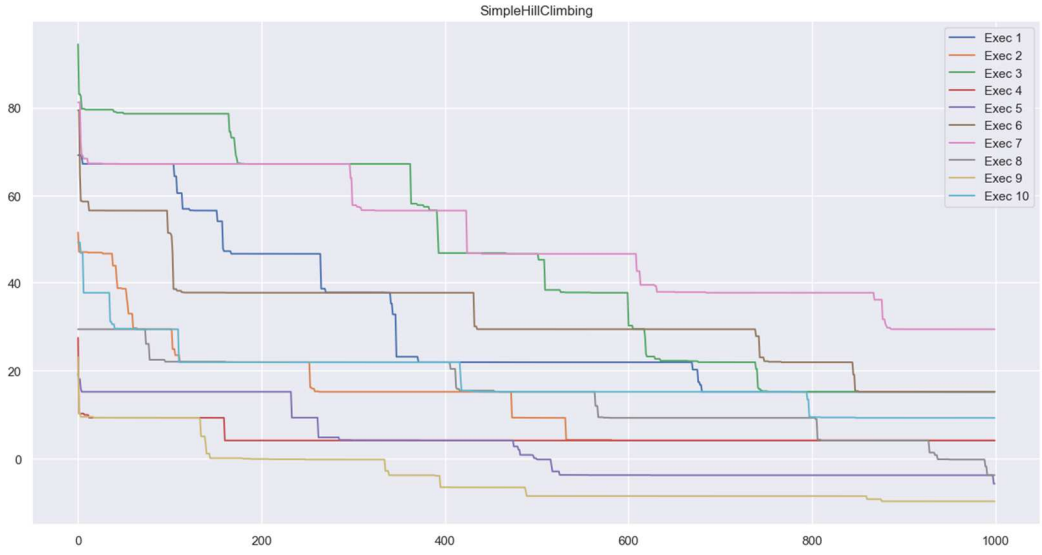
Projeto de Programação - Otimização

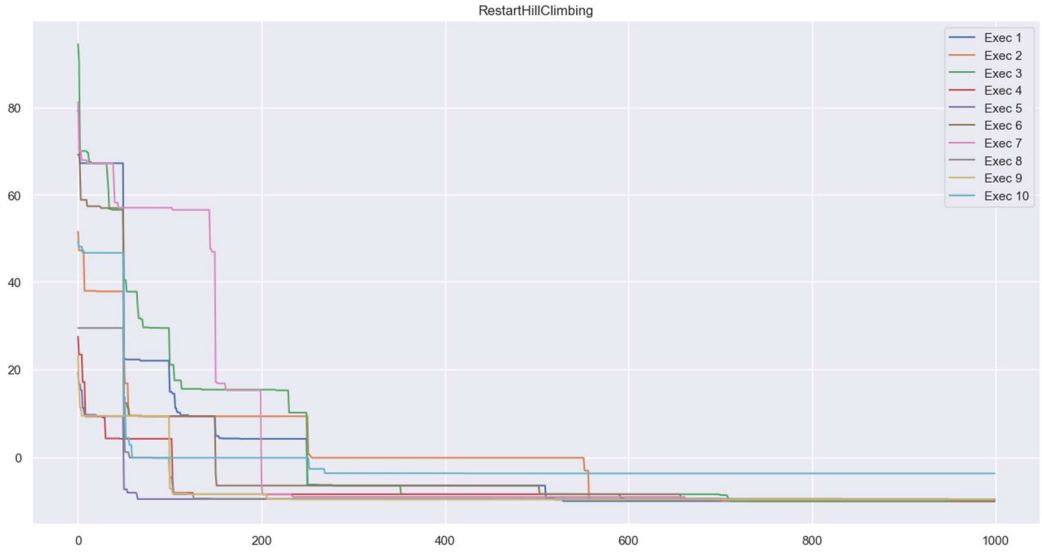
Problema 1:

Minimizar a função objetivo $f(x) = \frac{x^2}{100} + 10\,sin\left(x - \frac{\pi}{2}\right)$ *no intervalo [-100, 100]*

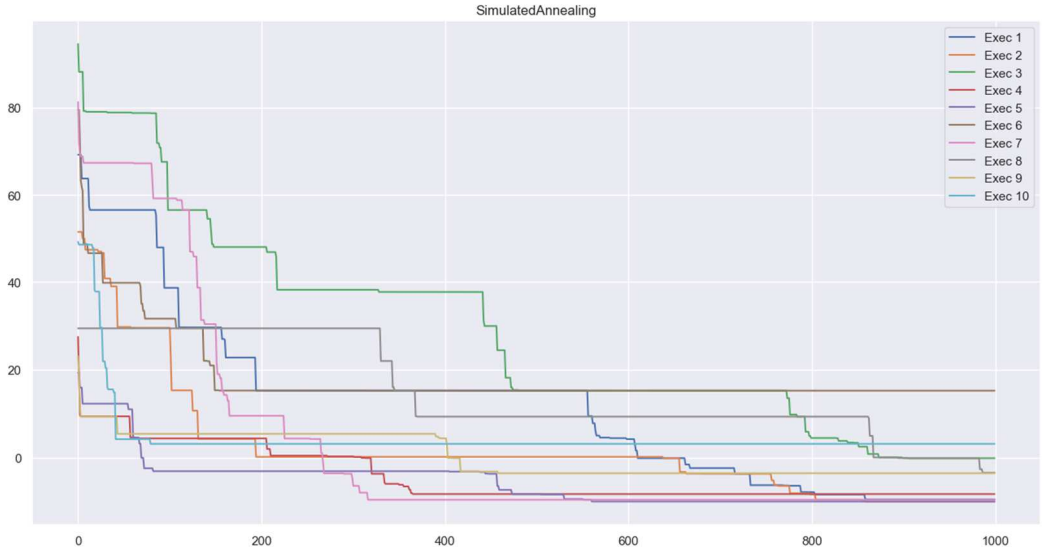
HillClimbing simples:



HillClimbing com restart:



Simulated Annealing:



Genetic Algorithm:

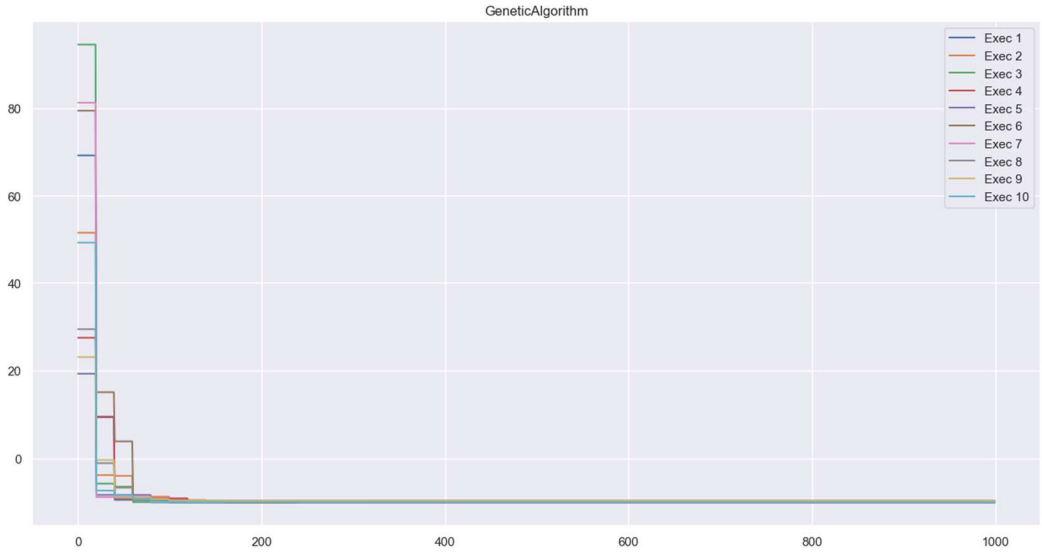
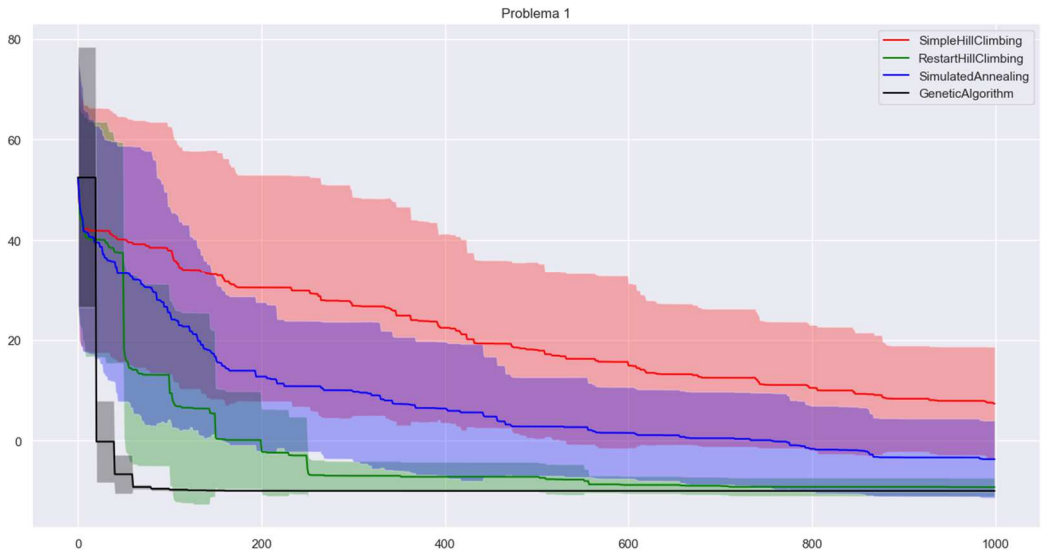


Tabela comparativa:

Algoritmo	Max	Min	Média	Desvio Padrão
HillClimbing simples	94.49527992891872	-9.605934063410883	20.603600536163665	21.119527468109293
HillClimbing c/ restart	94.49527992891872	-9.999664241669759	-3.2635857543627784	14.390050531121833
Simulated Annealing	94.49527992891872	-9.998021496897998	7.270527998635835	17.502610349244865
Genetic Algorithm	94.49527992891872	**-10.0**	9.62485322774159	9.62485322774159

Comparativo dos algoritmos:



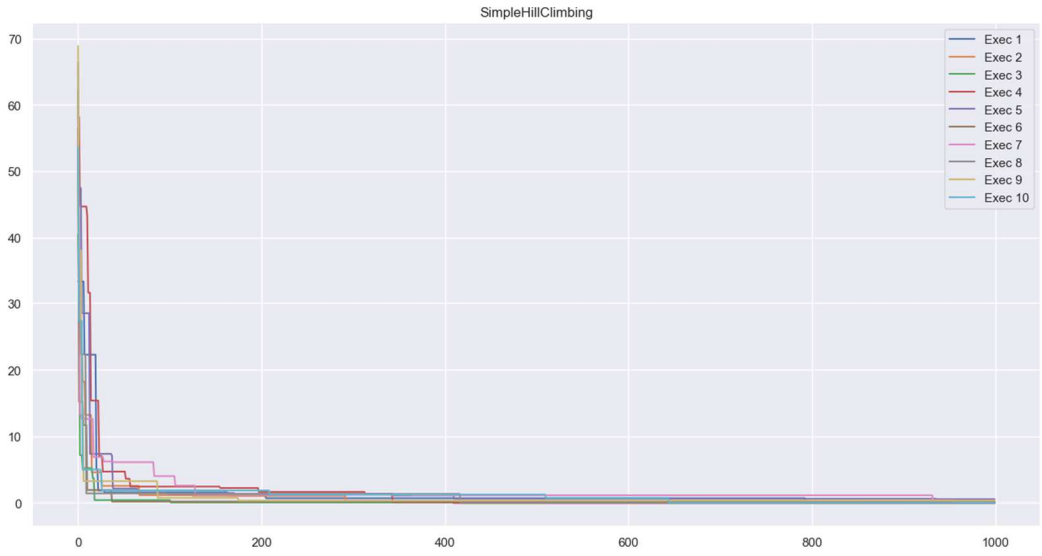
Discussão:

Todos os algoritmos apresentaram bom desempenho para o problema 1. O HillClimbing simples apresentou dificuldades para superar alguns mínimos locais com a distribuição normal de desvio padrão 2.0, mas ainda sim venceu muitos. O HillClimbing com restart explorou várias partes do intervalo válido, o que proporcionou boas oportunidades para encontrar soluções melhores. O Simulated Annealing também explorou partes do intervalo, só que próximas da mesma vizinhança inicial. Para este problema, o alto valor para aceitar vizinhos piores fez com que o Simulated Annealing explorasse regiões não promissoras, subindo na curva. O Genetic Algorithm, para o problema 1, apresentou bom desempenho devido a ser um algoritmo populacional com a oportunidade de “exploração paralela” de vários vizinhos, que foi favorecida pela função do problema. Ele encontrou o ótimo global.

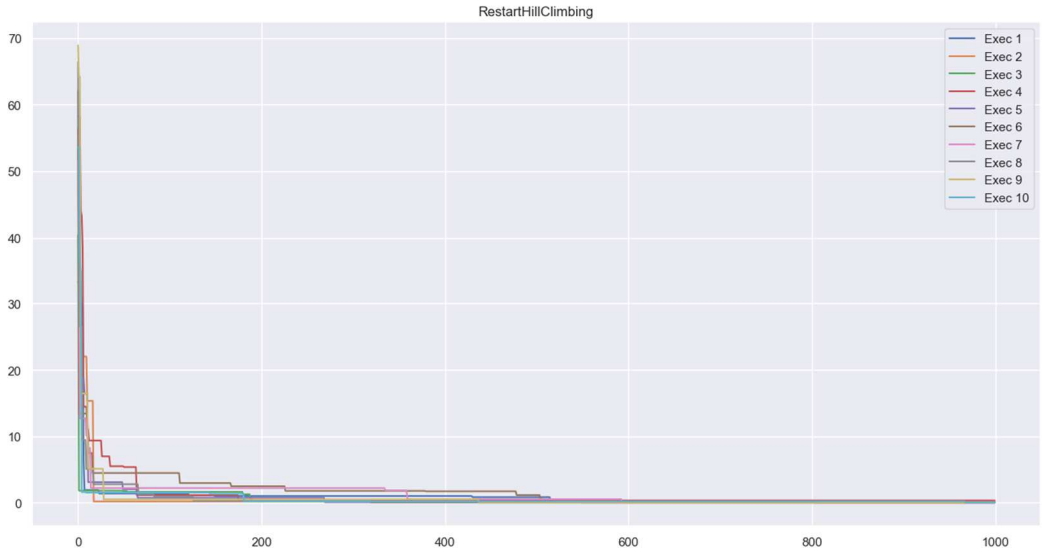
Problema 2:

Minimizar Rastrigin: $f(x,y) = 20 + (x^2 - 10\,cos(2\pi x) + (y^2 - 10\cos(2\pi y))$ *em [-5.12, 5.12]*

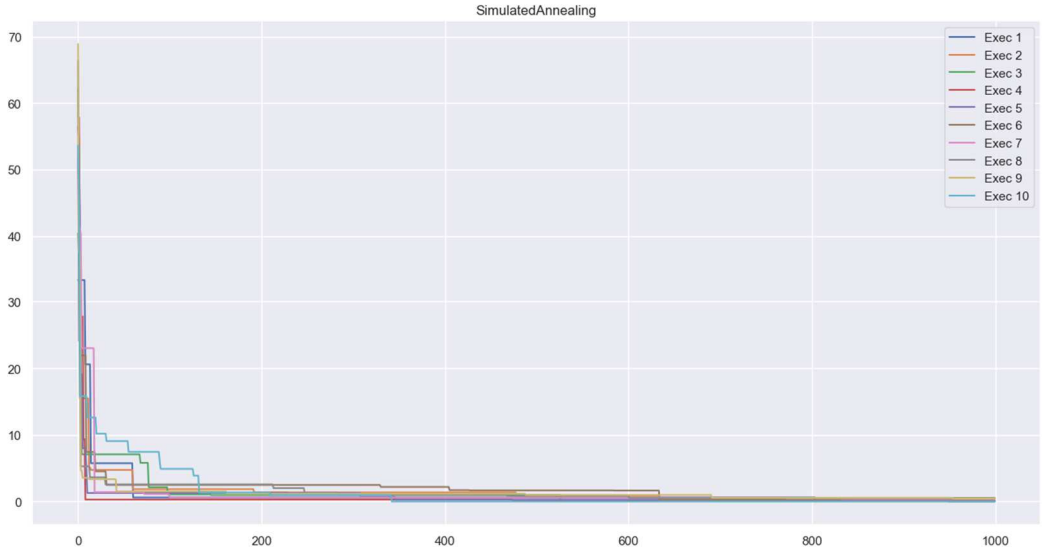
HillClimbing simples:



HillClimbing com restart:



Simulated Annealing:



Genetic Algorithm:

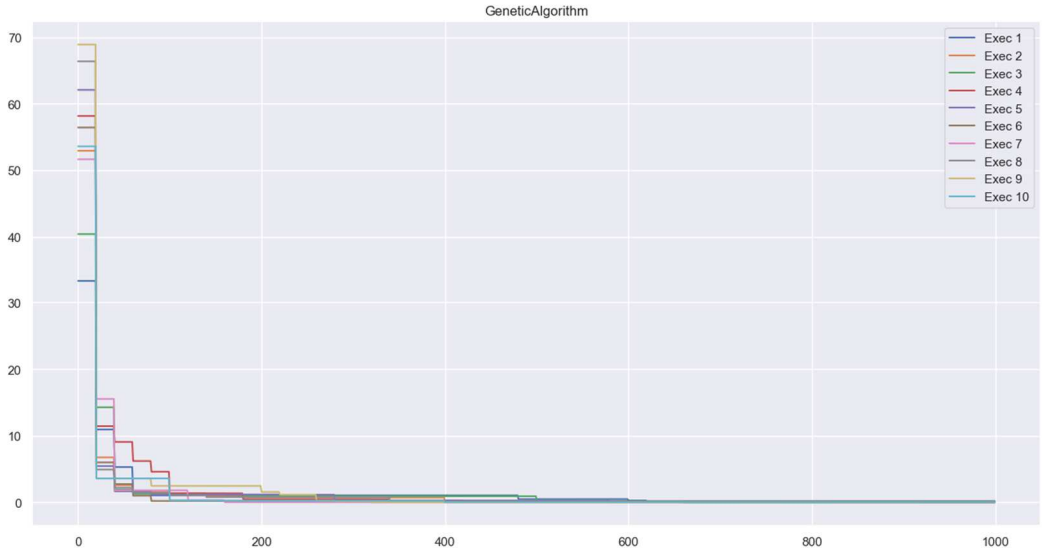
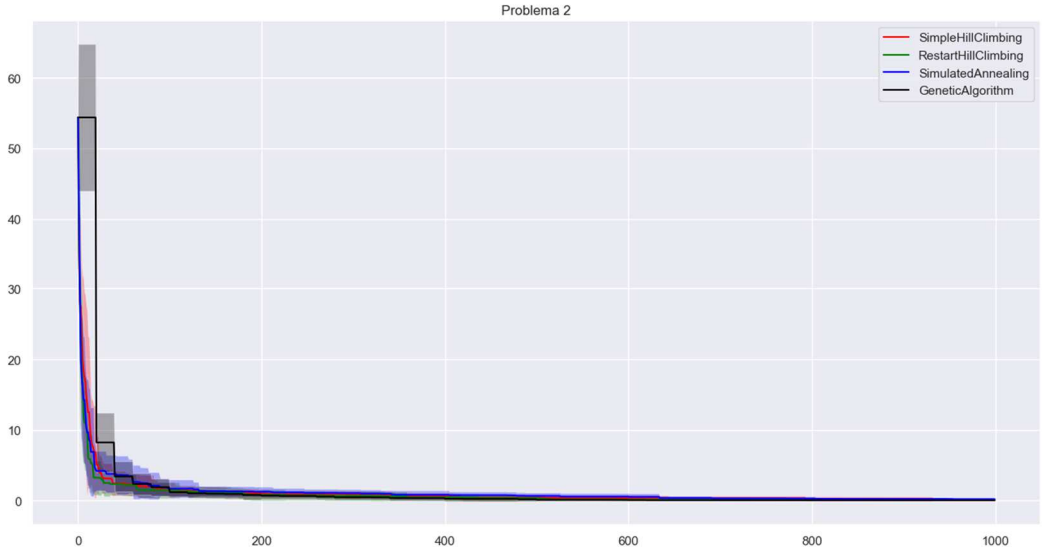


Tabela comparativa:

Algoritmo	Max	Min	Média	Desvio Padrão
HillClimbing simples	68.88754837396415	0.0017206568809271516	1.0459503824422114	3.3422903273795512
HillClimbing c/ restart	68.88754837396415	0.0065170233741618230	0.8093127985001215	2.9850032714012810
Simulated Annealing	68.88754837396415	0.0009753086338903927	1.1616722246724340	3.1108564281512447
Genetic Algorithm	68.88754837396415	**0.0001971769719570915**	1.6753939761692394	7.8131288562002610

Comparativo dos algoritmos:



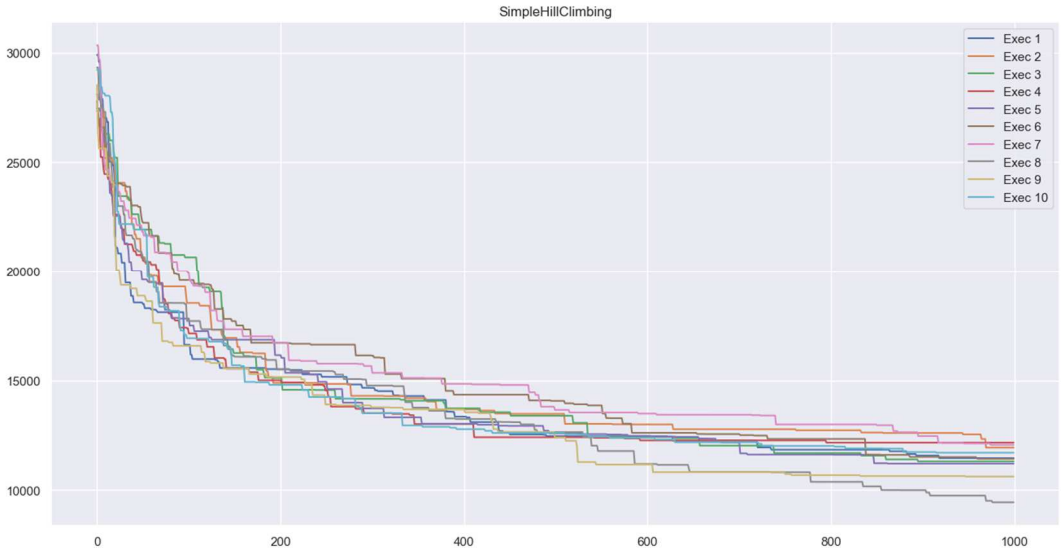
Discussão:

Todos os algoritmos apresentaram bom desempenho para o problema 2. O HillClimbing simples apresentou dificuldades para sair de mínimos locais com a distribuição normal de desvio padrão 2.0, mas ainda venceu alguns. O ponto de inicialização influenciou no resultado. O HillClimbing com restart explorou várias partes do intervalo válido, o que proporcionou boas oportunidades para encontrar soluções melhores, mas devido ao espaço de pesquisa ser mais amplo que o do problema 1, teve pouca oportunidade (iterações) de buscar um mínimo local melhor em cada restart. O Simulated Annealing também explorou partes do intervalo próximas a sua inicialização e teve a oportunidade de vencer vários mínimos locais aceitando vizinhos piores. O Genetic Algorithm apresentou o melhor desempenho. A operação de crossover e mutação geraram soluções que puderam cobrir muitos pontos do espaço de pesquisa. Não percebi convergência dos indivíduos com o crossover utilizado para a função de Rastrigin.

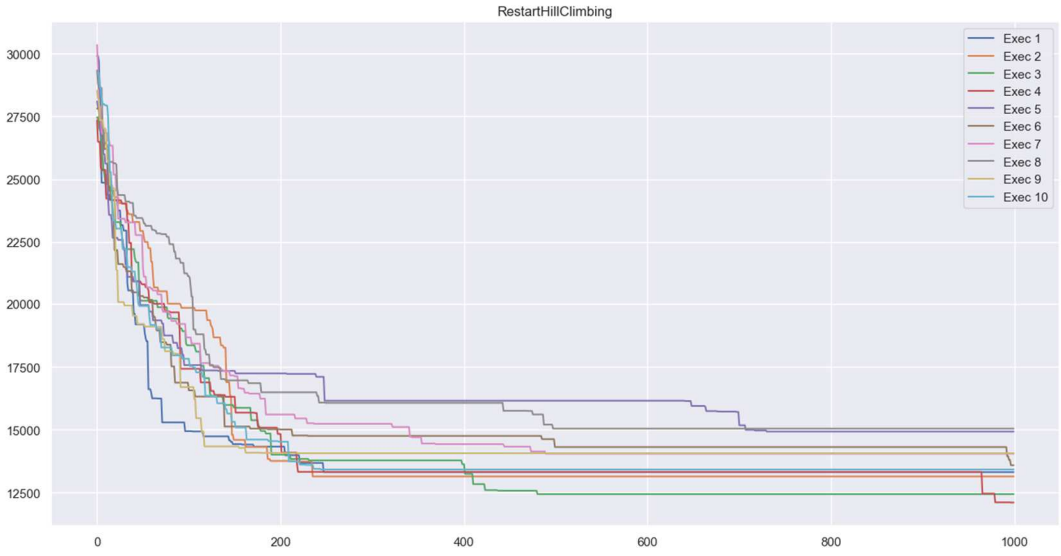
Problema 3:

Minimizar TSP de 38 cidades

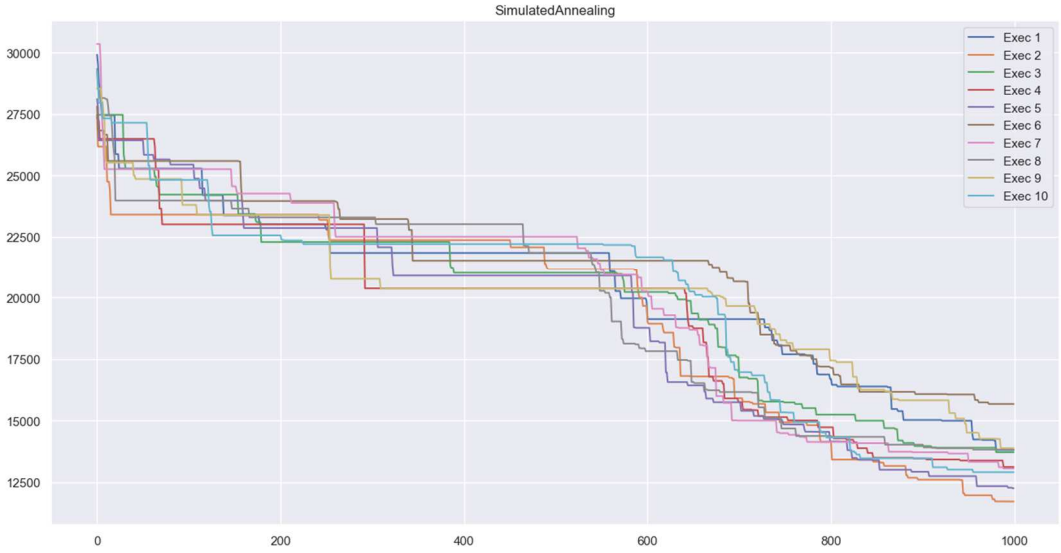
HillClimbing simples:



HillClimbing com restart:



Simulated Annealing:



Genetic Algorithm:

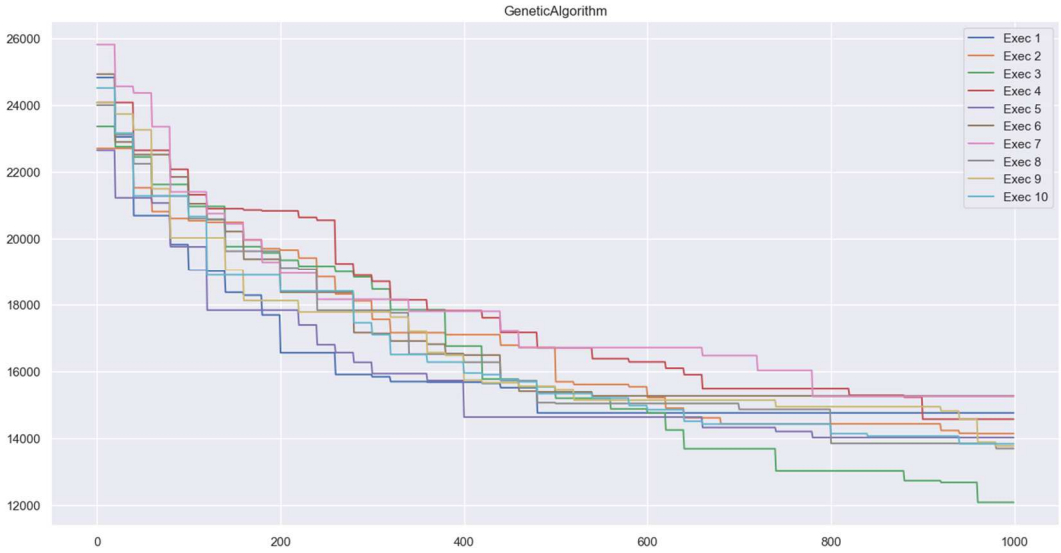
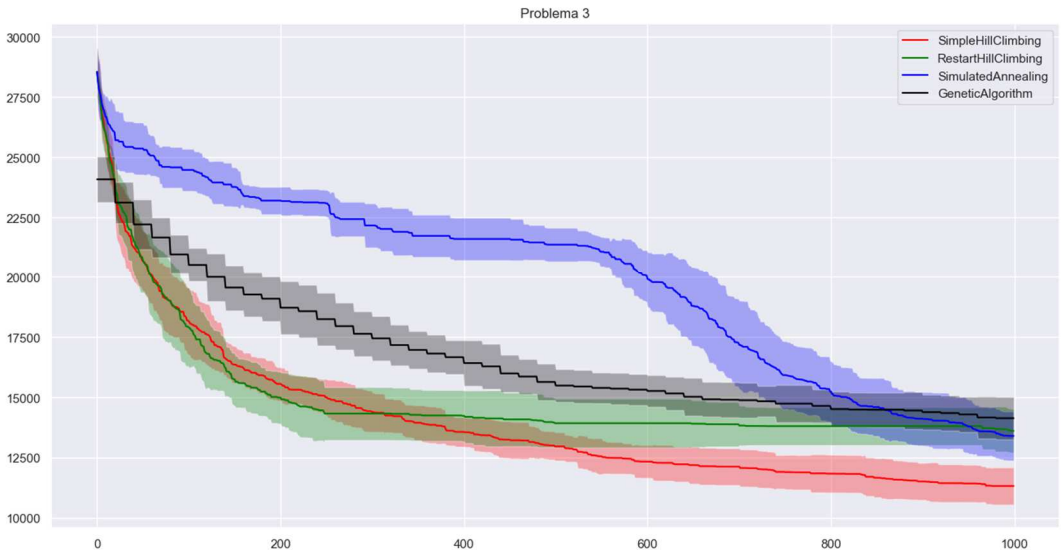


Tabela comparativa:

Algoritmo	Max	Min	Média	Desvio Padrão
HillClimbing simples	30344.23910360661	**9430.247200792006**	14010.164121676013	3164.1583117547057
HillClimbing c/ restart	30344.23910360661	12107.760768964139	14957.289053714416	2658.822387862522
Simulated Annealing	30344.23910360661	11720.442902220122	19949.136216161085	3954.838260144989
Genetic Algorithm	25802.159720445085	12080.264978438023	16731.755156331950	2679.259083071553

Comparativo dos algoritmos:



Discussão:

Os algoritmos não apresentaram bom desempenho em relação ao ótimo global conhecido para o problema 3. O HillClimbing simples apresentou o melhor resultado, apesar de ser, teoricamente, o algoritmo menos elaborado. O HillClimbing com restart, com a configuração de 1000 iterações e restart a cada 200 ficou ligeiramente melhor que com restart a cada 50, pois conseguiu "tempo" (iterações) para melhorar o cada estado inicial. O Simulated Annealing, segundo melhor resultado, também teve dificuldade e ficou com resultado bem acima do ótimo global. Para este problema, achei que o Simulated Annealing precisaria de rodadas de aquecimentos e resfriamentos para melhor exploração do espaço de pesquisa. Após alteração de 90% para 65% das iterações para parar de ter chances de aceitar estados piores, o algoritmo também apresentou uma pequena melhora. O Genetic Algorithm apresentou o segundo pior resultado dentre os 4, o que foi uma surpresa. Além de resultados ruins, para o tamanho do problema, o algoritmo também ficou mais lento (grande parte por conta implementação resultado da minha total inexperiência com Python). O algoritmo apresentou uma melhora quando mudei a forma de seleção dos pais para o crossover. A saber, seleciono os 2 primeiros candidatos através de sorteio entre os 20% melhores da população. Fico com o melhor dos 2. Para a escolha do outro pai, seleciono por sorteio 2 dentre todos os indivíduos da população e novamente fico com o melhor. Obs: todos os comentários e parâmetros ajustados consideram que o número máximo de chamadas da FO é limitado a 1000.