PRO5826 - Estudo de Heurísticas e Meta-heurísticas para Problemas de Produção

Implementação III

Jhonatan Albertini

NUSP: 11887309

Luiza Pellin Biasoto

NUSP: 6846703

Docente: Debora Pretti Ronconi

14 de setembro de 2021

1 Introdução

Este relatório visa descrever as premissas, métodos e o passo-a-passo da elaboração de uma heurística construtiva, heurística de melhoria e metaheurística para otimização do problema de planejamento da produção em uma única máquina com datas de entrega em comum, considerando penalidades de adiantamento e de atraso de entrega. A modelagem do problema considerado NP-difícil tem como base o modelo descrito por Biskup e Feldmann[1]. Os testes serão executados em 280 problemas distintos: 7 conjuntos de 10 instâncias, com dimensões de 10, 20, 50, 100, 200, 500 e 1000 tarefas cada, sendo otimizados sob 4 diferentes valores para o parâmetro h, que determina o quão restritiva é a data de entrega do problema. Os resultados das otimizações utilizando a heurística e métodos descritos neste trabalho serão então comparados com os resultados obtidos por Biskup e Feldmann. Nas conclusões, serão apresentadas as suas vantagens e limitações, além de direcionadores para os próximos passos deste trabalho.

1.1 Organização do texto

O problema de planejamento da produção em uma única máquina com data de entrega em comum será definido na Seção 2. O objetivo deste relatório é descrever a implementação de uma heurística construtiva, uma heurística de melhoria e uma metaheurística.

A elaboração da heurística e suas premissas serão descritas na Seção 3. Resultados computacionais da heurística construtiva são comparados com os do artigo do Biskup serão apresentados na Seção 3.3. Em seguida será apresentada a busca local, definindo a vizinhança, o movimento, o critério de parada e os resultados. Por último serão apresentados com detalhes a metaheurística escolhida, algoritmo genético. A implementação e os resultados da busca local serão apresentados na seção 4 e as conclusões mostradas na Seção 6.

2 Descrição do problema

O problema de planejamento da produção em uma única máquina com datas de entrega em comum considera diferentes penalidades para adiantamento e atraso das tarefas executadas. O problema possui n tarefas que precisam ser processadas uma única vez e por uma única máquina. As $i = 1, ..., n$ tarefas possuem tempos de processamento determinísticos e conhecidos, p_i, e cada uma possui penalidades por unidade de tempo para adiantamento a_i e atraso b_i distintos. A data de entrega de todas as tarefas é então definido como uma única data d em comum. Caso a data de término C_i de uma tarefa seja menor do que a data de entrega comum d, o adiantamento E_i é definido por: $E_i = d - C_i$. Por outro lado, caso a data de término de uma tarefa seja maior que d, o atraso desta é definido por: $T_i = C_i - d$.

Em problemas de planejamento com data de entrega em comum, d é uma informação determinística, fornecida ou calculada, que sensibiliza o quão restritivo é o problema. Uma data de entrega $d \geq \sum^{\mathrm{n}}{}_{\mathrm{i=i}} p_{\mathrm{i}}$, i.e., igual ou maior que a soma de todos os tempos de processamento das tarefas, é considerada uma variável de decisão, não restritiva e permite o sequenciamento das tarefas sem considerá-la. Caso seja menor que o tempo total de processamento, ela se torna restritiva e passa a ser considerada para o sequenciamento das tarefas.

A complexidade do problema também depende dos termos de penalidades por unidade de tempo de atraso e adiantamento que, quando combinados com a data de entrega em comum restritiva ou não restritiva, determinam se o problema é solucionável em tempo polinomial ou NP-difícil. Biskup resumiu essas relações na Tabela 1.

A data de entrega em comum para os problemas considerados neste trabalho é restritiva e definida por:

$$d = h * \sum_{i=1}^{n} p_i \tag{1}$$

sendo que h pode assumir o valor de 0,2, 0,4, 0,6 ou 0,8 dependendo do nível de restrição do problema. As penalidades por unidade de tempo de atraso e adiantamento também são determinísticas e distintas para cada um dos problemas. Portanto, este é definido

Tabela 1: Resumo das complexidades dos problemas para datas de entrega comuns restritivas e não restritivas.

para i = 1, ..., n:	**d é não restritiva**	**d é restritiva**
$\mathbf{a_i = b_i = 1}$	Solucionável em tempo polinomial	NP-difícil, solucionável em tempo pseudo-polinomial
$\mathbf{a_i = a}$ **e** $\mathbf{b_i = b}$	Solucionável em tempo polinomial	NP-difícil, solucionável em tempo pseudo-polinomial
$\mathbf{a_i = b_i}$ **(problema balanceado)**	NP-difícil, solucionável em tempo pseudo-polinomial	NP-difícil, solucionável em tempo pseudo-polinomial
$\mathbf{a_i, b_i}$ **(problema generalizado)**	NP-difícil, aberto	NP-difícil, aberto

como NP-difícil e aberto, i.e., torna improvável encontrar algoritmos eficientes capazes de obter a solução ótima do problema.

2.1 Propriedades

Biskup descreve duas importantes propriedades do problema considerado e que são ilustradas na Figura 1.

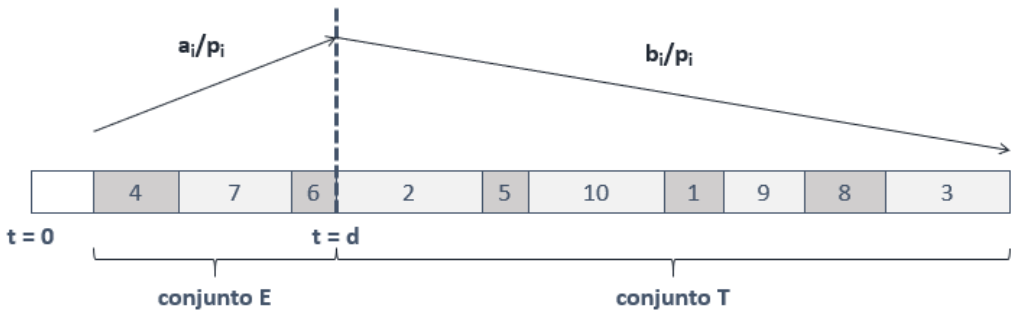


Figura 1: Exemplo de um sequenciamento ótimo de 10 tarefas, ilustrando as Propriedades I e II.

2.1.1 Propriedade I: Não há tempo ocioso entre tarefas sucessivas em um planejamento ótimo

É possível que, em um planejamento ótimo, a primeira tarefa não inicie no instante 0. Porém, não há intervalos ociosos entre tarefas sucessivas.

Cada sequência de tarefas pode ser dividida em 3 subconjuntos E, D e T, com $|D| \leq 1$. O subconjunto E possui as tarefas que terminam antes da (ou exatamente na) data de entrega comum d, o subconjunto T possui as tarefas que iniciam depois da (ou exatamente

na) data de entrega comum, e o conjunto D possui no máximo uma tarefa que inicia estritamente antes de d e finaliza após d.

2.1.2 Propriedade II: Todo planejamento ótimo possui uma propriedade em forma de V

As tarefas de E são ordenadas em ordem crescente de acordo com a relação $\mathrm{a_i/p_i}$, e as tarefas de T são ordenadas em ordem decrescente de acordo com $\mathrm{b_i/p_i}$.

2.2 Formulação matemática

Para instâncias menores, é possível encontrar o planejamento ótimo S* através da formulação com programação inteira mista descrita nesta seção. s_i e x_{ik} são as variáveis de decisão que, juntas com d, definem S*. s_i é o tempo de início de uma tarefa, x_{ik} recebe 1 se a tarefa i iniciar antes de k (senão, 0) e R é um número suficientemente grande (big-M). A soma das penalidades de atrasos e adiantamentos das tarefas do problema é definida conforme a equação (2) a seguir:

$$f(S) = \sum_{i=1}^{n} a_i E_i + \sum_{i=1}^{n} b_i T_i \tag{2}$$

sendo S uma solução factível para o problema. O objetivo é encontrar um planejamento S^* que minimize a equação (2) sujeito a:

$$T_i \geq s_i + p_i - d, i = 1, ..., n \tag{3}$$

$$E_i \geq d - s_i - p_i, i = 1, ..., n \tag{4}$$

$$s_i + p_i \leq s_k + R(1 - x_{ik}), i = 1, ..., n-1, k = i+1, ..., n \tag{5}$$

$$s_k + p_k \leq s_i + Rx_{ik}, i = 1, ..., n-1, k = i+1, ..., n \tag{6}$$

$$T_i, E_i, s_i \geq 0, i = 1, ..., n \tag{7}$$

$$x_{ik} \in \{0, 1\}, i = 1, ..., n-1, k = i+1, ..., n \tag{8}$$

As restrições (3) e (4) determinam a quantidade de atraso e adiantamento, respectivamente, de cada tarefa i. As restrições (5) e (6) determinam o instante do processamento das tarefas: se i for executada antes de k, a equação (5) é restritiva para s_i, pois $x_{ik} = 1$ (sendo p_i maior que 0, s_k obrigatoriamente será maior que s_i, deixando a equação (6) irrestrita para s_k); caso k seja executada antes de i, $x_{ik} = 0$ e o inverso ocorre: a equação (6) passa a ser restritiva para s_k e a (5) passa a ser irrestrita para s_i.

3 Heurísticas propostas

3.1 Ideia geral

As heurísticas propostas neste trabalho se baseiam na heurística definida por Jarboui et al.[3], denominada Hte. Considere dois conjuntos de tarefas, E e T. Ambos são divididos pela data de entrega em comum *d*. O tempo de processamento total de E é limitado por *d*, enquanto a primeira tarefa de T tem $s_i = d$. Seu princípio consiste de inserir no conjunto T, a cada iteração, a tarefa que, *a priori*, é menos interessante quando consideramos as primeiras tarefas a serem executadas, e reciprocamente para o conjunto E. As tarefas são ordenadas de forma decrescente em a_i/p_i (respectivamente b_i/p_i) para serem adicionadas ao conjunto T (respectivamente E). Quando não for mais possível alocar tarefas em E (no caso de E atingir o limite de processamento d) ou não houver mais tarefas a serem alocadas, o processamento para. No limite, será possível alocar no máximo $n/2$ tarefas ao conjunto E, e por isso é necessária uma fase de busca local para refinar os resultados da heurística.

A Hte considera que todo planejamento ótimo do problema considerado tem uma propriedade em formato de **V**, conforme descrito na Seção 2.1 deste trabalho. As tarefas i do conjunto E são ordenadas crescentemente em relação a a_i/p_i, e as tarefas i no conjunto T são ordenadas decrescentemente em relação a b_i/p_i.

Jarboui et al. demonstram que essa heurística Hte apresenta resultados satisfatórios e analisando como ela se comportava na prática, pode-se observar que algumas tarefas com alto b_i/a_i, indicando afinidade pelo conjunto T, eventualmente acabava vendo alocada no conjunto E. Considerando que tanto a heurística Hte quanto a heurística II proposta por Biskup produzem resultados satisfatórios, foi elaborada uma heurística que aproveitasse o melhor de cada. Considerando que uma tarefa com relação b_i/a_i alta, é esperado que ela seja alocada no conjunto T.

3.2 Detalhamento do método

A heurística desenvolvida neste trabalho consiste de duas etapas: uma alocação inicial de tarefas em dois conjuntos com base em um valor de corte para um indicador z, definido a seguir, e uma segunda etapa de alocação baseada na heurística Hte.

3.2.1 Etapa I: Pré-alocação baseada em z_{corte}

Na primeira etapa, é escolhido um valor de corte para a relação b_i/a_i e todos serviços que excedem esse valor são alocados neste conjunto. De forma a uniformizar essa medida, em vez de utilizar a relação b_i/a_i, é proposto um indicador equivalente $z = (b_i - a_i)/(b_i+a_i)$, que pode variar entre -1 e 1, dependendo do conjunto que a tarefa tem mais afinidade. O valor de corte é chamado de z_{corte}. As tarefas com $z > z_{corte}$, indicando $b_i \gg a_i$ são alocados no conjunto E, e os itens com $z < -z_{corte}$ são alocados no conjunto T, sempre observando se o conjunto E ainda possui capacidade para mais tarefas.

Figura 2: Fluxo da heurística proposta.

3.2.2 Etapa II: Alocação baseada em Hte e busca determinística

Na segunda etapa, os itens que restaram no grupo intermediário, chamado de **grupo sem preferência**, são alocados conforme preconiza a heurística Hte e reordenados da forma ótima conforme a propriedade em V. Vale ressaltar que o valor z de cada serviço permanece como critério de desempate entre tarefas com relações a_i/p_i e b_i/p_i idênticas.

Uma busca determinística também foi inserida nesta etapa, de forma a procurar em E tarefas interessantes de se enviar para T. Com base no maior a_i/p_i das tarefas do conjunto E, a busca inicia removendo a tarefa mais interessante de E e inserindo em T. As tarefas são então reordenadas de acordo com a propriedade em V e o objetivo é calculado. Caso este melhore em relação ao anterior, a busca continua. Senão, o penúltimo objetivo é utilizado.

3.2.3 Etapa III: Busca em z's

Foi observado que melhores objetivos de algumas instâncias eram obtidos com z_{corte} mais apertados (entre 0,75 e 0,9), e para outras com valores menos restritivos ($z_{corte} = 0{,}5$). Por esse motivo foi inserida uma terceira etapa, que resolve os problemas com z_{corte} de 0,25, 0,50, 0,75, 0,9 e 2,0, e consolida os menores valores encontrados para cada problema.

No caso de z=2,0, não existe a pré-alocação na Etapa I, pois $z \in \{0, 1\}$. Neste caso, a heurística passa a funcionar de forma similar à Hte, com a adição de uma ordenação baseada no indicador z, que passa a funcionar como critério de desempate nos casos em que a_i/p_i (ou b_i/p_i) de tarefas sucessivas das listas ordenadas são iguais.

3.3 Resultados computacionais - Heurística construtiva

Na Tabela 2 são apresentados os resultados dos experimentos dos 280 problemas em comparação aos resultados dos ótimos/limites obtidos por Biskup. Nesta, conseguimos observar que, para problemas mais restritivos (h = 0,2 ou h = 0,4), a heurística é efetiva

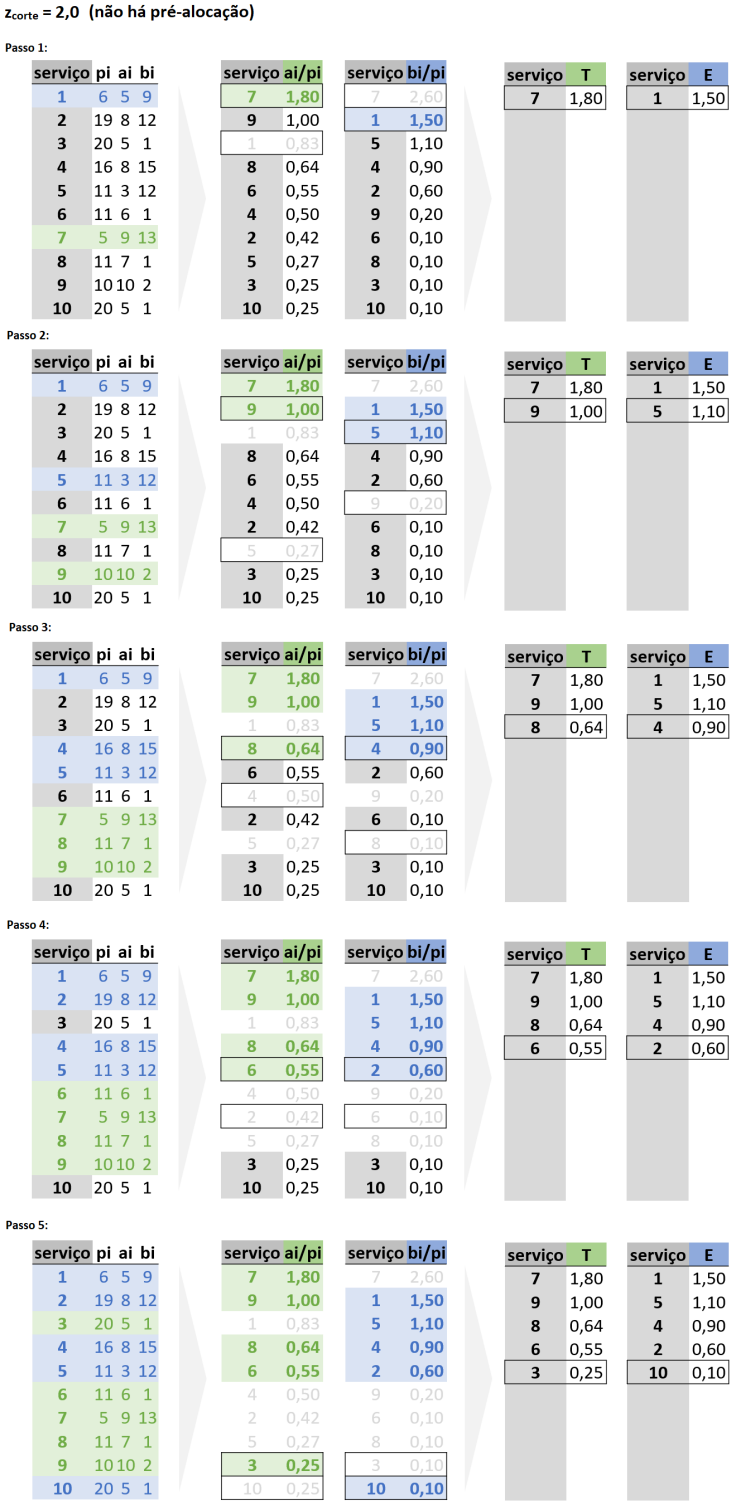


Figura 3: Exemplo gráfico da heurística construtiva com $z_{corte} = 2{,}0$ (sem pré-alocação de tarefas).

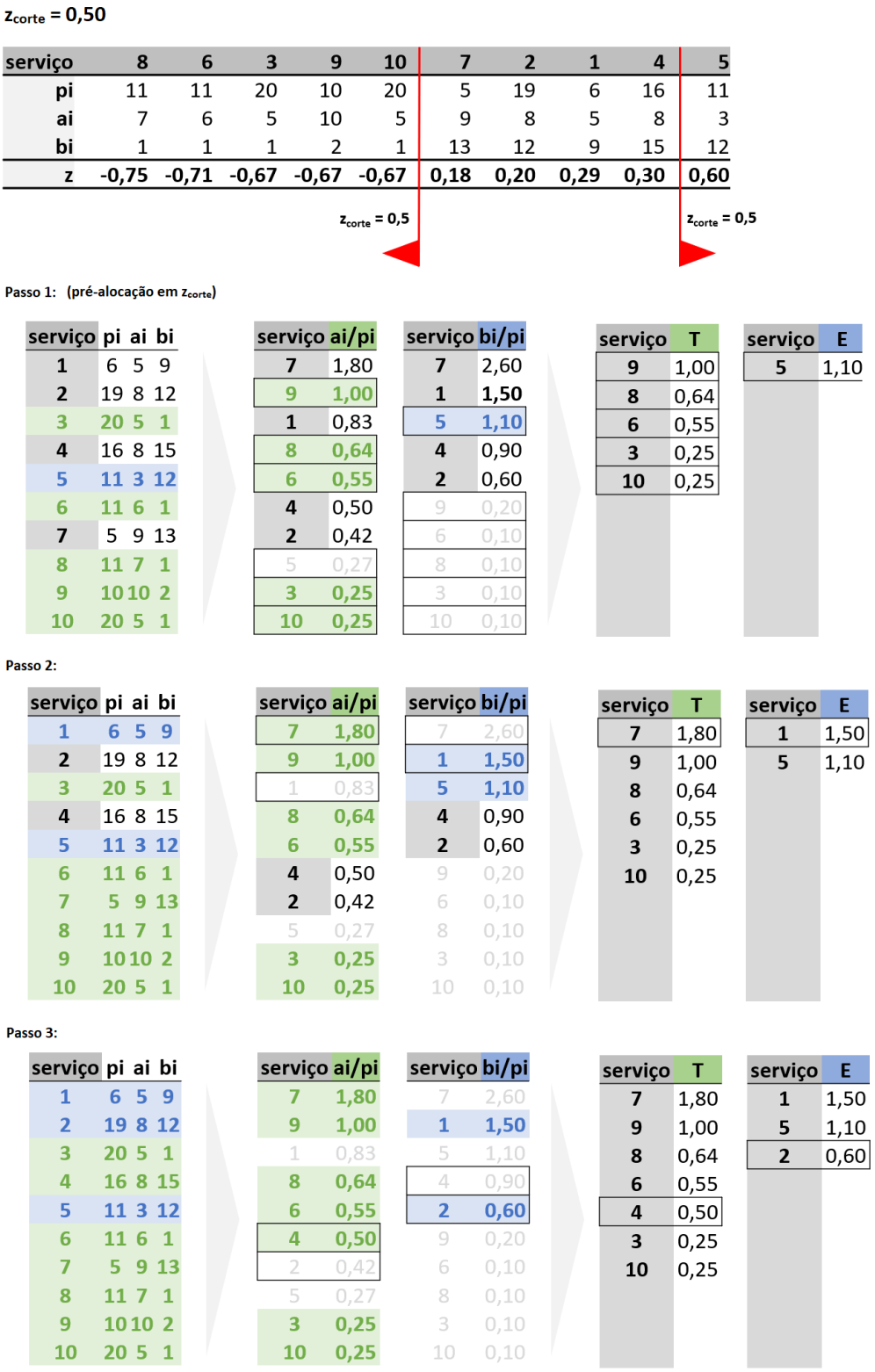


Figura 4: Exemplo gráfico da heurística construtiva com $z_{corte} = 0{,}50$ (com pré-alocação de tarefas).

Tabela 2: Comparação dos objetivos obtidos pelos experimentos da heurística construtiva com os apresentados por Biskup e Feldmann[1]. Cada célula representa a variação (em porcentagem) da média dos 10 problemas de cada experimento para os conjuntos n e parâmetros h.

Dif.(%) / **h**	**n** **10**	**20**	**50**	**100**	**200**	**500**	**1000**	**Média**	**Hte**
0,2	6,74	-1,29	-4,30	-5,10	-4,88	-5,35	-5,88	**-2,87**	**-3,89**
0,4	12,09	0,67	-2,59	-3,39	-2,44	-2,23	-3,07	**-0,14**	**-0,26**
0,6	1,61	0,36	0,39	0,25	0,06	0,01	0,04	**0,39**	**1,63**
0,8	2,15	0,48	0,07	0,23	0,06	0,01	0,04	**0,43**	**0,86**
Média	**5,65**	**0,05**	**-1,61**	**-2,00**	**-1,80**	**-1,89**	**-2,22**	**-0,55**	**-0,42**

e apresentou um melhor resultado para os problemas com pelo menos 50 tarefas. Além disso, há uma melhora gradual conforme o aumento do número de tarefas, sendo o pior desempenho para problemas de 10 tarefas e o melhor para 1000 tarefas.

Se forem desconsiderados os problemas pequenos (até 20 tarefas) e menos restritivos (h maior ou igual a 0,6), esta encontrou objetivos entre 2,23% e 5,88% menores do que Biskup. Considerando todos os problemas, a heurística deste trabalho obteve um resultado em média -0,55% menor.

Os melhores resultados obtidos para h a partir de 0,6 são provenientes de $z_{corte} = 0{,}5$, enquanto os melhores resultados obtidos para h até 0,4 são provenientes de $z_{corte} = 0{,}9$ e $z_{corte} = 2{,}0$ (dados completos por z_{corte} disponíveis no arquivo **export.xlsx** anexo). Isso mostra que a busca em z_{corte} foi essencial para a obtenção dos melhores resultados consolidados, e que a elaboração de uma busca local mais robusta pode melhorar os resultados obtidos para h a partir de 0,6.

Na última coluna da Tabela 2 estão os resultados disponíveis da heurística original Hte na literatura. No geral, a Hte teve um desempenho melhor para cenários menos restritivos (h a partir de 0,6), mas a melhoria proposta neste trabalho trouxe melhores resultados consolidados (média de -0,55% contra 0,42% do original).

3.4 Tempos de processamento

Na Tabela 3 estão disponíveis os tempos médios de execução para cada conjunto de problemas de dimensões n. Os tempos de processamento aumentam linearmente conforme a dimensão do problema aumenta, indicando uma complexidade O(n) para o algoritmo proposto. A Figura 5 indica a relação linear entre os tempos de execução e as dimensões do problema: a regressão linear entre ambos apresentou um R^2 de 99,9%.

Os experimentos foram realizados com programação na linguagem Python e em uma máquina com as seguintes configurações:

Tabela 3: Tempos médios de processamento dos experimentos da heurística construtiva para as dimensões n.

n	**10**	**20**	**50**	**100**	**200**	**500**	**1000**
t (s)	0,003	0,005	0,011	0,021	0,042	0,112	0,237

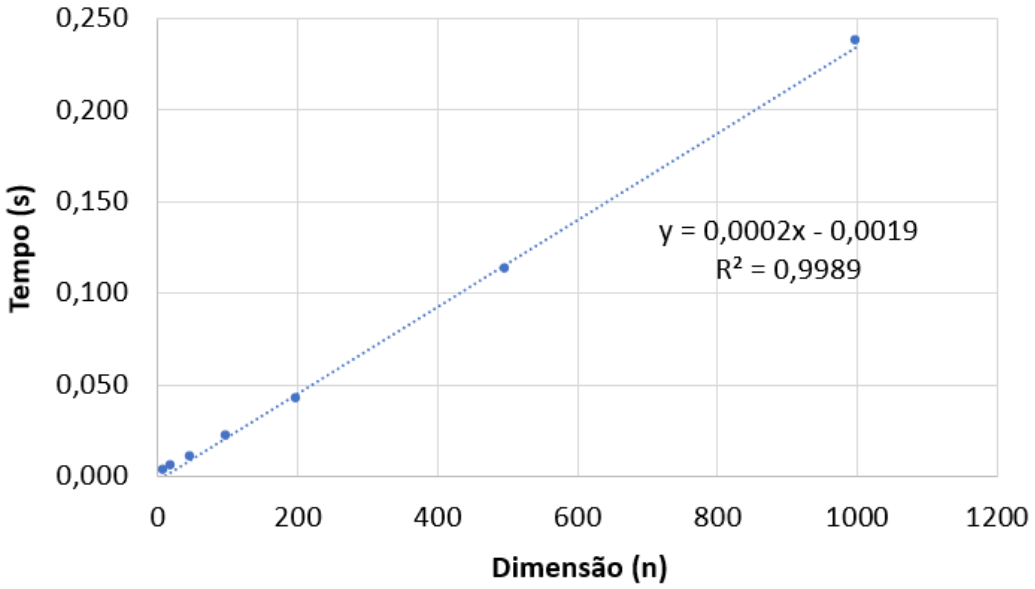


Figura 5: Tempos de execução em segundos vs. dimensões n.

- Processador: Intel(R) Core(TM) i7-8750H CPU @ 2,20GHz 2,21GHz.
- Memória RAM: 8GB.

4 Busca Local

Nessa seção será descrita a etapa da busca local. A busca local será definida pela vizinhança, movimento e critério de parada, em seguida serão apresentados os resultados, evidenciando a melhora sobre a solução obtida com a heurística construtiva e comparando com o Biskup. A busca é realizada com cada uma das soluções obtidas com os diferentes z_{corte}.

4.1 Vizinhança e movimento

Desconsiderando a possibilidade do último serviço no grupo adiantado não terminar na data de entrega e lembrando que a posição dos serviços podem ser determinadas pela

propriedade V-shape e que o problema possui apenas dois grupos, pode-se definir de maneira mutualmente excludente e completamente exaustiva qual a sequência das tarefas definindo apenas se uma tarefa está adiantada ou não. Em outras palavras, se um serviço não termina antes da data, ele termina depois e a sequência dos serviços serão definidos pela sequência V-shape.

Essas considerações permitem que uma solução seja definida unicamente por uma lista do tamanho do número de tarefas com a informação verdadeiro ou falso, sendo que os serviços indicados com verdadeiro está programado para terminar antes da data de entregue.

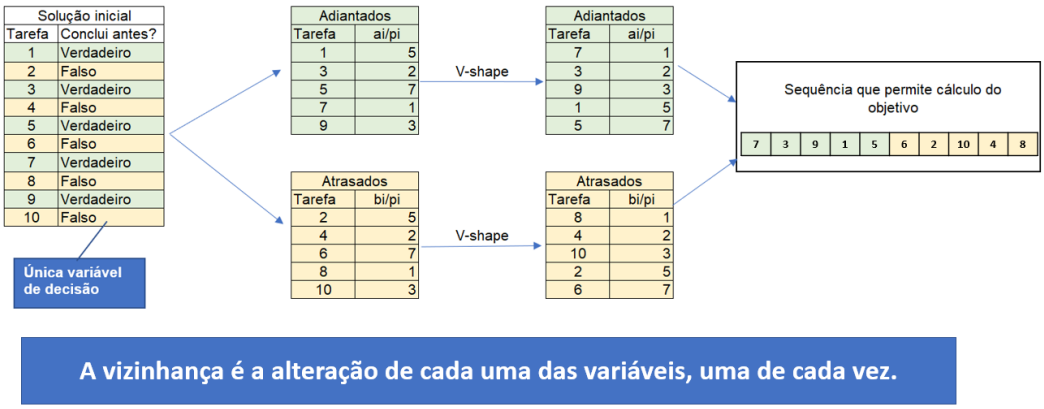


Figura 6: Exemplo gráfico da vizinhança para uma instância de dimensão 10.

O movimento considerado na implementação aqui apresentada é a inserção de cada tarefa no conjunto oposto ao inicialmente programado na solução inicial, também é conhecido por operador de Hamming, que consiste em inverter um único bit. [4]

O movimento escolhido é o **best-improvement**, isto é, toda a vizinhança de uma solução é avaliada e a solução que mais melhora o objetivo é escolhida - a factibilidade de cada movimento também é avaliada nesta etapa. A nova vizinhança é então explorada e o algoritmo continua até atingir o parâmetro que limita o máximo de vizinhanças exploradas **LV**.

Caso não haja movimentos que melhorem a solução incumbente, o movimento é realizado, i.e., aceita-se movimentos piores limitado a um parâmetro de número máximo de soluções piores aceitas **LSP**.

4.2 Critério de parada

Foram adotados dois critérios de parada para o algoritmo da busca local:

- **LV = 100** - Limite de vizinhanças exploradas
- **LSP = máximo(n*0,10, 10)** - Número máximo de soluções piores aceitas

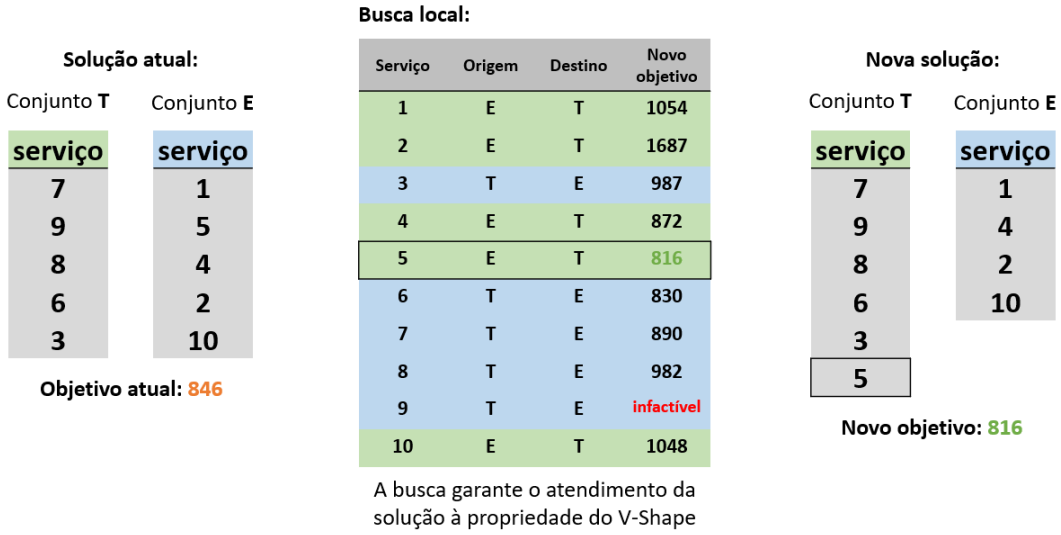


Figura 7: Exemplo gráfico indicando o funcionamento do movimento de inserção para uma instância de dimensão 10.

O limite de vizinhanças exploradas é fixo, igual a 100, enquanto o número máximo de soluções piores aceitas depende da dimensão do problema. Este último, por ser dependente da dimensão, evita que iterações desnecessárias entre vizinhanças ocorram para as dimensões menores, da mesma forma que garante um número de iterações suficientes para que as dimensões maiores saiam do ótimo local.

O algoritmo para quando o primeiro critério de parada, entre **LV** e **LSP**, for atingido.

4.3 Resultados computacionais - Busca local

Na Tabela 4 são apresentados os resultados dos experimentos da combinação da heurística construtiva mais a busca local dos 280 problemas em comparação aos resultados dos ótimos/limites obtidos por Biskup. Os melhores resultados continuam sendo para problemas mais restritivos (h até 0,4) e dimensões a partir de 50. O resultado final foi de -1,22% em relação à média geral encontrada por Biskup.

Apesar de as dimensões 10 e 20 ainda serem as que mais precisam de melhoria no resultado, quando os resultados da busca local são comparados com os da heurística construtiva, conforme mostra a Tabela 5, houve uma melhora considerável destas dimensões (-2,58% e -0,66%, respectivamente). Quando analisamos a Tabela 2 com a Tabela 4, vemos que o resultado médio das dimensões 10 e 20 passou de 5,65% para 2,72% e 0,05% para -0,61%, respectivamente. No geral, a busca local apresentou uma melhoria média de -0,63% em relação à heurística construtiva desenvolvida neste trabalho.

Tabela 4: Comparação dos objetivos obtidos pelos experimentos da busca local com os apresentados por Biskup e Feldmann[1]. Cada célula representa a variação (em porcentagem) da média dos 10 problemas de cada experimento para os conjuntos n e parâmetros h.

Dif.(%) **h** \ **n**	**10**	**20**	**50**	**100**	**200**	**500**	**1000**	**Média**
0,2	4,60	-2,01	-4,88	-5,46	-5,02	-5,43	-5,88	**-3,44**
0,4	6,26	0,13	-2,93	-3,83	-2,64	-2,29	-3,10	**-1,20**
0,6	0,02	-0,32	0,08	-0,14	-0,15	-0,11	-0,06	**-0,10**
0,8	0,00	-0,25	-0,22	-0,17	-0,15	-0,11	-0,06	**-0,14**
Média	**2,72**	**-0,61**	**-1,99**	**-2,40**	**-1,99**	**-1,99**	**-2,27**	**-1,22**

Tabela 5: Comparação dos objetivos obtidos pelos experimentos da heurística construtiva mais a busca local com os experimentos obtidos pela heurística construtiva, apenas. Cada célula representa a variação (em porcentagem) da média dos 10 problemas de cada experimento para os conjuntos n e parâmetros h.

Dif.(%) **h** \ **n**	**10**	**20**	**50**	**100**	**200**	**500**	**1000**	**Média**
0,2	-2,00	-0,73	-0,60	-0,38	-0,14	-0,09	0,00	**-0,56**
0,4	-4,87	-0,54	-0,35	-0,46	-0,21	-0,06	-0,03	**-0,93**
0,6	-1,47	-0,67	-0,31	-0,39	-0,22	-0,12	-0,09	**-0,47**
0,8	-2,00	-0,73	-0,29	-0,39	-0,22	-0,12	-0,09	**-0,55**
Média	**-2,58**	**-0,66**	**-0,39**	**-0,40**	**-0,19**	**-0,10**	**-0,05**	**-0,63**

Tabela 6: Tempos médios de processamento dos experimentos para as dimensões n.

n	**10**	**20**	**50**	**100**	**200**	**500**	**1000**
t (s)	0,176	0,015	0,035	0,085	0,452	4,951	21,145

4.4 Tempos de processamento

Os tempos médios de processamento de cada dimensão estão disponíveis na Tabela 6 e na Figura 8.

Os experimentos foram realizados com programação na linguagem Python e em uma máquina com as seguintes configurações: Processador: Intel(R) Core(TM) i5-8265U CPU @ 1,60GHz 1,80GHz; Memória RAM: 8GB. Pode-se observar que o tempo de processamento da heurística de melhoria implementada não foi linear com a dimensão dos problemas, devido a necessidade de se calcular o objetivo de todos vizinhos e esse cálculo também depender do tamanho da instância.

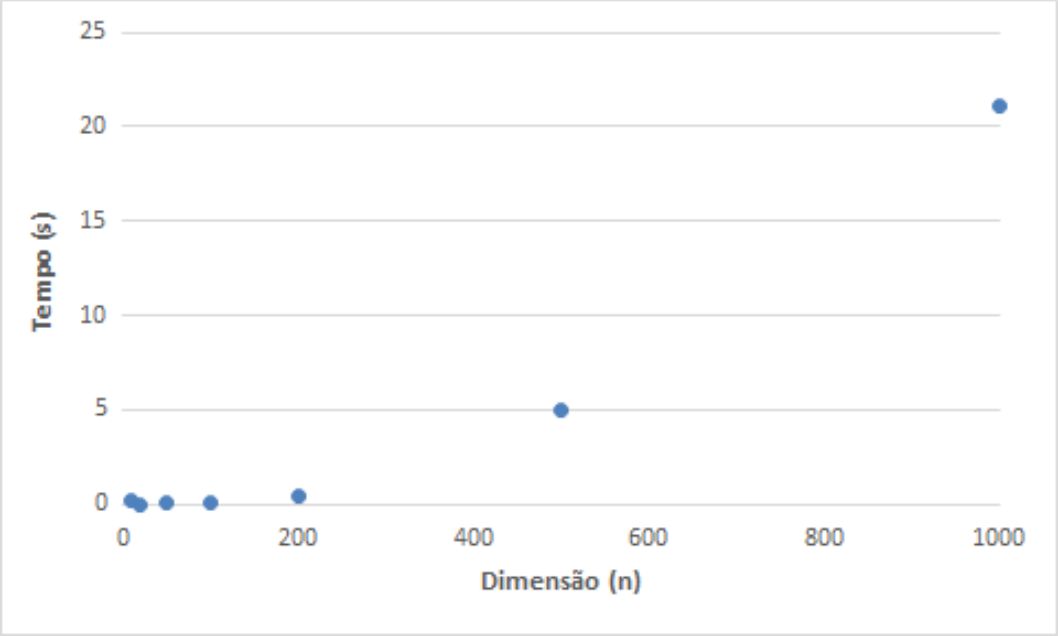


Figura 8: Tempos de execução da busca local em segundos vs. dimensões n.

5 Metaheurística - Algoritmo Genético

5.1 Ideia geral

Baseado na teoria da seleção natural, o **algoritmo genético** foi escolhido para a aplicação da metaheurística neste trabalho. O algoritmo genético é um algoritmo probabilístico, no qual é gerada uma população inicial de **indivíduos** (soluções), representados na forma de **cromossomos** (transcrição da solução em um *array* binário, de forma que cada número deste seja um de seus genes), e a cada **geração** (iteração) os melhores indivíduos possuem **maiores chances** (avaliados por uma função de *aptidão*) de **sobreviverem** (escolha dos sobreviventes), se **reproduzirem** (cruzamento entre os reprodutores) e evoluírem. Na Seção 5.2

No final das gerações, o melhor indivíduo estará presente na população e representará a melhor solução encontrada pelo algoritmo.

Em algoritmos genéticos existe um balanço que precisa ser ponderado entre a pressão evolutiva e a diversidade da população. Quão maior a pressão evolutiva, mais rapidamente o algoritmo converge para um ótimo local. Por outro lado, quão maior a diversidade da população, este tende a convergir mais lentamente, mas para soluções melhores. Diversos parâmetros são utilizados para controlar esse balanço, e os utilizados neste trabalho serão especificados na Seção 5.3.

5.2 Estrutura do algoritmo genético

A cada geração do algoritmo genético a população de indivíduos passa por uma série de operações, conforme descritas na seção 5.1, e que, juntas, permitem a evolução da população conforme as gerações passam.

A Figura 9 mostra o fluxo das operações do algoritmo genético implementado. Nas seções seguintes, detalhamos a lógica de cada decisão que estas tomam.

5.2.1 Representação cromossômica

Algoritmos genéticos requerem que soluções sejam codificadas. Essa codificação faz analogia ao DNA, que através de cromossomos definem as características dos seres vivos.

A codificação adotada para as soluções do problema, chamadas de cromossomos, precisam ser representadas de maneira mutualmente excludente e completamente exaustiva, isto é, a partir de um cromossomo deve ser possível definir completamente uma única solução. Esse processo é chamado decodificação.

Também é interessante definir os cromossomos de formas que seja conveniente realizar as operações de codificação, decodificação, cruzamento e mutação.

Considerando que se for definido quais tarefas terminam antes da data e quais depois, a ordem e os tempos de início de cada uma das tarefas é definida pelas propriedades I e II, descritas na Seção 2.1. O que resta a ser decidido no problema estudado é quais tarefas estarão antes (no subconjunto dos adiantados) ou depois (no subconjunto dos atrasados) da data de entrega em comum.

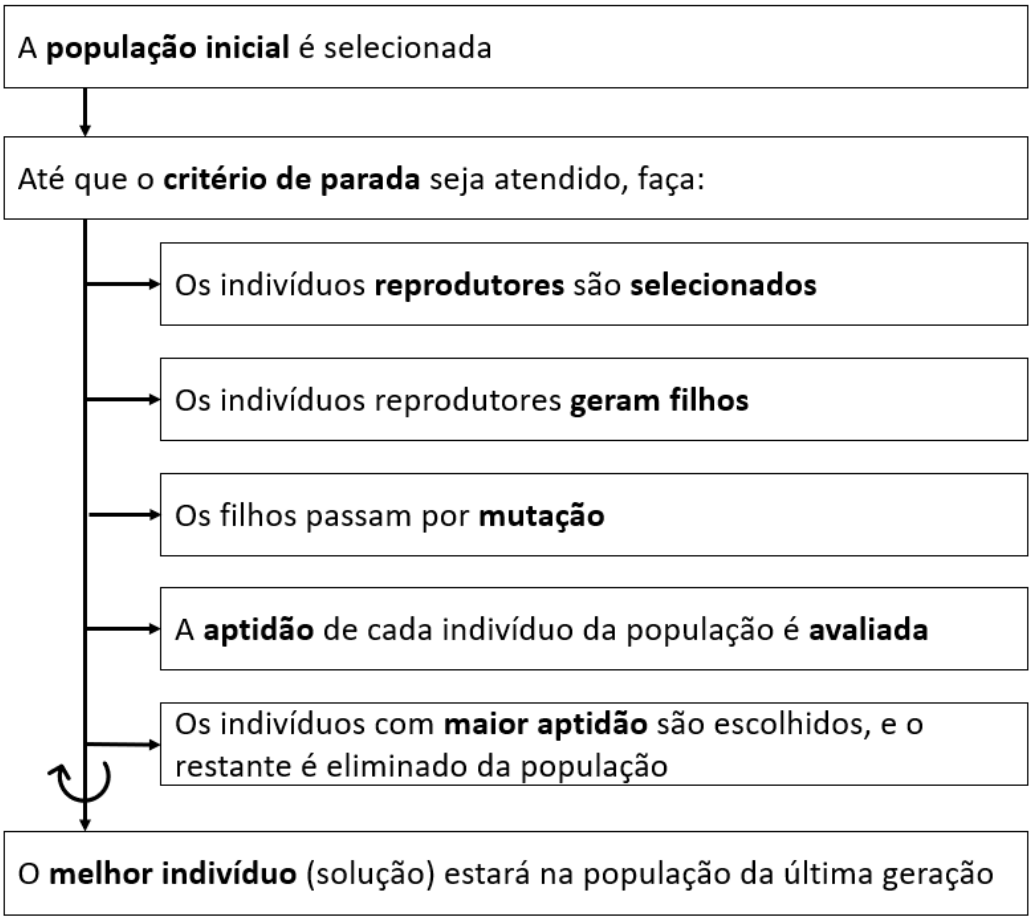


Figura 9: Fluxograma geral do algoritmo genético implantado.

Dessa forma, a representação cromossômica adotada é um *array* de booleanos, no qual a posição no array representa o número da tarefa, e o valor em cada uma dessas posições indica se a tarefa em questão está adiantada (recebe 1) ou atrasada (recebe 0). Na Figura 10 está a representação cromossômica de uma programação exemplo.

5.2.2 População inicial

Na Seção 3.2 definimos o indicador z_{corte}, responsável pela alocação inicial das tarefas na heurística construtiva. Para cada valor de z_{corte}, a heurística construtiva e a busca local são executadas. No final da execução dessas duas etapas, existe uma melhor solução para cada z_{corte} configurado, e estas melhores soluções são definidas como a **população inicial** e **reprodutores iniciais** do algoritmo genético, servindo de sementes para a formação da população total da primeira geração.

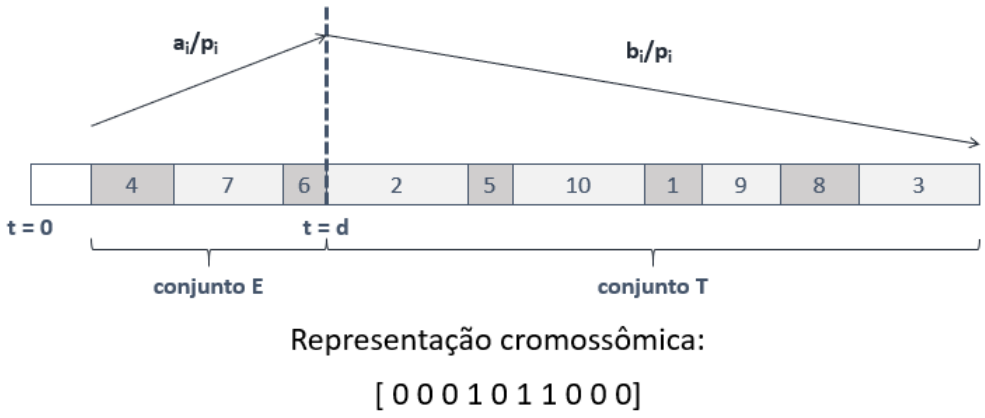


Figura 10: Representação cromossômica de uma solução que segue as propriedades do problema estudado.

5.2.3 Seleção dos reprodutores

A **população de pais** é definida por uma porcentagem fixa da população total ordenada crescentemente pela melhor aptidão. Para cada filho gerado por um cruzamento, é necessário escolher dois reprodutores pertencentes à população de pais pré-definida. Para essa seleção, foram definidos dois métodos distintos:

- **Coliseu**: No Coliseu, é gerado um torneio para cada reprodutor selecionado. O torneio é uma competição entre pais competidores, que são sorteados com a mesma chance entre todos da população de pais, para definir quem é o melhor reprodutor (maior aptidão). Um parâmetro **n** define quantos pais participarão de cada torneio, que neste trabalho foi fixado em 2 pais. Para cada filho gerado de um cruzamento, são executados dois coliseus. Neste método, a pior aptidão da população de pais não possui chances de se reproduzir, a menos que esteja empatada com os **n** piores.

- **Roleta**: Na Roleta é realizado um sorteio envieasado, onde as melhores aptidões da população de pais recebem um incentivo para serem sorteadas a partir da população de pais. Uma parcela dos melhores pais, definidos pela variável *taxa_elitismo*, recebem 33% das chances totais de serem sorteados da roleta. Os 66% restantes de chance são distribuídos entre o restante dos pais da população de pais. Neste método, as chances dos piores pais serem sorteados é reduzida, mas não nula.

O tipo de seleção de parentes adotado é controlado pelo parâmetro *"usa_coliseu"*.

5.2.4 Cruzamento

Cruzamento é uma das operações de diversificação da população. A operação é análoga ao cruzamento de seres vivos, onde a prole herda características dos parentes.

Consiste em gerar uma nova solução unindo partes de indivíduos da população, que devido a seleção de indivíduos mais aptos, contém características de interesse.

Foram implementados dois tipos de cruzamento:

- **Cruzamento simples**: O indivíduo é gerado a partir da primeira metade do cromossomo do primeiro parente, com a segunda metade do cromossomo do segundo parente. Neste tipo de cruzamento um determinado par de parentes sempre gerará sempre o mesmo indivíduo.
- **Cruzamento aleatório**: É realizado um sorteio para cada gene que define de qual parente o indivíduo herdará o gene. Neste tipo de cruzamento um par de parentes pode gerar indivíduos distintos.

  Caso o indivíduo gerado seja infactível ele será reparado.

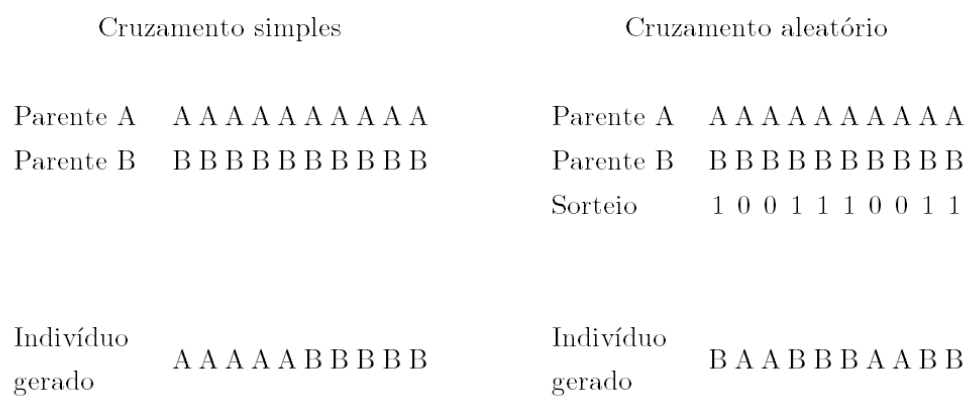


Figura 11: Exemplo de cada tipo de cruzamento.

5.2.5 Mutação

Mutação é uma das operações de diversificação da população. A operação é análoga a mutação de seres vivos, onde um individuo pode apresentar características não encontradas em seus parentes. Caso a característica adquirida por meio da mutação melhore a aptidão do individuo, ela aumentará a chance deste individuo se reproduzir e se disseminará pela população. No algoritmo genético a operação de mutação permite que características não encontradas na população tenham a chance de serem adquiridas, visto que se fossem realizados apenas cruzamentos, uma característica não fosse encontrada na população nunca surgiria. Na implementação realizada foi adotado um parâmetro que controla a probabilidade de cada gene da solução ser modificada. Devido a variedade na dimensão dos problemas, foi necessário tornar essa taxa proporcional ao tamanho do problema. O parâmetro de controle é chamado *"taxa_mutação_inicial"*. A probabilidade de cada gene ser modificado é definido pela *taxa mutação inicial dividida pelo tamanho do problema*, de forma a manter a proporção de genes modificados a cada operação de mutação

constante entre as diversas dimensões dos problemas. Nesta etapa também ocorre o reparo de indivíduos infactíveis.

5.2.6 Reparação de infactibilidade

Cada filho gerado, seja por cruzamento ou mutação, passa por uma avaliação de factibilidade, i.e., é verificado se a soma dos tempos de processamento das tarefas adiantadas da solução que o filho representa cabem no prazo de entrega **d**.

Caso o filho seja infactível, segue-se o seguinte procedimento: i) as tarefas do conjunto de adiantadas são ordenadas de forma decrescente pelo indicador **z**, definido na Seção 3.2.1; ii) identifica-se, na ordenação decrescente em z, o mínimo de tarefas que é necessário transferir do conjunto dos adiantados para o dos atrasados, de forma a viabilizar a solução; iii) a transferência é então efetivada, sensibilizando a troca dos genes das tarefas identificadas do filho (de 1 para 0) e retornando-o para o fluxo do algoritmo.

5.2.7 Avaliação da aptidão

A função de avaliação da aptidão tem a função de terminar a qualidade do indivíduo. E a qualidade do indivíduo é determinada pelas características que se deseja que o individuo tenha.

Nesta implementação a aptidão é relacionada com um custo menor.

A função de avaliação da aptidão calcula o objetivo do cromossomo transformando-o numa sequência que segue a propriedade do V-shape com a última tarefa adiantada terminando exatamente em "d"ou com a primeira tarefa adiantada começando em t=0, o que for mais vantajoso.

5.2.8 Escolha dos sobreviventes

Após a realização dos cruzamentos e mutações os indivíduos gerados são temporariamente incluídos na população. Neste momento é realizada uma operação para selecionar os indivíduos que irão compor a próxima geração. A etapa de escolha de sobreviventes implementada conta com uma parcela das vagas dos sobreviventes reservadas para soluções de elite, ordenadas pela aptidão, e o restante é distribuído aleatoriamente entre as demais soluções. A parcela da população reservada para as melhores soluções é determinada pelo parâmetro *"taxa_elitismo"*. Essa forma de escolher os sobreviventes foi escolhida com o objetivo de aumentar a pressão evolutiva.

5.2.9 Critério de parada

O problema, principalmente nas instâncias com maior número de tarefas, não conta com uma maneira eficiente de verificar se a otimalidade foi atingida. Por esse motivo, é preciso definir um critério de parada para o algoritmo. Critérios de parada usuais são número de gerações, número de genes avaliados e número de gerações sem melhoria e tempo de execução. O critério de parada adotado foi o número de gerações, e foi escolhido por ser a

forma mais simples de tornar o tempo de execução previsível e proporcional ao tamanho do problema.

5.3 Parametrização

O algoritmo genético é composto por diversos processos sujeitos a parâmetros de controle, alguns usuais em algoritmos genéticos, outros específicos da implementação desenvolvida neste trabalho. O funcionamento detalhado dos parâmetros já foi descrito nos itens anteriores. A seguir eles serão relacionados dando ênfase ao seu efeito no funcionamento do algoritmo:

- **Tamanho da população**: O tamanho da população é o número de indivíduos que passam de uma geração para a outra. Influi tanto na pressão evolutiva quanto na diversidade, em problemas com espaço de solução muito grande é recomendado adotar valores maiores. Adotar um valor pequeno pode restringir a diversidade e causar a convergência prematura da população, que passaria a ser composta de diversas réplicas das mesmas soluções.

- **Número de gerações**: O número de gerações controla quantas vezes o algoritmo é repetido. Está relacionado com a área do espaço de soluções que deseja-se que seja explorado. Valores baixos produzirão resultados pobres, e valores muito grandes pode ser um desperdício de recursos.

- **Taxa de mutação**: Controla a inclusão de novos genes na população. Valores muito altos torna a busca aleatória. Valores muito baixos restringe a diversidade, causando uma convergência prematura, interrompendo o processo de evolução.

- **Tamanho do torneio**: Define a quantidade de indivíduos que disputam uma vaga de parente. É um número inteiro maior que dois.

- **Elitismo - Seleção de parentes**: Controla o viés utilizado na seleção dos indivíduos que produzirão a próxima geração. Valores muito altos permitirão apenas que uma pequena parcela dos melhores se reproduzam, restringindo a diversidade. Valores baixos limitará a pressão evolutiva, tornando a busca aleatória e menos alinhada ao conceito da evolução.

- **Elitismo - Seleção de sobreviventes**: Controla o viés utilizado na seleção dos indivíduos que sobrevivem para a próxima geração. O efeito é idêntico ao do elitismo na seleção de parentes.

- **Tipo de cruzamento**: Define qual dos cruzamentos implementados é utilizado. A seleção de pais baseada em torneio, além do viés introduzido pelo elitismo, incentiva a seleção de parentes melhores, por comparar dois indivíduos antes de o escolher para cruzar. Por exemplo, a melhor solução tem mais chances que a segunda melhor, pois se forem selecionadas para um confronto, a primeira será selecionada. A seleção por sorteio não tem preferência entre os melhores. Apenas separa os

indivíduos entre soluções de elite, com maior chance, e os demais. Neste caso, a primeira e segunda melhores soluções têm a mesma chance de cruzar.

5.3.1 Considerações sobre a definição dos parâmetros

Conhecendo o efeito dos parâmetros, resta definir qual o valor que deve ser escolhido para cada um deles. Esta não é uma pergunta simples, e a literatura conta com diversos meios iterativos e não iterativos para responder essa pergunta. Nos métodos não iterativos são definidos algumas alternativas para cada parâmetro e são realizados testes, da análise desses testes os parâmetros são definidos. Entre os métodos iterativos testes iniciais dão origem a mais testes que vão melhorando a qualidade dos parâmetros iterativamente. Não existem parâmetros que são ótimos para diversos tipos de problema, e encontra-se exemplo de diferença de performance dos parâmetros entre instâncias do mesmo problema. Eiben e Smith apresentam um capítulo inteiro dedicado a métodos para definir os parâmetros de algoritmos evolutivos[2].

5.3.2 Testes e premissas para configuração dos parâmetros

A principio foram realizados alguns testes para identificar faixas onde o algoritmo era capaz de evoluir a população. Nos testes preliminares, por exemplo, utilizando a taxa_elitismo=0.1 a população não evoluiu, nem com 5000 iterações. Nesse ponto foi necessário adotar algumas premissas para simplificar o testes, pois o algoritmo possuía pelo menos 7 parâmetros, que se tivessem duas opções cada, seriam necessários 128 execuções. Considerando a diferença de performance entre instâncias de problemas idênticos, seria necessário realizar os testes em pelo menos duas instâncias de cada uma das 7 dimensões de problema para um dos quatro níveis de restrição da data de entrega. Apenas esse teste exigiria 7168 repetições, considerando um tempo médio de 45s por repetição, tomaria mais de 90 horas. Por esse motivo foram adotadas a seguintes premissas:

- **Tamanho da população**: Foi fixado em 500 indivíduos.
- **Número de gerações**: Foi fixado em 1000 gerações.
- **Tamanho do torneio**: Foi fixado em 2 competidores.
- **Elitismo - seleção de parentes e de sobreviventes**: Foi unido em apenas um parâmetro - *taxa_elitismo*.
- **Tipo de crossover**: Foi adotado o cruzamento simples.

Escolheu-se testar os demais parâmetros com os seguintes valores:

- **taxa_mutacao_inicial**: 0.5, 1 ou 5, lembrando que o valor efetivo da mutação é este número dividido pelo tamanho do problema. O valor maior foi escolhido de forma que a taxa de mutação não fosse maior que 50% para nenhuma dimensão de problema, o que tornaria a mutação menos aleatória.

- **taxa_elitismo**: 35%, 75% ou 100%.
- **usa_coliseu**: 0 ou 1.

Nestes testes também identificamos que o tipo de cruzamento não tinha efeito aparente no objetivo encontrado, porém o cruzamento simples era dez vezes mais rápido, por isso optou-se pela sua escolha.

Foi definido também que os testes seriam realizados em duas das dez instâncias de cada tamanho, para cada limitação da data. Para limitar o efeito da aleatoriedade nos resultados, cada instância foi otimizada três vezes. O valor considerado na análise dos resultados é a média das três execuções.

5.3.3 Resultados da otimização dos parâmetros

O resultado dos testes de parametrização foram, quanto ao objetivo, inconclusivos, isto é, alterar os parâmetros do algoritmo dentro da faixa estabelecida no item anterior não teve impacto significativo na qualidade das soluções. Por este motivo, o critério de desempate entre as opções foi a performance, que também só foi significativa para a escolha do tipo de seleção de parentes, onde a seleção por roleta demandou menos recursos que a seleção por torneio.

Algumas figuras ilustrando os resultados da parametrização são apresentadas a seguir:

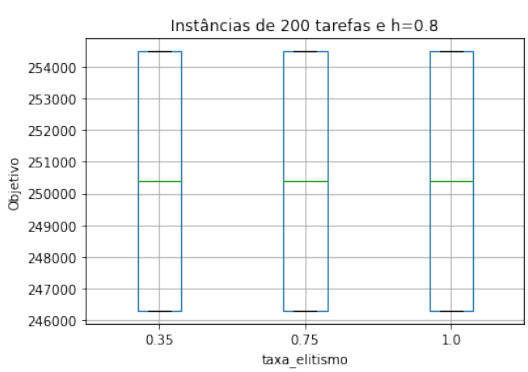
5.4 Resultados computacionais

5.4.1 Ajuste no cálculo do objetivo

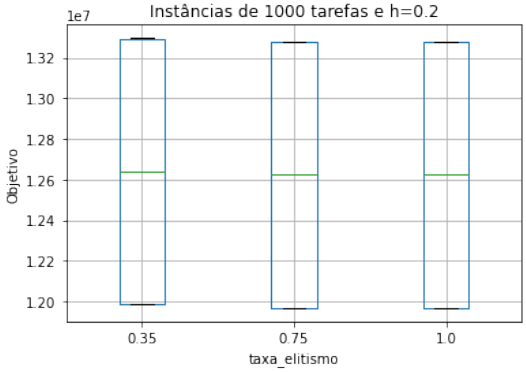
Houve um ajuste no cálculo do objetivo, que foi aplicado também na etapa da heurística construtiva e busca local, além do algoritmo genético. Por este ter tido um impacto relevante na melhoria dos objetivos encontrados, isolamos o efeito desta alteração do algoritmo genético. Os resultados desta subseção podem ser considerados uma atualização dos resultados obtidos na Seção 4.3 e, também, um novo ponto de partida para avaliação do impacto da metaheurística desenvolvida.

O método de cálculo anterior considerava que a última tarefa do conjunto dos adiantados finalizava exatamente na data de entrega comum **d**. No novo método, consideramos, também, a opção de a primeira tarefa do conjunto dos adiantados iniciar em 0, conforme propriedade I do problema, já descrita neste trabalho na Seção 2.1.1. Assim, o objetivo calculado é o menor entre iniciar no tempo 0 ou finalizar no tempo d. Vale reforçar aqui que não há alteração nos conjuntos dos adiantados e atrasados, i.e., caso a "folga"entre o tempo 0 e o início da primeira tarefa do conjunto dos adiantados seja maior que o tempo de processamento da primeira tarefa do conjunto dos atrasados, a solução não vai ter opção de cálculo de objetivo iniciando no tempo 0. Outro ponto importante é que a solução de qualquer um dos objetivos calculados obedecem ao V-shape. A Figura 14 exemplifica a diferença entre os dois métodos e como ambos respeitam as propriedades do problema.

Na Tabela 7 está a comparação do ajuste no cálculo objetivo com os valores obtidos por Biskup e Feldmann[1]. Comparando em relação a Biskup, os objetivos desta etapa

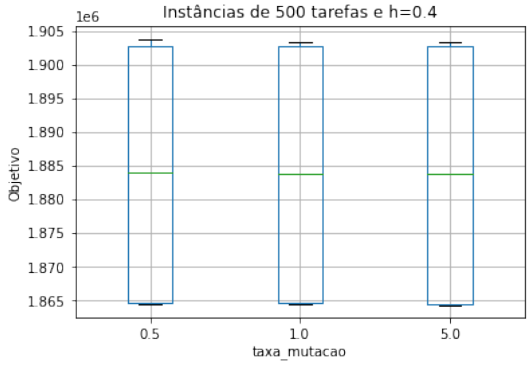


(a) Instâncias de tamanho 200 para h=0.8

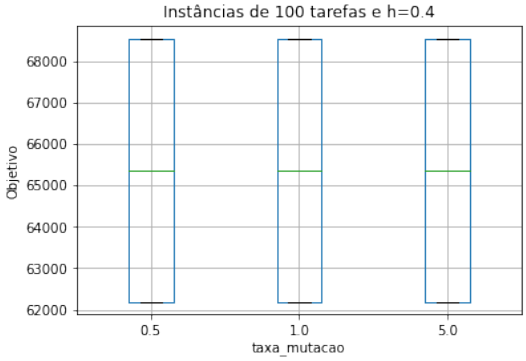


(b) Instâncias de tamanho 1000 para h=0.2

Figura 12: Exemplos da variação no objetivo para alterações na taxa_elitismo



(a) Instâncias de tamanho 500 para h=0.4



(b) Instâncias de tamanho 100 para h=0.6

Figura 13: Exemplos da variação no objetivo para alterações na taxa_mutacao

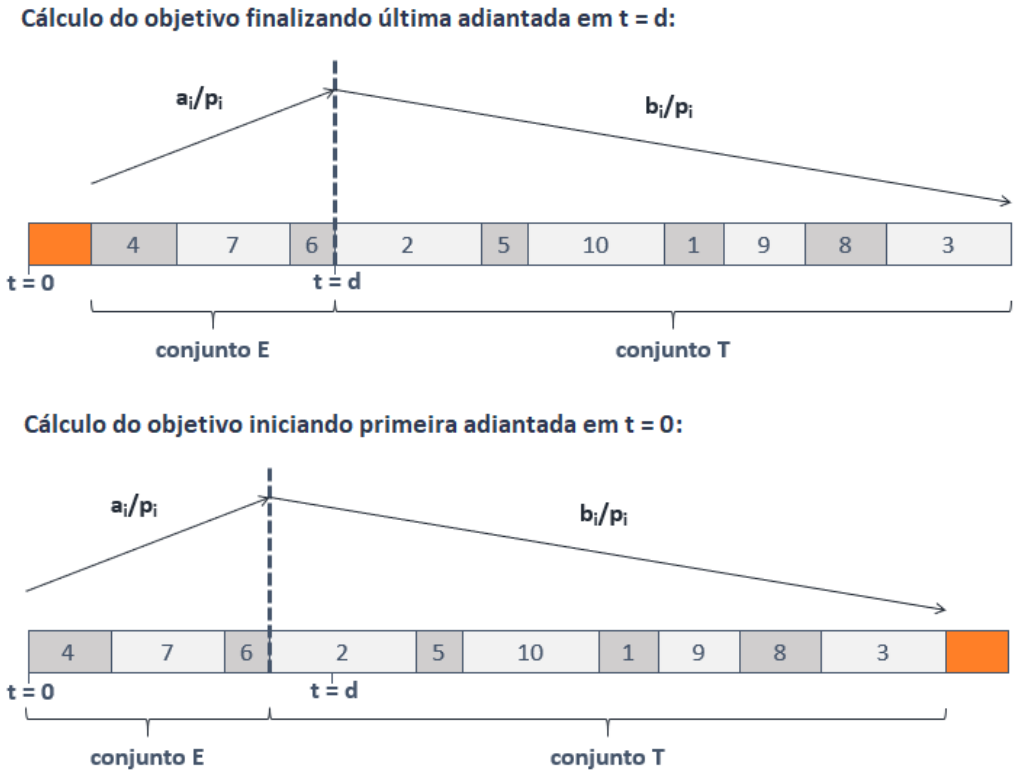


Figura 14: Exemplo gráfico da diferença entre os diferentes cálculos de objetivo para uma mesma instância. O objetivo final será o menor entre os dois objetivos calculados.

ficaram em média **1,47%** menores, sendo que as mais impactadas foram as instâncias de h = 0,2.

Na Tabela 8 está a comparação com os resultados obtidos anteriormente com a heurística construtiva e busca local. Em relação aos resultados obtidos anteriormente com a heurística construtiva e busca local, a média dos objetivos ficou em **0,24%** menor, melhorando principalmente as instâncias dos conjuntos 10 e 20, para h até 0,4.

5.4.2 Resultados do algoritmo genético

Os parâmetros utilizados para os testes do algoritmo genético estão definidos a seguir:

- **População inicial**: 1000
- **Método de seleção dos reprodutores**: Roleta
- **Taxa mutação**: 1/conjunto (variável conforme dimensão do problema)
- **Taxa de elitismo**: 1.0

Tabela 7: Comparação dos objetivos obtidos pelos experimentos da heurística construtiva mais a busca local após o ajuste no cálculo do objetivo com os apresentados por Biskup e Feldmann[1]. Cada célula representa a variação (em porcentagem) da média dos 10 problemas de cada experimento para os conjuntos n e parâmetros h.

Dif.(%)	**n**							
h	**10**	**20**	**50**	**100**	**200**	**500**	**1000**	**Média**
0,2	0,97	-2,80	-4,94	-5,51	-5,02	-5,43	-5,88	**-4,09**
0,4	4,56	-0,24	-3,13	-3,89	-2,75	-2,29	-3,11	**-1,55**
0,6	0,01	-0,42	-0,08	-0,14	-0,15	-0,11	-0,06	**-0,11**
0,8	0,00	-0,25	-0,23	-0,17	-0,15	-0,11	-0,06	**-0,14**
Média	**1,38**	**-0,93**	**-2,06**	**-2,43**	**-2,02**	**-1,99**	**-2,28**	**-1,47**

Tabela 8: Comparação dos objetivos obtidos pelos experimentos da heurística construtiva mais a busca local **após o ajuste** no cálculo do objetivo com os experimentos da heurística construtiva mais a busca local **antes do ajuste** do objetivo. Cada célula representa a variação (em porcentagem) da média dos 10 problemas de cada experimento para os conjuntos n e parâmetros h.

Dif.(%)	**n**							
h	**10**	**20**	**50**	**100**	**200**	**500**	**1000**	**Média**
0,2	-3,36	-0,80	-0,06	-0,06	-0,00	-0,00	-0,00	**-0,61**
0,4	-1,54	-0,36	-0,21	-0,06	-0,12	-0,01	-0,01	**-0,33**
0,6	-0,01	-0,10	0,00	0,00	0,00	0,00	0,00	**-0,02**
0,8	0,00	0,00	-0,02	0,00	0,00	0,00	0,00	**0,00**
Média	**-1,23**	**-0,31**	**-0,07**	**-0,03**	**-0,03**	**-0,00**	**-0,00**	**-0,24**

Tabela 9: Comparação dos objetivos obtidos pelos experimentos do algoritmo genético com os apresentados por Biskup e Feldmann[1]. Cada célula representa a variação (em porcentagem) da média dos 10 problemas de cada experimento para os conjuntos n e parâmetros h.

Dif.(%)	**n**							
h	**10**	**20**	**50**	**100**	**200**	**500**	**1000**	**Média**
0,2	0,12	-3,84	-5,70	-6,19	-5,78	-6,41	-6,72	**-4,93**
0,4	0,19	-1,62	-4,65	-4,94	-3,75	-3,58	-4,39	**-3,24**
0,6	0,01	-0,72	-0,34	-0,15	-0,15	-0,11	-0,06	**-0,22**
0,8	0,00	-0,41	-0,24	-0,18	-0,15	-0,11	-0,06	**-0,16**
Média	**0,08**	**-1,65**	**-2,73**	**-2,86**	**-2,46**	**-2,56**	**-2,81**	**-2,14**

- **Critério de parada**: limite de 1000 gerações

Foram executadas 4 repetições do algoritmo genético. O resultado final consolida a média dos objetivos mínimos de cada instância de cada repetição.

Na Tabela 9 estão os resultados finais do algoritimo genético comparado com Biskup. Este melhorou os objetivos principalmente das instâncias de h até 0,4, sendo mais efetivo para instâncias de conjuntos maiores ou iguais a 50. Os resultados finais ficaram em média **2,14%** menores do que as referências de Biskup, sendo que para o conjunto 10 chegou-se muito próximo dos valores ótimos do problema.

Na Tabela 10 está a comparação dos resultados do algoritmo genético em relação aos resultados etapa da heurística construtiva mais a busca local, após o ajuste do cálculo do objetivo. No geral, houve **0,65%** em média de redução nos objetivos, melhorando principalmente as instâncias dos conjuntos 10 e 20 e de h 0,4.

5.4.3 Comparação com resultados da literatura

Na Figura 15 está a comparação dos resultados obtidos neste trabalho com demais resultados encontrados na literatura[3]. Destacados em caixas verdes, estão os melhores resultados disponíveis para cada instância, e que então foram resumidos na coluna **Melhores**. Na última coluna, são identificadas as instâncias que ainda há oportunidade de melhoria por parte deste trabalho, considerando como mínimo os melhores resultados obtidos na literatura.

No geral, os melhores resultados da literatura alcançaram em média resultados **2,16%** menores do que os de Biskup. O algoritmo desenvolvido neste trabalho alcançou os melhores resultados disponíveis na literatura para 18 das 28 instâncias (**ca. 64%** do total) e, em outras 6 (**ca. 21%**), ficou a no máximo **0,02%** de diferença do melhor. Há oportunidade de melhoria para as instâncias do conjunto 10, no qual ainda é possível atingir os seus ótimos globais.

Tabela 10: Comparação dos objetivos obtidos pelos experimentos do algoritmo genético com os experimentos da heurística construtiva mais a busca local **após o ajuste** do objetivo. Cada célula representa a variação (em porcentagem) da média dos 10 problemas de cada experimento para os conjuntos n e parâmetros h.

Dif.(%) **h** \ **n**	**10**	**20**	**50**	**100**	**200**	**500**	**1000**	**Média**
0,2	-0,83	-1,06	-0,80	-0,72	-0,80	-1,04	-0,89	**-0,88**
0,4	-3,40	-1,39	-1,57	-1,09	-1,03	-1,32	-1,32	**-1,59**
0,6	0,00	-0,30	-0,41	-0,01	0,00	0,00	0,00	**-0,10**
0,8	0,00	-0,16	-0,01	-0,01	0,00	0,00	0,00	**-0,02**
Média	**-1,06**	**-0,73**	**-0,70**	**-0,46**	**-0,46**	**-0,59**	**-0,55**	**-0,65**

n	h	FB (2003)	HGT (2005)	GA (2005)	VNS/TS (2007)	LCC (2007)	LCY (2007)	Nosso algoritmo	Melhores	Ainda temos a melhorar
10	0.2	0.00	−0.12	−0.12	0.00	0.00	−0.08	-0,12%	0,00%	0,12%
	0.4	0.00	−0.19	−0.19	0.00	0.00	0.00	-0,19%	0,00%	0,19%
	0.6		−0.01	−0.01	0.00	0.00	−0.01	-0,01%	0,00%	0,01%
	0.8		0.00	0.00	0.00	0.00	0.00	0,00%	0,00%	0,00%
20	0.2	3.84	3.84	3.84	3.84	3.84	3.79	3,84%	3,84%	0,00%
	0.4	1.63	1.62	1.62	1.63	1.63	1.58	1,62%	1,63%	0,01%
	0.6		0.71	0.68	0.72	0.72	0.64	0,72%	0,72%	0,00%
	0.8		0.41	0.28	0.41	0.41	0.39	0,41%	0,41%	0,00%
50	0.2	5.65	5.70	5.68	5.70	5.70	5.58	5,70%	5,70%	0,00%
	0.4	4.64	4.66	4.60	4.66	4.66	4.42	4,65%	4,66%	0,02%
	0.6		0.31	0.31	0.34	0.34	0.31	0,34%	0,34%	0,00%
	0.8		0.23	0.19	0.24	0.24	0.24	0,24%	0,24%	0,00%
100	0.2	6.18	6.19	6.17	6.19	6.19	6.12	6,19%	6,19%	0,00%
	0.4	4.94	4.93	4.91	4.94	4.94	4.85	4,94%	4,94%	0,00%
	0.6		0.04	0.12	0.15	0.15	0.15	0,15%	0,15%	0,00%
	0.8		0.11	0.12	0.18	0.18	0.18	0,18%	0,18%	0,00%
200	0.2	5.73	5.76	5.74	5.78	5.78	5.76	5,78%	5,78%	0,00%
	0.4	3.79	3.75	3.75	3.75	3.75	3.73	3,75%	3,75%	0,00%
	0.6		0.07	0.13	0.15	0.15	0.15	0,15%	0,15%	0,00%
	0.8		0.07	0.14	0.15	0.15	0.15	0,15%	0,15%	0,00%
500	0.2	6.40	6.41	6.41	6.42	6.43	6.43	6,41%	6,43%	0,02%
	0.4	3.52	3.58	3.58	3.56	3.58	3.57	3,58%	3,58%	0,00%
	0.6		0.15	0.11	0.11	0.11	0.11	0,11%	0,15%	0,04%
	0.8		0.13	0.11	0.11	0.11	0.11	0,11%	0,13%	0,02%
1,000	0.2	6.72	6.74	6.75	6.75	6.77	6.77	6,72%	6,77%	0,05%
	0.4	4.30	4.39	4.40	4.37	4.40	4.40	4,39%	4,40%	0,01%
	0.6		0.42	0.05	0.05	0.06	0.06	0,06%	0,06%	0,00%
	0.8		0.40	0.05	0.05	0.06	0.06	0,06%	0,06%	0,00%
Average			2.06	2.12	2.15	2.16	2.12	**2,14%**	**2,16%**	0,02%

Figura 15: Comparação dos resultados obtidos pelo nosso trabalho com os melhores resultados disponíveis da literatura[3]. Nas caixas em verdes estão destacados os melhores resultados para cada conjunto **n** e **h**.

Tabela 11: Tempos médios de processamento dos experimentos do algoritmo genético para as dimensões n.

n	**10**	**20**	**50**	**100**	**200**	**500**	**1000**
t (s)	16,15	20,81	37,69	63,68	67,19	128,40	212,41

5.4.4 Tempos de processamento

Os tempos médios de processamento de cada dimensão estão disponíveis na Tabela 11 e na Figura 16. Estes tempos referem-se somente à etapa de execução do algoritmo genético, desconsiderando os demais tempos de execução da heurística construtiva com busca local.

Os experimentos foram realizados com programação na linguagem Python e em uma máquina com as seguintes configurações: Processador: Intel(R) Core(TM) i5-8265U CPU @ 1,60GHz 1,80GHz; Memória RAM: 8GB. O crescimento do tempo de execução em face do aumento da dimensão **n** mostra que, na etapa do algoritmo genético, a complexidade ficou **menor** do que a **O(n)** da primeira etapa de heurística construtiva (da dimensão 10 para dimensão 1000 - 100 vezes maior - o tempo médio de execução aumentou ca. 10 vezes).

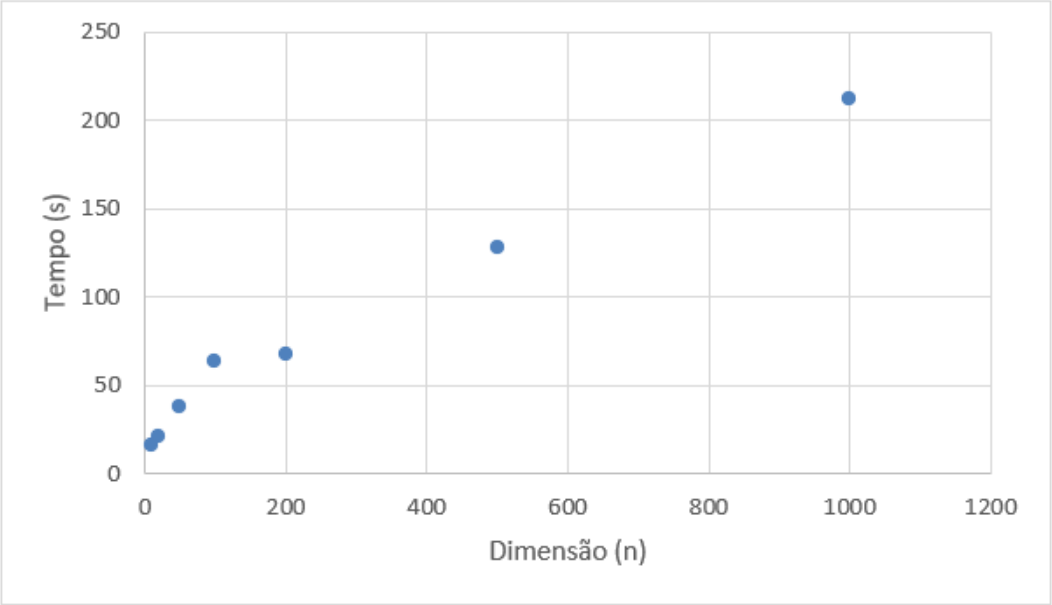


Figura 16: Tempos de execução do algoritmo genético em segundos vs. dimensões n.

6 Conclusão

Os resultados obtidos demonstram que uma abordagem mista que a princípio priorize os serviços pela diferença entre os seus coeficientes a_i e b_i em seguida por quanto os serviços remanescentes penalizarão a função objetivo é uma estratégia válida, principalmente em problemas com grande número de itens e restrição na capacidade do conjunto E, visto que a heurística de referência H_{te} apresentava resultados melhores justamente nos casos com h mais alto.

Os próximos passos para continuar aprimorando a heurística pode envolver uma limitação na quantidade de serviços que seriam pré-alocados, seja esse limitante uma parcela do número de serviços do problema ou como uma parcela do tempo disponível no conjunto E.

O conceito do z_{corte} também poderia ter sido implementado em conjunto de outras heurísticas para avaliar seu potencial.

Na implementação da etapa de busca pode-se observar que mesmo uma busca local simples, com poucos recursos para sair de ótimos locais, é capaz de produzir melhorias, mesmo em soluções muito próximas ou melhores que as encontradas na literatura. Observa-se na prática a capacidade do algoritmo genético de obter resultados melhores em pouco tempo, mesmo em problemas com grande número de soluções.

Mesmo abordando um único problema, pode-se observar comportamento distinto entre diferentes iterações e principalmente em problemas com características diferentes, aqui representado pela variação na restrição da data de entrega, onde soluções muito próximas dos melhores resultados foram encontrados ainda na etapa da heurística construtiva, por exemplo os problemas com 1000 tarefas e h=0,8. A dificuldade em se atingir o melhor resultado aumenta conforme a restrição da data de entrega aumenta.

Entender a fundo o funcionamento dos algoritmos de metaheurísticas, assim como gerar indicadores que podem auxiliar na análise e visualização da execução destes (ex: diversidade da população ao longo das gerações para o algoritmo genético), também foi importante para se atingir melhores objetivos.

Uma das conclusões mais valiosas está relacionada com a importância da atenção as características do problema na escolha e no desenvolvimento do método de otimização. Neste trabalho, uma interpretação cuidadosa das propriedades do problema direcionou com sucesso os esforços, resultando em soluções no nível do estado da arte.

Resumo da comparação dos resultados obtidos a cada etapa deste trabalho com os de referência do Biskup:

- **Heurística costrutiva**: -0,55%
- **Busca local**: -1,22% e -1,47% (pré e após ajuste do cálculo no objetivo, respectivamente)
- **Algoritmo Genético**: -2,14%

7 Anexos

Aqui estão descritos os arquivos que seguem anexos a este relatório:

- **export_GA.xlsx**: planilha Excel com os resultados dos experimentos do algoritmo genético.
- **GA_parametrizado.py**: código Python com o fluxo de execução da solução completa.
- **funcoes_BuscaLocal.py, funcoes_GA.py, funcoes_gerais.py**: códigos Python com os métodos utilizados no GA_parametrizado.py.
- **ApresentacaoGA.pptx**: apresentação da etapa de metaheurística.
- **resultado_final.xlsx**: planilha com os dados dos testes e resultados finais.

Referências

[1] Dirk Biskup and Martin Feldmann. Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. *Computers & Operations Research*, 28(8):787–801, 2001.

[2] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer Berlin Heidelberg, 2015.

[3] Bassem Jarboui, Patrick Siarry, and Jacques Teghem. *Metaheuristics for Production Scheduling (ISTE)*. Wiley-IEEE Press, 1st edition, 2013.

[4] R.C. Martí, P.M. Pardalos, and M.G.C. Resende. *Handbook of Heuristics*. Springer, 2018.