0.1. Results obtained

In the tests performed, we used a 64-bit, 16GB, 3.6GHz, eight-core processor with *Linux Ubuntu 22.04.1 LTS 64-bit* as the operational system and *Python 3.10.4* as the programming language. We also used the well-known solver *Gurobi* (www.gurobi.com), version 9.5.2.

We ran Algorithm **??** in the 5 scenarios described in Table **??**, considering 6 methods for node-by-node solution: Gurobi (see **??**), ACO, NMO, TS, GRASP, and Shims (Algorithm **??**). In generating the items in each node, we consider 3 values for the parameter *volume*. The results obtained for the function f, with the corresponding run time in seconds, are shown in Tables 1 ($volume = 1.2$), 2 ($volume = 1.5$) and 3 ($volume = 2.0$). For each *method*, *scenario* and *volume*, 7 different instances were generated. Therefore, 105 tests (5 scenarios $\times$ 3 volumes $\times$ 7 instances) were performed for each method.

The average values were presented for f_π and, for the run time, the worst result obtained. To facilitate the comparison between the methods, we added a last column in these tables, where two values are indicated:

- **Normalized**: value between 0 and 1, which corresponds to the ratio between the sum of f values obtained by the method in all scenarios and the sum of the best values obtained among all methods in all scenarios. The higher the value of **Normalized**, the closer the method approached the best solutions found.
- **Speed-up**: ratio of the sums of the worst runtimes of all scenarios and the sum of the method runtimes in all scenarios. The method with the highest **Speed-up** is the fastest.

In each *scenario*, we indicate in bold the best value of f found. In each table, we also indicate in bold the best **Normalized** and **Speed-up** values.

Table 1: Methods and scenarios with $volume = 1.2$

method	**Results**	**1**	**2**	**3**	**4**	**5**	**Normalized Speed-up**
Gurobi	f	6.10	**10.02**	**10.82**	**12.13**	**12.10**	**0.997**
	run time (s)	14	51	243	1,417	9,720	1.000
TS	f	6.56	8.49	9.19	11.25	11.48	0.906
	run time (s)	4	17	86	516	3,600	2.709
GRASP	f	6.76	8.85	9.65	11.75	11.54	0.937
	run time (s)	4	17	80	472	3,265	2.984
ACO	f	**6.77**	8.85	9.65	11.75	11.54	0.937
	run time (s)	4	17	86	513	3,596	2.714
NMO	f	6.74	8.80	9.61	11.76	11.50	0.934
	run time (s)	4	17	86	508	3,480	2.796
Shims	f	**6.77**	8.92	9.60	11.78	11.61	0.939
	run time (s)	2	8	31	166	1,087	**8.856**

Table 2: Methods and scenarios with $volume = 1.5$

method	**Results**	**1**	**2**	**3**	**4**	**5**	**Normalized Speed-up**
Gurobi	f	4.756	8.84	11.94	16.21	**16.47**	0.852
	run time (s)	18	68	326	1,906	12,960	1.000
TS	f	8.97	11.44	12.92	15.20	15.39	0.936
	run time (s)	4	18	88	525	3,660	3.555
GRASP	f	9.36	12.34	13.63	16.18	15.44	0.981
	run time (s)	4	17	86	515	3,564	3.651
ACO	f	9.36	12.34	13.63	16.18	15.44	0.981
	run time (s)	5	19	90	539	3,720	3.492
NMO	f	9.36	12.33	13.60	16.16	15.43	0.979
	run time (s)	5	19	91	544	3,780	3.442
Shims	f	**9.39**	**12.42**	**13.72**	**16.28**	15.49	**0.986**
	run time (s)	3	11	48	265	1,720	**7.470**

Table 3: Methods and scenarios with *volume* = 2.0

method	**Results**	**1**	**2**	**3**	**4**	**5**	**Normalized Speed-up**
Gurobi	*f*	4.92	6.58	6.34	7.24	8.15	0.334
	run time (s)	29	106	493	2,833	19,380	1.000
TS	*f*	13.72	17.53	19.28	21.78	22.35	0.951
	run time (s)	5	19	93	556	3,900	4.992
GRASP	*f*	13.99	19.39	19.86	23.50	22.57	0.998
	run time (s)	6	21	97	571	3,960	4.906
ACO	*f*	13.99	19.39	19.86	23.50	22.57	0.998
	run time (s)	6	23	105	597	4,080	4.745
NMO	*f*	13.99	19.39	19.85	23.50	22.56	0.998
	run time (s)	9	27	120	681	4,620	4.184
Shims	*f*	**14.00**	**19.44**	**19.94**	**23.55**	**22.58**	**1.000**
	run time (s)	7	20	84	450	2,918	**6.570**

It is important to note that the best tour found in each test was seldom the tour with the lowest cost (the minimal cost tour without considering the CG deviation), but any of the first 25% of the best tours. This indicates that our strategy of enumerating and solving all tours was assertive, although it is possible to devise a more proactive strategy to discard most of the higher-cost tours and improve performance. This must be one of the next steps in this research.

If a posterior application of this method requires 8, 9, or more nodes, it is possible to use this 25% as a threshold to discard the higher-cost tours and keep the run time within an operational acceptable limit for the client.

In Tables 1, 2 and 3, *Gurobi* was the slowest method and *Shims* was the fastest. This was expected, as *Gurobi* is a MIP solver, which seeks an optimal solution within the established constraints, usually reaching its time limit on each node, while *Shims* is a fast heuristic. This way, *Gurobi* even demands prohibitive times (more than 5 hours in scenario 6 of Table 3), while the maximum time required by *Shims* did not reach one hour.

Considering the **Normalized** values, in Table 1 *Gurobi* was the best (0.997), followed closely by *Shims* (0.939). In Tables 2 and 3, *Shims* was the best (0.986 and 1.000). As expected, the performance of *Gurobi* drops sharply as the volume of shipments at the nodes increases: 0.852 in Table 2 and 0.334 in Table 3.

It is worth noting that the other heuristics always obtained **Normalized** values greater than or equal to 0.9, and were always faster than *Gurobi*.

It is not possible to state how far these results are from the optimal solutions, but we can report that all occupancy rates in terms of volume were above 97%, and in terms of weight above 79%, which denotes that the results achieved are good, considering that the CG deviation constraints keeps these values from going higher.

0.1.1. Proposed strategy assessment

Another important test is to compare the best tour score/cost ratio with the least expensive tour score/cost ratio. This is important to see how far our solution strategy is from the simplest form, just solving a TSP.

For this achievement, we ran *Shims* in 5 scenarios with 7 instances each, and 3 volume surpluses, collecting the averages of the best tour result and the least cost tour result.

Table 4: Shims best and least cost tour results

surplus	**Tour**	**1**	**2**	**3**	**4**	**5**	**Sum**	**Improvement**
1.2	Least cost	6.23	11.56	12.88	13.54	12.47	56.68	
	Best	7.96	13.04	14.30	15.56	15.28	66.14	16.7%
1.5	Least cost	8.44	16.61	18.18	19.68	17.64	80.55	
	Best	11.47	18.25	20.61	22.22	20.94	93.49	16.1%
2.0	Least cost	13.38	22.46	25.97	26.23	22.51	110.55	
	Best	17.93	25.59	28.61	31.28	29.09	132.50	19.9%
						Average improvement		17.6%

As the selection of the best tour, considering what was collected and delivered in each node, and considering

the cost increase due to CG deviation, was 17.6% better than the option for a simple TSP solution (*Least cost* results), this confirms that our solution strategy was adequate.