A variable neighborhood search based matheuristic for a Waste Cooking Oil collection network design problem

Omer Berk Olmez, Ceren Gultekin, Burcu Balcik, Ali Ekici, Okan Örsan Özener

INTRODUCTION

- **Improper disposal** of **Waste Cooking Oil** can cause **ecological contamination** and **damage** in the **sewage** system.
- **WCO** is mainly **produced** by **households**, but only **5.6%** of it is **recycled**, while 83.8% of oil produced by businesses is recycled.
- A **strategy** to **collect WCO** produced by households is necessary.

INTRODUCTION

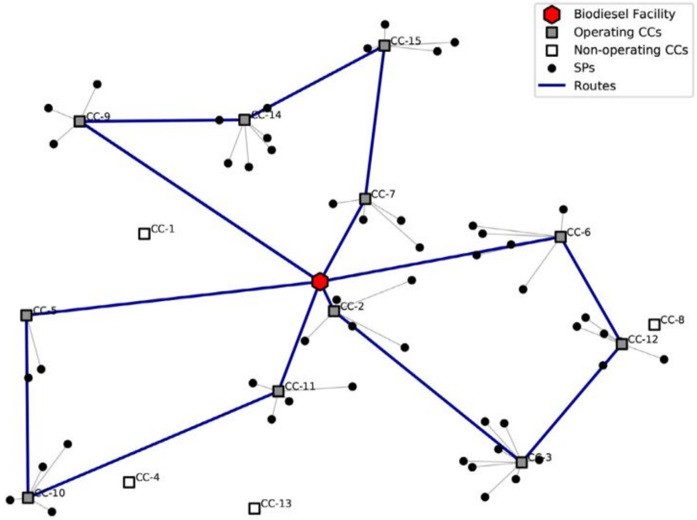
- The **aim** of the paper is to **design** an **efficient network** to **increase** the amount of **WCO collected** to then be recycled by a biodiesel facility.
- **Households** are **assigned** a **Collection Center** where to **deposit** the **WCO into bins**.
- A **fleet of vehicles weekly collects and replace** the **bins** at the CCs.

PROBLEM DEFINITION

The **models** proposed, starting from a set of candidate CCs, **optimize**:

- The **location** of **CCs**.
- The **number** of **bins** to place **at each CC**.
- The **assignment** of **households** to **CCs**.
- The **route collection** of **vehicles**.

while minimizing the **cost of opening CCs** and **placing bins at the CCs**, and **transportation cost** of **collection vehicles**.



PROBLEM DEFINITION: constants

- **S** is the set of s Source Points, each producing d_s litres of oil.
- **P** is the set of p candidate Collection Centers, each with a capacity of C_i bins; bins all have a capacity of **B** litres. f_i is the cost of opening CC i and **b** is the bin cost.
- P^s is the set of CCs close enough to SP s and P_0 is the set P plus the facility at the end.

PROBLEM DEFINITION: constants

- The distance between a pair of nodes i,j is $c_{i,j}$ and the travel cost per unit distance is α.
- The **WCO** collected at CCs are **picked up** by **collection vehicles** and **transported to** the biodiesel **facility**.
- V is the set of v homogenous capacitated vehicles, each vehicle has capacity **Q**.
- Each **vehicle tour starts** and **ends** at the biodiesel **facility**. Tours longer than a certain distance R are not allowed.

PROBLEM DEFINITION: variables

The decision variables are:

- **x_i** (binary) which is 1 if the CC i is opened.
- **z_si** (binary) which is 1 if the SP s is assigned to CC i.
- **c_ijk** (binary) which is 1 if vehicle k travels from location i to j.
- **t_ik** (integer) which is the number of bins picked up by vehicle k at CC i.
- **u_i** (integer) which is an auxiliary variable for subtour elimination.

PROBLEM DEFINITION: model

$$\min \quad \alpha \sum_{i \in \mathcal{P}_0} \sum_{j \in \mathcal{P}_0} \sum_{k \in \mathcal{V}} c_{ij} v_{ijk} + \sum_{i \in \mathcal{P}} f_i x_i + b \sum_{i \in \mathcal{P}} \sum_{k \in \mathcal{V}} t_{ik}$$

The **objective function minimizes** the **total cost** that incurs in each collection period, including the **vehicle routing cost**, the **cost** of **opening CCs** and the **bin cost.**

$$\sum_{s \in \mathcal{S} | i \in \mathcal{P}^s} d_s z_{si} \leq B \sum_{k \in \mathcal{V}} t_{ik} \qquad \forall i \in \mathcal{P}$$

Ensure that the total amount of WCO generated by the SPs assigned to a CC is not higher than the total capacity of the bins placed at that CC.

PROBLEM DEFINITION: model

$$\sum_{k \in \mathcal{V}} t_{ik} \leq C_i x_i \qquad \forall i \in \mathcal{P}$$

Limit the capacity of each CC in terms of the number of bins.

$$z_{si} \leq x_i \qquad \forall i \in \mathcal{P}^s, s \in \mathcal{S}$$

Force a CC to be opened if an SP is assigned to it.

$$\sum_{i \in \mathcal{P}^s} z_{si} = 1 \qquad \forall s \in \mathcal{S}$$

Assign SP to only one close CC.

$$\sum_{i \in \mathcal{P}} v_{0ik} \leq 1 \qquad \forall k \in \mathcal{V}$$

Each vehicle leave the facility once.

$$t_{ik} \leq Q \sum_{j \in \mathcal{P}_0 \setminus \{i\}} v_{jik} \qquad \forall i \in \mathcal{P}, k \in \mathcal{V}$$

A vehicle can pick up bins only if it visits the CC.

PROBLEM DEFINITION: model

$$\sum_{i \in \mathcal{P}} t_{ik} \leq Q \qquad \forall k \in \mathcal{V}$$

Number of bins picked up by a vehicle cannot be greater than vehicle capacity.

$$\sum_{j \in \mathcal{P}_0} v_{jik} = \sum_{j \in \mathcal{P}_0} v_{ijk} \qquad \forall i \in \mathcal{P}_0, k \in \mathcal{V}$$

Route continuity constraints.

$$\sum_{j \in \mathcal{P}_0 \setminus \{i\}} \sum_{k \in \mathcal{V}} v_{ijk} = x_i \qquad \forall i \in \mathcal{P}$$

$$\sum_{j \in \mathcal{P}_0 \setminus \{i\}} \sum_{k \in \mathcal{V}} v_{jik} = x_i \qquad \forall i \in \mathcal{P}$$

Vehicles only visits opened CCs.

$$u_i - u_j + P \sum_{k \in \mathcal{V}} v_{ijk} \leq P - 1 \qquad \forall i, j \in \mathcal{P}, i \neq j$$

Eliminates the sub-tours.

PROBLEM DEFINITION: model

$$\sum_{i \in \mathcal{P}_0} \sum_{j \in \mathcal{P}_0} c_{ij} v_{ijk} \le R \qquad \forall k \in \mathcal{V}$$

Limit the maximum route length of a vehicle.

$$x_i \in \{0, 1\} \qquad \forall i \in \mathcal{P}$$

$$z_{si} \in \{0, 1\} \qquad \forall s \in \mathcal{S}, i \in \mathcal{P}^s$$

$$v_{ijk} \in \{0, 1\} \qquad \forall i, j \in \mathcal{P}_0, k \in \mathcal{V}$$

$$t_{ik} \ge 0 \quad integer \qquad \forall i \in \mathcal{P}, k \in \mathcal{V}$$

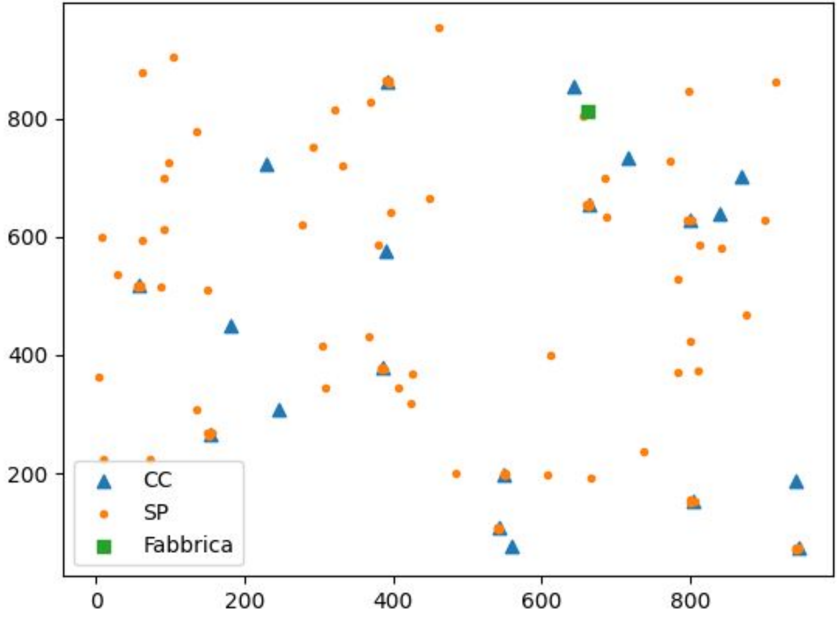
$$u_i \ge 0 \qquad \forall i \in \mathcal{P}$$

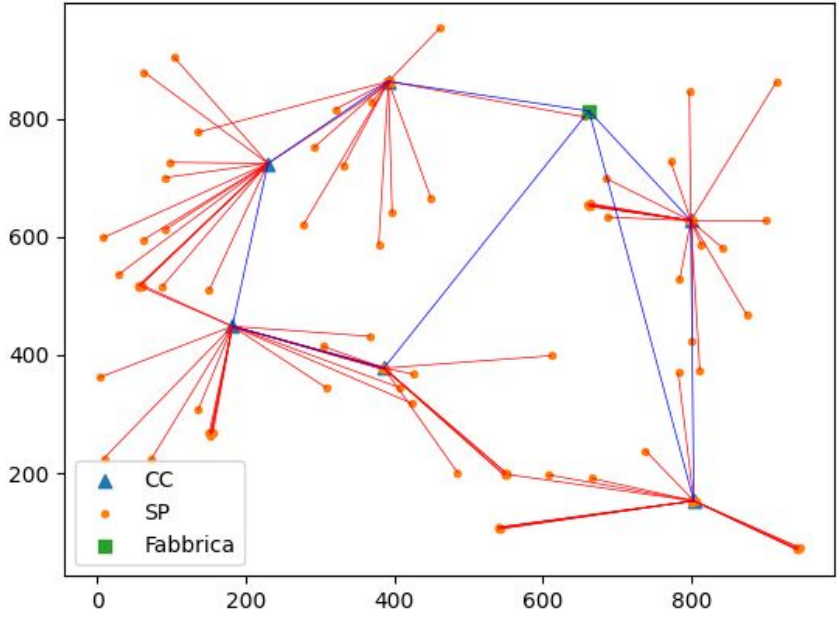
Domain of the decision variables.

DATA GENERATION

- Test **instances** are **randomly generated**.
- The position of the **CCs**, **SPs** and of the **facility** are **uniformly generated** in a **1000x1000 map**.
- To represent **dense areas**, clusters of SPs, **centered around** a **CC** are created. **Half** of the **nodes belong** to **clusters.**
- The **cost to open** a **CC**, their **capacity** and the **WCO produced by** an **SP** are all also **uniformly generated**.

GUROBI: solution example





n_CC=20 and n_SP=100

GUROBI: dimensional analysis

- **Start** from **20 CCs** and **100 SPs** and gradually **increase**, with steps of 20 CCs and 100 SPs, **up to 100 CCs** and **500 SPs**.
- **5 instances** are **generated** and **solved** for each couple.
- Run on Ubuntu 22.04.3 LTS, CPU AMD Ryzen 3 3250U 2.60 GHz and 8 GB RAM

GUROBI: dimensional analysis

n_CC	n_SP	Time[s]	Gap[%]
20	100	0	/
20	100	10800	15.0
20	100	732	0.0
20	100	0	/
20	100	782	0.0
40	200	10800	26.1
40	200	10800	35.3
40	200	10800	25.5
40	200	10800	38.3
40	200	10800	18.8
60	300	10800	35.2
60	300	10800	inf
60	300	10800	inf
60	300	10800	inf
60	300	10800	inf

DIMENSIONAL ANALYSIS

80	400	10800	29.6
80	400	10800	inf
80	400	10800	inf
80	400	10800	inf
80	400	10800	inf
100	500	10800	inf
100	500	10800	inf
100	500	10800	inf
100	500	10800	inf
100	500	10800	inf

DIMENSIONAL ANALYSIS

- At **higher n_CC** the **gurobi solver** struggles to find a solution.
- The **Vehicle Routing Problem** is **NP hard**.
- **Metaheuristic algorithms** are required to **optimize** the **problem.**
- An initial **sub-optimal solution** can be **constructed**, this **solution can** then **be used** to **implement** other algorithms such as the **Variable Neighborhood Search**.

VNS: constructive heuristic

Algorithm 1: Constructive Heuristic.

for $\omega \in \{0.0, 0.1, \dots, 1.0\}$ **do**
  for $n \in \{0, 1, \dots, n_{\max} = \lceil \frac{P}{2} \rceil\}$ **do**
    $\mathcal{S}' = \mathcal{S}, \widehat{\mathcal{P}} = \emptyset$
    $\widehat{\mathcal{P}} = \bigcup_{s \mid |\mathcal{P}^s| = 1} \mathcal{P}^s$
    $\mathcal{S}' = \mathcal{S}' \setminus \{s \mid |\mathcal{P}^s| = 1\}$
    while $\mathcal{S}' \neq \emptyset$ **do**
      Calculate A_i for each $i \in \mathcal{P} \setminus \widehat{\mathcal{P}}$
      $i^* = \operatorname{argmax}_{i \in \mathcal{P} \setminus \widehat{\mathcal{P}}} \{A_i\}$
      $\widehat{\mathcal{P}} = \widehat{\mathcal{P}} \bigcup \{i^*\}$
      Set the residual capacity and set of accessible SPs: $C' = BC_{i^*}, \mathcal{S}'' = \{s \in \mathcal{S}' \mid i^* \in \mathcal{P}^s\}$
      while $\mathcal{S}'' \neq \emptyset$ **do**
        $s^* = \operatorname{argmin}_{s \in \mathcal{S}''} \{c_{s,i^*}\}$
        $\mathcal{S}'' = \mathcal{S}'' \setminus \{s^*\}$
        if $C' \geq d_{s^*}$ **then**
          $C' = C' - d_{s^*}, \mathcal{S}' = \mathcal{S}' \setminus \{s^*\}$
        end if
      end while
    end while
    Solve AP-S (or AP-F) to select CCs, assign SPs and bins.
    Implement the modified savings algorithm to construct the collection routes.
  end for
end for

First find all SPs covered by a single CC.

Among the remaining CCs find the one with the max attractiveness, then find all its compatible SPs according to distance.

$$A_i = \frac{\min\{C_i, \sum_{s \in \mathcal{S}' \mid i \in \mathcal{P}^s} d_s\}}{\omega c_{i0} + (1-\omega)\frac{1}{n}\sum_{j=1}^{n} c_{i,i_j}}$$

Assign the SPs to the CC based on distance and residual capacity.
Once the CC capacity is full, or there are no more SPs, start again.

VNS: constructive heuristic

Algorithm 1: Constructive Heuristic.

```
for ω ∈ {0.0, 0.1, ..., 1.0} do
  for n ∈ {0, 1, ..., n_max = ⌈P/2⌉} do
    S' = S, P̂ = ∅
    P̂ = ⋃_{s | |P^s|=1} P^s
    S' = S' \ {s | |P^s| = 1}
    while S' ≠ ∅ do
      Calculate A_i for each i ∈ P \ P̂
      i* = argmax_{i∈P\P̂} {A_i}
      P̂ = P̂ ⋃ {i*}
      Set the residual capacity and set of accessible SPs:
      C' = BC_{i*}, S'' = {s ∈ S' | i* ∈ P^s}
      while S'' ≠ ∅ do
        s* = argmin_{s∈S''} {c_{s,i*}}
        S'' = S'' \ {s*}
        if C' ≥ d_{s*} then
          C' = C' − d_{s*}, S' = S' \ {s*}
        end if
      end while
    end while
    Solve AP-S (or AP-F) to select CCs, assign SPs and bins.
    Implement the modified savings algorithm to construct
    the collection routes.
  end for
end for
```

Iterate until all SPs are assigned.

Starting from the sets just found, solve an optimization problem to select CCs and assign SPs and bins. The problem is the same as before minus the vehicle parts. The AP-F, differently from the AP-S, forces all CCs found to be open.

A sub-optimal route is then created using a modified Clarke-Wright savings algorithm.

CLARKE-WRIGHT ALGORITHM

- **First consider** a **route** for **each CC**.
- **For each route pair** the **savings if the routes were combined** are **computed**.
- **Starting** with the **highest savings**, the **route pairs** are iteratively **merged**, only **if the resulting route does not** already **exist** and **if capacity** and **route length constraints** are **satisfied**.
- The **merging** of routes continue **until no further improvements** can be made **or until all** route **pairs** have been **considered**.

VNS

- Starting from the solution found with the Constructive Heuristic it is possible to **implement** a **VNS** which **uses four different operations**:
  1. insert a route segment in another route (**insert**).
  2. swap segments of different routes(**swap**).
  3. remove a segment of CCs from a route and add a number of unopened CCs   to the route (**insert & remove**).
  4. Remove a CC from a route (**Remove**).
- All operations, except Remove, can operate on **segments** of **length greater** than **one.**
- If the CCs are closed/opened The AP must be run again to assign the SPs to the CCs.

VNS

Algorithm 2: Variable Neighborhood Search based Matheuristic for the WCO-CNDP.

1: **input:** An initial solution β, and the set of neighborhood structures θ_k $(k = 1, \ldots, k_{\max})$
2: $t_{\max}$: maximum time that can be spent in each neighborhood
3: **repeat**
4: $\quad k = 1$
5: $\quad$ **while** $k \leq k_{\max}$ **do**
6: $\quad\quad t_{current} \leftarrow 0$
7: $\quad\quad \beta' \leftarrow Shaking(\theta_k(\beta))$
8: $\quad\quad$ **if** $\bar{f}(\beta') \leq (1+r)f(\beta)$ **then**
9: $\quad\quad\quad$ **if** AP-F infeasible **then**
10: $\quad\quad\quad\quad$ **if** $t_{current} \leq t_{\max}$ **then**
11: $\quad\quad\quad\quad\quad$ go to Step 7
12: $\quad\quad\quad\quad$ **else**
13: $\quad\quad\quad\quad\quad$ go to Step 23
14: $\quad\quad\quad\quad$ **end if**
15: $\quad\quad\quad$ **end if**
16: $\quad\quad\quad \beta'' \leftarrow LocalSearch(\beta')$
17: $\quad\quad\quad$ **if** $f(\beta'') \leq f(\beta)$ **then**
18: $\quad\quad\quad\quad \beta = \beta''$
19: $\quad\quad\quad\quad k = 1$
20: $\quad\quad\quad$ **else if** $t_{current} \leq t_{\max}$ **then**
21: $\quad\quad\quad\quad$ go to Step 7
22: $\quad\quad\quad$ **else**
23: $\quad\quad\quad\quad k = k + 1$
24: $\quad\quad\quad$ **end if**
25: $\quad\quad$ **else if** $t_{current} \leq t_{\max}$ **then**
26: $\quad\quad\quad$ go to Step 7
27: $\quad\quad$ **else**
28: $\quad\quad\quad k = k + 1$
29: $\quad\quad$ **end if**
30: $\quad$ **end while**
31: **until** Termination condition is met

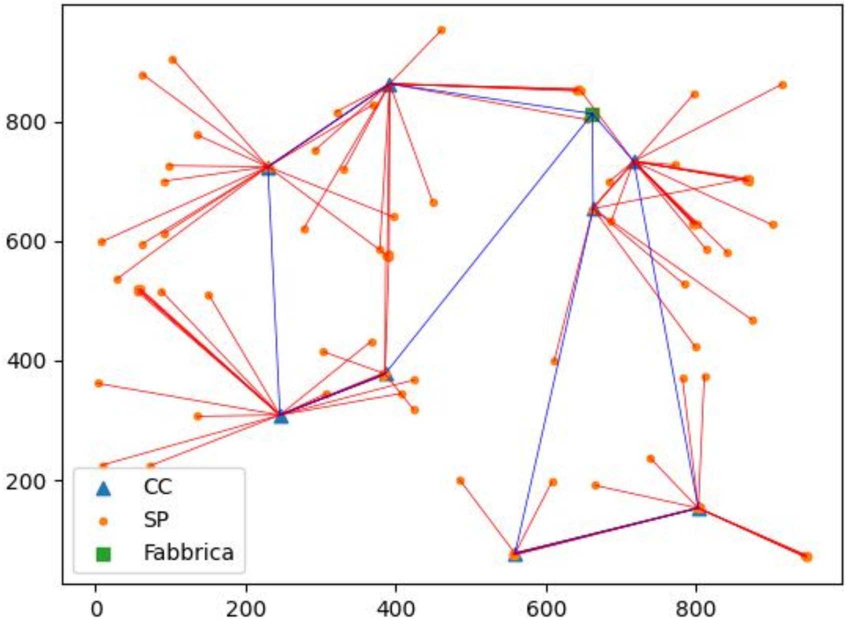
Need an initial solution and the set of operations as input.

Try to escape from the local minimum applying one of the operations.

Apply swap and insertion with segment length 1, if the new routes enhance the objective function, and if the constraints are satisfied, update the solution.

Termination condition is either time based or based on number of times it fails to find better solution or escape from the minimum

VNS: solution example



- n_CC=20 and n_SP=100

VNS: dimensional analysis

- Only the **AP-F** was **tested**.
- Instead of finding an initial solution for all **(n, omega)** pairs **only** a **pair** with **intermediate values** was used.
- **Start** from **20 CCs** and **100 SPs** and gradually **increase**, with steps of 20 CCs and 100 SPs, **up to 100 CCs** and **500 SPs**.
- Due to lack of time and resources only **one instance** is **generated** and **solved** for each couple.

VNS: dimensional analysis

n_CC	n_SP	CH time [s]	VNS time [s]	Number of routes
20	100	3.46	509.92	2
40	200	28.96	2016.65	2
60	300	109.42	9859.84	3
80	400	736.14	8190.73	4
100	500	638.57	4765.35	5

- Solution time decrease because the algorithm fails to find a better solution earlier.

VNS: Issues with my implementation

- The **Clarke-Wright algorithm** implementation is **not very efficient**, so for bigger cases creating **multiple initial solution** is **not viable**.
- For **bigger n_CCs** the **maximum route constraint** had to be **increased** to 3600, otherwise no solutions would be found, at the biggest case it had to be increased to 5000.
- The **stop conditions** are too **conservative**.
- Given this reasons, while a **solution** is **found** for **every dimensions**, **not** necessarily it is the **best one**.

CONCLUSIONS

- The VNS, differently from Gurobi, will always find a solution.
- In the paper it is shown that a better implementation of a **VNS** is **able** to **find** an **optimal solution** for **all couples of n_CC** and **n_SP**, within a **reasonable time-limit**, and that the **AP-S** is **quicker**, but the **AP-F** leads to **better solutions** since it expands the search space.
- It is also shown that it would be **better** (but slower) to **initialize** the **VNS** using **multiple starting solution**, found with CH ran with **different couples** of **(n, omega)**.