DM812
METAHEURISTICS

Lecture 9
No Free Lunch Theorems

Marco Chiarandini

Department of Mathematics and Computer Science
University of Southern Denmark, Odense, Denmark
<marco@imada.sdu.dk>

Outline

Outline

SMTWTP

Linear permutations problems

Construction graph:
Fully connected and the set of vertices consists of the n jobs and the n positions to which the jobs are assigned.

Constraints: all jobs have to be scheduled.

Pheromone Trails: τ_{ij} expresses the desirability of assigning job i in position j (cumulative rule)

Heuristic information: $\eta_{ij} = \frac{1}{h_i}$ where h_i is a dispatching rule.

Generalized Assignment

Input:

- a set of jobs $J = \{1, \ldots, n\}$ and a set of agents $I = \{1, \ldots, m\}$.
- the cost c_{ij} and the resource requirement a_{ij} of a job j assigned to agent i
- the amount b_i of resource available to agent i

Task: Find an assignment of jobs to agents $\sigma : J \to I$ such that:

$$\begin{aligned} \min \quad & f(\sigma) = \sum_{j \in J} c_{\sigma(j)j} \\ \text{s.t.} \quad & \sum_{j \in J, \sigma(j)=i} a_{ij} \leq b_i \quad \forall i \in I \end{aligned}$$

Assignment problems

Construction Graph:
a complete graph with vertices $I \cup J$ and costs on edges. An ant walk must then consist of n couplings (i, j).

(alternatively the graph is given by $I \times J$ and ants walk through the list of jobs choosing agents. An order must be decided for the jobs.)

Constraints:

- if only feasible: the capacity constraint can be enforced by restricting the neighborhood, ie, N_i^k for a ant k at job i contains only those agents where job i can be assigned.
- if also infeasible: then no restriction

Pheromone:

Two choices:

- which job to consider next
- which agent to assign to the job

Pheromone and heuristic on:

- desirability of considering job i_2 after job i_1
- desirability of assigning job i on agent j

Subset problems

Construction graph:
Fully connected with set of vertices that corresponds to the set of columns plus a dummy vertex from where all the ants depart.

Constraints: each vertex can be visited at most once and all rows must be covered.

Pheromone Trails: associated with components (vertices); τ_j measures the desirability of including column j in solution.

Heuristic information: on the components as function of the ant's partial solution.
$\eta_j = \frac{e_j}{c_j}$ where e_j is the # of additional rows covered by j.

Capacited Vehicle Routing

Input:

- complete graph $G(V, A)$, where $V = \{0, \ldots, n\}$
- vertices $i = 1, \ldots, n$ are customers that must be visited
- vertex $i = 0$ is the single depot
- arc/edges have associated a cost c_{ij} $(c_{ik} + c_{kj} \geq c_{ij}, \forall\, i, j \in V)$
- costumers have associated a non-negative demand d_i
- a set of K identical vehicles with capacity C $(d_i \leq C)$

Task: Find collection of K circuits with minimum cost, defined as the sum of the costs of the arcs of the circuits and such that:

- each circuit visit the depot vertex
- each customer vertex is visited by exactly one circuit; and
- the sum of the demands of the vertices visited by a circuit does not exceed the vehicle capacity C.

ACO and other Metaheuristics

Greedy Randomized "Adaptive" Search Procedure (GRASP):
while *termination criterion* is not satisfied **do**
  generate candidate solution s using
   subsidiary greedy randomized constructive search
  perform subsidiary local search on s

Adaptive Iterated Construction Search:
initialize weights
while *termination criterion* is not satisfied: **do**
  generate candidate solution s using
  subsidiary randomized constructive search
  perform subsidiary local search on s
  adapt weights based on s

Squeaky Wheel:
Construct, Analyze, Prioritize

Iterated Greedy (IG):
destruct, reconstruct, acceptance criterion

Outline

No Free Lunch Theorems

Focus on general purpose black-box optimization algorithms
(= they exploit little knowledge of the problem)

Examples: simulated annealing, evolutionary algorithms

Goal analyze the matching algorithms to class of problems

$\Longrightarrow$ What can be said a priori on the performance of one or more algorithms when run once on all problems?

One would expect hill climbing outperforms hill descending and random walk...

Simplified Proof

Definitions:

No Free Lunch Theorems

NFL1: Optimal Strategy Selection

- General-purpose universal optimization strategy is theoretically impossible.
- The only way one strategy can outperform another is if it is specialized to the specific problem under consideration.

NFL2: Search Algorithms

- Assume all algorithms use the same information rationally
- No search algorithm, no matter how sophisticated,
  absent any a priori assumptions about the cost function being worked on,
  have the same expected performance.
- No search algorithm can a priori be expected to perform any better than blind random picking, which is rather inefficient
- Hill climbing strategy should perform the same as hill descending strategy, (they still return the minimal solution seen during their trajectories).

NFL3: Encoding and Neighborhood Structures

- For a given search algorithm, and absent any prior information about which $f \in F$ we are working on, no encoding can be expected to result in better performance than any other.

NFL4: Stochastic Optimization

- For problems of the form:

$$\min_{x \in X} E_{\Omega}[\ell(x, \omega)]$$

  without prior knowledge about the function ℓ or the probability distribution Ω that reflects uncertainty, there is no universal strategy

- Conservation of robustness
  (robust = perform reasonably well on a set of functions at the cost of not extremely performing well on any set of functions.) if any algorithms is robust then every algorithm is robust and if some algorithm is not robust then no algorithm is robust.
- Prior Knowledge
- Performance/Sensitivity trade off

Original Proof

Original proof of NFL theorem is cast in Cast in probability theory. Motivations:

- easy generalization to stochastic algorithms
- simple and consistent framework that works for both deterministic and stochastic settings
- immediate advantage of the use of the probability distribution:

$$\Pr(f) = \Pr\left(f(x_0), f(x_1), \ldots, f(x_{|X|-1})\right)$$

Theorem (Wolpert and Macready, 1996)

For any pair of algorithms A_1 and A_2

$$\sum_{f \in F} \Pr(T_m^y \mid f, m, A_1) = \sum_{f \in F} \Pr(T_m^y \mid f, m, A_2)$$

Corollary For any performance measure M over T_m^y the average overall f of $\Pr(M(T_m^y \mid f, m, A)$ is independent of A

Criticism and Research

- NFL applies to large sets of functions and it is unclear if the NFL applies to small sets or to real world problems of practical interest.
- Research focuses on finding classes of problems over which the NFL does not hold. In particular, does the NFL hold over the instances of a particular combinatorial optimization problem?

Sharpened version of NFL: the NFL holds over classes of functions much smaller than the set of all functions.

Sharpened NFL

Def.: π is a permutation of X ($\pi \in \Pi(X)$), $\pi : X \mapsto Y$
Def.: $\pi f := f(\pi^{-1}(x))$
Def.: The set of functions F is closed under permutation if for all $f \in F$ and $\pi \in \Pi(X)$: $\pi f \in F$

Theorem (Schumacher et al. 2001)

The NFL theorem holds for a set of functions if and only if that set of functions is closed under permutation.

Criticism

The set of all possible discrete functions is incredibly large and most are incompressible (need a full enumeration look-up table for their representation)
Restriction to permutation closure still need exponential space description.

In practice, the function class F has restricted complexity.

Complexity:

- limited computation time for $f(x)$
- limited description space (Kolmogorov complexity)
- limited circuit size ...

Further developments: Almost No Free Lunch theorem

Learnings

- Should we conclude that all algorithms are equal?

  No: we can restrict the study to functions that we believe are representative of the problems we actually want to solve. But we

  must be careful with generalizations because algorithms that perform well on some benchmark problems may perform bad on others.

- An algorithm performance is determined by how "aligned" it is with the underlying probability distribution over the problems on which it is run.

- NFL is an argument in favor of algorithm specialization. Exploit problem specific knowledge