Optimization Methods

1. Introduction.
2. Greedy algorithms for combinatorial optimization.
3. LS and neighborhood structures for combinatorial optimization.
4. Variable neighborhood search, neighborhood descent, SA, TS, EC.
5. Branch and bound algorithms, and **subset selection algorithms**.
6. Linear programming problem formulations and applications.
7. Linear programming algorithms.
8. Integer linear programming algorithms.
9. Unconstrained nonlinear optimization and gradient descent.
10. Newton's methods and Levenberg-Marquardt modification.
11. Quasi-Newton methods and conjugate direction methods.
12. Nonlinear optimization with equality constraints.
13. Nonlinear optimization with inequality constraints.
14. Problem formulation and concepts in multi-objective optimization.
15. Search for single final solution in multi-objective optimization.
16: Search for multiple solutions in multi-objective optimization.

Subset Selection Problems

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, ..., s_n\}$)

Objective: Maximize $f(S)$ or Minimize $f(S)$ under $S \subseteq S_C$

Constraint: $g(S) = G$ or $g(S) \leq G$

Output: Selected item set: S

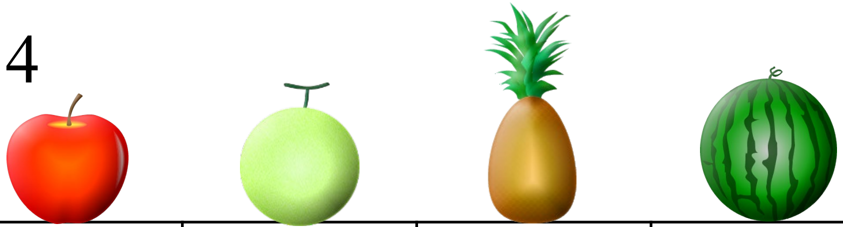
[Example: Knapsack Problem]

Maximize $4x_1 + 5x_2 + 12x_3 + 14x_4$

Subject to $2x_1 + 3x_2 + 5x_3 + 6x_4 \leq 9$

$x_i = 0$ or 1, $i = 1, 2, 3, 4$

W = 9 kg



Item (i)	1	2	3	4
value (v_i)	4 $	5 $	12 $	14 $
weight (w_i)	2 kg	3 kg	5 kg	6 kg

Simplest Version of Subset Selection Problems

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, ..., s_n\}$)

Objective: Maximize $f(S)$ or Minimize $f(S)$ under $S \subseteq S_C$

Constraint: $|S| = k$ **(Selection of *k* items)**

Output: Selected item set: S

[Simple Example]

Maximize $4x_1 + 5x_2 + 12x_3 + 14x_4$

Subject to $x_1 + x_2 + x_3 + x_4 = 2$ (i.e., $k = 2$)

$x_i = 0$ or 1, $i = 1, 2, 3, 4$

Item (i)	1	2	3	4
value (v_i)	4 $	5 $	12 $	14 $
weight (w_i)	1 kg	1 kg	1 kg	1 kg

W **= 2 kg**

Optimal solution: $S = \{s_3, s_4\}$, $x_1 = x_1 = 0$, $x_3 = x_4 = 1$.

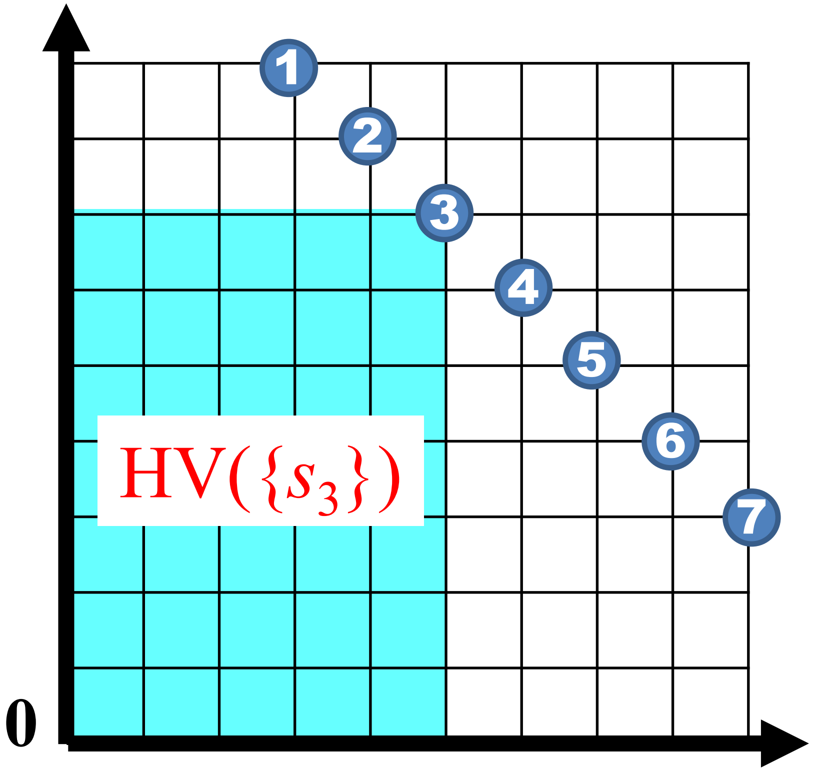
An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, ..., s_n\}$)

Objective: Maximize $HV(S) = Volume\left(\bigcup_{s_i \in S} \{\boldsymbol{x} \mid 0 \le \boldsymbol{x} \le s_i\} \right)$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S



s_i : Point in a space

Search Space Size: ${}_nC_k$

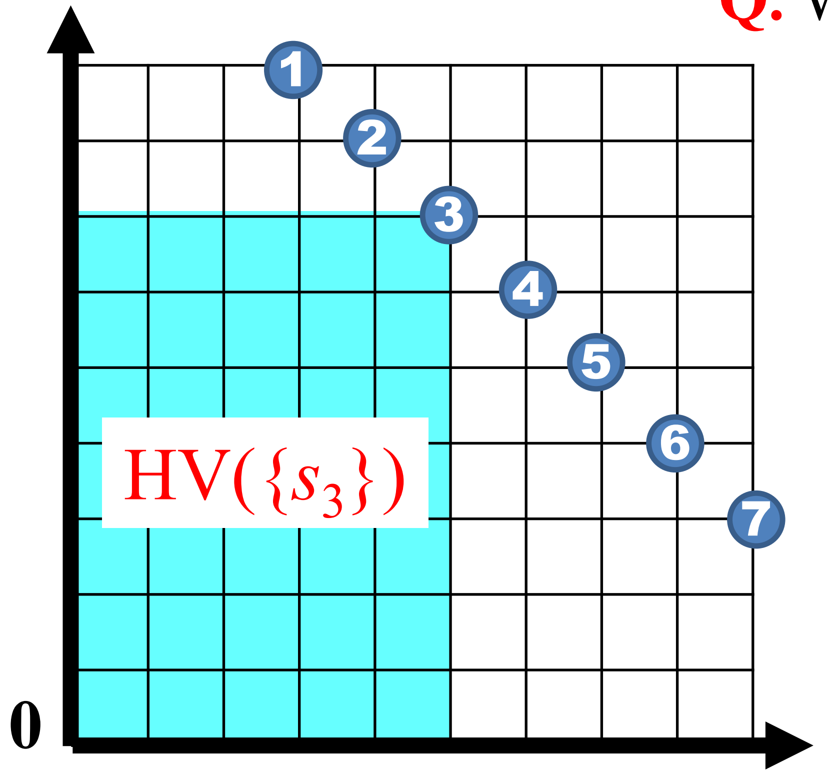
An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, \ldots, s_n\}$)

Objective: Maximize $HV(S) = Volume\left(\bigcup_{s_i \in S} \{\boldsymbol{x} \mid 0 \le \boldsymbol{x} \le s_i\} \right)$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S



Q. What is the optimal solution for $k = 1$?

Your answer: $\{s_?\}$

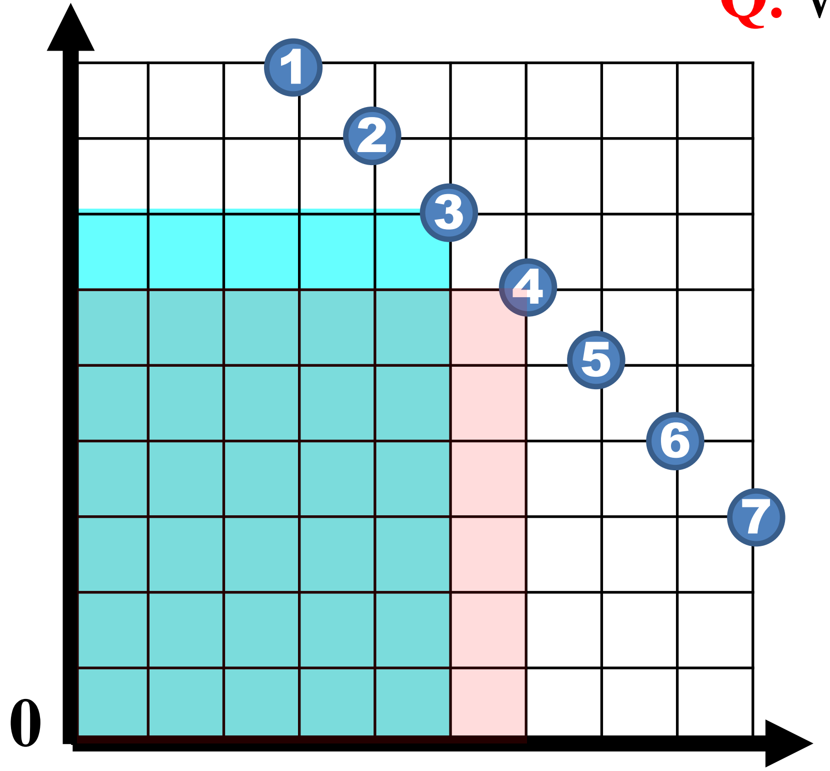
An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, ..., s_n\}$)

Objective: Maximize $HV(S) = Volume\left(\bigcup_{s_i \in S} \{\boldsymbol{x} \mid 0 \le \boldsymbol{x} \le s_i\}\right)$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S



Q. What is the optimal solution for $k = 1$?

Your answer: $\{s_4\}$

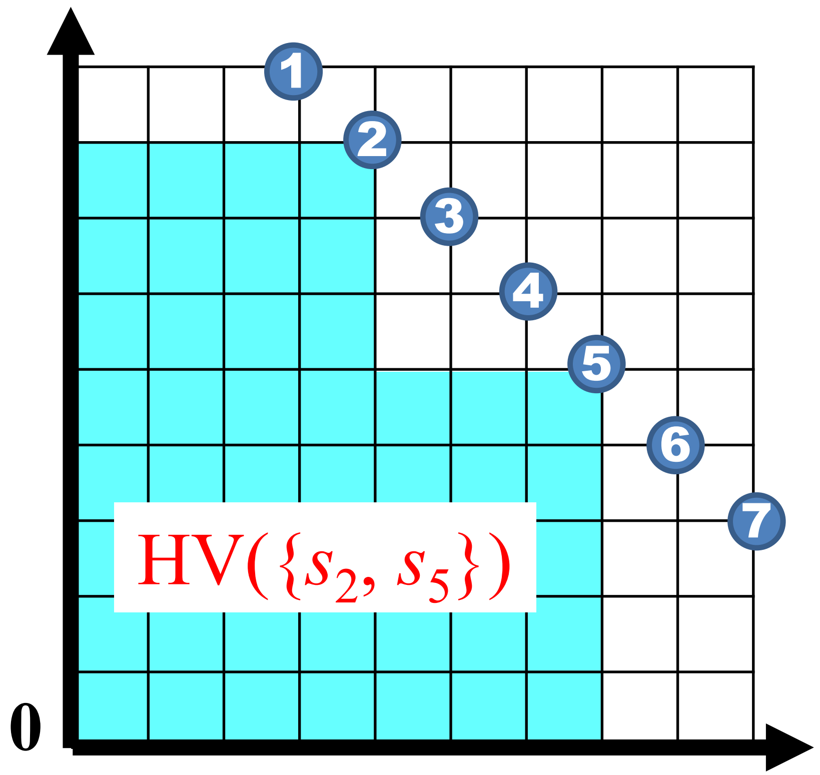
An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, ..., s_n\}$)

Objective: Maximize $HV(S) = Volume\left(\bigcup_{s_i \in S} \{\boldsymbol{x} \mid 0 \le \boldsymbol{x} \le s_i\} \right)$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S



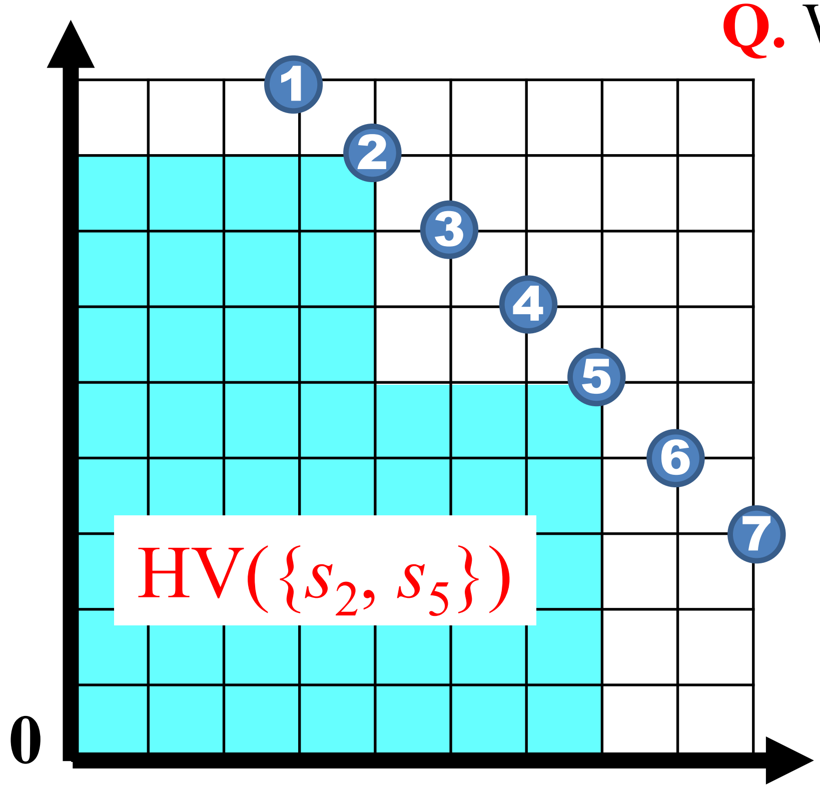
An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, \ldots, s_n\}$)

Objective: Maximize $HV(S) = Volume\left(\bigcup_{s_i \in S} \{\boldsymbol{x} \mid 0 \le \boldsymbol{x} \le s_i\}\right)$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S



Q. What is the optimal solution for $k = 2$?

Your answer: $\{s_?, s_?\}$

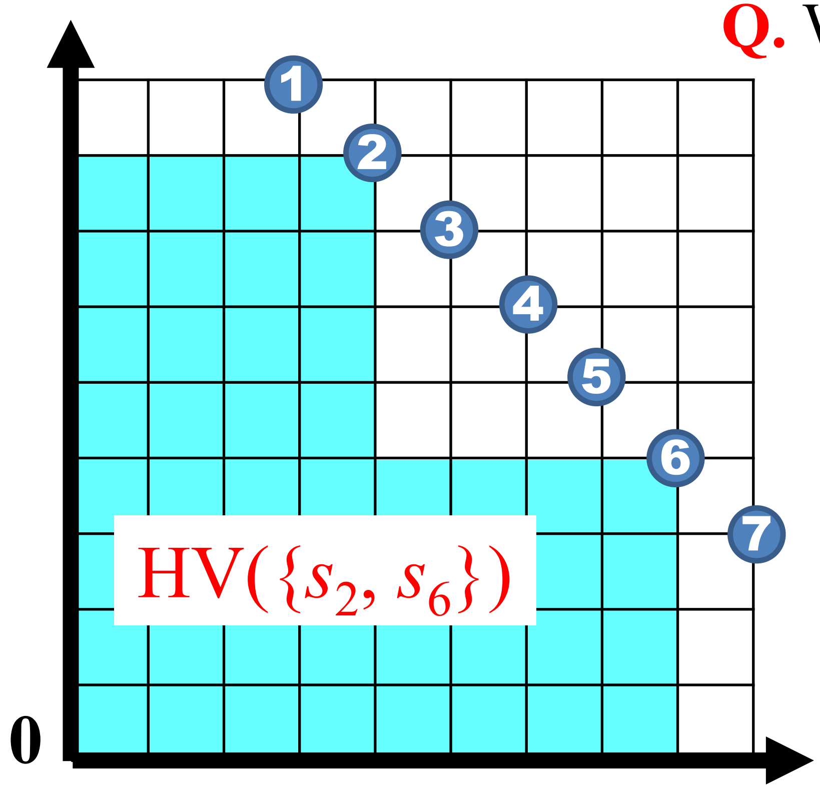
An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, \ldots, s_n\}$)

Objective: Maximize $HV(S) = Volume\left(\bigcup_{s_i \in S} \{\boldsymbol{x} \mid 0 \leq \boldsymbol{x} \leq s_i\}\right)$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S



Q. What is the optimal solution for $k = 2$?

Your answer: $\{s_2, s_6\}$

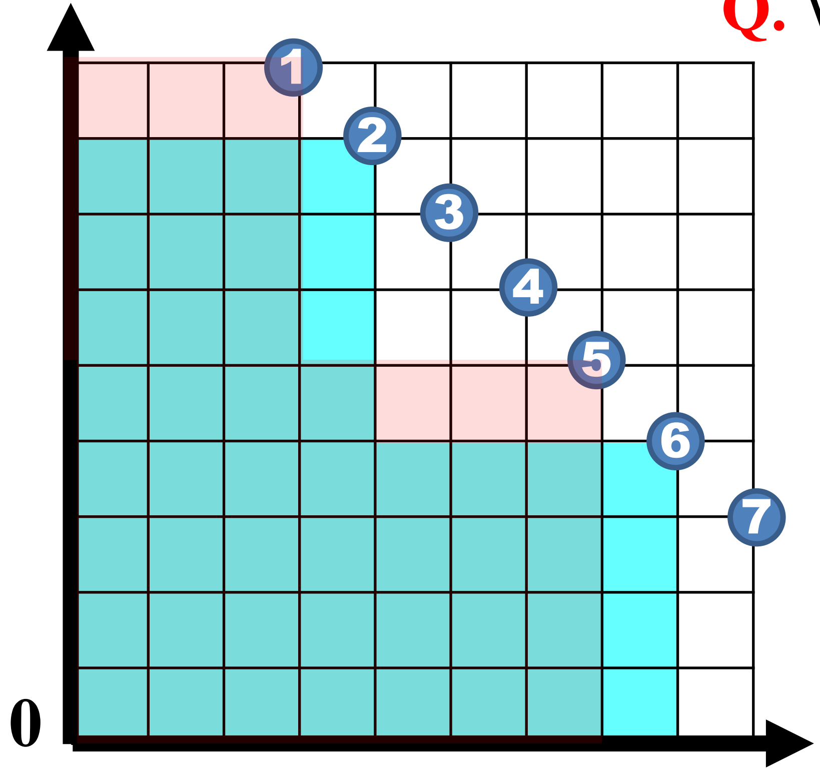
An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, ..., s_n\}$)

Objective: Maximize $HV(S) = Volume\left(\bigcup_{s_i \in S} \{\boldsymbol{x} \mid 0 \le \boldsymbol{x} \le s_i\} \right)$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S



Q. What is the optimal solution for $k = 2$?

Your answer: $\{s_2, s_6\}$

$$HV(\{s_1, s_5\}) < HV(\{s_2, s_6\})$$

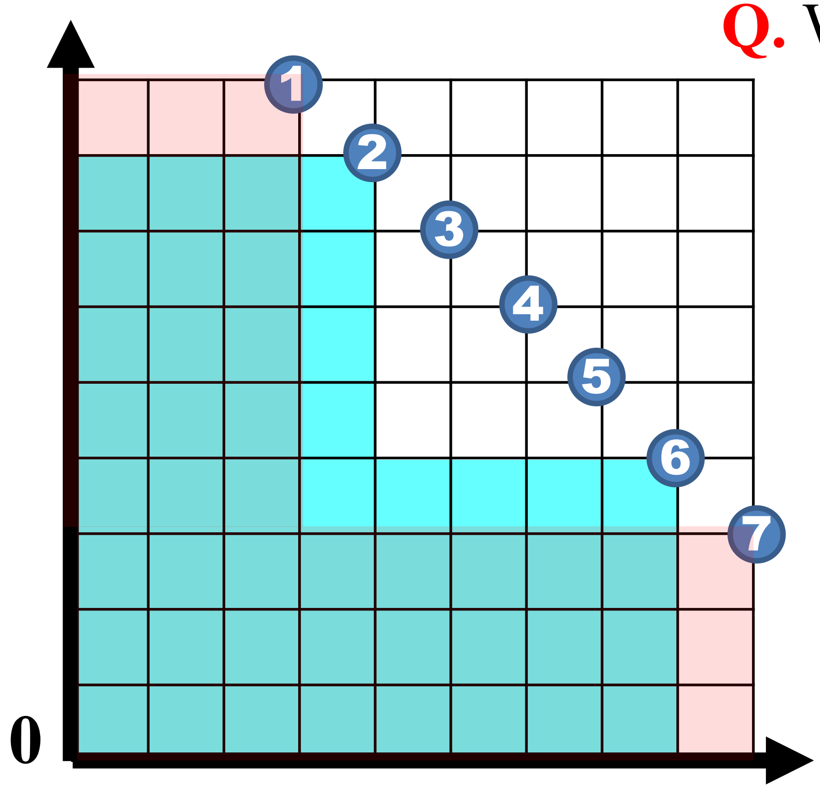
An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, ..., s_n\}$)

Objective: Maximize $HV(S) = Volume\left(\bigcup_{s_i \in S} \{\boldsymbol{x} \mid 0 \leq \boldsymbol{x} \leq s_i\} \right)$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S



Q. What is the optimal solution for $k = 2$?

Your answer: $\{s_2, s_6\}$

$$HV(\{s_1, s_7\}) < HV(\{s_2, s_6\})$$

An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, ..., s_n\}$)

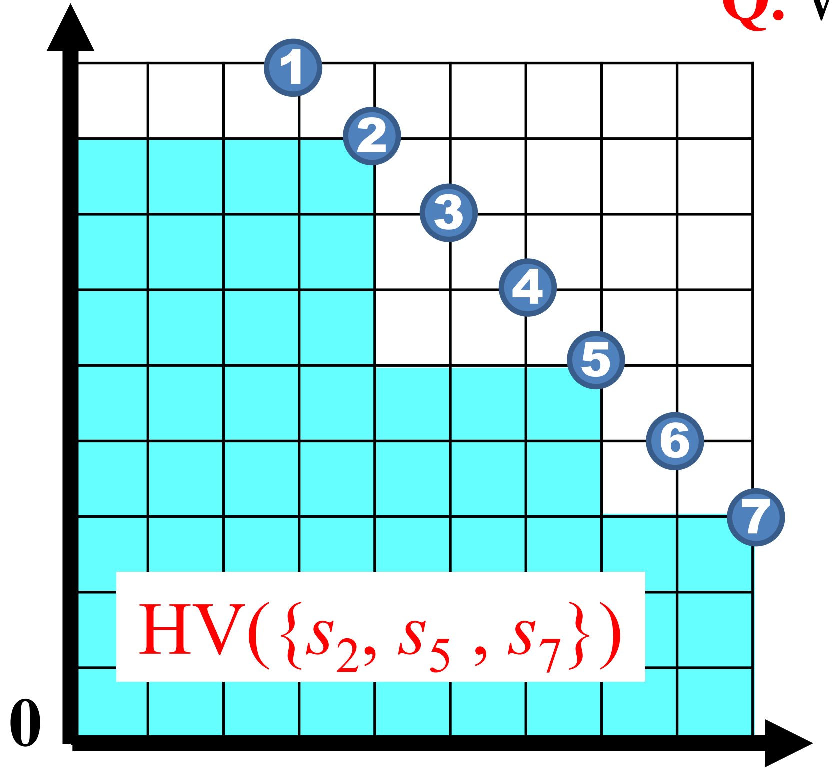
Objective: Maximize $HV(S) = Volume\left(\bigcup_{s_i \in S} \{\boldsymbol{x} \mid 0 \leq \boldsymbol{x} \leq s_i\}\right)$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S

Q. What is the optimal solution for $k = 3$?

Your answer: $\{s_?, s_?, s_?\}$



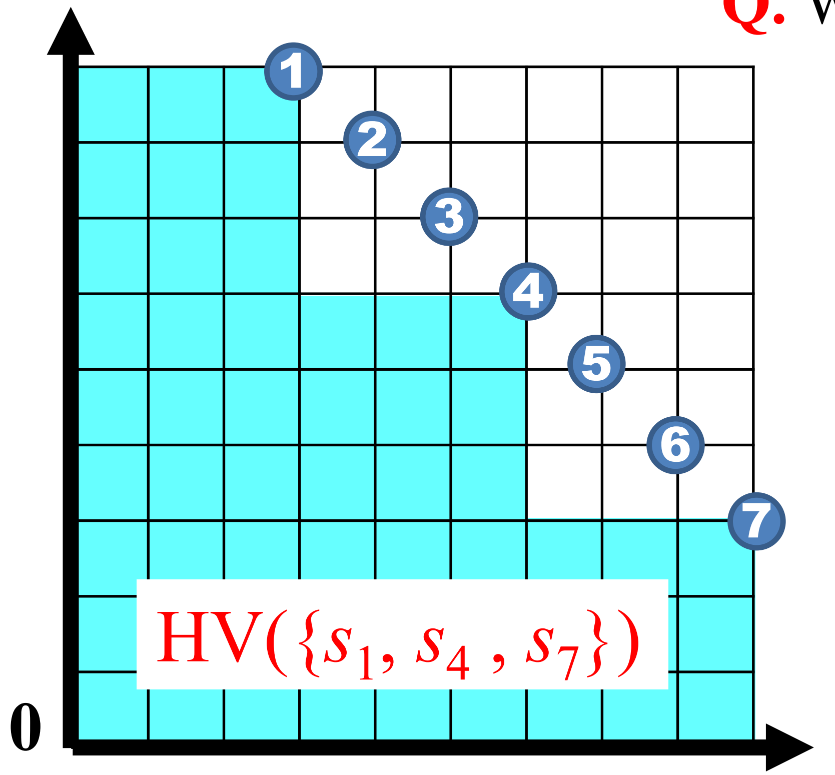
An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, \ldots, s_n\}$)

Objective: Maximize $HV(S) = Volume\left(\bigcup_{s_i \in S} \{\boldsymbol{x} \mid 0 \le \boldsymbol{x} \le s_i\} \right)$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S



Q. What is the optimal solution for $k = 3$?

Your answer: $\{s_1, s_4, s_7\}$

An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, ..., s_n\}$)

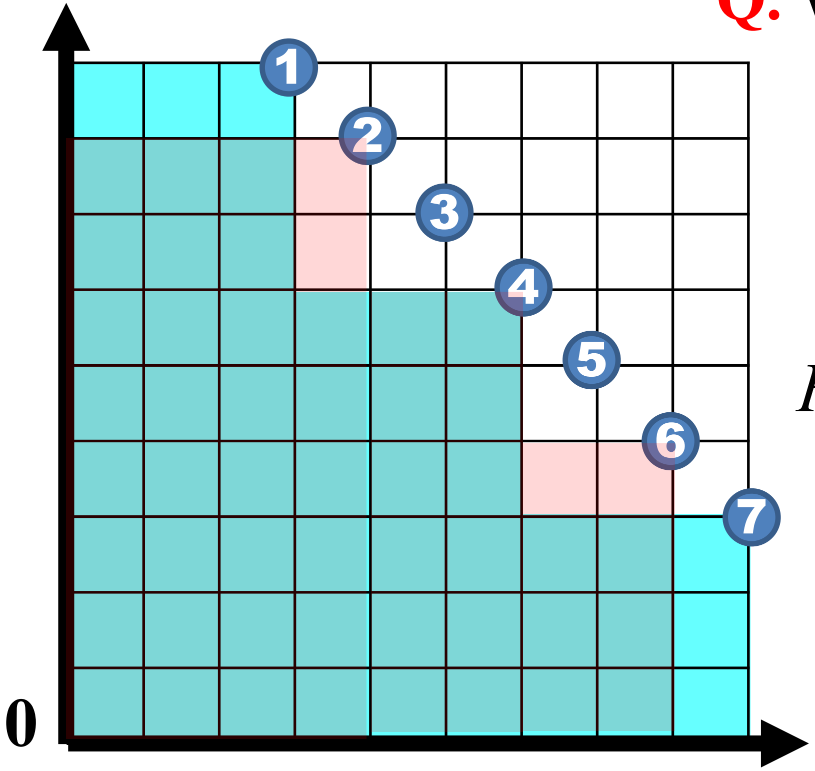
Objective: Maximize $HV(S) = Volume\left(\bigcup_{s_i \in S} \{\boldsymbol{x} \mid 0 \le \boldsymbol{x} \le s_i\}\right)$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S

Q. What is the optimal solution for $k = 3$?

Your answer: $\{s_1, s_4, s_7\}$



$HV(\{s_2, s_4, s_6\}) < HV(\{s_1, s_4, s_7\})$

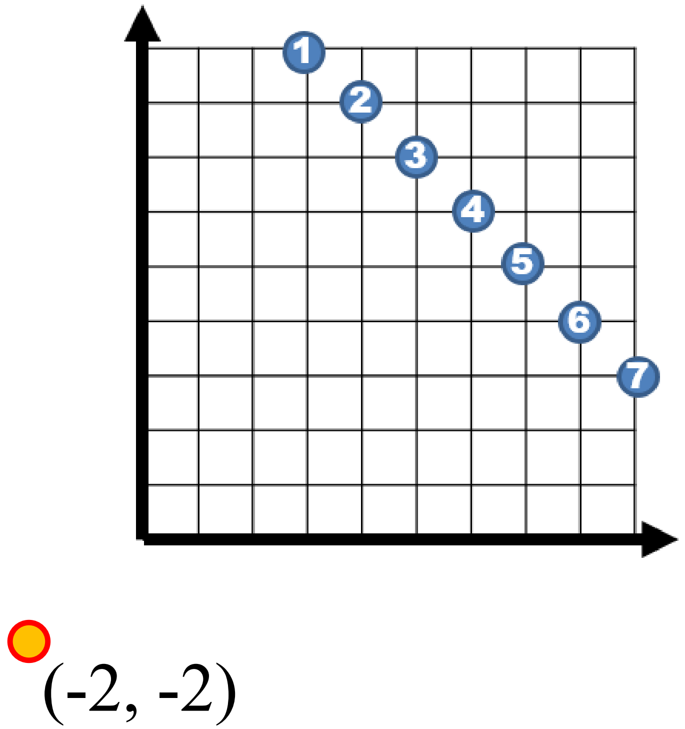
An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, ..., s_n\}$)

Maximize $HV(S) = Volume\left(\bigcup_{s_i \in S} \{\boldsymbol{x} \mid (-2, -2) \leq \boldsymbol{x} \leq s_i\} \right)$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S



Q. What is the optimal solution for $\boldsymbol{k = 2}$?

Your answer: $\{s_?, s_?\}$

The reference point: The lower bound for HV calculation.

The optimal subset depends on the reference point specification.

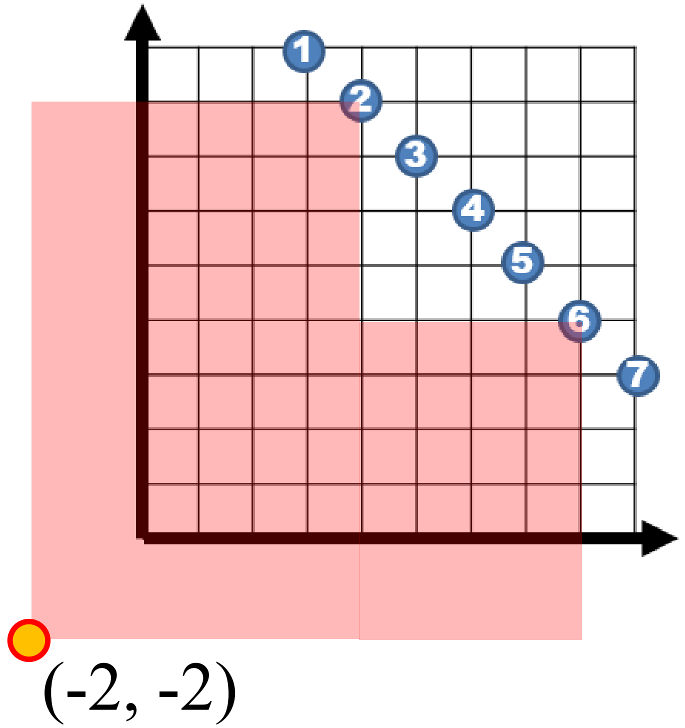
An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, ..., s_n\}$)

Maximize $HV(S) = Volume\left(\bigcup_{s_i \in S} \{ \boldsymbol{x} \mid (-2, -2) \le \boldsymbol{x} \le s_i \} \right)$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S



Q. What is the optimal solution for $\boldsymbol{k = 2}$?

Your answer: $\{s_?, s_?\}$

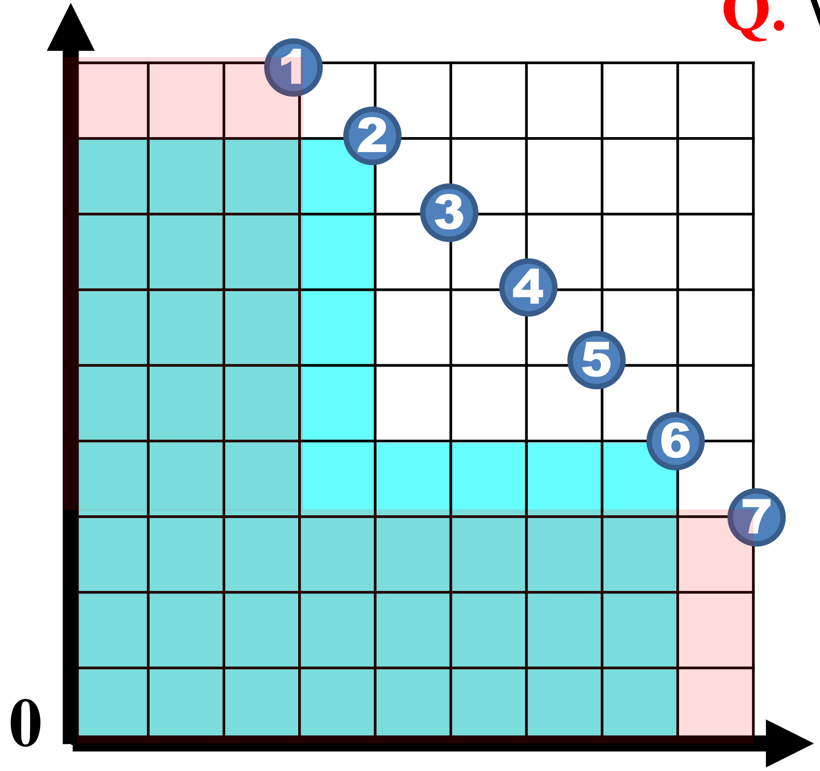
An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, \ldots, s_n\}$)

Objective: Maximize $HV(S) = Volume\left(\bigcup_{s_i \in S} \{\boldsymbol{x} \mid 0 \le \boldsymbol{x} \le s_i\}\right)$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S



Q. What is the optimal solution for $k = 2$?

Your answer: $\{s_2, s_6\}$

$HV(\{s_1, s_7\}) < HV(\{s_2, s_6\})$

The reference point: The lower bound for HV calculation.

The optimal subset depends on the reference point specification.

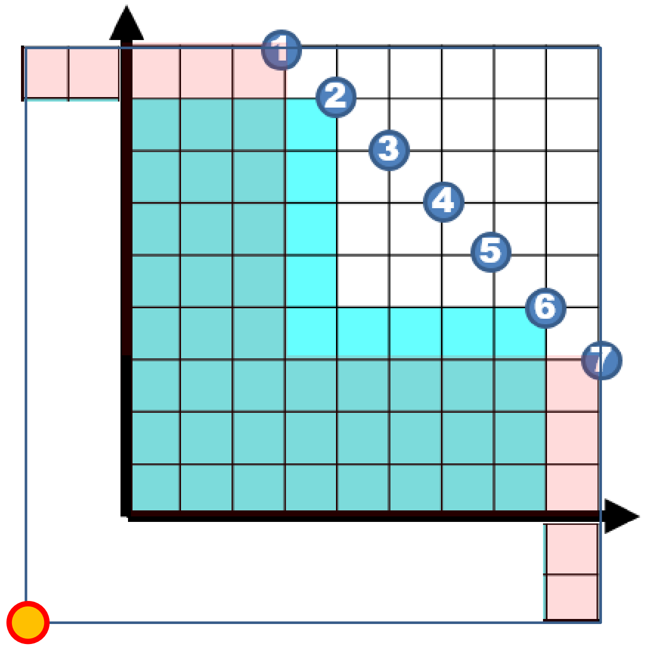
An Example: Hypervolume Subset Selection

Input: Candidate item set S_C: n items ($S_C = \{s_1, s_2, ..., s_n\}$)

Maximize $$HV(S) = Volume\left(\bigcup_{s_i \in S} \{\boldsymbol{x} \mid (-2, -2) \le \boldsymbol{x} \le s_i\} \right)$$

Constraint: $|S| = k$ (Selection of k items) and $S \subseteq S_C$

Output: Selected item set: S



(-2, -2)

Q. What is the optimal solution for $k = 3$?

Your answer: $\{s_1, s_7\}$

$HV(\{s_1, s_7\}) > HV(\{s_2, s_6\})$

The reference point: The lower bound for HV calculation.

The optimal subset depends on the reference point specification.

An Example: Hypervolume Subset Selection

Search Space Size: $_nC_k$

How to solve this problem:

Small size problem (e.g., n = 20, k = 2)

We can find the optimal subset by examining all combinations.

Research topic: Design an efficient optimization algorithm.

(to decrease the computation time.)

An Example: Hypervolume Subset Selection

Search Space Size: ${}_{n}C_{k}$

How to solve this problem:

Small size problem (e.g., $n = 20$, $k = 2$)
We can find the optimal subset by examining all combinations.
Research topic: Design an efficient optimization algorithm.
(to decrease the computation time.)

Medium size problem (e.g., $n = 200$, $k = 20$)
Various approximation algorithms can be used. LS, SA, EA, ...
Research topic: Search for near-optimal subsets efficiently.
(i.e., to decrease the computation time, and to increase the subset quality).

An Example: Hypervolume Subset Selection

Search Space Size: ${}_nC_k$

How to solve this problem:

Small size problem (e.g., $n = 20$, $k = 2$)

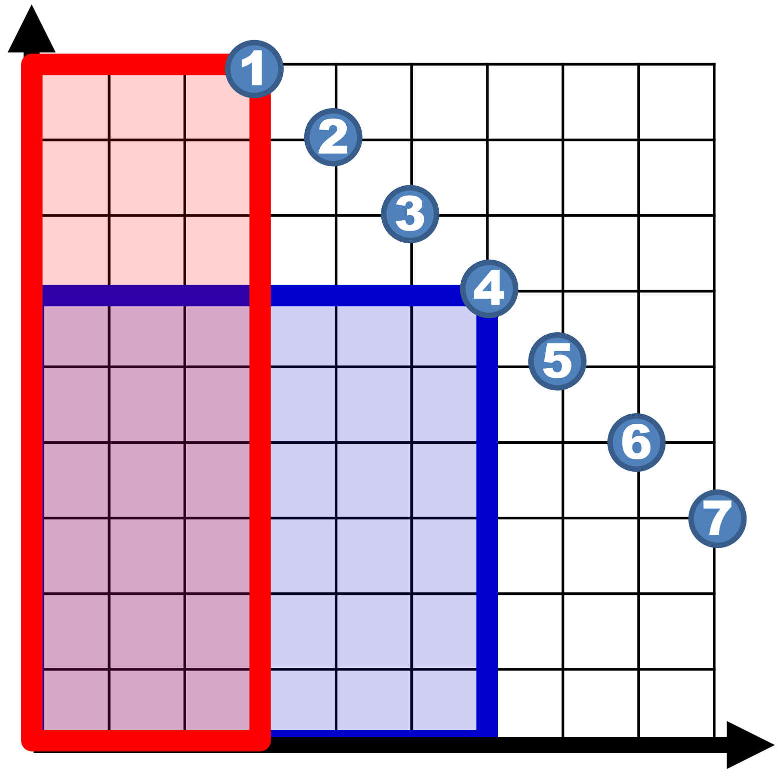
We can find the optimal subset by examining all combinations.
Research topic: Design an efficient optimization algorithm.
(to decrease the computation time.)

Medium size problem (e.g., $n = 200$, $k = 20$)

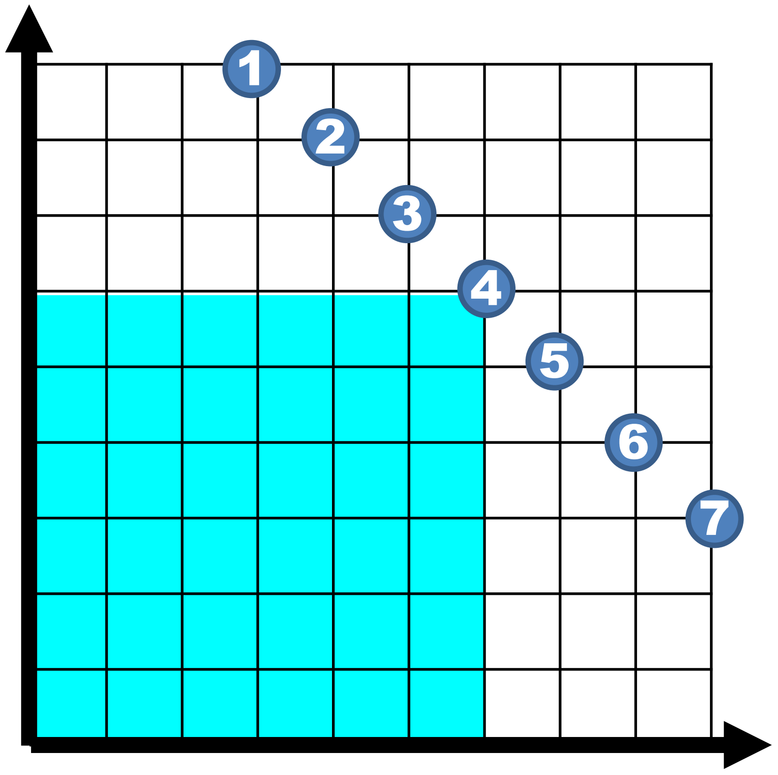
Various approximation algorithms can be used. LS, SA, EA, ...
Research topic: Search for near-optimal subsets efficiently.
(i.e., to decrease the computation time, and to increase the subset quality).

Greedy algorithm

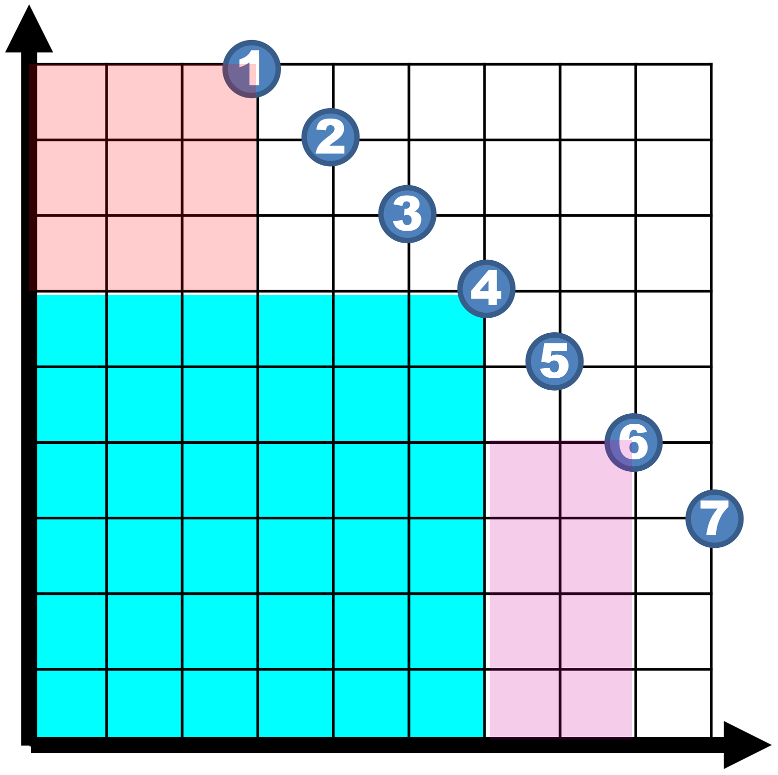
The first item is the single best item. After that, the best item with the largest contribution is selected by examining all the remaining items one by one.



Step 1: Each item is separately evaluated. Item 4 is the best item, which is selected.

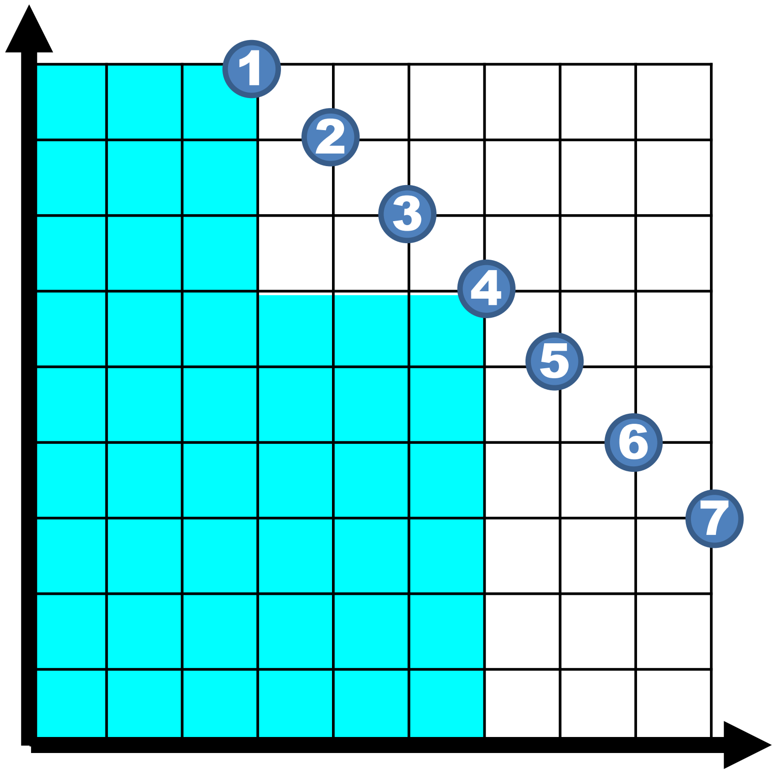


Step 1: Each item is separately evaluated. Item 4 is the best item, which is selected.



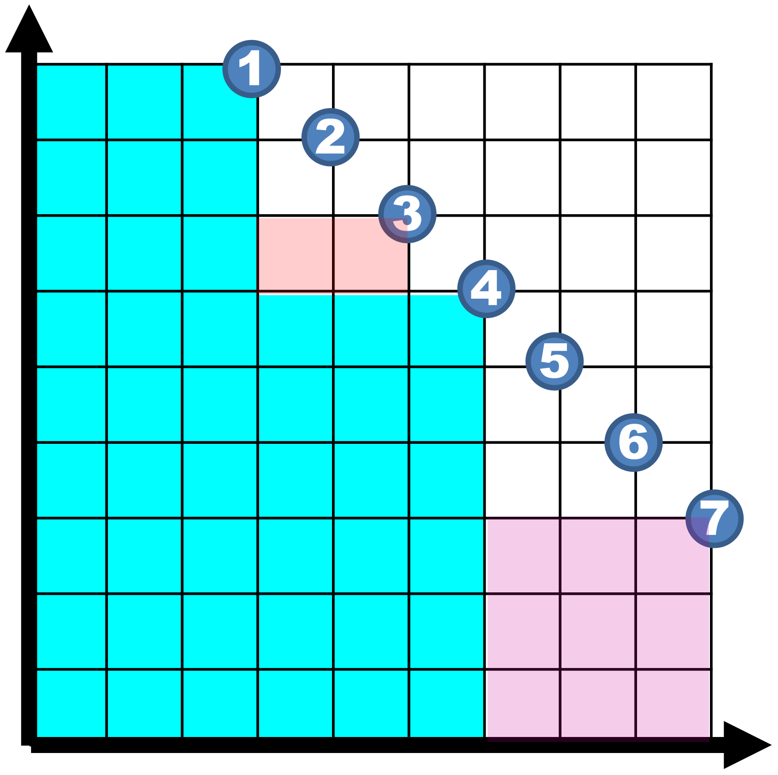
Step 1: Each item is separately evaluated. Item 4 is the best item, which is selected.

Step 2: The contribution of each of the remaining items is examined (random tie-break). Item 1 and Item 7 have the same largest contribution. Item 1 is selected.



Step 1: Each item is separately evaluated. Item 4 is the best item, which is selected.

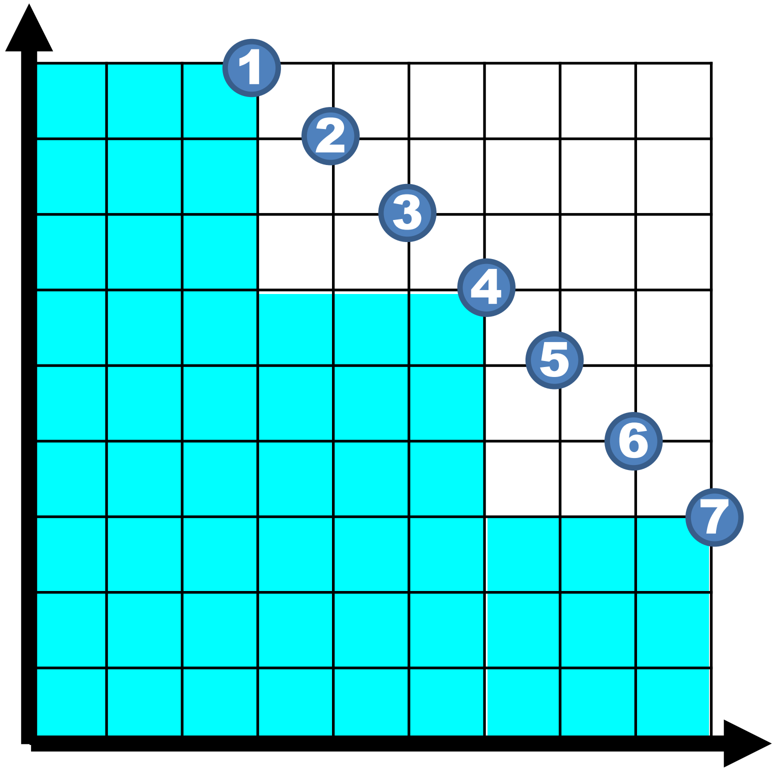
Step 2: The contribution of each of the remaining items is examined (random tie-break). Item 1 and Item 7 have the same largest contribution. Item 1 is selected.



Step 1: Each item is separately evaluated. Item 4 is the best item, which is selected.

Step 2: The contribution of each of the remaining items is examined (random tie-break). Item 1 and Item 7 have the same largest contribution. Item 1 is selected.

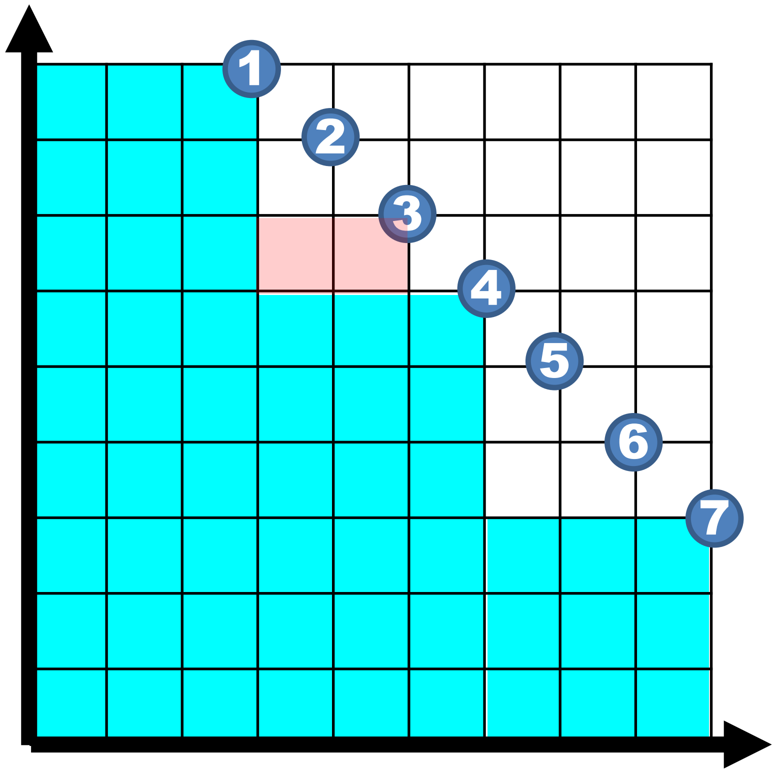
Step 3: The contribution of each of the remaining items is examined (random tie-break). Item 7 have the same largest contribution. Item 7 is selected.



Step 1: Each item is separately evaluated. Item 4 is the best item, which is selected.

Step 2: The contribution of each of the remaining items is examined (random tie-break). Item 1 and Item 7 have the same largest contribution. Item 1 is selected.

Step 3: The contribution of each of the remaining items is examined (random tie-break). Item 7 have the same largest contribution. Item 7 is selected.



Step 1: Each item is separately evaluated. Item 4 is the best item, which is selected.

Step 2: The contribution of each of the remaining items is examined (random tie-break). Item 1 and Item 7 have the same largest contribution. Item 1 is selected.

Step 3: The contribution of each of the remaining items is examined (random tie-break). Item 7 have the same largest contribution. Item 7 is selected.

Step 4: The contribution of each of the remaining items is examined (random tie-break). ...

An Example: Hypervolume Subset Selection

How to solve this problem:

Small size problem (e.g., $n = 20$, $k = 2$)

We can find the optimal subset by examining all combinations.
Research topic: Design an efficient optimization algorithm.

Medium size problem (e.g., $n = 200$, $k = 20$)

Various approximation algorithm can be used. LS, SA, EA, ...
Research topic: Search for near-optimal subsets efficiently.

Greedy algorithm

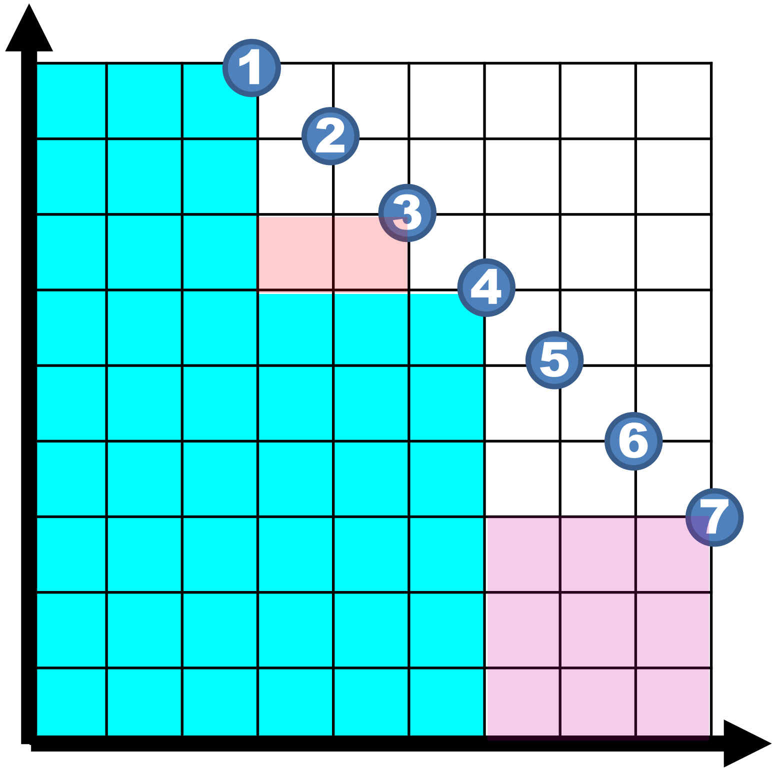
The first item is the single best item. After that, the best item with the largest contribution is selected by examining all the remaining items one by one.

Large size problem (e.g., $n = 2000$, $k = 200$)

Q1. Is the greedy algorithm practically useful ? Yes or No ?

Extremely large size problem (e.g., $n = 500{,}000{,}000$, $k = 500$)**:**

Q2. Is the greedy algorithm practically useful ? Yes or No ?

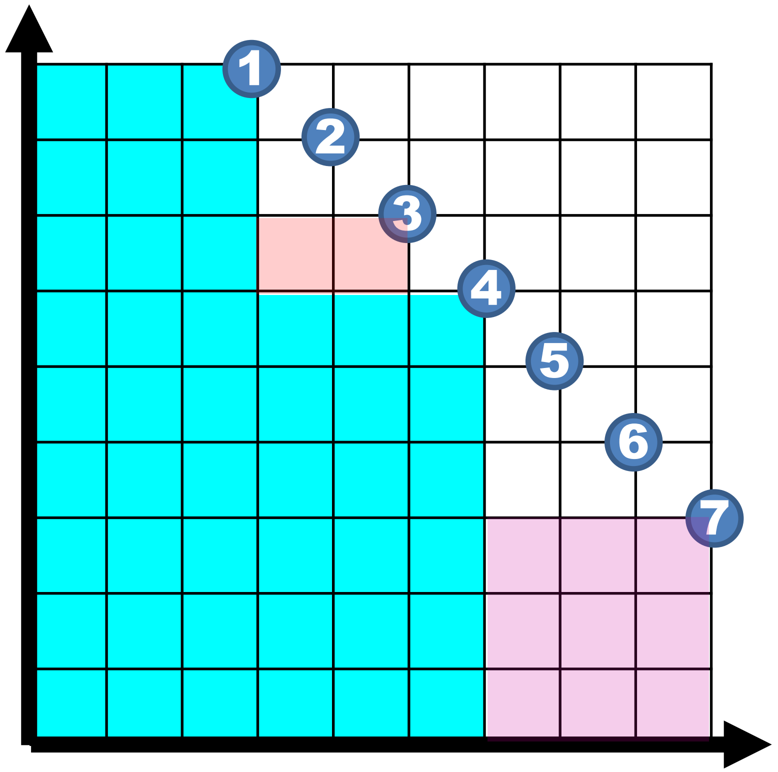


Step 1: Each item is separately evaluated. Item 4 is the best item, which is selected.

Step 2: The contribution of each of the remaining items is examined (random tie-break). Item 1 and Item 7 have the same largest contribution. Item 1 is selected.

Step 3: The contribution of each of the remaining items is examined (random tie-break). Item 7 have the same largest contribution. Item 7 is selected.

Q3. How to decrease the computation time ? **Your idea**

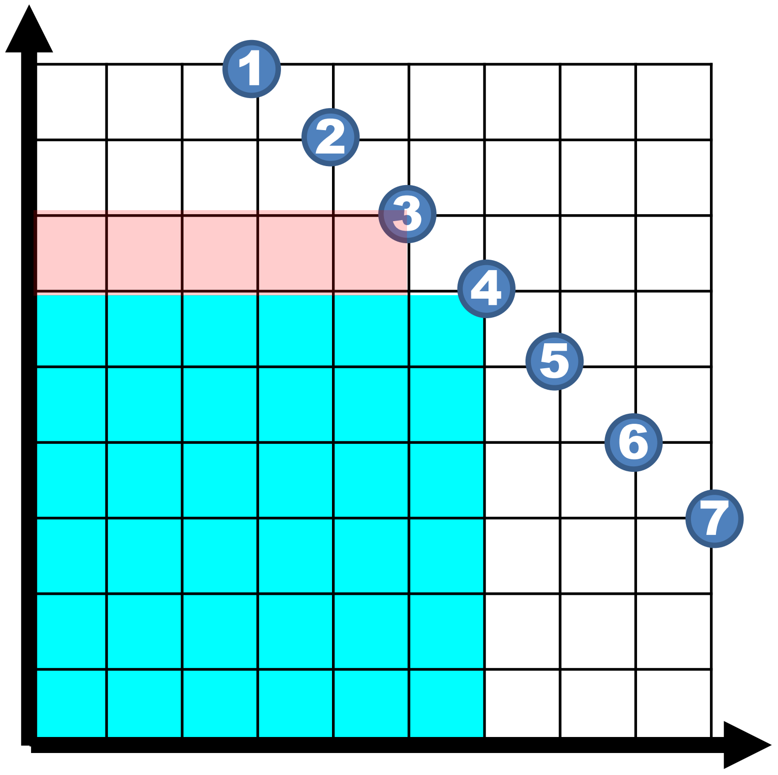


Step 1: Each item is separately evaluated. Item 4 is the best item, which is selected.

Step 2: The contribution of each of the remaining items is examined (random tie-break). Item 1 and Item 7 have the same largest contribution. Item 1 is selected.

Step 3: The contribution of each of the remaining items is examined (random tie-break). Item 7 have the same largest contribution. Item 7 is selected.

Q3. How to decrease the computation time ? **Your idea**



Step 1: Each item is separately evaluated. Item 4 is the best item, which is selected.

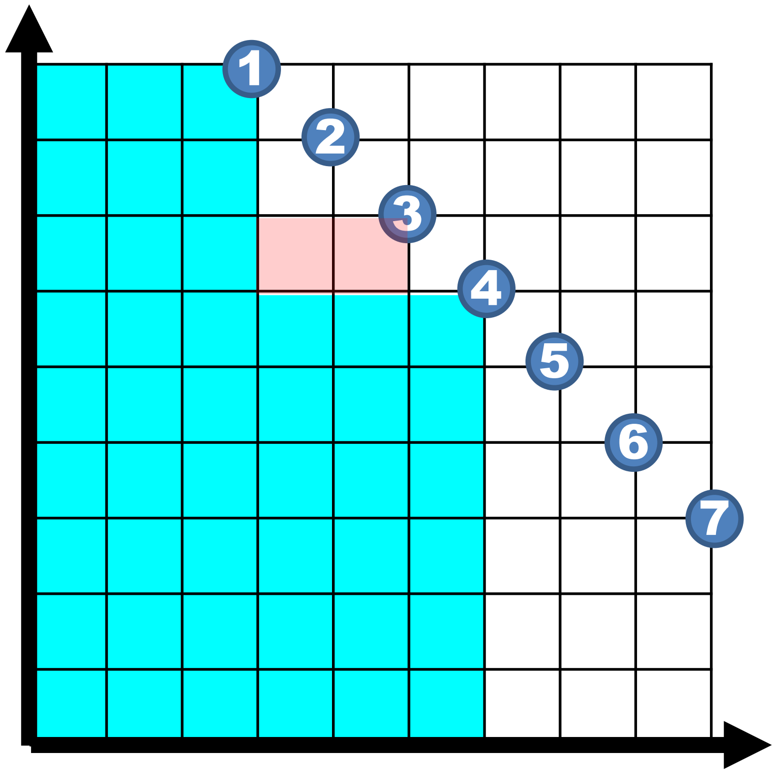
Step 2: The contribution of each of the remaining items is examined (random tie-break). Item 1 and Item 7 have the same largest contribution. Item 1 is selected.

How to decrease the computation time ?

Basic Idea: Use of the following mathematical feature:

Contribution of each item never increases.

Contribution of Item 3: 5



Step 1: Each item is separately evaluated. Item 4 is the best item, which is selected.

Step 2: The contribution of each of the remaining items is examined (random tie-break). Item 1 and Item 7 have the same largest contribution. Item 1 is selected.

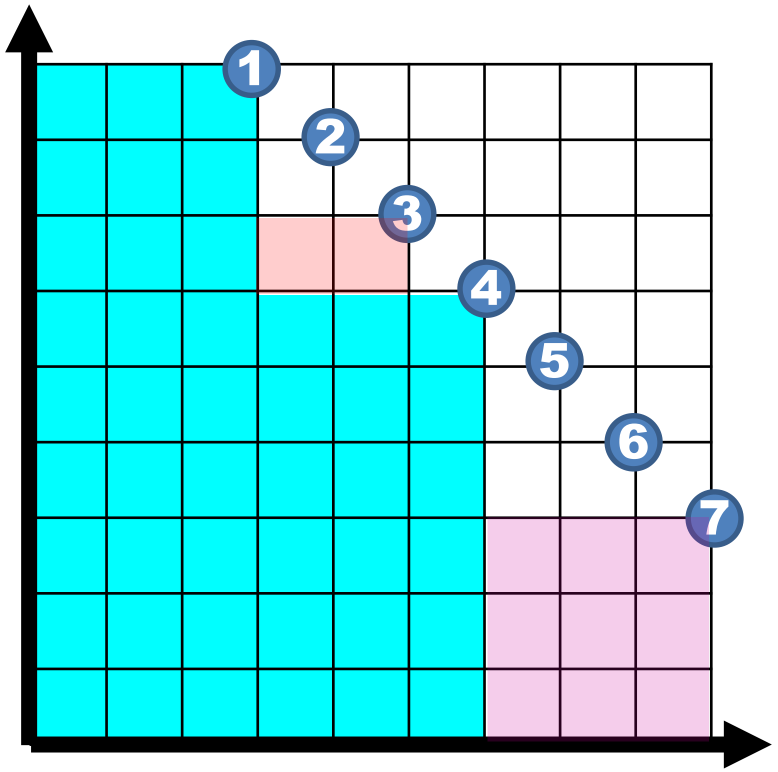
Step 3: The contribution of each of the remaining items is examined (random tie-break). Item 7 have the same largest contribution. Item 7 is selected.

How to decrease the computation time ?

Basic Idea: Use of the following mathematical feature:

Contribution of each item never increases.

Contribution of Item 3: 5 ==> 2



Step 1: Each item is separately evaluated. Item 4 is the best item, which is selected.

Step 2: The contribution of each of the remaining items is examined (random tie-break). Item 1 and Item 7 have the same largest contribution. Item 1 is selected.

Step 3: The contribution of each of the remaining items is examined (random tie-break). Item 7 have the same largest contribution. Item 7 is selected.

How to decrease the computation time ?

Basic Idea: Use of the following mathematical feature:

Contribution of each item never increases.

Contribution of Item 3: 5 ==> 2 (In Step 2, this recalculation is not needed since 5 is smaller than 9 (Item 7's contribution).

An Example: Hypervolume Subset Selection

Search Space Size: $_nC_k$

How to solve this problem:

Small size problem (e.g., $n = 20$, $k = 2$)

We can find the optimal subset by examining all combination.
Research topic: Design an efficient optimization algorithm.

Medium size problem (e.g., $n = 200$, $k = 20$)

Various approximation algorithm can be used. LS, SA, EA, ...
Research topic: Search for near-optimal subsets efficiently.

Greedy algorithm

The first item is the single best item. After that, the best item with the largest contribution is selected by examining all the remaining items one by one.

Local improvement (from an initial subset with k items)

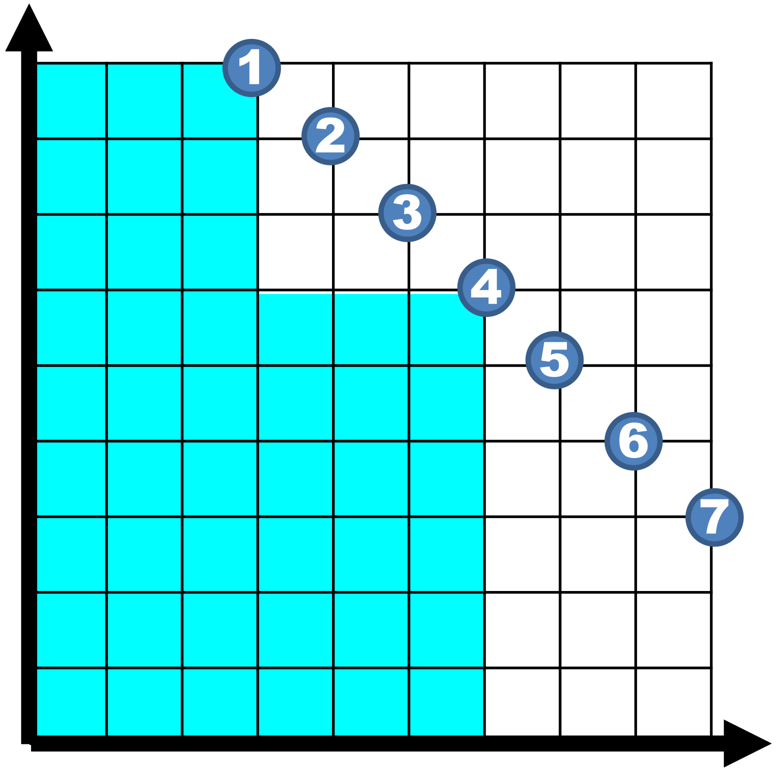
Iterate the following two steps :

1. Randomly add one of the remaining items to the selected item set.
2. Remove the worst item with the smallest contribution.

Explanation for *k* = 2

Greedy subset: {Item 1, Item 4}

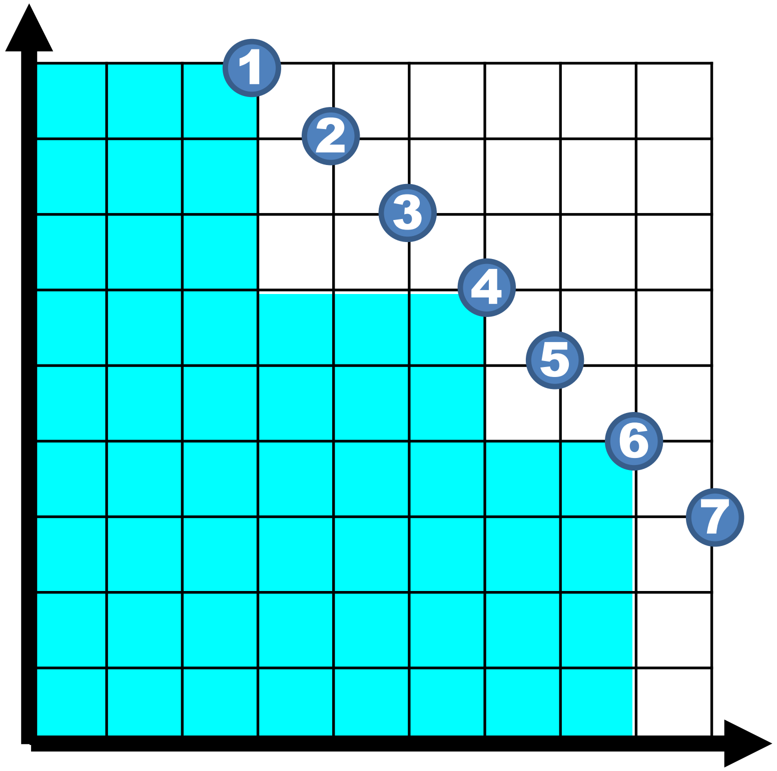
1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).

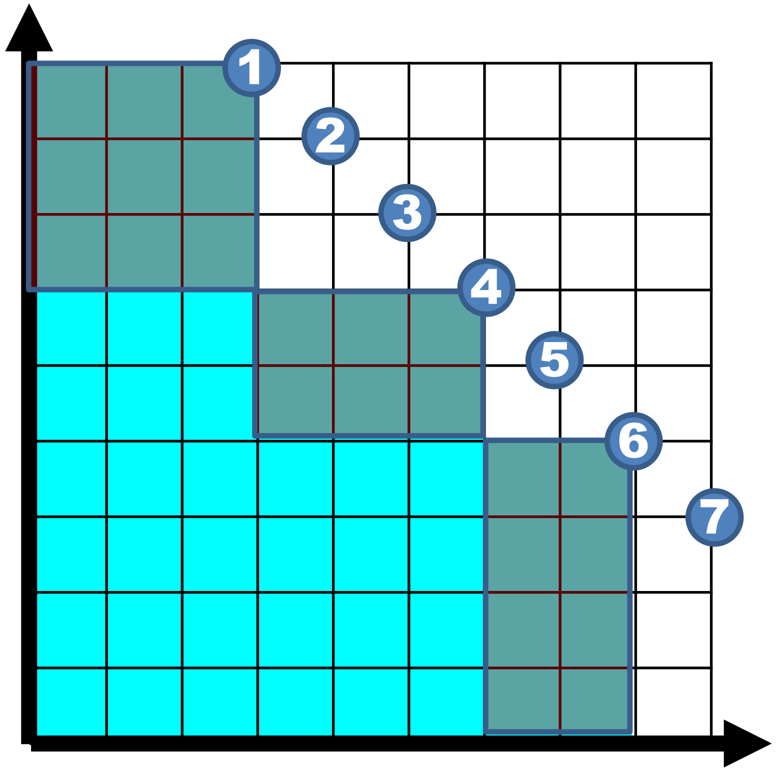


Explanation for $k = 2$

Greedy subset: {Item 1, Item 4}

1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).



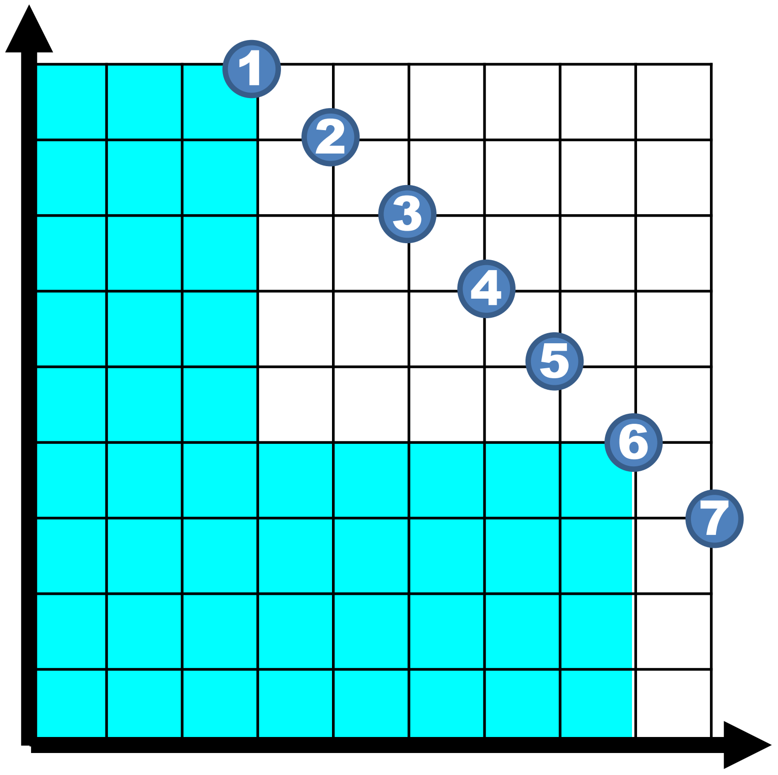


Explanation for *k* = 2

Greedy subset: {Item 1, Item 4}

1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).

2. Contribution of each item is evaluated. The worst item (Item 4) is removed.

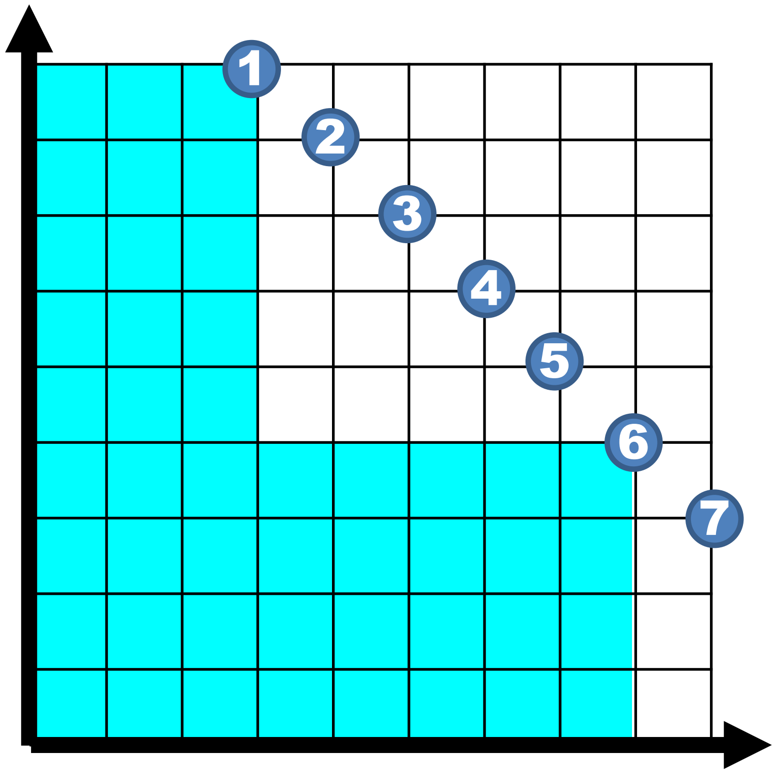


Explanation for $k = 2$

Greedy subset: {Item 1, Item 4}

1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).

2. Contribution of each item is evaluated. The worst item (Item 4) is removed.



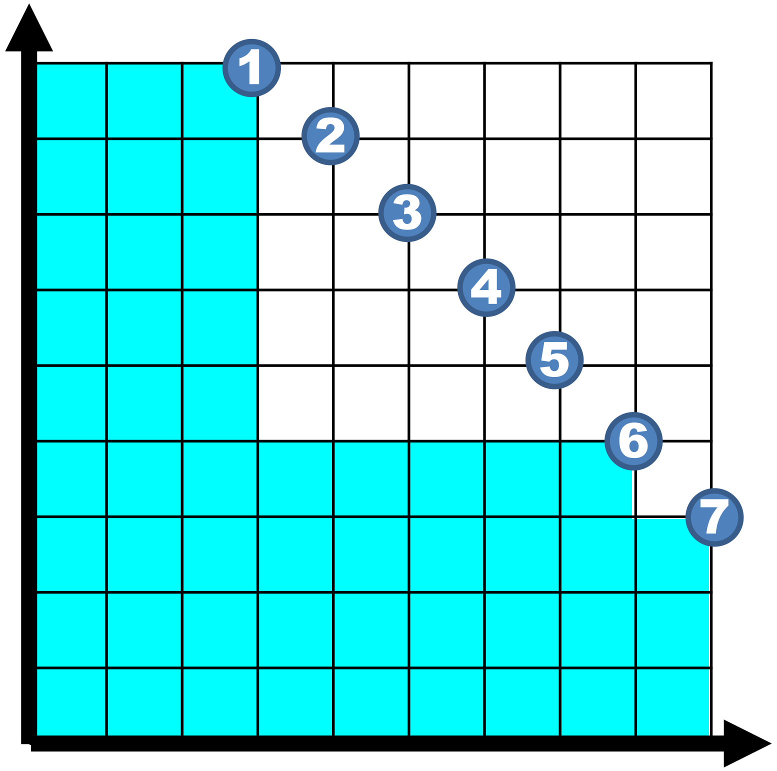
Explanation for *k* = 2

Greedy subset: {Item 1, Item 4}

1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).

2. Contribution of each item is evaluated. The worst item (Item 4) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 7 is added).



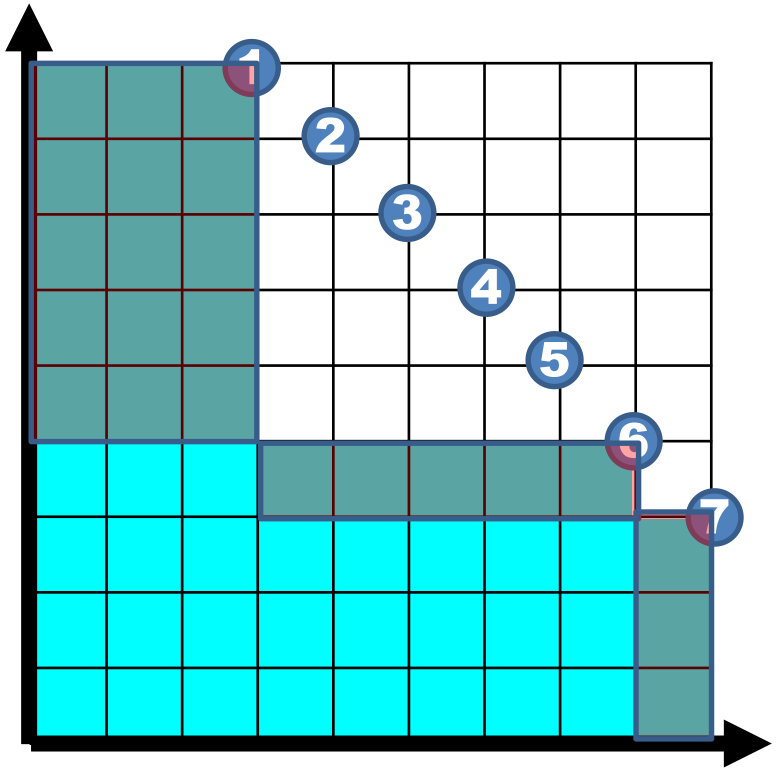
Explanation for *k* = 2

Greedy subset: {Item 1, Item 4}

1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).

2. Contribution of each item is evaluated. The worst item (Item 4) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 7 is added).



Explanation for *k* = 2

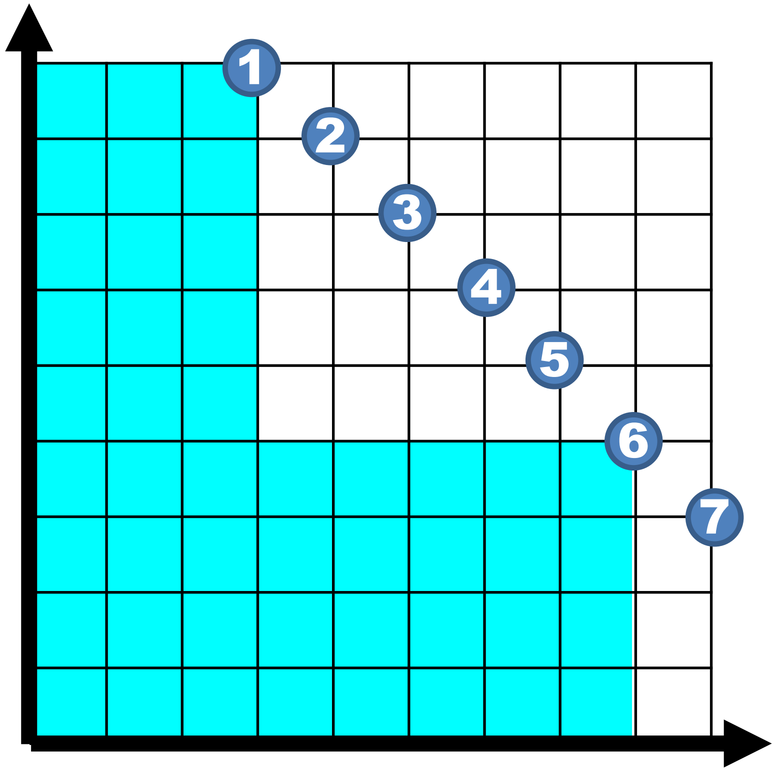
Greedy subset: {Item 1, Item 4}

1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).

2. Contribution of each item is evaluated. The worst item (Item 4) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 7 is added).

2. Contribution of each item is evaluated. The worst item (Item 7) is removed.



Explanation for k = 2

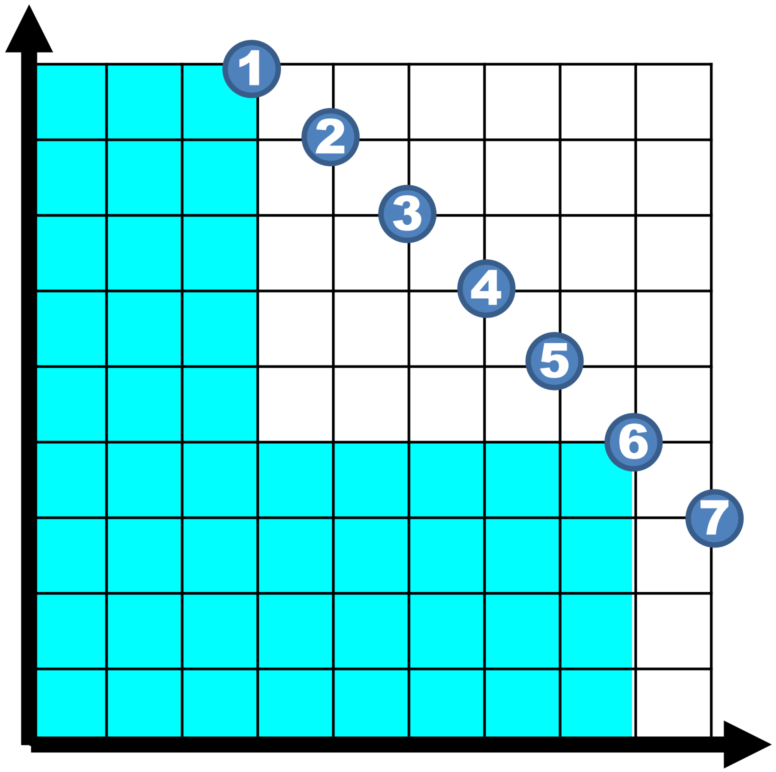
Greedy subset: {Item 1, Item 4}

1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).

2. Contribution of each item is evaluated. The worst item (Item 4) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 7 is added).

2. Contribution of each item is evaluated. The worst item (Item 7) is removed.



Explanation for *k* = 2

Greedy subset: {Item 1, Item 4}

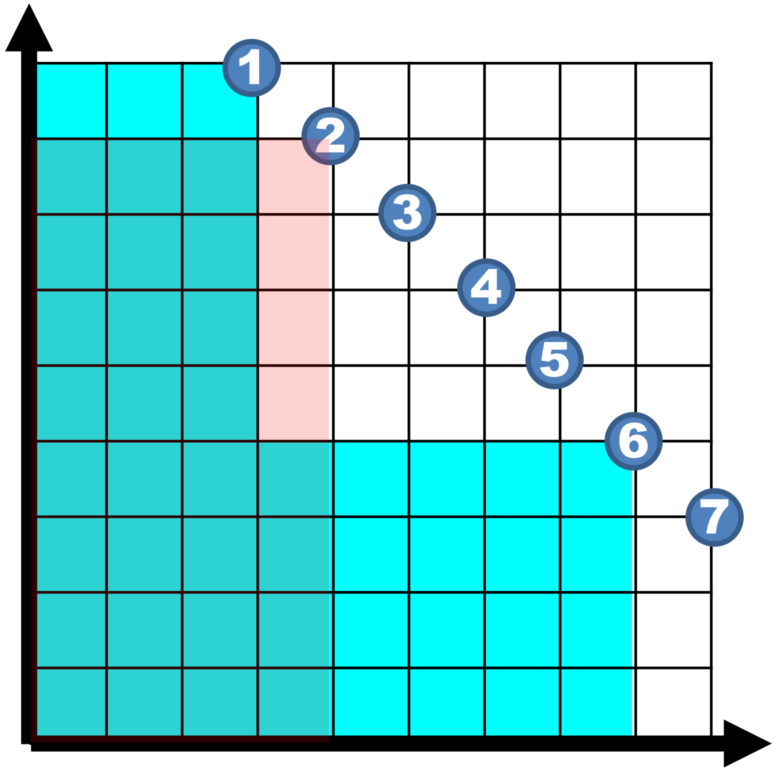
1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).

2. Contribution of each item is evaluated. The worst item (Item 4) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 7 is added).

2. Contribution of each item is evaluated. The worst item (Item 7) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 2 is added).



Explanation for *k* = 2

Greedy subset: {Item 1, Item 4}

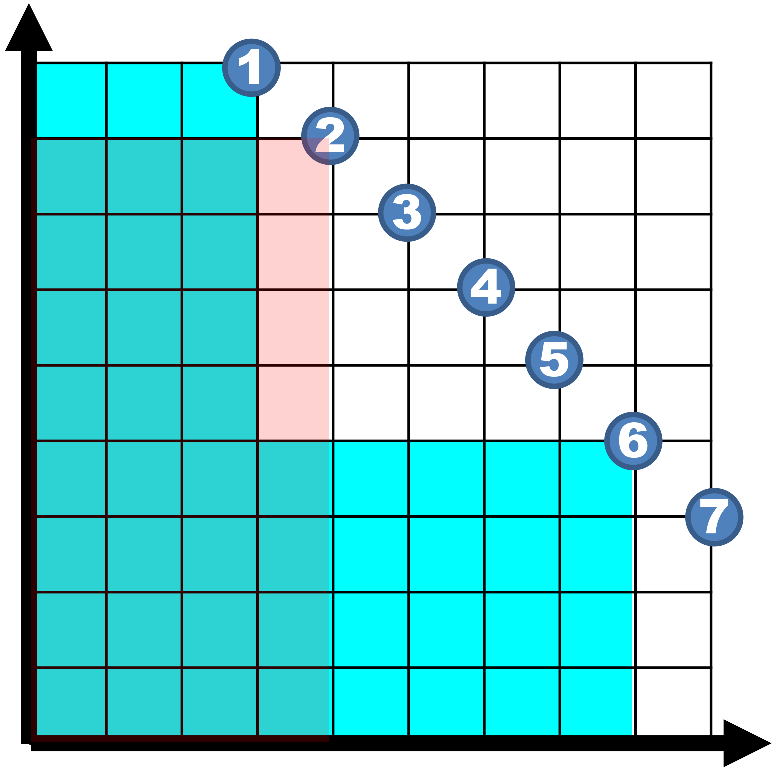
1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).

2. Contribution of each item is evaluated. The worst item (Item 4) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 7 is added).

2. Contribution of each item is evaluated. The worst item (Item 7) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 2 is added).



Explanation for *k* = 2

Greedy subset: {Item 1, Item 4}

1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).

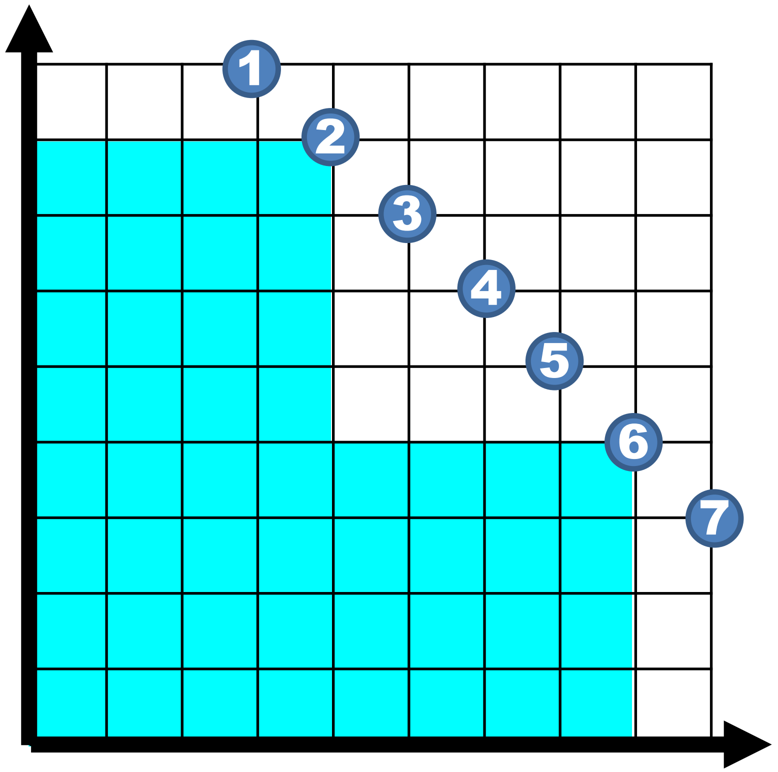
2. Contribution of each item is evaluated. The worst item (Item 4) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 7 is added).

2. Contribution of each item is evaluated. The worst item (Item 7) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 2 is added).

2. Contribution of each item is evaluated. The worst item (Item 1) is removed.



Explanation for *k* = 2

Greedy subset: {Item 1, Item 4}

1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).

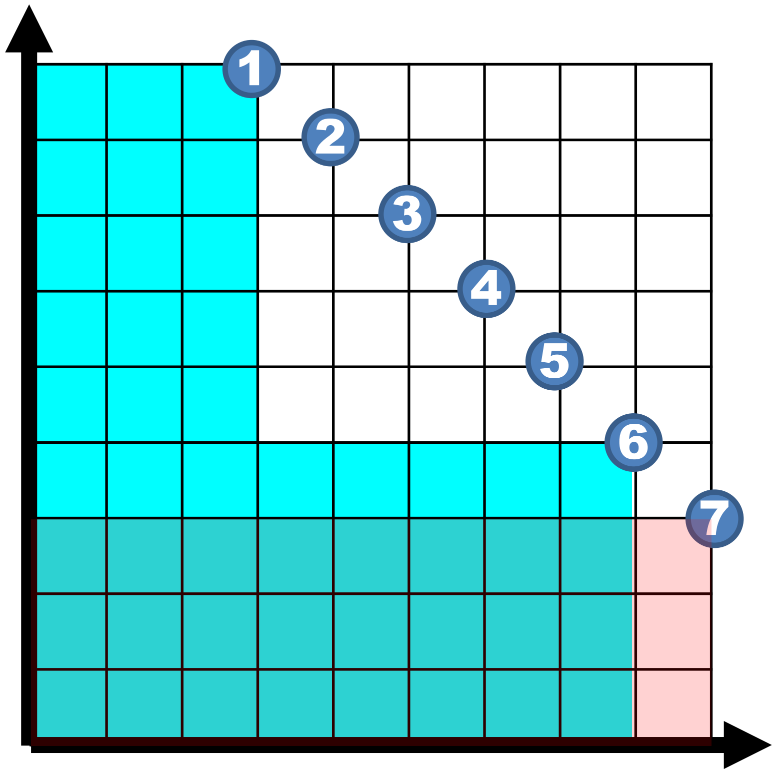
2. Contribution of each item is evaluated. The worst item (Item 4) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 7 is added).

2. Contribution of each item is evaluated. The worst item (Item 7) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 2 is added).

2. Contribution of each item is evaluated. The worst item (Item 1) is removed.



Explanation for *k* = 2

Greedy subset: {Item 1, Item 4}

1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).

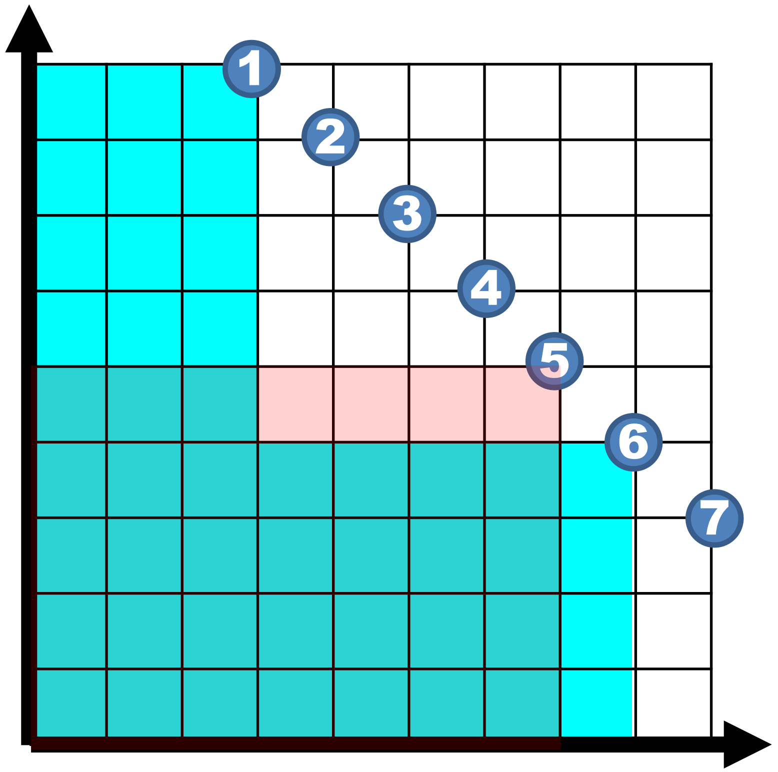
2. Contribution of each item is evaluated. The worst item (Item 4) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 7 is added).

2. Contribution of each item is evaluated. The worst item (Item 7) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (**Item ? is added**).

Important Issue: How to choose one item to be added ? Random is not efficient. **Item 7 is not good.**



Explanation for *k* = 2

Greedy subset: {Item 1, Item 4}

1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).

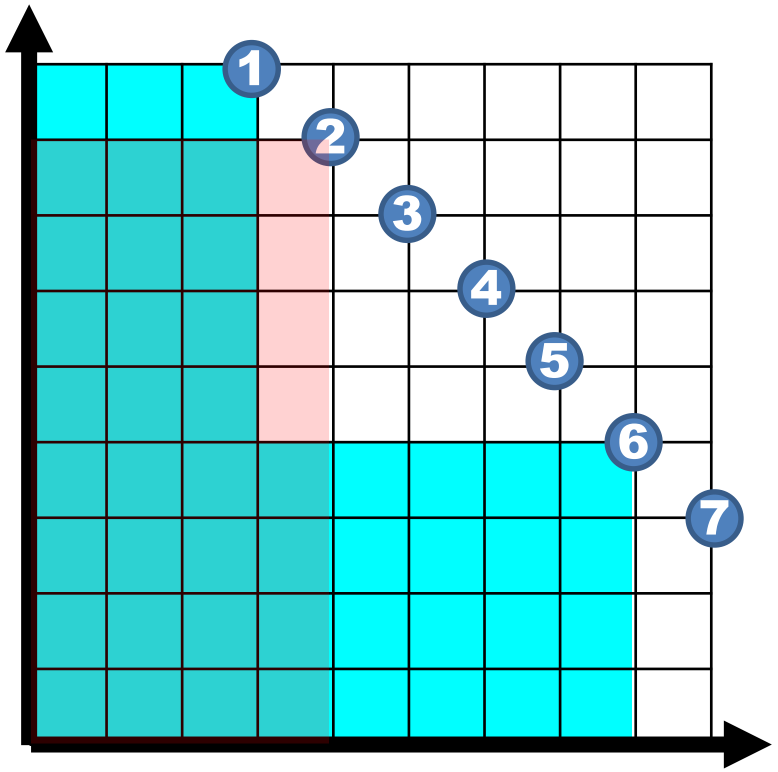
2. Contribution of each item is evaluated. The worst item (Item 4) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 7 is added).

2. Contribution of each item is evaluated. The worst item (Item 7) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (**Item ? is added**).

Important Issue: How to choose one item to be added ?
Random is not efficient. **Item 5 is not good.**



Explanation for *k* = 2

Greedy subset: {Item 1, Item 4}

1. One of the remaining items (2, 3, 5, 6, 7) is randomly added (Item 6 is added).

2. Contribution of each item is evaluated. The worst item (Item 4) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (Item 7 is added).

2. Contribution of each item is evaluated. The worst item (Item 7) is removed.

1. One of the remaining items (2, 3, 4, 5, 7) is randomly added (**Item ? is added**).

Important Issue: How to choose one item to be added ?
Random is not efficient. **Item 2 is good.**

An Example: Hypervolume Subset Selection

Search Space Size: ${}_{n}C_{k}$

How to solve this problem:

Small size problem (e.g., $n = 20$, $k = 2$)

We can find the optimal subset by examining all combination.
Research topic: Design an efficient optimization algorithm.

Medium size problem (e.g., $n = 200$, $k = 20$)

Various approximation algorithm can be used. LS, SA, EA, ...
Research topic: Search for near-optimal subsets.

Large size problem (e.g., $n = 2000$, $k = 200$)

Greedy algorithm: In the i-th iteration, $(n - i + 1)$ solutions are examined (e.g., n solutions are examined when $i = 1$) to choose one solution to be added.

Local improvement: In each iteration, $(k + 1)$ solutions are examined to find the worst solution in the current solution set.

An Example: Hypervolume Subset Selection

Search Space Size: ${}_nC_k$

How to solve this problem:

Small size problem (e.g., $n = 20$, $k = 2$)

We can find the optimal subset by examining all combination.
Research topic: Design an efficient optimization algorithm.

Medium size problem (e.g., $n = 200$, $k = 20$)

Various approximation algorithm can be used. LS, SA, EA, ...
Research topic: Search for near-optimal subsets.

Large size problem (e.g., $n = 2000$, $k = 200$)

Greedy algorithm: In the i-th iteration, $(n - i + 1)$ solutions are examined (e.g., n solutions are examined when $i = 1$) to choose one solution to be added.

Local improvement: In each iteration, $(k + 1)$ solutions are examined to find the worst solution in the current solution set.

Extremely large size problem (e.g., $n = 500{,}000{,}000$, $k = 500$):

Which algorithm can we use? Greedy, local, both, or none?

Large size problem (e.g., $n = 2000$, $k = 200$)

Greedy algorithm: In the i-th iteration, $(n - i + 1)$ solutions are examined (e.g., n solutions are examined when $i = 1$) to choose one solution to be added. For example, when $n = 2{,}000$, we need to examine about 2,000 solutions to choose one solution in each iteration.

Local improvement: In each iteration, $(k + 1)$ solutions are examined to find the worst solution in the current solution set. For example, when $k = 200$, we need to examine 201 solutions in each iteration.

Extremely large size problem (e.g., $n = 500{,}000{,}000$, $k = 500$):

Greedy algorithm: When $n = 500$ million, we need to examine about 500 million solutions in each iteration.

Q1. How can we improve the efficiency of the greedy algorithm?

Your idea? (We need to examine 500,000,000 items)

Local improvement: When $k = 500$, we need to examine 501 solutions in each iteration.

Q2. How to choose a single item to be added? **Your idea ?**

Lab Session

Task 1. When the candidate set size is huge (e.g., 500 million items), the greedy algorithm will become very slow since it needs to examine a huge number of items (about 500 million items) in each iteration to choose a single item. Please explain your own idea about how to significantly decrease the computation time of the greedy algorithm at the cost of slight deterioration of the selected subset quality.

Task 2. When the candidate set size is huge (e.g., 500 million items), random selection of one item in local improvement seems to be ineffective. Please explain your own idea about how to choose a single item to be added to the current subset in local improvement.