Fundamentals of Optimization Heuristics

Pham Quang Dung

dungpq@soict.hust.edu.vn

Department of Computer Science

Content

- Overview
- Examples

Overview

- Exact methods (Constraint Programming, Branch and Cut, ..) cannot handle large-scale combinatorial optimization problems
- In practice, high quality solutions found in a reasonable computation time are required

Overview

- **S**: a solution is represented by a set of components
- **C**: set of candidates of components to be added to the solution
- **select(C)**: select the most promising component among candidates
- **solution(S)**: return true if S is a solution to the original problem
- **feasible(S)**: return true if S does not violate any constraints

```
Greedy() {
  S = {};
  while C ≠ ∅ and
        not solution(S){
    x = select(C);
    C = C \ {x};
    if feasible(S ∪ {x}) {
      S = S ∪ {x};
    }
  }
  return S;
}
```

TSP

- Given n points 1, 2, …, n in which d(i,j) is the distance from point i to point j. Find the shortest closed tour starting from 1 visiting other points and terminating at 1 such that the total travel distance is minimal

TSP

- Greedy idea
  - The tour is initialized by point 1
  - At each step
    - Select the nearest point to the last point of the tour under construction and add this point to the end of the tour

TSP

- Greedy idea
  - The tour is initialized by point 1
  - At each step
    - Select the nearest point to the last point of the tour under construction and add this point to the end of the tour

Multi knapsack problem

- Given unlimited number of bins having capacity Q and n items 1,2,…, n in which the weight of item i is w(i). How to put these n items into bins such that the total weight of items put into each bin cannot exceed Q and the number of bins used is minimal