Introduction to Meta-Heuristics
Team Work I
Maximum clique problem

m5221148 Yohei Shimmyo
m5221105 Hiroshi Naito
m5221130 Ryota Fukuzawa
m5211158 Yuji Murakami

Roles

Yohei Shimmyo (Leader)
- Implementation of simulated annealing
- Testcase generating, experiments

Hiroshi Naito
- Implementation of Iterative Local Search
- Make slides

Ryota Fukuzawa
- Implementaton of TabuSearch
- Make slides

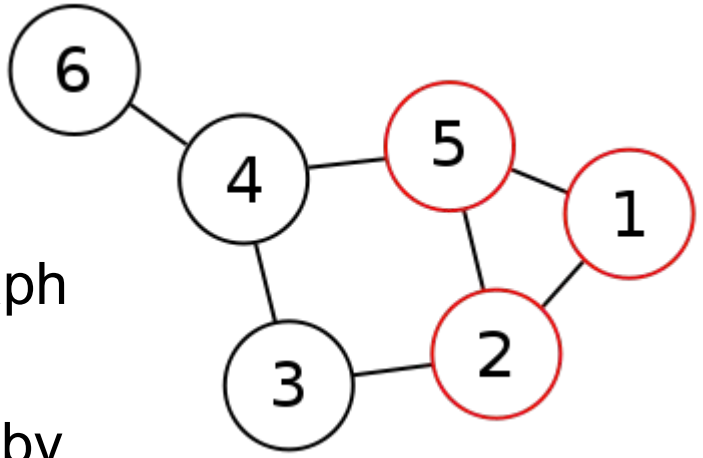
Yuji Murakami
- Implementation of Guided Local Search
- Make slides

Implementation of GLS and TS could not make until deadline.

Problem

Maximum clique problem

- Given an undirected graph G=(V, E)
- Find maximum clique V’
  - Clique : a subgraph of G whose vertices compose a complete graph
    - Complete graph: any pair of distinct vertices is connected by a unique edge



https://en.wikipedia.org/wiki/Clique_problem

Result: Objective Function Value 10 times

testcase (# vertices)	SA avg.	SA min.	SA max.	ILS avg.	ILS min.	ISL max.
hand_00(6)	3	3	3	3	3	3
hand_01(6)	6	6	6	6	6	6
hand_02(8)	5	5	5	5	5	5
hand_03(6)	4	4	4	4	4	4
rnd_00(122)	3.7	3.0	5.0	6	6	6
rnd_01(201)	4.4	3.0	6.0	8	8	8
rnd_02(27)	22	22	22	22	22	22
rnd_03(202)	4	3.0	7.0	7	7	7
rnd_04(267)	3.5	2.0	4.0	8	8	8

Result: Execution Time [seconds] 10 times

testcase (# vertices)	SA avg.	SA min.	SA max.	ILS avg.	ILS min.	ISL max.
hand_00 (6)	0.4653	0.4615	0.4858	8.028e-04	3.370e-04	0.0022
hand_01 (6)	0.5419	0.5319	0.5541	0.0013	9.200e-04	0.0015
hand_02 (8)	0.5005	0.4838	0.5764	7.181e-04	4.830e-04	0.0012
hand_03(6)	0.4744	0.4681	0.4902	4.383e-04	3.480e-04	5.750e-04
rnd_00(122)	2.7093	2.6148	2.8838	0.1663	0.1499	0.2420
rnd_01(201)	6.9969	6.7584	7.3392	2.4336	2.2889	2.8854
rnd_02(27)	1.1271	1.0877	1.1600	0.6595	0.6457	0.6720
rnd_03(202)	6.7217	6.6027	6.8633	1.3850	1.2626	1.5224

Result: Memory Usage

- SA
  - $O(V^2)$
- ILS
  - $O(V+E)$

It depends on how the graph is represented.

Properties of SA

- T = 50:-0.00001:1e-2
- State
  - Set of vertices
- Neighborhoods:
  - Add a vertex to a previous state at random
  - Remove a vertex from a previous state at random
- Graph Representation:
  - Adjacency matrix

Discussion of SA

- SA is not appropriate for the Maximum Clique Problem compared to ILS
  - The objective function value might change dramatically in neighborhoods
    - Suppose a subgraph G' is a clique
    - Neighborhoods 1: G' and a vertex
    - Neighborhoods 2: G' minus a vertex
    - if a vertex is added/removed from a complete graph, the subgraph might become a non-complete graph
      →Objective function value decreases dramatically
  - SA might employ this result with a probability

Properties of ILS

- Initial state : Random
- Local search : Hill Climb
- Terminate Condition : search history : The search history which contains the best results is empty. If a search history can get a better result, the value of the search history is updated.
- Objective Function : The number of node of clique
- State : Set of vertices

Discussion of ILS

- ILS is proper to Maximum clique problem.
- The search method is Hill climbing.
- If ILS got a local optimum, ILS can escape the local optimum by changing an initial value.

  → In Maximum clique problem, it is important to find a proper initial value, so we consider that ILS is proper to find initial values.

Discussion of Maximum clique problem

Selection of the initial state is important, because the evaluation function is similar to discrete

- In SA, its not suitable to escape from the local solution.
- In ILS, its suitable because it conducts iterative search from different initial state.
- In TS, its necessary to try all neighborhood once, so TS is not effective.
- In GLS, the costs for actions and attributes are all equal.

additional slides

Comments from Prof.zhao

- Comments
  - Not trustable solution because inside algorithms are different.

    ➔ We reviewed our programs (not to use our implementation but use MATLAb libraries)

  - revised points
    - ➔ We changed current state representation from list such as [2 4] into binary representation such [0 1 0 1].
    - ➔ SA function was used from MATLAB library.
    - ➔ We used a common evaluation function.
    - ➔ We resolved Maximum Clique Problem by SA and TS.

Simulated Annealing

MATLAB Library: simulannealbnd

- apply our objective and anneal (neighborhood) function
  - objective function is negated because simulannealbnd is for minimization problem and maximum clique problem is maximum problem
- Objective function inputs subgraph of a given graph and returns the number of vertices of clique. If the subgraph is not clique, return -1.
- Neighborhood
  - Add a vertex to the subgraph
  - Remove a vertex from the subgraph

Tabu Search

We refered to ypea116-tabu-search to make TS program.

- TS used same Neighborhood and Evaluation functions SA used.
  - TS searches scores of all neighbourhood and selects the best one in the neighbourhood.
  - Then, TS operates tabu list.
    - Basically, a current state does not move to the states in tabu list.
  - As an exception, current states move to the state evaluated the best neighbourhood in iteration.
    - Tabu list was saved in queue.

Environment

Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz

MATLAB_R2018b

Linux 4.9.0 (debian)

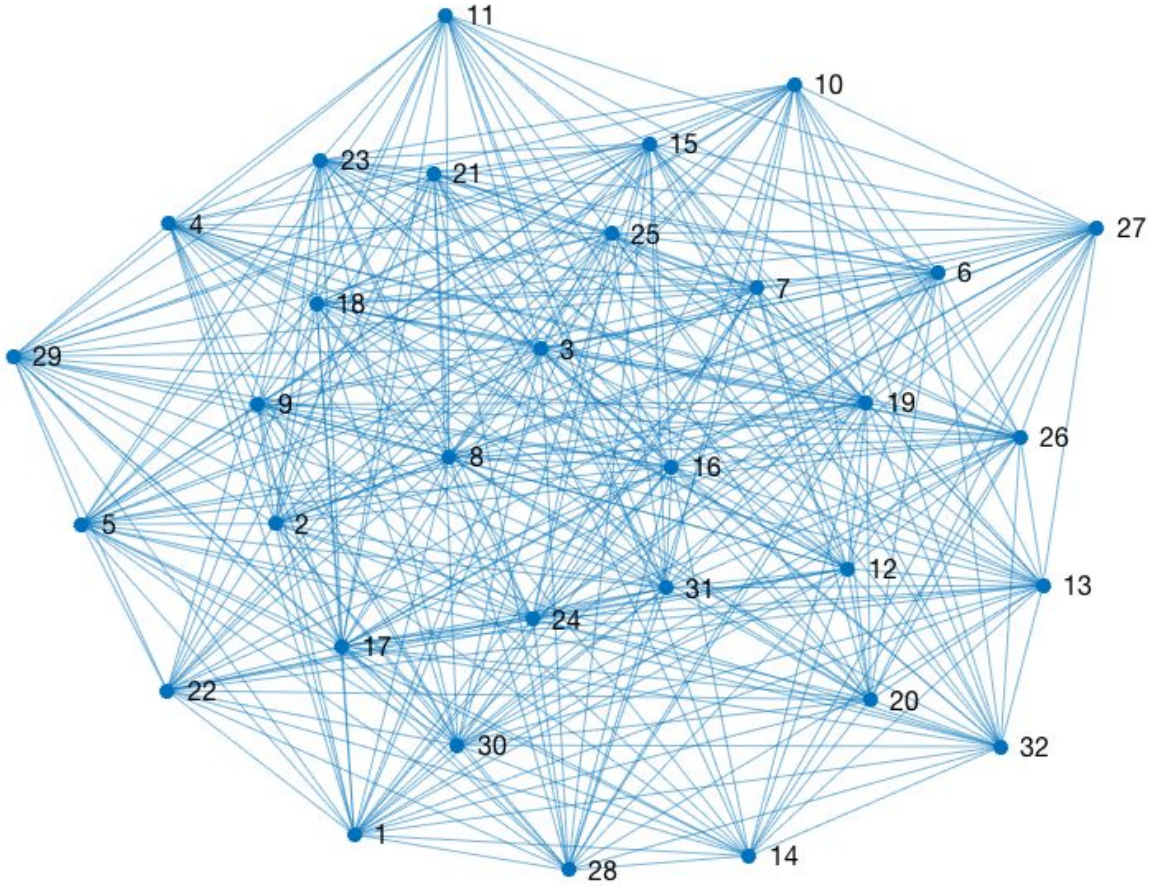
Result: Objective Function Value 10 times

testcase (# vertices)	SA avg.	SA min.	SA max.	TS avg.	TS min.	TS max.
hand_00(6)	2.8000	1	3	3	3	3
hand_01(6)	6	1	6	6	6	6
hand_02(8)	4.8000	1	5	5	5	5
hand_03(6)	4	1	4	4	4	4
rnd_00(32)	3.3	1	5	9	9	9
rnd_01(87)	1.9000	1	3	7	7	7
rnd_02(79)	3.3000	1	6	14	14	14
rnd_03(78)	5	1	10	20	20	20

Result: Execution Time [seconds] 10 times

testcase (# vertices)	SA avg.	SA min.	SA max.	TS avg.	TS min.	TS max.
hand_00 (6)	13.4950	11.0522	15.6396	0.2092	0.1941	0.2352
hand_01 (6)	11.0798	10.6093	11.8158	0.2786	0.2716	0.2865
hand_02 (8)	11.6497	11.2203	12.7827	0.3778	0.3648	0.4020
hand_03(6)	11.5965	11.2322	12.9180	0.2165	0.2059	0.2745
rnd_00(32)	14.2499	13.6432	15.0163	8.3888	8.0266	9.5017
rnd_01(87)	11.7813	10.9269	12.7644	83.7158	80.9691	87.5815
rnd_02(79)	12.6523	10.3767	25.8952	95.0464	89.2705	107.2564
rnd_03(78)	9.9625	9.4162	10.4410	124.7628	123.3009	126.7900

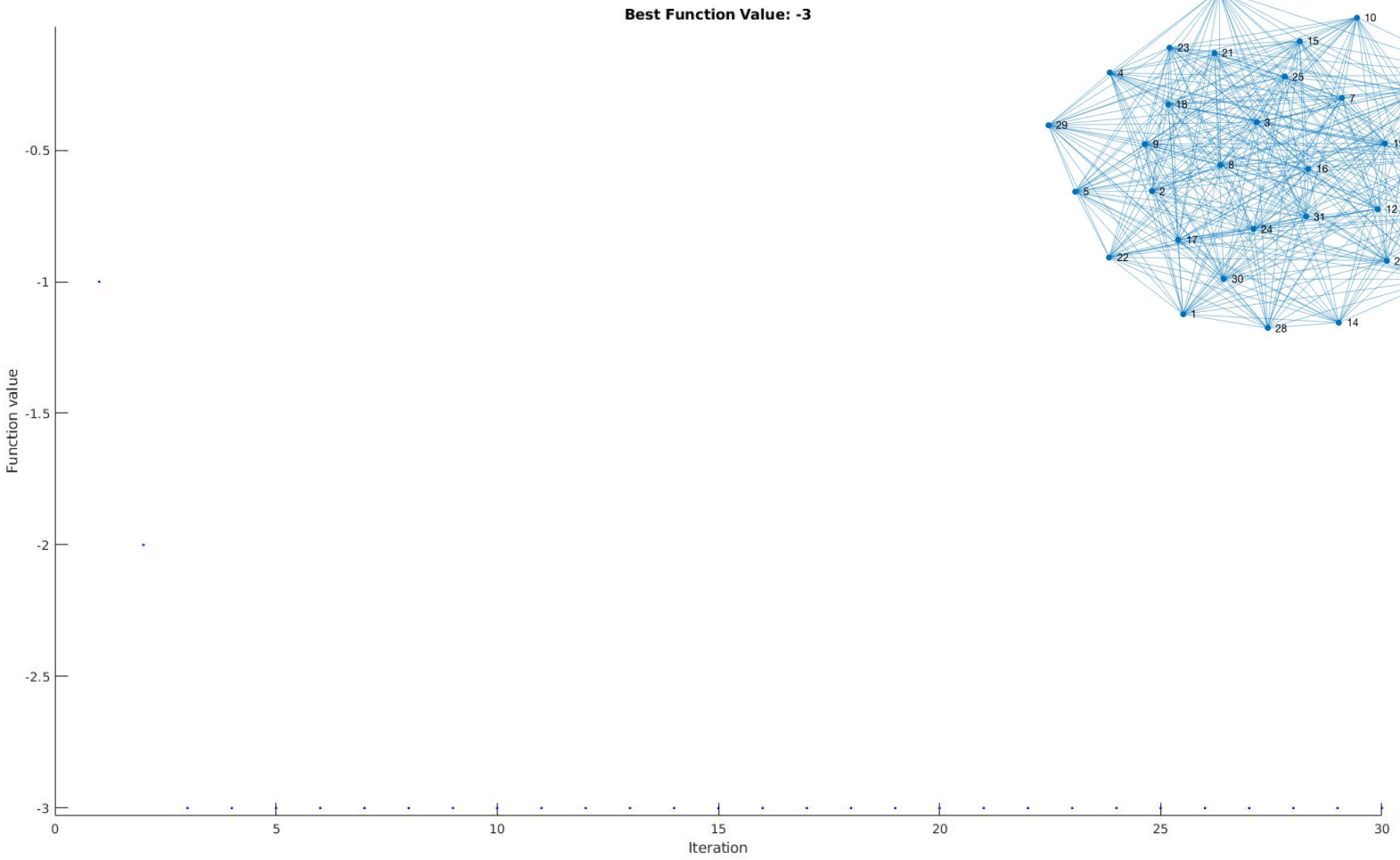
One of Target Graph



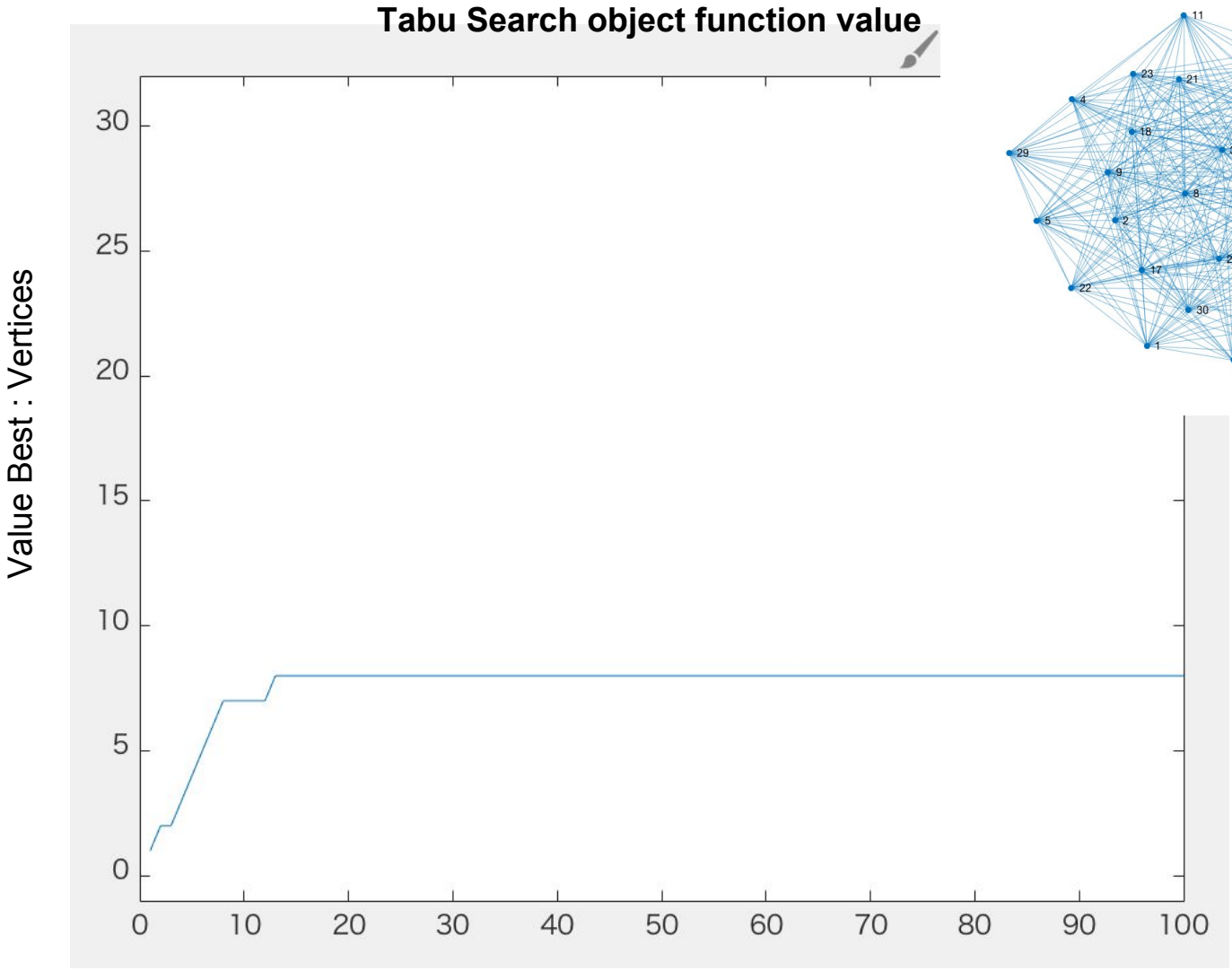
V = 32

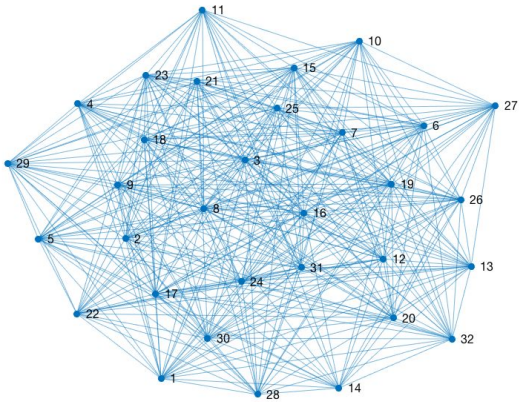
E = 375

One of objective function graph for SA



One of Object function graph for TS





Result

We confirmed that both algorithms correctly move states and recognize correct cliques for small cases (hand_*).

Tabu search is faster and has larger objective function values than simulated annealing for small cases.

For the large cases, tabu search is slower than simulated annealing. Still the objective function values of tabu search are better than that of simulated annealing.

Result: Memory Usage

- SA
  - $O(V^2)$
- TS
  - $O(V^2)$
    The size of tabu list is V-1.

Discussion

As a result table,

- SA’s result is different for each run, but TS’s result is same.

  ->SA is difficult to get out of local solution.

  ->TS might be also difficult to get out unless using LTM.

- The maximum clique problem is prone to local solution.

  ->In SA, the transition from initial state is random.

  ∗ The influence of initial state is low.

  ->In TS, it transits from initial state to the better neighborhood.

  ∗ The influence of initial state is strong.

  (In ILS, The local search method implemented is better for TS.)

Conclusion

We compared tabu search and simulated annealing for “Maximum Clique Problem.”

Tabu search has better objective function values than simulated annealing.

Therefore, we conclude that tabu search is more suitable algorithm than simulated annealing for this problem.