Greedy algorithm for local heating problem

Description:

In the following we present a mathematical description of the studied model and a summary of the results of this paper.

First of all, we consider a discrete time model for the considered problem, meaning that we split the planning period into T time intervals of the same length resulting in a set $\mathcal{T} = \{1, \dots, T\}$ of time intervals. In this paper, the letter t is always used as an index of time intervals.

For the heating system, we consider a simple converter which has only two states: In every time interval the converter is either turned on or turned off. The amount of produced heat during one time interval in which the converter is turned on is denoted by H. If the converter is turned off, then it consumes and produces no energy. Let $x_t \in \{0,1\}$ be the variable indicating whether the converter is running in time interval $t \in \mathcal{T}$ or not. Furthermore, if the converter is running, then it consumes some amount of electricity which costs P_t in time interval $t \in \mathcal{T}$. In another words, P_t is the price for running the converter in time interval $t \in \mathcal{T}$. Summarizing, the objective of the planning problem is minimizing the cost for producing the heat, which is given by $\sum_{t\in\mathcal{T}} P_t x_t$.

Coupled to the heating system is a buffer. The state of charge of the buffer in the beginning of time interval $t \in \mathcal{T}$ is denoted by s_t and represents the amount of heat in the buffer. Note that s_{T+1} is the state of charge at the end of planning period. Based of the physical properties of the buffer, the state of charge s_t is limited by a lower bound L_t and an upper bound U_t. These two bounds are usually constant over time: the upper bound U_t is the capacity of buffer and the lower bound L_t is zero. But it may be useful to allow different values, e.g. a given initial state of charge can be modelled by setting L_1 and U_1 equal to the initial state. In this paper, we always assume that $L_1 = U_1$, meaning that the initial state of charge s_1 is fixed.

The (predicted) amount of consumed heat by the inhabitants of the house from the boiler during time interval $t \in \mathcal{T}$ is denoted by D_t. This amount is assumed to be given and is called the demand. In this paper, we study off-line version of the problem, so we assume that both the demands D_t and also the prices P_t are given for the whole planning period already at time the beginning of the planning period.

The variables x_t specifying the operation of the converter and the states of charge of the buffer s_t are restricted by the following constraints.

$$s_{t+1} = s_t + Hx_t - D_t \quad \text{for} \quad t \in \mathcal{T} \tag{1}$$

$$L_t \le s_t \le U_t \quad \text{for} \quad t \in \{1, \dots, T+1\} \tag{2}$$

$$x_t \in \{0,1\} \quad \text{for} \quad t \in \mathcal{T} \tag{3}$$

Algorithm Design and Proof of Correctness

In [3] a reformulation of the problem presented in Section 1.1 is given, which enables a better presentation and analysis of our algorithm. For sake of completeness, we give this reformulation in this section. We show that conditions (1) and (2) can be replaced by one condition (8).

First, we expand the recurrence formula (1) into an explicit equation

$$s_{t+1} = s_1 + \sum_{i=1}^{t} Hx_i - \sum_{i=1}^{t} D_i.$$

Since we assume that the initial state of charge satisfies $s_1 = L_1 = U_1$, we can replace s_1 by L_1 and substitute this into inequalities (2), leading to

$$L_{t+1} \leq L_1 + H\sum_{i=1}^{t} x_i - \sum_{i=1}^{t} D_i \leq U_{t+1}$$

which can be rewritten as

$$\frac{L_{t+1} - L_1 + \sum_{i=1}^{t} D_i}{H} \leq \sum_{i=1}^{t} x_i \leq \frac{U_{t+1} - L_1 + \sum_{i=1}^{t} D_i}{H}.$$

Since the sum $\sum_{i=1}^{t} x_i$ is an integer between 0 and t we obtain the following simple constrains for this sums

$$A'_t \leq \sum_{i=1}^{t} x_i \leq B'_t \text{ for } t \in \mathcal{T} \tag{4}$$

where

$$A'_t = \max\left\{0, \left\lceil \frac{L_{t+1} - L_1 + \sum_{i=1}^{t} D_i}{H} \right\rceil\right\}, \tag{5}$$

$$B'_t = \min\left\{t, \left\lfloor \frac{U_{t+1} - L_1 + \sum_{i=1}^{t} D_i}{H} \right\rfloor\right\}. \tag{6}$$

The values of A'_t and B'_t for every $t \in \mathcal{T}$ can easily be computed in time $\mathcal{O}(T)$. In the rest of this section we study properties of sequences $A'_1, \ldots, A'_T$ and $B'_1, \ldots, B'_T$ which are shortly denoted by $(A'_t)_t$ and $(B'_t)_t$, respectively.

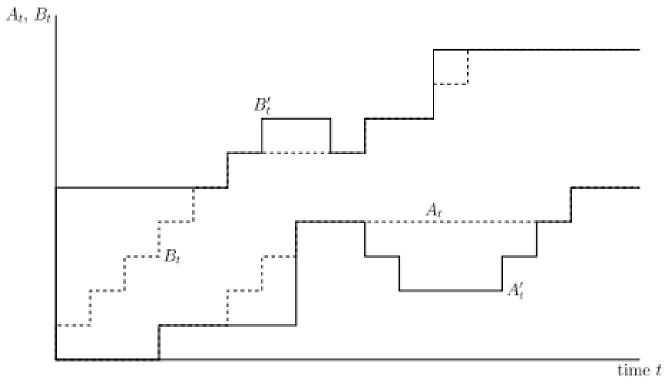
Observe that the sequence of partial sums $\sum_{i=1}^{t} x_i$ for $t = 1, \ldots, T$ is non-decreasing and the difference of two consecutive elements is at most 1. We say that a sequence $(Z_t)_t$ of $T+1$ integers $Z_0, Z_1, \ldots, Z_T$ satisfies (7) if

$$\begin{aligned} &Z_0 = 0, \\ &Z_{t-1} \leq Z_t \leq Z_{t-1} + 1 \text{ for all } t \in \mathcal{T}. \end{aligned} \tag{7}$$

We show that we can replace parameters A'_t and B'_t by parameters A_t and B_t such that sequences $(A_t)_t$ and $(B_t)_t$ satisfy (7) and the binary variables x_t satisfy (4) if and only if they satisfy

$$A_t \leq \sum_{i=1}^{t} x_i \leq B_t \text{ for } t \in \mathcal{T}. \tag{8}$$

The idea of the replacement is presented in Fig. 1. It is easy to see that for this example every solution $(x_t)_t$ that satisfies (4) also satisfies (8) and vice versa. In [3] an algorithm to obtain required sequences $(A_t)_t$ and $(B_t)_t$ is given.



4. Greedy algorithm

In this section we present a greedy algorithm to the problem of fulfilling the heat demand with minimal cost which is based on solution that can be formulated as

$$\begin{aligned}
&\text{Minimize} && \sum_{t\in\mathcal{T}} P_t x_t \\
&\text{such that } A_t \le \sum_{i=1}^{t} x_i \le B_t && \text{for } t \in \mathcal{T} \\
&x_t \in \{0,1\} && \text{for } t \in \mathcal{T}
\end{aligned} \tag{9}$$

For the following, we assume that the bounds A_t and B_t are precomputed, meaning that the sequences $(A_t)_t$ and $(B_t)_t$ satisfy (7).

The first natural question is under which conditions problem (9) has a feasible solution. An obvious necessary condition for the existence of a feasible solution x_t is that $A_t \le B_t$ for every $t \in \mathcal{T}$. This condition is also sufficient, since in this case $x_t = A_t - A_{t-1}$ for $t \in \mathcal{T}$ gives a feasible solution. Summarizing, we get the following lemma.

Lemma 4.1. *The problem (9) has a feasible solution if and only if*

$$A_t \le B_t \textit{ for every } t \in \mathcal{T}. \tag{10}$$

Since the condition (10) can easily be evaluated in linear time, we assume in the remainder of the paper that the problem (9) has a feasible solution. To solve the problem, we use the classical greedy approach. First, the time intervals are sorted by prices P_t. Then, time intervals are processed in order of increasing prices and the converter is turned on in time interval $t^\star \in \mathcal{T}$, if there exists a feasible solution satisfying $x_{t^\star} = 1$. Such a feasible solution implies

$$A_{t^\star-1} \le \sum_{i=1}^{t^\star-1} x_i < \sum_{i=1}^{t^\star} x_i \le B_{t^\star}$$

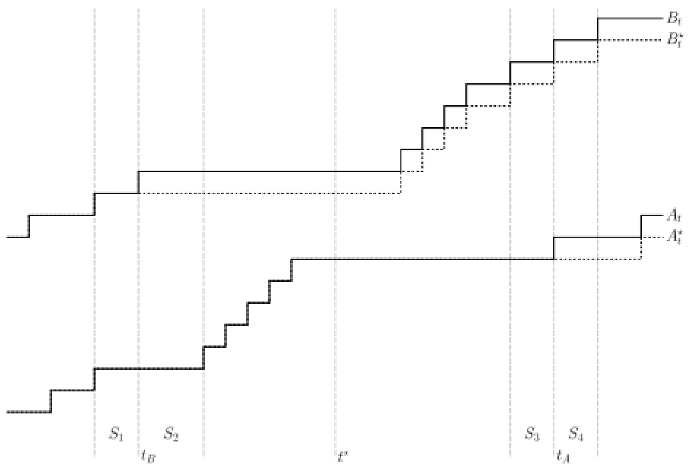


Fig. 2. Sequences $(A_t)_t$, $(B_t)_t$, $(A_t^\star)_t$ and $(B_t^\star)_t$ according to Lemma 4.3.

Lemma 4.2. *If the problem (9) has a feasible solution $(x_t)_t$ satisfying $x_{t^\star} = 1$ for a given $t^\star \in \mathcal{T}$, then the inequality $A_{t^\star-1} < B_{t^\star}$ holds.*

Note that the condition $A_{t^\star-1} < B_{t^\star}$ in the lemma is well-defined also for $t^\star = 1$ since we consider that $A_0 = 0$. The condition $A_{t^\star-1} < B_{t^\star}$ in Lemma 4.2 is also sufficient (if (10) is satisfied) and is the base for the presented approach. At this point, we do not give a formal proof of this implication, but the proof follows from the further considerations and lemmas (e.g. 4.3 or 4.5).

The greedy algorithm starts with the (infeasible) solution $x_t = 0$ for every $t \in T$. It finds the cheapest time interval $t^\star$ satisfying $A_{t^\star-1} < B_{t^\star}$ and it sets $x_{t^\star} := 1$. After choosing $t^\star$ and setting $x_{t^\star} = 1$, the condition $A_{t-1} < B_t$ do no longer imply that x_t can be set to 1. We first have to adopt the values of sequences $(A_t)_t$ and $(B_t)_t$ to incorporate the choice $x_{t^\star} = 1$. The following lemma specifies how the values of $(A_t)_t$ and $(B_t)_t$ have to be updated in every step and shows that this update is correct.

The lemma finds the time interval t_A where the sequence $(A_t)_t$ has the first increasing value after the time interval $t^\star$, and the time interval t_B where the sequence $(B_t)_t$ has the last increasing value before the time interval $t^\star$. These increases at time intervals t_A and t_B are removed from sequences $(A_t)_t$ and $(B_t)_t$, respectively; that is values of sequences are decreased by one after these time intervals (see Fig. 2).

Lemma 4.3. *Let $t^\star \in \mathcal{T}$ satisfy $A_{t^\star-1} < B_{t^\star}$, and let*

$$t_A = 1 + \max\{t \in \mathcal{T}\,;\ A_t = A_{t^\star-1}\} \qquad t_B = \min\{t \in \mathcal{T}\,;\ B_t = B_{t^\star}\}$$

$$A_t^\star = \begin{cases} A_t & \text{if } t < t_A \\ A_t - 1 & \text{if } t \geq t_A \end{cases} \qquad B_t^\star = \begin{cases} B_t & \text{if } t < t_B \\ B_t - 1 & \text{if } t \geq t_B. \end{cases}$$

Then, sequences $(A_t^\star)_t$ and $(B_t^\star)_t$ satisfy (7) and for every 0–1 sequence $(x_t)_t$ with $x_{t^\star} = 1$ the condition (9) holds if and only if it holds

$$A_t^\star \leq \sum_{\substack{i=1 \\ i \neq t^\star}}^{t} x_i \leq B_t^\star \text{ for every } t \in \mathcal{T}. \tag{11}$$

Before we prove Lemma 4.3 we first explain why we need the equivalence in the lemma. The idea is that sequences $(A_t^\star)_t$ and $(B_t^\star)_t$ form another instance of the problem (9) where the time interval $t^\star$ is skipped. When we find an optimal solution $(x_t)_t$ for the instance of the problem 4 with $(A_t^\star)_t$ and $(B_t^\star)_t$ where $t \in \mathcal{T} \setminus \{t^\star\}$, then we obtain an optimal solution for the original instance with $(A_t)_t$ and $(B_t)_t$ where $t \in \mathcal{T}$ by setting $x_{t^\star} := 1$. Note that Lemma 4.3 does not guarantee the optimality but only the feasibility and that time interval $t^\star$ needs not to be removed in the greedy algorithm since the greedy algorithm process every time interval at most once.

Proof of Lemma 4.3. Note that if $A_{t^\star-1} = A_T$, then $t_A = T+1$ and the sequences $(A_t)_t$ and $(A_t^\star)_t$ are the same. Otherwise, if $A_{t^\star-1} < A_T$, then $A_{t_A-1} = A_{t_A} - 1$. Furthermore, it holds that $B_{t_B-1} = B_{t_B} - 1$. Therefore, the sequences $(A_t^\star)_t$ and $(B_t^\star)_t$ differ from sequences $(A_t)_t$ and $(B_t)_t$ by removing (at most) one step of the step function. This implies that they satisfy (7).

In order to prove the second part, let $(x_t)_t$ be a 0–1 sequence with $x_{t^\star} = 1$. For such a sequence, the condition (9) is equivalent to

$$\begin{aligned} A_t &\le \sum_{\substack{i=1\\ i\neq t^\star}}^{t} x_i \le B_t && \text{for every } t < t^\star \text{ and} \\ A_t - 1 &\le \sum_{\substack{i=1\\ i\neq t^\star}}^{t} x_i \le B_t - 1 && \text{for every } t \ge t^\star. \end{aligned} \tag{12}$$

Thus, it remains to prove that conditions (11) and (12) are equivalent.

First, we consider the lower bounds. Observe that the lower bounds of (11) and (12) only differ for time intervals $t \in \mathcal{T}$ with $t^\star \le t < t_A$. For such t it holds that $A_t^\star = A_t$ and, thus, we only have to prove that (12) implies (11) since the lower bound in (11) is stronger. However, the implication follows directly from

$$A_t^\star = A_t = A_{t^\star-1} \le \sum_{i=1}^{t^\star-1} x_i \le \sum_{\substack{i=1\\ i\neq t^\star}}^{t} x_i.$$

The upper bounds can be considered similar. Observe that the upper bounds of (11) and (12) differ only for time intervals $t \in \mathcal{T}$ satisfying $t_B \le t < t^\star$. For such t it holds that $B_t^\star = B_t - 1$ and we only have to prove that (12) implies (11) since the upper bound in (11) is stronger. The implication follows from

$$\sum_{\substack{i=1\\ i\neq t^\star}}^{t} x_i \le \sum_{\substack{i=1\\ i\neq t^\star}}^{t^\star} x_i \le B_{t^\star} - 1 = B_t - 1 = B_t^\star. \quad \square$$

In practice, the price of electricity is usually positive. However, we present a greedy algorithm which also works if the price P_t is negative for some $t \in \mathcal{T}$. If all prices are non-negative, then without loss of generality we can assume that $A_T = B_T$ since there is an optimal solution which turns the converter on only A_T-times (that is, there exists an optimal solution with $\sum_{t\in\mathcal{T}} x_t = A_T$). In the general case where prices can be negative, the value of the objective function may be improved by turning the converter on more often. In the later case, we need to extend the condition $A_{t^\star-1} < B_{t^\star}$ of the greedy algorithm to a condition which also considers negative prices. The new condition is

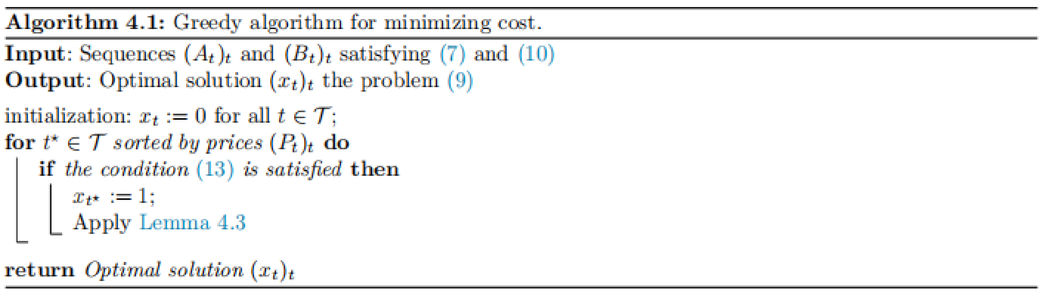
$$A_{t^\star-1} < B_{t^\star} \text{ and } (A_{t^\star-1} < A_T \text{ or } P_{t^\star} < 0). \tag{13}$$

Note, that the condition $A_{t^\star-1} < B_{t^\star}$ is already necessary for setting $x_{t^\star} = 1$ by Lemma 4.2. If $A_{t^\star-1} = A_T$ then the total minimal number of runs of the converter has to be reached already before the time interval $t^\star$, so the lower bound $(A_t)_t$ does not force the converter on in the time interval $t^\star$. In this case, it is obvious that an optimal solution satisfies $x_{t^\star} = 0$ unless the price $P_{t^\star}$ is negative.

As already mentioned above, we can assume that $A_T = B_T$ if all prices are non-negative. In this case the condition $A_{t^\star-1} < B_{t^\star}$ implies $A_{t^\star-1} < A_T$, so the condition (13) reduces to the condition $A_{t^\star-1} < B_{t^\star}$.

The greedy algorithm is summarized in Algorithm 4.1. In the following, mathematical induction is used to prove that this greedy algorithm finds an optimal solution. The following lemma provides the base of the induction.

Pseudo code



Algorithm 4.1: Greedy algorithm for minimizing cost.

```
Input: Sequences $(A_t)_t$ and $(B_t)_t$ satisfying (7) and (10)
Output: Optimal solution $(x_t)_t$ the problem (9)
initialization: $x_t := 0$ for all $t \in \mathcal{T}$;
for $t^\star \in \mathcal{T}$ sorted by prices $(P_t)_t$ do
    if the condition (13) is satisfied then
        $x_{t^\star} := 1$;
        Apply Lemma 4.3
return Optimal solution $(x_t)_t$
```

Time Complexity

In each loop, it needs to update the elements of A vector after ta and update the elements of B vector after tb, which takes O(T) on average, so the whole process takes O(T^2).

Improvement

Considering the need for ta in each update of the loop, the element after tb is subtracted by 1. Therefore, an array is maintained that allows the elements of an interval to be subtracted by a number, allowing the value of the subscript i to be interrogated. Using a line tree and using a difference array to update the values of the interval, assume that d[i] = a[i] - a[i-1] (d[0] = a[0]) , so subtracting a from the value in [l, r] requires only d[l] = d[l] - a, d[r + 1] = d[r + 1] - a, which requires O(1) operations. Applying the operation to the line tree, it takes O(log T) because each operation changes only one value at each level of the line tree. The query for the value of subscript i needs to consider the changes in the previous logi layer, and the query requires O(logT). In summary, the line-segment tree difference array can optimize the time complexity to O(TlogT).

1. The proof of correctness of segment tree

Theorem: A line tree of [1,n] can decompose any subinterval [L,R] of [1,n] into no more than $2\lfloor \log_2(n-1) \rfloor$ subintervals for n >= 3.

Using mathematical induction, prove the above theorem as follows. First, for n=3,4,5, it is not difficult to prove the theorem by exhaustive enumeration. Suppose that for n= 3,4,5,... ,k-1 the above equation holds, the following is to prove that for n=k (k>=6) holds.

The proof is divided into 4 cases.
Case 1: [L,R] contains the root node (L=1 and R=n), at this time, [L,R] is decomposed into one node and the theorem holds.

Case 2: [L,R] contains the left child node of the root node, at this time [L,R] must not contain the right child node of the root (because if it does, it can merge the left and right child nodes and replace it with the root node, this is case one). At this point, the number of elements of the tree with the right child node as the root is $\left\lfloor \frac{k}{2} \right\rfloor \geq 3$.
The subinterval into which [L,R] is divided consists of two parts.
1. the left child node of the root, the number of intervals is 1
2. the interval query in the tree rooted by the right child node of the root, and this can be used recursively using this theorem.
From the induction hypothesis, it follows that [L,R] is divided into a total of

$1+2\left\lfloor \log_2\left(\left\lfloor \frac{k}{2} \right\rfloor -1\right) \right\rfloor$ intervals.

Case 3: symmetric to case 2, unlike case 2, the number of elements of the tree rooted at the left child node of the root is $\left\lfloor \frac{k+1}{2} \right\rfloor \geq 3$

.

The tree [L,R] is divided into a total of $1+2\left\lfloor \log_2\left(\left\lfloor \frac{k+1}{2} \right\rfloor -1\right) \right\rfloor$ intervals. From the formula, we can see that the number of intervals in case two is less than or equal to the number of intervals in case three, so we only need to prove that the number of intervals in case three meets the condition.

$$
\begin{aligned}
&1+2\left\lfloor \log_2\left(\left\lfloor \frac{k+1}{2}\right\rfloor -1\right)\right\rfloor \\
&=1+2\left\lfloor \log_2\left(\left\lfloor \frac{k-1}{2}\right\rfloor\right)\right\rfloor \\
&\leq 1+2\left\lfloor \log_2\left(\frac{k-1}{2}\right)\right\rfloor \\
&=1+2\lfloor \log_2(k-1)-1\rfloor \\
&=2\lfloor \log_2(k-1)\rfloor -1<2\lfloor \log_2(k-1)\rfloor
\end{aligned}
$$

Thus, the theorems of case two and case three hold.

Case 4: [L,R] does not include the root node and the left and right children of the root node. Thus, the remaining layer $\lfloor \log_2(k-1)\rfloor$ has at most two nodes per layer (refer to the content in the rough proof). Thus [L,R] is decomposed into at most $2\lfloor \log_2(k-1)\rfloor$ intervals and the theorem holds. The above only proves that $2\lfloor \log_2(k-1)\rfloor$ is an upper bound, but, in fact, it is the minimum upper bound. When n=3,4, there are many groups of intervals that can be decomposed to reach the minimum upper bound. When n>4, the decomposition of the interval [L,R] can reach the minimum upper bound $2\lfloor \log_2(k-1)\rfloor$ when and only when n=2^t (t>=3),L=2,R=2^t - 1.

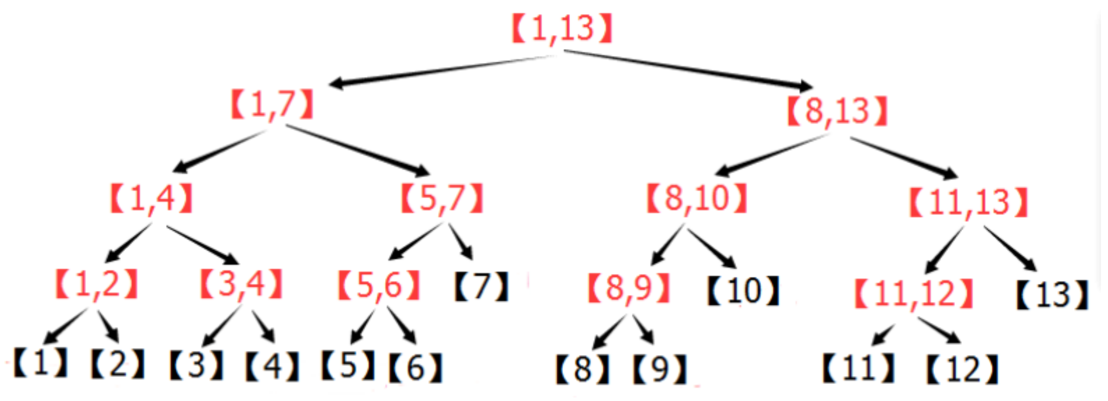
Example

A line tree is a decomposition of each interval [L,R] into [L,M] and [M+1,R] (where M=(L+R)/2 where the division is integer division, i.e., rounding down the result) until L==R.

The tree starts with the interval [1,n] , and is decomposed step by step by recursion, assuming that the height of the root is 1, and the maximum height of the tree is $\lfloor \log_2(n-1)\rfloor + 2$ (n>1).

The decomposition of the tree is unique for each n, so the tree structure is the same for the same n. This is also the basis for the implementation of a persistent tree.

The following figure shows the decomposition of the interval [1,13].



Division of Labor

韩子睿: program code
周德程: experimental report
孙宇祥: slides and presentation