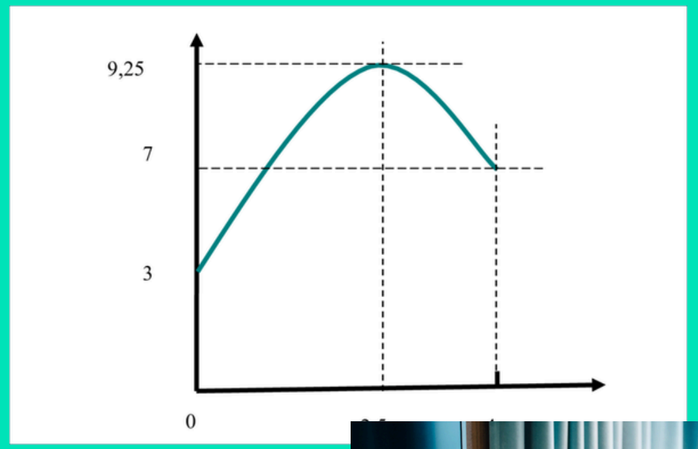
MULTI KNAPSACK PROBLEM.

By: Team E
May 2024.

Importance of the Multiknapsack problem..

- Critical for solving complex optimization problems.
- Improves operational efficiency in various sectors.
- Enables better resource management.



Real world applications

- Logistics: Optimal loading of trucks and containers.
- Resource Allocation: Efficient budget and manpower distribution.
- Scheduling: Assigning tasks to time slots or resources.

Mathematical formulation

Data/Parameters

n: number of items

m: number of knapsacks

w_i: weight of item i

v_i: value of item i

c_j: capacity of knapsack j

Decisions

x_{ij}: 1 if item i is placed in knapsack j, 0 otherwise

Objetive

$$\sum_{j=1}^{m} \sum_{i=1}^{n} v_i \cdot x_{ij}$$

Contraints

$$\sum_{i=1}^{n} w_i \cdot x_{ij} \leq c_j \text{ for each knapsack } j$$

$$\sum_{j=1}^{m} x_{ij} \leq 1 \text{ for each item } i$$

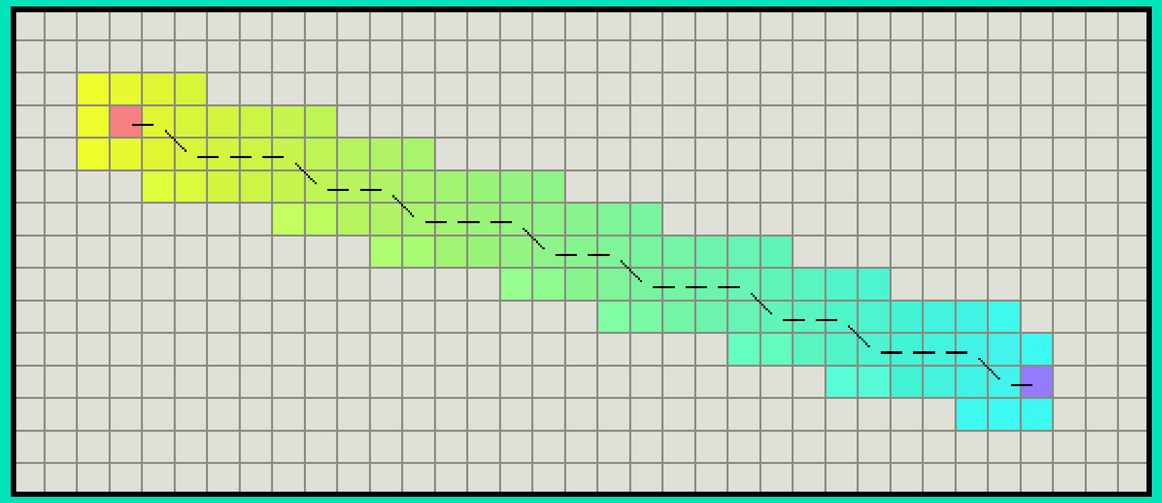
$$x_{ij} \in \{0, 1\}"$$

How does a simple example begin?

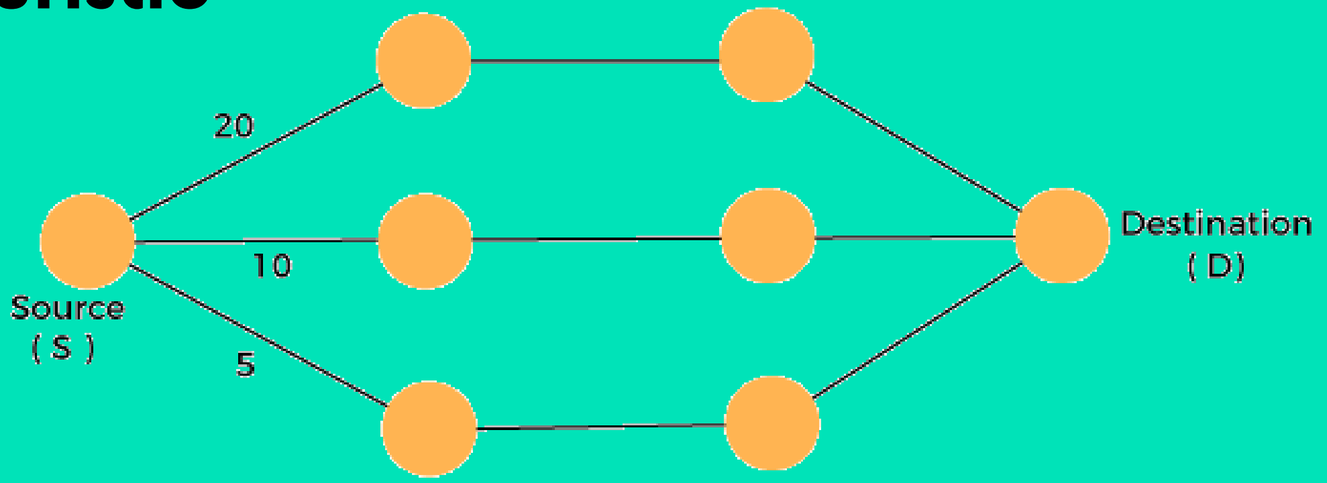
- Item 1: weight = 20, value = 40
- Item 2: weight = 30, value = 50
- Item 3: weight = 40, value = 60

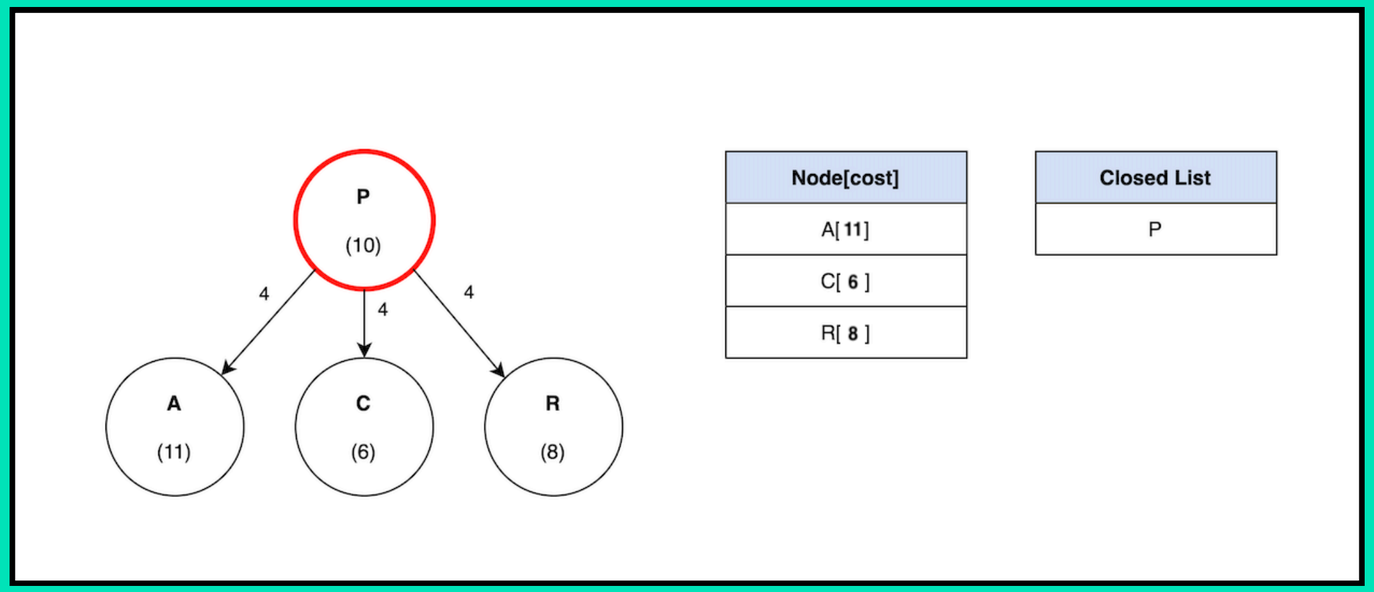
- Max capacity per knapsack: 60 units

Metrics for a Solution to be Considered

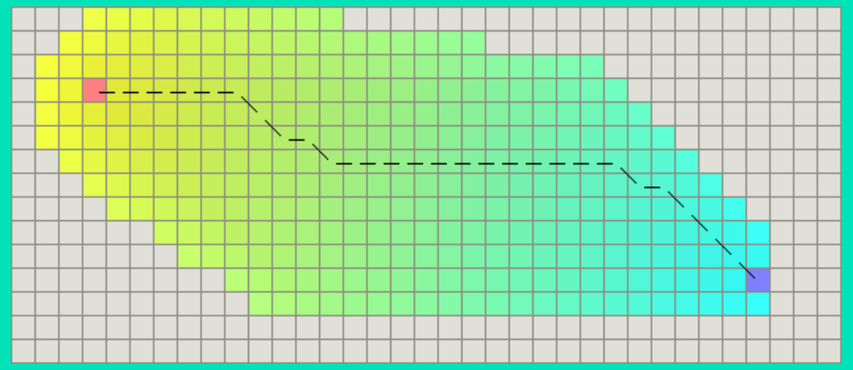


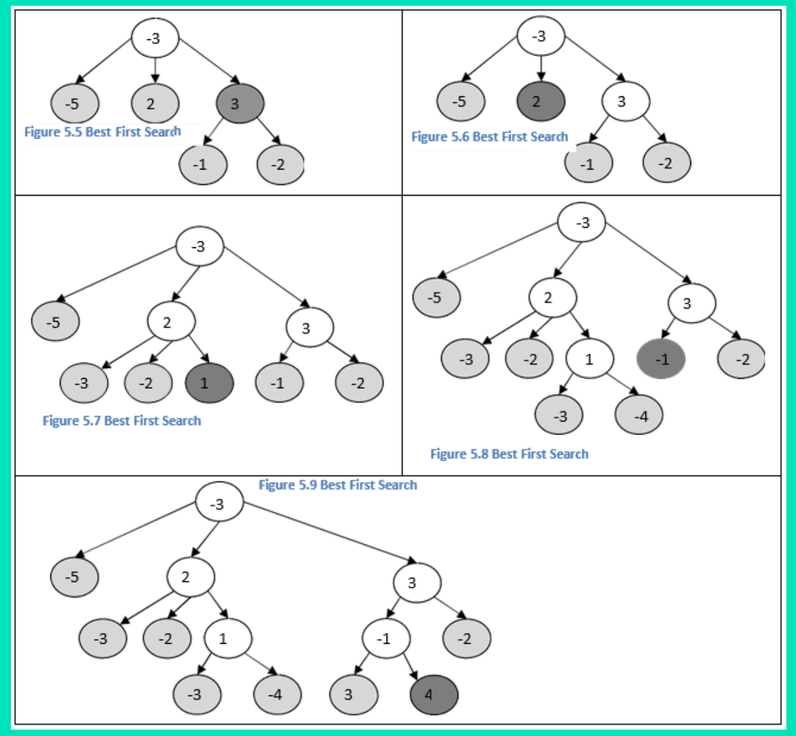
Construction heurístic: Highest value first heuristic





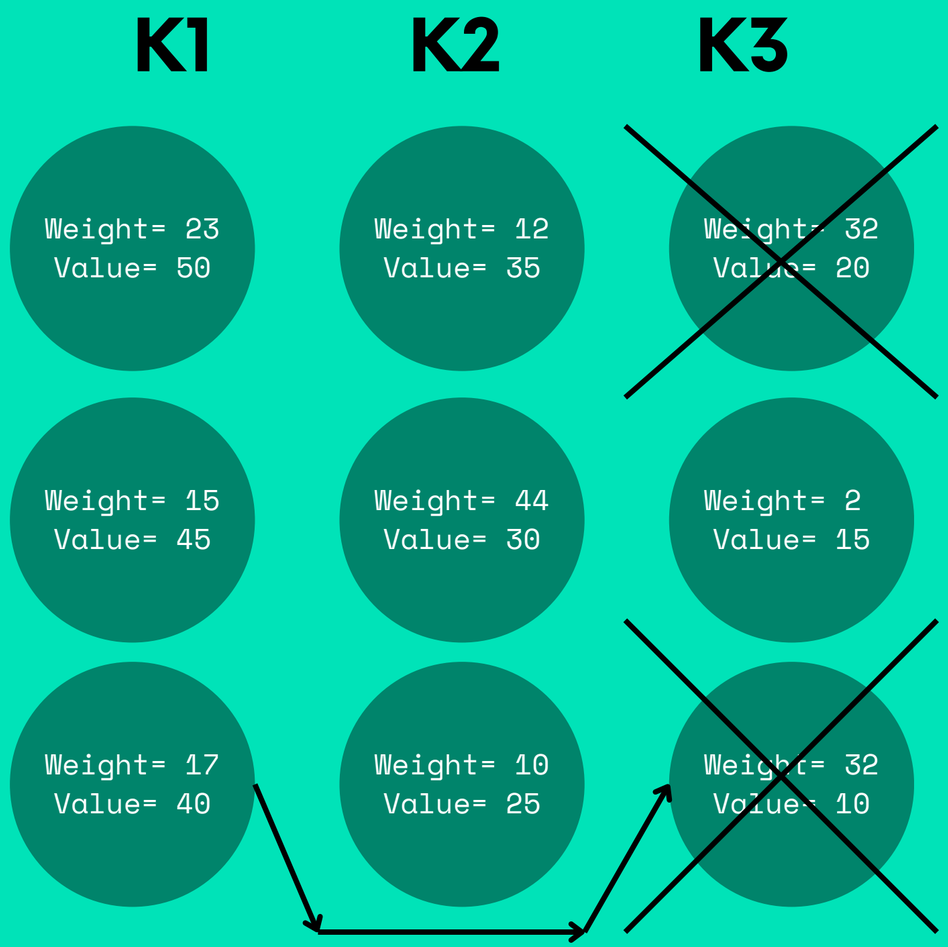
Spirit and pseudocode

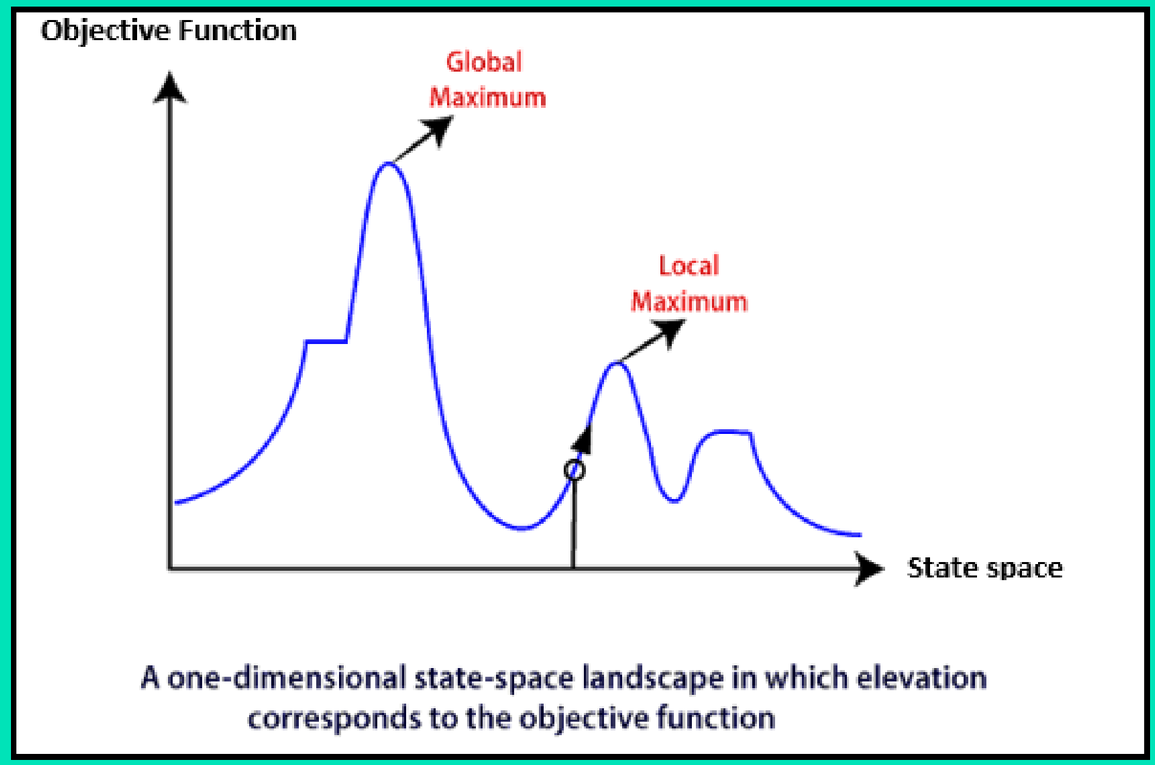




Oa->Kaj
a+1
Wj+=WOa
if Wj>>MWj-> remove Oa from Kaj and j+1
O=Object
K=Knapsack
W=Weight from knapsack
WO= Weight from object
MW= Max Weight for the napsack
j= Napsack counter
a= Object counter

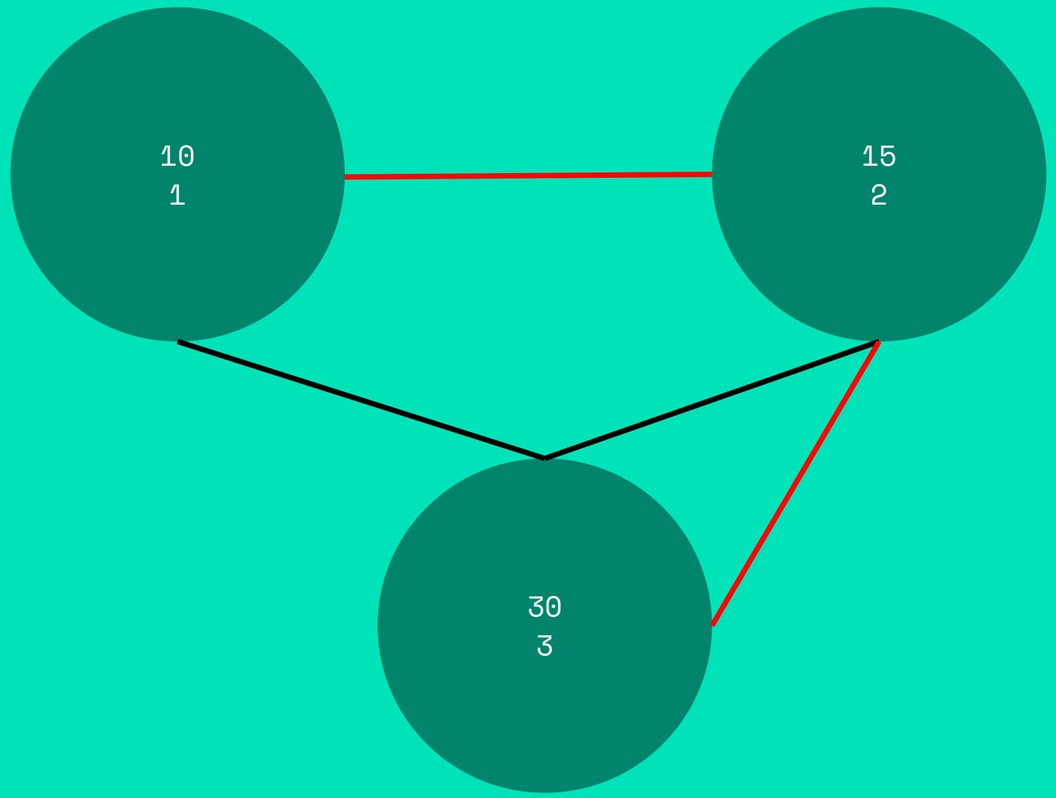
Local Search Heuristic: Remove 2 items from a Napsack to put an item from another one



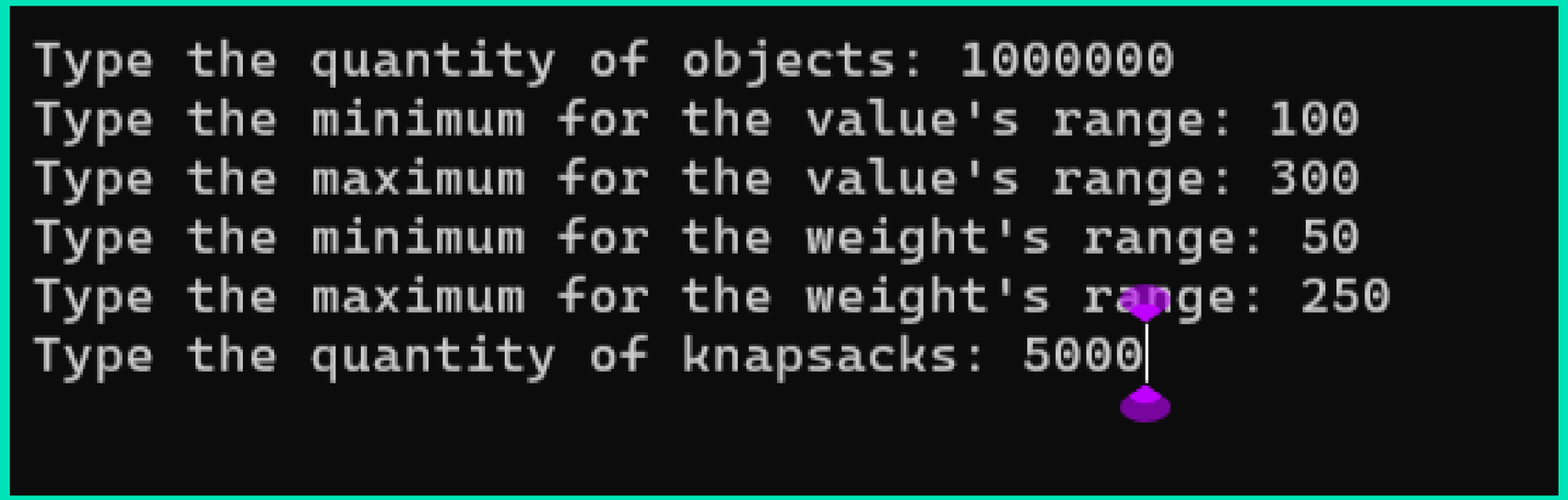


A one-dimensional state-space landscape in which elevation corresponds to the objective function

Methodology Usted: First Found



Instance Generator



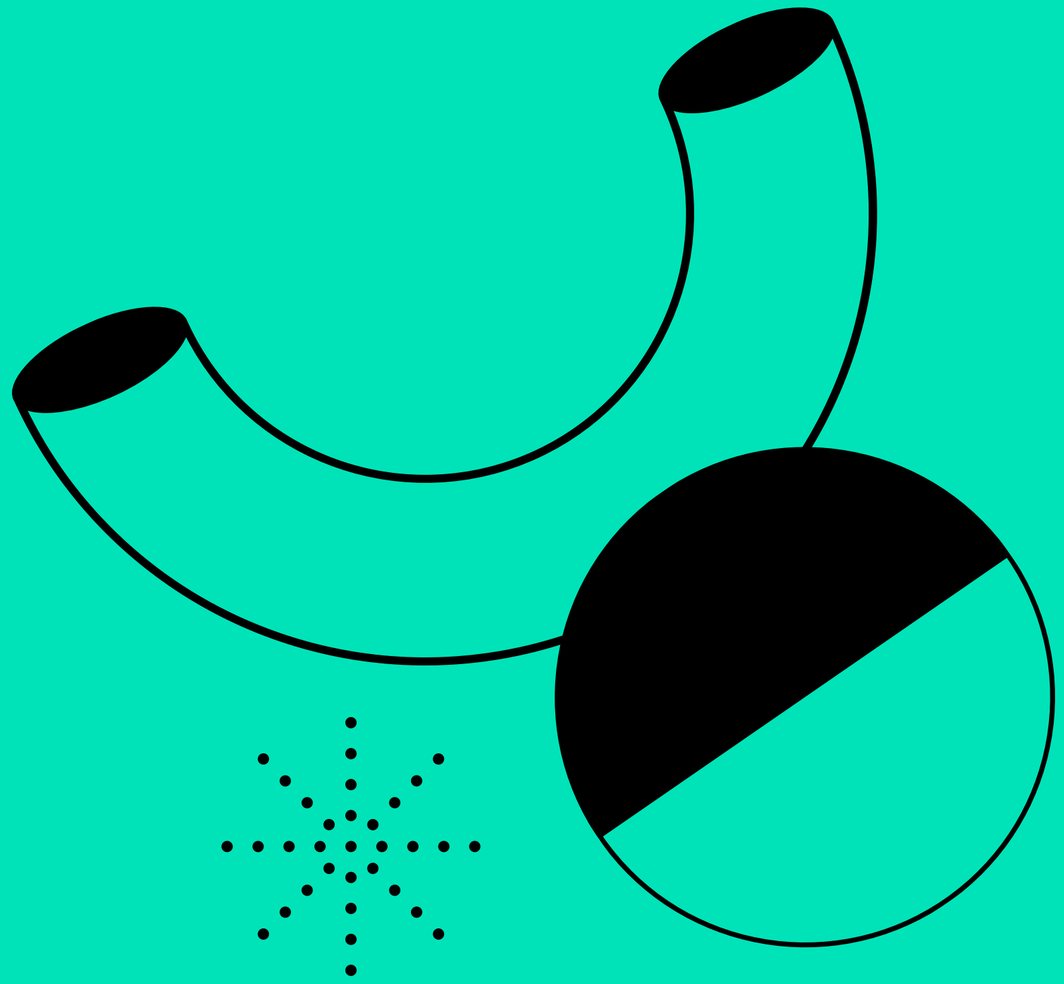
Experimental process

1. Instance Generator → `Knapsack.dat`

2. Heuristic 1 →
`Knaps_value.dat`
`Knaps_weight.dat`
`unused.dat`
`data.dat`

2. Local Search → `Final Solutions.`

FINAL RESULTS.



Small instances

Set: Small
 # Items: 1000
 # knapsacks: 5
Max weight p/knapsack: 30000

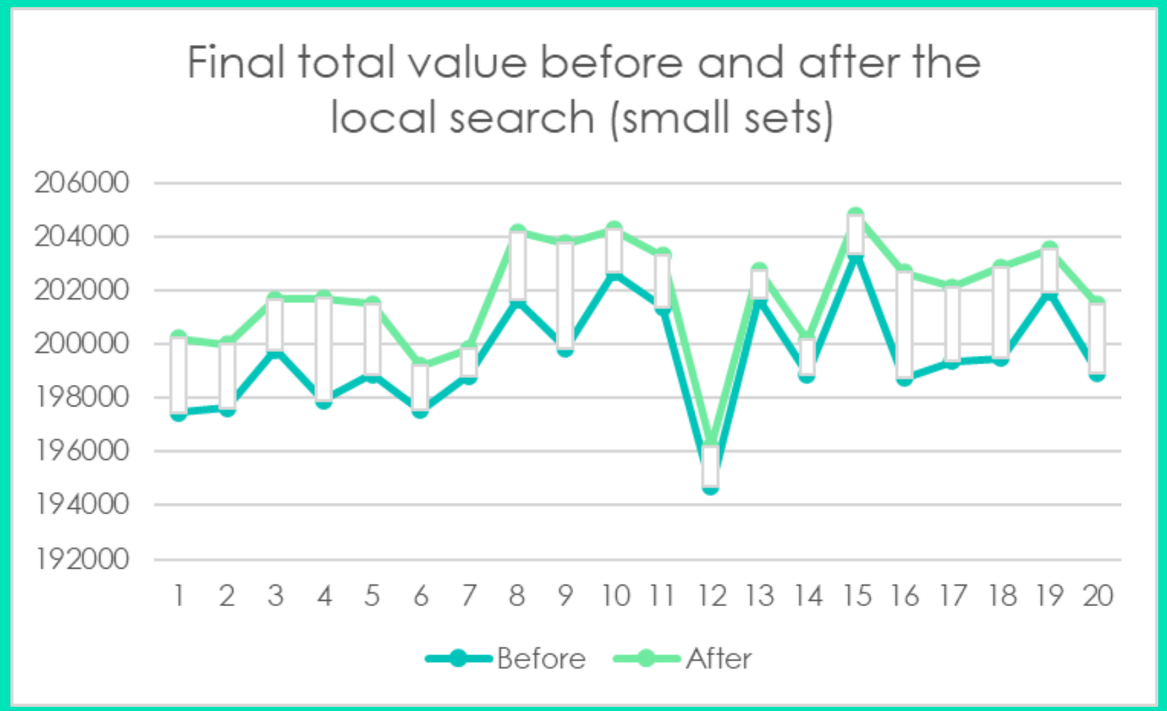
Value range: 100-300
Weight range: 50-250

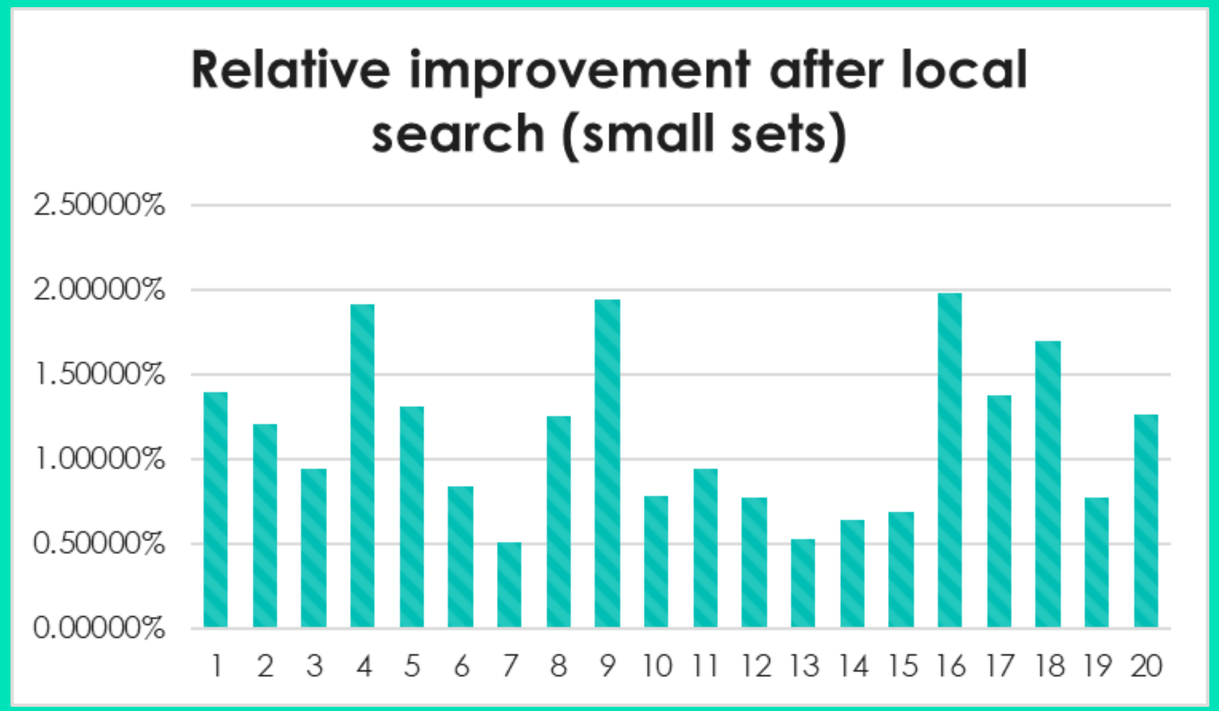
Heuristic used: value-sorted
Local search heuristic used: 2-opt
Algorithm used: First Found

Sample	Final Combined Value	Execution's time
1	197467	0.00001
2	197613	0.000009
3	199786	0.00001
4	197897	0.000009
5	198882	0.000009
6	197538	0.000009
7	198818	0.00001
8	201641	0.000009
9	199846	0.00001
10	202666	0.000009
11	201382	0.00001
12	194691	0.00001
13	201699	0.00001
14	198879	0.000009
15	203363	0.000009
16	198727	0.000008
17	199369	0.00001
18	199473	0.00001
19	201963	0.00001
20	198943	0.00001

Sample	Final Combined Value	Execution's time	Absolute improvement	Relative improvement
1	200220	0.86346	2753	1.39416%
2	200000	0.97508	2387	1.20792%
3	201678	0.93531	1892	0.94701%
4	201680	1.00281	3783	1.91160%
5	201486	1.01391	2604	1.30932%
6	199187	0.80393	1649	0.83478%
7	199826	0.75668	1008	0.50700%
8	204163	0.99691	2522	1.25074%
9	203729	1.04725	3883	1.94300%
10	204257	0.74694	1591	0.78504%
11	203277	0.94280	1895	0.94100%
12	196193	0.96368	1502	0.77148%
13	202757	0.81492	1058	0.52454%
14	200152	0.78398	1273	0.64009%
15	204765	0.74971	1402	0.68941%
16	202670	0.88750	3943	1.98413%
17	202120	0.77429	2751	1.37985%
18	202853	0.95343	3380	1.69446%
19	203528	0.88025	1565	0.77489%
20	201455	0.82404	2512	1.26267%
		Average	2267.65	1.13765%

Small instances





Medium instances

Set: Medium
 # Items: 100000
 # knapsacks: 500
Max weight p/knapsack: 22500

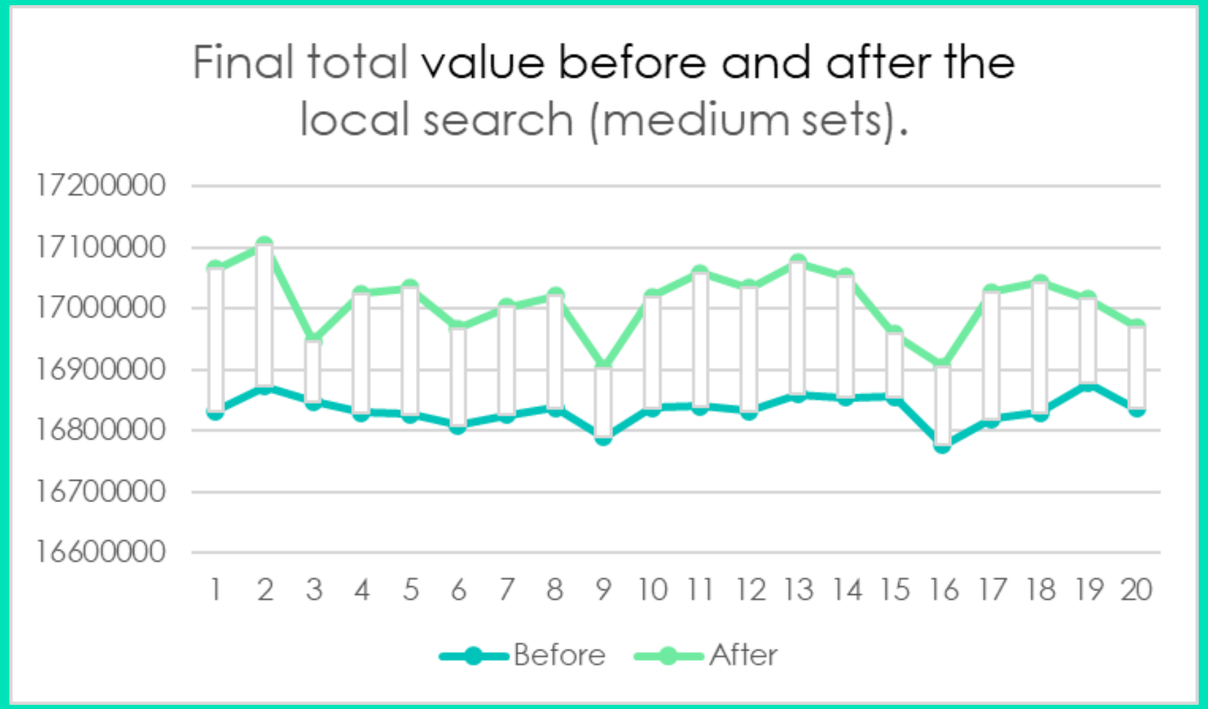
Value range: 100-300
Weight range: 50-250

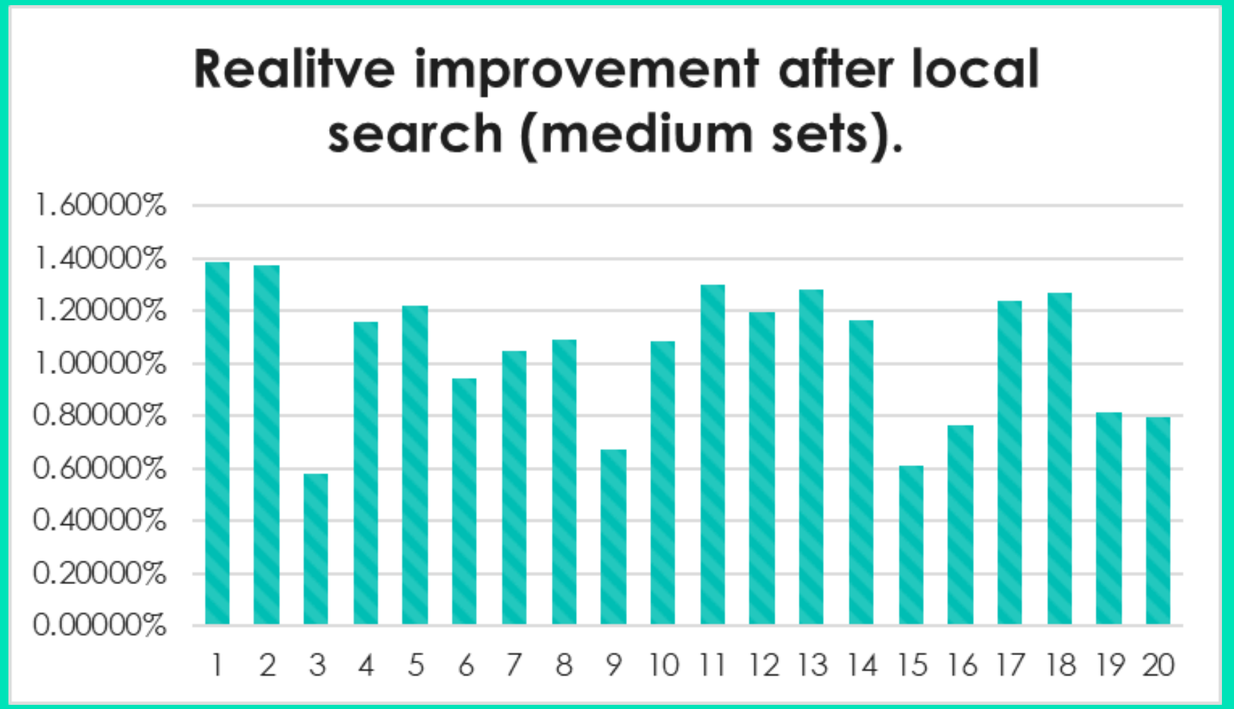
Heuristic used: value-sorted
Local search heuristic used: 2-opt
Algorithm used: First Found

Sample	Final Combined Valu	Execution's time
1	16831771	0.000009
2	16872269	0.00001
3	16848033	0.000009
4	16829585	0.00001
5	16827491	0.000009
6	16809195	0.000009
7	16826197	0.00001
8	16837344	0.000009
9	16789786	0.00001
10	16837587	0.000009
11	16839615	0.00001
12	16831825	0.00001
13	16860155	0.000009
14	16854953	0.00001
15	16855415	0.00001
16	16776632	0.00001
17	16818979	0.00001
18	16829969	0.000009
19	16878397	0.00001
20	16836469	0.000009

Sample	Final Combined Value	Execution's time	Absolute improvement	Relative improvement
1	17065027	37.564	233256	1.38581%
2	17103828	33.898	231559	1.37242%
3	16946329	37.493	98296	0.58343%
4	17024333	35.014	194748	1.15718%
5	17033354	34.443	205863	1.22337%
6	16967279	36.900	158084	0.94046%
7	17002226	39.100	176029	1.04616%
8	17021088	35.714	183744	1.09129%
9	16902372	38.285	112586	0.67056%
10	17020173	36.055	182586	1.08440%
11	17058149	36.919	218534	1.29774%
12	17033446	34.944	201621	1.19786%
13	17076211	35.492	216056	1.28146%
14	17051638	34.484	196685	1.16693%
15	16958923	34.732	103508	0.61409%
16	16905263	39.135	128631	0.76673%
17	17027522	39.056	208543	1.23993%
18	17043154	35.411	213185	1.26670%
19	17015559	37.381	137162	0.81265%
20	16969956	36.709	133487	0.79284%
		Average	176708.15	1.04960%

Medium instances





Large instances

Set: Large
 # Items: 1000000
 # knapsacks: 5000
Max weight p/knapsack: 225000

Value range: 100-300
Weight range: 50-250

Heuristic used: value-sorted
Local search heuristic used: 2-opt
Algorithm used: First Found

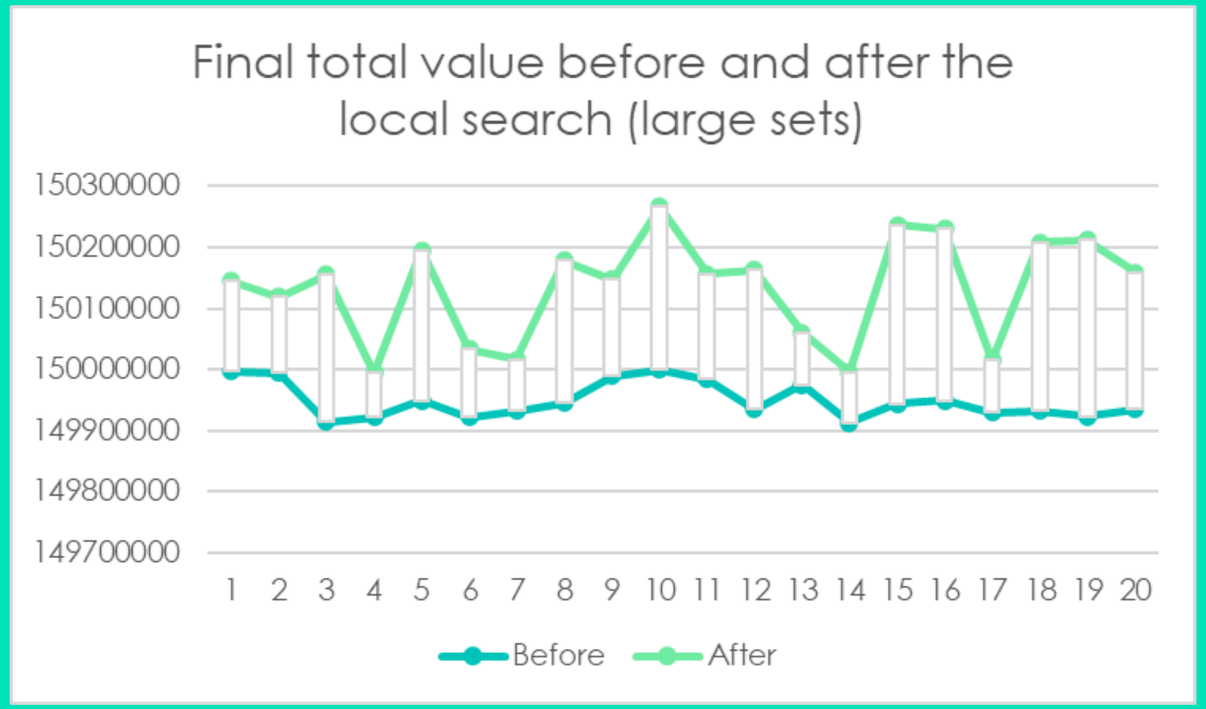
Sample	Final Combined Valu	Execution's time
1	149997809	0.03964
2	149994524	0.03964
3	149914887	0.03964
4	149921792	0.03964
5	149949880	0.03963
6	149921916	0.03964
7	149932848	0.03965
8	149945940	0.03964
9	149989870	0.03964
10	149999965	0.03965
11	149983950	0.03964
12	149934878	0.03964
13	149974851	0.03964
14	149912941	0.03965
15	149943835	0.03964
16	149948866	0.03964
17	149929943	0.03964
18	149932919	0.03964
19	149923952	0.03963
20	149934953	0.03964

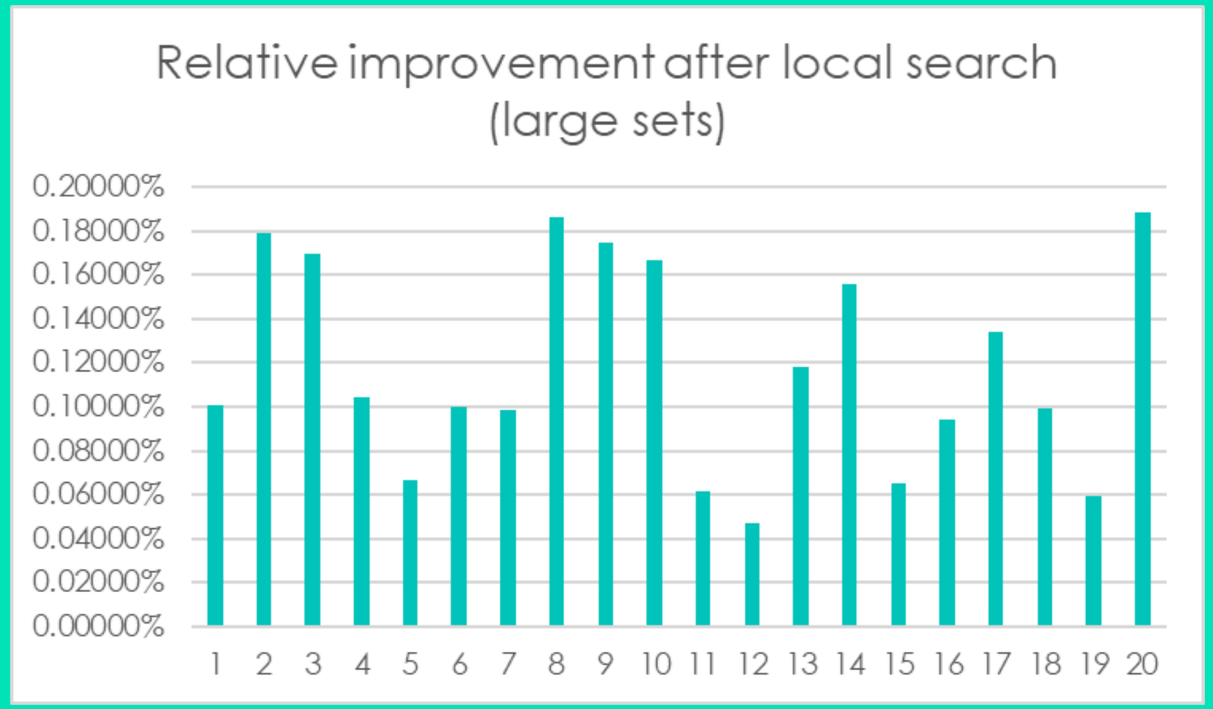
*Time limit: 10 minutes

Sample	Final Combined Valu	Execution's time*	Absolute improvemer	Relative improvemen
1	150145477	600.00	151230	0.10082%
2	150118676	600.00	268864	0.17925%
3	150154933	600.00	254287	0.16962%
4	149994762	600.00	156352	0.10429%
5	150196038	600.00	100334	0.06691%
6	150033357	600.00	149663	0.09983%
7	150017212	600.00	147619	0.09846%
8	150178623	600.00	279459	0.18637%
9	150147812	600.00	261856	0.17458%
10	150266410	600.00	250099	0.16673%
11	150156796	600.00	92839	0.06190%
12	150163486	600.00	70361	0.04693%
13	150060339	600.00	176713	0.11783%
14	149995484	600.00	233656	0.15586%
15	150237468	600.00	98400	0.06562%
16	150230176	600.00	141536	0.09439%
17	150014843	600.00	200586	0.13379%
18	150209072	600.00	148626	0.09913%
19	150212592	600.00	88828	0.05925%
20	158662	600.00	282172	0.18820%
		Average	177674	0.11849%

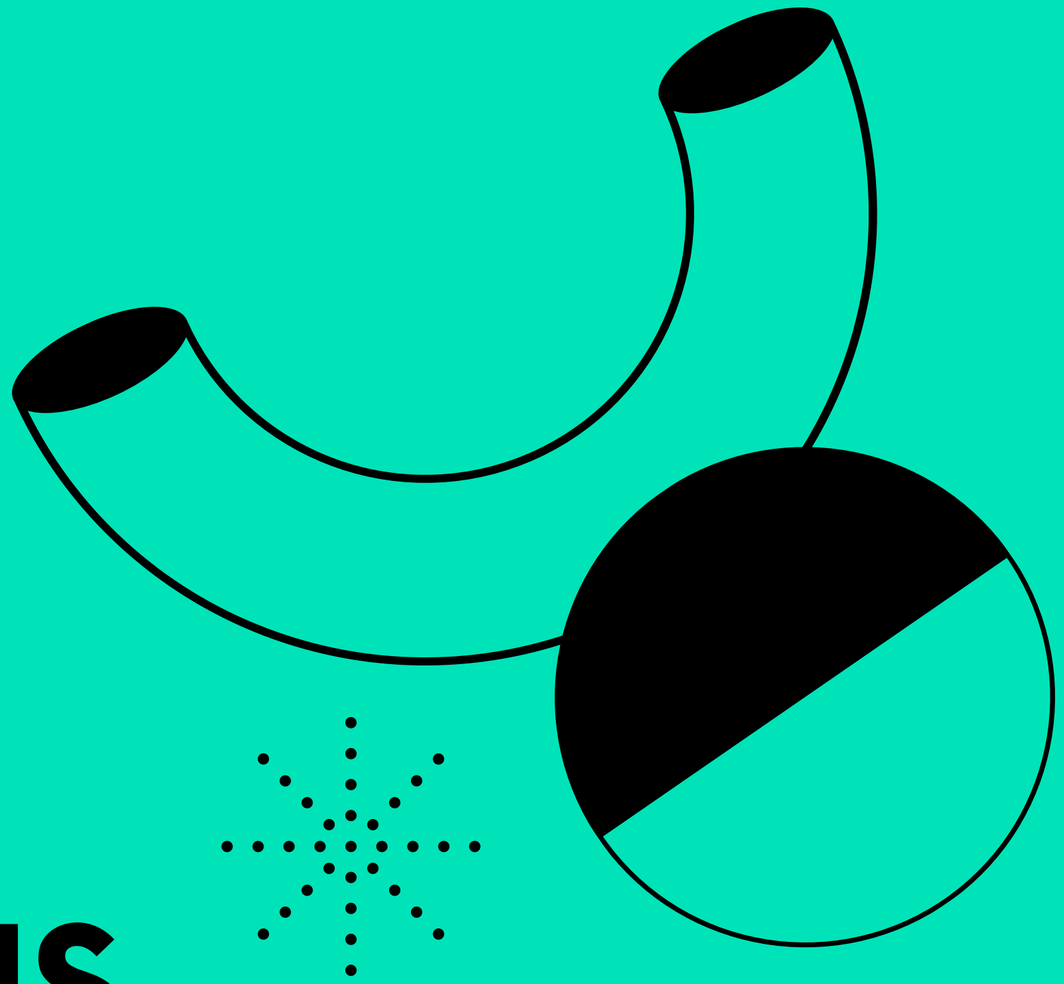
Título del gráfico

Large instances

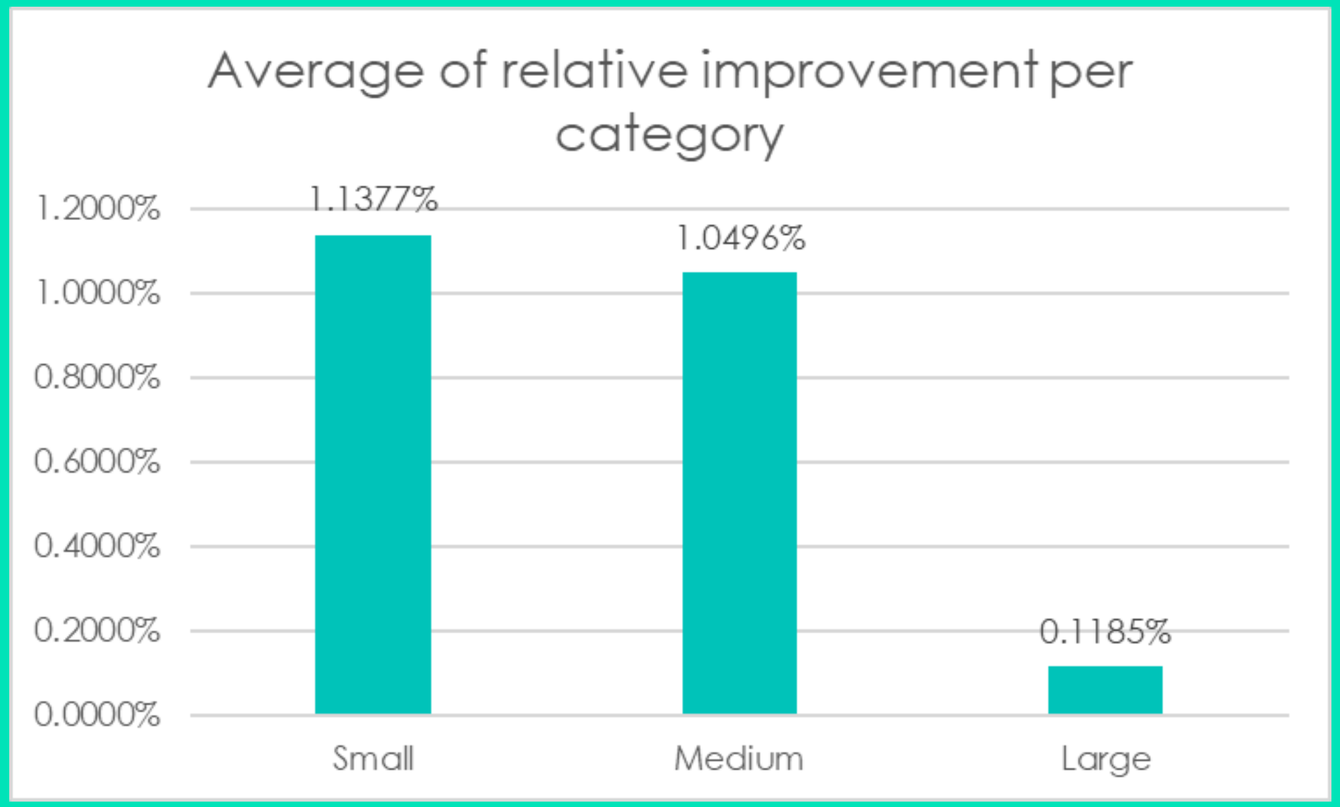




CONCLUSIONS.



Conclusions



Regardless of the instance size, the final solution will always improve unless there are resource constraints, such as time, which prevented us from making more extensive improvements in the larger instances. We propose conducting tests without such limitations to achieve better results.