施設配置問題を解く効率的なアルゴリズムに関する研究

電子情報・数理領域　赤木 俊裕
指導教員　中野 眞一 教授

平成30年1月31日

目 次

図 目 次

概要

施設配置問題とは, 一般に, 指定したコストが最小となるような施設の配置を求める問題である. 多くの自然な問題があり, 様々な先行研究がある. また, データのクラスタリングや情報の匿名化などにも関連する. 代表的な施設配置問題に k-median 問題と k-center 問題がある. 利用者の集合と施設配置候補地の集合とそれらの間の距離が与えられたとき, k-median 問題は, 各利用者と最寄りの施設の間の距離の総和が最小となるように k 個の施設を配置する問題であり, k-center 問題は, 各利用者と最寄りの施設の間の距離の最大値が最小となるように k 個の施設を配置する問題である. しかし, k-median 問題や k-center 問題の解においては, 利用者が極端に多い施設や極端に少ない施設があるかもしれない.

これに対して, 本研究では, 各施設を必ず指定した人数 (r とする) 以上が利用するような施設の配置と, 利用者の施設への割り当てについて考える. このような制約を持つ施設配置問題を r-gathering 問題と呼ぶ. 本研究は, この問題および関連する問題を効率的に解くアルゴリズムを設計する.

また, 一般に, 施設配置問題では, 指定したコストが最小となるような施設の配置を求める. これに対し, 指定したコストが最大になるような施設配置問題を, 特に, dispersion 問題という. なるべく離れ離れに分散して何かを配置することが望ましいときに, そのような配置を求める問題である. 大量のデータから多様性のあるような少数のデータを選ぶ問題などにも関連する.

施設配置問題に関連する問題の多くは NP 困難と呼ばれる計算量のクラスに属し, 問題を解く多項式時間のアルゴリズムの設計は非常に困難であると予想される. そこで, 入力に制約を加えた問題に対する多項式時間アルゴリズムや, 最適解ではないが最適解に近い近似解を求める多項式時間アルゴリズムを設計することを目標とする.

本研究の主な成果について説明する. まず, r-gathering 問題を次のように定義する. n 人の利用者の集合と m 人の施設配置候補地の集合とそれらの間の距離が与えられたとき, どの開設施設も利用者が r 人以上であるようにいくつかの施設を開設し, 各利用者をいずれかの開設施設に割り当てたい. 利用者と割り当てた施設の間の距離の最大値をコストと呼び, このとき, このコストを最小化したい. この問題は NP 完全であることが知られている. この問題について, 以下の成果が得られた. まず, 利用者や施設配置候補地がすべて直線上の点とみなせるとき, r-gathering 問題を解く $O((m+n)\log(m+n))$ 時間のアルゴリズムを設計した. さらに, 利用者の集合から指定した人数の利用者を (外れ値として) 除外できるとき, 最適解を求めるアルゴリズムを設計した. また, 距離が三角

不等式を満たすとき, 最適なコストの高々3 倍のコストの解を求める $O(mn)$ 時間の近似アルゴリズムを設計した.

次に, dispersion 問題を次のように定義する. n 個の施設配置候補地から k 個の施設を開設したい. このとき, 開設した施設の間の距離の最小値をコストと呼び, このコストを最大化したい. この問題につい, 以下の成果が得られた. 施設配置候補地が円周上の点集合とみなせるとき, $k = 3$ のときの dispersion 問題を解く $O(n)$ 時間のアルゴリズムを設計した.

第1章　序論

1.1　背景

施設配置問題が古くから研究されている [9, 10]. 一般的な施設配置問題は, (1) 利用者の集合 C, (2) 施設の集合 F, (3) 施設の開設コスト $op : F \to \mathbb{R}$, (4) 利用者と施設の接続コスト $co : C \times F \to \mathbb{R}$, が与えられたとき, 指定されたコストが最小となるような開設施設の集合 $F' \subset F$ と割当 $A : C \to F'$ を求める問題である. 施設配置問題に関連する問題としてクラスタリング問題や dispersion 問題がある.

一般的なクラスタリング問題は, (1) 集合 C, (2) C 中 2 つの要素間の距離コスト $d : C \times C \to \mathbb{R}$, 整数 k, が与えられたとき, 指定されたコストが最小となるような集合 C の k 個の部分集合への分割 $\mathcal{C}$ を求める問題である. これは, $F = C$, $op = 0$ とした施設配置問題の特殊な場合の問題とみなすことができる.

一般的な dispersion 問題は, (1) 施設の集合 F, (2) F 中の 2 つの要素間の距離コスト $d : P \times P \to \mathbb{R}$, 整数 k, が与えられたとき, 指定されたコストが最大となるような k 個の開設施設の集合 $F' \subset F$ を求める問題である.

本論文では, 最近提唱された施設配置問題である r-gathering 問題 [7], およびこれに関連する k-anonymity 問題 [1], ℓ-diversity 問題 [17], dispersion 問題 [15, 16, 20, 23] を扱う.

従来の施設配置問題の解において, 利用者が極端に少ない開設施設があるかもしれない. どの開設施設も指定した人数以上の利用者がいることが望ましいことが多い. そこで, このような開設施設の選び方と利用者の施設への割当を求めたい. 各開設施設に利用者が r 人以上割り当てられるような, 利用者 C の開設施設 $F' \subset F$ への割当 A を r-gathering という. 従来の施設配置問題の割当は $r = 1$ のときの r-gathering に相当する. 本文では, r-gathering のコストを $\max\{\max_{c \in C}\{co(c, A(c))\}, \max_{f \in F'}\{op(f)\}\}$ と定義する. このコストが最小となる r-gathering を求める問題を r-gathering 問題という. 一方, 関連する問題として, コストを min-sum で定義した r-gathering 問題もある. これについては [7] を参照されたい.

この問題は, 例えば次のような避難計画問題のモデルになっている. n 人の住民の集合を C とし, m 個の避難場所の候補地の集合を F とする. 住民 $c \in C$ が避難場所 $f \in F$ へ避難するのにかかる避難時間を $co(c, f)$ とし, 避難場所 $f \in F$ の開設にかかる準備時間を $op(f)$ とする. このとき, 各開設避難場所に r 人以上の住民が避難するような, 住民の

避難場所への割当が r-gathering に相当する．また, 避難が完了するのにかかる時間を最小にする避難場所の選び方と住民の避難場所への割当を求める問題が r-gathering 問題である．

Armon は r-gathering 問題を解く $O(mn + rn + n \log n)$ 時間の 3 倍近似アルゴリズムを与え, また, $P \neq NP$ ならば, 任意の $r \geq 3$ について, 近似比 3 を改善することは NP 困難であることを証明した [7]．本論文では C と F が直線上の点集合とみなせるとき, r-gathering 問題を解く $O((m+n)\log(m+n))$ 時間アルゴリズムを与える．また, Armon の 3 近似アルゴリズムの計算時間を $O(mn)$ 時間に改善したアルゴリズムを与える．

個人の秘密を保護しつつ, データを公開することがデータの活用において望ましい．氏名やマイナンバーなどの個人を特定する情報を “識別子” といい, 生年月日, 性別, 郵便番号などの他のデータと組み合わせることで高い確率で個人を特定できる情報を “準識別子” という．また, 病気のような秘密にしたい情報を “秘密情報” という．識別子を削除すればデータを公開しても個人の秘密は保護できるわけではないことが報告されている [21]．データから識別子を削除しても, 選挙人名簿などの他の公開情報と組み合わせることで, 高い確率でデータに対応する個人が特定できる [21]．そこで, データに対応する個人が特定できないように, 公開するデータを匿名化する手法やモデルが必要となる．

匿名化の手法のひとつに, 準識別子の値に幅をもたせたり, 準識別子を上位の概念に置き換えたりすることで曖昧性を持たせる手法がある．これをデータの “一般化” という．この一般化によりデータに対応する個人が特定されにくくなるが, データが曖昧になってしまう．一般に, 匿名性を保証しつつ, 曖昧性ができるだけ少なくなることが望ましい．

準識別子の各データについて, 同じ値のデータを持つレコードが k 個以上あるとき, そのレコードの集合は k-anonymity 性を持つという [1]．例えば, ある人が 25 歳で体重が 65kg であることを知っているとき, 表 1.1 のレコードの集合では, その人はレコード A に対応することがわかってしまう．一方, 表 1.2 のレコードの集合中では, データの一般化により, その人は 3 個のレコード A,D,F のうち, どのレコードがその人に対応するかはわからない．k-anonymity 性を持つレコードの集合は, どのレコードも, 同じ準識別子を持つデータのレコードが他に $k-1$ 個以上あり, 他の公開情報と組み合わせても, どのレコードが指定した人のものなのかを特定することは困難である．このとき, 準識別子の一般化による情報損失をコストと定義し, コストが最小であるように, k-anonymity 性を持つようにデータを一般化する問題を k-anonymity 問題という．この問題は NP 困難であることが知られている [14]．Aggarwal らは, レコードの集合を点集合とみなせるとき, k-anonymity 問題を解く 2 近似アルゴリズムを与えた [1].

一般に, 施設配置問題では, 倉庫や図書館など, 利用者の近くにある事が望ましい施設の最適な配置について考えている．これに対して, チェーンストアやごみ処理施設などなるべく分散して配置することが望ましい施設の配置を求める問題が dispersion 問題である．

氏名	年齢	体重	病歴
A	25	65	心疾患
B	37	81	心疾患
C	49	78	脳血管
D	33	69	脳血管
E	67	74	肺炎
F	40	57	肺炎
G	29	77	心疾患
H	52	63	脳血管

表 1.1: オリジナルデータ

氏名	年齢	体重	病歴
A	25∼40	57∼69	心疾患
D	25∼40	57∼69	脳血管
F	25∼40	57∼69	肺炎
B	29∼37	77∼81	心疾患
G	29∼37	77∼81	心疾患
C	49∼67	63∼78	脳血管
E	49∼67	63∼78	肺炎
H	49∼67	63∼78	脳血管

表 1.2: 2-anonymity 性を持つレコードの集合

n 個の施設候補地の集合 P から k 個の開設施設の集合 S を, 指定したコストが最大となるように選ぶ問題を dispersion 問題という [20, 23]. たとえば, 代表的な S のコストは $\min_{u,v\in S}\{d(u,v)\}$ である. また, この他にも, 次に定義するような partial c sum コストがある [15, 16].

各開設施設 $p \in S$ のコストは, p に最も近い S 中の c 個の開設施設への距離の和とし, これらの最小値を S の partial c sum コストとする. この partial c sum コストが最大となるような $S \subset P$ を求める問題を partial c sum dispersion 問題という. 従来の dispersion 問題は $c = 1$ のときの partial c sum dispersion 問題に相当する.

Dispersion 問題は NP 困難であることが知られている [20, 23]. また, P が直線上の点集合とみなせるとき, dipsersion 問題を解く動的計画法による $O(n \log n + kn)$ 時間アルゴリズム [20] が知られている. また, パス分割問題へ帰着してこれを行列探索法 [13] で解くことにより, $O(n)$ 時間で解くこともできる.

本論文では, P が直線上の点集合とみなせるとき, partial 2 sum dispersion 問題を解く $O(n \log n)$ 時間アルゴリズムを与える.

1.2 論文構成

本文の構成は次の通りである.

2 章では, 本論文で扱う問題について定義する. 3 章では, r-gathering 問題とそれに関連する問題を解くアルゴリズムを与える. 4 章では, r-gathering 問題とそれに関連する問題を高速に解く近似アルゴリズムを与える. 5 章では, dispersion 問題を解くアルゴリズムを与える. 6 章では, dispersion 問題に関連する問題を解くアルゴリズムを与える. 最後に, 7 章は結論である.

第2章　定義

2.1　r-gathering 問題

2.1.1　r-gathering 問題

集合 T の各要素に集合 S の要素を r 個以上割り当てた割当 $A : S \to T$ を S の T への r-gathering という.

利用者の集合 C, 施設の集合 F, 利用者から施設への接続コスト $op : C \times F \to \mathbb{R}$, 施設の開設コスト $op : F \to \mathbb{R}$, 整数 r, が与えられたとする. C の $F' \subset F$ への r-gathering A の接続コストの最大値 $\max_{c \in C}\{co(c, A(c))\}$ と開設コストの最大値 $\max_{f \in F'}\{op(f)\}$ の最大値を A のコストとする. このとき, コストが最小となるような, 開設施設の集合 $F' \subset F$ と利用者の開設施設への r-gathering $A : C \to F'$ を求める問題を r-gathering 問題という.

2.1.2　r-gathering with h-outlier 問題

利用者の集合 C, 施設の集合 F, 利用者から施設への接続コスト $op : C \times F \to \mathbb{R}$, 施設の開設コスト $op : F \to \mathbb{R}$, 整数 r, h, が与えられたとする.

このとき, A のコストが最小となるような, 高々 h 人の利用者の集合 C' と開設施設の集合 $F' \subset F$, と利用者の部分集合 $C \backslash C'$ の開設施設 $F' \subset F$ への r-gathering $A : C \backslash C' \to F'$ を求める問題を r-gathering with h-outlier 問題という.

2.1.3　reserved r-gathering 問題

利用者の集合 C, 施設の集合 F, 施設の部分集合 $F^o \subset F$, 利用者から施設への接続コスト $op : C \times F \to \mathbb{R}$, 施設の開設コスト $op : F \to \mathbb{R}$, 整数 r, が与えられたとする.

このとき, A のコストが最小となるような, 開設施設の集合 $F^o \subset F' \subset F$ と利用者の部分集合 $C \setminus C'$ の開設施設 $F' \subset F$ への r-gathering $A : C \setminus C' \to F'$ を求める問題を reserved r-gathering 問題という.

2.2 Dispersion 問題

2.2.1 dispersion 問題

施設候補地の集合 P, 施設間の距離コスト $d : P \times P \to \mathbb{R}$, 整数 k, が与えられたとする. ここで, $|S| = k$ である P の部分集合 $S \subset P$ を考える. 各点 $u \in S$ から最も近い $S \setminus \{u\}$ 中の点への距離 $\min_{v \in S \setminus \{u\}}\{d(u, v)\}$ を点 $u \in S$ のコスト $cost(u)$ とし, これらのうち最小のコスト $\min_{u \in S}\{cost(u)\}$ を S のコストとする. このとき, コストが最大となるような, $|S| = k$ である P の部分集合 $S \subset P$ を求める問題を dispersion 問題 [15, 16] という.

2.2.2 partial c sum dispersion 問題 (PcS-dispersion 問題)

施設候補地の集合 P, 施設間の距離 $d : P \times P \to \mathbb{R}$, 整数 k, c, が与えられたとする. ここで, $|S| = k$ である P の部分集合 $S \subset P$ を考える. 各点 $u \in S$ から最も近い $S \setminus \{u\}$ 中の c 個の点への距離の和を点 $u \in S$ のコスト $cost_{PcS}(u)$ とし, これらのうち最小のコスト $\min_{u \in S}\{cost_{PcS}(u)\}$ を S のコストとする. このとき, コストが最大となるような, $|S| = k$ である P の部分集合 $S \subset P$ を求める問題を partial c sum dispersion 問題 [15, 16] という.

第3章　直線上のr-gathering問題を解くアルゴリズム

本章では, r-gathering 問題を扱う. 3.1 節では, C と F が直線上の点集合とみなせるときの r-gathering 問題を定義する. 3.2 節では, この直線上の r-gathering 問題の判定問題を $O(m+n)$ 時間で解くアルゴリズムを与える. 3.3 節では, 直線上の r-gathering 問題を $O((m+n)\log(m+n))$ 時間で解くアルゴリズムを与える. 3.4 節と 3.5 節では, 直線上の r-gathering 問題を一般化した問題を定義し, これを解くアルゴリズムを与える. ここで, $|C|=n$, $|F|=m$ である.

3.1　直線上のr-gathering問題

水平直線上の, 点集合 (利用者の集合)$C=\{c_1,c_2,\ldots,c_n\}$ と, 点集合 (施設候補地の集合)$F=\{f_1,f_2,\ldots,f_m\}$ と, 施設の開設コスト $op:F\to\mathbb{R}$ と, 整数 r が与えらえれたとする. ただし, 点集合の各点は左から右へ順に並んでいると仮定する. 利用者 $c\in C$ が開設施設 $f\in F'\subset F$ に割り当てられたとき, 接続コスト $co(c,f)$ は 2 点間の距離とする. 各 $f\in F'\subset F$ について $|\{c\mid A(c)=f\}|\geq r$ を満たすような, C から $F'\subset F$ への割当 A を r-gathering という. r-gathering A のコストを $\max\{\max_{c\in C}\{co(c,A(c))\},\max_{f\in F'}\{op(f)\}$ とする. コストが最小の r-gathering を求める問題を r-gathering 問題という. 特に $r=1$ のときは, 通常の施設配置問題である.

r-gathering $A:C\to F$ において, $i'<i$ なる $c_{i'},c_i\in C$ で $A(c_{i'})>A(c_i)$ なるものがあるとき, A 中で $c_{i'}\to A(c_{i'})$ と $c_i\to A(c_i)$ は **overlap** であるという. (図 3.1, 3.2 参照.) 次の補題が成り立つ.

補題 3.1.　r-gathering 問題に解が存在するとき, overlap がない解がある.

証明.　ある r-gathering 問題において, overlap がある解しかないと仮定する. このとき, overlap が最小個の r-gathering を $A:C\to F'$ とする. また, A のコストを OPT とする.

$c_h,c_i\in C$ $(h<i)$, $f_j,f_k\in F$ $(j<k)$ とし, $A(c_h)=f_k$ が $A(c_i)=f_j$ より右にあり, A 中の c_h と c_i が overlap であるとする.

ここで, A 中の c_h,c_i の割当のみを $A(c_h)=f_j$, $A(c_i)=f_k$ のように変更した割当を A' とする.

図 3.1: overlap がない割当の例.

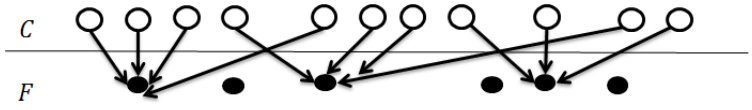

図 3.2: overlap がある割当の例.

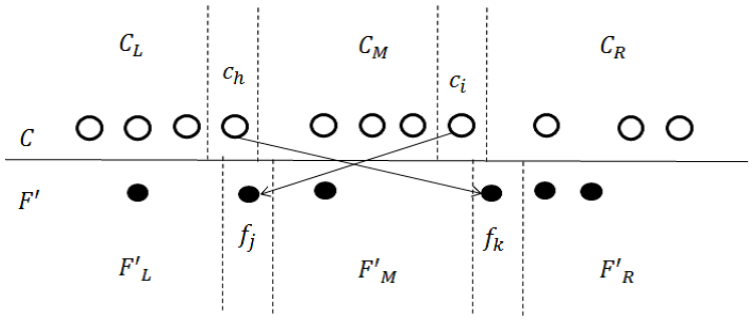

図 3.3: c_h と c_i の overlap の例.

A' の各開設施設に割り当てられている利用者の人数は A と等しいので, A' は r-gatehring である.

また, $h < i$ かつ $j < k$ より $\max\{co(c_h, f_k), co(c_i, f_j)\} \geq \max\{co(c_h, f_j), co(c_i, f_k)\}$ が成り立つ. すなわち, A' のコストが OPT より大きくなることはない. (コストは接続コストか開設コストの最大値であることに注意する.)

次に, A' の overlap の個数は A より 1 個以上少ないことを示す.

c_h より左側の利用者の集合 $\{c_x \in C \mid x < h\}$ を C_L とし, c_h より右側かつ c_i より左側の利用者の集合 $\{c_x \in C \mid h < x < i\}$ を C_M とし, c_i より右側の利用者の集合 $\{c_x \in C \mid i < x\}$ を C_R とする. 同様に, f_j より左側の開設施設の集合 $\{f_y \in F' \mid y < j\}$ を F'_L とし, f_j より右側かつ f_k より左側の開設施設の集合 $\{f_y \in F' \mid j < y < k\}$ を F'_M とし, f_j より右側の開設施設の集合 $\{f_y \in F' \mid k < y\}$ を F'_R とする.

このとき, A から A' への変更による ovprlap の個数の変化について考える.

$c_l \in C_L$ から $f_l \in F'_L$ への割当において, c_l は A 中の c_h, c_i とも A' 中の c_h, c_i とも overlap しない. よって, 変更により overlap の個数は変わらない. $c_r \in C_R$ から $f_r \in F'_R$ への割当についても同様である.

$c_l \in C_L$ から $f_r \in F'_R$ への割当において, c_l は A 中の c_h, c_i と 1 回ずつ overlap し, A'

中の c_h, c_i とも 1 回ずつ overlap する．よって, 変更により overlap の個数は変わらない．$c_r \in C_R$ から $f_l \in F'_L$ への割当についても同様である．

$c_l \in C_L$ から $f_m \in F'_M$ への割当において, c_l は A 中の c_i と 1 回 overlap し, A' 中の c_h とも 1 回 overlap する．よって, 変更により overlap の個数は変わらない $c_r \in C_R$ から $f_m \in F_M$ への割当, $c_m \in C_M$ から $f_l \in F_L$ への割当, $c_m \in C_M$ から $f_r \in F_R$ への割当についても同様である．

$c_m \in C_M$ から $f_m \in F'_M$ への割当はにおいて, c_m は A 中の c_h, c_i と 1 回ずつ overlap し, A' 中の c_h, c_i とは overlap しない．

また, 変更により A' 中の c_h と c_i の overlap が解消する．

以上より, 変更により overlap の個数は 1 つ以上少なくなる．

よって, 解のコストが同じで overlap の個数がより少ない解があることが示せる．これは仮定に矛盾する．したがって, overlap がない解がある． □

コストが λ 以下の r-gathering があるかを判定する問題を (λ, r)-gathering 問題という．本章で設計する r-gathering 問題を解くアルゴリズムは, この (λ, r)-gathering 問題を解くアルゴリズムをサブルーチンとして利用する．(λ, r)-gathering 問題を解くアルゴリズムについては 3.2 節で説明する．

水平直線上の, 点集合 (利用者の集合) $C = \{c_1, c_2, \ldots, c_n\}$ と，点集合 (施設候補地の集合) $F = \{f_1, f_2, \ldots, f_m\}$, 施設候補地の開設コスト $op : F \to \mathbb{R}$, 整数 r と値 λ が与えらえれたとする．利用者 $c \in C$ が開設施設 $f \in F' \subset F$ に割り当てられたとき, 接続コスト $co(c, f)$ は 2 点間の距離とする．次の条件 (i)(ii)(iii) を満たす C から $F' \subset F$ への割当 A を (λ, r)-gathering という．

(i) 各 $f \in F'$ について $|\{c \mid A(c) = f\}| \geq r$

(ii) 各 $c \in C$ について $co(c, A(c)) \leq \lambda$

(iii) 各 $f \in F'$ について $op(f) \leq \lambda$

ここで (i) は割当 A が r-gathering であるための条件であり, (ii)(iii) は r-gathering A のコストが λ 以下であるための条件である．

コストが λ 以下の r-gathering があるかどうかを判定する問題を (λ, r)-gathering 問題という．この問題を解く $O(m + n)$ 時間アルゴリズムを設計した [5].

r-gathering のコストは, ある $c \in C$ と $f \in F$ の接続コスト $co(c, f)$, もしくは，ある $f \in F$ の開設コスト $op(f)$，のいずれかに等しい．接続コスト $co(c, f)$ の候補は高々 mn 個であり, 開設コスト $op(f)$ の候補は高々 m 個である．すなわち r-gathering 問題の解のコストは, これら高々 $mn + m$ 個のコストのうちのいずれかである．これらをソートし, 3.2 節で与える $O(m + n)$ 時間の判定アルゴリズムを用いて，(λ, r)-gathering 問題が解を持つような最小の λ を二分探索で見つけることにより, r-gathering 問題の解の割当とそのコストが計算できる．

ソートには $O(mn\log(mn))$ 時間かかり, 二分探索には, $O(m+n)$ 時間の判定アルゴリズムを $\log(mn+m))$ 回実行するため，$O((m+n)\log(mn+m))$ 時間かかる．したがって, このアルゴリズムの計算時間は $O(mn\log(m+n))$ である．

しかし, 文献 [12] の手法 (行列探索法) を用いることで，より速い $O((n+m)\log(n+m))$ 時間アルゴリズムを設計することができる [5]．これが本章の主な結果である．同様の手法は [11, 18] の点集合を step 関数で近似する問題を解くアルゴリズムにも使われている．

3.2 直線上の (λ, r)-gathering 問題を解くアルゴリズム

この節では，(λ, r)-gathering 問題を $(m+n)$ 時間で解くアルゴリズム [2, 5] について説明する．このアルゴリズムは動的計画法に基づく．

3.2.1 定義

$c_i \in C, f_j \in F$ をそれぞれ水平直線上の座標値とみなす．ある $c_i \in C$ について, 区間 $[c_i-\lambda, c_i+\lambda]$ に施設候補地がないとき, (λ, r)-gathering 問題に解はない．よって, そのような c_i はないものとする．ある $f_j \in F$ について, 区間 $[f_j-\lambda, f_j+\lambda]$ に利用者が高々 $r-1$ 人しかいないとき, (λ, r)-gathering 問題のいずれの解においても f_j は F' に含まれない．よって, そのような f_j はないものとする．同様に, 開設コスト $op(f_j)$ が λ より大きい施設 f_j は, いずれかの解においても F' に含まれない．よって, そのような f_j はないものとする．それらについては $O(m+n)$ 時間で取り除くことができる．擬似コード **remove facility** を以下に示す．

$C_i = \{c_1, c_2, \ldots, c_i\}$ とし, $F_j = \{f_1, f_2, \ldots, f_j\}$ とする．

3.2.2 アルゴリズム

整数 $j \in [1, m]$，$i \in [1, n]$ が与えられたとき，次の条件 (i)(ii)(iii)(iv) を満たす C_i から $F'_j \subset F_j$ への割当 A を求める問題を $\mathrm{SP}(j, i)$ とする．

(i) 各 $f \in F'_j$ について $|\{c \mid A(c) = f\}| \geq r$
(ii) 各 $c \in C_i$ について $co(c, A(c)) \leq \lambda$
(iii) 各 $f \in F'_j$ について $op(f) \leq \lambda$
(iv) $f_j \in F'_j$

ここで，(i) は各開設施設には利用者が r 人以上割り当てられる，(ii) は接続コストが λ 以下である，(iii) は開設コストが λ 以下である，(iv) は最も右の開設施設が f_j である，ことをそれぞれ意味している．

Algorithm 1 remove facility(C, F, r)

$left(j) = \infty$ $(j = 1, 2, \ldots, m)$ に初期化
$i = 1$, $j = 1$
while $j \leq m$ **and** $i \leq n$ **do**
  if $f_j - \lambda \leq c_i$ **then**
    $left(j) = i$ /* 区間 $[f_j - \lambda, f_j + \lambda]$ 中の最も左の利用者が c_i */
    $j = j + 1$
  else
    $i = i + 1$
  end if
end while
$right(j) = -\infty$ $(j = 1, 2, \ldots, m)$ に初期化
$i = n$, $j = m$
while $j \geq 1$ **and** $i \geq 1$ **do**
  if $c_i \leq f_j + \lambda$ **then**
    $right(j) = i$ /* 区間 $[f_j - \lambda, f_j + \lambda]$ 中の最も右の利用者が c_i */
    $j = j - 1$
  else
    $i = i - 1$
  end if
end while
/* 区間 $[f_j - \lambda, f_j + \lambda]$ に利用者が少なくとも r 人いる $f_j \in F$ を加える */
$\dot{F} = \emptyset$
for $j = 1$ to m **do**
  if $right(j) - left(j) > r$ **then**
    $\dot{F} = \dot{F} \cup f_j$ /* 区間 $[f_j - \lambda, f_j + \lambda]$ 中に利用者が $right(i) - left(i) + 1$ 人いる */
  end if
end for
return $\dot{F}$

補題 3.1 より，SP(j,i) に解があるならば，SP(j,i) に overlap がない解があることがわかる．また，SP(j,i) に解があり，$co(c_{i+1}, f_j) \leq \lambda$ ならば，SP$(j,i+1)$ にも解があることがわかる．

SP(j,i) に解があるとき，SP(j,i') が解を持つ中で最小の i' を $s(f_j)$ とする．条件 (iv) $f_j \in F'_j$ は $c_{s(j)}$ が区間 $[f_j - \lambda, f_j + \lambda]$ 中にあることを意味する．$s(f_j)$ (とそれを実現する割当) を求める問題を部分問題 SP(j) とする．もしある j について，すべての i に対して SP(j,i) に解がないならば，SP(j) に解はない．これ以外の場合は SP(j) に解がある．

補題 3.2. $j' < j$ なる $f_{j'}, f_j \in F$ について，$s(f_{j'}) \leq s(f_j)$ である．

証明. 背理法で証明する．$s(f_{j'}) > s(f_j)$ が成り立つと仮定する．$C_{s(f_j)}$ から F'_j への $s(f_j)$ に関する割当を次のように修正する．f_j に割り当てている利用者を $f_{j'}$ に割り当て，f_j を閉設する．これは，$C_{s(f_j)}$ からへ $F'_{j'}$ の r-gathering であり，$s(f_{j'}) = s(f_j)$ である．よって，仮定に矛盾する． □

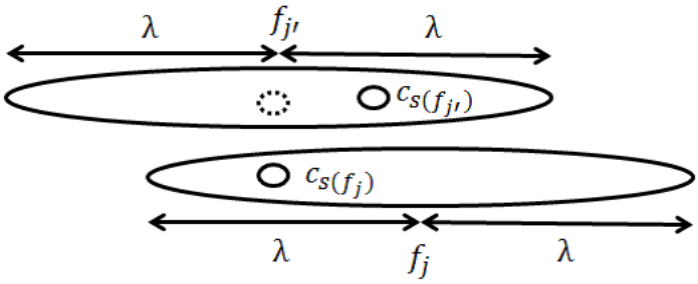

図 3.4: $s(f_{j'}) > s(f_j)$ の場合．

SP(j) に解があり，$c_1 < f_j - \lambda$ であるとき，その解に関連する割当には f_j 以外の開設施設が 1 つ以上ある．SP(j) の解について，右から 2 番目の開設施設を $f_{j'}$ とする．$f_{j'}$ を f_j のメイトといい，$mate(f_j) = f_{j'}$ と表記する．f_j のメイト $f_{j'}$ について，3 つの場合がある．(図 3.4 参照.)

Type 1: $f_{j'} + \lambda < f_j - \lambda$ かつ 区間 $[f_{j'} + \lambda, f_j - \lambda]$ 中に利用者がおらず, 区間 $[f_j - k, f_j + k]$ 中に利用者が r 人以上いる

Type 2: $f_{j'} + \lambda \geq f_j - \lambda$ かつ $c_{s(f_{j'})} \geq f_j - k$ であり, 区間 $(s(f_{j'}), f_j - \lambda]$ 中に利用者が r 人以上いる

Type 3: $f_{j'} + \lambda \geq f_j - \lambda$ かつ $c_{s(f_{j'})} < f_j - k$ であり, 区間 $(c_{s(f_{j'})}, f_j - \lambda]$ 中に利用者が r 人以上いる

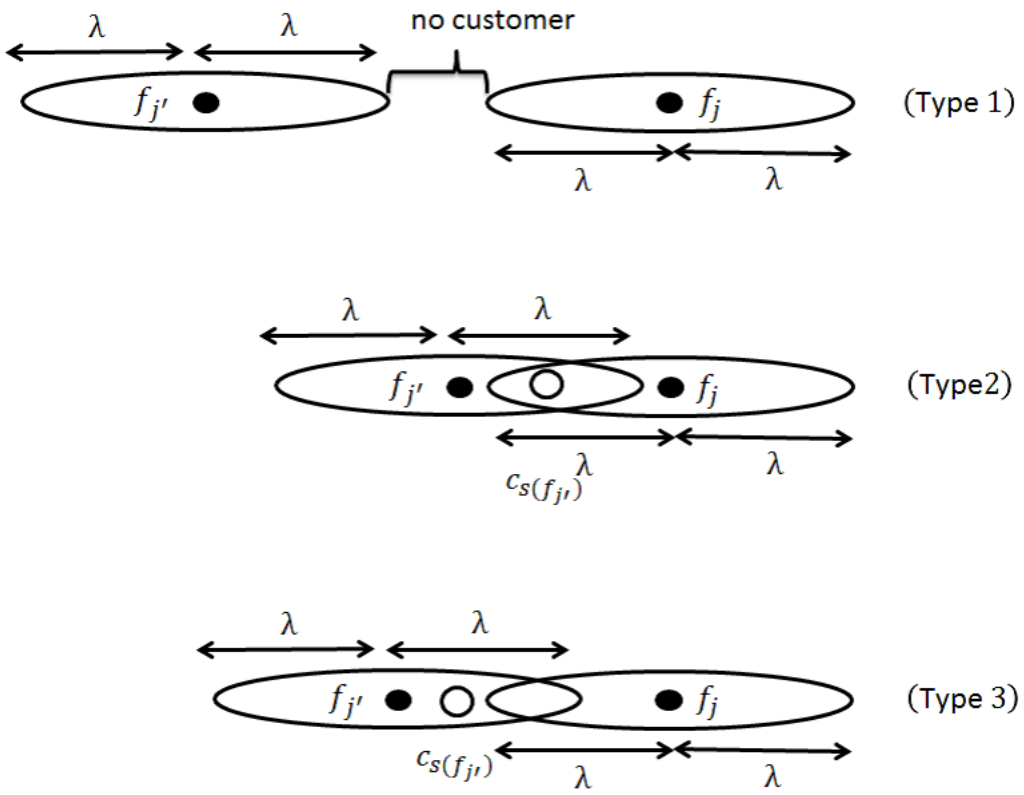

図 3.5: $mate(f_j)$ による 3 つの Type.

各 f_j について，上記の 3 つの場合でメイト $f_{j'}$ の候補をすべてチェックすることで，動的計画法に基づいた $O(m^2 + n)$ 時間アルゴリズムを設計することができる．しかし，それらのチェックの大半を省略できることを次の補題で示す.

補題 3.3. (a) SP(j) に解があり，SP$(j+1)$ にも解があるならば，$mate(f_j) \leq mate(f_{j+1})$ が成り立つ.

(b) 各 $f_j \in F$ について，次の条件 (i)(ii)(iii) を満たす $f_{j'}$ があるならば，それらの中で最小の $f_{j'}$ を f_{min} とする.

(i) SP(j') に解がある,

(ii) $f_{f'} + \lambda \geq f_j - \lambda$,

(iii) $j' < j$.

右から 2 番目の開設施設を f_{min} とした SP(j) に解がないならば，(b1) $f_{min} < f_{j''} < f_j$

を満たす任意の $f_{j''}$ は f_j のメイトではなく，SP(j) に解がない．(b2) $mate(f_{j+1})$ があるならば，$f_{min} \leq mate(f_{j+1})$ が成り立つ．

証明．(a) 背理法で証明する．$mate(f_{j+1}) + \lambda < f_j - \lambda$ が成り立つと仮定すると，$mate(f_{j+1})$ も f_j のメイトであるため矛盾．$mate(f_{j+1}) + \lambda \leq f_j - \lambda$ が成り立つと仮定すると，補題 3.2 より $mate(f_{j+1})$ も f_j のメイトであるため矛盾．

(b1) 補題 3.2 より，明らかである．

(b2) 背理法で証明する．$mate(f_{j+1}) + \lambda < f_j - \lambda$ が成り立つと仮定すると，$mate(f_{j+1})$ も f_j のメイトであるため矛盾．$mate(f_{j+1}) + \lambda \geq f_j - \lambda$ が成り立つと仮定すると，f_{min} は $mate(f_j)$ ではなく $mate(f_{j+1})$ であるので矛盾． □

補題 3.3 は，ある $f_{j'}$ まで f_j のメイトを調べた後，次に f_{j+1} のメイトを調べるとき，$f_{j'}$ から調べ始められることを意味する．

上記の補題に基づきアルゴリズム **find** (λ, r)**-gathering** を設計する．

ある区間に利用者がいるかどうかや $s(f_j)$ の値を高速に計算するため，前処理を行う．前処理には $O(m+n)$ 時間かかる．常に $j' \leq j$ が成り立つので．$s(f_j)$ を計算するための最も内部の処理は高々 $2m$ 回実行される．このアルゴリズムは $O(m)$ 時間で実行できる．以上より，次の定理が示せる．

定理 3.4． 直線上の (λ, r)-gathering 問題を解く $O(m+n)$ 時間アルゴリズムがある．

3.2.3 疑似コード

(λ, r)-gathering 問題を解くアルゴリズム **find** (λ, r)**-gathering** を示す．

Algorithm 2 find (λ, r)-gathering

$j = 1$
/* 開設施設が 1 つの場合 */
while 区間 $[f_j - \lambda, f_j + \lambda]$ 中に c_1 と c_r がある **do**
  r 番目の利用者が $c_{s(f_j)}$ となるよう $s(f_j) = r$ とする
  $j = j + 1$
end while
/* 開設施設が 2 つ以上の場合 */
$j' = 1$
while $j \leq m$ **do**
  while $j' < j$ **and** (区間 $(f_{j'} + \lambda, f_j - \lambda)$ 中に利用者がいる **or** SP(j') に解がない) **do**
    $j' = j' + 1$
  end while
  if $j' < j$ **then**
    /* 区間 $(f_{j'} + \lambda, f_j - \lambda)$ に利用者がいない **and** SP(j') が解を持つ */
    if $f_{j'} + \lambda < f_j - \lambda$ **and** 区間 $[f_{j'} + \lambda, f_j - \lambda]$ 中に利用者がいない **and** 区間 $[f_j - \lambda, f_j + \lambda]$ 中に利用者が r 人以上いる **then**
      区間 $[f_{j'} + \lambda, f_j - \lambda]$ 中の r 番目の利用者が $c_{s(f_j)}$ となる $s(f_j)$ を求める (Type 1)
    else if $f_{j'} + \lambda \geq f_j - \lambda$ **and** $c_{s(f_{j'})} \geq f_j - \lambda$ **and** 区間 $(s(f_{j'}), f_j - \lambda]$ 中に利用者が r 人以上いる **then**
      区間 $(c_{s(f_{j'})}, f_j - \lambda]$ 中の r 番目の利用者が $c_{s(j)}$ となる $s(f_j)$ を求める (Type 2)
    else if $f_{j'} + \lambda \geq f_j - \lambda$ **and** $c_{s(f_{j'})} < f_j - \lambda$ **and** 区間 $(c_{s(f_{j'})}, f_j - \lambda]$ 中に利用者が r 人以上いる **then**
      区間 $[f_{j'} + \lambda, f_j - \lambda]$ 中の r 番目の利用者が $c_{s(f_j)}$ となる $s(f_j)$ を求める (Type 3)
    end if
    /* SP(j) は解なし */
  end if
  $j = j + 1$
end while
if 点 c_n から距離 λ 以内のある f_j について, $s(f_j)$ が定義されている **then**
  return Yes
else
  return No
end if

3.3 直線上の r-gathering 問題を解くアルゴリズム

この章では, C と F が直線上の点集合とみなせるとき, r-gathering 問題を解く $O((n+m)\log(n+m))$ 時間アルゴリズムを与える. 本アルゴリズムは, 文献 [12, 11] の手法 (行列探索法) を用いる.

$C = \{c_1, c_2, \ldots, c_n\}, F = \{f_1, f_2, \ldots, f_m\}$ とする. これらの要素を水平直線上の点や施設とみなす. $c_1 \leq c_2 \leq \ldots \leq c_n$ かつ $f_1 \leq f_2 \leq \ldots \leq f_m$ と仮定する.

i 行 j 列の行列 M'_C の各要素を $m_{i,j} = c_i - f_j$ とする. M'_C の任意の要素について $m_{i,j} \geq m_{i,j+1}$ と $m_{i,j} \leq m_{i+1,j}$ が成り立つ. これを M'_C の各行や各列では要素がソートされているという. 同様に, i 行 j 列の行列 M'_F の各要素を $m'_{i,j} = f_j - c_i$ とする. M'_F の各行や各列では要素がソートされている.

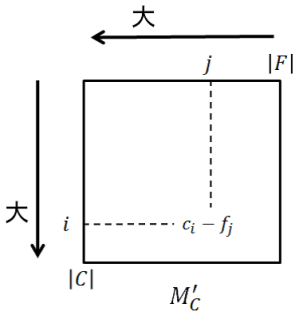

図 3.6: M'_C の要素.

r-gathering 問題の (最適) 解の (最小) コスト λ^* は, (i) M'_C 中の要素, (ii) M'_F 中の要素, もしくは (iii) いずれかの $f \in F$ の開設コスト, のいずれかである. まず, M'_C 中の要素 k で, (λ, r)-gathering 問題が解を持つ最小の λ を求める方法を示す.

$\max\{n, m\}$ 以上の最小の 2 のべき乗である整数を n とする. M'_C の最大の要素は第 n 行第 1 列の要素 $m_{n,1}$ である. M'_C が n 行 n 列の行列となるように, $m_{n,1}$ からなる行や列を M'_C の最も下の行や最も左の列として必要な数だけ追加する. 得られた n 次正方行列を M_C とする. M_C においても各行や各列はソートされている.

アルゴリズムはいくつかのステージ $s = 1, 2, \ldots, \log n$ からなる. 各ステージ s では M_C の部分行列の集合 L_s を保持する. アルゴリズムは, 常に M_C 中の要素 λ で (λ, r)-gathering 問題に解がある最小の λ が L 中に残ることを保証しつつ, それ以外の要素しか含まない部分行列を効率的に削除していく.

初めに, $L_0 = \{M_C\}$ とする.

まず, L_{s-1} から L_s を次のように作る. L_{s-1} 中の各部分行列 M は $n/2^{s-1}$ 次正方行列である. 各 M を 4 個の $n/2^s$ 次正方行列に分割し, L_s に追加する.

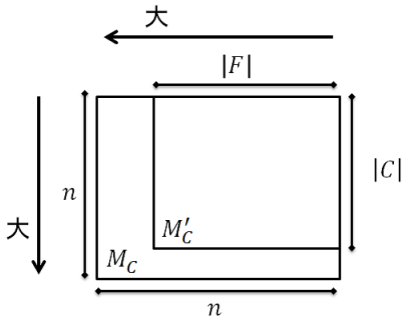

図 3.7: M'_C と M_C のサイズ.

次に, L_s 中の各部分行列の右上隅の要素の集合を考え, これらの中央値を λ_{min} とし, $\lambda = \lambda_{min}$ として (λ, r)-gathering 問題を解く. このとき, 2 つの場合がある.

場合 1: (λ, r)-gathering 問題に解があるとき.
$\lambda_{min} \geq \lambda^*$ である. L_s から右上隅の（最も小さい）要素が λ_{min} より大きい部分行列を取り除く. $\lambda_{min} < \lambda^*$ より, 取り除かれる行列に λ^* が含まれることはない. L_s から $|L_s|/2$ 個の部分行列を取り除くことができる.

場合 2: (λ, r)-gathering 問題に解がないとき.
$\lambda_{min} < \lambda^*$ である. L_s から左下隅の（最も大きい）要素が λ_{min} より小さい部分行列を取り除く. $\lambda_{min} < \lambda^*$ より, 取り除かれる行列に λ^* が含まれることはない. 取り除かれる部分行列の個数を見積もる. 左下から右上への対角線が M_C 中で同一直線上にあるような部分行列の集合を “chain” とする. chain 中の行列の対角要素はソートされている. よって, 各 chain について, chain 中の行列の右上隅の要素が λ_{min} より小さい部分行列は高々1 つしか残らない. したがって, D_s を chain の本数に 1 を加えた数, つまり, $D_s = 2^{s+1}$ とすると, $|L_s|/2 > D_s$ ならば少なくとも $|L_s|/2 - D_s$ 個の部分行列が L_s から取り除かれる.
同様に, L_s 中の部分行列の左下隅の集合を考え, これらの中央値を λ_{max} とし, $\lambda = \lambda_{max}$ として (λ, r)-gathering 問題を解く. 同様に L_s からいくつかの部分行列を取り除く. ここまでがステージ s である.

ステージ $s = \log n$ の終了時に, $L_{\log n}$ 中の各部分行列はちょうど 1 つの要素からなる. よって通常の二分探索により $O(|L_{\log n}| \log |L_{\log n}|)$ 時間で λ^* を計算することができる.

補題 3.5. ステージ s の終了時に L_s の部分行列の個数は高々$2D_s$ 個である. （証明略）

最後に, r-gathering 問題を解くアルゴリズムの計算時間について考えよう.

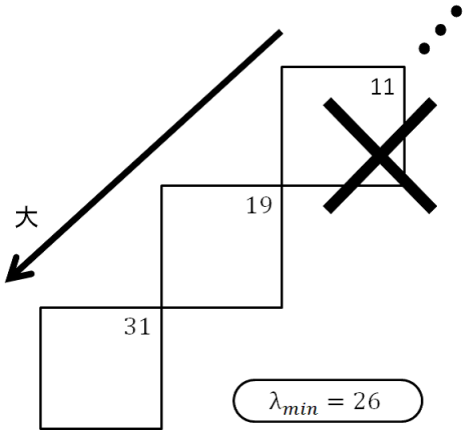

図 3.8: chain の例.

(λ, r)-gathering 問題を解く線形時間決定アルゴリズムの呼び出しを除いて, 各ステージ $s = 1, 2, \ldots, \log n$ には $O(|L_{s-1}|) = O(D_s)$ 時間かかる. $D_0 + D_1 + \cdots + D_{\log n} = 2 + 4 + \ldots + 2^{\log n} = 2 \cdot 2^{\log n} \leq 2n$ が成り立つので, 合計 $O(n)$ 時間かかる. （ここでは中央値を求めるために線形時間アルゴリズムを用いている.）

各ステージで線形時間決定アルゴリズムを 2 回呼び出すので, この部分には合計 $O(n \log n)$ 時間かかる.

ステージ $s = \log n$ の後, 各行列はちょうど 1 つの要素からなり, 高々 $L_{\log n} \leq 2D_{\log n} = 4n$ 個の要素が残る. よって, 3 章の線形時間判定アルゴリズムを用いて, 高々 $\log 4n$ 回呼び出し, 二分探索をすることにより, (λ, r)-gathering 問題に解がある最小の λ を計算できる. これには $O(n \log n)$ 時間かかる.

すなわち, 合計 $(n \log n)$ 時間で M'_C 中の要素 λ で (λ, r)-gathering 問題に解がある最小の λ を計算できる.

同様に, M'_F 中の要素 λ で (λ, r)-gathering 問題に解がある最小の λ を計算できる.

また, いずれかの $f \in F$ の開設コスト λ のうち, (λ, r)-gathering 問題に解がある最小の λ も, 線形時間の判定アルゴリズムを用いた通常の二分探索により $O((m + n) \log m)$ 時間で計算できる.

これら 3 つのうち最小の値が解のコストである.

定理 3.6. 直線上の r-gathering 問題を解く $O((m + n) \log(m + n))$ 時間アルゴリズムがある.

3.4 直線上の r-gathering with h-outlier 問題

本節では, r-gathering 問題を一般化した問題を考える. ここでは, 高々h 人の利用者を割り当てなくてもよいとする.

例えば, 1 人の利用者が他のすべての利用者と離れていたとき, その 1 人の利用者により r-gathering のコストが大きくる場合がある. そこで, 大量の利用者に対して極小数の利用者を無視（特別扱い）することで、特殊な利用者に左右されないような, コストの小さい r-gathering を見つけたい.

水平線上の, 点集合 (利用者の集合)$C = c_1, c_2, \ldots, c_n$ と, 点集合 (施設候補地の集合)$F = f_1, f_2, \ldots, f_m$, 施設候補地の開設コスト $op : F \to \mathbb{R}$, 整数 r に加え, 整数 h が与えられたとする. 利用者 $c \in C$ が開設施設 $f \in F' \subset F$ に割り当てられたとき, 接続コスト $co(c, f)$ は 2 点間の距離とする.

ここで, 割り当てなくてもよい高々h 人の利用者の部分集合を $C' \subset C$ とする. 各 $f \in F'$ について $|\{c \mid A(c) = f\}| \geq r$ であるような, $C \backslash C'$ から $F' \subset F$ への割当 A を r-gathering with h-outlier という.

r-gathering with h-outlier $A : C \setminus C' \to F'$ のコストを $\max\{\max_{c \in C \setminus C'}\{co(c, A(c)\}, \max_{f \in F'}\{op(f)\}\}$ とする.

コストが最小の r-gathering with h-outlier を見つける問題を r-gatherig with h-outlier 問題という. また, 実数 λ が与えられたとき, コストが λ 以下の r-gathering with h-outlier が存在するか判定する問題を (λ, r)-gathering with h-outlier 問題という.

r-gathering 問題と同様に, r-gatherig with h-outlier 問題の解のコストは, ある $c \in C$ と $f \in F$ の接続コスト $co(c, f)$, もしくは, ある $f \in F$ の開設コスト $op(f)$ のいずれかに等しく, それらは高々$mn + m$ 通りである. よって, r-gathering 問題と同様に, 判定問題を解く $O(h^2m + n)$ 時間アルゴリズムを設計し, 行列探索法 [12] を用いることで, 最適化問題を解く $O((h^2m + n)\log(n + m))$ 時間アルゴリズムを設計する.

r-gathering with h-outlier $A : C \setminus C' \to F'$ において, $i'' < i' < i$ なる $c_{i''}, c_{i'}, c_i \in C$ で $A(c_{i''}) = A(c_i)$ かつ $c_{i'} \in C'$ なるものがあるとき, A 中で $c_{i'}$ を **crack** という. (図 3.9, 3.10 参照.) 次の補題が成り立つ.

補題 3.7. r-gahtering with h-outlier 問題に解が存在するとき, overlap も crack のない解がある.

証明. 補題 3.1 より, overlap がない解がある. よって, crack がない解があることを示す.

ある r-gathering with h-outlier 問題において, crack がある解しかないと仮定する. このとき, crack の個数が最小の r-gatheirng with h-outlier を $A : C \setminus C' \to F'$ とする. また, A のコストを OPT とする.

A 中の crack のうち, 最も左側の crack を $c_i \in C'$ とする. このとき, $c_{i-1} \in C \setminus C'$ であり, $A(c_{i-1}) = f_j$ とする. また, f_j に割り当てられている利用者のうち, 最も左側の利用者を $c_{i'} \in C \setminus C'$ とする.

ここで, A 中の $c_{i'}$, c_i のみを $c_{i'}$ を新たに crack とし, $A'(c_i) = f_j$ のように変更した割当を A' とする. この変更により, A' の crack の個数は A の crack の個数より 1 少ない.

A' 中で f_j に割り当てられている利用者の人数は A と等しく, A' 中の $|C'|$ の人数も A と等しい. よって, A' も r-gathering with h-outlier である. また, $c_{i'}$ は A 中で f_j に割り当てられていた最も左側の利用者なので, $co(c_i, f_j) \leq co(c_{i'})$ である. したがって, A' のコストは A のコスト以下である.

以上より, A' は A より crack の個数が少ない解である. これは仮定に矛盾する. □

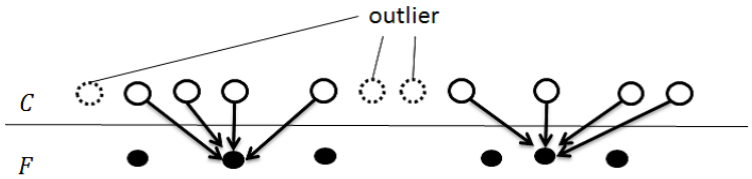

図 3.9: crack がない割当の例.

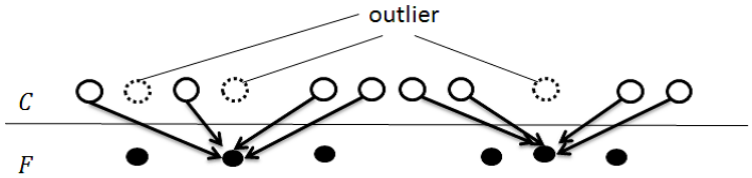

図 3.10: crack がある割当の例.

(λ, r)-gathering with h-outlier 問題の部分問題を定義する. C の部分集合 $\{c_1, c_2, \ldots, c_i\}$ を C_i とし, F の部分集合 $\{f_1, f_2, \ldots, f_j\}$ を F_j とする.

ここで, 3.2.1 節のように, 前処理により, 開設コストが λ より大きいような施設候補地は取り除かれているとする. つまり, 開設コストが λ 以下の施設候補地の集合が F であり, その個数が m である.

整数 $j \in [1, m]$, $i \in [1, n]$, $k \in [0, h]$ が与えられたとき, 次の条件 (i)(ii)(iii)(iv)(v) を満たす $C_i \backslash C'$ から $F'_j \subset F_j$ への割当 A を求める問題を $SP(j, i, k)$ とする.

(i) 各 $f \in F'_j$ について $|\{c \mid A(c) = f\}| \geq r$
(ii) $|C'| = k$
(iii) 各 $c \in C_i$ について $co(c, A(c)) \leq \lambda$
(iv) 各 $f \in F_j$ について $op(f) \leq \lambda$
(v) $f_j \in F'_j$

指定した j, k に対して, $SP(j, i, k)$ がいずれかの i において解を持つとき, $SP(j, i, k)$ が解を持つ中で最小の i を $s(f_j, k)$ とする.

$s(f_j, k)$ とそれに対応する r-gathering を見つける問題を部分問題 $SP(j, k)$ とする. すべての i に対して $SP(j, i, k)$ が解を持たないならば, $SP(j, k)$ は解なしという.

r-gathering 問題に対する補題 3.2 や補題 3.3 と同様に, 次の補題が成り立つ.

補題 3.8. $j' < j$ であり $SP(f_{j'}, k)$ と $SP(f_j, k)$ に解がある $f_{j'}, f_j \in F$ について, $s(f_{j'}, k) \leq s(f_j, k)$ が成り立つ.

証明. 背理法で証明する. $C_{s(f_j,k)} \setminus C'_{s(f_j,k)}$ から F'_j への割当を次のように修正する. f_j に割り当てられている利用者を $f_{j'}$ に割り当て, f_j を閉鎖する. これは $C_{s(f_j,k)} \setminus C_{s(f_j,k)}$ から $F'_{j'}$ への r-gathering with k-outlier であり, $s(f_{j'}, k) = s(f_j, k)$ である. よって, 仮定に矛盾する. □

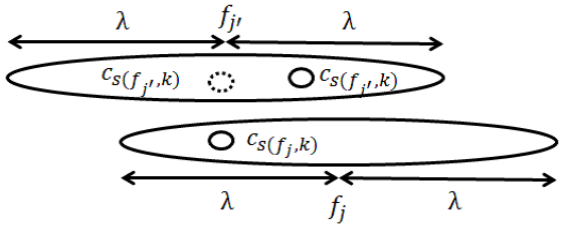

図 3.11: $s(f_{j'}, k) > s(f_j, k)$ の場合.

開設施設 $f_j \in F'$ について, $c_{s(f_j)}$ までに outlier が k 個であることを f_j with k-outlier とし, $f_{j,k}$ と書く.

$SP(j, k)$ に解があり, $c_{1+k} < f_{j,k} - \lambda$ であるとき, その解に関連する割り当てには f_j 以外の開設施設が 1 つ以上ある. $SP(j, k)$ の解について, 右から 2 番目の開設施設を $f_{j',k'}$ とし, $c_{s(f_{j'},k')}$ までの outler の個数が k' であるとする. つまり, 区間 $(c_{s(f_{j'},k')}, c_{s(f_j,k)})$ 中の outlier の人数は $k - k'$ 人である. $f_{j'}$ with k'-outlier を f_j with k-outlier のメイトといい, $mate(f_{j,k}) = f_{j',k'}$ と書く.

$f_{j,k}$ のメイト $f_{j',k'}$ について, 3 つの場合がある.

Type 1: $f_{j',k'} + \lambda < f_{j,k} - \lambda$ かつ 区間 $[f_{j',k'} + \lambda, f_{j,k} - \lambda]$ 中に利用者が $k - k'$ 人以下であり, 区間 $[f_{j,k} - \lambda, f_{j,k} + \lambda]$ 中に利用者が $r + \max\{k - k' - \alpha, 0\}$ 人以上いる

ここで, α は区間 $(c_{s(f_{j'},k')}, f_{j,k} - \lambda)$ 中の利用者の人数である

Type 2: $f_{j',k'} + \lambda \geq f_{j,k} - \lambda$ かつ $c_{s(f_{j'},k')} \geq f_{j,k} - \lambda$ であり, 区間 $(c_{s(f_{j'},k')}, f_{j,k} - \lambda]$ 中に利用者が $r + k - k'$ 人以上いる

Type 3: $f_{j',k'} + \lambda \geq f_{j,k} - \lambda$ かつ $c_{s(f_{j',k'})} < f_{j,k} - \lambda$ であり, 区間 $(c_{s(f_{j',k'})}, f_{j,k} - \lambda]$ 中に利用者が $r + k - k'$ 人以上いる

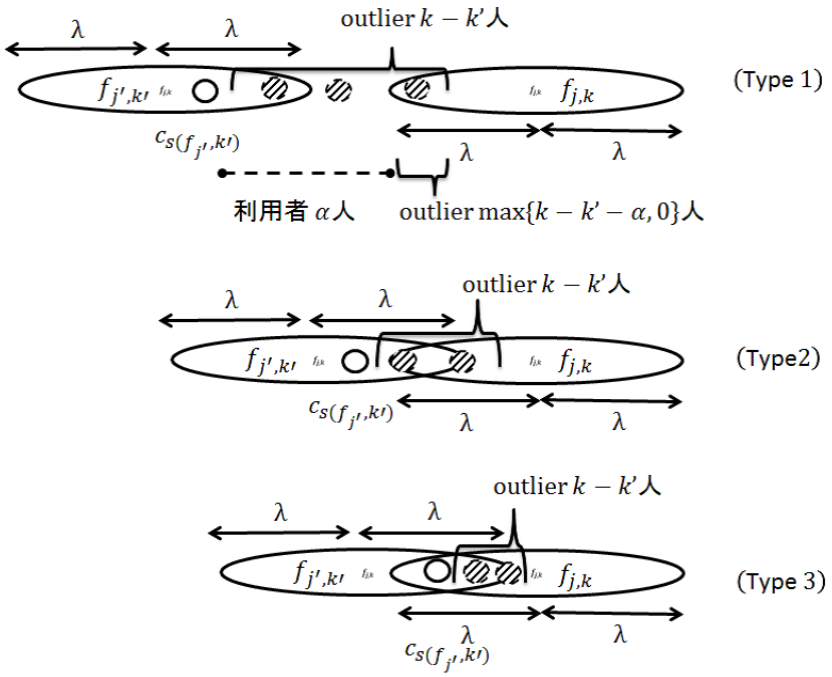

図 3.12: $mate(f_j, k)$ による 3 つの Type.

補題 3.3 と同様に, 次の補題が成り立つ.

補題 3.9. (a) $SP(j, k)$ に解があり, $SP(j + 1, k)$ にも解があるならば, $mate(f_{j,k}) \leq mate(f_{j+1,k})$ が成り立つ.

(b) 各 $f_{j,k} \in F$ について, それぞれの $k' < k$ に対して, 次の条件 (i)(ii)(iii) を満たす $f_{j',k'}$ があるならば, それらの中で最小の $f_{j',k'}$ を $f_{min,k'}$ とする.

(i) $SP(j', k')$ に解がある.

(ii) $f_{j',k'} + \lambda \geq f_{j,k} - \lambda$

(iii) $j' < j$

右から 2 番目の開設施設を $f_{min,k'}$ とした $SP(j, k)$ に解がないならば, (b1) $f_{min,k'} < f_{j'',k'} < f_{j,k}$ を満たす任意の $f_{j'',k'}$ は $f_{j,k}$ のメイトではなく, $SP(j, k)$ に解がない. (b2) $mate(f_{j+1}, k)$ があるならば, $f_{min,k'} \leq mate(f_{j+1}, k)$ が成り立つ.

証明 (a) 背理法で証明する. $mate(f_{j+1}, k) + \lambda < f_{j,k} - \lambda$ が成り立つと仮定すると, $f_{meta(j+1,k)}$ も $f_{j,k}$ のメイトであるため矛盾. $f_{mate(j+1,k)} + \lambda \leq f_{j,k} - \lambda$ が成り立つと仮定

すると, 補題 3.7 より $mate(f_{j+1}, k)$ も $f_{j,k}$ のメイトであるため矛盾.

(b1) 補題 3.8 より, 明らかである.

(b2) 背理法で証明する. $mate(f_{j+1}, k) + \lambda < f_{j,k} - \lambda$ が成り立つと仮定すると, $mate(f_{j+1}, k)$ も $f_{j,k}$ のメイトであるため矛盾. $mate(f_{j+1}, k) + \lambda \geq f_{j,k} - \lambda$ が成り立つと仮定すると, $f_{min,k'}$ は $mate(f_j, k)$ ではなく $mate(f_{j+1}, k)$ であるので矛盾. □

各 $k' < k$, $j' < j$ の $f_{j',k'}$ を網羅的に調べることで $f_{j,k}$ のメイトを計算できる. しかし, メイトの場合分けや補題により調べる部分問題を省略できる.

メイトの場合分けでは, $k'(< k)$ を固定したとき, 調べるべき $f_{j',k'}$ が 3 種類であることを示した.

また, 補題 3.9 は, $k' < k$ を固定したとき, $f_{1,k'}$ から $f_{j',k'}$ まで $f_{j,k}$ のメイトを調べた後, 次に $f_{j+1,k}$ のメイトを調べるとき, 固定した $k' < k$ に対して, $f_{j',k'}$ から調べ始められることを意味する.

上記の補題に基づきアルゴリズム **find** (λ, r)**-gathering with** h**-outlier** を設計する.

ある区間に利用者がいるかどうかや $s(f_j, k)$ の値を高速に計算するため, 前処理を行う. 前処理には $O(m+n)$ 時間かかる. 常に $j' \leq j$ が成り立つので, $k' < k$ なる $k', k \in [0, h]$ に対して, $s(f_j, k)$ を計算するための最も内部の処理はそれぞれ高々 $2m$ 回実行される. よって, $s(f_j, k)$ を計算するための最も内部の処理は高々 $2h^2m$ 回実行される. このアルゴリズムは $O(h^2m + n)$ 時間で実行できる. 以上より, 次の定理が示せる.

定理 3.10 直線上の (λ, r)-gathering with h-outlier 問題を解く $O(h^2m + n)$ 時間アルゴリズムがある.

r-gathering 問題の解のコストはいずれかの接続コストの値である. ここで, 接続コストの値は高々 mn 通りである. よって, 3.3 節のように, 行列探索法 [12, 11] を用いることで, (λ, r)-gathering with h-outlier 問題を $O(\log(m+n))$ 回解くことで, r-gatehring with h-outlier 問題の解を計算できる.

したがって, 次の定理が成り立つ.

定理 3.11 直線上の r-gathering with h-outlier 問題を解く $O((h^2m + n)\log(m+n))$ 時間アルゴリズムがある.

Algorithm 3 find (λ, r)-gathering with h-outlier

for $k = 0$ to h **do**
  for $j = 1$ **do**
    /* 開設施設が 1 つの場合 */
    if 区間 $[f_{j,k} - \lambda, f_{j,k} + \lambda]$ 中に c_{1+k} と c_{r+k} がある **then**
      $r + k$ 番目の利用者が $c_{s(f_j,k)}$ となるよう $s(f_j, k) = r + k$ とする
    else
      /* 開設施設が 2 つ以上の場合 */
      $tmp = \infty$
      $j'[k'] = 1$, $(k' = 0, 1, \ldots, k)$
      for $k' = 0$ to k **do**
        $cost(j, k, k') = \infty$
        while $j'[k'] < j$ **and** (区間 $(f_{j'[k']} + \lambda, f_{j,k} - \lambda)$ 中に利用者が $k - j'[k']$ 人より多くいる **or** $SP(j'[k'], k')$ に解がない) **do**
          $j'[k'] = j'[k'] + 1$
        end while
        if $j'[k'] < j$ **then**
          /* 区間 $(f_{j'[k'],k'} + \lambda, f_{j,k} - \lambda)$ に利用者が $k - j'[k']$ 以下いる **and** $SP(j'[k'], k')$ が解を持つ */
          /* 区間 $(c_{s(f_{j'[k']})}, f_j - \lambda)$ の利用者の人数を α とする */
          if $f_{j'[k'],k'} + \lambda < f_{j,k} - \lambda$ **and** 区間 $(f_{j'[k'],k'} + \lambda, f_{j,k} - \lambda)$ 中に利用者が $k - j'[k']$ 以下いる **and** 区間 $[f_{j,k} - \lambda, f_j + \lambda]$ 中に利用者が $r + \max\{k - j'[k'] - \alpha, 0\}$ 人以上いる **then**
            区間 $[f_{j,k} - \lambda, f_{j,k} + \lambda]$ 中の $r + \max\{k - j'[k'] - \alpha, 0\}$ 番目の利用者が $c_{s(f_j,k)}$ となる $s(f_j, k)$ を求める (Type 1)
          else if $f_{j'[k'],k'} + \lambda \geq f_{j,k} - \lambda$ **and** $c_{s(f_{j'[k'],k'})} \geq f_j - \lambda$ **and** 区間 $(s(f_{j'[k']}), f_j - \lambda]$ 中に利用者が $r + k - j'[k']$ 人以上いる **then**
            区間 $(c_{s(f_{j'[k']})}, f_j - \lambda]$ 中の $r + k - k'$ 番目の利用者が $c_{s(j)}$ となる $s(f_j, k)$ を求める (Type 2)
          else if $f_{j'[k']} + \lambda \geq f_j - \lambda$ **and** $c_{s(f_{j'[k']})} < f_j - \lambda$ **and** 区間 $(c_{s(f_{j'[k']})}, f_{j,k} - \lambda]$ 中に利用者が $r + k - j'[k']$ 人以上いる **then**
            区間 $[f_{j'[k']} + \lambda, f_{j,k} - \lambda]$ 中の $r + k - j'[k']$ 番目の利用者が $c_{s(f_j,k)}$ となる $s(f_j, k)$ を求める (Type 3)
          end if
          /* $SP(j, k)$ は解なし */
        end if
        if $s(f_j, k) < tmp$ **then**
          $tmp = s(f_j, k)$
        end if
      end for
      $s(f_j, k) = tmp$
    end if
  end for
end for
Ans = No
for $\alpha = 0$ to h **do**
  for $\beta = h - \alpha$ downto 0 **do**
    if 点 $c_{n-\alpha}$ から距離 λ 以内のある $f_{j,\beta}$ について, $s(f_j, \beta)$ が定義されている **then**
      Ans = Yes
    end if
  end for
end for
return Ans

3.5 直線上の reserved r-gathering 問題

本説では, r-gathering 問題を一般化した問題を考える. ここでは, いくつかの施設が事前に開設することが決まっているとする. ただし, 開設コストはすべて 0 とする.

開設することが決まっている施設を $F^o \subset F$ とする. 各 $f \in F'$ について $|\{c \mid A(c) = f\}| \geq r$ であるような, C から $F^o \subset F' \subset F$ への割当 A を reserved r-gathering という.

reserved r-gathering $A : C \to F'$ のコストを $\max_{c \in C}\{co(c, A(c)\}$ とする.

コストが最小の reserved r-gathering を見つける問題を reserved r-gatherig 問題という. また, 実数 λ が与えられたとき, コストが λ 以下の reserved r-gathering が存在するか判定する問題を reserved (λ, r)-gathering 問題という. (開設コストが 0 であるため, コストとして考えるのは接続コストのみであることに注意する.)

reserved (λ, r)-gathering 問題の部分問題を定義する. C の部分集合 $\{c_1, c_2, \ldots, c_i\}$ を C_i とし, F の部分集合 $\{f_1, f_2, \ldots, f_j\}$ を F_j とする. また, F^o と F_j の共通部分集合 $F^o \cap F_j$ を F_j^o とする.

整数 $j \in [1, m]$, $i \in [1, n]$ が与えられたとき, 次の条件 (i)(ii)(iii) を満たす C_i から $F_j^o \subset F_j' \subset F_j$ への割当 A を求める問題を $SP^o(j, i)$ とする.

(i) 各 $f \in F_j'$ について $|\{c \mid A(c) = f\}| \geq r$
(ii) 各 $c \in C_i$ について $co(c, A(c)) \leq \lambda$
(iii) $f_j \in F_j'$

$SP^o(j, i)$ がいずれかの i において解を持つとき, $SP^o(j, i)$ が解を持つ中で最小の i を $s^o(f_j)$ とする.

$s^o(f_j)$ とそれに対応する r-gathering を見つける問題を部分問題 $SP^o(j)$ とする. すべての i に対して $SP^o(j, i)$ が解を持たないならば, $SP^o(j)$ は解なしという.

通常の r-gatherign 問題に対する補題 3.2 や補題 3.3 と同様に, 次の補題が成り立つ.

補題 3.12. $j' < j$ であり $SP^o(f_{j'})$ と $SP^o(f_j)$ に解がある $f_{j'}, f_j \in F$ について, $s^o(f_{j'}) \leq s^o(f_j)$ が成り立つ.

証明. 2 つの場合を考える. Case 1: $j' < j'' < j$ なる $f_{j''} \in F^o$ が存在しない.

背理法で証明する. $s^o(f_{j'}) > s^o(f_j)$ が成り立つと仮定する. $C_{s^o(f_j)}$ から F_j' への $s^o(f_j)$ に関する割当を次のように修正する. f_j に割り当てている利用者を $f_{j'}$ に割り当て, f_j を閉設する. これは, $C_{s^o(f_j)}$ からへ $F_{j'}'$ の r-gathering であり, $s^o(f_{j'}) = s^o(f_j)$ である. よって, 仮定に矛盾する.

Case 2: $j' < j'' < j$ なる $f_{j''} \in F^o$ が存在する.

背理法で証明する. $s^o(f_{j'}) > s^o(f_j)$ が成り立つと仮定する. $j' < j'' < j$ なる $f_{j''} \in F^o$ が存在するが, 仮定より, $j' < j'' < j$ であるような, ある $f_{j'} \in F$ が存在する. $C_{s^o(f_j)}$ から

F'_j への $s^o(f_j)$ に関する割当を次のように修正する．f_j に割り当てている利用者を $f_{j'}$ に割り当て，f_j を閉設する．これは，$C_{s^o(f_j)}$ からへ $F'_{j'}$ の r-gathering であり，$s^o(f_{j'}) = s^o(f_j)$ である．よって，仮定に矛盾する． □

$SP^o(j)$ に解があり, $c_1 < f_j - \lambda$ であるとき．その解に関連する割り当てには f_j 以外の開設施設が 1 つ以上ある．$SP^o(j)$ の解について, 右から 2 番目の開設施設を $f_{j'}$ とする．$f_{j'}$ を f_j のメイトといい, $mate^o(f_j) = f_{j'}$ と書く．$SP^o(f_j)$ が解を持つならば, 区間 $(f_{j'}, f_j)$ には F^o の施設は存在しない．f_j のメイト $f_{j'}$ について, 3 つの場合がある．

Type 1: $f_{j'} + \lambda < f_j - \lambda$ かつ 区間 $[f_{j'} + \lambda, f_j - \lambda]$ 中に利用者がおらず, 区間 $[f_j - \lambda, f_j + \lambda]$ 中に利用者が r 人以上いる

Type 2: $f_{j'} + \lambda \geq f_j - \lambda$ かつ $c_{s^o(f_{j'})} \geq f_j - \lambda$ であり, 区間 $(c_{s^o(f_{j'})}, f_j - \lambda]$ 中に利用者が r 人以上いる

Type 3: $f_{j'} + \lambda \geq f_j - \lambda$ かつ $c_{s^o(f_{j'})} < f_j - \lambda$ であり, 区間 $(c_{s^o(f_{j'})}, f_j - \lambda]$ 中に利用者が r 人以上いる

補題 3.3 と同様に, 次の補題が成り立つ.

補題3.13. (a) $SP^o(j)$ に解があり，$SP^o(j+1)$ にも解があるならば $mate^o(f_j) \leq mate^o(f_{j+1})$ が成り立つ．

(b) 各 $f_j \in F$ について，次の条件 (i)(ii)(iii) を満たす $f_{j'}$ があるならばそれらの中で最小の $f_{j'}$ を f_{min} とする．

(i) $SP^o(j')$ に解がある,

(ii) $f_{f'} + \lambda \geq f_j - \lambda$,

(iii) $j' < j$.

右から 2 番目の開設施設を f_{min} とした $SP^o(j)$ に解がないならば，(b1) $f_{min} < f_{j''} < f_j$ を満たす任意の $f_{j''}$ は f_j のメイトではなく，$SP^o(j)$ に解がない．(b2) $mate^o(f_{j+1})$ があるならば，$f_{min} \leq mate^o(f_{j+1})$ が成り立つ．

証明. (a) 背理法で証明する．$mate^o(f_{j+1}) + \lambda < f_j - \lambda$ が成り立つと仮定すると, $mate^o(f_{j+1})$ も f_j のメイトであるため矛盾．$mate^o(f_{j+1}) + \lambda \leq f_j - \lambda$ が成り立つと仮定すると, 補題 3.9 より $mate^o(f_{j+1})$ も f_j のメイトであるため矛盾.

(b1) 補題 3.12 より，明らかである．

(b2) 背理法で証明する．$f_{mate^o(j+1)} + \lambda < f_j - \lambda$ が成り立つと仮定すると, $f_{mate^o(j+1)}$ も f_j のメイトであるため矛盾．$f_{mate^o(j+1)} + \lambda \geq f_j - \lambda$ が成り立つと仮定すると, f_{min}

は $f_{mate^o(j)}$ ではなく $f_{mate^o(j+1)}$ であるので矛盾. □

補題 3.13 は，ある $f_{j'}$ まで f_j のメイトを調べた後，次に f_{j+1} のメイトを調べるとき，$f_{j'}$ から調べ始められることを意味する．

上記の補題に基づきアルゴリズム **find reserved** (λ, r)**-gathering** を設計する．

ある区間に利用者がいるかどうかや $s^o(f_j)$ の値を高速に計算するため，前処理を行う．前処理には $O(m+n)$ 時間かかる．常に $j' \leq j$ が成り立つので．$s^o(f_j)$ を計算するための最も内部の処理は高々$2m$ 回実行される．このアルゴリズムは $O(m)$ 時間で実行できる．以上より，次の補題が示せる．

補題 3.14. reserved (λ, r)-gathering 問題を解く $O(m+n)$ 時間アルゴリズムがある．

reserved r-gathering 問題の解のコストはいずれかの接続コストの値である．ここで，接続コストの値の種類は高々nm 通りである．

よって, 3.3 節のように, 行列探索法 [12, 11] を用いることで, reserved (λ, r)-gathering 問題を $O(\log(m+n))$ 回解くことで, reserved r-gathering 問題の解を計算できる．

したがって, 次の定理が成り立つ.

定理 3.15. reserved r-gathering 問題を解く $O((m+n)\log(m+n))$ 時間アルゴリズムがある.

Algorithm 4 find reserved (λ, r)-gathering

$j = 1$
/* 開設施設が 1 つの場合 */
while 区間 $[f_j - \lambda, f_j + \lambda]$ 中に c_1 と c_r がある **and** 区間 $(-\infty, f_j)$ 中に F^o の施設がない **do**
  r 番目の利用者が $c_{s^o(f_j)}$ となるよう $s^o(f_j) = r$ とする
  $j = j + 1$
end while
/* 開設施設が 2 つ以上の場合 */
$j' = 1$
while $j \leq m$ **do**
  while $j' < j$ **and** (区間 $(f_{j'} + \lambda, f_j - \lambda)$ 中に利用者がいる **or** $SP^o(j')$ に解がない **or** 区間 $(f_{j'}, f_j)$ 内に F^o の施設がある) **do**
    $j' = j' + 1$
  end while
  if $j' < j$ **then**
    /* 区間 $(f_{j'} + \lambda, f_j - \lambda)$ に利用者がいない **and** $SP^o(j')$ が解を持つ **and** 区間 $(f_{j'}, f_j)$ 中に F^o の施設がない */
    if $f_{j'} + \lambda < f_j - \lambda$ **and** 区間 $[f_{j'} + \lambda, f_j - \lambda]$ 中に利用者がいない **and** 区間 $[f_j - k, f_j + k]$ 中に利用者が r 人以上いる **then**
      区間 $[f_{j'} + \lambda, f_j - \lambda]$ 中の r 番目の利用者が $c_{s^o(f_j)}$ となる $s^o(f_j)$ を求める (Type 1)
    else if $f_{j'} + \lambda \geq f_j - \lambda$ **and** $c_{s^o(f_{j'})} \geq f_j - k$ **and** 区間 $(s^o(f_{j'}), f_j - \lambda]$ 中に利用者が r 人以上いる **then**
      区間 $(c_{s^o(f_{j'})}, f_j - \lambda]$ 中の r 番目の利用者が $c_{s(j)}$ となる $s^o(f_j)$ を求める (Type 2)
    else if $f_{j'} + \lambda \geq f_j - \lambda$ **and** $c_{s^o(f_{j'})} < f_j - k$ **and** 区間 $(c_{s^o(f_{j'})}, f_j - \lambda]$ 中に利用者が r 人以上いる **then**
      区間 $[f_{j'} + \lambda, f_j - \lambda]$ 中の r 番目の利用者が $c_{s^o(f_j)}$ となる $s^o(f_j)$ を求める (Type 3)
    end if
    /* $SP^o(j)$ は解なし */
  end if
  $j = j + 1$
end while
if 点 c_n から距離 λ 以内のある f_j について, $s^o(f_j)$ が定義されている **then**
  return Yes
else
  return No
end if

第4章　r-gathering問題を解く3近似アルゴリズム

Armon は r-gathering 問題を $O(mn + rn + n \log n)$ 時間で解く 3 近似アルゴリズムを設計した [7]. また, P≠NP ならば, 近似比 3 は改善できないことを示した [7]. 本章では, Armon のアルゴリズムに, (i) ソートをしない, (ii) 効率的なデータ構造を用いるという工夫を加えることで, アルゴリズムの計算時間を $O(mn)$ に改善する.

ここで, 利用者の集合のサイズを n, 施設候補地の集合のサイズを m である.

説明を簡単にするために, 接続コストの値はすべて異なると仮定する.

4.1　r-gathering問題を解く3近似アルゴリズム

Armon のアルゴリズム [7] を説明する. このアルゴリズムは 3 つのステップからなる.

ステップ 1 では, 各利用者 $c \in C$ について, その利用者 c にとって最適な, (1) 開設施設 $f \in F$ と, (2) f に c と一緒に割り当てる $r-1$ 人の利用者の集合を計算する.

ステップ 2 では, ステップ 1 で計算した各利用者にとって最適な部分割当を貪欲法に基づき確定していく. ただし, 既に割り当てた他の部分割当と競合するときは, その部分割当は行わないことにする. 利用者はいずれかの開設施設に割り当てなくてはならないので, この部分の割当コストは r-gathering のコストの下界であることがいえる.

ステップ 3 では, ステップ 2 で割り当てられなかった各利用者を, その利用者に最も近い開設施設に割り当てる.

以下にアルゴリズムの詳細を説明する.

ステップ 1 の詳細について説明する.

まず各施設 $f \in F$ ごとに, 施設 f を開設する場合にかかる最小のコストを次にのように計算する. f の開設コストは $op(f)$ である. これに加えて開設施設 f には r 人以上の利用者を割り当てる必要があるので, この部分のコストについて考えよう. f への接続コストが r 番目に小さい利用者を $c^r(f) \in C$ とする. また, f への接続コストが $co(c^r(f), f)$ 以下の c^r を含む r 人の利用者の集合を $N^r(f) \subset C$ とする. 開設施設 $f \in F' \subset F$ の接続コストは少なくとも $co(c^r(f), f)$ である.

指定した利用者 c を開設施設 f に割り当てるとき, この部分の割当にかかるコストの下界を次のように定義し, $lb(c,f)$ とする. 2 つの場合がある. $c \in N^r(f)$ ならば $lb(c,f) = \max\{co(c^r(f),f), op(f)\}$ とし, $c \notin N^r(f)$ ならば $lb(c,f) = \max\{co(c,f), op(f)\}$ とする.

指定した利用者 c をいずれかの開設施設に割り当てるとき, この部分の割当にかかるコストの下界を次にように定義し, $lb(c)$ とする. 利用者 $c \in C$ はいずれかの開設施設に割り当てるので, $lb(c) = \min_{f \in F}\{lb(c,f)\}$ とする.

$lb(c) = lb(c,f)$ となる f を c のベスト施設といい, $bestf(c)$ と書く. c のメイト $mate(c)$ を次のように定義する. 2 つの場合がある. $c \in N^r(bestf(c))$ ならば c の $mates(c)$ は $N^r(bestf(c)) \setminus \{c\}$ とし, $c \notin N^r(bestf(c))$ ならば c の $mates(c)$ は $N^r(bestf(c)) \setminus \{c^r(bestf(c))\}$ とする. すなわち, c を $bestf(c)$ に割り当てるとき, 接続コストが最小であるような他の $r-1$ 人の利用者の集合が c のメイトである.

任意の r-gatheirng A において, 各 $c \in C$ はいずれかの開設施設に割り当てるので, A のコストは少なくとも $lb(c)$ である. すなわち, A のコストは少なくとも $\max_{c \in C}\{lb(c)\}$ である.

ステップ 2 の詳細について説明する.

まず, $lb(c)$ で C をソートする. 各利用者 $c \in C$ に対して $lb(c)$ の昇順に, 次の処理をする. もし, c のベスト施設 $bestf(c)$ が未開設であり, かつ, c と c のメイト $mates(c)$ のすべての利用者が未割り当てならば, ベスト施設 $bestf(c)$ を開設し, c とメイト $mates(c)$ をベスト施設 $bestf(c)$ に割り当てる. 一方, ベスト施設 $bestf(c)$ が既に開設済み, もしくは c またはメイト $mates(c)$ のいずれかの利用者が既に割り当て済みならば, c はこのステップでは割り当てない. c は次のステップ 3 で割り当てる.

ステップ 3 の詳細について説明する. ステップ 2 で未割り当てである各利用者 c を, その c に最も近い開設施設に割り当てる.

r-gathering 問題を解く 3 近似アルゴリズム **Best-or-Rest** [7] を Algorithm5 に示す.

次に本文で提案するアルゴリズム **Best-or-Rest-without-Sort** を Algorithm6 に示す. ソートを行わず, 各施設 $f_j \in F$ ごとにフラグ $flag(f_j)$ を持つ.

r-gathering 問題の最適解のコストを OPT とする. Best-or-Rest-without-Sort について, 次の補題が成り立つ.

Algorithm 5 Best-or-Rest(C, F, r)

```
for c_i ∈ C do
  lb(c_i), bestf(c_i), mates(c_i) を計算する
end for
lb(c_i) で C をソートする
for lb(c_i) の昇順に各 c_i ∈ C do
  if mates(c_i) 中のすべての利用者がまだ割り当てられていない then
    bestf(c_i) を開設する
    c_i を bestf(c_i) に割り当てる
    for c_j ∈ mates(c_i) do
      c_j を bestf(c_i) に割り当てる                    /* Best-Assignment */
    end for
  end if
end for
for まだ割り当てられていない c_i ∈ C do
  c_i を c_i に最も近い開設施設に割り当てる              /* Rest-Assignment */
end for
```

補題 4.1. Best-or-Rest-without-Sort はコストが $3 \cdot OPT$ 以下の r-gathering を見つける.

証明. 各利用者 $c_i \in C$ の割当コストについて考える.

まず, Best-Assignment の部分のコストについて考えよう. Best-Assignment において. 各利用者 $c_i \in C$ は $mates(c_i)$ と共に $bestf(c_i)$ へ割り当てられる. このとき, この部分の接続コストと開設コストは $lb(c_i)$ である. $lb(c_i)$ は c_i を含む r-gathering のコストの下界であるため, $lb(c_i) \leq OPT$ である.

次に, Rest-Assignment の部分のコストについて考えよう. Rest-Assignment において, 各利用者 $c_i \in C$ は $bestf(c_i)$ ではないが, Best-Assignment 開設された施設のうち, 最も近い施設へ割り当てられる. 新たに施設を開設することはないので, この接続コストのみを考えればよい.

利用者 $c_i \in C$ を開設施設に割り当てるとする. Best-Assignment において, c_i は割り当てられなかったので, いずれかの $c_{i'} \in mates(c_i)$ を既に (1) $bestf(c_{i'})$ に割り当てた, もしくは (2) $c_{i'} \in mates(c_{i''})$ なる他の施設 $best(c_{i''}) \in F$ に割り当てた, のいずれかである.

(2) について考えよう. 接続コスト $co(c_i, bestf(c_{i''}))$ について, $co(c_i, bestf(c_{i''})) \leq co(c_i, bestf(c_i)) + co(c_{i'}, bestf(c_i)) + co(c_{i'}, bestf(c_{i''})) \leq lb(c_i) + lb(c_i) + lb(c_{i''}) \leq 3 \cdot OPT$ が成り立つ. (図 4.1 参照.) (1) の場合も同様に, $co(c_i, bestf(c_{i'})) \leq 3 \cdot OPT$ が成り立つ.

最後に, フラグに関する正当性について考えよう. $c_i \in C$ の Best-Assignment では, $flag(bestf(c_i))$ が off の場合のみ c_i の $bestf(c_i)$ への割当を試みる. $flag(bestf(c_i))$ が on

Algorithm 6 Best-or-Rest-without-Sort(C, F, r)

```
for c_i ∈ C do
  lb(c_i), bestf(c_i), mates(c_i) を計算する
end for
for f ∈ F do
  flag(f) を off とする
end for
for c_i ∈ C do
  if flag(bestf(c_i)) が off then
    if mates(c_i) 中のすべての利用者がまだ割り当てられていない then
      /* c_i のベスト施設が未開設かつ c_i のメイトがすべて未割り当て */
      flag(bestf(c_i)) を on にする
      bestf(c_i) を開設する
      c_i を bestf(c_i) に割り当てる
      for c_j ∈ mates(c_i) do
        c_j を bestf(c_i) に割り当てる                    /* Best-Assignment */
      end for
    else
      /* c_i のベスト施設が未開設または c_i のメイトのいずれかが割り当て済み */
      flag(bestf(c_i)) を on にする                  /* mates(c_i) のいずれかの利用者が
      割り当て済みなので, 以降 bestf(c_i) の開設は試みない */
    end if
  else
    /* flag(bestf(c_i)) が on */
  end if
end for
for まだ割り当てられていない c_i ∈ C do
  c_i を c_i に最も近い開設施設に割り当てる                /* Rest-Assignment */
end for
```

ならば, 既に, ある利用者 $c_{i'} \in C$ が $bestf(c_i)$ に $c_{i'}$ のメイトである他の $r-1$ 人の利用者と共に割当を試みている. したがって, c_i の Best-Assignment では, $mates(c_i)$ が既に割り当て済みかどうかを調べなくてよい.

したがって, Best-or-Rest-without-Sort は正しく動き, すべての利用者 $c_i \in C$ の開設施設への接続コストは高々 $3 \cdot OPT$ である. □

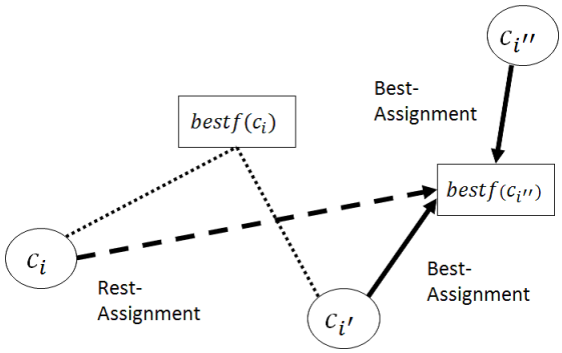

図 4.1: Rest-Assignment における接続コストの 3 近似性

次に, 提案した Best-or-Rest-without-Sort の計算時間が $O(mn)$ であることを示す. 一方, Armon の Best-or-Rest の計算時間は $O(mn + rn + n \log n)$ である.

補題 4.2. Best-or-Rest-without-Sort の計算時間は $O(mn)$ である.

証明. 選択アルゴリズム [8] を用いることで, 各施設 $f_j \in F$ に対して, $c^r(f_j)$ をそれぞれ $O(n)$ 時間で計算できる. $c^r(f_j)$ を用いて $N(f_j)$ をそれぞれ $O(n)$ 時間で計算できる. この処理に合計 $O(mn)$ 時間かかる. 各利用者 $c_i \in C$ に対して, 各 f_j について $c^r(f_j)$ と $N^r(f_j)$ が計算済みのとき, $lb(c_i)$, $bestf(c_i)$, $mates(c_i)$ を計算するのに $O(m)$ 時間かかる. よって, 合計 $O(mn)$ 時間かかる.

Best-Assignment では, 各利用者に対してフラグ $flag(bestf(c_i))$ が off であるかをチェックする. もしフラグが off ならば $mates(c_i)$ の $r-1$ 人が既に割り当て済みかどうかをチェックする. フラグを先にチェックするため, 各施設 $f_j \in F$ ごとに $r-1$ 人のチェックは高々 1 回しか行わない. よって, Best-Assignment には $O(rm + n)$ 時間かかる. $r > n$ ならば自明に解なしであるため, $r \leq n$ であることに注意する.

Rest-Assignment では, 各未割り当てな利用者に対して, 最も近い開設施設を $O(m)$ 時間で計算する. よって, Rest-Assignment には合計 $O(mn)$ 時間かかる.

□

補題 4.1, 4.2 より次の定理が成り立つ.

定理 4.3. r-gathering 問題を $O(mn)$ 時間で解く 3 近似アルゴリズムがある.

4.2 r-gathering with h-outlier 問題を解く 3 近似アルゴリズム

Best-or-Rest-without-Sort を少し改良すると r-gathering with h-outlier 問題を解く 3 近似アルゴリズムになる. (Best-or-Rest[7] も少しの改良で r-gathering with h-outlier 問題を解く 3 近似アルゴリズムになる.)

各利用者 $c_i \in C$ について $lb(c_i)$, $bestf(c_i)$, $mates(c_i)$ を計算した後, $lb(c_i)$ のうち, $n-h$ 番目に大きい値 lb^h を計算する. 少なくとも $n-h$ 人の利用者を開設施設に割り当てなければならないので, $lb^h \leq OPT$ である. ($lb^h < lb(c_i)$ であるような $c_i \in C$ について, $lb(c_i) \leq OPT$ は自明ではないことに注意する.)

次に, $lb(c_i) \geq lb^h$ であるような利用者 $c_i \in C$ の集合を C' とし, C から C' を取り除いた集合 $C \setminus C'$ を計算する. $|C'|$ のサイズは h であり, $|C \setminus C'|$ のサイズは $n-h$ である.

$C \setminus C'$ に対して Best-or-Rest-without-Sort のアルゴリズムを実行するとこの問題の近似解が得られる.

正当性について説明する. $c_i \in C \setminus C'$ の Best-Assignment において, $mates(c_i)$ 中に $c_{i'} \in C'$ であるような利用者 $c_{i'}$ がいないことを示せばよい.

$c_{i''} \in mates(c_i)$ ならば $lb(c_{i''}) \leq lb(c_i)$ であるため, $c_i \in C \setminus C'$ かつ $c_{i''} \in mates(c_i)$ ならば $c_{i''} \in C \setminus C'$ であり, 矛盾である. よって, $c_i \in C \setminus C'$ の $mates(c_i)$ に利用者 $c_{i'} \in C'$ は含まれない.

したがって, 次の定理が成り立つ.

定理 4.4. r-gathering with h-outlier 問題を $O(mn)$ 時間で解く 3 近似アルゴリズムがある.

第5章　円周上のdisperison問題を解くアルゴリズム

n 点の集合 P, P の 2 点間の距離 $d : P \times P \to \mathbb{R}$, 整数 k が与えられたとき, 指定されたコストを最大化するような k 個の点集合 $S \subset P$ を求める問題を dispersion 問題という. 指定するコストにより様々な問題が知られている [15, 16, 20, 23].

P が平面上の点集合とみなせるとき dispersion 問題は NP 困難である [20, 23]. P が直線上の点集合とみなせるとき, dipsersion 問題を解く動的計画法による $O(n \log n + kn)$ 時間アルゴリズム [20] が知られている. また, パス分割問題へ帰着してこれを行列探索法 [13] で解くことにより, $O(n)$ 時間で解くこともできる. これを拡張して, P が円周上の点集合とみなせるときも $O(n)$ 時間で解くことができる [22]. しかし, P が円周上の点集合とみなせるときの dispersion 問題を解く $O(n)$ 時間アルゴリズム [22] はとても複雑であり, 実装が難しい.

本節では, P が円周上の点集合とみなせるときの, $k = 3$ のときの dispersion 問題を解く単純な $O(n)$ 時間アルゴリズムを与える.

S 中の最も近い 2 点間の距離 $\min_{p,q \in S}\{d(p, q)\}$ を集合 $S \subset P$ のコスト $cost(S)$ とする. この $\min_{p,q \in S}\{d(p, q)\}$ が最大の $S \subset P$ を求める問題を dispersion 問題 (k-dispersion 問題) という.

5.1　円周上の 3-dispersion 問題を解くアルゴリズム

本節では $P = \{p_1, p_2, \ldots, p_n\}$ が円周上の点集合とみなせるときの 3-dispersion 問題を解く $O(n)$ 時間アルゴリズムを与える. ここで, $p_1, p_2, \ldots$ は円周上に時計回りに現れるとする.

$S \subset P$ ($|S| = 3$) 中の最も近い 2 点間の (円弧の) 距離 $\min_{p,q \in S}\{d(p, q)\}$ を S のコスト $cost(S)$ とする.

いくつかの定義を行う. $p_i \in P$ が与えられたとき, 点 p_i^ℓ, p_i^r を p_i, p_i^ℓ, p_i^r が円周上の正三角形の角であるような点と定義する.

中心角が 120° の p_i と p_i^ℓ 間の円弧を $A_\ell = (p_i, p_i^\ell)$ とし, 中心角が 120° の p_i^ℓ と p_i^r 間の (開) 円弧を $A_t = (p_i^\ell, p_i^r)$, 中心角が 120° の p_i^r と p_i 間の円弧を $A_r = (p_i^r, p_i)$ とする. (図

5.1 参照.)

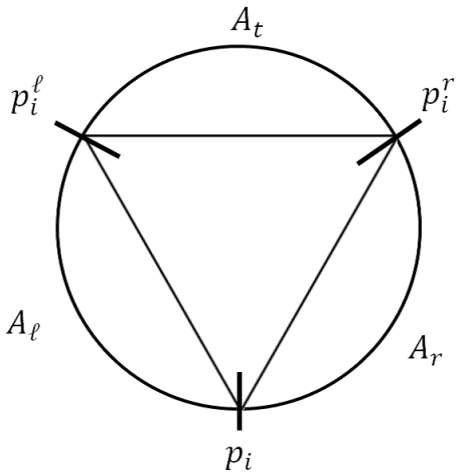

図 5.1: A_ℓ, A_t, A_r の図.

p_i を基準に, $p_\ell \in A_\ell$ かつ $p_r \in A_r$ である集合 $S = \{p_i, p_\ell, p_r\}$ を Type-LR と定義する. 同様に, p_i を基準に, $p_\ell \in A_\ell$ かつ $p_r \in A_\ell$ である S を Type-LL と定義し, p_i を基準に, $p_\ell \in A_r$ かつ $p_r \in A_r$ である S を Type-RR, p_i を基準に, $p_\ell \in A_\ell$ かつ $p_r \in A_t$ である S を Type-LT, p_i を基準に, $p_\ell \in A_t$ かつ $p_r \in A_r$ である S を Type-TR, p_i を基準に, $p_\ell \in A_t$ かつ $p_r \in A_t$ である S を Type-TT と定義する.

次の補題が成り立つ.

補題.5.1. P が円周上の点集合とみなせるとき, 3-dispersion 問題の最適解 S はある点 $p_i \in S$ を基準にした Type-LR または Type-TT である.

証明. $S = \{p_i, p_\ell, p_r\}$ とし, p_i を基準として p_ℓ と p_r は円周上に時計回りで現れるとする.

もし S が p_i を基準とした Type-LL ならば, S は p_ℓ を基準とした Type-LR である. (図 5.2 参照.) もし S が p_i を基準とした Type-RR ならば, S は p_r を基準とした Type-LR である. (図 5.2 と同様.) もし S が p_i を基準とした Type-LT ならば, (1) p_ℓ と p_r 間の円弧の中心角が 120° 未満であり, S は p_ℓ を基準とした Type-LR である, (図 5.3 参照.) または (2) p_ℓ と p_r 間の円弧の中心角が 120° 以上であり, S は p_r を基準とした Type-TT である. (図 5.4 参照.) もし S が p_i を基準とした Type-TR ならば, (1) p_ℓ と p_r 間の円弧の中心角が 120° 未満であり, S は p_r を基準とした Type-LR である, (図 5.3 と同様.) または (2) p_ℓ と p_r 間の円弧の中心角が 120° 以上であり, S は p_ℓ を基準とした Type-TT である. (図 5.3 と同様.) □

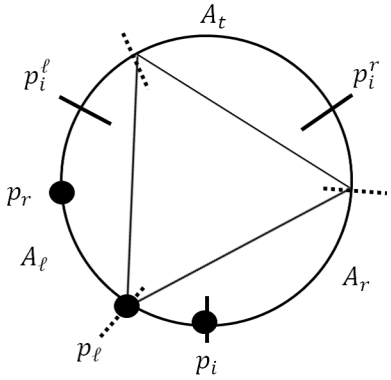

図 5.2: p_i を基準とした S が Type-LL の例.

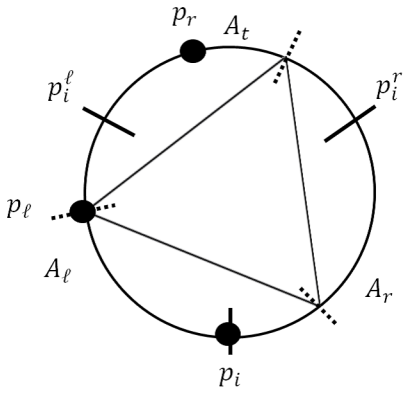

図 5.3: p_i を基準とした S が Type-LT で (1) p_ℓ と p_r 間の円弧の中心角が 120° 未満の例.

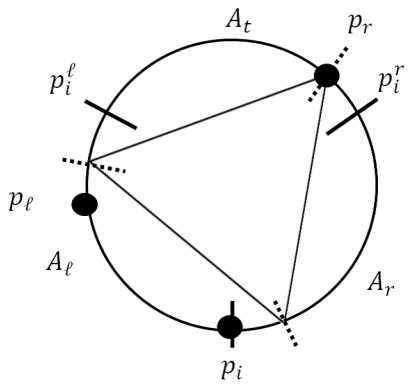

図 5.4: p_i を基準とした S が Type-LT で (2) p_ℓ と p_r 間の円弧の中心角が 120° 以上の例.

補題 5.2. (a) もし解 $S = \{p_i, p_\ell, p_r\}$ が p_i を基準とした Type-TT ならば, 時計回りに p_i^ℓ の次の P の点が p_ℓ であり, 反時計回りに p_i^r の次の P の点が p_r である.

(b) もし解 $S = \{p_i, p_\ell, p_r\}$ が p_i を基準とした Type-LR ならば, 反時計回りに p_i^ℓ の次の P の点が p_ℓ であり, 時計回りに p_i^r の次の P の点が p_r である.

証明. (a) S は Type-TT なので, $\min\{d(p_i, p_\ell), d(p_\ell, p_r), d(p_r, p_i)\} = d(p_\ell, p_r)$ である. よって, $d(p_\ell, p_r)$ が最大となるように S を選択すると, S は最大コストとなる.

(b) S は Type-LR なので,$\min\{d(p_i, p_\ell), d(p_\ell, p_r), d(p_r, p_i)\} \neq d(p_\ell, p_r)$ である. よって, $d(p_i, p_\ell)$ と $d(p_i, p_r)$ が最大となるように S を選択すると, S は最大コストとなる. □

各 $p_i \in P$ について, 時計回りに p_i^ℓ の次の P の点を計算する. この計算にかかる時間は $O(n)$ である. 同様に, 各 $p_i \in P$ について, 反時計回りに p_i^r の次の P の点を計算する.

この計算にかかる時間は $O(n)$ である. これらを前処理として実行する.

前処理より, 各 $p_i \in P$ について, $O(1)$ 時間で (1) コスト $cost(S)$ が最大である p_i を基準とした Type-LR の S, (2) コスト $cost(S)$ が最大である p_i を基準とした Type-TT の S を計算し, コストが大きい方を選択できる.

したがって, 次の定理が成り立つ.

定理 5.3. P を円周上の点集合とみなせるとき, 3-dispersion 問題を解く $O(n)$ 時間アルゴリズムがある.

第6章　直線上のPcS-dispersion問題を解くアルゴリズム

dispersion 問題は指定するコストにより様々な問題が知られている [15, 16, 20, 23].

本章では, 文献 [15, 16] 中のいくつかの問題について考える. n 点の集合 P, P の 2 点間の距離 $d : P \times P \to \mathbb{R}$, 整数 k が与えられたとき, 各点 $u \in S$ から最も近い S 中の他の c 点への距離の和を点 u のコスト $cost_{PcS}(u)$ とし, これらの最小値 $\min_{u \in S}\{cost_{PcS}(u)\}$ を集合 $S \subset P$ のコスト $cost_{PcS}(S)$ とする. この $\min_{u \in S}\{cost_{PcS}(u)\}$ が最大の $S \subset P$ を求める問題を partial c sum dispersion 問題 (PcS-dispersion 問題) という.

6.1　直線上のP2S-dispersion問題

本節では $P = \{p_1, p_2, \ldots, p_n\}$ が水平直線上の点集合とみなせるときの partail 2 sum dispersion 問題を解くアルゴリズムを 2 つ与える. ここで $p_1, p_2, \ldots$ は水平線上に左から右に順に現れるとする. 1 つ目は動的計画法による $O(kn^2 \log n)$ 時間アルゴリズムである. 2 つ目は行列探索法 [12] による $O(n \log n)$ 時間アルゴリズムである.

6.1.1　動的計画法によるアルゴリズム

部分問題 $P2S(h, i; k)$ を定義する.

P_i を P の部分集合 $\{p_1, p_2, \ldots, p_i\}$ とする. $p_h \in P_i$ と整数 $k \geq 3$ が与えられたとき, $|S| = k$ かつ, S の最も右側の 2 つの要素が p_h と p_i $(h < i)$ であるような, P_i の部分集合 S を考える. $p \in S$ の点コスト $cost_{P2S}(p)$ は S 中の p に最も近い他の 2 点への距離の和であり, S のコスト $cost_{P2S}(S)$ は $\min_{p \in S}\{cost_{P2S}(p)\}$ である. このとき, コストが最大である S を求める問題を部分問題 $P2S(h, i; k)$ という. また, $P2S(h, i; k)$ の解のコストを $cost_{P2S}(h, i; k)$ とする. つまり, この部分問題は S の右端の 2 点が指定された P2S-dispersion 問題に相当する.

もし p_1 が S に含まれないならば, S 中の最も左側の点を S から取り除き, その後 p_1 を加えた集合のコストは $cost_{P2S}(S)$ 以上である. p_i についても同様のことがいえる. よって, 部分問題 $P2S(h, i; k)$ について, P_i の両端点 p_1, p_i を含む $P2S(h, i; k)$ の解 S が存在する. 次の補題が成り立つ.

補題 6.1. $k = 3$ ならば, $cost_{P2S}(h, i; k) = d(p_1, p_i)$ である.

証明. $P2S(h, i; 3)$ の解は $\{p_1, p_h, p_i\}$ である. よって, $\{p_1, p_h, p_i\}$ 中の点のコストはそれぞれ $cost_{P2S}(p_1) = d(p_1, p_h) + d(p_1, p_i)$, $cost_{P2S}(p_h) = d(p_1, p_h) + d(p_h, p_i) = d(p_1, p_i)$, $cost_{P2S}(p_i) = d(p_1, p_i) + d(p_h, p_i)$ である. $\{p_1, p_h, p_i\}$ のコストは点のコストの最小値であるので, $cost_{P2S}(h, i; 3) = d(p_1, p_i)$ である. □

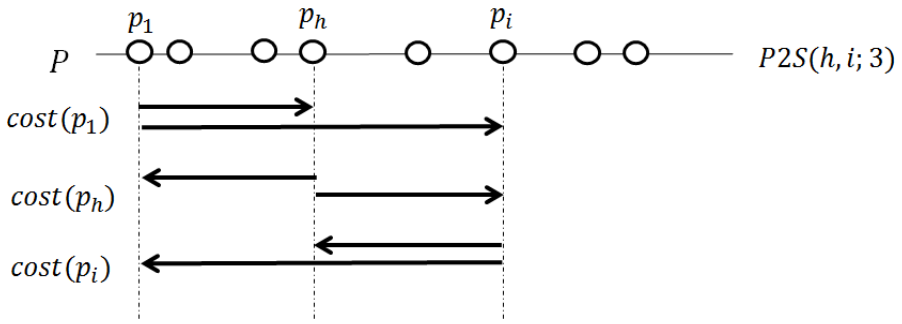

図 6.1: $P2S(h, i; 3)$ の点コストの例図.

$cost_{P2S}(h, i; k)$ は S の点 $p \in S$ のコストの最小値である. したがって, $cost_{P2S}(p_i) > cost_{P2S}(p_h)$ が常に成り立つため, $cost_{P2S}(h, i; k)$ を計算するとき, p_i の点コスト $cost_{P2S}(p_i)$ を求めなくてよい.

補題 6.2. $k \geq 4$ ならば, $cost_{P2S}(h, i; k) = \max_{h'=k-2,k-1,\ldots,h-1} \min\{cost_{P2S}(h', h; k - 1), d(p_{h'}, p_i)\}$ が成り立つ.

証明. $SP(h, i; k)$ の解を S とし, $p_{h'}$ を S 中の左から 3 番目の点とする. $|S| = k$ より, $h' \geq k - 2$ が成り立つ.

点 $p_x \in S$ において $cost_{P2S}(h, i; k) = cost_{P2S}(p_x)$ が成り立つとする. 次の 3 つの場合がある.

Case 1: $x < h$.

$cost_{P2S}(p_x) = cost_{P2S}(h', h; k - 1)$ が成り立ち, $cost_{P2S}(p_x) \leq cost_{P2S}(p_h) \leq d(p_{h'}, p_i)$ が成り立つ. したがって, $cost_{P2S}(p_x) = \min\{cost_{P2S}(h', h; k - 1), d(p_{h'}, p_i)\}$ が成り立つ. (図 6.2 参照.)

Case 2: $x = h$.

更に 2 つの場合がある. $p_{h''}$ を S の左から 4 番目の点とする.

もし $cost_{P2S}(p_x) = d(p_{h'}, p_h) + d(p_{h''} + p_h)$ ならば, $cost_{P2S}(p_x) = d(p_{h'}, p_h) + d(p_{h''} + p_h) > cost_{P2S}(p_{h'})$ が成り立つ. これは, $cost_{P2S}(h, i; k) = cost_{P2S}(p_x)$ に矛盾する. (図 6.3 参照.)

もし $cost_{P2S}(p_x) = d(p_{h'}, p_h) + d(p_h, p_i)$ ならば, $cost_{P2S}(p_x) = d(p_{h'}, p_i) \leq cost_{P2S}(h', h; k-1)$ が成り立つ. したがって, $cost_{P2S}(p_x) = \min\{cost_{P2S}(h', h; k-1), d(p_{h'}, p_i)\}$ が成り立つ. (図 6.4 参照.)

Case 3: $x = i$.

$cost_{P2S}(p_h) < cost_{P2S}(p_i)$ より, この場合は生じない. (図 6.5 参照.)

よって, 考慮すべきすべての h' に対して $\min\{cost_{P2S}(h', h; k-1), d(p_{h'}, p_h)\}$ の値を計算し, それらのうちの最大値を求めることは, $cost_{P2S}(h, i; k)$ を求めることと同値である. □

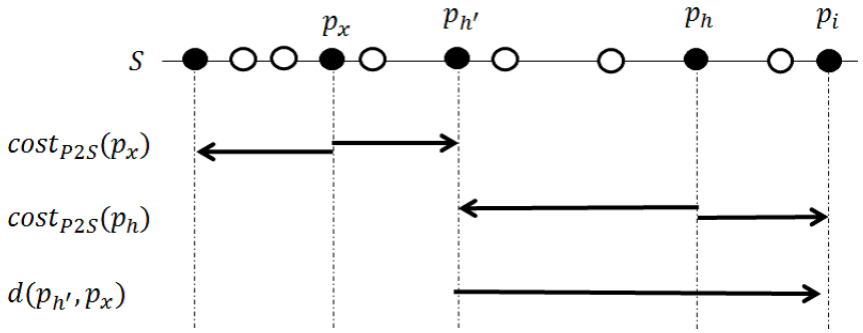

図 6.2: Case 1 の例.

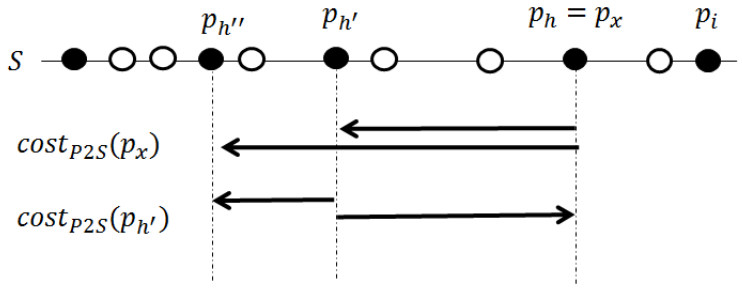

図 6.3: Case 2 で $cost_{P2S}(p_x) = d(p_{h'}, p_h) + d(p_{h''} + p_h)$ であるときの例.

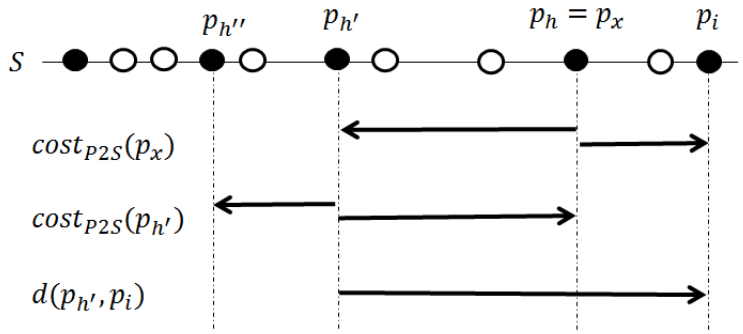

図 6.4: Case 2 で $cost_{P2S}(p_x) = d(p_{h''}, p_{h'}) + d(p_{h''} + p_h)$ であるときの例.

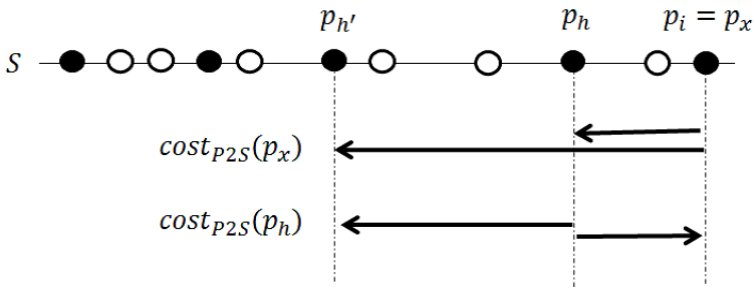

図 6.5: Case 3 の例.

部分問題の個数は高々kn^2 個であり, 補題 6.2 より, 各部分問題を解く計算時間は $O(n)$ である.

P2S-dispersion 問題を解くアルゴリズム Find-P2S-dispersion を示す.

したがって, 次の定理が成り立つ.

定理 6.3. P2S-dispersion 問題を解 $O(kn^3)$ 時間アルゴリズムがある.

部分問題にコストについて, 次の補題が成り立つ.

補題 6.4. $cost_{P2S}(h', h; k-1)$ は h' に対して単調非減少である.

証明. 単調非減少でないと仮定する. $h_L < h_R$ であるような, P 中の点 p_{h_L} と p_{h_R} において, $cost_{P2S}(h_L, h; k-1) > cost_{P2S}(h_R, h; k-1)$ が成り立つ. ここで, $cost_{P2S}(h', h; k-1)$ は $\min_{p \in S}\{cost_{P2S}(p)\}$ であることに注意する.

$P2S(h_L, h; k-1)$ の解を S_L とし, S_L から p_{h_L} を取り除き, p_{h_R} を加えた集合を S' とする.

Algorithm 7 Find-P2S-dispersion(P, k)

/* $P(h, i; 3)$ を計算 */ /* Case $k = 3$ */
for $i = 3, 4, \ldots, n$ **do**
  for $h = 2, 3, \ldots, i-1$ **do**
    $cost_{P2S}(h, i; 3) = d(p_1, p_i)$
  end for
end for
/* $P(h, i; k)$ を計算 */ /* Case $k > 4$ */
for $k' = 4, 5, \ldots, k$ **do**
  for $i = k', k'+1, \ldots, n$ **do**
    for $h = k'-1, k', \ldots, i-1$ **do**
      $cost_{P2S}(h, i; k') = 0$
      /* 最大コストを計算 */
      for $h' = k'-2, k'-1, \ldots, h-1$ **do**
        if $cost_{P2S}(h, i; k) > \max\{cost_{P2S}(h, i; k'), \min\{cost_{P2S}(h', h; k'-1), d(p_{h'}, p_i)\}\}$ **then**
          $cost_{P2S}(h, i; k) = \max\{cost_{P2S}(h, i; k'), \min\{cost_{P2S}(h', h; k'-1), d(p_{h'}, p_i)\}\}$
          /* $P(h, i; k')$ の解の右から 3 番目の点 $3rm(h, i; k')$ を $p_{h'}$ とする */
          $3rm(h, i; k') = h'$
        end if
      end for
    end for
  end for
end for
/* 最適コストを計算 */
$cost = 0$
$i_1 = 1, i_k = n$
for $h = k-1, k, \ldots, n-1$ **do**
  if $cost_{P2S}(h, n; k) > cost$ **then**
    $cost = cost_{P2S}(h, n; k)$
    /* 解の右から 2 番目の点を p_h とする */
    $i_{k-1} = h$
  end if
end for
/* 最適解を計算 */
for $h = k-1, k, \ldots, n-1$ **do**
  $i_{k'} = 3rm(i_{k'+1}, i_{k'+2}; k'+2)$
end for
return $S = \{p_{i_1}, p_{i_2}, \ldots, p_{i_k}\}$

S_L 中の p_{h_L} の左隣の点を p_x とし, その P_x の左隣の点を p_y とする.

S_L 中の点 p_x のコスト $cost_{P2S}(p_x)$ は, S' 中の点 p_x のコスト $cost_{P2S}(p_x)$ 以下である. また, S_L 中の点 p_y のコスト $cost_{P2S}(p_y)$ は, S' 中の点 p_y のコスト $cost_{P2S}(p_x)$ 以下である.

S_L 中の点 p_{h_L} のコスト $cost_{P2S}(p_{h_L})$ は $\min\{d(p_x, p_h), d(p_y, p_{h_L}) + d(p_x, p_{h_L})\}$ である. また, S' 中の点 p_{h_R} のコスト $cost_{P2S}(p_{h_R})$ は $\min\{d(p_x, p_h), d(p_y, p_{h_R}) + d(p_x, p_{h_R})\}$ である. $y < x$ より, $d(p_y, p_{h_L}) + d(p_x, p_{h_L}) < d(p_y, p_{h_R}) + d(p_x, p_{h_R})$ が成り立つ.

よって, S_L 中の点 p_{h_L} のコスト $cost_{P2S}(p_{h_L})$ は, S' 中の点 p_{h_L} のコスト $cost_{P2S}(p_{h_L})$ 以下である.

したがって, $cost_{P2S}(h_L, h; k-1) \leq \min_{p \in S_L}\{cost_{P2S}(p)\} \leq \min_{p \in S'}\{cost_{P2S}(p)\} \leq cost_{P2S}(h_R, h; k-1)$ が成り立つ. これは仮定に矛盾する. □

よって, h' の増加に対して, $\min\{cost_{P2S}(h', h; k-1), d(p_{h'}, p_i)\}$ は単調非減少である. (つまり, h' の減少に対して, $\min\{cost_{P2S}(h', h; k-1), d(p_{h'}, p_i)\}$ は単調非増加である.) したがって, 二分探索を $\log n$ 回行うことで, $\min\{cost_{P2S}(h', h; k-1), d(p_{h'}, p_i)\}$ が最大である $p_{h'}$ を見つけることができる. この手法により, 部分問題を $O(\log n)$ 時間で解くことができる.

したがって, 次の定理が成り立つ.

定理 6.5. P2S-dispersion 問題を解く $O(kn^2 \log n)$ 時間アルゴリズムがある.

6.1.2 行列探索法によるアルゴリズム

まず, P2S-dispersion 問題の判定問題を解くアルゴリズムを与える.

整数 k と実数 λ が与えられたとき, $|S| = k$ であり, かつ S のコスト $cost_{P2S}(S)$ が λ 以上であるような P の部分集合 S が存在するかどうかを判定する問題を (λ, k)-P2S-dispersion 問題という.

補題 6.6. (λ, k)-P2S-dispersion 問題の解が Yes ならば, $\{p_1, p_2, p_n\}$ は S に含まれる.

証明. $p_1, p_n \in P$ を含む解 S が存在する. したがって,$p_2 \in P$ を含む解が存在することを示す.

まず, p_1, p_n を含み p_2 を含まないような, $|S| = k$ である, (λ, k)-P2S-dispersion 問題の解 $S = \{p_1, s_2, \ldots, s_{k-1}, p_n\} \subset P$ を考える. またここで, $S' = \{p_1, p_2, s_3, \ldots, s_{k-1}, p_n\}$ であるような部分集合を考える.

S は (λ, k)-P2S-dispersion 問題の解なので, S 中の点 s_2 のコストは $cost_{P2S}(s_2) = \min\{d(p_1, s_2) + d(s_2, s_3), d(s_2, s_3) + d(s_2, s_4)\} = \min\{d(p_1, s_3), d(s_2, s_3) + d(s_2, s_4)\} \geq \lambda$ である. S' 中の点 p_2 のコストは $cost_{P2S}(p_2) = \min\{d(p_1, s_2) + d(p_2, s_3), d(p_2, s_3) + d(p_2, s_4)\} = \min\{d(p_1, s_3), d(p_2, s_3) + d(p_2, s_4)\}$ である.

$d(p_2, s_3) + d(p_2, s_4) \geq d(s_2, s_3) + d(s_2, s_4)$ より, $cost_{P2S}(p_2) \geq cost_{P2S}(s_2) \geq \lambda$ が成り立つ. したがって, $cost_{P2S}(S') \geq \lambda$ であり, p_2 を含む解が存在する. □

(λ, k)-P2S-dispersion 問題を解くアルゴリズム Decide-P2S-dispersion を示す. 次の補題が成り立つ.

Algorithm 8 Decide-P2S-dispersion(P, k, λ)

```
s_1 = p_1, s_2 = p_2
c = 3
for i = 3, 4, ..., n do
  if d(s_{c-2}, p_1) ≥ λ then
    s_c = p_i
    c = c + 1
  end if
end for
if c > k then
  return Yes
else
  return No
end if
```

補題 6.7. $|S| = k$ であり, かつ S のコスト $cost_{P2S}(S)$ が λ 以上であるような P の部分集合 S が存在するかどうかをアルゴリズム (λ, k)-P2S-dispersion は正しく判定する.

証明. 背理法で証明する. $|S'| = k$ かつ $cost_{S'} \geq \lambda$ であるような $S' = \{s'_1, s'_2, \ldots, s'_k\} \subset P$ が存在するが, アルゴリズムは NO を返すとする.

アルゴリズムが選択した点集合を $S = \{s_1, s_2, \ldots\}$ とする. このとき, p_1 と p_2 は S に含まれている. 補題 5.7 より, $s'_1 = p_1$ と $s'_2 = p_2$ が成り立つ.

ここで, 点 $s \in P$ の座標値を $x(s)$ とする. $x(s_j) > x(s'_j)$ である最小の添字を j とする. (そのような j が存在しないならばアルゴリズムは Yes を返すため, 矛盾である.) このとき, $x(s_{j-1}) \leq x(s'_{j-1})$ と $x(s_{j-2}) \leq x(s'_{j-2})$ が成り立つ.

2 つの場合がある.

S' 中の点 s'_{j-1} のコスト $cost_{P2S}(s'_{j-1})$ が $d(s'_{j-2}, s'_j)$ ならば, $\lambda \leq d(s'_{j-2}, s'_j)$ と $\lambda \leq d(s_{j-2}, s_j)$ が成り立つ. このとき, アルゴリズムは s'_j かそれより左側の点を選択できた. これは, アルゴリズムが s_j を選択したことに矛盾する.

S' 中の点 s'_{j-1} のコスト $cost_{P2S}(s'_{j-1})$ が $d(s'_{j-2}, s'_j)$ でないならば, s'_{j-1} の近くの 2 点は, s'_{j-3} と s'_{j-2} の組, または s'_j と s'_{j+1} の組である. どちらの組だとしても, $cost_{P2S}(s'_{j-1}) <$

$d(s'_{j-2}, s'_j)$ が成り立つ. したがって, $\lambda \leq cost_{P2S}(s'_{j-1}) < d(s'_{j-2}, s'_j)$ が成り立つ. よって, この場合もアルゴリズムが s_j を選択したことに矛盾する. □

定理 6.8. (λ, k)-P2S-dispersion 問題を解く $O(n)$ 時間アルゴリズムがある.

3.3 節のように行列探索法 [13] を用いることで, (λ, k)-P2S-dispersion 問題を $O(\log n)$ 回解くことで, P2S-dispersion 問題を解くことができる. したがって, 次の定理が成り立つ.

定理 6.9. P2S-dispersion 問題を解く $O(n \log n)$ 時間アルゴリズムがある.

6.2 直線上の PcS-dispersion 問題を解くアルゴリズム

本節では, P が直線上の点集合とみなせるときの PcS-dispersion 問題を解くアルゴリズムを与える. このアルゴリズムは動的計画法に基づく.

部分問題を定義する.

$p_{h_{c-1}}, p_{h_{c-1}}, \ldots, p_{h_1} \in P_i$ と整数 $k \geq c+1$ が与えられたとき, $|S| = k$ かつ, S の最も右側の c 個の要素が $p_{h_{c-1}}, p_{h_{c-1}}, \ldots, p_{h_1}$ と p_i $(h_{c-1} < h_{c-2} < \cdots < h_1 < i)$ であるような, P_i の部分集合 S を考える. このとき, コストが最大である S を求める問題を部分問題 $PcS(h_{c-1}, h_{c-2}, \ldots, h_1, i; k)$ という. また,$PcS(h_{c-1}, h_{c-2}, \ldots, h_1, i; k)$ の解のコストを $cost_{PcS}(h_{c-1}, h_{c-2}, \ldots, h_1, i; k)$ とする. つまり, この部分問題は S の右端の c 点が指定された PcS-dispersion 問題に相当する.

ある部分集合の点 $p_x \in S$ について, $cost_{PcS}(h_{c-1}, h_{c-2}, \ldots, h_1, i; k) = cost_{PcS}(p_x)$ とする. このとき, 次の 2 つの補題が成り立つ.

補題 6.10. S 中の p_x に近い左側の $\lceil c/2 \rceil$ 点と p_x に近い右側の $\lfloor c/2 \rfloor$ 点のそれぞれから, p_x までの距離の和を $c(p_x)$ とする. このとき, $cost_{PcS}(p_x) = c(p_x)$ が成り立つ.

証明. $cost_{PcS}(p_x) = c(p_x)$ が成り立たないと仮定する.

ある整数 $g \neq 0$ に対して, S 中の p_x に近い左側の $\lceil c/2 \rceil + g$ 点と p_x に近い右側の $\lfloor c/2 \rfloor - g$ 点のそれぞれから, p_x までの距離の和が $cost_{PcS}(p_x)$ である場合を考えよう.

まず, c が偶数であるときを考える. S 中の p_x の左隣の点を p_L とし, p_x の右隣の点を p_L とする. $g > 0$ ならば, $cost_{PcS}(p_x) > cost_{PcS}(p_L)$ が成り立ち, $g < 0$ ならば, $cost_{PcS}(p_x) > cost_{PcS}(p_R)$ が成り立つ. よって, これは矛盾である.

次に, c が奇数であるときを考える. S 中の p_y に近い左側の $\lfloor c/2 \rfloor$ 点と p_y に近い右側の $\lceil c/2 \rceil$ 点のそれぞれから, p_y までの距離の和を $c(p_y)$ とする. c が奇数ならば, $c(p_x) = c(p_y)$ が成り立つことに注意する. 後は奇数の場合と同様に, 矛盾を示すことができる. □

補題 6.11. S 中の右側の $\lceil c/2 \rceil$ 点の中に p_x は含まれない.

証明. 補題 6.10 より明らか. □

補題 6.11 より, 点 $p \in S$ の $cost_{PcS}(p)$ が最小である $cost_{PcS}(h_{c-1}, h_{c-2}, \ldots, h_1, i; k)$ を計算するとき, 右側の $\lceil c/2 \rceil$ 点について省略できる.

補題 6.6 に見られるように, $PcS(h_{c-1}, h_{c-2}, \ldots, h_1, i; k)$ の解 S に p_1 が含まれる. 次の 3 つの補題が成り立つ.

補題 6.12. $\{p_1, p_{h_{c-1}}, p_{h_{c-2}}, \ldots, p_{h_1}, p_i\}$ 中の, 左から $\lfloor c/2 \rfloor + 1$ 番目の点を p_y とする. このとき, $k = c + 1$ ならば, $cost_{PcS}(h_{c-1}, h_{c-2}, \ldots, h_1, i; k) = c(p_y)$ が成り立つ.

証明. 補題 6.1 と同様. $S = \{p_1\} \cup \{p_{h_{c-1}}, p_{h_{c-2}}, \ldots, p_{h_1}, p_i\}$ である. ここで, $|S| = c + 1$ より, 補題 5.11 から, $cost_{PcS}(h_{c-1}, h_{c-2}, \ldots, h_1, i; k) = c(p_y)$ が成り立つ. □

補題 6.13. c が偶数ならば, $cost_{PcS}(h_{c-1}, h_{c-2}, \ldots, h_1, i; k) = \max_{h'=k-c,k-c+1,\cdots,h_{c-1}-1} \min\{cost_{PcS}(h', h_{c-1}, h_{c-2}, \ldots, h_1; k-1), c(p_{h_{c/2}})\}$ が成り立つ.

証明. $cost_{PcS}(h_{c-1}, h_{c-2}, \ldots, h_1, i; k)$ の解を S とし, S 中の右端から $c + 1$ 番目の点を $p_{h'}$ とする.

$|S| = k$ より, $h' \geq k - c$ が成り立つ. S 中のある点 p_x において, $cost_{PcS}(h_{c-1}, h_{c-2}, \ldots, h_1, i; k) = cost_{PcS}(p_x)$ と仮定する. 次の 3 つの場合がある. このとき, S 中の点 $p_{h_{c/2}}$ の右側には点が $c/2$ 個あることに注意する.

Case 1: $x < h_{c/2}$

ここで, $cost_{PcS}(p_x) = cost_{PcS}(h', h_{c-1}, h_{c-2}, \ldots, h_1; k-1)$ と $cost_{PcS}(p_x) \leq cost_{PcS}(p_{h_{c/2}}) \leq c(p_{h_{c/2}})$ が成り立つ. よって, $cost_{PcS}(p_x) = \min\{cost_{PcS}(h', h_{c-1}, h_{c-2}, \ldots, h_1; k-1), c(p_{h_{c/2}})\}$ が成り立つ.

Case 2: $x = h_{c/2}$

ここで, $cost_{PcS}(p_x) = c(p_{h_{c/2}}) \leq cost_{PcS}(h', h_{c-1}, h_{c-2}, \ldots, h_1; k-1)$ が成り立つ. よって, $cost_{PcS}(p_x) = \min\{cost_{PcS}(h', h_{c-1}, h_{c-2}, \ldots, h_1; k-1), c(p_{h_{c/2}})\}$ が成り立つ.

Case 3: $x > h_{c/2}$

補題 6.11 より, この場合は起こりえない. □

補題 6.14. c が奇数ならば, $cost_{PcS}(h_{c-1}, h_{c-2}, \ldots, h_1, i; k) = \max_{h'=k-c,k-c+1,\cdots,h_{c-1}-1} \min\{cost_{PcS}(h', h_{c-1}, h_{c-2}, \ldots, h_1; k-1), c(p_{h_{\lfloor c/2 \rfloor}})\}$ が成り立つ.

証明. 補題 6.13 と同様. □

c を定数とすると, $c(p)$ は $O(1)$ 時間で計算できる. 部分問題の個数は高々 kn^c 個であり, 1つの部分問題を解くのに $O(n)$ 時間かかる. したがって, $O(kn^{c+1})$ 時間で PcS-dispersion 問題を解くことができる.

定理 6.15. PcS-dispersion 問題を解く $O(kn^{c+1})$ 時間アルゴリズムがある.

第7章　結論

一般的な r-gathering 問題は NP 困難であり, これを解く多項式時間アルゴリズムの設計は困難である. そこで, 多項式時間で解くことのできるクラスの解明や, 近似アルゴリズムの設計が望まれる. 本文では, r-gatheirng 問題および関連する問題を解く様々なアルゴリズムを設計した.

利用者の集合や施設の集合が直線上の点集合とみなせるとき, r-gathering 問題やそれと関連する問題を解く多項式時間アルゴリズムを設計した.
多項式時間アルゴリズムの設計できるクラスについて解明するため, 直線上の点集合より一般的なクラスの問題を多項式時間で解くことができるか検討したい. 例えば, 利用者の集合や施設の集合が木グラフ上の点集合とみなせるときに, r-gathering 問題や関連する問題を多項式時間で解くアルゴリズムを設計したい.

一方, 既存の近似アルゴリズムより高速な近似アルゴリズムも設計した. 近似比の改善やより高速な近似アルゴリズムを設計したい.

本研究で設計した複数のアルゴリズムは, 行列探索法を用いているが, 行列探索法の実装は非常に複雑である. そこで, より単純で実装が容易なアルゴリズムを設計したい.

本文では, 各開設施設に少なくとも r 人以上の利用者がいるような割当について考えた. さらに, 各開設施設に r 人以上 q 人以下の利用者がいるというような制限を加えた割当は, 自然な広い応用が期待される. このような割当に関する各種の問題についても考えてみたい.

謝辞

本研究を進めるにあたり, 研究面だけでなく様々な面において，多大なるご指導，ご鞭撻を賜った群馬大学大学院理工学府 中野眞一 教授に心より厚く御礼申し上げます. また, 日頃からご指導いただいた同大学 宮田洋行助教に感謝の意を表します.

本論文を主査していただいた山崎浩一教授をはじめ，日頃からお世話くださった，天野一幸教授，荒木徹准教授, 藤田憲悦准教授, 鏑木喜雄技術職員に深く感謝いたします.

そして，日頃の様々なイベントでお世話になった中野研究室の修士 2 年の小川航平君, 小池優君, バトバヤル ドゥラムダリさん, 修士 1 年の神谷瑠飛君，小島大輝君, 学部 4 年の兜森崇平君，佐藤寛斗君，平澤紹君，ALIVIA AYU SAVIRA さん，及び歴代のメンバー, 日頃お世話になっている皆様に心より感謝を申し上げます.

最後に, これまで見守ってくれた両親に深く感謝いたします。

参考文献

[1] G. Aggarwal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas and A. Zhu, *"Achieving anonymity via clustering",* Transactions on Algorithms, 6, Article No.49 (2010).

[2] T. Akagi and S. Nakano, *"On (k, r)-gatherings on a Road",* Proc. of Forum on Information Technology, FIT 2013, RA-001 (2013).

[3] T. Akagi and S. Nakano, *"On min-max (k,r(f))-gatherings",* The 16th Japan Conference on Discrete and Computational Geometry and Graphs, Tokyo, September 18, (2013).

[4] T. Akagi, R. Arai and S. Nakano *"Faster min-max r-gatherings",* IEICE Transactinos on Fundamentals of Electronic, Communications and Computer Science, Vol. E99A, No.6, pp.1149-1151, (2016).

[5] T. Akagi and S. Nakano, *"On r-gatherings on the Line",* IEICE Transactions on Information and Systems, Vol.100D, No.3, pp.428-433, (2017).

[6] T. Akagi, T. Araki, H, Ishikawa and S. Nakano, *"The Partial Sum Dispersion Problme on the Line",* Proc. of JCDCGˆ3, 2017, Tokyo University of Science, (2017).

[7] A. Armon, *"On min-max r-gatherings",* Theoretical Computer Science, 412, pp.573-582, (2011).

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms, Third Edition, MIT Press, (2009).

[9] Z. Drezner, *"Facility Location: A Survey of Applications and Methods",* Springer, (1995).

[10] Z. Drezner and H.W. Hamacher, *"Facility Location: Applications and Theory",* Springer, (2004).

[11] H. Fournier, and A. Vigneron, *"Fitting a Step Function to a Point Set",* Proc of ESA 2008, Lecture Notes in Computer Science, 5193, pp.442-453, (2008).

[12] G. Frederickson and D. Johnson, *"Generalized Selection and Ranking: Sorted Matrices",* SIAM Journal on Computing, 13, pp.14-30, (1984).

[13] G. Frederickson, *"Optimal Algorithms for Tree Partitioning",* Proc. of SODA '91, pp.168-177, (1991).

[14] A. Meyerson and R. Williams, *"On the Complexity of Optimal k-Anonymity",* In Proc. SIGMOD-SIGACT-SIGART symposium on Princeiples of database system on ACM, 23rd, pp.223-228, (2004).

[15] T. L. Lei and R. L. Church, *"A Unified Model for Dispersion Facilities",* Geographich Analysis, 45, pp.401-408, (2013).

[16] T. L. Lei and R. L. Church, *"On the Unified Dispersion Problem : Efficient Formulations and Exact Algorithms",* European Journal of Operational research, 241, pp.622-630, (2015).

[17] J. Li, K. Yi and Q. Zhang, *"Clustering with Diversity",* Proc. of ICALP 2010, Lecture Notes in Computer Science, 6198, pp.188-200, (2010).

[18] J. Y. Liu, *"A Randomized Algorithm for Weighted Approximation of Points by a Step Function",* Proc. of COCOA 2010, pp.300-308, (2010).

[19] A. Machanavajjhala, D. Kifer, J. Gehrke and M. Venkitasubramaniam, *"ℓ-diversity : Privacy beyond k-anonymity",* J. ACM Transactions on Knowledge Discovery from Data, 1, Article 3, (2007).

[20] S. S. Ravi, D, J. Rosenkrantz and G. K. Tayi, *"Hueristic and Special Case Algorithms for Dispersion Problems",* Operations Resarch, 42, pp.299-310, (1994).

[21] L. Sweeney, *"Simple Demographics Often Identify People Uniquely",* Carnegie Mellon University, Data Privary Wlrking Paper 3, Pittsburgh, (2000).

[22] K. H. Tsai and D. W. Wang, *"Optimal Algorithms for Circle Partitioning",* Proc. of COCOON 1997, LNCS 1276, pp.304-310, (2014).

[23] D. W. Wang and Y. S. Kuo, *"A Study on Two Geometric Location Problems",* Information Processing Letters, 28, pp.281-286, (1988).