CD-002

# 多様性を考慮した最大内積探索

平田 皓平, 天方 大地, 原 隆浩
大阪大学
{hirata.kohei,amagata.daichi,hara}@ist.osaka-u.ac.jp

藤田 澄男
ヤフー株式会社
sufujita@yahoo-corp.jp

## アブストラクト

最大内積探索（または $k$-MIPS）は，ユーザに対して好ましいアイテムを推測し，推薦システムにおける基本的な操作である．大規模な推薦システムをサポートするために，既存研究ではスケーラブルな $k$-MIPS アルゴリズムが設計されている．しかし，多様なアイテムを推薦することはユーザ満足度を向上させるにも関わらず，これらの研究では多様性について考慮されていない．そこで，我々は新しい問題である，多様性を考慮した $k$-MIPS を定式化する．この問題では，ユーザは推薦リストの多様性をパラメータで制御することが可能である．しかし，この問題を正確に解くことは NP 困難であり，多様性を考慮した $k$-MIPS 問題に対する効率的，効果的，および実用的なアルゴリズムを考案することは挑戦的である．本論文では，この課題を克服し，新しい早期終了およびスキップ技術を貪欲アルゴリズムに組み込んだ IP-Greedy を提案する．実データを用いた大規模な実験を行い，本アルゴリズムの効率性および有効性を実証する．また，実世界のデータセットにおいて多様性を考慮した $k$-MIPS 問題のケーススタディを行い，推薦リスト中のアイテムベクトルとユーザベクトルとの高い内積を維持し，推薦リストの多様性を確保できることを確認する．

## 1 はじめに

近年，電子商取引 [37]，ニュースサイト [25]，ビデオオンデマンドサービス [39] などにおいて，推薦システムはユーザ体験を向上させ，大きな利益をもたらすため必要不可欠である．そのため，学術界 [49, 51] および産業界 [16, 26, 29] では推薦システムの設計が行われている．通常，推薦システムはユーザとアイテムの関係を利用し，ユーザおよびアイテムの埋め込みを学習する．大まかに，ユーザとアイテムの関係を学習する手法は，行列分解（MF）と（深層）ニューラルネットワーク [4, 36] の 2 つが存在する．近年の研究 [4, 36] では，ニューラルネットワークよりも MF を用いた推薦システムの方が高性能であることが確認されているため，本論文では MF を用いた推薦システムを想定する．

MF に基づく推薦システムでは，与えられたユーザベクトルとの内積が上位 $k$ 個となるアイテムベクトルを求める $k$ 最大内積探索（$k$-MIPS）が基本操作である．ユーザはインタラクティブに推薦リストを取得する必要があるため，$k$-MIPS アルゴリズムは大量のアイテムにスケールする必要がある．これを満たすために，既存研究は（近似）$k$-MIPS アルゴリズムを開発した [2, 5, 21, 31, 35, 41–43]．ここで，ユーザの嗜好は推薦システムの重要な要素であり，これらの $k$-MIPS の研究はこの要素を満たしている．また，推薦システムにおいて多様性も重要な要素であり，多様性を推薦システムに取り入れる企業も存在する [1, 3]．しかし，$k$-MIPS の研究ではこの要素は無視されている．

**例 1.** 表1は MovieLens[1]のデータセットにおける，あるユーザ（ユーザ A）の推薦リストを示す．行列分解によりユーザベクトルおよびアイテムベクトルを取得し，ユーザ A に対する 5-MIPS 問題の解を推薦リストとした．ユーザ A の推薦リストは大きな偏りがあり，ハリーポッターの映画しか含まれていない．

**表 1: 5-MIPS による推薦リストの例**

| 順序 | ユーザ A |
|---|---|
| 1 | Harry Potter and the Deathly Hallows Part 2 |
| 2 | Harry Potter and the Deathly Hallows Part 1 |
| 3 | Harry Potter and the Half-Blood Prince |
| 4 | Harry Potter and the Order of the Phoenix |
| 5 | Harry Potter and the Goblet of Fire |

この例から，$k$-MIPS によって得られる推薦リストの多様性は極端に悪くなりうることが明らかである．[4] もこれを観察している．ここで，[9, 30, 32, 40, 48] は多様性を考慮している．しかし，これらは内積を考慮せず，また，ユーザはそれらのモデルにおいてオンデマンドで多様性の度合いを調節することはできない．これらの問題を解決するために，制御可能な多様性をどのように $k$-MIPS に組み込むかという問題が生じる．

**貢献.** 本論文では上記の問題に対して，多様性を考慮した最大内積探索問題（または，多様性を考慮した $k$-MIPS）である新たな問題を定式化する．詳細は 2.1 節に示す．MF モデルのもとでは，クエリベクトルが与えられた時，望ましい推薦リストは以下を満たす必要がある．(i) クエリと高い内積を持つアイテムを含み，(ii) リスト中のアイテムは互いに非類似であり，(iii) ユーザが多様性の度合いを調節可能である．多様性を考慮した $k$-MIPS 問題はこれらを満たすが，この問題を正確に解くことは NP 困難である．つまり，最適解は多項式時間では得られず，発見的手法のみが許される．そのため，克服しなければならない課題は，この問題をどのようにして効率的かつ効果的に解くかである．

本論文はこの課題に取り組む．$\mathbf{P}$ を $n$ 個のアイテムベクトルからなる集合とする．通常，NP 困難の問題で採用される素朴な貪欲アルゴリズムは，解を $\mathbf{S} \subset \mathbf{P}$（つまり，$k$ 個のアイテムの集合）とし，各反復で $\mathbf{S}$ にアイテムベクトル $\mathbf{p}$ を追加することで，$\mathbf{S}$ をインクリメンタルに更新する．ここで，$\mathbf{p}$ は $\mathbf{P}$ の線形スキャンによって得られるため，$i$ 回目の反復は固定された次元数において $\Theta(n-i)$ 時間必要である．（詳細は 2.2 節に示す）．これがこの貪欲アルゴリズムの主なボトルネックである．提案する貪欲アルゴリズムである IP-Greedy は，素朴な貪欲アルゴリズムと同じ解を保ち，新しい早期終了およびアイテムレベルのスキップ技術を用いることで，このボトルネックを取り除くことができる．IP-Greedy の時間計算量は $\tilde{O}((2-\alpha_1)n + k^2(1-\alpha_2)n)$ である．ここで，$\alpha_1$ および $\alpha_2$ は枝刈り率である[2]．実践的には，$\alpha_1$ および $\alpha_2$ は大きいため，時間計算量はおおよそ $\Theta(n)$ になる．

我々の主な貢献は以下の通りである．

- 多様性を考慮した $k$-MIPS 問題を定式化する（2 章）．また，本研究はこの問題に取り組んだ初めての研究である．
- この問題では，既存手法を単に適用する場合，全てのアイテムベクトルに $k$ 回アクセスする必要があるため，効率性が問題となる．我々はこの弱点を取り除き，貪欲アルゴリズムと同じ解を維持しつつ，無駄な計算を回避することができる IP-Greedy を提案する（3 章）．

[1] https://grouplens.org/datasets/movielens/

[2] $\tilde{O}(\cdot)$ はポリ対数の要素を省略する．

- 実世界のデータセットを用いた大規模な実験を行い，我々のアルゴリズムの効率性および有効性を実証する．素朴な貪欲法と比較して，IP-Greedy は通常少なくとも 10 倍高速であり，$k$ に対して頑健である．また，実世界のデータセットを用いたケーススタディを行い，多様性を考慮した $k$-MIPS の有効性を実証する．その結果，$k$-MIPS と比較して，ユーザベクトルと高い内積を持ちつつ，推薦リストが多様化されることを確認する（4 章）．

以上の内容に加え，5 章で関連研究を紹介し，最後に 6 章で本論文の結論を述べる．

## 2 予備知識

まず，本論文で取り組む多様性を考慮した $k$-MIPS 問題を定義し，次に，この問題を近似的に解くベースラインアルゴリズムを紹介する．本論文の目的は 2 つである．(i) $k$-MIPS の結果を（教師無しで）多様化すること，(ii) 多様性を考慮した $k$-MIPS 問題に対する効率的かつ効果的なアルゴリズムを考案することである．

### 2.1 問題定義

$\mathbf{p}$ を $d$ 次元のベクトルとし，$d$ は大きいものとする [2, 5, 21, 31, 41–43]．さらに，$\mathbf{P}$ を $n$ 個のアイテムベクトルの集合とし，$\mathbf{p}$ と $\mathbf{p}'$ の内積を $\mathbf{p}\cdot\mathbf{p}'$ で表す．次に，$k$-MIPS を定義する．

**定義 1（$k$-MIPS 問題）．** クエリベクトル $\mathbf{q}$，アイテムベクトルの集合 $\mathbf{P}$，および結果のサイズ $k$ が与えられた時，この問題は，$\mathbf{P}$ に含まれる $k$ 個のアイテムベクトルからなる集合 $\mathbf{S}$ を探索する．ただし，$\mathbf{p}\in\mathbf{S}$，$\mathbf{p}'\in\mathbf{P}\backslash\mathbf{S}$，$\mathbf{p}\cdot\mathbf{q}\geq\mathbf{p}'\cdot\mathbf{q}$ を満たす．

ここで，$k$-MIPS に多様性を組み込む．多様性に関して様々な要素があるが [19, 24, 28]，本論文では，[1, 23, 50] と同様にアイテム間の非類似度である距離を多様化のために採用する．$dist(\mathbf{p},\mathbf{p}')$ は $\mathbf{p}$ と $\mathbf{p}'$ 間の距離を表し，$dist(\cdot,\cdot)$ は三角不等式を満たすとする．行列分解によって作成されたベクトルは内積空間に存在し，これらのベクトルはこの内積空間に特有であるため，距離は有効ではない．そこで，各アイテムは内積空間とメトリック空間の 2 つのベクトルを持つとする．例えば，Item2Vec[6] を用いて，ユークリッド空間における各アイテムベクトルを生成し，任意の 2 つのアイテム間の距離を求めることができる．本論文では表現を簡単にするために，$\mathbf{p}\cdot\mathbf{q}$ の文脈において $\mathbf{p}$ は内積空間のアイテムベクトルを表し，$dist(\mathbf{p},\mathbf{p}')$ の文脈において $\mathbf{p}$ はメトリック空間のアイテムベクトルとする．

クエリベクトル $\mathbf{q}$ に対して，望ましい結果集合 $\mathbf{S}$ は，(i) $\mathbf{p}\cdot\mathbf{q}$ が高く，(ii) 各 $\mathbf{p},\mathbf{p}'\in\mathbf{S}$ において $dist(\mathbf{p},\mathbf{p}')$ が大きくなる $\mathbf{p}$ から構成される．すなわち，$\mathbf{S}$ の中のアイテムは $\mathbf{q}$ に関連し，かつ互いに非類似である，つまり多様である．このような結果集合を得るために，定義 1 を拡張する．

**定義 2（多様性を考慮した $k$-MIPS 問題）．** クエリベクトル $\mathbf{q}$，アイテムベクトルの集合 $\mathbf{P}$，結果のサイズ $k$，およびバランスパラメータ $\lambda\in[0,1]$ が与えられた時，この問題は以下によって定義される $\mathbf{S}^*$ を求めるものである．

$$\mathbf{S}^* = \underset{\mathbf{S}\subseteq\mathbf{P},|\mathbf{S}|=k}{\arg\max}\, f(\mathbf{S}) \tag{1}$$

$$f(\mathbf{S}) = \frac{\lambda}{k}\sum_{\mathbf{p}\in\mathbf{S}}\mathbf{p}\cdot\mathbf{q} + c(1-\lambda)\min_{\mathbf{p},\mathbf{p}'\in\mathbf{S}} dist(\mathbf{p},\mathbf{p}'), \tag{2}$$

ここで，$c$ は $dist(\cdot,\cdot)$ のスケールを内積空間のスケールに調節するためのパラメータである．

この定式化は MMR（maximal marginal relevance）[7] を採用している．[1, 32] などの近年の研究においても，MMR に似た定式化が採用されているが，細部が異なる．$\lambda=1$ の場合，本問題は $k$-MIPS になる．一方，$\lambda=0$ の場合，本問題は $k$-center 問題になる．$\lambda$ を用いて，ユーザは多様性の度合いを自由に調節することができ，$\lambda$ が小さいほど多様性に重きを置いていることになる．

MMR は NP 困難であることが知られており [14]，$\mathbf{S}^*$ を求めることも NP 困難であることが $k$-center 問題で示されている [20]．このことは，高い $f(\mathbf{S})$ を持つ $\mathbf{S}$ を得る発見的手法が必要であることを示唆している．本論文ではそのようなアルゴリズムを考案する．

### 2.2 貪欲アルゴリズム

多様性を考慮した $k$-MIPS 問題は MMR の変形である．そのため，経験的に有効な解を多項式時間で得るために通常採用される MMR の貪欲アルゴリズムをベースラインとして使用する [14, 44]．このアルゴリズムは以下のように解 $\mathbf{S}$ を得る．

(1) 与えられたクエリベクトル $\mathbf{q}$ と最大の内積となるアイテムベクトルで $\mathbf{S}$ を初期化する．（$S$ は任意のアイテムベクトルで初期化できるが，$k$-MIPS を考慮するためにこのようにする．）
(2) 各アイテムベクトル $\mathbf{p}\in\mathbf{P}\backslash\mathbf{S}$ に対して，このアルゴリズムは次の関数を評価する．

$$f(\mathbf{p},\mathbf{S}) = \lambda\mathbf{p}\cdot\mathbf{q} + c(1-\lambda)\min_{\mathbf{p}_a,\mathbf{p}_b\in\mathbf{S}\cup\{\mathbf{p}\}} dist(\mathbf{p}_a,\mathbf{p}_b). \tag{3}$$

そして，

$$\mathbf{p}^* = \underset{\mathbf{p}\in\mathbf{P}\backslash\mathbf{S}}{\arg\max}\, f(\mathbf{p},\mathbf{S}) \tag{4}$$

を $\mathbf{S}$ に追加する．(与えられた途中解 $\mathbf{S}$ に対して，$\mathbf{p}^*$ は式 (2) を最大化する.)
(3) $|\mathbf{S}|=k$ となるまでステップ 2 を繰り返す．

**分析．** ここで，貪欲アルゴリズムの時間計算量を分析する．以下，$d=O(1)$ とし，本論文で紹介する全てのアルゴリズムの時間計算量は $d$ に線形である．

最初の反復である MIPS は $\mathbf{P}$ の線形スキャンにより行われるため，$\Theta(n)$ の時間が必要である．$i$ 回目の反復 ($2\leq i\leq k$) では，式 (3) を単純に計算するのに $O(k)$ 時間を必要とするため，式 (4) を単純に計算するのに $O(kn)$ 時間を必要とする．ゆえに，貪欲アルゴリズムは $\Theta(n)+O((k-1)kn)=O(k^2n)$ 時間を必要とする．

**貪欲アルゴリズムの最適化．** MMR の貪欲アルゴリズムは $O(k^2n)$ 時間で実行されると言われているが [8, 17]，実際これはタイトではない．ここでは，貪欲アルゴリズムの時間計算量を $O(kn)$ および空間計算量を $O(n)$ で実装する方法を述べる．$2\leq i\leq k$ である $i$ 回目の反復について考える．アイテムベクトル $\mathbf{p}\in\mathbf{P}\backslash\mathbf{S}$ が与えられた時，単純に式 (3) を評価すると，$\min_{\mathbf{p}_a,\mathbf{p}_b\in\mathbf{S}\cup\{\mathbf{p}\}} dist(\mathbf{p}_a,\mathbf{p}_b)$ の計算により $O(k)$ 時間が発生する．ここで，以下に注目する．

$$\min_{\mathbf{p}_a,\mathbf{p}_b\in\mathbf{S}\cup\{\mathbf{p}\}} dist(\mathbf{p}_a,\mathbf{p}_b) = \min\{\min_{\mathbf{p}_a,\mathbf{p}_b\in\mathbf{S}} dist(\mathbf{p}_a,\mathbf{p}_b), \min_{\mathbf{p}_a\in\mathbf{S}} dist(\mathbf{p},\mathbf{p}_a)\}. \tag{5}$$

また，$\mathbf{p}^j$ を $j$ 回目の反復における $\mathbf{p}^*$ とする．このとき，$\mathbf{p}\in\mathbf{P}\backslash\mathbf{S}$ に対し，

$$\min_{\mathbf{p}_a\in\mathbf{S}} dist(\mathbf{p},\mathbf{p}_a) = \min\{dist(\mathbf{p},\mathbf{p}^1),...,dist(\mathbf{p},\mathbf{p}^{i-1})\} \tag{6}$$

この式から，各 $\mathbf{p}\in\mathbf{P}\backslash\mathbf{S}$ の $\min\{dist(\mathbf{p},\mathbf{p}^1),...,dist(\mathbf{p},\mathbf{p}^{i-2})\}$ を保持することにより，$\min_{\mathbf{p}_a\in\mathbf{S}} dist(\mathbf{p},\mathbf{p}_a)$ を $O(1)$ で得ることができる．これは，$\min_{\mathbf{p}_a\in\mathbf{S}} dist(\mathbf{p},\mathbf{p}_a)$ を $dist(\mathbf{p},\mathbf{p}^{i-1})$ の計算によって得ることができるためである．また，各

$\mathbf{p} \in \mathbf{P}\backslash\mathbf{S}$ における $\min\{dist(\mathbf{p}, \mathbf{p}^1), ..., dist(\mathbf{p}, \mathbf{p}^{i-2})\}$ は $O(1)$ の空間計算量のみを必要とする．

この方法と同様に，$\min_{\mathbf{p}_a,\mathbf{p}_b \in \mathbf{S}} dist(\mathbf{p}_a, \mathbf{p}_b)$ を常に保持する．式（6）における $O(1)$ 時間の評価と同じ要領により，$\mathbf{S}$ を更新する時に $\min_{\mathbf{p}_a,\mathbf{p}_b \in \mathbf{S}} dist(\mathbf{p}_a, \mathbf{p}_b)$ もインクリメンタルに更新可能である．

**例 2.** 2 回目の反復において，$\min_{\mathbf{p}_a,\mathbf{p}_b \in \mathbf{S}\cup\{\mathbf{p}^2\}} dist(\mathbf{p}_a, \mathbf{p}_b) = dist(\mathbf{p}^1, \mathbf{p}^2)$ である．この値を $\theta$ とし，$\theta$ は $O(1)$ 時間で得ることが可能である．3 回目の反復において，$\min_{\mathbf{p}_a,\mathbf{p}_b \in \mathbf{S}\cup\{\mathbf{p}^3\}} dist(\mathbf{p}_a, \mathbf{p}_b) = \min\{\theta, \min_{\mathbf{p}\in\mathbf{S}} dist(\mathbf{p}^3, \mathbf{p})\}$ である．$S$ の更新の時，既に $\min_{\mathbf{p}\in\mathbf{S}} dist(\mathbf{p}^3, \mathbf{p})$ を得ている（また，これは $O(1)$ 時間で得る）ため，$\min\{\theta, \min_{\mathbf{p}\in\mathbf{S}} dist(\mathbf{p}^3, \mathbf{p})\}$ の評価は $O(1)$ 時間である．残りの反復も同様の操作を行う．

各 $\mathbf{p} \in \mathbf{P}\backslash\mathbf{S}$ に対し，$O(1)$ の時間/空間で式 (3) を評価可能であることは明らかである．したがって，各反復は $O(n)$ 時間を必要とするため，時間計算量は $O(nk)$ となる．

## 3 IP-GREEDY

上記の最適化により，貪欲アルゴリズムは $n$ および $k$ に対して線形時間で実行できる．しかし，各反復で式 (4) の $\mathbf{p}^*$ を求めるために，$|\mathbf{P}\backslash\mathbf{S}|$ のアイテムベクトルにアクセスすることによるボトルネックが発生する．そのため，依然として貪欲アルゴリズムは非効率的である．例えば，$|\mathbf{P}| \approx 400,000$ および $k = 10$ の時，貪欲アルゴリズムは 1 つのクエリの $\mathbf{S}$ を得るのに，数秒を必要とする（表3を参照）．この処理時間は十分ではない．なぜなら，推薦システムは多くのユーザ（クエリ）をサポートする必要があり，ユーザはインタラクティブ性（例えば，様々な $k$ や $\lambda$ のクエリ）を要求するからである．ここで，各 $\mathbf{p} \in \mathbf{P}\backslash\mathbf{S}$ の式 (3) を計算せずに，$\mathbf{p}^*$ を特定できるのかという疑問が生じるが，これが可能であることを本章で証明する．なお，問題の NP 困難性のため，貪欲アルゴリズムと同じ解を得ることを考える．

### 3.1 早期終了技術

貪欲アルゴリズムは $\mathbf{P}\backslash\mathbf{S}$ の線形スキャンにより，式 (4) で示される $\mathbf{p}^*$ を求める．この効率を向上させる有望な方法は，残りの（つまり，未評価の）アイテムベクトルの中に $\mathbf{p}^*$ が存在しないことが保証される場合，線形スキャンを早期に終了させることである．この方法は，$\mathbf{P}\backslash\mathbf{S}$ の全てのアイテムベクトルにアクセスするのではなく，一部のみにアクセスすることで $\mathbf{p}^*$ を特定できる（すなわち，貪欲アルゴリズムと同じ解を得る）．我々はこの手法を実現する．説明を容易にするために，以下のように仮定する．

- $\mathbf{P}$ の全てのアイテムベクトル $\mathbf{p}_i$ は $\gamma_i = \min\{\|\mathbf{p}_i\|^{\mathcal{I}}, \frac{\mathbf{p}_i \cdot \mathbf{q}}{\|\mathbf{q}\|}\}$ の降順でソートされている．なお，$\|\mathbf{p}_i\|^{\mathcal{I}}$ と $\|\mathbf{q}\|$ はそれぞれ内積空間におけるアイテムベクトルとクエリベクトルのユークリッドノルムである．
- $\mathbf{P}$ のアイテムベクトルは $\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n$ の順に並んでいるとする．つまり，これは各 $i \in [1, n-1]$ で $\gamma_i\|\mathbf{q}\| \geq \gamma_{i+1}\|\mathbf{q}\|$ を意味する．
- $2 \leq i \leq k$ である $i$ 回目の反復において，上の順で $\mathbf{P}\backslash\mathbf{S}$ にアクセスする．また，$\mathbf{P}$ の $j$ 回目のアイテムベクトルにアクセスする時，暫定の $\mathbf{p}^*$ を $\mathbf{p}^*_{int}$ とする．

このとき，次の定理が成り立つ．

**定理 1.** $\mathbf{p}_j \in \mathbf{P}\backslash\mathbf{S}$ を評価する時，$\min_{\mathbf{p}_a,\mathbf{p}_b \in \mathbf{S}} dist(\mathbf{p}_a, \mathbf{p}_b) < \frac{f(\mathbf{p}^*_{int}, \mathbf{S}) - \lambda\gamma_j\|\mathbf{q}\|}{c(1-\lambda)}$ を満たす場合，$\mathbf{p}_j, \mathbf{p}_{j+1}, ...\mathbf{p}_n$ が $\mathbf{p}^*$ になることは不可能であり，$\mathbf{p}^*_{int} = \mathbf{p}^*$ である．

**証明.** 式 (5) から，$\min_{\mathbf{p}_a,\mathbf{p}_b \in \mathbf{S}\cup\{\mathbf{p}\}} dist(\mathbf{p}_a, \mathbf{p}_b) \leq \min_{\mathbf{p}_a,\mathbf{p}_b \in \mathbf{S}} dist(\mathbf{p}_a, \mathbf{p}_b)$ となる．また，コーシー・シュワルツの不等式から，$\mathbf{p} \cdot \mathbf{q} \leq \|\mathbf{p}\|^{\mathcal{I}}\|\mathbf{q}\|$ である．このとき，

$$f(\mathbf{p}_j, \mathbf{S}) \leq \lambda\mathbf{p}_j \cdot \mathbf{q} + c(1-\lambda) \min_{\mathbf{p}_a,\mathbf{p}_b \in \mathbf{S}} dist(\mathbf{p}_a, \mathbf{p}_b)$$
$$\leq \lambda\|\mathbf{p}_j\|^{\mathcal{I}}\|\mathbf{q}\| + c(1-\lambda) \min_{\mathbf{p}_a,\mathbf{p}_b \in \mathbf{S}} dist(\mathbf{p}_a, \mathbf{p}_b).$$

したがって，次を満たす時，$\mathbf{p}_j$ が $\mathbf{p}^*$ になることは不可能である．

$$\lambda\gamma_j\|\mathbf{q}\| + c(1-\lambda) \min_{\mathbf{p}_a,\mathbf{p}_b \in \mathbf{S}} dist(\mathbf{p}_a, \mathbf{p}_b) < f(\mathbf{p}^*_{int}, \mathbf{S}). \quad (7)$$

各 $j \in [1, n-1]$ において $\gamma_j\|\mathbf{q}\| \geq \gamma_{j+1}\|\mathbf{q}\|$ より，$\lambda\gamma_j\|\mathbf{q}\| + c(1-\lambda)\min_{\mathbf{p}_a,\mathbf{p}_b \in \mathbf{S}} dist(\mathbf{p}_a, \mathbf{p}_b) \geq \lambda\gamma_{j+1}\|\mathbf{q}\| + c(1-\lambda)\min_{\mathbf{p}_a,\mathbf{p}_b \in \mathbf{S}} dist(\mathbf{p}_a, \mathbf{p}_b)$ となる．つまり，不等式 (7) が成り立つとき，$\mathbf{p}_{j+1}, ...\mathbf{p}_n$ も $\mathbf{p}^*$ になることは不可能である． □

**備考 1.** この定理は $\mathbf{p}^*$ を得るために線形スキャンを正確性を失うことなく早期に終了できることを示唆している．また，$\min_{\mathbf{p}_a,\mathbf{p}_b \in \mathbf{S}} dist(\mathbf{p}_a, \mathbf{p}_b)$，$f(\mathbf{p}^*_{int}, \mathbf{S})$，および $\gamma_j$ は $\mathbf{p}_j$ へのアクセス時に既に計算されているため，$O(1)$ 時間で評価可能である．（$\gamma_j$ を実現する方法は後述する．）特に，$\gamma_j = \|\mathbf{p}_j\|^{\mathcal{I}}$ の時，クエリベクトルと $\mathbf{p}_{j'}$（ただし，$j' \geq j$）の内積を計算する必要はなく，$O(d)$ の計算を除くことができる．以上から，不等式 (7) を評価することも効率的であることが分かる．

**備考 2.** 定理 1 は 2 回目の反復後に機能する．1 回目の反復後は $|\mathbf{S}| = 1$ のため，$\min_{\mathbf{p},\mathbf{p}'\in\mathbf{S}} dist(\mathbf{p}, \mathbf{p}') = \infty$．それゆえ，2 回目の反復は全ての $\mathbf{P}\backslash\mathbf{S}$ に対し，定理 1 を使用することはできない．この事実は，少なくとも $\Omega(n)$ 個のアイテムベクトルへのアクセスが回避できないことを意味する．

### 3.2 アイテムレベルのスキップ技術

$\mathbf{p}_j$ が不等式 (7) を満たさない場合を考える．この場合，単純に式 (3) を計算することが素朴な方法である．貪欲アルゴリズムの最適化から，式 (3) の計算には $\Omega(d)$ 時間が必要であることは明らかである．2.1 節で述べたように $d$ は大きいため，この計算を省くことが望ましい．以下では，これが可能であること，すなわち，2 回目の反復も効率化できることを証明する．($3 \leq i \leq k$ である $j$ 回目の反復もこの恩恵を受ける．)

メトリック空間におけるアイテムベクトルのユークリッドノルムを $\|\mathbf{p}\|^{\mathcal{M}}$ とする．なお，このノルムはクエリに依存しないためオフラインで計算される．式 (3) の計算を省くことができる定理について説明する．

**定理 2.** $\mathbf{p}_j \in \mathbf{P}\backslash\mathbf{S}$ が $\|\mathbf{p}_j\|^{\mathcal{M}} + \min_{\mathbf{p}^{\mathcal{M}}\in\mathbf{S}} \|\mathbf{p}\| \leq \frac{f(\mathbf{p}^*_{int}, \mathbf{S}) - \lambda\gamma_j\|\mathbf{q}\|}{c(1-\lambda)}$ を満たす時，$\mathbf{p}_j \neq \mathbf{p}^*$ である．

**証明.** 定理 1 の証明と同様にして，式 (5)，コーシーシュワルツの不等式，および三角不等式から次のようになる．

$$f(\mathbf{p}_j, \mathbf{S}) \leq \lambda\gamma_j\|\mathbf{q}\| + c(1-\lambda) \min_{\mathbf{p}\in\mathbf{S}} dist(\mathbf{p}_j, \mathbf{p})$$
$$\leq \lambda\gamma_j\|\mathbf{q}\| + c(1-\lambda)(\|\mathbf{p}_j\|^{\mathcal{M}} + \min_{\mathbf{p}\in\mathbf{S}} \|\mathbf{p}\|^{\mathcal{M}})$$

このことから，$\lambda\gamma_j\|\mathbf{q}\| + c(1-\lambda)(\|\mathbf{p}_j\|^{\mathcal{M}} + \min_{\mathbf{p}\in\mathbf{S}} \|\mathbf{p}\|^{\mathcal{M}}) \leq f(\mathbf{p}^*_{int}, \mathbf{S})$ を満たす時，$\mathbf{p}_j \neq \mathbf{p}^*$ は明らかである． □

**備考 3.** $\|\mathbf{p}_j\|^{\mathcal{M}}$ は事前に計算でき，$\mathbf{S}$ の更新時に $\min_{\mathbf{S}} \|\mathbf{p}\|^{\mathcal{M}}$ はインクリメンタルに更新できるため，$\|\mathbf{p}_j\|^{\mathcal{M}} + \min_{\mathbf{p}\in\mathbf{S}} \|\mathbf{p}\|^{\mathcal{M}}$ を $O(1)$ 時間で計算することができる．したがって，定理 2 から正確性を保ちつつ，式 (3) で発生する $\Omega(d)$ 時間を省くことができる．

**Algorithm 1:** IP-Greedy

**Input:** $\mathbf{P}$, $\mathbf{q}$, $k$, and $\lambda$
**Output:** $\mathbf{S}$

1 Compute $\|\mathbf{q}\|$, $\mathbf{p}^* \leftarrow \emptyset$, $\mathbf{T} \leftarrow \emptyset$, $\tau \leftarrow 0$, $\mathbf{S} \leftarrow \emptyset$
2 **for** *each* $i \in [1, n]$ **do**
3  **if** $\|\mathbf{p}_i\|^{\mathcal{I}} \|\mathbf{q}\| \leq \tau$ **then**
4  **break**
5  Compute $\mathbf{p}_i \cdot \mathbf{q}$ and maintain the value
6  $\mathbf{T} \leftarrow \mathbf{T} \cup \{\mathbf{p}_i\}$
7  **if** $\mathbf{p}_i \cdot \mathbf{q} > \tau$ **then**
8  $\tau \leftarrow \mathbf{p}_i \cdot \mathbf{q}$, $\mathbf{p}^* \leftarrow \mathbf{p}_i$
9 $\mathbf{S} \leftarrow \mathbf{p}^*$
10 $\min_{\mathbf{p},\mathbf{p}' \in \mathbf{S}} dist(\mathbf{p}, \mathbf{p}') \leftarrow \infty$
11 $\min_{\mathbf{p} \in \mathbf{S}} \|\mathbf{p}\|^{\mathcal{M}} \leftarrow \|\mathbf{p}^*\|^{\mathcal{M}}$
12 **for** *each* $i \in [2, k]$ **do**
13  **for** *each* $\mathbf{p}_j \in \mathbf{T}$ **do**
14  Re-order $\mathbf{p}_j$, $\mathbf{T} \leftarrow \mathbf{T} \backslash \{\mathbf{p}_j\}$
15  $\mathbf{p}^*_{int} \leftarrow \emptyset$
16  **for** *each* $j \in [1, n]$ *such that* $\mathbf{p}_j \notin \mathbf{S}$ **do**
17  **if** $\min_{\mathbf{p}_a, \mathbf{p}_b \in \mathbf{S}} dist(\mathbf{p}_a, \mathbf{p}_b) < \frac{f(\mathbf{p}^*_{int}, \mathbf{S}) - \lambda \gamma_j \|\mathbf{q}\|}{c(1-\lambda)}$ **then**
18  **break**
19  **if** $\|\mathbf{p}_j\|^{\mathcal{M}} + \min_{\mathbf{S}} \|\mathbf{p}\|^{\mathcal{M}} > \frac{f(\mathbf{p}^*_{int}, \mathbf{S}) - \lambda \gamma_j \|\mathbf{q}\|}{c(1-\lambda)}$ **then**
20  $flag \leftarrow 1$
21  **if** $\mathbf{p}_j \cdot \mathbf{q}$ *has not been computed* **then**
22  Compute $\mathbf{p}_j \cdot \mathbf{q}$ and maintain the value
23  $\gamma_j \leftarrow \mathbf{p}_j \cdot \mathbf{q} / \|\mathbf{q}\|$, $\mathbf{T} \leftarrow \mathbf{T} \cup \{\mathbf{p}_j\}$
24  **if** *Now (line 17 is true)* $\vee$ *(line 19 is false)* **then**
25  $flag \leftarrow 0$
26  **if** $flag = 1$ **then**
27  **if** $L[\mathbf{p}_j] = \emptyset$ **then**
28  $L[\mathbf{p}_j] \leftarrow \langle 0, \infty \rangle$
29  $i' \leftarrow L[\mathbf{p}_j].first$, $\tau \leftarrow L[\mathbf{p}_j].second$
30  $\tau \leftarrow \min\{\tau, dist(\mathbf{p}_j, \mathbf{p}^{i'+1}), ..., dist(\mathbf{p}_j, \mathbf{p}^{i-1})\}$
31  $L[\mathbf{p}_j] \leftarrow \langle i-1, \tau \rangle$
32  **if** $f(\mathbf{p}^*_{int}, \mathbf{S}) < f(\mathbf{p}_j, \mathbf{S})$ **then**
33  $\mathbf{p}^*_{int} \leftarrow \mathbf{p}_j$
34  $\mathbf{S} \leftarrow \mathbf{S} \cup \{\mathbf{p}^*_{int}\}$
35  Update $\min_{\mathbf{p},\mathbf{p}' \in \mathbf{S}} dist(\mathbf{p}, \mathbf{p}')$ and $\min_{\mathbf{p} \in \mathbf{S}} \|\mathbf{p}\|^{\mathcal{M}}$

### 3.3 アルゴリズムの説明

提案アルゴリズムである IP-Greedy について説明する．このアルゴリズムは，1 回の前処理と与えられたクエリに対してオンライン処理を行う．

**前処理.** オフラインで各 $\mathbf{p} \in \mathbf{P}$ の $\|\mathbf{p}\|^{\mathcal{I}}$ および $\|\mathbf{p}\|^{\mathcal{M}}$ を計算する．これらのノルムはクエリに依存しない．次に，$\|\mathbf{p}\|^{\mathcal{I}}$ で全ての $\mathbf{p} \in \mathbf{P}$ を降順にソートする．

**オンライン処理.** アルゴリズム1は与えられたクエリに対する IP-Greedy を示す．全体のフレームワークは貪欲アルゴリズムと同様であるが，IP-Greedy は定理 1 および 2 から導かれるいくつかの操作を用いている．以下では，主な違いについて述べる．
(i) 最初の反復で，IP-Greedy はコーシー・シュワルツの不等式を利用し，網羅的な内積計算を回避しながら MIPS を実行する．そして，得られたベクトルで $\mathbf{S}$ を初期化する．また，IP-Greedy は $\mathbf{p} \cdot \mathbf{q}$ を既に計算されたアイテムベクトル $\mathbf{p}$ を保持するための集合 $\mathbf{T}$ を使用する．
(ii) その後，IP-Greedy は 12–35 行目に示す $\mathbf{S}$ のインクリメンタルな更新に移る．各反復において，まず $\gamma_j$ を更新することで $\mathbf{p}_j \in \mathbf{T}$ を再び順序付け，$\gamma$ に基づくソート順を維持する．そして，IP-Greedy は $\mathbf{P}$ の線形スキャンにより式 (4) の $\mathbf{p}^*$ を求める．この操作において，貪欲アルゴリズムとの違いが 3 つある．まず 1 つ目は，IP-Greedy が定理 1 から導かれる早期終了を利用していることである (17 行目)．2 つ目は，与えられたアイテムベクトルが早期終了の条件を満たさない場合においても，IP-Greedy はアイテムレベルのスキップにより，式 (3) の計算を避けようとすることである．（これは内積が未計算の場合において最大 2 回試行される．）3 つ目は，式 (6) を可能な限り効率的に計算するため，$\min\{dist(\mathbf{p}, \mathbf{p}^1), ..., dist(\mathbf{p}, \mathbf{p}^{i-2})\}$ を保持する代わりに，IP-Greedy は 28 行目で初期化されるログ集合 $L$ を保持している．早期終了とアイテムレベルのスキップにより，各 $\mathbf{p} \in \mathbf{P} \backslash \mathbf{S}$ が $\min\{dist(\mathbf{p}, \mathbf{p}^1), ..., dist(\mathbf{p}, \mathbf{p}^{i-2})\}$ を持っていることは保証されない．したがって，式 (6) を最初から計算することを避けるために，$L$ は式 (6) を最後に計算した反復と最小距離からなる組を保持する．

### 3.4 分析

IP-Greedy の空間計算量は $O(n)$ である．これは，(i) $|L| = n$ であるログ集合 $L$ および (ii) 各 $\mathbf{p} \in \mathbf{P}$ に対して，$\|\mathbf{p}\|^{\mathcal{I}}$，$\|\mathbf{p}\|^{\mathcal{M}}$，および $\mathbf{p} \cdot \mathbf{q}$（最悪の場合）を保持するからである．次に IP-Greedy の時間計算量を分析する．前処理に必要な時間は $O(n \log n)$ である．ここでは，オンライン時間に注目する．最初の繰り返し（1–11 行目）は $O((1-\alpha_1)n)$ 時間を必要とする．ただし，$\alpha_1$ はコーシー・シュワルツの不等式による枝刈り率である．$i$ 回目（$2 \leq i \leq k$）の反復の分析は最適化無しの貪欲アルゴリズムと同様である．しかし，IP-Greedy は早期停止を利用するため，$i$ 回目（$3 \leq i \leq k$）の反復において，$(1-\alpha_2)n$ 個のアイテムベクトルにしかアクセスしない．なお，$\alpha_2$ は $i$ 回目の反復における平均枝刈り率である．これより，各反復において再び順序付けるのに必要な時間は $O((1-\alpha_2)n \log n)$ であることが容易に分かる．アイテムレベルのスキップは $\Omega(d)$ 時間を除去するが，ここでは簡単のため $d = O(1)$ とする．したがって，IP-Greedy は 2 回目と $i$ 回目の反復で，それぞれ $\Theta(n)$ および $O((k + \log n)(1-\alpha_2)n)$ 時間を必要とする．また，$k$ はアクセスしたベクトルのログが $L$ に存在しないという最悪の場合から導かれる．以上より，IP-Greedy は $O((1-\alpha_1)n) + \Theta(n) + (k-2) \cdot O((k + \log n)(1-\alpha_2)n) = \tilde{O}((2-\alpha_1)n + k^2(1-\alpha_2)n)$ 時間を必要とする．

**備考 4.** IP-Greedy が各 $\mathbf{p} \in \mathbf{P}$ に対し，(i) $\mathbf{p} \cdot \mathbf{q}$ および (ii) $f(\mathbf{p}, \mathbf{S})$ を必ずしも求めるというわけでない．さらに，$\alpha_1$ および $\alpha_2$ が十分に大きい場合，上記の分析から IP-Greedy の時間計算量はおおよそ $\Theta(n)$ である．これは 4.2 節で示すように現実的に成立する．そして，多様化コストがほとんど無視でき，$k$ に対して頑健であることを意味する．（貪欲アルゴリズムと異なり，IP-Greedy の時間は $k$ に対して 2 次であるが，$\alpha_2$ により実際にはその影響は見られない．）以上の 2 点が貪欲アルゴリズムのような既存手法に対する IP-Greedy の主な利点である．

## 4 評価実験

### 4.1 設定

**データセット.** 1 から 5 段階のレーティングの集合である以下の 3 つの実世界のデータセットを使用した．

- Amazon-K [22]：Amazon Kindle の本に対するレーティングデータセットであり，アイテム数は 430,530.
- MovieLens：MovieLens 25M データセットであり，アイテム数は 59,047.
- Netflix[3]：Netflix Prize において用いられたレーティングデータセットであり，アイテム数は 17,770.

MF[11] を使用し，内積空間におけるクエリ（ユーザ）ベクトルおよびアイテムベクトルを生成した．また，アイテムについては Item2Vec[6] を使用し，ユークリッド空間への埋め込みを生成した．各ベクトルの次元数は 200 である．

**アルゴリズム.** 我々の実験では，以下のアルゴリズムを評価した．

- Swap [14]：これは MMR の発見的アルゴリズムである．$\mathbf{S}$ をランダムな $k$ 点で初期化する．そして，$\mathbf{S}\times\mathbf{P}\backslash\mathbf{S}$ において式 (2) の目的関数を最も改善する $\mathbf{p}'\in\mathbf{P}\backslash\mathbf{S}$ が存在する場合，それと $\mathbf{p}\in\mathbf{S}$ を入れ替える．一般に，この入れ替えは目的関数を改善できる組がなくなるまで繰り返される．我々は貪欲アルゴリズムと同様に，最大反復回数を $k-1$ とし，Swap は $O(k^3n)$ 時間を必要とする．
- Greedy：2.2 節で述べた $O(nk)$ 時間の貪欲アルゴリズム．
- IP-Greedy：3 章で述べたアルゴリズム．

我々は全てのアルゴリズムを C++ で実装し，g++ 5.5.0 に最適化オプション-O3 を付加し，シングルスレッドで実行した．

**評価指標.** 各アルゴリズムの性能を測定するため，以下の評価指標を使用した[4]．

- 実行時間：与えられたクエリの解 $\mathbf{S}$ を返すまでのオンライン時間．
- MMR：式 (2) の目的関数の値であり，高いほど良い．
- 最小距離：$\min_{\mathbf{p},\mathbf{p}'\in\mathbf{S}} dist(\mathbf{p},\mathbf{p}')$ の値．ただし，$c$ によってスケーリングはされていない．最小距離が大きいほど，$\mathbf{S}$ は多様化されていることを意味する．

ユーザベクトルの集合からランダムに 100 個のベクトルを選び，それらをクエリとした．各評価指標において平均値を報告する．なお，$k=10$ および $\lambda=0.5$ をデフォルトとする．

**実験環境.** 全ての実験は，3.0GHz の Core i9-9980XE CPU および 128GB の RAM を搭載した Ubuntu 16.04 LTS マシンで行った．

## 4.2 実験結果

**アブレーションスタディ.** 早期終了およびアイテムレベルのスキップ技術の性能が実行時間にどのように寄与しているかを確認する．表2は最初の反復処理とその他の反復処理で発生した実行時間を示す．

Greedy と IP-Greedy without skipping（定理 2 を用いない IP-Greedy）と比較すると，早期終了の影響が分かる．2 回目から 10 回目の反復に必要な時間に注目すると，早期終了は明らかに上手く機能している．IP-Greedy と IP-Greedy without skipping と比較すると，アイテムレベルのスキップの影響が分かる．その影響も明らかであり，IP-Greedy は 2 回目から 10 回目の反復に要する時間を大幅に短縮することができる．最後に，IP-Greedy が内積の計算回数を最小にすることにより，どれだけ効率を高めることができるかを確認する．そのために，IP-Greedy と IP-Greedy-exhaustive（最初の反復で全ての $\mathbf{p}\in\mathbf{P}$ について $\mathbf{p}\cdot\mathbf{q}$ を計算する IP-Greedy）を比較する．IP-Greedy はコーシー・シュワルツの不等式を利用して内積計算を削減するため，最初の反復に要する時間に関して，IP-Greedy-exhaustive に勝る．一方で, IP-Greedy は 2 回目から 10 回目の反復において IP-Greedy-exhaustive より通常遅くなる．これは当然な結果である．$\gamma\geq\mathbf{p}\cdot\mathbf{q}/\|\mathbf{q}\|$ より, IP-Greedy-exhaustive は IP-Greedy よりも常に定理 1 および 2 の上限がタイトであり（または同じである）からである．さらに, IP-Greedy は $\mathbf{p}\cdot\mathbf{q}$ が計算される場合，アイテムベクトル $\mathbf{p}$ を再び順序付けする必要がある．しかしながら，IP-Greedy は IP-Greedy-exhaustive よりも実行時間の合計が小さい．なぜなら，2 回目から 10 回目に反復における時間差は，最初の反復における時間差よりもはるかに小さいからである．

**$\lambda$ の影響.** 次に $\lambda$ が各アルゴリズムの性能にどのように影響するかを確認する．表3–5に実験結果を示す．Swap, Greedy，および IP-Greedy は $\lambda$ が異なる場合，解 $\mathbf{S}$ も異なる．そのため，MMR および最小距離は $\lambda$ に応じて変化する．なお，IP-Greedy は Greedy と同じ解を返すため，その MMR および最小距離は同じである．Swap と Greedy の実行時間は $\lambda$ が変化しても（ほとんど）変わらない．なぜなら，交換する組や追加するアイテムが網羅的に探索されるからである．一方，IP-Greedy の実行時間は $\lambda$ の影響を受ける．$\lambda$ が小さい場合，$f(\mathbf{p},\mathbf{S})$ は小さくなる傾向があるため，定理 1 および 2 の上限は $\lambda$ が小さいとルーズになる可能性がある．これは，$\lambda$ が小さいほど IP-Greedy の MMR が小さくなることから分かる．しかしながら，IP-Greedy は Greedy よりもはるかに高速である．

全体として，IP-Greedy の方が Swap よりも MMR が優れている．一方，Swap の最小距離が IP-Greedy のものよりも大きい場合がある．この場合，Swap が返す解のアイテムベクトルは，IP-Greedy が返すものより相対的に小さな内積を持ち，$\lambda$ が大きい場合には不適切である．（例えば，MovieLens や Netflix の場合である．）さらに，Swap は他の手法に比べ非常に遅く，IP-Greedy は Swap よりも最大で 100 倍速い．これらの結果から，IP-Greedy が Swap よりも有効性および効率性において優れていることが示される．

**$k$ の影響.** 次に，$k$ に対する頑健性を評価する．表3–5に示すように Swap は IP-Greedy より非常に遅い．したがって，Swap は考慮しない．

表6–8はそれぞれ $k$ が 5，15，および 20 の場合の結果である．（なお，$k=10$ のときは表3–5に示されている．）理論的に分析した結果通り，Greedy の実行時間は $k$ に対して線形である．IP-Greedy の実行時間は理論的には $k$ に対して 2 次であるが，現実的な実行時間は $k$ に対して劣線形（またはほとんど影響無し）である．この結果は，$\alpha_2$（3.4 節参照）が十分に大きいこと，つまり，$k$ が大きい場合においても定理 1 および 2 が上手く機能することを示唆している．これは，ユーザが大規模な推薦リストを望む場合における IP-Greedy の利点の 1 つでもある．例えば，（$k=10$ の元のリストから）$k=20$ のリストを見たい場合，IP-Greedy はリストを求めるのに 2 倍の時間を必要とせず，インタラクティブな推薦が可能である．

## 4.3 ケーススタディ

最後に，多様性を考慮した $k$-MIPS 問題の有効性を示すために定性的な評価を行った．MovieLens がほとんどのアイテム（映画）に関するメタ情報を持つのに対し，他のデータセットでは十分なメタ情報を持たないため，MovieLens を使用した．表1はユーザ A の 5-MIPS の結果（すなわち，$\lambda=1$ の場合）である．この結果が，多様性を考慮した $k$-MIPS 問題によって，どのように多様化するのかを示す．

[3] https://www.cs.uic.edu/liub/Netflix-KDD-Cup-2007.html
[4] 多様性を考慮した $k$-MIPS 問題は NP 困難のため，最良の解を得ることは現実的に不可能である．よって Recall のような精度を評価指標として使用することはできない．

**表 2: Greedy と IP-Greedy の分解時間 [msec]**

| | Amazon-K | | MovieLens | | Netflix | |
|---|---|---|---|---|---|---|
| 反復回数 | 1 | 2–10 | 1 | 2–10 | 1 | 2–10 |
| Greedy | 198.13 | 1873.81 | 23.99 | 230.42 | 6.77 | 57.47 |
| IP-Greedy without skipping | 62.36 | 509.21 | 7.21 | 59.24 | 2.03 | 16.19 |
| IP-Greedy-exhaustive | 197.46 | 70.69 | 23.53 | 7.94 | 5.84 | 2.31 |
| IP-Greedy | 61.97 | 85.89 | 6.67 | 7.79 | 1.53 | 2.90 |

**表 3: $\lambda$ が Amazon-K における実行時間 [msec]，MMR，および最小距離に与える影響**

| $\lambda$ | 0.25 | | | 0.5 | | | 0.75 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 実行時間 | MMR | 最小距離 | 実行時間 | MMR | 最小距離 | 実行時間 | MMR | 最小距離 |
| Swap | 12517.69 | 1.25 | 0.19 | 12542.50 | 2.46 | 0.18 | 12359.47 | 3.67 | 0.16 |
| Greedy | 2084.57 | 1.67 | 3.87 | 2071.96 | 2.50 | 0.87 | 2063.87 | 3.72 | 0.22 |
| IP-Greedy | 480.61 | 1.67 | 3.87 | 147.88 | 2.50 | 0.87 | 131.93 | 3.72 | 0.22 |

**表 4: $\lambda$ が MovieLens における実行時間 [msec]，MMR，および最小距離に与える影響**

| $\lambda$ | 0.25 | | | 0.5 | | | 0.75 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 実行時間 | MMR | 最小距離 | 実行時間 | MMR | 最小距離 | 実行時間 | MMR | 最小距離 |
| Swap | 1536.03 | 1.57 | 5.66 | 1555.24 | 2.62 | 4.55 | 1542.83 | 3.71 | 4.01 |
| Greedy | 253.36 | 1.59 | 5.94 | 254.42 | 2.63 | 3.86 | 253.51 | 3.76 | 3.58 |
| IP-Greedy | 30.90 | 1.59 | 5.94 | 14.48 | 2.63 | 3.86 | 14.30 | 3.76 | 3.58 |

**表 5: $\lambda$ が Netflix における実行時間 [msec]，MMR，および最小距離に与える影響**

| $\lambda$ | 0.25 | | | 0.5 | | | 0.75 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 実行時間 | MMR | 最小距離 | 実行時間 | MMR | 最小距離 | 実行時間 | MMR | 最小距離 |
| Swap | 397.64 | 1.50 | 3.58 | 403.42 | 2.39 | 2.74 | 408.11 | 3.37 | 2.14 |
| Greedy | 63.50 | 1.66 | 4.90 | 64.26 | 2.39 | 2.33 | 63.85 | 3.41 | 0.63 |
| IP-Greedy | 11.84 | 1.66 | 4.90 | 4.44 | 2.39 | 2.33 | 3.61 | 3.41 | 0.63 |

**表 6: $k$ が Amazon-K における実行時間 [msec]，MMR，および最小距離に与える影響**

| $k$ | 5 | | | 15 | | | 20 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 実行時間 | MMR | 最小距離 | 実行時間 | MMR | 最小距離 | 実行時間 | MMR | 最小距離 |
| Greedy | 1084.70 | 2.67 | 3.00 | 3064.10 | 2.48 | 0.50 | 4057.86 | 2.46 | 0.33 |
| IP-Greedy | 146.36 | 2.67 | 3.00 | 147.82 | 2.48 | 0.50 | 148.84 | 2.46 | 0.33 |

**表 7: $k$ が MovieLens における実行時間 [msec]，MMR，および最小距離に与える影響**

| $k$ | 5 | | | 15 | | | 20 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 実行時間 | MMR | 最小距離 | 実行時間 | MMR | 最小距離 | 実行時間 | MMR | 最小距離 |
| Greedy | 131.34 | 2.72 | 4.89 | 378.52 | 2.59 | 3.49 | 502.42 | 2.56 | 3.23 |
| IP-Greedy | 14.41 | 2.72 | 4.89 | 14.62 | 2.59 | 3.49 | 14.68 | 2.56 | 3.23 |

**表 8: $k$ が Netflix における実行時間 [msec]，MMR，および最小距離に与える影響**

| $k$ | 5 | | | 15 | | | 20 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 実行時間 | MMR | 最小距離 | 実行時間 | MMR | 最小距離 | 実行時間 | MMR | 最小距離 |
| Greedy | 27.38 | 2.56 | 4.56 | 93.60 | 2.32 | 1.34 | 123.20 | 2.28 | 0.78 |
| IP-Greedy | 2.80 | 2.56 | 4.56 | 4.65 | 2.32 | 1.34 | 4.67 | 2.28 | 0.78 |

$\lambda = 0.5$ とし，表9はユーザ A の場合を示す．このリストは IP-Greedy によって得られたものである．（Swap は非常に遅く，MMR が低いため考慮しない．）なお，$i$ 番目の映画は $\mathbf{p}^i$，つまり $i$ 番目の $\mathbf{p}^*$ に対応する．表9が示すように，明らかにユーザ A の推薦リストが多様化している．$\lambda = 1$ の場合，ハリーポッターの映画だけが含まれている．一方，$\lambda = 0.5$ の場合，リストにはファンタジー（Harry Potter and the Deathly Hallows Part 2），アニメーション（Darwyn Cooke's Batman 75th Anniversary），コメディ（A Midsummer Night's Dream および Jimmy Carr: Live），およびスリラー（Went the Day Well?）のジャンルが含まれており，ユーザとこれらの映画の内積は高い値を維持していることが分かる．この結果から，多様性を考慮した $k$-MIPS 問題は 1 章で述べた，(i) アイテムがユーザに関連し，(ii) リスト中のアイテムは互いに非類似であるという 2 つの重要な要素を満たしていることを示唆している．また，この結果から，ユーザは自分の要求に応じて多様性の度合いを調節できることを示している．

## 5 関連研究

**$k$-MIPS 問題.** 前述したように，これは MF に基づく推薦にとって必要不可欠な問題である．ユーザが推薦システム（またはサービス）を利用する時，できるだけ早く推薦リストを得るためには，$k$-MIPS 問題を効率的に解くことが重要である．そこで，多くの研究で $k$-MIPS に対する効率的なアルゴリズムが考案され，大規模な推薦システムをサポートしている．

**表 9: ユーザ A の推薦リストの多様化**

| | $\lambda = 1$ | | $\lambda = 0.5$ | |
|---|---|---|---|---|
| 順序 | タイトル | $\mathbf{p}^i \cdot \mathbf{q}$ | タイトル | $\mathbf{p}^i \cdot \mathbf{q}$ |
| 1 | Harry Potter and the Deathly Hallows Part 2 | 3.50 | Harry Potter and the Deathly Hallows Part 2 | 3.50 |
| 2 | Harry Potter and the Deathly Hallows Part 1 | 3.48 | Darwyn Cooke's Batman 75th Anniversary | 3.37 |
| 3 | Harry Potter and the Half-Blood Prince | 3.46 | A Midsummer Night's Dream | 3.31 |
| 4 | Harry Potter and the Order of the Phoenix | 3.45 | Jimmy Carr: Live | 3.29 |
| 5 | Harry Potter and the Goblet of Fire | 3.40 | Went the Day Well? | 3.21 |

[35] は，木構造に基づくデータ構造を用いて，正確性を保持しながら不要なベクトルの枝刈りを行うアルゴリズムを考案した．しかし，高次元空間では次元の呪いにより，木構造のデータ構造は機能しないことが知られている．そこで，最新の厳密な $k$-MIPS アルゴリズムである FEXIPRO[31] は線形スキャンに基づく手法を使用し，木構造に基づくアルゴリズムよりも非常に高速であることを確認した．

アプリケーションによっては，応答時間を短縮する必要がある場合，近似的な結果を許容するものもある．そこで，近似 $k$-MIPS アルゴリズムも考案された [12, 13, 21, 33, 38, 41]．これらのアルゴリズムは，サンプリング [13]，ハッシュ化 [38]，近接グラフ [33, 41]，または量子化 [12, 41] により，クエリベクトルと高い内積を持つであろうアイテムベクトルを含む $\mathbf{P}$ の部分集合にアクセスする．これらの近似アルゴリズムは，$k$-MIPS の処理を高速化できるが，与えられたクエリベクトルとの関連性のみに着目している．そのため，多様性を考慮した $k$-MIPS 問題を解くことはできない．

**結果の多様化.** Web[27]，機械学習 [47]，情報検索 [32]，データベース [18]，および推薦システム [48] においても多様化は最も重要な話題の 1 つである．文献 [14, 24] では，多様化の基準は，カバー率/カテゴリ，新規性，およびコンテンツ/セレンディピティの 3 つに大きく分かれると述べられている．カバー率/カテゴリは検索結果（あるいは推薦リスト）に含まれるカテゴリの数を最大化しようとするものである．新規性には厳密な定義は無く，“ユーザが知らないこと”，“ユーザがチェックしたアイテムと類似していないこと” などの要素がある．コンテンツ/セレンディピティは，新規性と密接に関連し [24]，結果集合における距離で測定される．本論文では，最後のケースを $k$-MIPS 問題に組み込み，距離（非類似度）を用いて結果を多様化する．4.3 節で示すように，この要素は実践的にカテゴリの多様化においても有効である．

結果集合を多様化するために，[15, 34] はリスト内の任意の 2 つのアイテム間の距離が少なくとも閾値 $r$ となるようにした．しかし，特に高次元空間では，与えられたデータセットのスケールや領域が不明確であることが多いため，適切な $r$ を知ることは困難である．我々の問題定義にはこの欠点が無い．既存研究 [9, 17, 18] では，top-$k$ 結果の多様化に取り組んでいる．[17] で提案された技術は低次元空間（$d \leq 3$）にのみ利用可能であり，[9, 18] は与えられたクエリとの関連性を考慮していない．学習に基づく結果の多様化も考案された．[47] はカバー率を考慮し，[52] は話題や URL などのメタ情報を使用した．[46] は MMR に基づくモデルを学習し，真のランキングを持つ学習データを必要とする．これらの学習手法は，特定の評価指標を損失関数として用いることで，その評価を向上させることを目的としている．したがって，上記の研究は我々の研究と全く異なっている．[30, 40, 46, 48] などの最近の研究では貪欲アルゴリズムが用いられているが，早期終了やスキップの手法については考慮されていない．我々の早期終了およびスキップ技術は，式（2）（より正確には式（4））に基づいて設計されているため，異なる問題に対する既存の早期終了およびスキップ技術は我々の問題に利用できない．

いくつかの研究 [10, 45] では，行列分解によってユーザベクトルおよびアイテムベクトルを生成する際に多様性を組み込んでいる．一見すると，それらは内積と多様性を考慮しているため我々の研究に非常に類似しているように思われるが，実際には異なる．例えば，[10] は多様化モデルを学習するためにラベル付きデータが必要であり，その推薦モデルはオンデマンドで多様性の度合いを調節できない．また，[45] も多様性の度合いをオンデマンドで調節できない．

## 6 まとめ

本論文では，行列分解に基づく推薦モデルの成功および多様化の有効性から，多様性を考慮した $k$-MIPS 問題を提案した．この問題を正確に解くことは NP 困難のため，貪欲アルゴリズムを検討した．しかし，既存の貪欲アルゴリズムを用いるだけでは不必要な計算コストが発生するため，我々は IP-Greedy を提案した．3 つの実世界データを用いた実験を行い，IP-Greedy の効率性および有効性を示した．さらに，実世界のデータセットを用いたケーススタディを行った．その結果，単純な $k$-MIPS 問題で得られるアイテム集合と比較して，多様性を考慮した $k$-MIPS 問題で得られるアイテム集合は多様であることが確認された．

## 謝辞

本研究の一部は，JST さきがけ（JPMJPR1931），JST CREST（JPMJCR21F2），および文部科学省科学研究費補助金・基盤研究 (A)(18H04095) の支援を受けたものである．

## 参考文献

[1] Mustafa Abdool, Malay Haldar, Prashant Ramanathan, Tyler Sax, Lanbo Zhang, Aamir Manaswala, Lynn Yang, Bradley Turnbull, Qing Zhang, and Thomas Legrand. 2020. Managing Diversity in Airbnb Search. In *KDD*. 2952–2960.

[2] Daichi Amagata and Takahiro Hara. 2021. Reverse Maximum Inner Product Search: How to efficiently find users who would like to buy my item?. In *RecSys*. 273–281.

[3] Ashton Anderson, Lucas Maystre, Ian Anderson, Rishabh Mehrotra, and Mounia Lalmas. 2020. Algorithmic Effects on the Diversity of Consumption on Spotify. In *The Web Conference*. 2155–2165.

[4] Vito Walter Anelli, Alejandro Bellogín, Tommaso Di Noia, and Claudio Pomo. 2021. Reenvisioning the Comparison between Neural Collaborative Filtering and Matrix Factorization. In *RecSys*. 521–529.

[5] Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. 2014. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *RecSys*. 257–264.

[6] Oren Barkan and Noam Koenigstein. 2016. Item2vec: Neural Item Embedding for Collaborative Filtering. In *International Workshop on Machine Learning for Signal Processing*. 1–6.

[7] Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*. 335–336.

[8] Ilio Catallo, Eleonora Ciceri, Piero Fraternali, Davide Martinenghi, and Marco Tagliasacchi. 2013. Top-k Diversity Queries over Bounded Regions. *ACM Transactions on Database Systems* 38, 2 (2013), 1–44.

[9] Laming Chen, Guoxin Zhang, and Eric Zhou. 2018. Fast Greedy MAP Inference for Determinantal Point Process to Improve Recommendation Diversity. *NeurIPS* 31 (2018).

[10] Peizhe Cheng, Shuaiqiang Wang, Jun Ma, Jiankai Sun, and Hui Xiong. 2017. Learning to Recommend Accurate and Diverse Items. In *World Wide Web*.

183–192.
[11] Wei-Sheng Chin, Bo-Wen Yuan, Meng-Yuan Yang, Yong Zhuang, Yu-Chin Juan, and Chih-Jen Lin. 2016. LIBMF: A Library for Parallel Matrix Factorization in Shared-memory Systems. *The Journal of Machine Learning Research* 17, 1 (2016), 2971–2975.
[12] Xinyan Dai, Xiao Yan, Kelvin KW Ng, Jiu Liu, and James Cheng. 2020. Norm-Explicit Quantization: Improving Vector Quantization for Maximum Inner Product Search. In *AAAI*. 51–58.
[13] Qin Ding, Hsiang-Fu Yu, and Cho-Jui Hsieh. 2019. A Fast Sampling Algorithm for Maximum Inner Product Search. In *AISTATS*. 3004–3012.
[14] Marina Drosou and Evaggelia Pitoura. 2010. Search Result Diversification. *ACM SIGMOD Record* 39, 1 (2010), 41–47.
[15] Marina Drosou and Evaggelia Pitoura. 2015. Multiple Radii Disc Diversity: Result Diversification Based on Dissimilarity and Coverage. *ACM Transactions on Database Systems* 40, 1 (2015), 1–43.
[16] Steven Essinger, Dave Huber, and Daniel Tang. 2021. AIR: Personalized Product Recommender System for Nike's Digital Transformation. In *RecSys*. 530–532.
[17] Piero Fraternali, Davide Martinenghi, and Marco Tagliasacchi. 2012. Top-k Bounded Diversification. In *SIGMOD*. 421–432.
[18] Xiaoyu Ge and Panos K Chrysanthis. 2020. Efficient PrefDiv Algorithms for Effective Top-k Result Diversification.. In *EDBT*. 335–346.
[19] Sreenivas Gollapudi and Aneesh Sharma. 2009. An Axiomatic Approach for Result Diversification. In *World Wide Web*. 381–390.
[20] Teofilo F Gonzalez. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science* 38 (1985), 293–306.
[21] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating Large-scale Inference with Anisotropic Vector Quantization. In *ICML*. 3887–3896.
[22] Ruining He and Julian McAuley. 2016. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-class Collaborative Filtering. In *World Wide Web*. 507–517.
[23] Zhengbao Jiang, Ji-Rong Wen, Zhicheng Dou, Wayne Xin Zhao, Jian-Yun Nie, and Ming Yue. 2017. Learning to Diversify Search Results via Subtopic Attention. In *SIGIR*. 545–554.
[24] Marius Kaminskas and Derek Bridge. 2016. Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-accuracy Objectives in Recommender Systems. *ACM Transactions on Interactive Intelligent Systems* 7, 1 (2016), 1–42.
[25] Mozhgan Karimi, Dietmar Jannach, and Michael Jugovac. 2018. News Recommender Systems–Survey and Roads Ahead. *Information Processing & Management* 54, 6 (2018), 1203–1227.
[26] Daniel James Kershaw, Rob Koeling, Stephan Bourgeois, Antonio Trenta, and Harriet J Muncey. 2021. Fairness in Reviewer Recommendations at Elsevier. In *RecSys*. 554–555.
[27] Vijay Keswani and L Elisa Celis. 2021. Dialect Diversity in Text Summarization on Twitter. In *The Web Conference*. 3802–3814.
[28] Matevž Kunaver and Tomaž Požrl. 2017. Diversity in Recommender Systems–A Survey. *Knowledge-based systems* 123 (2017), 154–162.
[29] Sudarshan Dnyaneshwar Lamkhede and Christoph Kofler. 2021. Recommendations and Results Organization in Netflix Search. In *RecSys*. 577–579.
[30] Chang Li, Haoyun Feng, and Maarten de Rijke. 2020. Cascading Hybrid Bandits: Online Learning to Rank for Relevance and Diversity. In *RecSys*. 33–42.
[31] Hui Li, Tsz Nam Chan, Man Lung Yiu, and Nikos Mamoulis. 2017. FEXIPRO: fast and exact inner product retrieval in recommender systems. In *SIGMOD*. 835–850.
[32] Yile Liang, Tieyun Qian, Qing Li, and Hongzhi Yin. 2021. Enhancing Domain-Level and User-Level Adaptivity in Diversified Recommendation. In *SIGIR*. 747–756.
[33] Jie Liu, Xiao Yan, Xinyan Dai, Zhirong Li, James Cheng, and Ming-Chang Yang. 2020. Understanding and Improving Proximity Graph Based Maximum Inner Product Search. In *AAAI*. 139–146.
[34] Lu Qin, Jeffrey Xu Yu, and Lijun Chang. 2012. Diversifying Top-k Results. *PVLDB* 5, 11 (2012), 1124–1135.
[35] Parikshit Ram and Alexander G Gray. 2012. Maximum inner-product search using cone trees. In *KDD*. 931–939.
[36] Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. 2020. Neural Collaborative Filtering vs. Matrix Factorization Revisited. In *RecSys*. 240–248.
[37] Brent Smith and Greg Linden. 2017. Two decades of recommender systems at Amazon.com. *IEEE Internet Computing* 21, 3 (2017), 12–18.
[38] Yang Song, Yu Gu, Rui Zhang, and Ge Yu. 2021. ProMIPS: Efficient High-Dimensional c-Approximate Maximum Inner Product Search with a Lightweight Index. In *ICDE*. 1619–1630.
[39] Harald Steck, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, and Justin Basilico. 2021. Deep Learning for Recommender Systems: A Netflix Case Study. *AI Magazine* 42, 3 (2021), 7–18.
[40] Zhan Su, Zhicheng Dou, Yutao Zhu, Xubo Qin, and Ji-Rong Wen. 2021. Modeling Intent Graph for Search Result Diversification. In *SIGIR*. 736–746.
[41] Shulong Tan, Zhaozhuo Xu, Weijie Zhao, Hongliang Fei, Zhixin Zhou, and Ping Li. 2021. Norm Adjusted Proximity Graph for Fast Inner Product Retrieval. In *KDD*. 1552–1560.
[42] Christina Teflioudi and Rainer Gemulla. 2016. Exact and Approximate Maximum Inner Product Search with LEMP. *ACM Transactions on Database Systems* 42, 1 (2016), 1–49.
[43] Christina Teflioudi, Rainer Gemulla, and Olga Mykytiuk. 2015. LEMP: Fast Retrieval of Large Entries in a Matrix Product. In *SIGMOD*. 107–122.
[44] Duong Chi Thang, Nguyen Thanh Tam, Nguyen Quoc Viet Hung, and Karl Aberer. 2015. An Evaluation of Diversification Techniques. In *DEXA*. 215–231.
[45] Jing Wang, Feng Tian, Hongchuan Yu, Chang Hong Liu, Kun Zhan, and Xiao Wang. 2017. Diverse Non-negative Matrix Factorization for Multiview Data Representation. *IEEE Transactions on Cybernetics* 48, 9 (2017), 2620–2632.
[46] Long Xia, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2015. Learning Maximal Marginal Relevance Model via Directly Optimizing Diversity Evaluation Measures. In *SIGIR*. 113–122.
[47] Le Yan, Zhen Qin, Rama Kumar Pasumarthi, Xuanhui Wang, and Michael Bendersky. 2021. Diversification-Aware Learning to Rank using Distributed Representation. In *The Web Conference*. 127–136.
[48] Rui Ye, Yuqing Hou, Te Lei, Yunxing Zhang, Qing Zhang, Jiale Guo, Huaiwen Wu, and Hengliang Luo. 2021. Dynamic Graph Construction for Improving Diversity of Recommendation. In *RecSys*. 651–655.
[49] Hamed Zamani and W Bruce Croft. 2020. Learning a Joint Search and Recommendation Model from User-Item Interactions. In *WSDM*. 717–725.
[50] Guangyi Zhang and Aristides Gionis. 2020. Maximizing Diversity over Clustered Data. In *SDM*. 649–657.
[51] Xing Zhao, Ziwei Zhu, Yin Zhang, and James Caverlee. 2020. Improving the Estimation of Tail Ratings in Recommender System with Multi-Latent Representations. In *WSDM*. 762–770.
[52] Yadong Zhu, Yanyan Lan, Jiafeng Guo, Xueqi Cheng, and Shuzi Niu. 2014. Learning for search result diversification. In *SIGIR*. 293–302.