Introduction to Optimization

Parin Chaipunya

[1] Department of Mathematics / [2] Center of Excellence in Theoretical and Computational Science
King Mongkut's University of Technology Thonburi

Multi-agent optimization
Mathematical economics
Alexandrov geometry in optimization
Optimal transport

Topic Overview

This introduction shall covers the following topics:

- Introduction to Optimization Modeling
- Unconstrained Optimization
  - Principles of Unconstrained Optimization
  - Gradient Descent Algorithm and its Variants
  - Numerical Examples
- Constrained Optimization
  - Principles of Constrained Optimization
  - Constrained Optimization Solvers
- Heuristic Approach

Constrained Optimization

Constrained Optimization

Recall that a constrained optimization problem requires two main ingredients, (1) an objective function f and (2) a constraint set C.

Constrained Optimization

Recall that a constrained optimization problem requires two main ingredients, (1) an objective function f and (2) a constraint set C.

The problem is formulated as

$$Min(f, C) \begin{cases} \min & f(x) \\ \text{s.t.} & x \in C. \end{cases}$$

Constrained Optimization

Recall that a constrained optimization problem requires two main ingredients, (1) an objective function f and (2) a constraint set C.

The problem is formulated as

$$Min(f, C) \begin{cases} \min & f(x) \\ \text{s.t.} & x \in C. \end{cases}$$

The presence of the constraint helps us narrow down the solution, but at the same time increase the complication of the problem.

Constrained Optimization

Recall that a constrained optimization problem requires two main ingredients, (1) an objective function f and (2) a constraint set C.

The problem is formulated as

$$Min(f, C) \begin{cases} \min & f(x) \\ \text{s.t.} & x \in C. \end{cases}$$

The presence of the constraint helps us narrow down the solution, but at the same time increase the complication of the problem.

A point x that satisfies the constraint (i.e. $x \in C$) is said to be *feasible*.

Structured Constrained Optimization Problem

Very often, the constraint set C is described by inequalities and equalities:

$$C = \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} g_i(x) \le 0, & \forall i = 1, 2, \cdots, r \\ h_j(x) = 0, & \forall j = 1, 2, \cdots, l \end{array} \right\}$$

Structured Constrained Optimization Problem

Very often, the constraint set C is described by inequalities and equalities:

$$C = \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} g_i(x) \le 0, & \forall i = 1, 2, \cdots, r \\ h_j(x) = 0, & \forall j = 1, 2, \cdots, l \end{array} \right\}$$

In this case, $Min(f, C)$ and $LMin(f, C)$ will be represented by $Min(f, g, h)$ $LMin(f, g, h)$, respectively, with

$$Min(f, g, h) \quad \left\{ \begin{array}{lll} \min & f(x) & \\ \text{s.t.} & g_i(x) \le 0 & \forall i = 1, 2, \cdots, r \\ & h_j(x) = 0 & \forall j = 1, 2, \cdots, l. \end{array} \right.$$

Structured Constrained Optimization Problem

Very often, the constraint set C is described by inequalities and equalities:

$$C = \left\{ x \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} g_i(x) \leq 0, & \forall i = 1, 2, \cdots, r \\ h_j(x) = 0, & \forall j = 1, 2, \cdots, l \end{array} \right\}$$

In this case, $Min(f, C)$ and $LMin(f, C)$ will be represented by $Min(f, g, h)$ $LMin(f, g, h)$, respectively, with

$$Min(f, g, h) \quad \left\{ \begin{array}{lll} \min & f(x) & \\ \text{s.t.} & g_i(x) \leq 0 & \forall i = 1, 2, \cdots, r \\ & h_j(x) = 0 & \forall j = 1, 2, \cdots, l. \end{array} \right.$$

If there is no equality constraints, then we just write $Min(f, g)$.

Karush-Kuhn-Tucker (KKT) Conditions

To solve a structured constrained optimization problem, one resorts to the KKT conditions:

$$\begin{cases} \nabla f(x) + \sum_{i=1}^{r} \lambda_i \nabla g_i(x) + \sum_{j=1}^{l} \mu_j \nabla h_j(x) = 0, \\ \lambda_i g_i(x) = 0 \qquad \text{for all } i = 1, \cdots, r, \end{cases}$$

for some scalars $\lambda_1, \cdots, \lambda_r \geq 0$ and $\mu_1, \cdots, \mu_l \in \mathbb{R}$.

Karush-Kuhn-Tucker (KKT) Conditions

To solve a structured constrained optimization problem, one resorts to the KKT conditions:

$$\begin{cases} \nabla f(x) + \sum_{i=1}^{r} \lambda_i \nabla g_i(x) + \sum_{j=1}^{l} \mu_j \nabla h_j(x) = 0, \\ \lambda_i g_i(x) = 0 \qquad \text{for all } i = 1, \cdots, r, \end{cases}$$

for some scalars $\lambda_1, \cdots, \lambda_r \geq 0$ and $\mu_1, \cdots, \mu_l \in \mathbb{R}$.

The second conditions are called the *complementarity conditions*, which states that if an inequality constraint $g_i(x) < 0$ holds strictly (i.e. it is inactive), then $\lambda_i = 0$. This 'deactivate' the participation of its gradient $\nabla g_i(x)$ in the first condition.

Karush-Kuhn-Tucker (KKT) Conditions

To solve a structured constrained optimization problem, one resorts to the KKT conditions:

$$\begin{cases} \nabla f(x) + \sum_{i=1}^{r} \lambda_i \nabla g_i(x) + \sum_{j=1}^{l} \mu_j \nabla h_j(x) = 0, \\ \lambda_i g_i(x) = 0 \qquad \text{for all } i = 1, \cdots, r, \end{cases}$$

for some scalars $\lambda_1, \cdots, \lambda_r \geq 0$ and $\mu_1, \cdots, \mu_l \in \mathbb{R}$.

The second conditions are called the *complementarity conditions*, which states that if an inequality constraint $g_i(x) < 0$ holds strictly (i.e. it is inactive), then $\lambda_i = 0$. This 'deactivate' the participation of its gradient $\nabla g_i(x)$ in the first condition.

We write $\bar{x} \in KKT(f, g, h)$ if the KKT conditions holds at $\bar{x}$ for some scalars $\lambda_1, \cdots, \lambda_r \geq 0$ and $\mu_1, \cdots, \mu_l \in \mathbb{R}$.

Principles of Structured Constrained Optimization Problems

Necessity: Under some ‘technical’ assumptions,

$$\bar{x} \in LMin(f, g, h) \implies \bar{x} \in KKT(f, g, h).$$

Principles of Structured Constrained Optimization Problems

Necessity: Under some 'technical' assumptions,

$$\bar{x} \in LMin(f, g, h) \implies \bar{x} \in KKT(f, g, h).$$

Sufficiency: If $\bar{x}$ is feasible and $\bar{x} \in KKT(f, g, h)$ with multipliers $\bar{\lambda}$ and $\bar{\mu}$ and the following holds:

$$d^\top \nabla_x^2 L(\bar{x}, \bar{\lambda}, \bar{\mu}) d > 0, \quad \forall d : \begin{cases} \nabla g_i^\top(\bar{x}) d \leq 0 & \text{if } g_i(\bar{x}) = 0, \\ \nabla g_i^\top(\bar{x}) d = 0 & \text{if } g_i(\bar{x}) = 0 \text{ and } \bar{\lambda}_i > 0, \\ \nabla h_j(\bar{x})^\top d = 0 & \text{for all } j = 1, \cdots, l, \end{cases}$$

then $\bar{x} \in LMin(f, g, h)$.

Principles of Structured Constrained Optimization Problems

Necessity: Under some ‘technical’ assumptions,

$$\bar{x} \in LMin(f, g, h) \implies \bar{x} \in KKT(f, g, h).$$

Sufficiency: If $\bar{x}$ is feasible and $\bar{x} \in KKT(f, g, h)$ with multipliers $\bar{\lambda}$ and $\bar{\mu}$ and the following holds:

$$d^\top \nabla_x^2 L(\bar{x}, \bar{\lambda}, \bar{\mu}) d > 0, \quad \forall d : \begin{cases} \nabla g_i^\top(\bar{x}) d \leq 0 & \text{if } g_i(\bar{x}) = 0, \\ \nabla g_i^\top(\bar{x}) d = 0 & \text{if } g_i(\bar{x}) = 0 \text{ and } \bar{\lambda}_i > 0, \\ \nabla h_j(\bar{x})^\top d = 0 & \text{for all } j = 1, \cdots, l, \end{cases}$$

then $\bar{x} \in LMin(f, g, h)$.

In the above expression, $L(x, \lambda, \mu) = f(x) + \sum_i \lambda_i \nabla g_i(x) + \sum_j \mu_j \nabla h_j(x)$.

Principles of Structured Constrained Optimization Problems

Necessity: Under some 'technical' assumptions,

$$\bar{x} \in LMin(f, g, h) \implies \bar{x} \in KKT(f, g, h).$$

Principles of Structured Constrained Optimization Problems

Necessity: Under some 'technical' assumptions,

$$\bar{x} \in LMin(f, g, h) \implies \bar{x} \in KKT(f, g, h).$$

Sufficiency (convex programs): If $\bar{x}$ is feasible, $f, g_1, \cdots, g_r$ are all convex and $h_1, \cdots, h_l$ are all affine, then

$$\bar{x} \in KKT(f, g, h) \implies \bar{x} \in LMin(f, g, h).$$

Principles of Structured Constrained Optimization Problems

Necessity (linear constraints): If all g_i's and h_j's are affine, then

$$\bar{x} \in LMin(f, g, h) \implies \bar{x} \in KKT(f, g, h).$$

Principles of Structured Constrained Optimization Problems

Necessity (linear constraints): If all g_i's and h_j's are affine, then

$$\bar{x} \in LMin(f, g, h) \implies \bar{x} \in KKT(f, g, h).$$

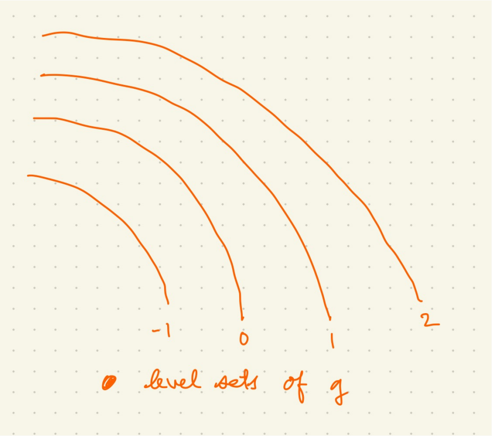
Sufficiency (convex programs): If $\bar{x}$ is feasible, $f, g_1, \cdots, g_r$ are all convex and $h_1, \cdots, h_l$ are all affine, then

$$\bar{x} \in KKT(f, g, h) \implies \bar{x} \in Min(f, g, h).$$

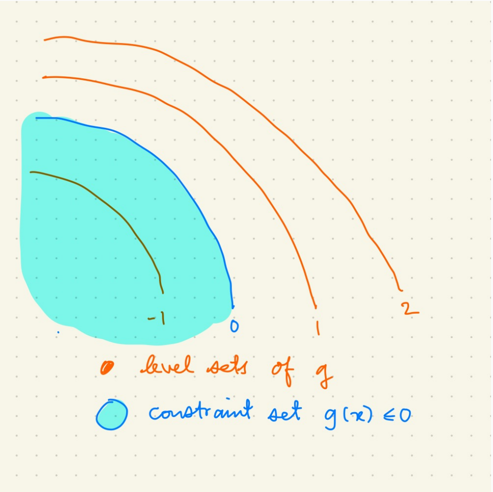
Principles of Structured Constrained Optimization Problems

To find a point $\bar{x} \in LMin(f, g, h)$:
Filter: Find all KKT points $\bar{x}$ with multipliers $\bar{\lambda}$ and $\bar{\mu}$.
Confirm: Check the positivity of $\nabla_x^2 L(\bar{x}, \bar{\lambda}, \bar{\mu})$ at each KKT points.

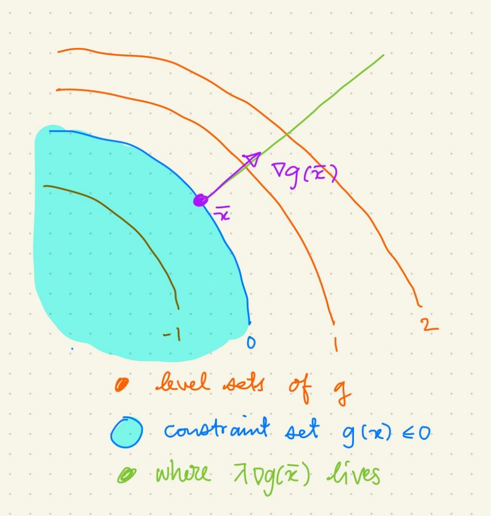
KKT conditions explained



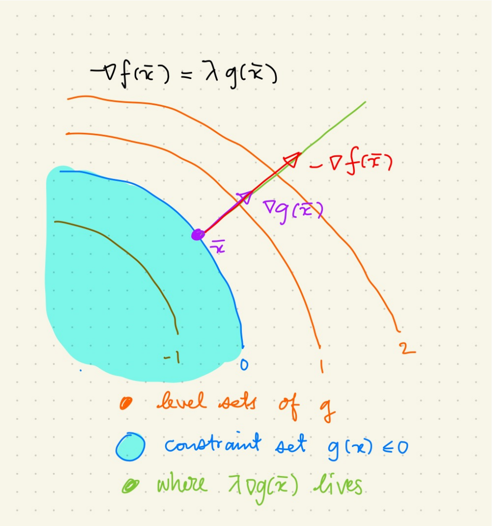
KKT conditions explained



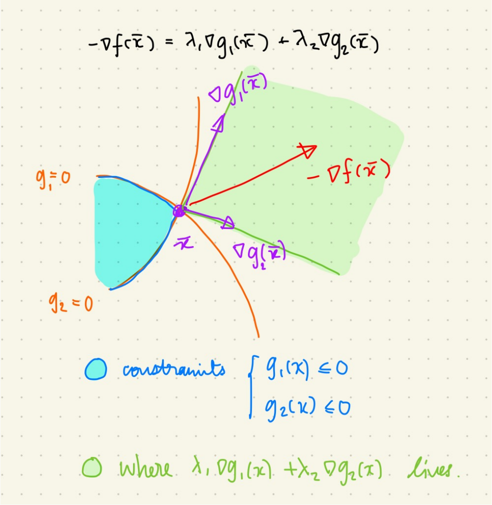
KKT conditions explained



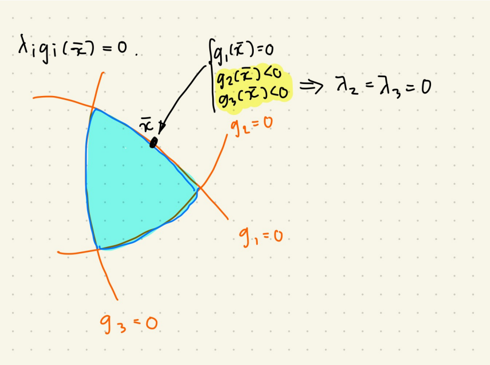
KKT conditions explained



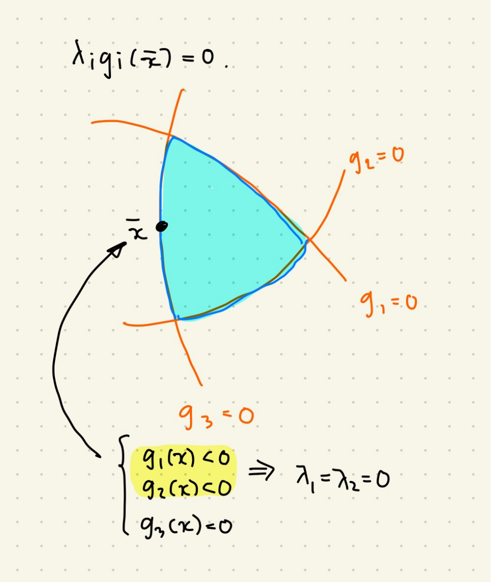
KKT conditions explained



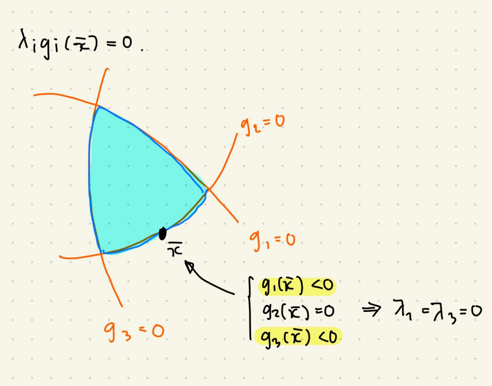
KKT conditions explained



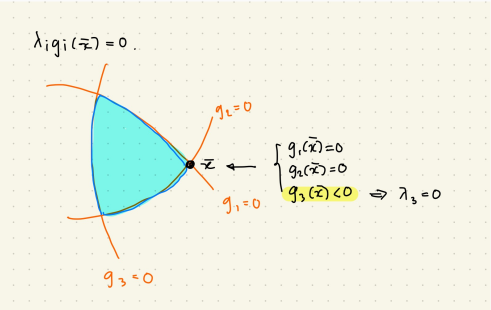
KKT conditions explained



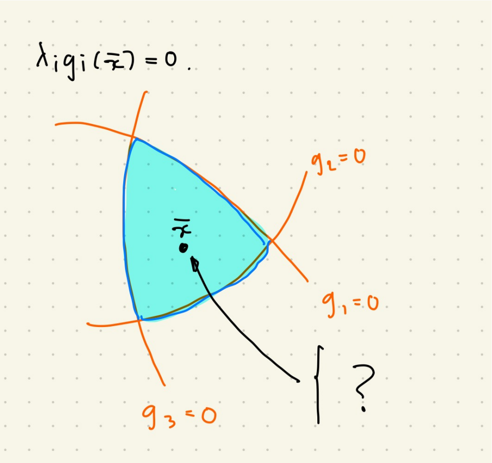
KKT conditions explained



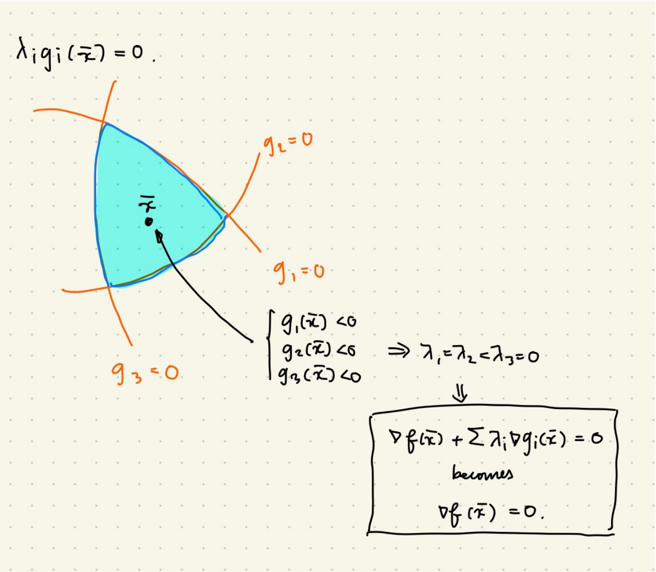
KKT conditions explained



KKT conditions explained



KKT conditions explained



KKT is somewhat sensitive.

The KKT conditions are somewhat sensitive.

KKT is somewhat sensitive.

The KKT conditions are somewhat sensitive. The same problem formulated differently may yield totally opposite results.

KKT is somewhat sensitive.

The KKT conditions are somewhat sensitive. The same problem formulated differently may yield totally opposite results.

Consider the problems

$$\begin{cases} \min & x_1 + x_2 \\ \text{s.t.} & x_1^2 + x_2^2 = 1. \end{cases} \tag{1}$$

and

$$\begin{cases} \min & x_1 + x_2 \\ \text{s.t.} & (x_1^2 + x_2^2 - 1)^2 = 0. \end{cases} \tag{2}$$

KKT conditions explained

$$\min \; x_1 + x_2 \quad \rightarrow f(x_1, x_2) \rightarrow \nabla f(x_1,x_2) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\text{s.t. } x_1^2 + x_1^2 - 1 = 0 \quad \rightarrow h_1(x_1, x_2) \rightarrow \nabla h_1(x_1,x_2) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}$$

$$\nabla f(x) + \mu \nabla h_1(x) = 0$$

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix} + \mu \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = 0$$

$$\left.\begin{array}{l} 1 + 2\mu x_1 = 0 \\ 1 + 2\mu x_2 = 0 \end{array}\right. \Rightarrow \left.\begin{array}{l} x_1 = -\frac{1}{2\mu} \\ x_2 = -\frac{1}{2\mu} \end{array}\right\} \; (*)$$

From the constraint,

$$x_1^2 + x_2^2 = 1$$

$$\frac{1}{4\mu^2} + \frac{1}{4\mu^2} = 1$$

$$\frac{1}{2\mu^2} = 1$$

$$\mu = \frac{1}{\sqrt{2}}$$

From (*), we get $x_1 = x_2 = -\frac{1}{\sqrt{2}}$.

The candidate for the local solution is $x = \begin{bmatrix} -1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}$.

KKT conditions explained

$$\min \quad x_1 + x_2$$
$$\text{s.t.} \quad (x_1^2 + x_2^2 - 1)^2 = 0$$

$$\nabla f(x) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\nabla h_2(x) = \begin{bmatrix} 4x_1 (x_1^2 + x_2^2 - 1) \\ 4x_2 (x_1^2 + x_2^2 - 1) \end{bmatrix}.$$

KKT condition:

$$\begin{bmatrix} -1 \\ -1 \end{bmatrix} = \mu \begin{bmatrix} 4x_1 (x_1^2 + x_2^2 - 1) \\ 4x_2 (x_1^2 + x_2^2 - 1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

← no solution on the constraint set.

If the constraint is satisfied, then $(x_1^2 + x_2^2 - 1) = 0$

The gradient $\nabla h_2(x) = 0$ for all $x \in C_2$.

Some MATLAB Solvers for Structured Problems

Well-known solvers

The following solvers are well known and many of them can be called in Matlab.

- `fmincon - General nonlinear programs`

Well-known solvers

The following solvers are well known and many of them can be called in Matlab.

- `fmincon - General nonlinear programs`
- `linprog - Linear programs`

Well-known solvers

The following solvers are well known and many of them can be called in Matlab.

- `fmincon - General nonlinear programs`
- `linprog - Linear programs`
- `quadprog - Linearly constrained quadratic programs (LQP)`

Well-known solvers

The following solvers are well known and many of them can be called in Matlab.

- `fmincon - General nonlinear programs`
- `linprog - Linear programs`
- `quadprog - Linearly constrained quadratic programs (LQP)`
- `intlinprog - Mixed integer linear programs (MILP)`

Well-known solvers

The following solvers are well known and many of them can be called in Matlab.

- `fmincon - General nonlinear programs`
- `linprog - Linear programs`
- `quadprog - Linearly constrained quadratic programs (LQP)`
- `intlinprog - Mixed integer linear programs (MILP)`
- `CVX - Structured convex programs`

Well-known solvers

The following solvers are well known and many of them can be called in Matlab.

- `fmincon - General nonlinear programs`
- `linprog - Linear programs`
- `quadprog - Linearly constrained quadratic programs (LQP)`
- `intlinprog - Mixed integer linear programs (MILP)`
- `CVX - Structured convex programs`
- `Gurobi / Mosek / CPLEX / AMPL / etc.`

Keep in mind that there is no algorithm that can be used to find a true (local) optimum for general nonlinear programs.

Well-known solvers

The following solvers are well known and many of them can be called in Matlab.

- `fmincon` - General nonlinear programs
- `linprog` - Linear programs
- `quadprog` - Linearly constrained quadratic programs (LQP)
- `intlinprog` - Mixed integer linear programs (MILP)
- `CVX` - Structured convex programs
- `Gurobi / Mosek / CPLEX / AMPL / etc.`

Keep in mind that there is no algorithm that can be used to find a true (local) optimum for general nonlinear programs.

Hence, eventhough the above solvers succeeded, the the reported result can be incorrect.

The tool `fmincon` solves a structured constrained optimization problem that takes the form

$$\begin{cases} \min & f(x) \\ \text{s.t.} & c_i(x) \leq 0 \quad \forall i = 1, 2, \cdots, r \\ & ceq_j(x) = 0 \quad \forall j = 1, 2, \cdots, l \\ & Ax \leq b \\ & Aeq \cdot x = beq \\ & lb \leq x \leq ub. \end{cases}$$

by the command

```
fmincon(f,x0,A,b,Aeq,beq,lb,ub,nonlcon)
```

where `nonlcon` outputs the nonlinear constraints *c* and *ceq* in a vector form.

Ex. Let's try `fmincon` with the soft drink manufacturing problem.

The tool `fmincon` solves a structured linear program that takes the form

$$\begin{cases} \min & f^\top x \\ \text{s.t.} & Ax \le b \\ & Aeq \cdot x = beq \\ & lb \le x \le ub. \end{cases}$$

by the command

```
linprog(f,A,b,Aeq,beq,lb,ub).
```

Ex. Let's try `linprog` with the portfolio optimization problem.

`linprog` for sequential investment planning (1)

General purpose. Suppose that you have an initial amount of money C_0 to invest over a time period of T years in N zero-coupon bonds. Each bond k pays an interest rate ρ_k that compounds each year, and pays the principal plus compounded interest at the end of a maturity period m_k. The objective is to maximize the total amount of money after T years.

linprog for sequential investment planning (2)

Ex. Invest \$1,000 in $N = 4$ types of bonds $B_1, \cdots, B_4$ over the period of $T = 5$ years.

`linprog` for sequential investment planning (2)

Ex. Invest \$1,000 in $N = 4$ types of bonds $B_1, \cdots, B_4$ over the period of $T = 5$ years.

B_1: Can be purchased in Year 1. Maturity period of 4 years.
Interest rate of 2%.

B_2: Can be purchased in Year 5. Maturity period of 1 year.
Interest rate of 4%.

B_3: Can be purchased in Year 2. Maturity period of 4 years.
Interest rate of 6%.

B_4: Can be purchased in Year 2. Maturity period of 3 years.
Interest rate of 6%.

`linprog` for sequential investment planning (2)

Ex. Invest \$1,000 in $N = 4$ types of bonds $B_1, \cdots, B_4$ over the period of $T = 5$ years.

B_1: Can be purchased in Year 1. Maturity period of 4 years. Interest rate of 2%.

B_2: Can be purchased in Year 5. Maturity period of 1 year. Interest rate of 4%.

B_3: Can be purchased in Year 2. Maturity period of 4 years. Interest rate of 6%.

B_4: Can be purchased in Year 2. Maturity period of 3 years. Interest rate of 6%.

We also create B_0 for the choice not to invest in any bonds. The interest rate for B_0 is bank interest rate $\rho_0 = 0.25\%$, which is assumed to be fixed over the period.

`linprog` for sequential investment planning (2)

<table>
<tr><th></th><th>Year 1</th><th>Year 2</th><th>Year 3</th><th>Year 4</th><th>Year 5</th></tr>
<tr><td>B_0</td><td>x_5
0.25%</td><td>x_6
0.25%</td><td>x_7
0.25%</td><td>x_8
0.25%</td><td>x_9
0.25%</td></tr>
<tr><td>B_1</td><td colspan="4">x_1
2%</td><td></td></tr>
<tr><td>B_2</td><td></td><td></td><td></td><td></td><td>x_2
4%</td></tr>
<tr><td>B_3</td><td></td><td colspan="4">x_3
6%</td></tr>
<tr><td>B_4</td><td></td><td colspan="3">x_4
6%</td><td></td></tr>
</table>

`linprog` for sequential investment planning (2)

<table>
<tr><th></th><th>Year 1</th><th>Year 2</th><th>Year 3</th><th>Year 4</th><th>Year 5</th></tr>
<tr><td>B_0</td><td>x_5
0.25%</td><td>x_6
0.25%</td><td>x_7
0.25%</td><td>x_8
0.25%</td><td>x_9
0.25%</td></tr>
<tr><td>B_1</td><td colspan="4">x_1
2%</td><td></td></tr>
<tr><td>B_2</td><td></td><td></td><td></td><td></td><td>x_2
4%</td></tr>
<tr><td>B_3</td><td></td><td colspan="4">x_3
6%</td></tr>
<tr><td>B_4</td><td></td><td colspan="3">x_4
6%</td><td></td></tr>
</table>

Let x_k denotes the amount of investment according to the above table and $r_k = (1 + \rho_k/100)^{m_k}$ denotes the net return of B_k.

`linprog` for sequential investment planning (2)

<table>
<tr><th></th><th>Year 1</th><th>Year 2</th><th>Year 3</th><th>Year 4</th><th>Year 5</th></tr>
<tr><td>B_0</td><td>x_5
0.25%</td><td>x_6
0.25%</td><td>x_7
0.25%</td><td>x_8
0.25%</td><td>x_9
0.25%</td></tr>
<tr><td>B_1</td><td colspan="4">x_1
2%</td><td></td></tr>
<tr><td>B_2</td><td></td><td></td><td></td><td></td><td>x_2
4%</td></tr>
<tr><td>B_3</td><td></td><td colspan="4">x_3
6%</td></tr>
<tr><td>B_4</td><td></td><td colspan="3">x_4
6%</td><td></td></tr>
</table>

Let x_k denotes the amount of investment according to the above table and $r_k = (1 + \rho_k/100)^{m_k}$ denotes the net return of B_k.

Profit after Year 5 = $r_2 x_2 + r_3 x_3 + r_9 x_9$.

`linprog` for sequential investment planning (2)

<table>
<tr><th></th><th>Year 1</th><th>Year 2</th><th>Year 3</th><th>Year 4</th><th>Year 5</th></tr>
<tr><td>B_0</td><td>x_5
0.25%</td><td>x_6
0.25%</td><td>x_7
0.25%</td><td>x_8
0.25%</td><td>x_9
0.25%</td></tr>
<tr><td>B_1</td><td colspan="4">x_1
2%</td><td></td></tr>
<tr><td>B_2</td><td></td><td></td><td></td><td></td><td>x_2
4%</td></tr>
<tr><td>B_3</td><td></td><td colspan="4">x_3
6%</td></tr>
<tr><td>B_4</td><td></td><td colspan="3">x_4
6%</td><td></td></tr>
</table>

Let x_k denotes the amount of investment according to the above table and $r_k = (1 + \rho_k/100)^{m_k}$ denotes the net return of B_k.

Profit after Year 5 = $r_2 x_2 + r_3 x_3 + r_9 x_9$. $\leftarrow$ maximized.

`linprog` for sequential investment planning (2)

	Year 1	Year 2	Year 3	Year 4	Year 5
B_0	x_5 0.25%	x_6 0.25%	x_7 0.25%	x_8 0.25%	x_9 0.25%
B_1	x_1 2%				
B_2					x_2 4%
B_3		x_3 6%			
B_4		x_4 6%			

Let x_k denotes the amount of investment according to the above table and $r_k = (1 + \rho_k/100)^{m_k}$ denotes the net return of B_k.

Profit after Year 5 = $r_2 x_2 + r_3 x_3 + r_9 x_9$. $\leftarrow$ maximized.

<u>Constraints:</u>

Initial capitol: $x_1 + x_5 = 1000$

`linprog` for sequential investment planning (2)

<table>
<tr><th></th><th>Year 1</th><th>Year 2</th><th>Year 3</th><th>Year 4</th><th>Year 5</th></tr>
<tr><td>B_0</td><td>x_5
0.25%</td><td>x_6
0.25%</td><td>x_7
0.25%</td><td>x_8
0.25%</td><td>x_9
0.25%</td></tr>
<tr><td>B_1</td><td colspan="4">x_1
2%</td><td></td></tr>
<tr><td>B_2</td><td></td><td></td><td></td><td></td><td>x_2
4%</td></tr>
<tr><td>B_3</td><td></td><td colspan="4">x_3
6%</td></tr>
<tr><td>B_4</td><td></td><td colspan="3">x_4
6%</td><td></td></tr>
</table>

Let x_k denotes the amount of investment according to the above table and $r_k = (1 + \rho_k/100)^{m_k}$ denotes the net return of B_k.

Profit after Year 5 = $r_2 x_2 + r_3 x_3 + r_9 x_9$. $\leftarrow$ maximized.

Constraints:

Initial capitol: $x_1 + x_5 = 1000$

Year 2 capitol: $x_3 + x_4 + x_6 = r_5 x_5$

`linprog` **for sequential investment planning (2)**

	Year 1	Year 2	Year 3	Year 4	Year 5
B_0	x_5 0.25%	x_6 0.25%	x_7 0.25%	x_8 0.25%	x_9 0.25%
B_1	x_1 2%				
B_2					x_2 4%
B_3		x_3 6%			
B_4		x_4 6%			

Let x_k denotes the amount of investment according to the above table and $r_k = (1 + \rho_k/100)^{m_k}$ denotes the net return of B_k.

Profit after Year 5 = $r_2 x_2 + r_3 x_3 + r_9 x_9$. $\leftarrow$ maximized.

Constraints:

Initial capitol: $x_1 + x_5 = 1000$

Year 2 capitol: $x_3 + x_4 + x_6 = r_5 x_5$

Year 3 capitol: $x_7 = r_6 x_6$

Year 4 capitol: $x_8 = r_7 x_7$

`linprog` for sequential investment planning (2)

<table>
<tr><th></th><th>Year 1</th><th>Year 2</th><th>Year 3</th><th>Year 4</th><th>Year 5</th></tr>
<tr><td>B_0</td><td>x_5
0.25%</td><td>x_6
0.25%</td><td>x_7
0.25%</td><td>x_8
0.25%</td><td>x_9
0.25%</td></tr>
<tr><td>B_1</td><td colspan="4">x_1
2%</td><td></td></tr>
<tr><td>B_2</td><td></td><td></td><td></td><td></td><td>x_2
4%</td></tr>
<tr><td>B_3</td><td></td><td colspan="4">x_3
6%</td></tr>
<tr><td>B_4</td><td></td><td colspan="3">x_4
6%</td><td></td></tr>
</table>

Let x_k denotes the amount of investment according to the above table and $r_k = (1 + \rho_k/100)^{m_k}$ denotes the net return of B_k.

Profit after Year 5 = $r_2 x_2 + r_3 x_3 + r_9 x_9$. $\leftarrow$ maximized.

<u>Constraints:</u>

Initial capitol: $x_1 + x_5 = 1000$

Year 2 capitol: $x_3 + x_4 + x_6 = r_5 x_5$

Year 3 capitol: $x_7 = r_6 x_6$

Year 4 capitol: $x_8 = r_7 x_7$

Year 5 capitol: $x_2 + x_9 = r_1 x_1 + r_4 x_4 + r_8 x_8$

The tool `fmincon` solves a structured linearly constrained quadratic program that takes the form

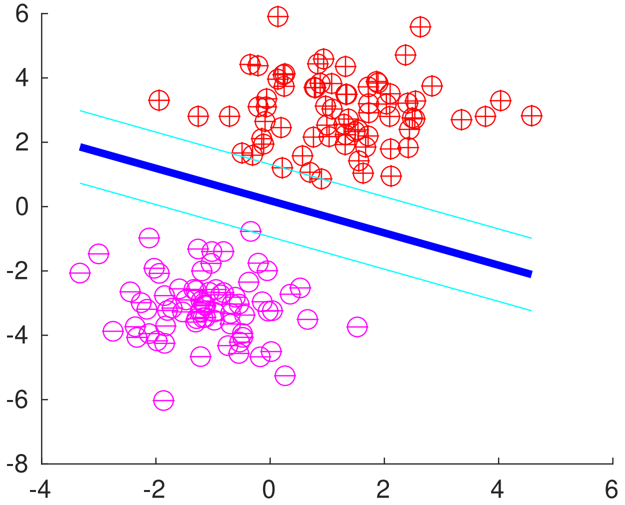
$$\begin{cases} \min & \frac{1}{2}x^\top Hx + f^\top x \\ \text{s.t.} & Ax \le b \\ & Aeq \cdot x = beq \\ & lb \le x \le ub. \end{cases}$$

by the command

```
quadprog(H,f,A,b,Aeq,beq,lb,ub).
```

Ex. Let's try to use `quadprog` to find an optimal separating plane in the SVM model.

`quadprog` for SVMs



The tool `fmincon` solves a structured linear program that takes the form

$$\begin{cases} \min & f^\top x \\ \text{s.t.} & Ax \le b \\ & Aeq \cdot x = beq \\ & lb \le x \le ub \\ & x_j \in \mathbb{Z}, \quad \text{for } j \in I \subset \{1, \cdots, n\} \end{cases}$$

by the command

```
intlinprog(f,I,A,b,Aeq,beq,lb,ub).
```

Ex. Let's try `intlinprog` with a service provider problem.

`linprog`/`intlinprog` for service provider problems

General statement.

A service company that has N hubs requires to serve its M clients.

`linprog`/`intlinprog` for service provider problems

General statement.

A service company that has N hubs requires to serve its M clients.

Each hub i has an ability to provide m_i of commodities.

linprog/intlinprog for service provider problems

General statement.

A service company that has N hubs requires to serve its M clients.

Each hub i has an ability to provide m_i of commodities.

Each client j has his/her demand r_i that needs to be satisfied.

`linprog`/`intlinprog` for service provider problems

General statement.

A service company that has N hubs requires to serve its M clients.
Each hub i has an ability to provide m_i of commodities.
Each client j has his/her demand r_i that needs to be satisfied.
This company aims to supply all the demand at the minimum cost, by sending y_{ij} commodities from the hub i to the client j that would then inflict some fixed service cost c_{ij}.

General statement.

A service company that has N hubs requires to serve its M clients.
Each hub i has an ability to provide m_i of commodities.
Each client j has his/her demand r_i that needs to be satisfied.
This company aims to supply all the demand at the minimum cost, by sending y_{ij} commodities from the hub i to the client j that would then inflict some fixed service cost c_{ij}.

Remark: This problem statement is generally linear, but more often than not, involves integrality constraints.

Total cost = $\sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} y_{ij}$

General statement.

A service company that has N hubs requires to serve its M clients.

Each hub i has an ability to provide m_i of commodities.

Each client j has his/her demand r_i that needs to be satisfied.

This company aims to supply all the demand at the minimum cost, by sending y_{ij} commodities from the hub i to the client j that would then inflict some fixed service cost c_{ij}.

Remark: This problem statement is generally linear, but more often than not, involves integrality constraints.

Total cost = $\sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} y_{ij} \leftarrow$ minimized.

General statement.

A service company that has N hubs requires to serve its M clients.
Each hub i has an ability to provide m_i of commodities.
Each client j has his/her demand r_i that needs to be satisfied.
This company aims to supply all the demand at the minimum cost, by sending y_{ij} commodities from the hub i to the client j that would then inflict some fixed service cost c_{ij}.

Remark: This problem statement is generally linear, but more often than not, involves integrality constraints.

Total cost = $\sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} y_{ij} \leftarrow$ minimized.
Constraints:

General statement.

A service company that has N hubs requires to serve its M clients.

Each hub i has an ability to provide m_i of commodities.

Each client j has his/her demand r_i that needs to be satisfied.

This company aims to supply all the demand at the minimum cost, by sending y_{ij} commodities from the hub i to the client j that would then inflict some fixed service cost c_{ij}.

Remark: This problem statement is generally linear, but more often than not, involves integrality constraints.

Total cost = $\sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} y_{ij} \leftarrow$ minimized.

Constraints:

Hub availability: At each i, $\sum_{j=1}^{M} y_{ij} \leq m_i$.

`linprog/intlinprog` for service provider problems

General statement.

A service company that has N hubs requires to serve its M clients.
Each hub i has an ability to provide m_i of commodities.
Each client j has his/her demand r_i that needs to be satisfied.
This company aims to supply all the demand at the minimum cost, by sending y_{ij} commodities from the hub i to the client j that would then inflict some fixed service cost c_{ij}.

Remark: This problem statement is generally linear, but more often than not, involves integrality constraints.

Total cost = $\sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} y_{ij} \leftarrow$ minimized.

Constraints:

Hub availability: At each i, $\sum_{j=1}^{M} y_{ij} \leq m_i$.

Demand satisfactory: At each j, $\sum_{i=1}^{N} y_{ij} \geq r_j$

General statement.

A service company that has N hubs requires to serve its M clients.
Each hub i has an ability to provide m_i of commodities.
Each client j has his/her demand r_i that needs to be satisfied.
This company aims to supply all the demand at the minimum cost, by sending y_{ij} commodities from the hub i to the client j that would then inflict some fixed service cost c_{ij}.

Remark: This problem statement is generally linear, but more often than not, involves integrality constraints.

Total cost = $\sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} y_{ij} \leftarrow$ minimized.

Constraints:

Hub availability: At each i, $\sum_{j=1}^{M} y_{ij} \leq m_i$.

Demand satisfactory: At each j, $\sum_{i=1}^{N} y_{ij} \geq r_j$

Integrality constraints: $y_{ij} \in \mathbb{Z}$ for $(i, j) \in I$.

`linprog`/`intlinprog` for service provider problems

General statement.

A service company that has N hubs requires to serve its M clients.
Each hub i has an ability to provide m_i of commodities.
Each client j has his/her demand r_i that needs to be satisfied.
This company aims to supply all the demand at the minimum cost, by sending y_{ij} commodities from the hub i to the client j that would then inflict some fixed service cost c_{ij}.

Remark: This problem statement is generally linear, but more often than not, involves integrality constraints.

Total cost = $\sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij}y_{ij} \leftarrow$ minimized.

Constraints:

Hub availability: At each i, $\sum_{j=1}^{M} y_{ij} \leq m_i$.
Demand satisfactory: At each j, $\sum_{i=1}^{N} y_{ij} \geq r_j$
Integrality constraints: $y_{ij} \in \mathbb{Z}$ for $(i,j) \in I$.
Other problem-specific constraints.

Ex. A transportation company has 5 hubs, namely Samut Prakan (city), Pathum Thani, Chachoengsao, Nakhon Nayok, and Rayong (city), that stock empty containers. This company needs to provide these containers to the 6 ports in Samut Prakan (Bang Pu), Samut Prakan (Suvarnabhumi), Prachin Buri, Sa Kaeo, Chonburi, and Rayong (Pluak Daeng).

`intlinprog` in logistics (1)

Ex. A transportation company has 5 hubs, namely Samut Prakan (city), Pathum Thani, Chachoengsao, Nakhon Nayok, and Rayong (city), that stock empty containers. This company needs to provide these containers to the 6 ports in Samut Prakan (Bang Pu), Samut Prakan (Suvarnabhumi), Prachin Buri, Sa Kaeo, Chonburi, and Rayong (Pluak Daeng).

The container stocks (x_i) and port demands (r_j) are as follows:

	Empty containers
Samut Prakan (city)	10
Pathum Thani	8
Chachoengsao	8
Nakhon Nayok	7
Rayong (city)	9

	Container demand
Samut Prakan (Bang Pu)	8
Samut Prakan (Suvarnabhumi)	7
Prachin Buri	7
Sa Kaeo	6
Chonburi	6
Rayong (Pluak Daeng)	7

`intlinprog` in logistics (3)

The containers are transported by lorries (the company possesses enough lorries). Each lorry carries up to 2 containers.

`intlinprog` in logistics (3)

The containers are transported by lorries (the company possesses enough lorries). Each lorry carries up to 2 containers.

The transportation cost bore by each lorry is directly proportional to the distance with the cost/km. = THB300. The (rounded) distances d_{ij} from hub i to port j are given in the following table.

		$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$
		BP	SVP	PB	SK	CB	PD
$i = 1$	SP	15	20	148	175	108	120
$i = 2$	PT	78	60	143	194	162	173
$i = 3$	CCS	100	92	69	95	72	83
$i = 4$	NN	132	107	59	110	153	161
$i = 5$	RY	153	154	186	213	54	47

`intlinprog` in logistics (3)

We can now formalize our optimization problem.

`intlinprog` in logistics (3)

We can now formalize our optimization problem.

Cost = $\sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} y_{ij}$

We can now formalize our optimization problem.

Cost = $\sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} y_{ij} \leftarrow$ minimized.
$c_{ij} = 300 d_{ij}$,
(y_{ij} the number of lorries sent from hub i to port j.)

We can now formalize our optimization problem.

Cost = $\sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} y_{ij} \leftarrow$ minimized.
$c_{ij} = 300 d_{ij}$,
(y_{ij} the number of lorries sent from hub i to port j.)

Constraints.

We can now formalize our optimization problem.

Cost = $\sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} y_{ij} \leftarrow$ minimized.
$c_{ij} = 300 d_{ij}$,
(y_{ij} the number of lorries sent from hub i to port j.)

Constraints.

Hub availability: At each i, $\sum_{j=1}^{M} x_{ij} \leq m_i$.

`intlinprog` in logistics (3)

We can now formalize our optimization problem.

Cost = $\sum_{i=1}^{N} \sum_{j=1}^{M} c_{ij} y_{ij} \leftarrow$ minimized.
$c_{ij} = 300 d_{ij}$,
(y_{ij} the number of lorries sent from hub i to port j.)

<u>Constraints.</u>

Hub availability: At each i, $\sum_{j=1}^{M} x_{ij} \leq m_i$.

Demand satisfactory: At each j, $\sum_{i=1}^{N} x_{ij} \geq r_j$

We can now formalize our optimization problem.

Cost = $\sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij}y_{ij} \leftarrow$ minimized.
$c_{ij} = 300d_{ij}$,
(y_{ij} the number of lorries sent from hub i to port j.)

Constraints.

Hub availability: At each i, $\sum_{j=1}^{M} x_{ij} \leq m_i$.

Demand satisfactory: At each j, $\sum_{i=1}^{N} x_{ij} \geq r_j$

Integrality constraints: $x_{ij}, y_{ij} \in \mathbb{Z}$ for all i, j.

We can now formalize our optimization problem.

Cost = $\sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij}y_{ij} \leftarrow$ minimized.
$c_{ij} = 300d_{ij}$,
(y_{ij} the number of lorries sent from hub i to port j.)

Constraints.

Hub availability: At each i, $\sum_{j=1}^{M} x_{ij} \leq m_i$.
Demand satisfactory: At each j, $\sum_{i=1}^{N} x_{ij} \geq r_j$
Integrality constraints: $x_{ij}, y_{ij} \in \mathbb{Z}$ for all i, j.
Lorry capacity constraint: $x_{ij} \leq 2y_{ij}$.

`intlinprog` in logistics (3)

We can now formalize our optimization problem.

Cost = $\sum_{i=1}^{N}\sum_{j=1}^{M} c_{ij}y_{ij} \leftarrow$ minimized.

$c_{ij} = 300 d_{ij}$,

(y_{ij} the number of lorries sent from hub i to port j.)

Constraints.

Hub availability: At each i, $\sum_{j=1}^{M} x_{ij} \leq m_i$.

Demand satisfactory: At each j, $\sum_{i=1}^{N} x_{ij} \geq r_j$

Integrality constraints: $x_{ij}, y_{ij} \in \mathbb{Z}$ for all i, j.

Lorry capacity constraint: $x_{ij} \leq 2y_{ij}$.

Non-negativity constraint: $x_{ij}, y_{ij} \geq 0$.

Metaheuristics

Heuristics & Metheuristics

Both heuristic and metaheuristic algorithms are *based on intuition, metaphor and natural experience* rather than systematical construction like the gradient descent or Newton's methods.

Heuristics & Metheuristics

Both heuristic and metaheuristic algorithms are *based on intuition, metaphor and natural experience* rather than systematical construction like the gradient descent or Newton's methods. Many of these methods do not have any supporting theory.

The goal of such approach is to find a "good enough" solution that meets certain criteria. These algorithms are claimed to be *global* optimization algorithms.

Heuristics & Metheuristics

Both heuristic and metaheuristic algorithms are *based on intuition, metaphor and natural experience* rather than systematical construction like the gradient descent or Newton's methods. Many of these methods do not have any supporting theory.

The goal of such approach is to find a "good enough" solution that meets certain criteria. These algorithms are claimed to be *global* optimization algorithms.

We have to be careful with the term *global* in this context, as it rather means a local solution that is better than some other local solutions than the true global optimum that one would imagined of.

Heuristics vs Metaheuristics

Heuristics refer to algorithms where the structures and contexts of a specific problem were taken into account, like the traveling salesman and knapsack problems.

Heuristics vs Metaheuristics

Heuristics refer to algorithms where the structures and contexts of a specific problem were taken into account, like the traveling salesman and knapsack problems.

Metaheuristics, on the other hand, are designed to work with any problems without using any problem-specific knowledge.

Heuristics vs Metaheuristics

Heuristics refer to algorithms where the structures and contexts of a specific problem were taken into account, like the traveling salesman and knapsack problems.

Metaheuristics, on the other hand, are designed to work with any problems without using any problem-specific knowledge. Well-known algorithms of this category are

- Particle Swarm Optimization (PSO)*
- Genetic Algorithm (GA)*
- Ant Colony Optimization
- Bee Colony Optimization
- Tabu Search
- etc.

Classical vs Metaheuristic Algorithms

Usually, classical iterative algorithms based on the gradient are more effective. The downside is that it does not always work and it requires computing the gradients or even the Hessians.

Classical vs Metaheuristic Algorithms

Usually, classical iterative algorithms based on the gradient are more effective. The downside is that it does not always work and it requires computing the gradients or even the Hessians.

In some applications, we need to minimize a function that we have no knowledge about its gradient or Hessian. And the metaheuristic algorithms are the only option here to derive an “acceptable” solution.

Classical vs Metaheuristic Algorithms

Usually, classical iterative algorithms based on the gradient are more effective. The downside is that it does not always work and it requires computing the gradients or even the Hessians.

In some applications, we need to minimize a function that we have no knowledge about its gradient or Hessian. And the metaheuristic algorithms are the only option here to derive an “acceptable” solution.

Applications that usually require metaheuristics are (hyper)parameter estimations.

PSO

PARTICLE SWARM OPTIMIZATION.

Initialization:
Generate an initial population of particles (points in the search space).
While: Not satisfied;
Evaluation: Evaluate the objective value at each candidate.
Local update: Each particle memorizes its own best-known position (local best). The new position is then compared with the previous local best.
Global update: All local bests are compared and the best one is memorized as the global best.
Movement: Each particle updates its position based on the local and global bests.

PARTICLE SWARM OPTIMIZATION.

Initialization:
Generate an initial population of particles (points in the search space).
While: Not satisfied;
Evaluation: Evaluate the objective value at each candidate.
Local update: Each particle memorizes its own best-known position (local best). The new position is then compared with the previous local best.
Global update: All local bests are compared and the best one is memorized as the global best.
Movement: Each particle updates its position based on the local and global bests.

GA

GENETIC ALGORITHMS.

Initialization:
Generate an initial population of candidate solutions. Each candidate is coded as a finite sequence of *genes*, called a *chromosome*.
While: Not satisfied;
Evaluation: Compute the fitness value of each candidates using a choice of fitness function.
Selection: Select the best candidates based on their fitness values.
Crossover: The selected candidates are paired and new candidates are generated by combining the chromosomes of their parents using the chosen crossover method.
Mutation: A percentage of the new offsprings are randomly mutated by introducing a small random changes.
Replacement: The new population is created from the few fittest candidates and the newly created candidates.

GENETIC ALGORITHMS.

Initialization:
Generate an initial population of candidate solutions. Each candidate is coded as a finite sequence of *genes*, called a *chromosome*.
While: Not satisfied;
Evaluation: Compute the fitness value of each candidates using a choice of fitness function.
Selection: Select the best candidates based on their fitness values.
Crossover: The selected candidates are paired and new candidates are generated by combining the chromosomes of their parents using the chosen crossover method.
Mutation: A percentage of the new offsprings are randomly mutated by introducing a small random changes.
Replacement: The new population is created from the few fittest candidates and the newly created candidates.

Conclusion and remarks

Conclusion and remarks

- To hand-calculate the constrained optimum, we solve it through the KKT conditions.
- Most of the classical gradient-based iterative methods are based on the KKT conditions.
- GA works best with discrete search spaces.
- PSO works best with continuous search spaces.
- Metaheuristics do not require gradient information.
- No best-for-all algorithm exists.

Thank you.