，論理時刻を用いる．ある $lasttime_i$ が更新されると論理時刻は 1 増加する．すべての $lasttime_i$ は 0 に初期化される．

同様に，次の情報を各辺に加える．

- $selcnt_{i,j}$：各探査における，辺 $e_{i,j}$ の通過回数．各探査が開始されるときにゼロに初期化される．

これらの情報を用いて，ノード s_j に隣接するノード s_i 上のエージェントに，次の規則が適用される．

(1) s_j が終点ノードであれば，s_j は最も優先される．

(2) s_j と，ある辺 $e_{i,j}$ について，$\underline{h}(s_j) \neq h(s_j)$ かつ $selcnt_{i,j} = 0$ であれば，s_j は 2 番目に優先される．この規則は未探査の辺の先を，その辺のコストが比較的大きくても，1 度は評価することを保証する．

(3) $vstcnt_j$ が小さいノード s_j が 3 番目に優先される．この規則により，エージェントは可能であれば閉路を避ける．

(4) $vstcnt_j$ が等しいとき，$\underline{h}(s_j)$ が小さいノード s_j が 4 番目に優先される．これは最良優先の戦略であり，上

記のルールによりガイドされる．

(5) $vstcnt_j$ と $\underline{h}(s_j)$ がともに等しいとき，$lasttime_j$ が古いノード s_j が 5 番目に優先される．この規則により，コスト値が同等のノードを決定論的に網羅する．

上記の発見的な境界の補正と探査戦略のため，この解法は厳密ではないが，比較的合理的なアプローチであると考える．初期の検討として，簡単なグラフにおける実験的な評価に適用する．このような vleximax のためのガイドされたアプローチの必要性は，このクラスの問題のためのオンライン型探索の設計における課題を示していると考えられる．

5. 評価

5.1 動的計画法に基づく経路探索

提案手法を実験により評価した．まず，vleximax を指標とするように拡張された A*アルゴリズムを評価した．以降の実験ではコスト値のベクトルをヒストグラムとして表現し配列を用いて実装した．

例題として，100×100 頂点の格子グラフを用いた[*2]．また，異なる長さの辺に応じたコスト値ベクトルを適用するための例題として，格子グラフの頂点の座標をランダムに変更した問題（rand-lattice）についても評価した[*3]．

左上の頂点を始点ノード s_s，右下の頂点を終点ノード s_g とした．また，格子グラフにおいてほぼ中央の頂点を始点ノード s_s とする場合（middle）についても評価した．

格子グラフについては，各辺のコスト値として，$[1,2]$，$[1,5]$ および $[1,10]$ の整数値を，一様分布に基づきランダムに選んだ．頂点の座標をランダムに変更した場合（rand-lattice）は，頂点間のユークリッド距離により求めた各辺の長さを整数値 l に丸め，l 個のコスト値からなるベクトルを辺のコストとした．すなわち，ある辺に与えられた重みが w であり，その辺のユークリッド距離に基づく長さが d であるとき，$l = \lfloor d \rfloor$ 個の w からなるベクトル $[w, \cdots, w]$ をその辺のコストとした．辺の重み w を格子グラフの場合と同様にランダムに与えた．格子グラフをランダムに歪めた頂点間の辺の長さ l は辺により異なり，たとえば，ある辺に与えた重みが 2 であり，その辺の長さが 4.3 のときは，その辺のコストは $[2,2,2,2]$ である．

それぞれのパラーメータについて 10 個の例題の結果を平均した．最適化する指標として，合計 ‘sum.’，leximax（vleximax）‘lxm.’，および拡大チェビシェフ関数 ‘awt.’ を用いる解法を比較した．‘awt.’ の解法では，A*アルゴリズムのコスト，加算および比較の演算が次のように置換される．スカラ値のコストの代わりに，コストの最大値と合計のペア (v_{max}, v_{sum}) を用いる．スカラ値のコスト 0 および，ある辺のコスト w に対応するペアは，それぞれ $(0,0)$，(w,w) である．2 つのペア (v_{max}, v_{sum}) と (v'_{max}, v'_{sum}) の合計は $(\max(v_{max}, v'_{max}), v_{sum} + v'_{sum})$ である．2 つのペア (v_{max}, v_{sum}) と (v'_{max}, v'_{sum}) の比較は論理的な表現の拡大チェビシェフ関数に基づき，$v_{max} < v'_{max} \vee (v_{max} = v'_{max} \wedge v_{sum} < v'_{sum})$ であるとき，かつそのときのみ $(v_{max}, v_{sum}) < (v'_{max}, v'_{sum})$ である．すなわち (v_{max}, v_{sum}) の方が小さい．実験には Core i7-3930K CPU（3.20 GHz），16 GB memory，Linux 2.6.32，g++（GCC）4.4.7 の環境を用いた．

格子グラフにおける左上–右下頂点間の経路の，解品質と計算コストを，**表 1** と**表 2** に示す．解品質を，経路コストの合計（sum.），経路中の辺のコストの最小値（min.）および最大値（max.），経路コストのベクトル長（len.），タイル尺度（theil）により評価する．タイル尺度は不公平性の尺度であり，小さい値のほうが不公平さが少ないことを表す．‘lxm.’ はコストの合計とのトレードオフにより，不公平さを削減し，‘sum.’ の場合よりも小さいタイル尺度となった．また，可能な場合に各区間のコストの最大値も抑制された．‘awt.’ は両者の中間的な解を得た．

計算コストは各頂点の近傍を操作した回数（iter.），展開された頂点の数（num. of opn. nodes），実行時間により評価する．表 2 のように，格子グラフとマンハッタン距離に基づく発見的距離関数はさほど効率的ではなく，最大コストを最小化する目的が，枝刈りに適さない性質を持つため，大半の頂点が展開された．‘lxm.’ の場合もわずかに枝刈りの効果が見られる場合があるが，発見的距離関数よりも，A*アルゴリズムが内包する分枝限定法の性質によるところが大きい．‘lxm.’ の実行時間はヒストグラム表現の整列ベクトルと vleximax の計算コストの影響のため，‘sum.’ よりも顕著に大きい．その一方で，計算の増加量はヒストグラムで表されるコスト値の種類の数について，おおむね線形のオーダである．

格子グラフにおける中間–右下頂点間の経路（middle）の，解品質と計算コストを，**表 3** と**表 4** に示す．結果には端点間の最適経路の場合と同様の傾向が見られるが，いずれの解法でも展開された頂点が比較的少ない．その一方で，‘lxm.’ における削減の効果は比較的少なく，これらの結果は，最大コストを最適化する際の効果的な枝刈りの難しさを示していると考えられる．

座標をランダムに変更した格子グラフ（rand-lattice）における，解品質と計算コストを，**表 5**，**表 6** および**表 7** に示す．ここでは辺のコストがベクトルとして定義されることから，コスト値のベクトル長（len.）が経路の辺の数と異なるため，最適経路に含まれる辺の数を別に示した（path）．結果には格子グラフの場合と同様の傾向が見られ

[*2] 予備実験により他の規模の問題でも同様の結果が得られたため，紙幅の都合上，代表的な結果として選んだ．

[*3] 簡易に平面直線グラフを生成するために，まず格子の座標の単位を 100 としてグラフを生成した後，一様分布の乱数に基づき，各頂点の縦横の座標を $[0,49)$ の範囲の整数値により増加または減少した．

表 1 解品質：100 × 100 nodes
Table 1 Solution qualities: 100 × 100 nodes.

cost	alg.	solution quality				
		sum.	min.	max.	len.	theil
[1, 2]	sum.	213.2	1	2	198	0.025
	awt.	214.4	1	2	199.2	0.025
	lxm.	286.9	1	2	282.8	0.006
[1, 5]	sum.	346.1	1	5	199.6	0.132
	awt.	368.1	1	3.6	211.6	0.107
	lxm.	446.7	1	3.6	275.6	0.085
[1, 10]	sum.	580.8	1	9.5	202.8	0.215
	awt.	630.6	1	6.8	215.2	0.167
	lxm.	960.3	1	6.8	350	0.128

表 2 計算コスト：100 × 100 nodes
Table 2 Computational cost: 100 × 100 nodes.

cost	alg.	iter.	num. of opn. nodes	exec. time [s]
[1, 2]	sum.	7217	7537	0.022
	awt.	7008	7333	0.022
	lxm.	8359	8883	0.070
[1, 5]	sum.	9989	9996	0.020
	awt.	9406	9755	0.041
	lxm.	9497	9923	0.078
[1, 10]	sum.	9996	9999	0.019
	awt.	9623	9957	0.041
	lxm.	8468	9182	0.172

表 3 解品質：100 × 100 nodes（middle）
Table 3 Solution qualities: 100 × 100 nodes (middle).

cost	alg.	solution quality				
		sum.	min.	max.	len.	theil
[1, 2]	sum.	108.5	1	2	100	0.027
	awt.	124.6	1	1.8	118	0.020
	lxm.	149.5	1	1.8	147	0.006
[1, 5]	sum.	178.5	1	4.9	100.8	0.132
	awt.	195.7	1	3.4	108.6	0.101
	lxm.	223.2	1	3.4	135	0.087
[1, 10]	sum.	301	1	9	103.2	0.213
	awt.	384.2	1	6.3	130.4	0.151
	lxm.	592.1	1	6.3	214.8	0.130

表 4 計算コスト：100 × 100 nodes（middle）
Table 4 Computational cost: 100 × 100 nodes (middle).

cost	alg.	iter.	num. of opn. nodes	exec. time [s]
[1, 2]	sum.	1993	2182	0.004
	awt.	3350	3905	0.019
	lxm.	7013	7371	0.066
[1, 5]	sum.	4702	4860	0.009
	awt.	6036	6455	0.025
	lxm.	9214	9702	0.105
[1, 10]	sum.	7023	7217	0.016
	awt.	7095	7645	0.046
	lxm.	7759	8402	0.175

る．この問題ではマンハッタン距離ではなくユークリッド距離に基づく発見的距離関数が適用されるため，枝刈りには不利である．また，各辺のコストは，辺の長さに応じた数のコスト値のベクトルとして表現されるため，小さなコスト値の長い辺は，最大コストを最小化する際の探索範囲を拡大すると考えられる．これらにより，‘lxm.’ は大半のノードを展開している．

表 5 解品質：100 × 100 nodes（rand-lattice）
Table 5 Solution qualities: 100 × 100 nodes (rand-lattice).

cost	alg.	solution quality					
		sum.	min.	max.	len.	path	theil
[1, 2]	sum.	20508.3	1	2	18422.5	199	0.034
	awt.	20630.7	1	2	18595.2	200.2	0.033
	lxm.	32316	1	2	32018.3	308.8	0.004
[1, 5]	sum.	33976.1	1	5	19912.9	203.4	0.142
	awt.	37775.9	1	3.6	21631.8	214.8	0.111
	lxm.	50931.7	1	3.6	31593.2	300.8	0.080
[1, 10]	sum.	56950	1	9.9	20217.5	204	0.222
	awt.	65476.8	1	6.8	22501	221	0.170
	lxm.	108888	1	6.8	39783.6	374	0.131

表 6 解品質：100 × 100 nodes（rand-lattice，middle）
Table 6 Solution qualities: 100 × 100 nodes (rand-lattice, middle).

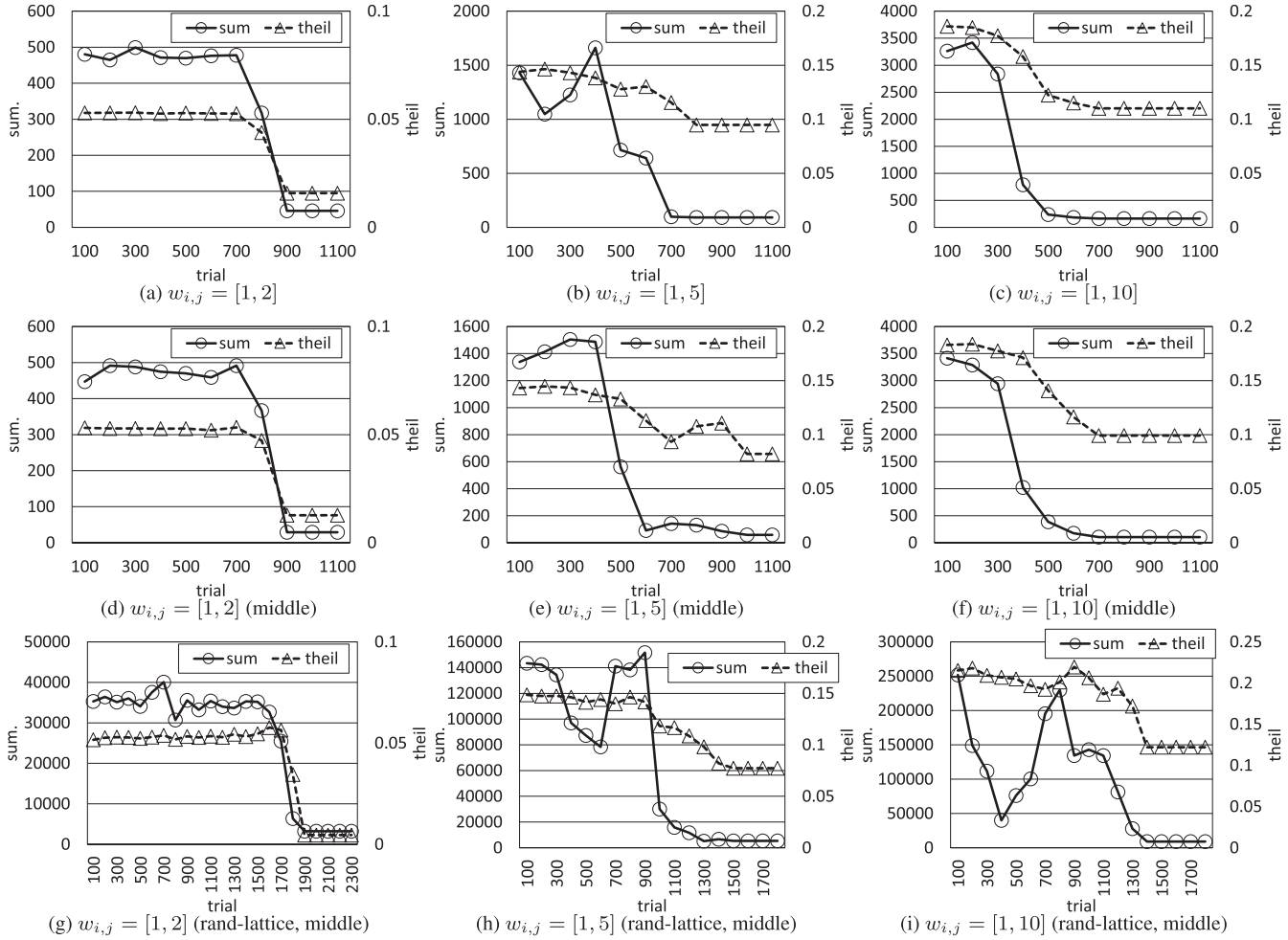
cost	alg.	solution quality					
		sum.	min.	max.	len.	path.	theil
[1, 2]	sum.	10519.8	1	2	9343.3	101	0.036
	awt.	11470.5	1	2	10466.1	108.8	0.030
	lxm.	19502	1	2	19319.8	185.6	0.004
[1, 5]	sum.	17657.1	1	5	9901.3	102.8	0.156
	awt.	20782.4	1	3.5	11480.9	114.4	0.102
	lxm.	27848.9	1	3.5	17232.8	164.4	0.081
[1, 10]	sum.	29986	1	9.9	10167.8	103.8	0.232
	awt.	38276.2	1	6.7	13062.5	126	0.153
	lxm.	64472.3	1	6.7	23267.1	218.8	0.131

表 7 計算コスト：100 × 100 nodes（rand-lattice，middle）
Table 7 Computational cost: 100 × 100 nodes (rand-lattice, middle).

cost	alg.	iter.	num. of opn. nodes	exec. time [s]
[1, 2]	sum.	3254	3448	0.008
	awt.	4786	5348	0.029
	lxm.	8955	9404	0.114
[1, 5]	sum.	5788	5982	0.015
	awt.	7344	7790	0.039
	lxm.	9335	9816	0.119
[1, 10]	sum.	8075	8277	0.023
	awt.	8558	9081	0.064
	lxm.	8490	9112	0.220

5.2 インクリメンタルな解法

次に，エージェントの探査と学習をともなうインクリメンタルな解法を評価した．ここでは，20 × 20 の格子グラフと座標を変更したグラフ（rand-lattice）の，左上–右下頂点および，中間–右下頂点の経路を求める問題を用い，vleximax に基づく A*アルゴリズム ‘lxm. A*’ および，探査と学習をともなう解法 ‘lxm. lrn.’ を適用した．エピソードの獲得と収束に十分なパラメータを予備実験により設定した．**表 8** と**表 9** に解品質を示す．これらの問題では，両者は同様の結果を得ており，提案手法の合理性が示されていると考えられる．



(a) $w_{i,j} = [1, 2]$ (b) $w_{i,j} = [1, 5]$ (c) $w_{i,j} = [1, 10]$

(d) $w_{i,j} = [1, 2]$ (middle) (e) $w_{i,j} = [1, 5]$ (middle) (f) $w_{i,j} = [1, 10]$ (middle)

(g) $w_{i,j} = [1, 2]$ (rand-lattice, middle) (h) $w_{i,j} = [1, 5]$ (rand-lattice, middle) (i) $w_{i,j} = [1, 10]$ (rand-lattice, middle)

図 3 Vleximax のインクリメンタルな最適化：20 × 20 nodes
Fig. 3 Incremental optimization with vleximax: 20 × 20 nodes.

表 8 解品質：20 × 20 nodes（lxm，A* and learning）
Table 8 Solution qualities: 20 × 20 nodes (lxm, A* and learning).

cost	alg.	solution quality				
		sum.	min.	max.	len.	theil
[1, 2]	lxm. A*	46.8	1	1.9	44.8	0.016
	lxm. lrn.	46.8	1	1.9	44.8	0.016
[1, 5]	lxm. A*	77.4	1	3.4	45.4	0.102
	lxm. lrn.	77.4	1	3.4	45.4	0.102
[1, 10]	lxm. A*	145.7	1	6.6	50.6	0.145
	lxm. lrn.	145.7	1	6.6	50.6	0.145

表 9 解品質：20×20 nodes（lxm, A* and learning, rand-lattice, middle）
Table 9 Solution qualities: 20 × 20 nodes (lxm, A* and learning, rand-lattice, middle).

cost	alg.	solution quality				
		sum.	min.	max.	len.	theil
[1, 2]	lxm. A*	2988.6	1	1.8	2893.7	0.012
	lxm. lrn.	2988.6	1	1.8	2893.7	0.012
[1, 5]	lxm. A*	5685.8	1	3.5	3310	0.087
	lxm. lrn.	5685.8	1	3.5	3310	0.087
[1, 10]	lxm. A*	10587.9	1	6.4	3570.3	0.131
	lxm. lrn.	10587.9	1	6.4	3570.3	0.131

図 3 にインクリメンタルな解法の収束状況の例を示す．これらのグラフは実際の探査結果についての解品質の anytime 曲線を表す．100 試行ごとに合計とタイル尺度を平均し，その変化を示した．オンライン型探索と同様に，発見的戦略と下界優先探索のために，収束の過程で解品質が大きく変動しうる．これらの結果では，合計およびタイル尺度ともに，最終的には減少し収束した．また，タイル尺度についてはコスト値のばらつきが小さければ，初期の段階から比較的抑制される傾向が見られた．

6. 議論

leximax についての最大化は，最悪の場合のコスト値の改善を優先するため，そのトレードオフによりコスト値の合計やそれに付随する指標の値は増加する．これは，公平性と効率性のトレードオフであることから不可避であり，また，(v)leximax は mini-max の拡張のパラメータなしの指標であることから本質的な性質である．コスト値の範囲が比較的広ければ，トレードオフは拡大し，合計の増加も大きくなる．また，ベクトル長すなわち，経路に含まれる辺の数も増加する．このようなコストの増加が極端な場合

には，コスト値の範囲の調整や，経路長とのトレードオフを制御する手法などの検討が必要と考えられる．解品質を表す数値のバランスについては，提案手法の枠組みに基づきユーザにより問題のコスト値の範囲を設定する対話的解法による調整が考えられる．その一方で，提案手法により得られる経路はある種の公平性に基づく下界となる解を分析するものととらえられる．すなわち，(v)leximax の適用による直接的な有効性と重要性は，最悪ケースと公平性を重視するパレート最適な指標に基づく解により，個々の問題の分析の根拠となりうることといえる．

部分経路のコストを平準化する指標の適用事例として，配送車両の移動経路計画の際に特定の近隣住民の不満や道路設備の劣化の程度が先鋭化するような経路を抑制する例や，アドホック通信網上のルーティングにおいて中継装置の電池の残量を維持する例への検討が考えられる．

(V)leximax の結果が良いか否かは多目的最適化における解の選択で本質的な主観的価値に依存する．たとえば，トラックなどの重量が大きな配送車両の移動経路を決定する際に，老朽化した道路の消耗を抑制することを最優先するときは，比較的消耗が少ないことを表すコストが小さな道路からなる経路を通ることが合理的であると考えられる．

また，今後の検討課題として leximax のように動的計画法による分解と整合し，かつ公平性と効率性のトレードオフを調整するパラメータを持つ指標の模索があげられる．その一方で，パレート最適性などの一般的に要求される性質を持ち，動的計画法とともに適用できる指標の模索には本研究とは独立した検討が必要と考えられる．

本研究では，格子グラフを前提とするマンハッタン距離に基づく発見的距離関数と，一般のグラフにおいて辺の長さに応じた個数のコスト値のベクトルが辺のコストである場合を前提としたユークリッド距離に基づく発見的距離関数を，許容的な下界として示した．このような下界のギャップは大きく，最大コスト最小化における枝刈りの効果は小さいが，これは問題の性質から自然であるとも考えられる．他のトポロジやより良い下界のためには異なる前提が必要であると考えられる一方で，A*アルゴリズムの最悪の場合がダイクストラ法であり，かつコスト値ベクトルのヒストグラム表現のための計算オーバヘッドが線形のオーダであることを考慮すれば，近年の潤沢な計算資源において，許容される規模の問題がありうると期待される．

一般に，厳密な公平性を扱う多くの指標については，動的計画法による部分問題への分解は簡単ではない．本研究では，leximin/leximax に基づく指標を経路探索問題のための動的計画法に適用する方法について検討した．これらの指標に基づく結合や比較の演算は，おおむね直接的に適用できたが，ベクトルの下界に依存する探査においては，閉路における下界の単調性の不整合を回避するための手法が必要である．これらの知見は，類似の指標に基づく強化学習への発展などにつながる可能性があると考えられる．

本研究では，基礎的な問題として単一エージェントの経路最適化に注目した．マルチエージェントシステムの設定として，各部分グラフが異なるエージェントに属する状況における最適化も考えられる．これような設定は，分散制約最適化問題に leximin を導入した従来研究 [8], [10] と類似する面がある一方で，解が経路に対応する可変長のベクトルとなることなどの変更が必要となる．

また，異なる多目的最適化問題の設定として，所要時間と環境への影響のような複数のコストの組が各区間に関連付けられた状況における経路最適化も考えられる．このような場合は，異なる種類のコストを結合するためのスカラ化を階層化する手法の検討が必要と考えられる．

7. おわりに

本研究では，leximax の最大化に基づき，経路における最大コストと公平性を考慮しつつコストを最適化する経路探索手法について検討した．提案手法により，最大コストとなる区間を抑制しつつ公平性を改善する経路が得られた．解法の効率化，合計コストや経路長との実際的なトレードオフなどが今後の課題としてあげられる．また，オンライン型探索や強化学習の可能性についてもさらに検討の余地があると考えられる．

謝辞 本研究の一部は科研費基盤研究（C）16K00301，19K12117 による．

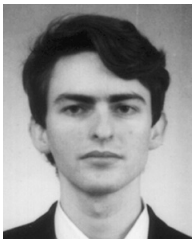
参考文献

[1] Barto, A.G., Bradtke, S.J. and Singh, S.P.: Learning to Act Using Real-time Dynamic Programming, *Artificial Intelligence*, Vol.72, No.1-2, pp.81–138 (1995).
[2] Bouveret, S. and Lemaître, M.: Computing Leximin-optimal Solutions in Constraint Networks, *Artificial Intelligence*, Vol.173, No.2, pp.343–364 (2009).
[3] Conceicao, P. and Ferreira, P.: The young person's guide to the Theil Index: Suggesting intuitive interpretations and exploring analytical applications, *UTIP Working Paper*, No.14, University of Texas (2000).
[4] Greco, G. and Scarcello, F.: Constraint Satisfaction and Fair Multi-objective Optimization Problems: Foundations, Complexity, and Islands of Tractability, *Proc. 23rd International Joint Conference on Artificial Intelligence*, pp.545–551 (2013).
[5] Hart, P.E., Nilsson, N.J. and Raphael, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths, *IEEE Trans. Syst. Science and Cybernetics*, Vol.4, No.2, pp.100–107 (1968).
[6] Hart, P.E., Nilsson, N.J. and Raphael, B.: Correction to 'A Formal Basis for the Heuristic Determination of Minimum-Cost Paths', *SIGART Newsletter*, No.37, pp.28–29 (1972).
[7] Marler, R.T. and Arora, J.S.: Survey of multi-objective optimization methods for engineering, *Structural and Multidisciplinary Optimization*, Vol.26, pp.369–395 (2004).
[8] Matsui, T., Silaghi, M., Hirayama, K., Yokoo, M. and

Matsuo, H.: Leximin Multiple Objective Optimization for Preferences of Agents, *Proc. 17th International Conference on Principles and Practice of Multi-Agent Systems*, pp.423–438 (2014).
[9] Matsui, T., Silaghi, M., Hirayama, K., Yokoo, M. and Matsuo, H.: Study of Route Optimization Considering Bottlenecks and Fairness Among Partial Paths, *Proc. 10th International Conference on Agents and Artificial Intelligence*, pp.37–47 (2018).
[10] Matsui, T., Silaghi, M., Okimoto, T., Hirayama, K., Yokoo, M. and Matsuo, H.: Leximin Asymmetric Multiple Objective DCOP on Factor Graph, *Proc. 18th International Conference on Principles and Practice of Multi-Agent Systems*, pp.134–151 (2015).
[11] Russell, S. and Norvig, P.: *Artificial Intelligence: A Modern Approach* (*2nd Edition*), Prentice Hall (2003).
[12] Sen, A.: *On Economic Inequality*, Clarendon Press (1973).
[13] Sen, A.K.: *Choice, Welfare and Measurement*, Harvard University Press (1997).
[14] Sutton, R.S. and Barto, A.G.: *Reinforcement learning: An introduction*, MIT Press (1998).

松井 俊浩 （正会員）

1995年名古屋工業大学電気情報工学科卒業．1999年同大学大学院博士前期課程修了．2006年同博士後期課程修了．2006年名古屋工業大学情報基盤センター助手．2007年同助教．2011年同准教授．現在に至る．分散協調処理，マルチエージェントシステム，分散制約最適化問題に関する研究に従事．博士（工学）．電子情報通信学会，人工知能学会各会員．



マリウス シラギ

1997年 Cluj-Napoca 工科大学卒業．1997年ブラウンシュヴァイク工科大学，Scientific Computing Laboratory，DAAD Researcher．1998–2002年スイス連邦工科大学 Artificial Intelligence Laboratory，Research Assistant．2002年スイス連邦工科大学，Ph.D．2002–2014年フロリダ工科大学，Department of Computer Sciences，Assistant Professor．2014年同大学，Department of Computer Engineering and Sciences，Associate Professor．現在に至る．Ph.D.（Computer Science）．AAAI（2000–2007），IEEE（2001–2008）各会員．

平山 勝敏 （正会員）

1990年大阪大学基礎工学部制御工学科卒業．1992年同大学大学院基礎工学研究科博士前期課程修了．1995年同大大学院基礎工学研究科博士後期課程修了．博士（工学）．1995年神戸商船大学助手．1997年同講師．2001年同助教授．2003年神戸大学海事科学部助教授（神戸大学と神戸商船大学の統合による）．2007年神戸大学大学院海事科学研究科准教授．2013年神戸大学大学院海事科学研究科教授．1999～2000年カーネギーメロン大学ロボティクス研究所客員研究員（文部省在外研究員）．マルチエージェントシステム，制約充足/SAT，組合せ最適化に関する研究に従事．2010年 IFAAMAS Influential Paper Award 受賞．2015年日本ソフトウエア科学会 2014年度基礎研究賞受賞．電子情報通信学会，人工知能学会，日本オペレーションズリサーチ学会，日本ソフトウェア科学会各会員．

横尾 真 （正会員）

1984年東京大学工学部電子工学科卒業．1986年同大学 大学院修士課程修了．同年NTTに入社．1990～1991年ミシガン大学客員研究員．2004年九州大学大学院システム情報科学研究院教授．2012年より同大学主幹教授 マルチエージェントシステム，制約充足問題に関する研究に従事．エージェントの合意形成メカニズム，制約充足/分散制約充足等に興味を持つ．博士（工学）．1992年，2002年人工知能学会論文賞，1995年情報処理学会坂井記念特別賞，2004年 Association for Computing Machinery（ACM）Special Interest Group on Artificial Intelligence（SIGART）Autonomous Agent Research Award，2005年ソフトウェア科学会論文賞，2006年学士院学術奨励賞，2010年人工知能学会業績賞，International Foundation for Autonomous Agents and Multiagent Systems influential paper award，2011年ソフトウェア科学会基礎研究賞，情報処理学会論文賞，2018年文部科学大臣表彰科学技術賞受賞．日本ソフトウェア科学会，AAAI フェロー，電子情報通信学会，人工知能学会各会員．本会フェロー．

松尾 啓志 （正会員）

1983 年名古屋工業大学情報工学科卒業．1985 年同大学大学院修士課程修了．同年松下電器産業株式会社入社．1989 年名古屋工業大学大学院博士課程修了．同年名古屋工業大学電気情報工学科助手．講師，助教授を経て，2003 年同大学大学院教授．2006 年同大学情報基盤センターセンター長（併任）．現在に至る．分散システム，分散協調処理に関する研究に従事．工学博士．電子情報処理学会，人工知能学会，IEEE 各会員．