CONSTRUCTIVE HEURISTIC FOR THE SCP

Heuristics and Metaheuristics | 2020/2021

Tiago Filipe Sousa Gonçalves | PDEEC

General Aspects: Solution Representation

Solution Representation: an array that contains the sets (columns indices) needed to cover the entire universe of elements (rows)

$$[\text{Column}_i, ..., \text{Column}_n] \text{ for } i \in [0, n) \text{ if } \text{Column}_i \in \text{Solution}$$

n = number of sets of the set covering instance

Note: Column indices start at 0 because the algorithms were implemented in Python

General Aspects: Monitored Variables

Rows already covered: the list of elements that our current solution already covers

Candidate columns to be added: the list of sets that are still available to be added to the solution

Number of columns that cover each row: the frequency of sets per element

Current solution: the sets that we added to our solution in a specific iteration

Current cost: the total cost of our solution in a specific iteration

General Aspects: Set Redundancy Removal

```
1) Start from the current solution
2) While the number of rows already covered < number of rows to be covered:
    a) Check the set (column) which covers more rows
    b) Check if this set (column) contains other set(s)
        i) If this set set contains other set(s):
            (1) Remove the other (smaller) sets from the current solution
3) The processed solution is the array of sets after Step 2)
```

General Aspects: Cost Effectiveness

Cost-Effectiveness (ɑ) is given by **ɑ = Cost(S) / |S - X|** where:

- **S** is the candidate set that is being evaluated in a specific iteration
- **Cost(S)** is the cost of the candidate set *S*
- **X** is the set of elements covered by the current solution
- **|S - X|** is the difference between the elements covered by set *S* and the elements covered by the current solution *X*

Constructive Heuristic #1

1) While the number of rows already covered < number of rows to be covered:
    a) Pick a random element (row) to be covered
    b) Check all the sets (columns) that contain this element (row)
    c) For each of these possible sets (columns) compute the **cost-effectiveness**
    d) Pick the set with **lower** cost-effectiveness value and add it to the solution
    e) Check the **rows that are covered by this set** and **update** the number of rows already covered
2) If post-processing is **enabled**, apply *Set Redundancy Removal*

Constructive Heuristic #2

1) While the number of rows already covered < number of rows to be covered:
    a) Check all the sets (columns) that are available to be added
    b) For each of these possible sets (columns) compute the **cost-effectiveness**
    c) Pick the set with **lower** cost-effectiveness value and add it to the solution
    d) Check the **rows that are covered by this set** and **update** the number of rows already covered
2) If post-processing is **enabled**, apply *Set Redundancy Removal*

Constructive Heuristic #3

1) Check the element (row) with higher frequency, ***f***
2) Solve LP and obtain x_j^*, $j \in [0, n)$
3) While the number of rows already covered < number of rows to be covered:
    a) For each $\mathbf{x_j^* > 1/f}$
        i) Add x_j to the solution
4) If post-processing is **enabled**, apply *Set Redundancy Removal*

Results

CH #	Average % Deviation from Best Known Solutions	Fraction of Instances that Profit from Redundancy Elimination
CH1	**17,86**	0
CH2	328,31	**1**
CH3	158,76	1/4

Discussion and Conclusions

- CH1 attained the best results, however, quite far from the optimal solution
- CH2 was the worst in terms of performance
- CH3 benefited most from the set redundancy routine

References

Akhter, Fatema. "A heuristic approach for minimum set cover problem." Int. J. Adv. Res. Artif. Intell 4 (2015): 40-45.

Beasley, John E. "An algorithm for set covering problem." European Journal of Operational Research 31.1 (1987): 85-93.