数直線へのエージェント割当問題

Efficient algorithm for assigning agents to a line

久保 宰 (東京工業大学 工学院)
Tsukasa Kubo
School of Engineering
Tokyo Instituite of Technology

松井 知己 (東京工業大学 工学院)
Tomomi Matsui
School of Engineering
Tokyo Instituite of Technology

1 はじめに

本研究では数直線へのエージェント割当問題を扱う．この問題は, Hougaard et al. [2] で初めて提案された問題で, 割当先が数直線上に並んでおり, 各エージェントはターゲットにできるだけ近い場所に割り当てられることを好むという性質を持った特殊な割当問題である．この問題では, エージェントが順番に割り当てられ, エージェントは限られた時間のみサービスを利用できるというマッチング問題に応用できる．また, 経済学の観点からの応用もあり, 本問題に似た仮定をおいた研究がされている (Procaccia and Tennenholtz [6], Bogomolnaia and Moulin [7] など)．Aziz et al. [1] では, 総ギャップ最小化割当を計算する $O\left(n^3\right)$ アルゴリズムが提案された．(ただし n はエージェント数である．) 本研究では, さらに高速なアルゴリズムの提案を行う．

2 数直線へのエージェント割当問題の定義

数直線へのエージェント割当問題では, 数直線上に等間隔でスロットが並んでいる．このスロットにはエージェントを高々 1 人割り当てることができる．また, 各エージェントの最も好ましいスロットをターゲットと呼ぶ．ここで, 下記のように文字と言葉を定義する．

$N = \{1, 2, \ldots, n\}$：エージェントの集合.
x_i：エージェント i $(i \in N)$ の割当先を表す変数.
t_i：エージェント i $(i \in N)$ のターゲット．ただし, $t_1 \leq t_2 \leq \cdots \leq t_n$ を満たす．(満たしていない場合は, 満たすようにエージェントを整列する．)
$g_i(x) = |t_i - x_i|$：エージェント i のギャップ.
島：割当が連続しているエージェントの集合のうち, 極大なもの．(例えば, スロット 1 からスロッ

ト 8 までエージェントが割り当てられており, スロット 0 とスロット 9 にはエージェントが割り当てられていないとする. この場合, スロット 1 からスロット 8 に割り当てられているエージェントの集合は島である.) 島を数える際は, 左から数えることとする. すなわち, ○番目の島という言い方をした際は, 左から数えて○番目ということとする.

$\mathcal{I}(x)$：割当 x の島全てが含まれる集合.

$LP(x) = \{i \in N | x_i < t_i\}$：割当 x において, ターゲットより左側に割り当てられたエージェントの集合.

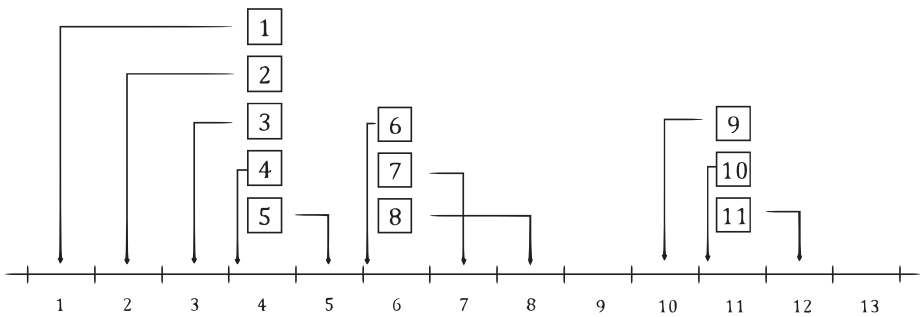
$TP(x) = \{i \in N | x_i = t_i\}$：割当 x において, ターゲット上に割り当てられたエージェントの集合.

$RP(x) = \{i \in N | x_i > t_i\}$：割当 x において, ターゲットより右側に割り当てられたエージェントの集合.

また, 島 I に含まれるエージェントのうち, 最も左に割り当てられたエージェントを $l(I)$, 最も右に割り当てられたエージェントを $r(I)$ とする.

上述の通り, エージェントのターゲットと割当先の距離をギャップと呼ぶが, 本研究では, 各エージェントのギャップの総和を最小化する割当を求めることを目的とする.

例 1. $N = \{1, 2, \ldots, 11\}, t = (4, 4, 4, 4, 4, 6, 6, 6, 11, 11, 11)$ とする. この時, 最適割当は $x = (1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12)$, 最適割当が形成する島は $\mathcal{I}(x) = \{\{1, 2, 3, 4, 5, 6, 7, 8\}, \{9, 10, 11\}\}$ である.



3 定式化と最適性条件

本節では, 数直線へのエージェント割当問題の定式化を行い, 双対問題を求めることで, 最適性条件を導く. まず, エージェントはターゲットが小さい順に左から並ぶことが Aziz et al. [1] で証明されているため, 定式化は以下のようにできる.

$$\text{(P): Minimize} \quad \sum_{i=1}^{n} |x_i - t_i| \tag{3.1}$$
$$\text{subject to} \quad x_i + 1 \leq x_{i+1} \ (i = 1, 2, \ldots, n-1), \tag{3.2}$$
$$x_i \in \mathbb{Z}. \ \ (i = 1, 2, \ldots, n). \tag{3.3}$$

この主問題 (P) は行列形式で書くと次のような問題 (P') として書くことができる.

$$\text{(P')}: \text{Minimize} \quad e^{\mathrm{T}}|x - t| \tag{3.4}$$
$$\text{subject to} \quad Hx \geq e, \tag{3.5}$$
$$x \in \mathbb{Z}^n. \tag{3.6}$$

ここで, e は各成分が 1 である $n-1$ 次元の列ベクトル, $|x-t|$ は第 i 成分が $|x_i - t_i|$ である $n-1$ 次元の列ベクトルである. また,

$$H = \begin{pmatrix} -1 & 1 & & & 0 \\ & -1 & 1 & & \\ & & & \ddots & \\ 0 & & & -1 & 1 \end{pmatrix}$$

である.

主問題 (P') に対し, 双対問題 (D) は以下のように定式化される.

$$\text{(D): Maximize} \quad (e^{\mathrm{T}} - t^{\mathrm{T}} H^{\mathrm{T}})v \tag{3.7}$$
$$\text{subject to} \quad |H^{\mathrm{T}} v| \leq e, \tag{3.8}$$
$$v \geq 0, v \in \mathbb{Z}^{n-1}. \tag{3.9}$$

この双対問題の算出を行うと同時に, 相補性条件が導出されるため, これが最適性条件となる. 最適性条件は下記のようになる.

定理 3.1. 次の命題 (A) と命題 (B) は同値である.

(A) 割当 x が最適割当である.

(B) 割当 x に対し, 以下の条件を全て満たす $d = (d_1, d_2, \ldots, d_n)$ が存在する.

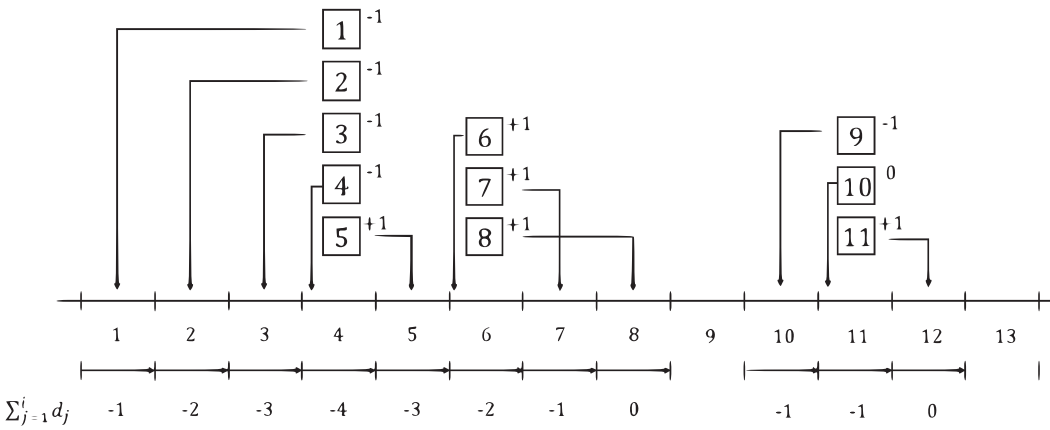
$$d_i = \begin{cases} -1 & (\ i \in LP(x) \text{ のとき}), \\ 1 & (\ i \in RP(x) \text{ のとき}), \\ -1, 0, 1 & (\ i \in TP(x) \text{ のとき}). \end{cases} \tag{3.10a}$$

任意の島 $I \in \mathcal{I}(x)$ に対して, $\sum_{i \in I} d_i = 0.$ (3.10b)

任意のエージェント $i \in N$ に対して, $\sum_{k=1}^{i} d_k \leq 0.$ (3.10c)

任意の島 $I \in \mathrm{I}(x)$ に対して, $\{i_1, i_2, \ldots, i_p\} = I \cap TP(x)\,(i_1 < i_2 < \cdots < i_p)$ としたとき, $d_{i_1} \leq d_{i_2} \leq \cdots \leq d_{i_p}$であり, $d_{i_1}, d_{i_2}, \ldots, d_{i_p}$のうち 0 のものは高々 1 つ. (3.10d)

例 2. 例 1 と同じ割当を考える. $d = (-1, -1, -1, -1, +1, +1, +1, +1, -1, 0, +1)$ とすると (3.10) をすべて満たすため, この割当は最適割当であることがわかる.



4 アルゴリズム

4.1 アルゴリズム 1

Aziz et al. [1] では, 総ギャップ最小化割当を計算する $O\left(n^3\right)$ アルゴリズムが提案された. このアルゴリズムではターゲットが同じエージェントを 1 つのエージェント集合とみなし, 左の集合から順に割り当て, 最適でない場合は解が改善しなくなるまで左にシフトする. 本節では, Aziz et al. [1] の $O\left(n^3\right)$ アルゴリズムのアイデアを利用した, アルゴリズム 1 を提案する. アルゴリズム 1 では, 番号が小さい順, すなわちターゲットが左にあるエージェントから 1 つずつ順に割り当てる. 割り当てる場所は, ターゲット上に他エージェントが割り当てられていなかった場合はターゲット上, そうでない場合は割当の再右端の一つ右とする. この割り当ての操作を行っているときに, 割当が最適でなくなった場合は, 即座に最も右の島を左に 1 つシフトする. 左に 1 つシフトすると割当が最適になるが, これは最適性条件を用いて証明できる. このアルゴリズム 1 の動きを疑似コード

で記述したものが下記になる．ただし, 島の接続とは, 島全体をシフトした際, 隣の島との間に空のスロットがなくなり, 隣の島と 1 つの島になったことを表すとする．

Algorithm アルゴリズム 1

入力: エージェントの集合 $N = \{1, \ldots, n\}$, ターゲット $t_i (i \in N)$ $(t_1 \leq t_2 \leq \cdots \leq t_n)$
出力: 割当 $x_i (i \in N)$

初期化 :$k \leftarrow 1, RI \leftarrow \emptyset$
LOOP Process
while $k \leq n$ **do**
  if $x_{k-1} < t_k - 1$ **then**
    $x_k \leftarrow t_k, d_k \leftarrow 0, RI \leftarrow \{k\}$
  else if $x_{k-1} = t_k - 1$ **then**
    $x_k \leftarrow x_{k-1} + 1, d_k \leftarrow 1, RI \leftarrow RI \cup \{k\}$, $m = \min\ RI \cap TP(x) \cap \{i | d_i \neq -1\}$ としたとき, $d_m \leftarrow d_m - 1$
  else
    $x_k \leftarrow x_{k-1} + 1, d_k \leftarrow 1, RI \leftarrow RI \cup \{k\}$
    if $d_j \neq -1\, (i \in RI \cap TP(x))$ となる j が存在する． **then**
      $m = \min\ RI \cap TP(x) \cap \{i | d_i \neq -1\}$ としたとき, $d_m \leftarrow d_m - 1$
    else
      全ての $i \in RI$ に対し, $x_i \leftarrow x_i - 1$
      if RI が左隣の島と接続した． **then**
        $RI \leftarrow RI \cup (RI$ の左隣の島$)$
      end if
      d_i を d **の設定方法**で設定する．
    end if
  end if
  $k \leftarrow k + 1$
end while
LOOP Process 終了

アルゴリズム 1 に対して, 以下の定理が成り立つ．

定理 4.1. アルゴリズム 1 は最適割当を得て終了する．

定理 4.2. アルゴリズム 1 の実行時間は $O(n^2)$ である．

4.2 アルゴリズム 2

前節のアルゴリズム 1 は, データ構造については特に工夫をしなかった. 本節では, データ構造を工夫し, 計算量を $O(n)$ としたアルゴリズム 2 を提案する. アルゴリズム 2 では, 割り当てやシフトについては, アルゴリズム 1 とまったく同じ要領で行う. 異なる点としては, エージェントのギャップの情報は, $list$ という名前のサイズ n の配列を用意し, その中で, 島ごとにギャップの情報を保持する. また, LP,TP,RP のエージェントの個数は, $surplus$ という配列にその情報を保持する. アルゴリズム 1 とまったく同じ要領で割り当てやシフトを行う際に, 割当が変わった場合は, $list$ と $surplus$ も更新する. また, シフトするかどうかの判断は, アルゴリズム 1 ではギャップの総和が小さくなるかどうかを計算することで行うが, アルゴリズム 2 では $surplus$ の値を見ることで判断できる. このことは最適性条件から証明できる. 具体的なアルゴリズムの動きは, 下記のようになる. ただし, この疑似コードでの文字は下記のようになる.

l, r : $l[q]$ は $q+1$ 番目の島の左端の位置, $r[q]$ は $q+1$ 番目の島の右端の位置を表す.

I_r : 反復終了時点での割当の島の個数から 1 を引いた値.

$list, position, ceiling$: $position, ceiling$ は昇順でソートされている. また, $position[q] > ceiling[q-1]$ (ただし $q = 0$ は除く) である. $list$ の $position[q]$ 番目から $ceiling[q]$ 番目までのセルには, $x' = \{x_i | i \in (q+1$ 番目の島$)\}$ としたとき, セル番号が小さい順に $TP(x'), RP_1(x'), RP_2(x'), \ldots$ が入る.

$surplus$: $x' = \{x_i | i \in (q+1$ 番目の島$)\}$ としたとき, $surplus[q] = |LP(x')| + |TP(x')| - |RP(x')|$.

ただし, 割当 x は, 上記の l, r から定まる割当である.

Algorithm アルゴリズム 2 Part 1

入力: エージェントの集合 $N = \{1, \ldots, n\}$, ターゲット $t_i (i \in N)\, (t_1 \leq t_2 \leq \cdots \leq t_n)$

出力: 割当先 $x_i (i \in N)$

初期化 :$k \leftarrow 1, I_r \leftarrow -1$,$list, position, ceiling, surplus, l, r \leftarrow$ サイズ n の空の配列
LOOP Process
while $k \leq n$ **do**
    if $r[I_r] < t_k - 1$(ただし $I_r = -1$ の時は真とする) **then**
        I_r += 1
        $ceiling[I_r], position[I_r] \leftarrow ceiling[I_r - 1] + 1$(ただし $I_r = -1$ の時は $\leftarrow 1$)
        $list[ceiling[I_r]] \leftarrow 1, surplus[I_r] \leftarrow 1, l[I_r], r[I_r] \leftarrow t_k$
    else if $r[I_r] = t_k - 1$ **then**
        $list[position[I_r]] \leftarrow 1, surplus[I_r]$ += $1, r[I_r] \leftarrow t_k$
    else
        $list[position[I_r] + r[I_r] + 1 - t_k]$ += $1, surplus[I_r]$ −= $1, r[I_r]$ += 1
        if $position[I_r] + r[I_r] + 1 - t_k > ceiling[I_r]$ **then**
            $ceiling[I_r]$ += 1
        end if
        if $surplus[I_r] = -1$ **then**
            $l[I_r]$ −= $1, r[I_r]$ −= $1, position[I_r]$ += 1
            $surplus[I_r] \leftarrow list[position[i_r]] * 2 - 1$
        end if
        if $r[I_r - 1] + 1 = l[I_r]$ **then**
            if $ceiling[I_r] - position[I_r] \geq ceiling[I_r - 1] - position[I_r - 1]$ **then**
                $I \leftarrow I_r, Z \leftarrow \{z \in \mathbb{Z} | position[I_r - 1] \leq z \leq ceiling[I_r - 1]\}$
            else
                $I \leftarrow I_r - 1, Z \leftarrow \{z \in \mathbb{Z} | position[I_r] \leq z \leq ceiling[I_r]\}$
            end if
            for i *in* Z **do**
                $list[i + position[I_r] - position[I_r - 1]]$ += $list[i]$
                $list[i] \leftarrow 0$
            end for
            $position[I_r - 1] \leftarrow position[I], ceiling[I_r - 1] \leftarrow ceiling[I]$
            $surplus[I_r - 1] \leftarrow surplus[I_r - 1] + surplus[I_r]$
            $r[I_r - 1] \leftarrow r[I_r]$
            $position[I_r], ceiling[I_r], surplus[I_r], l[I_r], r[I_r] \leftarrow 0$
            I_r −= 1
        end if
    end if
    $k \leftarrow k + 1$
end while
LOOP Process 終了

Algorithm アルゴリズム 2 Part 2

```
39: k ← 0
40: for i ← 0 to I_r do
41:     for j ← l[i] to r[i] do
42:         x_k ← j
43:         k ← k + 1
44:     end for
45: end for
```

このアルゴリズム 2 は, 最適割当を $O(n)$ の計算時間で算出できる. (ただし, エージェントをターゲットが小さい順にソートする $O(n \log n)$ の前処理が必要である.)

定理 4.3. アルゴリズム 2 は最適割当を得て終了する.

定理 4.4. アルゴリズム 2 の実行時間は $O(n)$ である. (ただし, エージェントをターゲットが小さい順にソートする $O(n \log n)$ の前処理が必要である.)

5 結論と考察

数直線へのエージェント割当問題を解くアルゴリズムとして, Aziz et al. [1] では, ギャップの総和を最小化する割当を計算する $O\left(n^3\right)$ アルゴリズムが提案された. 本研究では, まず, 数直線へのエージェント割当問題の最適性条件を導出した. 次に, Aziz et al. [1] の $O\left(n^3\right)$ アルゴリズムに対し, シフトするか判定する際に最適性条件を用いることと, エージェントを 1 人ずつ割り当てるようにする改良を施した, $O(n^2)$ アルゴリズムを提案した. さらに, この $O(n^2)$ アルゴリズムにおいて, ギャップの情報を保持するデータ構造を特別な配列にした, $O(n)$ アルゴリズムを提案した. (ただし, エージェントをターゲットが小さい順にソートする $O(n \log n)$ の前処理が必要である.) 数直線へのエージェント割当問題は, 例えばギャップがターゲットまでの距離であることや, スロットについての制約がないことなど, 考えやすい仮定が置かれている. このアルゴリズムをより一般的な問題設定でも応用できないかが, 今後の課題である.

参考文献

[1] Haris Aziz, Jens L. Hougaard, Juan D. Moreno-Ternero, and Lars P. Østerdal, "Computational aspects of assigning agents to a line," *Mathematical Social Sciences*, **86** (2017), pp. 68–74.

[2] Jens L. Hougaard, Juan D. Moreno-Ternero, and Lars P. Østerdal, "Assigning agents to a line," *Games and Economic Behavior*, **87** (2014), pp. 539–553.

[3] Harold W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, **2** (1955), pp. 83–97.

[4] Dimitri P. Bertsekas, "A distributed algorithm for the assignment problem," *Technical Report,MIT*, 1979.

[5] Dimitri P. Bertsekas, "The auction algorithm: A distributed relaxation method for the assignment problem," *Annals of Operations Research*, **14** (1988), pp. 105–123.

[6] Ariel D. Procaccia, and Moshe Tennenholtz, "Approximate mechanism design without money," *ACM Transactions on Economics and Computation*, **1** (2013).

[7] Anna Bogomolnaia, and Hervé Moulin, "A new solution to the random assignment problem," *Journal of Economic Theory*, **100(2)** (2001), pp. 295–328.

[8] Olvi L. Mangasarian, "Absolute value programming," *Computational optimization and Applications*, **36** (2007), pp. 43–53.