Outline

1. Implementation Contest
2. Other Metaheuristics
   - Evolutionary Algorithm Extensions
   - Model Based Metaheuristics
3. Particle swarm optimization (PSO)
4. Resume
   - Capacitated Vehicle Routing

Median and Best results

<table>
<thead>
<tr>
<th>Variable</th>
<th>Alg</th>
<th>Inst</th>
<th>Sol</th>
<th>Alg</th>
<th>Monotone</th>
<th>Inst</th>
<th>Sol</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>191076</td>
<td>G-1000-0.5-50-25.col</td>
<td>48</td>
<td>191076</td>
<td>G-1000-0.5-50-49.col</td>
<td>43</td>
<td></td>
</tr>
<tr>
<td></td>
<td>270383</td>
<td>G-1000-0.5-50-25.col</td>
<td>57</td>
<td>270383</td>
<td>G-1000-0.5-50-49.col</td>
<td>47</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS</td>
<td>G-1000-0.5-50-25.col</td>
<td>42</td>
<td>TS</td>
<td>G-1000-0.5-50-49.col</td>
<td>42</td>
<td></td>
</tr>
<tr>
<td></td>
<td>191076</td>
<td>G-1000-0.5-60-30.col</td>
<td>56</td>
<td>191076</td>
<td>G-1000-0.5-60-59.col</td>
<td>53</td>
<td></td>
</tr>
<tr>
<td></td>
<td>270383</td>
<td>G-1000-0.5-60-30.col</td>
<td>59</td>
<td>270383</td>
<td>G-1000-0.5-60-59.col</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TS</td>
<td>G-1000-0.5-60-30.col</td>
<td>49</td>
<td>TS</td>
<td>G-1000-0.5-60-59.col</td>
<td>49</td>
<td></td>
</tr>
</tbody>
</table>

Flat

<table>
<thead>
<tr>
<th>Alg</th>
<th>Inst</th>
<th>Sol</th>
</tr>
</thead>
<tbody>
<tr>
<td>191076</td>
<td>flat-1000-50-0-1.col</td>
<td>101</td>
</tr>
<tr>
<td>270383</td>
<td>flat-1000-50-0-1.col</td>
<td>96</td>
</tr>
<tr>
<td>TS</td>
<td>flat-1000-50-0-1.col</td>
<td>90</td>
</tr>
<tr>
<td>191076</td>
<td>flat-1000-60-0-1.col</td>
<td>101</td>
</tr>
<tr>
<td>270383</td>
<td>flat-1000-60-0-1.col</td>
<td>96</td>
</tr>
<tr>
<td>TS</td>
<td>flat-1000-60-0-1.col</td>
<td>90</td>
</tr>
</tbody>
</table>
Scatter Search and Path Relinking

Key idea: maintain a small population of reference solutions and combine them to create new solutions.

Differ from EC by providing unified principles for recombining solutions based on generalized path constructions in Euclidean or neighborhood spaces.

Scatter Search and Path Relinking:
- generate \( \mathbf{s_p} \) with a diversification generation method
- perform subsidiary perturbative search on \( \mathbf{s_p} \)
- update reference set \( \mathbf{r_s} \) from \( \mathbf{s_p} \)

while termination criterion is not satisfied: do
- generate subset \( \mathbf{s_b} \) from \( \mathbf{r_s} \)
- apply solution combination to \( \mathbf{s_b} \) to obtain \( \mathbf{s_c} \)
- perform subsidiary perturbative search on \( \mathbf{s_c} \)
- update reference set \( \mathbf{r_s} \) from \( \mathbf{r_s} \cup \mathbf{s_c} \)

Note:
- A large number of solutions is generated by the diversification generation method while about 1/10 of them are chosen for the reference set.
- In more complex implementations the size of the subset of solutions \( \mathbf{s_c} \) may be larger than two.

Scatter Search
Solutions are encoded as points of an Euclidean space and new solutions are created by building linear combinations of reference solutions using both positive and negative coefficients.

Path Relinking
Combinations are reinterpreted as paths between solutions in a neighborhood space. Starting from an initiating solution moves are performed that introduces components of a guiding solution.
**Model Based Metaheuristics**

**Key idea** Solutions generated using a parameterized probabilistic model updated using previously seen solutions.

1. Candidate solutions are constructed using some parameterized probabilistic model, that is, a parameterized probability distribution over the solution space.
2. The candidate solutions are used to modify the model in a way that is deemed to bias future sampling toward low cost solutions.

---

**Stochastic Gradient Method**

- \( \{P(s, \bar{\theta}) \mid \bar{\theta} \in \Theta \} \) family of probability functions defined on \( s \in S \)
- \( \Theta \subset \mathbb{R}^m \) m-dimensional parameter space
- \( P \) continuous and differentiable

Then the original problem may be replaced with the following continuous one:

\[
\arg \min_{\bar{\theta} \in \Theta} \mathbb{E}_{\bar{\theta}}[f(s)]
\]

**Gradient Method:**
- start from some initial guess \( \bar{\theta}_0 \)
- at stage \( t \), calculate the the gradient \( \nabla \mathbb{E}_{\bar{\theta}_t}[f(s)] \) and update \( \bar{\theta}_{t+1} \) to be \( \bar{\theta}_t + \alpha_t \nabla \mathbb{E}_{\bar{\theta}_t}[f(s)] \) where \( \alpha_t \) is a step-size parameter.

---

**Cross Entropy Method**

**Key idea:** use rare event-simulation and importance sampling to proceed toward good solutions

- Generate random solution samples according to a specified mechanism
- update the parameters of the random mechanism to produce better “sample”

**Probability Density Function**

- \( p(s, \bar{\theta}) \mid \bar{\theta} \in \Theta \) probability density function on \( s \in S \)
- \( \mathbb{E}_{\bar{\theta}}[f(s)] = \sum_{s \in S} f(s)p(s, \bar{\theta}) \)

If we are interested in the probability that \( f(s) \) is smaller than some threshold \( \gamma \) under the probability \( p(\cdot, \bar{\theta}^*) \) then:

\[
Pr(f(s) \geq \gamma, \theta^*) = \mathbb{E}_{\bar{\theta}^*}[I[f(s) \geq \gamma]]
\]

if this probability is very small then we call \( \{f(s) \geq \gamma\} \) a rare event

**Monte-Carlo simulation:**
- draw a random sample
- \( \frac{1}{N} \sum_{i=1}^{N} I[f(S_i) \geq \gamma] \)

**Importance sampling:**
- use a different probability function \( h \) on \( S \) to sample the solutions
- \( \frac{1}{N} \sum_{i=1}^{N} I[f(S_i) \geq \gamma] \frac{p(S_i, \theta^*)}{h(S_i)} \)
Cross Entropy Method (CEM):
Define $\hat{\theta}_0$. Set $t = 1$
While termination criterion is not satisfied:
| generate a sample $(s_1, s_2, \ldots, s_N)$ from the pdf $p(\cdot; \hat{\theta}_{t-1})$
| set $\hat{\gamma}_t$ equal to the $(1 - \rho)$-quantile with respect to $f$ ($\hat{\gamma}_t = S(\lceil(1 - \rho)N\rceil)$)
| use the same sample $(s_1, s_2, \ldots, s_N)$ to solve the stochastic program
\[
\hat{\theta}_t = \arg\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} I(f(S_i) \leq \hat{\gamma}_t) \ln p(S_i; \theta)
\]
Generates a two-phase iterative approach to construct a sequence of levels $\hat{\gamma}_1, \hat{\gamma}_2, \ldots, \hat{\gamma}_t$ and parameters $\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_t$ such that $\hat{\gamma}_t$ is close to optimal and $\hat{\theta}_t$ assigns maximal probability to sample high quality solutions

Example: TSP

- Solution representation: permutation representation
- Probabilistic model: matrix $P$ where $p_{ij}$ represents probability of vertex $j$ after vertex $i$
- Tour construction: specific for tours
  Define $P^{(1)} = P$ and $X_1 = 1$. Let $k = 1$
  While $k < n - 1$
  | obtain $P^{(k+1)}$ from $P^{(k)}$ by setting the $X_k$-th column of $P^{(k)}$ to zero and normalizing the rows to sum up to 1.
  | Generate $X_{k+1}$ from the distribution formed by the $X_k$-th row of $P^{(k)}$
  | set $k = k + 1$
- Update: take the fraction of times transition $i$ to $j$ occurred in those paths the cycles that have $f(s) \leq \gamma$
Estimation of Distribution Algorithms

**Key idea** avoid the problem of breaking good building blocks of EC by estimating a probability distribution over the search space which is then used to sample new solutions

- Candidate solutions constructed by a parametrized probabilistic model
- The candidate solutions are used to modify the model in order to bias toward high quality solutions

**Needed:**
- A probabilistic model
- An update rule for the model’s parameter and/or structure

**Estimation of Distribution Algorithm (EDA):**

1. **generate an initial population** \( sp \)
2. **While** termination criterion is not satisfied:
   - **select** \( sc \) from \( sp \)
   - estimate the probability distribution \( p_i(x_i) \) of solution component \( i \) from the highest quality solutions of \( sc \)
   - **generate a new** \( sp \) by sampling according to \( p_i(x_i) \)

**Probabilistic Models**

**No Interaction**
- weighted frequencies over the population (a mutation operator can be applied to the probability)
- classical selection procedures
- incremental learning with binary strings:
  \[ p_{t+1,i}(x_i) = (1 - \rho)p_{t,i}(x_i) + \rho x_i \] with \( x_i \in S_{\text{best}} \)

**Pairwise Interaction**
- chain distribution of neighboring variables (conditional probabilities constructed using sample frequencies)
- dependency tree
- forest

**Multivariate**
- independent clusters based on minimum description length
- Bayesian optimization: Bayesian networks learning

**Particle swarm optimization (PSO)**

- Inspired by social system, the collective behaviors of simple individuals interacting with their environment and each other.
- Is a population based stochastic optimization technique
- In PSO, each single solution is a "bird" in the search space. We call it "particle".
- All of particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.
Elements of PSO:
- Solution representation, e.g., binary string
- Initialization: group of random particles (solutions)
- Evaluation, Compare, Imitate using two simple sociometric principles:
  - the best solution it has achieved so far (pbest)
  - the best value, obtained so far by any particle among the neighbors (lbest)
  - the best value, obtained so far by any particle in the population (gbest)
After finding the two best values, the particle updates its velocity and positions with following equation (a) and (b).

(a) \[ v_{td}^{t+1} = v_{td}^t - c_1 \cdot r\cdot \cdot (p_{best,d} - x_{td}^t) + c_2 \cdot r\cdot \cdot (g_{best,d} - x_{td}^t) \]

(b) \[ x_{td}^{t+1} = x_{td}^t + v_{td}^{t+1} \]
In binary string representation (rather than real numbers) the velocity is used as a probability threshold to determine whether to flip or not the value of \( x_{td} \).

The pseudo code of the procedure is as follows

Initialize particles
While maximum iterations or minimum error criteria is not attained
  For each particle
    Calculate fitness value
    If the fitness value is better than the best fitness value (pBest) in history
      set current value as the new pBest
  End
  Choose the particle with the best fitness value of all the particles as the gBest
  For each particle
    Calculate particle velocity according equation (a)
    Update particle position according equation (b)
  End
Particles’ velocities on each dimension are bounded by \( v_{max} \).

Comparisons between Genetic Algorithm and PSO

Most of evolutionary techniques have the following procedure:
1. Random generation of an initial population
2. Reckoning of a fitness value for each subject. It will directly depend on the distance to the optimum.
3. Reproduction of the population based on fitness values.
4. If requirements are met, then stop. Otherwise go back to 2.
- PSO does not have genetic operators like crossover and mutation.
- Particles update themselves with the internal velocity. They also have memory, which is important to the algorithm.

PSO parameter control

Parameters in PSO:
- Dimension and range of particles is determined by the problem to be solved.
- Number of particles: the typical range is 20 - 40. But also 10 or 100 or 200 might be used.
- \( v_{max} \): usually set as the range of the particle \([-v_{max}, v_{max}]\)
- Learning factors: usually \( c_1 = c_2 \) and ranges from \([0, 4]\)
- The stop condition:
  - global version is faster but might converge to local optimum for some problems.
  - local version is a little bit slower but not easy to be trapped into local optimum.
  - Combined version: use global version to get quick result and use local version to refine the search.
Construction Heuristics

- Greedy heuristics
- Two steps heuristics
  - Choose variable
    - Most constrained first
    - Most constraining first (higher degree)
  - Choose value
- Look ahead features
- Add or drop approach
- Decomposition/partitioning

Moreover heuristics can be
- static, ie, order decided at the beginning
- dynamic, ie, order redecided after every decision.

Classification of Metaheuristics

- Trajectory methods vs discontinuous methods
- Population-based vs single-point search
- Memory usage vs memory-less methods
- One vs various neighborhood structures
- Dynamic vs static objective function
- Nature-inspired vs non-nature inspiration
- Instance based vs probabilistic modeling based

Local Search

Four typical solution representation and their neighborhood operators:
- Linear permutation (Scheduling)
- Circular permutation (Routing)
- Assignment (coloring)
- Subset (set covering)

Outline

1. Implementation Contest
2. Other Metaheuristics
   - Evolutionary Algorithm Extensions
   - Model Based Metaheuristics
3. Particle swarm optimization (PSO)
4. Resume
   - Capacited Vehicle Routing
Capacitated Vehicle Routing (CVRP)

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
▶ arcs/edges have associated a cost \( c_{ij} \) \( (c_{ik} + c_{kj} \geq c_{ij}, \forall i, j \in V) \)
▶ customers 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 \( C \).

Lower bound to \( K \): \( K \geq K_{\text{min}} \) where \( K_{\text{min}} \) is the number of bins in the associated Bin Packing Problem

CVRP

Construction Heuristics
▶ Nearest neighbors
▶ Savings heuristics (Clarke and Wright)
▶ Insertion heuristics
▶ Route-first cluster-second
▶ Cluster-first route-second
▶ Sweep algorithm
▶ Generalized assignment
▶ Location based heuristic
▶ Petal algorithm

Perturbative Search
▶ Solution representation: sets of integer sequences, one per route
▶ Neighborhoods structures:
  ▶ intra-route: 2-opt, 3-opt
  ▶ inter-routes: \( \lambda \)-interchange, relocate, exchange, CROSS, ejection chains, GENI

Construction Heuristics specific for TSP
▶ Heuristics that Grow Fragments
  ▶ Nearest neighborhood heuristics
  ▶ Double-Ended Nearest Neighbor heuristic
  ▶ Multiple Fragment heuristic (aka, greedy heuristic)
▶ Heuristics that Grow Tours
  ▶ Nearest Addition
  ▶ Farthest Addition
  ▶ Random Addition
  ▶ Clarke-Wright savings heuristic
▶ Heuristics based on Trees
  ▶ Minimum span tree heuristic
  ▶ Christofides' heuristics
  ▶ Fast recursive partitioning heuristic