Artificial Intelligence: Assignment 1

Jury Andrea D'Onofrio

October 2022

1 Double Bridge

I implement the two functions `perturb_solution()` and `difference_cost()`. The steps are the following:

1. I pick four random nodes and I sort them.
2. I call `difference_cost()` to remove the current cost of the edges between the four nodes and add the new ones. Then, I return the difference between these two values.
3. I calculate the new cost as the current plus the gain returned before.
4. I connect the four random nodes to get the new sequence of vertices.
5. Finally, I return the new solution and the updated cost.

2 Instance "d198.tsp"

As expected, ILS-RW is always the worst one. As you can see from Figure 1, ILS-LSMC is the best in terms of the gap. However, in seed 333, both ILS-better and ILS-LSMC obtain the same result.
Moreover, the last column of Figure 1 shows that ILS-better performs a bit more iterations compared to the others. This means that it is a bit faster; although, for seed 0, the most rapid is ILS-LSMC since it carries out a greater number of iterations. Figure 2 confirms that ILS-LSMC is the best one.

I use a starting temperature value of 100 with the following update rule:

$$temperature = temperature * 0.93$$

I decided to choose $\alpha = 0.93$ otherwise I move towards 0 too fast.

In Figure 1 note that, for each seed, my best result is very close to the optimal solution. In particular, the optimal length is 15780.0 and I get:

- seed 0 $\rightarrow$ 15939.0
- seed 123 $\rightarrow$ 15900.0
- seed 333 $\rightarrow$ 16000.0

problem	optimal length	method	seed	tour length	gap	time to solve	calls Local Search
d198.tsp	15780.0	initialized with random, improved with ILS-better	0	15950.0	1.08	180.569	231
		initialized with random, improved with ILS-RW	0	16420.0	4.06	180.317	227
		initialized with random, improved with ILS-LSMC	0	15939.0	1.01	180.397	235
		initialized with random, improved with ILS-better	123	15967.0	1.19	180.796	230
		initialized with random, improved with ILS-RW	123	16503.0	4.58	180.398	228
		initialized with random, improved with ILS-LSMC	123	15900.0	0.76	180.259	227
		initialized with random, improved with ILS-better	333	16000.0	1.39	180.604	230
		initialized with random, improved with ILS-RW	333	16548.0	4.87	180.316	222
		initialized with random, improved with ILS-LSMC	333	16000.0	1.39	180.345	229

Figure 1: Table of results for d198.tsp

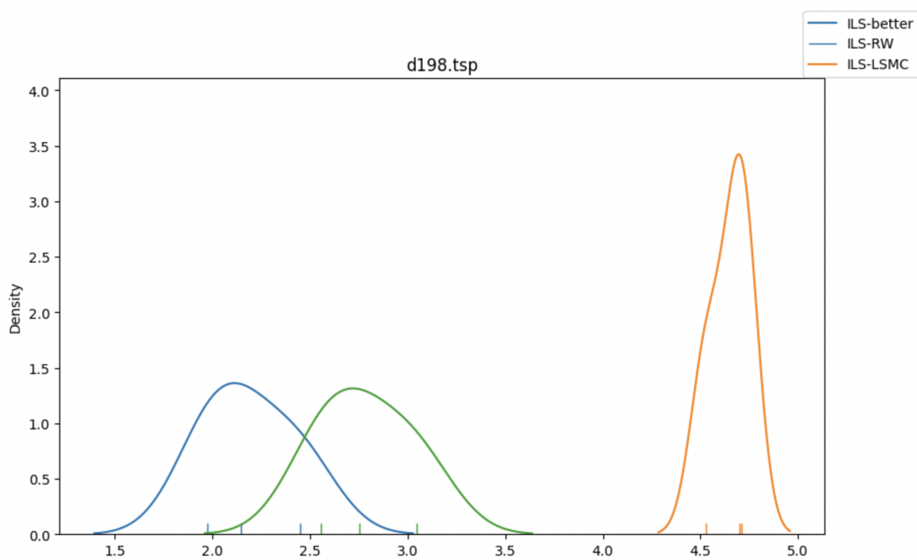

Figure 2: Plot of results for d198.tsp

Then, I run the script using `twoOpt_with_cl`. I expect to obtain bigger gaps compared to the optimal length but to be faster than before. Indeed, as you can compare between Figure 1 and Figure 3, it always performs more iterations, and it gets worse gaps. Surprisingly, Figure 3 shows that ILS-RW is now the best in terms of gap. This is also confirmed in Figure 4.

problem	optimal length	method	seed	tour length	gap	time to solve	calls Local Search
d198.tsp	15780.0	initialized with random, improved with ILS-better	0	19898.0	26.10	180.213	378
		initialized with random, improved with ILS-RW	0	17126.0	8.53	180.389	355
		initialized with random, improved with ILS-LSMC	0	18336.0	16.20	180.334	356
		initialized with random, improved with ILS-better	123	17068.0	8.16	180.559	342
		initialized with random, improved with ILS-RW	123	16983.0	7.62	180.483	339
		initialized with random, improved with ILS-LSMC	123	17800.0	12.80	180.086	339
		initialized with random, improved with ILS-better	333	21944.0	39.06	180.008	371
		initialized with random, improved with ILS-RW	333	18583.0	17.76	180.345	380
		initialized with random, improved with ILS-LSMC	333	21870.0	38.59	180.056	370

Figure 3: Table of results for d198 with cl.tsp

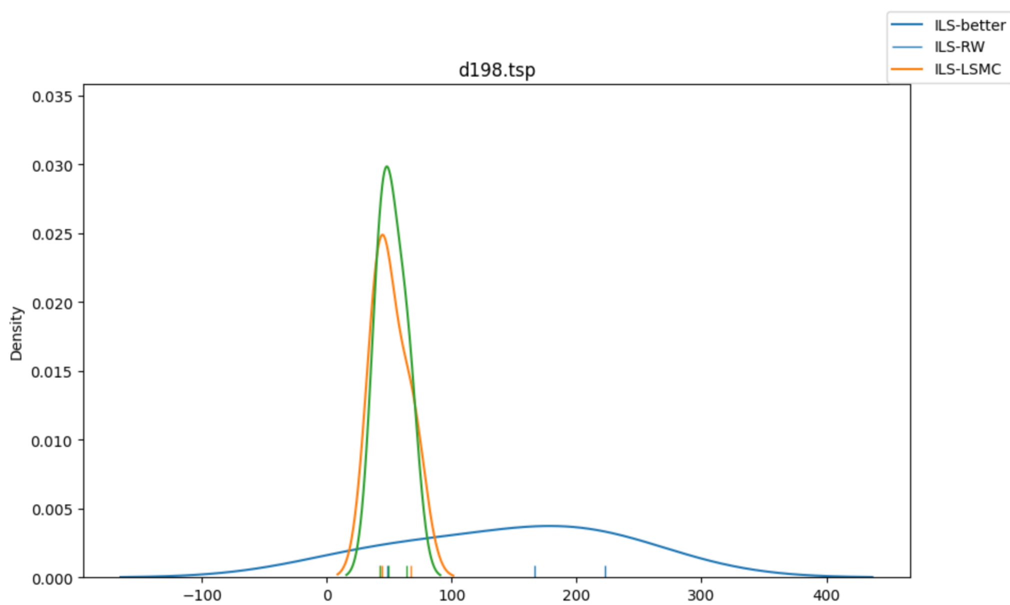

Figure 4: Plot of results for d198 with cl.tsp

3 Instance ”pr439.tsp”

As expected, ILS-RW is always the worst one. Figure 5 shows that ILS-LSMC is the best in terms of gap. However, in seed 123, ILS-better outperforms ILS-LSMC.
As the last column of Figure 5 shows, there is no method that is particularly faster than the others. This is also confirmed in Figure 6 which shows the density graph.

I use a starting temperature value of 100 with the following update rule:

$$temperature = temperature * 0.93$$

As before, I decided to choose $\alpha = 0.93$ otherwise I move towards 0 too fast.

In Figure 5 note that, for each seed, my best result is quite close to the optimal solution. In particular, the optimal length is 107217.0 and I get:

- seed 0 $\rightarrow$ 110064.0
- seed 123 $\rightarrow$ 110096.0
- seed 333 $\rightarrow$ 111790.0

problem	optimal length	method	seed	tour length	gap	time to solve	calls Local Search
pr439.tsp	107217.0	initialized with random, improved with ILS-better	0	112558.0	4.98	181.891	29
		initialized with random, improved with ILS-RW	0	119676.0	11.62	184.180	32
		initialized with random, improved with ILS-LSMC	0	110064.0	2.66	182.993	31
		initialized with random, improved with ILS-better	123	110096.0	2.69	182.996	31
		initialized with random, improved with ILS-RW	123	119165.0	11.14	182.198	31
		initialized with random, improved with ILS-LSMC	123	113244.0	5.62	182.160	32
		initialized with random, improved with ILS-better	333	112738.0	5.15	182.897	32
		initialized with random, improved with ILS-RW	333	117777.0	9.85	180.644	30
		initialized with random, improved with ILS-LSMC	333	111790.0	4.27	183.063	30

Figure 5: Table of results for pr439.tsp

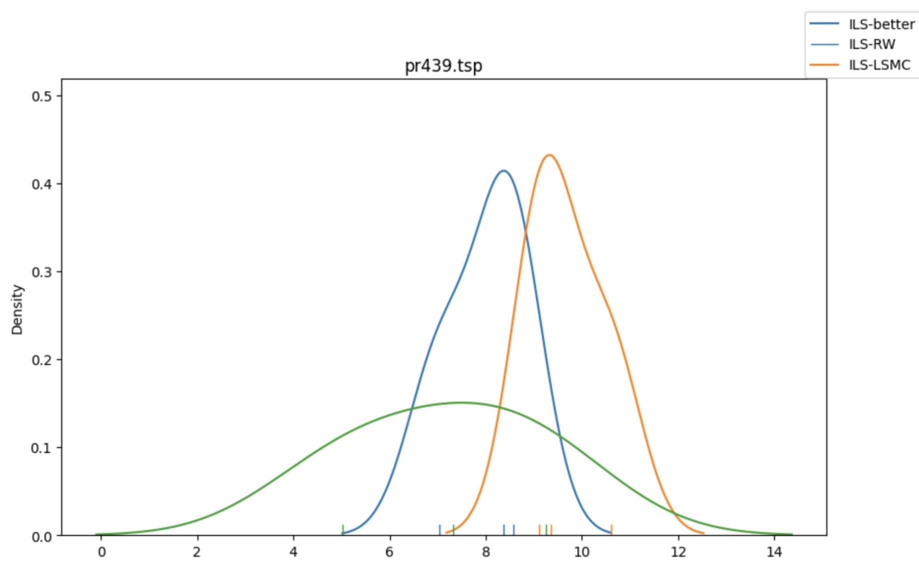

Figure 6: Plot of results for pr439.tsp

Then, I run the script using `twoOpt_with_cl`. Also here, I expect that Figure 7 shows bigger gaps compared to the optimal length but to be faster. Indeed, it performs more iterations and it gets worse gaps. Surprisingly, Figure 7 shows that ILS-RW is now the best in terms of gap. Furthermore, ILS-better is become the most rapid in two seeds out of three. Figure 8 show the corresponding results.

problem	optimal length	method	seed	tour length	gap	time to solve	calls Local Search
d198.tsp	15780.0	initialized with random, improved with ILS-better	0	19898.0	26.10	180.213	378
		initialized with random, improved with ILS-RW	0	17126.0	8.53	180.389	355
		initialized with random, improved with ILS-LSMC	0	18336.0	16.20	180.334	356
		initialized with random, improved with ILS-better	123	17068.0	8.16	180.559	342
		initialized with random, improved with ILS-RW	123	16983.0	7.62	180.483	339
		initialized with random, improved with ILS-LSMC	123	17800.0	12.80	180.086	339
		initialized with random, improved with ILS-better	333	21944.0	39.06	180.008	371
		initialized with random, improved with ILS-RW	333	18583.0	17.76	180.345	380
		initialized with random, improved with ILS-LSMC	333	21870.0	38.59	180.056	370

Figure 7: Table of results for pr439.tsp with cl

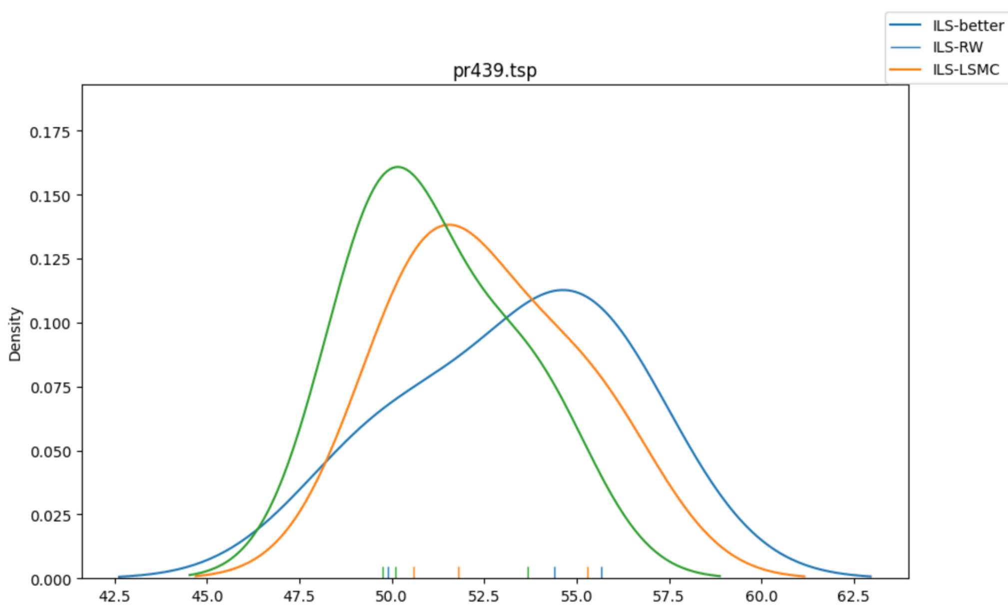

Figure 8: Plot of results for pr439.tsp with cl

4 Instance ”u1060.tsp”

I expect unreliable results since the instance is considerably large and the 3-minute constraint affects the performance. Indeed, the script executes only two iterations for each entry.
Figure 9 shows that there is no method that outperforms the others, although ILS-better gets the best gap in two out of three seeds. Furthermore, for seed 123, ILS-RW is very close to ILS-better. This, in my opinion, confirms that the size of the instance and the time constraint falsify the results. Finally, in Figure 10, you can see how the density graph for each method is roughly the same.

I use a starting temperature value of 100 with the following update rule:

$$temperature = temperature * 0.93$$

As before, I decided to choose $\alpha = 0.93$ otherwise I move towards 0 too fast.

In Figure 9 note that, for each seed, my best result is quite far to the optimal solution. In particular, the optimal length is 224094.0 and I get:

- seed 0 $\rightarrow$ 241976.0
- seed 123 $\rightarrow$ 246391.0
- seed 333 $\rightarrow$ 246205.0

problem	optimal length	method	seed	tour length	gap	time to solve	calls Local Search
u1060.tsp	224094.0	initialized with random, improved with ILS-better	0	241976.0	7.98	259.786	2
		initialized with random, improved with ILS-RW	0	243433.0	8.63	205.178	2
		initialized with random, improved with ILS-LSMC	0	244430.0	9.07	308.013	2
		initialized with random, improved with ILS-better	123	246391.0	9.95	411.774	2
		initialized with random, improved with ILS-RW	123	246412.0	9.96	401.805	2
		initialized with random, improved with ILS-LSMC	123	248169.0	10.74	387.152	2
		initialized with random, improved with ILS-better	333	246205.0	9.87	233.063	2
		initialized with random, improved with ILS-RW	333	248444.0	10.87	276.265	2
		initialized with random, improved with ILS-LSMC	333	247954.0	10.65	247.589	2

Figure 9: Table of results for u1060.tsp

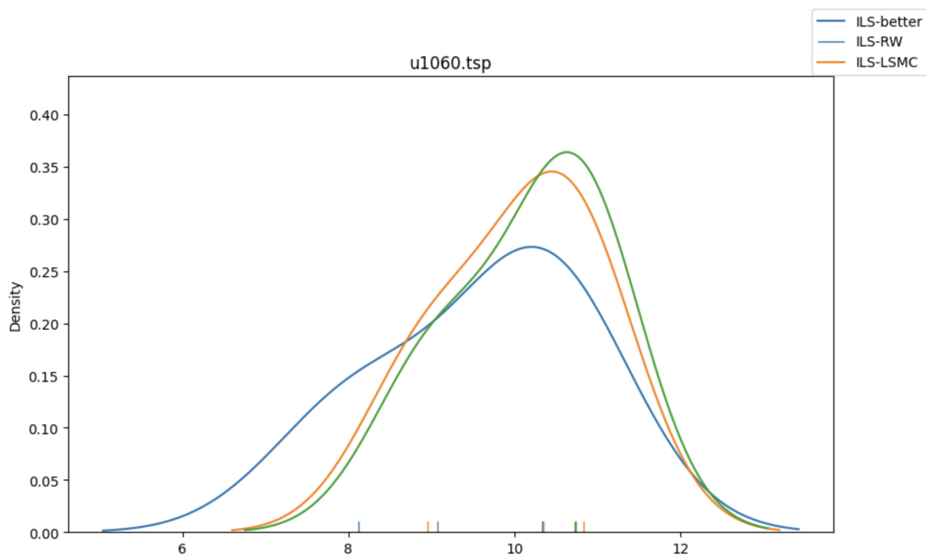

Figure 10: Plot of results for u1060.tsp

Then, I run the script using `twoOpt_with_cl`. Also here, I expect bigger gaps compared to the optimal length but to be faster. Indeed, it always performs more iterations and it gets worse gaps. Moreover, as you can see in Figure 11, ILS-RW outperforms the others. Finally, as the density graph of Figure 12 confirms the performance of ILS-better and ILS-LSMC work as ILS-RW.

problem	optimal length	method	seed	tour length	gap	time to solve	calls Local Search
u1060.tsp	224094.0	initialized with random, improved with ILS-better	0	335586.0	49.75	182.392	45
		initialized with random, improved with ILS-RW	0	289632.0	29.25	186.921	46
		initialized with random, improved with ILS-LSMC	0	318408.0	42.09	185.696	43
		initialized with random, improved with ILS-better	123	326185.0	45.56	181.158	42
		initialized with random, improved with ILS-RW	123	272483.0	21.59	182.910	40
		initialized with random, improved with ILS-LSMC	123	305590.0	36.37	180.946	40
		initialized with random, improved with ILS-better	333	309106.0	37.94	184.023	44
		initialized with random, improved with ILS-RW	333	318008.0	41.91	182.237	46
		initialized with random, improved with ILS-LSMC	333	313881.0	40.07	184.756	41

Figure 11: Table of results for u1060.tsp with cl

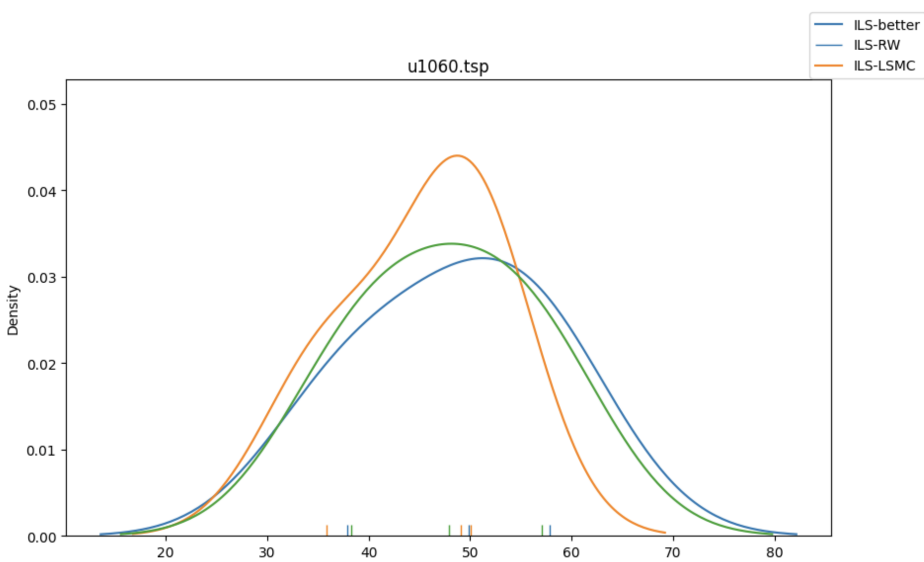

Figure 12: Plot of results for u1060.tsp with cl