**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY**

**──────── \* ───────**

**AI Project Report**

**A Two-Index Formulation for the Fixed-Destination Multi-Depot Asymmetric Travelling Salesman Problem Algorithm**

|  |  |  |
| --- | --- | --- |
| Instructor: | Assoc.Prof. Phạm Văn Hải |  |
| Group 1: | Lê Kỳ Anh | 20207586 |
|  | Lê Anh Quân | 20207626 |

**Hanoi – 2023**

**Table of content**

Table of Contents

[1. Mathematical Formulations for the FD-mATSP: 3](#_Toc128092831)

[1.1. General Formulations for the FD-mATSP: 3](#_Toc128092832)

[1.2. Compact SECs: 4](#_Toc128092833)

[KB-SECs 4](#_Toc128092834)

[GG-SECs 4](#_Toc128092835)

[1.3. Compact FDCs: 5](#_Toc128092836)

[FDCs1: 5](#_Toc128092837)

[FDCs2: 5](#_Toc128092838)

[1.4. Proposed FDCs: 6](#_Toc128092839)

[FDCs3: 6](#_Toc128092840)

[Lemma 1. FDCs3 is equivalent to FDCs1. 6](#_Toc128092841)

[2. Pseudocode of the algorithm: 7](#_Toc128092842)

[3. Example of FD-mATSP: 9](#_Toc128092843)

# **Mathematical Formulations for the FD-mATSP:**

## **General Formulations for the FD-mATSP:**

Let:

N: the set of nodes, N = {1,...,n} , N = D ∪ C

D: the set of depots, D = {1,..., |D|}

C: the set of customers to be visited, C = {|D| + 1,...,n}

cij: the distance from node i to node j

md: the number of salesmen located at depot d ∈ D

xij: determines whether arc (i,j) is used in the solution(xij = 1) or not (xij = 0)

Fullfilling:

,

Enforcing md salesmen to depart from depot d

Enforcing md salesmen to return to depot d

Prohibiting a tour with a unique customer

Each customer is visited exactly once

Each customer is departed exactly once

,

Domain of decision variables

Subtour elimination constraints (SECs),

Fixed destination constraints (FDCs).

Minimize:

## **Compact SECs:**

### KB-SECs

Let ui be a real number that represents the order in which a customer i is visited in an optimal tour. The KB-SECs is as follows:

,

,

,

.

K and L are the minimum and the maximum number of nodes a salesman can visit. KB-SECs are used to break any infeasible subtours and collectively impose the bounding limitations.

### GG-SECs

Let yij be the flow on arc (i, j ), i ≠ j , ∀i, j ∈ N. The GG-SECs is as follows:

,

,

,

,

,

.

The constraints impose the graph connection produced by the x-variables. They also impose the load balancing restrictions with each path carries at most L and at least K units of flow.

## **Compact FDCs:**

The following constraint forces md units of flow to depart from and return to each depot d utilizing only those arcs for which xij > 0 in order to enforce the problem's fixed-destination feature.

zijd : The flow on arc (i, j ) from depot d, ∀i, j ∈ N, i = j , ∀d ∈ D.

Propose the following FDCs:

### FDCs1:

0 ≤ zdij ≤ xij,∀i, j ∈ N , ∀d ∈ D

= = md ∀d ∈ D

- = 0 ∀d ∈ D, ∀i ∈ C, i ≠ j

Making ki a variable that represents the label given to customer i, ∀i ∈ N. We proposed the following FDCs( FDCs2)

### FDCs2:

In order to prevent a contradiction from arising from an exchange of salesmen across depots, the constraints below mark the nodes visited from each d D based on the specific index d.

kd = d, ∀d ∈ D,

ki – kj + (|D| - 1)(xij + xji) ≤ |D| - 1, ∀i, j ∈ N, i ≠ j,

kj – ki + (|D| - 1)(xij + xji) ≤ |D| - 1, ∀i, j ∈ N, i ≠ j,

Constraint capture the domain of the decision variables:

ki ≥ 0, ∀ i ∈ N

## **Proposed FDCs:**

y’ij : the label assigned to arc (i, j ), i ≠ j , ∀i, j ∈ N

We proposed the following FDCs ( FDCs3):

### FDCs3:

All salesman departing from depot d are given the label "d," which serves as a flow that is maintained throughout the salesmen's tours from that depot.

y’di ≤ |D|xij, ∀i, j ∈ N ,

Constraints enforce flow to only occur on arcs for which xij > 0.

y’di = dxdi, ∀d ∈ D, ∀i ∈ C,

y’di = dxid, ∀d ∈ D, ∀i ∈ C,

d units are required by constraints to leave and return to each base using arcs for which, respectively, xdi > 0 and xid > 0, ∀d ∈ D, an d∀ I ∈ C, respectively.

- = 0

Constraints above are the standard flow conservation constraints.

y’ij ≥ 0, ∀i, j ∈ N.

Constraints represent the domain of the y’ij -variables

Note: While FDCs2 labels the nodes, FDCs3 labels the arcs visited by each d ∈ D.

### Lemma 1. FDCs3 is equivalent to FDCs1.

Proof: Verifying the equivalence between FDCs1 and FDCs3 allows us to demonstrate this statement. If we let y’ij = then, y’ij , ∀i, j ∈ N satisfy constraints in FDCs3. Similar to this, setting = 1, ∀i, j on this path, and = 0 otherwise, ∀d ∈ D, results in a feasible path for a d in FDCs3. Consequently, , ∀i, j ∈ N, ∀d ∈ D, will satisfy Constraints in FDCs1. FDCs1 and FDCs3 are hence equivalent.

# **Pseudocode of the algorithm:**

# Input: A complete graph G = (N,A) with vertex set N = D ∪ C, where D is a set of depots and C is a set of customers.

# Output: A set of m tours T = {T1, ..., Tm} such that each tour Ti starts and ends at a depot di ∈ D and visits exactly one customer cj ∈ C.

* Initializes an empty set of tours T and a set of unvisited customers U,
* For each depot di in D, it creates a tour Ti that starts and ends at di,
* Finds a customer cj in U that has the lowest cost of traveling to and from di,
* Adds cj to Ti and removes cj from U,
* Repeats this process until all customers are visited,
* Returns T as the solution.

|  |  |
| --- | --- |
| **FD-mATSP** | |
| **1:** | **T = {}** |
| **2:** | U = C |
| **3:** | **For** di in D: |
| **4:** | Ti = [] |
| **5:** | Ti.append(di) |
| **6:** | cj = min(U, key=lambda j: cij + cji) |
| **7:** | Ti.append(cj) |
| **8:** | U.remove(cj) |
| **9:** | Ti.append(di) |
| **10:** | T.add(Ti) |
| **11:** | **Endfor** |
| **12:** | return T |

|  |  |
| --- | --- |
| **Compact SECs** | |
| **1:** | **For** each i in C: |
| **2:** | 1 ≤ ui ≤ |C| |
| **3:** | **For** each d in D |
| **4:** | ui + (L-2) – ≤ L - 1 |
| **5:** | ui + + (2 – K) ≥ 2 |
| **6:** | **End For** |
| **7:** | **For** each j in C |
| **8:** | **If** I != j: |
| **9:** |  |
| **10:** | **End If** |
| **11:** | **End For** |
| **12:** | **End For** |

|  |  |
| --- | --- |
| **FDCs 1** | |
| **1:** | **For** each i,j in N: |
| **2:** | **For** each d in D: |
| **3:** | 0 ≤ zdij ≤ xij |
| **4:** | **End for** |
| **5:** | **End For** |
| **6:** | **For** each i in C |
| **7:** | **For** each d in D |
| **8:** | = = md |
| **9:** | **If** i != j: |
| **10:** | **For** each j in N |
| **11:** | - = 0 |
| **12:** | **End For** |
| **13:** | **End If** |
| **14:** | **End For** |
| **15:** | **End For** |

|  |  |
| --- | --- |
| **Pseudocode** | |
| **1:** | **For** epoch = 0 to number of epochs - 1: |
| **2:** | **For** iter = 0 to number of iterations per epoch - 1: |
| **3:** | batch\_x, batch\_y = get\_batch() |
| **4:** | l = loss(f(batch\_x, A), batch\_y) |
| **5:** | **For** i = 0 to number of parameters - 1: |
| **6:** | ; |
| **7** | **End For** |
| **8:** | **End For** |
| **9:** | **End For** |

|  |  |
| --- | --- |
| **FDCs 3** | |
| **1:** | // Initialize epoch and iter to zero  epoch = 0  iter = 0 |
| **2:** | **For** each j in N: |
| **3:**  4:  5: | 0 ≤ y’di ≤ |D|xij  **End For**  **End For** |
| **6:** | **For** each d in D: |
| **7:** | **For** each i in C: |
| **8:** | **If** i != f: |
| **9:** | - = 0 |
| **10:** | **End If** |
| **11:** | y’di = dxdi |
| **12:** | y’id = dxdi |
| **13:** | **End For** |
| **14:** | **End For** |

# **Example of FD-mATSP:**

* D = {1, 2} are depots
* C = {3, 4, 5} are customers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 1 | 0 | 0 | 5 | 7 | 9 |
| 2 | 0 | 0 | 6 | 8 | 10 |
| 3 | 12 | 11 | 0 | 0 | 0 |
| 4 | 14 | 13 | 0 | 0 | 0 |
| 5 | 16 | 15 | 0 | 0 | 0 |

* Salesman A starts from depot 1, visits customer 3 and returns to depot 1 with a total cost of (5 +12) =17
* Salesman B starts from depot 2, visits customers 4 and then customer 5 before returning to depot 2 with a total cost of (8 +10 +15) =33
* Minimum cost = 17 + 33 = 50

Example:

A courier company that has multiple warehouses and multiple delivery vehicles. Each warehouse has a fixed number of vehicles that depart from and return to the same warehouse after delivering a set of packages to different locations. The travel time and cost between locations are not symmetric, meaning that they depend on factors such as traffic, road conditions, tolls, etc. The goal is to find the optimal routes for each vehicle that minimize the total delivery time and cost.

Constraint:

The constraint for FD-mATSP is that each vehicle must deliver exactly one package per location and must not visit any location more than once. Additionally, each vehicle must return to their original warehouse after delivering all packages.

* Subtour elimination constraints (SECs): These are constraints that prevent cycles or loops in the routes that do not include the depot. For example, a SEC could be that the sum of binary variables indicating whether an arc is used in a route must be less than or equal to the number of locations visited by that route minus one.
* Table constraint: This is a constraint that specifies a set of allowed tuples for a set of variables. For example, a table constraint could be used to define the asymmetric travel costs between locations by listing all possible pairs of locations and their corresponding costs.
* Regular constraint: This is a constraint that enforces that a sequence of variables belongs to a regular language defined by a finite automaton. For example, a regular constraint could be used to ensure that each vehicle returns to its original depot after visiting all locations by defining a finite automaton with states corresponding to depots and transitions corresponding to locations.