Day 20 Notes

Zach Neveu

June 10, 2019

1 Agenda

- Recap of local search
- Variable-depth search
- Simulated Annealing
- Quiz tomorrow
- Project 4 recap

2 Local Search Recap

- Initial solution, made better by looking at neighbors
- TABU search: pick best neighbor even if it is worse
- Stop at time limit, when best answer stops getting better
- Items in TABU list can just be changes that were made, not complete solutions
- TABU list doesn't have to be very long to work
- Aspiration Level conditions: break tabu list rules for really good solutions
- Intensification/Diversification: alternate between searching for best in local area vs. moving to another area.

3 Variable Depth Search

- Consider uniform graph partitioning problem again.
- Break a graph into groups with equal number of nodes
- Minimize weight of inter-group edges
- Size of neighborhood determines how similar neighbors are.
- Larger neighborhood = fewer steps to optimum, slower steps
- Smaller neighborhood = smaller steps to optimum, faster steps
- Tradeoff between neighborhood sizes.

- Consider a potential swap of (a,b)
- Perform the swap, but mark it as "tentative".
- Swap **gain** $g(a,b)$: the decrease in cost function of making swap (a,b)
- Put both a and b on tabu list: neither can be moved
- Choose (a,b) such that $g(a,b)$ is maximized (steepest descent)
- Repeat until no swaps possible (all on tabu list)
- Let cumulative gain $G(k) = \sum_{i=1}^{k} g(a_i, b_i)$ be the gain after k swaps.
- After $\frac{n}{2}$ swaps, partition is back where it started with all items flip-flopped
- $G(k)$ has maximum at k^* which is somewhere between $\{0, \frac{n}{2}\}$
- Take the first k^* steps to get the next neighbor
- This neighborhood function allows the solver to take larger steps without requiring significantly longer steps
- This is basis for Lin-Kernighan TSP

4 Lin-Kernighan

- 2-opt neighborhoods require $O(n^2)$ - too lengthy
- Consider smaller neighborhood
- Consider HCs as paths. Select an edge to delete. Reconnect end of path to first side of deleted edge. Essentially, take a suffix of the path and flip it. Only single way to reconnect, n ways to delete an edge, $O(n)$ neighborhood.
- 2 tabu lists.
  - First list: edges that are added. Once an edge is added, it cannot be deleted.
  - Second list: deleted edges. Once an edge is deleted, it cannot be added.
- Use variable-depth search as optimization technique. Using tabu lists, make up to $\frac{n}{2}$ moves. Look find best out of these solutions and use it as the next solution.
- Turns out this is really effective! Long standing TSP champion algorithm. 1-2% from bound for Euclidean instances.
- Key concepts used: restrictive tabu list, steepest descent, variable depth search, small neighborhood

```
# Local search standard form (before tabu)
def local_search():
    x = initial_soln()
    done = False
    while not done:
```

```
        xp = minimum(c(Neighbors(x))
        if c(xp) < c(x):
            x = xp
        else
            done = True

# local search tabu search form
x = initial_soln()
xbest = x
k = 0
while xxx:
    xp = generate(N(x))

    # true if xp better than x
    # also true if xp worse than x up to a point
    if c(xp) - c(x) < t(k)
        x = xp
        if c(x) < c(xp)
            xbest = x
    k += 1
```

- If $t(k) = 0$, then this gets stuck in any local optimum
- If $t(k) = \infty$, then first neighbor always visited
- $t(k)$ controls tradeoff between finding optimum and exploring.
- Start with large value of $t(k)$, anneal to reach small $t(k)$ by the end.
- $t(k) \geq t(k+1)$, $lim_{k \to \infty} t(k) = 0$. Threshold Accepting.

5 Simulated Annealing

- Instead of $t(k)$ having fixed value, we let $t(k)$ be a random variable > 0
- Consider

$$p(N_i) = \begin{cases} 1 & c(xp) \leq c(x) \\ e^{\frac{c(x)-c(xp)}{d_k}} & c(xp) > c(x) \end{cases}$$

- Better solutions always excepted
- Worse potentially excepted. Worse solutions have smaller probability.
- d_k adjusts curve of falloff. Larger d_k means higher chance of going to worse neighbors. Smaller d_k means smaller chance of visiting worse neighbors.
- d_k varies with k. Can start large to diversify, then get smaller to intensify
- **cooling schedule:** how fast does d_k get smaller?

- Name: from quantum physics, simulating particle motion. d_k is temperature. Cooling schedule has very physical meaning here!

Simulated Annealing TSP

- SA1 algorithm
- Neighborhood function: 2-opt
- Set $d_0 \approx \infty$ - accept like 95% of answers
- set $d_k = d_{k-1}^{0.95}$
- Temperature length: $N(N-1)$ - how long to spend at each temperature
- Results (see 1): SA1 slow! Can be improved greatly using 2-opt afterwords.
- How to speed up?
- Neighborhood pruning. Smaller neighborhood → faster annealing → faster result.
  - Neighborhood: select a random edge to delete, select one of its 20 closest neighbors to delete. Then reconnect. This takes us from $0(n^2) \rightarrow O(n)$
  - Temp length = $\alpha * 20n$
- Low-temperature start, and don't use random initial solution
  - Start with good heuristic solution
  - Lower initial temperature
  - Speeds up search

Table 1: SA1 Results: Size vs. Algorithm

N=	100	316	1000
SA1	5.2(12.4s)	4.1(188s)	4.1(3170s)
SA1+2-opt	3.4	3.7	4.0
2-opt	4.5(0.03s)	4.8 (0.09s)	4.9 (0.34s)
3-opt	2.5(0.04s)	2.5 (0.1s)	3.1 (0.4s)
LK	1.5(0.06)	1.7(0.2)	2.0(0.77s)