UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE CIÊNCIAS APLICADAS

PO240 – Introdução à Meta-Heurística

Tema: Algoritmo Genético para sequenciamento de tarefas em uma única máquina

Orientadores:

Prof. Dr. Priscila Rampazzo

Guilherme Lima Correa - 173811

Limeira, 09 de Junho de 2020

1.Resumo

Propõe-se neste trabalho três tipos de implementações computacionais (em Python) para o problema de sequenciamento de tarefas em uma máquina. O primeiro tipo consiste na solução exata com um solver (CBC), o segundo a implementação de um algoritmo genético (AG) com codificação inteira e, por último, a implementação de um AG com codificação em ponto flutuante.

Palavra Chaves: Sequenciamento, Algoritmo Genético, Solver.

2. Introdução

Em um mundo capitalista onde o tempo acaba sendo um dos fatores decisivos para uma empresa ser competitiva lidamos com os problemas de scheduling, bastante explorado pela Pesquisa Operacional. Problemas relacionados a atrasos na produção passam a implicar diretamente no lucro de uma empresa, já que o atraso gera insatisfação com o nível de serviço aos clientes incorrendo em problemas com a reputação da empresa.

Dessa forma, não é incomum empresas colocarem esforços na redução do atraso em suas atividades, consequentemente a literatura científica também acompanha estes movimentos e nota-se uma imensa quantidade de trabalhos com diversas abordagens acerca deste problema.

O problema abordado neste trabalho é conhecido como sequenciamento de tarefas em uma única máquina. Onde o desafio consiste em alocar todas as tarefas em uma única máquina na qual, cada tarefa deve ser processada apenas uma vez e procura-se sequenciar as tarefas de maneira a minimizar o Término Ponderado. Segundo Pinedo (2002), "O objetivo a ser minimizado é sempre uma função do tempo de conclusão dos jobs, que é claro, depende do sequenciamento".

Feili, Haddad, Ghanbari (2012) também trabalharam com um modelo matemático exato para resolver este tipo de problema e, em seguida, duas metaheurísticas; Algoritmo Genético (GA) e Simulated Annealing (SA), obtendo resultados satisfatórios.

O objetivo do trabalho é avaliar o desempenho das três implementações para problemas de pequena e grande escala. Vale ressaltar, que o algoritmo genético não garante a solução ótima do problema como é obtida através do Solver. Portanto, será avaliado se a solução proposta pelo AG é viável. Levando em consideração o valor da função objetivo e o tempo para obtê-la.

Muitos problemas possuem um grande gasto computacional para encontrar a solução ótima e, consequentemente, torna o tempo de operação inviável. Portanto, a

solução oferecida por um algoritmo genético, apesar de não ótima, pode ser suficiente para a empresa reduzir seus gastos com um tempo operacional aceitável.

3. Modelo Matemático

3.1 Índices:

I = 1,..., n Representa todas as *n* tarefas.

K = 1,...,k Representa todas as *n* tarefas.

3.2 Dados:

R_i Representa o tempo de chegada de cada tarefa i.

P_i Representa o tempo de processamento de cada tarefa i.

D_i Representa o prazo de término de cada tarefa i.

W_i Representa o peso de importância de cada tarefa i.

3.3 Variáveis de Decisão:

X_i Representa o tempo de início de cada tarefa i.

Y_{ik} 1 , se a tarefa i é processada antes da tarefa k

0, caso contrário.

3.4 *Modelo:*

$$Min \sum_{i=1}^{n} Wi * (Xi + Pi) \qquad \forall i = 1, \dots, n \qquad (1)$$

s.a:

$$Xi \geq Ri \qquad \forall i = 1, \dots, n \qquad (2)$$

$$Xi + Pi \leq Xk + M(1 - Yik) \qquad \forall i = 1, \dots, n \; e \; k > i \qquad (3)$$

$$Xk + Pk \leq Xi + MYik \qquad \forall i = 1, \dots, n \; e \; k > i \qquad (4)$$

$$Xi + Pi \leq Di \qquad \forall i = 1, \dots, n \qquad (5)$$

$$Xi \geq 0 \qquad \forall i = 1, \dots, n \qquad (6)$$

$$Yik \in \{0,1\} \qquad \forall i = 1, \dots, n \; e \; k > i \qquad (7)$$

Este modelo matemático foi implementado através do solver (CBC) e testado para diversas instâncias, cada uma delas possui um número diferente de tarefas. O Solver apresentou um bom desempenho para instancias até 80 tarefas, para números superiores de tarefas o tempo de otimização foi superior a 1 hora (tempo definido como limite para a otimização) e em muitas instâncias nem foi possível encontrar uma solução. Sendo assim, o algoritmo genético acaba sendo necessário, pois com tempos menores de otimização é possível encontrar soluções que satisfaçam o problema.

4. Algoritmo Genético

Para modelar um algoritmo genético é preciso codificar o problema. As codificações escolhidas para este projeto foi a Codificação Inteira e a Codificação em Ponto Flutuante (PI). Vale ressaltar, que as duas já foram modeladas antes do início do projeto, ou seja, são como pontos partidas. Além da codificação é necessário definir alguns parâmetros para o AG, tanto o parâmetro de tamanho da população quanto o número de gerações foram definidos empiricamente. O Algoritmo foi criado e testado 10 vezes para cada instância e para cada codificação, para definir quais são os melhores parâmetros. Os testes levaram em conta apenas o valor da função objetivo, visto que, o maior tempo de processamento era inferior a 10 minutos (ANEXO A) , o que obteve melhores resultados foi quando o tamanho da população e gerações são iguais a 200 tanto na codificação inteira como na codificação em ponto flutuante. Portanto, para a comparação, ambas as codificações tiveram seus parâmetros fixos em 200.

5. Algoritmo Genético com Codificação Inteira

Com a inicialização da população, a decodificação e o cálculo da função objetivo já desenvolvidos. Precisou-se implementar um operador de seleção de Pais para o Crossover, optou-se por um torneio binário. Neste procedimento, sorteiam-se dois indivíduos ao acaso, comparam-se suas aptidões e o mais apto destes dois é selecionado, neste caso como a função objetivo é de minimização, seleciona-se aquele com a menor valor. Este procedimento é repetido até que toda a população tenha participado do torneio. Neste procedimento houve o cuidado para não selecionar pais idênticos para o torneio, e além disso, também foi incentivada a variabilidade genética, ou seja, caso exista indivíduos selecionados iguais que participaram de torneios diferentes, o segundo indivíduo é substituído pelo indivíduo derrotado diretamente no torneio (Tabela 1). Caso, o tamanho da população não seja par, o modelo avisa para realizarmos uma alteração. Após o torneio ter terminado, obtém-se uma população com a metade do tamanho da população inicial.

Tabela 1: Modificação caso pais selecionados sejam iguais.

Indivíduo	F.O			
1	100	Torneio	Vencedor	Selecionados
2	130	**1 x 3**	1 (f.o. = 100)	1 (f.o. = 100)
3	245	**2 x 4**	4 (f.o. = 100)	2 (f.o. = 130)
4	100			

Após selecionarmos os pais, realizamos o operador genético predominante. Através de cruzamentos são criados indivíduos novos misturando as características de dois pais. O operador desenvolvido neste trabalho foi o crossover de 1 ponto. Para isso, em um cruzamento onde temos dois pais, são gerados dois filhos idênticos aos pais. Estipula-se uma taxa de crossover de 65%. Caso ocorra o crossover, sorteia aleatoriamente o ponto de corte, deste ponto em diante o filho selecionado recebe as características do segundo pai (Tabela 2). Lembrando que este filho selecionado é cópia exata do primeiro pai até a realização do crossover.

Tabela 2: Operador de Crossover para Codificação Inteira

Indivíduo 1	1	1	0	1	0	1	1	1
Indivíduo 2	0	1	1	1	1	1	0	0
Descendente 1	1	1	0	1	0	1	0	0
Descendente 2	0	1	1	1	1	1	1	1

Dessa maneira, o crossover precisa de uma reparação, visto que, pode ocorrer de um filho repetir tarefas ou não possuir alguma tarefa. Para isso, após a combinação de características, averígua-se se o filho tem todas as tarefas, caso ele não tenha, as tarefas faltantes são copiadas da mesma ordem que são dispostas no segundo pai. Após a mutação temos a uma quantidade de filhos igual a quantidade de pais selecionados.

Em seguida implementou-se um operador de mutação. Este modifica aleatoriamente alguma característica do indivíduo sobre o qual é aplicado. Desta forma, a mutação assegura que a probabilidade de se chegar a qualquer ponto do espaço de busca possivelmente não será zero. O operador de mutação é aplicado aos indivíduos através de uma taxa de mutação de 5%. A mutação proposta é simples, escolhe-se duas tarefas aleatórias e mudam elas de posição.

Tabela 3: Operador de Mutação para Codificação Inteira

Descendente 2	0	1	1	1	1	1	1	1
Novo Descende	0	1	1	1	1	1	1	1

Por fim, implementou-se um operador para selecionar os indivíduos para uma nova população. Este operador concatena a população inicial desta geração com os filhos e seleciona-se os indivíduos com a melhor função objetivo até que se obtenha o tamanho da população padrão. Caso tenha indivíduos repetidos, opta-se por retirá-los e selecionar o melhor dentre aqueles que ainda não foram escolhidos.

6. Algoritmo Genético com Codificação em Ponto Flutuante

Para este algoritmo foram reaproveitados alguns operadores como o operador de seleção de pais para Crossover e o operador para seleção da próxima geração. Sendo assim, além da codificação diferente foi proposto operadores de crossover e mutação diferentes.

O Crossover neste algoritmo é uniforme, ou seja, são gerados classificadores aleatórios (0 e 1) do tamanho do número de tarefas desta instância. Ou seja, uma instancia com 8 tarefas teremos umas sequências com 8 classificadores (Tabela 4):

*Tabela 4:*Operador de Crossover para Codificação em Ponto Flutuante

Classificador	1	1	0	1	0	1	0	1
Filho	1	1	1	1	1	1	0	1

Caso o classificador seja igual a 1, esta posição terá a tarefa copiada do pai 1 e caso o classificador seja 0, esta posição terá a tarefa copiada do pai 2. Vale ressaltar que a probabilidade de ocorrer o crossover é de 65%. Já o operador de mutação, escolhe-se um ponto aleatório do filho e soma-se a aquela posição da codificação em ponto flutuante um valor aleatório entre 0 e 1. Se valor se tornar maior que um, ou menor que zero, este valor é reparado. Pois, o menor valor possível tem que ser zero e o maior tem que ser 1. A taxa de mutação é de 5%.

7. Resultados

Para validarmos os resultados de cada algoritmo genético todas instâncias analisadas foram resolvidas o máximo de vezes possível em 15 minutos. A partir disso, destaca-se o melhor resultado obtido, ou seja, o menor valor que a função objetivo alcançou. Além disso, evidenciou-se tanto tempo médio de execução como o valor médio dos melhores indivíduos de cada geração.

Assim pode-se observar se o algoritmo converge melhor para a mesma solução na codificação inteira.

Tabela 5: Resultados do Solver, CDI,CPF.

	Solver (CBC)		Codificação Inteira				Codificação em Ponto Flutuante			
	Tempo (s)	Função objetivo	Tempo (s)	Função objetivo	Médias das funções objetivos	Número de rodadas	Tempo (s)	Função objetivo	Médias das funções objetivos	Número de rodadas
Instância 10	1.34	335	13,34	340	363	40	8,02	**336**	366,15	40
Instância 20	601.15*	3127	14,87	3672	3898,68	40	9,53	**3600**	3969,25	40
Instância 30	600.74*	12752	18,79	**16031**	210700	40	10,73	16644	424325	40
Instância 40	600.82*	26141	21,8	**36833**	3479123	40	12,19	39820	6713079	40
Instância 60	600.38*	81087	28,39	**128801**	7576268	32	14,85	134411	1,4E+07	40
Instância 80	1199.98	223743	47,98	**1E+08**	2,8E+08	19	17,31	1,7E+08	3,3E+08	40
Instância 100	3600.97	466794	45,93	**3E+08**	6,6E+08	20	20,1	3,7E+08	8,4E+08	40
Instância 200	x	x	83,57	**2E+10**	2E+10	12	44,1	1,5E+10	2E+10	21
Instância 500	x	x	301	2E+12	2,1E+12	3	62,49	**1E+12**	1,9E+12	12

* Solver estourou o tempo limite (10 minutos)

Pode-se concluir que para as instâncias menores o algoritmo genético apresenta boas soluções já que o valor do melhor indivíduo da população se aproxima da otimalidade. Para instâncias maiores a análise é mais complexa, visto que, o valor da função objetiva é alto. Sendo assim, o AG acaba apresentando maiores divergências. Nota-se que neste caso, a codificação em inteira apresentou melhores soluções do que a codificação em ponto flutuante, somente em relação ao melhor indivíduo, pois em relação ao tempo a codificação em PI foi melhor. Porém, como as combinações e mutações divergem não é possível afirmar que a codificação inteira é melhor para o problema de sequenciamento em uma única máquina. Quando se analisa os gráficos de Fit e Gerações nota-se que a codificação inteira converge para uma solução mais vezes, ou seja, tem um desvio do melhor fit em cada geração menor do que na codificação PF.

8. Referências Bibliográficas

PINEDO, Michael. “Scheduling - Theory, Algorithms, and Systems”. 2. ed. New Jersey: Prentice-hall, 1995.

Coelho, Leandro dos S. "Fundamentos, Potencialidades e Aplicações de Algoritmos Evolutivos", SBMAC, 2003.

FEILI, H.; HADDAD, H.; GHANBARI, P. “Two Hybrid Algorithms for Single-Machine Total Weighted Tardiness Scheduling Problem with Sequence-Dependent Setup.” American Journal of Scientific Research, Volume 64, p. 22-29.

ANEXO

Anexo A: Decisão dos parâmetros para as codificações

Tabela 6: Decisão empírica dos parâmetros para Codificação Inteira .

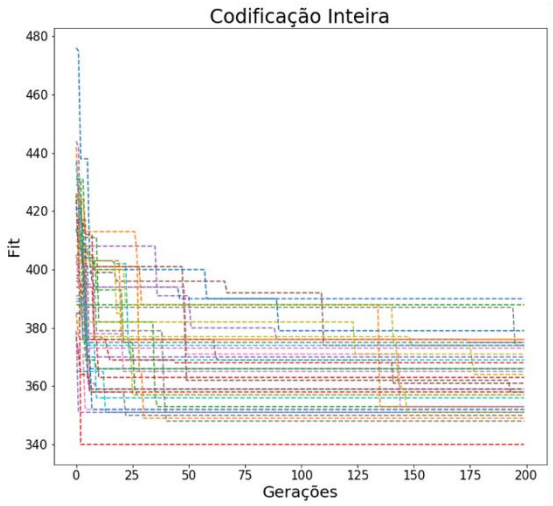
Tp	100	100	200	200	200	200	150	150
G	100	100	200	200	100	100	50	50
	melhor	média	melhor	média	melhor	média	melhor	média
Instância 10	360	384	340	358	351	363	361	373
Instância 20	3842	4061	3672	3928	3773	3946	3798	4124
Instância 30	18158	529230,4	16031	328759,6	17826	201632	18740	765929
Instância 40	2242474	9272196	36833	3809990	38825	3996727	1924002	4971372
Instância 60	124705	15685969	128801	9022231	128603	10740246	154267	10602789
Instância 80	2,56E+08	3,98E+08	1,14E+08	2,87E+08	79743266	3,05E+08	1,54E+08	2,68E+08
Instância 100	6,35E+08	9,89E+08	3,35E+08	58229379	4,59E+08	6,17E+08	3,31E+08	7,69E+08
Instância 200	1,72E+10	2,25E+10	1,53E+10	2,05E+10	1,48E+10	2,06E+10	1,43E+10	2,29E+10
Instância 1000	1,57E+12	2,09E+12	1,95E+12	1,84E+12	1,79E+12	1,94E+12	1,6E+12	1,98E+12

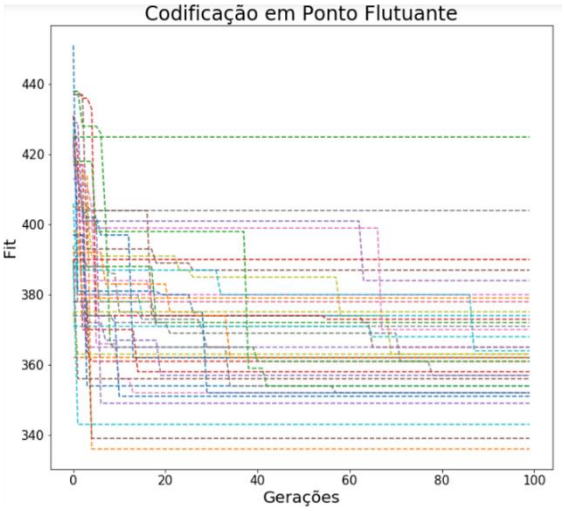
Tabela 7:Decisão empírica dos parâmetros para Codificação em Ponto Flutuante

Tp	100	100	200	200	200	200	150	150
G	100	100	200	200	100	100	50	150
	melhor	média	melhor	média	melhor	média	melhor	média
Instância 10	335.0	361.1	336	369	346	361	343	372
Instância 20	3882	4162,4	3600	4018,8	3821	3890	3736	4044
Instância 30	16530	1056963	16644	438099,8	18062	256661	19834	675130
Instância 40	37547	12224577	39820	6949351	666405	6069449	39050	7011458
Instância 60	5863588	23272831	134411	10886780	130490	11960750	1570705	14467295
Instância 80	2,41E+08	4,11E+08	1,72E+08	3,53E+08	1,28E+08	3,37E+08	2,71E+08	3,9E+08
Instância 100	7,64E+08	1,03E+09	3,69E+08	9,22E+08	5,03E+08	7,52E+08	6,32E+08	8,93E+08
Instância 200	2,14E+10	2,45E+10	1,52E+10	2,29E+10	1,49E+10	2,21E+10	1,63E+10	2,24E+10
Instância 1000	1,75E+12	2,1E+12	1,47E+12	2,07E+12	1,82E+12	2,03E+12	1,81E+12	1,98E+12

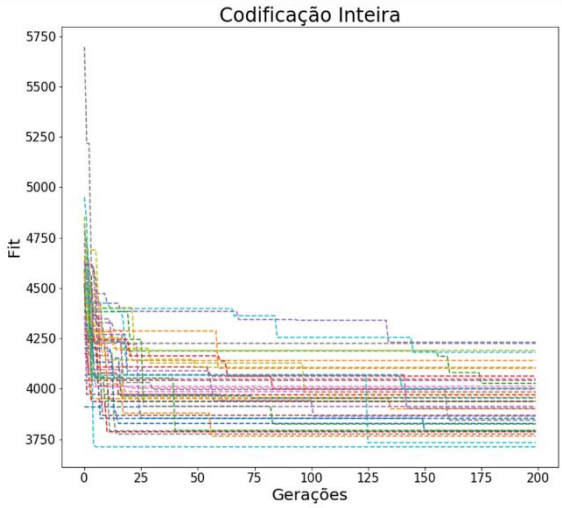
Anexo B: Gráfico Comparativo entre as codificações

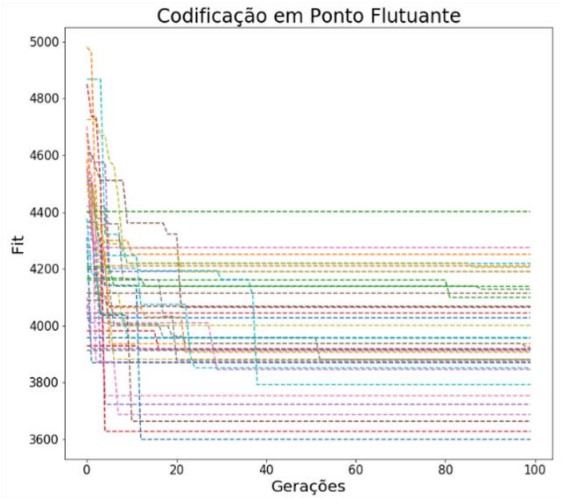
Instância 10



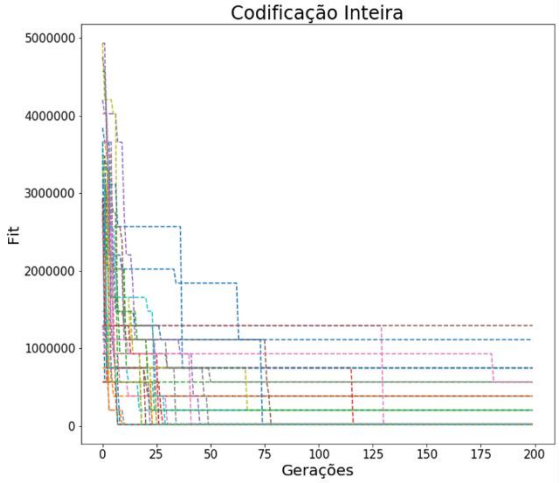


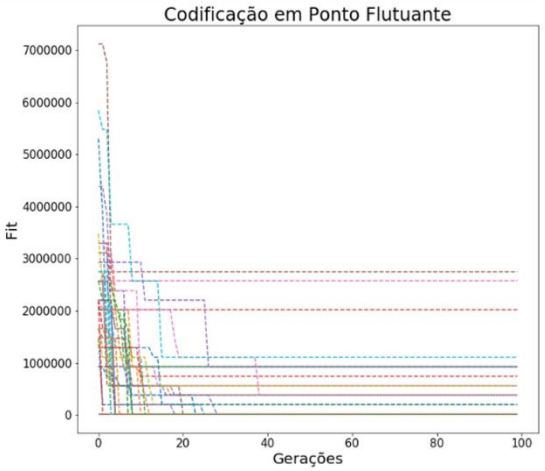
Instância 20



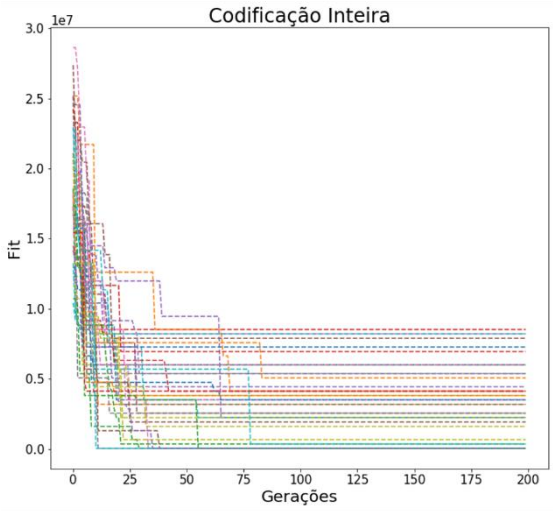


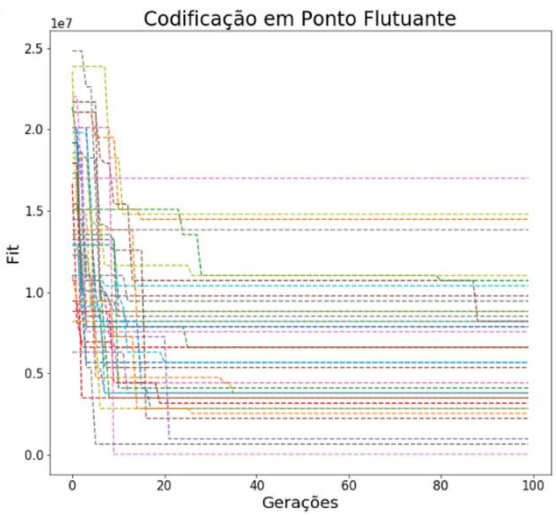
Instância 30



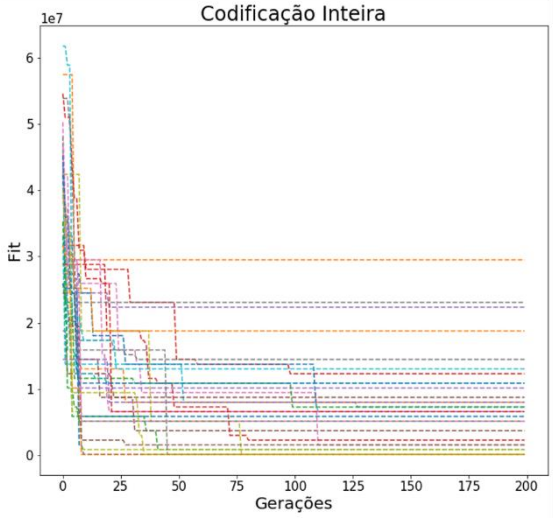


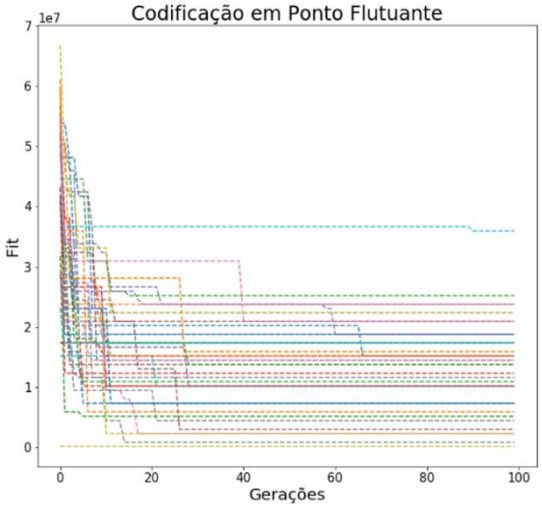
Instância 40



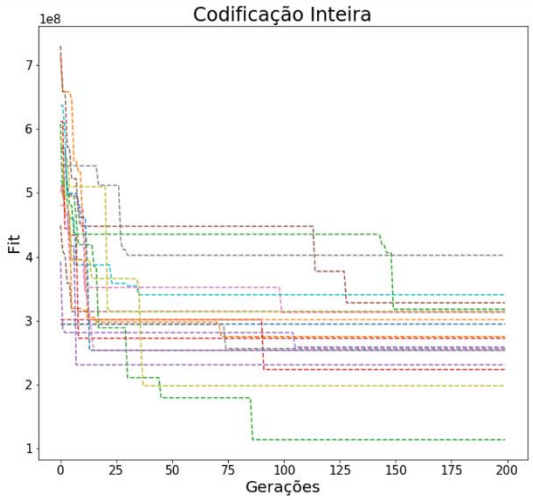


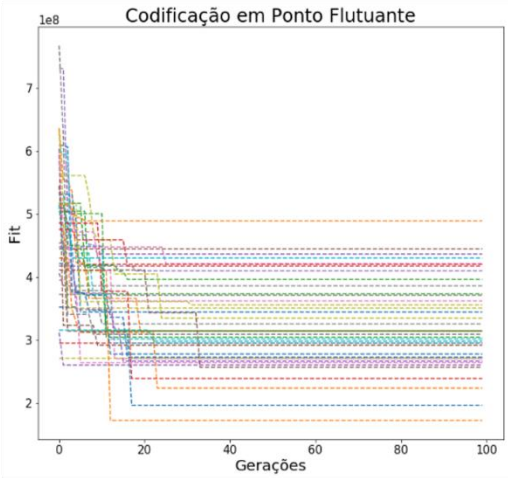
Instância 60



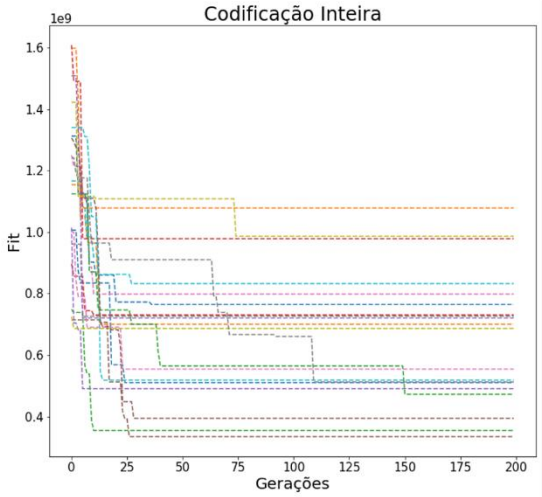


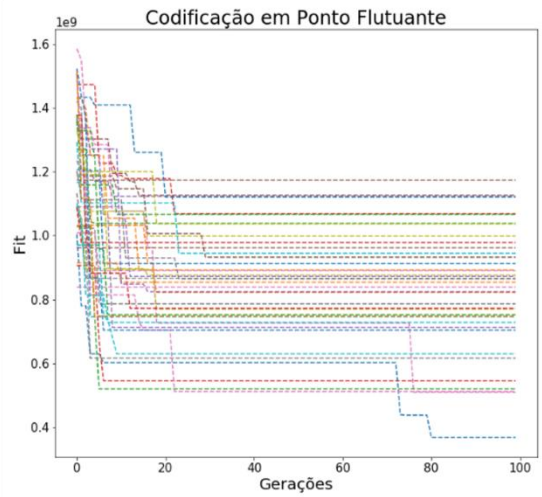
Instância 80



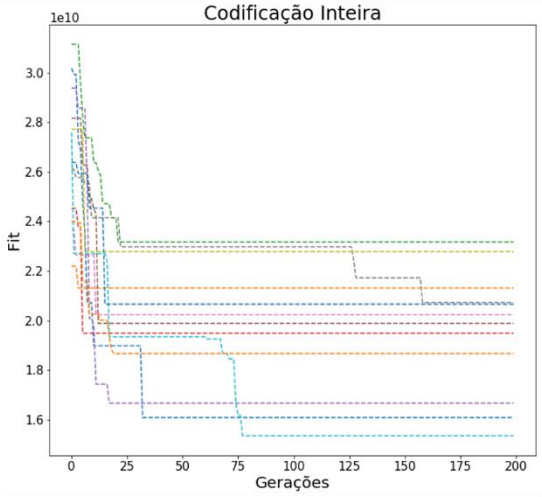


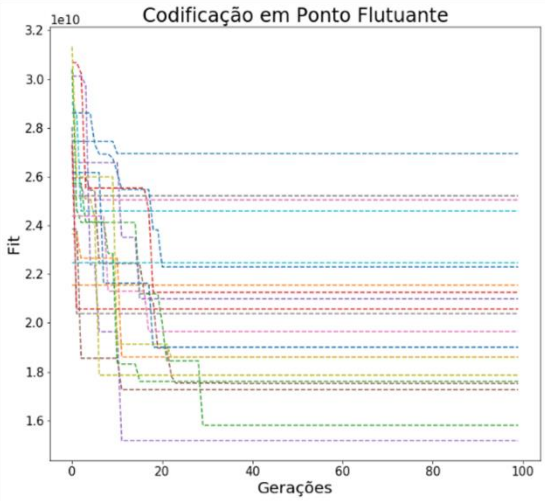
Instância 100





Instância 200





Instância 500