Project report

Authors: Matej Luburić, Filip Šimek

Problem Description

Problem title is Capacitated vehicle routing problem with time windows. Given information about depot, vehicles, and customers we need to find an optimal solution in which all customers are served with respect to all defined constraints. A depot is the starting point of all vehicle routes. Each vehicle has a capacity limit. Each customer has defined coordinates, demand value, service, ready and due time. Ready and due times are time constraints in this problem. A vehicle is serving customers fulfilling their demand. Arrival time to each customer is calculated as a maximum value between the customer's ready time and the sum of the previous customer's arrival time, its service time, and the upper ceiling of the distance between two customers. The primary objective of this problem is to minimize the number of vehicles needed to serve all customers. There is also a limit on the number of vehicles that can be used. The secondary objective is to minimize the sum of distances on all routes. As there are many customers, finding an exact solution to this problem is not an option, so we need to implement some optimization algorithm. Firstly, we construct an initial solution using a greedy algorithm and then we optimize it using a simulated annealing algorithm.

Algorithm description

Used languages: **Java** for project optimization, **Python** for visualization

Solution representation: Customer input information is represented by a Customer class. This information is immutable. Mutable customer information which is it's served time and position on the route are represented in a CustomerCalc class with reference to the corresponding customer. The Vehicle is represented by Vehicle class. It has information about customers served and route length. The whole solution is represented by a Solution class that stores a list of used vehicles.

Objective function: Solutions are compared primarily based on number of vehicles used in them. If these values are equal, a better solution is one that has a lower total route distance.

Fitness function/s: Solutions are compared primarily based on the number of vehicles used in them. Next, the number of customers in the shortest route is compared and a better solution has a lower number of customers in the shortest route. If all previous values are equal, the better solution is one with the lower total route distance.

Number of customers in the shortest route guides search to solutions with lower number of vehicles.

Initial solution construction: Initial solution is constructed with a greedy algorithm.

Initial solution optimization: Initial solution obtained by the greedy algorithm is optimized with **simulated annealing algorithm (SA)**.

Neighborhood solution in SA is generated by NeighborhoodGenerator class. Two types of operators for improving the current solution are implemented: inter operators and intra operators. As the names suggest, inter operators are applied between two vehicle routes, while intra operators are applied inside one vehicle route. Main purpose of inter operators is to try to lower number of vehicles, while intra operators lower distance and therefore lower the vehicle return time to depot.

Four operators are implemented: inter/intra route relocate and swap customers.

- Inter route relocate (RelocateCustomerInterOp class) operator tries to move customer from a short vehicle route to the longer one.
- Inter route swap (TwoCustomersSwapInterOp class) operator first random selects some vehicles from current solution and for each of them centroid of customers positions is calculated. Two vehicles with the nearest centroid are chosen for random swap of customers between them.
- Intra route relocate (RelocateCustomerIntraOp class) operator tries to move customer to random position inside the route.
- Intra route swap (TwoCustomersSwapIntraOp class) operator tries to swap customer supply order inside route.

Algorithm pseudocode

Greedy algorithm pseudocode:

Algorithm 1: Greedy algorithm

```
initialize solution;
while exists unserved customers do
    take new vehicle;
    while unfilled capacity exists and the vehicle can return to the depot before its closing
     do
        if vehicle in depot then
            find the unserved customer with the earliest ready time within some number
              (this is a parameter) of the furthest customers from the depot;
        else
            within a certain number of the nearest customers from the last served user on
              the route, select the one with the minimum difference between the current
              route time and the potential route time after its serving;
        end
    end
    return to depot;
    add vehicle to solution;
end
```

While there are still unserved customers, a new vehicle is taken to serve the remaining unserved users until there is enough capacity in the vehicle and until there is enough time for the vehicle to return to the depot before it closes.

The logic of the order of serving the unserved user differs when the vehicle starts from the depot and when the vehicle starts from the previously served user.

If the vehicle starts from the depot, the customer with the lowest earliest time among the predetermined number of the most distant unserved customers will be served first. **The percentage of the most distant unserved customers from depot is a parameter**. Increasing this parameter increases the number of customers among which we are looking for the customer with the lowest earliest time and therefore we prefer to serve the customer with the lowest earliest time first rather than serving the most distant customer from the depot first.

If the vehicle starts from the last served customer, a certain number of the nearest neighbors is chosen first. For each of these neighbors, the difference between the current route time and the current route time plus the amount when that neighbor would be served after the current customer, is calculated. Customer with minimal that difference is served first. The key is that customers can be close but with great ready time, so we have to 'wait' for it to become available and thus increase route time. So, it becomes more optimal to travel to a further customer who has less ready time. **Number of the nearest neighbors is a parameter**. Increasing it gives preference to serving those customers who are earlier ready to be served. Lower it gives preference to 'wait' for less distant customer to be available.

Simulated annealing algorithm pseudocode:

Algorithm 1: Simulated annealing algorithm

initialize constants $TIME\_LIMIT, MAX\_ITER\_WITHOUT\_IMPROVEMENT$;
generate greedy initial solution s_0;
select initial temperature T_0;
$s \leftarrow s_0; s^{best} \leftarrow s_0; T \leftarrow T_0;$
$iter \leftarrow 0; no\_improvement\_iters \leftarrow 0;$
while $time < TIME\_LIMIT$ **do**
  if $no\_improvement\_iters \geq MAX\_ITER\_WITHOUT\_IMPROVEMENT$ **then** $s \leftarrow s^{best}$;
  generate neighborhood solution s^{neigh};
  if $f_{fit}(s^{neigh}) < f_{fit}(s)$ **then** $s \leftarrow s^{neigh}$;
  else if $f_{fit}(s^{neigh}) \geq f_{fit}(s)$ *and* $rand[0,1) < e^{-\frac{\triangle f}{T}}$ **then** $s \leftarrow s^{neigh}$;
  if $f_{obj}(s) < f_{obj}(s^{best})$ **then**
    $s^{best} \leftarrow s;$
    $no\_improvement\_iters \leftarrow 0;$
    decrease T;
  end
  $iter \leftarrow iter + 1; no\_improvement\_iters \leftarrow no\_improvement\_iters + 1;$
end

Simulated annealing is applied after greedy initial solution.

Beside some standard parameters, max iterations without improvement parameter is introduced. This parameter is the number of iterations without improving the current solution after which we set the best seen solution to the current solution. With this parameter we allow only a limited divergence of the current solution from the best seen solution. By increasing this parameter we allow greater divergence than the currently best seen solution.

Neighborhood solution is generated by applying previously mentioned operators: inter/intra swap and relocate. In each iteration all operators are applied. Operators return unchanged solution if they fail to obtain feasible solution.

Very slow homogenous decrement function is used.

T <- T / (1 + βT)

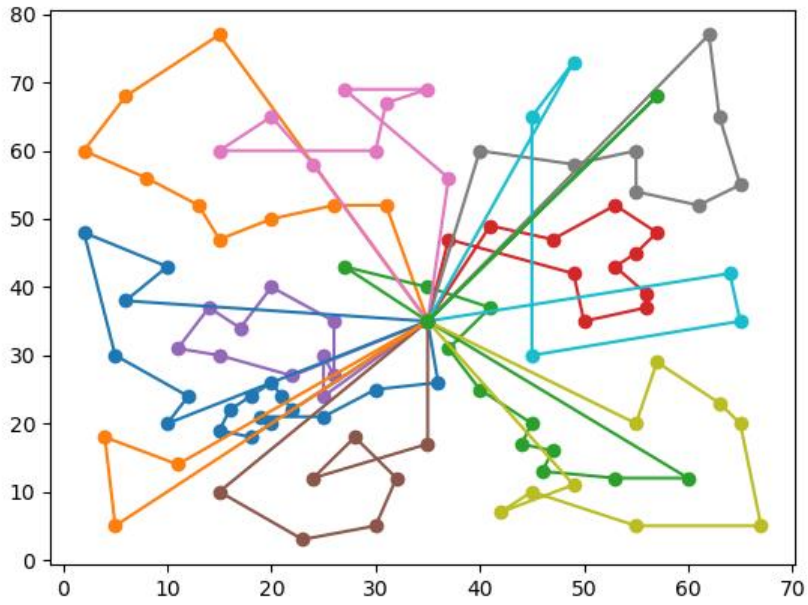
Where beta is very small real number.

To further slow down the temperature decrease it only decreases when the current solution is better than the best-seen solution. It was introduced not to exclude too early the option of accepting a worse solution than the current one and because it gives better results in practice.

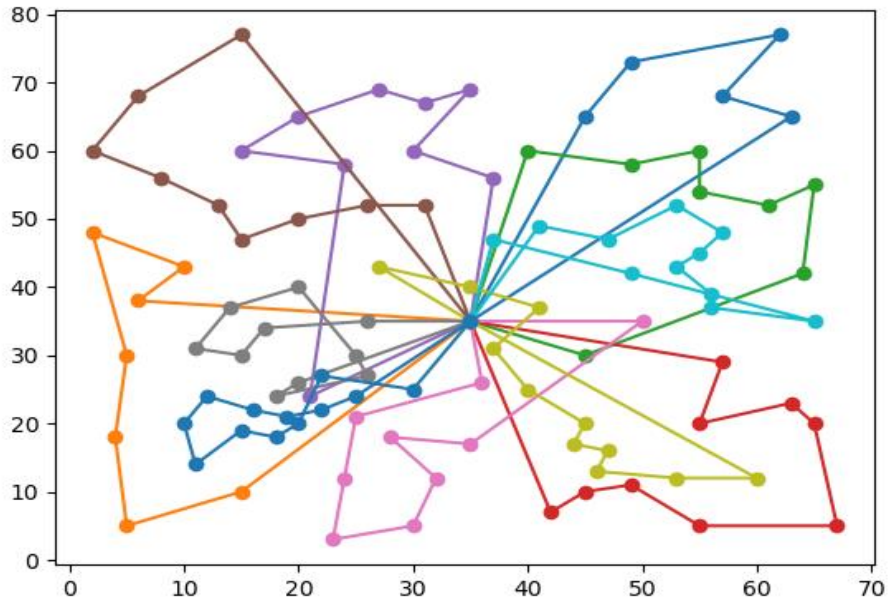
When comparing the neighboring solution with the current one, the fitness function is used, while when comparing whether the current solution is better than the best-seen solution, the

objective function is used. Fitness function, explained earlier, guides the search to solutions with fewer customers in the shortest route and thus to fewer vehicles in the end.

Analysis of results and discussion



Picture: Greedy output for instance 1



Picture: Best obtained output with SA algorithm for instance 1

The images above show the obtained solutions for the greedy algorithm and the SA that received the output from the greedy as input.

Instance	Total number of vehicles	Greedy algorithm	SA	Difference= SA - greedy
1	25	12	11	-1
2	50	32	19	-13
3	100	42	37	-5
4	150	36	20	-16
5	200	88	77	-11
6	250	67	19	-48

As shown in the table above, SA improves the initial solution for each instance. SA reduces the number of vehicles by 48 for instance 6. Intra operators contributed to reducing mileage on each of the routes, while inter operators managed to reduce the number of vehicles.

The percentage of the most distant unserved customers from depot

The best results after SA optimization were obtained by 75 percent, and that means that 25 percent of the closest customers from the depot are excluded when choosing the first customer. By increasing and decreasing this parameter, worse solutions are obtained.

Number of the nearest neighbor customers from previous served customer

This parameter varied from 3 to 7 depending on the size of the instance and the best solutions were obtained for them.

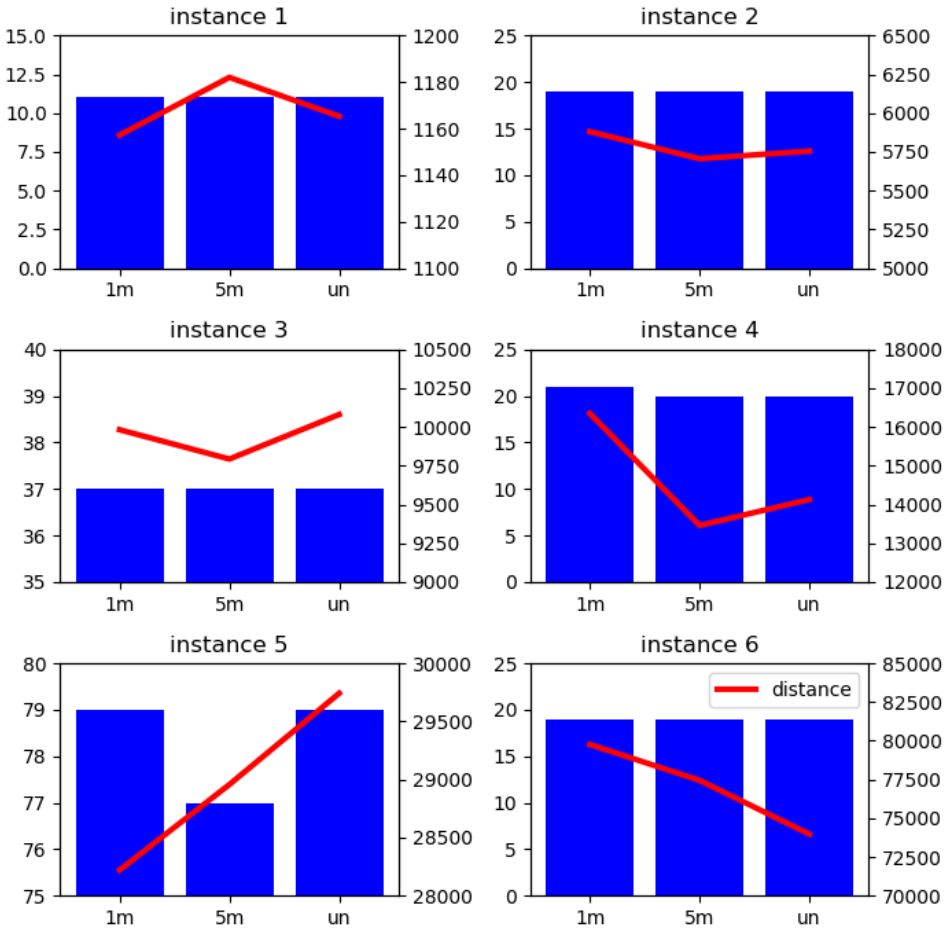
Initial temperature in SA

The initial temperature was varied from 25 to 100 depending on the number of customers. For a larger number of customers, a higher temperature was used to allow with higher probability worse solutions and therefore avoid getting stuck in the local optimum. Also, with more customers, there are more improvements to the current solution so worse solutions should be allowed with more probability to maintain balance.

Beta parameter for temperature decrease in SA

The beta parameter varied from 0.001 to 0.00025 and was smaller for instances with a larger number of customers. The goal was to achieve a slower temperature reduction for larger instances and to maintain a higher temperature for larger instances and thus the likelihood of choosing a worse solution.

Time limit parameter



For almost all instances and for all time limits the same number of vehicles is obtained with exception of instance 5 where the best solution is obtained in 5 minutes.

The correlation between the time limit and the number of kilometers is not uniform in all instances.

Conclusion

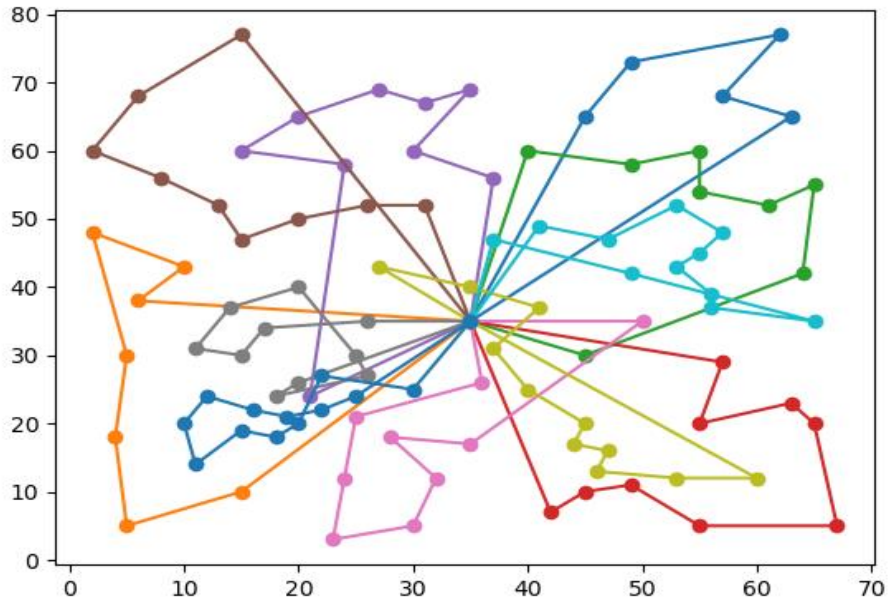
SA using inter and intra operators and fitness function managed to greatly improve the initial greedy solution in number of used vehicles.

Potential further improvements:

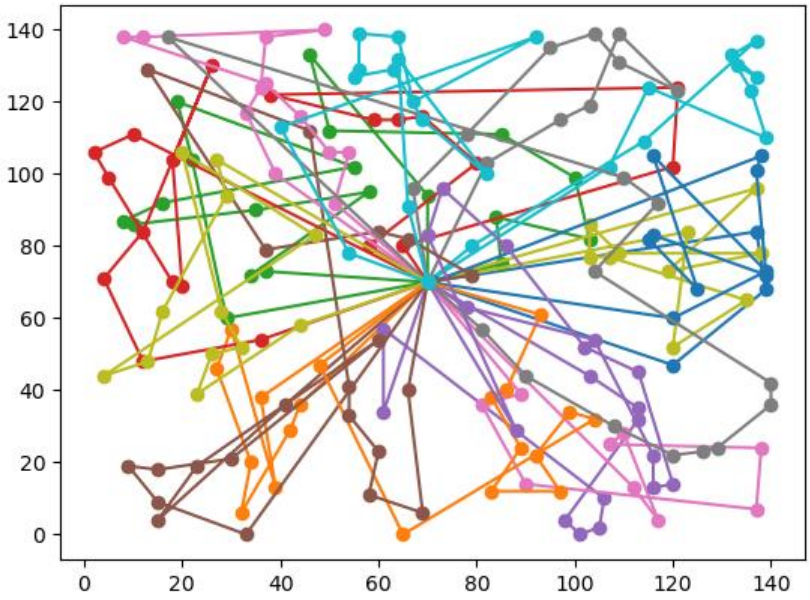
- More fitness functions and their combination could better guide the solution to better solution space, e.g. fitness function that will consider the number of vehicles and the number of customers not only in the shortest route but in some number of the shortest routes, because sometimes it is not optimal to break the shortest route to the end.
- Implementation of more inter and intra operators like 2 opt inter/intra swap, exchange operator.

Appendix

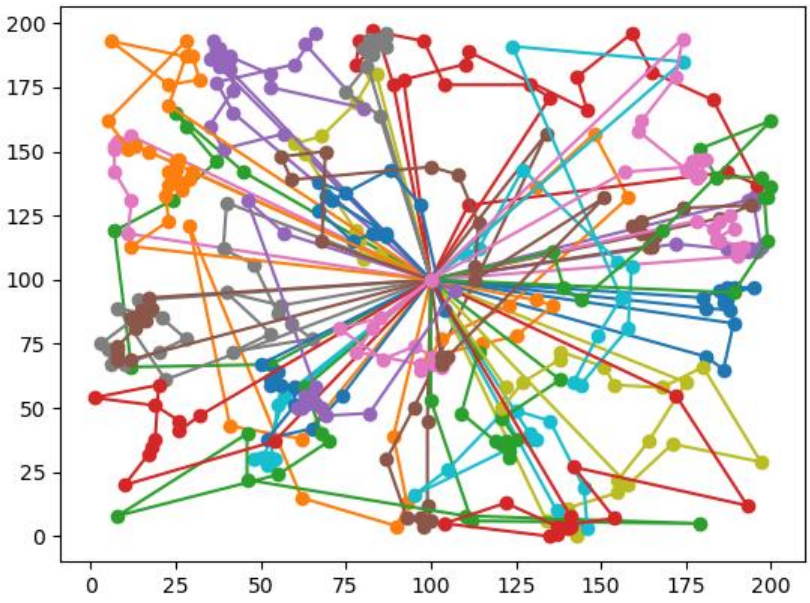
Best obtained solutions per instances



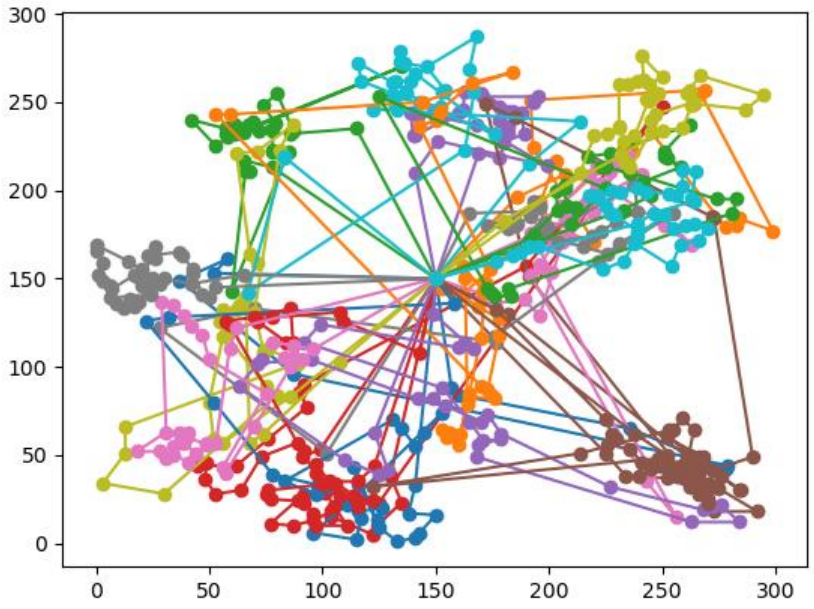
Picture: Best obtained solution for instance 1.



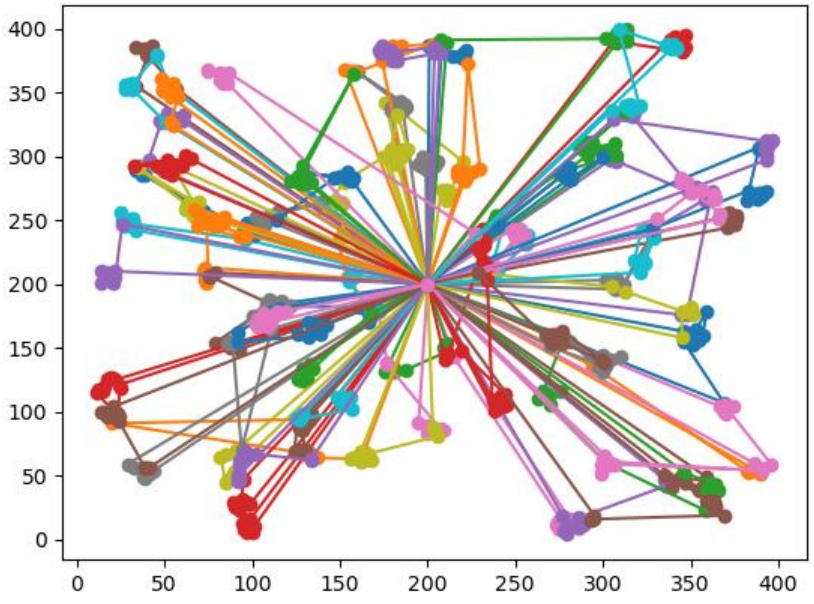
Picture: Best obtained solution for instance 2.



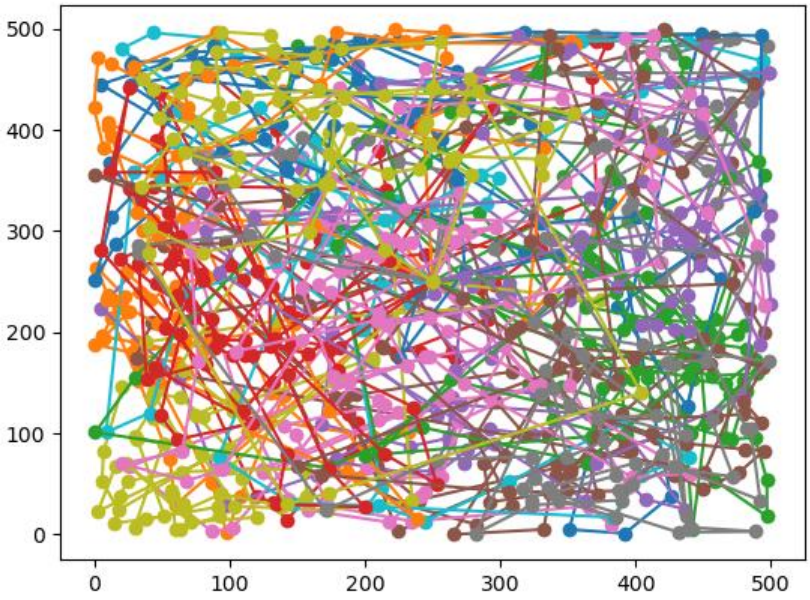
Picture: Best obtained solution for instance 3.



Picture: Best obtained solution for instance 4.



Picture: Best obtained solution for instance 5.



Picture: Best obtained solution for instance 6.