Nurse Scheduling

Fanying Chen, Shuyao Lu
Tianhao Lu, Sijia Zhu

Outline

- Nurse Scheduling Problem (NSP) review
- SA Algorithm
- GA Algorithm
- Comparing SA & GA
- Our experiment with simulated data
- Conclusion
- Further Questions to Consider
- Q&A

NSP - Background

- ❖ Goal: create a timetable that assign nurses to each shift, balancing the workload and preferences.
- ❖ Scheduling Period
- ❖ Professional Level of nurses
- ❖ Shift-Systems (Two-, Three-, Four- Shift)
  - ➢ Morning, Afternoon, Night, Long Shift
  - ➢ Off Days

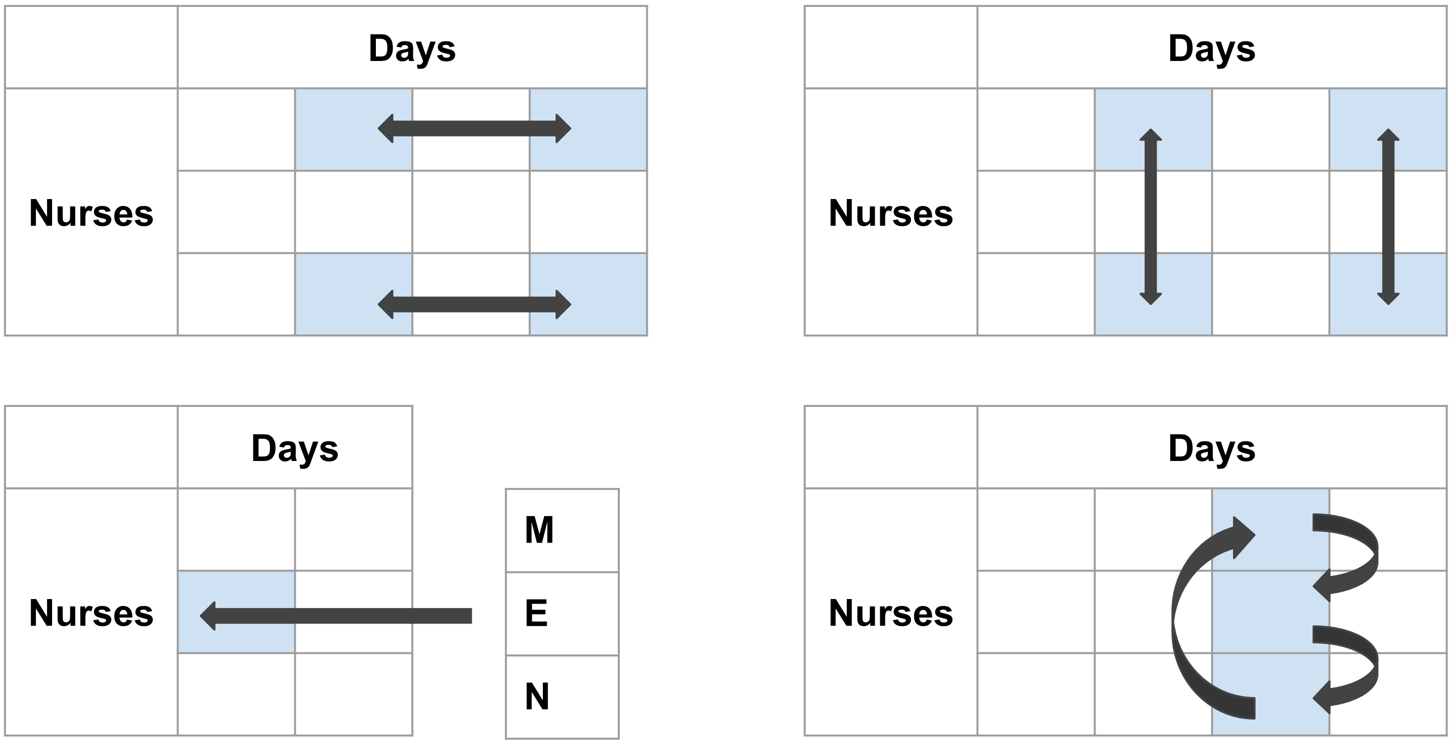
NSP - Constraints

- Hard Constraint ~> **Feasibility**
  - **Type 1 - Nurse Coverage:** for each shift, number of nurses assigned within a range, number of Top Level Nurses > 2, etc.
  - **Type 2 - Prohibited Working Patterns:** Morning shift after a Night Shift, 3 consecutive Night Shift, etc
- Soft Constraint ~> **Optimality**
  - **Type 3 - Satisfaction/ Preferences:** total number of Off-days = 4, total number of night-shifts = 3, etc.
- Objective: **Minimize** **C = w1*Cost1 + w2*Cost2 + w3*Cost3**

Simulated Annealing

- A total of N variables, N = number of nurses * number of days
- Assign Initial values randomly, calculate C, set as optimum
- Find a neighbor, compare objective value C.
  - If $dC < 0$, Accept the neighbor as new optimum
  - Otherwise,
    - Accept if $U < e^{(-dC/t)}$
    - Temperature t reduces in two different rates (depend on Changes/Trials)

Simulated Annealing - Neighborhood Structures



Genetic Algorithm

- Similar to evolutionary biology

Parent x Parent

↓ By random genetic mutation

Offspring

↓

The most fit survives

Constructed Operator

NSP solution x NSP solution

↓ A random change in the solution

New set of solutions

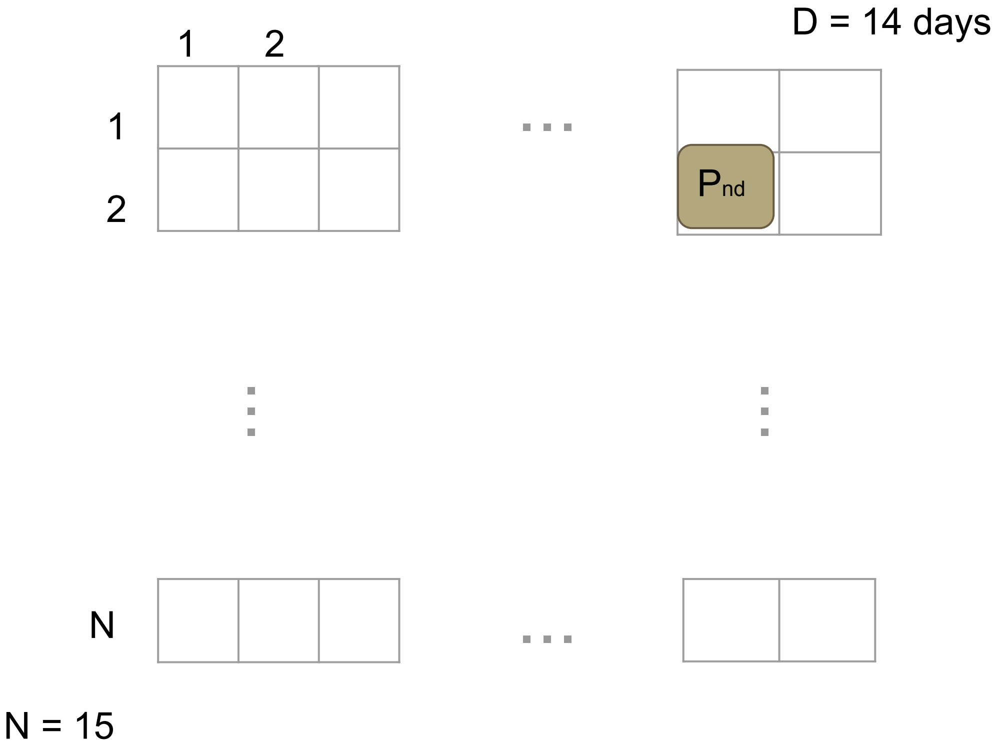
↓ Some fitness evaluator

Best result from the new Generation

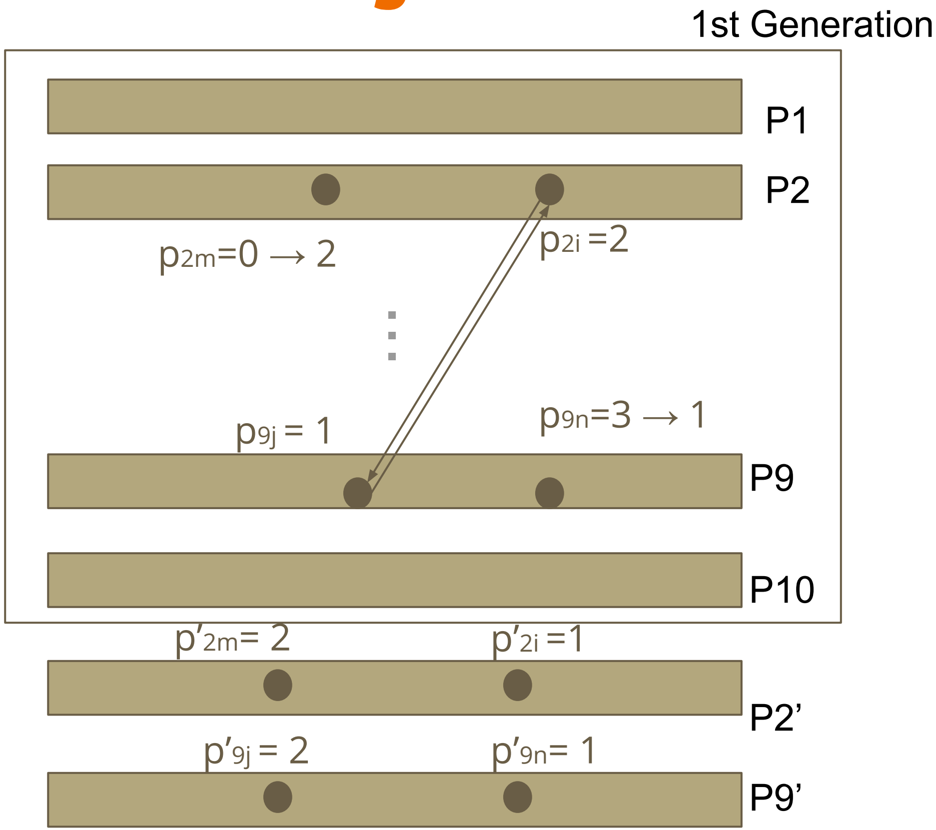
Genetic Algorithm

Variables:

$$P_{nd} = \begin{cases} 0, \text{ have the day off} \\ 1, \text{ 6AM - 12PM} \\ 2, \text{ 12PM - 6PM} \\ 3, \text{ 6PM - 6AM} \end{cases}$$



Genetic Algorithm



- Can view the N x D table as a 1 x (N + D) rectangle, which is a *chromosome*
- Each set of generation contains 10 solutions
- Randomly pick two solutions as "parent"
- Step 1: Pick an entry p_{2i} in P2 and p_{9j} in P9, switch the value of p_{2i} and p_{9j}
- Step 2: Randomly pick p_{2m}, and change its value; randomly pick p_{9n}, and change its value. Save as P2' and P9'
- For each generation, we keep the minimum as our result

Literature Review

- ❖ Most studies focus on either GA or SA; difficult to draw a fair comparison
- ❖ *S. Kundu, M. Mahato, B. Mahanty and S. Acharyya (2008)* made a comparison with simplified constraints
- ❖ Authors' conclusion: SA is more efficient than GA as it solves more problems on average, regardless of the random initialization;

Authors: SA discourages the bad move in short term while GA accepts all

We want to validate this result

Period (days)	Probs	Method	Solved	Cost	Time (sec)
7	100	SA	88	0.27	0.77
		GA	80	26.85	2.5
14	100	SA	92	0.11	2.85
		GA	73	4.72	7.21

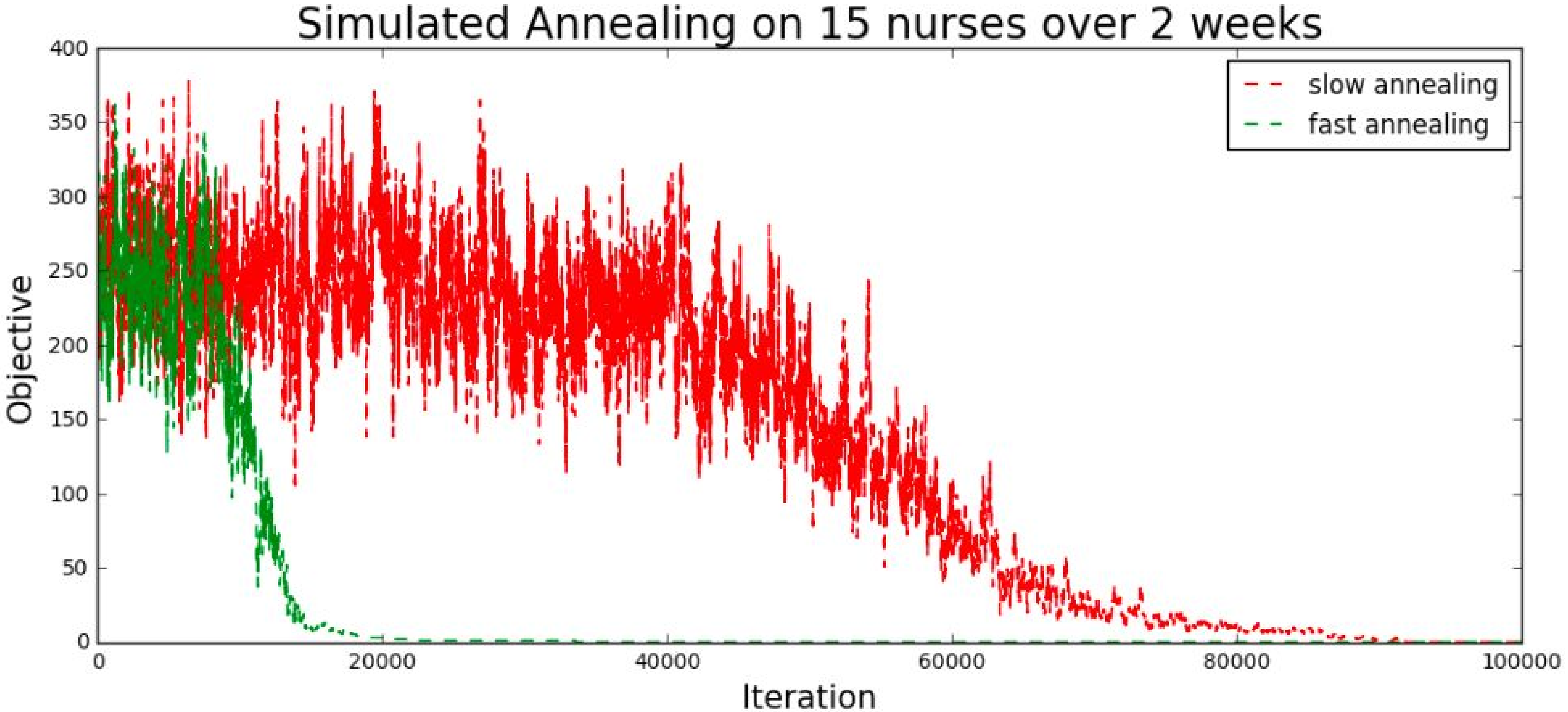
Experiment for Comparison

- 15 Nurses;
- 14 Days;
- Constraints:
  - Each day needs 4-6 nurses for morning/afternoon/night
  - No nurse works in morning shift right after a night shift
  - No nurse works 3 nights in a row

Experiment - Simulated Annealing

- Tried 2 settings:
  - **Fast Annealing** (temperature *=0.95 for every 1000 iterations)
  - **Slow Annealing** (temperature *=0.99 for every 1000 iterations)
- Both converge to optimum in 100,000 iterations

Both yield optimal solution - No Constraint Violated



Comparison

Random initialization

0	0	3	3	1	0	3	1	2	1	3	0	0	1
1	1	0	2	3	2	3	1	0	3	1	2	3	0
0	0	0	0	0	2	3	1	1	2	0	3	2	3
1	3	2	3	1	1	1	3	0	0	1	3	3	0
3	1	0	3	0	0	2	1	1	3	0	0	3	0
0	0	3	2	3	2	1	0	0	0	3	0	3	1
3	3	1	1	2	3	2	0	1	1	0	3	3	3
0	3	2	3	2	1	1	1	1	0	2	0	1	2
1	1	3	2	1	2	3	1	0	3	1	0	0	2
2	1	0	2	0	0	3	2	2	3	2	1	2	0
2	3	1	3	3	2	2	0	2	1	3	3	1	3
0	3	0	3	0	0	2	1	2	2	0	0	1	1
1	1	3	0	1	0	2	0	2	0	3	3	3	3
0	3	1	1	3	2	0	3	3	3	1	2	1	0

violation

Optimum schedule

3	2	2	2	0	3	3	0	0	2	3	3	0	2
3	2	3	3	1	3	0	2	2	3	0	2	3	3
0	0	3	3	2	0	2	3	3	2	3	2	2	3
1	1	0	2	3	1	2	2	1	2	3	0	0	0
1	1	2	1	0	3	0	1	1	1	0	0	2	0
2	2	1	1	0	2	0	0	1	0	0	2	2	2
3	0	1	2	3	2	0	0	2	1	1	1	2	0
1	0	0	1	0	3	0	3	3	2	2	0	1	1
0	1	2	2	1	3	3	2	3	0	1	2	3	1
0	0	2	1	2	0	0	1	2	0	3	2	3	1
0	0	1	0	3	2	1	2	3	3	1	1	1	0
1	2	3	1	0	1	3	2	0	2	2	1	0	1
0	3	3	2	2	2	2	1	2	3	0	1	1	1
2	3	0	1	2	0	2	3	2	3	2	1	3	3
2	3	2	0	2	1	2	0	3	3	1	1	1	1

Experiment - Genetic Algorithm

- Each population has 10 schedules;
- Choose 2 parents each time
- Better parent (with lower objective value) has higher chance to be chosen
- We used

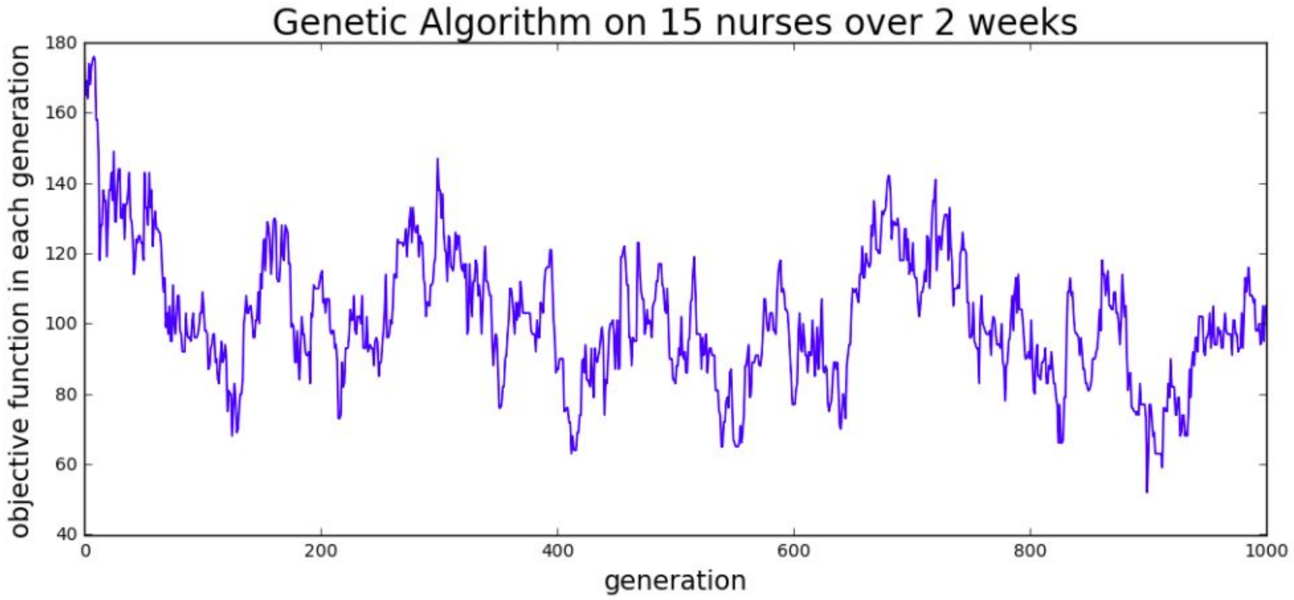
$$Pr(schedule\ i) \propto e^{-\sqrt{(score_i)}}$$

$$Pr(schedule\ i) \propto e^{-score_i}$$

$$Pr(schedule\ i) \propto (C - score_i)$$

Experiment - Genetic Algorithm

All the weight functions **failed to find global minimum** (which exists per Simulated Annealing)

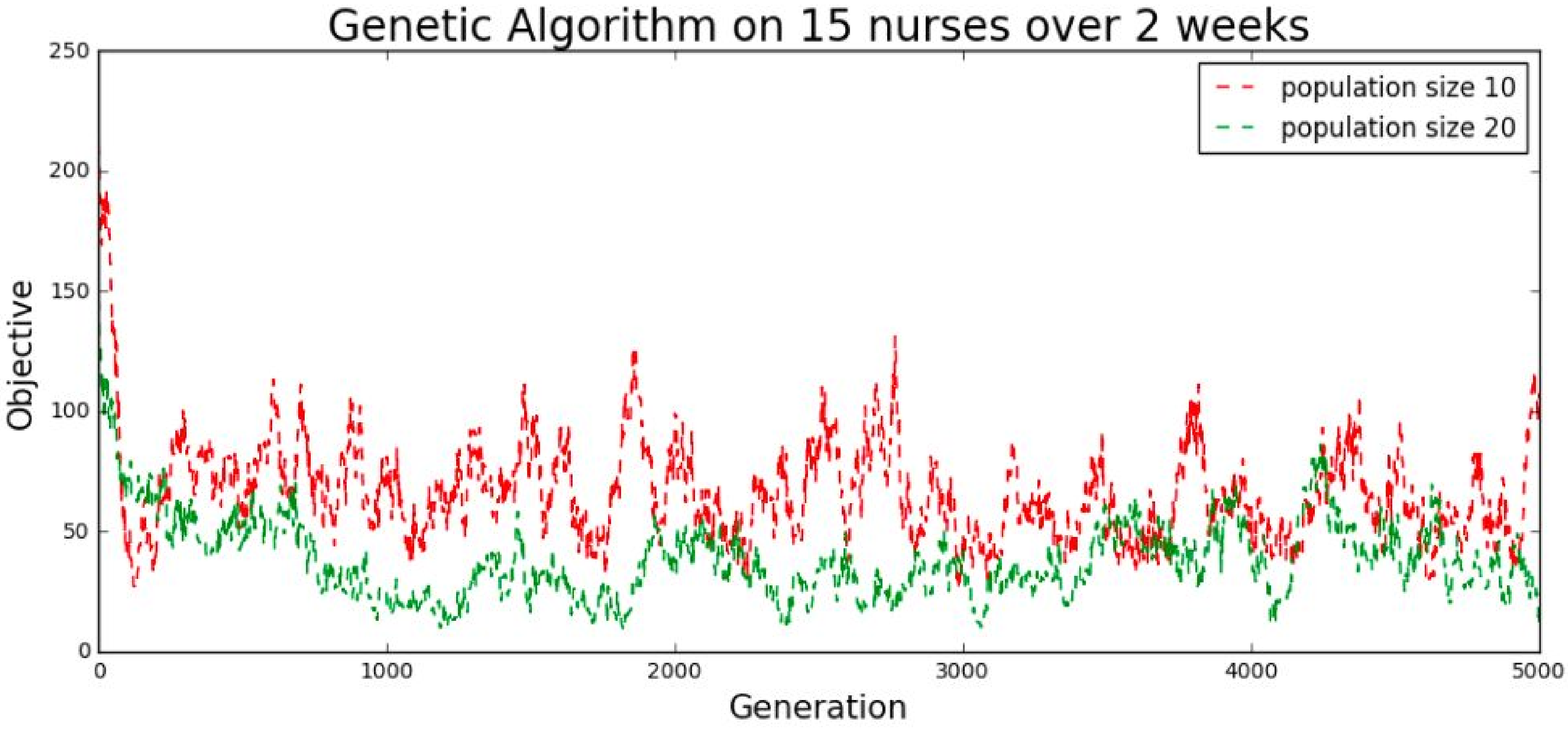


Experiment - Genetic Algorithm

- Also tried:
  - Hybrid approach
    - use 1 weight function for first 5000 generation
    - another one for the next 5000 generation
  - Only Cross-over without mutation
  - Neither significantly enhances the performance

Experiment - Genetic Algorithm

- When increasing the population size to 20, it enhances the performance:
  - The best schedule has objective value=9 (no hard constraint violated)
  - Still sub-optimal (compared to the 0-violation result from Simulated Annealing)


Genetic Algorithm on 15 nurses over 2 weeks
population size 10
population size 20
Objective
Generation
0
50
100
150
200
250
0
1000
2000
3000
4000
5000

Results and Conclusion

- NSP is **NP-hard** in general;
- We used **2 meta-heuristics** on the simulated data;
- **Simulated Annealing yields a more efficient result** than Genetic Algorithm, consistent with the authors' discovery
- One possible explanation: Simulated Annealing **discourages bad neighbors** while Genetic Algorithms **accept all results** to be in next generation.

Further Problems

- We **assumed the nurses are exchangeable**, while in real life nurses are different (skill sets, holiday preferences, etc);
- In Simulated Annealing, we search in the **entire neighborhood** within 1-shift difference; sometimes researchers suggest only looking into the **valid neighborhoods** (where no hard constraint is violated);
- The **weight function** in Genetic Algorithm could be more explicitly defined.

Questions?

References

[1]Y. Ko, D. Kim, M. Jeong, W. Jeon, S. Uhmn and J. Kim, *An Efficient Method for Nurse Scheduling Problem using Simulated Annealing*, 1st ed. Chuncheon, Gangwondo.

[2]H. Kawanaka, K. Yamamoto, T. Yoshikawa, T. Shinogi and S. Tsuruoka, *Genetic Algorithm with the Constraints for Nurse Scheduling Problem*, 1st ed. Mie, Japan, 2001.

[3]R. Namoco and R. Salazar, *Solving the Nurse Scheduling Problem of Private Hospitals in the Philippines using Various Operators for Genetic Algorithm*, 1st ed. Cagayan de Oro City, Philippines: Indian Journal of Science and Technology, 2016.

[4]U. Aickelin and K. Dowsland, "An indirect Genetic Algorithm for a nurse-scheduling problem", *Computers & Operations Research*, vol. 31, no. 5, pp. 761-778, 2004.

[5]S. Kundu, M. Mahato, B. Mahanty and S. Acharyya, *Comparative Performance of Simulated Annealing and Genetic Algorithm in Solving Nurse Scheduling Problem*, 1st ed. Hong Kong, China: Proceedings of the International MultiConference of Engineers and Computer Scientists, 2008.