最適化と行動モデル

浦田淳司（筑波大学）

第25回行動モデル夏の学校

2026年9月8日

尤度最大化

行動モデル（離散選択モデル）：

選択効用　$U_{ia} = \sum_k \beta_{ak} x_{iak} + \varepsilon_{ia}$

確定効用　誤差効用

選択確率　$P_i(a|\boldsymbol{\beta}) = \frac{\sum_k \beta_{ak} x_{ak}}{\sum_{\forall a} \sum_k \beta_{ak} x_{ak}}$

a: 選択肢
i: 個人
β: 選好パラメータ
x: 説明変数
δ_i: 選択結果
$\varepsilon \sim i.i.d\ Gumbel$

パラメータ推定（尤度最大化）　$\widehat{\boldsymbol{\beta}} = argmax \left(LL(\boldsymbol{\beta}|\boldsymbol{\delta}) = \sum_{\forall i} \delta_i(a) \log P_i(a|\boldsymbol{\beta}) \right)$

最適化

尤度最大化の手法

https://www.bin.t.u-tokyo.ac.jp/kaken/pdf/mnl.pdf より

9 対数尤度関数の最大化

```
76  ##### 対数尤度関数 fr の最大化#####
77
78  ##パラメータ値の最適化
79  res <- optim(b0,fr, method = "Nelder-Mead", hessian = TRUE, control=list(fnscale=-1))
80
```

パラメータ"b0"に対して対数尤度関数"fr"を最大化し，その結果を"res"に出力します．

※optim 関数について

optim 関数は以下のような引数をとります．

```
optim(par, fn, gr = NULL, method = "Nelder-Mead", lower = -Inf, upper = Inf,
control = list(), hessian = FALSE)
```

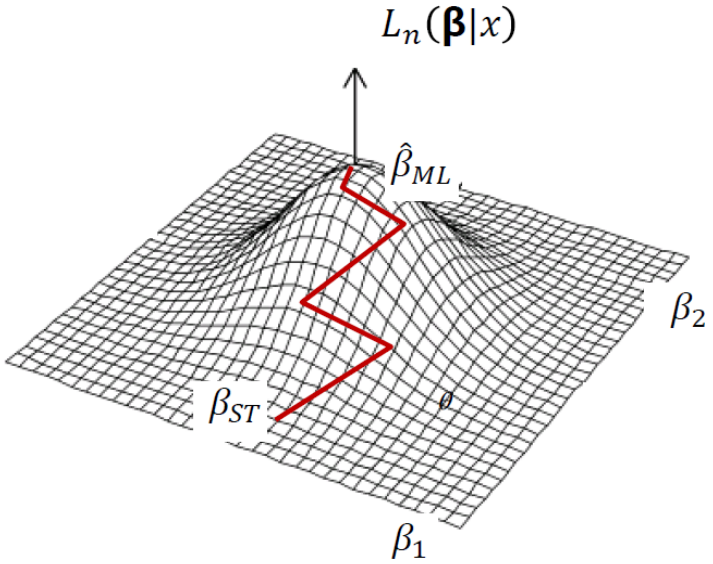
par	初期値(必須)
fn	最適化する目的関数(必須)
gr	一階偏微分関数の指定(NULL でデフォルト関数使用)
method	最適化手法の指定(5 種類から選ぶ，Nelder-Mead(デフォルト)，BFGS，CG，L-BFGS-B，SANN)
lower	L-BFGS-B 法での変数の下限(デフォルトは-Inf)
upper	L-BFGS-B 法での変数の上限(デフォルトは Inf)
control	制御パラメータ fnscale　関数に与える比例定数，optim 関数は最小化を行うので，最大化のときは，control=list(fnscale=-1)と指定します．ここでは尤度を最大化しています． hessian　最適解のヘッセ行列を返すかどうか，パラメータ推定では，t 値計算にヘッセ行列が必要なので，hessian = TRUE とします．

尤度（連続量）を
最大化するための手法

最大化アルゴリズムの考え方

周りがあまり見えない中で，近傍の情報から頂点を目指す

- 対数尤度関数の段階的な最大化
  - 初期値を与える
  - 初期値周りで勾配(1次微分）等を用いて次の推定値の方向を決める
  - 初期値付近で１次微分，２次微分を用いて適切に次の点を決めて推定値を得る
  - 収束基準(一次微分ベクトル)で判定し，収束していない場合は，現在の値から次の推定値に移る

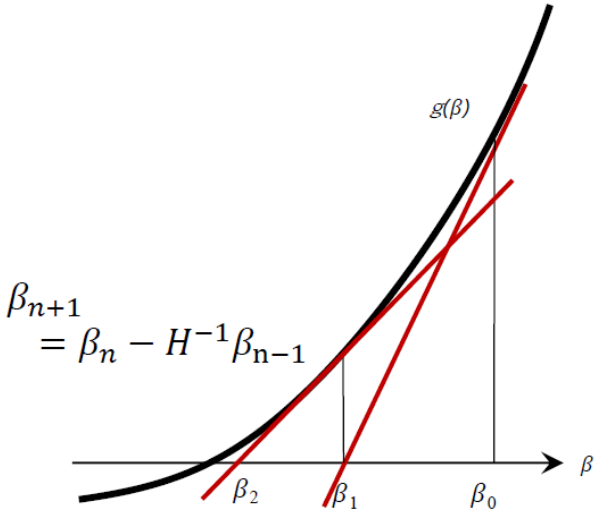


佐々木先生
BMSS2023

代表的な繰り返し計算法

尤度関数を最大化　尤度関数の一階微分＝０を解く

- Newton-Raphson法
  - テイラー展開の１次近似を利用して進める
- 準Newton法（BFGS，L-BFGS法）
  - ヘッセ行列を，パラメータの差分と一階微分の差分を用いて逐次近似する．
  - L-BFGSはヘッセ行列の更新式を展開して，初期値と差分の関数和で表す．



H:尤度関数の二階微分　ヘッセ行列
g: 尤度関数の一階微分

佐々木先生
BMSS2023

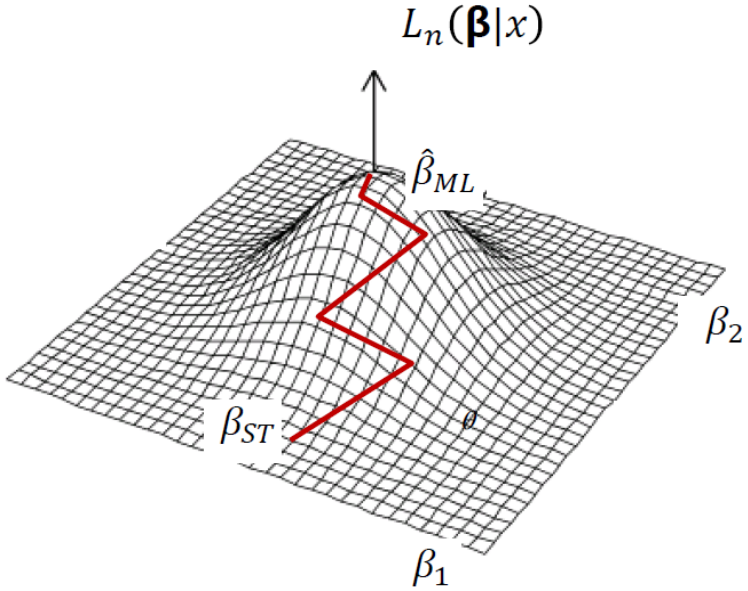
シミュレーション最適化

微分によらない最適化

Simulated ANNealing（SANN）

※Kirkpatrick et al. (1983), Belisle (1992)

$$\max_{\beta}\left(LL(\boldsymbol{\beta}|\boldsymbol{\delta}) = \sum_{\forall i}\delta_i(a)\log P_i(a|\boldsymbol{\beta})\right)$$



現在の解β_tから近傍解yを生成

$$y \sim q(y|\beta_t)$$

目的関数の差を算出

$$\Delta = LL(y) - LL(\beta_t)$$

改善解の場合

$$if\ \Delta \geq 0\ \Rightarrow \beta_{t+1} = y$$

悪化解の場合

$$if\ \Delta < 0\ \Rightarrow \beta_{t+1} = y\ with\ P(accept) = \exp\left(\frac{\Delta}{T_t}\right)$$

温度Tを徐々に冷却

$$T_{t+1} = \alpha T_t\ (0 < \alpha < 1)$$

1.尤度最大化（連続量最適化）

a. 基礎編

b.入れ子型の行動モデル

詳しくは22回のスライド

2.離散最適化と行動モデル

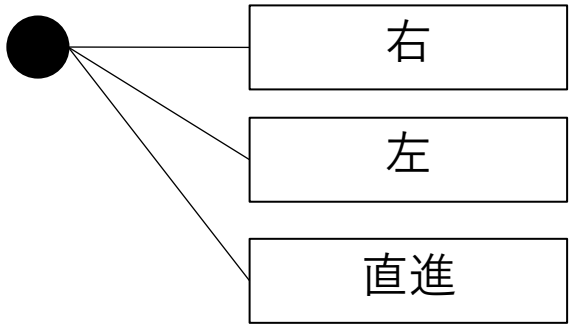
a. 解法

b. 目的関数

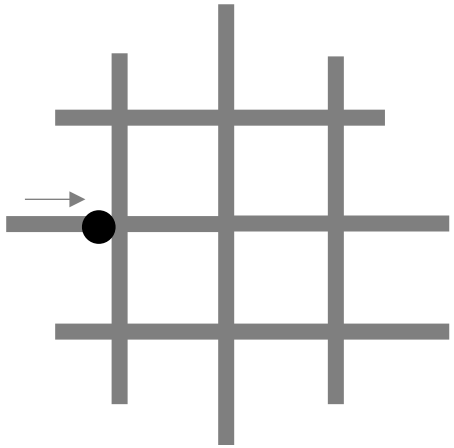
動的選択問題とは

静学問題

時刻t0

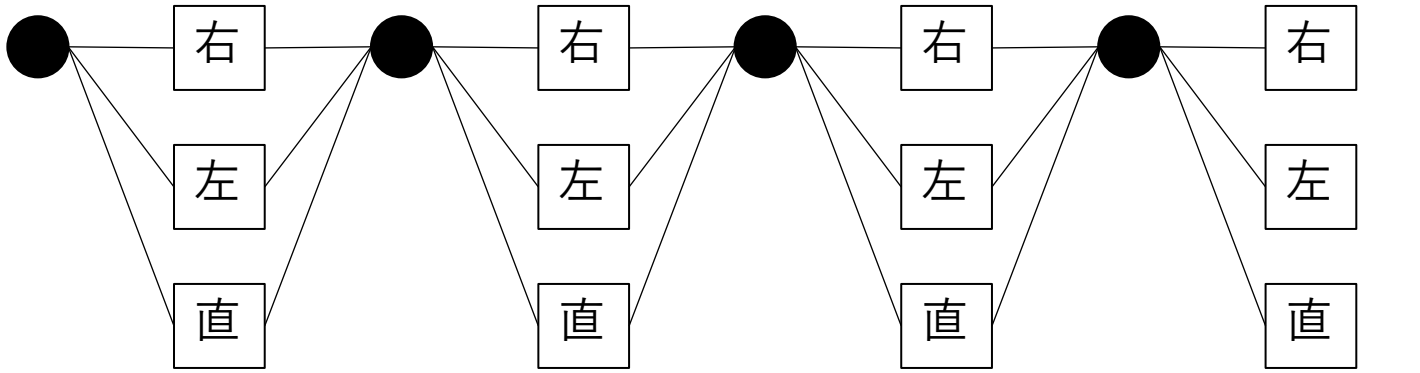


3選択肢



動学問題

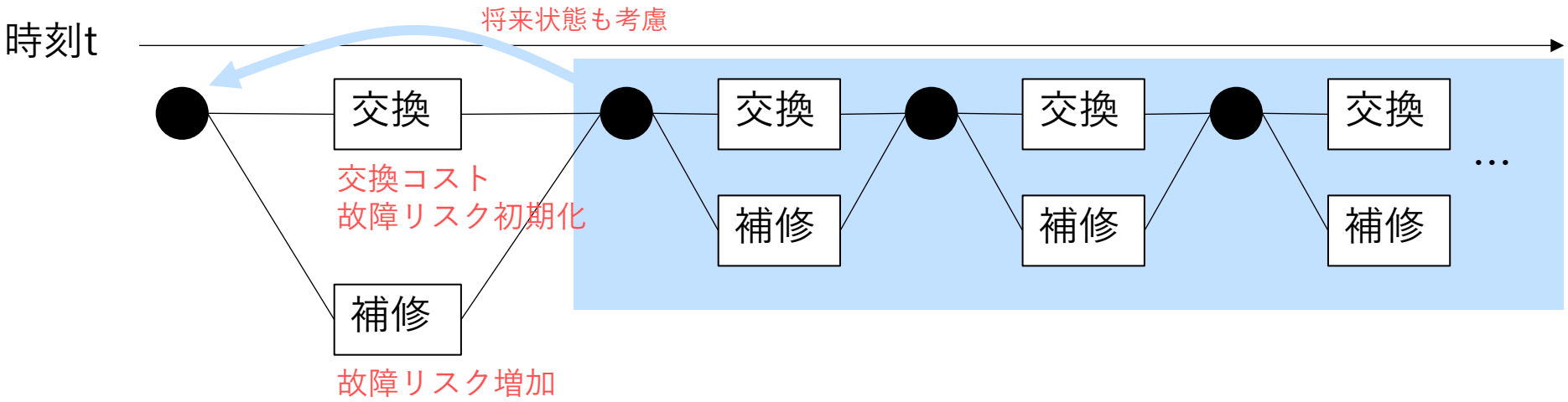
時刻t



3^4選択肢

動的離散選択モデル(概要)

Rust, J.(1987) のバスエンジン補修/交換の問題を例に説明



ある期の効用関数u

$$u(x, d, \varepsilon; \theta_1, \mathrm{RC}) = v(x, d; \theta_1, \mathrm{RC}) + \varepsilon(d)$$

$$v(x, d; \theta_1, \mathrm{RC}) = \begin{cases} -c(x; \theta_1), & \text{if } d = 0 \\ -\mathrm{RC} - c(0; \theta_1), & \text{if } d = 1 \end{cases}$$

状態xの推移確率（一次マルコフ性を仮定）

$$p(x_{t+1}, \varepsilon_{t+1} \mid x_t, \varepsilon_t, d_t; \theta_2, \theta_3)$$

d=1: 交換，d=0: 補修のみ
x: バスエンジンの状態
ε : 非観測項
c:補修コスト
θ_1: 補修パラメータ
RC: 交換コスト
θ_2, θ_3: 状態推移パラメータ

動的離散選択モデル(定式化)

将来の価値関数は，時間割引を考慮した効用の最大化となる

$$\mathrm{V}(x_t, \varepsilon_t) = \max_{\{d_t, d_{t+1}, d_{t+2}, \ldots\}} \mathbb{E}\left[\sum_{\tau=t}^{\infty} \beta^{\tau-t} u(x_\tau, d_\tau, \varepsilon_\tau; \theta_1, \mathrm{RC})\right]$$

時間割引率　$\beta \in (0,1)$

最適意思決定は無限期間先までを考慮しており，期によらず一定である

- 今期と次期の状態により，価値関数を定義
- 次期の期

入れ子になっているP・EVをどう計算するか

$$\mathrm{V}(x, \varepsilon) = \max_d \left\{ \underbrace{\nu(x, d; \theta_1, \mathrm{RC}) + \varepsilon(d)}_{\text{今期の効用}} + \beta \int_{x'} \int_{\epsilon'} \underbrace{\mathrm{V}(x', \varepsilon')}_{\text{次期の効用}} \underbrace{p(x', \varepsilon' | x, \varepsilon, d; \theta_2, \theta_3)}_{\text{今期の選択後の次期状態の推移確率}} dx'\, d\varepsilon' \right\}$$

次期の期待価値関数EV

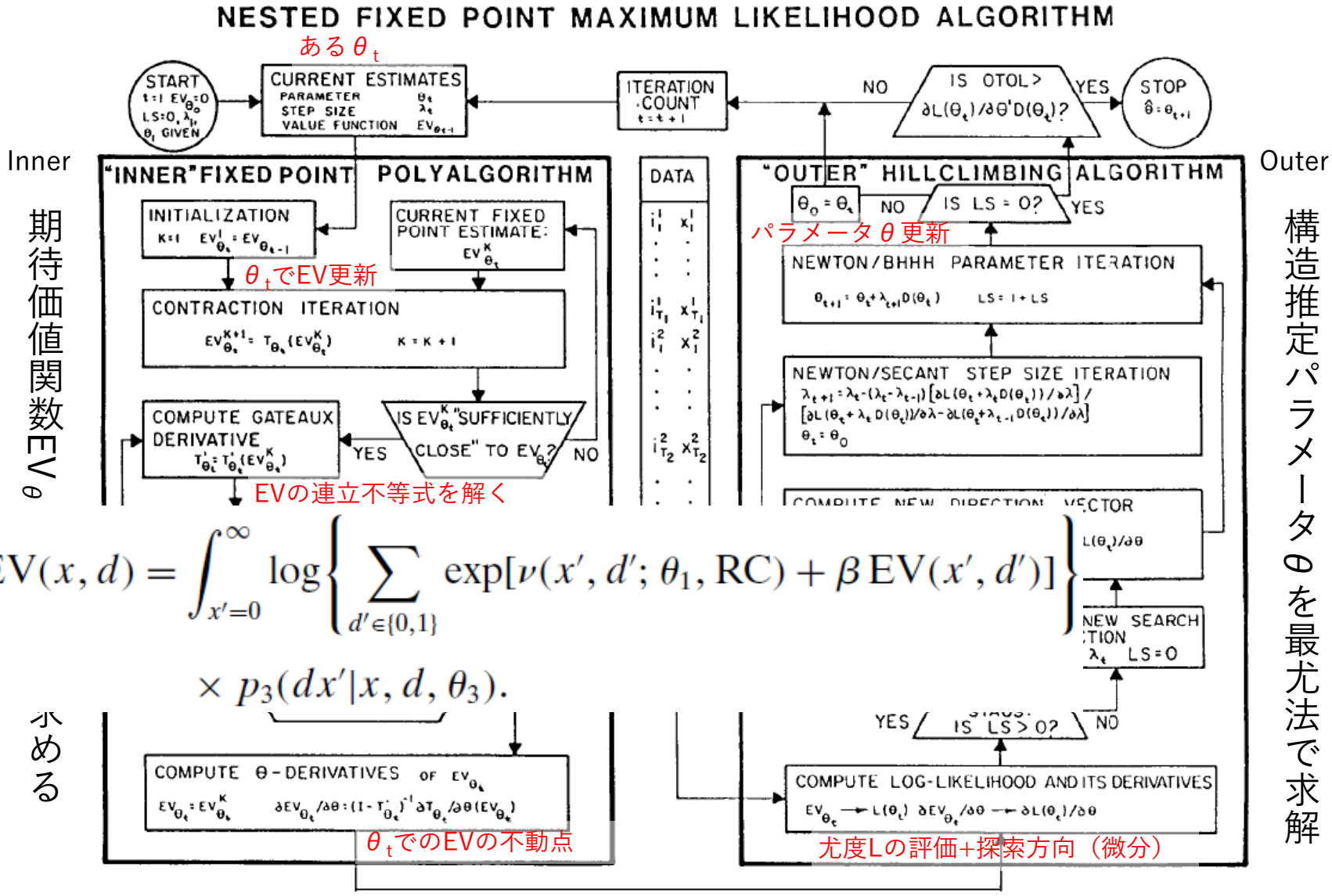
次期状態(x', ε')

選択確率　$P(d|x; \theta) = \dfrac{\exp[\nu(x, d; \theta_1, \mathrm{RC}) + \beta\, \mathrm{EV}(x, d)]}{\sum_{d' \in \{0,1\}} \exp[\nu(x, d'; \theta_1, \mathrm{RC}) + \beta\, \mathrm{EV}(x, d')]}$

尤度　$\ell(\theta) = \prod_{i=1}^{M} \ell_i(X^i; \theta) = \prod_{i=1}^{M} \prod_{t=2}^{T} P(d_t^i | x_t^i; \theta)\, p_3(x_t^i | x_{t-1}^i, d_{t-1}^i; \theta_3)$

パラメータ推定方法①：NFXP法

Rust, J.(1988) Statistical Models of Discrete Choice Processes, Transportation Research Part B, Vol. 22(2), pp. 125-158.



$$\mathrm{EV}(x,d) = \int_{x'=0}^{\infty} \log\left\{ \sum_{d' \in \{0,1\}} \exp[\nu(x', d'; \theta_1, \mathrm{RC}) + \beta\, \mathrm{EV}(x', d')] \right\} \times p_3(dx'|x, d, \theta_3).$$

期待価値関数EVθを求める

構造推定パラメータθを最尤法で求解

推定法②：NPL(擬似最尤推定法)

Aguirregabiria and Mira(2002)

Step1: EV_0をランダムに与える

Step2: EV_kを次式の右辺に代入し，尤度最大化するパラメータθ_kを求める

$$P(d|x;\theta) = \frac{\exp[\nu(x,d;\theta_1,\mathrm{RC}) + \beta\,\mathrm{EV}(x,d)]}{\sum_{d'\in\{0,1\}} \exp[\nu(x,d';\theta_1,\mathrm{RC}) + \beta\,\mathrm{EV}(x,d')]}$$

Step3: EV_kとθ_kを用いて，次式よりEv_{k+1}を求める ⇔NFXPのInner Algorithm

$$\mathrm{EV}(\hat{x}_k,d) = \sum_{j=0}^{J} \log\left\{ \sum_{d'\in\{0,1\}} \exp[\nu(x',d';\theta_1,\mathrm{RC}) + \beta\,\mathrm{EV}(x',d')] \right\} \times p_3(x'|\hat{x}_k,d,\theta_3).$$

仮のEVのまま，繰り返し計算する

収束判定: $|EV_{k+1} - EV_k|$と$|\theta_{k+1} - \theta_k|$が十分小さければ収束
収束していなければ，step2に戻る

無限回の繰り返し計算によって不動点を得ることができ，NFXPと漸近等価

推定法3：MPEC型アルゴリズム

(Su & Judd (2012))

MPEC（Mathematical Programming with Equilibrium Constraints)を用いた求解

等価な均衡制約条件付き最適化問題と定式化.

$$\max_{(\theta,\underline{\mathrm{EV}})} \frac{1}{M}\mathcal{L}(\theta, \mathrm{EV}) \qquad \text{（最適化問題）}$$

$$\text{subject to} \quad \mathrm{EV} = T(\mathrm{EV}, \theta) \qquad \text{（不動点制約）}$$

NFXPが不動点の算出プロセスを毎回解いているのに比べて，
Bellmanの等式を一度評価すればよいので，計算負荷は小さい.

1.尤度最大化（連続量最適化）

a. 基礎編

b.入れ子型の行動モデル

2.離散最適化と行動モデル

a. 解法

b. 目的関数

離散最適化と行動モデル

最適化問題とは

目的関数

$$\min_{x} f(x)$$

制御変数
（離散/連続）

$$\text{subject to } g(x) = 0, h(x) > 0$$

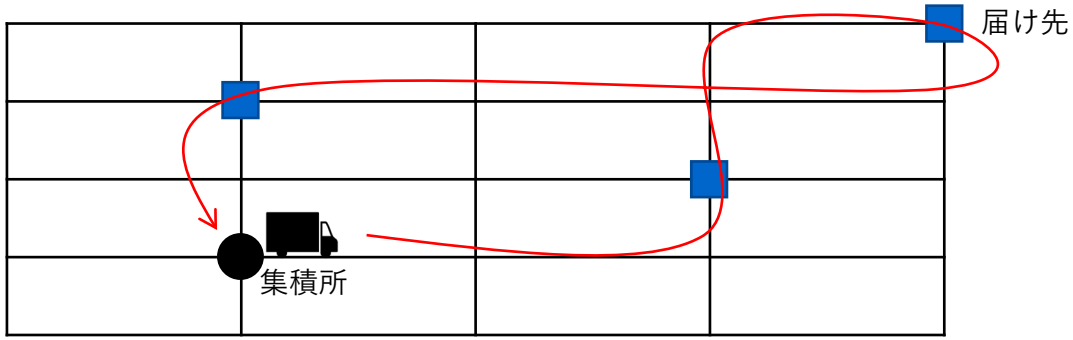
制約条件（等式制約，不等式制約）

交通分野の最適化とは？

$$\min_{x} f(x)$$

$$\text{subject to } g(x) = 0, h(x) > 0$$

	制御変数	目的関数	制約条件
トラック配送	配送経路	旅行時間最小化	配送場所

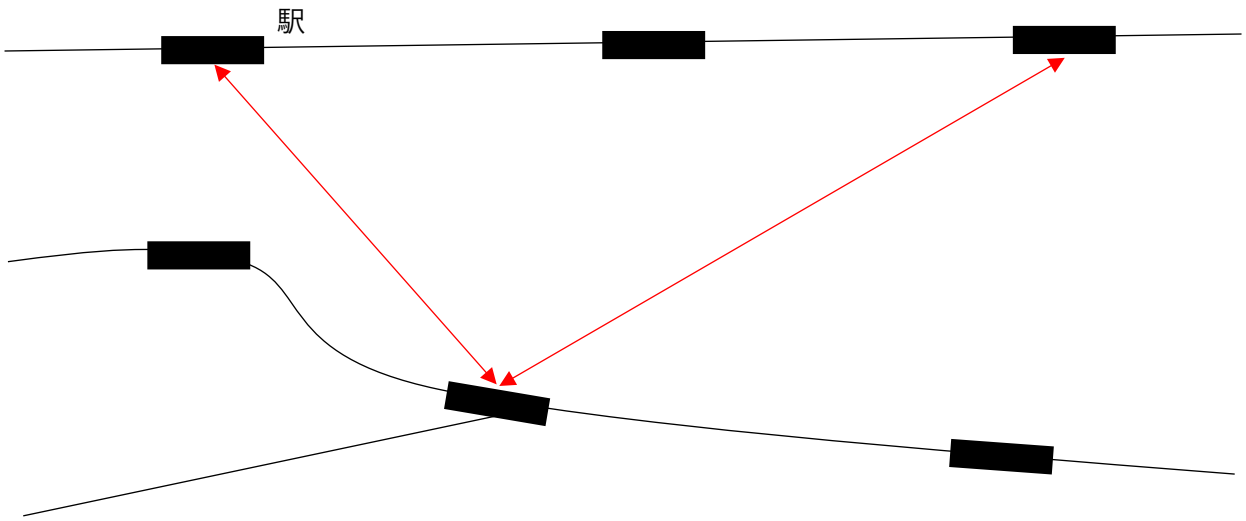


交通分野の最適化とは？

$$\min_{x} f(x)$$

$$\text{subject to } g(x) = 0, h(x) > 0$$

	制御変数	目的関数	制約条件
トラック配送	配送経路	旅行時間最小化	配送場所
バス路線	路線選定	収益最大化？ 乗客効用最大化？	車両台数



交通分野の最適化とは？

$$\min_{x} f(x)$$
$$\text{subject to } g(x) = 0, h(x) > 0$$

	制御変数	目的関数	制約条件
トラック配送	配送経路	旅行時間最小化	配送場所
バス路線	路線選定	収益最大化？ 乗客効用最大化？	車両台数
シェアサイクル	ポート位置	収益最大化	自転車配置数

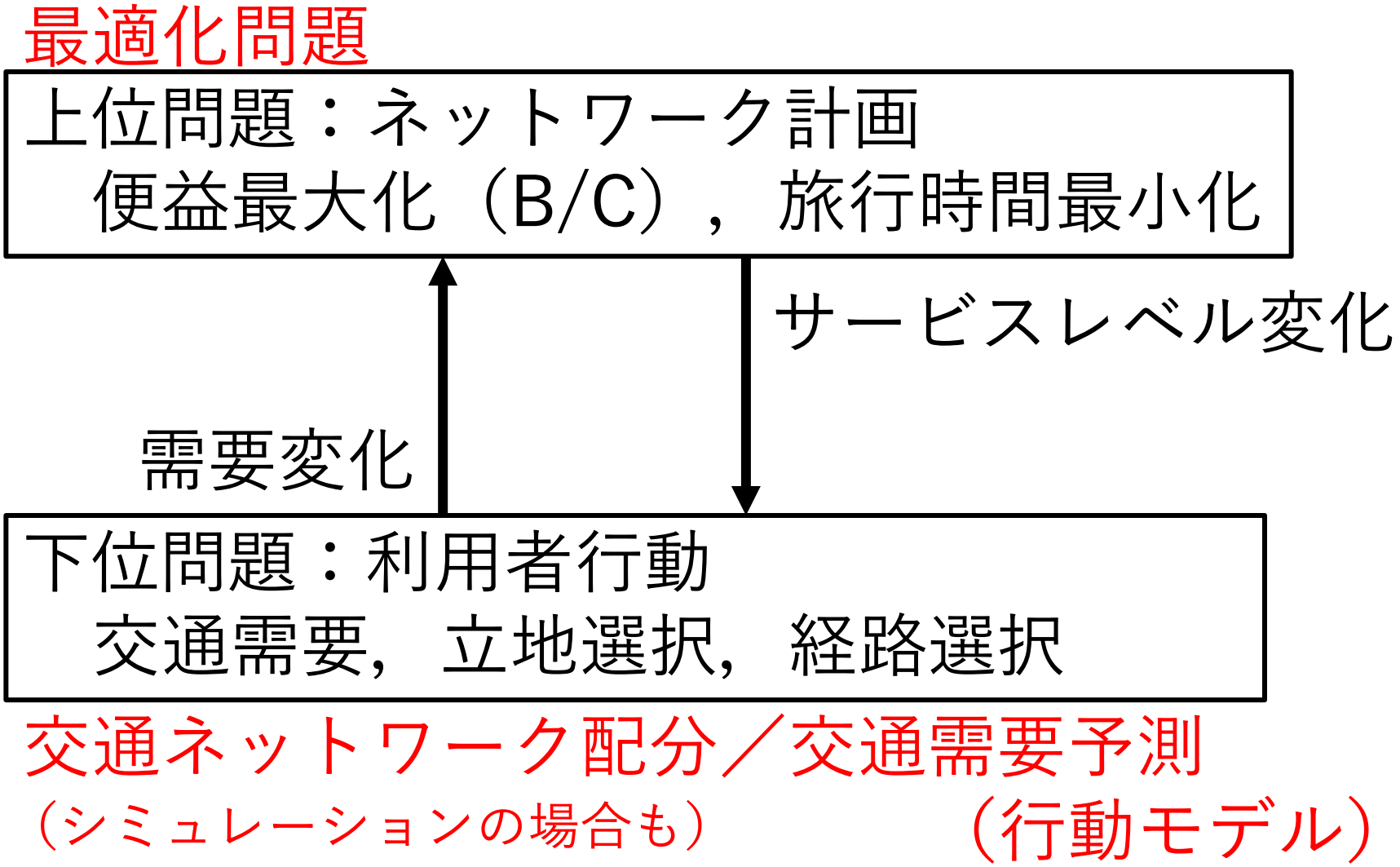
（シミュレーション評価）

交通分野の最適化とは？

$$\min_{x} f(x)$$
$$\text{subject to } g(x) = 0, h(x) > 0$$

	制御変数	目的関数	制約条件
トラック配送	配送経路	旅行時間最小化	配送場所
バス路線	路線選定	収益最大化？ 乗客効用最大化？	車両台数
シェアサイクル	ポート位置	収益最大化	自転車配置数
公共施設	位置配置	受益者利益？ まちなか回遊？	利用可能敷地 予算

最適化と行動モデルの組合せ（二段階最適化）



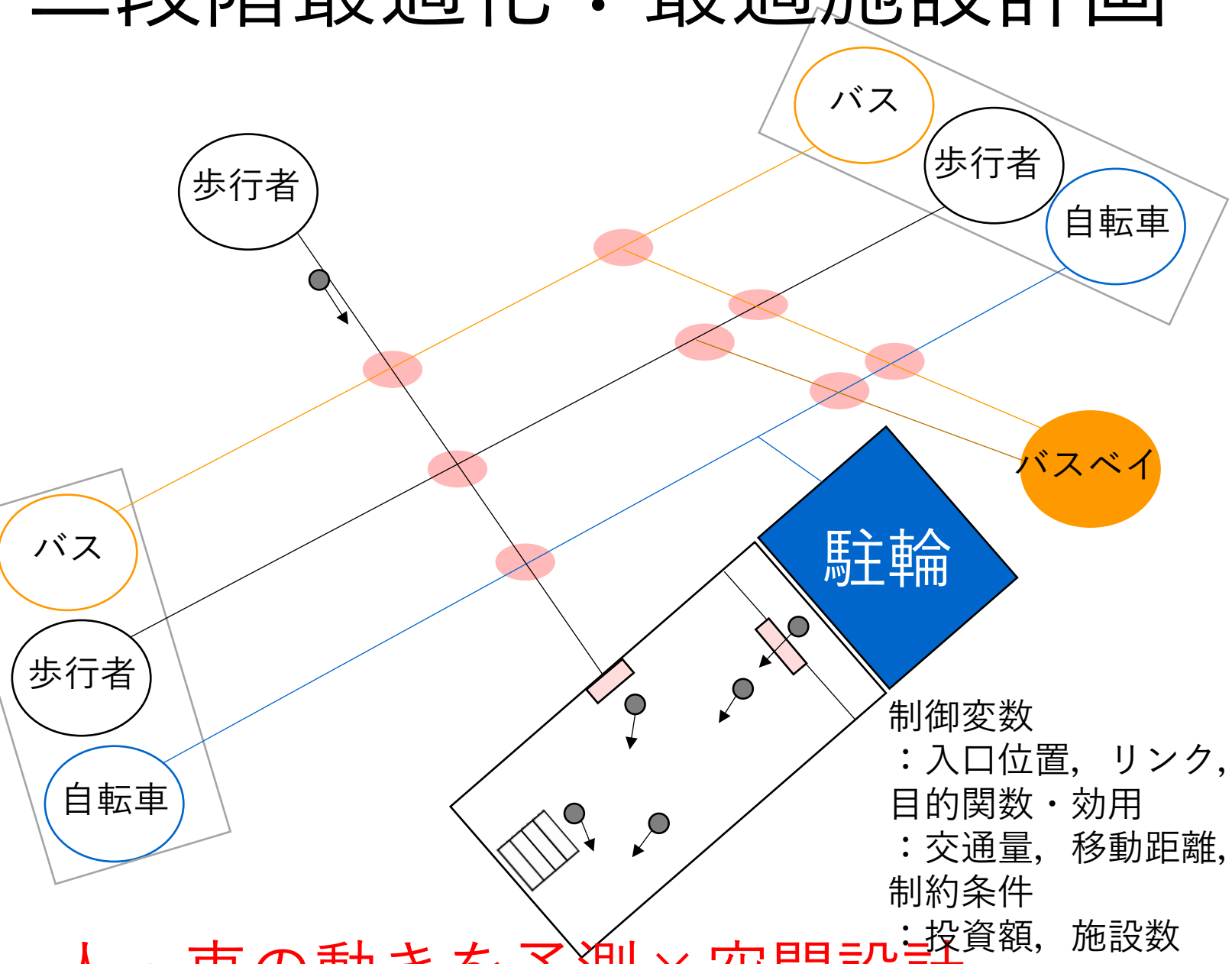
二段階最適化：鉄道整備事業

活動・移動パターン⇔鉄道ネットワーク

企業・住宅の立地パターン⇔鉄道ネットワーク

制御変数
：リンク
目的関数・効用
：交通量，移動時間，地価
制約条件
：投資額，開発余地

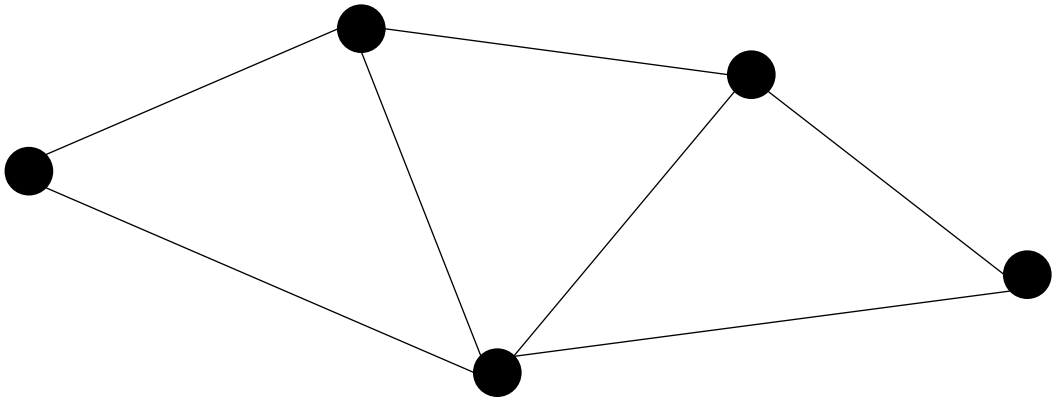
二段階最適化：最適施設計画



制御変数
：入口位置，リンク，施設位置
目的関数・効用
：交通量，移動距離，交差数
制約条件
：投資額，施設数

人・車の動きを予測×空間設計

離散最適化（組合せ最適化）



$x_{ij} = 0$ or 1

リンクの組み合わせ数　$2^{\text{リンク数}}$　→組み合わせ爆発

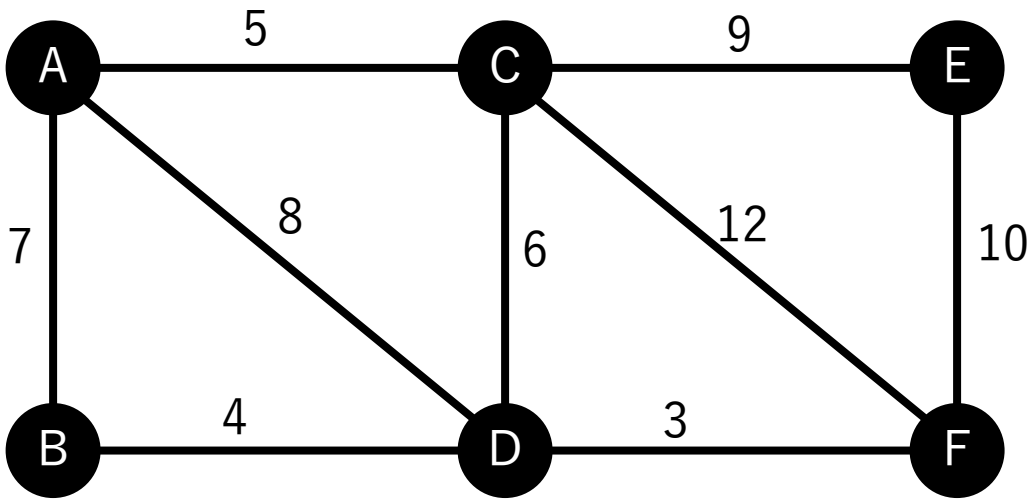
- 厳密解法：分枝限定法，動的計画法
- 近似解法：メタヒューリスティックス

※厳密解にこだわらない場合も⇔問題設定自体が近似

貪欲法：最小木問題

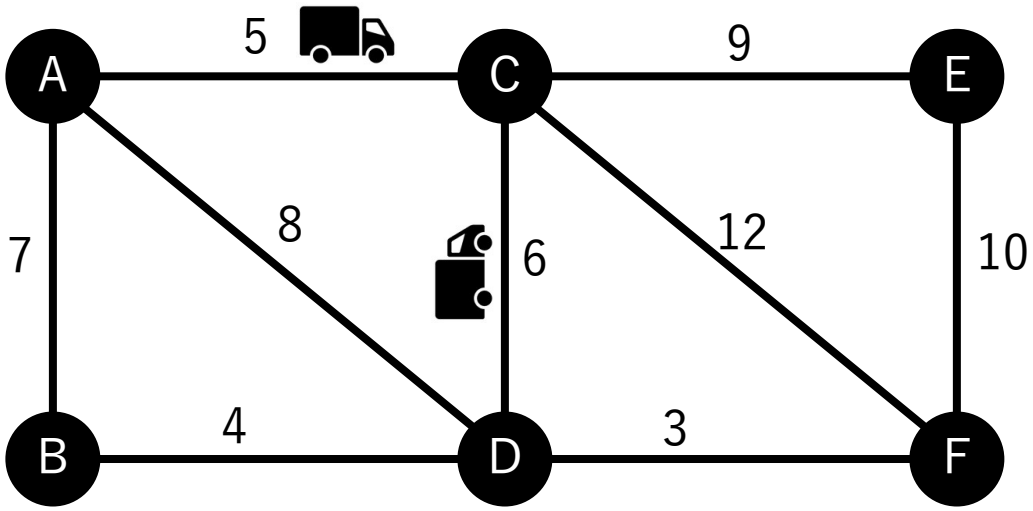
貪欲法(Greedy algorithm)：
　最も効果的なものから追加していく.
　　※問題によって，厳密解を得られる場合と，近似解しか
　　得られない場合とある

最小木問題(Minimum spanning tree)：
　最小となる全域木を求める問題.
　全域木：すべてのノードが連結し，閉路をもたない木



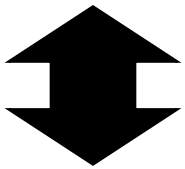
最小木問題→道路ネットワーク計画

6つの都市を，(最小のコストで)繋ぐ自動運転ネットワーク計画



メタヒューリスティクス

厳密解法（Exact Algorithm）
　最適性の保障された解を求める手法



近似解法（Approximate Algorithm）
　ある程度良い解を得られると保証されている手法

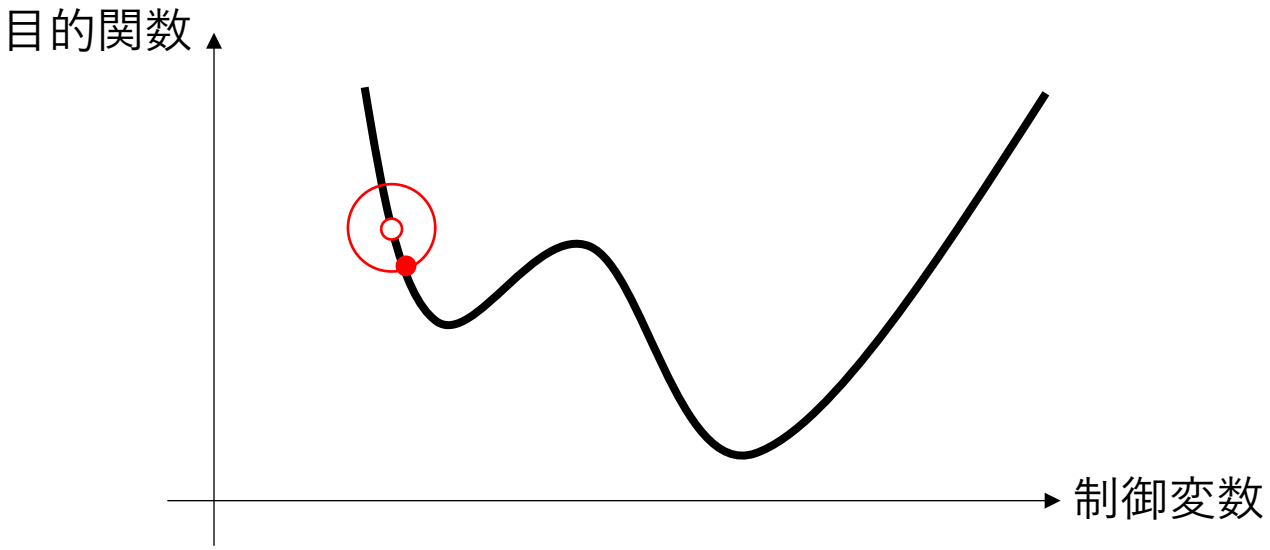
発見的解法（Heuristics）
　求められた解に精度の保証はないが，経験的に近似解が得られるとわかっている方法

汎用的発見的解法（Meta-Heuristics）
　最適化問題の特徴に依存せず，経験的に近似解が得られるとわかっている方法

メタヒューリスティクス

解くための工夫の仕方が色々ある.

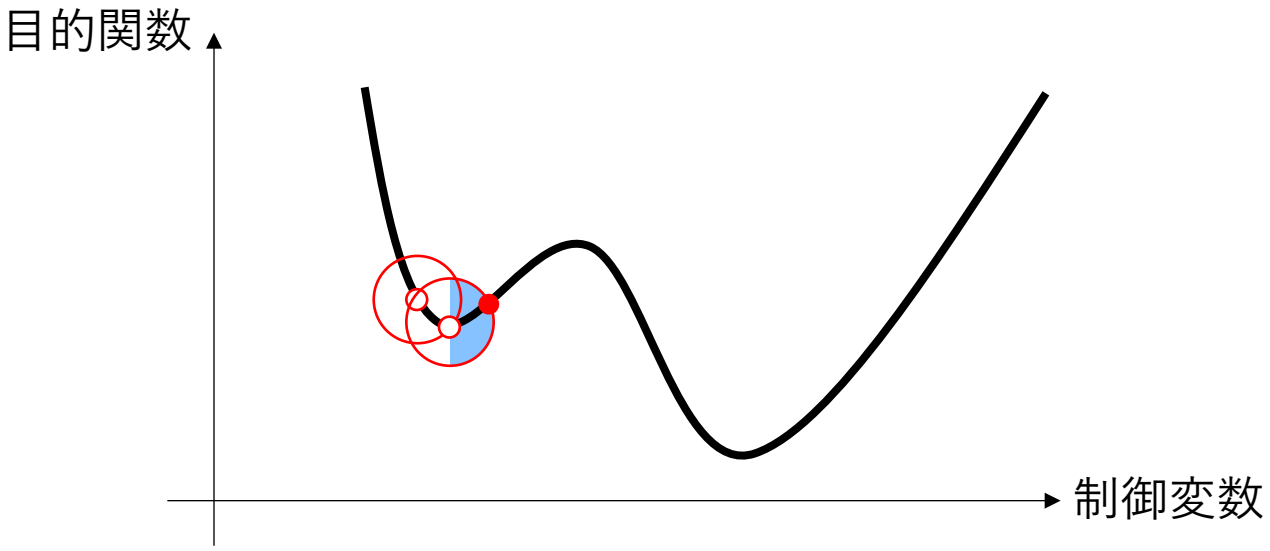
- 近傍の利用：暫定解の近傍を探索する
- 履歴：直前にダメだった解に戻らないよう探索する
- ランダム性：行き詰ったらランダムに解を生成
- 変形：目的関数／制約条件内の重みを変化
- 複数の解：並行して探索する



メタヒューリスティクス

解くための工夫の仕方が色々ある.

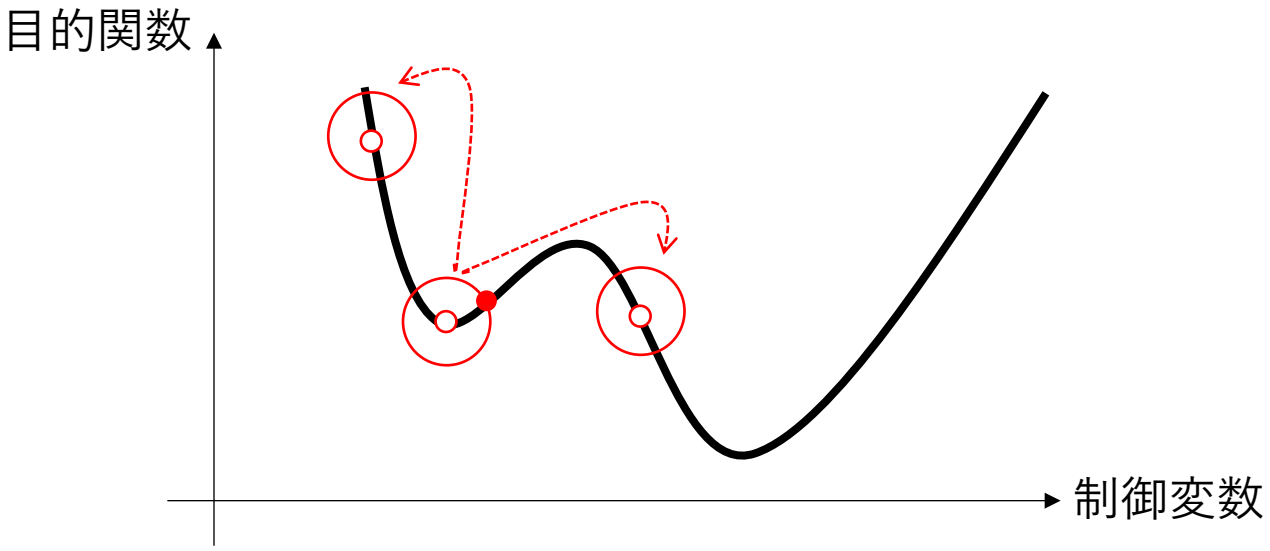
- 近傍の利用：暫定解の近傍を探索する
- 履歴：直前にダメだった解に戻らないよう探索する
- ランダム性：行き詰ったらランダムに解を生成
- 変形：目的関数／制約条件内の重みを変化
- 複数の解：並行して探索する



メタヒューリスティクス

解くための工夫の仕方が色々ある.

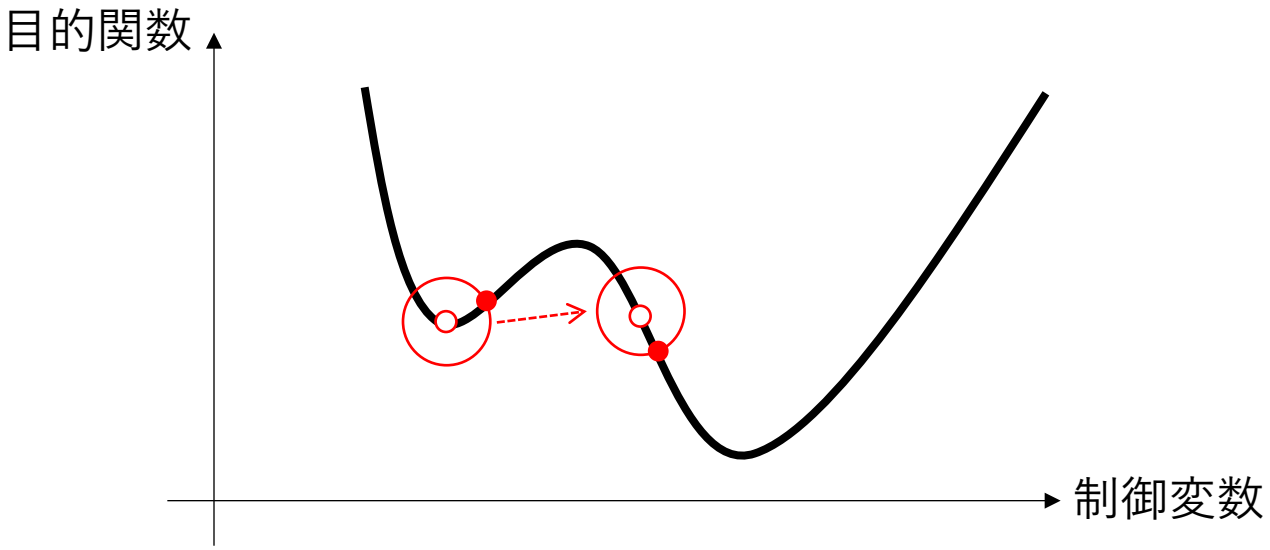
- 近傍の利用：暫定解の近傍を探索する
- 履歴：直前にダメだった解に戻らないよう探索する
- ランダム性：行き詰ったらランダムに解を生成
- 変形：目的関数／制約条件内の重みを変化
- 複数の解：並行して探索する



メタヒューリスティクス

解くための工夫の仕方が色々ある.

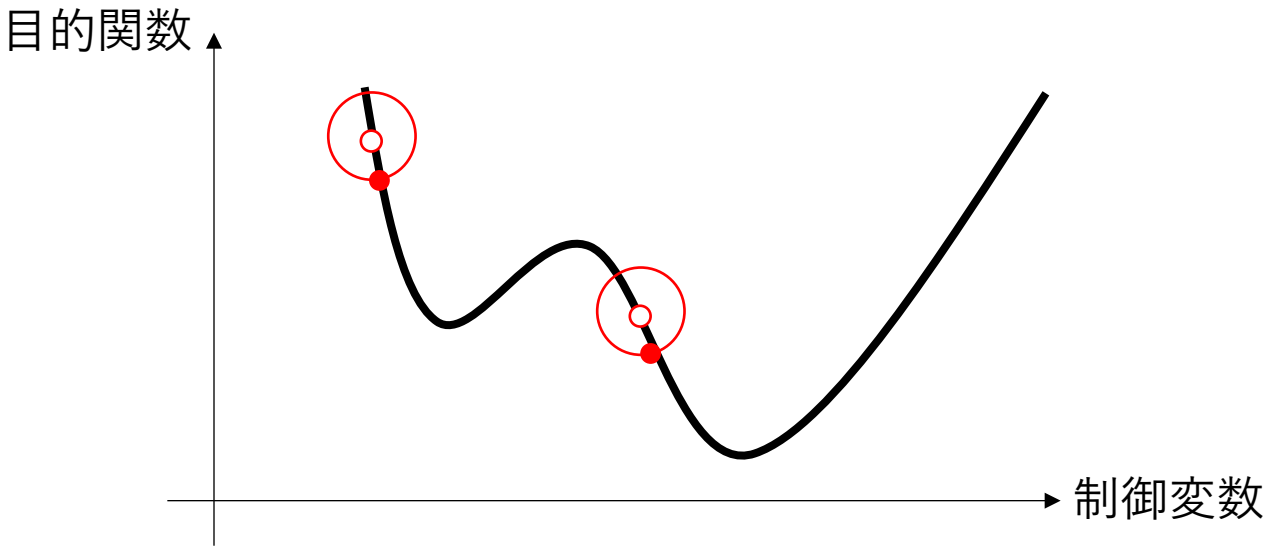
- 近傍の利用：暫定解の近傍を探索する
- 履歴：直前にダメだった解に戻らないよう探索する
- ランダム性：行き詰ったらランダムに解を生成
- 変形：目的関数／制約条件内の重みを変化
- 複数の解：並行して探索する



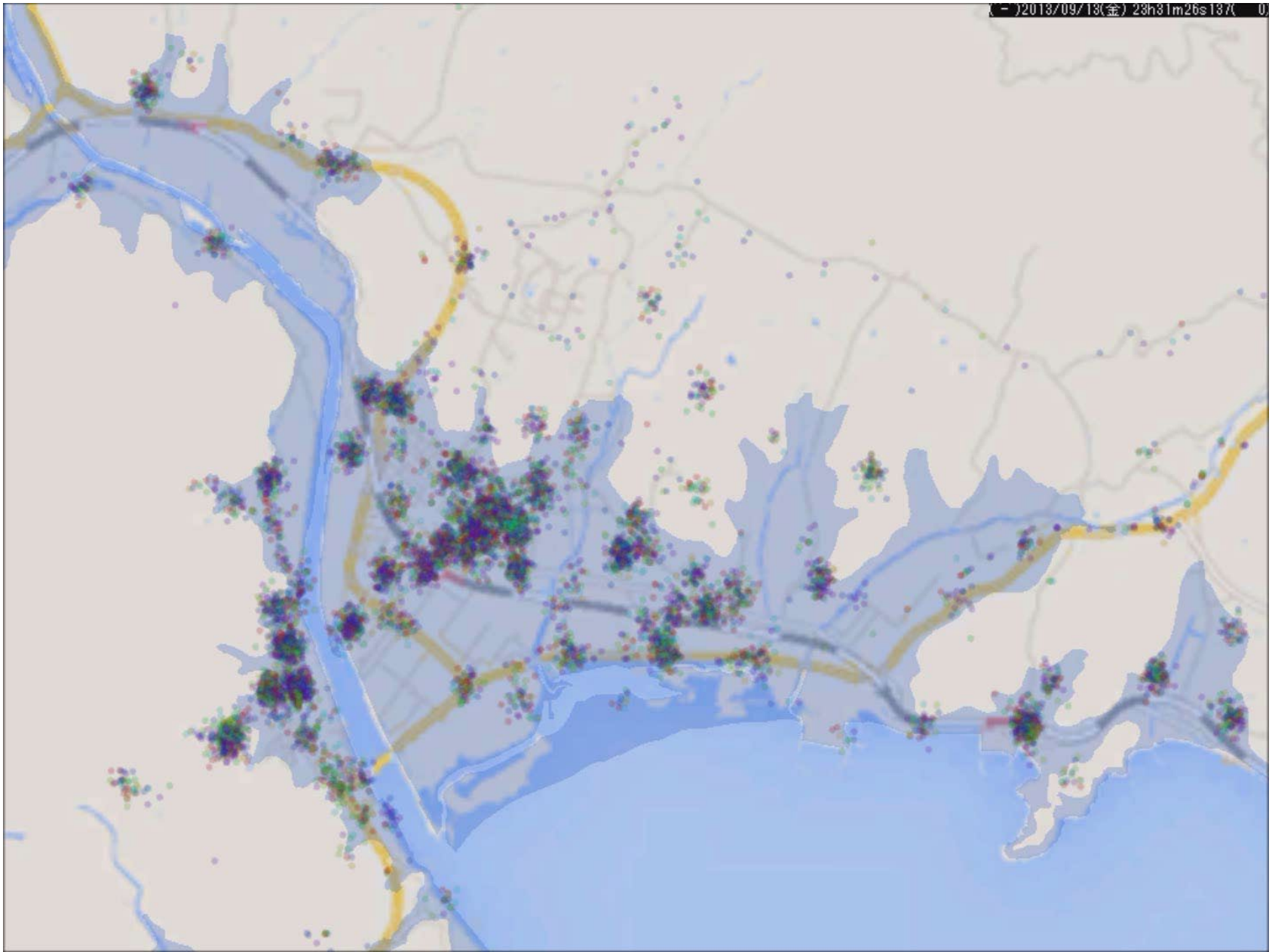
メタヒューリスティクス

解くための工夫の仕方が色々ある.

- 近傍の利用：暫定解の近傍を探索する
- 履歴：直前にダメだった解に戻らないよう探索する
- ランダム性：行き詰ったらランダムに解を生成
- 変形：目的関数／制約条件内の重みを変化
- 複数の解：並行して探索する



目的関数の設定～最適避難計画を例に～



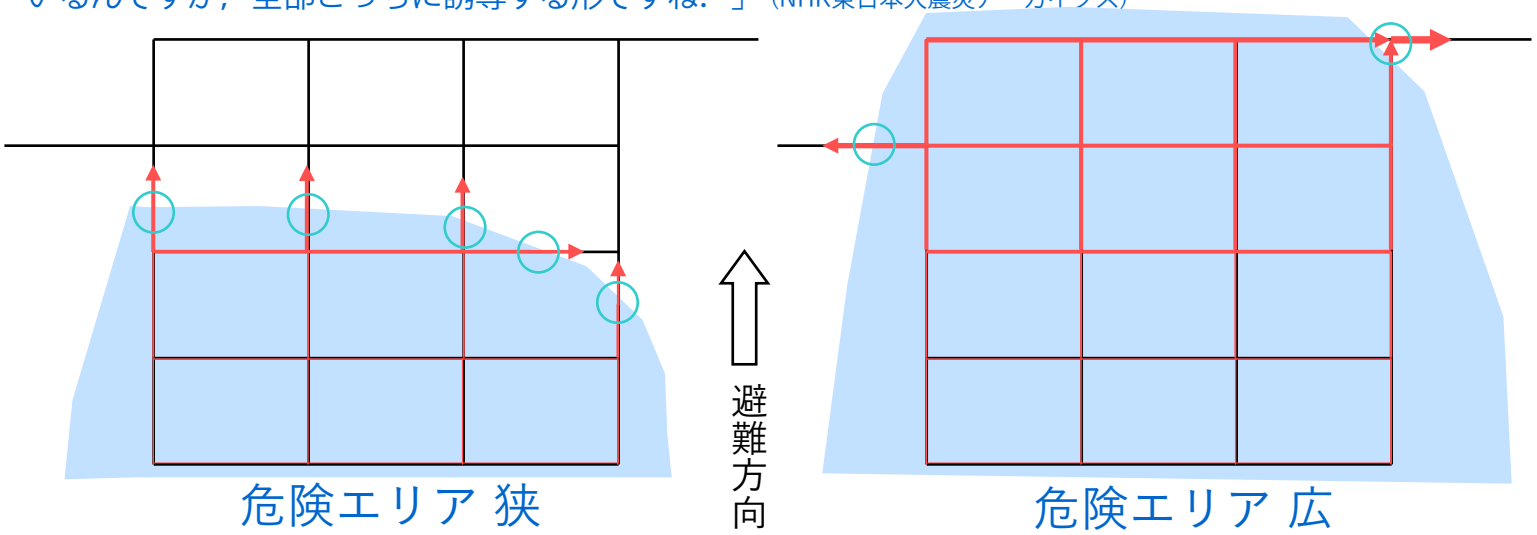
最適避難計画の考え方：流出時混雑

限られた流入路上で渋滞が発生

⇒ ①早期避難のための流出制御，ネットワーク計画

▼「もうここまで渋滞になっているわけさ，皆山に逃げるために～」（陸前高田市インタビュー）

▼「向こうに行く車は，完全に川とか海の方に行く道なので，を止めて，こっちの高田一中に向かうこの道になっているんですが，全部こっちに誘導する形ですね．」（NHK東日本大震災アーカイブス）



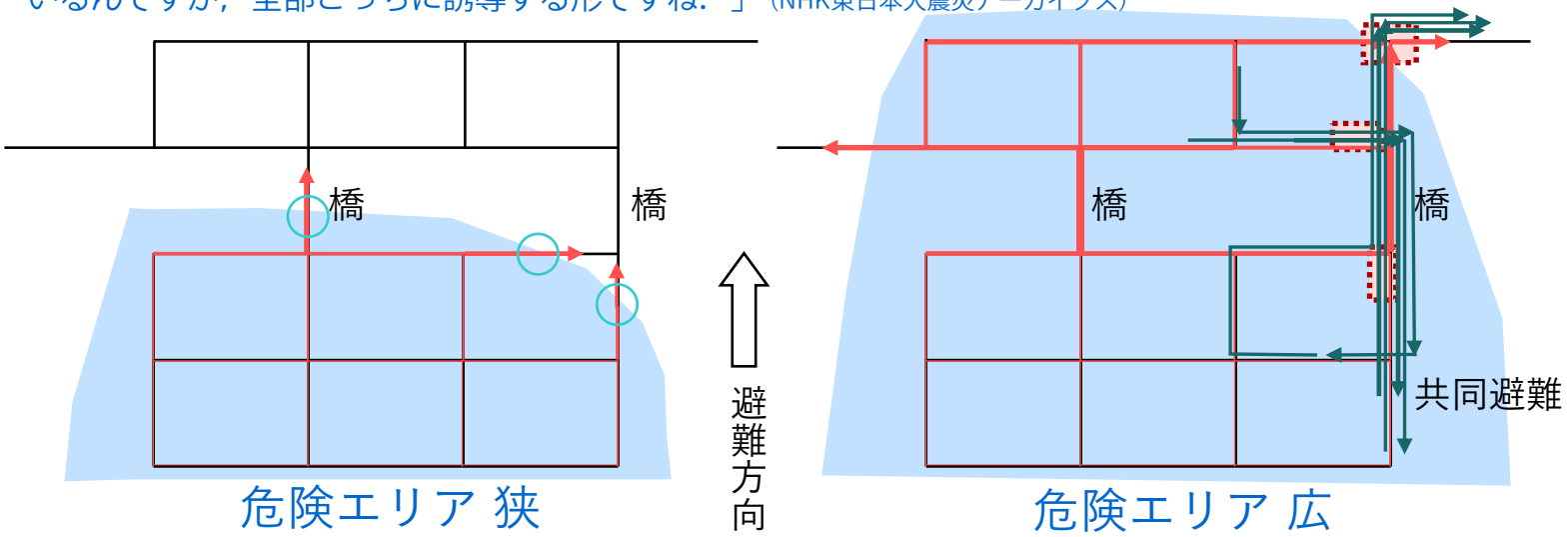
最適避難計画の考え方：共同避難

限られた流入路上で渋滞が発生

⇒ ①早期避難のための流出制御，ネットワーク計画

▼「もうここまで渋滞になっているわけさ，皆山に逃げるために～」（陸前高田市インタビュー）

▼「向こうに行く車は，完全に川とか海の方に行く道なので，を止めて，こっちの高田一中に向かうこの道になっているんですが，全部こっちに誘導する形ですね．」（NHK東日本大震災アーカイブス）



避難開始の遅れ

⇒ ②促進のための共同避難とその動的制御

共同避難が増えれば，避難促進は進むが，道路混雑悪化

▼「同居する両親が気掛かりで，迷わず～自宅に車を走らせた．～．日和大橋を抜ける道は大渋滞で，～．自宅前も渋滞しており，～．両親が出たのを確認後，隣に住むおじいちゃんと一緒に逃げようと思った」（三陸河北新報社インタビュー）

目的関数の色々

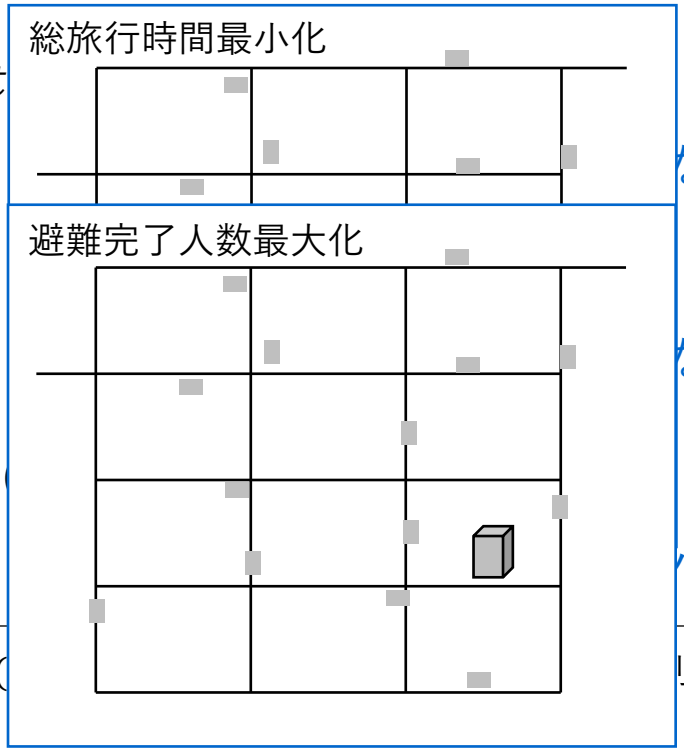
		避難猶予時間短への対応	全員避難の方策検討	時空間リスク分布	計算設定上の課題
総旅行時間最小化	$\min \int_0^T \sum_a x_a(t)\,dt$	×	○	×	–
避難完了人数最大化	$\max \int_0^T \mu_{out}(t$			×	× Tの設定
全員避難完了時刻最小化	$\min T$			×	○ Tの設定なし
総被災リスク最小化	$\min \int_0^T \sum_e R_e($			○	× R_eの設定

全員避難の猶予がある状況で，避難移動の効率化が目標

な人への対策・計画が必要

ないという最大目標と合致

バラつき・推移を反映可能



$x_a(t)$: 時刻t リンクaの交通量　$\mu_{out}($　　刻t ゾーンeのリスク　T: 終端時刻

最適化問題とは

目的関数

$$\min_{x} f(x)$$

制御変数
（離散/連続）

$$\text{subject to } g(x) = 0, h(x) > 0$$

制約条件（等式制約，不等式制約）

https://www.bu.edu/diversity/resource-toolkit/inequity-equality-equity-and-justice/