UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF MECHANICAL ENGINEERING

MIXED INTEGER PROGRAMMING MODELS FORMULATION FOR INBOUND TRUCKS TO DOOR ASSIGNEMENT ON CROSS DOCKING FACILITIES

POSTGRADUATE THESIS

of Magalios Vaios

Chemical Engineer A.U.Th.

ADVISOR: ***Saharidis K.D. George, Assistant Professor***

VOLOS 2017

Η έγκριση της μεταπτυχιακής εργασίας από το Τμήμα Μηχανολόγων Μηχανικών της Πολυτεχνικής Σχολής του Πανεπιστημίου Θεσσαλίας δεν υποδηλώνει αποδοχή των απόψεων του συγγραφέα (Ν. 5343/32 αρ. 202 παρ. 2).

Εγκρίθηκε από τα Μέλη της Τριμελούς Εξεταστικής Επιτροπής:

Πρώτος Εξεταστής (Επιβλέπων)	Δρ. Γεώργιος Σαχαρίδης Επίκουρος Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Δεύτερος Εξεταστής	Δρ. Γεώργιος Λυμπερόπουλος Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας
Τρίτος Εξεταστής	Δρ. Γεώργιος Κοζανίδης Επίκουρος Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών, Πανεπιστήμιο Θεσσαλίας

ACKNOWLEDGEMENTS

First of all, I would like to thank the advisor of this postgraduate thesis, Assistant Professor Mr. George Saharidis, for his valuable assistance and guidance during the writing of this thesis. Secondly, I am thankful for the members of the Advisory Committee, Professor Mr. George Liberopoulos and Assistant Professor Mr. George Kozanidis. I would also like to thank my friends and colleagues for their moral support and comprehension during the last months. Most of all, I am grateful to my parents for their love and support all these years. I would like to dedicate this thesis to my father and my mother.

ABSTRACT

This work aims to present the mathematical models used in cross docking facilities in order to assign inbound trucks to inbound doors.

It presents four mathematical models, two of continuous time representation and two of discrete time. Each time representation consists of one bi-index model and one three index model.

Finally, for each model an exact algorithm is formulated in C++ code and the numerical results are presented and discussed.

Contents

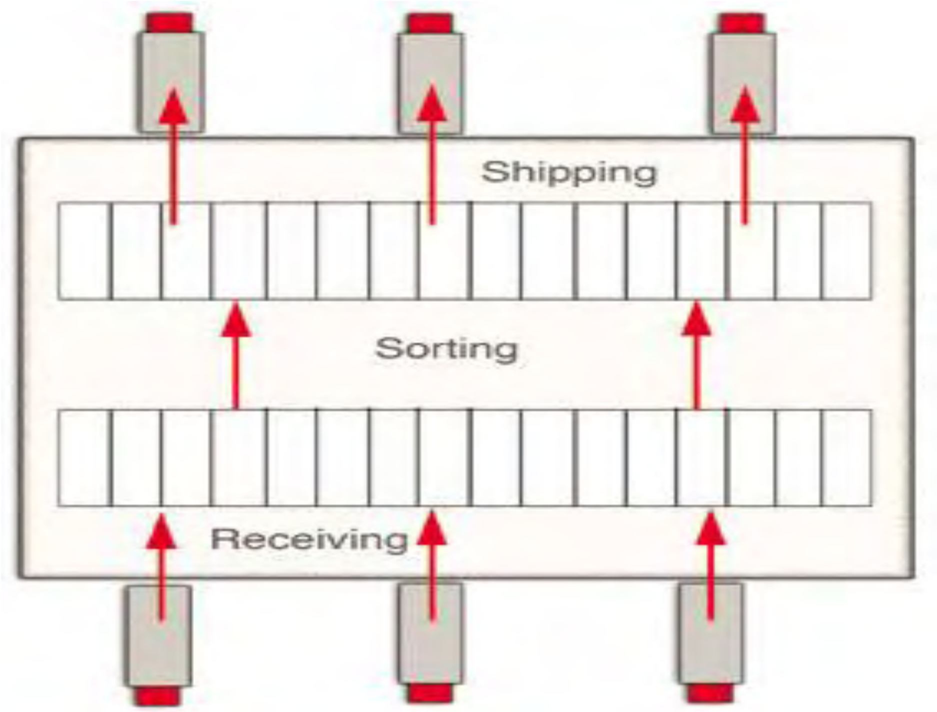
Contents of tables

Contents of shapes

1. INTRODUCTION

Speed and productivity of a supply chain has become an important factor of growth for organizations. Cross-docking is just one strategy that can be implemented to help achieve a competitive advantage. Implemented appropriately and in the right conditions, cross-docking can provide significant improvements in efficiency and handling times (Van Belle et al., 2012).

Cross docking is a logistics procedure where products from a supplier or manufacturing plant are distributed directly to a customer or retail chain with marginal to no handling or storage time. Cross docking takes place in a distribution docking terminal; usually consisting of trucks and dock doors on two (inbound and outbound) sides with minimal storage space. The name ‘cross docking’ explains the process of receiving products through an inbound dock and then transferring them across the dock to the outbound transportation dock (Van Belle et al., 2012).



Shape 1.1: Cross dock graphic (http://eurekapub.eu)

In simple terms, inbound products arrive through transportation such as trucks, and are allocated to a receiving dock on one side of the ‘cross dock’ terminal. Once the inbound transportation has been docked its products can be moved either directly or indirectly to the outbound destinations; they can be unloaded, sorted and screened to identify their end destinations. After being sorted, products are moved to the other end of the ‘cross dock’ terminal via a forklift, conveyor belt, pallet truck or another means of transportation to their destined outbound dock. When the outbound transportation has been loaded, the products can then make their way to customers (Boysen & Fleidner, 2010).

The process of cross docking will not suit every warehouses needs, it is therefore important to make an informed decision as to whether cross-docking will increase the productivity, costs and customer satisfaction for your specific business. Cross docking can advance the supply chain for a variety of specific products. Already packaged and sorted products ready for transportation to a particular customer can become a faster and more efficient process through cross docking (Van Belle et al., 2012).

Some of the main reasons cross docking is implemented is to:

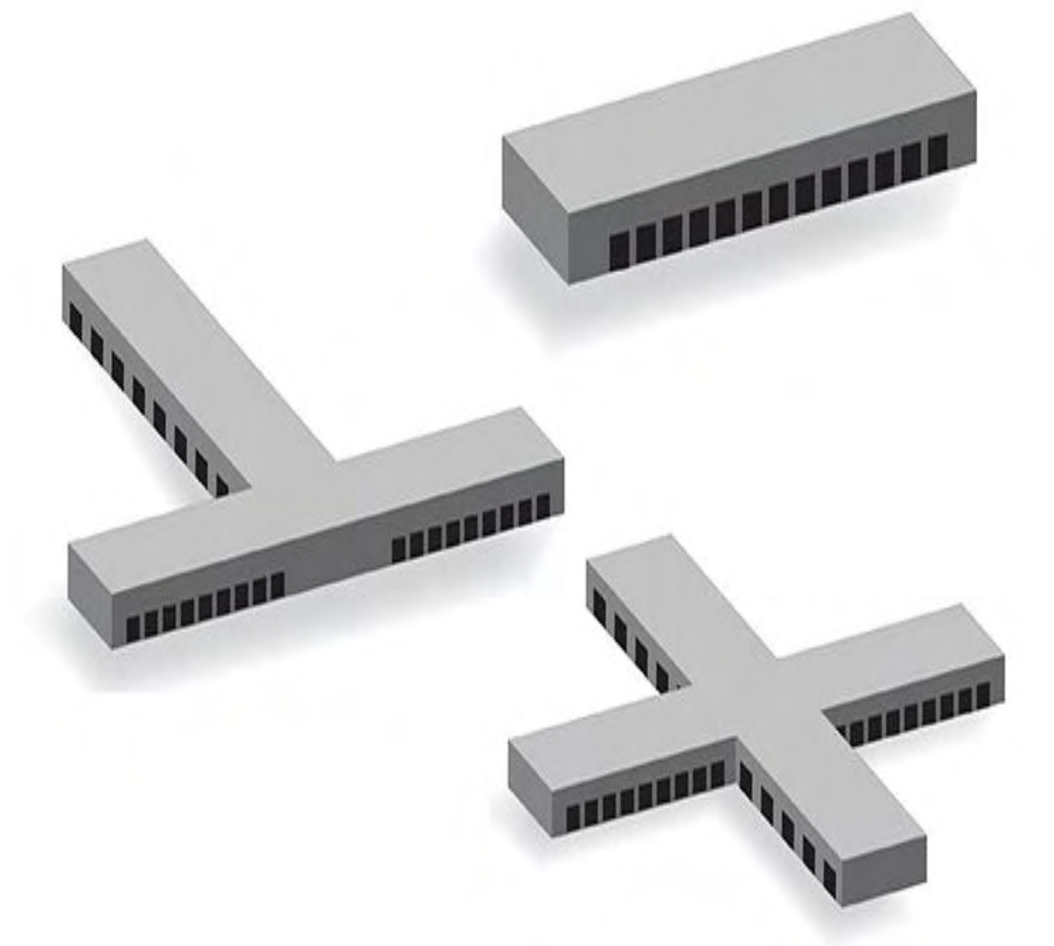
- Provide a central site for products to be sorted and similar products combined to be delivered to multiple destinations in the most productive and fastest method. This process can be described as ‘hub and spoke’.
- Combine numerous smaller product loads into one method of transport to save on transportation costs. This process can be described as ‘consolidation arrangements’.
- Break down large product loads into smaller loads for transportation to create an easier delivery process to the customer. This process can be described as ‘deconsolidation arrangements’.

Advantages of cross-docking:

- Streamlines the supply chain, from point of origin to point of sale
- Reduces costs through less inventory handling

- Reduces inventory holding costs by reducing storage times and potentially eliminating the need to retain safety stock
- Products reach the distributor, and consequently the customer, faster
- Reduces or eliminates warehousing costs
- Less risk of inventory handling

Cross dock facilities are generally designed in an "I" configuration, which is an elongated rectangle. The goal in using this shape is to maximize the number of inbound and outbound doors that can be added to the facility while keeping the floor area inside the facility to a minimum. There are also facilities that use L, U, H, T or E shapes (Saxena, 2007).



Shape 1.2: Shapes of cross dock facilities (http://eurekapub.eu)

Cross dock operations were first pioneered in the US trucking industry in the 1930s, and have been in continuous use in less-than-truckload (LTL) operations ever since. Wal-Mart is a well-known example that began using cross-docking in the retail sector in the late 1980s. Now Wal-Mart is able to offer lower prices than their competitors because of the ability to reduce its costs of sales (Boysen & Fleidner, 2010).

2. LITERATURE REVIEW

The assignment problem is one of the fundamental combinatorial optimization problems in the branch of optimization or operations research in mathematics. It consists of finding a maximum weight matching or minimum weight perfect matching in a weighted bipartite graph. In its most general form, the problem has a number of agents and a number of tasks. Any agent can be assigned to perform any task, incurring some cost that may vary depending on the agent-task assignment. It is required to perform all tasks by assigning exactly one agent to each task and exactly one task to each agent in such a way that the total cost of the assignment is minimized (Agustina et al., 2010)

If the numbers of agents and tasks are equal and the total cost of the assignment for all tasks is equal to the sum of the costs for each agent (or the sum of the costs for each task, which is the same thing in this case), then the problem is called the linear assignment problem (Miao et al., 2006).

The existing literature is limited because cross-docking has attracted the academicals attention during the recent years. After 2004 is published 85% of the academic papers due to the growing interest from industry (Van Belle et al., 2012). (Boysen & Fleidner, 2010) studied the truck scheduling problem classified the considered problems. Studies provide solutions on the door assignment and proposed mathematical models used in cross-docking (Agustina et al., 2010). The arrival and departure time, the operational time, the capacity of a cross-dock facility, the time windows, the number of trucks and their capacities and the number of servicing doors are parameters that can affect the formulation of the mathematical model. Many models are formulated to minimize total cost of cross-dock facilities and reduce the time that a truck or a product needs to remain in a facility (Soltani & Sadjadi, 2010).

This work aims to present the mathematical formulation of two continuous time models and two discrete time models for the scheduling of inbound trucks to doors at a cross-docking facility.

3. MODEL FORMULATION

3.1 General

The arrival time of each inbound truck should be known in a cross dock facility in order to assign the truck to an inbound door. The trucks are either immediately assigned to a door or have to wait in line in front of the inbound door that they are assigned. The assignment has to be planned before the arrival of trucks and has to be planned carefully. The 4 mathematical formulations have as objective the minimization of total waiting time and total handling time for all inbound trucks.

The developed models are mixed integer linear problems and based on branch-and-bound or branch-and-cut methods who intend to be very complex. An algorithm requires too much time to give the first integer-feasible solution. Many researchers developed heuristic algorithms for the assignment problem on dross docking and can solve large scale problems.

This work presents four formulations for the assignment of trucks at the inbound doors of a cross docking facility. The two first formulations are based on a continuous time representation but the second one has a reduced number of variables. Formulations three and four are based on a discrete time representation but the fourth has a reduced number of variables.

3.2 Mathematical model using continuous time formulation and three index decision variables

The first model presents the classical modeling approach. The developed model uses a three index decision variable. In the following tables are presented the indexes, the parameters and the decision variables of the mathematical model 1.

Table 3.1: Indexes of the mathematical model 1

Indexes	Definition
i	Index of inbound trucks
j	Index of inbound doors
k	Index of service order of inbound trucks

Table 3.2: Parameters of the mathematical model 1

Parameter	Definition
AT_j	Scheduled arrival time for truck j
$HT_{i,j}$	Handling time for truck j at door i
S_i	Time door i becomes available for the first time in the planning horizon
M	Big positive number

Table 3.3: Variables of the mathematical model 1

Variables	Definition
$X_{j,i,k}$	Binary variable, equals to 1 if truck j is going to be served at door i as the k^{th} truck
T_j^s	Continuous positive variable expresses the starting time of truck j

The first constraint of the developed model ensures that truck j is served in only one door i with only one order k which means that all truck should be served in only one door;

$$\sum_i \sum_k X_{j,i,k} = 1 \qquad \forall j \quad (1)$$

The second constraint ensures that each door i services at most one truck j with order k which mean that maximum one truck will be served with a specific order in a specific door;

$$\sum_j X_{j,i,k} \leq 1 \qquad \forall i,k \quad (2)$$

The third constraint defines that the starting time of truck j is after the arrival time of truck j which means that the starting time of service will be after the arrival time of a truck;

$$T_j^s \geq AT_j \qquad \forall j \quad (3)$$

Constraints 4a and 4b ensure that the starting time of service of truck j is after the time door i becomes available which means that a door can service for first time only after it is available;

$$T_j^s \geq S_i \qquad \forall j \quad (4a)$$

$$T_j^s \geq S_i - M * \left(1 - \sum_k X_{j,i,k}\right) \qquad \forall j,i,k > 1 \quad (4b)$$

Constraint 5a guarantees that if truck j is served at door i with service order k, then previously were served k-1 trucks at same door;

$$(k-1) * X_{j,i,k} \leq \sum_{0}^{h<k} \sum_{j' \neq j} X_{j',i,h} \qquad \forall j,i,k > 1 \quad (5a)$$

Constraint 5b expresses the same thing with 5a. If there is a truck served at door i with order k, then has been served another truck with order k-1;

$$\sum_{j} X_{j,i,k} \leq \sum_{j} X_{j,i,k-1} \qquad \forall i,k \geq 1 \quad (5b)$$

Also, if a truck j is served at door i with order k, then the sum of all trucks served at door i until order k-1 equals to 1;

$$X_{j,i,k} \leq \sum_{j' \neq j} X_{j',i,k-1} \qquad \forall j,i,k > 1 \quad (5c)$$

If k=1 then constraints 5a, 5b and 5c don't have effect on the mathematical model.

Constraint 6 combines the starting time of service of truck j' with order k at door i, with time that finishes the service of truck j with order k-1 at dock i;

$$T^{s}_{j' \neq j} + M * \left(1 - X_{j' \neq j,i,k}\right) \geq T^{s}_{j} + HT_{j,i} * X_{j,i,k-1} - M * \left(1 - X_{j,i,k-1}\right) \qquad \forall j',j,i,k \quad (6)$$

The objective function of Model 1 is the minimization of the total service time (starting and handling time) of all trucks to all docks;

$$Min\left(\sum_{j} T^{s}_{j} + \sum_{j,i,k} X_{j,i,k} * HT_{j,i}\right)$$

3.3 Mathematical model using continuous time formulation and two index decision variables

The second model is an extension of model 1 but uses two index decision variables. In the following tables are presented the indexes, the parameters and the decision variables of the mathematical model 2.

Table 3.4: Indexes of the mathematical model 2

Indices	Definition
i	Index of inbound trucks
j	Index of inbound doors
k	Index of service order of inbound trucks

Table 3.5: Parameters of the mathematical model 2

Parameter	Definition
AT_j	Scheduled arrival time for truck j
$HT_{i,j}$	Handling time for truck j at door i
S_i	Time door i becomes available for the first time in the planning horizon
M	Big positive number

Table 3.6: Variables of the mathematical model 2

Variables	Definition
$X_{j,i}$	Binary variable, equals to 1 if truck j is going to be served at door i
$W_{j,k}$	Binary variable, equals to 1 if truck j is assigned at a door with service order k
T_j^s	Continuous positive variable expresses the starting time of service of truck j

In model 2, constraint 1 of model 1 is replaced by constraints 7 and 8 that guarantee that each truck will be assigned to exactly one door i with exactly one order k which means that all trucks will be served with a specific order in a specific door;

$$\sum_i X_{j,i} = 1 \qquad \forall j \qquad (7)$$

$$\sum_k W_{j,k} = 1 \qquad \forall j \qquad (8)$$

Model 2 uses also constraints 3, 4a and 4b of model 1;

$$T_j^s \geq AT_j \qquad \forall j \qquad (3)$$

$$T_j^s \geq S_i \qquad \forall j \qquad (4a)$$

$$T_j^s \geq S_i - M * (1 - X_{j,i}) \qquad \forall j, i \quad (4b)$$

Constraint 9, which takes place of constraint 2 of model 1, guarantees that at each door i at most one truck j will be assigned in order k. If truck j and truck j' are assigned at the same door i then they can't be served in the same order. If $X_{j,i} = W_{j,k} = 1$ and $X_{j' \neq j,i} = 1$ then $W_{j' \neq j,k} \leq 0 \Longrightarrow W_{j' \neq j,k} = 0$;

$$W_{j' \neq j,k} - M * (1 - X_{j' \neq j,i}) \leq M * (1 - W_{j,k}) + M * (1 - X_{j,i}) \qquad \forall j, j' \neq j, i, k \quad (9)$$

In model 2 is not needed constraint 5 of model 1, because it is only needed to guarantee that after a truck is served in order k, the next truck will be served in order k+1. Constraint 10 guarantees that fact. If a truck j is assigned at door i in order k $W_{j,k} = 1 = X_{j,i}$ and truck j' is assigned at the same dock i $X_{j',i} = 1$ then $W_{j',k+1} = 1$ as $W_{j' \neq j,k+1} \geq 1$;

$$W_{j,k} - M * (1 - X_{j,i}) \leq W_{j' \neq j,k+1} + M * (1 - X_{j' \neq j,i}) \qquad \forall j, j' \neq j, i, k \quad (10)$$

Finally, constraint 11 replaces constraint 6 of model 1. Constraint 11 ensures that truck in order k+1 begins been served after the end of service of truck in order k;

$$T_j^s + HT_{j,i} * X_{j,i} - M * (1 - X_{j,i}) - M * (1 - W_{j,k}) \leq T_{j' \neq j}^s + M * (1 - X_{j' \neq j,i}) + M * (1 - W_{j'j,k+1}) \qquad \forall j, i, k > 1 \quad (11)$$

The objective function of Model 2 is the minimization of the total service time (starting and handling time) of all trucks to all docks;

$$Min\left(\sum_{j} T_j^s + \sum_{j,i} X_{j,i} * HT_{j,i}\right)$$

3.4 Mathematical model using discrete time formulation and three index decision variables

In discrete time representation the time horizon is divided into intervals of equal size. The main issue of this type formulations is the selection of the time interval size. The third model uses two three index decision variables and due to the discrete time is no involved the continuous decision variable used in previous models. In the following tables are presented the indexes, the parameters and the decision variables of the mathematical model 3.

Table 3.7: Indexes of the mathematical model 3

Indices	Definition
i	Index of inbound trucks
j	Index of inbound doors
t	Index of time

Table 3.8: Parameters of the mathematical model 3

Parameter	**Definition**
AT_j	Scheduled arrival time for truck j
$HT_{i,j}$	Handling time for truck j at door i
S_i	Time door i becomes available for the first time in the planning horizon
M	Big positive number

Table 3.9: Variables of the mathematical model 3

Variables	**Definition**
$CC_{j,i,t}$	Binary variable, equals to 1 if truck j is served during t period at dock i
$SC_{j,i,t}$	Binary variable, equals to 1 if truck j is assigned at t period at dock i

In model 3, constraint 12, which takes place of constraint 1 of model 1, guarantees that in only one period a truck j starts its service in only one door i;

$$\sum_t \sum_i SC_{j,i,t} = 1 \qquad \forall j \quad (12)$$

Constraint 13 is added in order to be completely expressed constraint 1 of model 1 and guarantees that in only one door i a truck j starts its service in period t;

$$\sum_i CC_{j,i,t} \leq 1 \qquad \forall j, t \quad (13)$$

Constraints 14a and 14b, take place of constraints 3 and 4 of model 1, ensure that a truck starts its service only after it has arrived at the facility;

$$\sum_{i}\sum_{t\geq ATj} SC_{j,i,t} = 1 \qquad \forall j \quad (14a)$$

$$\sum_{i,t} \left(t * SC_{j,i,t}\right) \geq AT_j \qquad \forall j \quad (14b)$$

Constraint 15 replaces constraint 2 of model 1 and ensures that no truck starts its service in a door i before this door is available;

$$\sum_{j}\sum_{t<S_i} SC_{j,i,t} = 0 \qquad \forall i \quad (15)$$

Constraint 16 expresses that maximum one truck j could be served in each door i during period t;

$$\sum_{j} CC_{j,i,t} \leq 1 \qquad \forall i,t \quad (16)$$

Constraint 6 of model 1 is expressed by constraints 17 and 18 which ensure that if a truck is assigned in a certain door i during a period t (for n=0, then $SC_{j,i,t} = 1$) then the truck j will be served for $HT_{i,j}$ periods;

$$\left(\sum_{n=0}^{HT_{j,i}} CC_{j,i,t+n}\right) - SC_{j,i,t} * \left(M + HT_{j,i}\right) \geq -M \qquad \forall j,i,t < \left(T - HT_{j,i}\right) \quad (17)$$

$$-\left(\sum_{n=0}^{HT_{j,i}} CC_{j,i,t+n}\right) + SC_{j,i,t} * \left(-M + HT_{j,i}\right) \geq -M \qquad \forall j,i,t < \left(T - HT_{j,i}\right) \quad (18)$$

Constraint 19 guarantees that there is a service process of a truck ($CC_{j,i,t} = 1$) during period t if this truck is assigned to a door during this period ($SC_{j,i,t} = 1$);

$$CC_{j,i,t} \geq SC_{j,i,t} \qquad \forall j,i,t \quad (19)$$

If there is no service process during period t-1 ($CC_{j,i,t-1} = 1$) and there is in period t ($CC_{j,i,t} = 1$) then an assignment is occurred during period t ($SC_{j,i,t} = 1$), and this fact is guaranteed by constraint 20;

$$SC_{j,i,t} \geq CC_{j,i,t} - CC_{j,i,t-1} \qquad \forall j, i, t > 1 \qquad (20)$$

The objective function of Model 3 is the minimization of starting time of service of each truck and handling time;

$$Min \sum_{j,i,t} \left((t-1) * SC_{j,i,t} + SC_{j,i,t} * HT_{j,i} \right)$$

It is also used an objective for the minimization of total time of service;

$$Min \sum_{j,i,t} \left((t-1) * CC_{j,i,t} + SC_{j,i,t} * HT_{j,i} \right)$$

3.5 Mathematical model using discrete time formulation and two index decision variables

The fourth model is an extension of model 3 but uses four two index decision variables. In the following tables are presented the indexes, the parameters and the decision variables of the mathematical model 4.

Table 3.10: Indexes of the mathematical model 4

Indices	Definition
i	Index of inbound trucks
j	Index of inbound doors
t	Index of time

Table 3.11: Parameters of the mathematical model 4

Parameter	Definition
AT_j	Scheduled arrival time for truck j
$HT_{i,j}$	Handling time for truck j at door i
S_i	Time door i becomes available for the first time in the planning horizon
M	Big positive number

Table 3.12: Variables of the mathematical model 4

Variables	Definition
$CC_{j,i}$	Binary variable, equals to 1 if truck j is served at door i
$SC_{j,i}$	Binary variable, equals to 1 if truck j is assigned at door i
$WCC_{j,t}$	Binary variable, equals to 1 if truck j is served in period t
$WSC_{j,t}$	Binary variable, equals to 1 if truck j is assigned at period t

In model 4 constraint 21, which takes place of constraint 1 of model 1, guarantees that for every truck j there is exactly one door i where it starts to be served;

$$\sum_{i} SC_{j,i} = 1 \qquad \forall j \quad (21)$$

Constraint 22, which replaces constraint 2 of model 1, guarantees that for every truck j there is exactly one period t when it starts to be served;

$$\sum_{t} WSC_{j,t} = 1 \qquad \forall j \quad (22)$$

Constraint 23 ensures that if two trucks j and j' are assigned at door i then they can't be served both during period t. Constraint 23 also guarantees that all four variables above cannot equal to 1 for given j, j'≠j, i, t;

$$CC_{j,i} + WCC_{j,t} + CC_{j'\neq j,i} + WCC_{j'\neq j,t} \leq 3 \qquad \forall j, j', i, t \quad (23)$$

Constraint 3 of model 1 is expressed by constraint 24 which ensures that every truck j can't be served before it has arrived at the facility;

$$\sum_{t<AT_j} \left(WCC_{j,t} + WSC_{j,t}\right) = 0 \qquad \forall j \quad (24)$$

Constraint 25, which replaces constraint 4 of model 1, guarantees that every truck j can't be served before every door i is available for service;

$$\sum_{t<S_i} WCC_{j,t} \leq M * (1 - CC_{j,i}) \qquad \forall j,i \quad (25)$$

Constraint 6 of model 1 is expressed by constraints 26 and 27 which ensure that if a truck is served in a certain door i during a period t (for n=0, then $WCC_{j,i,t} = 1$) then the truck j will be served for $HT_{i,j}$ periods that keeps the handling time of that truck in that door;

$$\sum_{n=0}^{HT_{i,j}} WCC_{j,t+n} \leq HT_{i,j} + M * [(1 - SC_{j,i}) + (1 - WSC_{j,t})] \qquad \forall j,i,t \quad (26)$$

$$-\sum_{n=0}^{HT_{i,j}} WCC_{j,t+n} \leq -HT_{i,j} + M * [(1 - SC_{j,i}) + (1 - WSC_{j,t})] \qquad \forall j,i,t \quad (27)$$

Constraints 28 and 29 guarantee that there is a service process of a truck ($CC_{j,i} = 1$, $WCC_{j,t} = 1$) during period t if this truck is assigned to a door ($SC_{j,i} = 1$) during this period ($WSC_{j,t} = 1$);

$$CC_{j,i} \geq SC_{j,i} \qquad \forall j,i \quad (28)$$

$$WCC_{j,t} \geq WSC_{j,t} \qquad \forall j,t \quad (29)$$

If there is no service process during period t-1 ($WCC_{j,i,t-1} = 1$) and there is in period t ($WCC_{j,i,t} = 1$) then an assignment is occurred during period t ($WSC_{j,i,t} = 1$), and this fact is guaranteed by constraint 30;

$$WSC_{j,t} \geq WCC_{j,t} - WCC_{j,t-1} \qquad \forall j, t > 1 \quad (30)$$

The objective function of Model 4 is the minimization of starting time of service of each truck and handling time;

$$\text{Min} \sum_{j,t} [(t+1) * WSC_{j,t}] + \sum_{j,i} (SC_{j,i} * HT_{i,j})$$

4. EXAMPLE AND RESULTS

4.1 General

Four C++ codes were formulated based on mathematical programming. Each code was formulated in different versions because of variety of constraints. For example, in model 1 constraint 4 is expressed by two different constraints, 4a and 4b. also constraint 5 has three different expressions, 5a, 5b and 5c. That means that run 6 different versions for model 1. Those versions are presented as 1-4a-5a, 1-4a-5b, 1-4a-5c, 1-4b-5a, 1-4b-5b and 1-4b-5c. same in model 2 there are versions 2-4a and 2-4b. Model 3 uses two different objective functions and two versions of constraint 13. So the versions are 3-obj1-13a, 3-obj1-13b, 3-obj2-13a and 3-obj2-13b. The codes were tested for a cross docking facility with five inbound doors, who were going to service twenty inbound trucks. Due to the complexity of problem, some of the codes did not give a feasible solution immediately but after a long period and it was necessary to stop running the programs after 2 hours. On the table below are presented the value of the objectives of each model, time took to give solution and Gap.

Table 4.1a: Value of models, CPU time, gap and value calculated based on objective function of model 1

Model	Value	Time (h)	Gap (%)	Value for model 1 objective
1-4a-5a	2381.667	2	52.16	2381.667
1-4a-5b	2227.07	2	48.06	2227.07
1-4a-5c	2271.42	2	48.59	2271.42
1-4b-5a	2452.24	2	52.18	2452.24
1-4b-5b	2236.38	2	48.27	2236.38
1-4b-5c	2226	2	48.06	2226

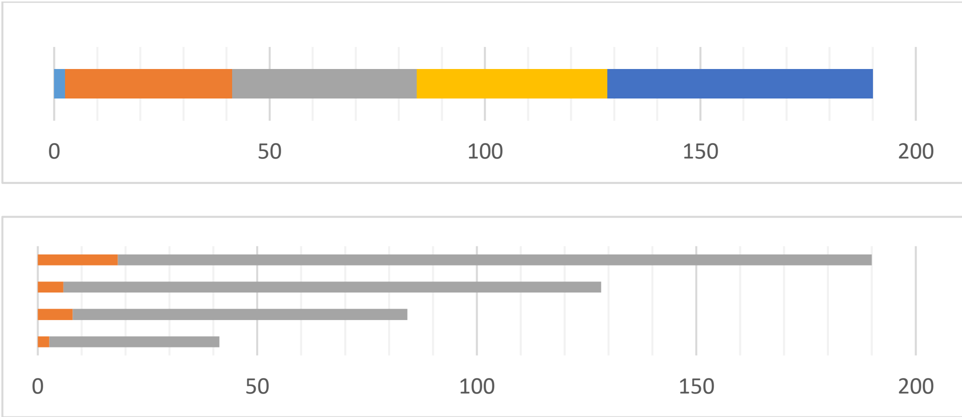
Table 4.1b: Value of models, CPU time, gap and value calculated based on objective function of model 1

Model	Value	Time (h)	Gap (%)	Value for model 1 objective
2-4a	2790.04	instantaneously	100	2790.04
2-4b	2850.86	instantaneously	100	2850.86
3-obj1-13a	264	instantaneously	100	2495,64
3-obj1-13b	247	instantaneously	100	2548,37
3-obj2-13a	264	instantaneously	100	2495,64
3_obj2_13b	247	instantaneously	100	2548,37
4	264	instantaneously	100	2446,31

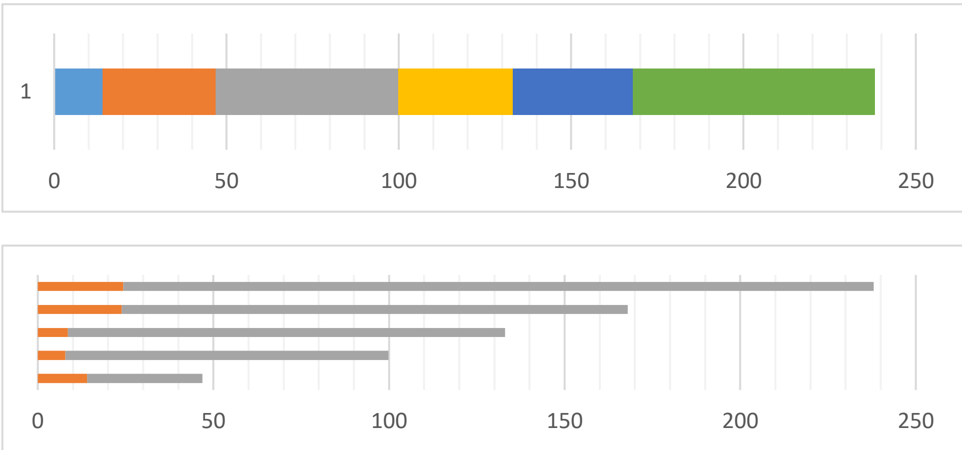
4.2 Results model 1-4a-5a

Table 4.2: Assignment of inbound trucks to inbound doors

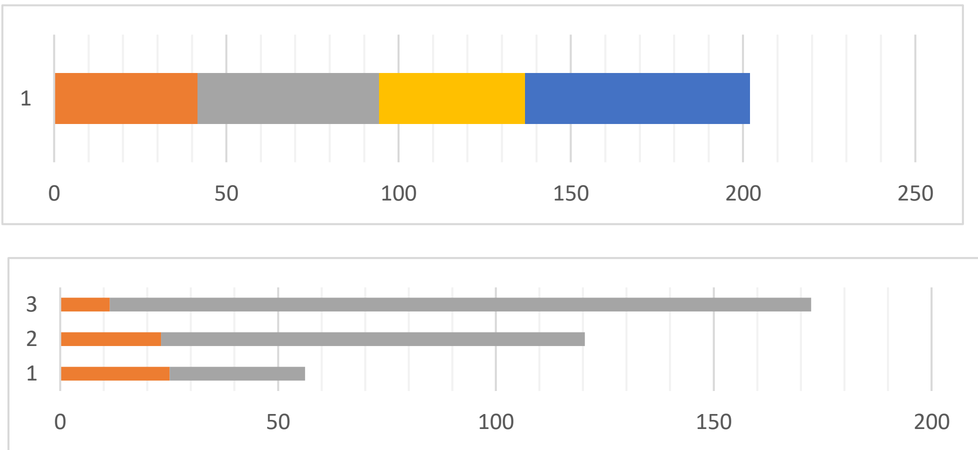
Door	Truck
0	1, 3, 2, 16
1	6, 12, 4, 18, 9
2	0, 15, 13, 5
3	19, 17, 4
4	10, 7, 11, 8



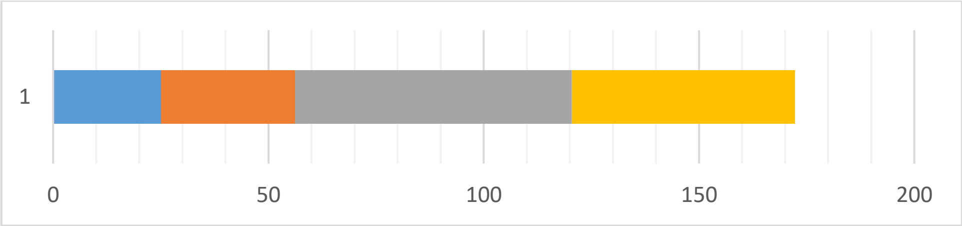
Shape 4.1a, b: Time of service in door 0 and arrival time to end of service assigned trucks

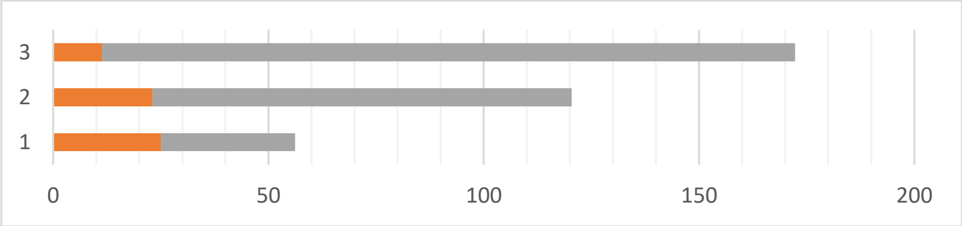


Shape 4.2a, b: Time of service in door 1 and arrival time to end of service assigned trucks

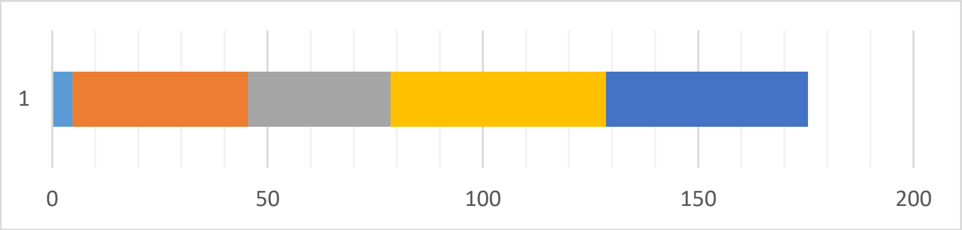


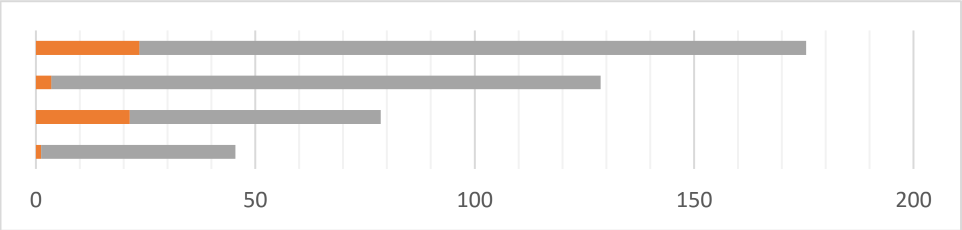
Shape 4.3a, b: Time of service in door 2 and arrival time to end of service assigned trucks





Shape 4.4a, b: Time of service in door 3 and arrival time to end of service assigned trucks



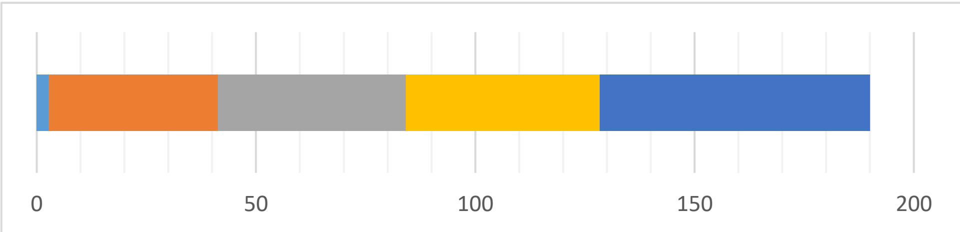


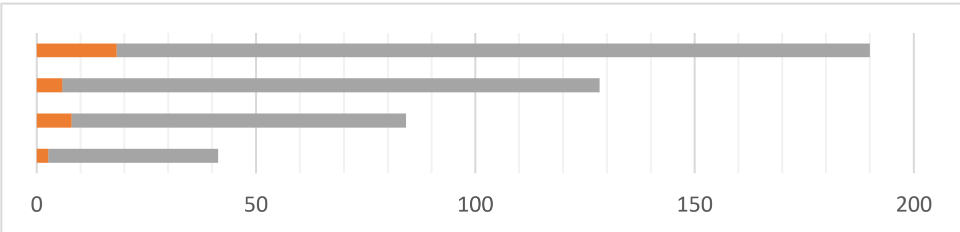
Shape 4.5a, b: Time of service in door 4 and arrival time to end of service assigned trucks

4.3 Results model 1-4a-5b

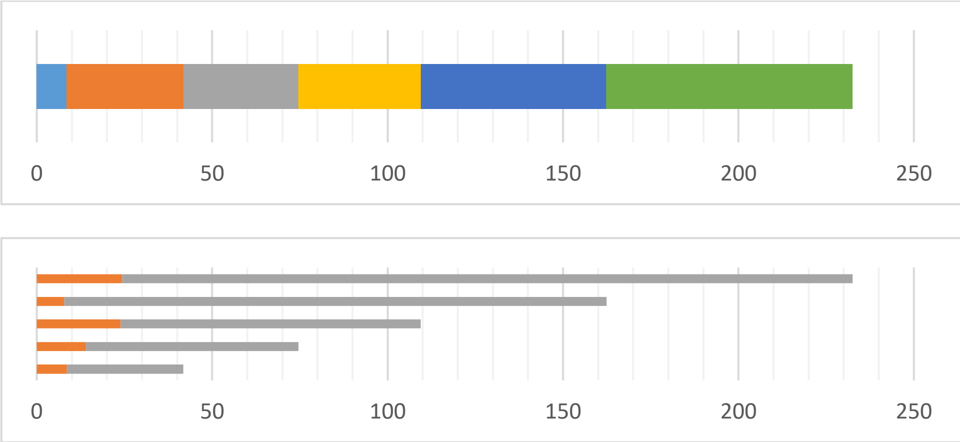
Table 4.3: Assignment of inbound trucks to inbound doors

Door	Truck
0	1, 3, 2, 16
1	4, 6, 18, 12, 9
2	0, 15, 11
3	13, 19, 14, 17
4	10, 7, 5, 8

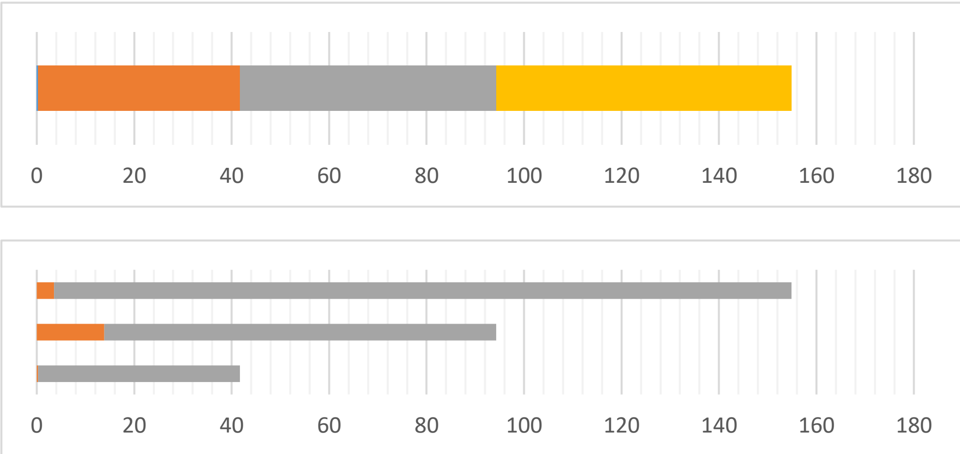




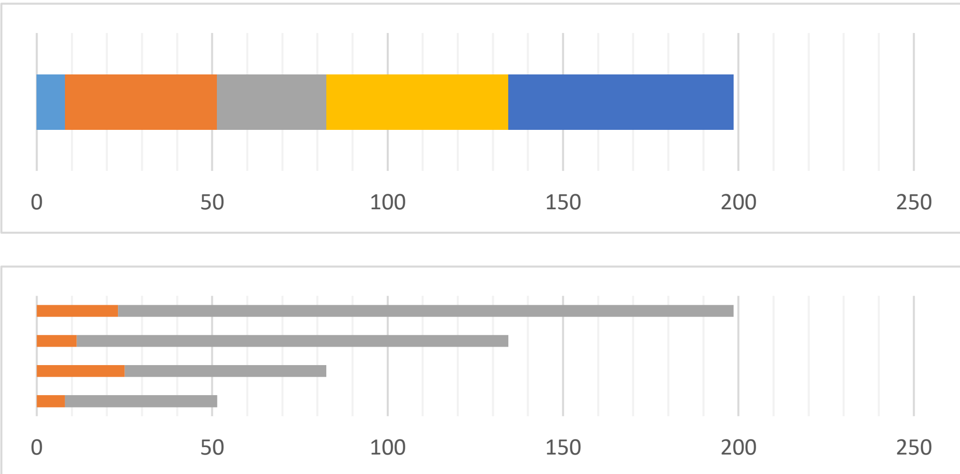
Shape 4.6a, b: Time of service in door 0 and arrival time to end of service assigned trucks



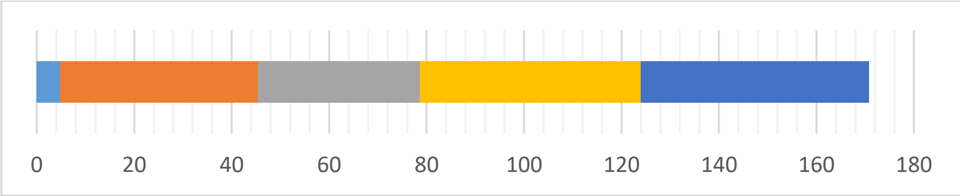
Shape 4.7a, b: Time of service in door 1 and arrival time to end of service assigned trucks

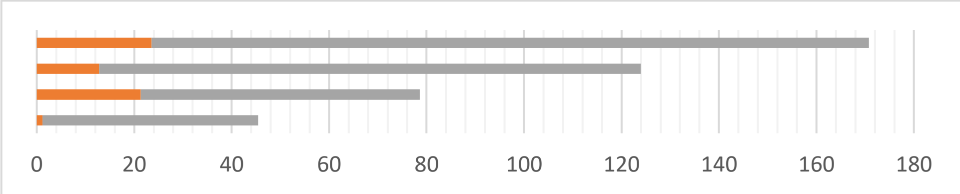


Shape 4.8a, b: Time of service in door 2 and arrival time to end of service assigned trucks



Shape 4.9a, b: Time of service in door 3 and arrival time to end of service assigned trucks



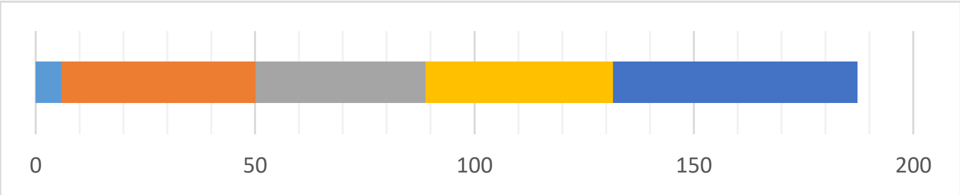


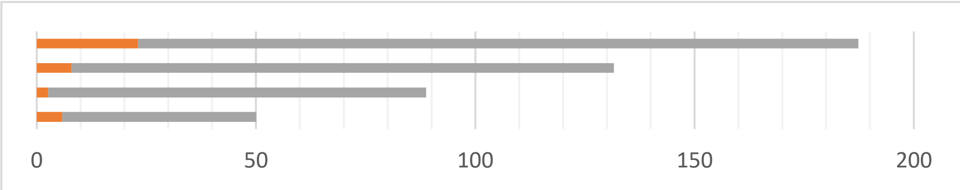
Shape 4.10a, b: Time of service in door 4 and arrival time to end of service assigned trucks

4.4 Results model 1-4a-5c

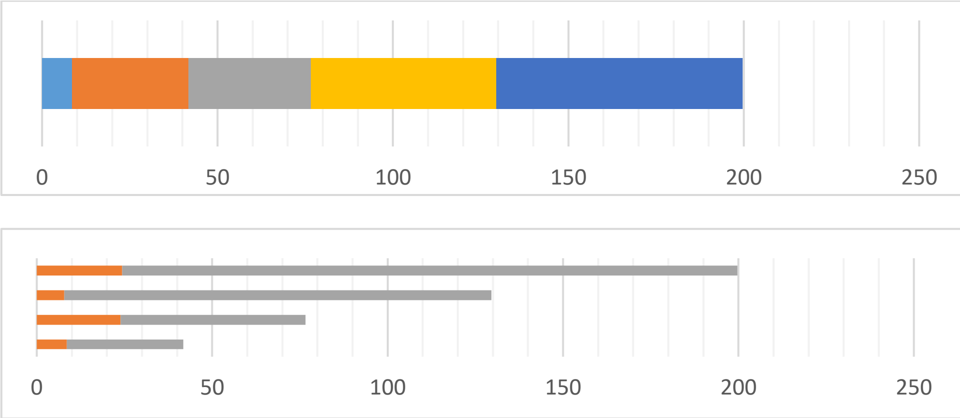
Table 4.4: Assignment of inbound trucks to inbound doors

Door	Truck
0	2, 1, 3, 17
1	4, 18, 12, 9
2	0, 13, 15, 11
3	6, 19, 14, 16
4	10, 7, 5, 8

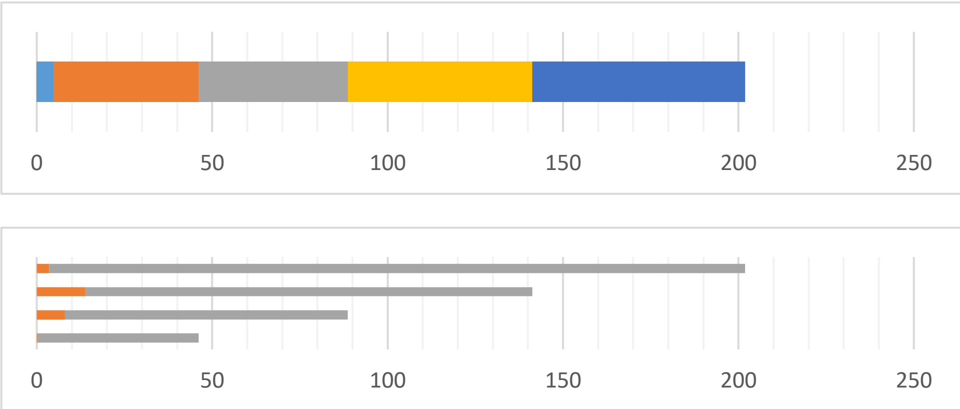




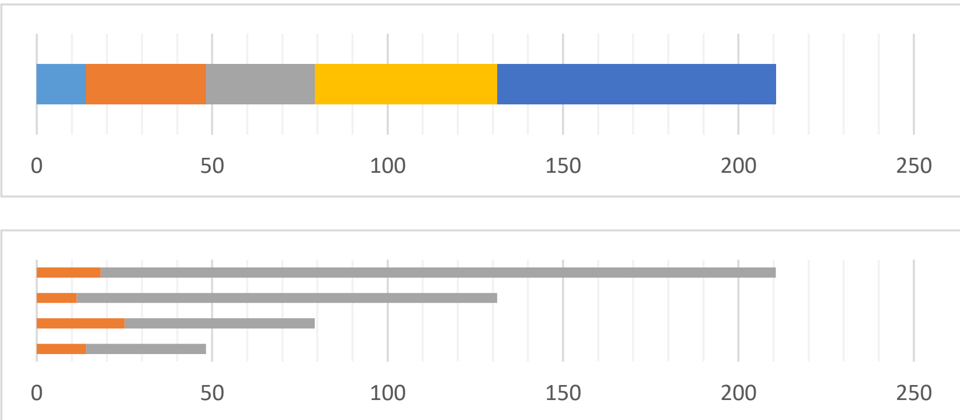
Shape 4.11a, b: Time of service in door 0 and arrival time to end of service assigned trucks



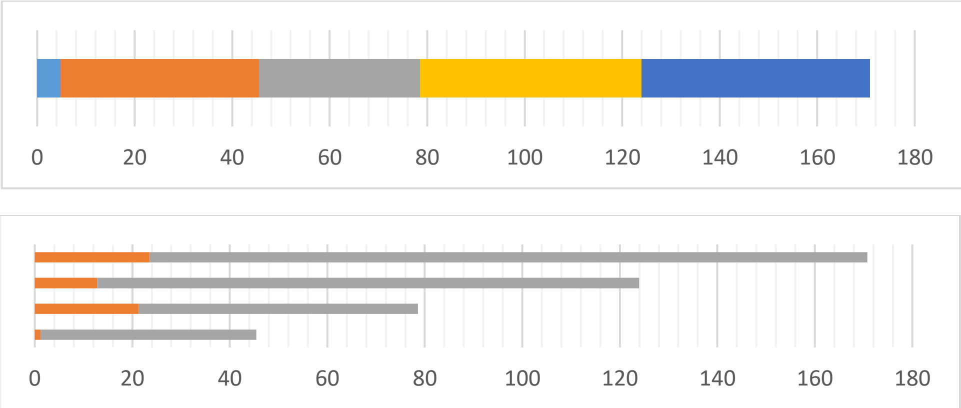
Shape 4.12a, b: Time of service in door 1 and arrival time to end of service assigned trucks



Shape 4.13a, b: Time of service in door 2 and arrival time to end of service assigned trucks



Shape 4.14a, b: Time of service in door 3 and arrival time to end of service assigned trucks

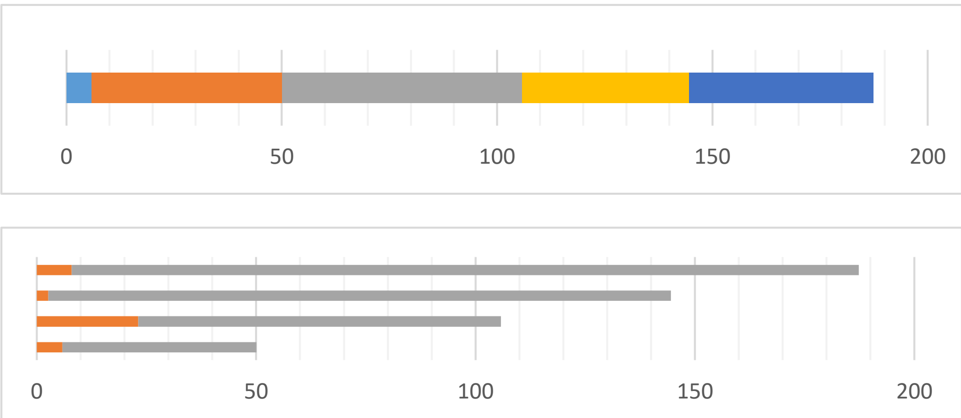


Shape 4.15a, b: Time of service in door 4 and arrival time to end of service assigned trucks

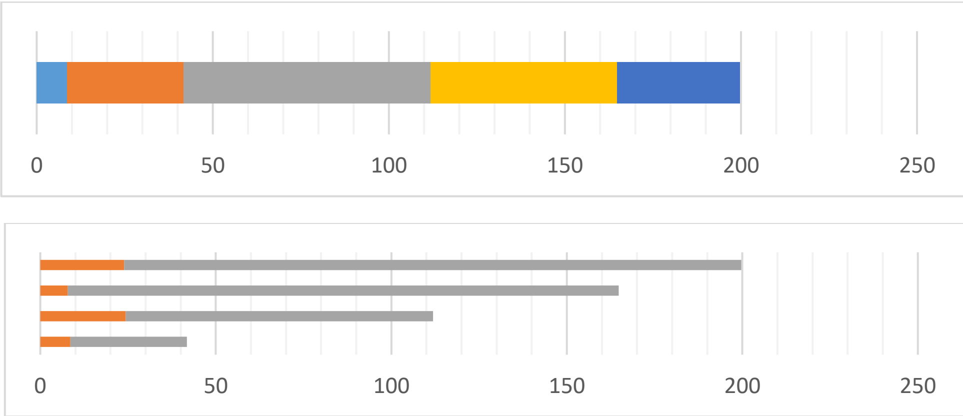
4.5 Results model 1-4b-5a

Table 4.5: Assignment of inbound trucks to inbound doors

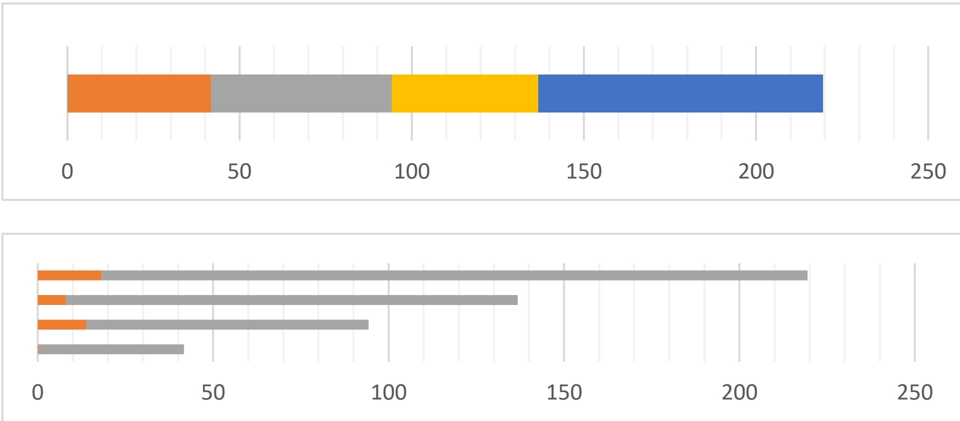
Door	Truck
0	2, 17, 1, 3
1	4, 9, 12, 18
2	0, 15, 13, 16
3	6, 19, 10, 14
4	5, 7, 11, 8



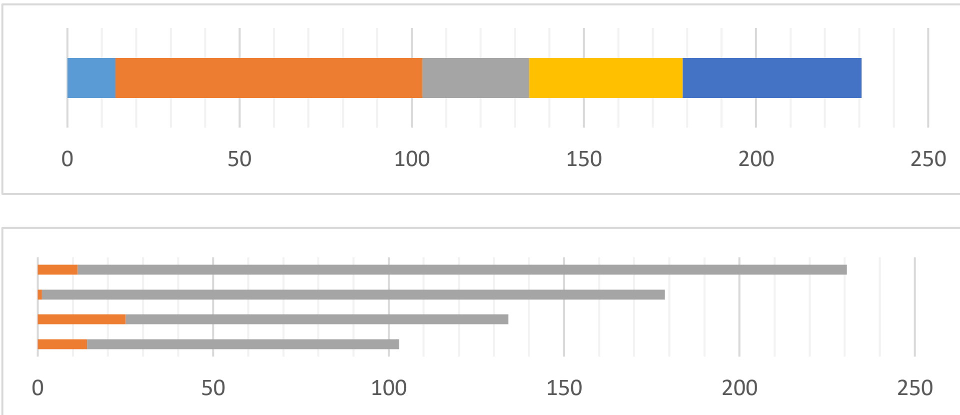
Shape 4.16a, b: Time of service in door 0 and arrival time to end of service assigned trucks



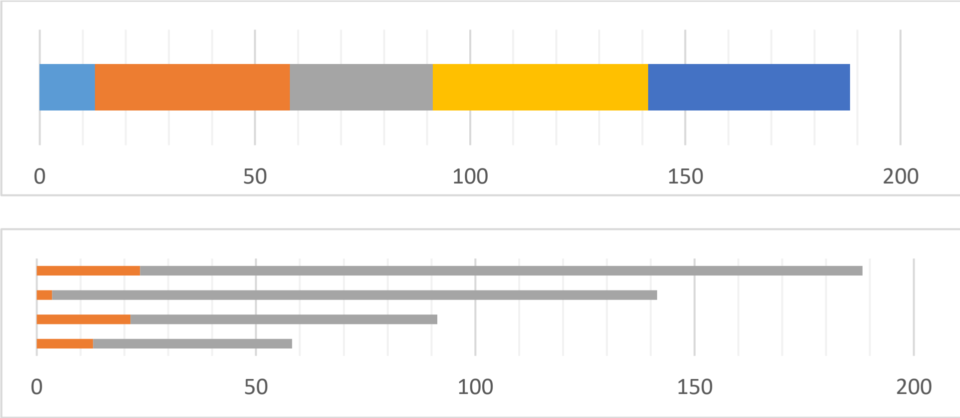
Shape 4.17a, b: Time of service in door 1 and arrival time to end of service assigned trucks



Shape 4.18a, b: Time of service in door 2 and arrival time to end of service assigned trucks



Shape 4.19a, b: Time of service in door 3 and arrival time to end of service assigned trucks

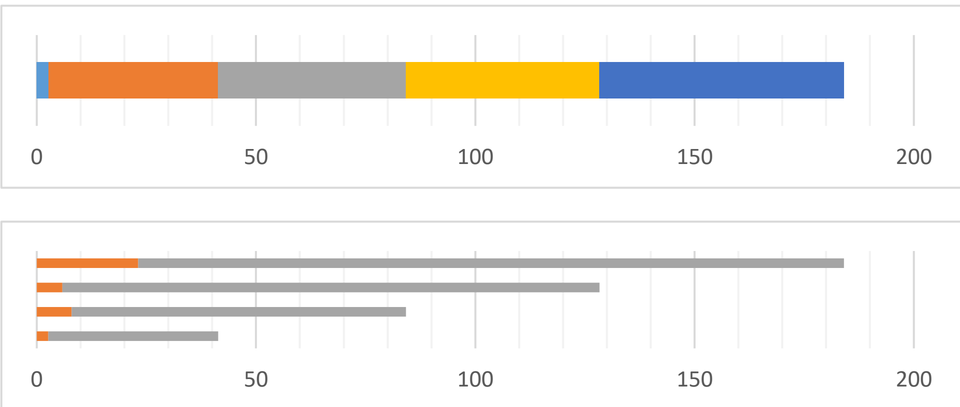


Shape 4.20a, b: Time of service in door 4 and arrival time to end of service assigned trucks

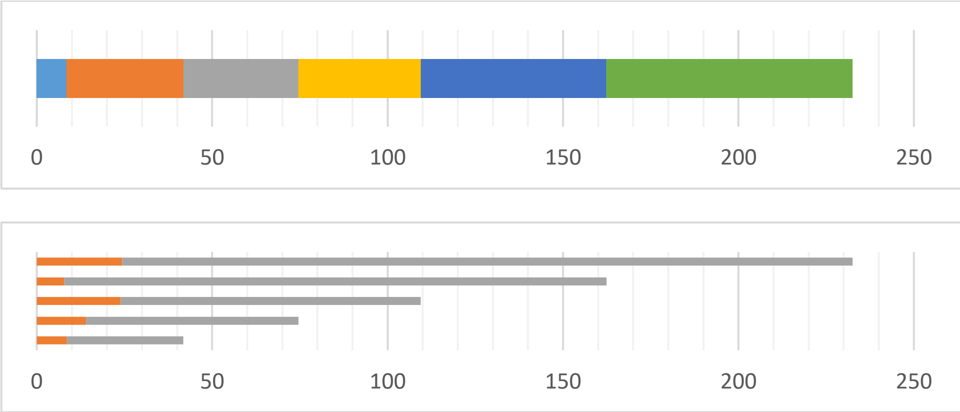
4.6 Results model 1-4b-5b

Table 4.6: Assignment of inbound trucks to inbound doors

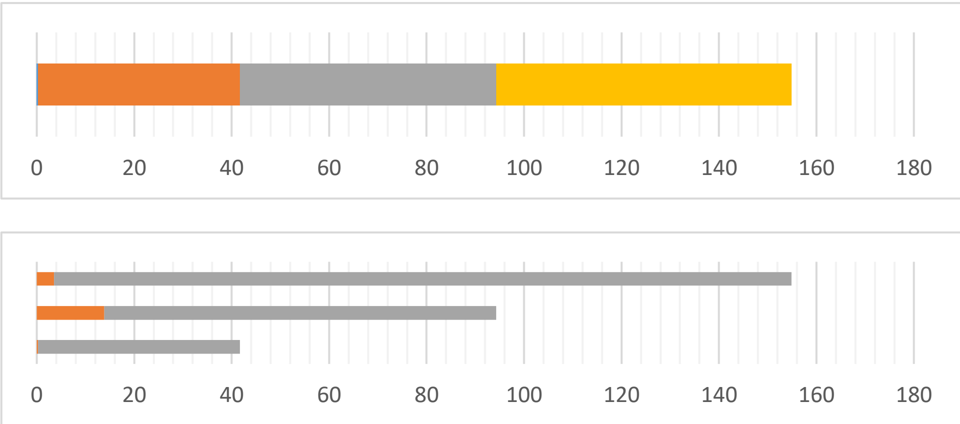
Door	Truck
0	1, 3, 2, 17
1	4, 6, 18, 12, 9
2	0, 15, 11
3	13, 19, 14, 16
4	10, 7, 5, 8



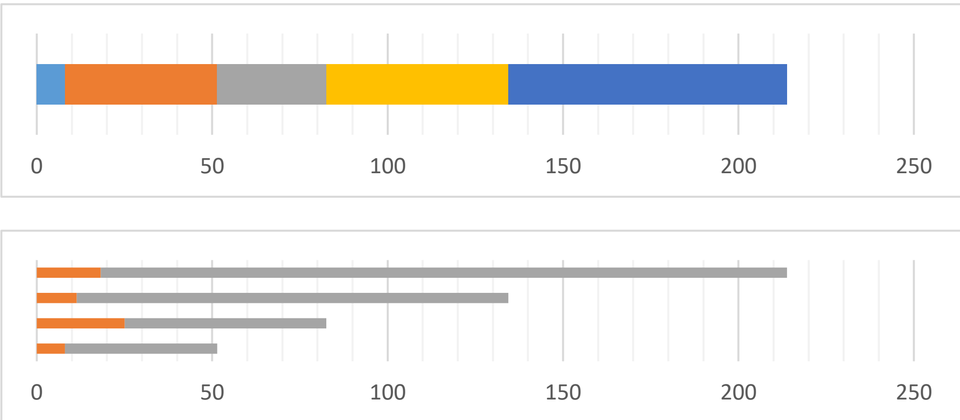
Shape 4.21a, b: Time of service in door 0 and arrival time to end of service assigned trucks



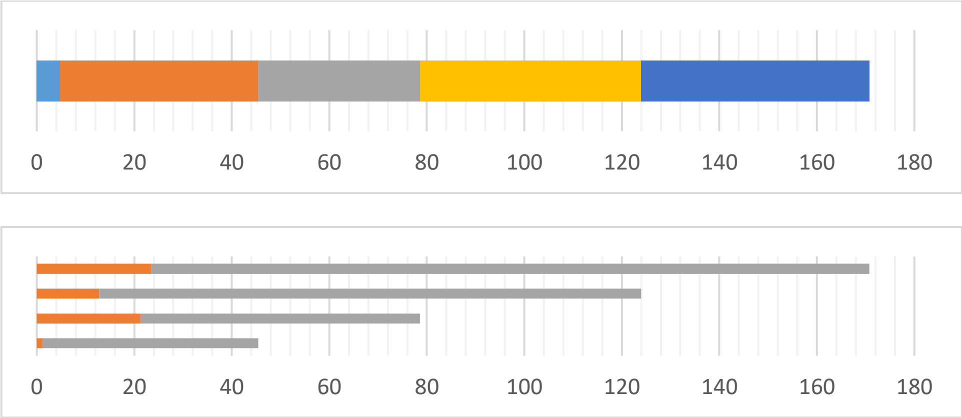
Shape 4.22a, b: Time of service in door 1 and arrival time to end of service assigned trucks



Shape 4.23a, b: Time of service in door 2 and arrival time to end of service assigned trucks



Shape 4.24a, b: Time of service in door 3 and arrival time to end of service assigned trucks

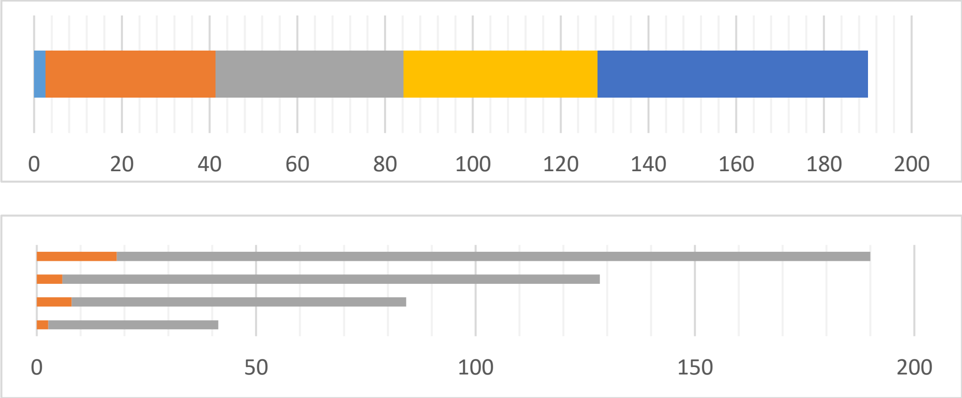


Shape 4.25a, b: Time of service in door 4 and arrival time to end of service assigned trucks

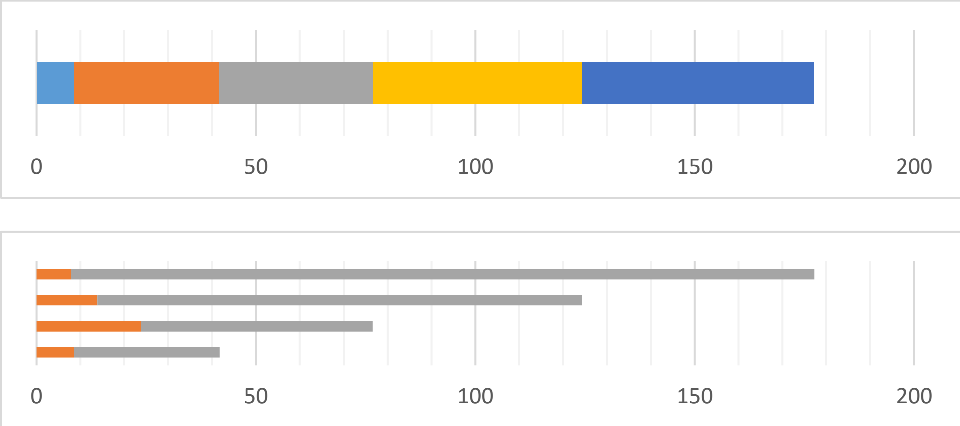
4.7 Results model 1-4b-5c

Table 4.7: Assignment of inbound trucks to inbound doors

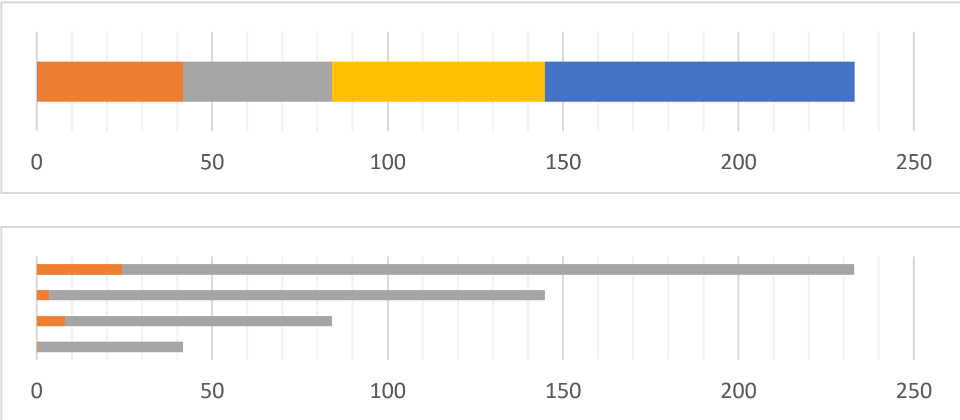
Door	Truck
0	1, 3, 2, 16
1	4, 18, 15, 12
2	0, 13, 11, 9
3	6, 19, 14, 17
4	10, 7, 5, 8



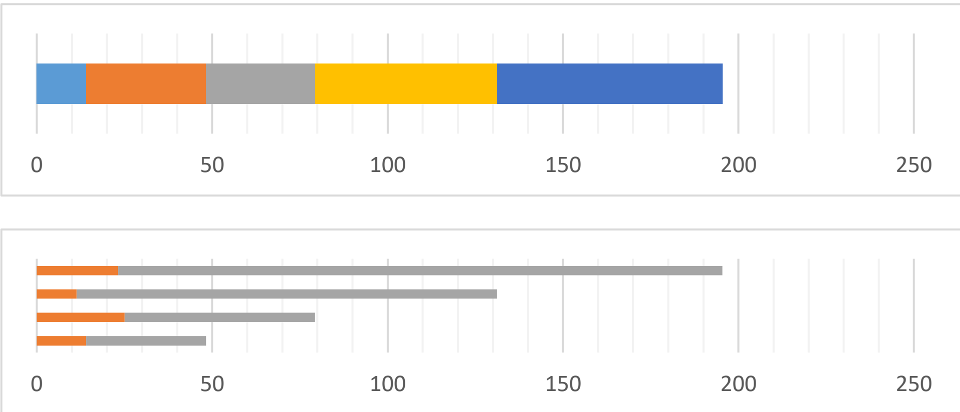
Shape 4.26a, b: Time of service in door 0 and arrival time to end of service assigned trucks



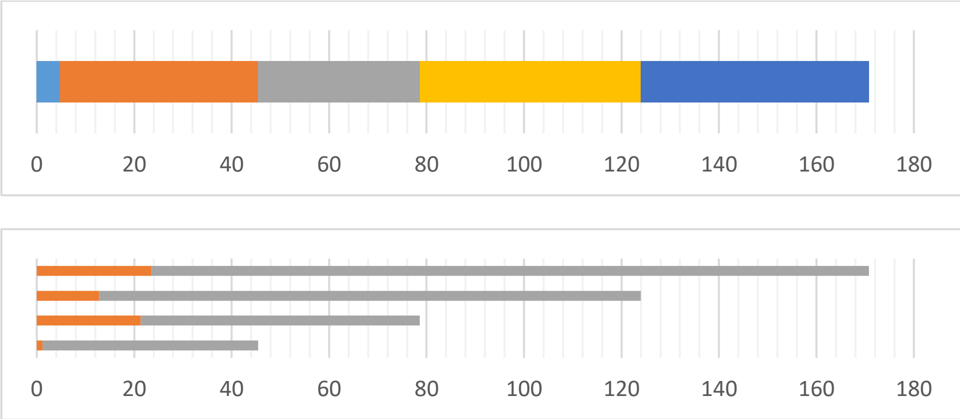
Shape 4.27a, b: Time of service in door 1 and arrival time to end of service assigned trucks



Shape 4.28a, b: Time of service in door 2 and arrival time to end of service assigned trucks



Shape 4.29a, b: Time of service in door 3 and arrival time to end of service assigned trucks

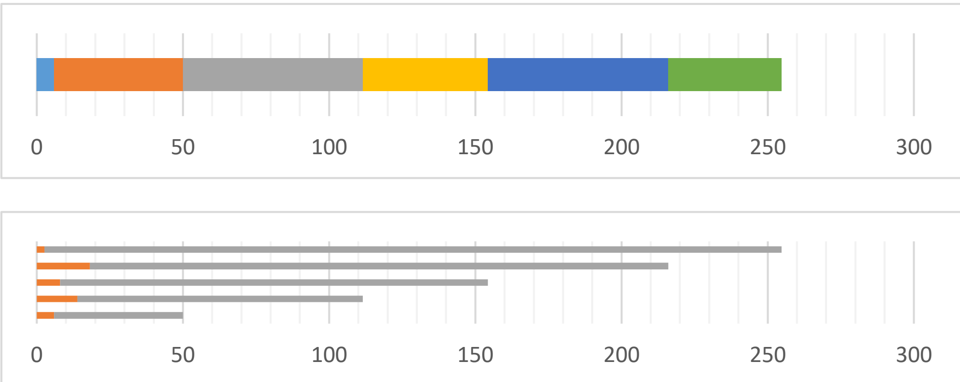


Shape 4.30a, b: Time of service in door 4 and arrival time to end of service assigned trucks

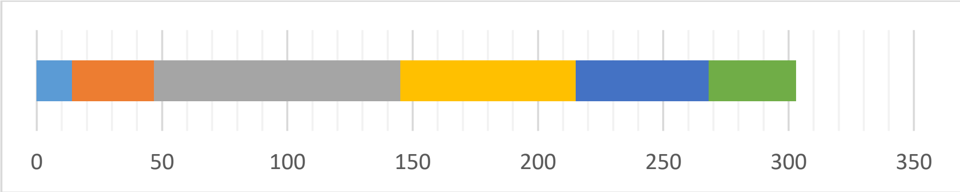
4.8 Results model 2-4a

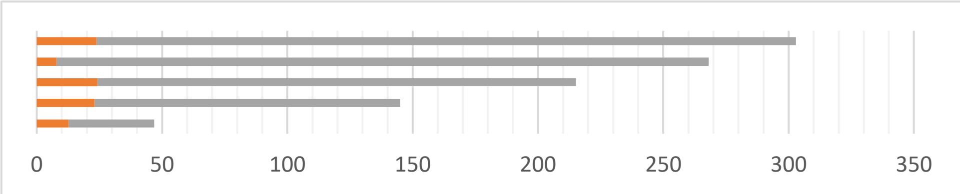
Table 4.8: Assignment of inbound trucks to inbound doors

Door	Truck
0	2, 15, 3, 16, 1
1	6, 17, 9, 12, 18
2	13, 0, 4
3	14, 19
4	5, 11, 7, 8, 10

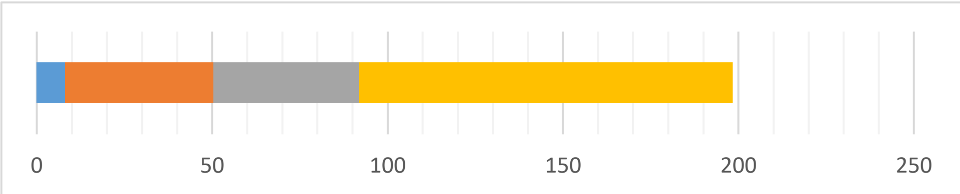


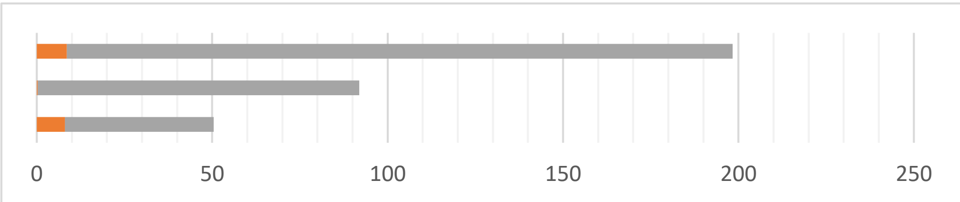
Shape 4.31a, b: Time of service in door 0 and arrival time to end of service assigned trucks



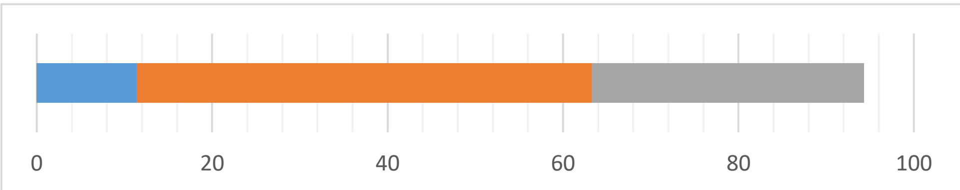


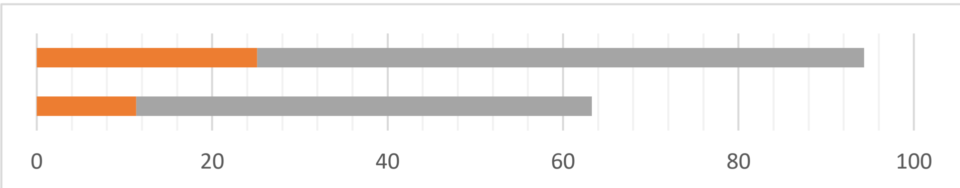
Shape 4.32a, b: Time of service in door 1 and arrival time to end of service assigned trucks



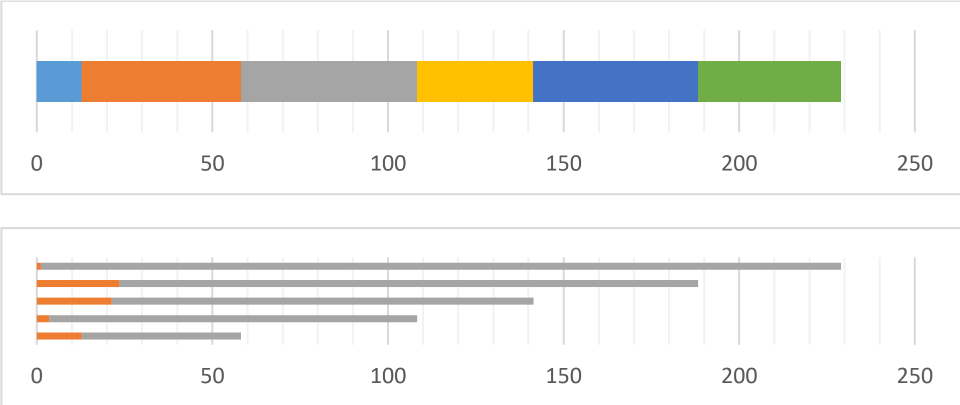


Shape 4.33a, b: Time of service in door 2 and arrival time to end of service assigned trucks





Shape 4.34a, b: Time of service in door 3 and arrival time to end of service assigned trucks

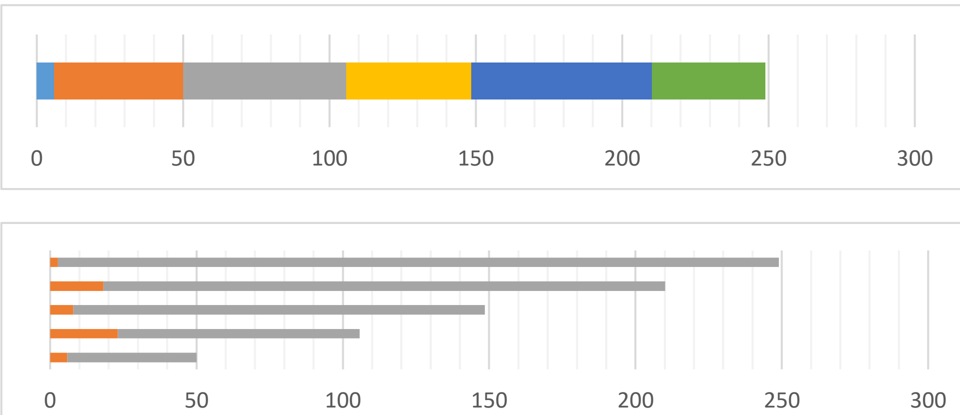


Shape 4.35a, b: Time of service in door 4 and arrival time to end of service assigned trucks

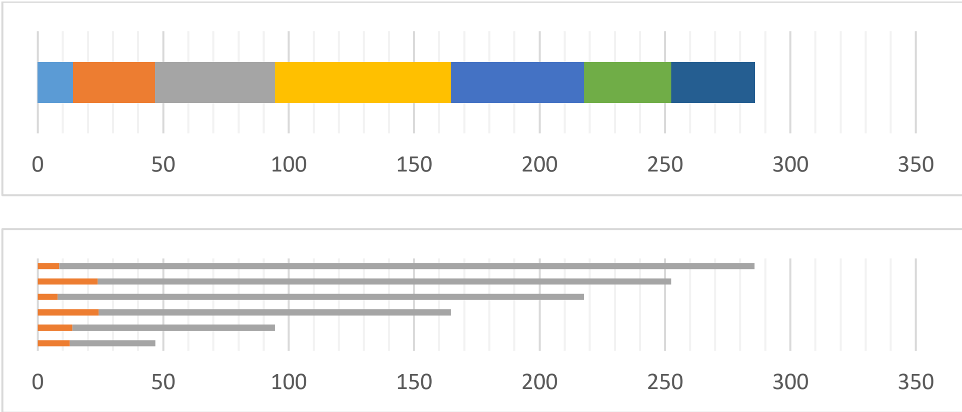
4.9 Results model 2-4b

Table 4.9: Assignment of inbound trucks to inbound doors

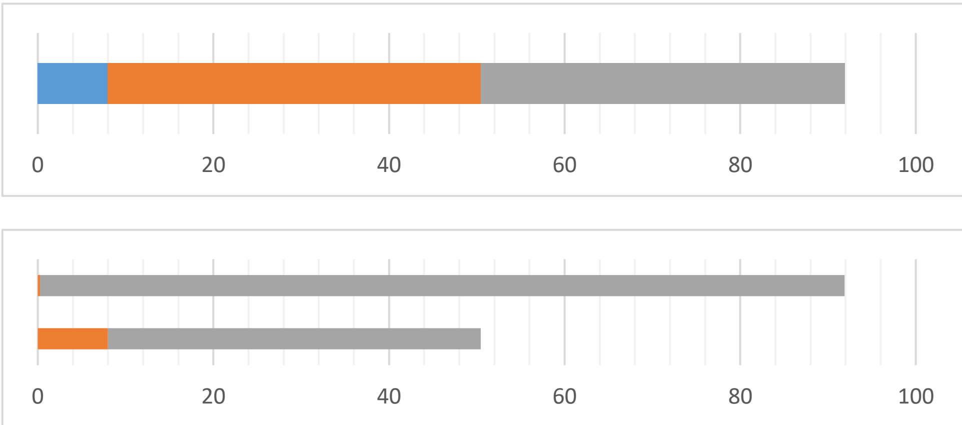
Door	Truck
0	2, 17, 3, 16, 1
1	6, 15, 9, 12, 18, 4
2	13, 0
3	14, 19
4	5, 11, 7, 8, 10



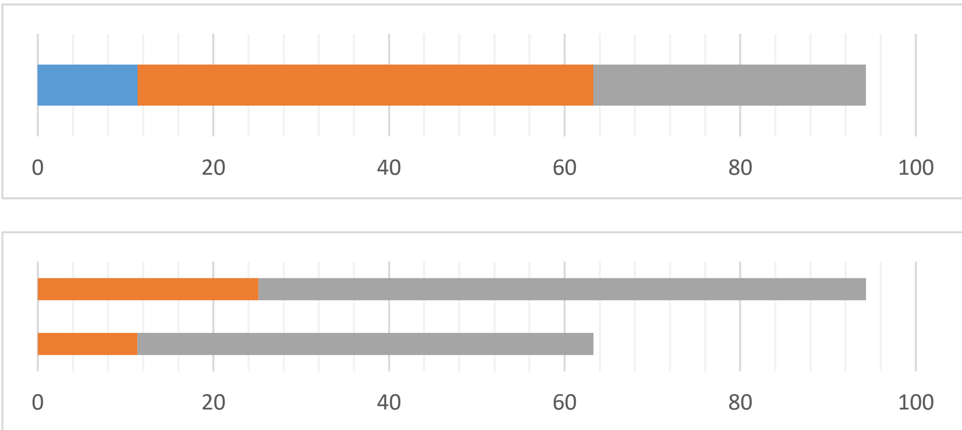
Shape 4.36a, b: Time of service in door 0 and arrival time to end of service assigned trucks



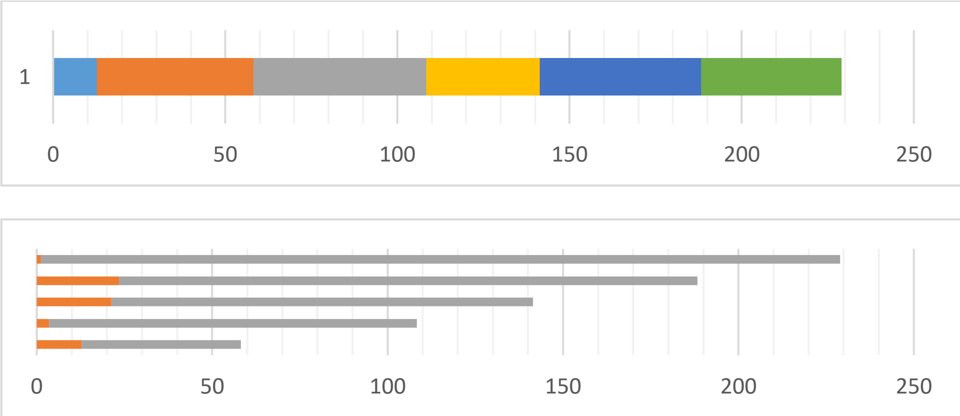
Shape 4.37a, b: Time of service in door 1 and arrival time to end of service assigned trucks



Shape 4.38a, b: Time of service in door 2 and arrival time to end of service assigned trucks



Shape 4.39a, b: Time of service in door 3 and arrival time to end of service assigned trucks

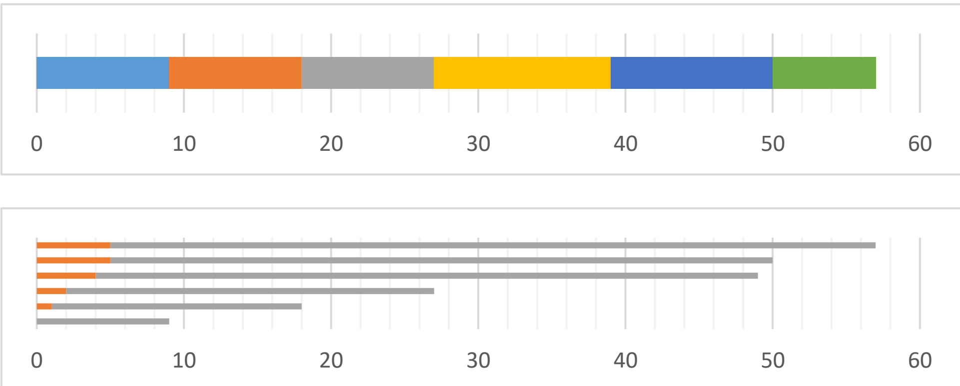


Shape 4.40a, b: Time of service in door 4 and arrival time to end of service assigned trucks

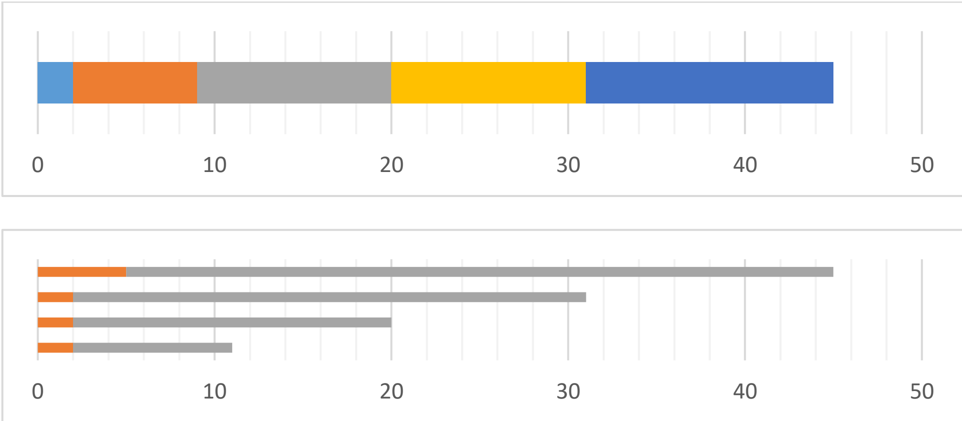
4.10 Results model 3-obj1-13a

Table 4.10: Assignment of inbound trucks to inbound doors

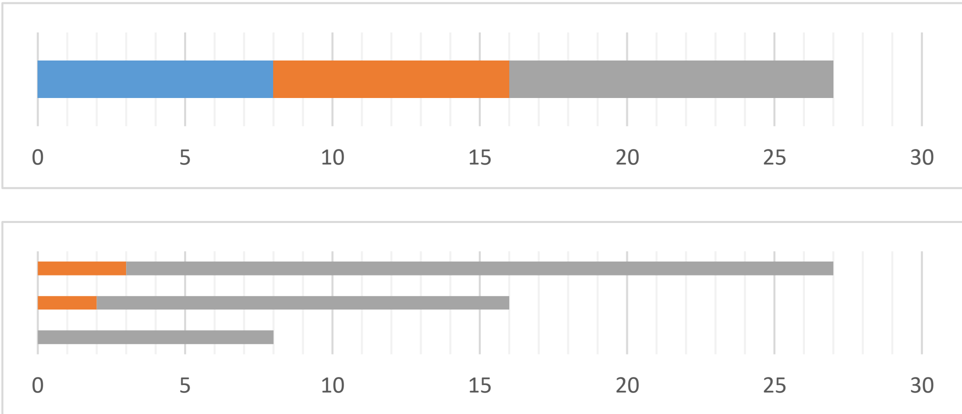
Door	Truck
0	1, 2, 3, 16, 17, 18
1	12, 4, 14, 9
2	0, 13, 15
3	10, 6, 19
4	11, 5, 7, 8



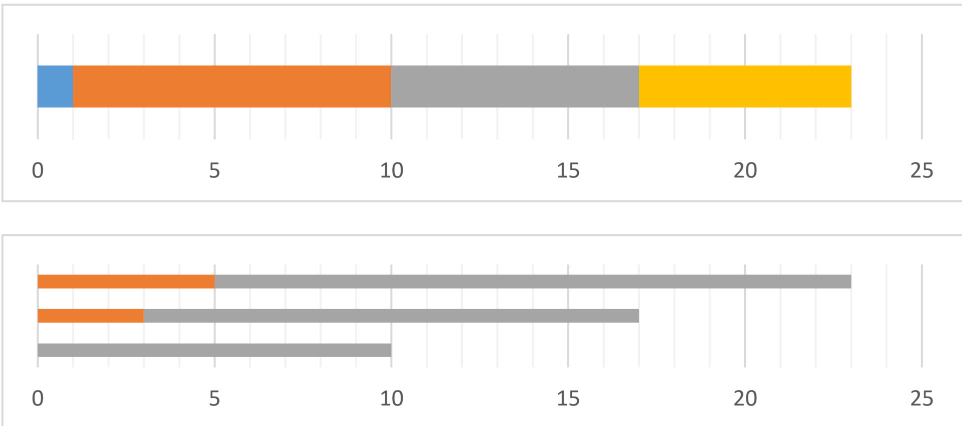
Shape 4.41a, b: Time of service in door 0 and arrival time to end of service assigned trucks



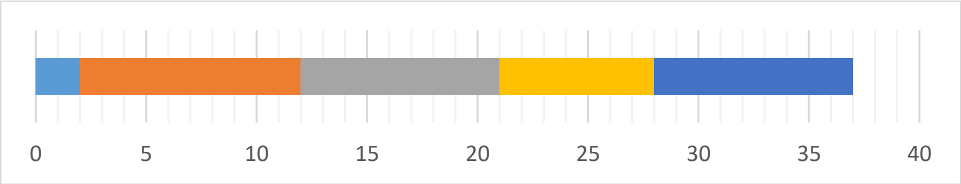
Shape 4.42a, b: Time of service in door 1 and arrival time to end of service assigned trucks

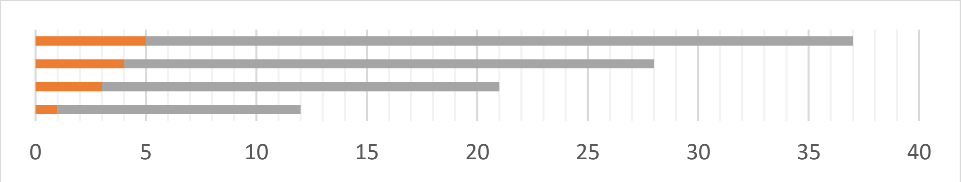


Shape 4.43a, b: Time of service in door 2 and arrival time to end of service assigned trucks



Shape 4.44a, b: Time of service in door 3 and arrival time to end of service assigned trucks



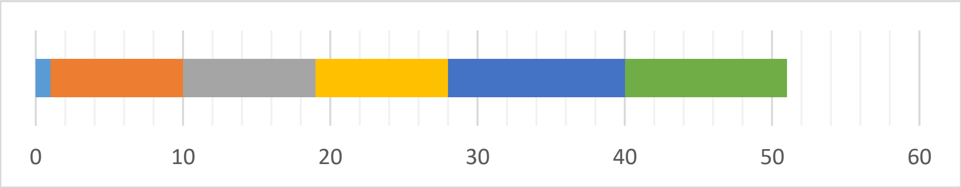


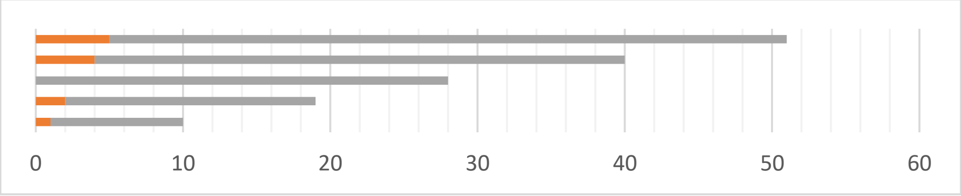
Shape 4.45a, b: Time of service in door 4 and arrival time to end of service assigned trucks

4.11 Results model 3-obj1-13b

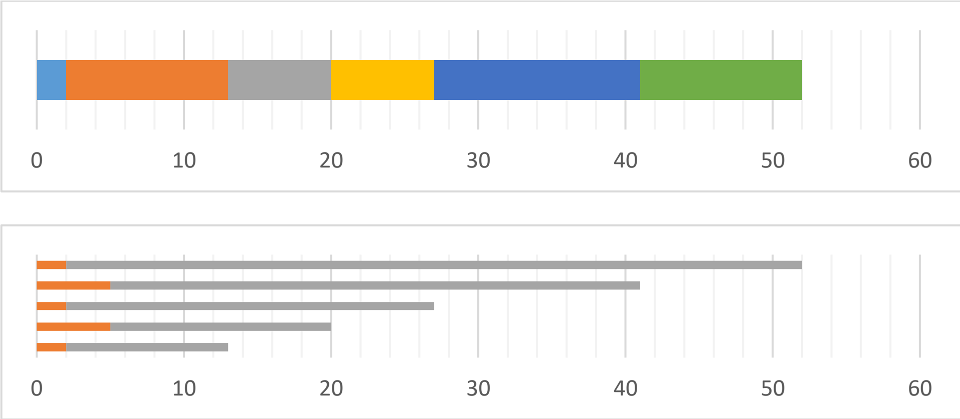
Table 4.11: Assignment of inbound trucks to inbound doors

Door	Truck
0	2, 3, 1, 16, 17
1	12, 18, 4, 9, 14
2	13, 0, 15
3	6, 10, 19
4	8, 11, 7, 5

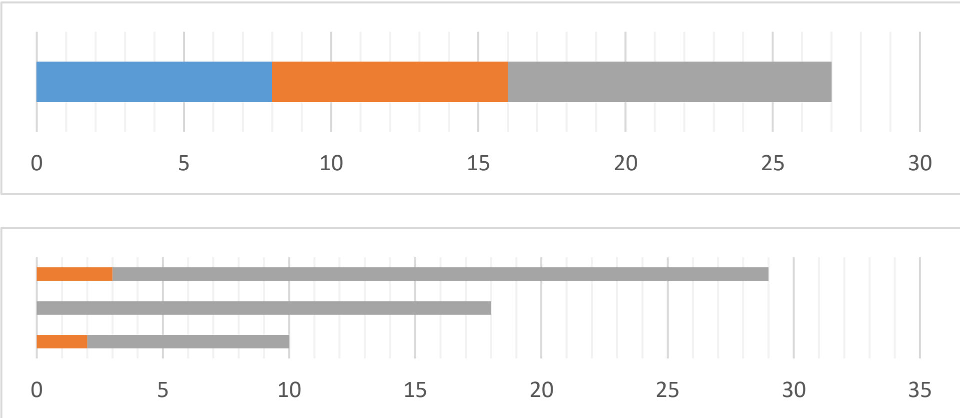




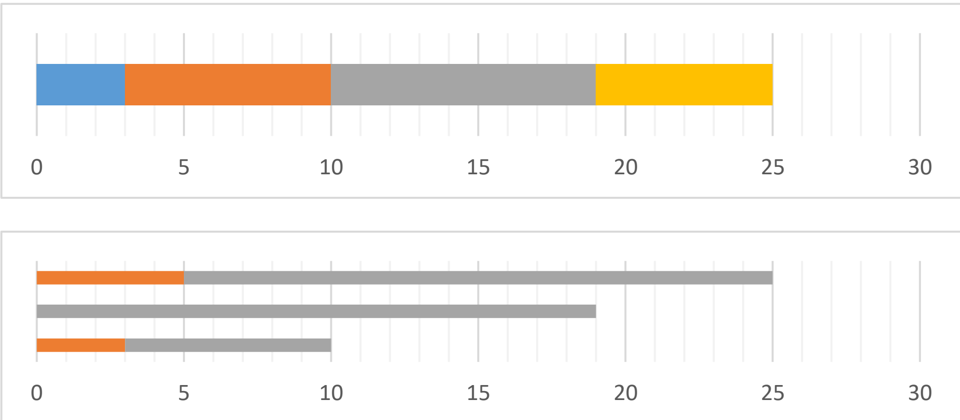
Shape 4.46a, b: Time of service in door 0 and arrival time to end of service assigned trucks



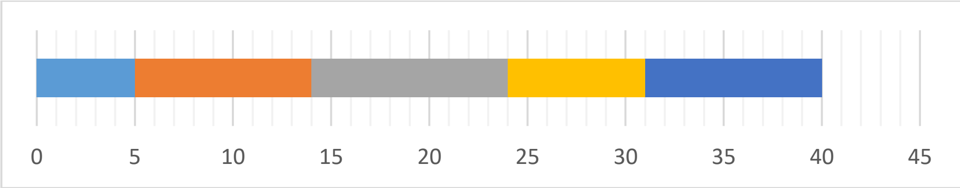
Shape 4.47a, b: Time of service in door 1 and arrival time to end of service assigned trucks

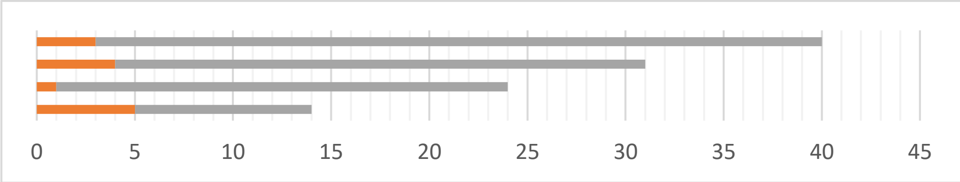


Shape 4.48a, b: Time of service in door 2 and arrival time to end of service assigned trucks



Shape 4.49a, b: Time of service in door 3 and arrival time to end of service assigned trucks



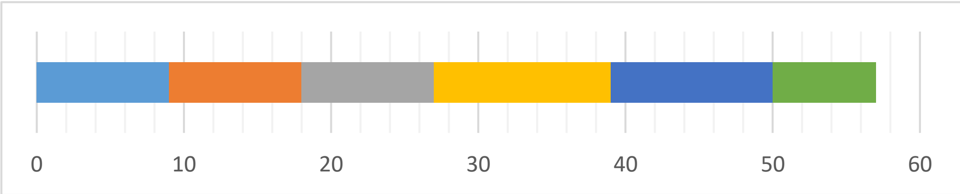


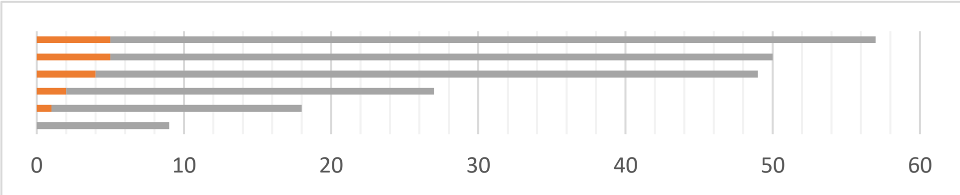
Shape 4.50a, b: Time of service in door 4 and arrival time to end of service assigned trucks

4.12 Results model 3-obj2-13a

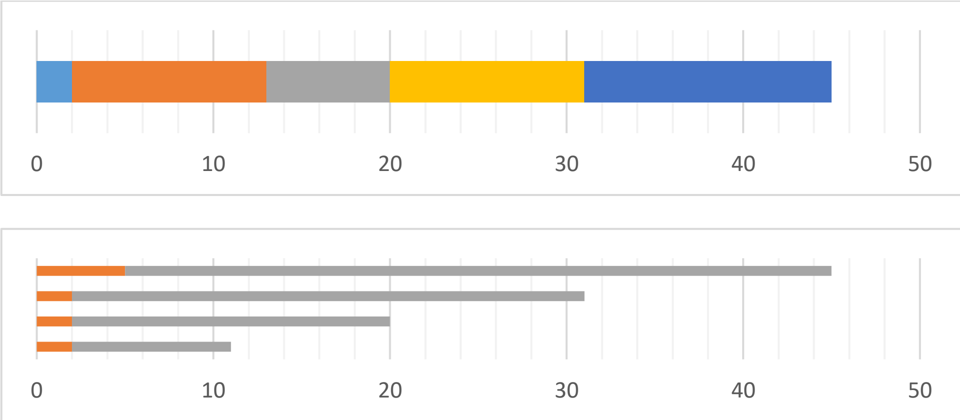
Table 4.12: Assignment of inbound trucks to inbound doors

Door	Truck
0	1, 2, 3, 16, 17, 18
1	12, 4, 14, 9
2	0, 13, 15
3	10, 6, 19
4	11, 5, 7, 8

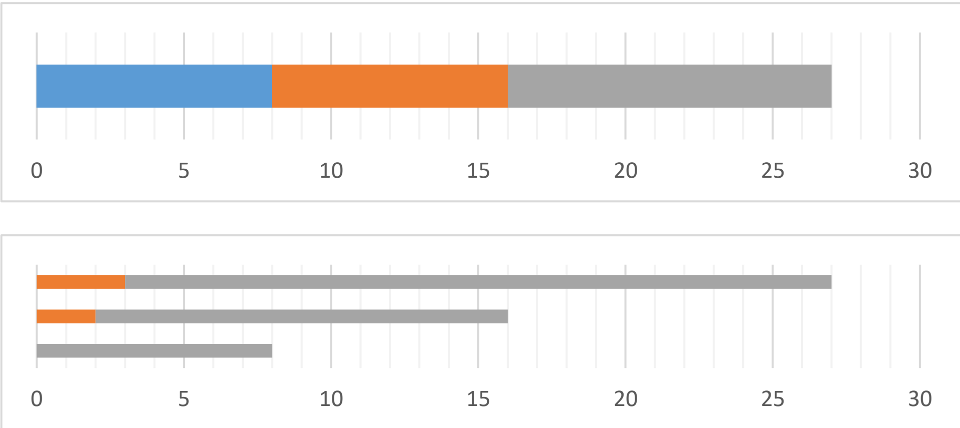




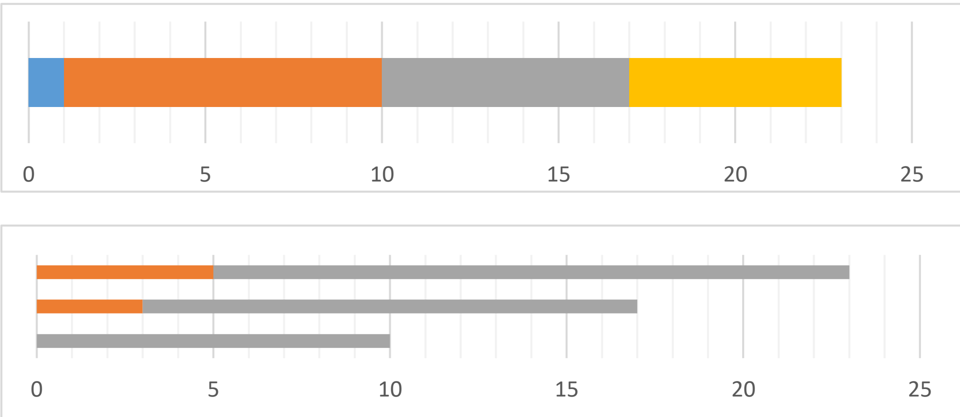
Shape 4.51a, b: Time of service in door 0 and arrival time to end of service assigned trucks



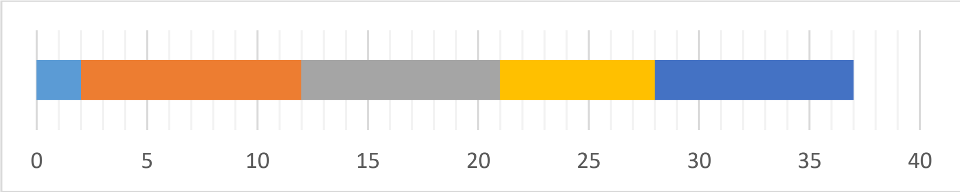
Shape 4.52a, b: Time of service in door 1 and arrival time to end of service assigned trucks

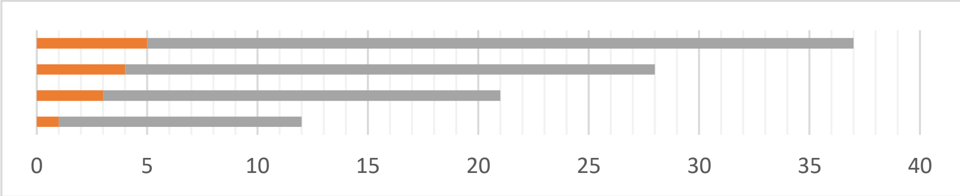


Shape 4.53a, b: Time of service in door 2 and arrival time to end of service assigned trucks



Shape 4.54a, b: Time of service in door 3 and arrival time to end of service assigned trucks



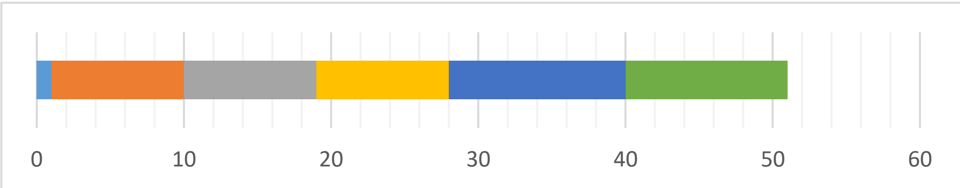


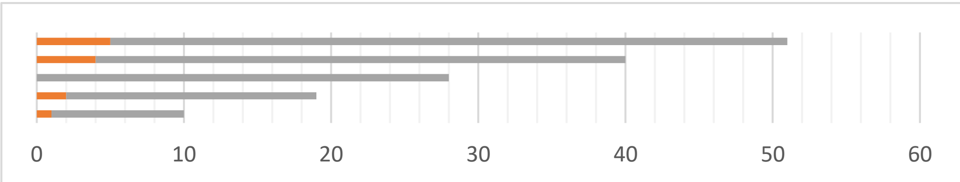
Shape 4.55a, b: Time of service in door 4 and arrival time to end of service assigned trucks

4.13 Results model 3-obj2-13b

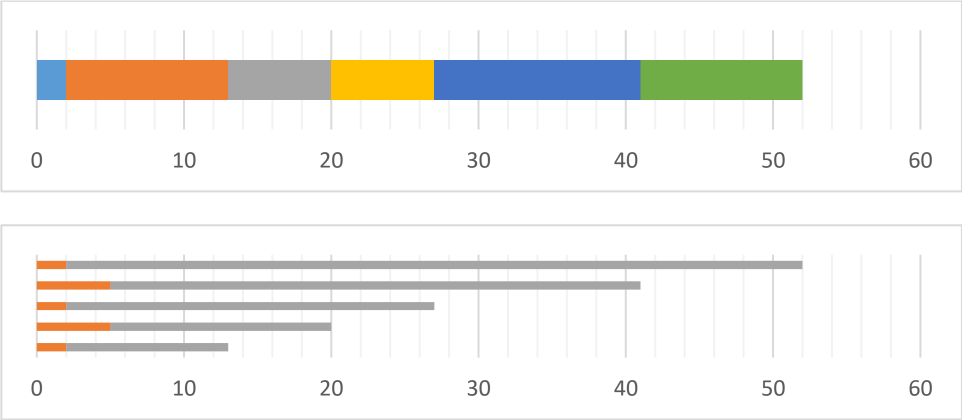
Table 4.13: Assignment of inbound trucks to inbound doors

Door	Truck
0	2, 3, 1, 16, 17
1	12, 18, 4, 9, 14
2	13, 0, 15
3	6, 10, 19
4	8, 11, 7, 5

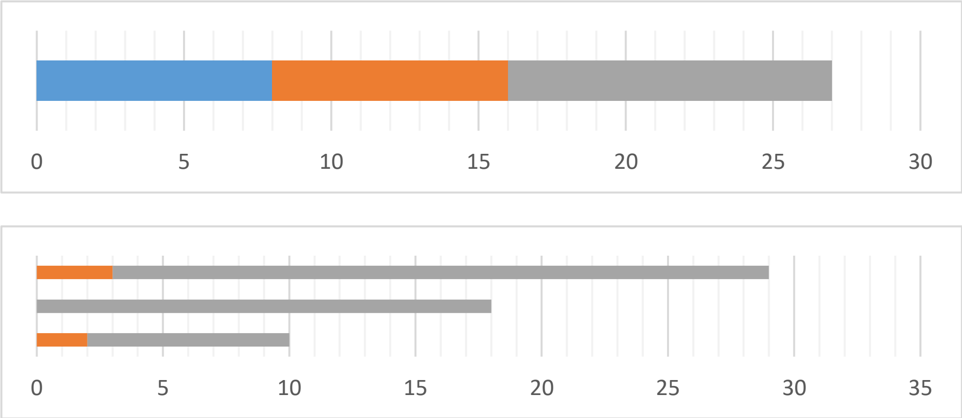




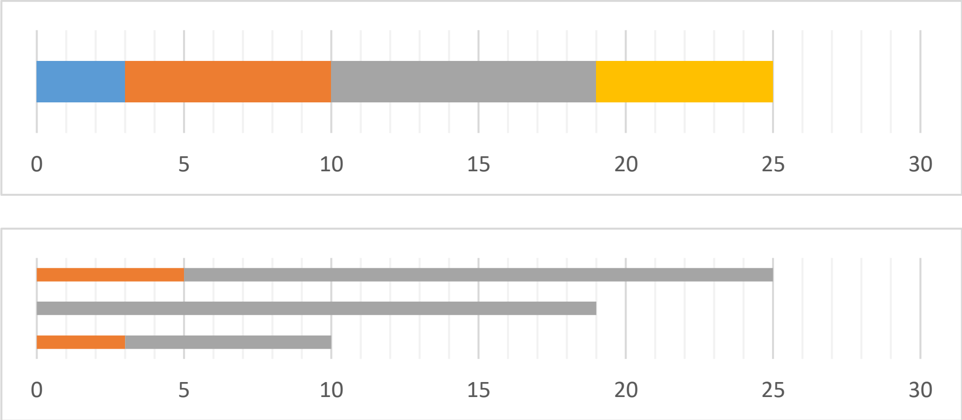
Shape 4.56a, b: Time of service in door 0 and arrival time to end of service assigned trucks



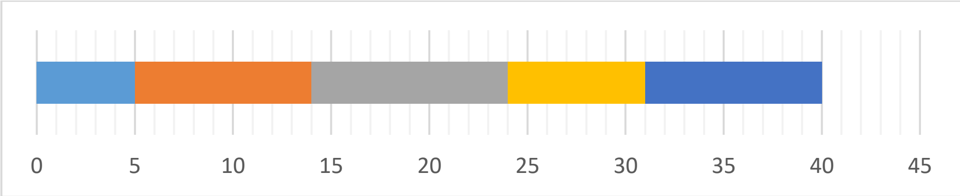
Shape 4.57a, b: Time of service in door 1 and arrival time to end of service assigned trucks

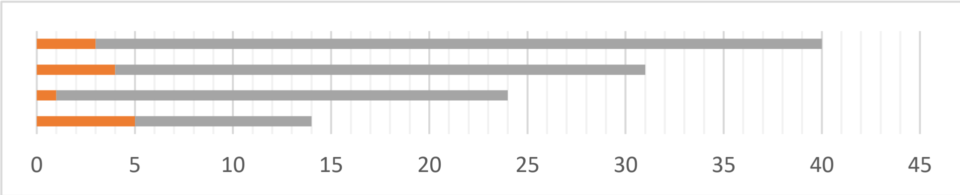


Shape 4.58a, b: Time of service in door 2 and arrival time to end of service assigned trucks



Shape 4.59a, b: Time of service in door 3 and arrival time to end of service assigned trucks



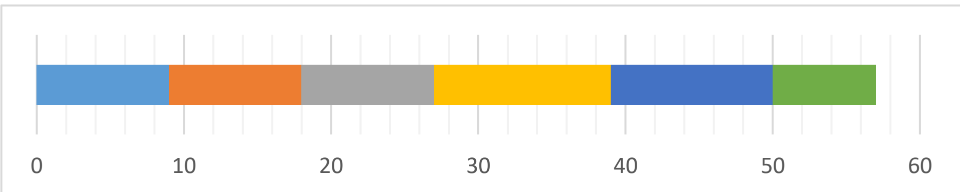


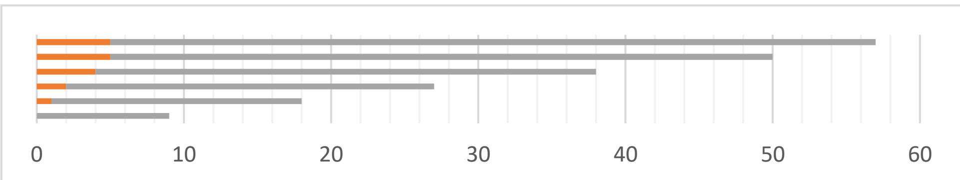
Shape 4.60a, b: Time of service in door 4 and arrival time to end of service assigned trucks

4.14 Results model 4

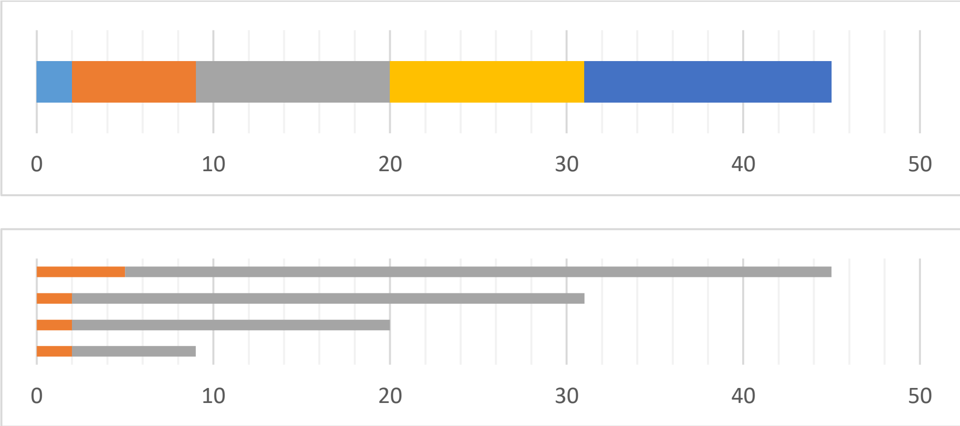
Table 4.14: Assignment of inbound trucks to inbound doors

Door	Truck
0	1, 2, 3, 16, 17, 18
1	4, 12, 14, 19
2	0, 13, 15
3	10, 6, 19
4	11, 5, 7, 8

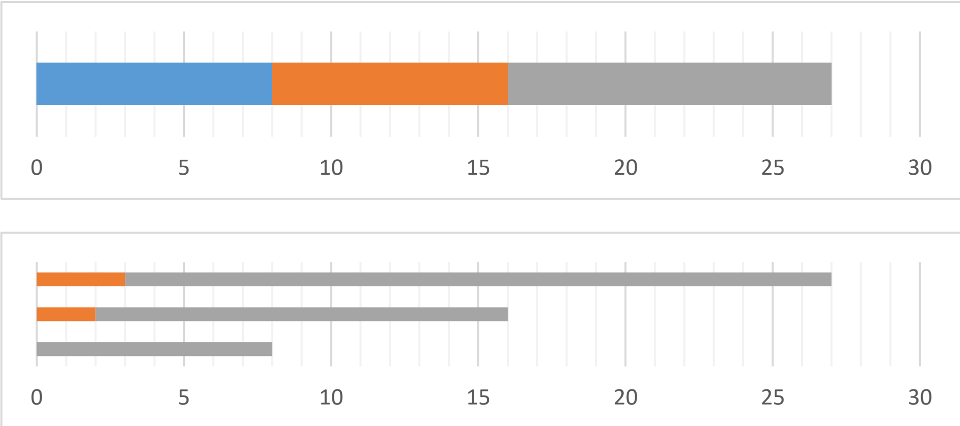




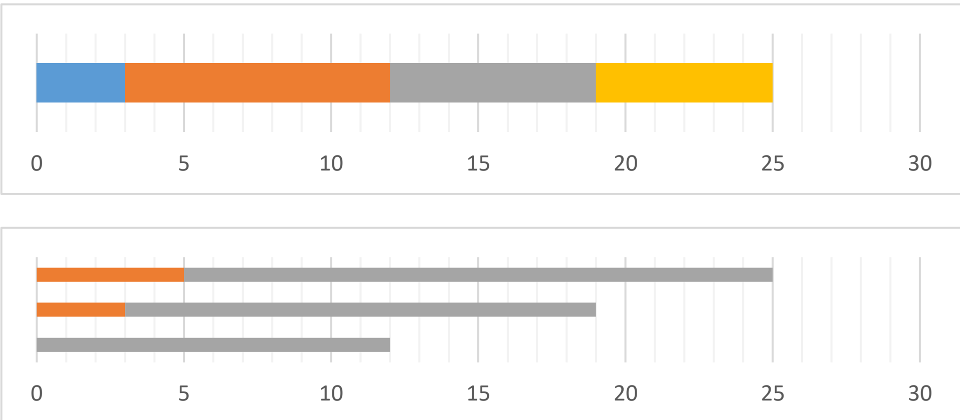
Shape 4.61a, b: Time of service in door 0 and arrival time to end of service assigned trucks



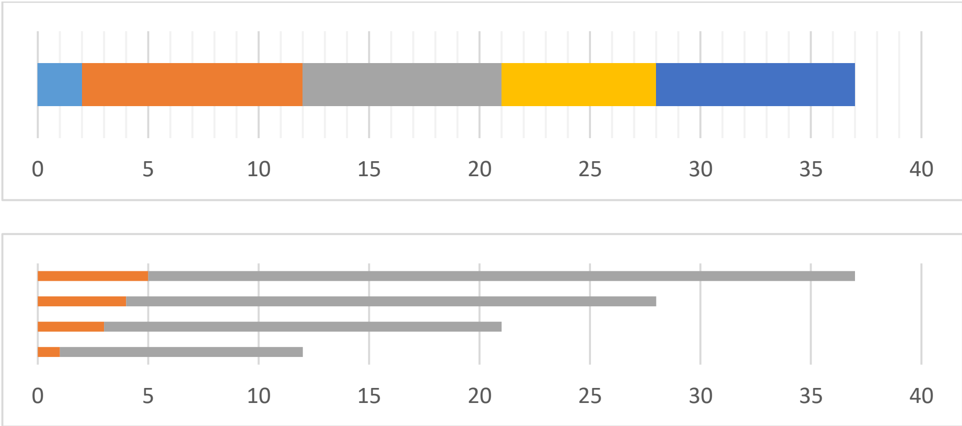
Shape 4.62: Time of service in door 1 and arrival time to end of service assigned trucks



Shape 4.63: Time of service in door 2 and arrival time to end of service assigned trucks



Shape 4.64: Time of service in door 3 and arrival time to end of service assigned trucks



Shape 4.65: Time of service in door 4 and arrival time to end of service assigned trucks

9. CONCLUSION

As can be noted, numerical results obtained by using all the formulated mathematical models showed that model 1 is giving optimal solution but the other three models need smaller CPU time to solve the problem, instantaneously in that case. The solutions of all models are compared by calculating the value based on the objective function of model 1. The fact that models 2, 3 and 4 give solution instantaneously is useful in case of a cross docking facility in which the philosophy is based on speed and saving time and money. Model 2 gives by far the worst solution which means that this model would not be preferred. On the other hand, a 10% difference in calculated values between model 1 and models 3 and 4 is also big but not prohibitive in cases that time counts. Perhaps model 1 would give a solution near to those of models 3 and 4 if its debugging time was smaller. But this needs more study to be answered.

Despite the fact that all models seem to be similar, the bigger difference between them is the objective function. A study that would focus on optimizing the objective function of models 3 and 4 might give better results. Also, model 1 could be helped by adding some valid inequalities in order the new code could give optimal solution in smaller CPU time which means that would drive model 1 to give an even better solution.

It is clear that cross docking facilities is the future in logistics and all it is fact that more and more researchers turn their attention to this direction. There are a lot of opportunities as cross docking still has many problems to be solved. All the algorithms presented until now focus on one problem each time, which means that all need further improvement and that should combine more than one. The assignment problem seems to be the bigger obstacle in order to minimize the cost in cross docking facilities and maximize the profit which is desideratum in modern businesses.

REFERENCES

A. Books and articles

Agustina D, Lee CKM, Piplani R., 2010. *A review: mathematical models for cross dock planning.* International Journal of Engineering Business Management. 2(2), 47–54.

Alpan G., Larbi R., Penz B., 2011. *A bounded dynamic programming approach to schedule operations in a crossdocking platform*. Computers & Industrial Engineering. 60(3), 385–96.

Boysen N., Fleidner M., 2006. *Cross dock scheduling: Classification, literature review and research agenda.* Review Article. Omega. Vol. 38, No. 6, pp. 413-422.

Bozer Y. A., Carlo H. J., 2008. *Optimizing inbound and outbound door assignments in less-than-truckload crossdocks.* IIE Transactions. Vol. 40, No. 11, pp. 1007-18.

Cook R. L., Gibson B., MacCurdy D., 2005. A lean approach to crossdocking. Supply Chain Management Review. 9(2), 54–9.

Galbreth M. R., Hill J. A., Handley S., 2008. *An investigation of the value of cross-docking for supply chain management.* Journal of Business Logistics, 29(1), 225–39.

Golias M.M., Saharidis G.K.D., Boile M., Theofanis S., 2012. *Scheduling of inbound trucks at a cross-docking facility: bi-objective vs bi-level modeling approaches.* International Journal of Information Systems and Supply Chain Management. 5(1), 20-37.

Li Y., Lim A., Rodrigues B., 2004. *Crossdocking-JIT scheduling with time windows.* Review Article. Omega. Vol. 38, No. 6, pp. 413-422.

Miao Z, Lim A, Ma H., 2006. *Truck dock assignment problem with operational time constraint within crossdocks.* Journal of the Operational Research Society. Vol. 55, pp. 1342-51.

Pearson Specter S., 2004. *How to crossdock successfully*. Modern Materials Handling. 59(1), 42–6.

Saharidis G.K.D., Minoux M., Dallery Y., 2009. *Scheduling of loading and unloading of crude oil in a refinery using event-based discrete time formulation.* Computer and Chemical Engineering. Vol. 33, No. 8, pp. 1413-1426.

Saxena R., 2007. *Cross-docking demystified.* Industrial Engineer. Vol. 39, Issue 7, p24.

Soltani R., Sadjadi S.J., 2010. *Scheduling trucks in cross-docking systems: A robust meta-heuristics approach.* Transportation Research Part E: Logistics and Transportation Review. Vol. 46, No. 5, pp. 650-666.

Tsui L, Chia-Hao C., 1992. *An optimal solution to a dock door assignment problem.* Computers and Industrial Engineering. Vol. 23, pp. 283-86.

Van Belle J., Valckenaers P., Cattrysse D., 2012. *Cross-docking: State of the art.* Omega. 40, 827–846.

Waller M. A., Cassady C. R., Ozment J., 2006. *Impact of cross-docking on inventory in a decentralized retailsupply chain.* Transportation Research Part E: Logistics

and Transportation Review. 42(5), 359–82.

Wang J. F., Regan A., 2008. *Real-time trailer scheduling for crossdock operations.* Transportation Journal. Vol. 47, No. 2, pp. 5-20.

Zhang T., Saharidis G.K.D., Theofanis S., Boile M., 2010. *Scheduling of inbound and outbound trucks at cross-docks: modeling and analysis.* Transportation Research Record: Journal of the Transportation Research Board. 2162, 9-16.

B. Internet

The online magazine for the materials handling professional, http://eurekapub.eu

APPENDIX

A. Computer specs

Dell Inspiron 15 7000 Series

Intel® Core™ i7-6700HQ CPU @2.60GHz

Ram: 8Gb

HDD: 1 TB 5.400 RPM

Operating System: Windows 10 Home

Microsoft Visual Studio 2010 Ultimate

IBM Cplex Studio 12.6.

B. C++ codes

B1. Data continuous time

```
double M=100000;

double HTij[imax][jmax];
for (i=0;i<imax;i++){
	for (j=0;j<jmax;j++){
HTij[i][j]=0;
	}
}
HTij[0][0]=103.3251;
HTij[1][0]=111.5213;
HTij[2][0]=41.4288;
HTij[3][0]=112.2038;
HTij[4][0]=86.9123;

HTij[0][1]=38.7786;
HTij[1][1]=55.0648;
HTij[2][1]=79.2193;
HTij[3][1]=116.1756;
HTij[4][1]=116.84;

HTij[0][2]=44.1852;
HTij[1][2]=117.3534;
HTij[2][2]=116.145;
HTij[3][2]=73.6838;
HTij[4][2]=102.0252;

HTij[0][3]=42.7698;
HTij[1][3]=67.9585;
HTij[2][3]=112.4162;
HTij[3][3]=101.2987;
HTij[4][3]=116.3543;

HTij[0][4]=89.0167;
HTij[1][4]=33.2141;
HTij[2][4]=106.4216;
HTij[3][4]=114.0594;
HTij[4][4]=91.0862;

HTij[0][5]=98.1966;
HTij[1][5]=96.8819;
HTij[2][5]=65.3004;
HTij[3][5]=88.993;
HTij[4][5]=45.4068;

HTij[0][6]=93.5441;
HTij[1][6]=32.865;
HTij[2][6]=54.9231;
HTij[3][6]=34.1554;
HTij[4][6]=38.7419;

HTij[0][7]=104.1112;
HTij[1][7]=92.5346;
HTij[2][7]=58.539;
HTij[3][7]=115.52;
HTij[4][7]=33.1001;

HTij[0][8]=69.487;
HTij[1][8]=64.3403;
HTij[2][8]=98.8965;
HTij[3][8]=101.568;
HTij[4][8]=46.8185;

HTij[0][9]=74.0788;
HTij[1][9]=70.1028;
HTij[2][9]=88.1682;
HTij[3][9]=93.8428;
HTij[4][9]=97.9218;
```

```
HTij[0][10]=54.8423;
HTij[1][10]=91.1732;
HTij[2][10]=88.9588;
HTij[3][10]=44.6351;
HTij[4][10]=40.7098;

HTij[0][11]=74.8528;
HTij[1][11]=116.377;
HTij[2][11]=60.6347;
HTij[3][11]=82.6741;
HTij[4][11]=50.1431;

HTij[0][12]=97.614;
HTij[1][12]=52.9586;
HTij[2][12]=75.5361;
HTij[3][12]=92.9169;
HTij[4][12]=110.1813;

HTij[0][13]=116.3362;
HTij[1][13]=79.2494;
HTij[2][13]=42.4762;
HTij[3][13]=43.4365;
HTij[4][13]=53.1757;

HTij[0][14]=105.6646;
HTij[1][14]=52.8854;
HTij[2][14]=103.2856;
HTij[3][14]=51.9172;
HTij[4][14]=113.6337;

HTij[0][15]=61.4985;
HTij[1][15]=47.6936;
HTij[2][15]=52.5975;
HTij[3][15]=85.444;
HTij[4][15]=72.596;

HTij[0][16]=61.6494;
HTij[1][16]=104.7746;
HTij[2][16]=82.6738;
HTij[3][16]=79.4751;
HTij[4][16]=112.5474;

HTij[0][17]=55.7255;
HTij[1][17]=98.148;
HTij[2][17]=97.8356;
HTij[3][17]=64.2401;
HTij[4][17]=81.1039;

HTij[0][18]=36.8269;
HTij[1][18]=34.8555;
HTij[2][18]=77.7718;
HTij[3][18]=100.1251;
HTij[4][18]=114.061;

HTij[0][19]=41.6916;
HTij[1][19]=81.1941;
HTij[2][19]=72.2452;
HTij[3][19]=31.0712;
HTij[4][19]=60.341;


double ATj[jmax];

for (j=0;j<jmax;j++){
ATj[j]=0;
}

ATj[0]=0.2481;
ATj[1]=2.6105;
ATj[2]=5.8558;
ATj[3]=7.9407;
ATj[4]=8.5386;
ATj[5]=12.8048;
ATj[6]=14.0426;
ATj[7]=21.3618;
ATj[8]=23.5718;
ATj[9]=24.3185;
ATj[10]=1.1904;
```

```
ATj[11]=3.5171;
ATj[12]=7.881;
ATj[13]=7.9856;
ATj[14]=11.3557;
ATj[15]=13.8456;
ATj[16]=18.2158;
ATj[17]=23.1397;
ATj[18]=23.8842;
ATj[19]=25.115;

double Si[imax];

for (i=0;i<imax;i++){
Si[i]=0;
}

Si[0]=1.05;
Si[1]=1.75;
Si[2]=0.1;
Si[3]=2.1;
Si[4]=4.75;
```

B2. Model 1

```
//------------------------------------MODEL1 ------------------------------------
------------------
#include <ilcplex/ilocplex.h>
#include <iostream>
using namespace std;

ILOSTLBEGIN

int main (){

IloEnv env;

typedef IloArray<IloNumArray> IloNumMatrix2x2;
typedef IloArray<IloNumMatrix2x2> IloNumMatrix3x3;
typedef IloArray<IloNumMatrix3x3> IloNumMatrix4x4;
typedef IloArray<IloNumMatrix4x4> IloNumMatrix5x5;
typedef IloArray<IloNumMatrix5x5> IloNumMatrix6x6;

typedef IloArray<IloNumVarArray> IloNumVarMatrix2x2;
typedef IloArray<IloNumVarMatrix2x2> IloNumVarMatrix3x3;
typedef IloArray<IloNumVarMatrix3x3> IloNumVarMatrix4x4;
typedef IloArray<IloNumVarMatrix4x4> IloNumVarMatrix5x5;
typedef IloArray<IloNumVarMatrix5x5> IloNumVarMatrix6x6;

typedef IloArray<IloRangeArray> IloRangeMatrix2x2;
typedef IloArray<IloRangeMatrix2x2> IloRangeMatrix3x3;
typedef IloArray<IloRangeMatrix3x3> IloRangeMatrix4x4;
typedef IloArray<IloRangeMatrix4x4> IloRangeMatrix5x5;
typedef IloArray<IloRangeMatrix5x5> IloRangeMatrix6x6;

int i,j,k,f,h,m; // f=i' , h=j' kai m=k'

const int imax=5;
```

```
const int fmax=5;
const int jmax=20;
const int hmax=20;
const int kmax=19;
const int mmax=19;

double UBGeneral=50000;
double LBGeneral=-50000;


IloModel model (env);




IloCplex cplex(env);


//---------------------------------------------------------------------------------
//------------------------- Decision Variables   ----------------------------
//---------------------------------------------------------------------------------
//---------------------------------------------------------------------------------
//------------------------- Decision Variable TS ----------------------------


IloNumVarArray TSj(env,0);
for (j=0;j<jmax;j++){
                    char TSServedTruck_j[70];
                    sprintf(TSServedTruck_j,"TS(j%d)",j);
                    IloNumVar TS(env,0,IloInfinity,ILOFLOAT,TSServedTruck_j);
                    TSj.add(TS);
}
//------------------------- Decision Variable X ----------------------------

IloNumVarMatrix3x3 Xjik(env,0);
for (j=0;j<jmax;j++){
       IloNumVarMatrix2x2 Xik(env,0);
       for (i=0;i<imax;i++){
              IloNumVarArray Xk(env,0);
              for (k=0;k<kmax;k++){
                    char ServedTruck_j_i_k[70];
                    sprintf(ServedTruck_j_i_k,"X(j%d,i%d,k%d)",j,i,k);
                    IloNumVar X(env,0,1,ILOINT,ServedTruck_j_i_k);
                    Xk.add(X);
              }
       Xik.add(Xk);
       }
Xjik.add(Xik);
}

//---------------------------------------------------------------------------------
//------------------------- Constraints ---------------------------------------
//---------------------------------------------------------------------------------
//------------------------- Constraint 1 ---------------------------------------

IloRangeArray Con1j(env,0);
for (j=0;j<jmax;j++){
       IloExpr expr(env,0);
```

```
for (i=0;i<imax;i++){
	for (k=0;k<kmax;k++){
expr+=Xjik[j][i][k];
	}
}
	char Con_1[60];
	sprintf(Con_1,"Con_1j(j%d)",j);
	float LB=1,UB=1;
	IloRange Con1(env,LB,expr,UB,Con_1);
	model.add(Con1);
	Con1j.add(Con1);
	expr.end();
}

//------------------------- Constraint 2 ----------------------------

IloRangeMatrix2x2 Con2ik(env,0);
for (i=0;i<imax;i++){
	IloRangeArray Con2k(env,0);
	for (k=0;k<kmax;k++){
			IloExpr expr(env,0);
for (j=0;j<jmax;j++){
expr+=Xjik[j][i][k];
}
			char Con_2[60];
			sprintf(Con_2,"Con_2ik(i%d,k%d)",i,k);
			float LB=-IloInfinity,UB=1;
			IloRange Con2(env,LB,expr,UB,Con_2);
			model.add(Con2);
			Con2k.add(Con2);
			expr.end();
		}
Con2ik.add(Con2k);
}

//------------------------- Constraint 3 ----------------------------

IloRangeMatrix3x3 Con3jik(env,0);
	for (j=0;j<jmax;j++){
		IloRangeMatrix2x2 Con3ik(env,0);
		for (i=0;i<imax;i++){
			IloRangeArray Con3k(env,0);
			for (k=0;k<kmax;k++){
		IloExpr expr(env,0);
expr+=TSj[j]-ATj[j];
		char Con_3[60];
		sprintf(Con_3,"Con_3jik(j%d,i%d,k%d)",j,i,k);
		float LB=0,UB=IloInfinity;
		IloRange Con3(env,LB,expr,UB,Con_3);
		model.add(Con3);
		Con3k.add(Con3);
		expr.end();
			}
	Con3ik.add(Con3k);
	}
Con3jik.add(Con3ik);
}
```

```
//------------------------- Constraint 4A ----------------------------

IloRangeMatrix2x2 Con4Aij(env,0);
for  (i=0;i<imax;i++){
       IloRangeArray Con4Aj(env,0);
              for (j=0;j<jmax;j++){
                     IloExpr expr(env,0);
expr+=TSj[j]-Si[i];
                                   char Con_4A[60];
                                   sprintf(Con_4A,"Con_4Aij(i%d,j%d)",i,j);
                                   float LB=0,UB=IloInfinity;
                                   IloRange Con4A(env,LB,expr,UB,Con_4A);
                                   model.add(Con4A);
                                   Con4Aj.add(Con4A);
                                   expr.end();
             }
              Con4Aij.add(Con4Aj);
}

//------------------------- Constraint 4B ----------------------------

int e=0;
for (i=0; i<imax; i++){
                                      e+=Si[i];
                              }
IloRangeMatrix2x2 Con4Bij(env,0);
       for (i=0;i<imax;i++){
              IloRangeArray Con4Bj(env,0);
                     for (j=0;j<jmax;j++){
                                   IloExpr expr(env,0);
          expr+=TSj[j];
                                   for (k=0;k<kmax;k++){
       expr+=M*(1-(Xjik[j][i][k]));
                                   }
                                   char Con_4B[60];
                                   sprintf(Con_4B,"Con_4Bij(i%d,j%d)",i,j);
                                   float LB=e,UB=IloInfinity;
                                   IloRange Con4B(env,LB,expr,UB,Con_4B);
                                   model.add(Con4B);
                                   Con4Bj.add(Con4B);
                                   expr.end();
                            }
              Con4Bij.add(Con4Bj);
              }

//------------------------- Constraint 5A --------------------------------

IloRangeMatrix3x3 Con5Aijk(env,0);
  for (i=0;i<imax;i++){
IloRangeMatrix2x2 Con5Ajk(env,0);
       for (j=0;j<jmax;j++){
         IloRangeArray Con5Ak(env,0);
              for (k=1;k<kmax;k++){
                                   IloExpr expr(env,0);
expr+=((k-1)*Xjik[j][i][k]);

       if(k>0){
       for (m=0;m<k;m++){
```

```
              for (h=0;h<hmax;h++){
expr+=-Xjik[h][i][m];
              }
       }
       }
                                   char Con_5A[60];

       sprintf(Con_5A,"Con_5Aijk(i%d,j%d,k%d)",i,j,k);
                                      float LB=-IloInfinity,UB=0;
                                      IloRange Con5A(env,LB,expr,UB,Con_5A);
                                      if(h!=j){
                                      model.add(Con5A);
                                      }
                                      Con5Ak.add(Con5A);
                                      expr.end();
                            }
              Con5Ajk.add(Con5Ak);
              }
       Con5Aijk.add(Con5Ajk);
 }

//------------------------- Constraint 5B ----------------------------

IloRangeMatrix2x2 Con5Bik(env,0);
       for (i=0;i<imax;i++){
              IloRangeArray Con5Bk(env,0);
                     for (k=1;k<kmax;k++){
                                      IloExpr expr(env,0);
                                      for (j=0;j<jmax;j++){
   expr+=Xjik[j][i][k];
                                      }
if (k>0){
       for (j=0;j<jmax;j++){
expr+=-Xjik[j][i][k-1];
              }
       }
                                      char Con_5B[60];
                                      sprintf(Con_5B,"Con_5Bik(i%d,k%d)",i,k);
                                      float LB=-IloInfinity,UB=0;
                                      IloRange Con5B(env,LB,expr,UB,Con_5B);
                                      model.add(Con5B);
                                      Con5Bk.add(Con5B);
                                      expr.end();
              }
}

//------------------------- Constraint 5C ----------------------------

IloRangeMatrix3x3 Con5Cjik(env,0);
    for (j=0;j<jmax;j++){
    IloRangeMatrix2x2 Con5Cik(env,0);
              for (i=0;i<imax;i++){
                     IloRangeArray Con5Ck(env,0);
                        for (k=1;k<kmax;k++){
                                      IloExpr expr(env,0);
expr+=Xjik[j][i][k];
if (k>0){
       for (h=0;h<jmax;h++){
```

```
expr+=-Xjik[h][i][k-1];
                        }
               }
                              char Con_5C[60];
                              sprintf(Con_5C,"Con_5Cjik(j%d,i%d,k%d)",j,i,k);
                              float LB=-IloInfinity,UB=0;
                              IloRange Con5C(env,LB,expr,UB,Con_5C);
                              if(h!=j){
                              model.add(Con5C);
                              }
                              Con5Ck.add(Con5C);
                              expr.end();
                         }
       Con5Cik.add(Con5Ck);
            }
 Con5Cjik.add(Con5Cik);
}

//------------------------- Constraint 6 ----------------------------

IloRangeMatrix4x4 Con6hjik(env,0);
for (h=0;h<hmax;h++){
       IloRangeMatrix3x3 Con6jik(env,0);
       for (j=0;j<jmax;j++){
              IloRangeMatrix2x2 Con6ik(env,0);
              for (i=0;i<imax;i++){
                     IloRangeArray Con6k(env,0);
                     for (k=1;k<kmax;k++){
                                   IloExpr expr(env,0);
expr+=-M*(1-Xjik[j][i][k-1])+TSj[j]+HTij[i][j]*Xjik[j][i][k-1]-TSj[h]-M*(1-
Xjik[h][i][k]);
                                   char Con_6[60];

       sprintf(Con_6,"Con_6hjik(h%d,j%d,i%d,k%d)",h,j,i,k);
                                   float LB=-IloInfinity,UB=0;
                                   IloRange Con6(env,LB,expr,UB,Con_6);
                                   if(h!=j){
                                   model.add(Con6);
                                   }
                                   Con6k.add(Con6);
                                   expr.end();
                            }
              Con6ik.add(Con6k);
              }
       Con6jik.add(Con6ik);
       }
Con6hjik.add(Con6jik);
}

//--------------------------------------------------------------------------------
-------------
//------------------------------------OBJECTIVE FUNCTION-------------------------
------------
//--------------------------------------------------------------------------------
------------

IloExpr expr1(env);
for (j=0;j<jmax;j++){
```

```
              expr1+=TSj[j];
}
              for (j=0;j<jmax;j++){
               for (i=0;i<imax;i++){
                      for (k=0;k<kmax;k++){
              expr1+=(Xjik[j][i][k]*HTij[i][j]);
              }
       }
}

model.add(IloMinimize(env, expr1));
expr1.end();

cplex.extract(model);
cplex.exportModel("onoma.lp");



cplex.setParam(IloCplex::Param::TimeLimit, 3600);
cplex.solve();



if (!cplex.solve ()){
                     env.error()<<"Failed to optimize LP."<<endl;
                                   }

              env.out()<<"Solution status = " <<cplex.getStatus()<<endl;
              env.out()<<"Solution value = " <<cplex.getObjValue()<<endl;

//-------------------PRINT tin X------------------------------------------
for (j=0;j<jmax;j++){
       for (i=0;i<imax;i++){
              for (k=0;k<kmax;k++){
                     float XN = cplex.getValue(Xjik[j][i][k]);
                     if(XN!=0)
cout<<"Xjik"<<"("<<j<<","<<i<<","<<k<<")"<<"="<<XN<<endl;
                     }
              }
       }


//-------------------PRINT tin TS------------------------------------------

         for (j=0;j<jmax;j++){
                     float TSN = cplex.getValue(TSj[j]);
                     if(TSN!=0) cout<<"TSj"<<"("<<j<<")"<<"="<<TSN<<endl;
                     }




system("pause");

return 0;
} //End main
```

B3. Model 2

```
//------------------------------------MODEL2 -----------------------------------
-
//-----------------------------------------------------------------------------
//------------------------- Decision Variables   ----------------------------
//-----------------------------------------------------------------------------
//------------------------- Decision Variable TS ---------------------------

IloNumVarArray TSj(env,0);
for (j=0;j<jmax;j++){
                     char TSServedTruck_j[70];
                     sprintf(TSServedTruck_j,"TS(j%d)",j);
                     IloNumVar TS(env,0,IloInfinity,ILOFLOAT,TSServedTruck_j);
                     TSj.add(TS);
}

//------------------------- Decision Variable X ---------------------------

       IloNumVarMatrix2x2 Xji(env,0);
       for (j=0;j<jmax;j++){
              IloNumVarArray Xi(env,0);
              for (i=0;i<imax;i++){
                     char ServingDoor_j_i[70];
                     sprintf(ServingDoor_j_i,"X(j%d,i%d)",j,i);
                     IloNumVar X(env,0,1,ILOINT,ServingDoor_j_i);
                     Xi.add(X);
              }
       Xji.add(Xi);
       }

//------------------------- Decision Variable W ---------------------------

       IloNumVarMatrix2x2 Wjk(env,0);
       for (j=0;j<jmax;j++){
              IloNumVarArray Wk(env,0);
              for (k=0;k<kmax;k++){
                     char ServedTruck_j_k[70];
                     sprintf(ServedTruck_j_k,"W(j%d,k%d)",j,k);
                     IloNumVar W(env,0,1,ILOINT,ServedTruck_j_k);
                     Wk.add(W);
              }
       Wjk.add(Wk);
       }

//-----------------------------------------------------------------------------
//------------------------- Constraints ---------------------------------------
//-----------------------------------------------------------------------------
//------------------------- Constraint 7 --------------------------------------

IloRangeArray Con7j(env,0);
for (j=0;j<jmax;j++){
       IloExpr expr(env,0);
for (i=0;i<imax;i++){
expr+=Xji[j][i];
}
       char Con_7[60];
       sprintf(Con_7,"Con_7j(j%d)",j);
```

```
        float LB=1,UB=1;
        IloRange Con7(env,LB,expr,UB,Con_7);
        model.add(Con7);
        Con7j.add(Con7);
        expr.end();
}


//------------------------- Constraint 8 --------------------------------------

IloRangeArray Con8j(env,0);
for (j=0;j<jmax;j++){
        IloExpr expr(env,0);
for (k=0;k<kmax;k++){
expr+=Wjk[j][k];
}
        char Con_8[60];
        sprintf(Con_8,"Con_8j(j%d)",j);
        float LB=1,UB=1;
        IloRange Con8(env,LB,expr,UB,Con_8);
        model.add(Con8);
        Con8j.add(Con8);
        expr.end();
}


//------------------------- Constraint 9 ----------------------------

IloRangeMatrix4x4 Con9hjik(env,0);
for (h=0;h<hmax;h++){
        IloRangeMatrix3x3 Con9jik(env,0);
        for (j=0;j<jmax;j++){
                IloRangeMatrix2x2 Con9ik(env,0);
                for (i=0;i<imax;i++){
                        IloRangeArray Con9k(env,0);
                        for (k=0;k<kmax;k++){
                                        IloExpr expr(env,0);
expr+=-M*(1-Xji[h][i])+Wjk[h][k]-M*(1-Xji[j][i])-M*(1-Wjk[j][k]);
                                        char Con_9[60];

        sprintf(Con_9,"Con_9hjik(h%d,j%d,i%d,k%d)",h,j,i,k);
                                        float LB=-IloInfinity,UB=0;
                                        IloRange Con9(env,LB,expr,UB,Con_9);
                                        if(h!=j){
                                        model.add(Con9);
                                        }
                                        Con9k.add(Con9);
                                        expr.end();
                                }
                Con9ik.add(Con9k);
                }
        Con9jik.add(Con9ik);
        }
Con9hjik.add(Con9jik);
}

//------------------------- Constraint 10 ---------------------------
```

```
IloRangeMatrix5x5 Con10mhjik(env,0);
for (m=k+1;m<mmax-1;m++){
 IloRangeMatrix4x4 Con10hjik(env,0);
  for (h=0;h<hmax;h++){
       IloRangeMatrix3x3 Con10jik(env,0);
        for (j=0;j<jmax;j++){
              IloRangeMatrix2x2 Con10ik(env,0);
               for (i=0;i<imax;i++){
                     IloRangeArray Con10k(env,0);
                      for (k=0;k<kmax;k++){
                                   IloExpr expr(env,0);
expr+=-M*(1-Xji[j][i])+Wjk[j][k]-M*(1-Xji[h][i])-Wjk[h][m];
                                   char Con_10[60];

       sprintf(Con_10,"Con_10mhjik(m%d,h%d,j%d,i%d,k%d)",m,h,j,i,k);
                                   float LB=-IloInfinity,UB=0;
                                   IloRange Con10(env,LB,expr,UB,Con_10);
                                   if(h!=j){
                                          if(m!=k){
                                   model.add(Con10);
                                          }
                                   }
                                   Con10k.add(Con10);
                                   expr.end();
                              }
                  Con10ik.add(Con10k);
                     }
            Con10jik.add(Con10ik);
               }
    Con10hjik.add(Con10jik);
      }
Con10mhjik.add(Con10hjik);
}

//------------------------- Constraint 11 ----------------------------

IloRangeMatrix5x5 Con11mhjik(env,0);
for (m=k+1;m<mmax-1;m++){
IloRangeMatrix4x4 Con11hjik(env,0);
for (h=0;h<hmax;h++){
       IloRangeMatrix3x3 Con11jik(env,0);
       for (j=0;j<jmax;j++){
              IloRangeMatrix2x2 Con11ik(env,0);
              for (i=0;i<imax;i++){
                     IloRangeArray Con11k(env,0);
                     for (k=1;k<kmax;k++){
                                   IloExpr expr(env,0);
expr+=-M*(1-Xji[j][i])-M*(1-Wjk[j][k])+TSj[j]+(HTij[i][j]*Xji[j][i])-TSj[h]-M*(1-
Xji[h][i])-M*(1-Wjk[h][m]);
                            char Con_11[60];

       sprintf(Con_11,"Con_11mhjik(m%d,h%d,j%d,i%d,k%d)",m,h,j,i,k);
                                   float LB=-IloInfinity,UB=0;
                                   IloRange Con11(env,LB,expr,UB,Con_11);
                                   if(h!=j){
                                          if(m!=k){
                                   model.add(Con11);
                                   }
```

```
                                    }
                                    Con11k.add(Con11);
                                    expr.end();
                             }
              Con11ik.add(Con11k);
              }
       Con11jik.add(Con11ik);
       }
Con11hjik.add(Con11jik);
}
Con11mhjik.add(Con11hjik);
}

//---------------------------------------------------------------------------------
-
//------------------------------------OBJECTIVE FUNCTION-------------------------
-
//---------------------------------------------------------------------------------
-

IloExpr expr1(env);
for (j=0;j<jmax;j++){
       for (i=0;i<imax;i++){
              expr1+=TSj[j]+(Xji[j][i]*HTij[i][j]);
       }
}

model.add(IloMinimize(env, expr1));
expr1.end();
```

B4. Data discrete time

```
double M=100000;

double
HTij[imax][jmax];
for (i=0;i<imax;i++){
        for
(j=0;j<jmax;j++){
HTij[i][j]=0;
        }
}

HTij[0][0]=21;
HTij[1][0]=22;
HTij[2][0]=8;
HTij[3][0]=22;
HTij[4][0]=17;

HTij[0][1]=9;
HTij[1][1]=11;
HTij[2][1]=16;
HTij[3][1]=23;
HTij[4][1]=23;

HTij[0][2]=9;
HTij[1][2]=24;
HTij[2][2]=23;
HTij[3][2]=15;
HTij[4][2]=21;

HTij[0][3]=9;
HTij[1][3]=14;
HTij[2][3]=23;
HTij[3][3]=20;
HTij[4][3]=23;

HTij[0][4]=18;
HTij[1][4]=7;
HTij[2][4]=21;
HTij[3][4]=23;
HTij[4][4]=18;

HTij[0][5]=20;
HTij[1][5]=19;
HTij[2][5]=13;
HTij[3][5]=18;
HTij[4][5]=9;

HTij[0][6]=19;
HTij[1][6]=7;
HTij[2][6]=11;
HTij[3][6]=7;
HTij[4][6]=8;

HTij[0][7]=21;
HTij[1][7]=19;
HTij[2][7]=12;
HTij[3][7]=23;
HTij[4][7]=7;

HTij[0][8]=14;
HTij[1][8]=13;
HTij[2][8]=20;
HTij[3][8]=20;
HTij[4][8]=9;

HTij[0][9]=15;
HTij[1][9]=14;
HTij[2][9]=18;
HTij[3][9]=19;
HTij[4][9]=20;

HTij[0][10]=11;
HTij[1][10]=18;
HTij[2][10]=18;
HTij[3][10]=9;
HTij[4][10]=8;

HTij[0][11]=15;
HTij[1][11]=23;
HTij[2][11]=12;
HTij[3][11]=17;
HTij[4][11]=10;

HTij[0][12]=20;
HTij[1][12]=11;
HTij[2][12]=15;
HTij[3][12]=19;
HTij[4][12]=22;

HTij[0][13]=23;
HTij[1][13]=16;
HTij[2][13]=8;
HTij[3][13]=9;
HTij[4][13]=11;

HTij[0][14]=21;
HTij[1][14]=11;
HTij[2][14]=21;
HTij[3][14]=20;
HTij[4][14]=23;

HTij[0][15]=12;
HTij[1][15]=10;
HTij[2][15]=11;
HTij[3][15]=17;
HTij[4][15]=15;

HTij[0][16]=12;
HTij[1][16]=21;
HTij[2][16]=17;
HTij[3][16]=16;
HTij[4][16]=23;

HTij[0][17]=11;
HTij[1][17]=20;
HTij[2][17]=20;
HTij[3][17]=13;
HTij[4][17]=16;

HTij[0][18]=7;
HTij[1][18]=7;
HTij[2][18]=16;
HTij[3][18]=20;
HTij[4][18]=23;

HTij[0][19]=8;
HTij[1][19]=16;
HTij[2][19]=14;
HTij[3][19]=6;
HTij[4][19]=12;


double ATj[jmax];

for (j=0;j<jmax;j++){
ATj[j]=0;
}
ATj[0]=0;
ATj[1]=0;
ATj[2]=1;
ATj[3]=2;
ATj[4]=2;
ATj[5]=3;
ATj[6]=3;
ATj[7]=4;
ATj[8]=5;
ATj[9]=5;
ATj[10]=0;
ATj[11]=1;
ATj[12]=2;
ATj[13]=2;
ATj[14]=2;
ATj[15]=3;
ATj[16]=4;
ATj[17]=5;
ATj[18]=5;
ATj[19]=5;

double Si[imax];
for (i=0;i<imax;i++){
Si[i]=0;
}
Si[0]=0;
Si[1]=0;
Si[2]=0;
Si[3]=1;
Si[4]=2;
```

B5. Model 3

```
//-----------------------------------MODEL3------------------------------------
-----------------
#include <ilcplex/ilocplex.h>
#include <iostream>
using namespace std;

ILOSTLBEGIN

int main (){

IloEnv env;

typedef IloArray<IloNumArray> IloNumMatrix2x2;
typedef IloArray<IloNumMatrix2x2> IloNumMatrix3x3;
typedef IloArray<IloNumMatrix3x3> IloNumMatrix4x4;
typedef IloArray<IloNumMatrix4x4> IloNumMatrix5x5;
typedef IloArray<IloNumMatrix5x5> IloNumMatrix6x6;

typedef IloArray<IloNumVarArray> IloNumVarMatrix2x2;
typedef IloArray<IloNumVarMatrix2x2> IloNumVarMatrix3x3;
typedef IloArray<IloNumVarMatrix3x3> IloNumVarMatrix4x4;
typedef IloArray<IloNumVarMatrix4x4> IloNumVarMatrix5x5;
typedef IloArray<IloNumVarMatrix5x5> IloNumVarMatrix6x6;

typedef IloArray<IloRangeArray> IloRangeMatrix2x2;
typedef IloArray<IloRangeMatrix2x2> IloRangeMatrix3x3;
typedef IloArray<IloRangeMatrix3x3> IloRangeMatrix4x4;
typedef IloArray<IloRangeMatrix4x4> IloRangeMatrix5x5;
typedef IloArray<IloRangeMatrix5x5> IloRangeMatrix6x6;

int i,j,t;

const int imax=5;
const int jmax=20;
const int tmax=100;

double UBGeneral=50000;
double LBGeneral=-50000;


IloModel model (env);


IloCplex cplex(env);


//-------------------------------------------------------------------------------
//------------------------- Decision Variables   ----------------------------
//-------------------------------------------------------------------------------
//------------------------- Decision Variable SC ----------------------------

IloNumVarMatrix3x3 SCjit(env,0);
for (j=0;j<jmax;j++){
       IloNumVarMatrix2x2 SCit(env,0);
       for (i=0;i<imax;i++){
              IloNumVarArray SCt(env,0);
```

```
				for (t=0;t<tmax;t++){
					char SCServedTruck_j_i_t[70];
					sprintf(SCServedTruck_j_i_t,"SC(j%d,i%d,t%d)",j,i,t);
					IloNumVar SC(env,0,1,ILOINT,SCServedTruck_j_i_t);
					SCt.add(SC);
				}
			SCit.add(SCt);
			}
		SCjit.add(SCit);
		}


		//------------------------- Decision Variable CC ---------------------------

		IloNumVarMatrix3x3 CCjit(env,0);
		for (j=0;j<jmax;j++){
			IloNumVarMatrix2x2 CCit(env,0);
			for (i=0;i<imax;i++){
				IloNumVarArray CCt(env,0);
				for (t=0;t<tmax;t++){
					char CCServingDoor_j_i_t[70];
					sprintf(CCServingDoor_j_i_t,"CC(j%d,i%d,t%d)",j,i,t);
					IloNumVar CC(env,0,1,ILOINT,CCServingDoor_j_i_t);
					CCt.add(CC);
				}
			CCit.add(CCt);
			}
		CCjit.add(CCit);
		}

		//--------------------------------------------------------------------------------
		//------------------------- Constraints ---------------------------------------
		//--------------------------------------------------------------------------------
		//------------------------- Constraint 12 -------------------------------------

		IloRangeArray Con12j(env,0);
			for (j=0;j<jmax;j++){
				IloExpr expr(env,0);
				for (i=0;i<imax;i++){
					for (t=0;t<tmax;t++){
		expr+=SCjit[j][i][t];
					}
				}
				char Con_12[60];
				sprintf(Con_12,"Con_12j(j%d)",j);
				float LB=1,UB=1;
				IloRange Con12(env,LB,expr,UB,Con_12);
				model.add(Con12);
				Con12j.add(Con12);
				expr.end();
		}

		//------------------------- Constraint 13--------------------------------------

		IloRangeMatrix2x2  Con13jt(env,0);
			for (j=0;j<jmax;j++){
			IloRangeArray Con13t(env,0);
			 for (t=0;t<tmax;t++){
```

```
                IloExpr expr(env,0);
                for (i=0;i<imax;i++){
expr+=CCjit[j][i][t];
                        }
                char Con_13[60];
                sprintf(Con_13,"Con_13jt(j%d,t%d)",j,t);
                float LB=-IloInfinity,UB=1;
                IloRange Con13(env,LB,expr,UB,Con_13);
                model.add(Con13);
                Con13t.add(Con13);
                expr.end();
         }
                Con13jt.add(Con13t);
}

//------------------------- Constraint 14A --------------------------------------
-

       IloRangeArray Con14Aj(env,0);
        for (j=0;j<jmax;j++){
                IloExpr expr(env,0);
                for (i=0;i<imax;i++){
                        for (t=ATj[j];t<tmax;t++){
expr+=SCjit[j][i][t];
                        }
                }
                char Con_14A[60];
                sprintf(Con_14A,"Con_14Aj(j%d)",j);
                float LB=1,UB=1;
                IloRange Con14A(env,LB,expr,UB,Con_14A);
                model.add(Con14A);
                Con14Aj.add(Con14A);
                expr.end();
 }
//------------------------- Constraint 14B --------------------------------------
-

int f=0;
for (i=0; i<imax; i++){
                                        f+=ATj[j];
                                }
       IloRangeArray Con14Bj(env,0);
        for (j=0;j<jmax;j++){
                IloExpr expr(env,0);
                for (i=0;i<imax;i++){
                        for (t=0;t<tmax;t++){
expr+=SCjit[j][i][t];
                        }
                }
                char Con_14B[60];
                sprintf(Con_14B,"Con_14Bj(j%d)",j);
                float LB=f,UB=IloInfinity;
                IloRange Con14B(env,LB,expr,UB,Con_14B);
                model.add(Con14B);
                Con14Bj.add(Con14B);
                expr.end();
 }
```

```
//------------------------- Constraint 15 ---------------------------------------

IloRangeArray Con15i(env,0);
       for (i=0;i<imax;i++){
              IloExpr expr(env,0);
              for (j=0;j<jmax;j++){
                     for (t=0;t<Si[i];t++){
expr+=SCjit[j][i][t];
                     }
              }
              char Con_15[60];
              sprintf(Con_15,"Con_15i(i%d)",i);
              float LB=0,UB=0;
              IloRange Con15(env,LB,expr,UB,Con_15);
              model.add(Con15);
              Con15i.add(Con15);
              expr.end();
}

//------------------------- Constraint 16 ---------------------------------------

IloRangeMatrix2x2  Con16it(env,0);
       for (i=0;i<imax;i++){
       IloRangeArray Con16t(env,0);
        for (t=0;t<tmax;t++){
              IloExpr expr(env,0);
                for (j=0;j<jmax;j++){
expr+=CCjit[j][i][t];
                     }
              char Con_16[60];
              sprintf(Con_16,"Con_16jt(i%d,t%d)",i,t);
              float LB=-IloInfinity,UB=1;
              IloRange Con16(env,LB,expr,UB,Con_16);
              model.add(Con16);
              Con16t.add(Con16);
              expr.end();
        }
              Con16it.add(Con16t);
}

//------------------------- Constraint 17 ---------------------------------------

int n=0;
for (i=0; i<imax; i++){
       for (j=0;j<jmax;j++){
                                      n+=HTij[i][j];
                              }
}
IloRangeMatrix3x3 Con17jit(env,0);
   for (j=0;j<jmax;j++){
   IloRangeMatrix2x2  Con17it(env,0);
          for (i=0;i<imax;i++){
          IloRangeArray Con17t(env,0);
           for (t=0;t<tmax-n;t++){
              IloExpr expr(env,0);
expr+=-(HTij[i][j]+M)*SCjit[j][i][t];
for (n=0;n<HTij[i][j];n++){
expr+=CCjit[j][i][t+n];
```

```
}
                char Con_17[60];
                sprintf(Con_17,"Con_17jit(j%d,i%d,t%d)",j,i,t);
                float LB=-M,UB=IloInfinity;
                IloRange Con17(env,LB,expr,UB,Con_17);
                model.add(Con17);
                Con17t.add(Con17);
                expr.end();
                 }
                Con17it.add(Con17t);
        }
  Con17jit.add(Con17it);
}

//------------------------- Constraint 18 ---------------------------------------

int m=0;
for (i=0; i<imax; i++){
       for (j=0;j<jmax;j++){
                                        m+=HTij[i][j];
                              }
}
IloRangeMatrix3x3 Con18jit(env,0);
   for (j=0;j<jmax;j++){
   IloRangeMatrix2x2  Con18it(env,0);
          for (i=0;i<imax;i++){
          IloRangeArray Con18t(env,0);
           for (t=0;t<tmax-m;t++){
              IloExpr expr(env,0);
expr+=(HTij[i][j]+M)*SCjit[j][i][t];
       for (m=0;m<HTij[i][j];m++){
expr+=-CCjit[j][i][t+m];
       }
                char Con_18[60];
                sprintf(Con_18,"Con_18jit(j%d,i%d,t%d)",j,i,t);
                float LB=-M,UB=IloInfinity;
                IloRange Con18(env,LB,expr,UB,Con_18);
                model.add(Con18);
                Con18t.add(Con18);
                expr.end();
                 }
                Con18it.add(Con18t);
        }
  Con18jit.add(Con18it);
}

//------------------------- Constraint 19 ---------------------------------------

IloRangeMatrix3x3 Con19jit(env,0);
   for (j=0;j<jmax;j++){
   IloRangeMatrix2x2  Con19it(env,0);
          for (i=0;i<imax;i++){
          IloRangeArray Con19t(env,0);
           for (t=0;t<tmax;t++){
              IloExpr expr(env,0);
expr+=SCjit[j][i][t]-CCjit[j][i][t];
                char Con_19[60];
                sprintf(Con_19,"Con_19jit(j%d,i%d,t%d)",j,i,t);
```

```
              float LB=-IloInfinity,UB=0;
              IloRange Con19(env,LB,expr,UB,Con_19);
              model.add(Con19);
              Con19t.add(Con19);
              expr.end();
               }
              Con19it.add(Con19t);
        }
  Con19jit.add(Con19it);
}

//------------------------- Constraint 20 -------------------------------------

IloRangeMatrix3x3 Con20jit(env,0);
   for (j=0;j<jmax;j++){
   IloRangeMatrix2x2  Con20it(env,0);
          for (i=0;i<imax;i++){
          IloRangeArray Con20t(env,0);
           for (t=1;t<tmax;t++){
              IloExpr expr(env,0);
expr+=SCjit[j][i][t]-CCjit[j][i][t]+CCjit[j][i][t-1];
              char Con_20[60];
              sprintf(Con_20,"Con_20jit(j%d,i%d,t%d)",j,i,t);
              float LB=-IloInfinity,UB=1;
              IloRange Con20(env,LB,expr,UB,Con_20);
              model.add(Con20);
              Con20t.add(Con20);
              expr.end();
               }
              Con20it.add(Con20t);
        }
  Con20jit.add(Con20it);
}

//-----------------------------------------------------------------------------
//---------------------------------Objective Function-1-------------------------
-
//-----------------------------------------------------------------------------

IloExpr expr456(env);
for (j=0;j<jmax;j++){
       for (i=0;i<imax;i++){
              for (t=0;t<tmax;t++){
              expr456+=(t+1)*(SCjit[j][i][t])*(HTij[i][j]);
              }
       }
}
//---------------------------------Objective Function-2-------------------------
-
//-----------------------------------------------------------------------------

IloExpr expr456(env);
for (j=0;j<jmax;j++){
       for (i=0;i<imax;i++){
              for (t=0;t<tmax;t++){
              expr456+=((t+1)*CCjit[j][i][t])+((HTij[i][j])*(SCjit[j][i][t]));
              }
       }
```

```
}
model.add(IloMinimize(env, expr456));
expr456.end();

cplex.extract(model);
cplex.exportModel("onoma.lp");


cplex.setParam(IloCplex::Param::TimeLimit, 3600);
cplex.solve();

if (!cplex.solve ()){
                    env.error()<<"Failed to optimize LP."<<endl;
                    //throw(-1);
             }


             env.out()<<"Solution status = " <<cplex.getStatus()<<endl;
             env.out()<<"Solution value = " <<cplex.getObjValue()<<endl;

//-------------------PRINT tin SC-------------------------------------------
for (j=0;j<jmax;j++){
       for (i=0;i<imax;i++){
             for (t=0;t<tmax;t++){
                    float SCN = cplex.getValue(SCjit[j][i][t]);
                    if(SCN!=0)
cout<<"SCjit"<<"("<<j<<","<<i<<","<<t<<")"<<"="<<SCN<<endl;
             }
       }
}

//-------------------PRINT tin CC-------------------------------------------

for (j=0;j<jmax;j++){
       for (i=0;i<imax;i++){
             for (t=0;t<tmax;t++){
                    float CCN = cplex.getValue(CCjit[j][i][t]);
                    if(CCN!=0)
cout<<"CCjit"<<"("<<j<<","<<i<<","<<t<<")"<<"="<<CCN<<endl;
             }
       }
}
system("pause");
return 0;
}
```

B6. Model 4

```
//------------------------------------MODEL4-----------------------------------
-
//--------------------------------------------------------------------------------
//------------------------- Decision Variables   ---------------------------
//--------------------------------------------------------------------------------
//------------------------- Decision Variable SC ---------------------------

IloNumVarMatrix2x2 SCji(env,0);
   for (j=0;j<jmax;j++){
         IloNumVarArray SCi(env,0);
```

```
            for (i=0;i<imax;i++){
                   char SCServedTruck_j_i[70];
                   sprintf(SCServedTruck_j_i,"SC(j%d,i%d)",j,i);
                   IloNumVar SC(env,0,1,ILOINT,SCServedTruck_j_i);
                   SCi.add(SC);
            }
       SCji.add(SCi);
}

//------------------------- Decision Variable CC ----------------------------
IloNumVarMatrix2x2 CCji(env,0);
       for (j=0;j<jmax;j++){
            IloNumVarArray CCi(env,0);
            for (i=0;i<imax;i++){
                   char CCServingDoor_j_i[70];
                   sprintf(CCServingDoor_j_i,"CC(j%d,i%d)",j,i);
                   IloNumVar CC(env,0,1,ILOINT,CCServingDoor_j_i);
                   CCi.add(CC);
            }
       CCji.add(CCi);
}

//------------------------- Decision Variable WSC ----------------------------
IloNumVarMatrix2x2 WSCjt(env,0);
   for (j=0;j<jmax;j++){
        IloNumVarArray WSCt(env,0);
            for (t=0;t<tmax;t++){
                   char WSCServedTruck_j_t[70];
                   sprintf(WSCServedTruck_j_t,"WSC(j%d,t%d)",j,t);
                   IloNumVar WSC(env,0,1,ILOINT,WSCServedTruck_j_t);
                   WSCt.add(WSC);
            }
       WSCjt.add(WSCt);
}
//------------------------- Decision Variable WCC ----------------------------

IloNumVarMatrix2x2 WCCjt(env,0);
       for (j=0;j<jmax;j++){
            IloNumVarArray WCCt(env,0);
            for (t=0;t<tmax;t++){
                   char WCCServingDoor_j_t[70];
                   sprintf(WCCServingDoor_j_t,"CC(j%d,t%d)",j,t);
                   IloNumVar WCC(env,0,1,ILOINT,WCCServingDoor_j_t);
                   WCCt.add(WCC);
            }
       WCCjt.add(WCCt);
}
//------------------------------------------------------------------------------
//------------------------- Constraints ---------------------------------------
//------------------------- Constraint 21 ---------------------------------------
IloRangeArray Con21j(env,0);
       for (j=0;j<jmax;j++){
            IloExpr expr(env,0);
             for (i=0;i<imax;i++){
expr+=SCji[j][i];
                   }
            char Con_21[60];
            sprintf(Con_21,"Con_21j(j%d)",j);
```

```
                float LB=1,UB=1;
                IloRange Con21(env,LB,expr,UB,Con_21);
                model.add(Con21);
                Con21j.add(Con21);
                expr.end();
}

//------------------------- Constraint 22 ---------------------------------------
IloRangeArray Con22j(env,0);
        for (j=0;j<jmax;j++){
                IloExpr expr(env,0);
                 for (t=0;t<tmax;t++){
expr+=WSCjt[j][t];
                        }
                char Con_22[60];
                sprintf(Con_22,"Con_22j(j%d)",j);
                float LB=1,UB=1;
                IloRange Con22(env,LB,expr,UB,Con_22);
                model.add(Con22);
                Con22j.add(Con22);
                expr.end();
}
//------------------------- Constraint 23 ---------------------------------------

IloRangeMatrix4x4  Con23jhit(env,0);
        for (j=0;j<jmax;j++){
    IloRangeMatrix3x3  Con23hit(env,0);
          for (h=0;h<hmax;h++){
      IloRangeMatrix2x2  Con23it(env,0);
             for (i=0;i<imax;i++){
             IloRangeArray Con23t(env,0);
            for (t=0;t<tmax;t++){
                IloExpr expr(env,0);
expr+=CCji[j][i]+WCCjt[j][t]+CCji[h][i]+WCCjt[h][t];
                char Con_23[60];
                sprintf(Con_23,"Con_23jhit(j%d,h%d,i%d,t%d)",j,h,i,t);
                float LB=-IloInfinity,UB=3;
                IloRange Con23(env,LB,expr,UB,Con_23);
                if(h!=j){
                model.add(Con23);
                }
                Con23t.add(Con23);
                expr.end();
                    }
                 Con23it.add(Con23t);
         }
     Con23hit.add(Con23it);
     }
  Con23jhit.add(Con23hit);
}
//------------------------- Constraint 24 ---------------------------------------
IloRangeArray Con24j(env,0);
    for (j=0;j<jmax;j++){
        IloExpr expr(env,0);
           for (t=0;t<ATj[j];t++){
expr+=WSCjt[j][t]+WCCjt[j][t];
                        }
                char Con_24[60];
```

```
                sprintf(Con_24,"Con_24j(j%d)",j);
                float LB=0,UB=0;
                IloRange Con24(env,LB,expr,UB,Con_24);
                model.add(Con24);
                Con24j.add(Con24);
                expr.end();
 }

//------------------------- Constraint 25 ---------------------------------------
IloRangeMatrix2x2  Con25ji(env,0);
       for (j=0;j<jmax;j++){
       IloRangeArray Con25i(env,0);
        for (i=0;i<imax;i++){
                IloExpr expr(env,0);
                expr+=M*(1-CCji[j][i]);
                  for (t=0;t<Si[i];t++){
        expr+=-WCCjt[j][t];
                      }
                char Con_25[60];
                sprintf(Con_25,"Con_25ji(j%d,i%d)",j,i);
                float LB=0,UB=IloInfinity;
                IloRange Con25(env,LB,expr,UB,Con_25);
                model.add(Con25);
                Con25i.add(Con25);
                expr.end();
        }
                Con25ji.add(Con25i);
}

//------------------------- Constraint 26 ---------------------------------------
int n=0;
for (i=0; i<imax; i++){
for (j=0;j<jmax;j++){
                                        n+=HTij[i][j];
                                }
          }
IloRangeMatrix3x3 Con26jit(env,0);
   for (j=0;j<jmax;j++){
   IloRangeMatrix2x2  Con26it(env,0);
          for (i=0;i<imax;i++){
          IloRangeArray Con26t(env,0);
           for (t=0;t<tmax-n;t++){
               IloExpr expr(env,0);
expr+=HTij[i][j]+M*((1-SCji[j][i])+(1-WSCjt[j][t]));
                  for (n=0;n<HTij[i][j];n++){
expr+=-WCCjt[j][t+n];
                  }
                char Con_26[60];
                sprintf(Con_26,"Con_26jit(j%d,i%d,t%d)",j,i,t);
                float LB=0,UB=IloInfinity;
                IloRange Con26(env,LB,expr,UB,Con_26);
                model.add(Con26);
                Con26t.add(Con26);
                expr.end();
                 }
                Con26it.add(Con26t);
        }
  Con26jit.add(Con26it);
```

```
}

//------------------------- Constraint 27 ---------------------------------------
int m=0;
for (i=0; i<imax; i++){
for (j=0;j<jmax;j++){
                                              n+=HTij[i][j];
                                       }
          }
IloRangeMatrix3x3 Con27jit(env,0);
   for (j=0;j<jmax;j++){
   IloRangeMatrix2x2  Con27it(env,0);
          for (i=0;i<imax;i++){
          IloRangeArray Con27t(env,0);
           for (t=0;t<tmax-n;t++){
              IloExpr expr(env,0);
expr+=-HTij[i][j]+M*((1-SCji[j][i])+(1-WSCjt[j][t]));
                   for (n=0;n<HTij[i][j];n++){
expr+=WCCjt[j][t+n];
                   }
              char Con_27[60];
              sprintf(Con_27,"Con_27jit(j%d,i%d,t%d)",j,i,t);
              float LB=0,UB=IloInfinity;
              IloRange Con27(env,LB,expr,UB,Con_27);
              model.add(Con27);
              Con27t.add(Con27);
              expr.end();
               }
              Con27it.add(Con27t);
        }
  Con27jit.add(Con27it);
}

//------------------------- Constraint 28 ---------------------------------------
IloRangeMatrix2x2  Con28ji(env,0);
       for (j=0;j<jmax;j++){
       IloRangeArray Con28i(env,0);
        for (i=0;i<imax;i++){
              IloExpr expr(env,0);
              expr+=CCji[j][i]-SCji[j][i];
              char Con_28[60];
              sprintf(Con_28,"Con_28ji(j%d,i%d)",j,i);
              float LB=0,UB=IloInfinity;
              IloRange Con28(env,LB,expr,UB,Con_28);
              model.add(Con28);
              Con28i.add(Con28);
              expr.end();
        }
              Con28ji.add(Con28i);
}
//------------------------- Constraint 29 ---------------------------------------
IloRangeMatrix2x2  Con29jt(env,0);
       for (j=0;j<jmax;j++){
       IloRangeArray Con29t(env,0);
        for (t=0;t<tmax;t++){
              IloExpr expr(env,0);
              expr+=WCCjt[j][t]-WSCjt[j][t];
              char Con_29[60];
```

```
            sprintf(Con_29,"Con_29jt(j%d,t%d)",j,t);
            float LB=0,UB=IloInfinity;
            IloRange Con29(env,LB,expr,UB,Con_29);
            model.add(Con29);
            Con29t.add(Con29);
            expr.end();
      }
            Con29jt.add(Con29t);
}

//------------------------- Constraint 30 --------------------------------------
IloRangeMatrix2x2  Con30jt(env,0);
       for (j=0;j<jmax;j++){
       IloRangeArray Con30t(env,0);
        for (t=1;t<tmax;t++){
            IloExpr expr(env,0);
            expr+=WSCjt[j][t]-WCCjt[j][t]+WCCjt[j][t-1];
            char Con_30[60];
            sprintf(Con_30,"Con_30jt(j%d,t%d)",j,t);
            float LB=0,UB=IloInfinity;
            IloRange Con30(env,LB,expr,UB,Con_30);
            model.add(Con30);
            Con30t.add(Con30);
            expr.end();
      }
            Con30jt.add(Con30t);
}

//--------------------------------------------------------------------------------
//---------------------------------Objective Function----------------------------
//--------------------------------------------------------------------------------
IloExpr expr456(env);
for (j=0;j<jmax;j++){
       for (t=0;t<tmax;t++){
              for  (i=0;i<imax;i++){
              expr456+=((t+1)*SCji[j][i]*HTij[i][j]);
        }
    }
}

model.add(IloMinimize(env, expr456));
expr456.end();
```