Day 21 Notes

Zach Neveu

June 11, 2019

1 Agenda

- Quiz
- Simulated Annealing
- Genetic Algorithms
- Reading
- Next Quiz Monday
- Homework Thursday

2 Simulated Annealing

- Review of details: probability of accepting solution. 1 if better, less than 1 if worse.
- Results from last class: very slow, not awesome results
- How to speed up?
- Long time needed at each temperature
- Smaller neighborhood lowers time needed at each temperature
- Neighborhood pruning: lowers size of 2-opt neighborhood $O(n^2) \rightarrow O(n)$
- Low-temp start with good initial solution - reduces number of steps needed in annealing 50% of neighbors accepted initially instead of 95%.
- Low-temp start + neighborhood pruning → SA2
- SA2 results in table 1
- Still quite slow, but can beat LK given enough time
- No bound needed, no gradient needed, very flexible algorithm
- Further Variations of SA
  - Adaptive scheduling technique: have temp lengths adapt to solution. For example, when champion solution isn't changing often, reduce the temperature.
  - Water level algorithms - instead of looking at relative goodness of neighboring solution compared to current solution, compare neighbor to a global threshold or "water level". Raise the water level over time to anneal. Idea: imagine tides here! The search is like a

sandpiper, explores when tide goes out, then runs back to best solution when tide comes in.

Table 1: Variants of SA and performance

N=	100	316	1000
SA1	5.2(12.4s)	4.1(188s)	4.1(3170s)
SA1+2-opt	3.4	3.7	4.0
SA2 $\alpha = 10$	1.6 (14.3s)	1.8 (50.35s)	2.0 (2290s)
SA2 $\alpha = 40$	1.3 (58s)	1.5 (204)	1.7 (80509s)
SA2 $\alpha = 100$	1.1 (141s)	1.3 (655s)	1.6 (191000s)
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)

3 Genetic Algorithms

- Famous example of how nature has inspired algorithms
- Includes local search, intensification/diversification, randomization, but does so in a unique way.
- Idea: population of organisms. Pairs of organisms reproduce, individual organisms mutate, offspring share characteristics with parents, and rates of survival/reproduction tied to "fitness".
- Translation to optimization
  - Collection of solutions. This is different from other solvers!
  - Pairs of solutions produce new solutions
  - Individual solutions change
  - New solutions are similar to their parents
  - Survival and reproduction related to objective value

```
# Genetic Agorithm
pop = generate_pop()
done = False
while not done:
    subs = select(subset(pop, size=[1,2]))
    children = [mutate(x) for x in subs if x.size ==  1]
    children += [crossover(x[0],x[1]) for x in subs if x.size ==  2]
    survivors = find_survivors(children)
return maximum([fitness(x) for x in survivors])
```

- Required features
  - Population size

  - How to select initial population
  - Mating strategy
  - Mutation strategy
  - Crossover strategy
  - Survival function
- Initial population
  - Heuristics for best solutions
  - Random
  - Should have some diversity
- Mating Strategy
  - Select randomly, in proportion to fitness
- Mutation
  - Solutions represented as string of digits, often bits.
  - Mutation, is pick a random bit and flip it
  - Can end up with unviable children not in F
- Crossover
  - Pick a crossover index in strings, choose first half of one, second half of the other
  - idea: why not flip a coin on each bit?
  - idea: what about probabilistic kind of truth table for each bit? i.e. if parents are both 1, child is 1 with p=0.99.
- Selection
  - Keep with probability proportional to fitness
  - Only keep offspring