Introduction to Local Search

Carlos M. Fonseca

University of Coimbra

Outline

- Decision space
- Neighbourhood structure
- Neighbourhood exploration
  - Local move enumeration
  - Random sampling with replacement
  - Random sampling without replacement

Decision space

The *decision space* is the *set of all candidate solutions* that may be *visited* during the solving process

- Only *feasible* solutions are considered in local search

  $\Rightarrow$ The objective function is defined on the whole decision space

  - The internal solution structure may not be specified
  - Solutions are *points* in decision space

- Initial feasible solution(s) may be:
  - Generated at random
  - Obtained by constructive search
  - Obtained heuristically in other ways

Decision space

Examples

How to generate solutions *uniformly at random* for the following problems?

- Travelling salesman

Decision space

Examples

How to generate solutions *uniformly at random* for the following problems?

- Travelling salesman
  - Generate a random permutation of the N cities
- Cable-trench problem (solutions are spanning trees)

Decision space

Examples

How to generate solutions *uniformly at random* for the following problems?

- Travelling salesman
  - Generate a random permutation of the N cities
- Cable-trench problem (solutions are spanning trees)
  - By Cayley's theorem, there are N^{N-2} labeled spanning trees in a complete graph with N nodes
  - Generate a vector of $N-2$ integers in $\{1, \ldots, N\}$ uniformly at random
  - Decode the resulting *Prüfer sequence* into a spanning tree in linear time

    (see `https://doi.org/10.4236/jsea.2009.22016`)

Decision space

Examples

How to generate solutions *uniformly at random* for the following problems?

- Binary knapsack problem

Decision space

Examples

How to generate solutions *uniformly at random* for the following problems?

- Binary knapsack problem
  - Generate a random binary vector of length equal to the number of available items and reject sets of items that do not meet the capacity constraint(s)

  ⚠ Possibly (very) inefficient if the problem is tightly constrained
- Clique partitioning problem

Decision space

Examples

How to generate solutions *uniformly at random* for the following problems?

- Binary knapsack problem
  - Generate a random binary vector of length equal to the number of available items and reject sets of items that do not meet the capacity constraint(s)

  ⚠ Possibly (very) inefficient if the problem is tightly constrained
- Clique partitioning problem
  - Draw a number of bins at random according to a *specific* distribution (see `https://doi.org/10.1016/0097-3165(83)90009-2`)
  - Randomly assign each vertex to one of those bins (some bins may end up empty)

Neighbourhood structure

What are similar solutions?

- “Parts” of the two solutions are somehow identical
- Similar performance (in most cases)
- Connect the whole space

Example 1

Symmetric Travelling Salesman Problem

- Tour length is the sum of the lengths of the tour edges
- Solutions are similar if they differ in a small number of edges
- 2-opt and 3-opt moves

Neighbourhood structure

Example 2

Asymmetric Travelling Salesman Problem

- Tour length is the sum of the lengths of the tour arcs
- Solutions are similar if they differ in a small number of arcs
- Due to asymmetry, 2-opt moves and some 3-opt moves are not suitable because they reverse parts of the tour
- Insertion move as a special case of 3-opt

Note: What about neighbourhood size?

Neighbourhood exploration

Local search in a nutshell

1. Visit neighbours of a current solution
2. Decide whether to reject them or to accept one as the next solution
3. Repeat

Local move generation

- Enumeration
- Random sampling with replacement
- Random sampling without replacement

Neighbourhood exploration

Local move enumeration

- Full neighbourhood exploration
- Enumeration order dictated by convenience
- Filter out invalid moves if needed
- Neighbourhood size may not be known in advance

Random sampling with replacement

- Sampling uniformly at random typically preferred
- Rejection sampling useful when neighbourhood size not known in advance

Neighbourhood exploration

Random sampling without replacement

- Partial to full neighbourhood exploration
- Random enumeration order to avoid search bias
- Filter out invalid moves if needed (rejection)
- Neighbourhood size may not be known in advance
- Generating (pseudo-)random permutations
  - Fisher-Yates shuffle
  - Linear congruential generator (low quality but constant space)
  - A plethora of other approaches

Neighbourhood exploration

Random sampling without replacement

Fisher-Yates shuffle

```
def fisher_yates_iter(n):
    p = list(range(n))
    for i in range(n-1, -1, -1):
        r = random.randrange(i+1)
        yield p[r]
        # p[r], p[i] = p[i], p[r] # preserve the permutation
        p[r] = p[i] # lazy
```

Neighbourhood exploration

Random sampling without replacement

Sparse Fisher-Yates shuffle

```
def sparse_fisher_yates_iter(n):
    p = dict()
    for i in range(n-1, -1, -1):
        r = random.randrange(i+1)
        yield p.get(r, r)
        if i != r:
            # p[r] = p.pop(i, i) # saves memory, takes time
            p[r] = p.get(i, i) # lazy, but faster
```

Concluding remarks

- Feasible solutions are just points in decision space
- Internal solution structure is not exposed
- Neighbourhood structure should
  - Capture the similarity of solutions
  - Connect the whole decision space
- Neighbourhood enumeration and random sampling

This presentation is based upon work from COST Action Randomised Optimisation Algorithms Research Network (ROAR-NET), CA22137, supported by COST (European Cooperation in Science and Technology). This work is funded by national funds through FCT – Foundation for Science and Technology, I.P., within the scope of the research unit UID/00326 – Centre for Informatics and Systems of the University of Coimbra.

COST (European Cooperation in Science and Technology) is a funding agency for research and innovation networks. Our Actions help connect research initiatives across Europe and enable scientists to grow their ideas by sharing them with their peers. This boosts their research, career and innovation.

www.cost.eu