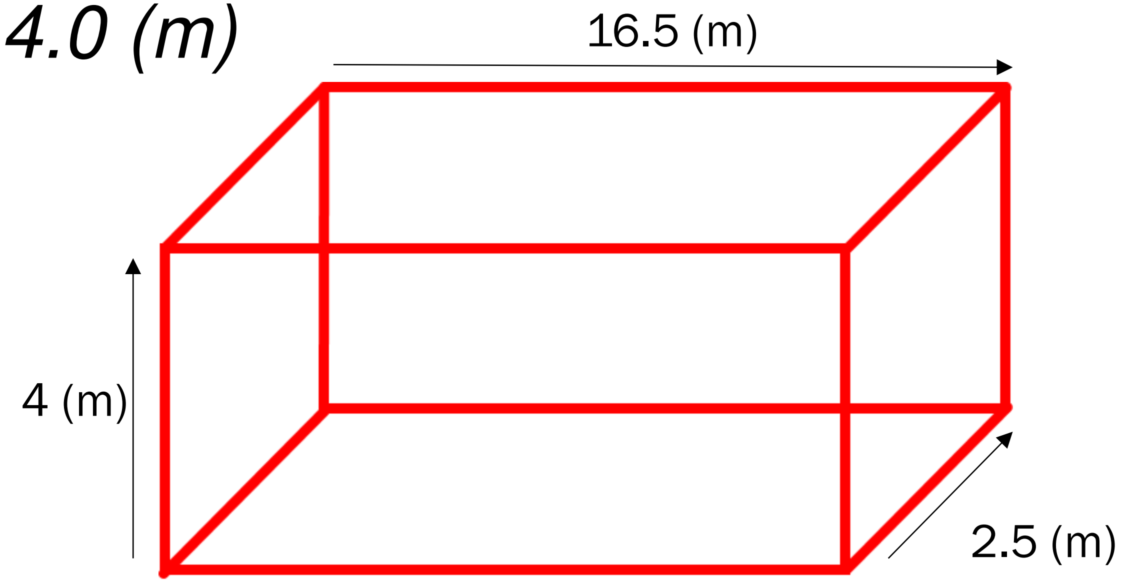
PACKING CARGO SPACES EFFICIENTLY

Group 8
Adam Eljasiak
Daniel Kaestner
Simon Wengeler
Raffaele Piccini
Nicola Gheza
Henri Viigimäe

CONTENTS

ASSIGNMENT DESCRIPTION

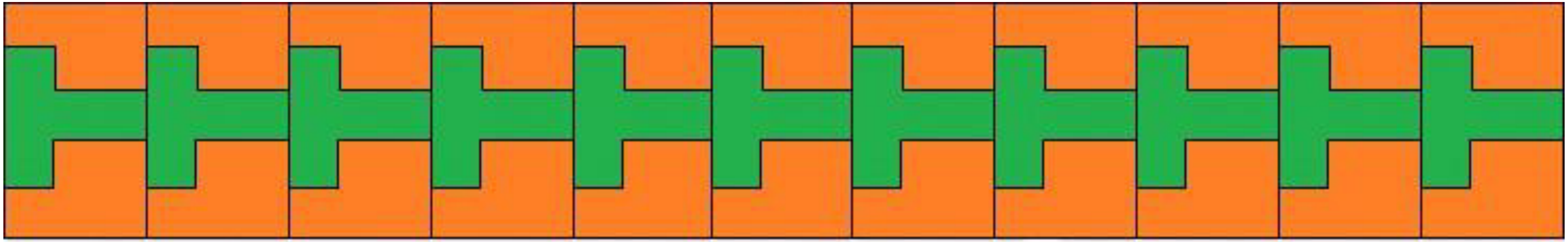
- Application for solving three-dimensional knapsack problems
- Three packages with certain values and a cargo space
  - *A: 1.0 x 1.0 x 2.0 (m) with a value of 3*
  - *B: 1.0 x 1.5 x 2.0 (m) with a value of 4*
  - *C: 1.5 x 1.5 x 1.5 (m) with a value of 5*
  - *Cargo space: 16.5 x 2.5 x 4.0 (m)*



ASSIGNMENT RESULTS

- No solution to filling the entire cargo space with A,B and/or C packages has been found.
- Best result found for maximizing a single packing: **240**
  - *Number of A packages: **30***
  - *Number of B packages :**10***
  - *Number of C packages: **22***

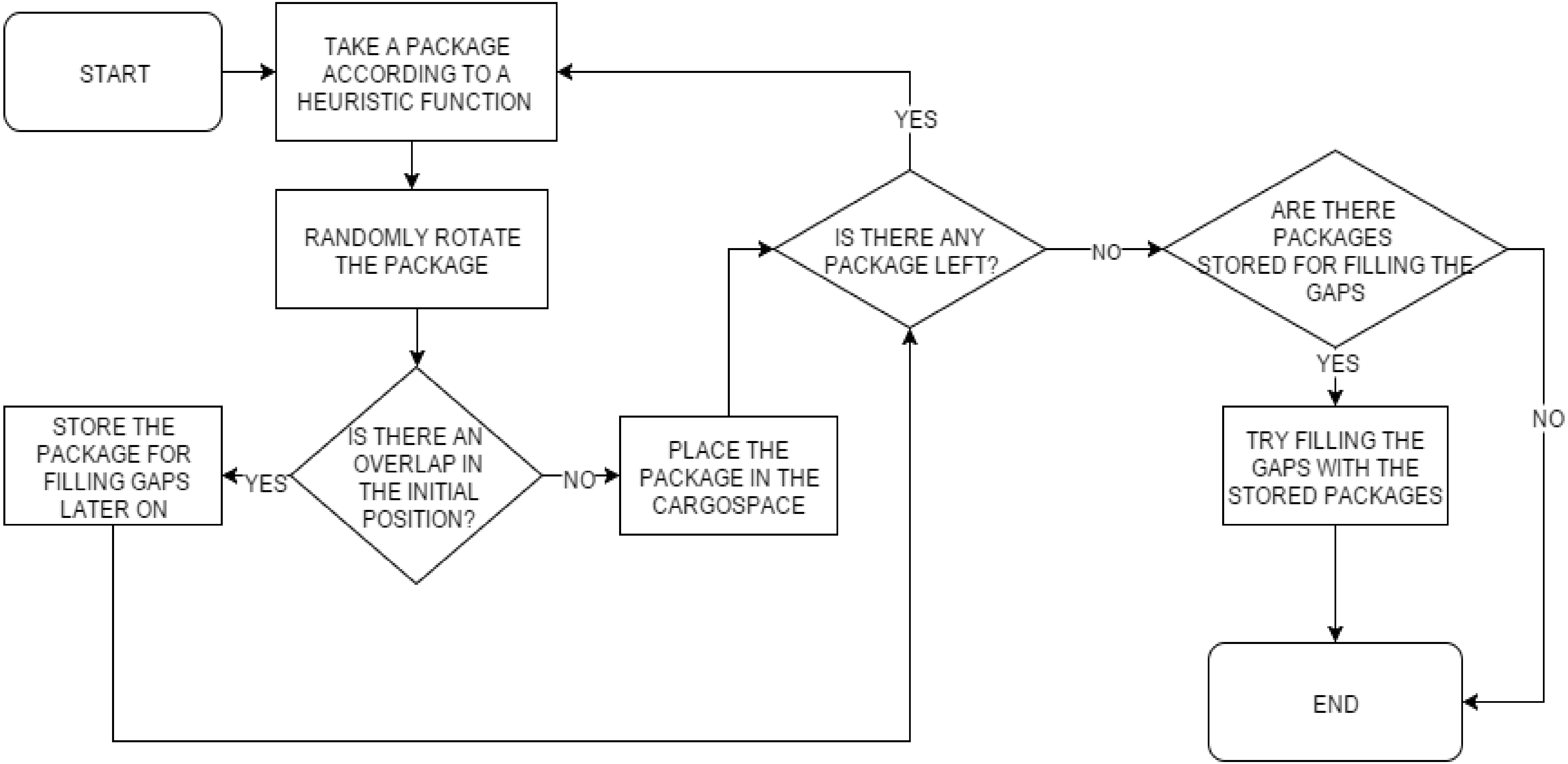
- It is possible to fill the entire cargo space with L,P and/or T packages.
- Best result found for maximizing a single packing: **1144**

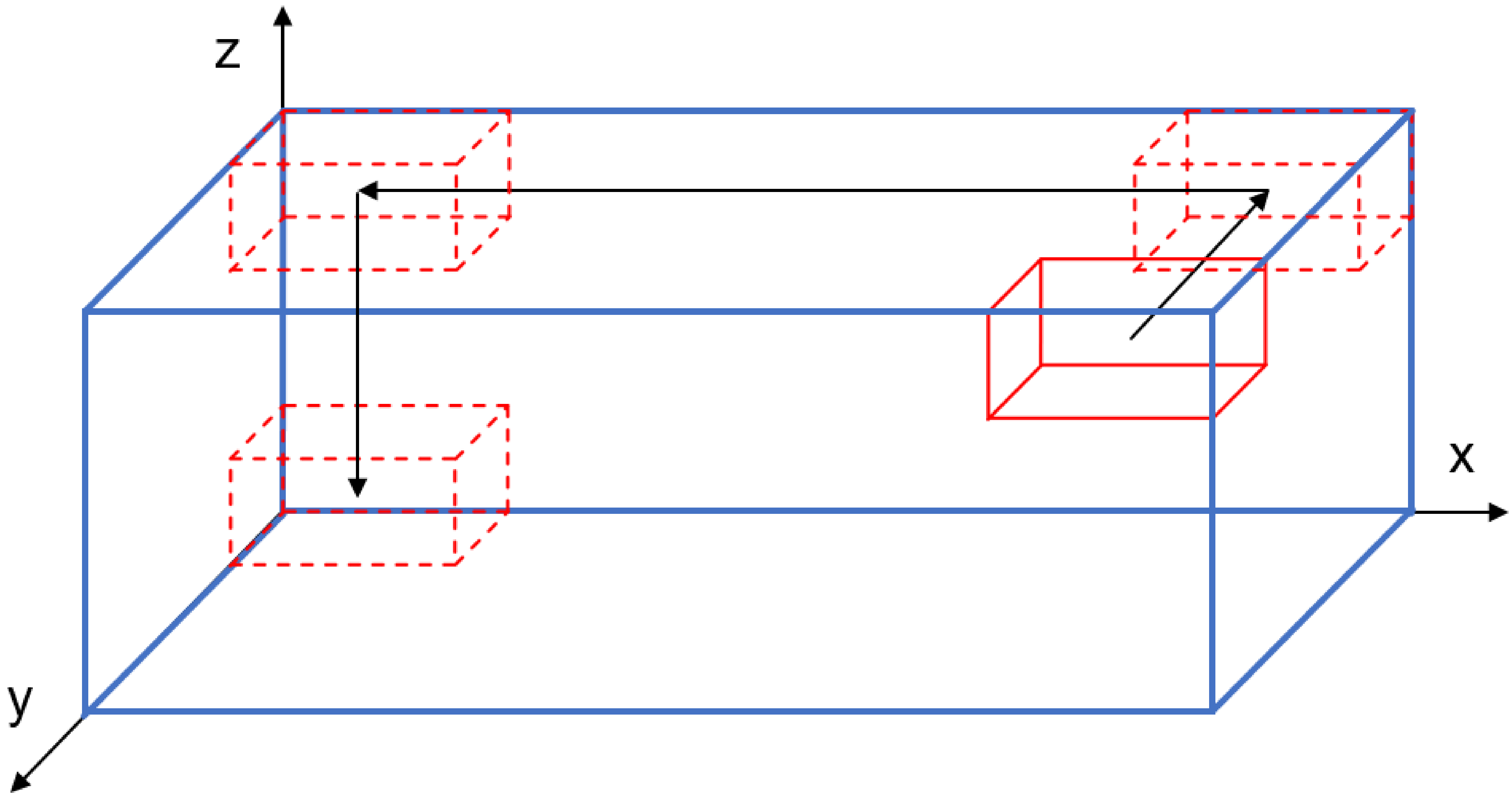


View from above (using P and T packages)

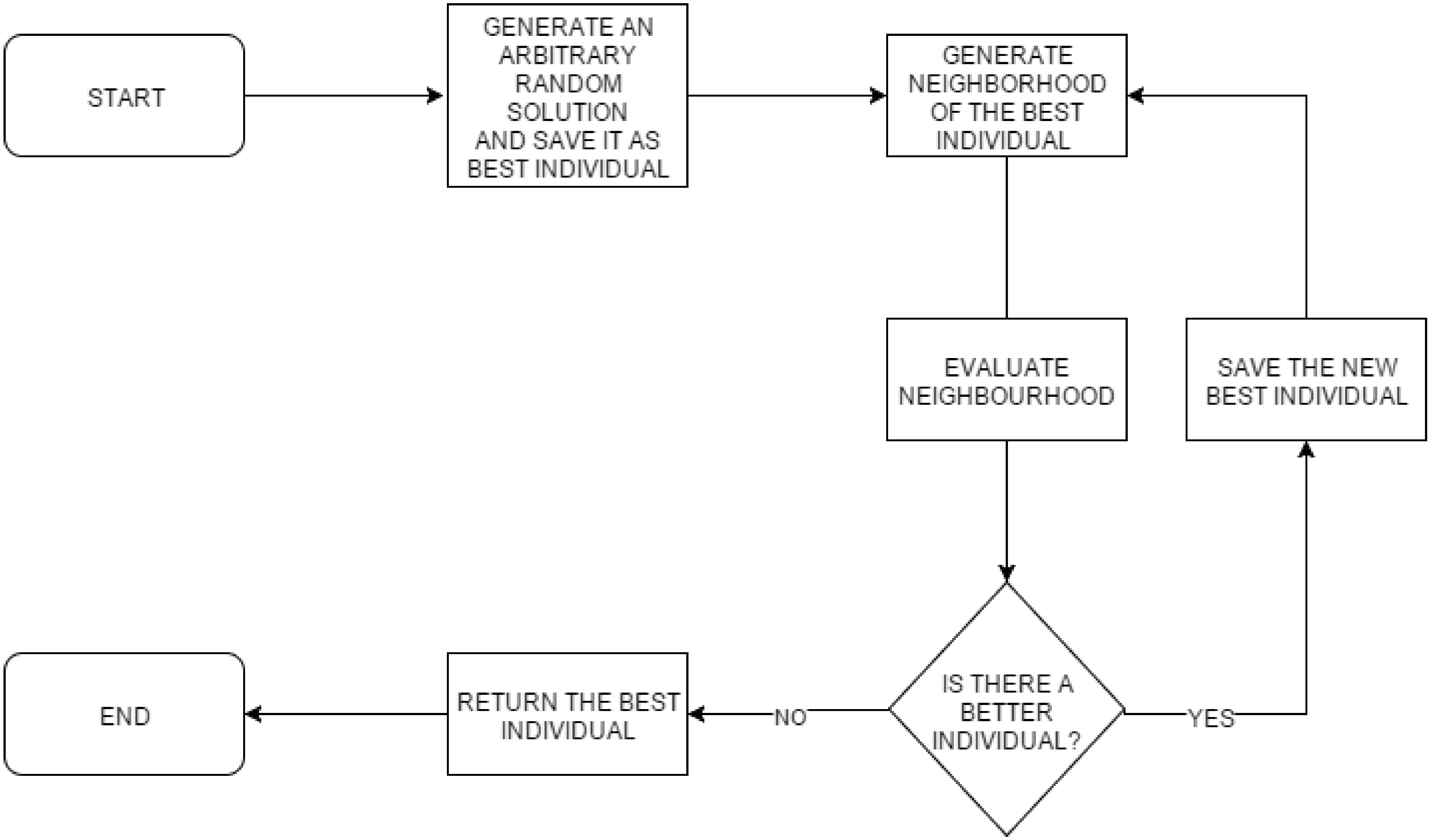
Number of T packages = 11 x 8 (heigth)
Number of P packages = 22 x 8
TOTAL VALUE = 1144

GREEDY ALGORITHM

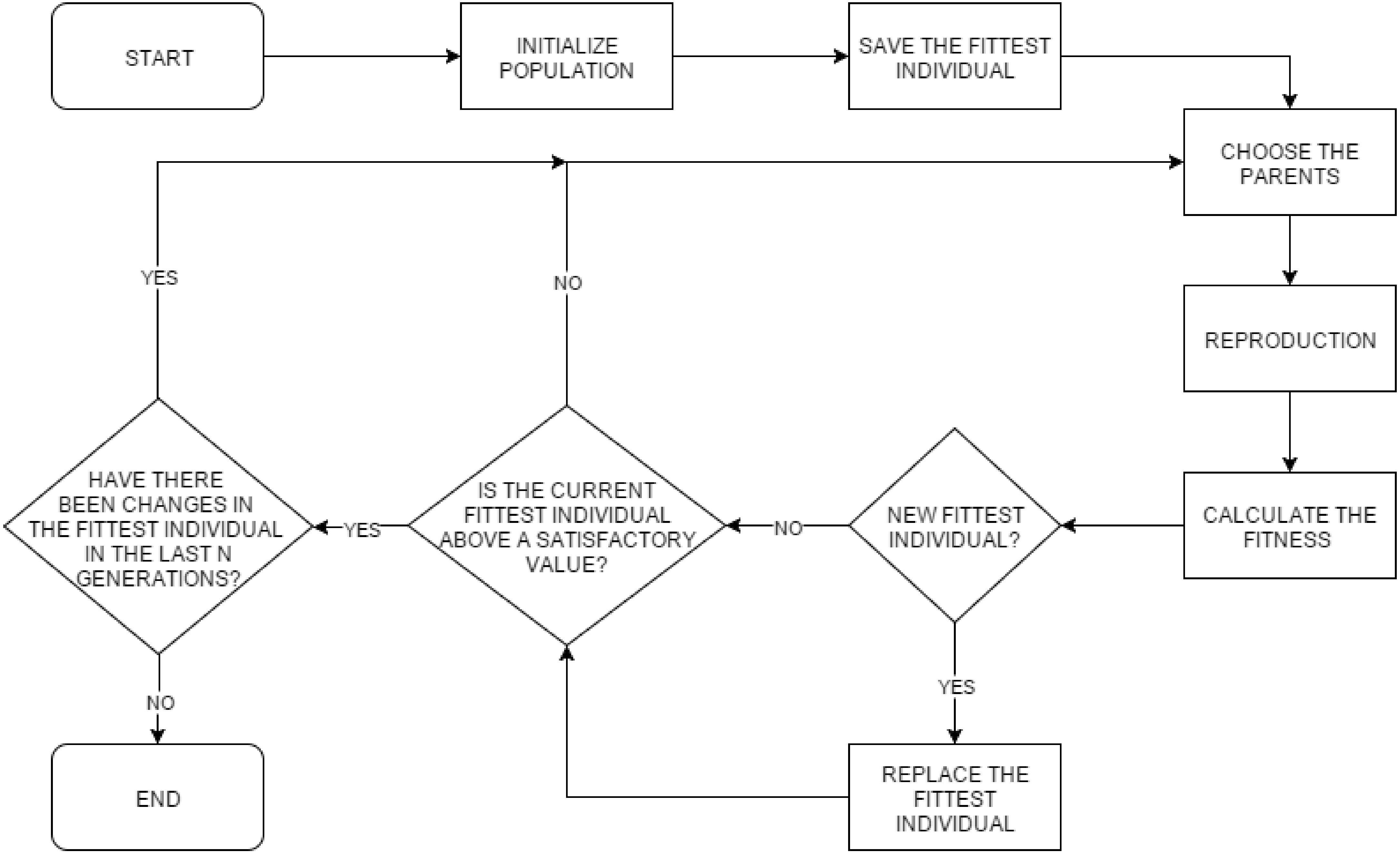

START
TAKE A PACKAGE ACCORDING TO A HEURISTIC FUNCTION
RANDOMLY ROTATE THE PACKAGE
IS THERE AN OVERLAP IN THE INITIAL POSITION?
YES
STORE THE PACKAGE FOR FILLING GAPS LATER ON
NO
PLACE THE PACKAGE IN THE CARGOSPACE
IS THERE ANY PACKAGE LEFT?
YES
NO
ARE THERE PACKAGES STORED FOR FILLING THE GAPS
YES
TRY FILLING THE GAPS WITH THE STORED PACKAGES
NO
END


z
x
y

HILL CLIMBING ALGORITHM

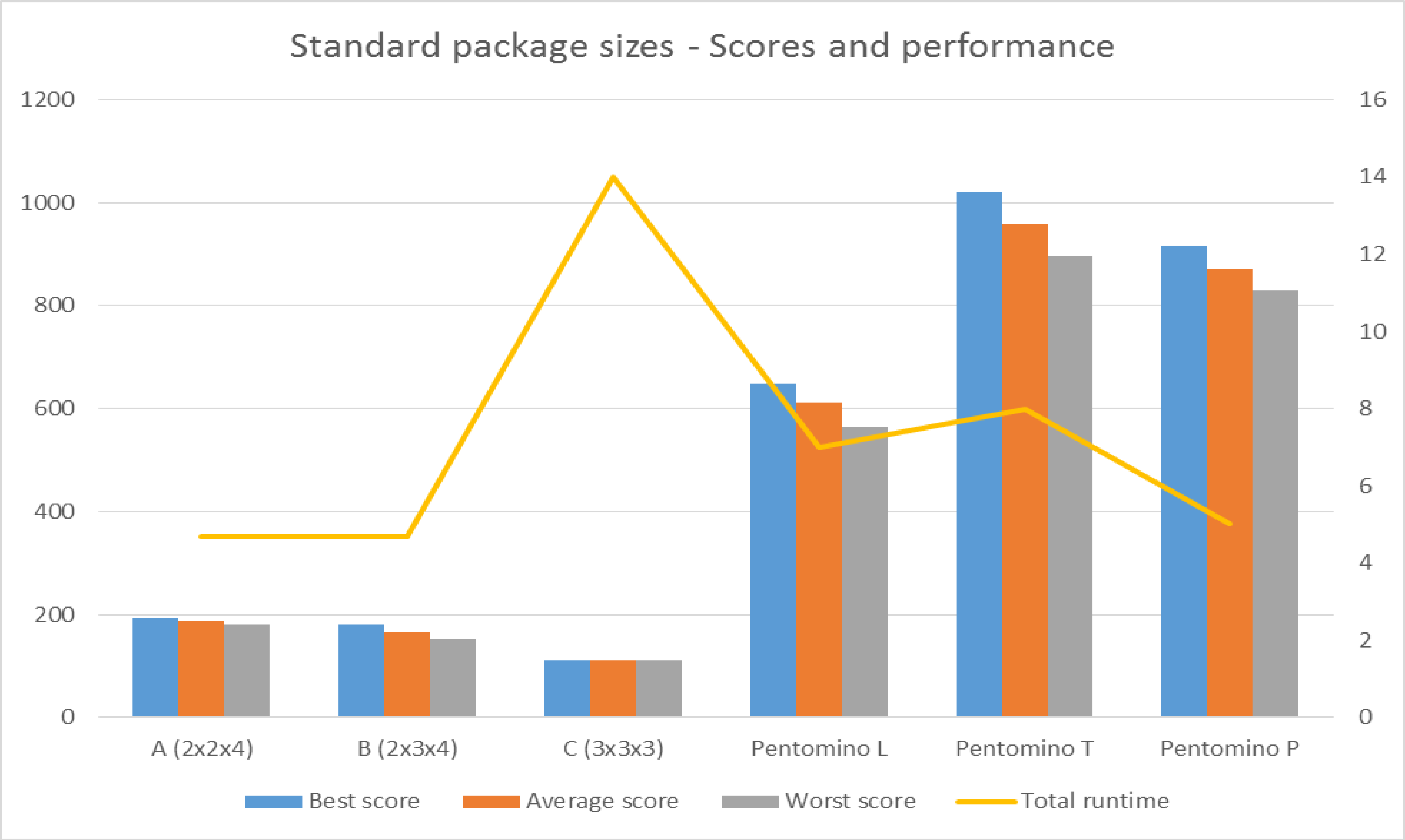

START
GENERATE AN ARBITRARY RANDOM SOLUTION AND SAVE IT AS BEST INDIVIDUAL
GENERATE NEIGHBORHOOD OF THE BEST INDIVIDUAL
EVALUATE NEIGHBOURHOOD
SAVE THE NEW BEST INDIVIDUAL
IS THERE A BETTER INDIVIDUAL?
YES
NO
RETURN THE BEST INDIVIDUAL
END

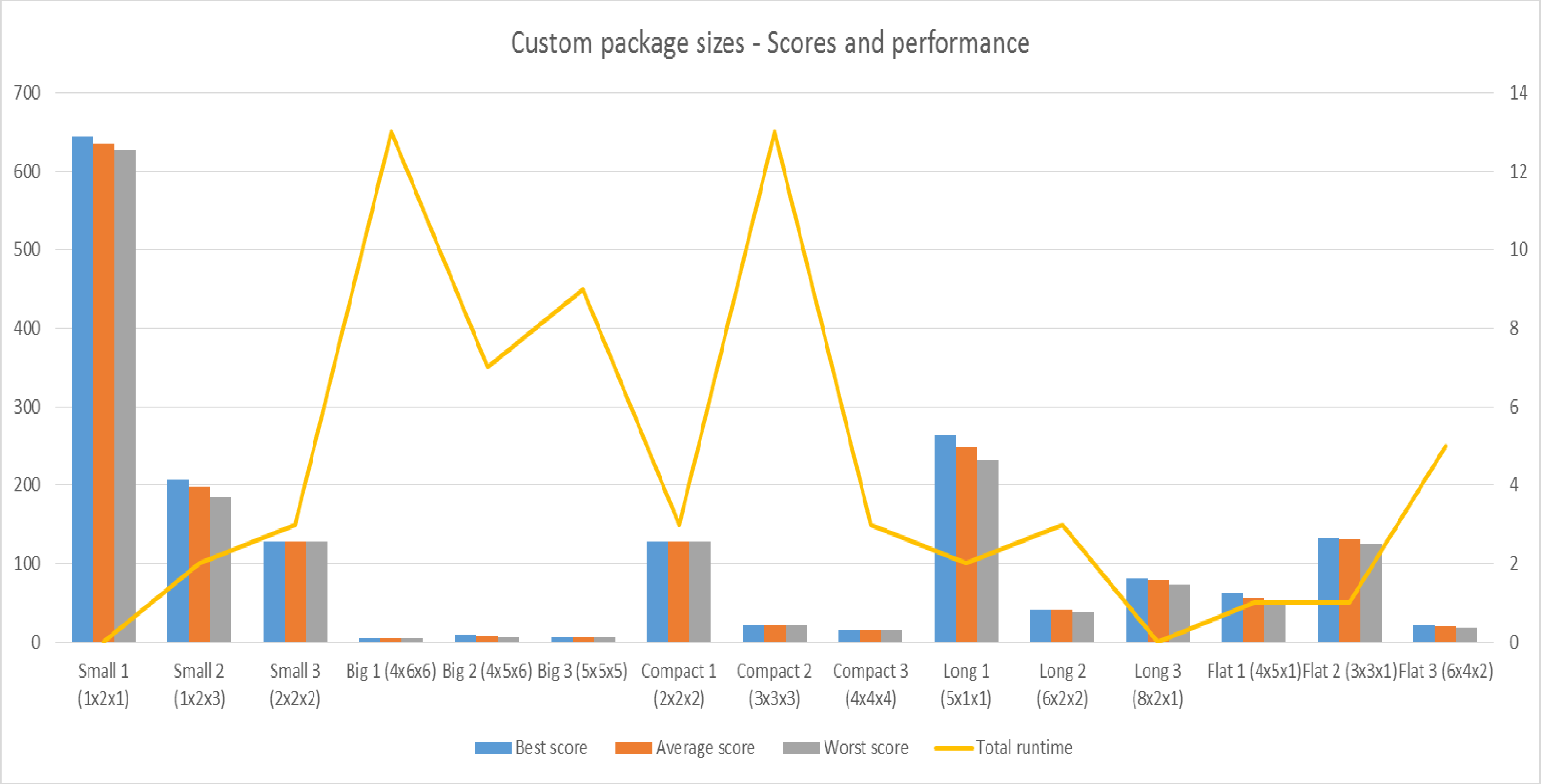
GENETIC ALGORITHM

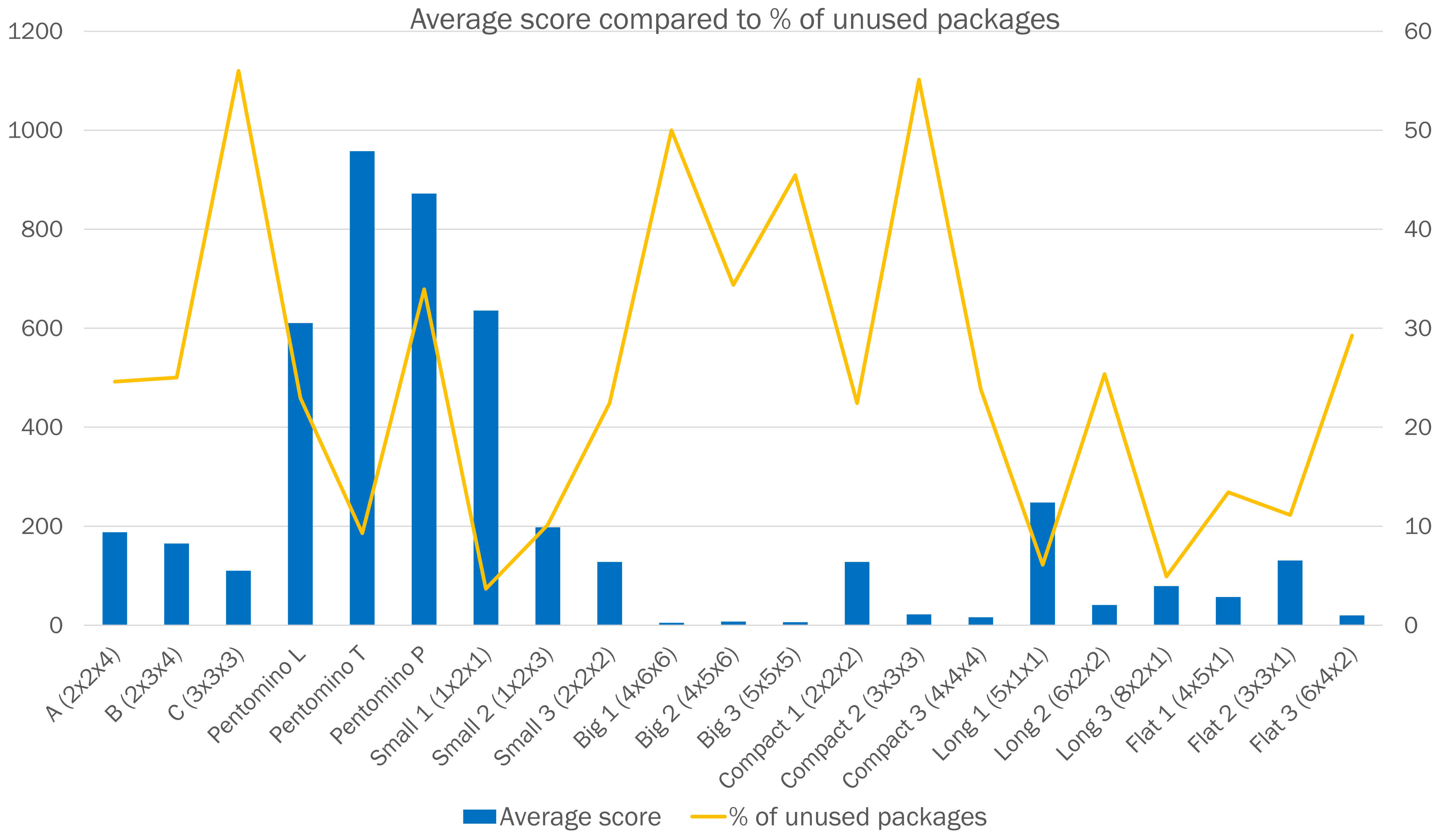

START
INITIALIZE POPULATION
SAVE THE FITTEST INDIVIDUAL
CHOOSE THE PARENTS
REPRODUCTION
CALCULATE THE FITNESS
NEW FITTEST INDIVIDUAL?
YES
REPLACE THE FITTEST INDIVIDUAL
NO
IS THE CURRENT FITTEST INDIVIDUAL ABOVE A SATISFACTORY VALUE?
NO
YES
HAVE THERE BEEN CHANGES IN THE FITTEST INDIVIDUAL IN THE LAST N GENERATIONS?
YES
NO
END

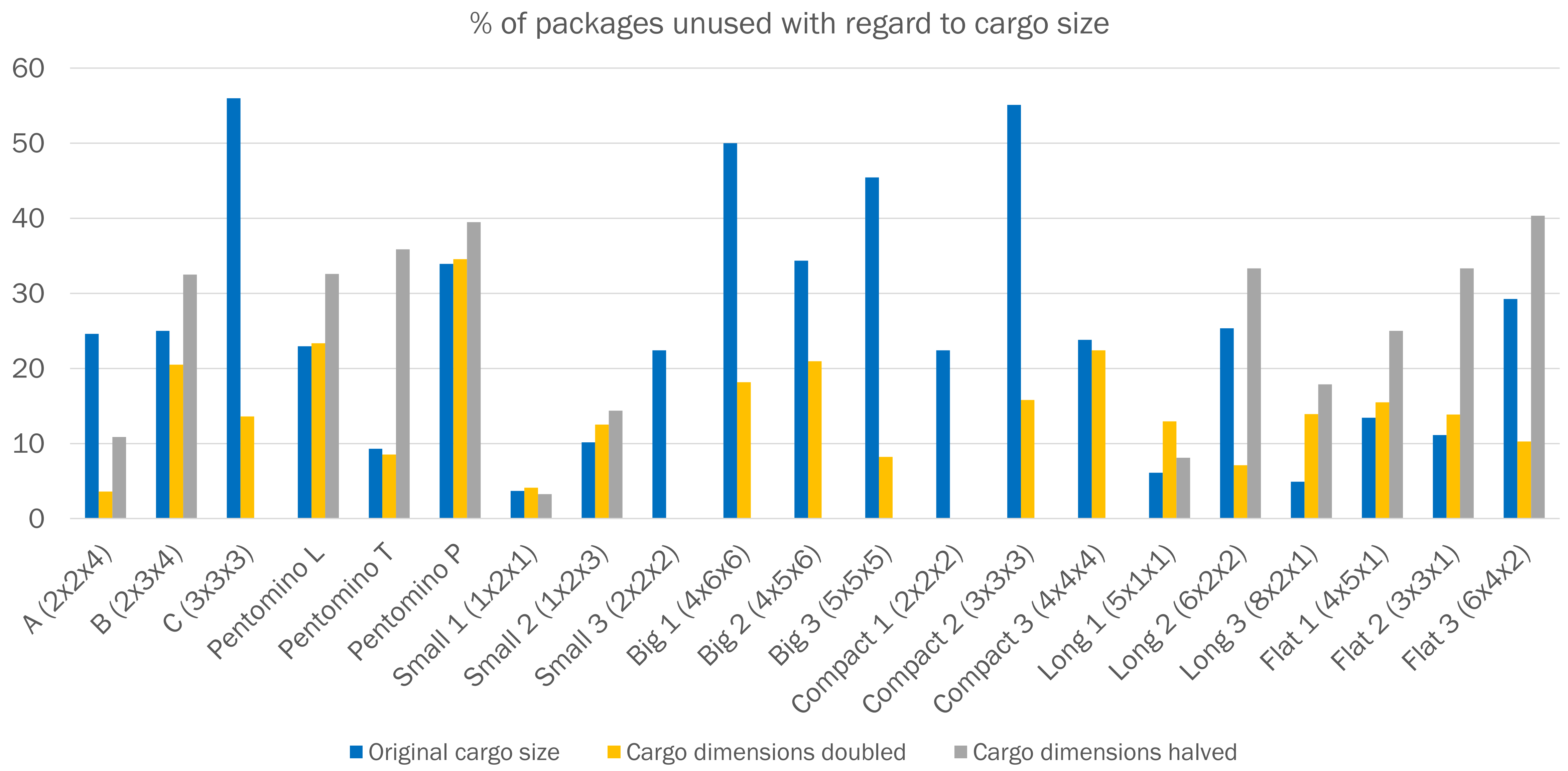
EXPERIMENTS AND RESULTS

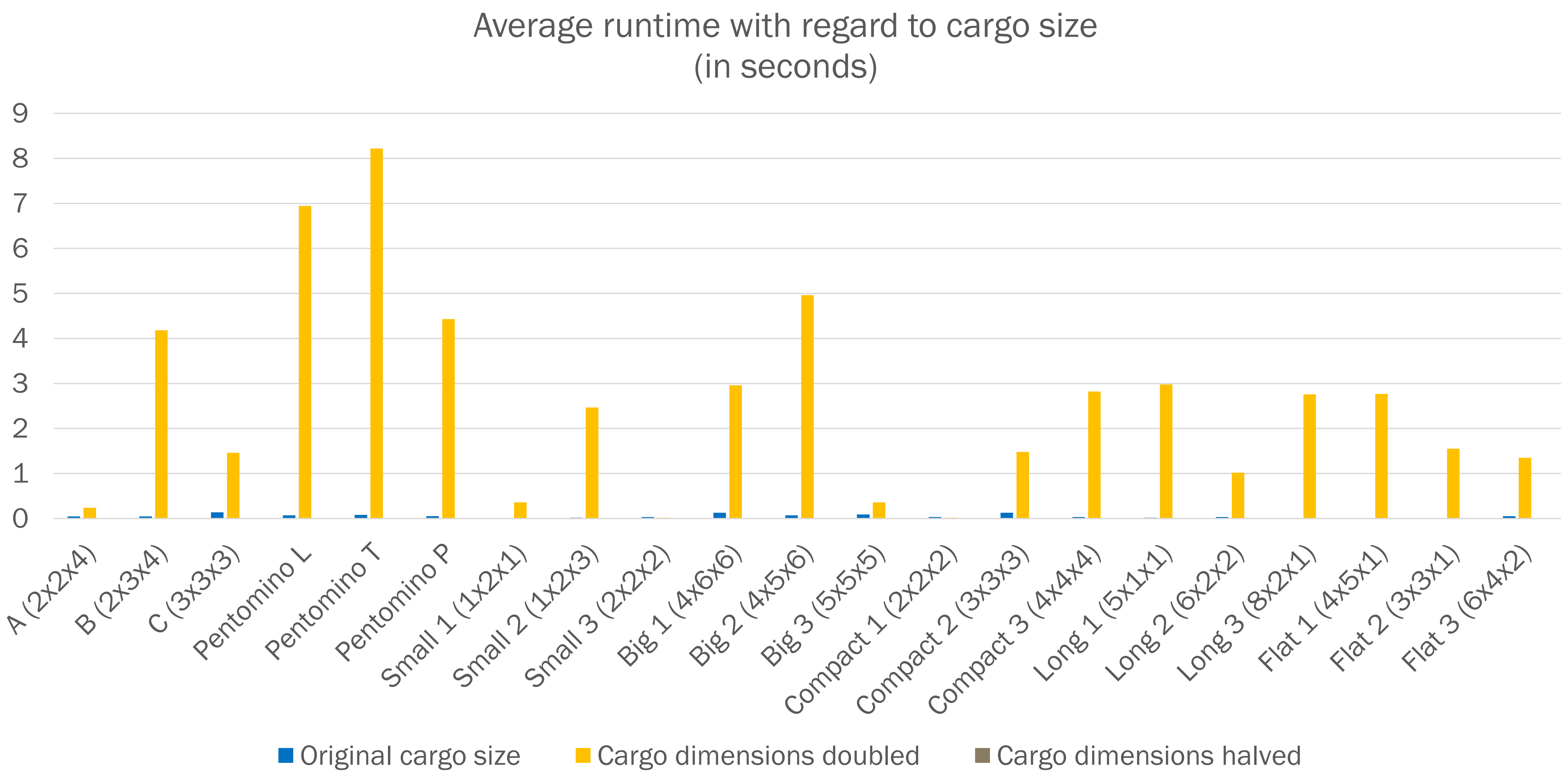
EXPERIMENTS AND RESULTS (GREEDY ALGORITHM)



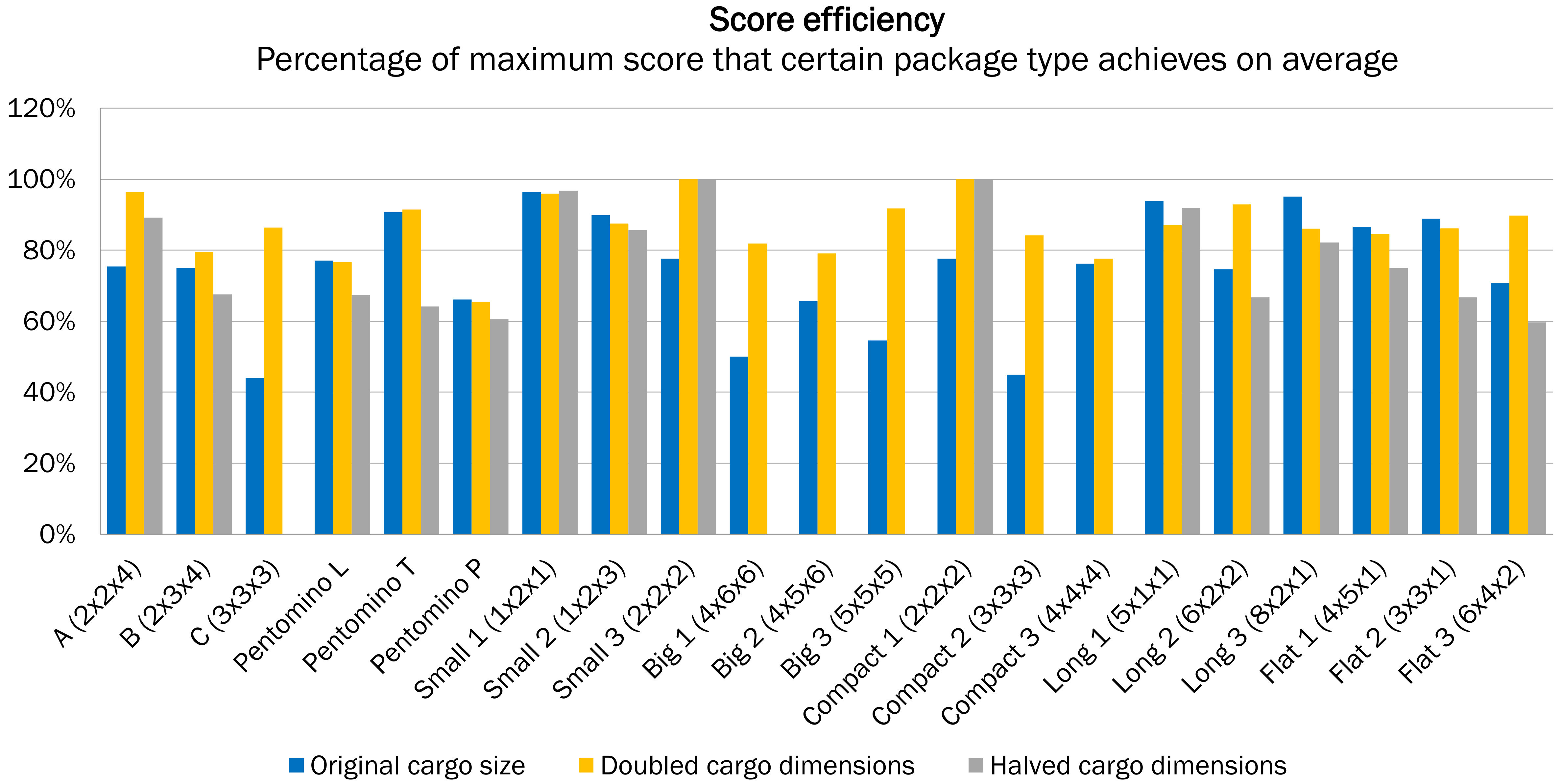

Custom package sizes - Scores and performance
700
600
500
400
300
200
100
0
14
12
10
8
6
4
2
0
Small 1 (1x2x1)
Small 2 (1x2x3)
Small 3 (2x2x2)
Big 1 (4x6x6)
Big 2 (4x5x6)
Big 3 (5x5x5)
Compact 1 (2x2x2)
Compact 2 (3x3x3)
Compact 3 (4x4x4)
Long 1 (5x1x1)
Long 2 (6x2x2)
Long 3 (8x2x1)
Flat 1 (4x5x1)
Flat 2 (3x3x1)
Flat 3 (6x4x2)
Best score
Average score
Worst score
Total runtime


Average score compared to % of unused packages
1200
1000
800
600
400
200
0
60
50
40
30
20
10
0
A (2x2x4)
B (2x3x4)
C (3x3x3)
Pentomino L
Pentomino T
Pentomino P
Small 1 (1x2x1)
Small 2 (1x2x3)
Small 3 (2x2x2)
Big 1 (4x6x6)
Big 2 (4x5x6)
Big 3 (5x5x5)
Compact 1 (2x2x2)
Compact 2 (3x3x3)
Compact 3 (4x4x4)
Long 1 (5x1x1)
Long 2 (6x2x2)
Long 3 (8x2x1)
Flat 1 (4x5x1)
Flat 2 (3x3x1)
Flat 3 (6x4x2)
Average score
% of unused packages

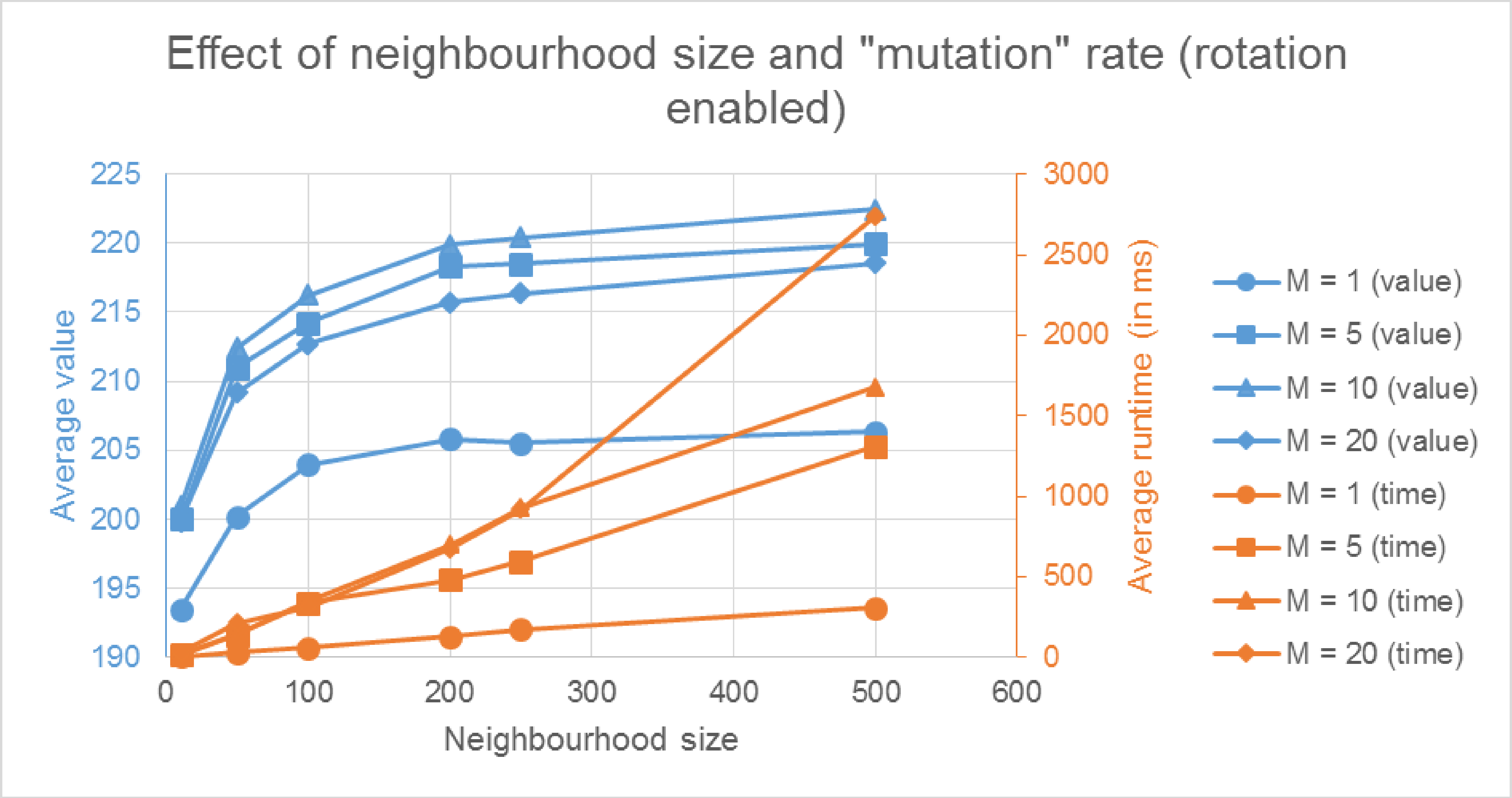

% of packages unused with regard to cargo size
60
50
40
30
20
10
0
A (2x2x4)
B (2x3x4)
C (3x3x3)
Pentomino L
Pentomino T
Pentomino P
Small 1 (1x2x1)
Small 2 (1x2x3)
Small 3 (2x2x2)
Big 1 (4x6x6)
Big 2 (4x5x6)
Big 3 (5x5x5)
Compact 1 (2x2x2)
Compact 2 (3x3x3)
Compact 3 (4x4x4)
Long 1 (5x1x1)
Long 2 (6x2x2)
Long 3 (8x2x1)
Flat 1 (4x5x1)
Flat 2 (3x3x1)
Flat 3 (6x4x2)
Original cargo size
Cargo dimensions doubled
Cargo dimensions halved



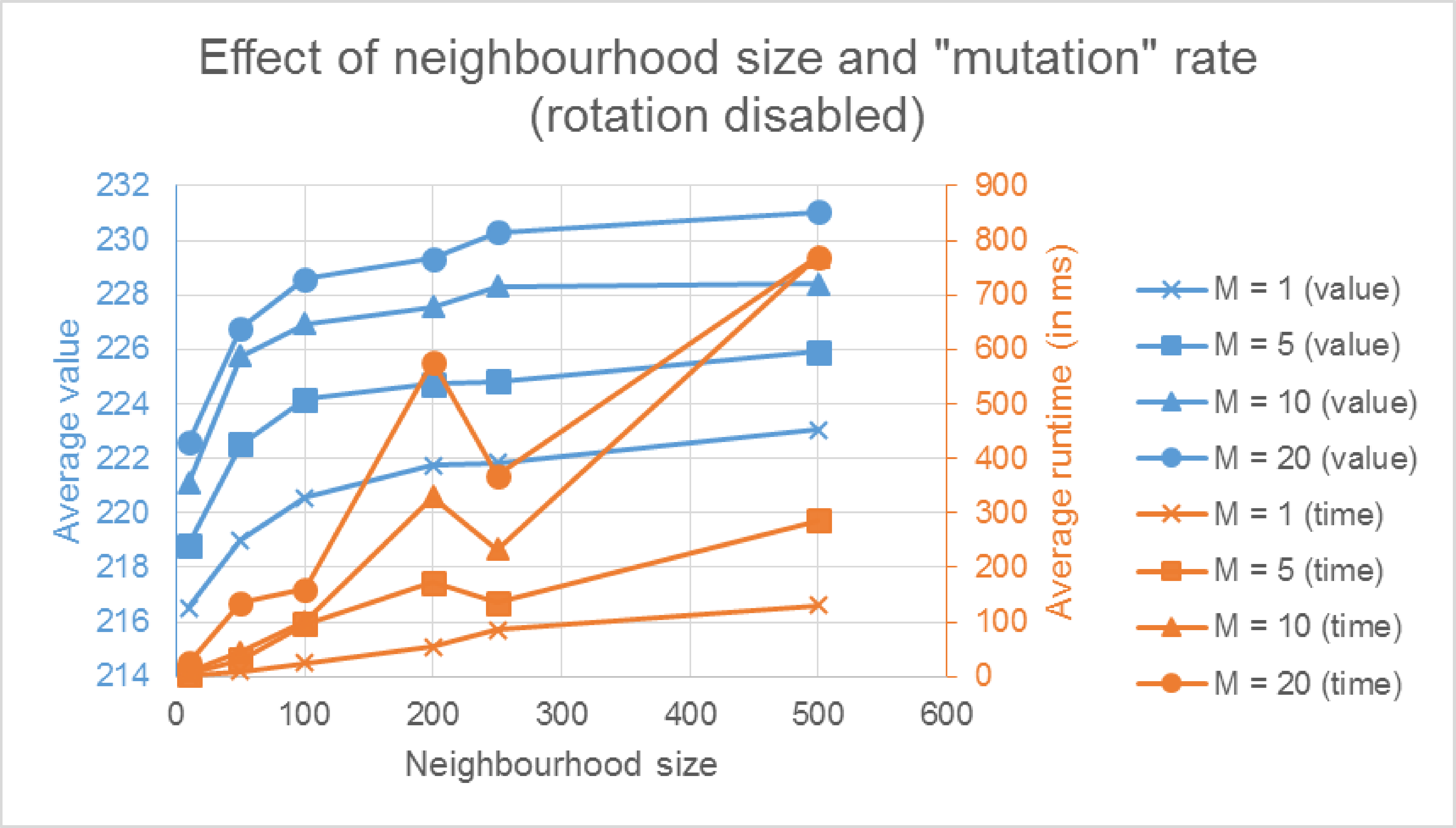
Average runtime with regard to cargo size
(in seconds)
9
8
7
6
5
4
3
2
1
0
A (2x2x4)
B (2x3x4)
C (3x3x3)
Pentomino L
Pentomino T
Pentomino P
Small 1 (1x2x1)
Small 2 (1x2x3)
Small 3 (2x2x2)
Big 1 (4x6x6)
Big 2 (4x5x6)
Big 3 (5x5x5)
Compact 1 (2x2x2)
Compact 2 (3x3x3)
Compact 3 (4x4x4)
Long 1 (5x1x1)
Long 2 (6x2x2)
Long 3 (8x2x1)
Flat 1 (4x5x1)
Flat 2 (3x3x1)
Flat 3 (6x4x2)
Original cargo size
Cargo dimensions doubled
Cargo dimensions halved


Score efficiency
Percentage of maximum score that certain package type achieves on average
120%
100%
80%
60%
40%
20%
0%
A (2x2x4)
B (2x3x4)
C (3x3x3)
Pentomino L
Pentomino T
Pentomino P
Small 1 (1x2x1)
Small 2 (1x2x3)
Small 3 (2x2x2)
Big 1 (4x6x6)
Big 2 (4x5x6)
Big 3 (5x5x5)
Compact 1 (2x2x2)
Compact 2 (3x3x3)
Compact 3 (4x4x4)
Long 1 (5x1x1)
Long 2 (6x2x2)
Long 3 (8x2x1)
Flat 1 (4x5x1)
Flat 2 (3x3x1)
Flat 3 (6x4x2)
Original cargo size
Doubled cargo dimensions
Halved cargo dimensions

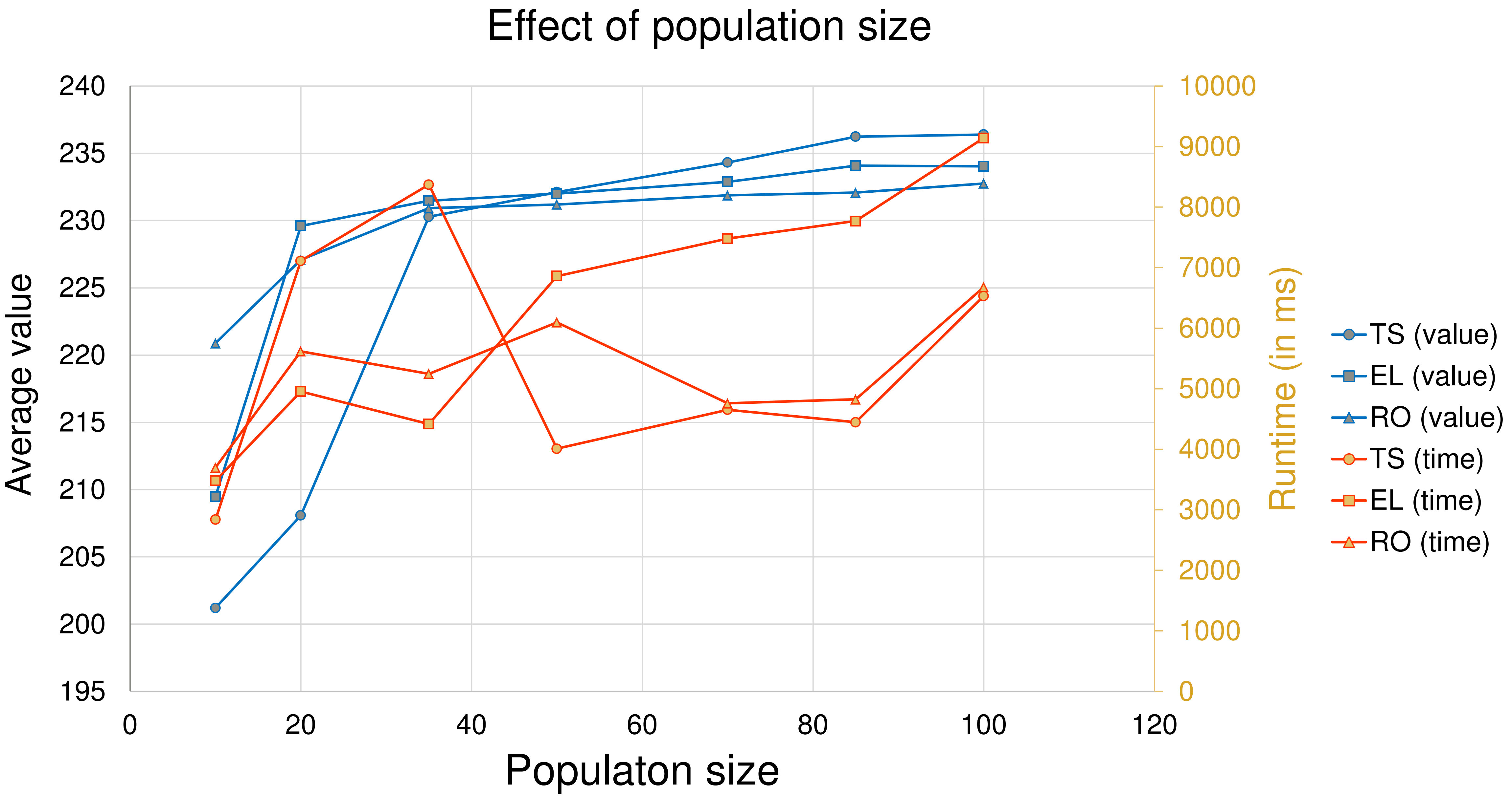
EXPERIMENTS AND RESULTS (HILL CLIMBING)



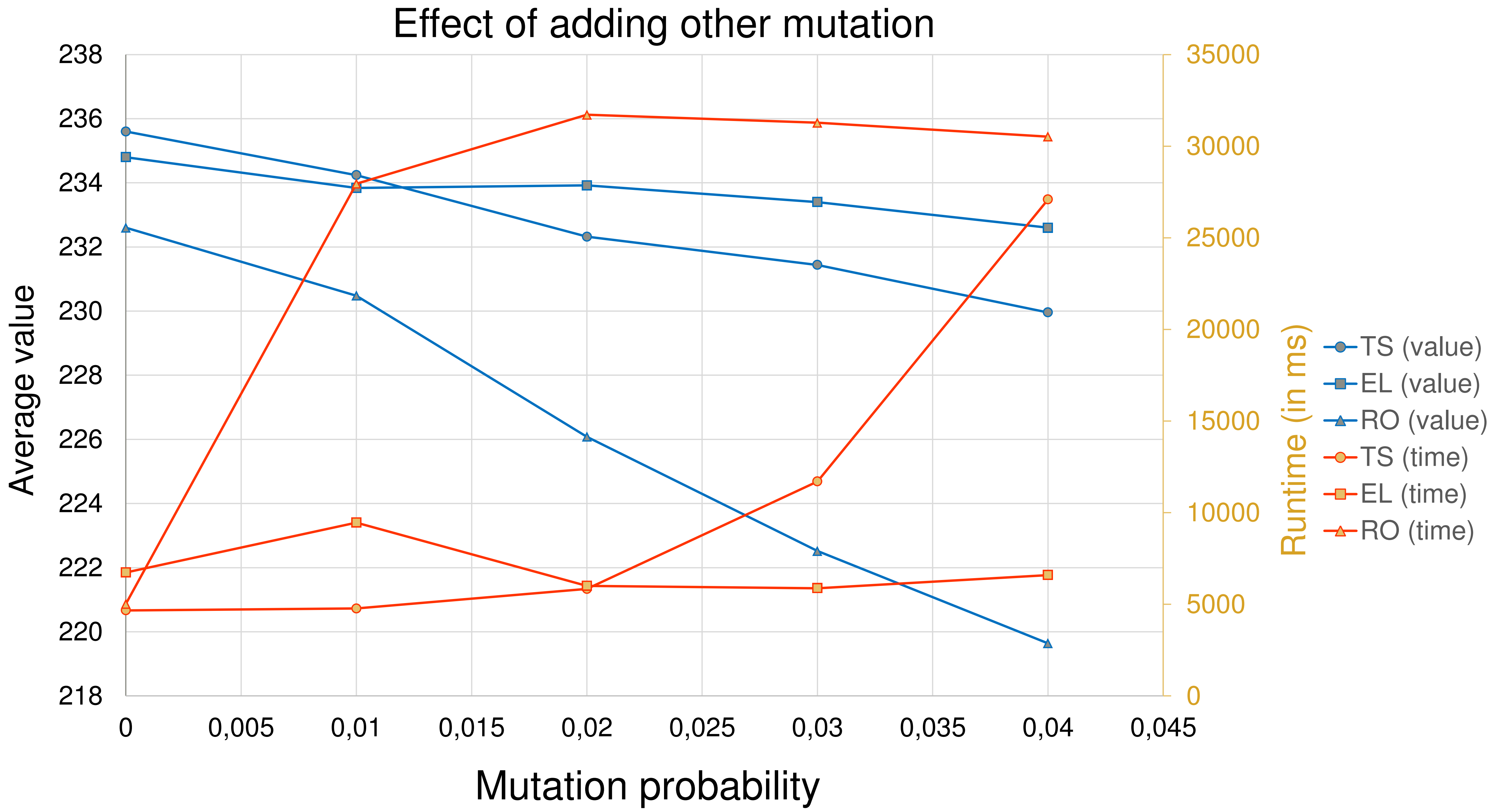
- Mutation rate can increase average runtime.
- Neighbourhood size is related to the performance.
- Better results with rotation disabled.


Effect of neighbourhood size and "mutation" rate
(rotation disabled)
232
230
228
226
224
222
220
218
216
214
Average value
900
800
700
600
500
400
300
200
100
0
Average runtime (in ms)
0
100
200
300
400
500
600
Neighbourhood size
M = 1 (value)
M = 5 (value)
M = 10 (value)
M = 20 (value)
M = 1 (time)
M = 5 (time)
M = 10 (time)
M = 20 (time)

EXPERIMENTS AND RESULTS (GENETIC ALGORITHM)



- Increase in population size yields better results
- Affects the tournament selection method the most
  - *Large increase in the average result achieved*
  - *Significant decrease in runtime*


Effect of adding other mutation
Average value
238
236
234
232
230
228
226
224
222
220
218
0
0,005
0,01
0,015
0,02
0,025
0,03
0,035
0,04
0,045
Mutation probability
Runtime (in ms)
35000
30000
25000
20000
15000
10000
5000
0
TS (value)
EL (value)
RO (value)
TS (time)
EL (time)
RO (time)

- Increasing the mutation probability **decreases** the performance of the algorithm (both value and runtime wise)
- This could be due to the way good solutions are achieved (no rotations)

EXTERNAL FACTORS

- Experiments with different numbers of packages and new package types (e.g. small, big, long etc.)
- Packages used have similar properties
- All with value 1

Defining properties	V	T (ms)	G	% 1D	Test No.
Small packages (inf)	289.26	12479.16	50	0.877	1
Big packages (inf)	10	61883.96	123	0.909	2
Cubes (inf)	108.56	45626.04	116	0.658	3
Long packages (inf)	248.6	85670.68	29.72	0.942	4
Flat packages (inf)	130.68	81555.08	121.44	0.895	5
A, B, C + 1 similar (15)	186.24	33130.76	165.52	0.955	6
A, B, C + 2 similar (15)	187.6	39986.68	163.8	0.947	7
A, B, C + 3 similar (10)	140.04	37154.92	220.4	0.946	8
A, B, C + 1 similar (inf)	213.36	51289.88	80.2	0.864	9
A, B, C + 2 similar (inf)	201.36	72838.04	66.56	0.815	10
A, B, C + 3 similar (inf)	204.12	73388.08	74.24	0.826	11

- Performance worse than with standard packages
- Still finds good solutions
- Could probably adapt parameters to fit other packages better

CONCLUSIONS

- *Greedy algorithm is easy to implement and is easy to change*
- *Hill climbing algorithm is fast and finds a good solution*
- *Genetic algorithm gets closest to the optimal solution and has a decent runtime*
- *GA struggles with high amounts of packages and different package types → can be improved by changing internal parameters*