大規模な推薦商品最適化問題に対する効率的な重み付き局所探索法

大阪大学・情報科学研究科　菅 貴博, 梅谷 俊治, 森田 浩
Takahiro Kan, Shunji Umetani, Hiroshi Morita
Graduate School of Information Science and Technology
Osaka University

§1　はじめに

近年, 多くのネット広告企業や電子取引では最適な商品やサービスを推薦するシステムが普及しており [1], 顧客に対する嗜好分析が行われている. 分析結果を元に行う商品推薦の例としては, ある顧客の嗜好と類似した他の顧客の情報を用いて自動的に推論を行う協調フィルタリング [2] や, 予め商品やサービスに応じてグループ化を行い, 顧客の嗜好と類似したものを推薦するコンテンツベースフィルタリング [3] などがある. 一方, 商品の多様性と顧客の効用にはトレードオフの関係があるため [4], そのバランスを保つために各商品の有用性と多様性を考慮した劣モジュラ関数が導入され [5], 貪欲法を用いた解法が一般的である [6]. しかし, 劣モジュラ関数を用いた定式化では多様な制約を扱うことは難しい.

そこで, 本研究では広告会社の事例を元に様々な制約を考慮した上で企業の利得を最大化する推薦商品最適化問題を考える. この問題では, 推薦時に発行するクーポンの費用に対する予算制約や各商品の最低利益を保証する利得制約なども考慮した 0-1 整数計画問題を扱う. 特に, 利得制約は推薦する商品の多様性という観点からも重要な制約である. 商品推薦に端を発する問題として同様に 0-1 整数計画問題で定式化した例 [7] では, 最大でも顧客数が数万, 商品数が数十程度のデータセットを扱っているが, 本研究では顧客数が数百万, 商品数が数百の大規模な実データを使用するため, この問題の線形緩和問題でさえ最適化ソルバーで解くことは難しい. この問題の実行可能解を求める方法として, 近傍を探索して解の改善を図る局所探索法は有効であるが, 一度の局所探索法では精度の悪い局所最適解に陥ることが多い.

そこで, 高精度な実行可能解を求めるために制約のペナルティ重みを適応的に更新しながら局所探索法を繰り返す重み付き局所探索法を用いる [11]. しかし, 本研究特有の問題点は大規模な問題を扱うため近傍が巨大であり, 妥当な計算時間内に一回の局所探索法が終了しないことである. そこで重み付き局所探索法の効率を向上させるために近傍を全て探索するのではなく, 近傍解のランダムサンプリングを導入する. 数値実験では, サンプリングによる効果を検証し, 提案手法を用いることで高精度な実行可能解を求められることを確認できた.

§2 推薦商品最適化問題

本節では, 推薦商品最適化問題に用いる数理計画法について説明する. 本研究では R 社から入手したデータを基に研究を行う. R 社では様々なサービスを提供するウェブサイトを運営している. まず本研究で扱う, あるウェブサイトの仕組みについて説明する. 顧客はウェブサイトを訪問し, 自分が利用したいと思う商品をウェブサイト上で探す. 顧客は商品を購入する時にウェブサイト上のクーポンを利用し, 購入する商品の値引きなどの特典を受けることが出来る. この時, クーポンの額面価格が R 社の費用として計上される. 顧客がクーポンを利用して商品を購入した時, 商品販売店から後日 R 社に謝礼が支払われる. この謝礼が R 社の利得になる. 以上がウェブサイトの仕組みである. また R 社は顧客に対してウェブサイトのトップページなどで顧客に商品を推薦できる. 推薦した商品のクーポン利用数はウェブサイト全体のクーポン利用数に占める割合が高いため, 顧客にどの商品を推薦するのかを決定することは R 社にとって重要な問題である.

本研究では R 社により顧客の嗜好分析は済んでおり, 任意の顧客と任意の商品の組に対してその顧客にその商品を推薦した時に発生する費用と利得は既知であると仮定する. その上で以下の 3 種類の制約の下でどの顧客にどの商品を推薦すれば利得の合計を最大化できるかを考える.

1. R 社の支払う費用の合計に上限を設ける (予算制約).
   一時的とはいえ, クーポンを発行したことにより, R 社は費用を負担する必要がある. その負担分に上限を設ける.
2. 各商品の利得の合計に下限を設ける (利得制約).
   R 社は実際に商品を提供しているのではなく, 商品の情報を提供している. また商品の提供者は広告掲載の代金を R 社に対して支払っている. したがって商品の提供者に対して推薦の実績を作ることはウェブサイトの運営上不可欠であり, 商品間で

不平等になりすぎないように商品の推薦の制約は必要である.

3. 各顧客に推薦する商品の数を一定にする.
   顧客への推薦手段として自社ウェブサイトへの掲載などが挙げられるが, 顧客によって掲載数が異なると都合が悪い. また大量の商品を推薦しても, 推薦する商品が少な過ぎても顧客のウェブサイトへの満足度が下がる. それらを防ぐための制約として本研究では顧客に推薦する商品の数は一定とする.

次に推薦商品最適化問題の定式化を行う. まず, 添字集合, 決定変数, 定数を定義しておく.

[添字集合]

I	: 顧客の添字集合
J	: 商品の添字集合

[決定変数]

x_{ij}	: 顧客 $i \in I$ に商品 $j \in J$ を推薦するならば 1, しないならば 0 をとる 0-1 変数

[決定変数]

c_{ij}	: 顧客 $i \in I$ に商品 $j \in J$ を推薦するためにかかる費用
g_{ij}	: 顧客 $i \in I$ に商品 $j \in J$ を推薦したときに得られる利得
C_{max}	: 顧客に商品を推薦するための総費用の上限
p_j	: 商品 $j \in J$ の利得の下限値
n	: 顧客 1 人に推薦する商品数

目的関数 $f(\boldsymbol{x})$ は次のように表現できる:

$$\text{maximize} \qquad f(\boldsymbol{x}) = \sum_{i \in I} \sum_{j \in J} g_{ij} x_{ij} \tag{1}$$

$\sum_{i \in I} \sum_{j \in J} g_{ij} x_{ij}$ は総利得を表し, その最大化が今回の目的である. 上記の制約を踏まえたうえで, 推薦商品最適化問題 (Goods Reccomendation Problem; GRP) は次のように定式化される.

$$
\begin{aligned}
\text{GRP}: \quad & \text{maximize} && \sum_{i\in I}\sum_{j\in J} g_{ij}x_{ij} && && \\
& \text{subject to} && \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} \le C_{max}, && && (2)\\
& && \sum_{i\in I} g_{ij}x_{ij} \ge p_j, && \forall j \in J, && (3)\\
& && \sum_{j\in J} x_{ij} = n, && \forall i \in I, && (4)\\
& && x_{ij} \in \{0,1\}, && \forall i \in I, \forall j \in J. && (5)
\end{aligned}
$$

1. 【(2):予算制約】
   左辺が総費用を表し, それを定数 C_{max} で上から抑える. これにより R 社の支払う費用の合計に上限を設ける制約を表現できる.
2. 【(3):利得制約】
   左辺が商品 $j \in J$ の利得の和を表し, それを定数 p_j で下から抑える. これにより各商品の利得の合計に下限を設ける制約を表現できる.
3. 【(4):各顧客に推薦する商品数の固定】
   左辺が顧客 $i \in I$ に推薦される商品数を表し, それが顧客に依存しない定数 n と等しくなる. これにより各顧客に推薦する商品の数を一定にする制約を表現できる.

また, 今回使用するデータは以下のような性質を持つ. 各商品に固有の α_j $(j \in J)$ という値を持ち, 以降はこの値を商品定数と呼ぶ.

$$
c_{ij} = \alpha_j g_{ij},\ \alpha_j \ge 0, \qquad \forall i \in I, \forall j \in J \tag{6}
$$

§3　局所探索法

前節の推薦商品最適化問題を解くために局所探索法を適用する. 局所探索法では総費用の上限は常に満たすように探索し, また目的関数の代わりに次の評価関数を導入する.

$$
z(\boldsymbol{x}) = f(\boldsymbol{x}) - \sum_{j\in J} w_j \max\left\{\frac{p_j - \sum_{i\in I} g_{ij}x_{ij}}{p_j}, 0\right\} \tag{7}
$$

この評価関数では, 元問題の目的関数から利得制約の違反量に対して重み $\boldsymbol{w}$ をかけたものの総和を差し引いている. この関数を用いる理由は目的関数値と利得制約違反の改善を

同時に行えるからである．また, ペナルティ重みの値を適応的に更新することで局所最適解に陥ることを防ぐことができる．ペナルティ重みの初期値を $\boldsymbol{w}$ とし, 更新を行うペナルティ重みを $\widetilde{\boldsymbol{w}}$ とする．

次に高澤 [10] によって提案された局所探索法で用いる二種類の近傍について説明する．一つ目の近傍は一顧客商品入替え近傍 (Shift Neighborhood) であり, この近傍では一人の顧客に推薦されている商品とそうでない商品を交換することで評価関数値を改善させる．一顧客商品入替え近傍 $N_i(\boldsymbol{x})$ は次のように定義される．

$$N_i(\boldsymbol{x}) = \{\boldsymbol{x}' \mid \text{現在解 } \boldsymbol{x} \text{ から一人の顧客 } i \text{ に推薦されている商品を入替えた集合 }\} \quad (8)$$

この近傍の大きさ $|N_i(\boldsymbol{x})|$ は, 各顧客に対し, $n(|J|-n)$ 通りの商品交換があり得るので,

$$|N_i(\boldsymbol{x})| = n|I|(|J|-n) = O(n|I||J|) \quad (9)$$

となる．最後にこのアルゴリズム (Shift Neighborhood Local Search, SHNLS) を **Algorithm 1** にまとめる．

Algorithm 1 SHNLS($\boldsymbol{x}$,$\widetilde{\boldsymbol{w}}$)

Input: 初期解 $\boldsymbol{x}$, ペナルティ重み $\widetilde{\boldsymbol{w}}$.

Output: 解 $\boldsymbol{x}$, 暫定解 $\boldsymbol{x}^*$.

```
START:
for i ∈ I do
    近傍 N_i(x) を生成する.
    for x' ∈ N_i(x) do
        if Σ_{i∈I} Σ_{j∈J} c_ij x'_ij ≤ C_max then
            if x' が実行可能解である then
                if f(x') > f(x*) then x* ← x'
            end if
            if z(x') > z(x) then
                x ← x'
                goto START
            end if
        end if
    end for
end for
```

二つ目の近傍は二顧客間商品交換近傍 (Swap Neighborhood) であり, この近傍では二人の顧客に推薦されている商品同士を交換することで評価関数値を改善させる. 一顧客商品入替え近傍と比較して, この近傍操作では利得制約に対する違反の変化量は小さい. 二顧客間商品交換近傍 $N_{i_1 i_2}(\boldsymbol{x})$ は次のように定義される.

$$N_{i_1 i_2}(\boldsymbol{x}) = \{\boldsymbol{x}' \mid \text{現在解 } \boldsymbol{x} \text{ から二人の顧客 } i_1, i_2 \text{ の間で推薦されている商品を交換した集合 }\} \tag{10}$$

各顧客に対し, n^2 通りの商品交換があり得るので, この近傍の大きさ $|N_{i_1 i_2}(\boldsymbol{x})|$ は

$$|N_{i_1 i_2}(\boldsymbol{x})| = n^2 \binom{|I|}{2} = O(n^2 |I|^2) \tag{11}$$

となる. 最後にこのアルゴリズム (Swap Neighborhood Local Search, SWNLS) を **Algorithm 2** にまとめる.

Algorithm 2 SWNLS($\boldsymbol{x}$,$\widetilde{\boldsymbol{w}}$)

Input: 初期解 $\boldsymbol{x}$, ペナルティ重み $\widetilde{\boldsymbol{w}}$.

Output: 解 $\boldsymbol{x}$, 暫定解 $\boldsymbol{x}^*$.

```
START:
for i_1, i_2 ∈ I (i_1 ≠ i_2) do
    近傍 N_{i_1 i_2}(x) を生成する.
    for x' ∈ N_{i_1 i_2}(x) do
        if Σ_{i∈I} Σ_{j∈J} c_ij x'_ij ≤ C_max then
            if x' が実行可能解である then
                if f(x') > f(x*) then x* ← x'
            end if
            if z(x') > z(x) then
                x ← x'
                goto START
            end if
        end if
    end for
end for
```

§4　大規模な問題に対する重み付き局所探索法

局所探索法では不十分な精度の局所最適解に陥ることが多いので, 様々な条件で局所探索法を繰り返すメタヒューリスティクスを考える. 本研究では重み付き局所探索法 (Weighting Local Search, WLS) を適用する [11]. 重み付き局所探索法とは, ペナルティ重みを適応的に更新しながら局所探索法を繰り返す手法であり, 解が局所最適解に陥ることを防ぐ.

重み付き局所探索法では, 局所探索法で解 $\boldsymbol{x}$ を得た後, ペナルティ重み $\widetilde{\boldsymbol{w}}$ を更新し, 再び局所探索法を行う. 梅谷 [8] によって提案されたペナルティ重み更新手続きに従い, ペナルティ重みの初期値 w_j $(j \in J)$ は十分大きな値に設定し, $\widetilde{\boldsymbol{w}} \leftarrow \boldsymbol{w}$ として以下のようにペナルティ重みを更新する. $\boldsymbol{x}^*$ を現在までに得られた最良の実行可能解 (暫定解) とする. $z(\boldsymbol{x}) \leq f(\boldsymbol{x}^*)$ となった場合, パラメータ γ $(0 < \gamma < 1)$ を用いて以下の式で均一にペナルティ重みを減少させる.

$$w_j \leftarrow \gamma w_j, \ \forall j \in J \tag{12}$$

そうでなければ, 次の更新式でペナルティ重みを増加させる.

$$w_j \leftarrow w_j + \frac{(z(\boldsymbol{x}) - f(\boldsymbol{x}^*))v_j(\boldsymbol{x})}{\sum_{j \in J} v_j(\boldsymbol{x})^2}, \ \forall j \in J \tag{13}$$

ただし, $v_j(x) = \max\left\{\frac{p_j - \sum_{i \in I} g_{ij}x_{ij}}{p_j}, 0\right\}$ $(j \in J)$ である. 最後にこのアルゴリズム (Weighting Local Search, WLS) を **Algorithm 3** にまとめる.

Algorithm 3 WLS($\boldsymbol{x}$)

Input: 初期解 $\boldsymbol{x}$.

Output: 最良暫定解 $\boldsymbol{x}^*$.

$\tilde{\boldsymbol{x}} \leftarrow \boldsymbol{x}, z^* \leftarrow -\infty, \widetilde{\boldsymbol{w}} \leftarrow \boldsymbol{w}$
repeat
    SHNLS($\tilde{\boldsymbol{x}}$,$\widetilde{\boldsymbol{w}}$) または SWNLS($\tilde{\boldsymbol{x}}$,$\widetilde{\boldsymbol{w}}$) を適用し, 改善解 $\tilde{\boldsymbol{x}}'$, 暫定解 $\boldsymbol{x}'$ を得る.
    $\tilde{\boldsymbol{x}} \leftarrow \tilde{\boldsymbol{x}}'$
    if $f(\boldsymbol{x}') > f(\boldsymbol{x}^*)$ **then** $\boldsymbol{x}^* \leftarrow \boldsymbol{x}'$
    if $z(\tilde{\boldsymbol{x}}) \leq f(\boldsymbol{x}^*)$ **then**
        $\widetilde{\boldsymbol{w}}$ を (12) に従って減少させる.
    else
        $\widetilde{\boldsymbol{w}}$ を (13) に従って増加させる.
    end if
until 制限時間に達する.

本研究で扱う大規模な問題では近傍の大きさも巨大であり, 実際の入力データであれば一回の局所探索法が制限時間以内に終了しなかった. そこで全近傍解を探索するのではなく, 全近傍解からランダムサンプリングを行い, 設定したサンプル数に達した時点で探索を打ち切り, 局所探索法を繰り返すことを考える. また, 一般に探索する近傍を大きくすれば, 得られる局所最適解の精度は向上するため [12], 一顧客商品入替え近傍と二顧客間商品交換近傍を交互に探索する. 一顧客商品入替え近傍では, 顧客全体の集合 I からランダムに顧客 i_1 を決め, その近傍 $N_i(\boldsymbol{x})$ から候補解を 1 つランダムサンプリングする. 二顧客間商品交換近傍では, 顧客全体の集合 I からランダムに顧客 i_1, i_2 $(i_1 \neq i_2)$ を決め, その近傍 $N_{i_1 i_2}(\boldsymbol{x})$ から候補解を 1 つランダムサンプリングする. 解の評価回数がサンプル数 s に達した場合, もしくは暫定解を更新した場合にアルゴリズムを終了する. 最後にこのアルゴリズム (Shift and Swap Neighborhood Randomized Local Search, SSNRLS) を **Algorithm 4** にまとめる.

Algorithm 4 SSNRLS($\boldsymbol{x}, \widetilde{\boldsymbol{w}}, s$)

Input: 初期解 $\boldsymbol{x}$, ペナルティ重み $\widetilde{\boldsymbol{w}}$, サンプル数 s.

Output: 解 $\boldsymbol{x}$, 暫定解 $\boldsymbol{x}^*$.

評価回数 $cnt \leftarrow 0$
while true **do**
  顧客全体の集合 I からランダムに顧客 i_1 を決める.
  近傍 $N_i(\boldsymbol{x})$ を生成し, ランダムに $\boldsymbol{x}' \in N_i(\boldsymbol{x})$ を得る.
  if $\sum_{i \in I} \sum_{j \in J} c_{ij} x'_{ij} \leq C_{max}$ **then**
    if $\boldsymbol{x}'$ が実行可能解である **then**
      if $f(\boldsymbol{x}') > f(\boldsymbol{x}^*)$ **then**
        $\boldsymbol{x}^* \leftarrow \boldsymbol{x}'$
        break
      end if
    end if
    if $z(\boldsymbol{x}') > z(\boldsymbol{x})$ **then** $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
  end if
  $cnt \leftarrow cnt + 1$
  if $cnt = s$ **then break**
  顧客全体の集合 I からランダムに顧客 i_1, i_2 $(i_1 \neq i_2)$ を決める.
  近傍 $N_{i_1 i_2}(\boldsymbol{x})$ を生成し, ランダムに $\boldsymbol{x}' \in N_{i_1 i_2}(\boldsymbol{x})$ を得る.
  if $\sum_{i \in I} \sum_{j \in J} c_{ij} x'_{ij} \leq C_{max}$ **then**
    if $\boldsymbol{x}'$ が実行可能解である **then**
      if $f(\boldsymbol{x}') > f(\boldsymbol{x}^*)$ **then**
        $\boldsymbol{x}^* \leftarrow \boldsymbol{x}'$
        break
      end if
    end if
    if $z(\boldsymbol{x}') > z(\boldsymbol{x})$ **then** $\boldsymbol{x} \leftarrow \boldsymbol{x}'$
  end if
  $cnt \leftarrow cnt + 1$
  if $cnt = s$ **then break**
end while

次に, 局所探索法で用いる初期実行可能解の生成方法について説明する. ある顧客 $i \in I$ と商品 $j \in J$ に対応する利得 g_{ij} を費用 c_{ij} で割った値は

$$\frac{g_{ij}}{c_{ij}} = \frac{1}{\alpha_j}, \qquad \forall i \in I, \forall j \in J \tag{14}$$

となり, 商品定数 α_j の逆数となる. この値が大きい商品を順に各顧客へ推薦することで初期解を構築する.

得られた初期解が実行不能であれば, 土谷 [9] によって提案されたペナルティ関数を利用して実行可能解へ変換する. この変換では初期解 $\boldsymbol{x}$ が与えられたとき, 次のペナルティ関数 $P(\boldsymbol{x})$ を目的関数とし, 一顧客商品入替え近傍と二顧客間商品交換近傍から交互に候補解 $\boldsymbol{x}'$ をランダムにサンプリングする. ペナルティ関数 $P(\boldsymbol{x}') = 0$ となるまで最小化を行い, 初期実行可能解 $\boldsymbol{x}'$ を得る.

$$P(\boldsymbol{x}) = \max\left\{\frac{\sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} - C_{max}}{C_{max}}, 0\right\} + \max\left\{\frac{p_j - \sum_{i\in I} g_{ij}x_{ij}}{p_j}, 0\right\} \tag{15}$$

§5 数値実験

本節では実際の入力データを用いた数値実験によって提案手法 SSNRLS を用いた重み付き局所探索法の性能を確認する. まず初めに 5.1 節で実験条件について説明する. 5.2 節では実際に R 社が解きたい問題規模である顧客数 $|I|$ =300 万人, 商品数 $|J|$ =500 個のデータを用意する.

5.1 実験条件

ここでは, 計算環境, 制約の設定, そして解の評価方法について述べる.

1. 計算環境
   - OS : macOS Sierra 10.12.6
   - プロセッサ : 2.7GHz 12cores intel Xeon E5
   - メモリ : 64GB 1866MHz DDR3
   - 言語 : C, コンパイラ : gcc4.2.1

2. 制約の制定

問題の制約に関するパラメータは土谷 [9] の方法を参考に定めた. 商品を顧客に推薦した場合の費用 c_{ij}, および利得 g_{ij} は R 社から提供されたデータを使用する. 推薦する商品数 N は 5 とし, $C_{max},\ p_j(j \in J)$ はそれぞれ

$$C_{max} = r_c \frac{N}{|J|} \sum_{i \in I} \sum_{j \in J} c_{ij}, \tag{16}$$

$$p_j = r_g \frac{N}{|J|} \sum_{i \in I} g_{ij} \qquad j \in J. \tag{17}$$

を用いる. $r_c,\ r_g\ (0 \leq r_c \leq 1,\ 0 \leq r_g \leq 1)$ はそれぞれ予算制約, 利得制約に関するパラメータである. 本研究のデータでは (6) の関係があるため, $r_c \geq r_g$ となり, $r_c - r_g$ が小さいほど問題の実行可能領域は狭くなる. 数値実験では, $r_c = 1.0$ で固定するため, r_g が 1.0 に近づくほど問題の難易度は上がる. 5.2 節では SHNLS, SWNLS, そして提案手法である SSNRLS(サンプル数 $S = 0.1 \times |I|, 1 \times |I|, 10 \times |I|, 100 \times |I|$) の全 3 種類の近傍探索を用いた重み付き局所探索法で, 顧客数 $|I|$ =300 万人, 商品数 $|J|$ =500 個, $(r_c, r_g) = (1.0, 0.5), (1.0, 0.7), (1.0, 0.9)$ の 3 通りの問題を解くことで, 提案手法の性能を評価する.

3. 解の評価方法

各問題について全手法の最良の下界値を求め, 相対ギャップを (最良の下界値 − 下界値)/(最良の下界値) として計算して解の評価を行う. また本研究では短い時間で高速に問題を解くことに重点を置いているので, 計算の制限時間は 3600 秒で一定とした. なお, データの入力時間は計算時間には含めない.

5.2 大規模な推薦商品最適化問題の求解

4 節で紹介した SHNLS, SWNLS, SSNRLS(サンプル数 $S = 0.1 \times |I|, 1 \times |I|, 10 \times |I|, 100 \times |I|$) の全 3 種類の近傍探索を用いた重み付き局所探索法で, 顧客数 $|I|$ =300 万人, 商品数 $|J|$ =500, $(r_c, r_g) = (1.0, 0.5), (1.0, 0.7), (1.0, 0.9)$ の 3 通りの問題を解いた結果を表 1〜表 3 にまとめた.

表 1　3 手法の比較, $(r_c, r_g) = (1.0, 0.5)$

近傍探索手法	サンプル数 S	重み更新回数	暫定解更新回数	相対ギャップ
SHNLS	-	0	23523590	2.39%
SWNLS	-	0	9201640	16.09%
SSNRLS	$0.1 \times \|I\|$	4945	4276	12.50%
SSNRLS	$1 \times \|I\|$	2058	1377	2.04%
SSNRLS	$10 \times \|I\|$	322	215	0.00%
SSNRLS	$100 \times \|I\|$	69	57	18.68%

表 2　3 手法の比較, $(r_c, r_g) = (1.0, 0.7)$

近傍探索手法	サンプル数 S	重み更新回数	暫定解更新回数	相対ギャップ
SHNLS	-	0	18110216	2.85%
SWNLS	-	0	7155793	17.16%
SSNRLS	$0.1 \times \|I\|$	4439	2822	5.76%
SSNRLS	$1 \times \|I\|$	1796	1056	1.02%
SSNRLS	$10 \times \|I\|$	289	182	0.00%
SSNRLS	$100 \times \|I\|$	54	42	10.94%

表 3　3 手法の比較, $(r_c, r_g) = (1.0, 0.9)$

近傍探索手法	サンプル数 S	重み更新回数	暫定解更新回数	相対ギャップ
SHNLS	-	0	6993931	2.71%
SWNLS	-	0	3442047	5.39%
SSNRLS	$0.1 \times \|I\|$	2129	4260	2.11%
SSNRLS	$1 \times \|I\|$	1587	804	0.27%
SSNRLS	$10 \times \|I\|$	257	151	0.00%
SSNRLS	$100 \times \|I\|$	43	31	0.17%

SHNLS については, 全問題に対して最も暫定解更新回数は多く, また問題の難易度差に対して相対ギャップは全て 2% 台で大きな差は見られなかった. 一方, SWNLS については, 難易度が一番高い $(r_c, r_g) = (1.0, 0.9)$ の問題の相対ギャップは $(r_c, r_g) = (1.0, 0.5), (1.0, 0.7)$ に比べて小さかった. このことから難易度が高い問題では, 利得

変化の小さい顧客間商品交換が発生しやすくなっていると考えられる．提案手法である SSNRLS について，サンプル数と重み更新回数の関係は，サンプル数が増加するほど重み更新回数は減少した．本実験ではサンプル数 $S = 10 \times |I|$ の時，$(r_c, r_g) = (1.0, 0.5), (1.0, 0.7), (1.0, 0.9)$ の 3 通りの全問題に対して最良の下界値を出力した．また，重み更新回数が増加すると暫定解更新回数も増加するが，これは重み更新により解が実行可能領域の境界へ移動しやすくなっているためである．

$(r_c, r_g) = (1.0, 0.9)$ の問題について，横軸に時間，縦軸に各手法が出力する下界値を取ったグラフを図 1 に示す．

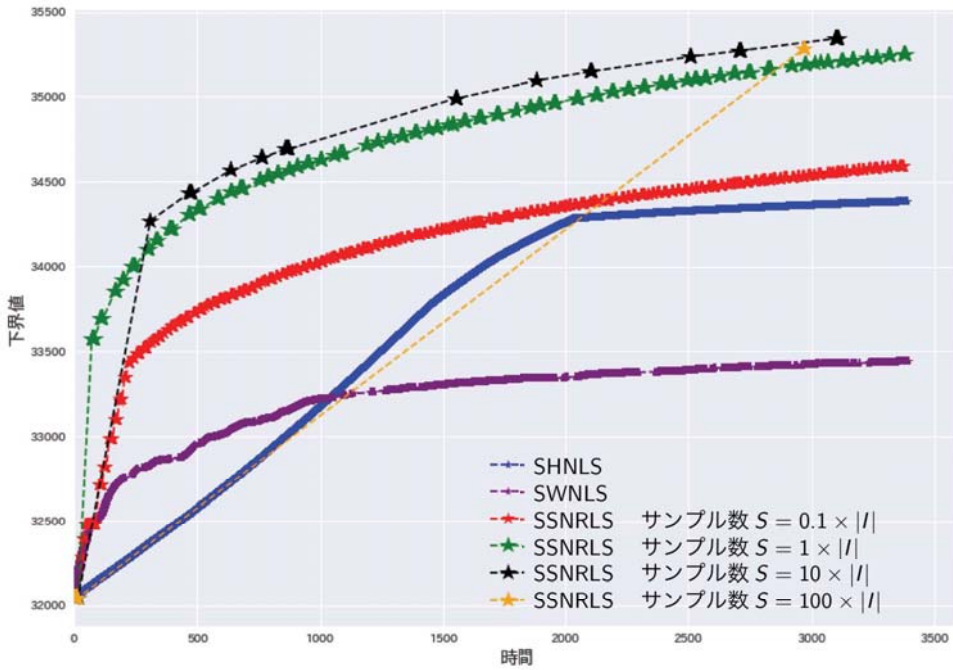

図 1　3 手法の時系列比較，$(r_c, r_g) = (1.0, 0.9)$

SHNLS, SWNLS は全近傍を探索してから重み更新を行うので，実行可能領域の境界付近を探索し，連続的に暫定解は更新される．一方，SSNRLS では暫定解が更新したらすぐに重み更新を行うため，実行不能領域の境界付近を探索し，断続的に暫定解が更新される．また，サンプル数が増加するほど暫定解の更新間隔は長くなるが，暫定解の精度は高くなる．

§6 結論

本研究では実務上必要な制約を満たした上でどの商品を顧客に薦めるかを決定する推薦商品最適化問題に取り組んだ．本研究で扱う問題は顧客数が数百万人, 商品数が数百個ある大規模な問題で解くことが難しく, また理論的にも NP-困難な問題クラスに属する．本研究ではこの問題の高精度な実行可能解を求めるために, 制約のペナルティ重みを適応的に更新しながら局所探索法を繰り返す重み付き局所探索法を提案した．本研究特有の問題点は大規模な問題を扱うため近傍が巨大であり, 妥当な計算時間内に一回の局所探索法が終了しないことである．そこで重み付き局所探索法の効率を向上させるために近傍を全て探索するのではなく, 近傍解のランダムサンプリングを導入した SSNRLS を提案した．

最後に数値実験を行い, 顧客数 300 万人, 商品 500 個の問題で, 提案手法である SSNRLS を用いた重み付き局所探索法がサンプル数 $S = 10 \times |I|$ の時, 最も高精度な下界値を得られることが確認できた．サンプル数と下界値の関係については, サンプル数が増加すると暫定解の更新回数は減少するが, 得られる暫定解の精度は高くなった．しかし, サンプル数を増やしすぎると下界値の精度が極端に減少する場合もあったため, 今回の提案手法では適切なサンプル数の設定が重要となる．

参考文献

[1] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors. *Recommender Systems Handbook*, Springer, 2011.

[2] Yehuda Koren and Robert M. Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145−186. 2011.

[3] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73−105. 2011.

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

[5] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *WSDM*, pages 5−14, 2009.

[6] G. L. Nemhauser, L. A.Wolsey, and M. L. Fisher. An analysis of approxima-

tions for maximizing submodular set functions - I. *Mathematical Programming*, 14(1):265−294, 1978.

[7] Fabrice Talla Nobibon, Roel Leus, Frits C.R.Spieksma, Optimization models for targeted offers in direct marketing: Exact and heuristic algorithms, *European Journal of Operational Research*, 210, pages 670−683, 2011.

[8] Shunji Umetani, Exploiting variable associations to configure efficient local search algorithms in large-scale binary integer programs, *European Journal of Operational Research*, 263, pages 72−81, 2017.

[9] 土谷拓人, 大規模な推薦商品最適化問題に対する確率的劣勾配法, 東京工業大学大学院, 2014.

[10] 高澤陽太朗, 大規模な推薦商品最適化問題に対する効率的ヒューリスティック解法の開発, 東京工業大学, 2015.

[11] Selman, B., Kautz, H. Domain-independent extensions to GSAT:Solving large structured satisability problems. Proceedings of *International Conference on Artificial Intelligence (IJCAI)*, pages 290−295.,1993.

[12] 柳浦睦憲, 茨木俊秀, 組合せ最適化 メタ戦略を中心として, 朝倉書店, 2001.

[13] T. Lu and C. Boutilier, Value-Directed Compression of Large-Scale Assignment Problems, *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 1182−1190, 2015.

Graduate School of Information Science and Technology
Osaka University
Osaka 565-0871
JAPAN
E-mail adress:takahiro.kan@ist.osaka-u.ac.jp

大阪大学・情報科学研究科　菅　貴博