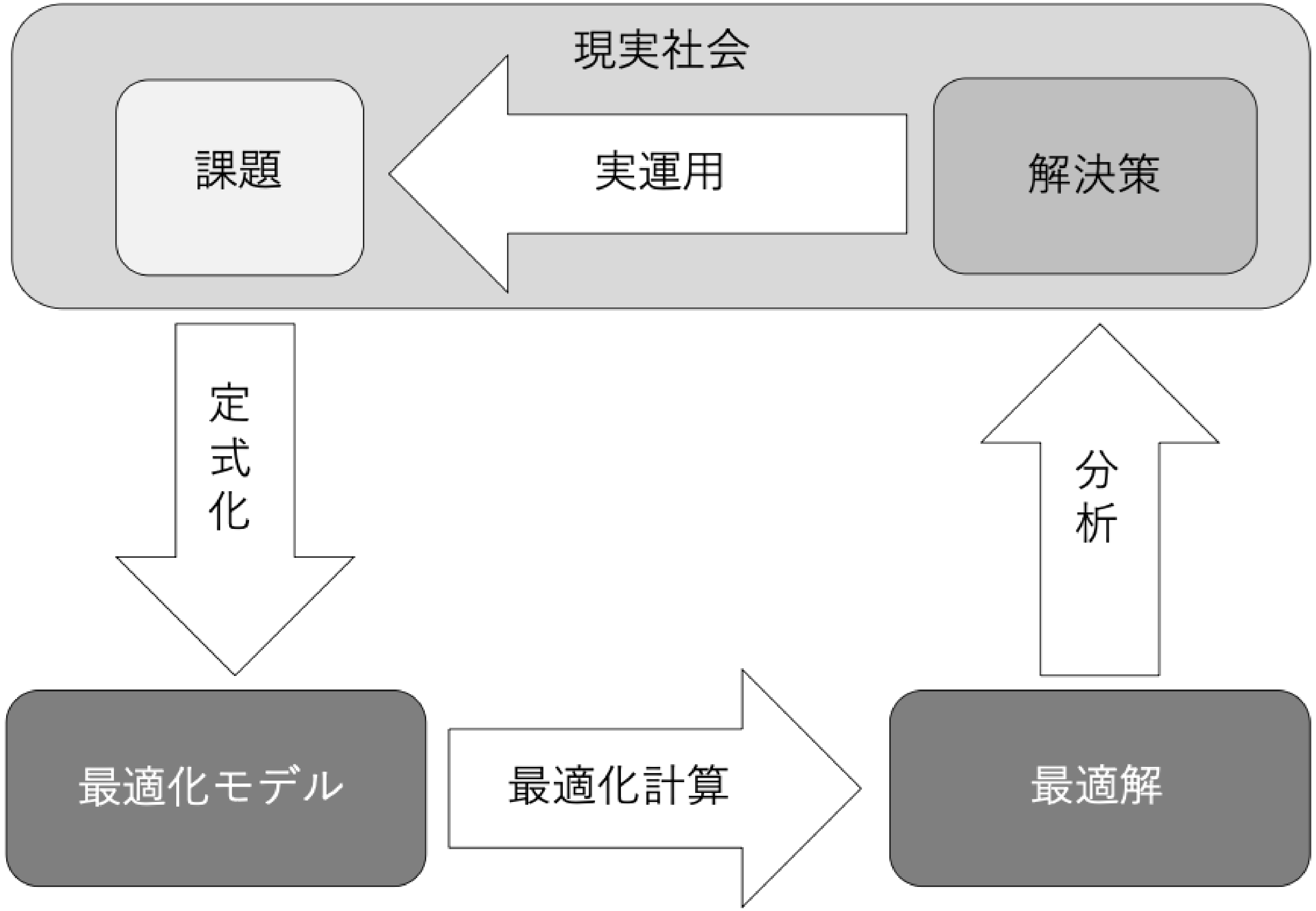
最適化

20260430 桑本球太郎

最適化の基本的な考え方



最適化問題の数学的定式化

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in X \end{array}$$

目的関数 $f(x)$ を最小化する

制約条件のもとで

X は実行可能集合　x は決定係数←選ぶ対象

３つの組み合わせによって数理モデルになる

←何を含めるか（含めないか）が重要

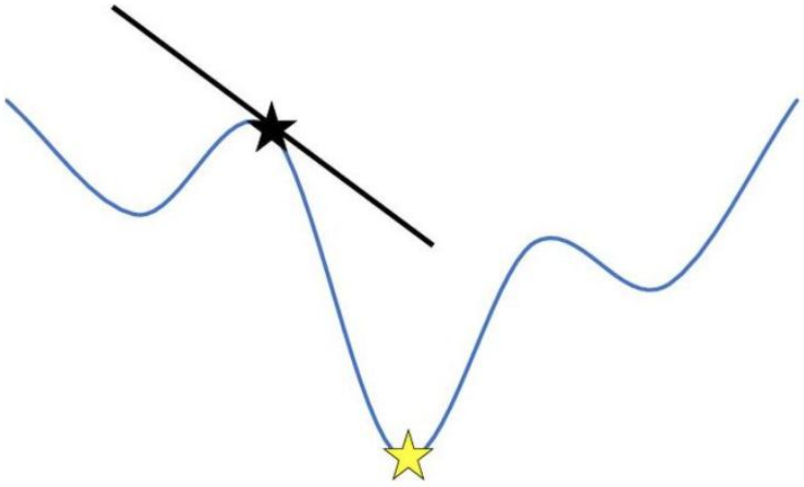
最大化問題と最小化問題

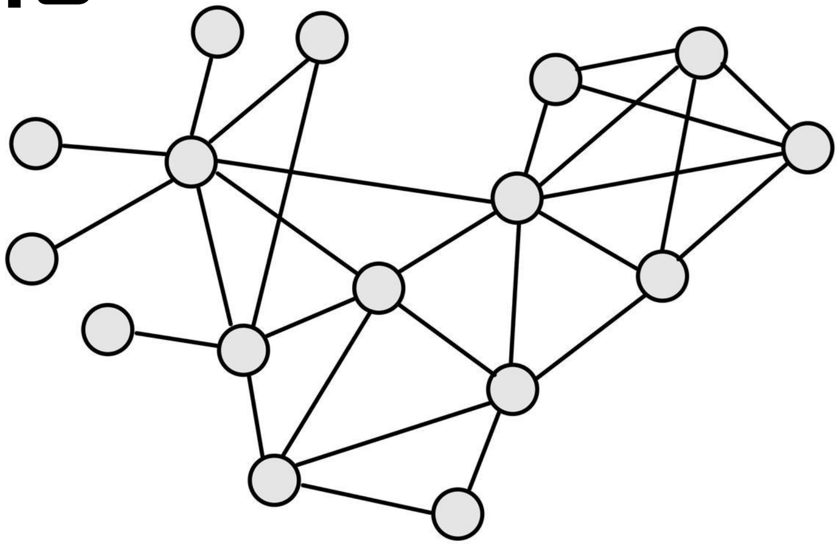
$$\max f(x) \quad \longleftarrow 等価 \longrightarrow \quad \min\{-f(x)\}$$

利益最大化　←等価じゃない→　費用最小化

目的関数が何を意味しているのか、
どのような価値判断を反映しているのかが重要

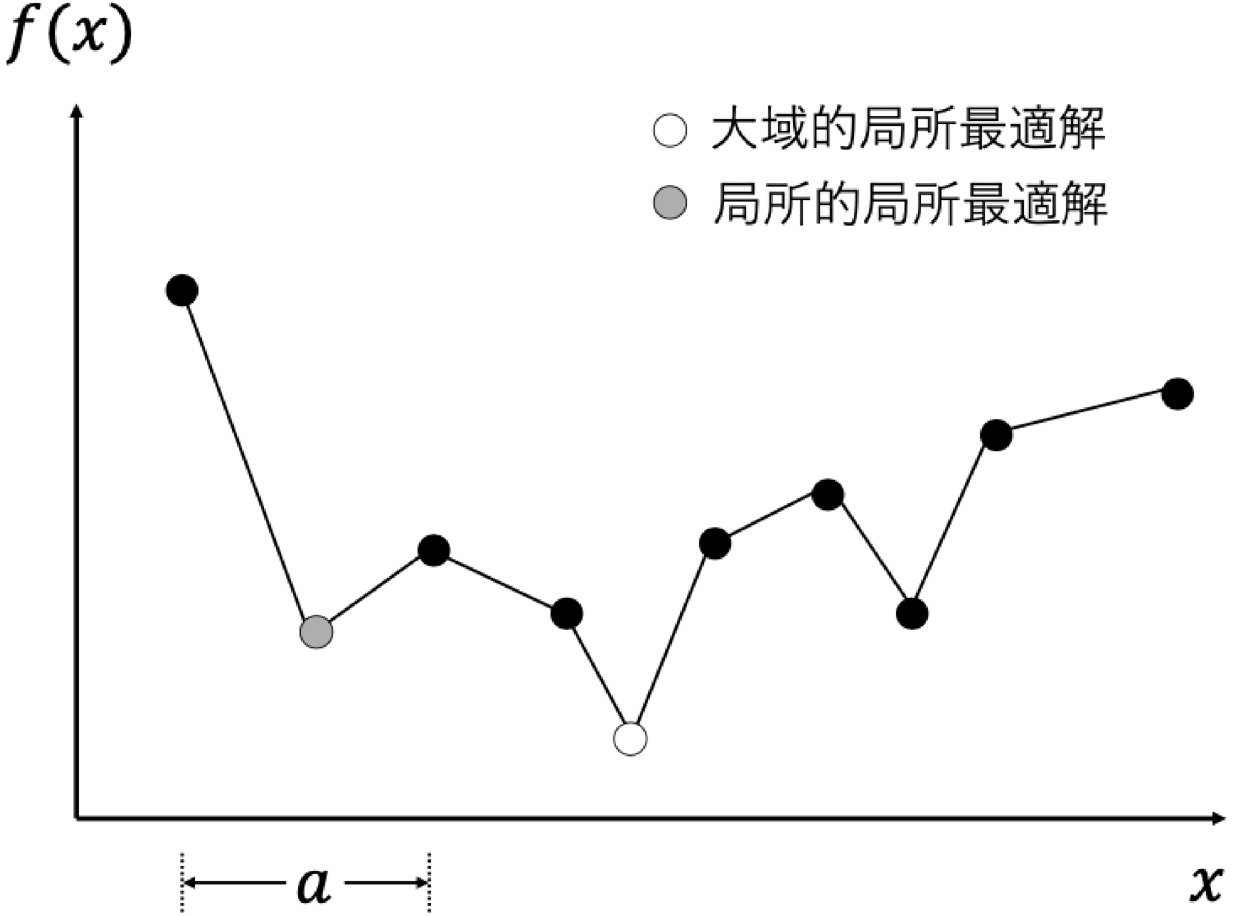
連続最適化と離散最適化





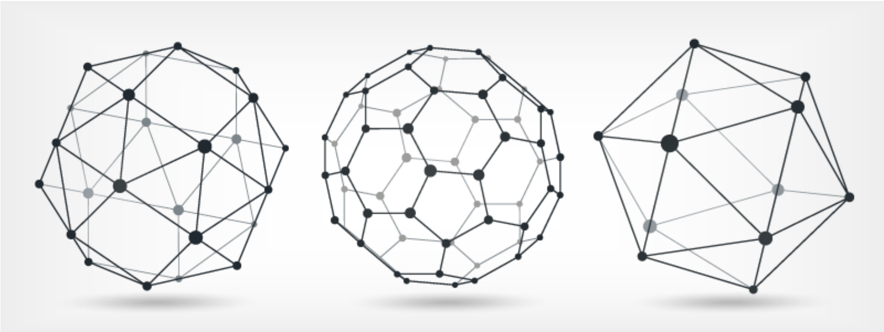
連続最適化	離散最適化
連続した値（量や割合）	離散的な値（選択や順序）
微分や勾配など解析的手法	組み合わせを探索する
滑らかな関数であれば 比較的解きやすい	組合せ爆発

局所最適解と大域的最適解



線形最適化と非線形最適化

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \le b \end{array}$$



実行可能領域が凸多面体
なので扱いやすい

単体法や内点法

$f(x) = x^2 \qquad f(x) = x_1 x_2$

$f(x) = x^4 - x^2$

実行可能領域が単純でなく、
局所と大域で最適が必ずしも
一致しないので扱いにくい

→線形で近似することも

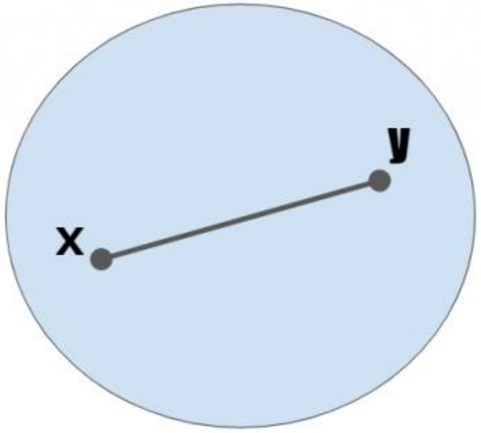
ただし、
凸最適化であれば扱いやすい

凸最適化問題の局所大域最適解

任意の局所最適解は大域最適解でもある

凸集合 $\theta y + (1-\theta)x \in S$

凸関数 $f(\theta y + (1-\theta)x) \leq \theta f(y) + (1-\theta) f(x)$



凸集合

x^* は局所最適（最小）解だが大域最適解ではないとすると

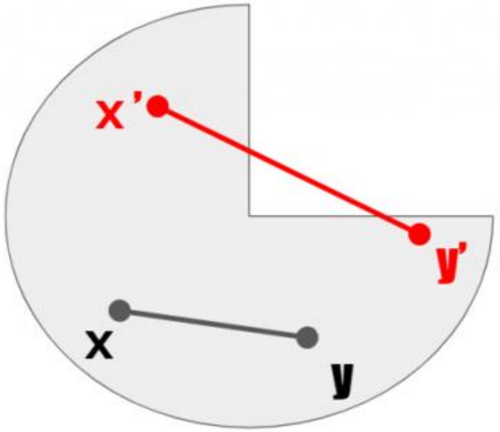
$f(y) < f(x^*)$ となる 点 y が存在する。

凸集合なので、$x_\theta = (1-\theta)x^* + \theta y \quad (0 < \theta \leq 1)$ は集合に属する。

凸関数なので、 $f(x_\theta) \leq (1-\theta) f(x^*) + \theta f(y)$ となる。よって、

$f(x_\theta) < (1-\theta) f(x^*) + \theta f(x^*)$ より $f(x_\theta) < f(x^*)$ となるが、

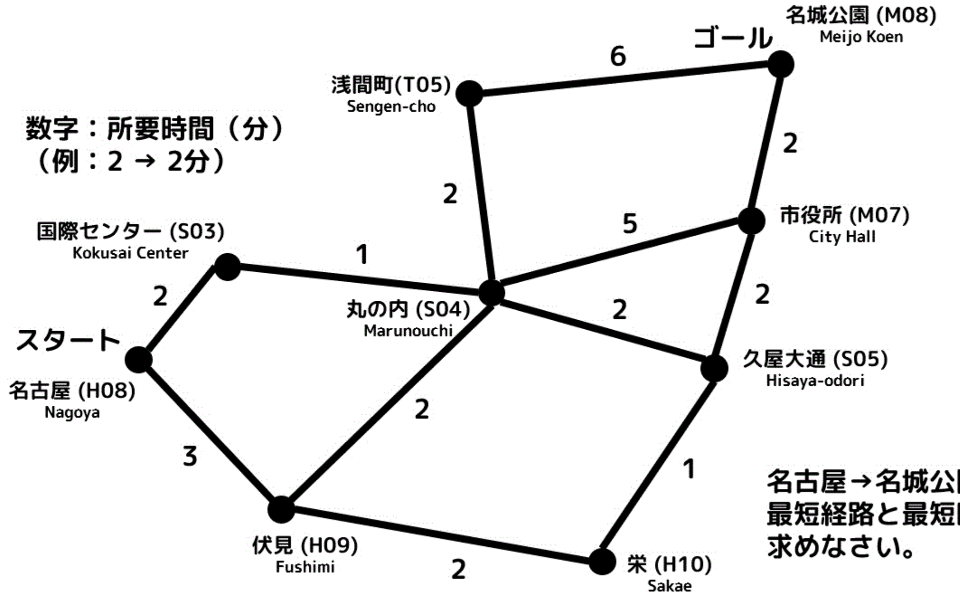
これがθゼロに限りなく近い正の値にしても成り立つので、
x^* が局所最適解であることに矛盾してしまう。



非凸集合

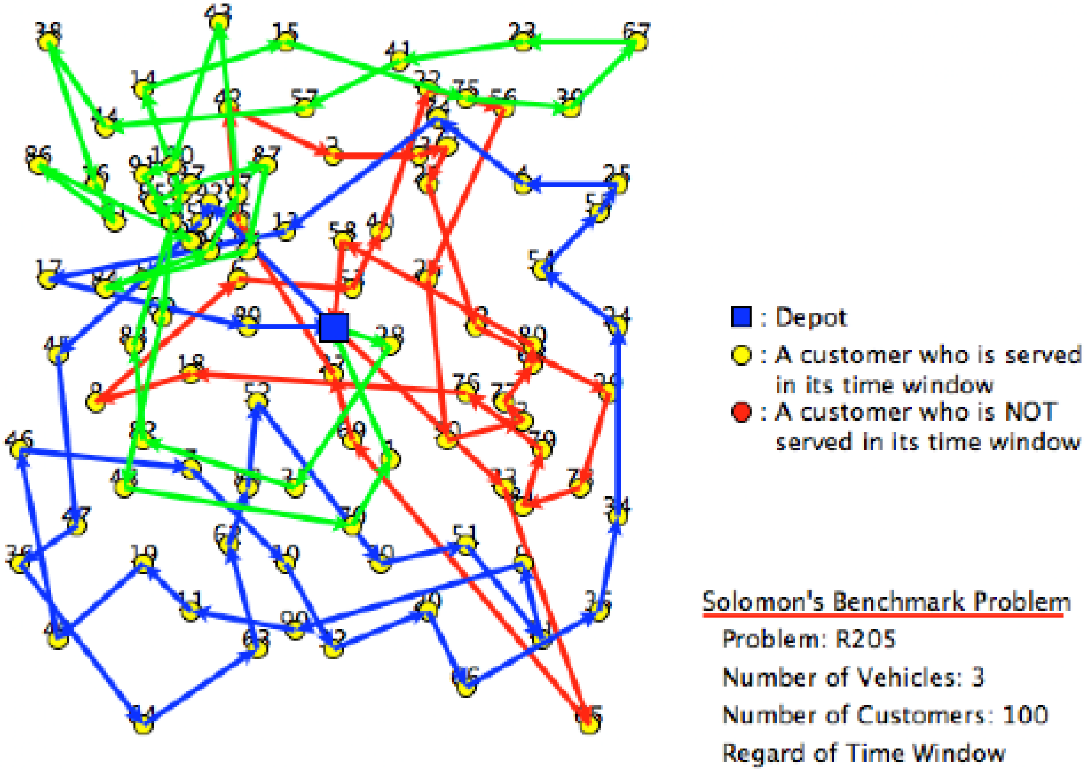
都市交通における最適化の例

最短経路問題



名古屋→名城公園の
最短経路と最短時間を
求めなさい。

配送計画問題(VRP)



線形計画法とは

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

$x \in \mathbb{R}^n$ 決定変数ベクトル

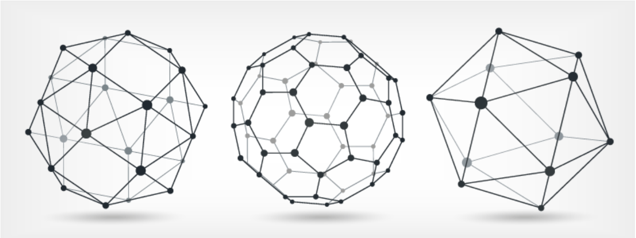
$c \in \mathbb{R}^n$ 目的関数係数ベクトル

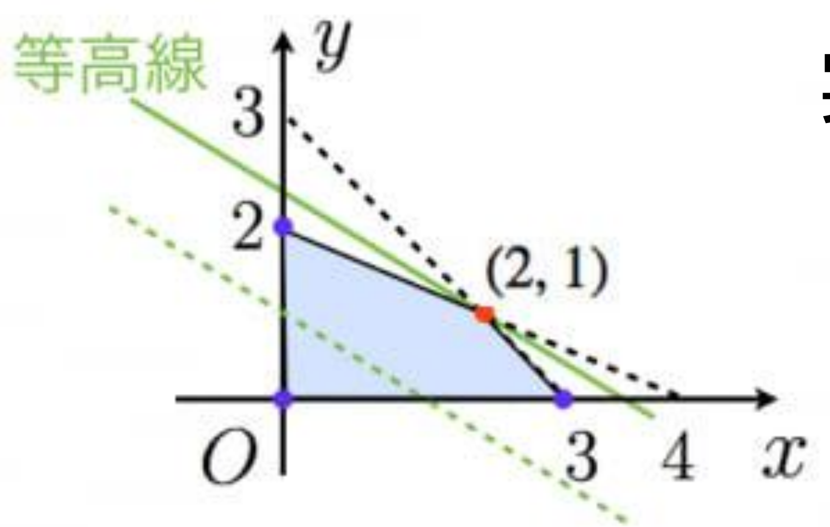
$A \in \mathbb{R}^{m \times n}$ 制約条件行列

$b \in \mathbb{R}^m$ 右辺ベクトル

線形関数(変数の一時結合で表される関数）と
一次不等式で目的関数と制約条件が表される。

線形計画問題の幾何学的理解





実行可能領域の凸多面体に

目的関数の超平面を押し当てるイメージ

→最適解は必ず頂点

(か、ある線分（面）全体)になる！

そもそも頂点の定義は異なる二点の凸結合で作れないことであった

$$x = ty + (1-t)z \implies y = z = x$$

連続無限個の候補が有限個の頂点に還元される！

→頂点探索型のアルゴリズム

線形計画法の標準形

標準形

$$\text{minimize} \quad c^T x$$

$$\text{subject to} \quad Ax = b$$

$$x \geq 0$$

不等式制約は等式制約に直す

$2x_1 + 3x_2 \leq 10 \qquad x_1 + x_2 \geq 5$

スラック変数 s サープラス変数 t を導入する。

$2x_1 + 3x_2 + s = 10 \quad (s \geq 0)$

$x_1 + x_2 - t = 5 \quad (t \geq 0)$

符号制約のない変数も０以上に直す

$x^+, x^- \geq 0$ を用いて $x = x^+ - x^-$ と表す。

標準形にすると、、

コンピュータが掃き出し法などの行列計算で解ける

ラグランジュの未定乗数法を適用しやすくなり、
双対問題の理論を利用できる

線形計画法の実行可能基底解

標準形

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

基底解

連立方程式 $Ax = b$

の非基底変数を０として解いた解

→実行可能領域の外側にもあり得る

実行可能基底解

基底解のうち、$x \geq 0$ を満たすもの

実行可能基底解は、幾何学的には実行可能領域の頂点に対応している

双対問題

ある工場の経営と買収を考える。

製品Aは1個3万円、製品Bは1個4万円 の利益が出る。
製品Aは1個で木材1mと労働2時間、製品Bは1個に、木材2mと 労働1時間が必要。
工場で使える資源の上限は、木材8m、労働7時間である。

経営：利益の最大化

$$\max \quad z = 3x_1 + 4x_2 \quad \text{(総利益)}$$

$$\text{s.t.} \quad x_1 + 2x_2 \leq 8 \quad \text{(木材の制約)}$$

$$2x_1 + x_2 \leq 7 \quad \text{(労働時間の制約)}$$

$$x_1, x_2 \geq 0 \quad \text{(生産量は非負)}$$

$$z = 18 \text{（万円）}$$

買収：価格の最小化

$$\min \quad w = 8y_1 + 7y_2 \quad \text{(総買収コスト)}$$

$$\text{s.t.} \quad y_1 + 2y_2 \geq 3 \quad \text{(製品Aを作る利益以上の価値を保証)}$$

$$2y_1 + y_2 \geq 4 \quad \text{(製品Bを作る利益以上の価値を保証)}$$

$$y_1, y_2 \geq 0 \quad \text{(価格は非負)}$$

$$w = 18 \text{（万円）}$$

そもそも同じ工場について考えているので2つは同じはず！

ラグランジュ緩和と双対問題

主問題

$$\max \quad z = 3x + 4y$$

$$\text{s.t.} \quad 3x + 2y \le 200$$

$$x + 2y \le 100$$

$$x, y \ge 0$$

双対問題

$$\min \quad w = 200v + 100w$$

$$\text{s.t.} \quad 3v + w \ge 3$$

$$2v + 2w \ge 4$$

$$v, w \ge 0$$

主問題の上限値を考える

$v \ge 0, w \ge 0$ として以下の式を考える

$$L(x,y,v,w) = \underbrace{3x + 4y}_{\text{元の利益}} + \underbrace{v(200 - 3x - 2y) + w(100 - x - 2y)}_{\text{非負の追加項}}$$

↓整理し直すと

$$L(x,y,v,w) = 200v + 100w + x(3 - 3v - w) + y(4 - 2v - 2w)$$

上限値をできるだけ小さくすることを考える

$$3 - 3v - w \le 0 \quad \Rightarrow \quad 3v + w \ge 3$$

$$4 - 2v - 2w \le 0 \quad \Rightarrow \quad 2v + 2w \ge 4$$

主問題と双対問題

主問題	双対問題
$\max \mathbf{c}^T\mathbf{x}$	$\min \mathbf{b}^T\mathbf{y}$
s.t. $A\mathbf{x} \leq \mathbf{b}$	s.t. $A^T\mathbf{y} \geq \mathbf{c}$
$\mathbf{x} \geq \mathbf{0}$	$\mathbf{y} \geq \mathbf{0}$

制約と変数を入れ替えられる
→計算量削減や双対シンプレックス法などに活きる

弱双対定理 $\mathbf{c}^T\mathbf{x} \leq \mathbf{b}^T\mathbf{y}$

$$\mathbf{c}^T\mathbf{x} \leq (A^T\mathbf{y})^T\mathbf{x} = \mathbf{y}^T A\mathbf{x}$$
$$\mathbf{y}^T(A\mathbf{x}) \leq \mathbf{y}^T\mathbf{b}$$
$$\mathbf{c}^T\mathbf{x} \leq \mathbf{y}^T A\mathbf{x} \leq \mathbf{b}^T\mathbf{y}$$

強双対定理 $\mathbf{c}^T\mathbf{x}^* = \mathbf{b}^T\mathbf{y}^*$

相補性条件（相補スラックネス定理）

$$x_j^* \times (A_j^T\mathbf{y}^* - \mathbf{c}_j) = 0$$

$$y_i^* \times (\mathbf{b}_i - A_i\mathbf{x}^*) = 0$$

KKT条件

$$
\begin{aligned}
\text{minimize} \quad & f(x) \\
\text{subject to} \quad & g_i(x) \le 0 \quad (i = 1, \ldots, m) \\
& h_j(x) = 0 \quad (j = 1, \ldots, p)
\end{aligned}
$$

※特定条件とは、有効な等式制約の勾配ベクトルがすべて一次独立であること。

この時、最適解 x^* は、特定条件のもとで以下を満たす

停留条件 $\nabla f(x^*) + \sum^{m} \lambda_i \nabla g_i(x^*) + \sum^{p} \mu_j \nabla h_j(x^*) = 0$

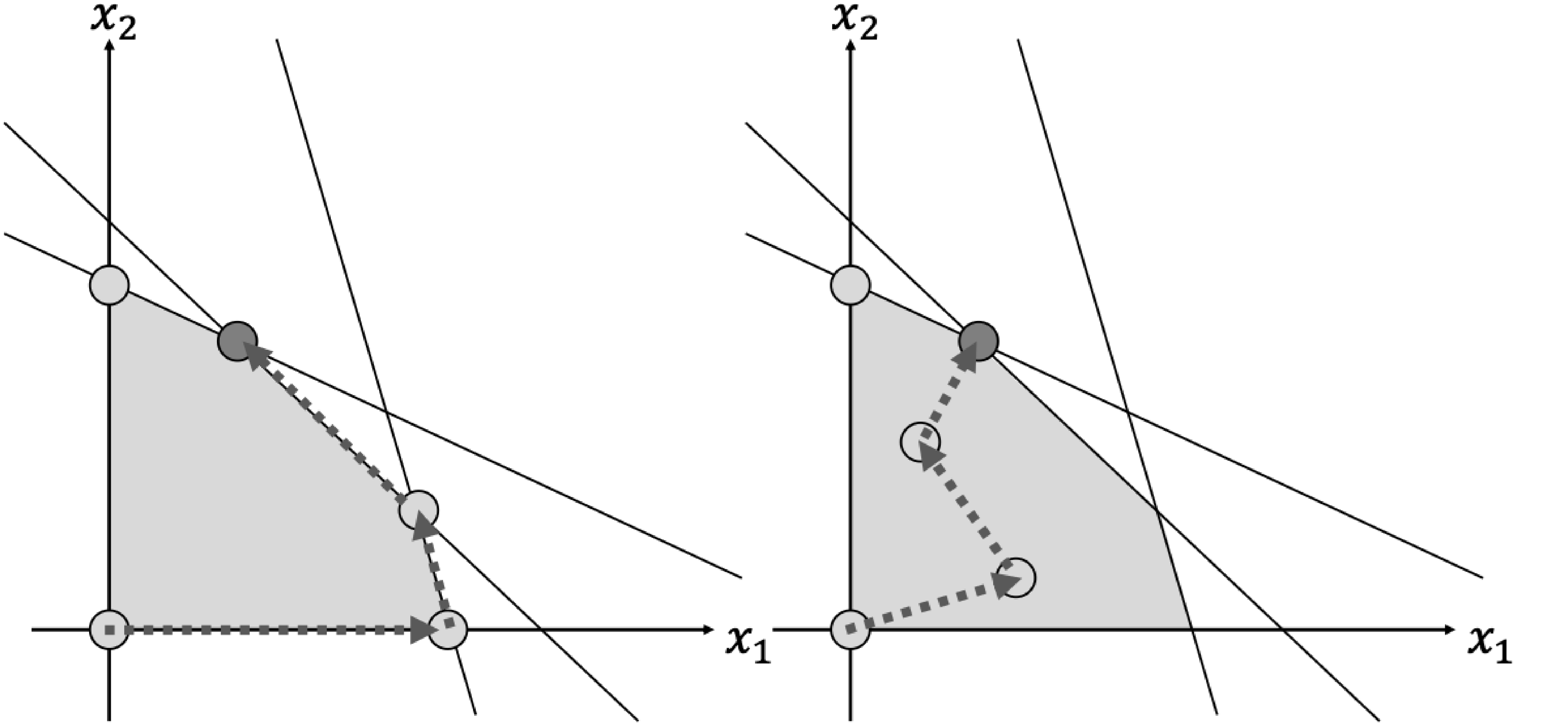
実行可能性 $g_i(x^*) \le 0, \quad h_j(x^*) = 0$

双対実行可能性 $\lambda_i \ge 0$

相補性条件 $\lambda_i \, g_i(x^*) = 0$

凸最適化であれば、最適解の必要十分条件になる！

代表的な解法



(a) 単体法　　(b) 内点法

単体法と内点法

$$\text{Minimize} \quad z = \boldsymbol{c}^T \boldsymbol{x} = \sum_{j=1}^{3000} c_j x_j \qquad \text{Subject to} \quad A\boldsymbol{x} \leq \boldsymbol{b} \qquad \boldsymbol{x} \geq \boldsymbol{0}$$

```
問題規模: 変数 3000 個 / 制約 1000 個
----------------------------------------
【単体法 (Simplex)】: 凸多面体の縁(頂点)を伝って探索
  計算時間      : 11.8777 秒
  反復回数      : 449 回 (頂点から頂点への移動回数)
  最適値        : 1.05

【内点法 (Interior Point)】: 空間の内部を突き抜けて探索
  計算時間      : 29.9228 秒
  反復回数      : 39 回 (中心軌道の更新回数)
  最適値        : 1.05
```

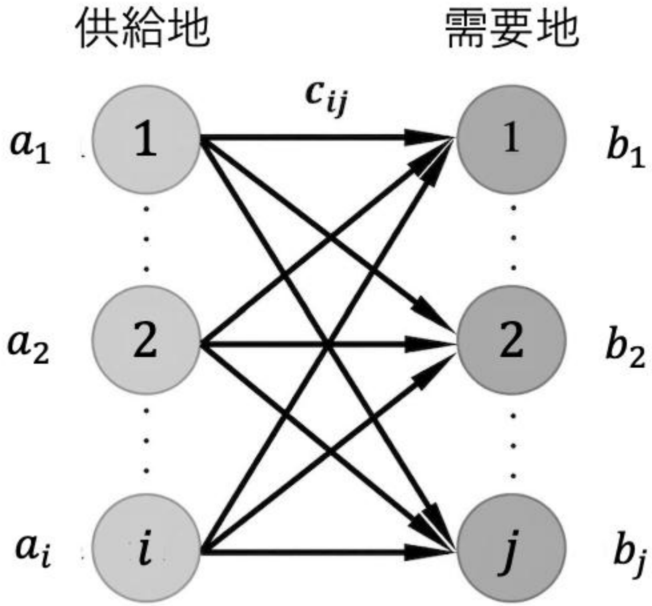
```
問題規模: 変数 3000 個 / 制約 1000 個 (密度 1%)
------------------------------------------
【単体法 (Simplex)】
  計算時間      : 3.1765 秒
  反復回数      : 4447 回


【内点法 (Interior Point)】
  計算時間      : 0.4535 秒
  反復回数      : 21 回
```

スパース性があれば内点法の方が圧倒的に速い
条件が少しだけ変わった時などは単体法の方が速い

線形計画法の例

輸送問題



$$\text{minimize} \quad \sum_i \sum_j c_{ij} x_{ij}$$

$$\text{subject to} \quad \sum_j x_{ij} = a_i \quad (all i)$$

$$\sum_i x_{ij} = b_j \quad (all j)$$

$$x_{ij} \geq 0$$

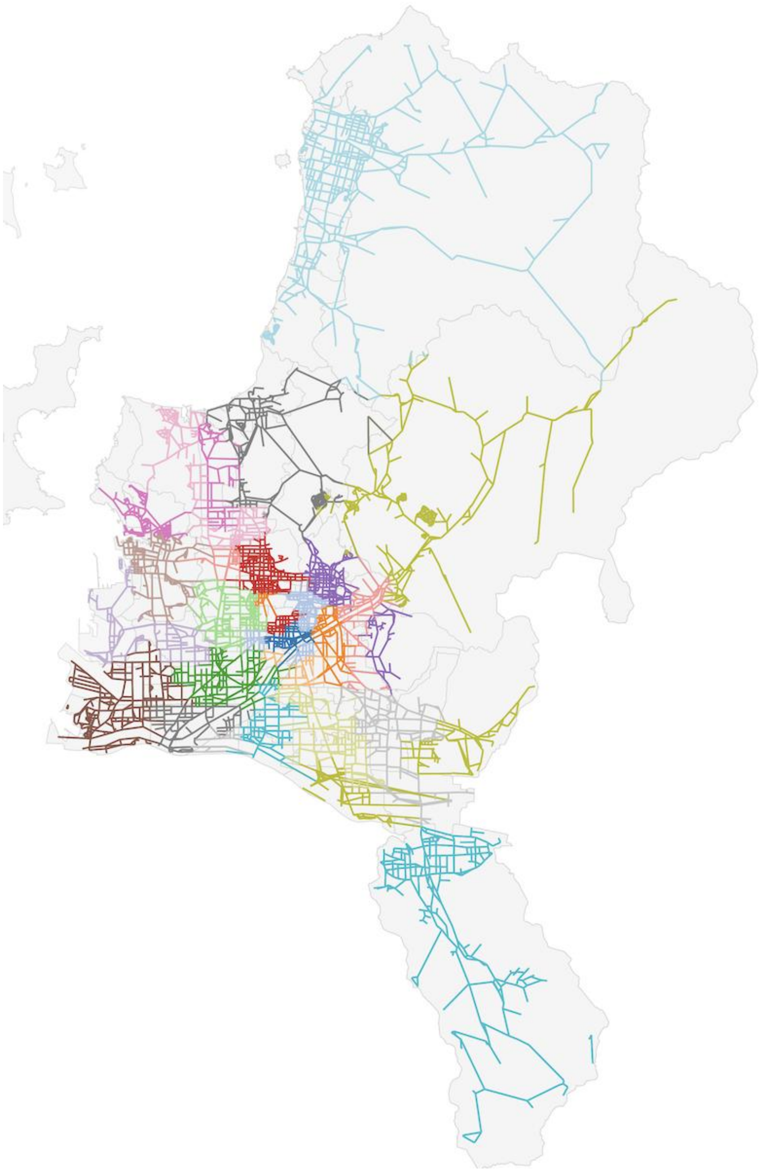
課題

社会便益を最大化するバスの最適ルートを探索する

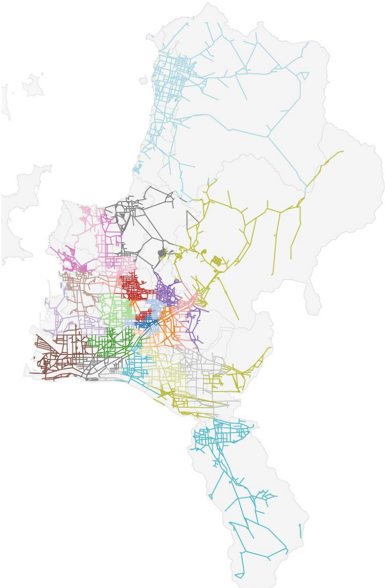
全ノードを対象とした離散最適化は
組み合わせが膨大

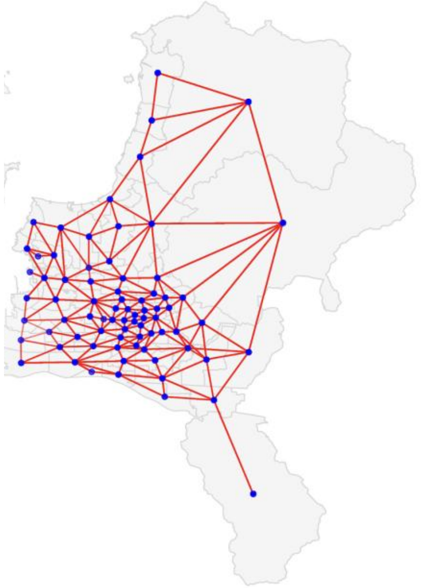
階層化: マクロ（ゾーン）で解き、
ミクロに射影する

メタヒューリスティクス: 遺伝的
アルゴリズム（GA）の活用



用いたデータ



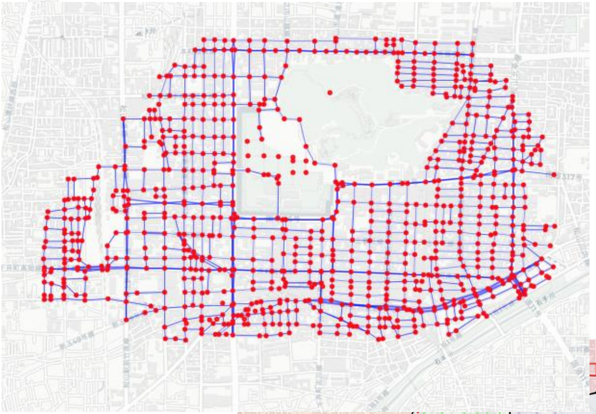


2007年松山都市圏パーソントリップ調査の個票データ

「出発地出発時刻時」が朝のピーク時（7時, 8時, 9時）のトリップに限定

「代表交通利用交通手段大分類 = 3（自動車）」のみを抽出

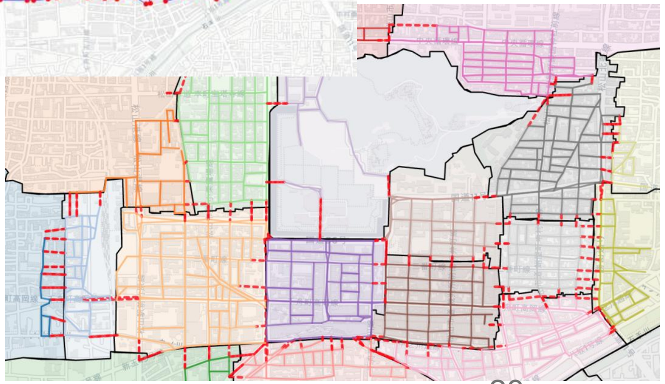
$$q_{rs} = \sum_{i \in \mathrm{trips}(r,s)} w_i$$



道路ネットワーク2種類

松山市全体のものと中心市街地のもの２種類

前者については、レーン数＝1に固定、すべて車走行可とした。
後者については、自動車通行可能なリンク（car == True）のみを計算対象（2140リンク）として抽出、車線数（lanes）データに欠損がある場合は、1車線として補完して計算。



マクロネットワークの構築と容量の定義

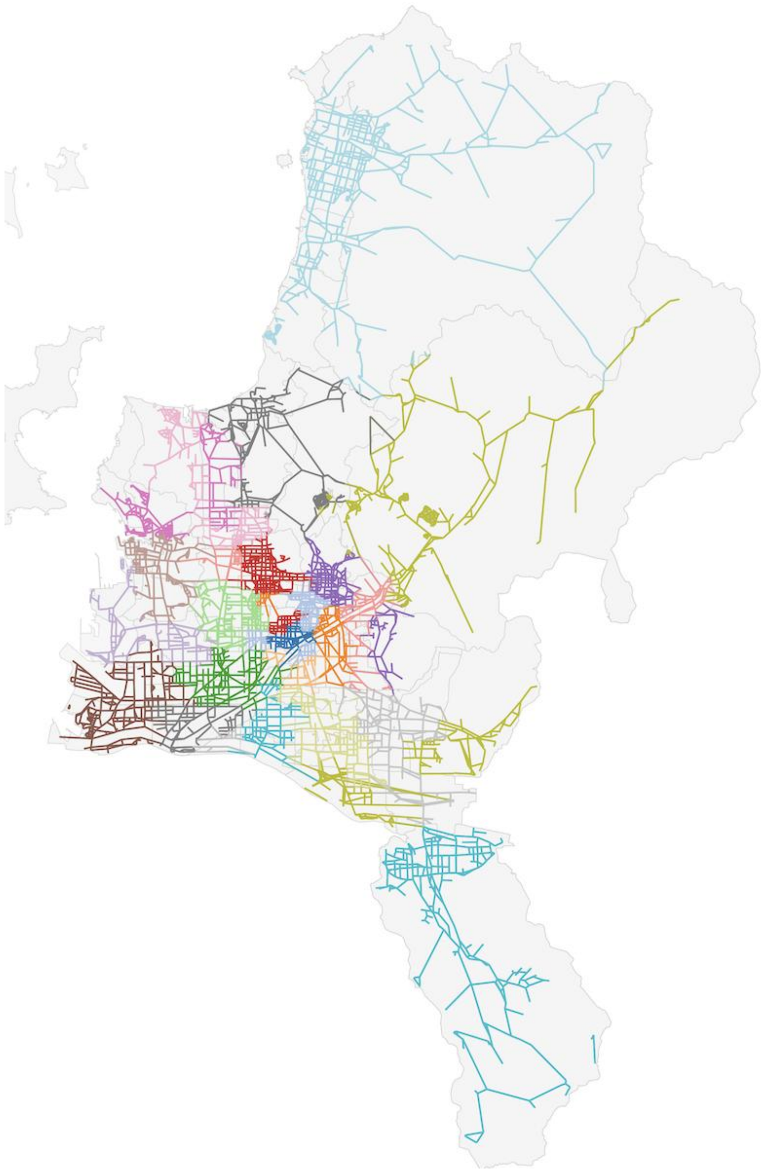
最大滞在可能台数の算出:

各ゾーン内に含まれるリンクの延長と車線数、および渋滞密度を用いて、ゾーン内に滞在できる理論上の最大車両数を定義しました。

$$N_{max,z} = \sum_{i \in z} (L_i \times \mathrm{lanes}_i) \times k_{jam}$$

マクロ距離の定式化:

ミクロなネットワークのダイクストラ探索を用いて、ゾーン内（Intra）、起点から境界（Origin）、通過（Transit）、境界から終点（Destination）という4パターンの「実効走行距離」を事前計算し、経路探索の基礎データとしました。



MFDに基づく動的旅行時間モデル

動的走行速度の計算:

自由旅行速度と、パラメータを用いたBPR関数型の速度低下モデルを採用しています。

$$v_z(n) = \frac{V_f}{1 + \alpha \left(\frac{n}{N_{max,z}} \right)^{\beta}}$$

リンク所要時間の算出:

マクロ距離と動的速度から、自動車の所要時間と、それに乗降ペナルティ等を加味したバスの所要時間を逐次計算しました

ロジットモデルによる効用計算と需要転換

効用関数:

時間価値、費用価値、およびバスの固有定数を用いて各手段の効用を計算します。

$$V_{car} = \beta_t \cdot t_{car} + \beta_c \cdot c_{car}$$

$$V_{bus} = \beta_t \cdot t_{bus} + \beta_c \cdot c_{bus} + ASC$$

選択確率:

$$P_{bus} = \frac{\exp(V_{bus})}{\exp(V_{car}) + \exp(V_{bus})}$$

社会便益の定式化とフィードバックループ

ログサム便益の算出:

バスという新たな選択肢が追加されたことによる便益の増加分を、時間価値を乗じて金銭的価値に換算します。

$$\Delta CS = \frac{\ln(\exp(V_{car}) + \exp(V_{bus})) - \ln(\exp(V_{car}))}{|\beta_t|} \times VOT$$

フィードバック構造:

「バスに乗客が移る」「道路の車が減る」「速度が上がり、車もバスも速くなる」「さらにバスの効用が上がる」という均衡計算（MAX_ITERによる反復）を実装した。

今回の最適化

決定変数:

バスが経由するゾーンの順番（ルート）

$$R = (z_1, z_2, \ldots, z_K) \quad \text{where} \quad z_k \in \mathbf{Z}$$

実行可能領域と制約条件:

①ルートの重複制約（ノードの単一訪問）
同じゾーンを2回以上経由する非効率なループ路線排除

$$z_i \neq z_j \quad (\forall i, j \in \{1, \ldots, K\}, i \neq j)$$

②路線の規模制約
広域をカバーするため、路線長（経由するゾーン数）は K個に固定

$$|R| = K$$

③輸送容量（定員）制約
ある区間におけるバスの通過乗客数は、提供されるバスの輸送力を上回ることはできません

$$q_{bus,i} \leq C_{capacity} \quad (\forall i \in R)$$

④交通均衡制約（フィードバック制約） 評価される便益や所要時間は、車からバスへの需要転換によって道路の滞在台数が減少し、それに伴い速度が向上した「事後」の均衡状態で計算されなければなりません。

$$v_z = \frac{V_f}{1 + \alpha \left(\frac{n - \Delta q}{N_{max,z}} \right)^{\beta}}$$

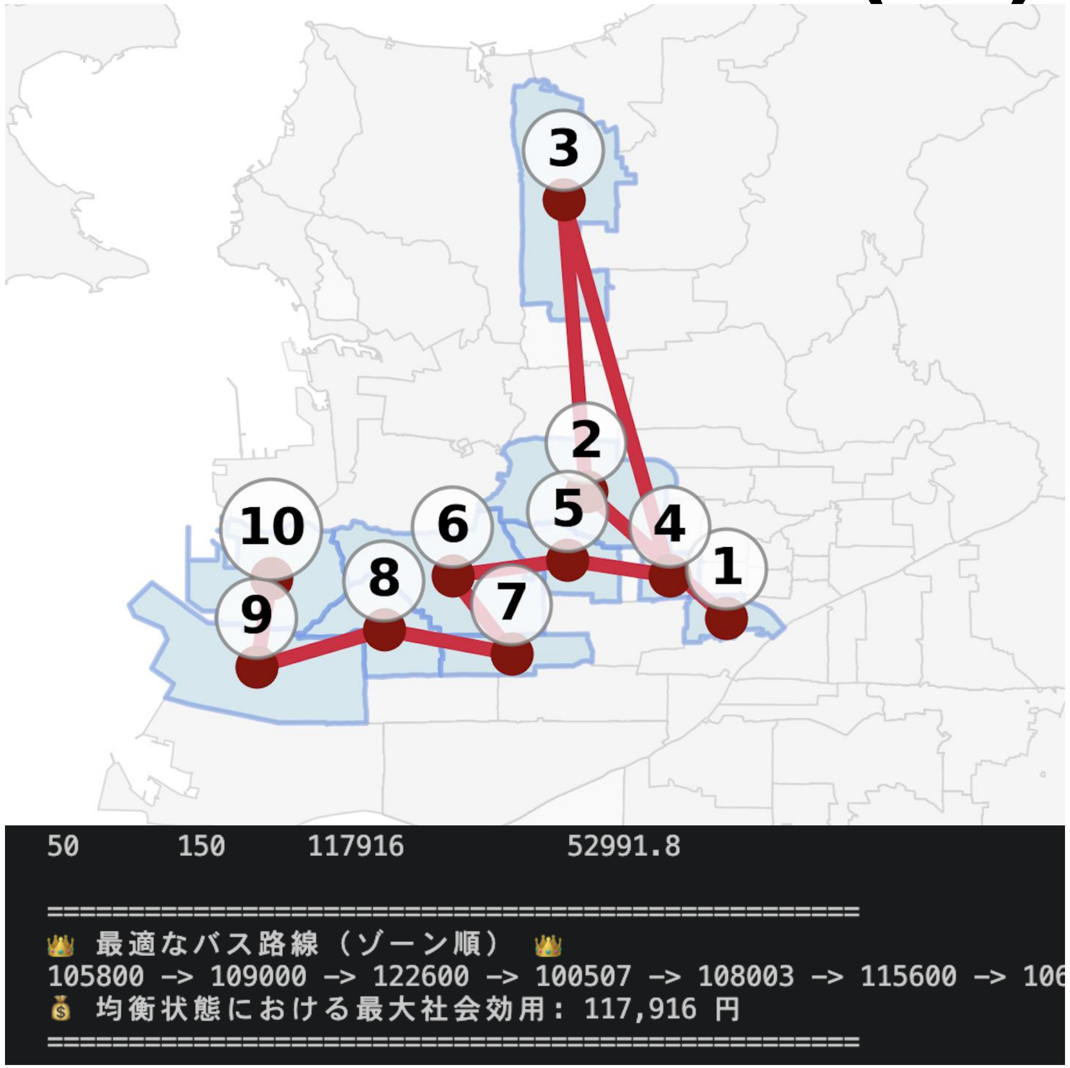
⑤リンクの連続性「直線距離×迂回ペナルティ」

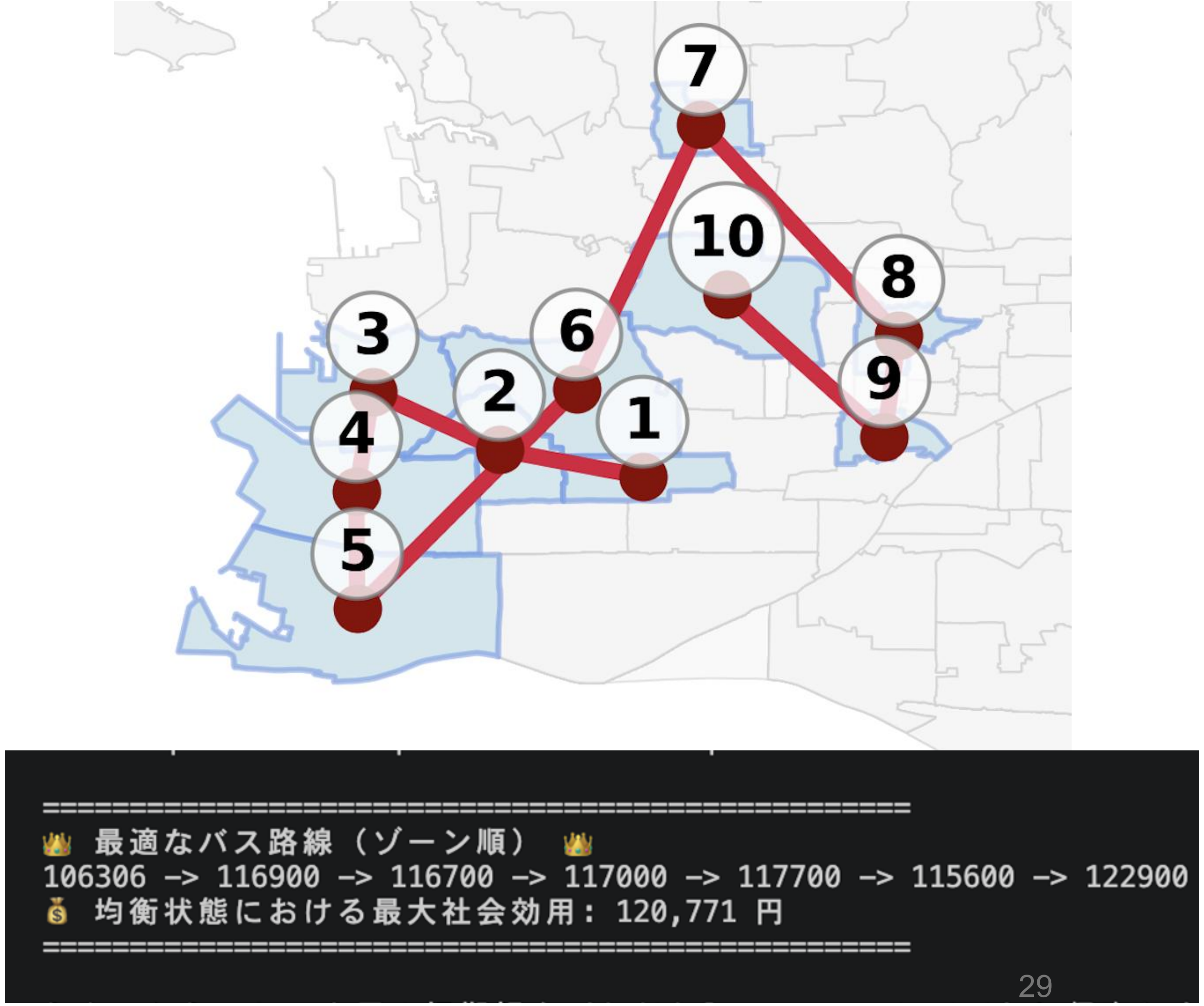
目的関数:

社会全体の純便益

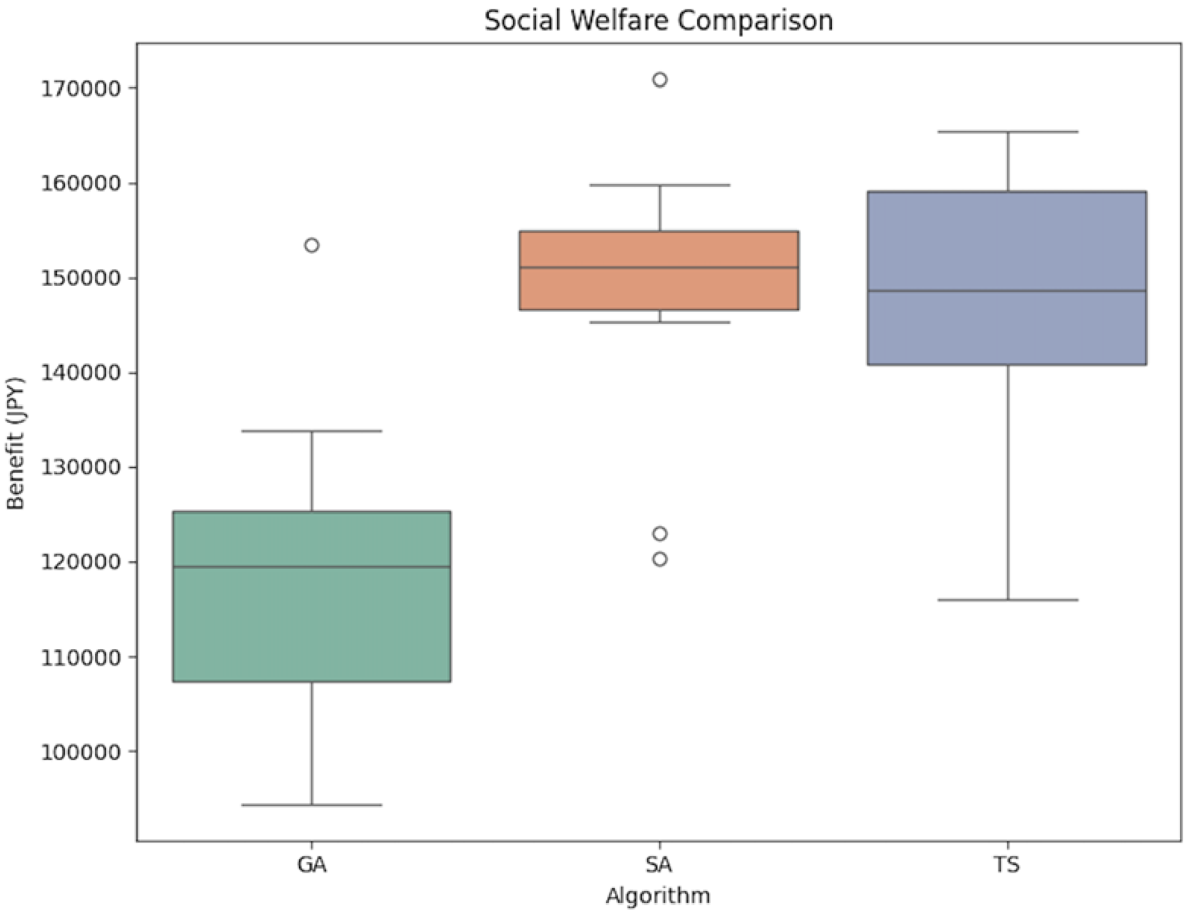
$$\max_R W(R) = \sum_{r \in \mathbf{Z}} \sum_{s \in \mathbf{Z}} \Delta CS_{rs}(R) - C_{op}(R)$$

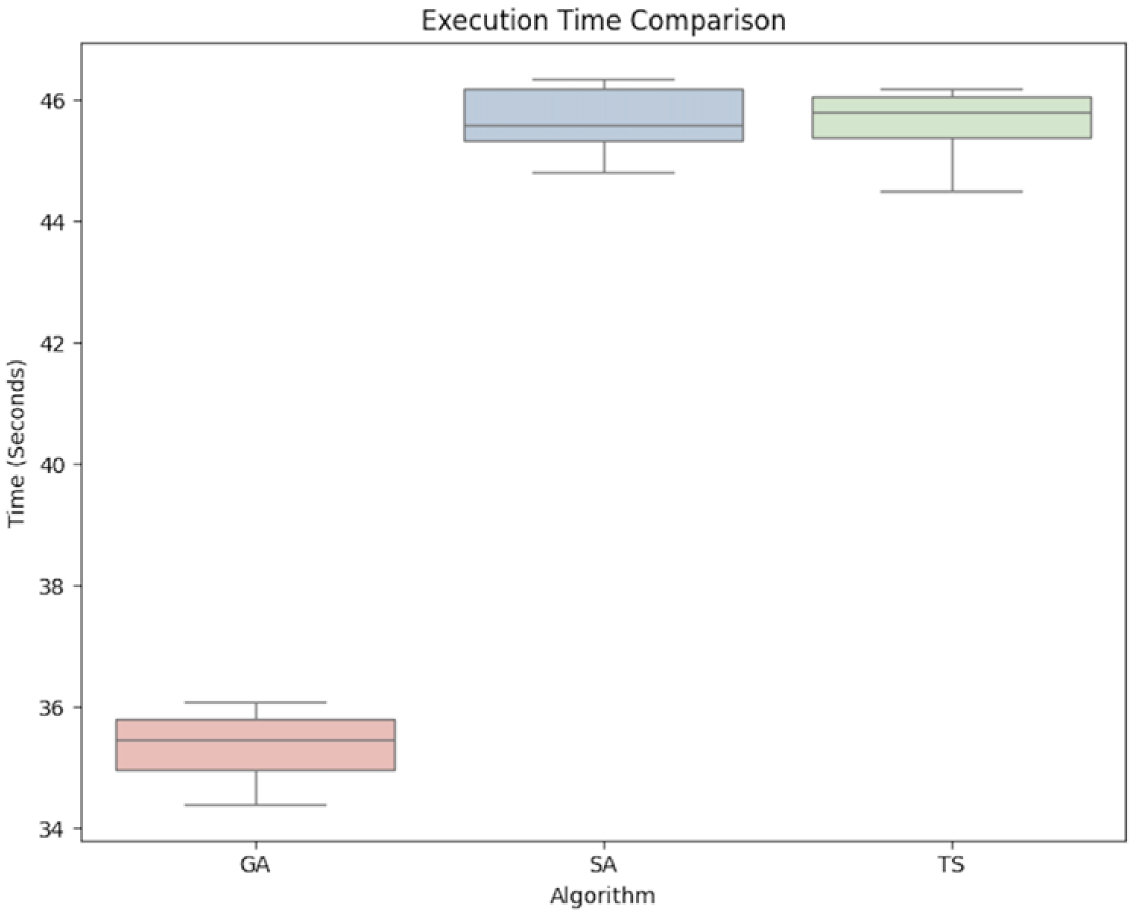
遺伝的アルゴリズム（左）と焼きなまし法による空間探索(右)



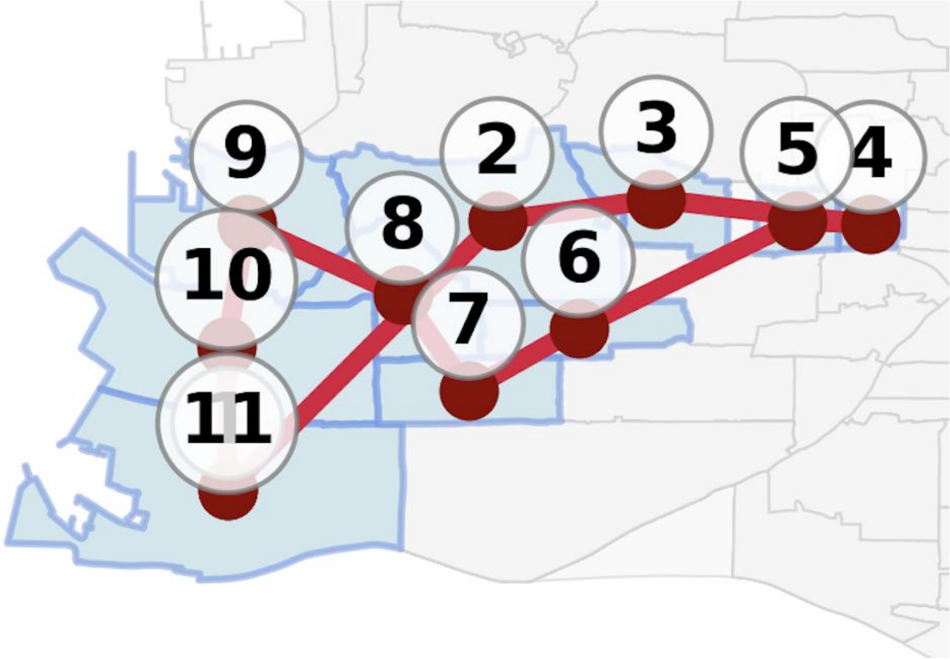


3種のヒューリスティックスの比較(評価回数 10000)

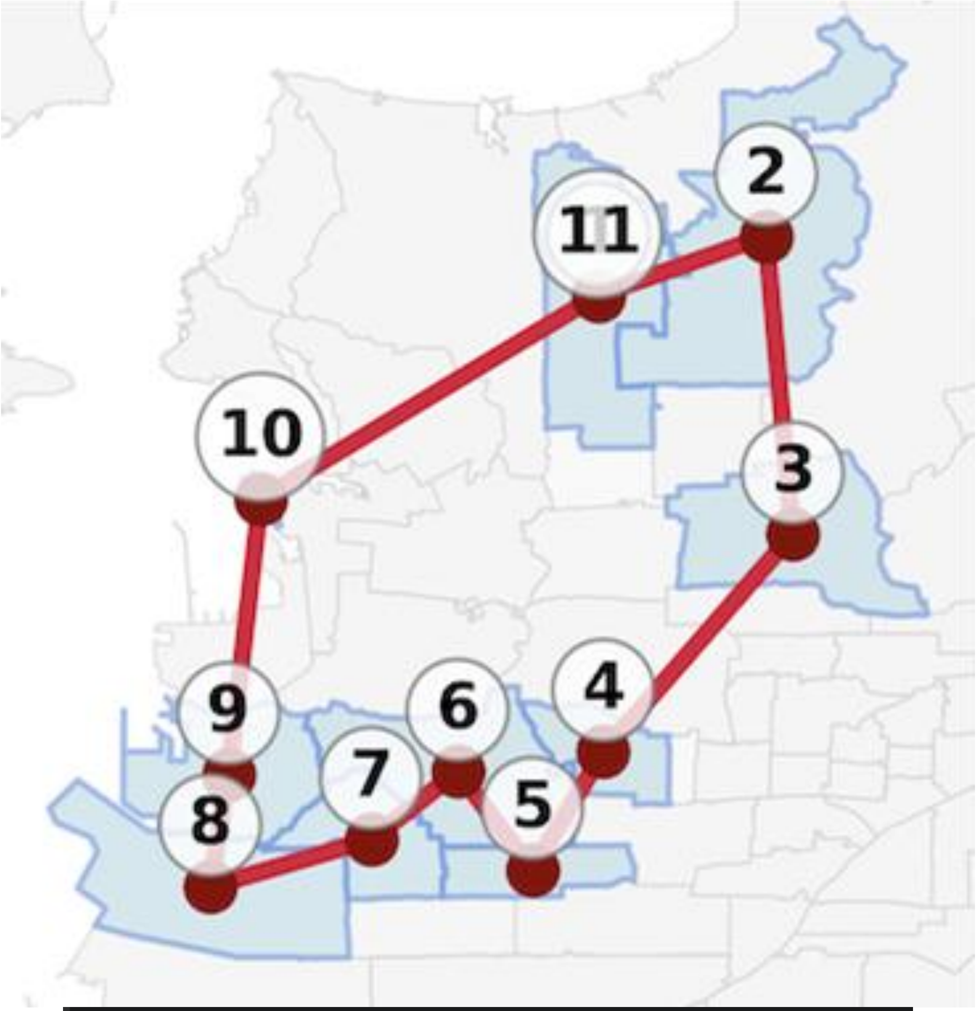




周回させると、、

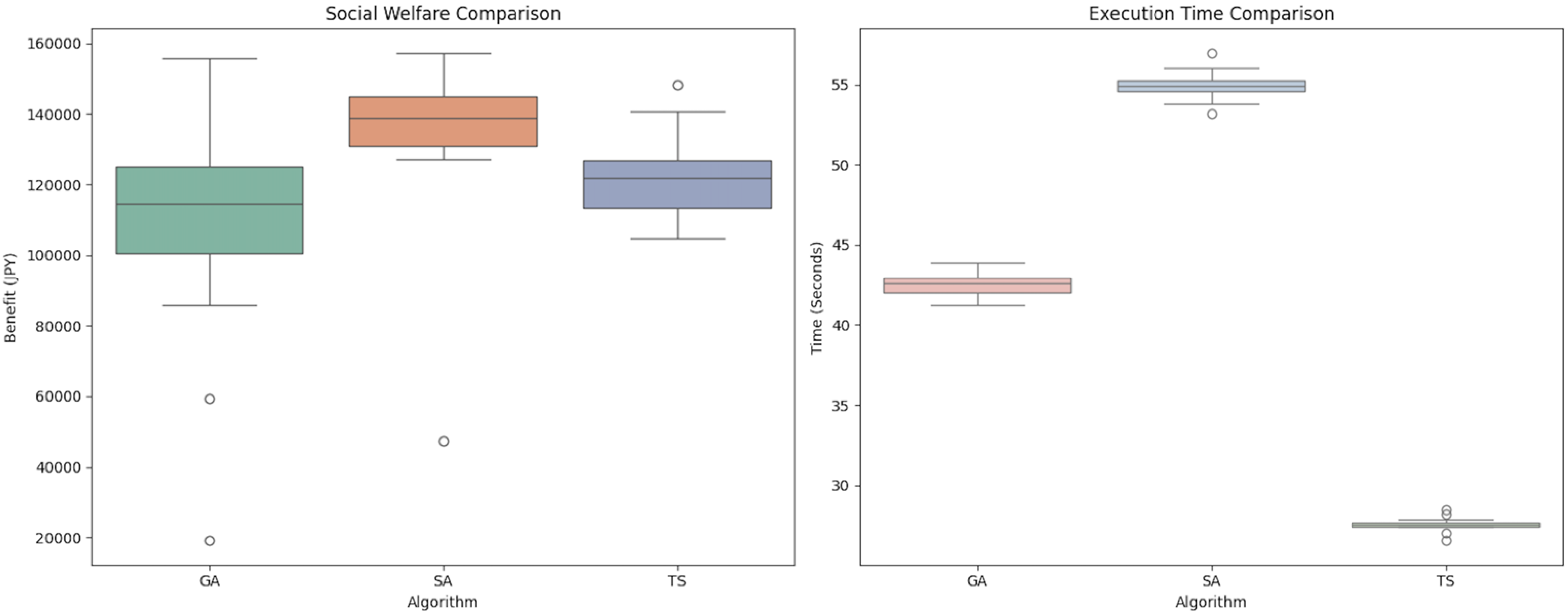



```
==================================================
👑 最適なバス路線（ゾーン順） 👑
125700 -> 122600 -> 106306 -> 115600 -> 116900 -> 1167
25700
💰 均衡状態における最大社会効用: 167,958 円
==================================================
```

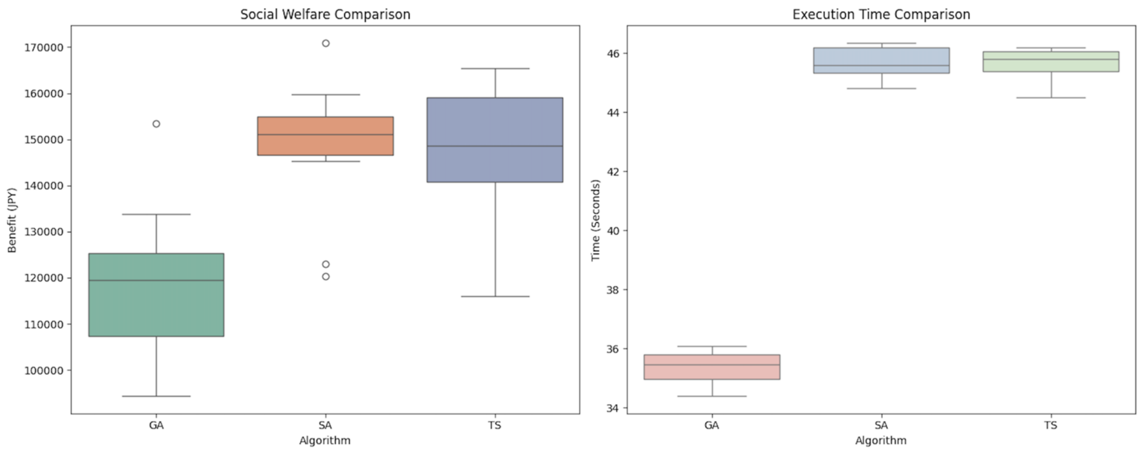


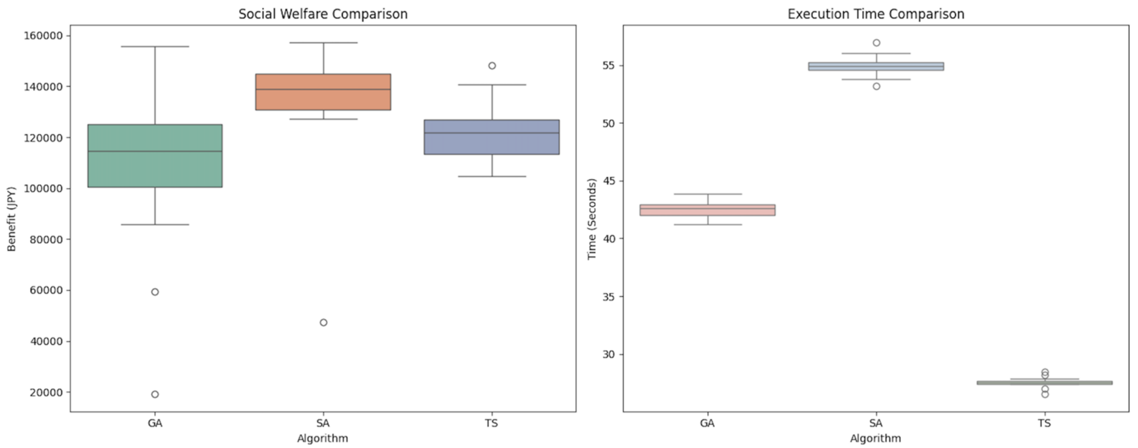

```
==================================================
👑 最適なバス路線（ゾーン順） 👑
117700 -> 115600 -> 108003 -> 100306 -> 100507 -> 1
17700
💰 均衡状態における最大社会効用: 121,376 円
==================================================
```

3種のヒューリスティックスの比較(評価回数10000)



3種のヒューリスティックスの比較





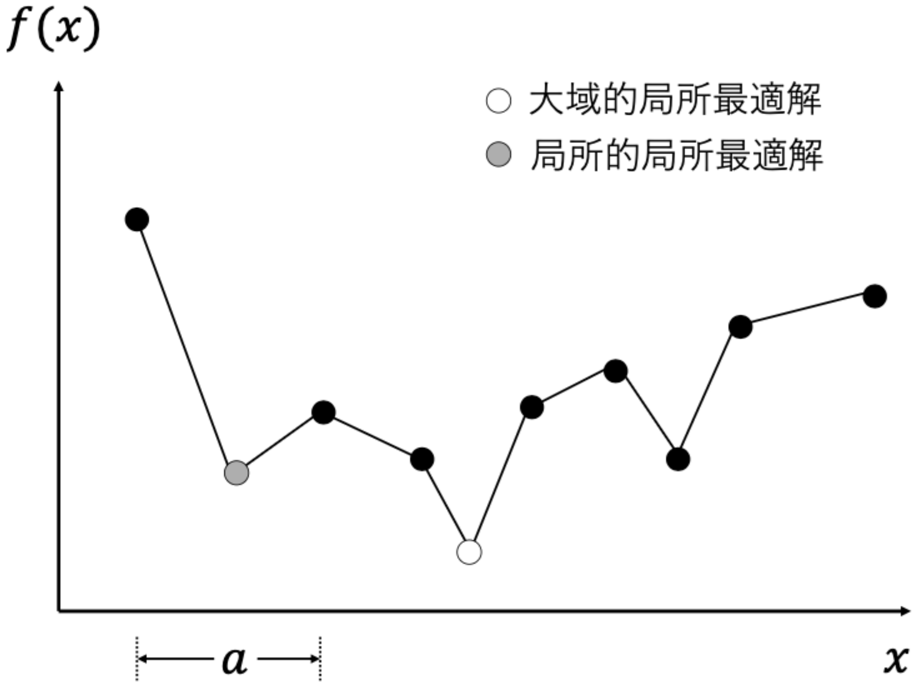
左でGAが速かったのは「交叉（配列を切り貼りするだけ）」の処理コストが、SAやTSの「毎回ルートを評価して近傍を作る」処理よりも相対的に軽かったためではないか

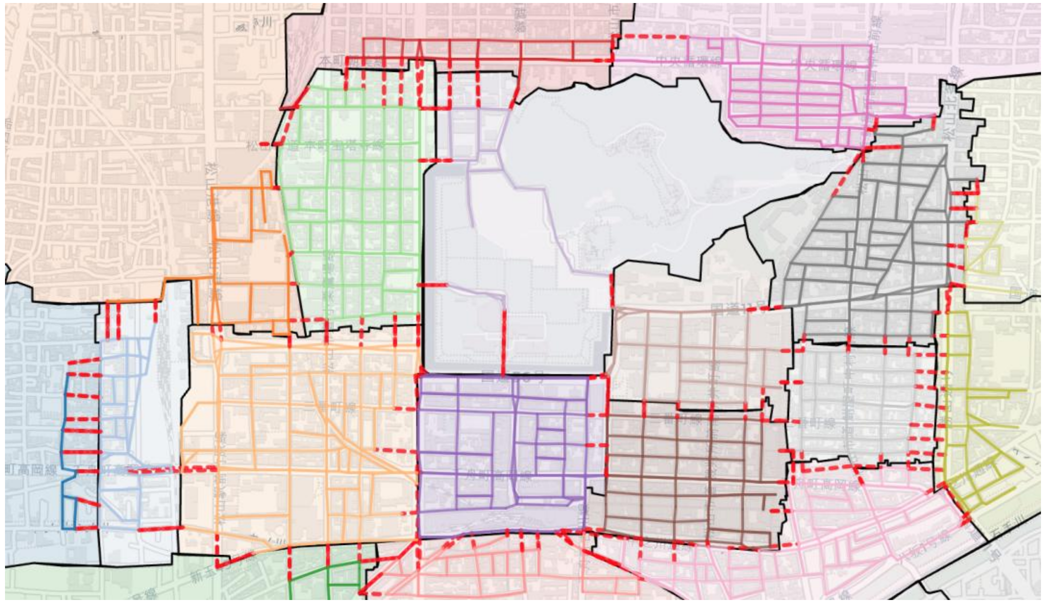
TSが非ループ路線でばらつきが大きくなったのは、「A→B」を「B→A」にスワップしてしまうと、その区間の需要を一切拾えなくなるという点が、ループの時と比べて変化が大きいためではないか

SAが安定して高いピークを見つけ出せているのは、「一時的にスコアが悪化しても、あえてその変更を受け入れる（確率的受理）」という性質が、非ループ路線のシビアな制約下でも強く機能しているためだろう

TSは早く求めたい時、SAは時間があって限界まで最大化したい時、GAは多様な解を得られる点で使い分けできるのではないか

ミクロな最適化の初期値にできないか





ミクロ経路への変換

代表ノードの抽出:

各Cゾーン内で、ミクログラフ上の次数（接続リンク数）が最大のノードを「代表バス停」と定義。

フィードバック構造:

$$t_a = t_0 \left(1 + 0.15 \left(\frac{v_a}{C_a} \right)^{4.0} \right)$$

実際の交通量をUEから取得し、リンク容量は１レーン600台として、BPR関数でリンク旅行時間を精密に再計算

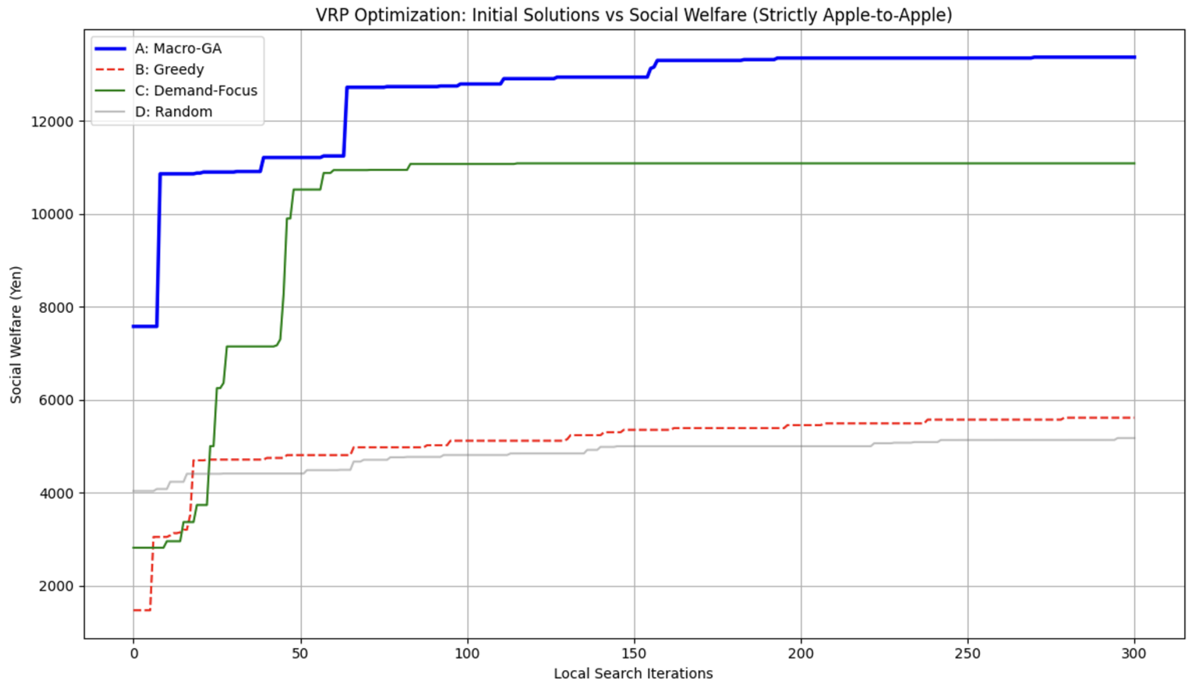
他の手法との比較を行う

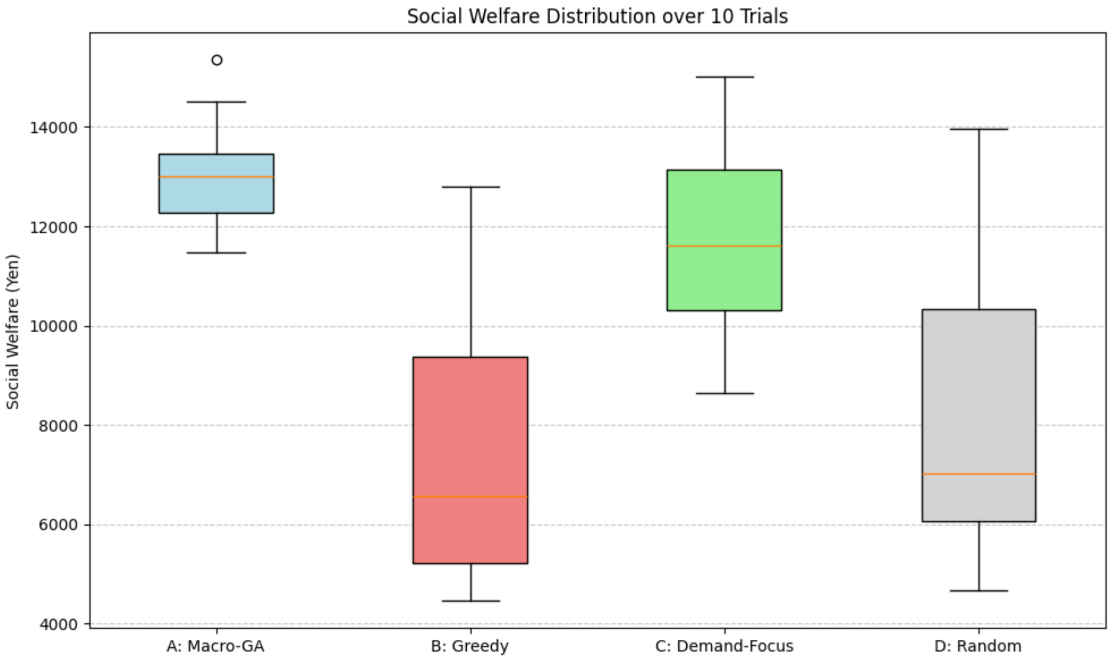
局所探索とミクロレベルのGA（Generations=50）を回す

Macro-GA（提案手法）	
Greedy（近傍探索）	最も需要の高いノードから開始し、物理的距離が最も近い未訪問ノードを繋ぐ
Demand-Focus	単純にOD需要の着発量が多い上位ノードをピックアップして繋ぐ
Random	完全ランダム

他の手法との比較を行う

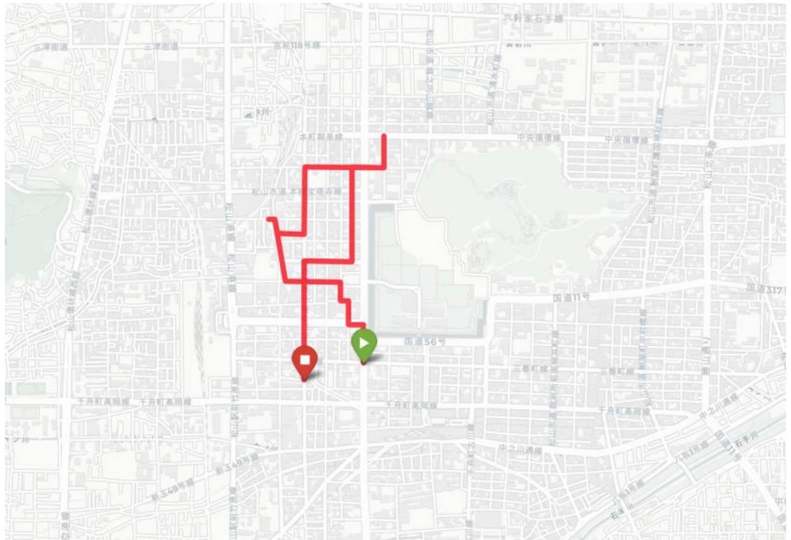
一番良さそう？←実際の都市との乖離がある

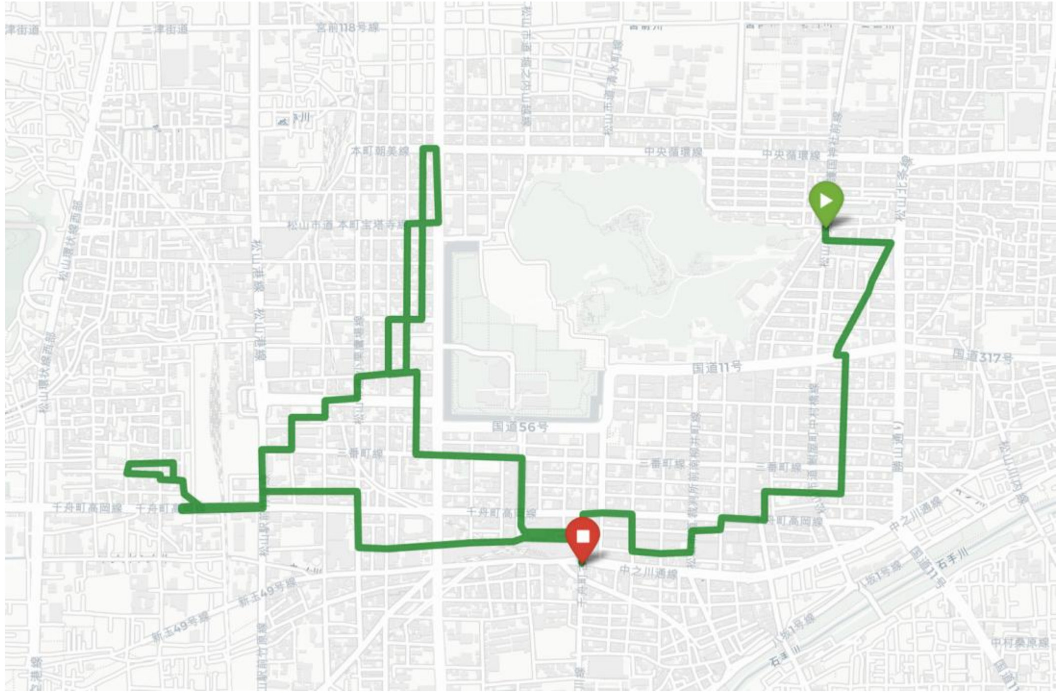


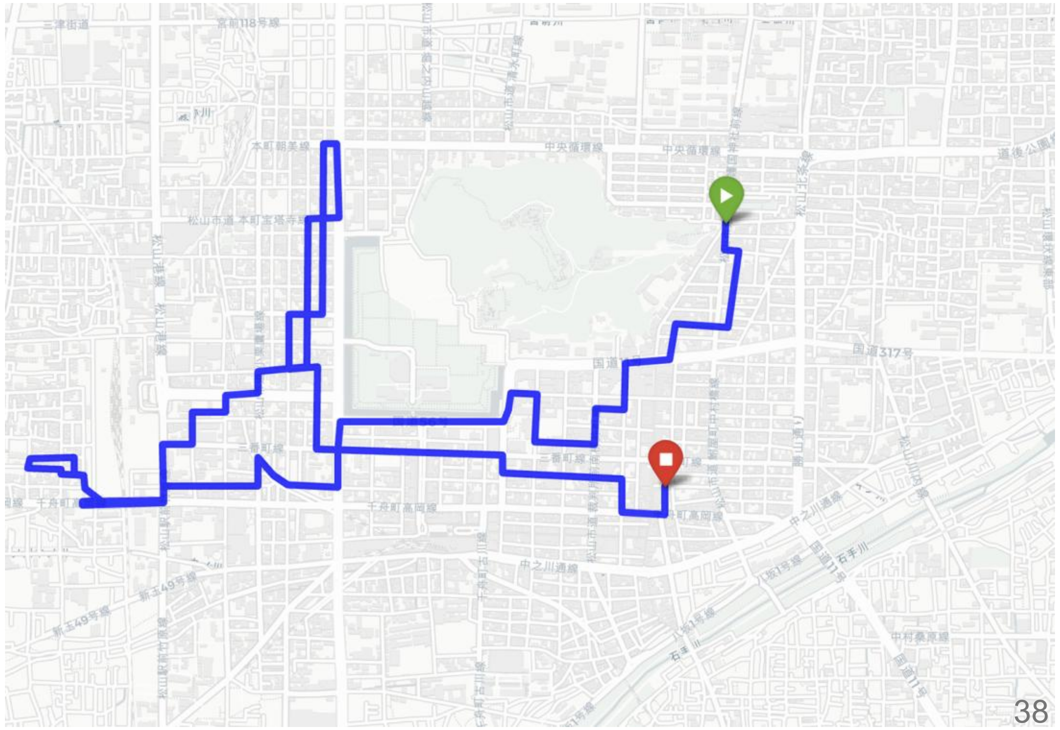


他の手法との比較を行う

青：GA 緑：Demand 赤：Greedy

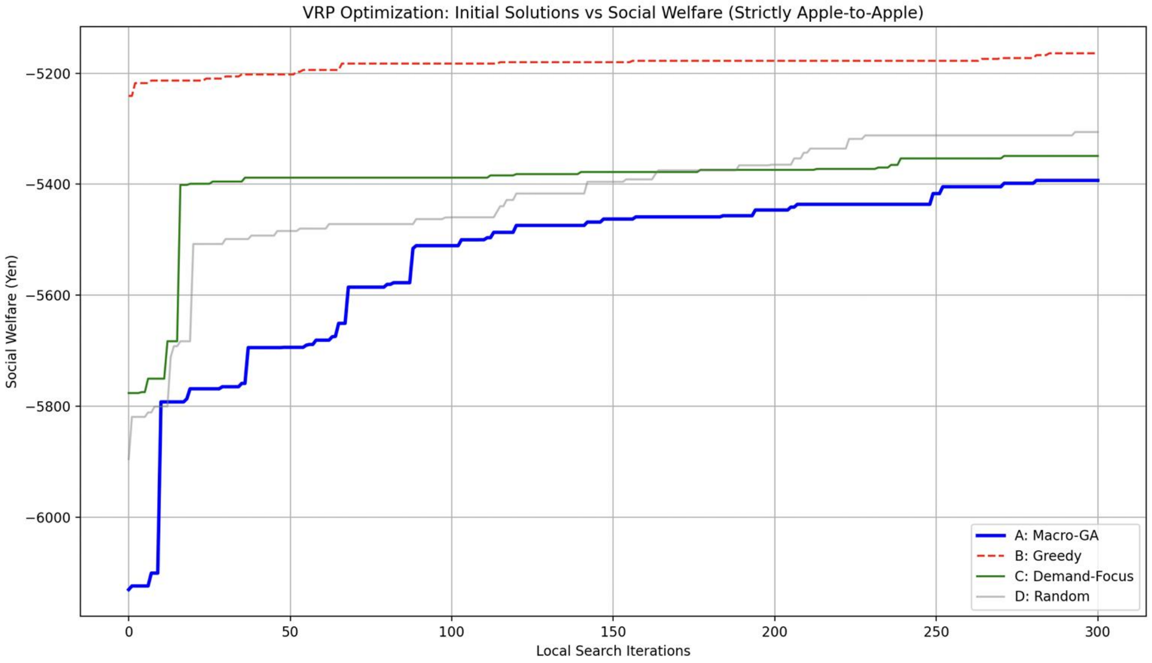


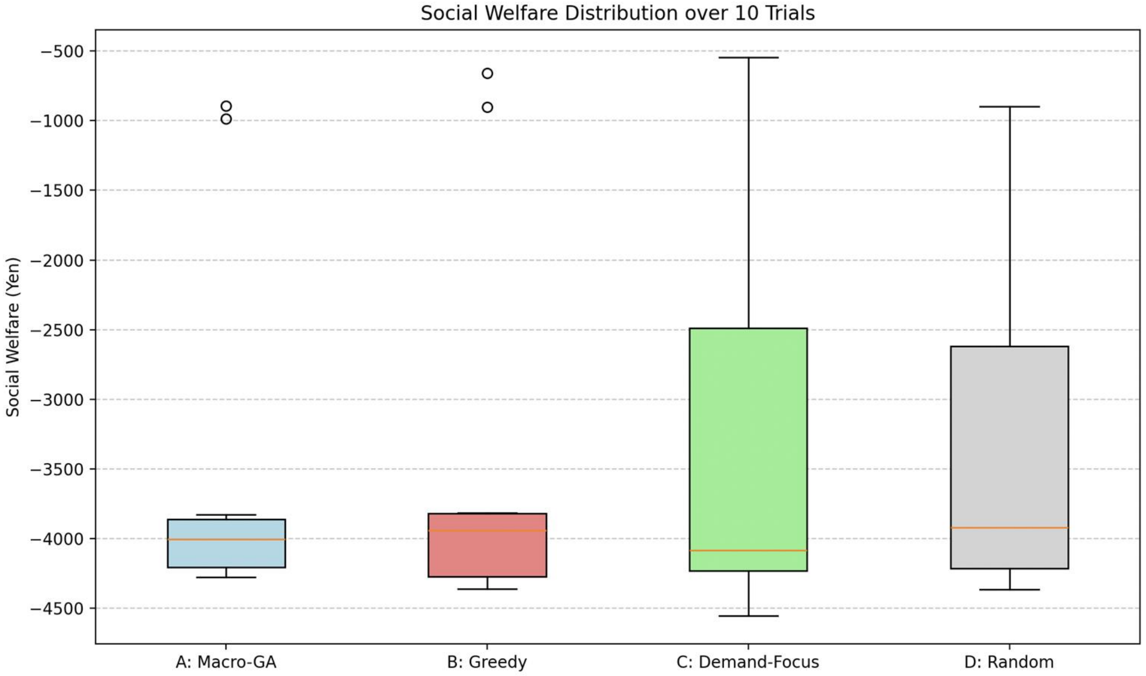




他の手法との比較を行う

マクロとミクロの不一致がある。そもそも導入しない方がいいときには不利な面がある。





階層化の限界と改善点

代表ノード固定化による経路の歪み

所要時間（コスト）の見積もり誤差が大きい

LLMについて

主にGeminiProを使用

コーディングもかなりGeminiにやってもらいました。
Geminiにコードを書かせる前になんと伝えるのかが難しかったです。
また、Geminiが書いたコードを見てて、あれおかしくね（理論的にも）、となることが多かったです。

時折無料版のChatGPTにも聞いてみると批判的な意見がもらえてよかったです。

ありがとうございました