

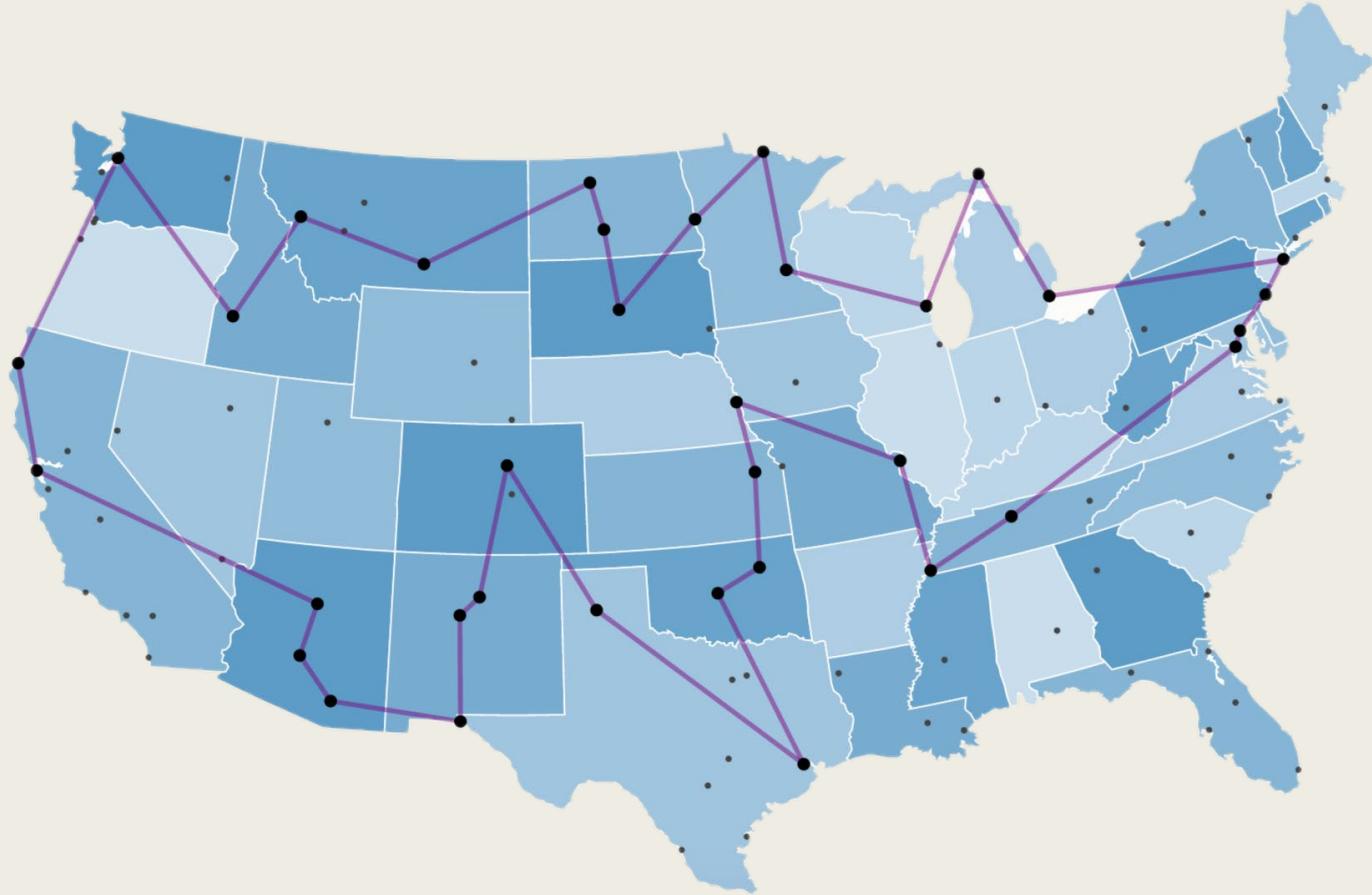
AN HYBRID METAHEURISTIC APPROACH TO THE TRAVELLING SALESMAN PROBLEM



DARIO BERTAZIOLI
FABRIZIO D'INTINOSANTE
MASSIMILIANO PERLETTI

INTRODUCTION

Travelling Salesman Problem



TSP

is an algorithmic problem tasked with finding the shortest route between a set of points and locations that must be visited

DATASETS

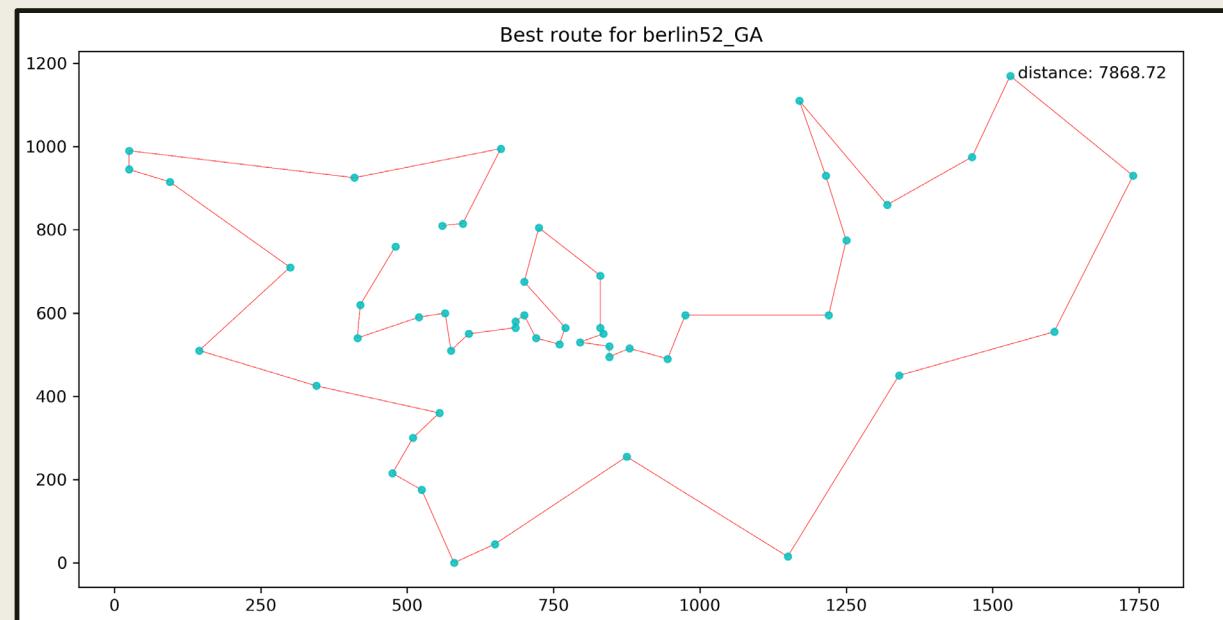
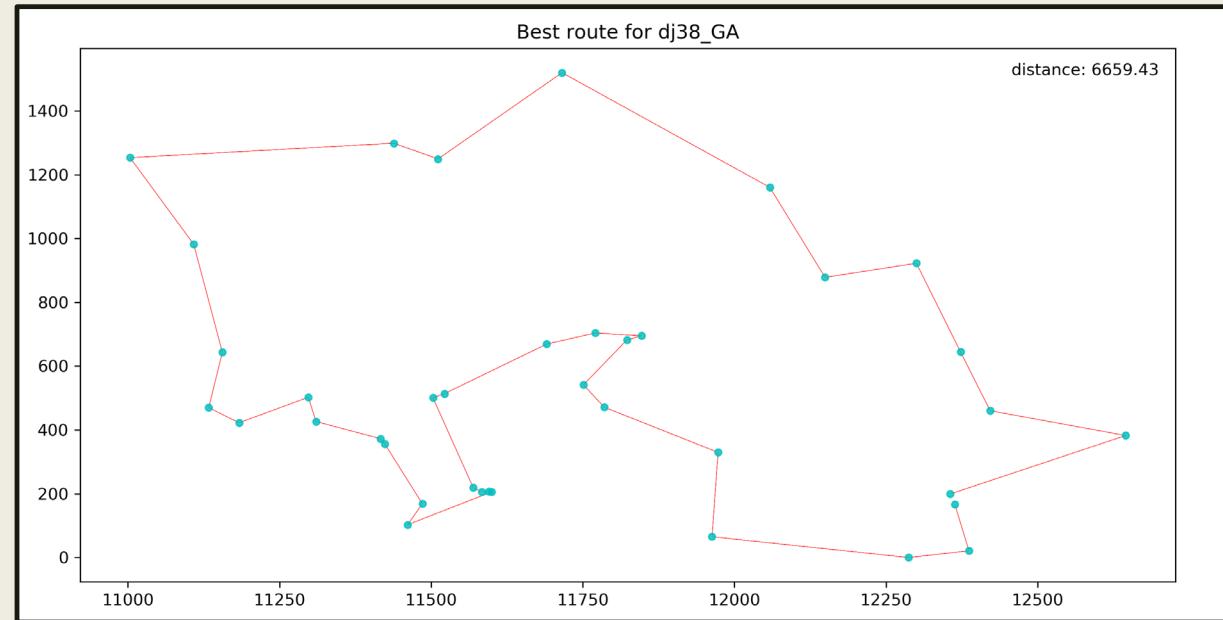
■ dj38

■ berlin52

■ ch130

■ d198

■ pr1002



METHODOLOGICAL APPROACH

ACO - Ant-Q - GA - KGA

ACO FAMILY ALGORITHMS

- inspiration from biology
- accomplish difficult tasks exploiting collaboration
- pheromone as a communication channel
- the more ants follow a specific path,
the more likely it becomes to be followed
- shortest path problems



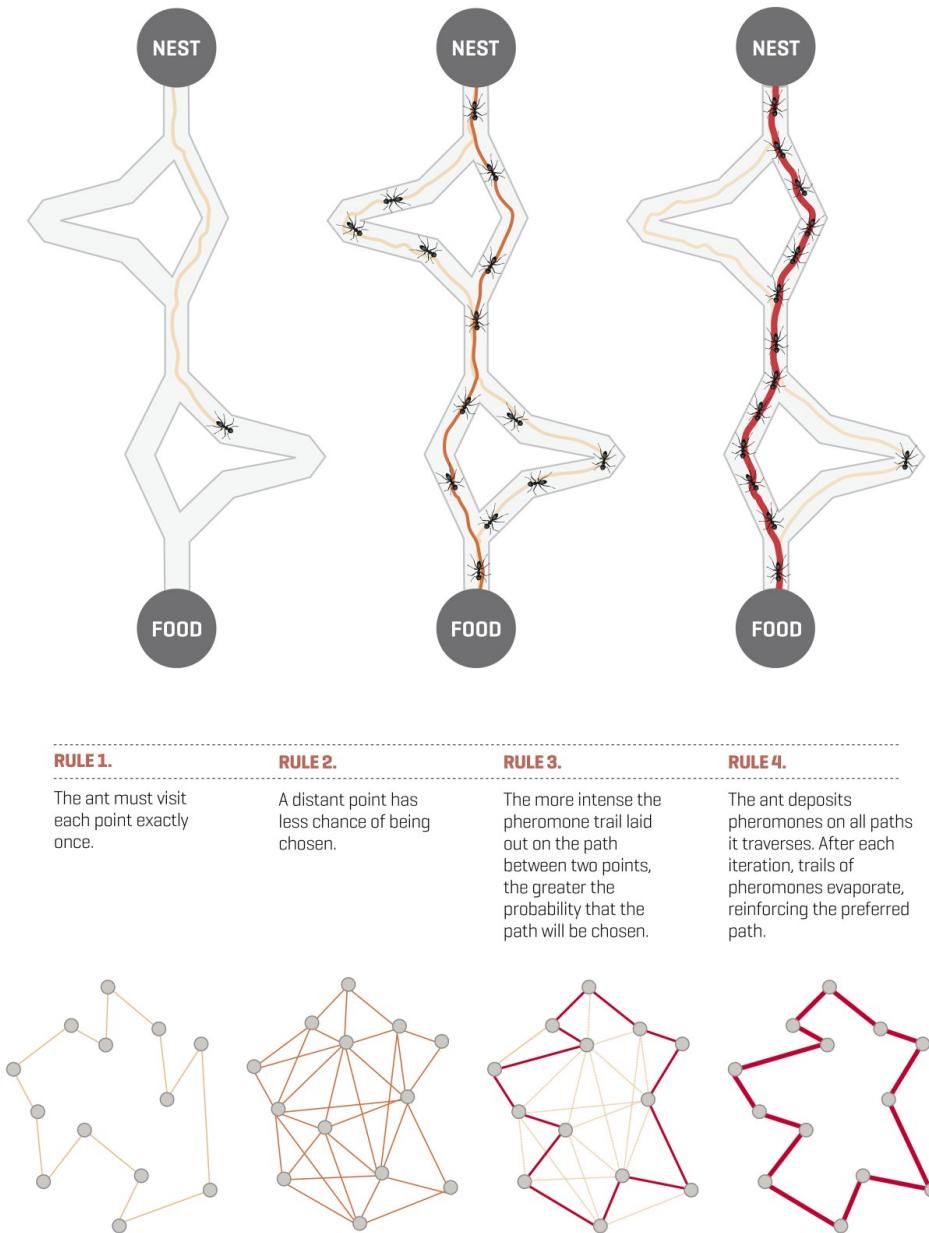
Ant Colony Optimization

- multi-agent collaborative approach
- pheromone evaporation and deposit
 - exploration vs exploitation tradeoff

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau_{ij}$$

- artificial ants might have different aims
 - pheromone guided moves vs heuristically driven ones

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta} \quad \text{if } j \in N_i^k$$



Dorigo, Gambardella, Ant Colony System:
A Cooperative Learning Approach to the Travelling Salesman Problem.
In: IEEE Transactions on Evolutionary Computation, Vol.1 No.1, pp.53-66

Ant-Q

a reinforcement learning approach

main idea

- integrate a Q-learning-like rule in the pheromone update procedure

$$\tau_{ij} = (1 - \alpha)\tau_{ij} + \alpha(\Delta\tau_{ij} + \gamma \max_{l \in N_j^k} \tau_{jl})$$

- and in the single decision process

$$s = \begin{cases} \arg \max_{a \in J_k(s)} [Q(s, a)]^\alpha [\eta(s, a)]^\beta & \text{if } q \leq q_0 \\ S & \text{otherwise} \end{cases}$$

Parallel Implementation Hints

- ACO algorithms are computational expensive, but... naturally parallelizable!
- parallel systems (*parent/children*)
 - running on each core
- local pheromone matrices
- global updates every iteration (*parent*)
 - providing each child with an updated mean pheromone matrix
- technically
 - Open MPI (<https://www.open-mpi.org>)
 - mpi4py (<https://mpi4py.readthedocs.io/en/stable>)

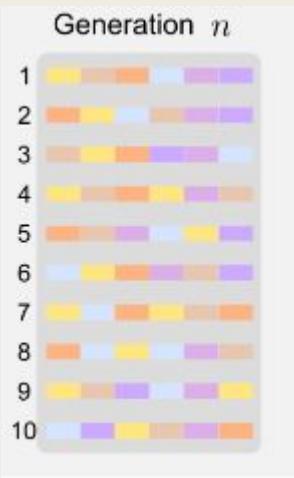


GENETIC ALGORITHM

- based on a biological metaphor (*Darwinism*)
- individuals as chromosomes (*genotypes*) into a population
- fitness function to evaluate each individual
- selection operator to choose best individuals (*natural selection*)
- crossover operator as gene transfer
- random mutation at each generation

Genetic Algorithm

generate random population
(chromosomes) - T_p



START

mutation - T_m



evaluation (rank pop)



selection mechanism and elitism - $elite_n$

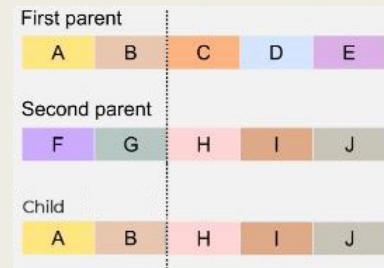
tournament selection



roulette-wheel selection

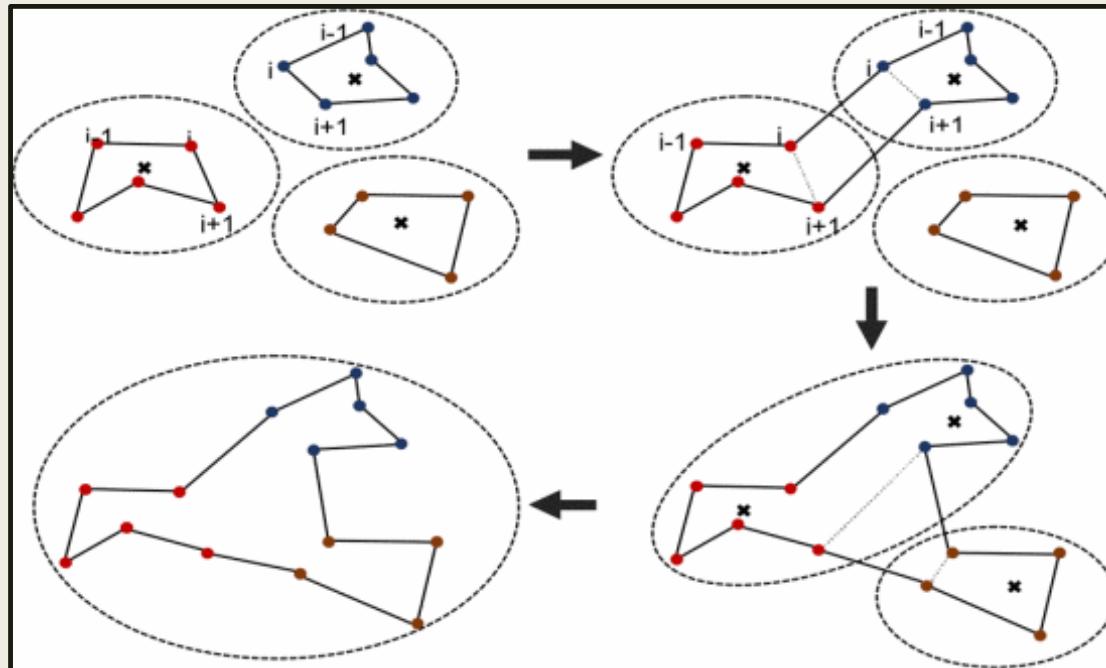


crossover (single-point)



K-means Genetic Algorithm

- hybrid implementation of GA
- transform wide problem into smaller sub-problems (clusters)



THREE STEPS

- create cluster (*k-means clustering algortihms*)
- solve sub-problems with GA (*intra-group evolution operation*)
- reconstruct optimal solution from sub-problems solutions (*inter-group connection*)

N.B. to achieve better performance use GA to order centroids

Test Parameters GA – KGA

G A

5 run

Tp = 300 (pr1002 Tp = 125)

elite_n = 30 (10%)

tournament selection

Tm = 0.01

5k generations

K G A

5 run

Tp = 125

elite_n = 25 (20%)

roulette-wheel selection

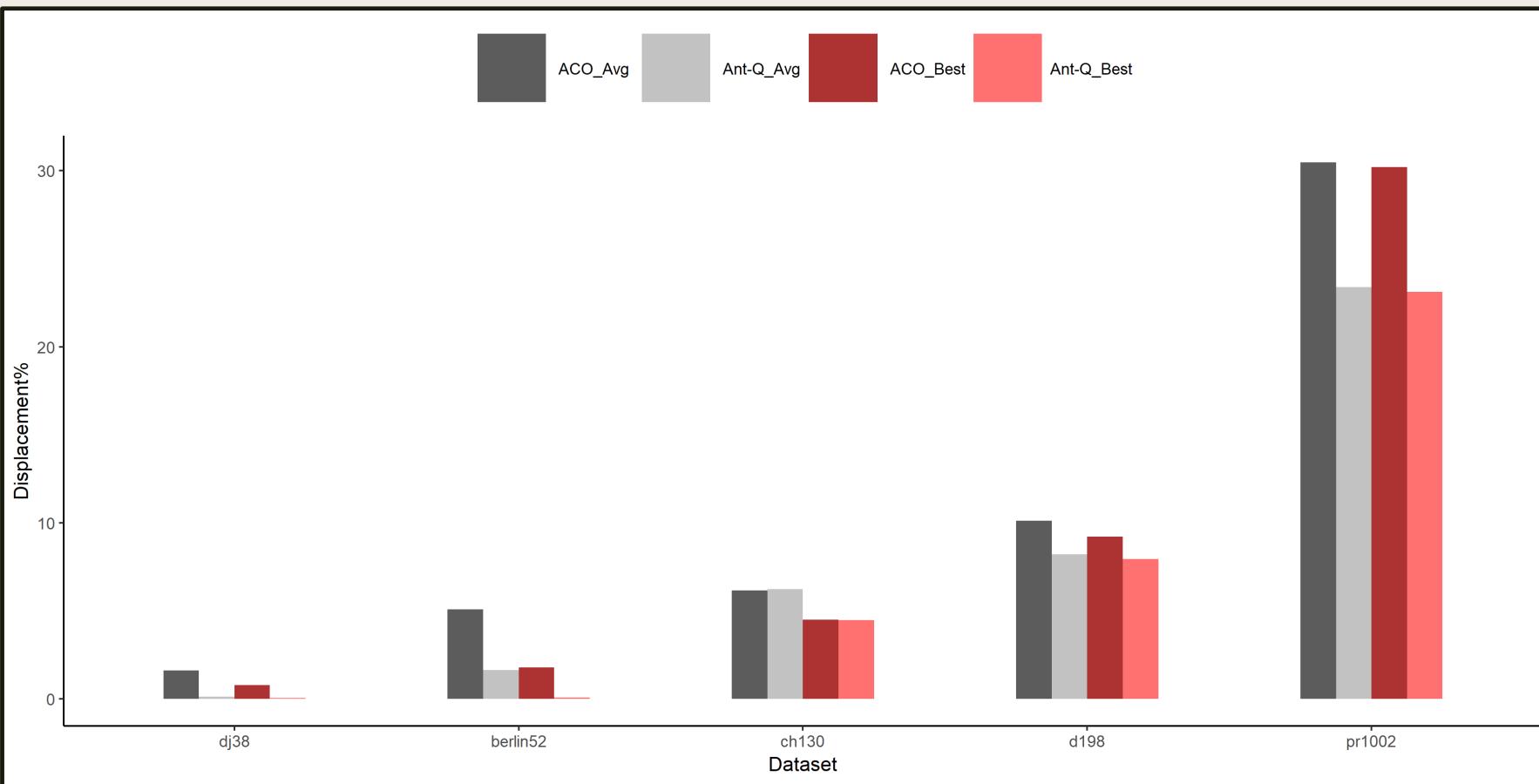
Tm = 0.01

1k generations (intra-group) + 1k generations (to order centroids)

K cluster = 2, 3, 8, 30, 60 k (initial dataset order, slide 5)

RESULTS

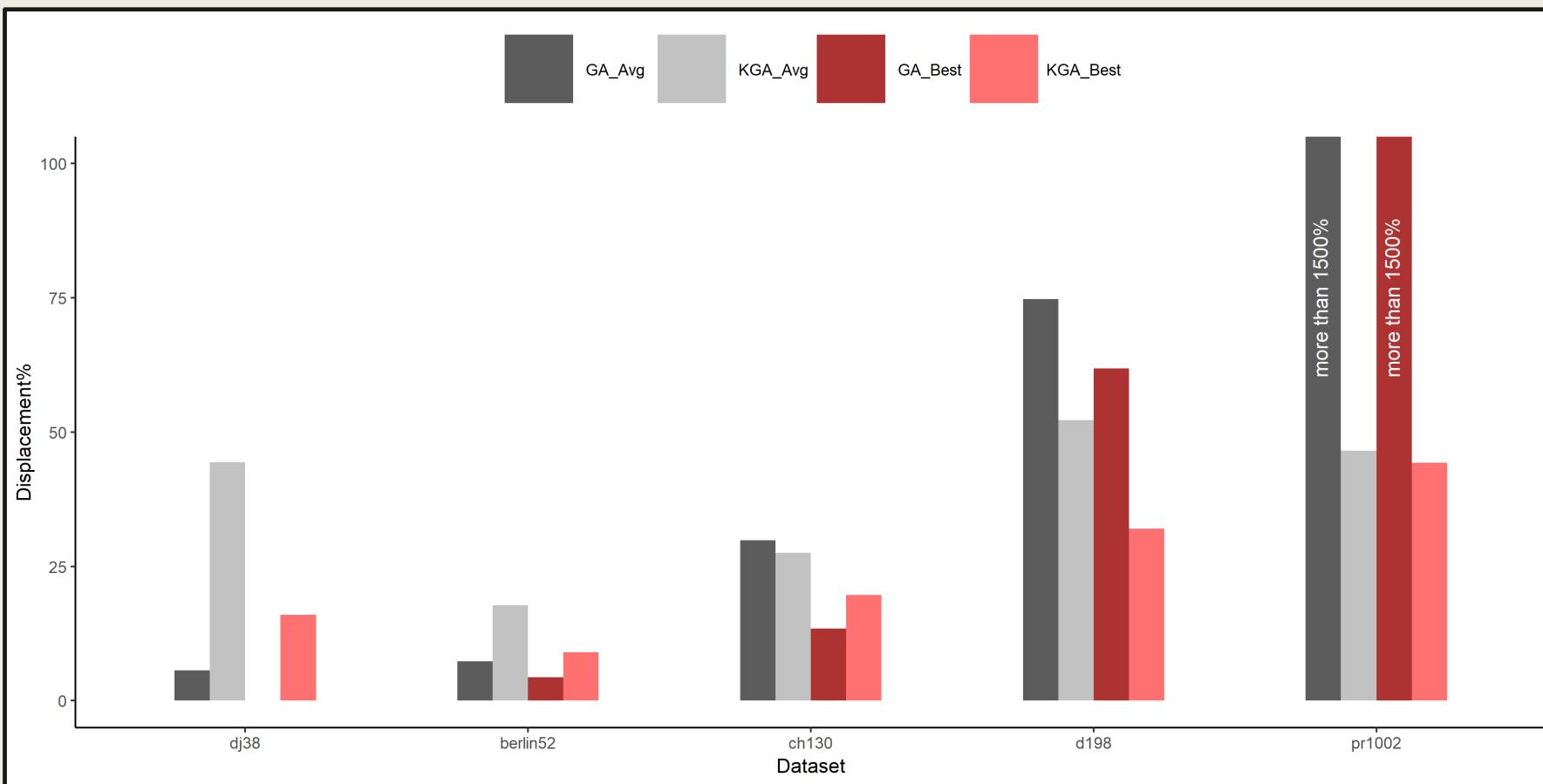
ACO - ANT-Q



17

Dataset	ACO							Dataset	Ant-Q						
	OptimalTour	Mean	SD	Best	% Avg Dist	% Best Dist	Avg time (s)		OptimalTour	Mean	SD	Best	% Avg Dist	% Best Dist	Avg time (s)
dj38 (1000 it)	6656	6763.03	75.16	6708.04	1.61	0.78	75	dj38 (1000 it)	6656	6663.99	4.16	6659.43	0.12	0.05	37
berlin52 (1000 it)	7542	7925.39	162.00	7677.66	5.08	1.80	320	berlin52 (1000 it)	7542	7666.17	113.87	7548.99	1.65	0.09	170
ch130 (500 it)	6110	6487.19	74.20	6385.46	6.17	4.51	9240	ch130 (500 it)	6110	6491.31	70.01	6383.42	6.24	4.47	8400
d198 (100 it)	15780	17376.23	105.81	17235.44	10.12	9.22	6540	d198 (100 it)	15780	17075.81	39.69	17032.75	8.21	7.94	2880
pr1002 (10 it)	259054	337976.61	833.96	337309.96	30.47	30.21	6180	pr1002 (10 it)	259054	319646.49	1009.39	318960.00	23.39	23.12	6120

GA - KGA



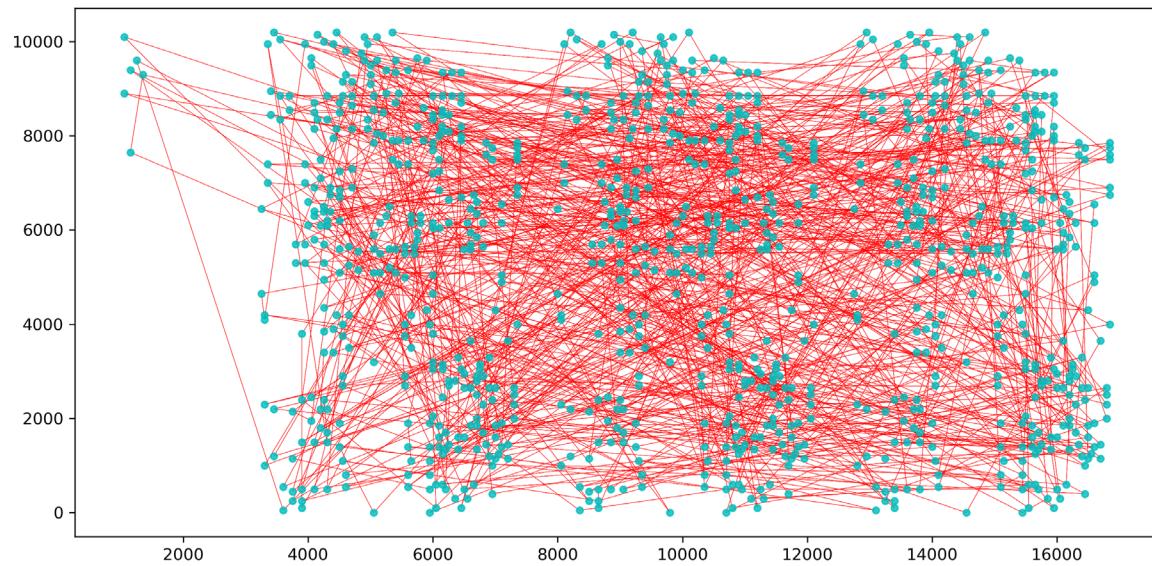
18

Dataset	GA						
	OptimalTour	Mean	SD	Best	% Avg Dist	% Best Dist	Avg time (s)
dj38	6656	7032.91	574.34	6659.43	5.66	0.05	263
berlin52	7542	8095.35	167.27	7868.72	7.34	4.33	286
ch130	6110	7933.02	1041.27	6929.81	29.84	13.42	435
d198	15780	27579.73	1869.54	25537.85	74.78	61.84	604
pr1002	259045	4666413.09	12608.57	4652556.08	1701.39	1696.04	2641

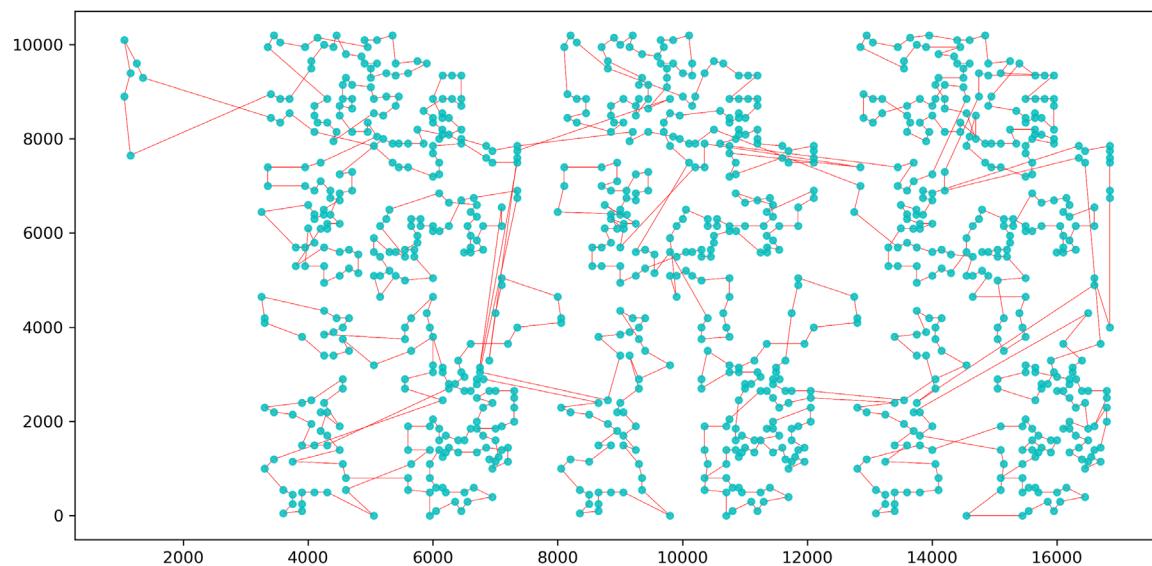
Dataset	KGA						
	OptimalTour	Mean	SD	Best	% Avg Dist	% Best Dist	Avg time (s)
dj38	6656	9608.77	1817.02	7720.03	44.36	15.99	140
berlin52	7542	8879.59	446.69	8218.95	17.74	8.98	188
ch130	6110	7790.85	312.17	7314.08	27.51	19.71	409
d198	15780	24023.86	3066.94	20830.94	52.24	32.01	1413
pr1002	259045	379495.87	6343.57	373865.26	46.50	44.32	4456

Wide tours comparison (GA - KGA)

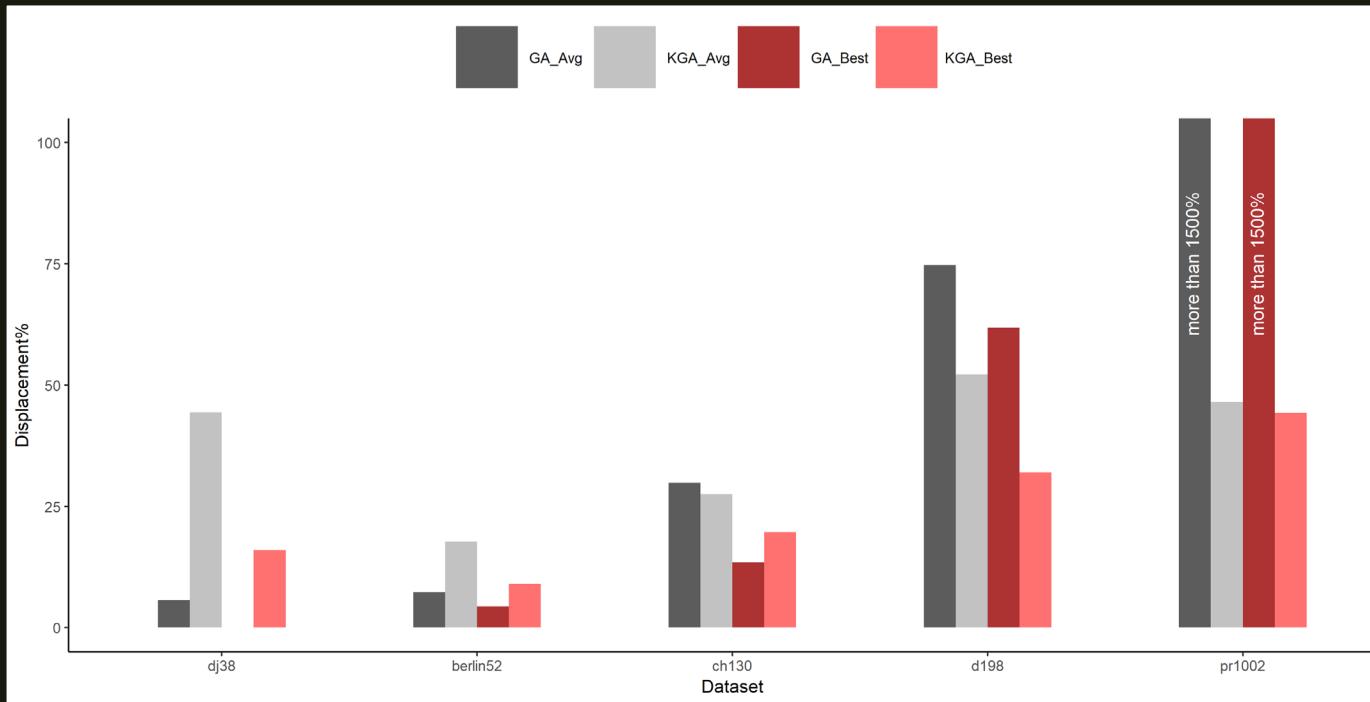
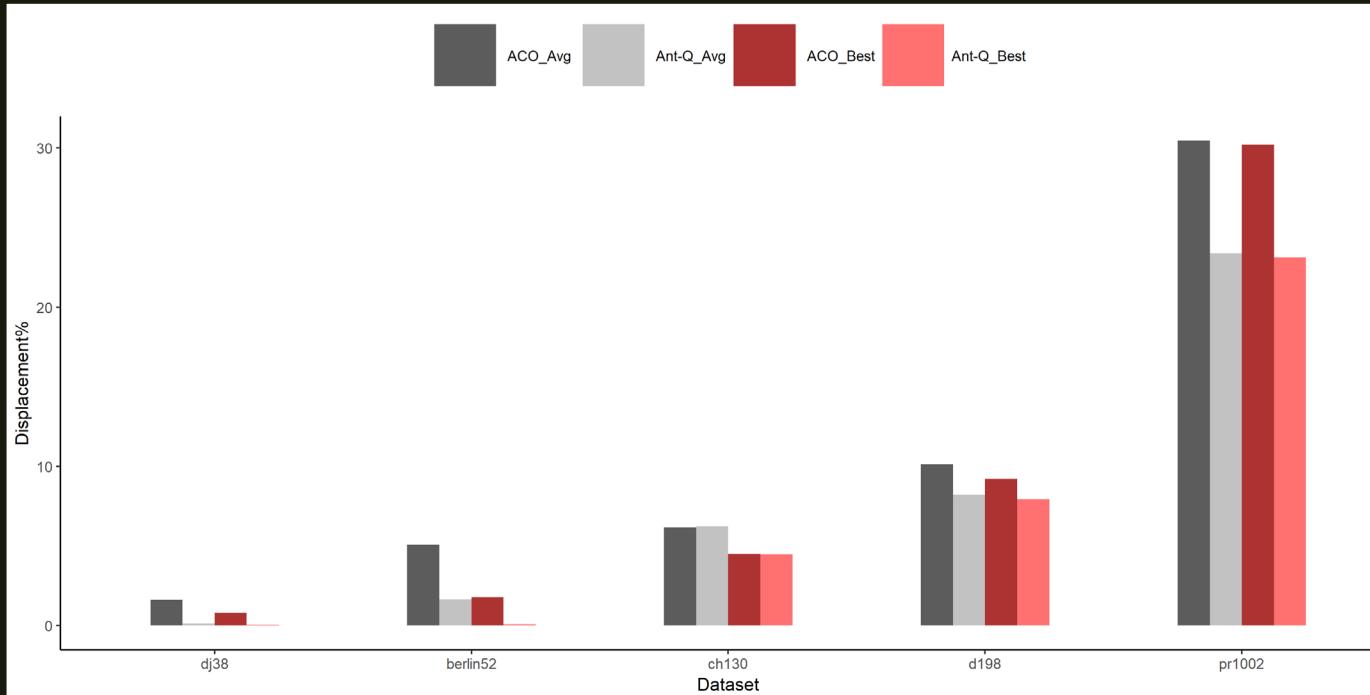
GA_pr1002_example route



KGA_pr1002_example route



Compare performances (ACO - Ant-Q - GA - KGA)



CONCLUSIONS

KEY POINTS



EFFECTIVE IMPROVEMENT OF THE
HYBRIDIZED APPROACH WITH
RESPECT TO THE CLASSIC VERSION
OF EACH ALGORITHM



ANT FAMILY – BEST PERFORMANCES
ON SMALL/MEDIUM SIZED PROBLEMS



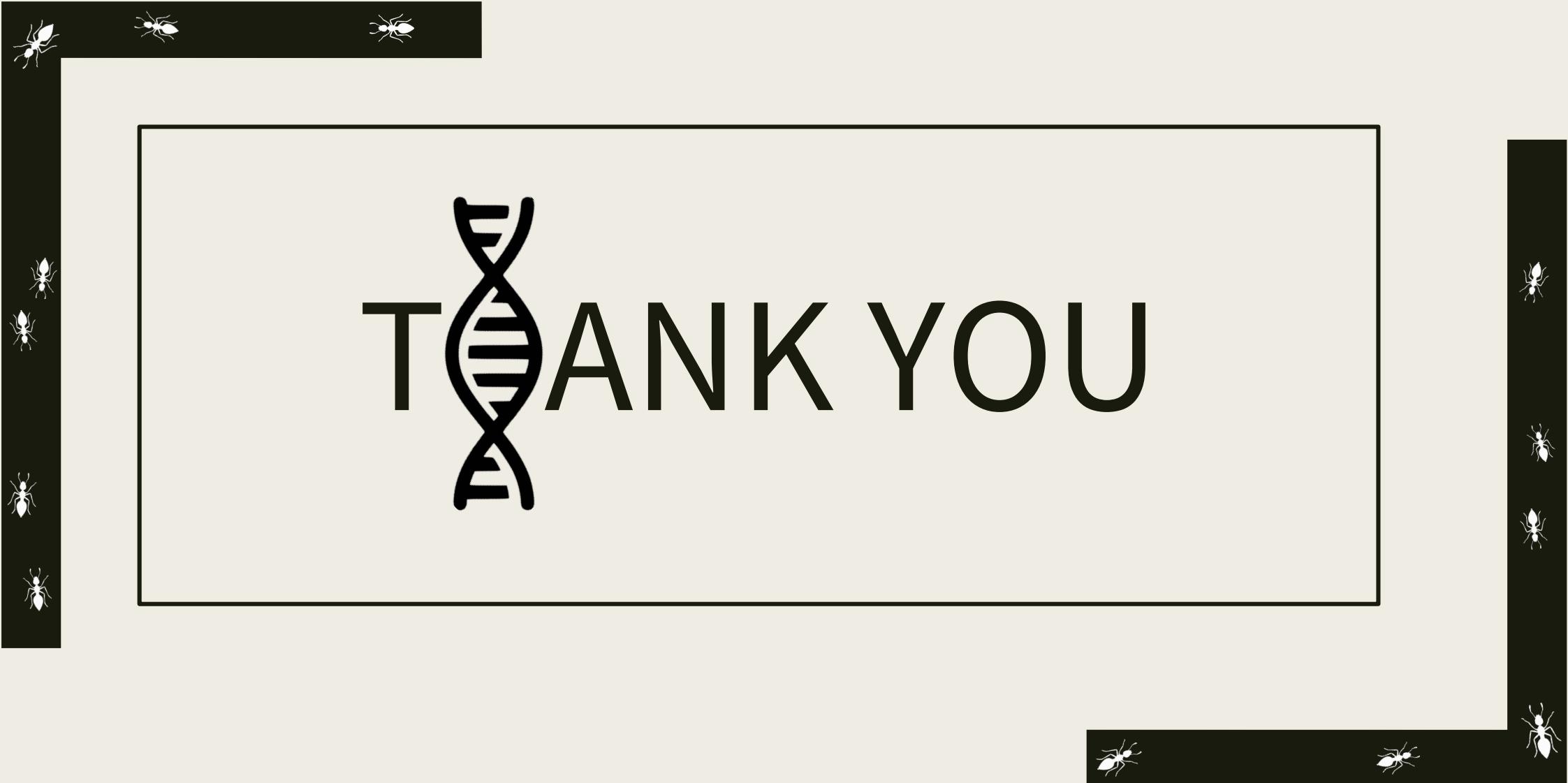
KGA – MORE PERFORMANT
ON LARGE DATASETS

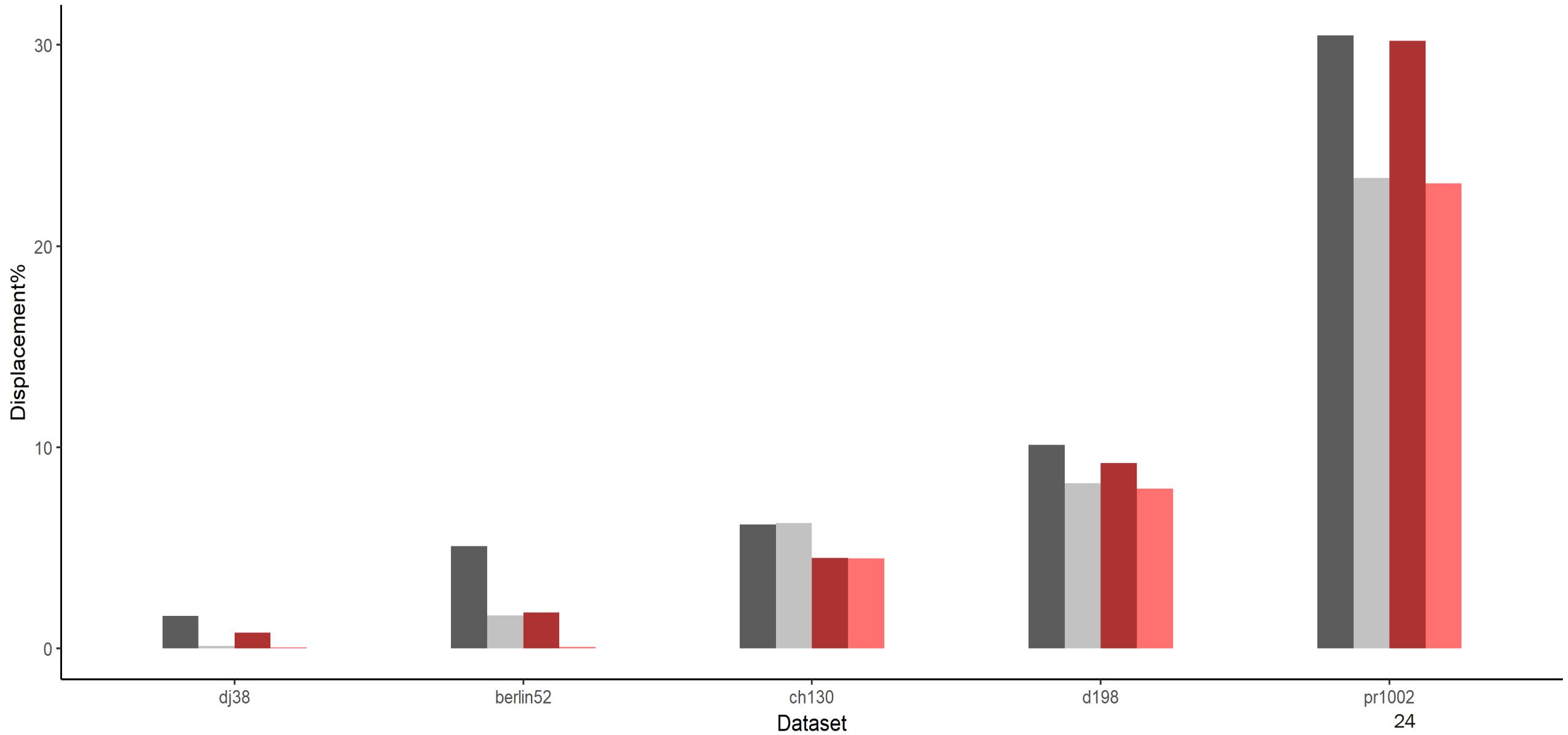


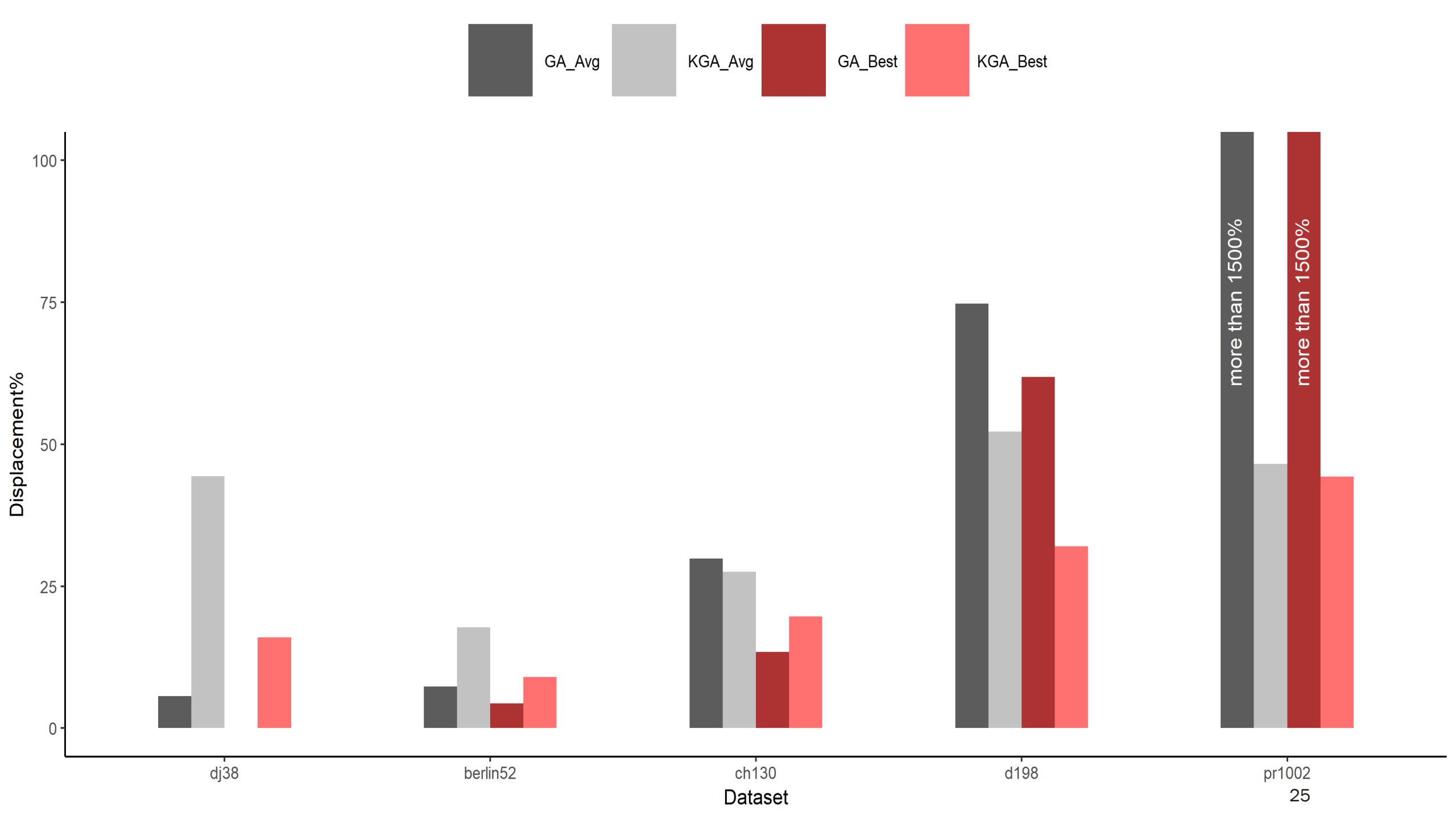
ACO AND ANT-Q SLOWER ON HIGHER
DIMENSIONS BUT FIND BETTER
SOLUTIONS THAN EA

FUTURE WORK

it could be interesting to further combine
Ant-Q and KGA







Dataset	ACO						
	OptimalTour	Mean	SD	Best	% Avg Dist	% Best Dist	Avg time (s)
dj38 (1000 it)	6656	6763.03	75.16	6708.04	1.61	0.78	75
berlin52 (1000 it)	7542	7925.39	162.00	7677.66	5.08	1.80	320
ch130 (500 it)	6110	6487.19	74.20	6385.46	6.17	4.51	9240
d198 (100 it)	15780	17376.23	105.81	17235.44	10.12	9.22	6540
pr1002 (10 it)	259054	337976.61	833.96	337309.96	30.47	30.21	6180

Dataset	Ant-Q						
	OptimalTour	Mean	SD	Best	% Avg Dist	% Best Dist	Avg time (s)
dj38 (1000 it)	6656	6663.99	4.16	6659.43	0.12	0.05	37
berlin52 (1000 it)	7542	7666.17	113.87	7548.99	1.65	0.09	170
ch130 (500 it)	6110	6491.31	70.01	6383.42	6.24	4.47	8400
d198 (100 it)	15780	17075.81	39.69	17032.75	8.21	7.94	2880
pr1002 (10 it)	259054	319646.49	1009.39	318960.00	23.39	23.12	6120

Dataset	GA							
	OptimalTour	Mean	SD	Best	% Avg Dist	% Best Dist	Avg time (s)	
dj38	6656	7032.91	574.34	6659.43	5.66	0.05	263	
berlin52	7542	8095.35	167.27	7868.72	7.34	4.33	286	
ch130	6110	7933.02	1041.27	6929.81	29.84	13.42	435	
d198	15780	27579.73	1869.54	25537.85	74.78	61.84	604	
pr1002	259045	4666413.09	12608.57	4652556.08	1701.39	1696.04	2641	

Dataset	KGA							
	OptimalTour	Mean	SD	Best	% Avg Dist	% Best Dist	Avg time (s)	
dj38	6656	9608.77	1817.02	7720.03	44.36	15.99	140	
berlin52	7542	8879.59	446.69	8218.95	17.74	8.98	188	
ch130	6110	7790.85	312.17	7314.08	27.51	19.71	409	
d198	15780	24023.86	3066.94	20830.94	52.24	32.01	1413	
pr1002	259045	379495.87	6343.57	373865.26	46.50	44.32	4456	

Algorithm Ant Colony Optimization

Main Algorithm

```
0: initialize best_dist and best_path to None
1: for generation in generations:
2:   create n_ants artificial ants
3:   for one_ant in ants:
4:     make a single ant path (see Make path)
5:     compute the path length
6:     update best_dist and best_path
7:   update the pheromone matrix
    (local update only for child processes, according to Eq. (10))
8:   every a certain n of iterations:
9:     update the global pheromone matrix
      (shared in MPI environment among master&child.)
10:  return best_dist, best_sol
```

Make path

```
1: start from a vertex
2: add start vertex to visited nodes
3: for each remaining vertex:
4:   list the neighbours
5:   list the not yet visited neighb
6:   calculate the probability of choosing a vertex (according to Eq. (7))
7:   choose the vertex according to probability
8:   add the chosen vertex to the visited list
9: return the chosen vertex id
```

Local update pheromone matrix

```
1: for ant in ant_colony :
2:   for each vertex of one_ant_path :
3:     increase pheromone_matrix between current and next vertex of  $\Delta\tau$ 
    (according to Eq. (10))
```

Global update pheromone matrix (parallelism)

```
1: gather from MPI env all the pheromone matrices
2: if process is the parent process (rank==0):
3:   for each element average over the n_cores matrices.
4: broadcast obtained pheromone matrix to the other
  processes
```

Algorithm Ant-Q algorithm

Main Algorithm

```
0: initialize best_dist and best_path to None
1: for generation in generations:
2:   create n_ants artificial ants
3:   for each ant:
4:     make a single ant path (see Make path)
5:     compute the path length
6:     update best_dist and best_path
7:   update pheromone matrix with delayed rewards
    (according to Eq. (14))
8:   update the global pheromone matrix
    (shared in MPI environment among master&child.)
9: return best_dist, best_sol
```

Make path

```
1: start from a vertex
2: add start vertex to visited nodes
3: for each remaining vertex:
4:   list the neighbours
5:   list the not yet visited neighb
6:   generate a random number  $q \in \{0, 1\}$ 
7:   if  $q < q_0$ : (threshold)
8:     select next vertex according to Eq. (12)
9:   else:
10:    calculate the probability of choosing a vertex
11:    (according to Eq. (7))
12:    choose the vertex according to probability
13:    add the chosen vertex to the visited list
14:    give local rewards (local update pheromone matrix)
15: return the chosen vertex id
```

Local update pheromone matrix

```
1: for ant in ant_colony :
2:   for each vertex of one_ant_path :
3:     increase pheromone_matrix between current
       and next vertex of a  $\Delta\tau$  (according to Eq. (11))
```

Global update pheromone matrix (parallelism)

```
1: gather from MPI env all the pheromone matrices
2: if process is the parent process (rank==0):
3:   for each element average over the n_cores matrices.
4: broadcast obtained pheromone matrix to the other
  processes
```

Algorithm Genetic Algorithm

```
1: procedure Genetic(Tm, Tp, elite_n, Selection, MaxGen)
2:   Pop ← GeneratePopulation(Tp)
3:   Pop ← Evaluation(Pop)
4:   for i = 1 ... MaxGen do
5:     Pop ← Selection(elite_n from Pop)
6:     Pop ← Crossover(Pop)
7:     With probability Tm do:
8:       Pop ← Mutation(Pop)
9:   end for
10:  return the best solution in Pop
11: end procedure
```

Algorithm K-Means Algorithm

```
1: Set the K cluster centers randomly;
2: repeat
3:   for each vertex do
4:     Calculate distance measure to each cluster;
5:     Assign it to the closest cluster;
6:   end
7:   recompute the cluster centers positions;
8: until stop criteria are met;
```

Algorithm KGA

```
1: input an TSP;
2: K-Means is adopted to cluster the TSP into k sub-problems
3: For each sub-prob  $i = 1$  to  $k$ , do:
4:   repeat
5:     GA procedure
6:   until stop criteria are met
7:   Output shortest path for sub-problem  $i$ ;
8: End
9: Seek for the best combining seq  $S$  with GA
10: Combine all those shortest path into one tour
11: Output the shortest whole travelling tour.
```

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}]^\alpha [\eta_{il}]^\beta} \quad \text{if } j \in N_i^k , \quad (7)$$

where

- η_{ij} indicates an heuristic value specified according to the problem; in the TSP case, we considered it to be $1/d_{ij}$, coherently with the choice of [2].
- τ_{ij} is the pheromone quantity on the path between the i -th and j -th nodes,
- N_i^k the neighbourhood region of the k -th ant.
- α and β are the parameters used to set the relative importance of the pheromone trail and the heuristic value. In our implementation, we attempted to change the values of those parameters in order to explore a set of possible configurations. We experienced that:
 - * as $\alpha \rightarrow 0$, the pheromone track become less important and the ants tend to choose the closest cities, resulting in a much more “greedy” search;
 - * vice-versa, when $\beta \rightarrow 0$, heuristic values are almost ignored and only the tracks are considered in the decision making;

Randomly initialise $Q(s, a)$

Repeat for each episode

 Initialise the current state s

 While s is not terminal state

 Choose action a at the current state s according to the policy (e.g. ϵ -greedy)

 Take action a , observe reward r and next state s'

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

$s \leftarrow s'$;

$$V^\pi(s) = E_\pi \left\{ \sum_i \gamma^i r_{t+1, i+1} \mid s_t = s \right\},$$

$$Q^\pi(s, a) = E_\pi \left\{ \sum_i \gamma^i r_{t+1, i+1} \mid s_t = s, a_t = a \right\},$$

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + \alpha(r + \gamma \max_{a'} Q(s', a') - Q(s, a)).$$