Evolutionary Algorithms: Final report

Jon Irastorza

May 20, 2024

1 Final design of the evolutionary algorithm (target: 3.5 pages)

1.1 The three main features

1. Fitness sharing in the elimination phase.
2. 2-opt and 3-opt local search methods.
3. Adaptivity and self-adaptivity schemes.

1.2 The main loop

See Figure 1.

1.3 Representation

The chosen candidate solution representation is to use a **permutation** of numbers in the range $[0, N)$, stored in a NumPy array. The population is then represented by a NumPy matrix of size λ x $N + 3$. N being the number of cities to visit and λ the size of the population. The last 3 positions are used for self-adaptivity, regarding mutation probability and two k-s for k-opt local search and k-tournament selection, which are explained in the following sections. I chose this representation for multiple reasons:

1. It guarantees that every node will be visited exactly once.
2. Directly captures the order in which citites are visited.
3. Easy to understand, manipulate and compute operations.
4. Efficient memory usage.

To maintain uniqueness and prevent representations that map to the same solution due to cyclic symmetry, all individuals start in the 0 city. This helps in avoiding redundant solutions that differ only by a cyclic shift.

I also considered using an adjacency matrix, where each candidate solution is represented by a binary matrix where entry (i, j) is 1 if there is an edge between city i and city j, and 0 otherwise. However, this representation introduces redundancy, as it requires N^2 entries to represent N cities, leading to increased memory usage and computational complexity, especially for large instances of the TSP.

1.4 Initialization

The initial population of my genetic algorithm is a **combination** of candidate solutions generated completely at **random** and others generated by a **heuristic**.

My heuristic operates by considering the previous city ($i - 1$) and arranging the remaining unvisited cities based on their distances from city $i - 1$. The level of greediness is adjustable through the parameter e, which influences an exponential probability distribution. A higher e promotes a greedy approach, while a e closer to 0 encourages a more exploratory strategy, for more detail see Figure 2. Cities at an infinite distance (indicating no direct path) are immediately excluded. If no candidate cities remain, the solution is discarded and the e is decreased by 0.1% to relax the greediness of the heuristic and prevent generating a non valid candidate again. When generating a completely random solution, a uniform distribution is used, meaning all cities at a finite distance from i-1 and not yet visited have an equal probability of being chosen.

The **flexibility** provided by choosing the parameter e and adjusting the proportion of random candidate solutions, which by default is 80%, ensures that the initial population is **sophisticated** while maintaining **sufficient diversity**. By default the value of e is $3 \cdot N$, so the bigger the problem, the more greedy heuristic solutions the initial population has.

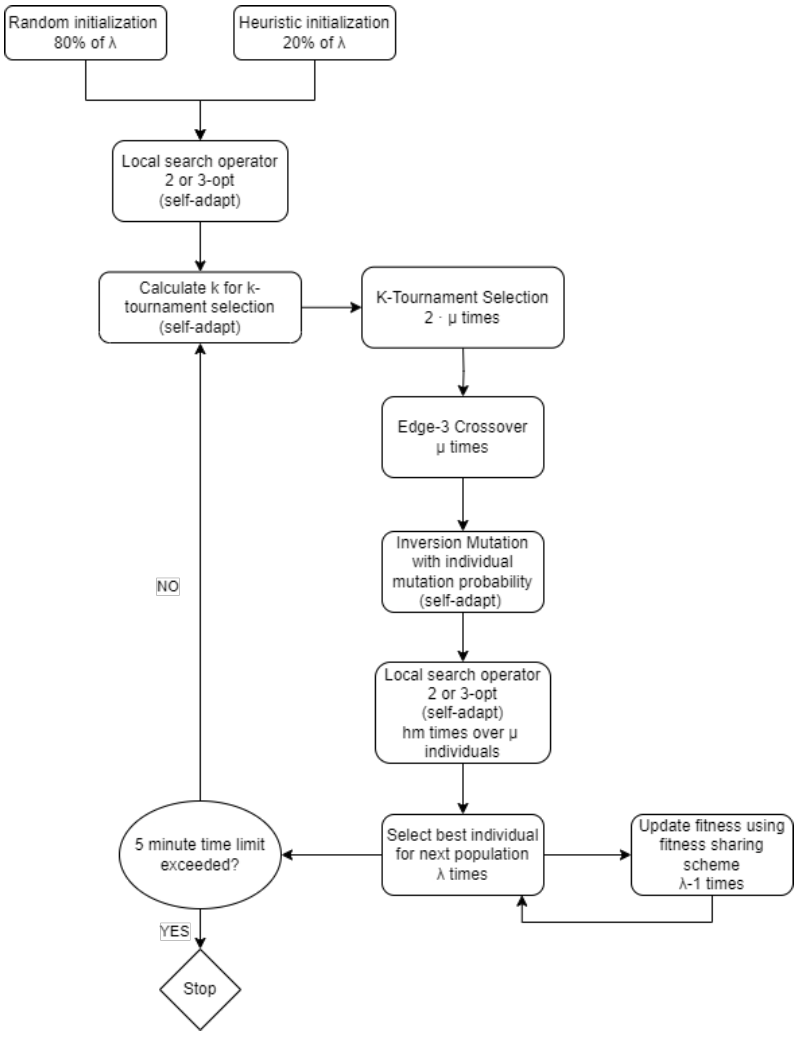
Random initialization 80% of λ
Heuristic initialization 20% of λ
Local search operator 2 or 3-opt (self-adapt)
Calculate k for k-tournament selection (self-adapt)
K-Tournament Selection 2 · μ times
Edge-3 Crossover μ times
Inversion Mutation with individual mutation probability (self-adapt)
Local search operator 2 or 3-opt (self-adapt) hm times over μ individuals
Select best individual for next population λ times
Update fitness using fitness sharing scheme λ-1 times
5 minute time limit exceeded?
NO
YES
Stop

Figure 1: Main loop diagram.

In addition, the last three positions of each candidate solution, which are utilized for **self-adaptivity**, are initialized **randomly**. The values range between 0.01 and 0.1 for the individual mutation probability, 2 or 3 for the k-opt local search value, and between $N \cdot 0.4$ and $N \cdot 0.8$ for the k-tournament selection. After the population is initialized, a **k-opt local search** is performed, where the value of k for each individual dictates the specific k value for the local search, which can be either 2 or 3.

Initially, the population was **completely initialized at random**. While this approach provides significant **diversity**, for problems with a high number of cities, it becomes **challenging** for the genetic algorithm to **converge** to optimal solutions within a 5-minute time frame, due to the high-dimensions of the problem. Therefore, there is interest in starting with a more **sophisticated initial population**.

The decision to maintain a population size of 200 allows for a **higher number of iterations**. Initially, the size was N or $N/2$. However, this approach was ultimately discarded. For larger problems, these numbers were deemed excessive, leading to a significantly **reduced number of iterations**.

1.5 Selection operators

I have implemented two selection algorithms: **k-tournament selection** and **exponentially decaying selection with a geometric decay schedule**. However, it is important to note that in the latter, the selection pressure on the best individuals in the population increases over iterations. Depending on the dimensionality of the problem and the chosen hyper parameter configuration, the total number of iterations can vary. As a result, determining the coefficient for the decay in the geometric schedule can be **challenging**.

For this reason, I **chose the k-tournament selection**, where the parameter to be adjusted is k—that is, the number of individuals randomly chosen from the population, followed by selecting the best among them. Importantly, the value of k is **automatically adjusted** by the genetic algorithm, through **self-adaptivity**. Each individual in the population has a specific k value, which after the elimination is used to calculate the **median** of the k values across all individuals in the population, thus determining the effective k for the tournament selection.

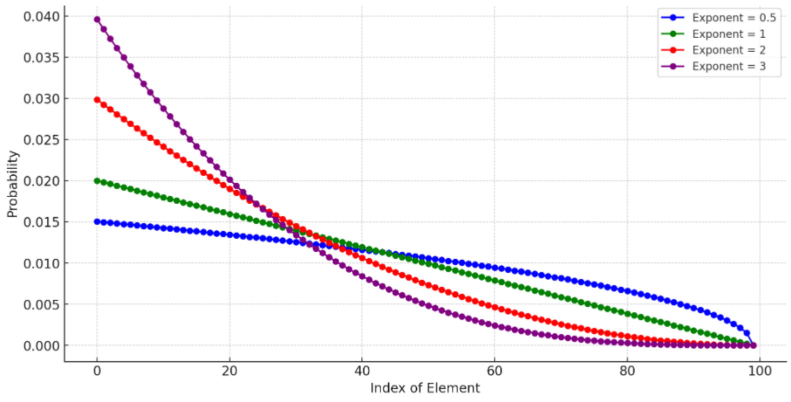

Figure 2: Probability curves for different exponents 'e'

1.6 Mutation operators

I have implemented two mutation operators, **swap mutation**, and **inversion mutation**. Ultimately, I have **chosen the inversion mutation**. This choice is motivated by the fact that it introduces **higher variation**. Not only does it alter two connections within the selected segment, but it also reverses the segment at both ends. This feature is particularly interesting when the distance matrix is asymmetric. When a segment is inverted, it has the potential to reduce the length of the path, offering a valuable mechanism for exploring and discovering novel solutions within the search space.

The mutation probability for each individual is determined within the individual's genome, as previously explained in Section 3.3. This value is adjusted through **self-adaptivity**, where an individual's mutation probability is set to a randomly weighted average of the mutation probabilities of its parents during the recombination phase. This self-adaptive mechanism ensures that the mutation rate **dynamically adapts** to the characteristics of the evolving population.

$$mprob_{\text{offspring}} = mprob_{\text{parent1}} \times X + mprob_{\text{parent2}} \times (1 - X)$$
$$\text{where} \quad X \sim \text{Uniform}[0, 1)$$

1.7 Recombination operators

I implemented two recombination operators: **Partially Mapped Crossover (PMX)** and **Edge Crossover (Edge-3)**. Initially, PMX was selected during the group phase for its **simplicity**. However, while with this method, a good amount of links present in the offspring are present in one or more of the parents, not all common information present in both parents may be reflected in the offspring, which **contradicts the expectation** that if a feature is present in both parents, it should be inherited by the offspring. Edge-3 crossover solves this, since it is based on the idea that offspring should be created as far as possible using only edges that are present in (one of) the parents, giving priority to those that appear in both parents.

Upon integrating Edge Crossover into the algorithm, notable improvements were observed in the results. Additionally, I successfully addressed the challenge of **increased computational cost** associated with Edge Crossover, which is caused by the identification of common edges, involving more computations than simpler operations like PMX. However, **Numba** library was used to improve the execution time, trying to mitigate the higher computational cost of this operator.

It is important to note that when there is minimal **overlap** between parents, **PMX tends to perform well**, maintaining gene order. On the other hand, **Edge Crossover**, while influenced by shared edges, may encounter **challenges** in generating meaningful offspring when parents exhibit minimal common structure.

The **PMX** operator can be configured with **parameters** to set a maximum or minimum size for the crossover slice. When the slice size so small or big in comparison to N, the offspring will contain much more information about one of the parents, which leads to a lack of diversity. On the other hand, I **have not utilized parameters** for the **edge-3** crossover.

While the idea of using a recombination operator with arity strictly greater than 2 crossed my mind, ultimately, given the **successful performance** of the algorithm with the Edge-3 Crossover operator (which has an arity of 2), I decided to **stick with it**.

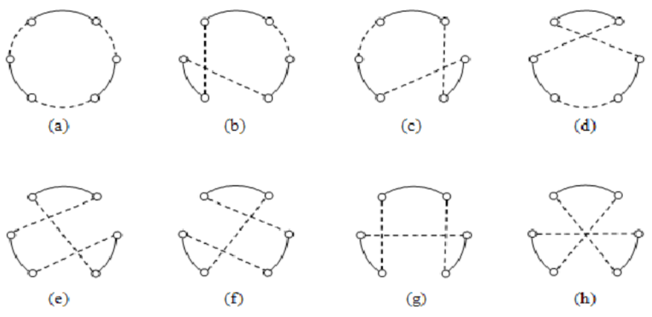

Figure 3: All possible 3-opt reconnection cases

1.8 Elimination operators

During the group phase we implemented the $\mu+\lambda$ elimination method, which I maintained for a while; however, I observed a noticeable **lack of diversity** in my genetic algorithm. The issue arose from consistently selecting the λ best individuals from $\mu+\lambda$ join population, leading to an **increasing similarity** among the individuals in the population. Therefore, after the discussion session of Module 3, I decided to implement a **diversity promotion scheme** specifically for the elimination phase, as explained in Section 3.10. This basically prevents the algorithm from being purely exploitative, which is crucial to **avoid premature convergence** and enhance the algorithm's ability **explore the solution space**.

1.9 Local search operators

I have implemented both 2**-opt** and 3**-opt** local search. Instead of performing a single local search on each individual, the number of searches can be specified using the parameter hm.

The type of local search, either 2-opt or 3-opt, for each individual is determined within the individual's genome, as previously explained in Section 3.3. This value is adjusted through **self-adaptivity**, where an individual's k is set by randomly choosing between the k of one of its parents.

$$k_{\text{opt}_{\text{offspring}}} = \begin{cases} k_{\text{opt}_{\text{parent1}}}, & \text{with probability } 0.5 \\ k_{\text{opt}_{\text{parent2}}}, & \text{with probability } 0.5 \end{cases}$$

When performing 2-opt, two edges are removed, leaving only **one way to reconnect the cities**. However, with 3-opt, where three edges are removed, there are 7 **ways to reconnect them**, resulting in a **larger neighborhood**, see Figure 3. To prevent individuals from leaning towards performing a 3-opt search hm times due to the difference in neighborhood sizes, those individuals on which a 2-opt search is conducted will actually undergo a $7 \cdot hm$ search. This is done to ensure that all individuals are on an **equal footing**, and the self-adaptivity genuinely tends towards the best k-opt value.

I have implemented an **adaptivity** scheme for the parameter hm. When the algorithm observes that its best solution has remained stagnant for 5 consecutive iterations, there is an **increment** of 10% in the value of hm. This adaptive strategy aims to **intensify the exploitation** phase of the algorithm, **countering stagnation** that may arise from a **rigorous diversity promotion strategy**, explained in Section 3.10. The dynamic adjustment of hm allows the algorithm to **adapt** to the changing characteristics of the search space.

The integration of local search operations into my genetic algorithm approach has proven **instrumental**. This tailored combination of global exploration and localized refinement enhances the algorithm's ability to converge towards **more optimal solutions**, since it helps to **guide the search** towards more promising areas.

1.10 Diversity promotion mechanisms

I have implemented a **shared fitness scheme in the elimination phase** of my algorithm. This scheme involves **adjusting the fitness** values of individuals based on their proximity or **similarity** within the already **selected individuals** for the next population. Each iteration, the individual with the lowest calculated fitness, is chosen to be promoted to the next iteration.

Initially, all the individuals have their original fitness $f(x)$. However, each time an individual y is promoted to the next iteration, the fitness values of the individuals from the added population $\mu+\lambda$, whose distance $d(x,y) \leq \sigma$,

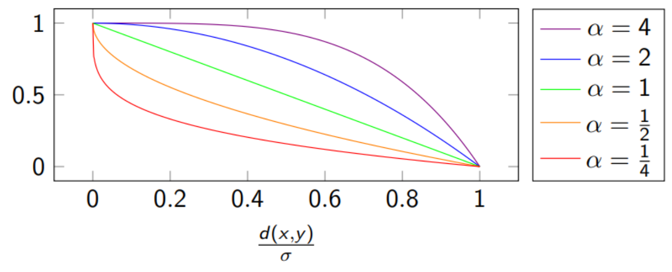

Figure 4: Influence of α over $1 + \beta$

$\sigma \geq 0$ receive a penalty, where their fitness is increased:

$$f'(x) = f(x) \underbrace{\left(1 + \left(\frac{d(x,s)}{\sigma}\right)^{\alpha}\right)}_{1+\beta}$$

Therefore, there are **two parameters** to determine: σ and α. The influence of α can be observed in the Figure 4, where larger values of α apply a larger penalty for each additional individual within a distance σ of y.

Given two candidate solutions, $s1$ and $s2$, the distance is calculated based on the number of edges they share n_s

$$d(s_1, s_2) = 1 - \frac{n_s}{n} \quad \text{where } n \text{ is the total number of pairs.}$$

In this way, when two individuals are **exactly identical**, their distance will be **0**, and when two individuals **do not share any edges**, their distance will be **1**.

This scheme significantly **enhances the performance** of my evolutionary algorithm by explicitly **preventing the promotion of similar individuals** to the next iteration. The mechanism degrades the fitness value of such individuals each time they are selected. Consequently, their likelihood of being chosen diminishes, strategically **promoting the diversity of the population**. This approach not only **safeguards** against the **premature convergence** of the algorithm but also promotes the **exploration** of the solution space by favoring the inclusion of **genetically diverse** individuals in the following iterations.

In order to prevent stagnation resulting from strict parameters, an **adaptive** mechanism has been implemented. If the best fitness does not improve over 5 consecutive iterations, the parameter σ is reduced by 10%, provided that it does not fall below a specified threshold ($\sigma_{threshold}$). This adaptive adjustment aims to prevent stagnation by dynamically refining the exploration-exploitation balance through the modulation of the σ parameter.

1.11 Stopping criterion

I have **not incorporated** any stopping criteria in my evolutionary algorithm, given that the **time limit** is set to 5 minutes. Although there are instances where the algorithm might stagnate before this time threshold, there remains the **possibility of further improvement** within the remaining time. This potential improvement is attributed to the effectiveness of the crossover, mutation, and local search operators, which continue to operate and may refine solutions within the specified time frame.

1.12 Parameter selection

The **population size** (λ) and **offspring size** (μ) have been adjusted, referring in part to the study conducted during the **group phase** using **Optuna**, a framework for hyper-parameter optimization. In this study, it was observed that employing a reduced λ and a similar μ proved beneficial, making our algorithm capable of conducting a more **exhaustive search** of the solution space while consuming **reduced memory** resources and requiring **less time per iteration**. Despite the adjustments have been made to the algorithm, as expressed throughout the entire report, it remains advantageous to adhere to these ideas. This is particularly true for larger problem instances, where the computational cost of calculating distances between individuals and performing crossover operations **increases significantly**.

Regarding the **fitness sharing** parameters, the **initial distance threshold** (σ_{init}) has been defined to 0.9, and this is because, as long as the algorithm does not stagnate, it is **interesting and vital** to try to maintain the **greatest** degree of **diversity**. Therefore, a value close to 1, such as 0.9, ensures this. There is also a **minimum distance threshold** ($\sigma_{threshold}$), which is the lower limit that σ can reach. It has been adjusted to 0.5 because, in

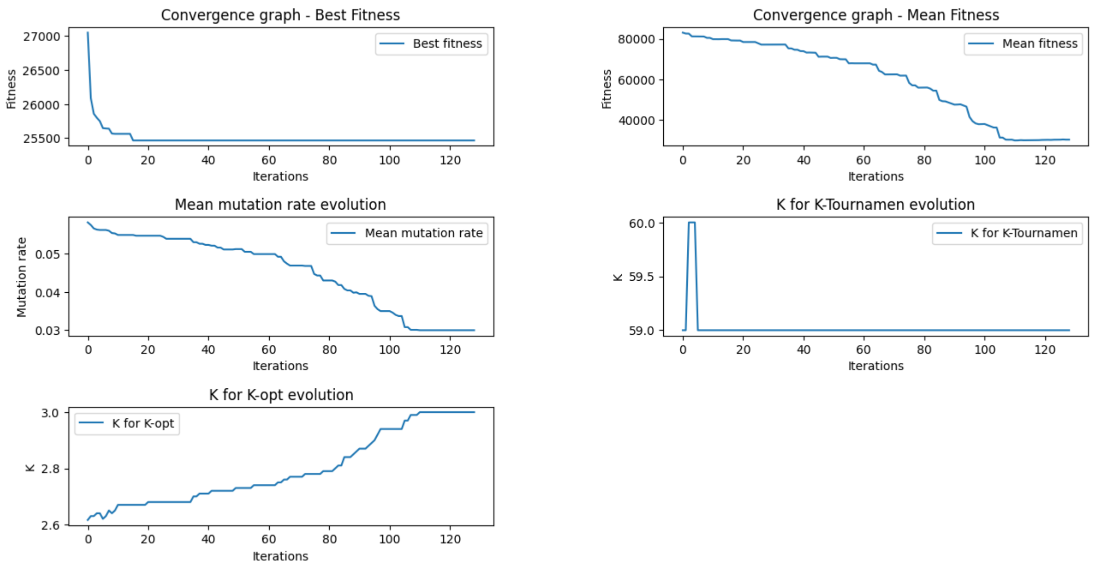

Figure 5: Curves tour50

addition to obtaining individual solutions of high quality, we aim to have a **diverse set of solutions** in the final population. This adjustment ensures that, at least, solutions are at a distance of 0.5, meaning that at least 50% of the connections between cities are different. The second parameter needed, alpha (α) has been adjusted after trying with all the values in the Figure 4 to $\frac{1}{2}$. This value has yielded positive results since, by setting a high value for σ, it is advantageous to **penalize** individuals that are **more distant** to a **lesser extent**.

The initial number of **local searches** to be conducted per individual (hm_{init}) has been determined through a **hyperparameter search**, ranging from 100 to 1000. Ultimately, the most favorable results were achieved with **500** local searches. The **shared fitness scheme**, as explained in Section 3.10, **prevents** the local search from easily **falling into local optima**.

2 Numerical experiments (target: 1.5 pages)

2.1 Metadata

The values of the parameters have been presented in the previous sections; however, for a quick reminder:

λ: 200; μ: 200; e: $3 \cdot N$; $random\%$: 0.8; σ_{init}: 0.9; $\sigma_{threshold}$: 0.5; α: $\frac{1}{2}$; hm_{init}: 500.

Specifications:

- **CPU:** Intel Core i7-10750H CPU @ 2.60GHz (12 CPUs)
- **RAM:** 16GB
- **Python version:** 3.10

2.2 tour50.csv

In this case, as you can see in Figure 5, the obtained results manage to **surpass the benchmark** set by the greedy heuristic (27723). However, upon observing the convergence mean fitness curve, it is noticeable to have a **stepped pattern**, which may be attributed to the adjustment made to the sigma value (which relaxes the diversity promotion scheme), as it decreases when the best fitness stagnates. The best fitness trend indicates a **stagnation** around iteration 20 with no further improvement, suggesting that it has likely reached or is **close to the global optimum**. It is intriguing how, in this instance, the **mean mutation rate decreases** with iterations while the **k** value for the **tournament** displays an initial spike in the early iterations and then it **stabilizes** and converges to the value of 56. Additionally, the **k** value for **local search** converges to 3, meaning that **all individuals** perform a search in the **3-opt neighborhood**.

Regarding the diversity of the final population, 95% of the individuals have distance between 0.6-0.9, which means that they only share 10-40% of the edges, denoting a **good amount of diversity**.

The best fitness obtained is **25434.91**.

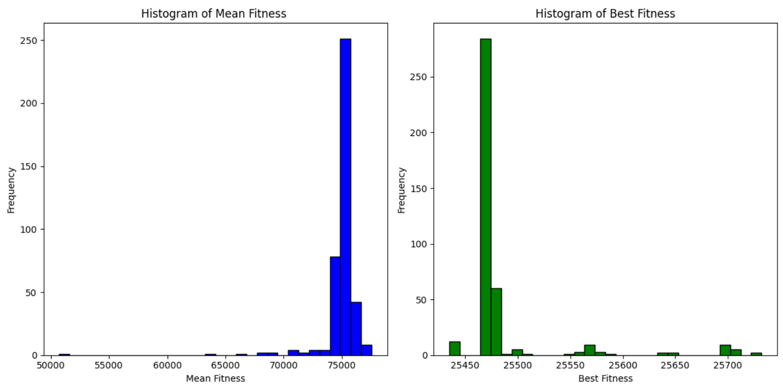

Figure 6: Histogram 500 runs tour50

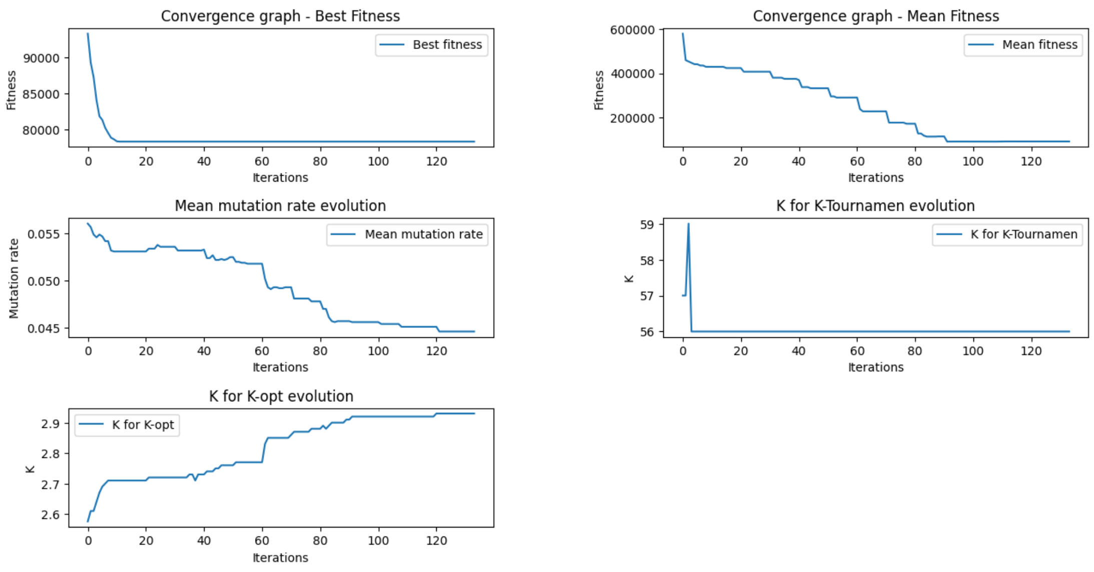

Figure 7: Curves tour100

In Figure 6, the final mean fitness and final best fitness of 500 runs over the tour50 are shown. The mean best fitness is 25484.78 with a std of 50.82 and the mean mean fitness is 74947.38 with a std of 1760.36. The best fitness and mean fitness are different, which means a **good diversity** is maintained over the population. Furthermore, the results present low standard deviations, meaning that the algorithm is **consistent** and **robust**. Due to the results, I would say that the **global optima** for this problem is **close**.

2.3 tour100.csv

In this case, as you can see in Figure 7, the obtained results manage to **surpass the benchmark** set by the greedy heuristic (90851). It can be observed that the **best fitness** curve **declines more steeply** than the mean fitness curve and **stagnates** around iteration 15, suggesting that it has likely reached or is **close to the global optimum**. The **stepped pattern** observed in the mean fitness curve can be attributed to the adjustment made to the sigma value (which relaxes the diversity promotion scheme), as it decreases when the best fitness stagnates.. In this case, the **mean mutation rate decreases** with iterations while the **k** value for the **tournament** spikes and then **converges** to 56. Additionally, the **k** value for **local search** tends to 3, meaning that **most individuals** perform a search in the **3-opt** neighborhood, which in this case **may work better than the 2-opt**.

Regarding the diversity of the final population, 93% of the individuals have distance between 0.5 and 0.9, which means that they only share 10-50% of the edges, which denotes a **good amount of diversity**.

The best fitness obtained is **78116.12**.

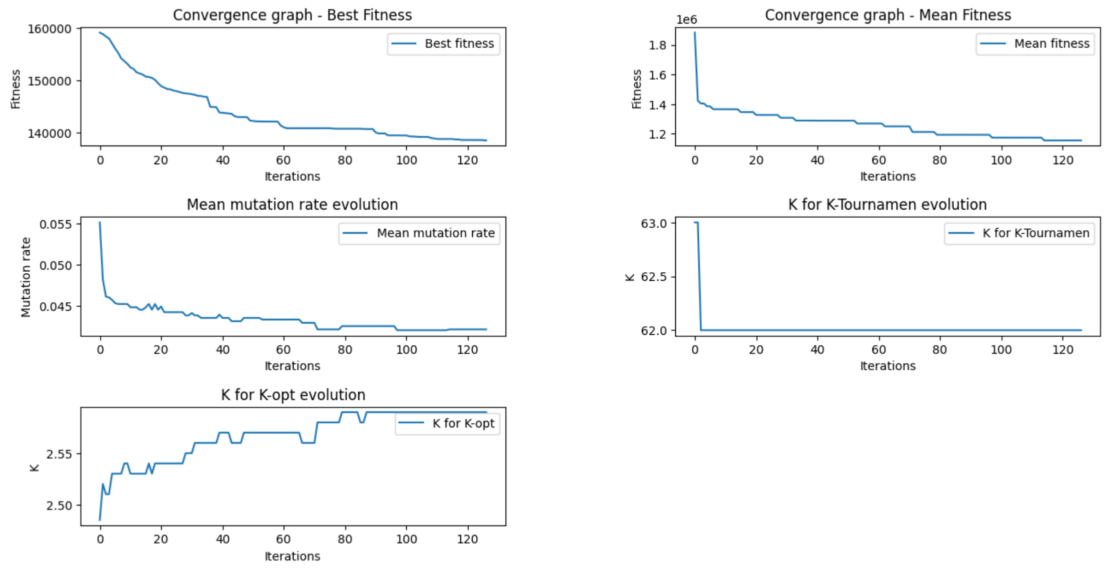

Figure 8: Curves tour500

2.4 tour500.csv

In this case, as you can see in Figure 8, the obtained results manage to **surpass the benchmark** set by the greedy heuristic (157034). However, in this case, **unlike with the smaller instance**, both the best and mean fitness **improve at a similar pace**. It is even noticeable that if the genetic algorithm was allowed to run for a longer duration, **further improvement** could have been achieved, suggesting that it may **not be as close to the global optimum**. Similar to the previous instances, the **mean mutation rate decreases** with iterations, and the **k** for **tournament** selection **stabilizes** around 62, a **slightly higher** value than in the previous instances, possibly due to a **larger search space**. Additionally, the average k value for **k-opt** remains around 2.5, indicating that roughly **half** of the individuals perform a **2-opt** search, and the other half perform a **3-opt** search. In this case, 3-opt may **not be as beneficial** as for smaller instances.

Regarding the diversity of the final population, 23% of the individuals have distance 1 and 56% of the individuals have $0.9 <$ distance < 1, which means that they share 0% and 10% of edges respectively with other solutions in the population. Therefore, the population exhibits a **high level of diversity**.

The best fitness obtained is **138613.96**.

2.5 tour1000.csv

In this case, as you can see in Figure 9, the obtained results manage to **surpass the benchmark** set by the greedy heuristic (195848). In this case, as with the tour500, both the best and mean fitness **improve at a similar pace**. It is even noticeable that if the genetic algorithm were allowed to run for a longer duration, **further improvement** could have been achieved, suggesting that it **may not be as close to the global optimum**. **Contrary** to the previous instances, the **mean mutation rate increases** with iterations, and the **k** for **tournament selection stabilizes** around 61, a **slightly higher** value than in the smaller instances, possibly due to a **larger search space**. Additionally, the average k value for **k-opt** remains around 2.5, indicating that roughly **half** of the individuals perform a **2-opt** search, and the other half perform a **3-opt** search. In this case, 3-opt **may not be as beneficial** as for smaller instances.

Regarding the diversity of the final population, 24% of the individuals have distance 1 and 63% of the individuals have $0.9 <$ distance < 1 this means that they share 0% and 10% of edges respectively with other solutions in the population. Therefore, the population exhibits a **high level of diversity**.

The best fitness obtained is **188061.11**.

3 Critical reflection (target: 0.75 pages)

1. **Strength 1:** Utilizing genetic algorithms offers the advantage that, with proper configuration promoting population **diversity** while attempting to guide the population towards more promising regions of the search space, it provides us with a **set** of **potentially useful solutions** to address the problem, rather than a

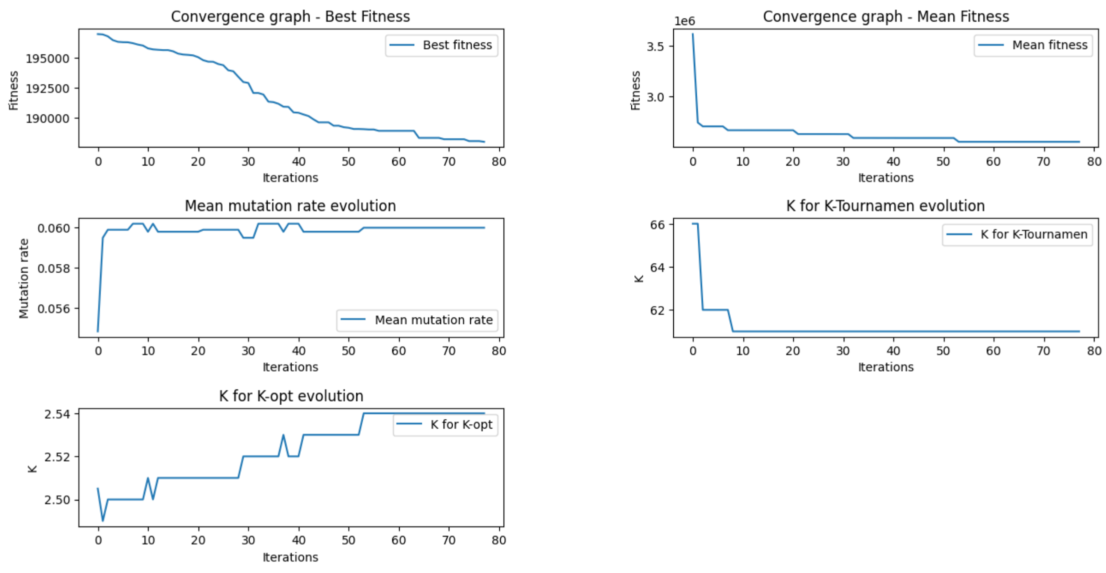

Figure 9: Curves tour1000

single solution, as is the case with other methods.

2. **Strength 2:** Genetic algorithms are inherently **adaptable** to different types of optimization problems. They do not rely on specific mathematical properties of the objective function, making them suitable for a wide range of applications. Whether the problem involves continuous or discrete variables, genetic algorithms can be adapted to handle the specific requirements of the task at hand.
3. **Strength 3:** Genetic algorithms demonstrate particular efficacy in scenarios characterized by the presence of multiple, often conflicting objectives. Traditional optimization methodologies typically center around the singular pursuit of a primary objective, such as maximizing profits or minimizing costs. In contrast, the application of genetic algorithms to **multi-objective optimization** involves the simultaneous consideration and pursuit of several objectives.

1. **Weakness 1:** Genetic algorithms are **susceptible to variations** in their performance based on the chosen **parameter values**. Parameters such as mutation rates or population sizes play pivotal roles in shaping the algorithm's behavior. The effectiveness of the genetic algorithm is contingent on finding the right balance among these parameters. However, identifying the optimal combination often requires thorough experimentation, and suboptimal parameter settings may lead to **diminished performance** or **premature convergence**.
2. **Weakness 2:** Genetic algorithms, while effective, can be **computationally demanding**, especially when dealing with complex problems or large solution spaces. The need to evaluate multiple potential solutions during the optimization process can lead to extended computation times.
3. **Weakness 3:** Genetic algorithms heavily depend on random processes, such as random initialization, more or less stochastic selection or mutation operators. This reliance introduces an element of **unpredictability** into the optimization journey. While randomness aids in exploring diverse solution spaces and avoiding getting stuck in local optima, it also means that the outcomes of the algorithm can vary between different runs.

As I delved into tackling the TSP, I **realized to be wrong** to believe that having a single dominant individual could be beneficial, but incorporating fitness sharing techniques shed light on the vital importance of maintaining diversity in the population. This realization led me to **understand** that genetic **variability** is crucial for the effective evolution of the genetic algorithm.

Another intriguing aspect was the successful **merging** of **genetic algorithms** with **local search techniques**. Even though I had previously studied these concepts separately, the ability to integrate them surprised me. The versatile combination of these strategies not only enhanced search efficiency but also showcased the adaptability and power of evolutionary approaches in solving complex problems like the Travel Salesman Problem.

This project has introduced me to **Numba**, which I have found incredibly effective in accelerating code execution, and may be a library that I tend to use more in other projects.

4 Best tours

tsp50: [0 26 5 8 46 35 16 4 15 47 42 11 3 19 38 10 13 7 41 14 1 43 30 9 48 6 22 49 20 18 24 45 2 31 29 37 39 25 12 27 28 36 23 44 17 40 33 34 21 32]

tsp100: [0 21 4 19 73 78 40 95 65 85 68 57 17 6 75 41 42 1 71 30 92 49 91 63 38 37 34 98 5 96 84 7 70 3 8 99 15 9 44 56 51 87 72 43 89 14 29 32 59 33 50 83 82 97 46 74 64 11 67 61 20 12 35 66 28 60 62 36 86 77 94 55 31 69 90 81 24 45 80 18 48 10 39 93 88 13 26 2 79 25 22 23 52 16 47 54 53 58 27 76]

tsp500: [0 148 392 120 310 450 215 77 46 71 11 138 246 174 476 78 69 185 263 31 351 422 62 28 197 370 384 130 458 308 234 369 96 236 48 38 497 251 214 166 301 322 328 390 94 414 169 460 202 68 150 225 336 149 248 15 55 200 2 335 50 378 466 444 483 324 417 146 74 408 239 312 403 104 37 463 436 442 108 452 232 165 459 477 92 256 472 299 420 475 359 143 393 425 229 133 86 43 366 183 495 496 87 350 330 119 29 1 123 394 469 182 91 283 307 365 321 245 244 121 303 402 34 170 186 6 243 375 284 339 158 353 9 206 227 47 90 53 220 428 171 160 176 474 45 211 274 438 440 316 224 196 413 269 320 332 493 401 311 65 152 490 387 261 226 142 313 341 338 281 10 276 404 400 52 41 376 221 254 3 367 259 354 213 124 231 126 262 348 49 304 136 432 237 249 360 264 272 242 93 449 106 132 137 318 380 407 489 66 296 238 162 277 293 491 230 317 292 54 89 290 437 358 199 271 223 433 167 473 486 147 391 79 192 286 448 306 468 349 76 102 193 342 42 161 385 156 25 20 453 410 326 478 372 190 498 117 88 439 180 139 434 255 298 35 60 22 51 59 397 178 421 315 275 418 265 189 480 302 209 210 451 122 377 343 388 487 107 33 115 260 216 80 109 431 289 7 347 175 56 499 58 112 406 295 285 398 98 381 57 141 157 184 191 267 279 416 181 386 85 208 327 110 471 412 159 446 105 435 368 204 345 72 257 266 163 297 415 187 173 379 426 357 118 240 131 355 99 340 374 18 26 125 470 481 155 39 114 364 13 461 195 111 424 488 70 455 81 4 319 329 219 16 168 140 177 482 382 346 64 27 253 153 151 212 32 250 201 218 24 395 309 205 127 396 172 14 194 97 344 5 282 40 12 419 462 198 278 300 362 268 492 485 36 467 429 188 427 21 17 207 100 334 44 252 144 128 63 129 154 383 61 373 145 135 464 361 67 164 445 280 273 405 233 399 465 441 84 411 305 179 337 75 314 291 288 270 371 287 331 456 457 494 323 241 356 333 228 95 479 363 247 423 113 82 352 235 8 258 454 389 19 116 73 101 430 134 294 203 217 30 325 409 484 222 443 23 103 447 83]

tsp1000: [0 255 105 416 754 457 371 894 986 329 207 430 499 35 81 316 349 743 437 397 241 181 25 123 498 139 882 296 115 113 857 347 753 137 474 257 188 900 644 31 988 114 17 120 539 90 38 955 517 445 526 807 674 252 519 780 34 479 475 381 423 509 889 773 272 476 713 313 177 932 218 145 507 254 7 60 545 632 899 864 392 502 211 341 134 406 677 990 602 292 421 690 523 786 943 462 758 552 841 521 391 804 683 725 768 595 132 159 365 234 66 366 696 776 639 634 239 974 987 614 46 896 204 314 537 831 738 844 162 486 650 852 881 190 631 684 615 699 281 73 953 512 351 273 173 908 622 991 477 80 695 756 157 550 664 611 324 623 367 723 156 956 278 231 150 348 685 675 306 508 976 141 995 40 321 944 360 681 19 63 368 148 560 47 604 327 904 176 345 730 856 212 777 146 192 575 440 179 668 49 629 817 557 766 816 985 335 688 41 962 866 489 178 704 625 333 464 830 409 708 645 515 230 764 534 308 846 868 919 763 388 202 796 280 164 122 229 9 301 267 924 467 442 94 578 898 103 978 824 911 183 525 171 897 133 820 873 670 344 546 251 945 57 45 842 52 187 973 540 839 403 663 792 374 808 532 127 656 303 887 850 718 589 364 404 98 923 323 380 712 649 128 463 744 643 628 193 82 779 520 720 125 299 262 165 914 531 567 553 140 485 726 518 484 963 468 957 542 983 220 248 393 227 424 250 893 431 184 992 478 755 198 795 43 166 263 441 307 765 244 941 414 453 576 340 101 222 707 343 304 228 596 480 242 245 798 770 126 561 163 221 259 287 386 325 511 236 402 858 870 759 940 569 965 279 637 832 934 342 89 586 129 772 444 268 210 801 194 447 11 891 219 247 77 835 398 420 514 851 78 74 454 106 180 926 93 555 969 573 789 213 405 660 516 731 715 929 160 682 751 450 271 837 373 436 613 285 288 427 996 931 997 65 588 491 385 182 967 821 653 572 579 487 843 196 42 85 800 232 282 189 549 370 676 311 562 950 910 551 240 590 716 20 425 691 330 91 706 785 749 274 977 802 884 298 946 833 680 302 527 154 275 144 980 901 797 384 149 822 168 921 826 640 669 563 32 434 745 697 446 67 71 938 186 13 433 70 92 760 26 410 740 854 610 747 888 290 319 79 774 742 256 960 728 44 51 968 711 739 970 37 118 3 679 155 999 226 813 564 598 860 810 448 355 533 452 18 748 377 54 982 138 769 652 331 390 543 305 312 383 665 752 829 472 849 108 246 556 666 959 878 27 678 147 96 635 48 191 692 939 762 265 23 235 318 698 286 655 548 174 815 415 135 855 315 733 554 710 443 362 15 599 701 356 95 630 658 297 358 359 206 83 201 876 724 671 469 727 847 261 12 574 689 886 823 818 411 638 378 5 490 869 269 998 597 949 276 989 473 594 143 672 119 538 693 151 736 781 379 565 456 705 387 646 64 124 862 28 205 928 317 382 535 647 624 757 277 24 536 616 930 295 641 570 121 86 958 686 916 571 794 605 352 834 266 544 903 558 530 376 72 336 591 803 197 513 601 496 169 993 109 249 375 859 503 504 417 927 458 621 110 642 913 865 116 912 293 488 607 497 346 223 714 339 659 771 721 97 618 492 21 828 805 238 332 153 185 243 338 460 827 915 506 981 603 890 76 170 99 522 399 482 673 874 925 117 703 61 994 582 451 428 214 966 260 36 142 783 58 55 363 568 917 600 907 310 326 84 459 617 619 161 627 587 871 654 328 401 16 233 408 369 845 422 872 892 788 354 867 69 195 112 979 400 732 729 429 609 100 863 111 702 791 583 577 59 29 56 361 717 592 200 39 322 580 283 395 104 309 2 438 935 510 501 961 50 495 396 784 735 471 885 224 493 948 529 937 418 687 750 494 620 951 389 131 62 741 225 87 500 734 470 947 918 466 53 819 972 541 782 694 811 130 439 761 814 413 880 291 806 426 136 8 840 215 432 75 922 936 964 719 337 584 799 217 775 581 709 585 853 167 14 651 203 264 920 812 861 909 483 22 566 394 216 88 30 838 608 481 879 320 289 253 667 722 449 4 258 606 612 767 68 902 906 636 102 435 547 524 787 790 809 883 895 208 746 353 778 528 152 33 825 412 455 334 294 875 372 700 793 593 300 954 1 284 237 505 350 6 657 10 461 971 905 465 175 107 836 199 209 158 559 407 737 633 975 648 357 626 662 661 952 848 942 270 172 984 419 877 933]