Short Notes about Discrete Optimization

References

[1] https://www.coursera.org/learn/discrete-optimization/home/welcome

Versions

10-AUG-2020 , First draft. Konstantin Burlachenko

Terminology

CP - Constraint Programming

LS - Local Search

MIP - Mixed Integer Programming

Discrete Optimization Tools

1. CP Solvers
2. MIP Solvers
3. LS Solvers

1. Modeling with Math Optimization

Step #1: Choose some decision variables

Step #2: Encode the result we are interested in terms of decision variables

Step #3: Express the problem constraints in terms of these variables

Step #4: Express the objective function. Objective function specifies the quality of each solution.

Optimization Models specify *"what"*, not *"how"*. There are many ways to model an optimization problems.

2. Greedy Algorithms for heuristic search

Greedy algorithms one of heuristics algorithm. Typically it consist of order of action in some way. The ordering means a lot. After specify ordering action is apply to optimization or feasibility problem without any backtracking.

Disadvantages of greedy algorithm:

1. For one problem, there are many possible greedy algorithms
2. Some do better on another on some specific problem data
3. In general no quality guarantees (in wild settings)
4. Quality can vary widely on the input

Advantages:

1. Quick to design and implement
2. In principle they are fast

3. Dynamic Programming as non-heuristic algorithm

Principle divide and conquer the problem and bottom up computation without repeating computation. Use already known things. Well-known technics from Computer Science.

4. Exhaustive search

Enumerate all possible solution (if they come from discrete space) and find the best one just via enumeration.

5. Branch and Bound a s non-heuristic algorithm

Branching - split the problem into a number of subproblems.

Search Strategies for perform branching: depth-first, best-first, least-discrepancy.

Bound - find an optimistic estimate of the best solution to the subproblem. Upper and Lower bound.

Key of algorithm how find an optimistic solution is relaxation of some constraints

Pruning: prunes when a node estimation is worse than the best found solution

6. Constraint Programming

Constraint programming or **CP** works with partial assignments that are being extended.

Computational paradigm is in fact Branch and Prune:

1. Use constraints to reduce the set of values that each variable can take (pruning). It can be done via directly operating in variables domains.
2. Remove values that cannot appear in any solution
3. Perform branching. Make a choice when no more deduction can be performed. In optimization most choice are wrong so solve should have ability to backtrack.

Modeling methodology:

1. Structure of the problem as explicitly as possible
2. Express substructures of the problem
3. Give solvers as much information as possible

Principles of Search/Branching:

1. Use feasibility information for branching
2. Try first decision where you are the most likely to fail
3. Implicit goal: creating small search trees
4. In need assign value for variable - choose the variable with the smallest domain
5. Choose the most constrained variable first

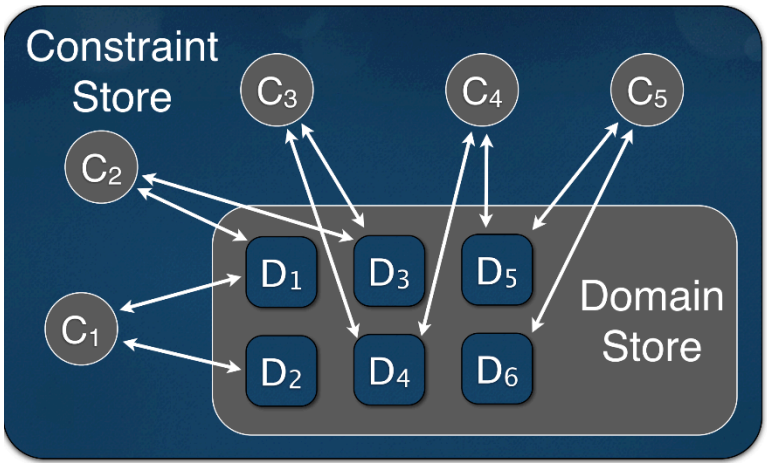
Role of constraints:

1. Feasibility checking. Can a constraint be satisfied given the values in the domains of its variables?

2. Pruning. If constraints are satisfiable, determine which values in the domains cannot be part of any solution. Pruning is performed in the propagation engine – core of constraint programming model
3. Key aspect of constraint programming is the ability to state complex, idiosyncratic constraints.
4. Constraints can have pretty general form:
    a. Element constraint
    b. Logical combination of constraints
    c. Ability to index an array/matrix with a variable or an expression containing variables
    d. Can be handled by reification for instance. Reification is transform constrain into signale (0/1) variable. The variable is **1 iff** constraint is satisfied**.**
    e. Array comprehension

Global Constraints:

1. Capture combinatorial substructures arising in many applications
2. Make modeling easier and more natural
3. Global constraints make it possible to prune the search space more



Redundant constraints:

1. Reduce the search space
2. Do not exclude any solution
3. They express properties of the solutions not captured by the model
4. Boost the propagation of other constraints

Constrain Programming allow perform optimization with following trick:

1. Solve a sequence of satisfaction problems
2. Find a solution
3. Impose a constraint that the next solution must be better then previous

7. Local Search

Key Idea: Move from configurations to configurations by performing local moves. It works with complete assignments to the decision variables and modify them.

Neighborhood:

Local moves define a neighborhood. It's configurations that are close.

In that sense Local search is a graph exploration of possible configurations.

Key Problems in Local Search:

Escaping local optima is a critical issue in local search.

How to drive a local search:

1. Start with an infeasible configuration and move towards feasibility
2. Start with suboptimal solutions and move towards optimal configuration
3. Local move – assign value to decision variable. One way:
    a. Choose a decision variable that appear in the most violations
    b. Assign to a value that minimizes its violations
4. Swaps – we have variable assignment configuration. Swap that configuration with another configuration to minimize the number of violations. Example: in Travel Salesman Problem find arbitrarily tour and then perform swapping via exchange two edges in tour.

Type of constraints during local search:

1. Soft constraints - may be violated during the search
2. Hard constraints - always feasible during the search

Steps for sophistication of Local Search:

1. After neighboring identification **N(S)** the next step split them into into two groups as legal move **L(N(S))** and illegal move.
2. Next within legal move we should have some principle to identify methods of selecting step **S(L(N(S)))**

Heuristics and Meta-Heuristics in Local Search:

1. **Heuristics drives local search**
    a. Choose the next neighbor
    b. Use local information for selection on next configuration
    c. Drive the search towards a local minimum
2. **Meta-heuristics**
    a. Aim at escaping local minima

b. Drive the search towards a global minimum
c. Typically include some memory or learning

Strategy to select one of the neighbor:

1. Select a neighbor at random
2. Use some principle

8. Local Search Meta-Heuristics

Iterated Local Search

Execute multiple local search from different starting configuration

Metropolis Heuristic

Accept a move if it improves the objective value.

In case it does not improve accept with some probability evaluated via $\exp\ exp\left(-\frac{f(n)-f(s)}{t}\right)$

Simulated Annealing

Use Metropolis algorithm but adjust the temperature (t) dynamically.

In addition technics such reheat and restarting can be applied.

Tabu Search

Maintain the sequence of nodes already visited. Select the best configurations that is not tabu, i.e., has not been visited before.

Main idea – select the best configurations that is not tabu, i.e., has not been visited before.

Key issue: expensive to maintain all the visited nodes

Intensification - store high-quality solutions and return to them periodically

Diversification - when the search is not producing improvement, diversify the current state

Strategic oscillation - change the percentage of time spent in the feasible and infeasible regions