NP 難題演算法系列 · 第 10 篇

集合覆蓋問題

Set Cover — The Greedy $\ln n$ Approximation

用最少的集合蓋住全部元素：貪心法的近似保證與它「已是最優」的驚人事實

Contents

1 問題是什麼

白話比喻

市政府要設消防站：每個候選站址能涵蓋幾個街區，全市街區都得有人罩。**最少建幾個站**？——街區是元素、站址是集合，這就是 Set Cover。

定義 1.1 (Set Cover)**.** 給定全集 U（$|U| = n$）與子集族 $\mathcal{S} = \{S_1, \ldots, S_m\}$（$\bigcup S_i = U$），找最小的 $\mathcal{C} \subseteq \mathcal{S}$ 使 $\bigcup_{S \in \mathcal{C}} S = U$。NP-Hard（Karp 21）；Vertex Cover 是它的特例（元素 = 邊、集合 = 每個點的鄰接邊集）。

2 貪心演算法

2.1 規則只有一句話

每一輪，挑「新覆蓋元素最多」的集合，直到蓋完。

```
while (covered != universe) {
    pick S maximizing |S \ covered|;    // 新增覆蓋最多
    covered |= S;
}
```

時間 $O(mn)$ 每輪、最多 n 輪；用優先佇列可到 $O(\sum |S_i| \log m)$。

2.2 近似保證

定理 2.1 (貪心 $= H_n$ 近似)**.** 貪心解 $\le H_n \cdot OPT$，其中 $H_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n} \le \ln n + 1$。

證明（攤帳論證）. 設最優用 $k = OPT$ 個集合。任何時刻若還剩 r 個元素未蓋，這些元素能被最優的 k 個集合蓋住，由鴿籠原理**存在**一個集合能蓋住其中 $\ge r/k$ 個——貪心挑的只會更多。所以每輪至少消滅剩餘的 $1/k$，r 輪後剩餘 $\le n(1-\frac{1}{k})^r < ne^{-r/k}$。當 $r = k \ln n$ 時剩餘 < 1 即為 0，故貪心至多 $k \ln n + k$ 輪。 □

筆記

更驚人的是下界：Dinur & Steurer (2014) 證明，除非 P=NP，多項式時間內**不可能**做出 $(1-\varepsilon)\ln n$ 倍的近似。也就是說——**這個一句話的貪心法，已經是人類可能做到的最好**。教科書上「貪心不保證最優」的刻板印象，在 Set Cover 這裡要反過來讀：不最優，但已是極限。

2.3 手算範例：貪心失手的瞬間

$U = \{1, \ldots, 14\}$，三個集合：

$$S_0 = \{1..7\}\ (\text{上半}), \quad S_1 = \{8..14\}\ (\text{下半}), \quad S_2 = \{1,2,3,4,8,9,10,11\}\ (\text{誘餌, 8 個元素})$$

最優解：$S_0 + S_1$，兩個集合。貪心：第一輪 S_2 最大（$8 > 7$）被咬走 $\Rightarrow$ 剩下上下各 3 個元素，還得拿 S_0、S_1——共 3 個。比率 3/2。把這種構造遞迴疊起來（2^k 個元素、誘餌一層比一層小），比率可以推到 $\Theta(\log n)$：貪心的上界是緊的。

3 精確解：小實例的位元枚舉

m 個集合枚舉 2^m 種組合，用 bitmask 秒級驗證覆蓋。$m \le 25$ 可行，價值同樣是**當標準答案**：本篇程式 Example 4 用它對 200 個隨機實例統計貪心的實際表現。

4 完整 C++ 程式

包含：貪心（含每輪 log）、位元枚舉精確解、教科書例、貪心失手構造、消防站選址、200 實例統計。編譯：

```
g++ -std=c++17 -O2 -Wall -Wextra -o set_cover set_cover.cpp
```

```
// set_cover.cpp
// ------------------------------------------------------------------
// Set Cover（集合覆蓋）：
//   1. 貪婪近似：每輪挑「新覆蓋元素最多」的集合，保證 <= (ln n + 1)· OPT
```

```cpp
//   2. 精確解：位元枚舉（子集合數量小時）
//   3. 貪婪比最優差的經典構造範例
//
// Compile: g++ -std=c++17 -O2 -Wall -Wextra -o set_cover set_cover.cpp
// ---------------------------------------------------------------
#include <iostream>
#include <vector>
#include <algorithm>
#include <cmath>

using namespace std;

// 全集 = {0, 1, ..., n-1}，每個集合用 bitmask（n <= 30）
// ---------- 1. 貪婪近似 ----------
vector<int> setCoverGreedy(int n, const vector<unsigned>& sets, bool
    verbose = false) {
    unsigned universe = (n == 32) ? ~0u : ((1u << n) - 1);
    unsigned covered = 0;
    vector<int> picked;
    int round = 0;
    while (covered != universe) {
        int best = -1, bestGain = 0;
        for (int i = 0; i < (int)sets.size(); ++i) {
            int gain = __builtin_popcount(sets[i] & ~covered);
            if (gain > bestGain) { bestGain = gain; best = i; }
        }
        if (best == -1) break;                              //
            蓋不完（輸入有問題）
        covered |= sets[best];
        picked.push_back(best);
        if (verbose)
            cout << "  round " << ++round << ": pick S" << best
                 << " (+" << bestGain << " new, covered "
                 << __builtin_popcount(covered) << "/" << n << ")\n";
    }
    return picked;
}

// ---------- 2. 精確解：枚舉所有集合組合 O(2^m) ----------
vector<int> setCoverExact(int n, const vector<unsigned>& sets) {
    int m = (int)sets.size();
    unsigned universe = (n == 32) ? ~0u : ((1u << n) - 1);
    int best = m + 1;
    unsigned bestMask = 0;
    for (unsigned mask = 0; mask < (1u << m); ++mask) {
        if (__builtin_popcount(mask) >= best) continue;
        unsigned covered = 0;
        for (int i = 0; i < m; ++i)
            if ((mask >> i) & 1) covered |= sets[i];
        if (covered == universe) {
            best = __builtin_popcount(mask);
            bestMask = mask;
        }
    }
    vector<int> out;
    for (int i = 0; i < m; ++i)
        if ((bestMask >> i) & 1) out.push_back(i);
    return out;
}
```

```cpp

static unsigned makeSet(const vector<int>& elems) {
    unsigned s = 0;
    for (int e : elems) s |= 1u << e;
    return s;
}

static void printPick(const string& tag, const vector<int>& p) {
    cout << tag << " uses " << p.size() << " sets: {";
    for (size_t i = 0; i < p.size(); ++i)
        cout << "S" << p[i] << (i + 1 < p.size() ? "," : "");
    cout << "}\n";
}

int main() {
    // ---- 範例 1：教科書例（貪婪=最優） ----
    cout << "=== Example 1: textbook instance ===\n";
    // 全集 {0..11}，6 個集合
    vector<unsigned> sets1 = {
        makeSet({0,1,2,3}),           // S0
        makeSet({4,5,6,7}),           // S1
        makeSet({8,9,10,11}),         // S2
        makeSet({0,4,8}),             // S3
        makeSet({1,5,9}),             // S4
        makeSet({2,3,6,7,10,11})      // S5
    };
    cout << "universe = {0..11}\n";
    auto g1 = setCoverGreedy(12, sets1, true);
    printPick("greedy", g1);
    printPick("exact ", setCoverExact(12, sets1));
    cout << "\n";

    // ---- 範例 2：讓貪婪失手的構造 ----
    cout << "=== Example 2: instance where greedy loses ===\n";
    // 全集 {0..13}。OPT = 2（S0, S1 各蓋一半）
    // 但貪婪先咬走大集合 S2（8 個），之後還要 2 個 -> 用 3 個
    vector<unsigned> sets2 = {
        makeSet({0,1,2,3,4,5,6}),        // S0: 上半
        makeSet({7,8,9,10,11,12,13}),    // S1: 下半
        makeSet({0,1,2,3,7,8,9,10}),     // S2: 誘餌（8 個元素，最大）
    };
    auto g2 = setCoverGreedy(14, sets2, true);
    printPick("greedy", g2);
    printPick("exact ", setCoverExact(14, sets2));
    cout << "greedy/OPT = " << (double)g2.size() / 2 << "  (bound:
       ln(14)+1 = "
         << log(14.0) + 1 << ")\n\n";

    // ---- 範例 3：消防站選址情境 ----
    cout << "=== Example 3: fire station placement ===\n";
    // 10 個街區，5 個候選站址，各站可涵蓋的街區：
    vector<unsigned> stations = {
        makeSet({0,1,2}),         // 站 0
        makeSet({2,3,4,5}),       // 站 1
        makeSet({5,6,7}),         // 站 2
        makeSet({7,8,9}),         // 站 3
        makeSet({1,3,6,8})        // 站 4
    };
    cout << "10 blocks, 5 candidate stations\n";
```

```
    auto g3 = setCoverGreedy(10, stations, true);
    printPick("greedy", g3);
    printPick("exact ", setCoverExact(10, stations));
    cout << "\n";

    // ---- 範例 4：理論界的驗證——隨機大量實例 ----
    cout << "=== Example 4: greedy vs OPT over random instances ===\n";
    unsigned seed = 555;
    auto nextRand = [&seed]() {
        seed = seed * 1103515245 + 12345;
        return (seed >> 16) & 0x7fff;
    };
    int n = 16, m = 12, trials = 200;
    double worstRatio = 1.0;
    int greedyWins = 0, ties = 0;
    for (int t = 0; t < trials; ++t) {
        vector<unsigned> ss;
        unsigned covered = 0;
        for (int i = 0; i < m; ++i) {
            unsigned s = 0;
            for (int e = 0; e < n; ++e)
                if (nextRand() % 100 < 30) s |= 1u << e;
            ss.push_back(s);
            covered |= s;
        }
        if (covered != (1u << n) - 1) { --t; continue; }  // 重抽：必須可覆蓋
        auto g = setCoverGreedy(n, ss);
        auto e = setCoverExact(n, ss);
        double r = (double)g.size() / e.size();
        worstRatio = max(worstRatio, r);
        if (g.size() == e.size()) ++ties;
        else ++greedyWins;                                  // 其實是 greedy 較差
    }
    cout << trials << " random instances (n=16, m=12, p=0.3):\n";
    cout << "greedy == OPT: " << ties << " times, greedy > OPT: "
         << greedyWins << " times\n";
    cout << "worst greedy/OPT ratio observed = " << worstRatio
         << "  (theory bound: ln(16)+1 = " << log(16.0) + 1 << ")\n";
    return 0;
}
```

程式碼 1: set_cover.cpp — 集合覆蓋貪心與精確解完整實作

5 執行結果與解讀

```
=== Example 2: instance where greedy loses ===
  round 1: pick S2 (+8 new, covered 8/14)
  round 2: pick S0 (+3 new, covered 11/14)
  round 3: pick S1 (+3 new, covered 14/14)
greedy uses 3 sets: {S2,S0,S1}
exact  uses 2 sets: {S0,S1}
greedy/OPT = 1.5  (bound: ln(14)+1 = 3.64)

=== Example 3: fire station placement ===
10 blocks, 5 candidate stations
  round 1: pick S1 (+4 new, covered 4/10)
  round 2: pick S3 (+3 new, covered 7/10)
```

```
  round 3: pick S0 (+2 new, covered 9/10)
  round 4: pick S2 (+1 new, covered 10/10)
greedy uses 4 sets: {S1,S3,S0,S2}
exact  uses 4 sets: {S0,S1,S2,S3}

=== Example 4: greedy vs OPT over random instances ===
200 random instances (n=16, m=12, p=0.3):
greedy == OPT: 151 times, greedy > OPT: 49 times
worst greedy/OPT ratio observed = 1.5  (theory bound: ln(16)+1 = 3.77)
```

筆記

Example 4 是「理論界 vs 實際表現」的好教材：理論保證 3.77 倍，但 200 個隨機實例中貪心 75% 直接命中最優、最差也只有 1.5 倍。**近似比是最壞情況的保險，不是日常的預期**——會構造壞例的人（Example 2）與會讀統計的人，才算真的懂近似演算法。

6 實務應用

- **設施選址**：消防站、基地台、疫苗接種點、無人機補給站的最少布點。
- **測試最小化**：用最少的測試用例覆蓋所有程式分支——CI 加速的核心手法。
- **病毒特徵庫**：挑最少的特徵碼集合偵測所有已知樣本。
- **機組排班**：航空公司的 crew scheduling 是加權 Set Cover 的巨型實例，以列生成 + LP 解。
- **資料摘要**：挑最少的句子覆蓋文件的所有主題詞（extractive summarization）。

7 練習題

1. 手算：$U = \{1..6\}$、$S_0 = \{1,2,3\}$、$S_1 = \{4,5\}$、$S_2 = \{6\}$、$S_3 = \{1,4,6\}$、$S_4 = \{2,5\}$，貪心與最優各是多少？
2. 把貪心改成**加權**版本（每集合有成本，每輪挑「成本/新覆蓋數」最小的），保證仍是 H_n 倍。
3. 依照手算範例的思路，構造一個貪心 $= \Theta(\log n)$ 倍的三層實例（$n = 2^3 \cdot 2$ 個元素）並用程式驗證。
4. 實作 LP 鬆弛 + 隨機捨入的近似（進階），與貪心比較。
5. 挑戰：Max Coverage 變形——最多挑 k 個集合，最大化覆蓋元素數。證明貪心給 $1 - 1/e$ 保證。

8 小結

方法	時間	品質	適用時機
貪心	$O(\sum \lvert S_i \rvert \log m)$	$\le (\ln n + 1)OPT$，理論最優	幾乎所有場合
位元枚舉	$O(2^m m)$	精確	$m \le 25$、驗證用
ILP 求解器	指數（實務常可行）	精確	中大型、有商用 solver
LP + 捨入	多項式	$O(\log n)$ 倍	理論分析、加權變形

延伸閱讀：Johnson (1974)、Chvátal (1979) 貪心分析；Dinur & Steurer (2014) $\ln n$ 不可近似性；Vazirani *Approximation Algorithms* 第 2 章。