25/5/2022

ORTA DOĞU TEKNİK ÜNİVERSİTESİ
MIDDLE EAST TECHNICAL UNIVERSITY

IE 252 – Network Flows and Integer Programming

Case Study 1

GROUP 7

Abderrahmane Harkat	2415297
Hasancan Özkalay	2233807
Kerem Genelioğlu	2234755
Youssef Nsouli	2487494

Table of Contents

Introduction

Philip is an industrial engineer at a pharmaceutical product distribution company, Drug-Pro. Philip needs to present Drug-Pro's board with some development project aiming at optimizing the distribution, control, and activities of the company in order for the rivalry to bite the dust. Not only that, but also Philip needs to also comply by the MHRA's regulation in every decision he needs to take.

Given the data and tasks at hand, Philip will need to use multiple algorithms and methods to tackle these problems. These methods will range from simple, solved-by-hand algorithms to sophisticated software like GAMS and Python.

Philip's solutions are provided in this report, from the transport of the drugs between Novensis and Drug-Pro's central warehouse; to the optimal transportation route between the supplier, warehouses and retailer; passing by the optimal layout connecting all fixed points to the central alarm system for the warehouse the company is buying; and the completion time of the warehouse's renovation with the critical path.

Task 1

It is in Phillip's interest to minimize the route between Novensis' warehouse and Drug-Pro's retail store in Brentwood. Finding the shortest route by enumeration is certainly not feasible, as there are at least $2^{no\ of\ roads}$ solutions. So, representing this problem as an integer programming model seems like a good solution.

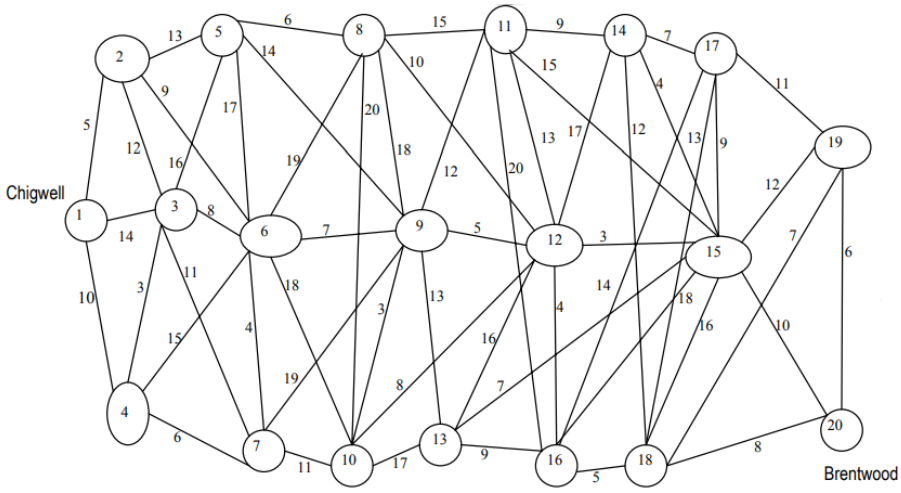


Figure 2: Network of Roads from Chigwell to Brentwood with the time needed to cross each road (in minutes).

Fortunately, since the model is a MCNFP, the constraint matrix is totally unimodular; its determinant is either -1, 0, or 1. So, the IP model can be relaxed into a LP, allowing Philip to use the LP optimization package, offered by the GAMS software.

The network must be transformed into a data structure comprehendible by GAMS to be able to use it. So, the network is transformed to a table (or 2-D matrix) where element (i, j) contains the time needed to traverse arc (i, j). In order to prevent GAMS from using arcs which do not exist in the network, a very large positive number (64-bit max integer number) is assigned to (i, j) if arc $(i\ j)$ is not part of the set containing all feasible arcs.

With the assumptions of proportionality, divisibility, additivity, and certainty, the model is as follows:

Mathematical Model:

Sets and Indices:

$i \in I = \{1, 2, \ldots 20\}$: Represents node i

$j \in J = \{1, 2, \ldots 20\}$: Representing node$j$, independent from node i

A: set of all feasible arcs

(i, j): Represents the arc from node i to node j for all arcs in A

Parameters:

t_{ij}: Time (in minutes) to traverse from node i to node j

Decision Variables:

x_{ij}: Represents whether the arc (i, j) has been traversed. If traversed, $x_{ij} = 1$. Otherwise, $x_{ij} = 0$.

Objective Function:

$$\textbf{Min}\ z = \sum_{(i,j)\in A} t_{ij} x_{ij}$$

$$\textbf{Min}\ z = [\text{Time taken to traverse each arc}] \times [\text{Arcs traversed}]$$

Subject to:

$$\sum_{j\in J} x_{ij} - \sum_{j\in J} x_{ij} = \begin{cases} 1, \text{for } i \in \text{Source Node} \\ 0, \forall\, i \notin \{\text{Source Node}, \text{Sink Node}\} \\ -1, \text{for } i \in \text{Sink Node} \end{cases}$$

Sign Restrictions:

$$x_{ij} \geq 0;$$

Since the constraint matrix is unimodular, it is possible to relax the model to use the simplex method to solve the model.

Results:

After solving the mathematical model that's presented above, the quickest route from Chigwell (or the supplier, Novensis) appears to have an average of 39 minutes commute time. The truck follows this route (also shown in figure 2):

$1 \rightarrow 2 \rightarrow 6 \rightarrow 9 \rightarrow 12 \rightarrow 15 \rightarrow 20.$

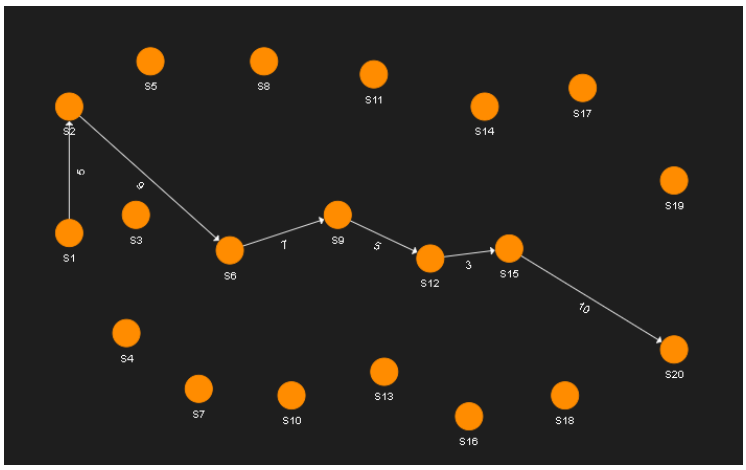


Figure 2: Optimal route-to-be-taken by the truck from Chigwell (S1 in figure) to Brentwood (S20).

Given a network of routes and possible road streams, Dijkstra's Algorithm is another way and is one of the best algorithms to use for this problem. To utilize Dijkstra's Algorithm, a Python script will be used (DijkstraAlgorithm.PY). The network will also be turned into a matrix (seen in Diagram_1.csv) and the same trick regarding non-existent arcs will be used. The output (seen by running Solution.PY) is as follows:

Nodes:	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20
Iter. 0:	(S1, 0/F)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)
Iter. 1:	(S1, 0/F)	(S1, 5/F)	(S1, 14)	(S1, 10)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)
Iter. 2:	(S1, 0/F)	(S1, 5/F)	(S1, 14)	(S1, 10/F)	(S2, 18)	(S2, 14)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)
Iter. 3:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18)	(S2, 14)	(S4, 17)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)
Iter. 4:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18)	(S2, 14/F)	(S4, 17)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)
Iter. 5:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18)	(S2, 14/F)	(S4, 17/F)	(S6, 33)	(S6, 21)	(S6, 32)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)
Iter. 6:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18/F)	(S2, 14/F)	(S4, 17/F)	(S6, 33)	(S6, 21)	(S7, 28)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)
Iter. 7:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18/F)	(S2, 14/F)	(S4, 17/F)	(S5, 24)	(S6, 21/F)	(S7, 28)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)
Iter. 8:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18/F)	(S2, 14/F)	(S4, 17/F)	(S5, 24/F)	(S6, 21/F)	(S9, 24)	(S9, 33)	(S9, 26)	(S9, 34)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)
Iter. 9:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18/F)	(S2, 14/F)	(S4, 17/F)	(S5, 24/F)	(S6, 21/F)	(S9, 24/F)	(S9, 33)	(S9, 26)	(S9, 34)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)
Iter. 10:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18/F)	(S2, 14/F)	(S4, 17/F)	(S5, 24/F)	(S6, 21/F)	(S9, 24/F)	(S9, 33)	(S9, 26/F)	(S9, 34)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)	(-, M)
Iter. 11:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18/F)	(S2, 14/F)	(S4, 17/F)	(S5, 24/F)	(S6, 21/F)	(S9, 24/F)	(S9, 33)	(S9, 26/F)	(S9, 34)	(S12, 43)	(S12, 29/F)	(S12, 30)	(-, M)	(-, M)	(-, M)	(-, M)
Iter. 12:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18/F)	(S2, 14/F)	(S4, 17/F)	(S5, 24/F)	(S6, 21/F)	(S9, 24/F)	(S9, 33)	(S9, 26/F)	(S9, 34)	(S15, 33)	(S12, 29/F)	(S12, 30/F)	(S15, 38)	(S15, 45)	(S15, 41)	(S15, 39)
Iter. 13:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18/F)	(S2, 14/F)	(S4, 17/F)	(S5, 24/F)	(S6, 21/F)	(S9, 24/F)	(S9, 33/F)	(S9, 26/F)	(S9, 34)	(S15, 33)	(S12, 29/F)	(S12, 30/F)	(S15, 38)	(S16, 35)	(S15, 41)	(S15, 39)
Iter. 14:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18/F)	(S2, 14/F)	(S4, 17/F)	(S5, 24/F)	(S6, 21/F)	(S9, 24/F)	(S9, 33/F)	(S9, 26/F)	(S9, 34)	(S15, 33/F)	(S12, 29/F)	(S12, 30/F)	(S15, 38)	(S16, 35)	(S15, 41)	(S15, 39)
Iter. 15:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18/F)	(S2, 14/F)	(S4, 17/F)	(S5, 24/F)	(S6, 21/F)	(S9, 24/F)	(S9, 33/F)	(S9, 26/F)	(S9, 34/F)	(S15, 33/F)	(S12, 29/F)	(S12, 30/F)	(S15, 38)	(S16, 35)	(S15, 41)	(S15, 39)
Iter. 16:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18/F)	(S2, 14/F)	(S4, 17/F)	(S5, 24/F)	(S6, 21/F)	(S9, 24/F)	(S9, 33/F)	(S9, 26/F)	(S9, 34/F)	(S15, 33/F)	(S12, 29/F)	(S12, 30/F)	(S15, 38)	(S16, 35/F)	(S15, 41)	(S15, 39)
Iter. 17:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18/F)	(S2, 14/F)	(S4, 17/F)	(S5, 24/F)	(S6, 21/F)	(S9, 24/F)	(S9, 33/F)	(S9, 26/F)	(S9, 34/F)	(S15, 33/F)	(S12, 29/F)	(S12, 30/F)	(S15, 38/F)	(S16, 35/F)	(S15, 41)	(S15, 39)
Iter. 18:	(S1, 0/F)	(S1, 5/F)	(S4, 13/F)	(S1, 10/F)	(S2, 18/F)	(S2, 14/F)	(S4, 17/F)	(S5, 24/F)	(S6, 21/F)	(S9, 24/F)	(S9, 33/F)	(S9, 26/F)	(S9, 34/F)	(S15, 33/F)	(S12, 29/F)	(S12, 30/F)	(S15, 38/F)	(S16, 35/F)	(S15, 41)	(S15, 39/F)

Figure 3: Output of Solution.PY concerning Dijkstra’s Algorithm

The problem was solved in 18 iterations. Tracing back from the final node, $S20$, the path taken by Dijkstra’s Algorithm turns out to be identical to the one found by the simplex algorithm:

$S1 \rightarrow S2 \rightarrow S6 \rightarrow S9 \rightarrow S12 \rightarrow S15 \rightarrow S20.$

Task 2

In order to connect all key points to ensure maximum security and lowest possible cost, a minimum spanning tree needs to be made from the 15 nodes. Said tree should not contain cycling paths, since that indicates that the cost can be minimized further, and connecting security points with more than one wire can be destructive, since that key point will have multiple voltage sources, causing it to burn out or causing a short circuit. The network is shown here:

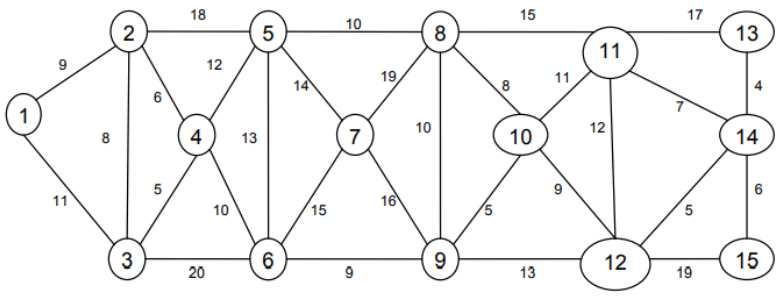


Figure 4: The network of the amount of cable needed to connect security key points (in yards).

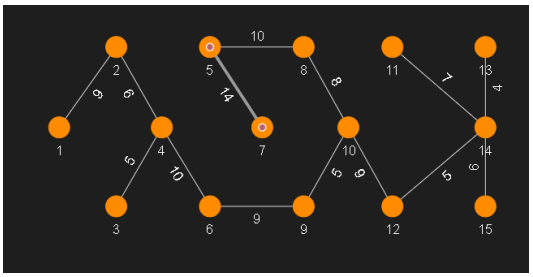
To tackle this, Kruskal's algorithm is used. The manually-worked iterations are in the word file, *Kruskal's Algorithm*. The arcs were listed in increasing order as the algorithm suggests, and arcs were added in such a way that no cycling occurs. Crossed-out arcs indicate that they cause cycling and are eliminated. The value, MST, indicates that amount of cable used so far. $LIST$ is the set of arcs that constitute the Minimum-spanning tree. The iterations are listed below, in figure 5, with their corresponding networks in Appendix A.

In Increasing Order

$(13, 14): 4$	$MST = 4$
$(12, 14): 5$	$MST = 9$
$(9, 10): 5$	$MST = 14$
$(3, 4): 5$	$MST = 19$
$(2, 4): 6$	$MST = 25$
$(14, 15): 6$	$MST = 31$
$(11, 14): 7$	$MST = 38$
~~$(2, 3): 8$~~	
$(8, 10): 8$	$MST = 46$
$(1, 2): 9$	$MST = 55$
$(6, 9): 9$	$MST = 64$
$(10, 12): 9$	$MST = 73$
~~$(8, 9): 10$~~	
$(4, 6): 10$	$MST = 83$
$(5, 8): 10$	$MST = 93$
~~$(10, 11): 11$~~	
~~$(1, 3): 11$~~	
~~$(11, 12): 12$~~	
~~$(4, 5): 12$~~	
~~$(5, 6): 13$~~	
~~$(9, 12): 13$~~	
$(5, 7): 14$	$MST = 107$

Ignore the rest (all nodes connected)

~~$(8, 11): 15$~~
~~$(6, 7): 15$~~
~~$(7, 9): 16$~~
~~$(11, 13): 17$~~
~~$(2, 5): 18$~~
~~$(7, 8): 19$~~
~~$(12, 15): 19$~~
~~$(3, 6): 20$~~



LIST:

$(13, 14): 4$	$MST = 4$
$(12, 14): 5$	$MST = 9$
$(9, 10): 5$	$MST = 14$
$(3, 4): 5$	$MST = 19$
$(2, 4): 6$	$MST = 25$
$(14, 15): 6$	$MST = 31$
$(11, 14): 7$	$MST = 38$
$(8, 10): 8$	$MST = 46$
$(1, 2): 9$	$MST = 55$
$(6, 9): 9$	$MST = 64$
$(10, 12): 9$	$MST = 73$
$(4, 6): 10$	$MST = 83$
$(5, 8): 10$	$MST = 93$
$(5, 7): 14$	$MST = 107$

Figure 5: Summary of Kruskal's Algorithm, and the final minimum spanning tree.

As seen, the algorithm was terminated as soon as all nodes are connected (i.e.: $n - 1$ arcs were added). The result turned out to be that 107 yards of cable are necessary.

Task 3

Philip now needs to renovate and automate the warehouse and has 11 activities that need to be done in the shortest time possible. The activities are shown in table 6:

Activity	Description	Immediate Predecessor	Time (in Weeks)
A	Determine new warehouse specifications	-	5
B	Obtain tenders for warehouse fitting	-	9
C	Select vendors	A, B	3
D	Order computerized surveillance system	C	12
E	Order computerized stock control interface	C	6
F	Install stock control interface	E	5
G	Train operators on stock control software	F	8
H	Construction of external fencing and Closed Circuit TV cameras	C	7
I	Interfacing of security system and alarm	D, H	4
J	Test all computer systems (stock and security)	G, I	5
K	Validation by MRHA inspectors	J	4

Table 6: Description of activities, their immediate predecessors, and time needed to complete.

As seen, each activity has a(n assumingly) fixed duration, and immediate predecessors. The path of activities and roadway to renovate the warehouse will be represented in a network, using the Arc-on-Activity representation, in figure 7. Since some activities have multiple immediate predecessors, a dummy arc will be used to indicate the predecessor relationships, like the arc from node 2 to node 3.

To solve this network using GAMS, the network will be transformed into a table, assigning large numbers to non-existent arcs. However, this time, the constraints (in GAMS) check the weight of the node, effectively ignoring them. For example, if the weight of the node is very large, GAMS will not include that arc in the set of constraints.

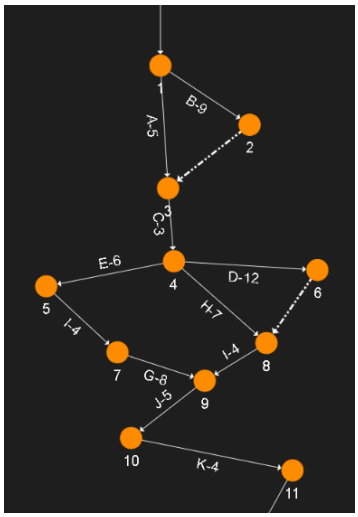


Figure 7: Arc-on-Activity representation of the warehouse renovation project.

Mathematical Model:

Sets and Indices:

$i \in I = \{1, 2, \dots 11\}$: Event i

$j \in J = \{1, 2, \dots 11\}$: Event j, independent from event i

A: Set of all feasible arcs

(i, j): Arc from event i to event j, representing an activity taking place

Parameters:

D_{ij}: Duration of the activity (in weeks) placed between event i and event j

Decision Variables:

t_i: The time where event i occurs (in weeks)

Objective Function:

$$\min z = t_{(11)} - t_1$$

$$\min z = [\text{time to end project};\ \text{time between project end and project beginning}]$$

Subject to:

$t_j - t_i \geq D_{ij} \; \forall \; (i,j) \in A$ (Activity-on-Arc constraint)

The time between two events should be at least equal to the duration of the activity contained in the arc.

$t_1 = 0$ (Initial time constraint)

Sets initial time. This allows Philip to see exactly which week an activity will start on.

Sign Constraints:

$t_i \; urs \; \forall \; i$

Result:

The result in GAMS is shown in table 8 below; the time to complete the project is 40 weeks:

Activity	Time of Event
t_1	0
t_2	9
t_3	9
t_4	12
t_5	18
t_6	24
t_7	23
t_8	24
t_9	31
t_{10}	36
t_{11}	40

Table 8: Time at which each event happens.

However, GAMS cannot inform Philip whether an activity can be delayed or not, or whether an activity is critical or not. It does not show the critical path either. So, the critical path algorithm will be implemented in Python (in CriticalPath.PY, appendix):

Activity	Earliest Start	Earliest Finish	Latest Start	Latest Finish	Total Slack	Free Slack
A	0	5	4	9	4	4
B	0	9	0	9	0	0
C	9	12	9	12	0	0
D	12	24	15	27	3	0
E	12	18	12	18	0	0
F	18	23	18	23	0	0
G	23	31	23	31	0	0
H	12	19	20	27	8	5
I	24	28	27	31	3	3
J	31	36	31	36	0	0
K	36	40	36	40	0	0

Table 9: Earliest start and finish, and latest start and finish of every activity, and their slack.

So, the critical path is: $B \rightarrow D \rightarrow E \rightarrow F \rightarrow G \rightarrow J \rightarrow K$. Any delay in these activities will cause a delay in the project end-time. The Gantt chart can now be made. Of course, critical activities have their least delay and utmost delay overlapping, since they should not be delayed at all:

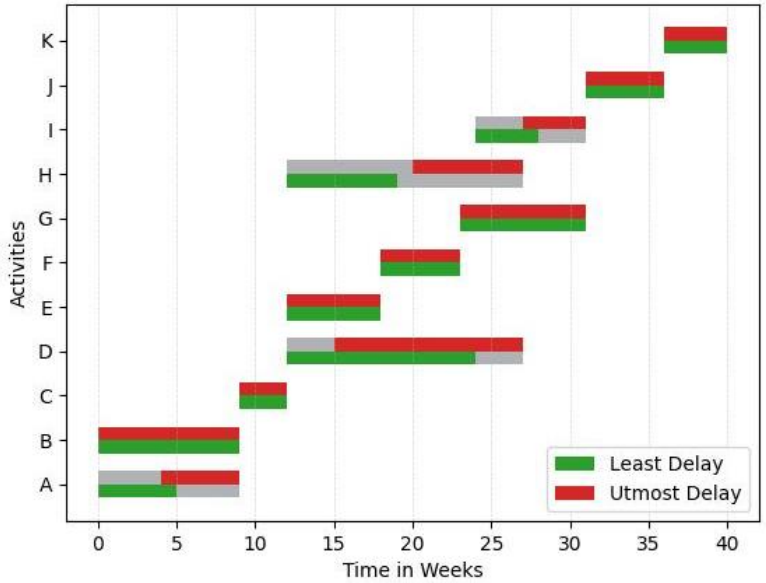


Figure 10: Gantt chart of the warehouse renovation project.

Task 4

Philip needs to distribute drugs from plants to retail stores while keeping costs as low as possible. There are two plants in Chigwell and Cambridge which will deliver drugs to three warehouses in Brentwood, Reigate, and Peterborough. The warehouses will then deliver to 16 retail stores across south England.

The plants in Chigwell and Cambridge have a supply of 12 and 7 million packets of drugs for a combined total of 17 million packets. The warehouses are assumed to have *unlimited* capacity, and the demand of the Better 4U retail stores are as follows.

Better 4U Retail Outlets	Demand (millions)
Luton	0.5
Northampton	1
Milton Keynes	0.2
Peterborough	0.7
Cambridge	1.4
Norwich	1.2
Ipswich	0.9
Chelmsford	0.4
Maidstone	1.8
Canterbury	2.1
Ashford	0.6
Royal Tunbridge Wells	1.3
Brighton	0.1
Worthing	1.5
Guildford	2
Winchester	0.3

Table 11: Demand of Better 4U retail stores in millions of packets.

The transportation costs, given in £ per thousand packets, are shown here:

	Warehouse		
Plant	Brentwood	Reigate	Peterborough
Chigwell	1	9	12
Cambridge	8	13	6

Table 12: Transportation costs (in £/1000 packets) between plants and warehouses.

	Better 4U Retail Stores															
Warehouse	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Brentwood	7	11	10	12	8	15	7	1	4	8	8	6	10	11	9	13
Reigate	9	13	12	19	13	21	14	8	4	10	8	4	6	7	3	7
Peterborough	8	6	7	1	6	11	12	11	17	20	20	18	22	23	17	23

Table 12: Transportation costs (in £/1000 packets) between warehouses and retail stores.

With certainty, divisibility, additivity, and proportionality assumptions in mind, the model is as follows:

Mathematical Model:

Sets and Indices:

$p \in P = \{1, 2\}$: Plant p

$w \in W = \{1, 2, 3\}$: Warehouse w

$r \in R = \{1, 2, \dots 16\}$: Retail store r

Parameters:

s_p: Supply of plant p

d_r: Demand of retail store r

t_{pw}^{P-W}: Transportation cost of 1000 packets from plant p to warehouse w

t_{wr}^{W-R}: Transportation cost of 1000 packets from warehouse w to retail store r

Decision Variables:

x_{pwr}: Amount of packets (in thousands) transported from plant p to warehouse w, and finally to retail store r

Objective Function:

$$\textbf{Min}\, z = \sum_{p\in P, w\in W, r\in R} \left(t_{pw}^{P-W} + t_{wr}^{W-R}\right) x_{pwr}$$

$$\textbf{Min}\, z = [\text{additive transp. costs}] \times [\text{amount transported}]$$

Subject to:

$\sum_{w\in W, r\in R} x_{pwr} \leq s_p \; \forall\, p \in P$ (Plant supply constraint)

Amount transported from each plant shouldn't be more than the supply

Since the capacity of each warehouse is considered to be unlimited, as well as holding costs to be negligible, no constraint will be added for warehouses.

$\sum_{p\in P, w\in W} x_{pwr} \geq d_r \; \forall\, r \in R$ (Retail store demand constraint)

Amount delivered to each retail store should surpass the demand

Sign Restrictions:

$x_{pwr} \geq 0 \; \forall\, p \in P, w \in W, r \in R$

Result:

The total cost is $z = 152900$ sterling pounds, the distribution of drugs is shown in the following table (next page):

		r															
$p=1$	w	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	0.5				0.5		0.9	0.4	1.8	2.1	0.6	1.3	0.1	1.5	2	0.3
	2																
	3																
$p=2$	w	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1																
	2																
	3		1	0.2	0.7	0.9	1.2										

Table 13: Optimal solution of the transshipment problem.

A notable remark is that each plant sends to one warehouse and warehouse 2 is never used in this scenario. This is to be expected, since the transportation costs to and from warehouse 2 are much higher. As for the former (plant to one warehouse only), it is actually more preferable, since packaging, handling, and unpacking to one warehouse and from one plant is easier than doing the same process over many warehouses.

However, the scenario holds only for warehouses with (presumably) unlimited capacities. This is, of course, an unreasonable scenario, since a warehouse stores many and multiple items and goods, not only those belonging to Drug-Pro. A good question might be to study the network if warehouses had a hard-set limit.

Adding a constraint, $\sum_{p\in P, r\in R} x_{pwr} \leq C \; \forall \, w \in W$, where C is the capacity of the warehouse (in millions), Philip achieves the following results for a value of C:

$C = 10, \qquad z = 156900,$

$C = 8, \qquad z = 163800,$

$C = 5, \qquad$ *Infeasible,*

A detailed representation of the optimal solution is presented in Appendix B1 and B2.

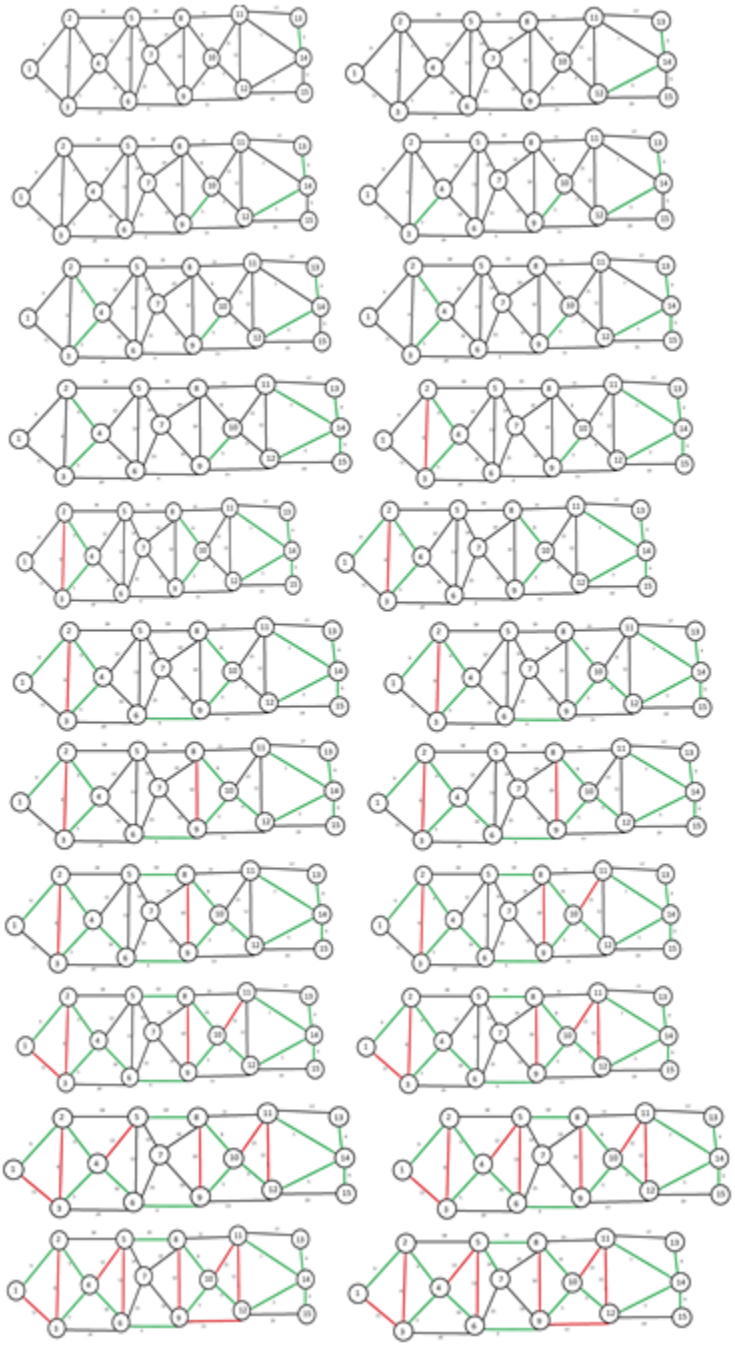
So, it seems that decreasing the capacity of warehouses will increase the objective function until it becomes infeasible.

Conclusion

In conclusion, the board will be satisfied with Phillip's work since it will save the company a very considerable amount time and resources that can be used now to focus on tackling other problems, like the one suggested (the implementation of these optimization methods for all clients, suppliers and warehouses of this company). On top of that, Philip was able to still comply with the MHRA regulations. This industrial engineer is very versatile since he could define optimal routes, optimal amount of resources used and the schedule of a renovation activity.

One can actually wonder, while this might be just a fractional amount of the impact an industrial engineer can have on businesses and societies, why are we still polluting and wasting resources and money, whilst harming our planet, ourselves and others. Is the world not ready yet for a reform in production, is the human too greedy or are industrial engineers just shy?

Appendix A



Appendix A1: Kruskal's Algorithm's iterations, explicitly on the security network. Iterations go from top to bottom, left to right.

Appendix B

		r															
$p=1$	w	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	0.5				0.5		0.9	0.4	1.8	2.1	0.6	1.3	0.1	1.5		0.3
	2															2	
	3																
$p=2$	w	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1																
	2																
	3		1	0.2	0.7	0.9	1.2										

Appendix B1: Distribution of drugs optimally when $C = 10$.

		r															
$p=1$	w	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	0.5						0.9	0.4	1.8	2.1	0.6	1.3	0.1	0.3		
	2														1.2	2	3
	3																
$p=2$	w	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1																
	2																
	3		1	0.2	0.7	1.4	1.2										

Appendix B2: Distribution of drugs when $C = 8$.

Interestingly, in all cases, it seems the algorithm tries to minimize the usage of warehouse 2. It is safe to say that it is due to the high cost of transporting from and to the second warehouse. In addition, plant 1 prefers warehouse 1, due to the 1 pound per 1000 packets cost as opposed to 9 and 12 for warehouses 2 and 3, respectively, and for the same reason, plant 3 prefers warehouse 3 (6 as opposed to 8 and 13 pounds/1000 for warehouses 3, 1, and 2 respectively).