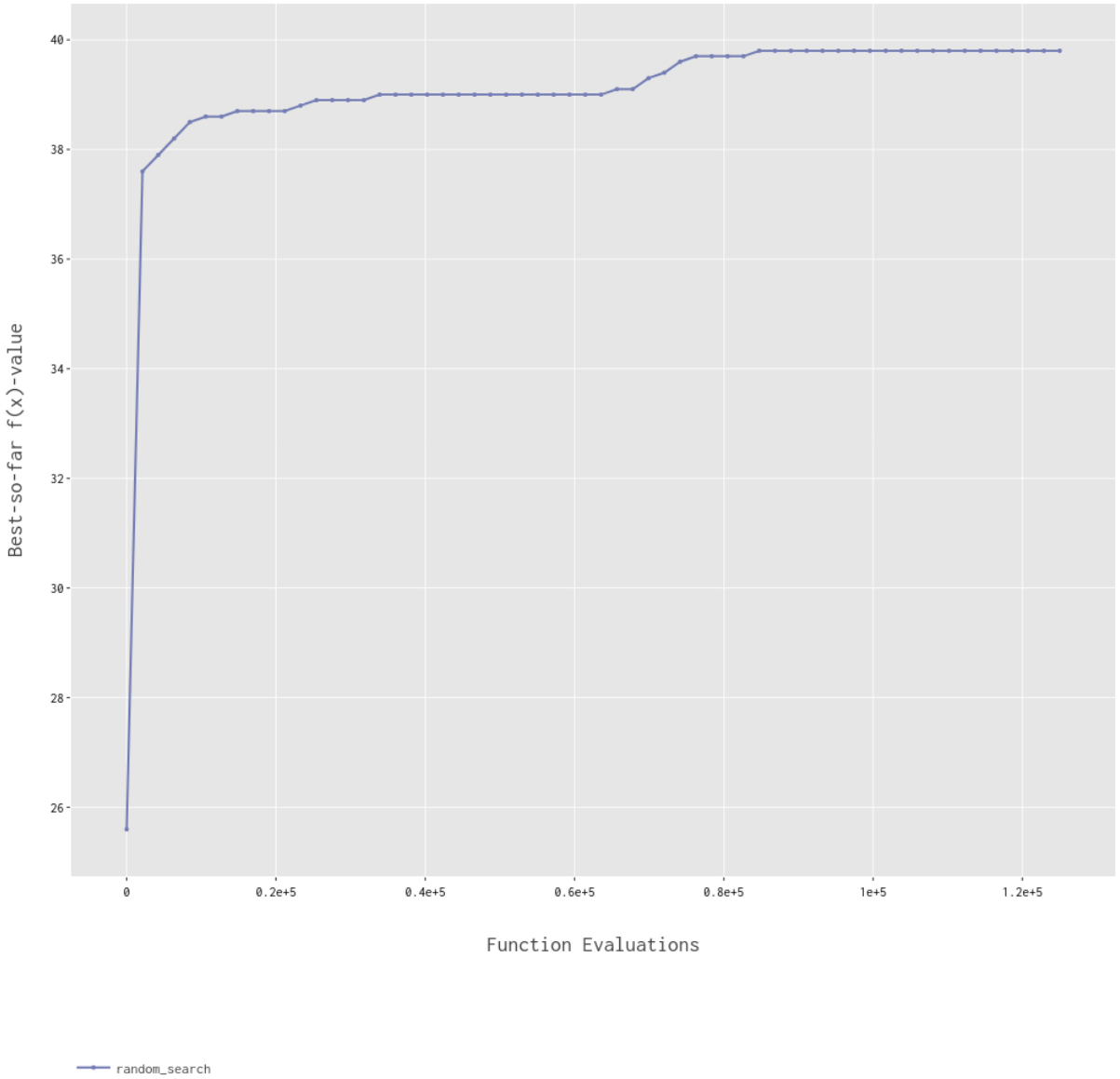
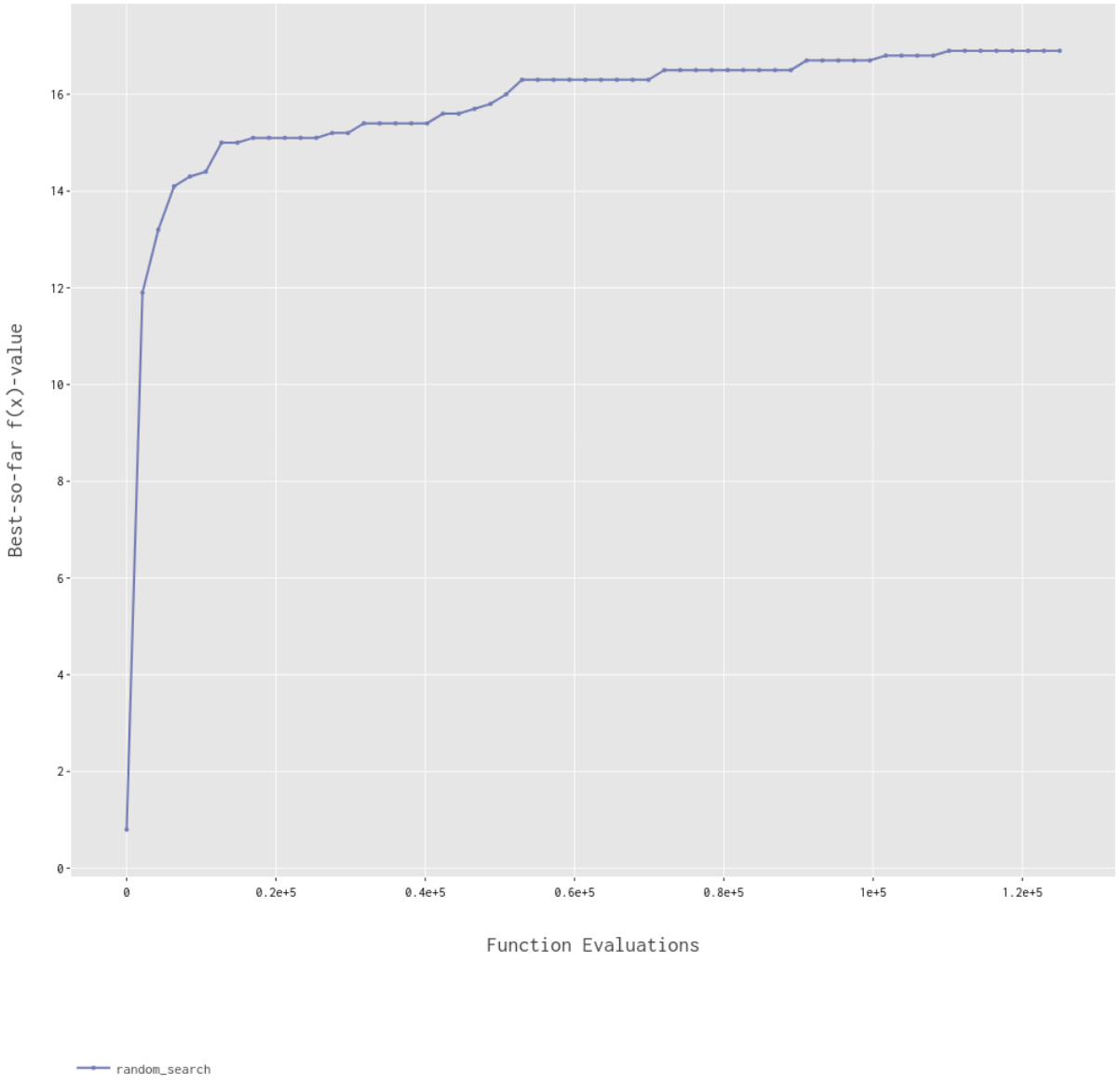
Evolutionary Computing Assignment 2 - Plots, and Analysis

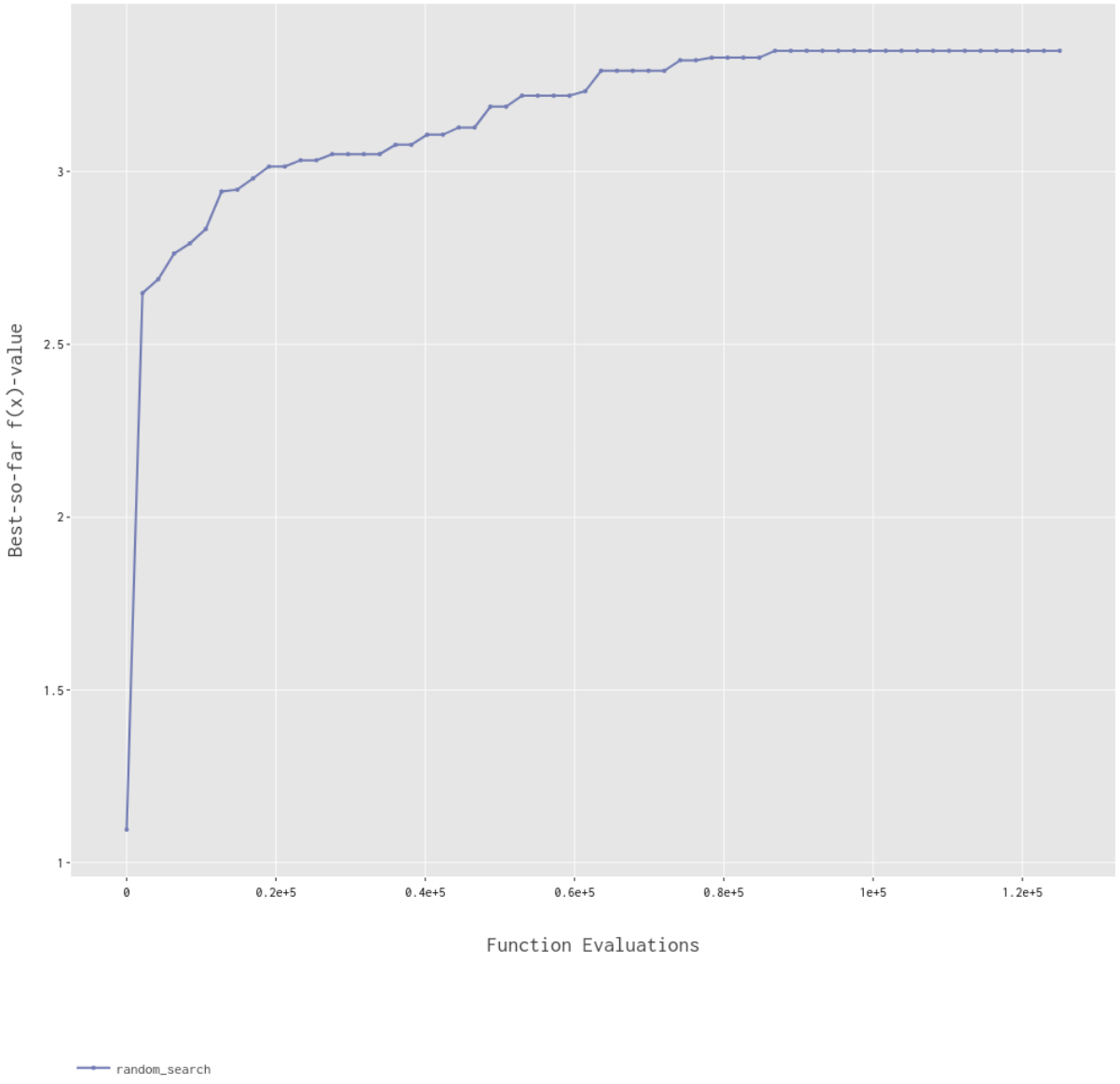
Question 1



Random Search F1



Random Search F2



Random Search F18

Question 2

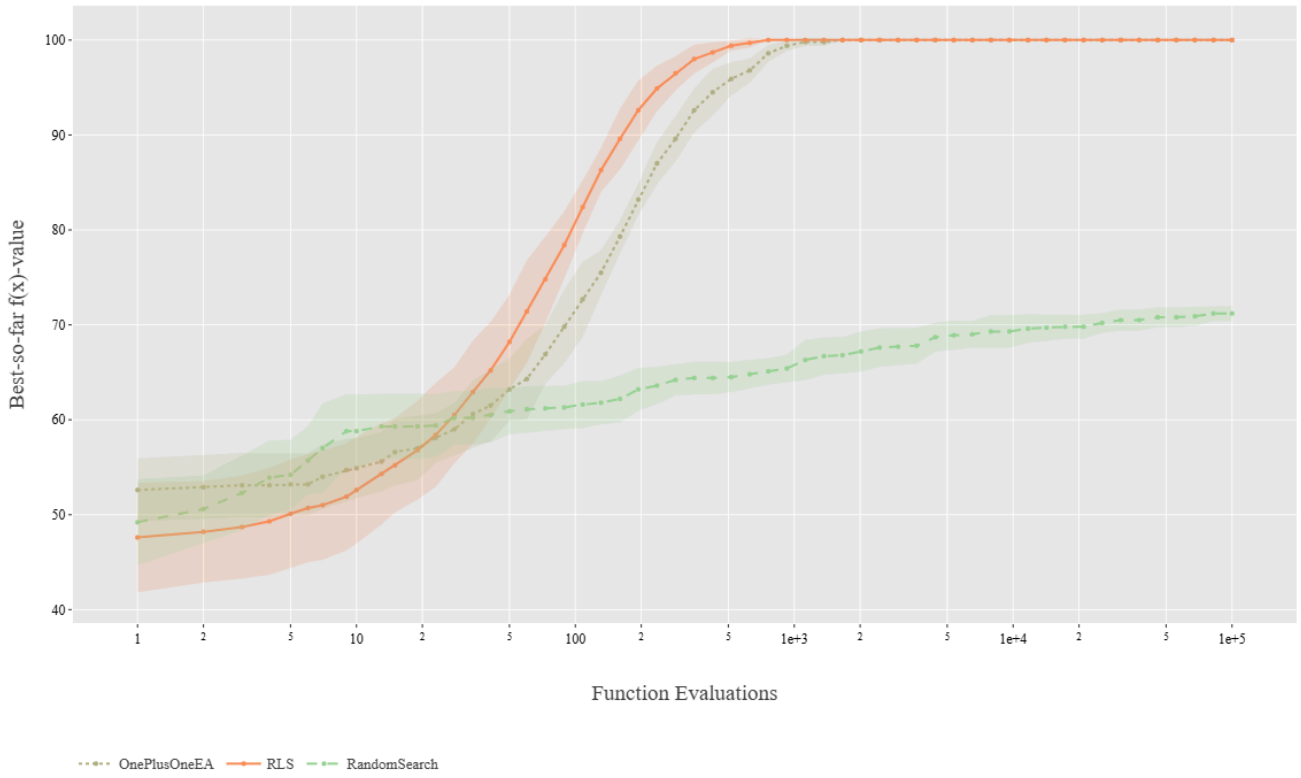


Figure: Exercise 2 - F1

From the plot, we can say that Random Search performs the weakest for One Max. It initially benefits from lucky guesses (achieving a better average fitness than RLS after the very first evaluation), but quickly stagnates. By around 30 evaluations, RLS and (1+1) EA both overtake it. RLS is the strongest throughout most of the run, since its mutation strategy (flipping exactly one bit at a time) is well suited for steadily increasing the number of ones.

The (1+1) EA starts better at the beginning, however, approximately around 30, it does not perform as well as RLS. The (1+1) EA starts slightly slower than RLS because its mutation flips multiple bits with probability 1/n, so many of the mutations early inadvertently flip a one bit to a zero bit, which do not improve the solutions. However, since only the best-so-far fitness is recorded, the curve never decreases, and over time the algorithm gradually learns to behave more like RLS. This trend is visible after around 1000 evaluations, where the performance curves of RLS and (1+1) EA effectively converge to the optimal solution at the end.

Standard deviations shrink over time for both, indicating stable convergence. Standard deviation shrinkage in Random Search is an artifact of best-so-far tracking, not genuine convergence like in RLS and EA.

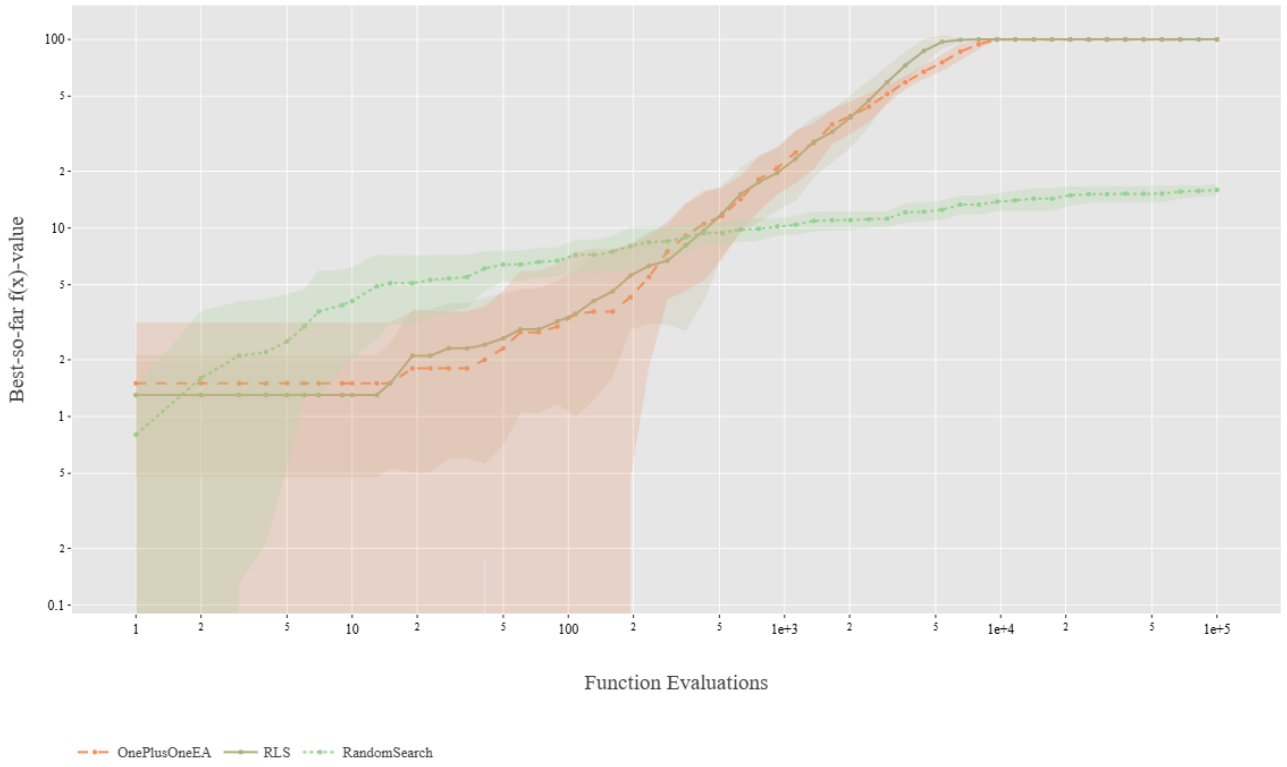


Figure: Exercise 2 - F2

Random Search performs poorly on Leading Ones because improvements require the first bits (from the left) to be ones. Since Random Search samples solutions completely at random, it rarely produces candidates that increase the number of leading ones early on, resulting in very slow progress. RLS is more effective because it flips one bit at a time and only accepts improvements, allowing it to steadily increase the number of leading ones.

The (1+1) EA starts slightly higher than RLS with the gap less than 1. However, flipping multiple bits can also disrupt already-correct leading bits, so early improvements vary between runs. This is reflected in the standard deviation: for the approximately first 200 evaluations, it forms a roughly rectangular block, showing that some runs improve faster while others lag behind. After that threshold, the standard deviation decreases, we can see a “corn hat” shape as improvements become more consistent across runs.

Overall, RLS quickly overtakes the (1+1) EA due to its steady one-bit improvements. Over time, the (1+1) EA catches up, sometimes even temporarily surpassing RLS, and both algorithms eventually reach the optimal solution. Best-so-far fitness increases monotonically for all algorithms. Standard deviations shrink for RLS and (1+1) EA as they converge, while Random Search shows similar patterns but for different reasons.

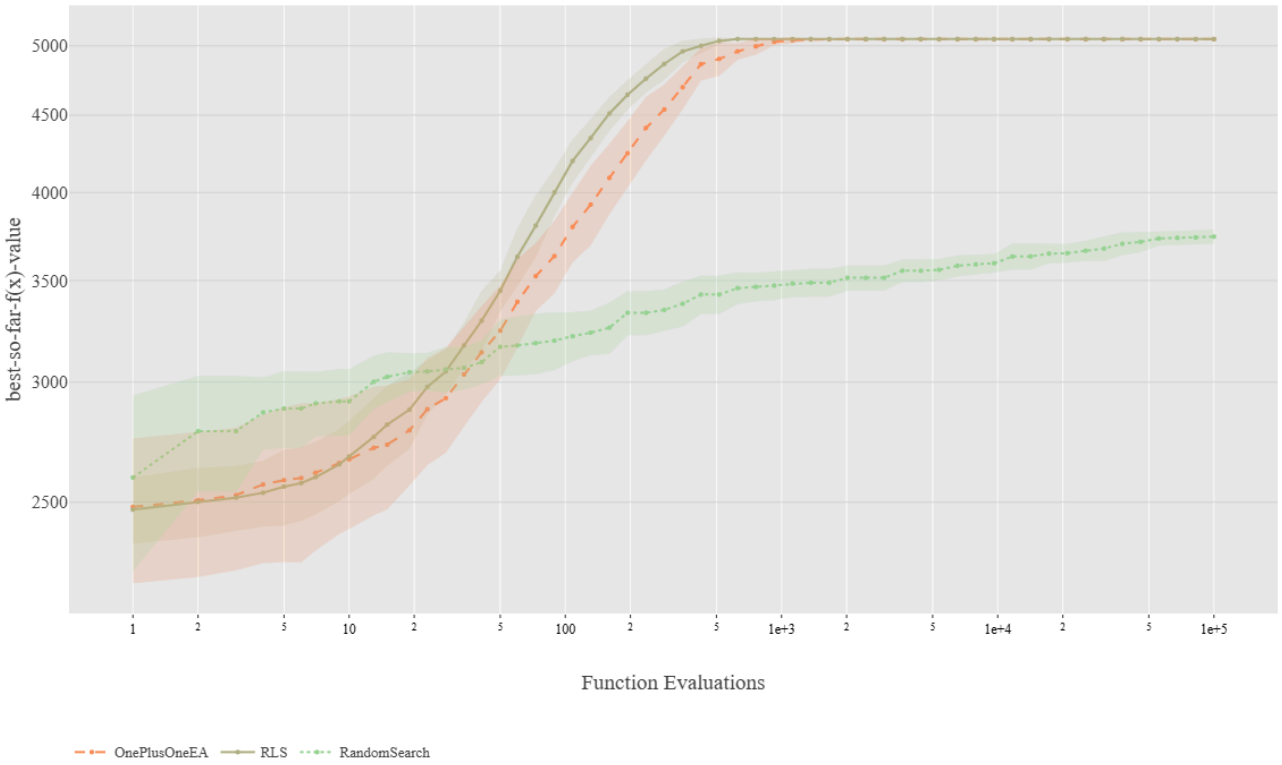


Figure: Exercise 2 - F3

Again, Random Search performs the worst, as it relies purely on chance to find improvements and struggles to escape local structures in the fitness landscape. For this problem function, it looks flat, resulting from slow progress across runs.

RLS outperforms the (1+1) EA mostly all the time because flipping one bit at a time allows steady, reliable improvements. The (1+1) EA improves slightly slower due to the possibility of multi-bit flips that do not always produce beneficial changes, but over time, its performance gradually catches up. Around 1000 iterations, the performance curves of RLS and (1+1) EA begin to merge, and both eventually reach the same point.

Best-so-far fitness increases monotonically for all algorithms. The Standard deviations shrink for all three algorithms, RLS and (1+1) EA as runs converge, and for Random Search is due to its purely random sampling.

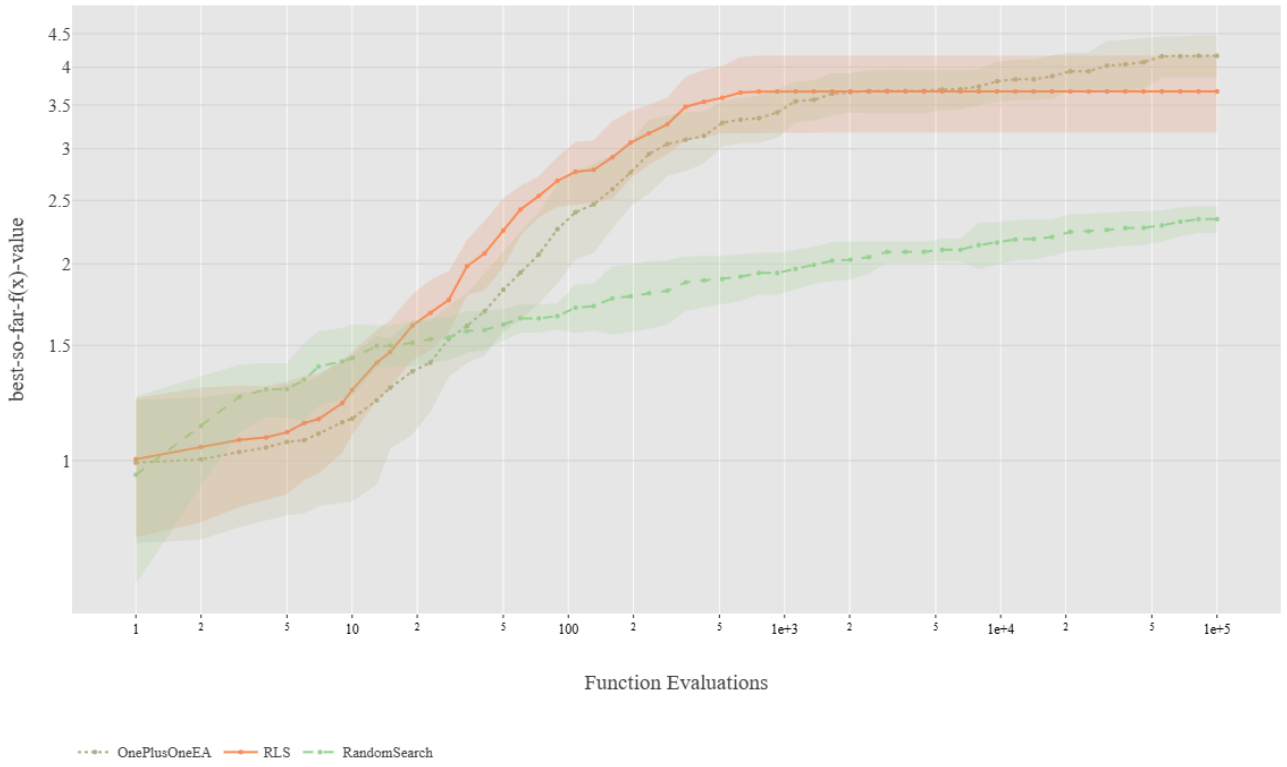


Figure: Exercise 2 - F18

F18 is a challenging problem with plateaus or deceptive regions, where many solutions have similar fitness, making consistent improvements difficult for simple search algorithms. Random Search performs the worst, even though it has good results at the beginning, it samples solutions completely at random and rarely finds higher-fitness solutions on the plateau.

RLS improves steadily in the early stages, but progress stalls around a fitness of 3.5, likely because it got trapped on a local optimum or plateau. I think this is because only flipping one bit is hard to get out of the local optima. On the other hand, the (1+1) EA starts slower, as flipping multiple bits can produce inconsistent improvements early on. However, it eventually outperforms RLS, since flipping multiple bits allows it to escape local optima and continue improving, reaching slightly higher fitness values, slightly above 4.

Fitness increases monotonically for all algorithms. Standard deviations shrink for RLS and (1+1) EA as the runs converge, while Random Search shows larger fluctuations. Both RLS and (1+1) EA eventually reach the optimum, but the (1+1) EA can surpass RLS slightly due to its ability to escape plateaus.

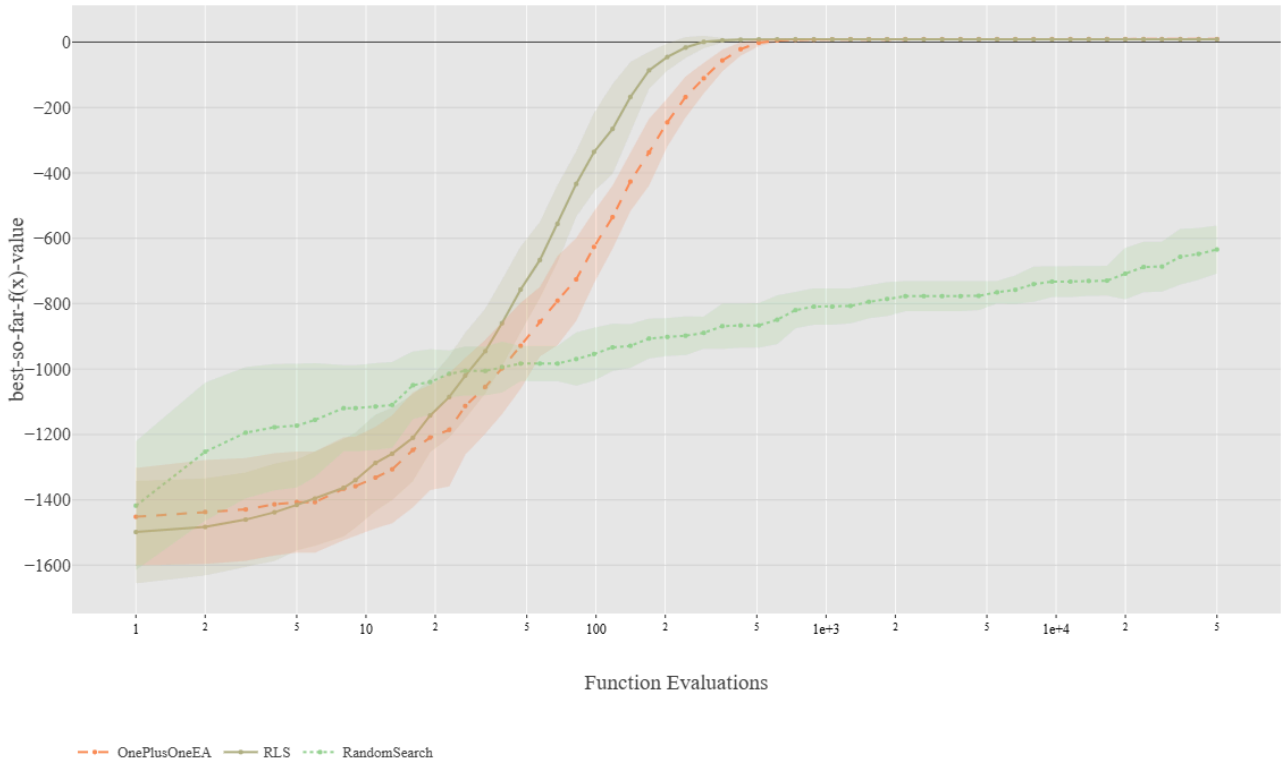


Figure: Exercise 2 - F23

One special observation for F23 is that the plots can appear “empty” when the y-axis is scaled. This happens because the problem’s objective function starts at negative values and is maximized toward 0 over time. To visualize the curves clearly, it is better to apply the log-transformation to the x-axis only (function evaluations), not the y-axis.

The F23 is a deceptive problem that starts with negative fitness values, which makes it difficult for algorithms to make early progress. Random Search performs the worst as usual, as it samples solutions completely at random and rarely finds candidates that significantly improve the fitness. It often gets ‘lucky’ at an early stage, but then stagnates for most of the run.

RLS generally performs better in all stages, it steadily increases its fitness by flipping one bit at a time and reaching 0 at around 300 evaluations. The (1+1) EA starts slightly higher in the early stage due to occasional lucky multi-bit flips but performs not as well as LRS later on. Over time, RLS and (1+1) EA converge and eventually reach the optimum.

The standard deviation for (1+1) EA, and RLS shrink as the algorithms converge. Random Search shows fluctuations throughout, consistent with its purely random sampling strategy.

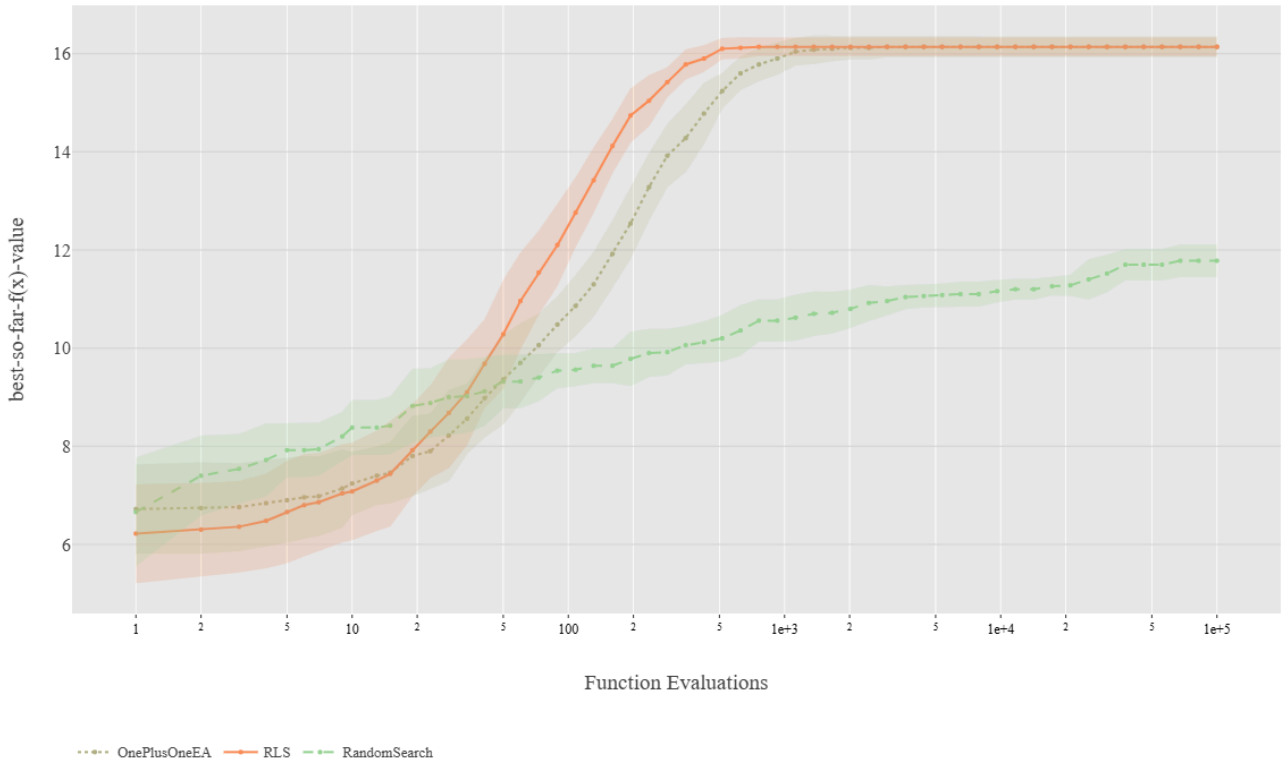


Figure: Exercise 2 - F24

Again, Random Search performs the worst. It steadily climbed throughout the iterations and finished near a fitness of 12 without ever reaching the global optimum (around 16) compared to the other two algorithms.

RLS initially starts slower than the (1+1) EA, but by 20 evaluations it consistently surpasses the (1+1)EA. From that point onward, RLS and the (1+1) EA improve together, gradually converge and eventually reach the optimum fitness of 16. This could be a trap as both algorithms cannot perform any better, which align with the deceptive characteristics of the problem function.

The standard deviation for both RLS and the (1+1) EA shrinks over time as the algorithms converge toward the optimum, reflecting stable performance across runs. Even though we still can see a little bit of fluctuation, it is more consistent than the Random Search. In contrast, Random Search exhibits more fluctuation in standard deviation, consistent with its randomness and lack of guided progress mechanism.

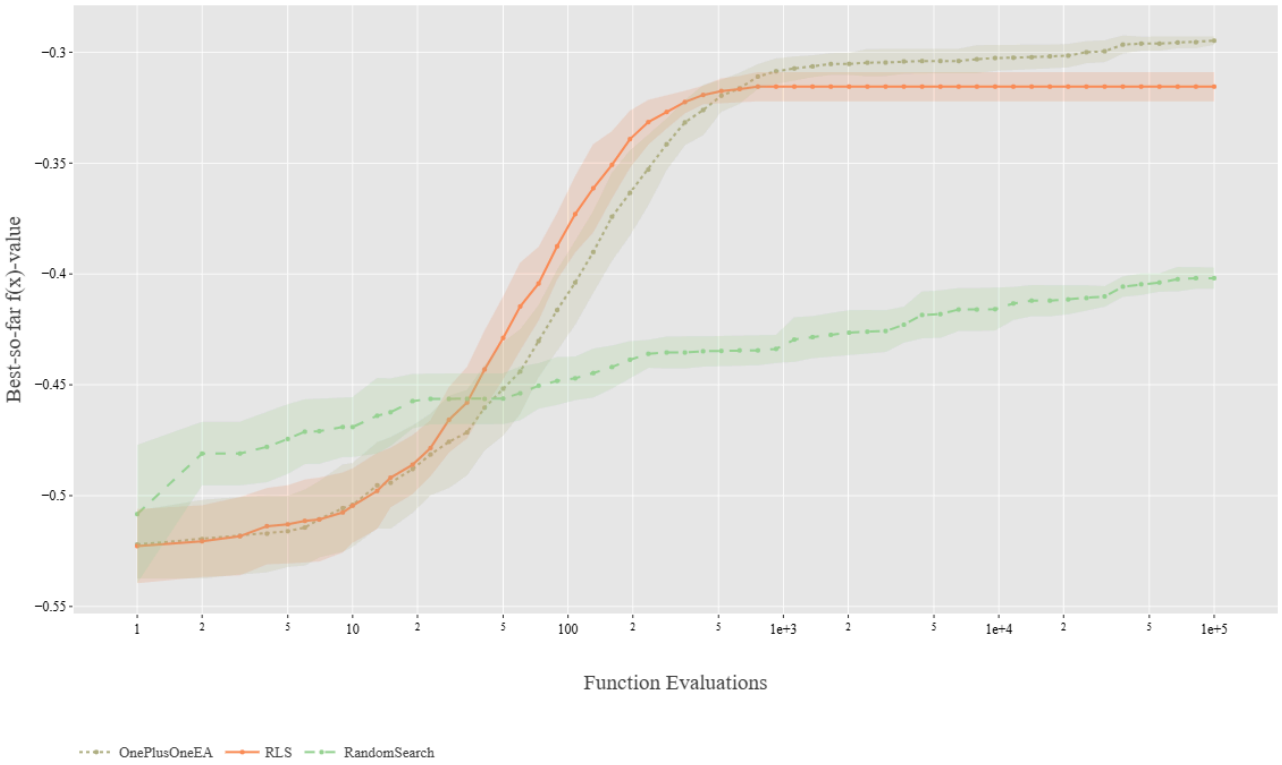


Figure: Exercise 2 - F25

At first glance, the plot appears almost empty when the y-axis is scaled. This occurs because the objective function's range for F25 (like F23) starts negative and is maximized toward 0 over time. Without applying a log-transformation to the x-axis, leaving the y-axis untouched, the plots are difficult to interpret. This highlights why scaling choices are crucial for interpreting results.

Random Search performs poorly, only performing a little bit better, and reaching the value of -0.4 at the end. It tends to make one early "lucky guess" that provides a decent solution, but afterward spends the majority of its evaluations (90–95%) without finding significant improvements.

RLS and the (1+1) EA show nearly identical performance curves. However, in the middle stage, the RLS perform slightly better than (1+1)EA. Both steadily improve after early iterations but for RLS, it is stuck in its local optima and cannot escape, while the other algorithm gets a better value at above -0.3.

The standard deviation decreases over time for both algorithms, indicating consistent convergence. However, RLS stuck at its local optima, the standard deviation remained unchanged. Finally, the Random Search retains high fluctuation due to its unguided nature.

Conclusion

Random Search typically makes one “lucky” improvement early on, then stagnates for most of the run. This explains why its curves across runs look almost identical and why progress flatlines after only 5–10% of evaluations.

Across all problems, Random Search consistently performs the worst, even with slightly better value at starting point, failing to make significant progress beyond early gains. Moving on to RLS, it performs best on unimodal functions and reliably reaches the optimum, however it can be stuck at local optima, and only changing one bit at a time is not good enough to help itself get out of trap.

The (1+1) EA tends to lag behind RLS initially but gradually converges, and sometimes it can even get out of its local optima as it flips multiple bits at once.

Overall, the results highlight the strength of guided local search strategies compared to unguided random sampling.

Question 3

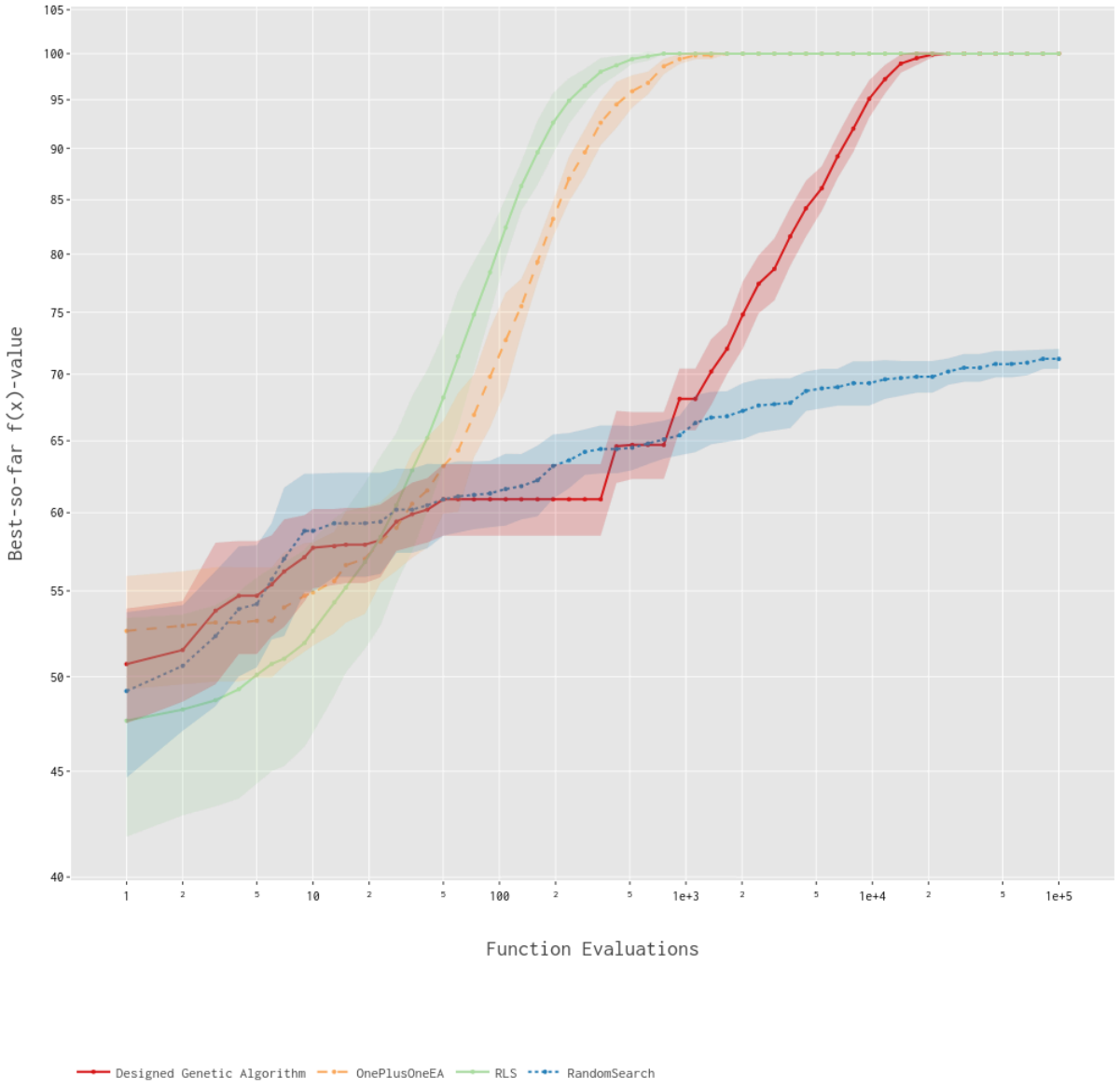


Figure: Exercise 3 - F1

In the OneMax problem, the Genetic Algorithm (GA) initially increases fitness quickly during the early iterations, but then its progress stalls, forming a plateau in the curve. Unlike RLS and (1+1) EA, which steadily climb and reach the optimum after only 1,000 evaluations, GA takes much longer to converge. After the plateau, it seems flat from 90 - 800 evaluations, the GA eventually escapes and dramatically improves, finally reaching the optimal solution around 20,000 evaluations.

The delayed convergence is likely due to the population initially exploring diverse solutions and relying on uniform crossover and mutation. We suggest that at an early stage, those good building blocks are scattered across individuals, so combining them effectively takes time. Once enough high-fitness individuals accumulate, GA can make rapid progress.

Standard deviation across GA runs is relatively high during the plateau, reflecting differences in how quickly individual populations escape early stagnation, and shrink as it converges.

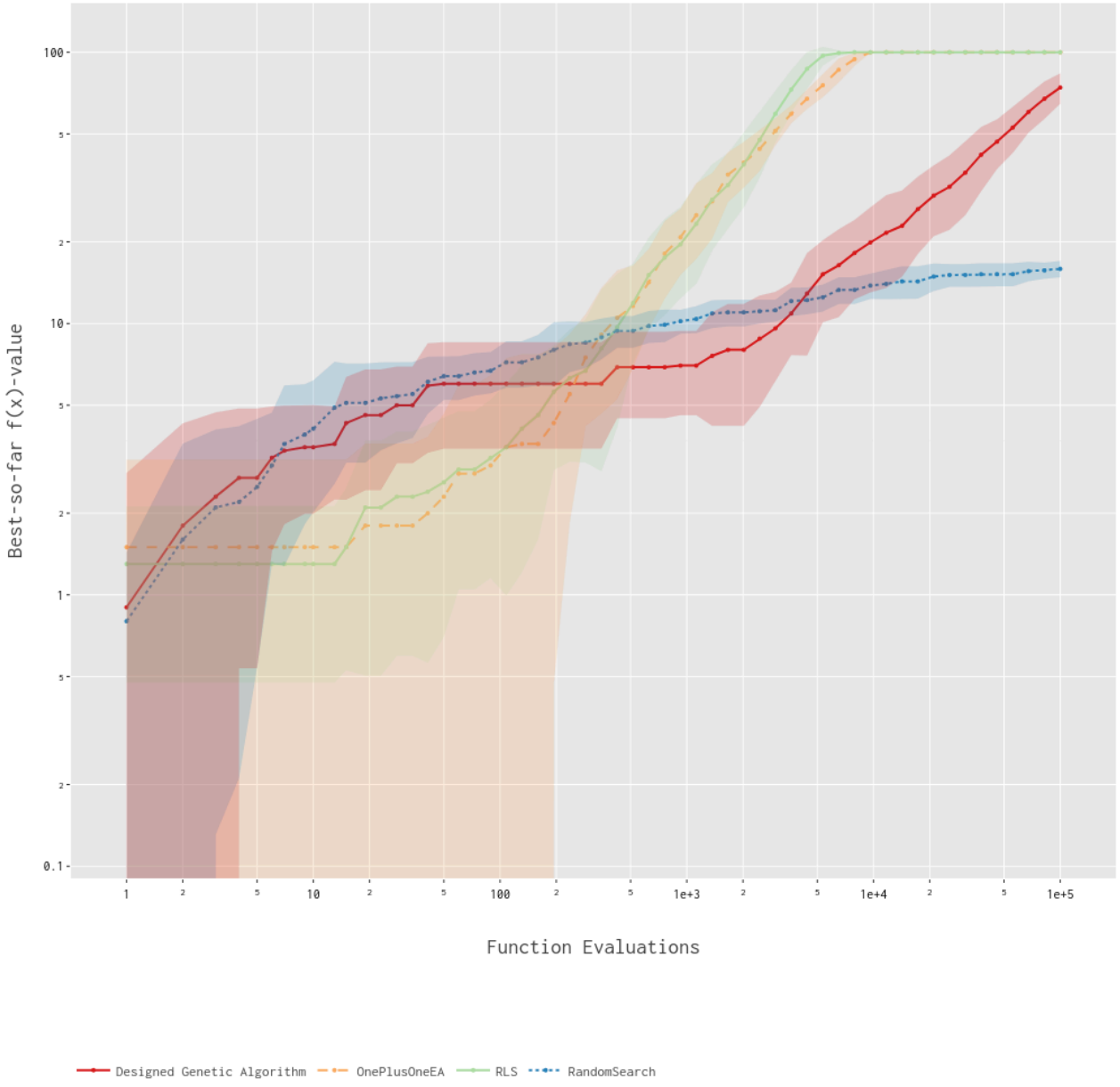


Figure: Exercise 3 - F2

On the LeadingOnes problem, RLS and (1+1) EA steadily improve and eventually reach the optimal solution. The Genetic Algorithm (GA), in contrast, shows a similar pattern to F1, which is to make early improvements but then plateaus. Unlike RLS and (1+1) EA, GA does not reach the optimum within the evaluation budget, only around a value of 90.

We suggest this weaker performance can be explained by the nature of LeadingOnes. The problem requires bits from the leftmost position onward to be correct, but uniform crossover in GA tends to disrupt already-correct leading bits when recombining individuals. While mutation can repair such disruptions, the process is slow, and progress becomes inconsistent.

GA shows slower and more uneven convergence. The higher standard deviation for the GA is a direct consequence of the inconsistent progress caused by crossover. As a result, the standard deviation stays larger and longer compared to RLS and (1+1) EA, and shrinks over time.

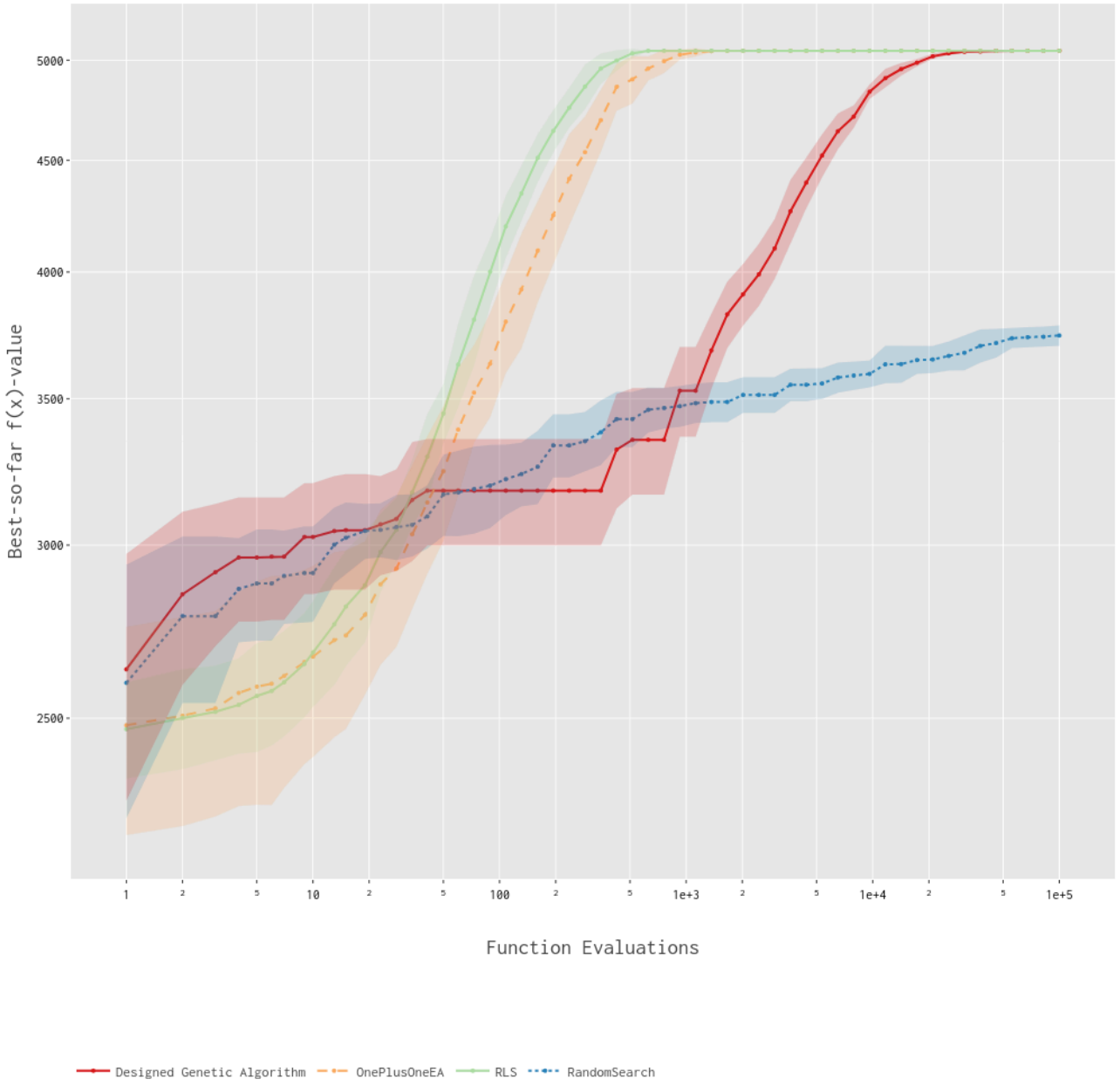


Figure: Exercise 3 - F3

From exercise 2, the Random Search performs the worst, while both RLS and the (1+1) EA reliably converge to the optimum within about 1,000 evaluations. The Genetic Algorithm shows a different pattern: it improves quickly at the start but then stalls between roughly 70 and 700 evaluations. Only much later, around 70,000 evaluations, GA manages to bridge the gap and eventually reach the optimum, as RLS and (1+1) EA do much earlier.

Overall, GA lags significantly behind RLS and (1+1) EA in both speed and consistency, though it eventually recovers after the plateau. This illustrates GA's strength in eventually exploring beyond local optima, but also its weakness in efficiency compared to simpler algorithms.

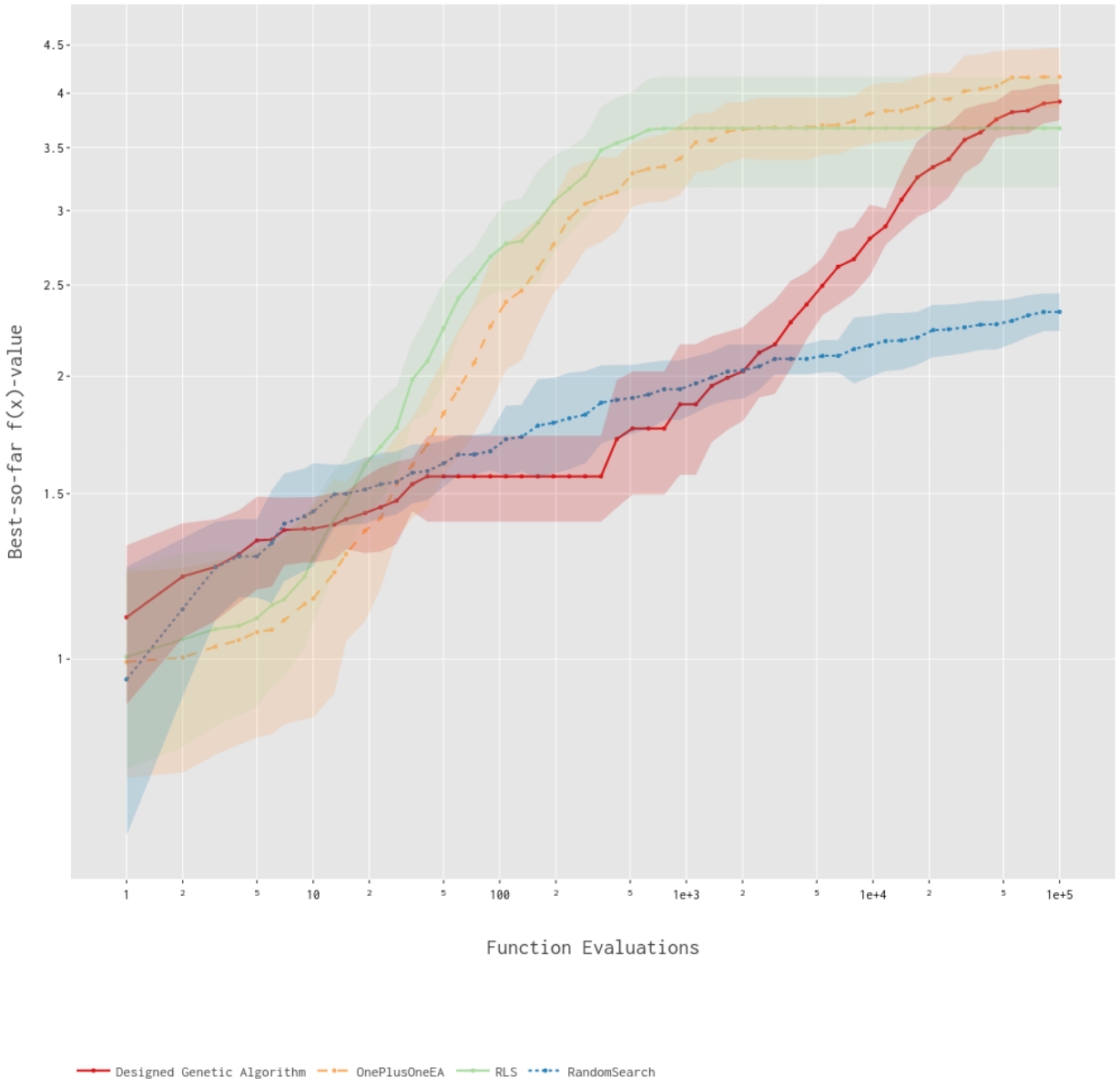


Figure: Exercise 3 - F18

Random Search again performs the worst, not making much progress beyond the start. RLS improves steadily at first but stalls around a fitness value of around 3.5, and it is unable to escape a local optimum. The (1+1) EA eventually outperforms RLS by flipping multiple bits, reaching fitness slightly above 4.

The Genetic Algorithm shows a similar shape to previous problems: a quick rise at the beginning, followed by an unchanged behavior. After that point, GA improved dramatically, eventually reaching a better fitness compared to RLS as it avoids being stuck in the same local optima. GA's final performance is slightly below that of the (1+1) EA, around the value of 4. This highlights GA's advantage in maintaining diversity across its population, which eventually allows it to discover better solutions, but only after a much longer delay compared to RLS and (1+1) EA.

The standard deviation for GA is quite interesting as it fluctuated throughout the run, reflecting the stochastic nature of crossover and mutation. This suggests that GA is less stable and more varied across runs.

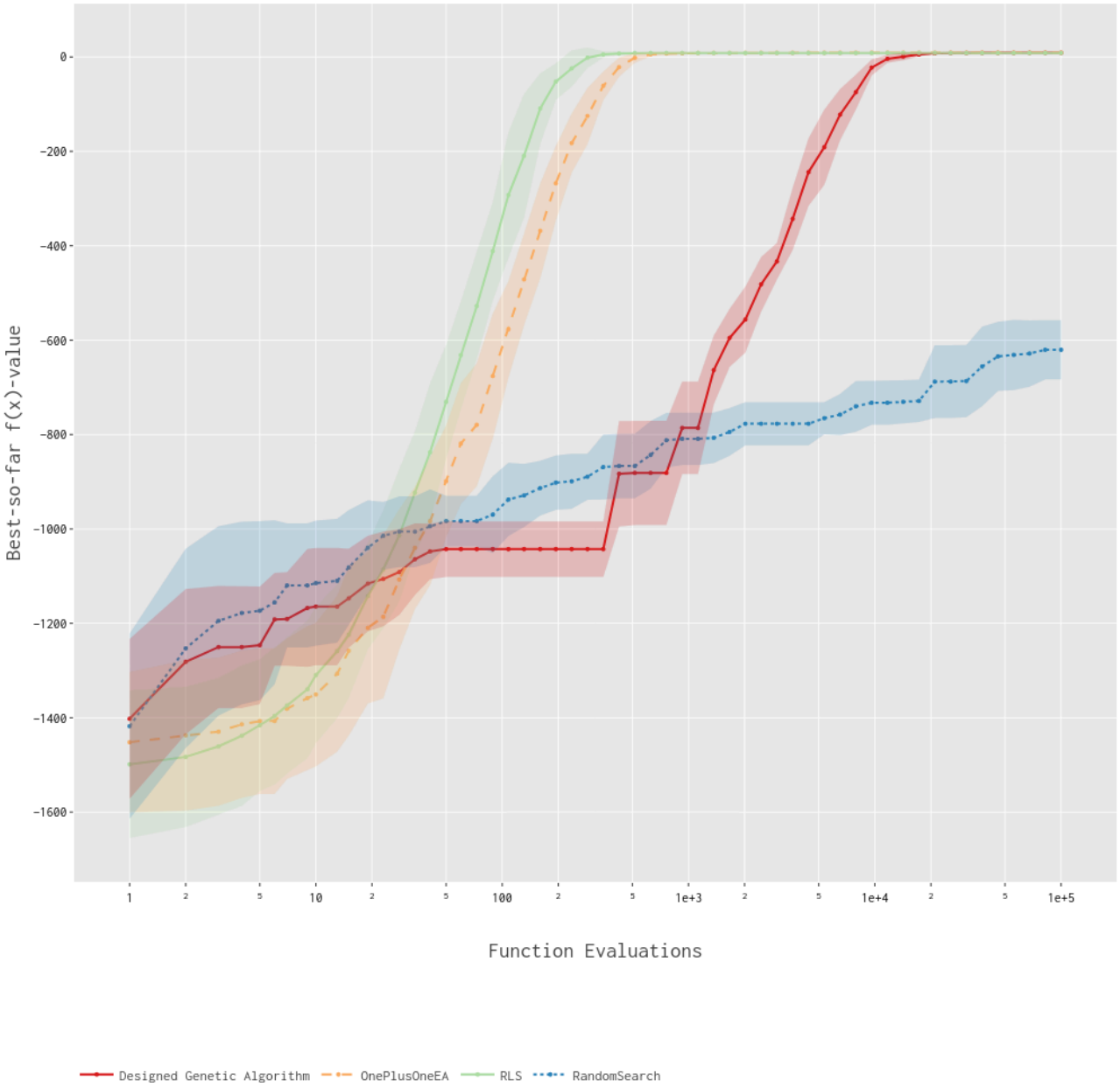


Figure: Exercise 3 - F23

Similar to the plots above, after around the 30th iteration, our GA enters a prolonged stagnation phase plateauing around -1100 to -1000 fitness while RLS and (1+1)-EA overtakes it. This results from premature convergence driven by strong tournament selection pressure (size 8 from population 44), which eliminates diversity, combined with uniform crossover's disruptive effect on the problem's high constraint nature. Moreover, the low mutation-rate is not enough in that phase to maintain diversity to escape local optima. Fortunately, like the above plots, the GA is able to make a come-back, as by this point the accumulated mutation rate provides enough variations for it to escape the local optima and ultimately converge at the global optimum but significantly slower than (1+1) EA and RLS – demonstrating that population-based recombination offers little advantages on constraint satisfaction problems (such as this one) where (immediate) local search through incremental single-bit modification proves far more effective.

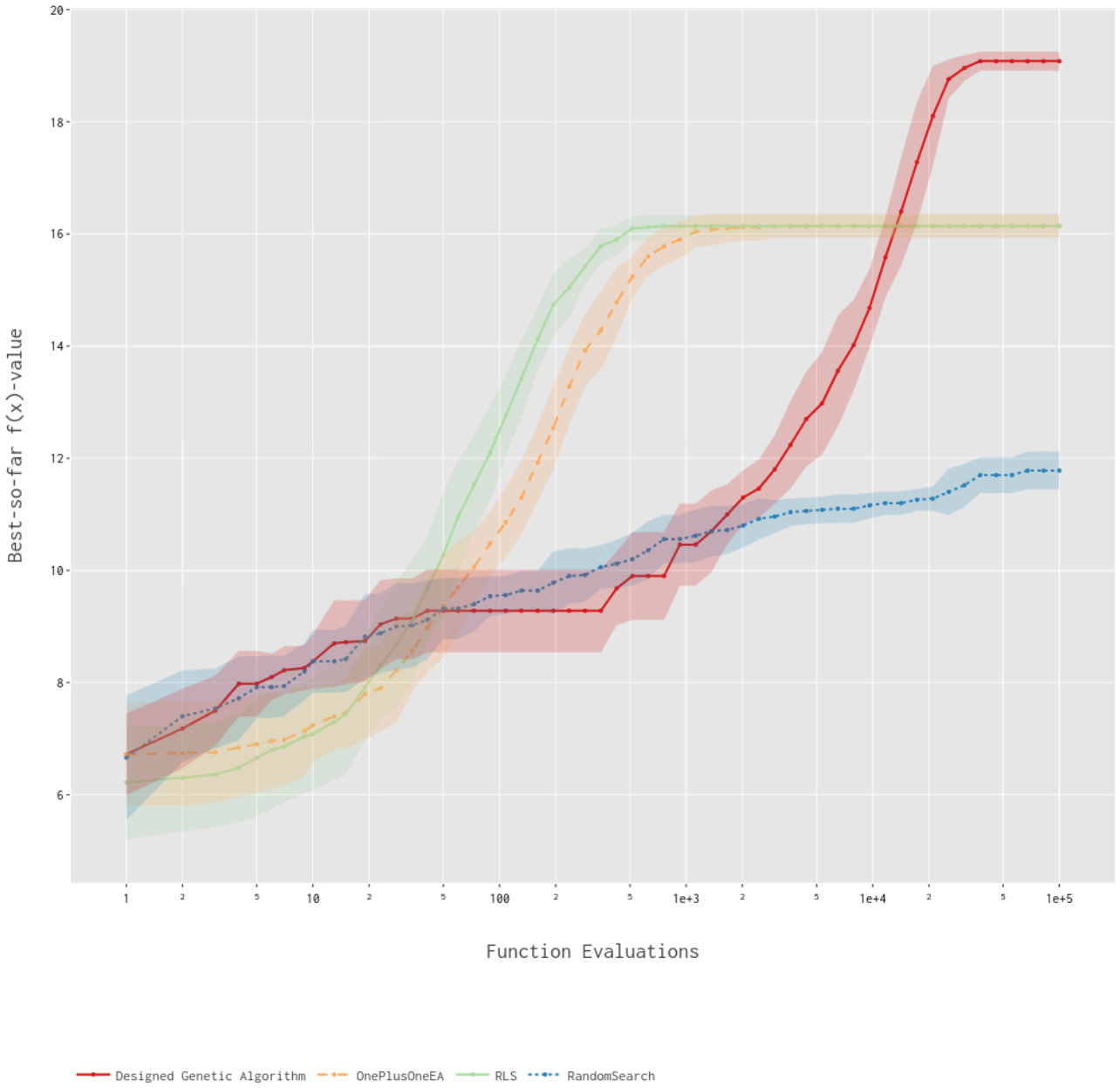


Figure: Exercise 3 - F24

The Designed GA demonstrates clear superiority on the Concatenated Trap problem, being the only algorithm to reach the global optimum (fitness n/k = 20 – where n = 100 and k = 5). While RLS and (1+1)EA plateau around fitness 16, trapped by the deceptive gradient that misleads single-bit mutations away from the optimal all-ones segments, the GA successfully overcomes this deception through crossover recombination. The uniform crossover operator combines building blocks (trap segments) from different individuals in the population, allowing the GA to assemble the 20 optimal five-bit segments required for maximum fitness. This validates the building block hypothesis: population-based recombination enables escape from local optima on deceptive problems where local search methods fail. RandomSearch performs worst (12), unable to systematically construct the required building blocks. Unlike N-Queens where crossover was disruptive, on Concatenated Trap it provides the critical mechanism for discovering global optima, demonstrating how problem structure determines algorithm effectiveness.

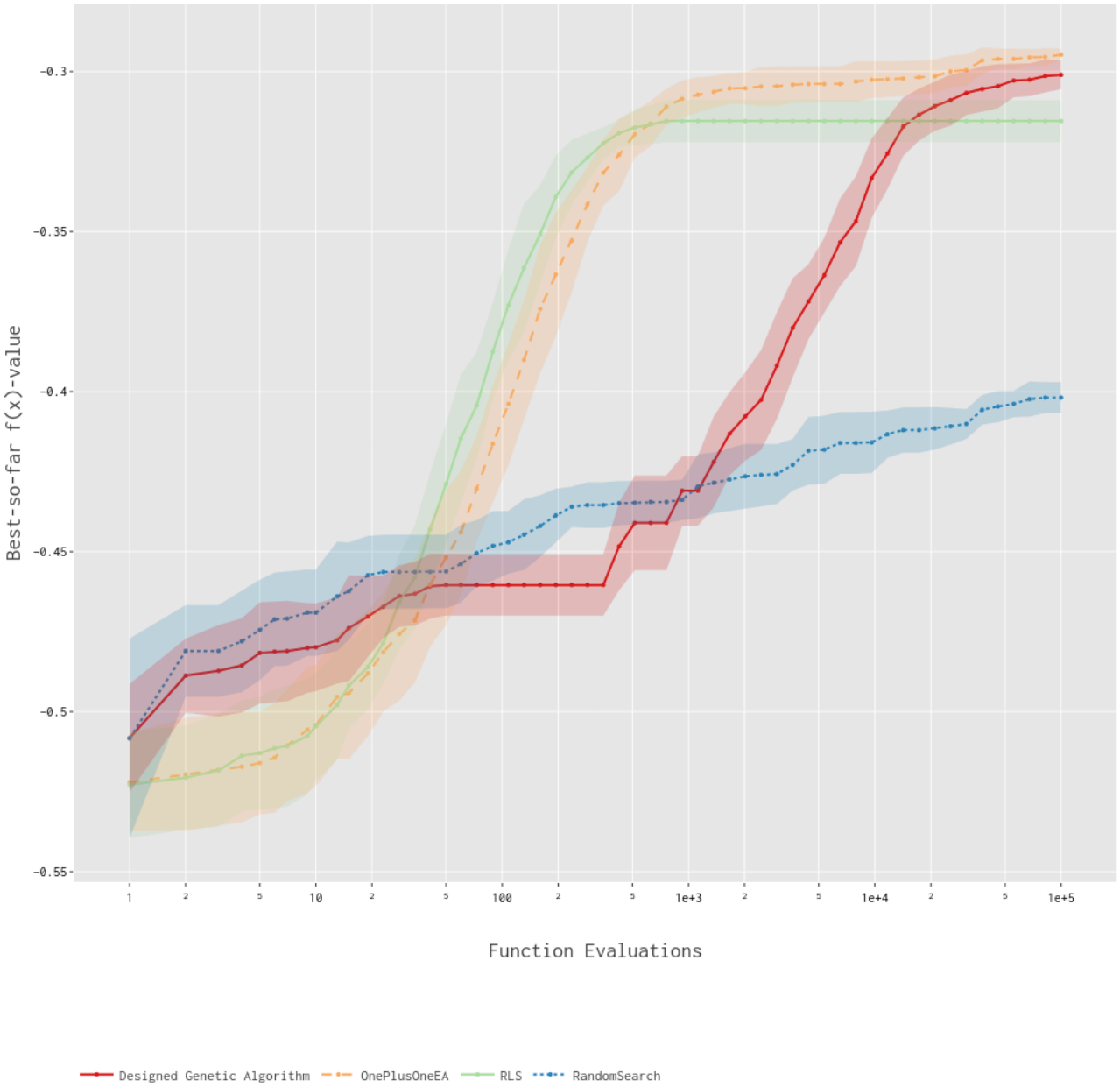


Figure: Exercise 3 - F25

On the NK-Landscape problem, Random Search performs worst, failing to make consistent progress. Both RLS and the (1+1)EA steadily improve, with the (1+1)EA achieving slightly better results by exploiting occasional multi-bit flips. The GA follows its now familiar trajectory: a rapid early improvement followed by a prolonged stagnation period where it lags behind RLS and (1+1)EA. This plateau arises from the strong tournament selection (size 8) and uniform crossover disrupting the fragile epistatic dependencies between bits, reducing effective diversity and preventing steady progress.

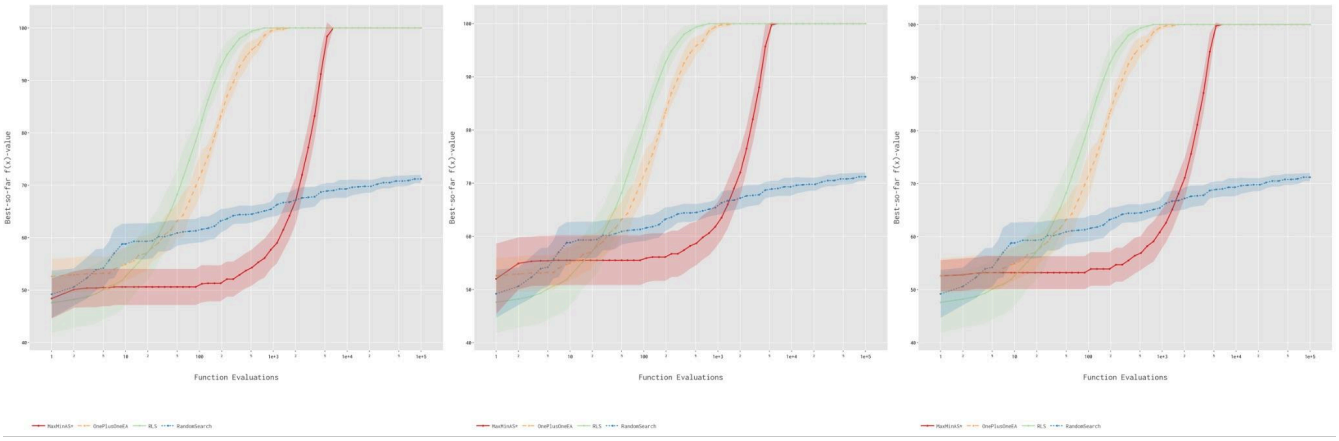
Nevertheless, over time, the GA recovers and nearly catches up to the (1+1)EA, though still more slowly. This late improvement is driven by accumulated mutations introducing enough diversity for recombination to occasionally produce fitter individuals. Thus, while RLS and (1+1)EA excel at incremental local search on rugged landscapes, the GA demonstrates long-term potential to assemble better solutions, but its efficiency is hindered by premature convergence and destructive crossover.

Conclusion

Across all tested problems, the Genetic Algorithm exhibits a characteristic performance pattern: a sharp improvement in the early phase, followed by a plateau due to premature convergence, and finally a late-stage rise as occasional mutations introduce new, fitter individuals. This explains why GA curves often look similar across different functions. While GA is able to eventually approach or reach the optimum on some problems, it generally lags behind RLS and the (1+1) EA, which progress more steadily and efficiently without suffering from the same stagnation. The results highlight that GA's population and crossover dynamics can limit short-term efficiency, but also provide long-term potential for escaping local optima, which could lead to a significantly better fitness like in F24 figure.

Question 4
In the order of small rho first from left to right. (1/n → 1/sqrt(n) → 1)

MMAS*
MMAS* The curve is in red.



MAS with F1*

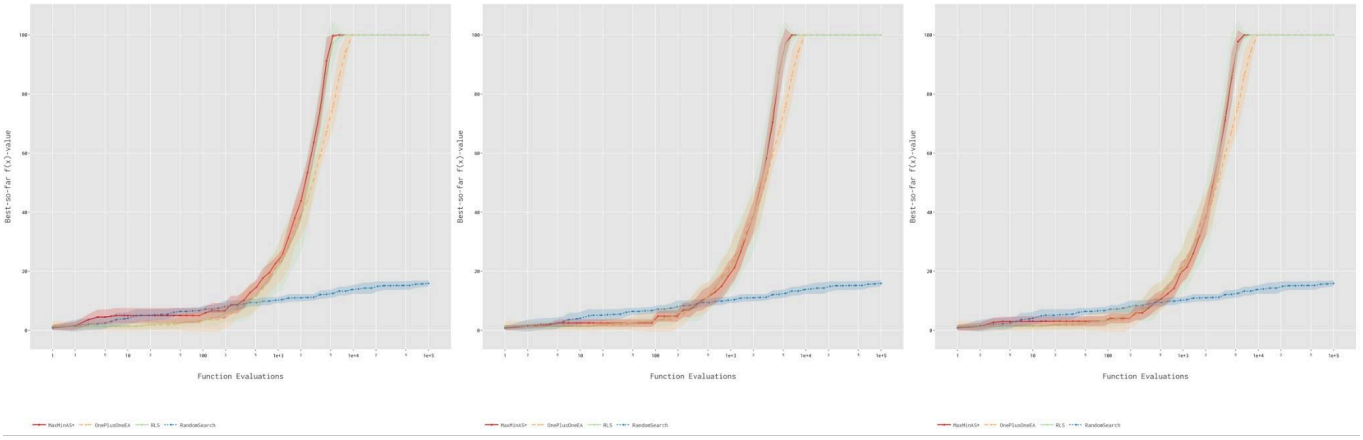
The first thing that we can see is the MMAS* perform not as good as the (1+1)EA and RLS as it reaches the same optimum at really later iterations, at 8000, while it is around 1000 for the other two algorithms.

The three evaporation rates mainly affect how the algorithms reach the optimum rather than the final outcome. We only see the slightly subtle difference as they start reaching the optimum at around 8000 iterations.

With ρ=1/n (small), pheromone changes are slow, so the algorithm explores longer. The fitness curve shows a relatively sharp climb once pheromones accumulate. At first it starts below the value of 50, and it seems to be flat in the early stage, followed by gentle rise, up to when they reach optimum.

With ρ=1/√n, the algorithm reinforces good solutions faster and the climb to optimal fitness occurs earlier and more smoothly. With ρ=1, updates are aggressive and convergence is fastest — for this unimodal OneMax problem this typically yields the quickest approach to the optimum.

Over time, the standard deviation shrinks, especially quite quickly when the fitness curve starts to rise up dramatically in the middle stage. This shows the runs become more consistent as the algorithm converges.

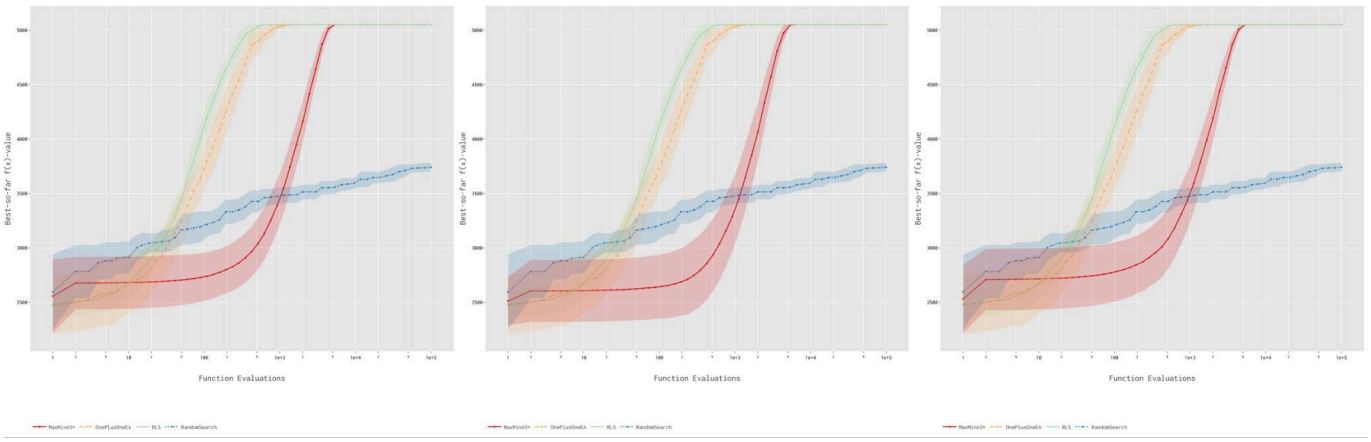


MMAS with F2*

This time, MMAS* performs as well as RLS and better than the (1+1) EA, we do not focus on Random Search as it clearly performs the worst.

For ρ = 1/n (small), the fitness curve rises slightly at first, then enters a plateau. After around 100 evaluations, it begins increasing more smoothly. Nearing the optimum, the curve flattens again, and it takes longer to fully reach the maximum.

For medium and large ρ values (1/√n and 1), they have the same shape with the smaller-rho plot. However, the curves do not rise as high initially, but the final climb toward the optimum is slightly steeper, allowing the algorithm to reach the maximum more efficiently once it reaches the optimum.

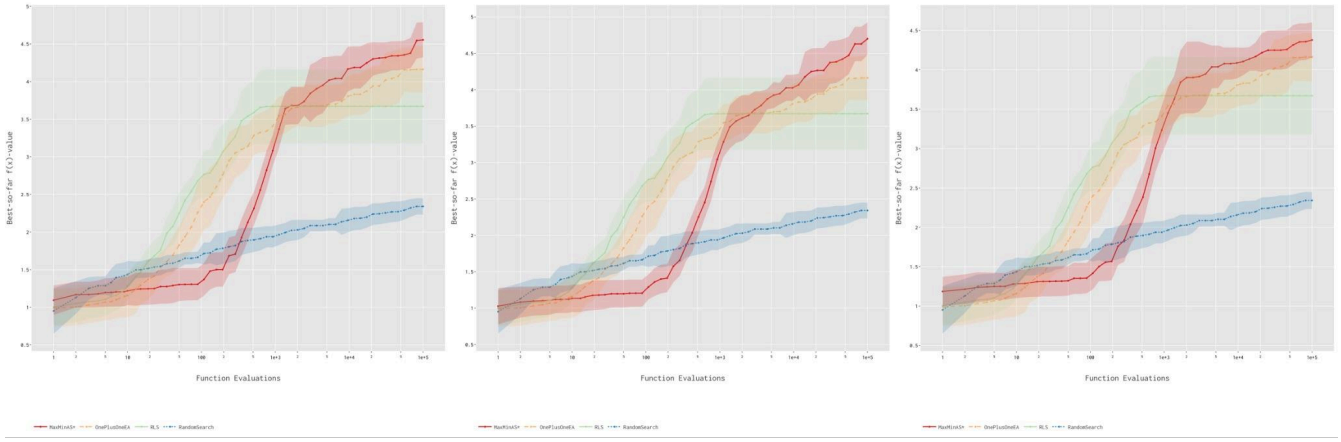


MMAS with F3*

The 3 plots have the same shape. The MMAS* perform worse than both (1+1)EA and RLS, as it needs around 3000 iterations to reach the same optimum, while the other two algorithms get the same result with only around 1000 evaluations.

This time the difference between plots with different evaporation rates is subtle, but the standard deviation has an interesting pattern. For ρ = 1/n, the variation is not so large, while for the other two, it is much more noticeable, especially the early stage.

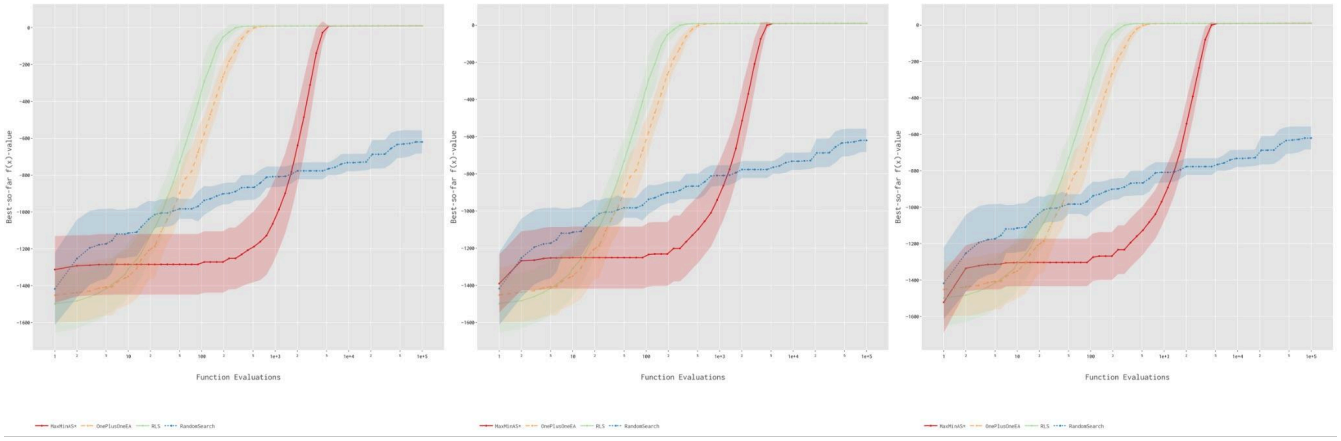
Over time, the standard deviations shrink as the algorithm converges. It shrinks drastically once it starts to escape the plateau and steadily climbs towards the optimum.



MMAS with F18*

On F18, MMAS* performs strictly better than all the algorithms from Exercise 2. Having said that, MMAS* climb slower than the other two, but it gets higher value at the end. This time the effect of the evaporation rate is clearly visible. We can see that the larger the ρ, the faster and the higher the fitness value reaches. For example, the first plot is only slightly higher than 4.5, the second is middle to 5, and the last one, it nearly gets up to the value of 5.

The standard deviation is large in the early stages, which reflects variations in how quickly different runs improve. As the algorithm starts climbing, variation decreases, but around 1000 evaluations, it starts fluctuating upon reaching the end of budget.



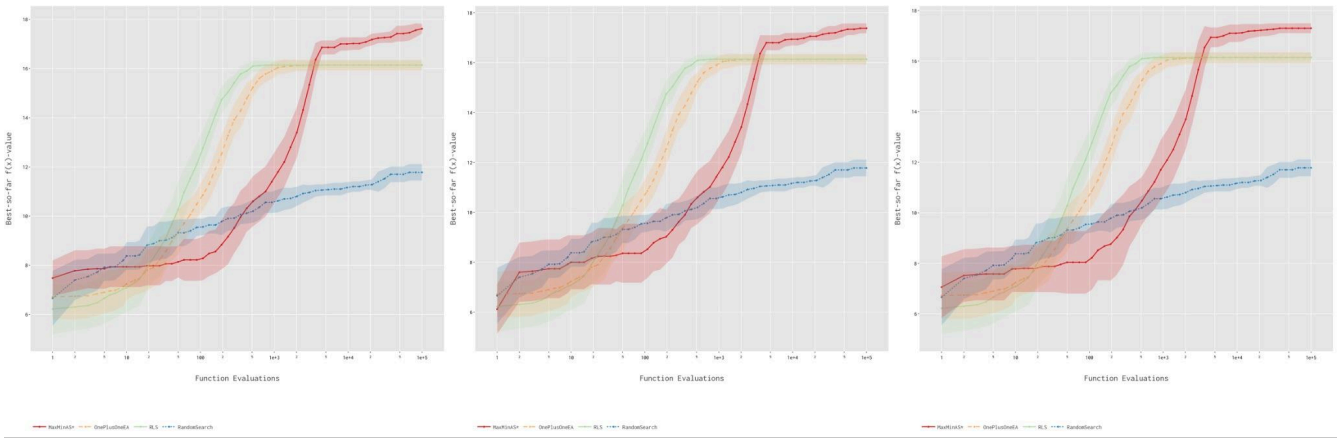
MMAS with F23*

For F23, the MMAS* perform worse than both (1+1)EA and RLS as it only takes 800 iterations, while it is roughly 8000 iterations for MMAS*.

In the early stage, the first plot remains almost flat for the first 200 iterations, whereas with ρ values, the curve starts lower but then rises around the point where the first plot starts, before stagnating. After that point, around 200 evaluations, all three variants begin a steady climb and converge at the optimum, with the fitness = 0. The difference is that with smaller ρ, the final climb is steeper, while it is smoother progress for the higher evaporation rate value.

This behavior reflects how the evaporation parameter affects exploration and exploitation. A small ρ slows down pheromone reinforcement, so the algorithm explores longer before suddenly accelerating once promising paths dominate (producing the steeper late climb). Larger ρ reinforces solutions faster, leading to earlier improvements, but the climb to the optimum is smoother.

Standard deviation is large at the beginning, reflecting high variability between runs, but shrink as it start climbing towards the optimum.

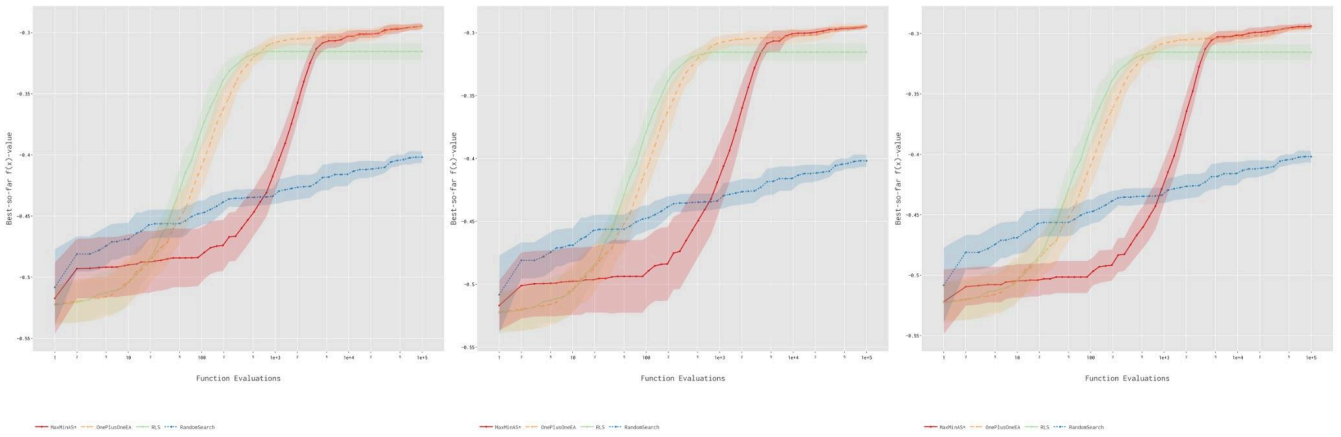


MMAS with F24*

This time for F24, the three plots for MMAS* have a similar shape. Compared to RLS and (1+1)EA, it climbs more slowly at first but achieves a much better result, reaching close to 18 fitness value, whereas others are stuck at a plateau around 16.

When it starts to converge, the difference between evaporation rates becomes visible. With ρ = 1/n, the curve rises more steadily and keeps improving with a steeper-look than the larger evaporation value. For larger ρ values, the curves look smoother and converge earlier, but it seems like they do not reach as high as the smaller ρ setting.

Standard deviation is large at the start, shrinks as the algorithm climbs, but does not fully stabilize, reflecting some variability in how different runs escape deceptive traps.



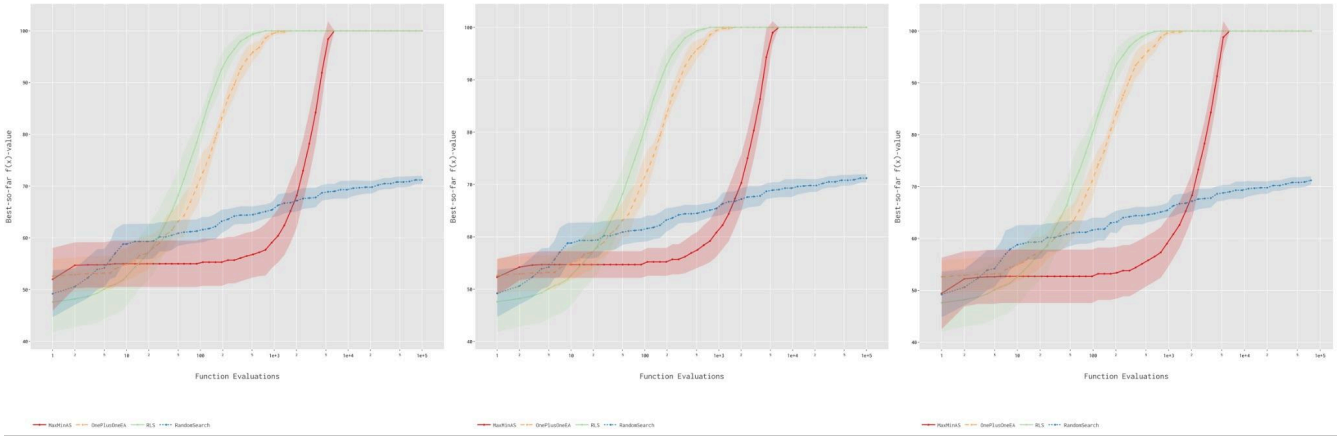
MMAS with F25*

On F25, the three MMAS* variants show very similar shapes, with no major differences between evaporation rates. Compared to the algorithms from Exercise 2, MMAS* performs as well as the (1+1)EA and clearly better than RLS, which becomes stuck in a local optimum.

Although MMAS* climbs slower at first, it eventually and finally reaches the same final result as the (1+1)EA, which demonstrates a reliable long term convergence.

The standard deviation is large for the small-ρ and medium-ρ settings in the early stages but shrinks steadily as the runs converge. For the largest ρ, the variance is smaller throughout, reflecting more consistent but less exploratory search behavior.

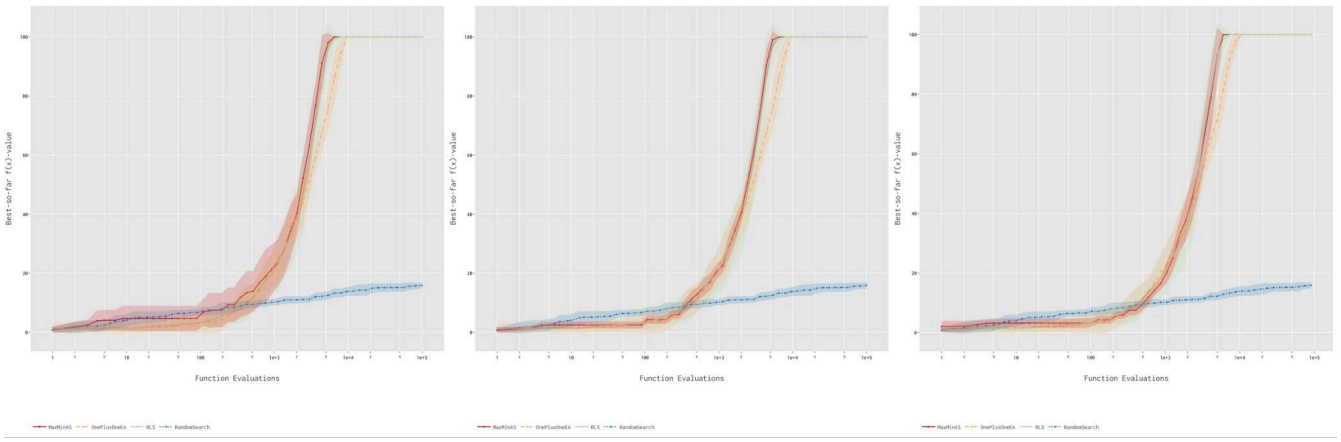
MMAS



MMAS with F1

There are not many significant differences between MMAS with ρ = 1/n and ρ = 1. However, the significant difference is with MMAS using ρ = 1/√n, particularly in the standard deviation: the other two have larger initial standard deviations that shrink slowly, but the one with ρ = 1/√n has a smaller initial standard deviation while still shrinking slowly. This shows that for all MMAS variants with different ρ values, there is stable convergence toward the optimum. After all, the overall performance across different ρ values is the same, so the analysis here applies to all of them. Random Search performs the worst here, after the 10th evaluation, it turns into linear growth and doesn't converge to the optimum yet. However, its standard deviation shows shrinkage, suggesting that with enough evaluations, it would reach the optimum, but not in this case. MMAS is worse than both (1+1)EA and RLS, the other two perform significantly better compared to MMAS, starting to converge around 1000 evaluations, unlike MMAS.

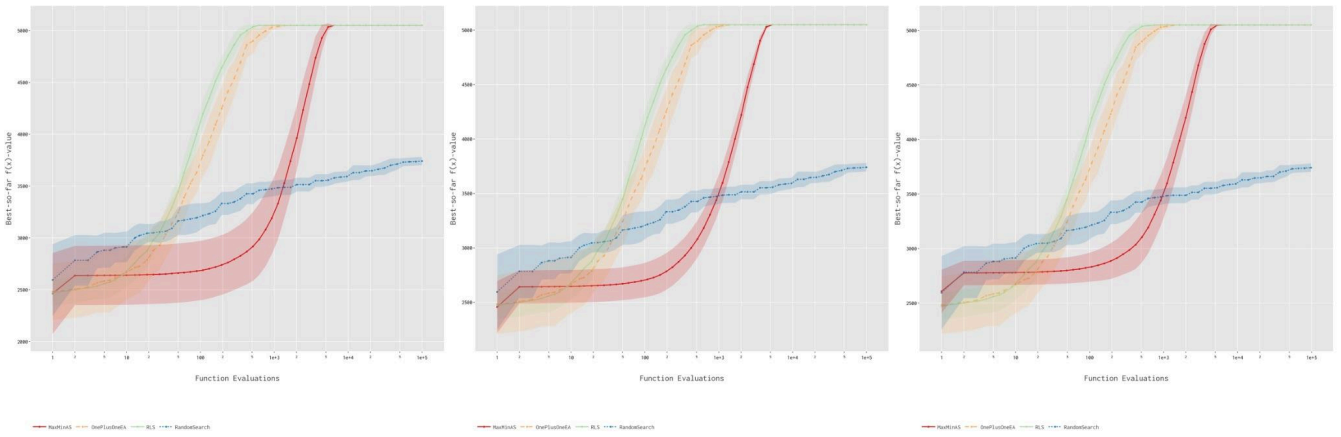
MMAS has a stagnation stage between 2 and 100 evaluations, where it shows no growth but just a straight line, reaching near optimum around 5000 evaluations. In contrast, the other algorithms exhibit consistent growth from start to end, even Random Search. Moreover, (1+1)EA and RLS show exponential growth, which is way better than MMAS. Overall, MMAS performs worse on average than (1+1)EA and RLS, and it is even worse than Random Search during MMAS's stagnation stage. Therefore, overall, MMAS is just better than Random Search.



MMAS with F2

There are not many significant differences between MMAS with ρ = 1/√n and ρ = 1. However, the significant difference is with MMAS using ρ = 1/n, particularly in the standard deviation: the other two have smaller initial standard deviations that shrink slowly, but the one with ρ = 1/n has a bigger initial standard deviation while still shrinking slowly. All four algorithms—MMAS, RLS, (1+1)EA, and Random Search—start at the same point on the Leading Ones problem. MMAS using ρ = 1/√n and ρ = 1 performs very close to RLS, and together with RLS, they both outperform (1+1)EA. Thus, in this problem, MMAS, RLS, and (1+1)EA exhibit similar exponential growth. The standard deviation is very small and consistent throughout the evaluation process.

However, from evaluations 1 to 100, MMAS is outperformed by Random Search, similar to the first OneMax problem. Regarding MMAS with ρ = 1/n, it exhibits the same growth rate but with a much higher standard deviation; however, from the 2000th evaluation onward, the three MMAS variants with different evaporation rates show nearly the same standard deviation. The standard deviation shrinkage fully occurs when it hits the optimum for all ρ values after 5000 evaluations. In this problem, MMAS performs as well as RLS and better than (1+1)EA.

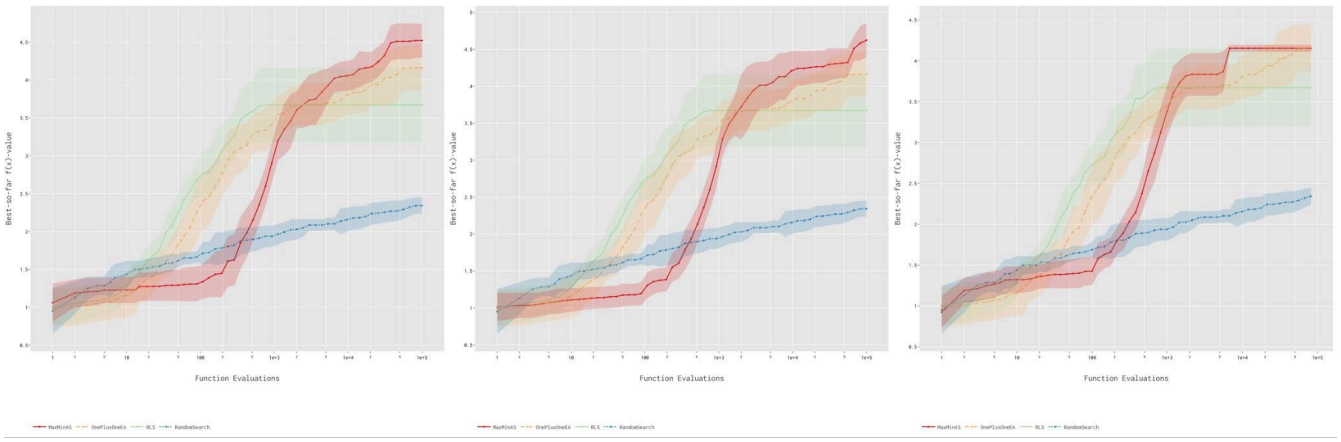


MMAS with F3

There are not many significant differences between MMAS with ρ = 1/√n and ρ = 1. However, the significant difference is with MMAS using ρ = 1/n, particularly in the standard deviation: the other two have smaller initial standard deviations that shrink slowly, but the one with ρ = 1/n has a bigger initial standard deviation while still shrinking slowly.

In this Harmonic Weight Linear Function problem, MMAS performs very poorly during the initial 1000 evaluations; it even performs worse than Random Search. RLS and (1+1)EA exhibit early exponential growth, which allows them to reach the optimum quickly around 1000 evaluations, whereas MMAS requires around 5000 evaluations. This can also be attributed to the stagnation stage in MMAS, where from 2 to 100 evaluations, MMAS shows no significant progress.

On the other hand, the starting points of MMAS for ρ = 1/n and 1/√n are almost the same as those of RLS and (1+1)EA. Random Search always has the highest initial starting point, but when using ρ = 1, MMAS can start at a point that is much higher than both RLS and (1+1)EA, which is almost the same as Random Search. MMAS still exhibits exponential growth later, similar to RLS and (1+1)EA, from 200 to 5000 evaluations, where it reaches the optimum.

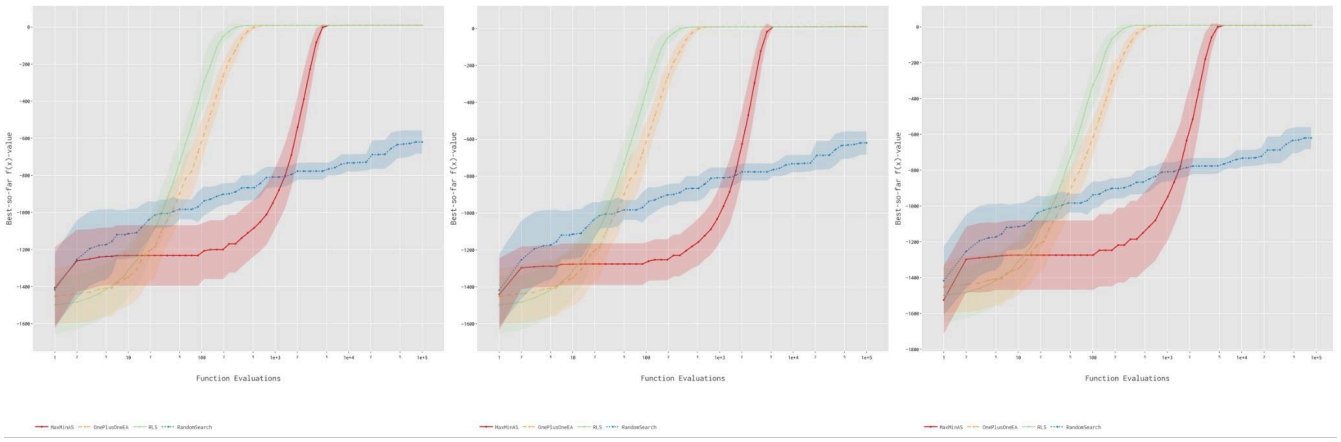


MMAS with F18

There are not many significant differences between MMAS with ρ = 1/√n and ρ = n. However, the significant difference is with MMAS using ρ = 1, particularly in the standard deviation: the other two have smaller end standard deviations that shrink slowly, but the one with ρ = 1 has a significantly bigger end standard deviation while still shrinking slowly.

In this Low Autocorrelation Binary Sequences (LABS) problem, MMAS outperforms RLS and (1+1)EA in the later stages, particularly in the end stage where, after 1000 evaluations, MMAS grows quicker than both RLS and (1+1)EA. Especially with ρ = 1, from evaluation 5000 to the end, it seems to reach the optimum, as shown by a straight flat line. In comparison, (1+1)EA reaches the optimum at around 10^4 evaluations with a high standard deviation, which may suggest that (1+1)EA is getting stuck in local optima.

At the initial starting point, MMAS performs as well as the other algorithms, such as RLS, (1+1)EA, and Random Search. But in the slow growth stage from 1 to 100 evaluations, MMAS exhibits linear growth that is even slower than Random Search's linear growth. MMAS later exhibits exponential growth in the quick growth stage from 100 to 5000 evaluations. From there to the end, MMAS turns back into linear growth but better than RLS and (1+1)EA. We can see that, compared to RLS and (1+1)EA, MMAS doesn’t have consistent growth, as it features two linear growth phases and an exponential growth phase. In contrast, those two have exponential growth at the beginning and then, after 5000 evaluations, turn into linear growth that is worse than MMAS.

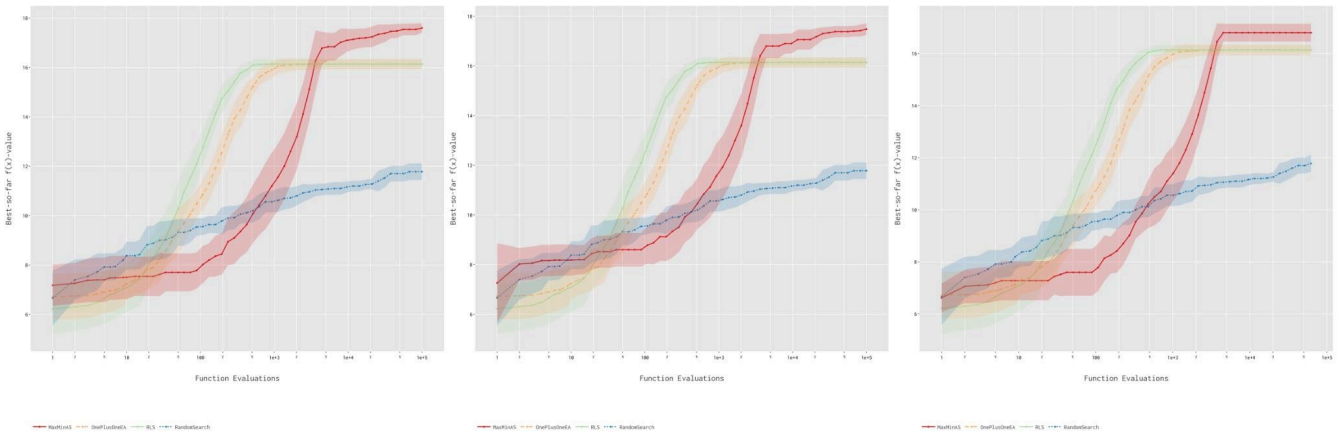


MMAS with F23

There are not many significant differences between MMAS with ρ = 1/n and ρ = 1. However, the significant difference is with MMAS using ρ = 1/√n, particularly in the standard deviation: the other two have bigger initial standard deviations that shrink slowly, but the one with ρ = 1 has a smaller initial standard deviation while still shrinking slowly.

In this N-Queen Problem, MMAS performs very poorly during the initial 1000 evaluations; it even performs worse than Random Search. RLS and (1+1)EA exhibit early exponential growth, which allows them to reach the optimum quickly around 1000 evaluations, whereas MMAS requires around 5000 evaluations. This can also be attributed to the stagnation stage in MMAS, where from 2 to 100 evaluations, MMAS shows no significant progress.

On the other hand, the starting points of MMAS for ρ = 1/n and 1/√n are almost the same as those of RLS and (1+1)EA. Random Search always has the highest initial starting point, but when using ρ = 1/n, MMAS can start at a point that is much higher than both RLS and (1+1)EA, which is almost the same as Random Search. MMAS still exhibits exponential growth later, similar to RLS and (1+1)EA, from 200 to 5000 evaluations, where it reaches the optimum.

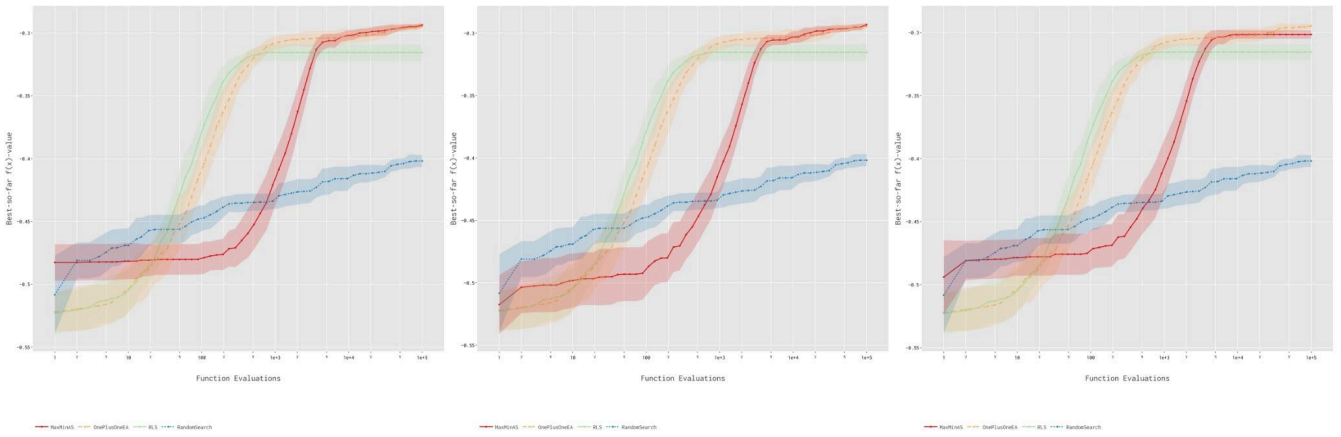


MMAS with F24

In this F24: Concatenated Trap problem, the starting point for MMAS when using ρ = 1/n and ρ = 1/√n is way higher than all of the other algorithms, except for when using ρ = 1, where the starting point is a bit lower and almost the same as Random Search.

However, after initialization, MMAS suffers from a slow linear growth stage until the 100th evaluation. Despite that, MMAS still manages to yield better results than RLS and (1+1)EA on average for all ρ values between evaluations 1 and 10. But after that, the exponential growth of RLS and (1+1)EA overtakes MMAS.

The exponential growth of MMAS really hits after the 100th evaluation until the 5000th evaluation, where MMAS enters its exponential growth stage. It grows so fast that it even surpasses RLS and (1+1)EA, reaching a higher optimum. As for ρ = 1/n and ρ = 1/√n, it seems to continue with linear growth after the 5000th evaluation, but for ρ = 1, it suffers from local optima, as suggested by the growth of the other two ρ values and the standard deviation of this ρ. Significantly, MMAS with ρ = 1/n and ρ = 1/√n seems to find a better optimum, whereas RLS and (1+1)EA also suffer from the same local optima like ρ = 1.



MMAS with F25

There are not many significant differences between MMAS with ρ = 1 and ρ = 1/n. However, the significant difference is with MMAS using ρ = 1/√n, particularly in the standard deviation: the other two have smaller initial standard deviations that shrink slowly, but the one with ρ = 1/√n has a significantly bigger initial standard deviation while still shrinking slowly.

In the F25: NK Landscapes (NKL) problem, both MMAS using ρ = 1 and ρ = 1/n have the highest starting points compared to the other algorithms. But both also suffer from a stagnation stage during evaluations 1 to 100. Even in such a stage, between evaluations 1 and 10, it is still better than RLS and (1+1)EA; however, after that, the exponential growth of RLS and (1+1)EA kicks in, and MMAS loses to those two. However, for those two ρ values, the exponential growth kicks in after evaluations 100 to 5000, where ρ = 1/n beats RLS after the 5000th evaluation and hugs (1+1)EA while having gradual linear growth to the optimum. On the other hand, ρ = 1 suffers from local optima, where it seems to stop growing and falls behind (1+1)EA but still outperforms RLS.

Regarding MMAS using ρ = 1/√n, the stagnation stage between evaluations 1 and 100 is not as linear compared to the other ρ values; it still shows gradual and insignificant linear growth. But after evaluations 100 to 5000, the behavior is similar to MMAS using ρ = 1/n.

Conclusion

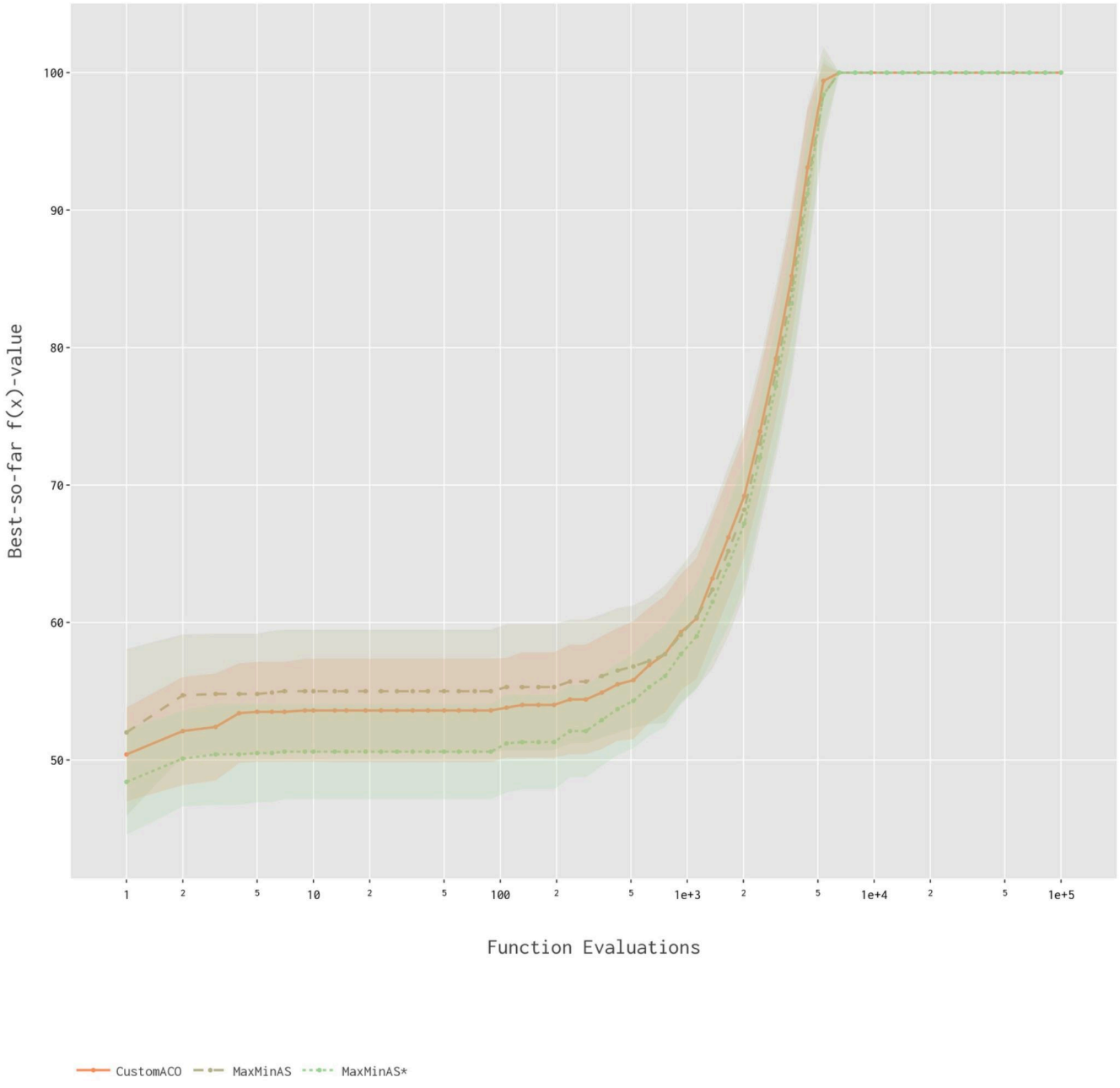
The evaporation parameter ρ primarily controls how much the algorithm explores and exploits. Small ρ values, like 1/n, update the pheromone slowly, encouraging more exploration. Therefore, it leads to a later but sharper, steeper climb to the optimum. On the other hand, with medium and large ρ values (1/√n or 1) show more exploitation, it reinforces good solutions faster, leading to earlier and smoother convergence.

On unimodal problems like F1, all ρ values eventually reach the optimum, but larger ρ generally allows faster convergence.

On more complex landscapes with plateaus or deceptive regions, smaller ρ would help avoid premature convergence, while larger ρ could cause the algorithm to lock into suboptimal solutions prematurely. Overall, tuning ρ is a key to balance the exploration and exploitation in MMAS-based algorithms, yet, we still need to consider the nature of the problem.

MMAS* behaves like MMAS but slightly more exploitative as it only accepts the strictly better solutions, so it often converges marginally earlier. But this also has the cost of higher risk of premature locking (earlier convergence) on harder landscapes.

Question 5: ACO-MMAS-MMAS*
Parameter:
- Number of ants: 10
- Rho = 0.01 for all.
- Distinct for our own ACO:
- **Apply the local search probability: 0.6** - probability of applying local search on a solution
- **Top ants rate: 0.2** - fraction of best ants will be used to update pheromone, instead of only 1 best ant reinforcing.



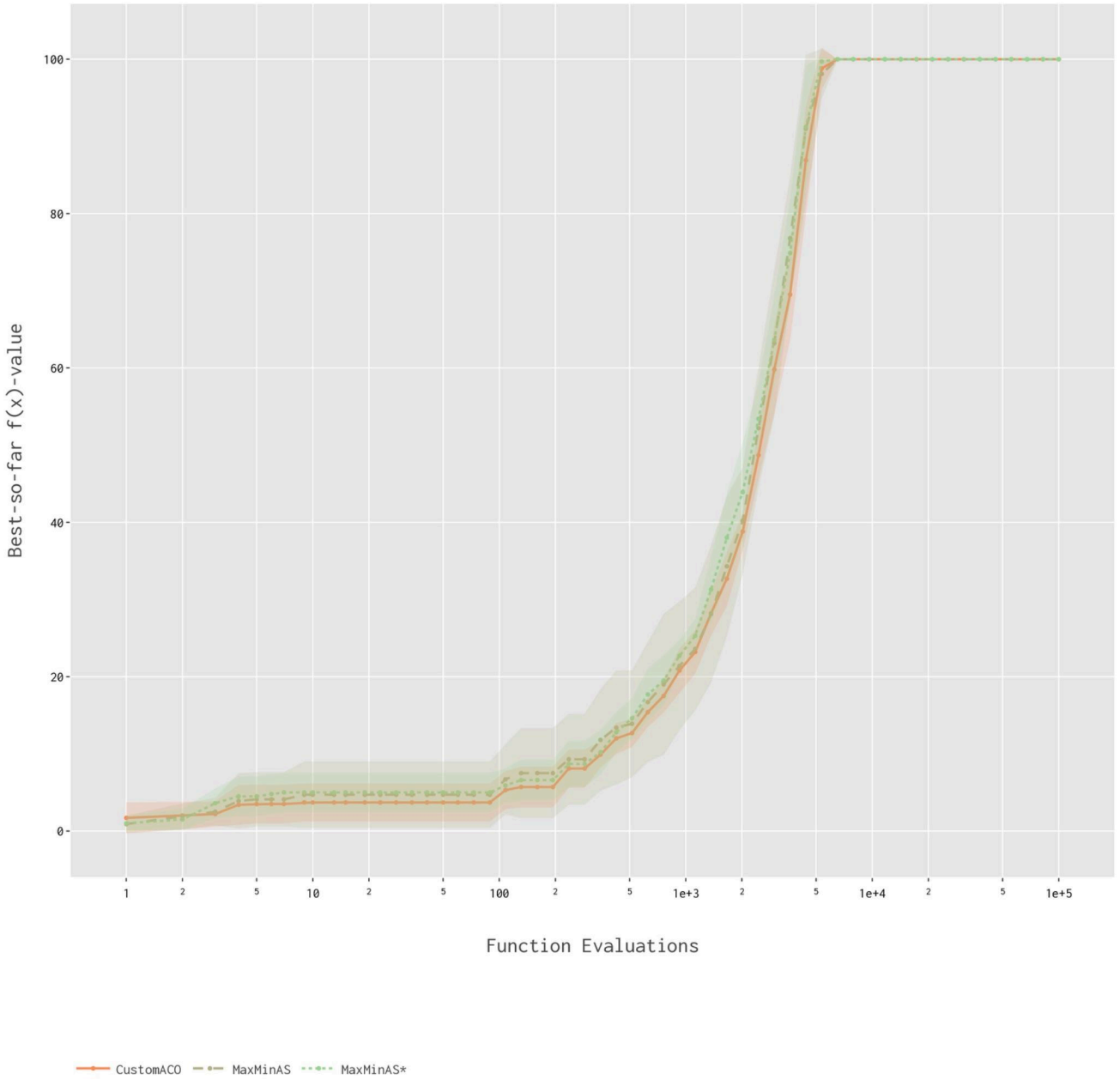
ACO-MMAS-MMAS with F1*

From the plot, we can see that MMAS and MMAS* form an upper bound and a lower bound for ACO, respectively. All three algorithms start with around 50 ones and begin to stagnate from the 5th evaluation until around the 500th evaluation. After that, they rise sharply to reach the optimum around

the 500th evaluation. Since the OneMax problem is unimodal, flipping bits to ones brings the algorithm closer to the maximum. This suggests that, theoretically, for OneMax, there won't be huge bumps, which is consistent with the evaluation graph.

I suggest that the stagnation is caused by the low ρ value because, with a low ρ, the exploration rate is low, and old paths are kept being used over and over again. This creates huge plateaus along with expensive local search executions. ACO doesn't suffer from extensive stagnation like MMAS and MMAS* because it only runs *local_search* with a probability of 0.6. However, after 500 evaluations, ACO starts to take off and outperforms both MMAS and MMAS* thanks to its probabilistic nature. MMAS* performs the worst because evaporation occurs only once per iteration, and it is strict to the best-so-far only if improved. This intensifies early good solutions but adapts slowly overall, as shown by the fact that MMAS* starts to catch up with MMAS after 500 evaluations.

Standard deviations shrink over time for all algorithms, indicating stable convergence. The standard deviation shrinkage in ACO shows a bit more fluctuation than the others, which demonstrates the stochasticity of the local search implemented in ACO.



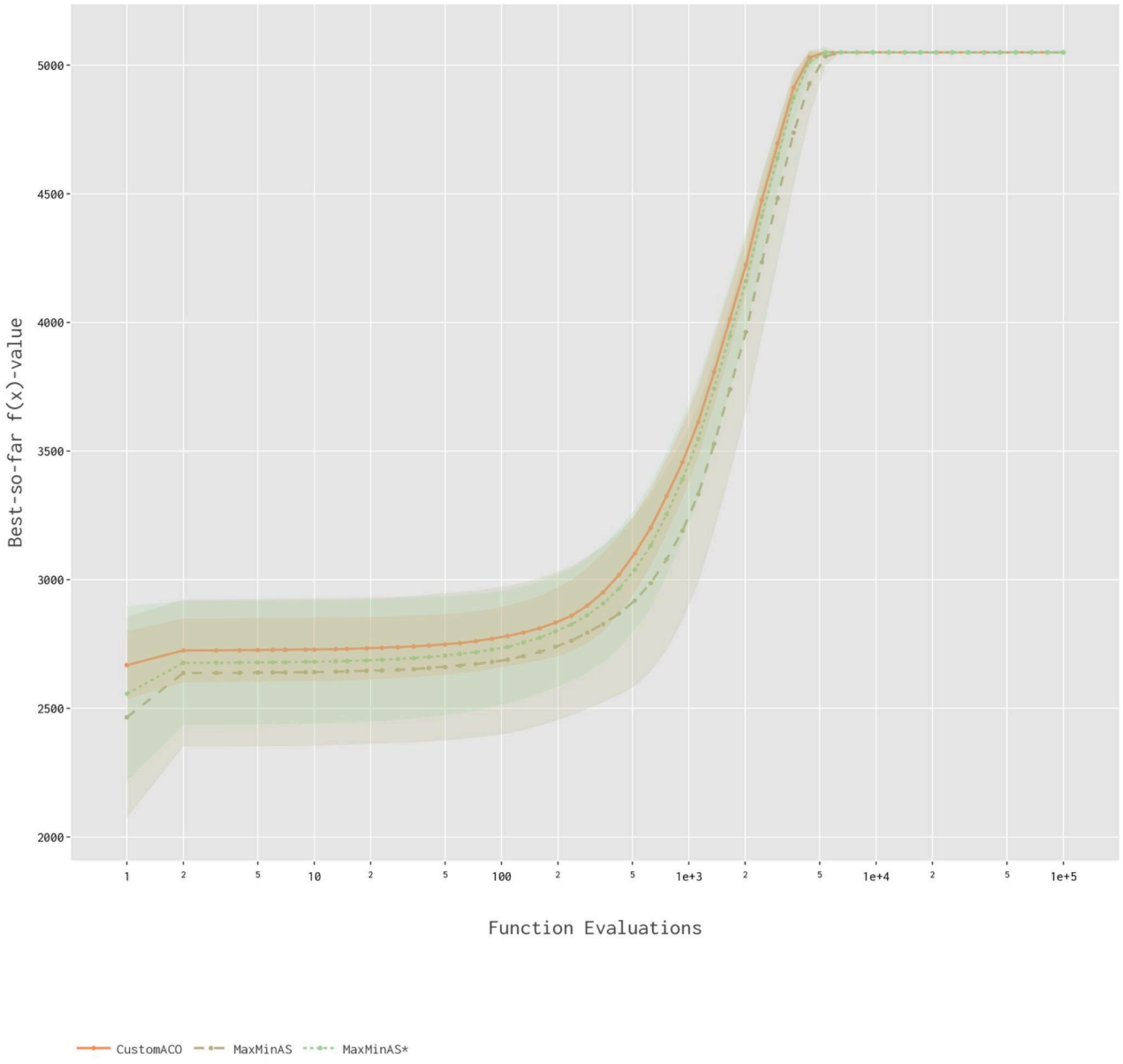
ACO-MMAS-MMAS with F2*

From the plot, we can see that MMAS and ACO form an upper bound and a lower bound for MMAS*, respectively. All three algorithms start very low at just over 0, around 1-2 leading ones, and begin to stagnate after less than 5 evaluations until around the 100th evaluation. After that, they rise sharply to reach the optimum around the 500th evaluation. Since the LeadingOnes problem is unimodal, flipping the first 0 to a 1 brings the algorithm closer to the maximum. This suggests that, theoretically, for LeadingOnes, there won't be huge bumps, which is consistent with the evaluation graph.

I suggest that the stagnation is caused by the low ρ value because, with a low ρ, the exploration rate is low, and old paths are kept being used over and over again. This creates huge plateaus along with expensive local search executions. ACO suffers from extensive stagnation more than MMAS and MMAS* because it only runs local search with a probability of 0.6, which means many ants contribute unoptimized, low-fitness solutions that dilute pheromone updates on this problem where precise prefix extensions are crucial. However, between 100 and 500 evaluations, MMAS* starts to take off and sharply increases way better than MMAS, potentially due to a performance outlier where its best-so-far

captures a rare breakthrough. MMAS performs the best overall because its effective higher evaporation allows faster adaptation and focused updates, while MMAS* adapts slowly with true low ρ and strict best-so-far rules, though it catches up in bursts as shown after 100 evaluations.

Standard deviations shrink over time for all algorithms, indicating stable convergence. The standard deviation shrinkage in ACO shows a bit more fluctuation than the others, which demonstrates the stochasticity of the local search implemented in ACO.



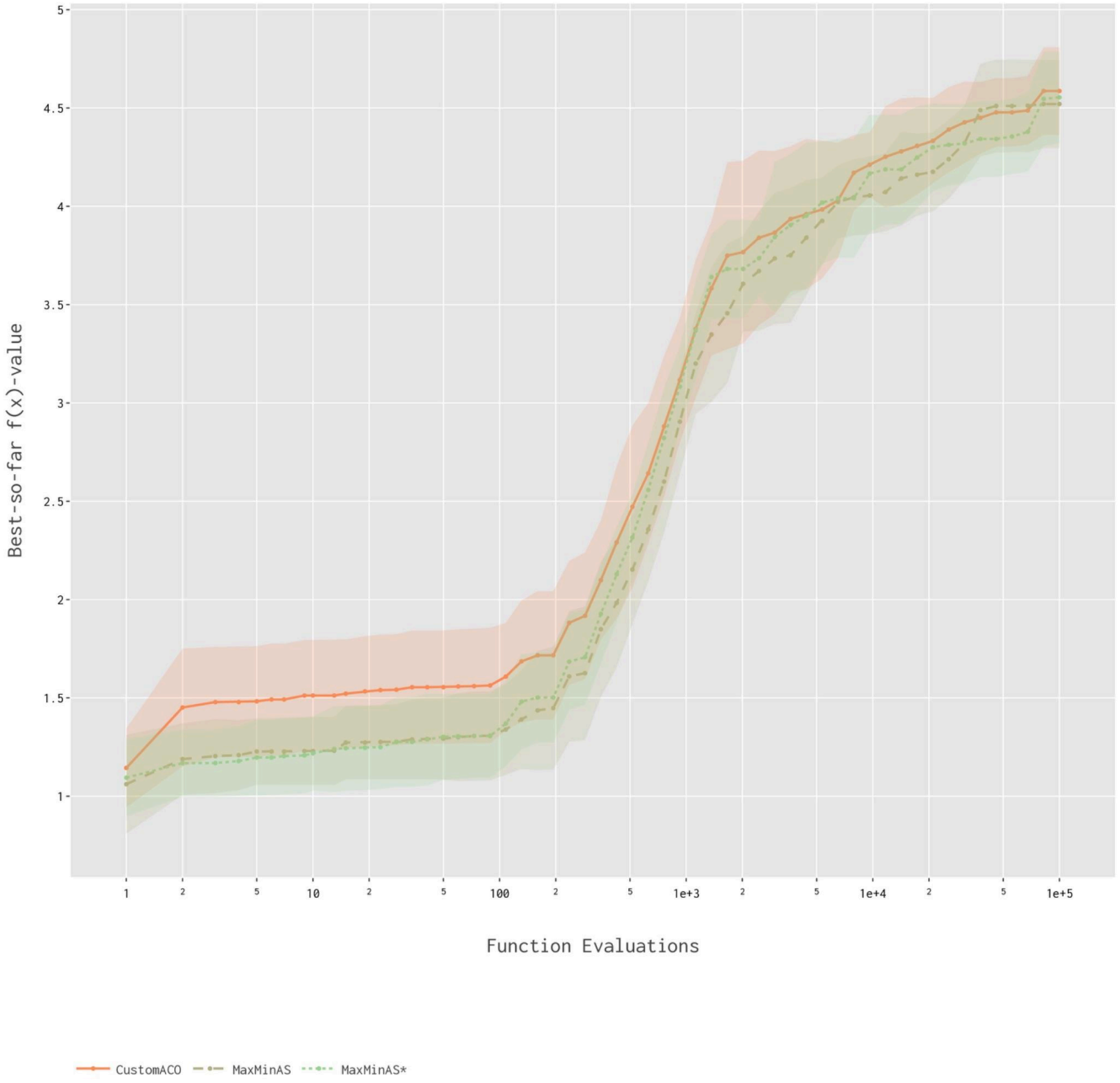
ACO-MMAS-MMAS with F3*

From the plot, we can see that ACO and MMAS form an upper bound and a lower bound for MMAS*, respectively. All three algorithms start at a very high value around 2500. They begin to stagnate from the 2nd evaluation until the 50th evaluation, but with a slight slope suggesting insignificant growth. After the 50th evaluation, they rocket to the optimum in the range of 50 to 10,000 evaluations; this suggests that the sharp growth is steadier than in other problems. Since the Linear problem (maximizing the function \sum w_i x_i with linear weights) is unimodal and very similar to OneMax but with harmonic multipliers on each variable making growth easier to observe, there are theoretically no huge bumps, which is consistent with the evaluation graph.

I suggest that the stagnation is caused by the low ρ value because, with a low ρ, the exploration rate is low, and old paths are kept being used over and over again. This creates huge plateaus along with expensive local search executions. ACO doesn't suffer from extensive stagnation like MMAS and MMAS* because it only runs local search with a probability of 0.6, allowing more iterations and updates within the budget while its multiple ants updates provide diversity to bias toward high weight bits. However, after 50 evaluations, all algorithms start to take off, with ACO outperforming the others thanks to its probabilistic nature and spread of reinforcements. MMAS performs the worst because its

effective higher evaporation may cause overdecay of pheromone memory on this weighted problem, leading to slower adaptation compared to MMAS*, which balances intensification on best-so-far with true low ρ and catches up steadily as biases toward high weight 1s accumulate.

Standard deviations shrink over time for all algorithms, indicating stable convergence.



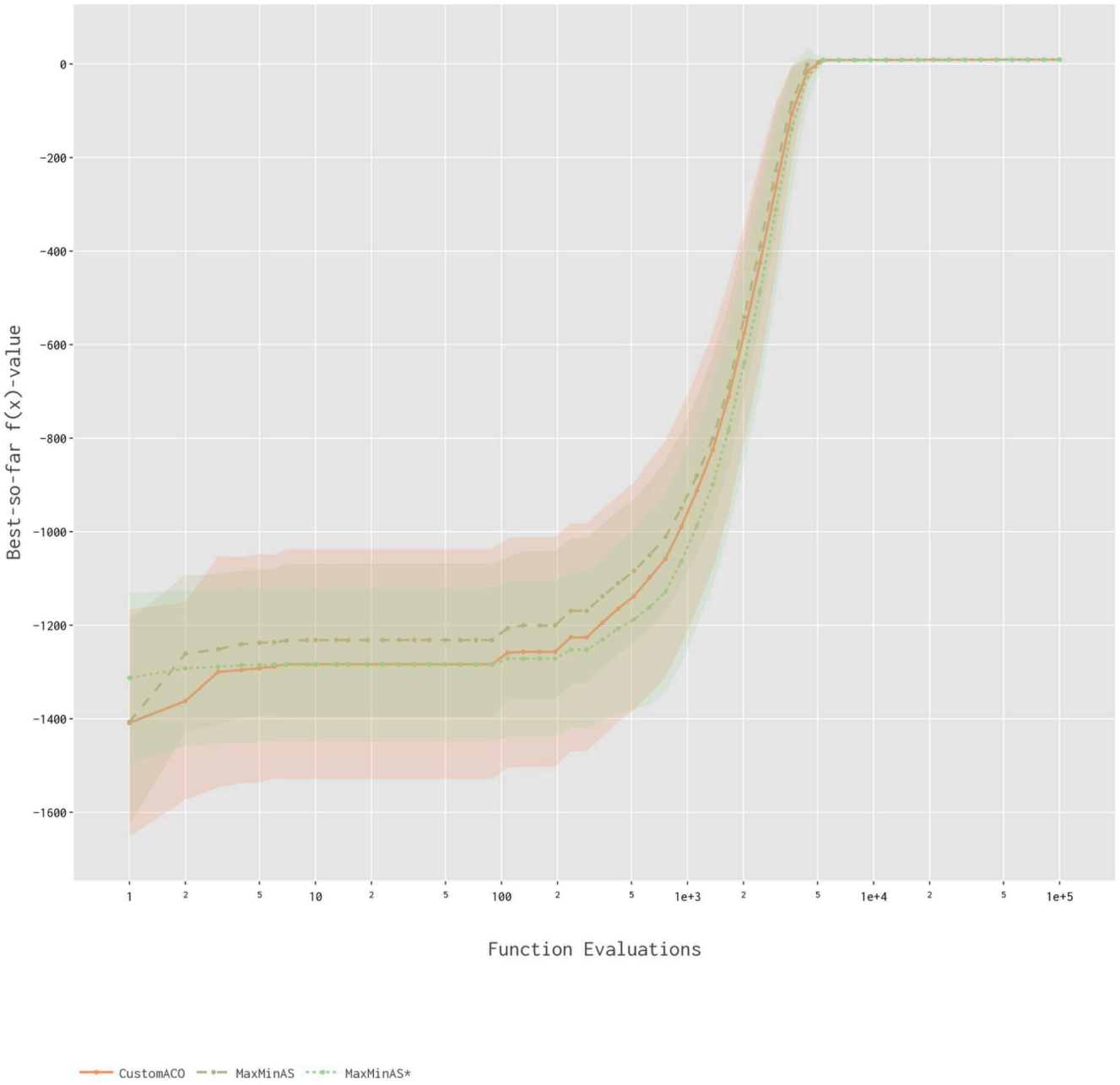
ACO-MMAS-MMAS with F18*

From the plot, we can see that it has a widened integral symbol shape. In contrast to the previous problems, none of the three algorithms reach the optimum. Additionally, in the high-value and high-evaluation range from 1000 onward, the algorithms are always growing, but after prolonged evaluations, they are supposed to reach the optimum with less fluctuation; however, in this case, the fluctuation is intensified. All three start very low at around 1. Here, the stagnation is less clear but replaced with a gradual increase between evaluations 2 and 100, followed by a sharp increase from 100 to 1000. In the first gradual growth segment, ACO outperforms the others, while MMAS and MMAS* hug each other closely. But in the second fast-growing segment, MMAS* rises up and beats ACO. In the fluctuating segment, the three fluctuate between each other, with none seeming to be the

highest. Since the Low Autocorrelation Binary Sequences (LABS) problem is multimodal with a rugged landscape full of local optima, there are theoretically many bumps and plateaus, which is consistent with the evaluation graph showing gradual, sharp, and fluctuating phases.

I suggest that the less pronounced stagnation, which is replaced by gradual growth, is caused by the low ρ value because, with a low ρ, the exploration rate is low, and old paths are kept being used over and over again. This creates extended plateaus along with expensive local search executions, but on a multimodal problem like LABS, it allows slow pheromone buildup to gradually bias toward low-autocorrelation patterns without premature convergence to poor local optima. ACO outperforms early because it only runs local search with a probability of 0.6, allowing more iterations and updates within the budget while its multi-ant updates provide diversity to explore different sequence modes and escape initial traps. However, after 100 evaluations, all algorithms start to take off in the sharp phase, with MMAS* outperforming the others thanks to its best-so-far intensification, which focuses on promising low-autocorrelation foundations once initial bases accumulate. MMAS performs the worst overall because its effective higher evaporation may cause overdecay of pheromone memory on this tricky problem, leading to faster convergence to local optima compared to MMAS*, which balances intensification on best-so-far with true low ρ and catches up steadily as biases toward better sequences accumulate, though the end fluctuations show ongoing struggles with multimodality.

Standard deviations do not shrink over time for all algorithms, indicating that there is no stable convergence. Furthermore, the fluctuation of the standard deviation at the last segment indicates that the three algorithms are suffering from local optima and there won’t be convergence soon.



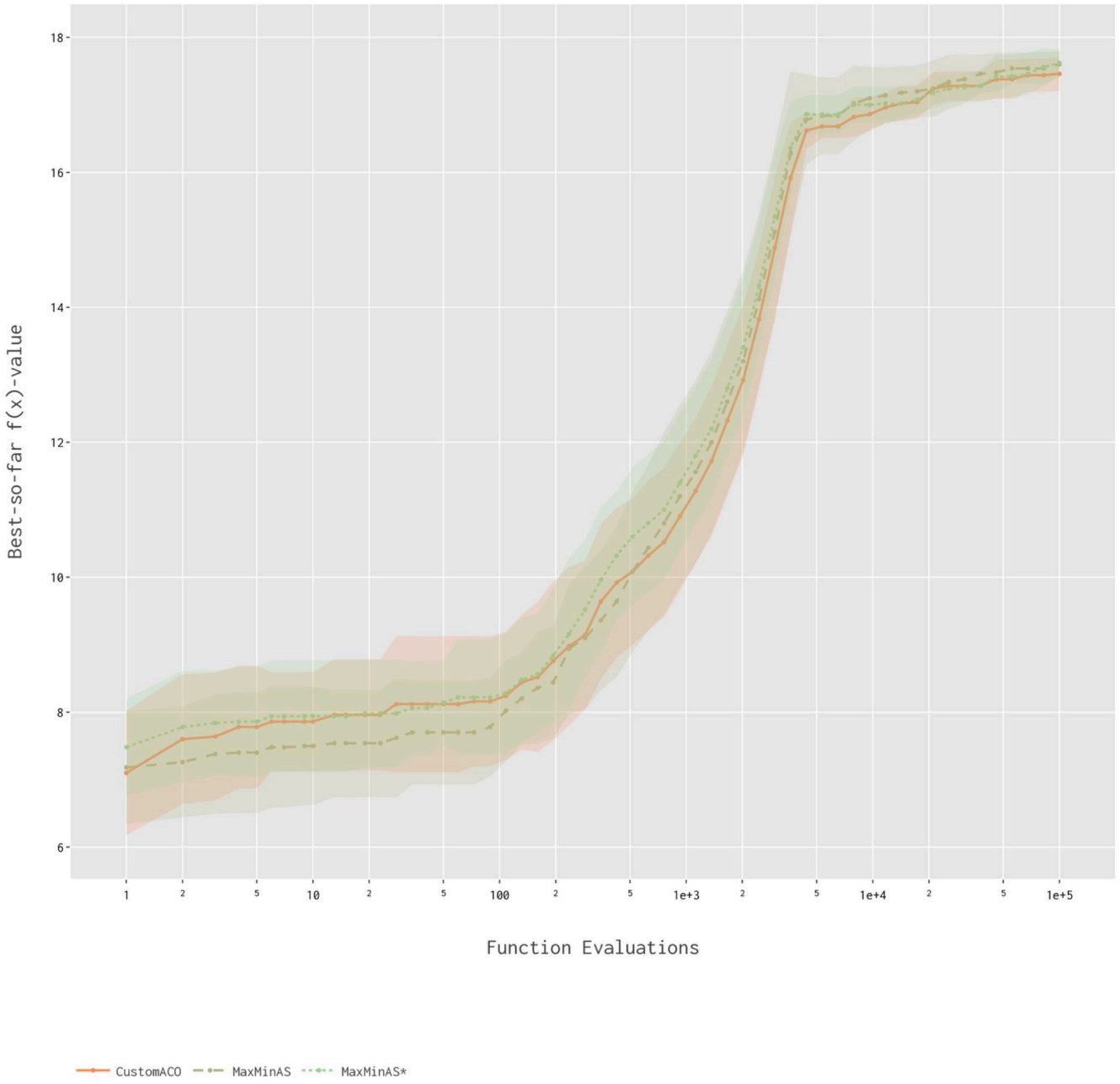
ACO-MMAS-MMAS with F23*

From the plot, we can see that the stagnation segment appears clearly again from the 5th evaluation until the 100th evaluation. Additionally, during stagnation, ACO performs as good as MMAS*, with their graph lines hugging each other closely. After the stagnation stage, from over 100 evaluations until 5000 evaluations where they reach the optimum after that, MMAS and MMAS* form an upper bound and a lower bound for ACO, respectively. Initially, all three algorithms start nearly at the same point. Since the N-Queens problem is multimodal with a highly constrained and tricky landscape full of local optima, there are theoretically many plateaus, which is consistent with the evaluation graph showing prolonged stagnation and followed by breakthroughs.

I suggest that the stagnation is caused by the low ρ value because, with a low ρ, the exploration rate is low, and old paths are kept being used over and over again. This creates huge plateaus along with expensive local search executions, particularly on a constrained problem like N-Queen where random constructions often violate many constraints, making local search inefficient and prone to getting stuck. ACO performs similarly to MMAS* during stagnation because its probabilistic local search of 0.6

chance and multiple ants updates dilute focus on poor initial pheromone, mirroring MMAS*'s slow adaptation with true low ρ. However, after 100 evaluations, all algorithms start to take off, with MMAS outperforming the others thanks to its effective higher evaporation, allowing faster pheromone resets and better exploration of queen placements to escape local optimas. MMAS* performs the worst overall because it balances intensification on best-so-far with true low ρ but adapts too slowly on this tricky problem, leading to prolonged trapping in local optima compared to ACO's diversity.

Standard deviations are intensified at the initial segment and the stagnation segment. In contrast, they also converge quickly, which seems to be happening extremely fast. From a wide standard deviation at the start to convergence at the end, this indicates that there is a stable optimum for the algorithms on this problem.



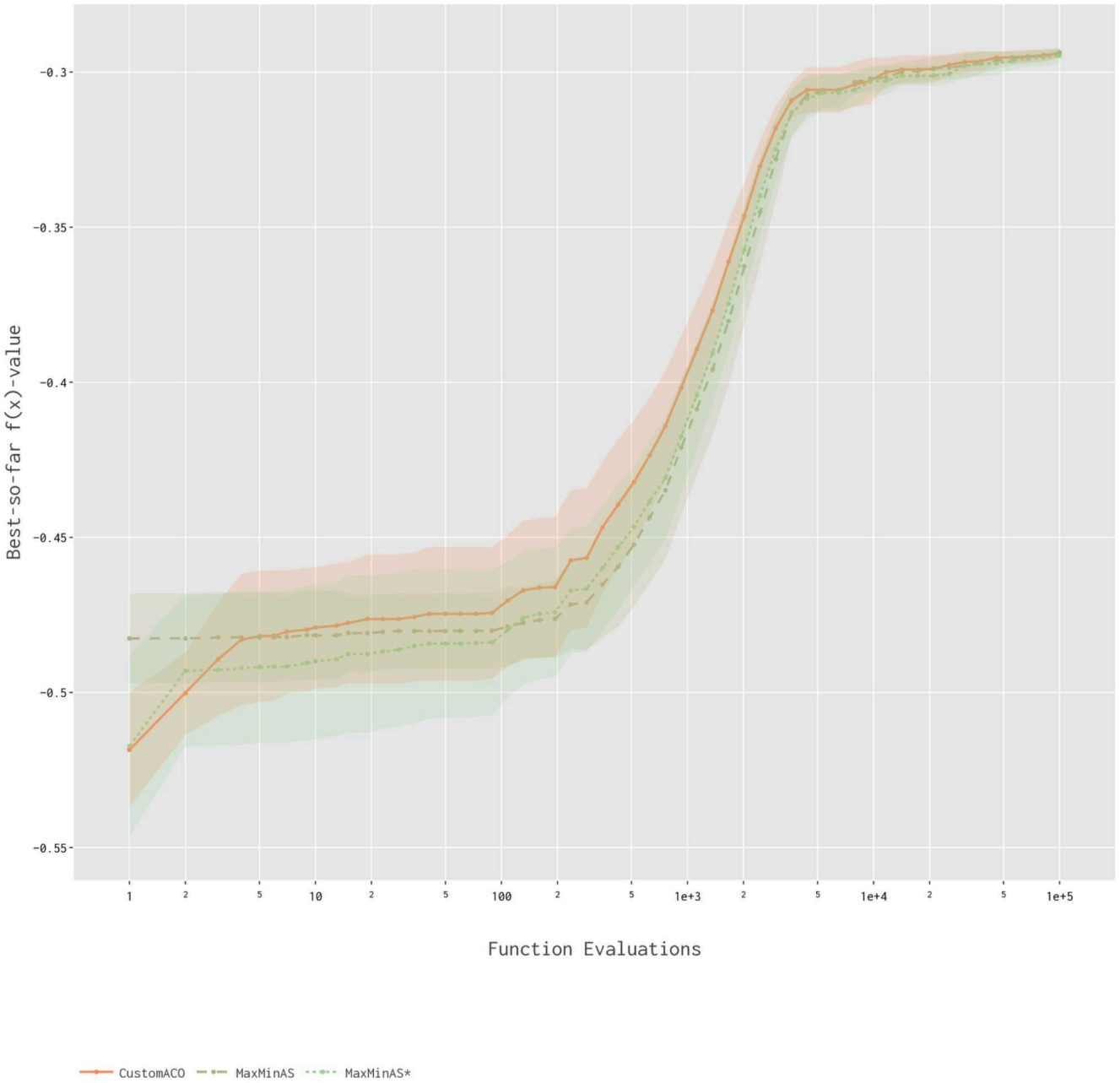
ACO-MMAS-MMAS with F24*

From the plot, we can see that it has a widened integral symbol shape, which is similar to the previous F18 problem (Low Autocorrelation Binary Sequences (LABS)). The big difference here is that not only

the near optimum segment is fluctuating but also the initial segment shows fluctuations. All three algorithms start at nearly the same value. From evaluations 1 to 100, the algorithms grow steadily, but the gradual growth has microfluctuations that happen very often. In this segment, ACO hugs MMAS* closely, while MMAS performs the worst. But in the quick-growing segment between over 100 and 5000 evaluations, MMAS gets better, outperforming ACO and positioning itself between MMAS* and ACO as its upper bound and lower bound, respectively. The graph shows that it doesn't reach the optimum, as indicated by the last segment that still exhibits gradual growth and thus some more fluctuations. Since the F24: Concatenated Trap problem is multimodal with a tricky landscape full of local optimas, there are theoretically many bumps and plateaus, which is consistent with the evaluation graph showing gradual, quick, and fluctuating phases.

I suggest that the less pronounced stagnation, which is replaced by gradual growth with microfluctuations, is caused by the low ρ value because, with a low ρ, the exploration rate is low, and old paths are kept being used over and over again. This creates extended plateaus along with expensive local search executions, but on a tricky problem like Concatenated Trap, it allows slow pheromone buildup to gradually escape local optimas without premature convergence. ACO hugs MMAS* early because its probabilistic local search with a 0.6 chance and multiple ants updates provide diversity to explore different block configurations and avoid deep traps initially, similar to MMAS*'s slow adaptation with true low ρ. However, after 100 evaluations, all algorithms start to take off in the quick phase, with MMAS* outperforming the others. MMAS performs the worst in the early segment because its effective higher evaporation may cause over-decay of pheromone memory on this tricky problem, leading to faster resets but initial struggles compared to MMAS*, which balances intensification on best-so-far with true low ρ and catches up steadily although the end fluctuations show ongoing struggles.

Standard deviations shrink over time for all algorithms, but very little, and they don't seem to shrink at the end of the segment, indicating that there is no stable convergence for the three algorithms.



ACO–MMAS–-MMAS with F25*

From the plot, we can see that it has a widened integral symbol shape. ACO and MMAS* start nearly at the same point around -0.52, but MMAS starts significantly higher at around -0.3 compared to the others. However, when it reaches the stagnation segment between 4 and 100 evaluations, MMAS gets outperformed by ACO, which has a quick rise between 1 and 4 evaluations that beats MMAS. There are microfluctuations in the stagnation stage and a very insignificant slope of growth. In the quick growth stage from over 100 to 5000 evaluations, ACO outperforms the two MMAS and MMAS*, where the latter two are hugging each other closely and ACO quickly rises above them. It seems like there are not enough evaluations for the three algorithms to reach the optimum. Furthermore, in the optimum stage, there are also microfluctuations. Since the F25: NK Landscapes (NKL) problem is multimodal with a rugged landscape full of local optimas, there are theoretically many bumps and plateaus, which is consistent with the evaluation graph showing initial rises, stagnation with microfluctuations.

I suggest that the stagnation, marked by microfluctuations and insignificant growth, is caused by the low ρ value because, with a low ρ, the exploration rate is low, and old paths are kept being used over and over again. This creates extended plateaus along with expensive local search executions, but on a rugged multimodal problem like NKL, it allows slow pheromone buildup to gradually navigate correlated variables and escape local optima without premature convergence to tricky basins. ACO quickly rises early and outperforms during stagnation because its probabilistic local search with a 0.6 chance and multiple ants updates provide diversity to explore different interaction configurations and avoid initial traps, outperforming MMAS's initial higher start which gets bogged down. However, after 100 evaluations, all algorithms start to take off in the quick phase, with ACO continuing to outperform the others thanks to its spread of reinforcements that better handles NKL's correlations. MMAS performs worse than expected early because its effective higher evaporation may cause over decay of pheromone memory on this deceptive problem, leading to faster resets but struggles in building stable biases compared to MMAS*, which balances intensification on best-so-far with true low ρ and hugs MMAS closely, though the microfluctuations near the optimum show ongoing challenges with ruggedness.

Standard deviations of the three algorithms shrink steadily over time, and they seem to be reaching a stable convergence, but because of not enough evaluations, the shrinkage is not fully fulfilled.