Large Scale Optimization

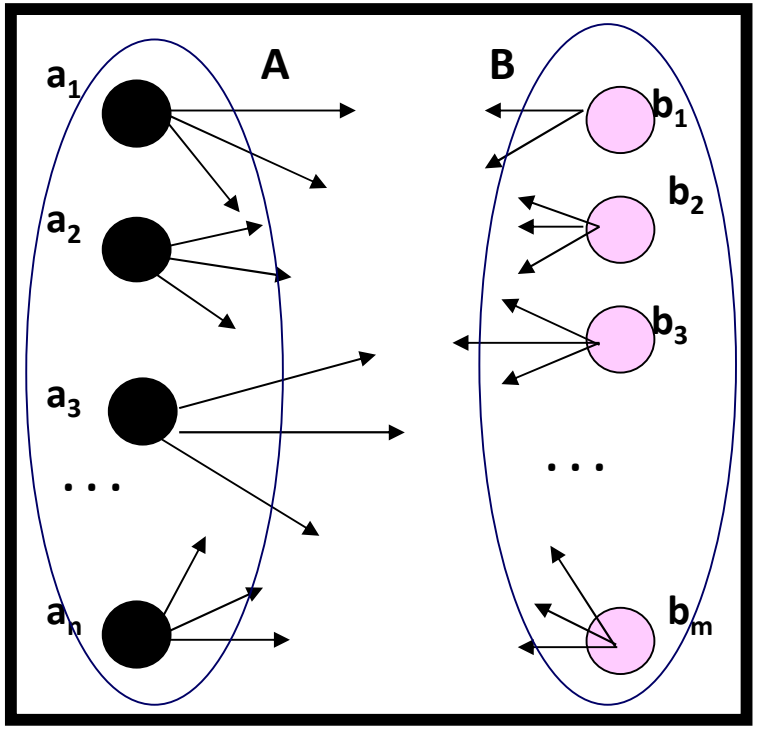
Emmanouil Zachariadis, Assistant Professor
Department of Management Science and Technology, Athens University of Economics and Business
E: ezach@aueb.gr
A: Evelpidon 47A and Lefkados 33 street, 9th floor, Room 906, 11362, Athens, Greece
T: +30-210-8203 674

2nd Combinatorial Structure: Assignment

- Assignment Problems:

  The solution structure of an assignment problem is composed by assignments between elements of two (or multiple) sets

- Different assignments between elements represent different solutions



Assignment Problem Examples

- Assignment Problems call for the identification of the optimal assignment between elements of different sets, so that an operational objective achieved

- Assignment Problem Applications:
  - Assignment of facilities to locations
  - Assignment of employees to timeslots
  - Assignment of products to shelves
  - Assignment of tourists to buses
  - Assignment of TV commercials to time zones

BIN PACKING PROBLEMS

- Several Applications of Management Science call for the insertion of items into larger containers
  - Products into Containers
  - Tasks into Shifts
  - Tourists into Buses
- Such problems are called packing problems
- Solution Structure: Assignment of items to containers
- The packing problems can have multiple objectives
  - Maximization of space utilization
  - Stability of the cargo
  - Easy unloading operations
  - …

BIN PACKING PROBLEM - BPP

- Bin Packing Problem
- One of the most basic assignment problems
- Objective: Insert *all* items into containers (bins), so that the number of containers used is minimized
  - Items: Various sizes
  - Containers: Same size

BIN PACKING PROBLEM - BPP

- BPP Formal Description:
  - Set of n items J
  - Each item $j \in J$ has size w_j $(j = \{1, \dots, n\})$
  - Consider a set of containers each with maximum capacity Q
  - The total weight of all items assigned to the same container must not exceed the capacity Q
  - Therefore, for a BPP to be feasible, the following: $w_j \leq Q, \forall j = 1, \dots, n$
  - Objective of the problem is to minimize place (assign) all items to containers so that the number of used containers is minimized

BIN PACKING PROBLEM - BPP

- Suggest an obvious lower bound for the objective function.
  - What is a lower bound of the number of containers needed?
- Suggest an obvious upper bound for the objective function?
  - What is an upper bound of the number of containers needed?

BIN PACKING PROBLEM - BPP

- Lower Bound on the number of containers needed

$$LB = \left\lceil \frac{\sum_{i=1}^{n} w_i}{Q} \right\rceil$$

- Lower Bound on the number of containers needed

$$UB = n$$

BIN PACKING PROBLEM - BPP

- Objective Function:

$$minimize\ B = \sum_{i=1}^{n} y_i$$

- Constraints:

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad \forall j \in \{1, \dots, n\}$$

$$\sum_{j=1}^{n} w_j \cdot x_{ij} \leq Q \cdot y_i, \qquad \forall i \in \{1, \dots, n\}$$

$y_i \in \{0,1\}, \qquad \forall i \in \{1, \dots, n\}$ The container i is used

$x_{ij} \in \{0,1\}, \qquad \forall i \in \{1, \dots, n\}, \forall j \in \{1, \dots, n\}$ Item j into Container i

A Greedy Algorithm for the BPP

- Solution Structure

  A set of :

  – $\boldsymbol{n}$ assignments

  – Each assignment*:*

  (Item j, Bin i)

- Solution Structure:

  A single assignment Item to Bin.

BPP – Greedy Algorithms

- **Online Bin-Packing**

Decisions are taken on the fly (as items arrive)

Subsequent items not known, when placement of an item is decided

- Two well known algorithms derived by two selection criteria:

- 1st Criterion: Assign the current item into the first bin that can feasibly accommodate the item (First-Fit Heuristic)

- 2nd Criterion: Assign the current item into the most utilized (fullest) bin that can feasibly accommodate the item (Best-Fit Heuristic)

BPP – Greedy Algorithms

Offline Bin-Packing

Decisions are taken when the complete item set is known

- The First-Fit and Best-Fit Algorithms are executed, after an initialization step:

  Arrange items in decreasing order of size

  (largest item first, smallest item last)

- When applying this ordering the algorithms are called
  - First-Fit Decreasing
  - Best-Fit Decreasing

BPP – Greedy Algorithms

- Objective Function:

  Number of non-empty bins after all items have been packed

BPP - Example

- 11 items with the following size:

$$w_1 = 8, w_2 = 7, w_3 = 14, w_4 = 9, w_5 = 6, w_6 = 9\ , w_7 = 2,$$
$$w_8 = 15, w_9 = 6, w_{10} = 7, w_{11} = 8$$

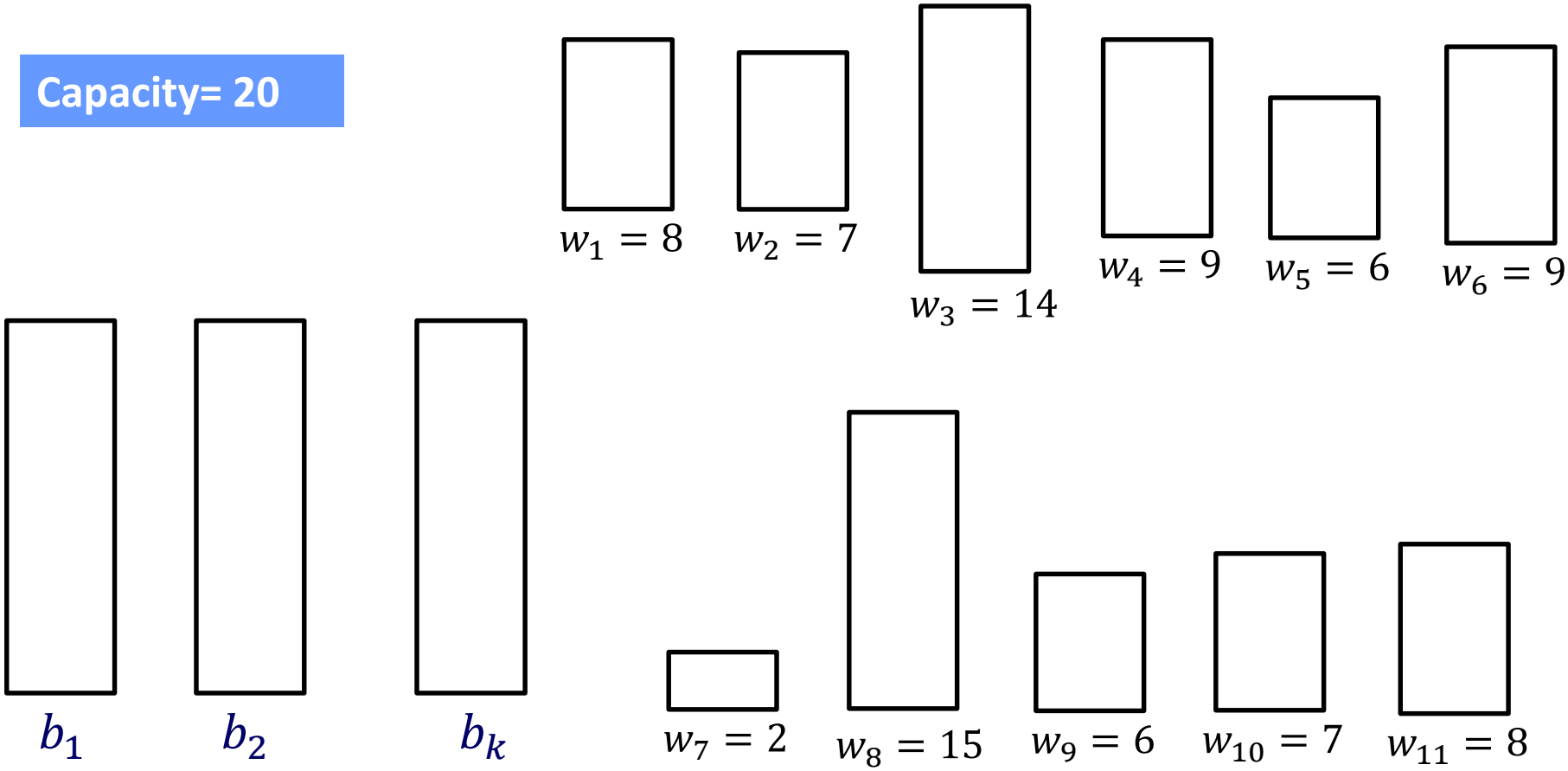
- Let $B = \{b_1, b_2, \dots, b_n\}$ be the container set, each with capacity $Q = 20$

- Our objective is to minimize the containers needed to store all items.

- Apply the First-Fit Decreasing Algorithm for the problem

BPP - Example

Capacity= 20



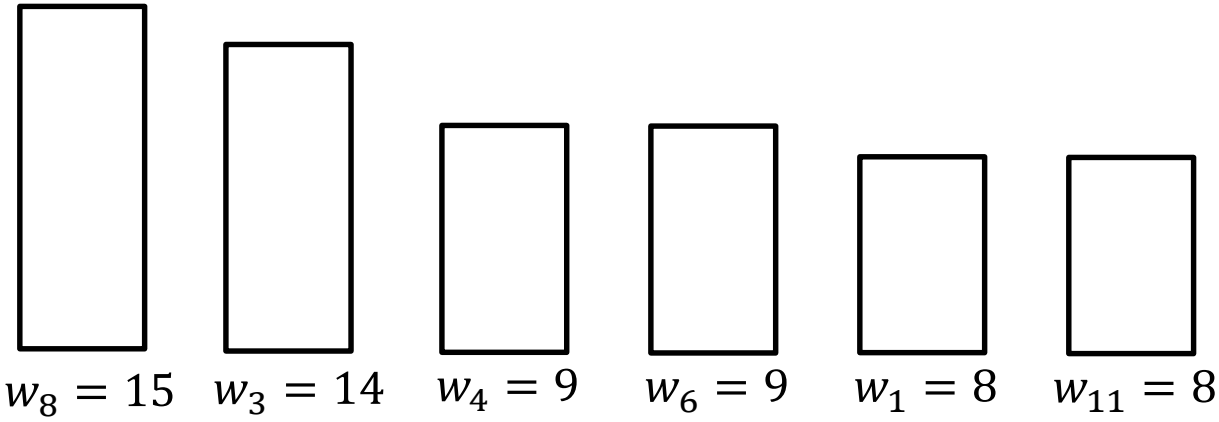
BPP - Example

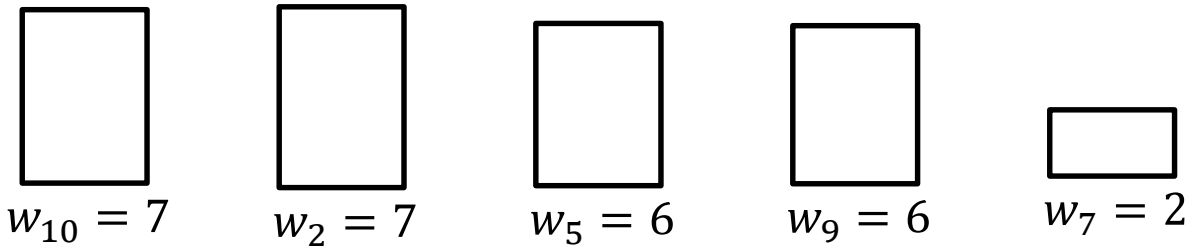
- Solution Structure: A set of 11 assignments of items to containers.
- Solution Feature: An item-container assignment
- Selection Criterion: Order the items in decreasing size order. Iteratively, select every item and insert it in the first container that can feasibly store it

  (First-Fit Decreasing)

BPP - Example

Ordering of Items





BPP - Example

Applying the First-Fit Decreasing Method:

- Iteration 1. Item $w_8 = 15$ inserted in Container 1,

$$s = \{(w_8, b_1)\}$$

- Iteration 2. Item $w_3 = 14$ inserted in Container 2,

$$s = \{(w_8, b_1), (w_3, b_2)\}$$

- Iteration 3. Item $w_4 = 9$ inserted in Container 3

$$s = \{(w_8, b_1), (w_3, b_2), (w_4, b_3)\}$$

BPP - Example

- Iteration 4. Item $w_6 = 9$ inserted in Container 3:
  $s = \{(w_8, b_1), (w_3, b_2), (w_4, b_3), (w_6, b_3)\}$

- Iteration 5. Item $w_1 = 8$ inserted in Container 4.
  $s = \{(w_8, b_1), (w_3, b_2), (w_4, b_3), (w_6, b_3), (w_1, b_4)\}$

- Iteration 6. Item $w_{11} = 8$ inserted in Container 4:

$s = \{(w_8, b_1), (w_3, b_2), (w_4, b_3), (w_6, b_3), (w_1, b_4), (w_{11}, b_4)\}$

BPP - Example

- Iteration 7. Item $w_2 = 7$ inserted in Container 5

$$s = \{(w_8, b_1), (w_3, b_2), (w_4, b_3), (w_6, b_3), (w_1, b_4), (w_{11}, b_4), (w_2, b_5)\}$$

- Iteration 8. Item $w_{10} = 7$ inserted in Container 5

$$s = \{(w_8, b_1), (w_3, b_2), (w_4, b_3), (w_6, b_3), (w_1, b_4), (w_{11}, b_4), (w_2, b_5), (w_{10}, b_5)\}$$

- Iteration 9. Item $w_5 = 6$ inserted in Container 2

$$s = \{(w_8, b_1), (w_3, b_2), (w_4, b_3), (w_6, b_3), (w_1, b_4), (w_{11}, b_4), (w_2, b_5), (w_{10}, b_5), (w_5, b_2)\}$$

BPP - Example

- Επανάληψη 10. Item $w_9 = 6$ inserted in Container 5

$$s = \{(w_8, b_1), (w_3, b_2), (w_4, b_3), (w_6, b_3), (w_1, b_4), (w_{11}, b_4), (w_2, b_5), (w_{10}, b_5), (w_5, b_2), (w_9, b_5)\}$$

- Επανάληψη 11. Item $w_7 = 2$ inserted in Container

$$s = \{(w_8, b_1), (w_3, b_2), (w_4, b_3), (w_6, b_3), (w_1, b_4), (w_{11}, b_4), (w_2, b_5), (w_{10}, b_5), (w_5, b_2), (w_9, b_5), (w_7, b_1)\}$$

BPP - Example

s :

b1:	w_8	w_7	
b2:	w_3	w_5	
b3:	w_4	w_6	
b4:	w_1	w_{11}	
b5:	w_2	w_{10}	w_9

Objective Function of the Final Solution $z(s) = 5$

BPP – Greedy Algorithm with alternative selection criterion

- Solving the same problem with the Best-Fit Decreasing algorithm
- Arrange the items in decreasing size order
- Iteratively, assign the current item into the most utilized (fullest) bin that can feasibly accommodate the item
- Most Utilized Container: The container that currently maximizes the total size of the accommodated items

BPP – Greedy Algorithm with alternative selection criterion

The greedy algorithm is applied to the problem as follows:

- Iteration 1. Item W_8=15 inserted in Container 1

  s: (W_8b1).

- Iteration 2. Item W_3=14 inserted in Container 2

  s: (W_8b1, W_3b2).

- Iteration 3. Item W_4=9 inserted in Container 3

  s: (W_8b1, W_3b2, W_4b3).

BPP – Greedy Algorithm with alternative selection criterion

- Iteration 4. Item W_6=9 inserted in Container 3

s: (W_8b1, W_3b2, W_4b3, W_6b3)

- Iteration 5. Item W_1=8 inserted in Container 4

s: (W_8b1, W_3b2, W_4b3, W_6b3, W_1b4)

- Iteration 6. Item W_{11}=8 inserted in Container 4

s: (W_8b1, W_3b2, W_4b3, W_6b3, W_1b4, W_{11}b4)

BPP – Greedy Algorithm with alternative selection criterion

- Iteration 7. Item W_2=7 inserted in Container 5

s: (W_8b1, W_3b2, W_4b3, W_6b3, W_1b4, W_{11}b4, W_2b5).

- Iteration 8. Item W_{10}=7 inserted in Container 5

s: (W_8b1, W_3b2, W_4b3, W_6b3, W_1b4, W_{11}b4, W_2b5, W_{10}b5).

- Iteration 9. Item W_5=6 inserted in Container 2

s: (W_8b1, W_3b2, W_4b3, W_6b3, W_1b4, W_{11}b4, W_2b5, W_{10}b5, W_5b2).

- Iteration 10. Item W_9=6 inserted in Container 5

s: (W_8b1, W_3b2, W_4b3, W_6b3, W_1b4, W_{11}b4, W_2b5, W_{10}b5, W_5b2, W_9b5).

BPP – Greedy Algorithm with alternative selection criterion

- Iteration 11. Item W_7=2 inserted in Container 5

s: (W_8b1, W_3b2, W_4b3, W_6b3, W_1b4, W_{11}b4, W_2b5, W_{10}b5, W_5b2, W_9b5, W_7b3)

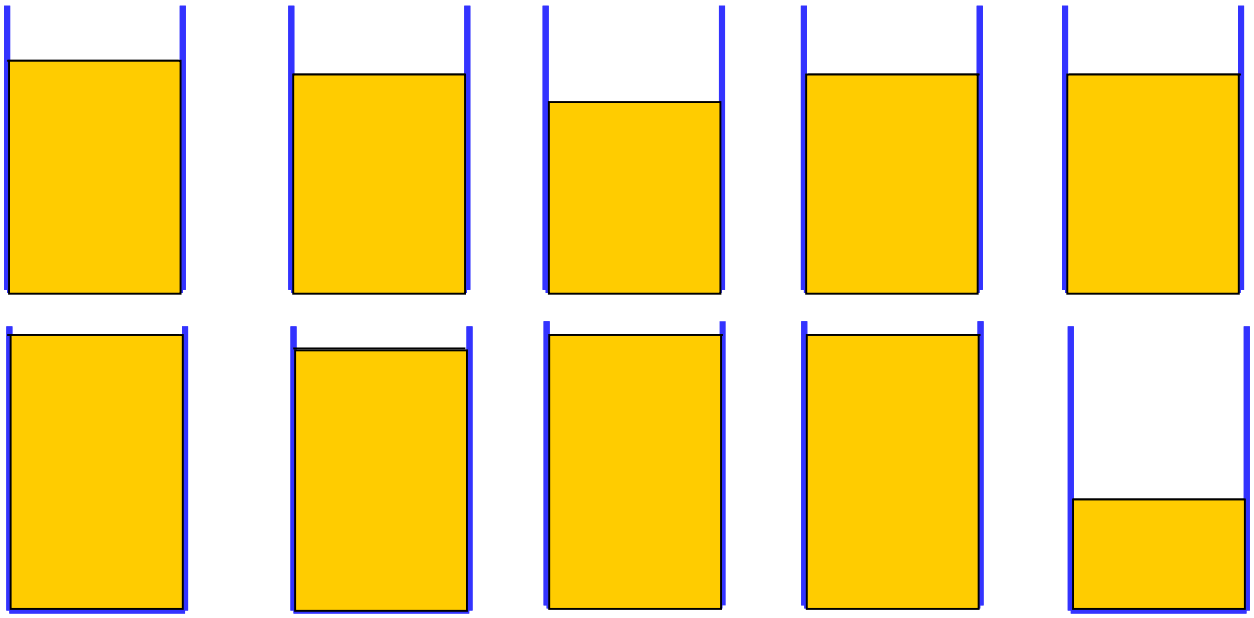
s :

b1:	W_8		
b2:	W_3	W_5	
b3:	W_4	W_6	W_7
b4:	W_1	W_{11}	
b5:	W_2	W_{10}	W_9

Objective Function of the Final Solution z(s)=5

Alternative Objective Function for the BIN PACKING PROBLEM

BPP – Alternative objective function

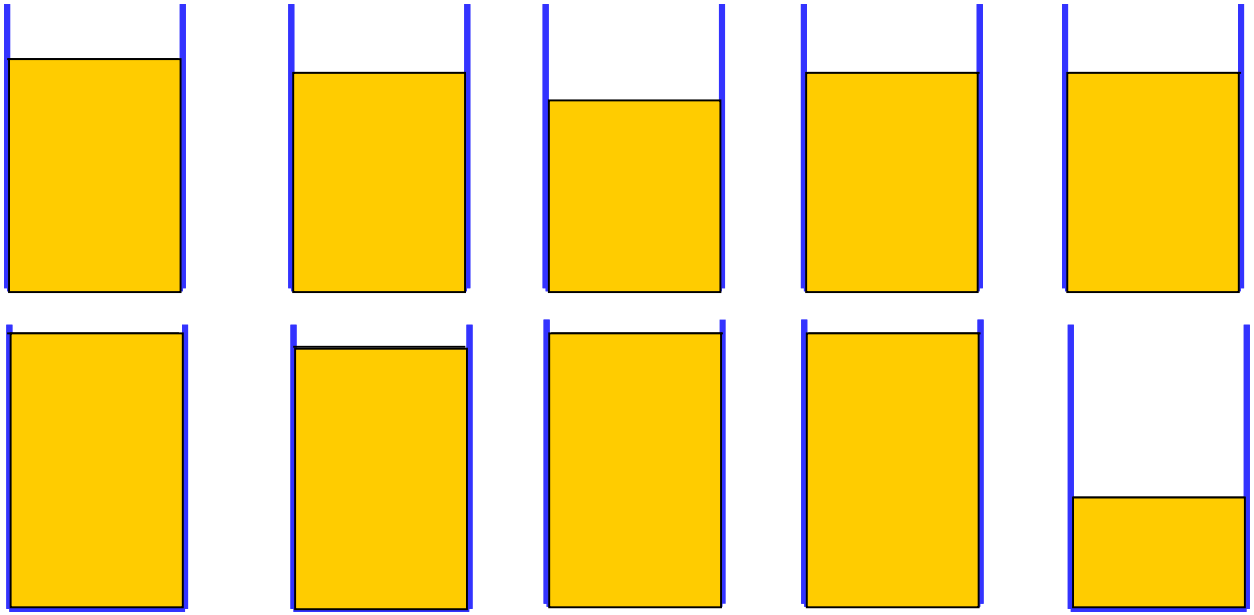


Solution 1

Solution 2

- Consider a BPP with two solutions as presented above, constructed with different selection criteria
- Note that with respect to the most common BPP objective function, both solutions have the same value of 5 as they both use 5 bins.

BPP – Alternative objective function



- However, one can claim that solution 2 is better due to the better utilization of the bins space
- Therefore, an alternative objective function must be formed to capture the superiority of solution 2

BPP – Alternative objective function

- Alternative objective function
- Assume a solution s as follows:
  - Uses p bins
  - Let u_i be the used space consumed by items assigned to bin $i = 1, \dots, p$
  - Bins $i = 1, \dots, p$, contain q_i objects
  - Let t_{ij} be the size of the object j stored in bin i, $(i = 1, \dots, p, and\ j = 1, \dots, q_i)$
- Objective function

$$z(s) = \sum_{i=1}^{p} {u_i}^2 = \sum_{i=1}^{p} \left(\left(\sum_{j=1}^{q_i} t_{ij} \right)^2 \right)$$

Increases as the used space of the bins is increased

BPP – Alternative objective function

- Therefore, the maximation of the new objective function
  - Is in favor of solutions that use the least possible number of bins
  - Is in favor of the used space increase of the used bins

- Which of the First-Fit and Best-Fit greedy algorithms better match the new objective function?

$$z(s) = \sum_{i=1}^{p} \left(\left(\sum_{j=1}^{q_i} t_{ij} \right)^2 \right)$$

BPP – Alternative objective function

- The evaluation of the solution constructed via a First Fit greedy algorithm, according to the new objective function is:

s :	b1:	W_8	W_7	
	b2:	W_3	W_5	
	b3:	W_4	W_6	
	b4:	W_1	W_{11}	
	b5:	W_2	W_{10}	W_9

Objective function value:

$$z(s) = 17^2 + 20^2 + 18^2 + 16^2 + 20^2 = 1669$$

BPP – Alternative objective function

- The evaluation of the solution constructed via a First Fit greedy algorithm, according to the new objective function is:

s :

b1:	W_8		
b2:	W_3	W_5	
b3:	W_4	W_6	W_7
b4:	W_1	W_{11}	
b5:	W_2	W_{10}	W_9

Objective function value:

$$z(s) = 15^2 + 20^2 + 20^2 + 16^2 + 20^2 = 1681$$

Problem

- Distribution of goods has a crucial impact on the effectiveness of the overall supply chain
- Milk producing company with a private fleet of 8 identical trucks
- The company can use up to 8 trucks based at the production site (Node 0) to serve 9 customer orders
- Customer nodes: {1,2,..,9}
- The capacity of each truck is 17,000 lt

Problem

- The trucks start off their trips from the depot, then visit customers, and finally return to the depot
- A truck may serve (visit) multiple customers
- Each customer is served by exactly one visit of exactly one customer
- Obviously, the total load carried by a single truck cannot exceed the vehicle capacity of 17,000 lt.

Problem

- The distances between node pairs are given in the following table:

	0	1	2	3	4	5	6	7	8	9
0	0	21	56	89	56	25	64	41	48	55
1	21	0	29	34	52	36	12	55	35	67
2	56	29	0	32	72	56	64	81	89	26
3	89	34	32	0	36	64	81	95	99	18
4	56	52	72	36	0	29	55	47	65	39
5	25	36	56	64	29	0	40	56	62	46
6	64	12	64	81	55	40	0	29	31	43
7	41	55	81	95	47	56	29	0	11	46
8	48	35	89	99	65	62	31	11	0	36
9	55	67	26	18	39	46	43	46	36	0

- The demand of the customers is given below:

Customer	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Demand	2,000	4,000	5,000	1,000	7,000	3,000	5,000	4,000	4,000

Problem

- We have two goals
  - Primary Goal: Minimization of the number of trucks dispatched to the customers
  - Secondary Goal: Minimization of the total kilometers travelled by the trucks to serve the customers
- Provide a suitable objective for the problem
- Design and apply a greedy algorithm for the problem

Management Science & Assignment Problems

- A maritime company manages five tanker vessels
- Depending on their characteristics,
  - Scrubbers
  - Fuel Efficiency
  - Age
  - Draft
  - Ice-Breaking capabilities

  Vessels have different running costs when operating in different areas
- To ensure a global presence the maritime company has to ensure that its vessels are spread across different geographic areas
- The commercial department of the company has constructed a table with the expected daily costs of each vessel for each of the five geographic areas

Assignment problem: The linear sum assignment

Vessel

Area

	A	B	C	D	E
Med	38	53	61	36	66
Continent	100	60	9	79	34
Baltic	30	37	36	72	24
Arabian Gulf	61	95	21	14	64
W. Africa	89	90	4	5	79

Assignment problem: The linear sum assignment

- The aim is to determine one area for each of our vessels

Constraint

- Each area can be assigned to exactly one vessel

Objective Function

- The total daily costs for all vessels must be minimized

Assignment problem: The linear sum assignment

- The problem is an assignment problem
- Set 1: Areas
- Set 2: Ships
- Each element of Areas must be assigned to exactly one element of Ships
- Objective: Minimization of the total costs incurred by the five (Ship-Area) assignments

Assignment problem: The linear sum assignment

- Solution Structure:

$$s = \{(Med, ship1), (Cont, ship2), (Baltic, ship3), (AG, ship4), (USG, ship5)\}$$

- How many feasible solutions?
  - Let's lock the areas in the solution representation

$$s = \{(\boldsymbol{Med}, ***), (\boldsymbol{Cont}, ***), (\boldsymbol{Baltic}, ***), (\boldsymbol{AG}, ***), (\boldsymbol{WAfr}, ***)\}$$

  - By locking the order of Areas, then each permutation of the ships represent a distinct solution to the problem
  - In total, $5!$ Feasible solutions

Assignment problem: The linear sum assignment

- Objective Function:

Solution

$$s = \{(Med, A), (Cont, C), (Baltic, B), (AG, D), (WAfr, E)\}$$

Objective function value:

$$z(S) = 38 + 9 + 37 + 14 + 79$$

	A	B	C	D	E
Med	**38**	53	61	36	66
Cont	100	60	**9**	79	34
Baltic	30	**37**	36	72	24
AG	61	95	21	**14**	64
WAfr	89	90	4	5	**79**

Assignment problem: The linear sum assignment

- Suggest a Greedy Algorithm for the problem

Linear sum assignment - A Greedy Algorithm

- A Greedy Algorithm for the problem could work as follows:

  Let R_{comp} is the set of areas and V_{comp} is the set of vessels

  $s = \{\}$

  $V = \{\}$

  $R = \{\}$

  while $|s| < 5$

  Identify the assignment $(i, j), i \in R_{comp}, j \in V_{comp}, i \notin R, j \notin V$ with the minimum cost

  $s \longleftarrow s \cup \{(i, j)\}$

  $R \longleftarrow R \cup \{i\}$

  $V \longleftarrow V \cup \{j\}$

Linear sum assignment - A Greedy Algorithm

- Iteration 1

$s = \{\}$
$V = \{\}$
$R = \{\}$

$(i, j) = (WAfr, C)$

$s = \{(WAfr, C)\}$
$V = \{C\}$
$R = \{WAfr\}$

	A	B	C	D	E
Med	38	53	61	36	66
Cont	100	60	9	79	34
Baltic	30	37	36	72	24
AG	61	95	21	14	64
WAfr	89	90	**4**	5	79

Linear sum assignment - A Greedy Algorithm

- Iteration 2

$$s = \{(WAfr, C)\}$$
$$V = \{C\}$$
$$R = \{WAfr\}$$

$$(i, j) = (AG, D)$$

$$s = \{(WAfr, C), (AG, D)\}$$
$$V = \{C, D\}$$
$$R = \{WAfr, AG\}$$

	A	B	C	D	E
Med	38	53	61	36	66
Cont	100	60	9	79	34
Baltic	30	37	36	72	24
AG	61	95	21	**14**	64
WAfr	89	90	4	5	79

Linear sum assignment - A Greedy Algorithm

- Iteration 3

$$s = \{(WAfr, C), (AG, D)\}$$
$$V = \{C, D\}$$
$$R = \{WAfr, AG\}$$

$$(i, j) = (Baltic, E)$$

$$s = \{(WAfr, C), (AG, D), (Baltic, E)\}$$
$$V = \{C, D, E\}$$
$$R = \{WAfr, AG, Baltic\}$$

	A	B	C	D	E
Med	38	53	61	36	66
Cont	100	60	9	79	34
Baltic	30	37	36	72	**24**
~~**AG**~~	~~61~~	~~95~~	~~21~~	~~14~~	~~64~~
~~**WAfr**~~	~~89~~	~~90~~	~~4~~	~~5~~	~~79~~

Linear sum assignment - A Greedy Algorithm

- Iteration 4

$s = \{(WAfr, C), (AG, D), (Baltic, E)\}$
$V = \{C, D, E\}$
$R = \{WAfr, AG, Baltic\}$

$(i, j) = (Med, A)$

$s = \{(WAfr, C), (AG, D), (Baltic, E), (Med, A)\}$
$V = \{C, D, E, A\}$
$R = \{WAfr, AG, Baltic, Med\}$

	A	B	C	D	E
Med	**38**	53	61	36	66
Cont	100	60	9	79	34
Baltic	30	37	36	72	24
AG	61	95	21	14	64
WAfr	89	90	4	5	79

Linear sum assignment - A Greedy Algorithm

- Iteration 5

$s = \{(WAfr, C), (AG, D), (Baltic, E), (Med, A)\}$
$V = \{C, D, E, A\}$
$R = \{WAfr, AG, Baltic, Med\}$

$(i, j) = (Cont, B)$

	A	B	C	D	E
Med	38	53	61	36	66
Cont	100	**60**	9	79	34
Baltic	30	37	36	72	24
AG	61	95	21	14	64
WAfr	89	90	4	5	79

$s = \{(WAfr, C), (AG, D), (Baltic, E), (Med, A), (Cont, B)\}$
$V = \{C, D, E, A, B\}$
$R = \{WAfr, AG, Baltic, Med, Cont\}$

Linear sum assignment - A Greedy Algorithm

- The algorithm terminates with the final solution

$$s = \{(WAfr, C), (AG, D), (Baltic, E), (Med, A), (Cont, B)\}$$

with objective value:

$$z(S) = 4 + 14 + 24 + 36 + 60 = 138$$

	A	B	C	D	E
Med	38	53	61	36	66
Cont	100	60	9	79	34
Baltic	30	37	36	72	24
AG	61	95	21	14	64
Wafr	89	90	4	5	79

Hungarian Algorithm

- Is this the optimal solution?
  - We don't know
  - It is just a good quality solution speedily produced by a greedy method
- For this type of problem, another Greedy method has been proposed (more elaborate than the one we presented)
- The algorithm has been proposed by Kuhn & Munkres
- Based on the research of Dénes Kőnig and Jenő Egerváry (Both Hungarian -> Hungarian Algorithm)
- Solves minimization problems in defined square cost matrices
- Note: We will se how maximization problems in rectangular matrices can be solved via the Hungarian algorithm

Hungarian Algorithm

- Algorithm Steps

  Step 1. Find the minimal value of each row and subtract it from all entries of the row

  Step 2. Find the minimal value of each column and subtract it from all entries of the column

  Step 3. Cover all zero values with the minimal number of straight lines (vertical: lines covering columns, horizontal: lines covering rows)

  If the required lines are less than n, Go to Step 4

  Step 4. Identify the minimal uncover entry (θ).

  Subtract θ from all uncovered elements

  Add θ to any entry covered by two lines.

  Go to Step 3

Hungarian Algorithm

Step 1. Find the minimal value of each row and subtract it from all entries of the row

	A	B	C	D	E
Med	38	53	61	**36**	66
Cont	100	60	**9**	79	34
Baltic	30	37	36	72	**24**
AG	61	95	21	**14**	64
WAfr	89	90	**4**	5	79

	A	B	C	D	E
Med	2	17	25	0	30
Cont	91	51	0	70	25
Baltic	6	13	12	48	0
AG	47	81	7	0	50
WAfr	85	86	0	1	75

Hungarian Algorithm

Step 2. Find the minimal value of each column and subtract it from all entries of the column

	A	B	C	D	E
Med	**2**	17	25	**0**	30
Cont	91	51	0	70	25
Baltic	6	**13**	12	48	**0**
AG	47	81	**7**	**0**	50
WAfr	85	86	0	1	75

	A	B	C	D	E
Med	0	4	25	0	30
Cont	89	38	0	70	25
Baltic	4	0	12	48	0
AG	45	68	7	0	50
WAfr	83	73	0	1	75

Hungarian Algorithm

Step 3. Cover all zero values with the minimal number of straight lines (vertical: lines covering columns, horizontal: lines covering rows)

If the required lines are less than n, Go to Step 4

	A	B	C	D	E
~~Med~~	~~0~~	~~4~~	~~25~~	~~0~~	~~30~~
Cont	89	38	0	70	25
~~Baltic~~	~~4~~	~~0~~	~~12~~	~~48~~	~~0~~
~~AG~~	~~45~~	~~68~~	~~7~~	~~0~~	~~50~~
WAfr	83	73	0	1	75

Notice that required lines are $4 < 5$, therefore we go to Step 4

Hungarian Algorithm

Step 4. Identify the minimal uncover entry (θ).

Subtract θ from all uncovered elements

Add θ to any entry covered by two lines.

Go to Step 3

$$\theta = 1$$

	A	B	C	D	E
~~Med~~	~~0~~	~~4~~	~~25~~	~~0~~	~~30~~
Cont	89	38	0	70	25
~~Baltic~~	~~4~~	~~0~~	~~12~~	~~48~~	~~0~~
~~AG~~	~~45~~	~~68~~	~~7~~	~~0~~	~~50~~
WAfr	83	73	0	***1***	75

	A	B	C	D	E
Med	0	4	**26**	0	30
Cont	**88**	**37**	0	**69**	**24**
Baltic	4	0	**13**	48	0
AG	45	68	**8**	0	50
WAfr	**82**	**72**	0	**0**	**74**

Hungarian Algorithm

Step 3. Cover all zero values with the minimal number of straight lines (vertical: lines covering columns, horizontal: lines covering rows)

If the required lines are less than n, Go to Step 4

	A	B	C	D	E
~~Med~~	~~0~~	~~4~~	~~26~~	~~0~~	~~30~~
Cont	88	37	0	69	24
~~Baltic~~	~~4~~	~~0~~	~~13~~	~~48~~	~~0~~
AG	45	68	8	0	50
WAfr	82	72	0	0	74

Notice that required lines are $4 < 5$, therefore we go to Step 4

Hungarian Algorithm

Step 4. Identify the minimal uncover entry (θ).

Subtract θ from all uncovered elements

Add θ to any entry covered by two lines.

Go to Step 3

$$\theta = 24$$

	A	B	C	D	E
~~Med~~	~~0~~	~~4~~	~~26~~	~~0~~	~~30~~
Cont	88	37	0	69	***24***
~~Baltic~~	~~4~~	~~0~~	~~13~~	~~48~~	~~0~~
AG	45	68	8	0	50
WAfr	82	72	0	0	74

	A	B	C	D	E
Med	0	4	**50**	**24**	30
Cont	**64**	**13**	0	**69**	**0**
Baltic	4	0	**37**	**72**	0
AG	**21**	**44**	8	0	**26**
WAfr	**58**	**48**	0	0	**50**

Hungarian Algorithm

Step 3. Cover all zero values with the minimal number of straight lines (vertical: lines covering columns, horizontal: lines covering rows)

If the required lines are less than n, Go to Step 4

	A	B	C	D	E
~~Med~~	0	4	50	24	30
~~Cont~~	~~64~~	~~13~~	~~0~~	~~69~~	~~0~~
~~Baltic~~	~~4~~	~~0~~	~~37~~	~~72~~	~~0~~
~~AG~~	~~21~~	~~44~~	~~8~~	~~0~~	~~26~~
~~WAfr~~	~~58~~	~~48~~	~~0~~	~~0~~	~~50~~

Observe that $5 = n$, thus the algorithm is terminated

Hungarian Algorithm

How is the final Assignment determined;

From the final matrix, we select all zero values, such that each row and each column has exactly one selected zero

	A	B	C	D	E
Med	**0**	4	50	24	30
Cont	64	13	0	69	**0**
Baltic	4	**0**	37	72	0
AG	21	44	8	**0**	26
WAfr	58	48	**0**	0	50

The final solution (assignment set) is determined by the selected zero values:

$$s = \{(WAfr, C), (AG, D), (Baltic, B), (Med, A), (Cont, E)\}$$

Hungarian Algorithm

The final Hungarian Algorithm solution is **:**

$$s = \{(WAfr, C), (AG, D), (Baltic, B), (Med, A), (Cont, E)\}$$

Objective function:

$$z(s) = 4 + 14 + 37 + 38 + 34 = 127$$

	A	B	C	D	E
Med	38	53	61	36	66
Cont	100	60	9	79	34
Baltic	30	37	36	72	24
AG	61	95	21	14	64
WAfr	89	90	4	5	79

Hungarian Algorithm

Comment 1:

- Step 3 suggests “Cover all zero values with the minimal number of straight lines”
- This is not always a straightforward procedure
- Solution to a set covering problem (we will see it later)
  - Universe: All zero values in the table
  - Candidate Elements: Candidate Lines
  - Each line is assigned to a subset of the universe (zeroes to be covered by the line)
  - Minimum number of the candidate elements
- A greedy method is to iteratively select the lines that cover the maximum number of uncovered zeroes
- However, when ties are observed, the random selection of lines may lead to suboptimal selection of lines
- For the small-scale problems faced in the class, the selection will be straightforward

Hungarian Algorithm

Comment 2:

- When the algorithm terminates, selection of zeroes (assignments) may not be obvious
- For the small-scale demo problems, this selection will be easy to make by a trail and error procedure

- Overall, a simple implementation of the Hungarian Problem is not always sufficient for large-scale problems
- We will use a Python package for a more effective implementation

Hungarian Algorithm

- Consider the case when the matrix gives the vessel profitability
- This is a Maximization Problem
- How should we work to identify the assignment providing the maximum profit?
  - Could the Hungarian algorithm (solving minimization problems) be applied?

	A	B	C	D	E
Med	38	53	61	36	66
Cont	100	60	9	79	34
Baltic	30	37	36	72	24
AG	61	95	21	14	64
WAfr	89	90	4	5	79

Hungarian Algorithm

- Hungarian Algorithm tackles Minimization Problems
- Thus the first step is to convert profits to losses ($\times\ -1$)

	A	B	C	D	E
Med	38	53	61	36	66
Cont	100	60	9	79	34
Baltic	30	37	36	72	24
AG	61	95	21	14	64
WAfr	89	90	4	5	79

	A	B	C	D	E
Med	-38	-53	-61	-36	-66
Cont	-100	-60	-9	-79	-34
Baltic	-30	-37	-36	-72	-24
AG	-61	-95	-21	-14	-64
WAfr	-89	-90	-4	-5	-79

Hungarian Algorithm

- Hungarian algorithm is applied on non-negative matrices
- Identify the minimum value (which is going to be negative)
- Add the absolute of this minimum value to all table entries
- Proceed, by applying the Hungarian Algorithm

	A	B	C	D	E
Med	-38	-53	-61	-36	-66
Cont	-100	-60	-9	-79	-34
Baltic	-30	-37	-36	-72	-24
AG	-61	-95	-21	-14	-64
WAfr	-89	-90	-4	5	-79

	A	B	C	D	E
Med	62	47	39	64	34
Cont	0	40	91	21	66
Baltic	70	63	64	28	76
AG	39	5	79	86	36
WAfr	11	10	96	95	21

Hungarian Algorithm

- Observe the Assignment maximizing the profit in the original table $(Cont, A)$, with profit 100
- The assignment $(Cont, A)$ has the minimal cost in the cost matrix
- It is the most "desirable" assignments for both tables

Profit matrix

	A	B	C	D	E
Med	38	53	61	36	66
Cont	100	60	9	79	34
Baltic	30	37	36	72	24
AG	61	95	21	14	64
WAfr	89	90	4	5	79

Cost matrix

	A	B	C	D	E
Med	62	47	39	64	34
Cont	0	40	91	21	66
Baltic	70	63	64	28	76
AG	39	5	79	86	36
WAfr	11	10	96	95	21

Hungarian Algorithm – Non-square matrices

- Consider the case of a rectangular table (not a square table)
- For instance, more geographic areas than ships

Cost Matrix

	A	B	C	D	E
Med	38	53	61	36	66
Cont	100	60	9	79	34
Baltic	30	37	36	72	24
AG	61	95	21	14	64
WAfr	89	90	4	5	79
USG	72	64	12	25	41
Spore	45	47	52	36	70

Hungarian Algorithm – Non-square matrices

- In this example, each ship must be assigned to exactly one area
- One area must be assigned to up to one ship
- Obviously, two areas will be left without any ship

Cost Matrix

	A	B	C	D	E
Med	38	53	61	36	66
Cont	100	60	9	79	34
Baltic	30	37	36	72	24
AG	61	95	21	14	64
WAfr	89	90	4	5	79
USG	72	64	12	25	41
Spore	45	47	52	36	70

Hungarian Algorithm – Non-square matrices

- How would this problem be solved with thew Hungarian Algorithm
- Creation of new columns representing mock ships
- All values of this new column must be equal
  - Preferably 0 for the minimization problems
  - Preferably the maximum table value for the maximization problems
  - This will lead to an algorithm ready table with zeroes, thus the algorithm will run faster

Cost Matrix

	A	B	C	D	E	V1	V2
Med	38	53	61	36	66	0	0
Cont	100	60	9	79	34	0	0
Baltic	30	37	36	72	24	0	0
AG	61	95	21	14	64	0	0
WAfr	89	90	4	5	79	0	0
USG	72	64	12	25	41	0	0
Spore	45	47	52	36	70	0	0

Hungarian Algorithm – Non-square matrices

- By adding two columns, every assignment of a mock ship to an area contributes to the total cost equally
- Thus every mock ship assignment does not differentiate the optimal assignment of the rest of the ships to the areas available
- Thus, the optimal solution of the modified table will "include" the optimal assignment of the original table to the various areas

Cost Matrix

	A	B	C	D	E	V1	V2
Med	38	53	61	36	66	0	0
Cont	100	60	9	79	34	0	0
Baltic	30	37	36	72	24	0	0
AG	61	95	21	14	64	0	0
WAfr	89	90	4	5	79	0	0
USG	72	64	12	25	41	0	0
Spore	45	47	52	36	70	0	0

Hungarian Algorithm – Non-square matrices

- Consider the solution
  $s = \{(USG, C), (AG, D), (Baltic, B), (Med, A), (Cont, E), \sout{(Spore, V1), (WAfr, V2)}\}$
- Ignore the assignments of the mock ships
- The final solution to the original problem is
  $$s = \{(USG, C), (AG, D), (Baltic, B), (Med, A), (Cont, E)\}$$
- Areas $WAfr$ and $Cont$ are not covered by any of the five ships

Cost Matrix

	A	B	C	D	E	V1	V2
Med	38	53	61	36	66	0	0
Cont	100	60	9	79	34	0	0
Baltic	30	37	36	72	24	0	0
AG	61	95	21	14	64	0	0
WAfr	89	90	4	5	79	0	0
USG	72	64	12	25	41	0	0
Spore	45	47	52	36	70	0	0

Hungarian Algorithm - Assignment 1-M

- Another variant of the problem is when the 1-1 is generalized to 1-m
- For example, areas may host more that 1 vessels
  - Med can host up to 2 vessels
  - Baltic can host up to 3 vessels
- How should the table be converted to deal with the modified operational scenario?

Cost Matrix

	A	B	C	D	E
Med	38	53	61	36	66
Cont	100	60	9	79	34
Baltic	30	37	36	72	24
AG	61	95	21	14	64
WAfr	89	90	4	5	79
USG	72	64	12	25	41
Spore	45	47	52	36	70

Hungarian Algorithm - Assignment 1-M

- Addition of new rows (clones) for each line allowing multiple assignments
- The total rows for an area must be equal to the upper bound of the assignments allowed
- Creation
  - 1 extra row for Med: Med2

  (2 Med row in total)
  - 2 extra rows for Baltic: Baltic2, Baltic3

  (3 Baltic rows in total)
- Comment: If ships were in the rows, obviously we would have to create column clones to allow multiple area assignments

	A	B	C	D	E
Med	38	53	61	36	66
Med2	38	53	61	36	66
Cont	100	60	9	79	34
Baltic	30	37	36	72	24
Baltic2	30	37	36	72	24
Baltic3	30	37	36	72	24
AG	61	95	21	14	64
WAfr	89	90	4	5	79
USG	72	64	12	25	41
Spore	45	47	52	36	70

Hungarian Algorithm

- In an aluminum company, there are 5 machines (Extrusion presses): M1-M5
- We have to produce 4 product types: P1-P4
- Based on the machines and types, the following profit table provides the profitability of assigning each product to each machine

	P1	P2	P3	P4
M1	32	10	30	20
M2	45	34	50	15
M3	21	52	55	23
M4	30	20	**62**	40
M5	60	30	44	55

Hungarian Algorithm

- Preprocessing Step 1: Profit Matrix → Cost Matrix

	P1	P2	P3	P4
M1	32	10	30	20
M2	45	34	50	15
M3	21	52	55	23
M4	30	20	**62**	40
M5	60	30	44	55

	P1	P2	P3	P4
M1	30	52	32	42
M2	17	28	12	47
M3	41	10	7	39
M4	32	42	0	22
M5	2	32	18	7

Hungarian Algorithm

- Preprocessing Step 2: Rectangular Matrix → Square Matrix

	P1	P2	P3	P4
M1	30	52	32	42
M2	17	28	12	47
M3	41	10	7	39
M4	32	42	0	22
M5	2	32	18	7

	P1	P2	P3	P4	P5
M1	30	52	32	42	0
M2	17	28	12	47	0
M3	41	10	7	39	0
M4	32	42	0	22	0
M5	2	32	18	7	0

Hungarian Algorithm

- Proceed with applying the Hungarian Algorithm to the modified Table

	P1	P2	P3	P4	P5
M1	30	52	32	42	0
M2	17	28	12	47	0
M3	41	10	7	39	0
M4	32	42	0	22	0
M5	2	32	18	7	0

Hungarian Algorithm

Step 1. Find the minimal value of each row and subtract it from all entries of the row (no change, all rows contain a zero entry)

	P1	P2	P3	P4	P5
M1	30	52	32	42	0
M2	17	28	12	47	0
M3	41	10	7	39	0
M4	32	42	0	22	0
M5	2	32	18	7	0

Hungarian Algorithm

Step 2. Find the minimal value of each column and subtract it from all entries of the column

	P1	P2	P3	P4	P5
M1	30	52	32	42	0
M2	17	28	12	47	0
M3	41	**10**	**7**	39	0
M4	32	42	0	22	0
M5	**2**	32	18	**7**	0

	P1	P2	P3	P4	P5
M1	28	42	32	35	0
M2	15	18	12	40	0
M3	39	0	7	32	0
M4	30	32	0	15	0
M5	0	22	18	0	0

Hungarian Algorithm

Step 3. Cover all zero values with the minimal number of straight lines (vertical: lines covering columns, horizontal: lines covering rows)

If the required lines are less than n, Go to Step 4

	P1	P2	P3	P4	P5
M1	28	42	32	35	0
M2	15	18	12	40	0
M3	39	0	7	32	0
M4	30	32	0	15	0
M5	0	22	18	0	0

Hungarian Algorithm

Step 4. Identify the minimal uncover entry (θ). Subtract θ from all uncovered elements

Add θ to any entry covered by two lines.

Go to Step 3

$$(\theta = 12)$$

	P1	P2	P3	P4	P5
M1	28	42	32	35	0
M2	15	18	12	40	0
M3	39	0	7	32	0
M4	30	32	0	15	0
M5	0	22	18	0	0

	P1	P2	P3	P4	P5
M1	16	30	20	23	0
M2	3	6	0	28	0
M3	39	0	7	32	12
M4	30	32	0	15	12
M5	0	22	18	0	12

Hungarian Algorithm

Step 3. Cover all zero values with the minimal number of straight lines (vertical: lines covering columns, horizontal: lines covering rows)

If the required lines are less than n, Go to Step 4

$$lines = 4$$

	P1	P2	P3	P4	P5
M1	16	30	20	23	0
M2	3	6	0	28	0
M3	39	0	7	32	12
M4	30	32	0	15	12
M5	0	22	18	0	12

Hungarian Algorithm

Step 4. Identify the minimal uncover entry (θ). Subtract θ from all uncovered elements
Add θ to any entry covered by two lines.
Go to Step 3

$$(\theta = 3)$$

	P1	P2	P3	P4	P5
M1	16	30	20	23	0
M2	3	6	0	28	0
M3	39	0	7	32	12
M4	30	32	0	15	12
M5	0	22	18	0	12

	P1	P2	P3	P4	P5
M1	13	27	20	20	0
M2	0	3	0	25	0
M3	39	0	10	32	15
M4	27	29	0	12	12
M5	0	22	21	0	15

Hungarian Algorithm

Step 3. Cover all zero values with the minimal number of straight lines
(vertical: lines covering columns, horizontal: lines covering rows)
If the required lines are less than n, Go to Step 4

$$lines = 5$$

	P1	P2	P3	P4	P5
M1	~~13~~	~~27~~	~~20~~	~~20~~	~~0~~
M2	~~0~~	~~3~~	~~0~~	~~25~~	~~0~~
M3	~~39~~	~~0~~	~~10~~	~~32~~	~~15~~
M4	~~27~~	~~29~~	~~0~~	~~12~~	~~12~~
M5	~~0~~	~~22~~	~~21~~	~~0~~	~~15~~

Hungarian Algorithm

Termination & Final Optimal Solution Identification

From the Table, we select the zero values, such that every row and every column has exactly one selected zero

The final optimal solution corresponds to the following assignment set:

$$s = \{(M1, P5), (M2, P1), (M3, P2), (M4, P3), (M5, P4)\}$$

Order P5 is a mock order, thus the final solution is (by ignoring the P5 assignment):

$$s = \{(M2, P1), (M3, P2), (M4, P3), (M5, P4)\}$$

	P1	P2	P3	P4	P5
M1	13	27	20	20	0
M2	0	3	0	25	0
M3	39	0	10	32	15
M4	27	29	0	12	12
M5	0	22	21	0	15

Hungarian Algorithm

$$s = \{(M2, P1), (M3, P2), (M4, P3), (M5, P4)\}$$

The total profit of the final assignment is:

$$z(s) = 45 + 52 + 62 + 55 = 214$$

	P1	P2	P3	P4
M1	32	10	30	20
M2	45	34	50	15
M3	21	52	55	23
M4	30	20	62	40
M5	60	30	44	55