Traveling Salesman Problem

From a set of \( n \) locations:

Find the best tour that visits all locations (and returns to the origin).

\[ \text{Figure 1: Locations} \]

\[ \text{Figure 2: TSP tour} \]

\( k \)-opt Heuristics

\( k \) edge swaps (\( k \)-opt moves) resulting in a shorter tour. Complexity: \( O(n^k) \). Simpler: Consider 2-opt and 3-opt moves.

Edge Selection: Usually Greedy e.g. First Improvement (FI), Best Improvement (BI).

Learn better 2-opt strategies?

Learn 2-opt strategies without supervision

Can be expanded to accommodate \( k \)-opt.

Trained on smaller instances and transferred to larger instances.

Markov Decision Process

States: A tuple \( S = (S_t, S') \), where \( S \) and \( S' \) are the current and lowest-cost solution.

Actions: \( (\alpha_1, \alpha_2) \) composed of two node indices.

Transitions: 2-opt operation to solution \( S \).

Rewards: Improvement upon the current best-found solution, i.e. \( R_t = L(S_t') - L(S_{t+1}) \).

Policy and Value Approximation

Encoder: Graph topology, node ordering and symmetry.

Policy decoder: probability of a \( k \)-opt move as

\[ \pi_{\theta}(A|S) = \frac{1}{Z} \exp \left( \phi A + \theta S \right) \]

Policy Gradient updates of the form, \( b \)-th instance:

\[ \nabla_b J(\theta) \approx T^{b-1} \nabla_b \log \pi_{\theta}(S^b_t|A^b_t, G^b_t) \]

Value decoder: trained on Monte Carlo returns

\[ L(\phi) \approx T^{\phi-1} (G^\phi_t - \bar{V}(S^\phi_t))^2 \]

Results

TSP instances of 3 sizes, 20, 50 and 100 nodes. Nodes drawn from a uniform in \([0,1]^2\). Performance:

<table>
<thead>
<tr>
<th>Method (nr. of samples)</th>
<th>Solver</th>
<th>TSP20</th>
<th>TSP50</th>
<th>TSP100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ours {500}</td>
<td>RL</td>
<td>3.84</td>
<td>5.71</td>
<td>7.83</td>
</tr>
<tr>
<td></td>
<td>3.84</td>
<td>5.71</td>
<td>7.83</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.84</td>
<td>5.71</td>
<td>7.83</td>
<td></td>
</tr>
<tr>
<td>Ours {1000}</td>
<td>RL</td>
<td>3.84</td>
<td>5.71</td>
<td>7.83</td>
</tr>
<tr>
<td></td>
<td>3.84</td>
<td>5.71</td>
<td>7.83</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.84</td>
<td>5.71</td>
<td>7.83</td>
<td></td>
</tr>
<tr>
<td>Ours {2000}</td>
<td>RL</td>
<td>3.84</td>
<td>5.71</td>
<td>7.83</td>
</tr>
<tr>
<td></td>
<td>3.84</td>
<td>5.71</td>
<td>7.83</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3.84</td>
<td>5.71</td>
<td>7.83</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Performance of TSP methods w.r.t. Concorde (optimal), SL Supervised Learning, RL Reinforcement Learning, S: Heuristic Solver

Results (cont.)

Figure 3: Tour cost comparison of learned and greedy policies.

Conclusion

Learning 2-opt policies can yield better policies than greedy ones.

Adapting such policies to more general TSP may be necessary for instances coming from different distributions.

Learning methods are slowly approaching the quality of handcrafted heuristics, but still cannot compete in scalability and depend on high training times.

References