Optimizing Production Distribution from Multiple Sources

Bouchra Er-rabbany (bouchra@aims.ac.za)
African Institute for Mathematical Sciences (AIMS)

Supervised by: Dr. Surafel Luleseged Tilahun
University of Zululand, South Africa

14 May 2020

Submitted in partial fulfillment of a structured masters degree at AIMS South Africa

Abstract

Distribution of goods sometimes follows a familiar pattern, yet it still remains a challenge for most companies. This distribution can be modeled using Traveling Salesman Problem, TSP, but multiple sources product distribution is more complicated than single source. In this project, the problem will be formulated. Previous studies proposed many algorithms to solve this problem, so that firstly, an overview of existing algorithms is given with the classification of optimization problems and methods, then two different approaches will be proposed. The first approach requires determining which cities to assign to each manufacturer, as well as the optimal ordering of the cities in each cluster. The second approach is to work directly with multiple sources. This project compared the results for multiple sources product distribution, solved with developed Genetic Algorithm, swarm intelligence algorithm based on Prey-Predator Algorithm for the first approach and Nearest Neighbor Algorithm for the second one. In order to include some special cases like the size of customers and factories, connected and disconnected networks for different datasets was used.

Declaration

I, the undersigned, hereby declare that the work contained in this research project is my original work, and that any work done by others or by myself previously has been acknowledged and referenced accordingly.

[illegible]

Bouchra Er-rabbany, 14 May 2020

Contents

1. Introduction

Competition in the industrial world increases day by day. To have a large market share, it is necessary to combine between the parameters: price, quality and time. Distribution logistics is concerned with the flow of physical flows through the distribution network. Evolution of technology and the adoption of e-commerce makes the distribution very intensive because manufacturers want their products to be present in a maximum number of sale points in order to capitalize on profit generation. The ultimate hope is to achieve the greatest possible market coverage. This strategy however involves high distribution costs.

1.1 Objectives

In this work, we are interested in the problem of transport within a distribution network, which remains a very complex problem. The aforesaid complexity, in most cases, compares to the organization of industrial manufacturing itself. Many decisions must be taken, incorporating constraints of a varied nature. These constraints arise naturally due to factors such as the characteristics of the product and the quantities involved, and their destinations. Choosing a means of transport that is not only suitable but also one that maximises the profit margins thus remains central to any organization.

The main difficulty faced by a decision-maker, in the presence of an optimization problem, is the choice of an efficient method capable of producing an optimal solution in a time of reasonable calculation. In this context, the objective of our work is to study the optimization of distribution costs in the logistics chain through the minimization of transport costs in the distribution chain, and to study application problems of formal scientific methods. One of those problems is to distribute goods from multiple sources to different customers within different locations.

In order to model this problem, the multiple travelling salesman problem (mTSP) can be used. The mTSP, mathematically formulated by W.R. Hamilton and T. Kirkman (1800), is a key component in distribution and logistics management. However, it is more complicated than TSP since it requires determining which cities to assign to each salesman, as well as the optimal ordering of each city.

Although there are different case-based methods for the production phase, the distribution of goods sometimes follows a familiar pattern. Our aim in this project is to propose a new mathematical model that minimizes the cost of transport in the network while guaranteeing the satisfaction of customer orders, respecting the constraints of the capacity of deposits and the customer demands.

This research project is part of the optimization problems applied in transport and logistics, more precisely the vehicle routing problems VRP, It first appeared in a paper by Dantzig and Ramser (1959), they consist in determining the routes of a set of trucks, with a limited capacity, in order to visit a group of customers. A common assumption is to assume that a customer should only be visited by one vehicle and only once. The goal is to minimize the cost of transportation in terms of distance or time. One way to model them is by using a graph. The representation $G = (V, E, C)$ is indeed perfectly used to design any network. Where, it is necessary to define some elements.

- V is a set of vertices which represents cities, correspond to a site location (factory, depot, platform,...).
- E is a set of pairs of vertices called edge, it symbolizes the passage from one city to another.
- C represent a weight associated to each edge, which can a cost, or even a distance.

In this case we are interested in undirected weighted connected network $G = (V, E, C)$, where none of the edges have any direction; if there is an edge from A to B, it necessarily means that there's an edge from B to A. Weights (numerical values) are assigned to the edges in a network to signify the relative importance of edges.

In vehicle routing problems we mainly have: TSP or Traveling Salesman Problem and VRP or Vehicle Routing Problems.

- Vehicle Routing Problems (VRP)

  Problem which aims to determine the optimal set of routes for a fleet of vehicles to traverse in order to deliver to a given set of customers.

- Traveling Salesman Problem (TSP)

  Consist of finding in graphs a cycle passing through all the vertices only once (such a cycle is said to be "Hamiltonian") and which is of minimum length. Although TSP is the simplest node routing problem, it is already NP-hard. For solving this problem requests the intervention of algorithms with "nondeterministic polynomial time".

1.2 Layout of the thesis

The organization of this project is as follows :

In order to have a better understanding of this project, chapter 2 gives an overview of existing methods and their classification, Then, gives the details of some of them. Chapter 3, forms the main chapter of this project, since the contribution is mentioned on it. Firstly, a special formulation of the problem will be proposed to form multi-source model. In order to solve multi-source product distribution, two different approaches will be discussed, one consist of studying directly multi-source product distribution, while the other aims to simplify the problem by transforming multiple sources to single source. Then, In order to have good results, at the end, many algorithms will be used, where some of them are developed on this project. Finally, in chapter 4, we test the model and all the algorithms created, using different datasets, discuss and interpret the results obtained.

2. Existing Methods

Vehicle routing problems are very common in logistics. There are multitude of variants, the presence of practical cases, always makes the problem more complicated, a motivating phenomena for most reseach work, this project is focused on the study of Capacitated Vehicle Routing Problem (CVRP) in more general case of Traveling Salesman Problem (TSP).

In order to have an overview about the different optimization problems and their methods used to solve them. The following diagram summarizes the classification of the common problems and methods used in literature.

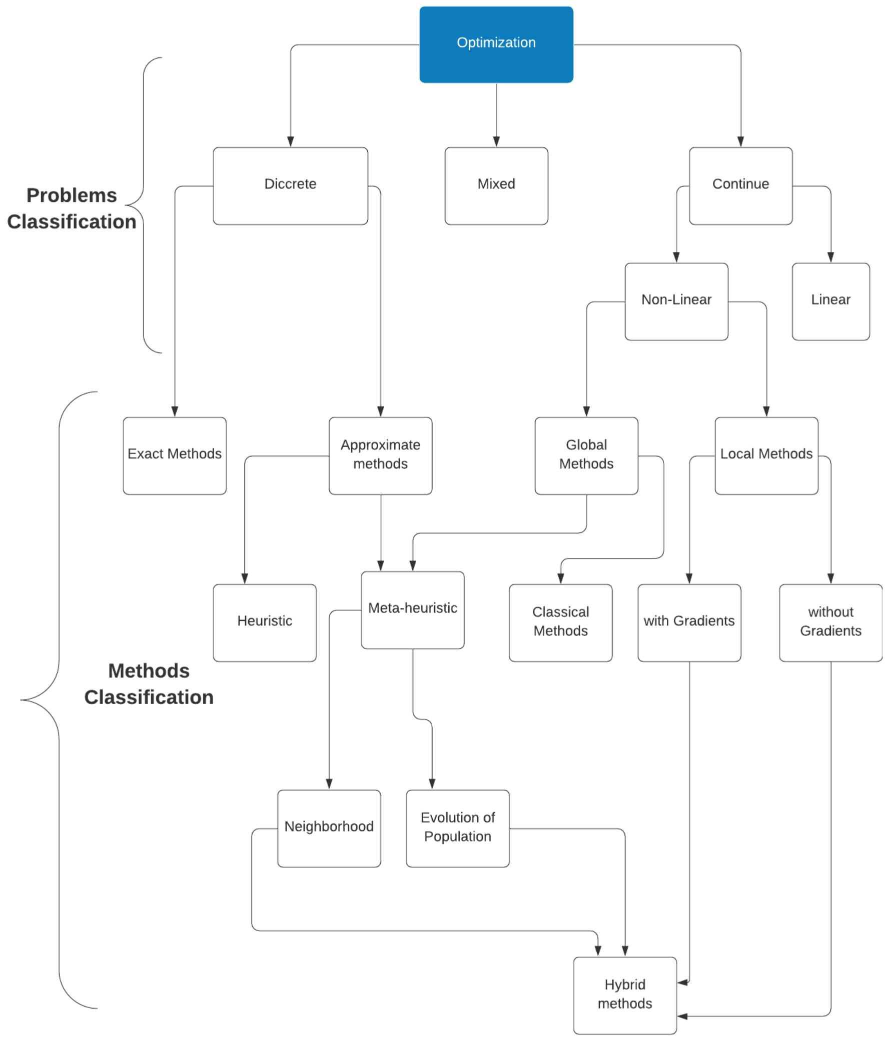

Figure 2.1: Classification of optimization problems and methods.

In this project, we will focus on combinatorial optimization, which consists of finding a better choice among an often very large finite set, by designing efficient algorithms, exact or approximate. It determines the optimum of a function under the given constraints. One common problem of combinatorial optimization is the traveling salesman problem (TSP).

It consists of choosing from a set of alternatives which satisfy a certain property, that which optimizes (minimizes or maximizes) such a cost function.

Denote by (f,ω), where:

- ω : all the alternatives
- f : $\omega \to \mathbb{R}$ a cost function

the aim is finding s^* in ω such that $f(s) \geqslant f(s^*)$ for all s in ω, where ω is finite or countable set of potential solutions.

Solving combinatorial problems is quite difficult, numerous resolution methods have been developed to solve this problem, and can be classified into two categories.

Exact resolution methods make it possible to obtain a solution whose optimallity is guaranteed, but when the number of possible combinations becomes exponential compared to the size of the problem, then the computation time becomes critical.

Approximate methods seek good quality solutions, without guaranteeing optimallity, but with a profile of reduced computation time. Among these methods, there are metaheuristic methods .

Metaheuristic is a strategy that allows to guide research towards an acceptable solution within a reasonable running time. The aim is to explore the search space effectively in order to determine solutions (almost) optimal. Metaheuristics are generally non-deterministic and give no guarantee of optimallity.

Based on the number of solutions (population size) and their interactions, metaheuristic algorithms can be grouped into two classes:

a Trajectory method

   It manipulates only one solution, and iteratively tries to improve this solution. They build a trajectory in the solution space, with an aim of moving towards an optimal solution.

   (a) Local search
   (b) Simulated annealing Kirkpatrick et al. (1983)
   (c) Tabu search Glover (1986)
   (d) Variable neighborhood search VNS Mladenović and Hansen (1997)

b Population based

   (a) Ant colony algorithm
   (b) Genetic algorithm John Holland (1960), **?**
   (c) Prey Predator algorithm

2.1 Dynamic programming

Dynamic programming was developed by Dreyfus (2002). Instead of solving one difficult problem, it solved a collection of overlapping sub-problems where any sub-problem refers to a partial solution.

Therefore we need to define the subproblems:

Let S $\subset \{1, ..., n\}$ and the vertex 1 and i in S, let $C(S, i)$ be the length of the shortest path that starts from 1 and ends at i and visit all vertices from S exactly once.

Assume that:

$$\begin{cases} C(\{1\}, 1) & = 0 \\ C(S, 1) & = \infty \quad when \quad |S| > 1. \end{cases}$$

We need to find the recurrence relation:

Consider the second to last vertex j on the required shortest path from 1 to i visiting all vertices in S.

The subpath from 1 to j is the shortest one visiting all vertices from $S \setminus \{i\}$ exactly once. Hence, $C(S, i) = \min\{C(S - i, j) + d_{ij}\}$, where the minimum is over all j∈ S, such that j $\neq$ i.

Algorithm:

```
Input  : Set of all vertex (cities)
Output : Sequence of cities with the shortest distance
for S from 2 to n do
    for $1 \in S \subset \{1, ..., n\}$ of size S do
        C(S,1) $\leftarrow \infty$;
        for $i \in S \quad and \quad i \neq 1$ do
            for $j \in S \quad and \quad j \neq 1$ do
                C(S,i) $\leftarrow \min\{C(S - i, j) + d_{ij}\}$;
            end
        end
    end
end
return $(min\{C(\{1, ..., n\}, i) + d_{ij}\})$
```

Algorithm 1: Dynamic programming algorithm

2.2 Local search

In computer science, local search is a heuristic method for solving computationally hard optimization problems. A local search algorithm takes the neighbor relation, and working according to the following high level scheme. At all times, it maintains a current solution. Throughout the execution of the algorithms, it remembers the minimum cost solution that it has seen so far.

A search strategy which searches the most promising branch of the state space first can:

- Find a solution more quickly.
- use very little memory
- Find solution even when there is limited time available.
- Often find a better solution, since more profitable part of the state space can be examined, while ignoring the unprofitable parts.

Local search algorithms move from solution to solution in the space of candidate solutions (the search space) by applying local changes, until a solution deemed optimal is found or a time bound is elapsed.

```
**Input**  : $S^{'}$ in neighborhood of S
**Output** : local optimal solution
S ← $S_{initial}$;
while $S^{'}$ better than $S$ do
    S ← $S^{'}$;
end
return ($S$)
```

Algorithm 2: Local search algorithm

Denote by $N(S, r)$ the neighborhood with center S and radius r: all cycles with distance at most r from S. When the neighborhood is larger (r is big), the resulting solution will be better and the running time will be higher.

2.3 Tabu search

Tabu search, created by Glover (1986) and formalized in 1989, is a local search algorithm. The idea of tabu search consists of starting from a given position, in exploring its neighborhood, then, choosing a position in this neighborhood which minimizes or mazimize the objective function. Using the concept of memory in the strategy to explore the search space, which guarantees the prohibition to resume recently visited solutions. In each iteration 'the least bad neighbor' is chosen.

- Types of memory

  Short-term: The list of solutions recently considered. If a potential solution appears on the tabu list, after a certain number of iterations it reaches the expiration point.

  Intermediate-term: Intensification rules intended to bias the search towards promising areas of the search space.

  Long-term: Rules that promote diversity in the search process; it can explore regions of the solutions space that have not been considered before.

- Size of taboo list:

  Size L to be determined empirically, Neither too long nor too small, Static/dynamic rules

- Selection of the best neighbor:

$$\begin{cases} \text{Best Fit:} & \text{whole neighborhood is explored} \\ \text{First Fit:} & \text{part of the neighborhood is explored} \end{cases}$$

Algorithm:

```
**Input**  : initial solution $S^* = S_0, C^* = f(S_0), S_c = S_0$ , $TabuList = \emptyset$
**Output** : Best solution
$S_n \in N(S_0)$ such that ∀ S ∈ $N(S_c)$,$f(S_n) \leqslant f(S)$ and $S_n \neq TabuList$;
if $f(S_n) \leqslant C^*$ then
    $S^* = S_n$,$C^* = f(S_n)$;
    Update Tabu List
end
return ($S$)
```

Algorithm 3: Tabu search algorithm

2.4 Variable neighborhood descent (VND)

Variable neighbourhood descent (VND), introduced by Mladenović and Hansen (1997), adopts a change of neighbourhoods in a deterministic way. VND algorithms assumes an initial condition, say x, and utilize local search heuristics on a set of neighbourhoods to attain an optimum solution. Since the aforementioned local search methods use few neighbourhoods, VND increases the likelihood of achieving a global optimum solution as opposed to when single neighbourhood structures are used.

```
Input  : Initial solution $x^{'}$, $k_{max}$ neighborhood structures function f to evaluate solutions
Output : a potentially improved $x^{'}$
k ← 1;
while $k \leqslant k_{max}$ do
    Find the best neighbor $x^{"} \in N_k(x^{'})$ ;
    $x^{"} \leftarrow$ LocalSearch($x^{'}$);
    if $f(x^{"}) > f(x^{'})$ then
        $x^{'} \leftarrow x^{"}$ ;
    end
    else
        $k \leftarrow k + 1$
    end
end
```

Algorithm 4: Variable neighborhood descent algorithm used as LocalSearch of VNS.

2.5 Ant colony algorithm

The ant colony optimization algorithm (ACO) proposed by Dorigo and Gambardella (1997), remains instrumental when good paths are to be obtained through graphs. From a computer science and operational research perspective, ACO is a probabilistic procedure for solving problems, often computational. Optimization tasks which involves graphs, such as vehicle routine tasks are best solved by incorporating local search algorithms and Artificial Ants; multi-agent methods influenced by behaviors of real ants. The notion of Artificial Ants remains an active and robust research area with their potential commercial applications conjured by specialized companies with each passing day.

Mathematical Model:

In order to represent the pheromone level on a graph a mathematical model is used.

Define variables:

α and β parameters control the exploitation and exploration behaviour of the ants by setting the attractiveness of pheromone deposits or the 'shortness' of the path.

τij : amount of pheromone deposited on edge i $\rightarrow$ j.

$\Delta\tau_{ij}^{k}$:Amont of phermon deposit on the edge i $\rightarrow$ j by the kth ant.

L_k : cost of the k^{th} ant's tour (typically length).

ρ : pheromone evaporation coefficient.

Amount of pheromone deposited by kth ant, typically given for a Traveling Salesman Problem TSP (with moves corresponding to arcs of the graph) by:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{L_k} & \text{kth ant travel on the edge i } \rightarrow j \\ 0 & \text{otherwise.} \end{cases}$$

- Pheromone update

  Trails are usually updated when all ants have completed their solution, increasing or decreasing the level of trails corresponding to moves that were part of "good" or "bad" solutions, respectively.

- Amount of phermon without vaporation

$$\tau_{ij} = \textstyle\sum_{k=1}^{m} \Delta\tau_{ij}^{k}.$$

- Amount of phermon with vaporation

$$\tau_{ij} = (1-\rho)\tau_{ij} + \textstyle\sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad \text{where} \quad 0 < \rho < 1.$$

- Edge selection

  Each ant needs to construct a solution to move through the graph. to select the next edge in its tour, an ant will consider the length of each edge available from its current position, as well as the corresponding pheromone level.

$$P_{ij} = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum(\tau_{ij}^{\alpha}\eta_{ij}^{\beta})}.$$

  where $\eta_{ij} = \frac{1}{L_{ij}}$.

2.6 Genetics algorithm

A genetic algorithm is a search heuristic that is inspired by Charles Darwin's theory of natural evolution, used for optimization purposes according to survival of the fittest idea. Evolve a population of randomly generated algorithms so that successive populations improve their ability to achieve a goal. They generate high-quality solutions usually in time. The genetic algorithm depends on selection criteria, crossover, and mutation operators, To solve the traveling salesman problem using genetic algorithms, there are different tour representations such as in the form of binary, path, adjacency and matrix representations. The genetic algorithms are useful for NP-hard problems, especially the traveling salesman problem, In order to minimize total cost, we will propose a new crossover to solve traveling salesman problem. This approach uses the path representation, which is the most natural way to represent a legal tour. Some traditional representation methods like partially mapped and order crossovers along with new cycle crossover operator are reported.

The algorithm starts from a population of randomly generated individuals, and is an iterative process, with the population in each iteration called a generation. They produce offspring which inherit the characteristics of the parents and will be added to the next generation. If parents have better fitness, their offspring will be better than parents and have a better chance at surviving. In each generation, the fitness which is the value of the objective function of every individual in the population is evaluated. The population has a fixed size. As new generations are formed, individuals with the least fitness die, providing space for new offspring. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population.

Typical genetic algorithm requires:

- Genetic representation of the solution domain.

- Fitness function to evaluate the solution domain.

Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators.

1. Initial population

   The process start with population which is a set of individuals, Each individual is a candidate solution to the problem required to solve. To encode the genes in a chromosome there is to many representations of set of genes of an individual.

2. Selection

   In order to produce a new generation two pairs of individuals (parents) are selected from the existing population, the different selection strategy used in the GA process will affect differently the performance of the algorithm. Individual solutions are usually selected throught a fitness-based process , where fitter solutions are typicaly more likely to be selected .

3. Crossover

   The operator which define the child production process is called crossover, it is the most significant phase in a genetic algorithm needed to increase the average quality of the population. There exist different crossover operator like Single Point Crossover ,double Point Crossover ,Partially-mapped crossover and Cycle crossover.

4. Mutation

   Some new offspring formed, of their genes can be subjected to a mutation with a low random probability, which means that some of the bits in the bit string can be flipped. It is necessary to explore new states and helps the algorithm to avoid local optima.

   Genetic algorithms are not the only methods but are the mostly used to solve Traveling Salesman Problem.

2.6.1 Genetic representation.

In genetic Algorithms, there have been many different representations, Among them; binary, path, adjacency, ordinal, and matrix representations are the most important. We are limiting ourself only with the path representation which is most natural and legal way to represent a tour for the tours representation.

a Binary presentation

Binary representation of the n-cities TSP is encoded as a string of $[log_2(n)]$, an individual is a string of n$[log_2(n)]$, hence the solutions are represented in binary as strings of 0s and 1s. Example : for n= 6 ,six cities are represented by 3 bits :

i	city i
1	000
2	001
3	010
4	011
5	100
6	101

Following the binary representation encode the tour (1-2-3-4-5-6) .

$$(000,001,010,011,100,101)$$

b Path presentation

The most natural way to present a legal tour is probably, path representation. A tour is represented as a list of n cities, for example the tour 6-2-4-8-7-1-5-3 is simply represented by:

$$(6\ 2\ 4\ 8\ 7\ 1\ 5\ 3)$$

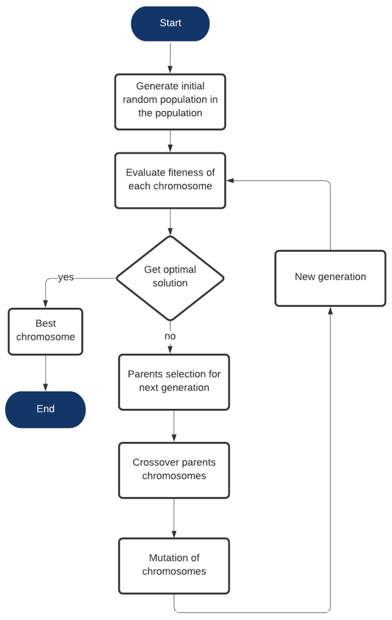
Start
Generate initial random population in the population
Evaluate fiteness of each chromosome
Get optimal solution
yes
no
Best chromosome
End
Parents selection for next generation
Crossover parents chromosomes
Mutation of chromosomes
New generation

Figure 2.2: logigramme Procedure of Genetic Algorithm GA

Partially-mapped crossover :

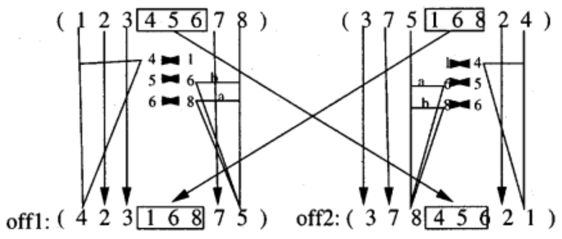

Figure 2.3: Partially-mapped crossover PMX, Larranaga et al. (1999)

The algorithm of PMX is given below.

Input : two pairs of solutions (parents)
Output : two pairs of produced solutions(new generation)
1.Substring selection: Cut each parent into two substrings, and then select one substring for each parent at random.
2.Substring exchange: Exchange the two selected substrings to produce proto-offspring.
3.Mapping list determination: Determine the mapping relationship based on the selected substrings.
4.Offspring legalization: Legalize proto-offspring with the mapping relationship.

Algorithm 5: Partially mapped crossover (PMX) Algorithm, Ting et al. (2010)

- Substring selection

  Consider the two pairs of solution (parents)

Parent1	1	2	3	4	5	6	7	8

Parent2	3	7	5	1	6	8	2	4

- Substring exchange

  To create an offspring by using this operator, it selects randomly two cut points along the parent tours. For example lets select the first cut point between the third and the fourth string element, and the second one between the sixth and seventh string element. Hence, we get:

Offspring1	1	2	3	4	5	6	7	8
Offspring2	3	7	5	1	6	8	2	4

  The mapping sections is defined as the substrings between the cut points, on this step of algorithm the parents exchange mapping section, then we get the following results.

Offspring1	1	2	3	1	6	8	7	8
Offspring2	3	7	5	4	5	6	2	4

- Mapping list determination Some of strings in the Offstring is repeted therefore we use the following mapping to avoid duplicate strings . For example, the first element of Offspring1 is 1 which is already on the mapping section , so we exchange 1 with 4 .

Offspring1	Offspring2
4↔1	1↔4
5↔6	6↔5
6↔8	8↔6

  After useing the mapping we get the final results as new generation in the population .

Offspring1	4	2	3	1	6	8	7	5

Offspring2	3	7	8	4	5	6	2	1

Cycle crossover (CX) :

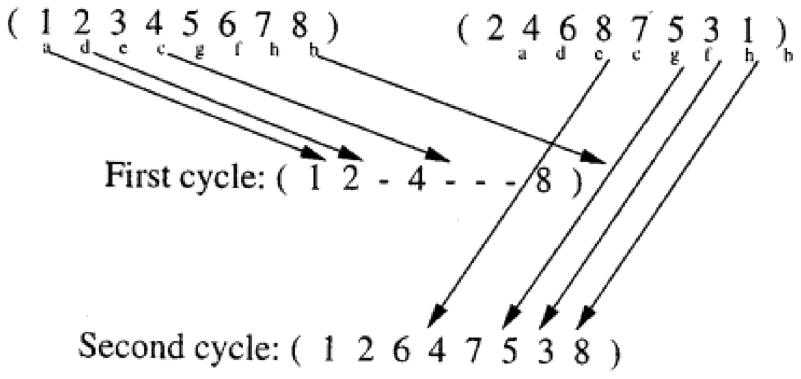

Figure 2.4: Cycle crossover (CX),Larranaga et al. (1999)

Consider the two pairs of tours (parents).

Parent1	1	2	3	4	5	6	7	8

Parent2	2	4	6	8	7	5	3	1

Parent2	1	2	6	4	7	5	3	8

2.7 Prey predator algorithm

Prey-predator algorithm is a swarm-based algorithm, that was proposed by Tilahun and Ong (2015), it is an evolutionary algorithm inspired by biological phenomenon. This algorithm consists of first generating the initial population of individuals randomly, then evaluating the fitness of the solutions based on the objective function. The solutions will be grouped into three, namely the predator with the minimum performance of all the solution determined by this algorithm in each iteration, best prey with the best performance of all the solution determined by this algorithm in each iteration and ordinary prey which are the rest of the solutions. The predator move randomly with maximum step length, λ_{max} is a parameter. The best prey improve the performance by generating q directions and choose the one which will improve its performance. The ordinary prey takes into consideration two cases either to influenced by the best prey or the predator. The main algorithm is described in the following, introduced by Tilahun and Ong (2015) for single source, developed on this project to be adapted for multiple source problem :

```
Input  : $P_{c1}, P_{c2}, Pf, q, f(x), S$
Output : Best solution $x_i$
-Random generated N feasible solution say $(x_1, ..., x_N)$with $x_i$ in S;
- Evaluate the solutions based on the objective function f(x),sorted the solutions;
$y_1 \leftarrow x_p$ ;
$y_N \leftarrow x_b$ ;
for $i = 2 : N - 1$ do
    $y_i^{'} \leftarrow x_i^{'}$ ;
end
update$y_1 \leftarrow$ update-predator$(x_p)$ ;
update$y_N \leftarrow$ update-best-prey$(x_b)$ ;
for $i = 2 : N - 1$ do
    update$y_i \leftarrow$ update-ordinary -prey$(x_i)$ ;
end
- Replace x by y and empty y ;
- If termination criterion is not met go back to line 4 ;
return Best solution $x_i$
```

Algorithm 6: Prey-predator Algorithm for discrete problem, Tilahun and Ong (2015)

The standard prey-predator algorithm is designed for a continuous problem, so that it was modified to be used for the discrete problem. Hence, the following pseudo code will be used to update predator.

```
Input  : $x_p, P_{c1}, f(x)$
Output : updated $x_p$
for $i = 1 : n$ do
    generate randomly rand ;
    if $rand < P_{c1}$ then
        change $x_p$ randomly ;
        adjust-feasibility ;
    end
    choose $x_p$ with least performance
end
return $x_p$
```

Algorithm 7: Update-predator Algorithm for discrete problem, Tilahun and Ong (2015)

The following pseudo code will be used to update the best prey. The best solution among the q directions will be taken as the updates solution of x_b

```
Input  : $x_b, P_{c2}, f(x)$
Output : updated $x_b$
for $i = 1 : n$ do
    generate randomly rand ;
    if $rand < P_{c2}$ then
        change $x_b$ randomly ;
        adjust-feasibility ;
    end
    choose $x_b$with best performance
end
return $x_b$
```

Algorithm 8: Update-Best-Prey Algorithm for discrete problem, Tilahun and Ong (2015)

```
Input   : $x_l, P_f, f(x)$
Output : updated $x_l$
for $i = 1 : n$ do
    generate randomly rand ;
    if $rand < P_f$ then
        for $j = i + 1 : N$ do
            $x_l(i) \leftarrow round(rand(x_j(i) - x_l(i)))$ ;
            adjust-feasibility ;
        end
    end
    else
        $x_l(i) \leftarrow round(rand(x_l(i) - x_1(i)))$ ;
        adjust-feasibility ;
    end
end
```

Algorithm 9: Update-Ordinary-Prey Algorithm for discrete problem, Tilahun and Ong (2015)

3. Multi-Source Product Distribution

3.1 Multi-source model (PDM)

We want to distribute a product made in factories located in n different cities. Each factory characterized by its unit production cost and its capacity, want to optimize production and distribution to meet the demand of customers, who are located in m different cities.

Model 1:

Assumptions:

Assume that we are in case of uni-product and the capacity of trucks is not a constraint.

C_j : Customer j ,for j$\in \{1, 2, \ldots, n\}$

M_i : Manufacure i ,for i $\in \{1, 2, \ldots, m\}$

T_i : Trucks i ,for i$\in \{1, 2, \ldots, t\}$

U_i : Number of places the trucks visits , present set of depots and customers, for i$\in \{1, 2, \ldots, t\}$

d_{ij} : Distance between two cities i and j

Matrix : CxM : X_{ij}

Decision variables:

$$Y_i = (y_{i,1}, y_{i,2}, ..., y_{i,n}, y_{i,n+1}, y_{i,n+2}, ...y_{i,n+m}) \in \{1, 2...n, n+1, ..., n+m\}$$

Where $y_i \in \{1, 2...n\}$ are customers that trucks T_i need to visit and $y_i \in \{n+1, ..., n+m\}$ are manufactures that trucks T_i need to visit.

Objective functions:

We need to optimize production and distribution to best satisfy the demand of customers. Therefore for each track T_i, we need to determine a sequence of centres (manufactures, customers), that is represented by the decision variables Y_i. Suppose that trucks T_i pass by u_i centers then we get the following sequence:

$$S_i = (y_{i,0} - y_{i,1} - y_{i,2}, ..., y_{i,u_i-1} - y_{i,0})$$

Where $y_{i,0} \in \{n+1, ..., n+m\}$ is the starting manufacture

We need to determine these sequenceS_i that minimise the cost or distance travel. Hence the objective function for each truck tour's is:

$$c_i = d(y_{i,0} - y_{i,1}) + \textstyle\sum_{j=1}^{u_i-2} d(y_{i,j} - y_{i,j+1}) + d(y_{i,u_i-1} - y_{i,0})$$

Therfore the objective function is:

$$min\ Z = \sum_{i=1}^{t} c_i \qquad (3.1.1)$$

Constraints:

- Set of customers and manufactures are a positive integers

$$y_{ij} \geqslant 0 \quad \forall i \in \{1,...,t\} \quad \forall j \in \{1...n+m\}$$

- Number of places visited by truck T_i is a positive integers

$$u_i \geqslant 0 \quad \forall i \in \{1,...,t\}$$

- Truck T_i can't visit a customer more than one time , but it can visit a manufacture more than one time

$$y_{i,j} \neq y_{i,k} \quad \forall j \neq k \; and \quad \forall i \in \{1,...,t\} \; and \quad \forall y_{i,j} \in \{1...n\}$$

3.2 Solving the multi-source PDM

The previous chapter gave an overview on many methods that can be used to solve multiple Traveling Salesman Problem mTSP, In this chapter, we will follow two approaches. The idea of the first approach is to transform multiple sources to single source, then we will use two methods, Genetic Algorithm (GA) and Nearest Neighbor Algorithm (NNA). In Genetic Algorithm we will use the new crossover operator. and the second approach based on using multiple sources directly. In order to solve this problem the Nearest Neighbor Algorithm (NNA) is used.

The following diagram clearly explains, the two approaches to follow:

This project considers multiple Traveling Salesman Problem mTSP as a symmetric problem. mTSP involves assigning m salesmen to n cities, and each city must be visited by a salesman while requiring a minimum total cost. where the number of customers is n and the number of manufactures is m and t is the number of trucks, such that $n > m$, $t > 0$ and $n > 0$ and $m > 0$ are integers.

The $n + m$ cities must be partitioned into t tours, with each tour resulting in a TSP for one salesman. The aim is to minimize the total cost (distance) for all salesman.

The requirements on the set of routes are:

- All of the routes must start and end at the (same) depot.
- Each city must be visited exactly once by only one salesman.

3.2.1 First Approach.

Clustering

In this project, mTSP is solved using two approaches. In order to partition the n+m cities into t tours, we use clustering technique, then, each cluster or tour is solved using GA as a TSP to find the shortest path for each salesman and thus to minimize the total distance for all salesmen.

The choice of clustering is justified by the concept of barycentre used in transport and distribution, barycentre is a calculation method which allows to know the center of gravity between several points.used in logistics and transport (during the geographical location of a factory or a new warehouse for example). This method determines the center of a network of points to serve whose coordinates are weighted by a traffic indicator which can be expressed in weight, volume, distance, number of lines of orders, turnover.

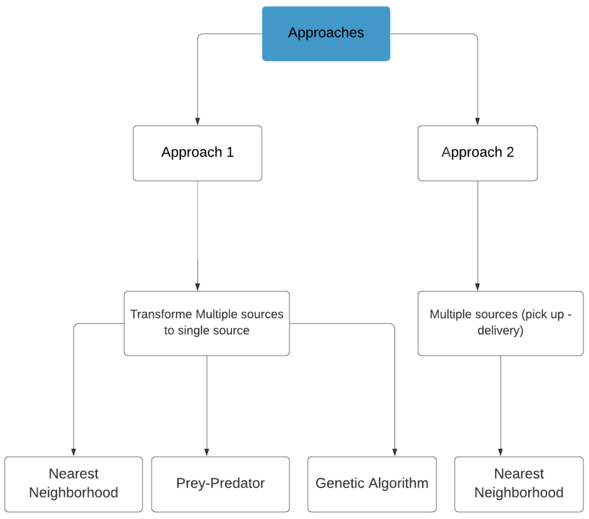

Figure 3.1: Aproaches methodology .

The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume t clusters) fixed a priori. Regarding the number of clusters c, it should be "less" or equal to the number of trucks, the reason why it should be "less" is that it could be a case of using less than available trucks for cost-effective delivery.

The following figure present an example of distribution of manufactures and customers, where A, B, C, D, E are the manufactures and x's are the customers.

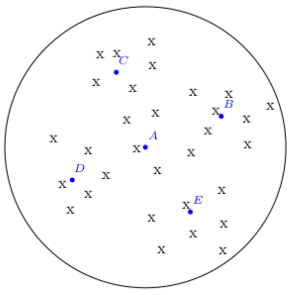

Figure 3.2: Manufactures and Customers distribution

proposed algorithm

```
Input  : $(M_1, M_2, M_3......M_m)$ set of cluster(Manufacturs), $(C_1, C_2, C_3......C_n)$ sets of
         customers need's,distance matrice
Output : the elements of each cluster
for each manufacture $M_i$ do
    manufacture-capacity $\leftarrow$ Capacity of manufacture $M_i$ ;
    Cluster list $\leftarrow [i]$ ;
    for each Customer $C_j$ do
        if manufacture-capacity $> 0$ then
            Customer-id ,distance-manufacture-customer $\leftarrow$ myArgmin(Matrice-distance[i]) ;
            if manufacture-capacity $>$ Customer-Need then
                - add customer-id to Cluster-list;
                - substract Customers-need from manufacture-capacity;
            end
        end
    end
    -add cluster to clusters list;
end
return Clusters
```

Algorithm 10: First Clustering Algorithm

developed algorithm

```
Input  : $(M_1, M_2, M_3......M_m)$ set of cluster(Manufacturs), $(C_1, C_2, C_3......C_n)$ascending sorted
         set of customers needs, distance matrice
Output : the elements of each cluster
for each Customer $C_i$ do
    manufacture-id $\leftarrow$ my-argmin(distance-cm[i]) ;
    manufacture-capacity $\leftarrow$ Capacity of manufacture $M_i$ ;
    if manufacture-capacity $>$ customer[i] then
        - add customer-id to Cluster-list of manufacture (manufacture-id );
        - update the current Capacity of manufacture (manufacture-id );
    end
    else
        - update the current Capacity of manufacture (manufacture-id );
    end
    for each Manufacture $M_j$ do
        manufacture-capacity $\leftarrow$ Capacity of manufacture $M_i$ ;
    end
end
return Clusters ,manufacture-capacity
```

Algorithm 11: Second Clustering Algorithm

```
Input   : Customer we want to insert $i$,saturated manufacture ,Clusters
Output : Customer inserted in its cluster
for each Customer $C_i$ do
    manufacture-id $\leftarrow$ my-argmin(distance-cm[i]) ;
    manufacture-capacity $\leftarrow$ Capacity of manufacture $M_i$ ;
    customer-need $\leftarrow$ Need of customer i ;
    while manufacture-capacity $>$ customer[i] do
        - add customer-id to Cluster-list of manufacture (manufacture-id );
        - update the current Capacity of manufacture (manufacture-id );
    end
    - update the current Capacity of manufacture (manufacture-id );
    - add customer i to Cluster-list of manufacture (manufacture-id );
end
return Clusters ,manufacture-capacity
```

Algorithm 12: Insert-Customer Algorithm

After applying clustering algorithm we get the following new distribution. Hence we transform mTSP to a TSP.

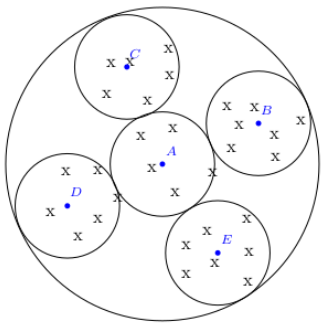

Figure 3.3: Clustering of Manufactures and Customers

Nearest Neighbor Algorithm NNA

After Clustering process we will apply the Nearest Neighbor Algorithm in each Cluster (TSP), Nearest Neighbor Algorithm NNA is a meta-heuristic method where the salesman starts at the manufacture (depot) and repeatedly visits the nearest city (customers) until all have been visited. The algorithm quickly yields a short tour.

These are the steps of the algorithm:

Input : set of Customers and manufacture's locations

Output: sequence of the visited vertices

i Initialize all vertices as unvisited.

ii Select the manufacture , set it as the current vertex u. Mark u as visited.

iii Find out the shortest edge connecting the current vertex u and an unvisited vertex v.

iv Set v as the current vertex u. Mark v as visited.

v If all the vertices in the domain are visited, then terminate. Else, go to step 3.

The nearest neighbour algorithm is easy to implement and executes quickly.

Genetic Algorithm GA

Proposed Crossover Operators

After Clustering process, we will apply Genetic Algorithm GA in each Cluster (TSP), by using the proposed crossover.

Since crossover is the main phase in genetic algorithm, a new crossover operator is proposed. From the existing crossover we notice that most of those operators are based on local search;the new generation always inherited part of their parents, which makes this search local. The idea is to browse the solution space in a global way. In order to do this we will exploit the idea of ideal riffle shuffle.

An ideal riffle shuffle is a riffle shuffle where the deck is cut right in the middle in two equal packets and the interleave exactly alternates the cards of both packets with the convention that the first card is always put in first position (on the top). For example with a deck of 10 cards:

Parent	1	2	3	4	5	6	7	8	9	10

Cut in the midle the dick

Parent	1	2	3	4	5	6	7	8	9	10

Apply the ideal riffle shuffle

Parent	1	6	2	7	3	8	4	9	5	10

The **fitness** which is the objective function in this case is the total cost (distance) in each cycle.

3.2.2 Lemma. Let N be the number of card in a deck When $N = 2^n$ after n ideal riffle shuffles the deck is necessarily back at its initial ordering.

Proof. Let $N = 2^n$, label the cards 0, 1, 2, . . . , N - 1

n binary digits are needed to code these N numbers: the card with label k is coded as $(i_{n-1}\ i_{n-2}...i_0)$ that is to say:

$k = i_{n-1}2^{n-1} + i_{n-2}2^{n-2} + ... + i_1 2^1 + i_0 2^0$

where the i's are binary digits equal either 0 or 1 consider card k to be in the left packet: then necessarily $i_{n-1} = 0$

card k is in position $k + 1$ after one ideal riffle shuffle it ends up being in position $2(k + 1) - 1$

this position is those of the card in the original ordered deck labelled as $2(k + 1) - 1 - 1$.

one has $2(k + 1) - 2 = 2(i_{n-1}2^{n-1} + i_{n-2}2^{n-2} + ... + i_1 2^1 + i_0 2^0 + 1) - 2$

where one has used $i_{n-1} = 0$

so we see that the binary coding of card k $(i_{n-1}\ i_{n-2}...i_0)$ becomes ($i_{n-2}...i_0 i_{n-1})$

With binary digits which are a circular permutation of those coding card k. one can do the same (mirror) reasoning if card k is in the right packet. This process repeats itself with each successive ideal riffle shuffle. It follows that necessarily after n ideal riffle shuffles one is back to the original ordered deck. □

From this lemma we conclude that the number of time we need to repeat the ideal riffle shuffle is n when $N = 2^n$.

3.2.3 Second Approach.

This approach is based on distributing goods from multiple source to customers.

Input : set of Customers and manufactures's locations.

Output: sequence of the visited vertices for each initial manufacture.

(a) Initialize all vertices as unvisited.

(b) Select randomly a manufacture , set it as the current vertex u. Mark u as visited.

(c) Find out the shortest edge connecting the current vertex u and an unvisited vertex v ,v could be a manufacture or customer if it was a customer we check if the current quantity can satisty its needs if it doesn't go to the manufacture.

(d) Set v as the current vertex u. Mark v as visited.

(e) If all the vertices in the domain are visited change the initial departure to another manufacture and go to step 1. Else, go to step 3.

(f) if all the manufactures in the domain are started by, terminate algorithm.

The following figure explain clearly the algorithm.

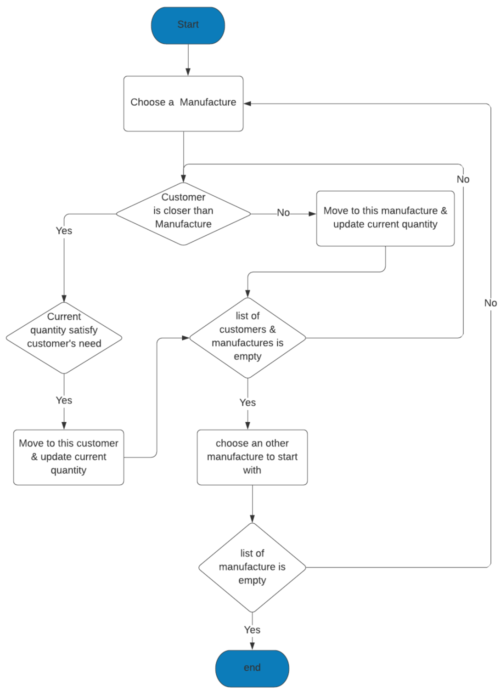
Start
Choose a Manufacture
Customer is closer than Manufacture
No
Move to this manufacture & update current quantity
Yes
Current quantity satisfy customer's need
list of customers & manufactures is empty
No
No
Yes
Yes
Move to this customer & update current quantity
choose an other manufacture to start with
list of manufacture is empty
Yes
end

Figure 3.4: Flowchart Nearest Neighbour Algorithm NNA for multiple source TSP

4. Numerical Results

4.1 Datasets

Using SAGE software, see the following link optimization code, six datasets were generated randomly, with n customers and m manufactures, three distance matrices, one represents the distance between customers, the other the distance between manufactures, the last the distance between customers and manufactures. To include global case these matrix are not symmetric. The following table summarizes the datasets used in this project.

Dataset	Connectedness	Customer number	Manufacture number	P_f	P_{c1}	P_{c2}	N	q	Iterations
Dataset 1	yes	30	5	0.8	0.65	0.25	10	8	12
Dataset 2	No	40	4	0.8	0.65	0.25	40	30	60
Dataset 3	No	60	5	0.8	0.65	0.25	30	25	20

Dataset 1 has thirty customers and five manufactures. All the centres are connected. Dataset 2 has forty customers and four manufactures, some of centres are not connected. Dataset 3 has sixty customers and five manufactures, some of centres are not connected.

4.2 Simulation results

4.2.1 Direct approach with multiple sources.
Dataset 1

Denote customers as $\{0, 1, ..., 29\}$ and manufactures as $\{0, -1, ..., -4\}$, after running the direct algorithm in SAGE the best solution found for this dataset is given by the following sequence : seq=[-4, 0, -3, 2, 1, -2, 10, 20, 9, 21, 26, 7, 18, 14, 0, 13, 15, 5, 3, -1, 25, 17, 28, 11, 16, 12, 4, 24, 29, 6, 22, 23, 27, 8, 19, 4], with the total distance being 344 and the total computational time 19.8 ms. That means truck T start from manufacture (-4) then pass by the customer 0, then to manufacture (-3) up to the last element of the sequence seq.
4.2.2 Clustering approach.
The developed algorithm of clustering assigns the customers to the manufactures. It takes into consideration the connectedness of the graph and gives the following clusters:

Dataset 1

This dataset gives four clusters which are:

Cluster 0 [0, 26, 0, 2, 25, 29, 19]

Cluster 1 [1, 3, 28, 22, 21, 24]

Cluster 2 [2, 10, 1, 15, 4, 16, 23, 5, 8, 9]

Cluster 3 [3, 17, 18, 13, 27, 14, 6, 11, 7, 20, 12]

with the total computational time is 10 ms.

Dataset 2

This dataset gives four clusters which are:

Cluster 0 [0, 18, 22, 4, 24, 17, 30, 28, 0, 9, 38, 23]

Cluster 1 [1, 8, 29, 19, 32, 21, 39, 7, 3, 12, 13]

Cluster 2 [2, 2, 26, 14, 20, 27, 11, 33, 36, 15, 5, 6, 10]

Cluster 3 [3, 16, 37, 31, 34, 25, 35, 1]

With the total computational time is 14.7 ms

Dataset 3

This dataset gives five clusters which are:

Cluster 0 [0, 34, 29, 31, 8, 4, 43, 9, 11, 5, 48, 27, 17, 6]

Cluster 1 [1, 44, 26, 53, 51, 42, 36, 46, 16, 47, 54, 56, 7]

Cluster 2 [2, 59, 12, 18, 13, 49, 57, 33, 14, 30, 3, 35, 22]

Cluster 3 [3, 58, 32, 15, 21, 45, 10, 39, 50, 28, 41, 24]

Cluster 4 [4, 1, 40, 19, 25, 20, 55, 37, 52, 0, 2, 23, 38]

with the total computational time is 15.8 ms.

Genetic algorithm GA

After applying Genetic algorithm GA on each dataset we get the following results:

Dataset 1

cluster	best order	minimum distance
0	[0, 26, 0, 2, 25, 29, 19,0]	238
1	[1, 3, 28, 22, 21, 24,1]	142
2	[2, 16, 10, 23, 1, 5, 15, 8, 4, 9,2]	433
3	[3, 27, 7, 13, 11, 18, 6, 17, 14, 20, 12,3]	415

With the total computational time of 22.9 ms.

Dataset 2

cluster	best order	minimum distance
0	[0, 18, 22, 4, 24, 17, 30, 28, 0, 9, 38, 23,0]	351
1	[1, 32, 3, 19, 7, 29, 39, 8, 21, 12, 13,1]	312
2	[2, 27, 5, 20, 15, 14, 36, 26, 33, 2, 11, 6, 10,2]	410
3	[3, 34, 16, 25, 37, 35, 31, 1,3]	285

With the total computational time of 15 ms.

Dataset 3

cluster	best order	minimum distance	Time
0	[0, 5, 4, 34, 48, 43, 29, 27, 9, 31, 17, 11, 8, 6,0]	433	
1	[1, 54, 47, 16, 46, 36, 42, 51, 53, 26, 44, 56, 7,1]	412	
2	[2, 33, 18, 3, 57, 12, 30, 49, 59, 14, 13, 35, 22,2]	492	
3	[3, 58, 32, 15, 21, 45, 10, 39, 50, 28, 41, 24,3]	489	
4	[4, 38, 23, 2, 0, 52, 37, 55, 20, 25, 19, 40, 1,4]	484	

With the total computational time of 16.3 ms

Prey predator algorithm PPA

After applying Prey predator algorithm, the results of each dataset are summarized in the following tables .

Dataset 1

cluster	best order	minimum distance	Time
0	[0, 26, 19, 2, 25, 29]	99	13.1 s
1	[28, 24, 3, 21, 22]	103	9.28 s
2	[16, 15, 5, 23, 10, 9, 8, 1, 4]	146	26.2 s
3	[27, 12, 14, 11, 7, 18, 6, 17, 20, 13]	153	32.1 s

Dataset 2

cluster	best order	minimum distance	Time
0	[28, 22, 0, 30, 38, 24, 4, 18, 9, 17, 23]	181	24.1 s
1	[19, 21, 13, 3, 29, 8, 12, 39, 7, 32]	119	21.1 s
2	[1, 25, 2, 20, 38, 0, 52, 23, 19, 37, 55, 40]	134	28.3 s
3	[20, 13, 11, 27, 7, 14, 17, 6, 12, 18]	140	20.9 s

Dataset 3

cluster	best order	minimum distance	Time
0	[8, 31, 6, 17, 9, 11, 48, 5, 43, 34, 4, 27, 29]	173	29.7 s
1	[16, 44, 56, 46, 54, 7, 42, 36, 53, 26, 47, 51]	90	27.4 s
2	[13, 14, 33, 30, 22, 18, 3, 35, 49, 12, 57, 59]	125	27.3 s
3	[24, 21, 15, 39, 28, 32, 41, 45, 58, 10, 50]	140	22.9 s
4	[1, 25, 2, 20, 38, 0, 52, 23, 19, 37, 55, 40]	134	28.2 s

In order to measure the performance of Prey-Predator Algorithm PPA and Genetic Algorithm GA, an experiment was carried out, its objective was to compute the objective function and the computational time of the different number of iterations. To do this, the three dataset were used. The following figures show the performance of those algorithms.

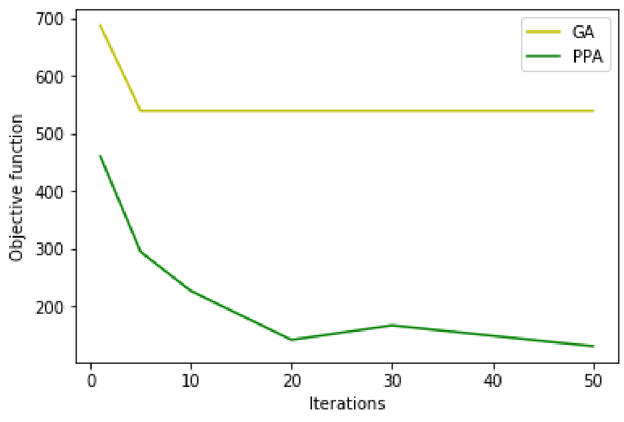

Figure 4.1: Performance of dataset 1

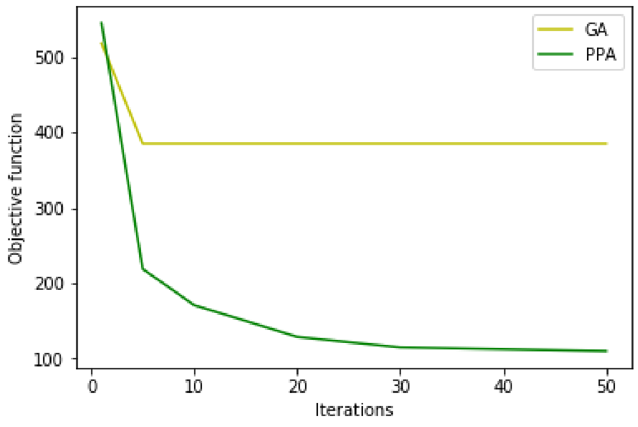

Figure 4.2: Performance for dataset 2

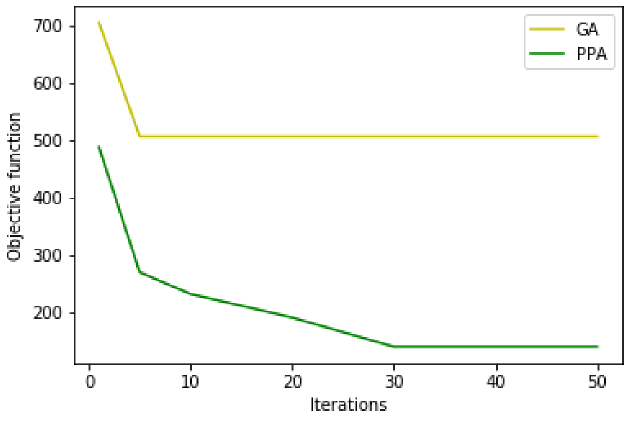

Figure 4.3: Performance of dataset 3

From these results, we notice that the performance of PPA is better than that of GA, because the performance of PPA increases with the number of iterations, since the objective function decreases.

To have a clear vision on the results of the algorithms used, the following tables summariesed results of each dataset:

Prey Predator algorithm

Datasets	Best $f(x)$	Average $f(x)$	Average Computational time
Dataset 1	131	213,5	13,5 s
Dataset 2	110	214	10,37 s
Dataset 3	140	243,5	15,6 s

Genetic algorithm

Datasets	Best $f(x)$	Average $f(x)$	Average Computational time
Dataset 1	540	564,6	4ms
Dataset 2	385	407,16	6ms
Dataset 3	507	540	7ms

4.3 Discussion and conclusion

4.3.1 Discussion.

Distributing product from multiple source to different customers was the problem studied in this project. Optimization model was introduced, respecting some assumptions, namely: a sufficent truck's capacity, single product, as well as no time constraints. To have a global vision of the optimal solution of this problem, two approaches were used. The first is based on direct distribution using the Nearest Neighbour Algorithm (NNA), the second one aims to transform multiple sources to a single source by introducing clustering concept adapted to this problem. With the aim of comparing algorithms, results for a different random dataset was used. The simulation result shows that Prey-Predator Algorithm is better than Genetic Algorithm, GA, and the performance of Prey-Predator algorithm improves though iterations.

4.3.2 future research.

Clustering approach was introduced for solving multiple sources product distribution. The performance of clustering approach can be developed since this approach divides the problem, it requires dealing with three problems clustering, determine number of clusters, and finding optimal route for each of the clusters. So, this project suggests the possibility for a future work on performance of this approach.

Due to limitation of time I didn't cover the second optimization model proposed, but that is one possible area of future work, since in this model the truck capacity limitation can be relaxed and one possible way of doing that is by having the following model:

Model 2:

In this model, we assume that we have a truck with limited carrying capacity needed to pick up or deliver multiple products at various locations. The number of trucks used in the solution is bounded above by a given value m each day (or at each depot). The products have a quantity, such as weight or volume, and the vehicles have a maximum capacity that they can carry. Also, each truck has the same number of compartments. The problem is to pick up or deliver the products at the minimum cost, while never exceeding the capacity of the vehicles.

Denote by:

C_j : Customer j ,for j$\in \{1,2,\ldots,n\}$.

M_i : Manufacure i ,for i $\in \{1,2,\ldots,m\}$.

T_k: Trucks k where k $\in$ {1,2,...,t}.

P_i: Product i where i $\in$ {1,2,...,p}.

Q_j: Compartiment j where j $\in$ {1,2,...,q}.

N_{C_j,P_i}: Need of customerC_j of Product P_i.

Cap_{Q_j,T_k}: Capacity of Compartiment Q_j in the trucks T_k.

d_{ij} : Distance between two cities i and j.

N_{ij}: Customer's need C_j of product P_i.

Decision variables:

$$Z_{ijk} = \begin{cases} 1 & \text{if i precede j in tour of } \quad T_k \\ 0 & \text{otherwise} \end{cases}$$

$$Y_{ikj} = \begin{cases} 1 & \text{if customer i receive product j by trucks } T_k \\ 0 & \text{otherwise} \end{cases}$$

$$X_{ikj} = \begin{cases} 1 & \text{if product i is affected to compartment j of trucks } T_k \\ 0 & \text{otherwise} \end{cases}$$

Constraints :

1. Capacity constraints

   Total number of products allocated to each compartment must not exceed its limited capacity.

$$\sum_{i=1}^{p} X_{ikj} \leqslant Cap_{Q_j,T_k} \quad \forall k \in \{1,...,t\} \quad \forall j \in \{1,...,q\}.$$

2. Satisfy customer needs

   Total number of products received by each customer must meet their needs

$$\sum_{k=1}^{t}\sum_{i=1}^{p} Y_{ikj} \geqslant \sum_{i=1}^{p} N_{ij} \quad \forall i \in \{1,...,p\} \quad \forall j \in \{1,...,n\}.$$

3. Compartment constraint

   Each compartment contain at most one product:

$$\sum_{i=1}^{p} X_{ikj} \leqslant 1 \quad \forall k \in \{1,...,t\} \quad \forall j \in \{1,...,q\}.$$

4. Product constraint

   Each product ordered by each customer is delivered by a single vehicle.

$$\sum_{k=1}^{t} Y_{ikj} \leqslant 1 \quad \forall j \in \{1,...,p\} \quad \forall i \in \{1,...,n\}.$$

4.3.3 Conclusion.

Multiple source product distribution problem was studied in this project with the aim of optimising the total traveling cost which can be a distance. Moreover, many algorithms was discussed, including Genetic Algorithms, NNA and Prey-Predator Algorithm. Different datasets were introduced to include different case studies.

Acknowledgements

I first would like to express my special thanks and gratitude to AIMS director Prof. Barry Green, as well as our academic director Dr. Simukai Utete for their guidance and support during this period, I am grateful to my tutor Dr. Prudence Djagba with who i have had the pleasure to work with during this and other related projects.

I would like to thank my supervisor, Dr. Surafel Luleseged Tilahun from University of Zululand, South Africa. who gave me the golden opportunity to do this wonderful project, For his patience, availability, support and above all for his full collaboration throughout this research experience. I extend my sincere thanks to the entire AIMS South Africa teaching staff who, through their advice and recommendations, were able to guide my reflections.Thank you for the good quality of their teaching and for their great efforts to ensure that their students receive up-to-date training.

I want to acknowledge AIMS and it's funders for supporting this work, because this work would not have been possible without the financial support of AIMS and it's funders. Thank you for giving the chance to African young scientists, to figure out their skills in mathematics.

Finally, I would like to express my deepest appreciation to all those who provided me the possibility to complete this report, in one way or another, and have contributed to the success of this work and who could not be cited here.

References

Dantzig, G. B. and Ramser, J. H. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

Dorigo, M. and Gambardella, L. M. Ant colonies for the travelling salesman problem. *biosystems*, 43 (2):73–81, 1997.

Dreyfus, S. Richard bellman on the birth of dynamic programming. *Operations Research*, 50(1):48–51, 2002.

Glover, F. Future paths for integer programming and links to ar tifi cial intelli g en ce. *Computers operations research*, 13(5):533–549, 1986.

John Holland. Computer programes that evolve in ways that resemble natural selection can solve complex problems even their creators do not fully understand , 1960.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *science*, 220 (4598):671–680, 1983.

Larranaga, P. and Kuijpers, C. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *ResearchGate*, 43, 1999.

Larranaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., and Dizdarevic, S. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artificial Intelligence Review*, 13(2):129–170, 1999.

Mladenović, N. and Hansen, P. Variable neighborhood search. *Computers & operations research*, 24 (11):1097–1100, 1997.

optimization code. python code. Python code , https://bitbucket.org/bouchraer-rabbany/optimization_project/src/aa65a594ead0ca723f47004ae951642ccee72cc9/Bouchra_errabany_sageCode.ipynb?at=master.

Tilahun, S. L. Swarm intelligence for a single source product distribution. 13:13, 2019.

Tilahun, S. L. and Ong, H. C. Prey-predator algorithm: a new metaheuristic algorithm for optimization problems. *International Journal of Information Technology & Decision Making*, 14(06):1331–1352, 2015.

Ting, C.-K., Su, C.-H., and Lee, C.-N. Multi-parent extension of partially mapped crossover for combinatorial optimization problems. *Expert Systems with Applications*, 37(3):1879–1886, 2010.

W.R. Hamilton and T. Kirkman. mathematical formulations of TSP, 1800.