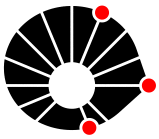

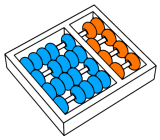

Comparação de algoritmos para o problema do *flowshop* com m máquinas

Felipe Pavan

Relatório Técnico - IC-PFG-19-54
Projeto Final de Graduação
2019 - Novembro

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Comparação de algoritmos para o problema do *flowshop* com m máquinas

Felipe Pavan*

Resumo

Este documento contém a descrição do trabalho sobre o problema do escalonamento de n tarefas com prazos de término em m máquinas utilizando-se diferentes algoritmos sendo eles, meta-heurística Busca Tabu, BRKGA (Bias Random-Key Genetic Algorithm) e variações de ambos. A escolha de diferentes algoritmos para a resolução do problema tem como objetivo determinar quais estratégias provém melhores soluções e discutir a relação entre a qualidade da solução e as ideias principais exploradas por cada algoritmo. Os resultados serão avaliados utilizando o mesmo conjunto de problemas para todas as implementações, com diferentes número de tarefas, número de máquinas e valores de prazo. Os dados serão coletados e analisados conjuntamente para se fazer um comparação uniforme com a mesma métrica entre todas as variações e métodos utilizados para a resolução do problema.

1 Introdução

O problema do escalonamento de tarefas (*flowshop problem*) é um problema muito estudado em otimização combinatória. Apesar ter uma definição simples, o problema possui um grau de dificuldade elevado, o que atrai a atenção de pesquisadores. No *flowshop problem* temos um conjunto de tarefas $N = 1, ..., n$ que necessitam processamento em um conjunto $M = 1, ..., m$ de máquinas. Todas as tarefas seguem o mesmo caminho nas máquinas e $p_{ij} \geq 0$ representa o tempo de processamento da tarefa $j \in N$ na máquina $i \in M$. Entre os critérios normalmente utilizados, a minimização do tempo de conclusão máximo é amplamente utilizado. A simplificação de considerar somente permutações de escalonamentos (a ordem de processamento das tarefas é a mesma para todas as máquinas) é muito comum e o problema resultante é chamado o problema da permutação do escalonamento de tarefas. Este problema foi mostrado por [1] como sendo NP-completo para $m \geq 3$. Em plantas industriais e situações da vida real, critérios baseados em prazos de entrega são mais importantes do que aqueles baseados em tempo de conclusão. Seja d_j, $j \in N$, o prazo para a tarefa j, o atraso da tarefa é definido como $T_j = max\{C_j - d_j, 0\}$, onde C_j é o tempo de conclusão da tarefa. O objetivo do problema é minimizar o atraso total de todas as tarefas: $\sum_{j=1}^{n} T_j$. Este problema é NP-difícil mesmo para uma máquina como mostrado em [2]. Temos um exemplo do problema para $n = 4$ e $m = 3$ na Figura 1.

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

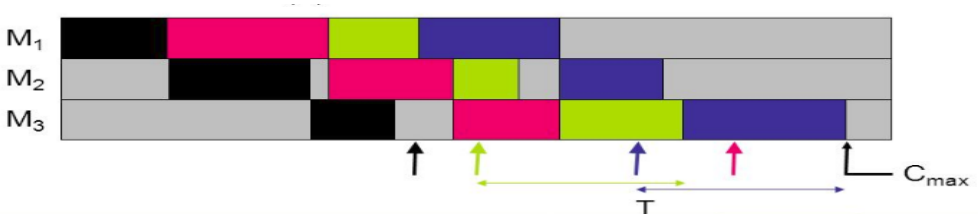

Figura 1: Exemplificação do problema do *flowshop*, onde as setas coloridas indicam os prazos de suas respectivas tarefas, neste caso da imagem temos duas tarefas atrasadas, concluídas em um tempo posterior ao indicado pela seta. Com o tempo de completude total indicado por *Cmax*.

Soluções ótimas para esse problema podem ser obtidas apenas através de métodos exatos como *branch-and-bound*. Porém esses métodos exigem quantidades muito grandes de computação e se tornam impraticáveis para problemas grandes. Em vista destes problemas surgiram heurísticas para auxiliar na solução do problema. A partir de uma solução inicial gerada por alguma heurística construtiva, são utilizadas heurísticas de melhoramento dessa solução, como técnicas de busca na vizinhança, utilizada em meta-heurísticas como Busca Tabu. Busca Tabu é um método de otimização baseado em busca local que é usado para resolver problemas de otimização combinatória e tem tido grande sucesso, especialmente em problemas da área de escalonamento. Ainda hoje soluções melhores são encontradas a partir de novas técnicas para o desenvolvimento de algoritmos para o *flowshop problem*. Em [3] é utilizado um algoritmo guloso iterativo híbrido. Em [4] é utilizado um design automático de algoritmos de busca local para se encontrar aquele que produz as melhores soluções. Em ambas referências são encontrados melhores resultados daqueles utilizados para a comparação neste trabalho [5].

Neste trabalho serão avaliados a meta-heurística Busca Tabu, o algoritmo genético BRKGA e variações de ambos algoritmos, com o intuito de obter melhores resultados, para o problema da permutação do escalonamento de n tarefas em m máquinas. Os resultados da meta-heurística Busca Tabu foram referentes à melhor implementação obtida, onde foram exploradas técnicas de diversificação e intensificação e métodos de aceleração para exploração das vizinhanças. Também será avaliada a implementação padrão do BRKGA

[1], onde o método de decodificação avalia o indivíduo da população, variando-se o tamanho da população. Também será avaliada uma versão que tenta melhorar o método de decodificação aplicando rotinas sobre o indivíduo com o objetivo de aprimorá-lo. Além disso, foi utilizado um conjunto de problemas para se avaliar os algoritmos implementados e assim poder fazer uma análise geral dos resultados. Trata-se de um subconjunto das instâncias de testes de [5].

A sequência deste trabalho está dividida da seguinte forma: na seção 2 são desenvolvidos os principais conceitos dos algoritmos implementados, suas características e variações utilizadas neste trabalho. A seção 3 elabora como foi feito a análise dos resultados e apresenta um exemplo de instância. Em seguida nas seções 4 e 5 temos os resultados e uma análise dos mesmos, respectivamente. Ao final são apresentadas conclusões tiradas a partir do desenvolvimento do trabalho.

2 Metodologia

Meta-heurísticas são métodos gerais que guiam a busca pelo espaço de soluções, complementado por alguma heurística e usualmente utilizando-se de busca local. Começando a partir de uma solução inicial, meta-heurísticas procuram melhorar esta solução inicial iterativamente até que algum critério de parada seja alcançado. O critério de parada pode ser número de iterações, número de avaliações da função objetivo ou tempo de processamento gasto pelo algoritmo.

2.1 Busca Tabu

A meta-heurística Busca Tabu parte de uma solução heurística inicial e aplica uma busca na vizinhança a cada iteração. Além disso, para evitar máximos locais, eventualmente são permitidas iterações nas quais não há melhora na função objetiva. Por fim, para evitar ciclagem, é mantida uma Lista Tabu, que contém pontos que não podem ser visitados, como por exemplo, os pontos que já foram avaliados.

Naturalmente, diversos parâmetros desse algoritmo influenciam seu comportamento e por consequência, a qualidade dos resultados apresentados: visitar a vizinhança com uma estratégia *best-improving* ou *first-improving*, tamanho da Lista Tabu - chamada *tabu tenure*, entre outros.

Para o problema proposto, uma solução foi dada por uma sequência de tarefas s, com $\mid s \mid \leq n$. Nas iterações da busca na vizinhança foram intercambiadas tarefas nesta solução em busca de alguma ação que melhore o valor da função objetivo.

É possível implementar estratégias alternativas para o algoritmo, sendo a Busca Tabu Probabilística e a Diversificação exemplos de tais estratégias. Outras variações podem ser encontradas em [6].

[1] http://mauricio.resende.info/src/brkgaAPI/

2.1.1 Espaço de busca e estrutura de vizinhança

O espaço de busca da Busca Tabu é simplesmente o espaço de todas a soluções possíveis que podem ser consideradas durante a busca. No caso do problema proposto, o espaço é composto por todas as permutações possíveis das n tarefas a serem escalonadas nas m máquinas. A definição de uma vizinhança não é muito distante de um espaço de busca. A cada iteração da Busca Tabu, as transformações possíveis que podem ser aplicadas na solução S em consideração, definem o conjunto de soluções vizinhas de S, chamado $N(S)$. No problema proposto a vizinhança de cada solução é uma troca de posições entre tarefas.

2.1.2 Lista tabu

Tabus são elementos que distinguem Busca Tabu quando comparada com uma busca local simples. Como mencionado anteriormente, a lista de tabus é utilizada para prevenir ciclagem quando se faz um movimento em outra direção do ótimo local através de movimentos que não melhoram a solução. Quando tal movimento ocorre o algoritmo precisa de uma memória dos movimentos anteriores para não retornar pelo mesmo caminho. Isto é alcançado através da lista tabu que consiste de movimentos que reverteriam o efeito de movimentos recentes. Para a implementação proposta a lista tabu consiste em um par de tarefas que foram recentemente intercambiadas, proibindo que uma troca envolvendo ambas tarefas seja feita.

2.1.3 Critério de aspiração

Em alguns casos a lista de tabus pode proibir movimentos atrativos, mesmo quando não há problema de ciclagem causada pelo movimento na vizinhança. Tendo isso em vista é necessário algum dispositivo algorítmico que permita tais movimentos que sobreponham os tabus. Estes dispositivos são chamados critérios de aspiração. O critério mais simples e mais comumente utilizado em implementações de Busca Tabu consiste em permitir um movimento, mesmo que presente na lista de movimentos proibidos, caso este gere um valor de solução objetivo melhor que o valor da melhor solução conhecida até agora.

2.1.4 Heurística de construção da solução inicial

Para a construção da solução inicial usaremos heurísticas classificadas como regra de despacho, amplamente conhecidas para a construção de um escalonamento. Esse tipo de regra é comumente utilizada na prática na sequência inicial utilizada em meta-heurísticas. Nesta seção vamos apresentar a heurística usada para o problema do atraso total do escalonamento em m máquinas e também uma segunda heurística que é utilizada pela primeira. Seja s a sequência de tarefas já escalonadas e t o tempo no qual as tarefas estão sendo consideradas para seleção. Seja $C_j(s)$ o tempo de conclusão da tarefa $j \notin s$ se a tarefa for escalonada no fim da sequência s.

- EDD: No tempo t, a tarefa com menor valor de d_j é selecionada
- NHE_{EDD}: Partindo-se de uma permutação obtida a partir da heurística anterior (EDD), a construção da nova solução se dá inicialmente pelas duas tarefas iniciais.

Em seguida, para cada tarefa ainda não escalonada na segunda solução, calcula-se a melhor posição para a inserção dessa tarefa e a insere nesta posição. Assim itera-se até todas as tarefas terem sidos escalonadas na nova solução. Esta heurística tem complexidade $O(n^2)$, mais custosa computacionalmente quando comparadas a outras heurísticas que possuem complexidade $O(n)$, porém esta apresenta melhores resultados [5].

2.1.5 Meta-heurísticas

Para este trabalho é proposto a implementação da seguinte meta-heurística Busca Tabu para o problema do escalonamento em m máquinas.

- TS1: Os movimentos na vizinhança são baseados na troca de posição entre duas tarefas e uma avaliação total da vizinhança é feita. A lista tabu consiste de pares de tarefas, ou seja, é proibido fazer a troca de posição entre duas tarefas que foram recentemente trocadas entre si.

2.1.6 Busca Tabu Probabilística

Na Busca Tabu é preciso avaliar a qualidade dos pontos na vizinhança da solução atual. Tal processo pode envolver um cálculo da função objetivo, potencialmente custoso, para avaliar se há melhora em cada vizinho, além de verificar se o vizinho sendo considerado viola a Lista Tabu. Uma forma de mitigar esses dois problemas é considerar apenas um subconjunto dos vizinhos definido aleatoriamente: tal aleatoriedade torna o passo do algoritmo de mover-se para uma solução vizinha menos custosa e a chance de ciclagem acontecer menor. A implementação desta técnica é interessante pois conforme as instâncias dos problemas tornam-se muito grandes, a computação necessária se torna cada vez maior, então reduzir a carga de trabalho do algoritmo pode ser interessante para explorar porções diferentes do espaço de busca em menos tempo.

2.1.7 Diversificação

Um dos problemas dos métodos baseados em busca local, incluindo Busca Tabu apesar dos benefícios da lista tabu, é a tendência de se restringir a uma porção pequena do espaço de busca na maior parte do seu tempo de procura pela solução. A principal consequência negativa disso seria que, apesar de possivelmente encontrar-se boas soluções, o algoritmo pode falhar em explorar as partes mais interessantes do espaço de busca, acabando sua execução com uma solução muito inferior à solução ótima. Diversificação é um mecanismo algorítmico que tenta resolver este problema forçando a busca a lugares inexplorados do espaço. Usualmente baseado em uma memória de longo termo da busca, registrando a frequência que vários componentes estiveram presentes nas soluções desde o começo da execução. A partir disso podemos implementar dois métodos. O primeiro, chamado diversificação por recomeço, envolve forçar o uso de componentes com pouca frequência de presença nas soluções e recomeçando a busca a partir deste ponto. O segundo método, diversificação contínua, integra diversificação direto no processo de busca (movimentos na vizinhança).

Isto é feito atribuindo-se pesos aos movimentos possíveis baseados nas frequências dos componentes da solução. A técnica de diversificação para se aprimorar a Busca Tabu é de suma importância devido às características da meta-heurística, podendo afetar amplamente as soluções encontradas se implementada de maneira correta.

2.1.8 Speed Up

Com o intuito de melhorar o desempenho da busca local implementada nos algoritmos implementados, foi também aplicado técnicas de *speed up* baseada em artigos sobre otimização de exploração de vizinhança [7] [3].

O *speed up* implementado nesse trabalho tira proveito dos tempos de completude das tarefas calculados previamente. Ao calcular-se o tempo de término da permutação ao fazer um movimento na vizinhança, por exemplo uma troca entre uma tarefa na posição i com uma tarefa na posição j, onde ij, aproveita-se o tempo de términos de todas as tarefas, já calculadas para esta mesma permutação num movimento na vizinhança anterior, nas posições ki.

Esse processo tem um *speed up* teórico de 50% [3], visto que o algoritmo percorre o vetor inteiro em busca de bons movimentos.

2.2 BRKGA

Biased Random-Key Genetic Algorithm (BRKGA) é uma meta-heurística inspirada no conceito de algoritmos genéticos (GA), que por sua vez também é uma meta-heurística. Em um GA cada cromossomo da população representa uma solução para o problema tratado. A busca por uma solução ótima segue através da passagem de gerações da população, nestas novos indivíduos são gerados a partir de procedimentos conhecidos como *crossover* (combinação de dois indivíduos da população) e mutação (alteração aleatória de indivíduos para gerar diversidade na população). No processo de *crossover* a qualidade da solução (indivíduo), chamado *fitness* pode ser levado em conta de maneira direta ou aleatória. Em [8] pode-se encontrar um estudo amplo sobre o BRKGA no contexto do *flowshop*.

2.2.1 Indivíduos

Agora para o BRKGA, cada indivíduo da população consiste em um vetor de números reais aleatórios, chamados de chaves (*keys*). A população inicial consiste de p cromossomos, onde cada chave desse cromossomo é gerada aleatoriamente no intervalo $[0, 1)$. Para a avaliação dos indivíduos, o BRKGA deve dispor de um método decodificador que recebe como entrada um cromossomo e deve retornar um valor de solução associado àquele indivíduo.

2.2.2 Passagem de gerações

Do mesmo modo que ocorre em GAs, a busca pela solução ótima decorre nas passagens de gerações da população inicial. Cada indivíduo de cada geração é avaliado para a determinação dos indivíduos elite, aqueles que apresentam as melhores soluções, e indivíduos não-elite.

A formação da nova população é dada da seguinte forma. Todos os indivíduos de elite da geração anterior são mantidos na nova geração. A proporção de indivíduos que serão considerados elite é dado por um parâmetro do BRKGA, p_e. Em seguida são inseridos mutantes na nova população. Mutantes são novos indivíduos gerados do mesmo modo que os indivíduos da população inicial, vetores aleatórios com chaves dento de $[0, 1)$. A porcentagem de cada nova população que consistirá de mutantes também é dado por um parâmetro do algoritmo, p_m.

O restante da população é constituído a partir do cruzamento de indivíduos da população anterior. Nesta parte da meta-heurística, BRKGAs e GAs se diferenciam, enquanto GAs, desprovidos do conceito de indivíduos de elite, selecionam os pais do novo indivíduo de maneira aleatória, um BRKGA faz o cruzamento de um indivíduo elite aleatório e um indivíduo não-elite aleatório. Desse modo as características dos indivíduos elite que resultam em boas soluções são preservadas, mas também não elitiza a população de um modo geral, mantendo a diversidade da população.

2.2.3 *Crossover*

No BRKGA o método de *crossover* utilizado chama-se *Crossover* Parametrizado Uniforme. Este método consiste em, dado uma probabilidade ρ, para cada chave em $N = 1, ..., n$, o novo indivíduo receberá em sua posição i a chave da mesma posição i de seu pai de elite com probabilidade ρ e a receberá a chave da posição i de seu pai não elite com probabilidade $1 - \rho$.

2.2.4 Decodificador

Como mencionado anteriormente, um BRKGA deve dispor de um método que seja capaz de entender um cromossomo da população como uma solução para o problema em questão. Desse modo o algoritmo pode avaliar cada indivíduo da população durante a execução.

O método de decodificação implementado neste trabalho, além de traduzir um cromossomo e avaliá-lo para o problema do *flowshop*, também tem a capacidade de aplicar melhorias em tais indivíduos, condicionalmente ou não, e também introduzir na população esse indivíduo modificado para que suas novas características estejam presentes na população e possam ser passadas em diante caso o indivíduo seja selecionado para reprodução.

Foram exploradas algumas variações do método decodificador.

- Padrão: apenas tradução e avaliação dos indivíduos da população, através da ordenação dos números reais do vetor e traduzindo sua posição no cromossomo ordenado para o número da tarefa. Como exemplo, o vetor $[0.23, 0.45, 0.65, 0.09, 0.98]$ seria avaliado como a permutação de tarefas $[2, 3, 4, 1, 5]$. O algoritmo padrão do BRKGA é disponibilizado pelos professores Rodrigo F. Toso and Mauricio G. C. Resende[2].

- NHE em todo avaliação: todo cromossomo avaliado, tem sua permutação modificada pela heurística NHE.

[2] http://mauricio.resende.info/src/brkgaAPI/

- Busca Local condicional: caso o *tardiness* total do cromossomo atinja um valor menor que (média - P(média - melhor solução)), onde a média refere-se aos valores de soluções médio dos cromossomos da população anterior e a melhor solução é o cromossomo com o melhor valor de solução também da população anterior. O parâmetro P, representa a parcela de soluções nas quais será aplicada a busca local. Durante os experimentos foram utilizados valores no intervalo $(0, 1)$, como será visto posteriormente. Para P igual a 1, somente se aplicará busca local naqueles indivíduos que apresentarem um *tardiness* menor que o da melhor solução da população anterior. Caso P seja igual a 0, todos indivíduos que representarem soluções melhores que a média dos indivíduos da população anterior serão submetidos à busca local.

- NHE em toda avaliação e Busca Local condicional: combinação das duas variações anteriores, com os mesmos princípios descritos.

Os métodos mencionados foram implementados e testados, os métodos que apresentaram os melhores resultados serão apresentados na seção de resultados e análise de dados.

3 Avaliação dos Resultados

Uma análise dos métodos propostos será feita a partir de um subconjunto das instâncias disponibilizadas por [5] no site Sistemas de Optimización Aplicada [3], com número de tarefas variando de 50 até 350 e o número de máquinas entre 10 e 50. Aplicaremos sempre os mesmos problemas e executaremos sempre no mesmo hardware para as variações implementadas com o objetivo de se fazer uma análise comparativa dos métodos.

Para o problema proposto as instâncias são os valores de processamento para as n tarefas nas m máquinas, juntamente com os prazos para cada tarefa. Um exemplo é apresentado Tabela 1.

Para cada variação da Busca Tabu e do BRKGA implementada será testado o subconjunto de instâncias. Os valores das soluções obtidos pelos algoritmos serão tabelados com o objetivo de se comparar as implementações entre si e com as melhores solução providas também por [5].

[3] http://soa.iti.es/files/Eva_Instances.zip

Máquina/Tarefa	1	2	3	4	5
1	1	9	70	49	15
2	4	48	85	15	15
3	87	8	72	88	51
4	21	85	31	29	3
Prazos	150	200	250	175	225

Tabela 1: Exemplo de instância do problema com os tempos de máquinas para cada tarefa e também seus prazos

Os parâmetros utilizados nas execuções foram: lista tabu de tamanho 50 e probabilidade de se explorar um vizinho de 20%, quando aplicada a Busca Tabu probabilística. Os parâmetros que variam dependendo da instância sendo testada podem ser vistos na tabela 2.

O tempo de execução de cada instância foi calculado a partir da fórmula $(n * m/2) * 90$, calculada em ms. Os limites de tempo de cada instância podem ser encontrados na tabela 3.

A partir dos resultados será possível ter uma avaliação da solução obtida pelo método em relação a melhor solução conhecida, analisar a qualidade da implementação, a eficiência das meta-heurísticas Busca Tabu e BRKGA para o problema proposto e como cada variação influencia nos resultados.

4 Resultados

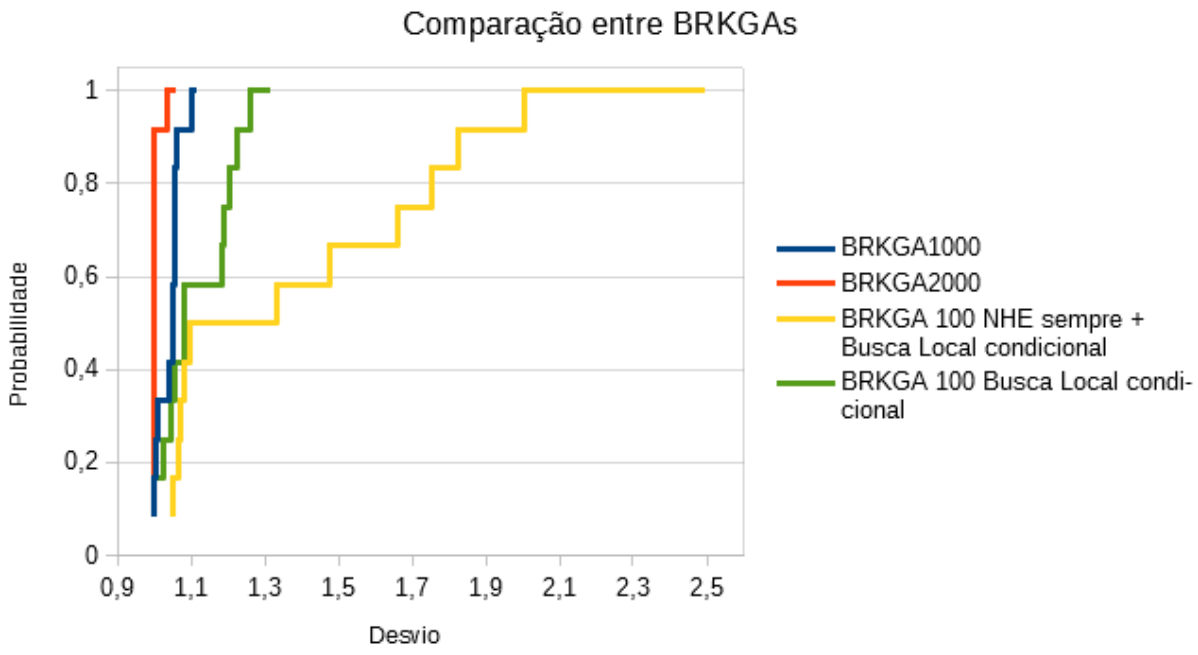

Figura 2: Performance Profile para variações do BRKGA.

Instância	Número de iterações sem melhora da solução para ativar Diversificação da Busca Tabu	Valor de P para BRKGA com NHE sempre + busca local condicional	Valor de P para BRKGA 100 com busca local condicional
I_50_10_1	1000	0.01	0.01
I_50_30_1	1000	0.01	0.01
I_50_50_1	1000	0.01	0.01
I_150_10_1	500	0.90	0.05
I_150_30_1	500	0.90	0.99
I_150_50_1	500	0.90	0.99
I_250_10_1	100	0.90	0.90
I_250_30_1	100	0.90	0.99
I_250_50_1	100	0.90	0.99
I_350_10_1	50	0.90	0.99
I_350_30_1	50	0.90	0.99
I_350_50_1	50	0.90	0.99

Tabela 2: Parâmetro utilizados para os experimentos

Instância	Tempo limite (s)
I_0,2_0,2_50_10_1	22
I_0,2_0,2_50_30_1	67
I_0,2_0,2_50_50_1	112
I_0,2_0,2_150_10_1	67
I_0,2_0,2_150_30_1	202
I_0,2_0,2_150_50_1	337
I_0,2_0,2_250_10_1	112
I_0,2_0,2_250_30_1	337
I_0,2_0,2_250_50_1	562
I_0,2_0,2_350_10_1	157
I_0,2_0,2_350_30_1	472
I_0,2_0,2_350_50_1	787

Tabela 3: Limites de tempo utilizados para limitar a execução dos experimentos.

Instance	TS	TS probabilística com diversificação
I_0,2_0,2_50_10_1	2044	2031
I_0,2_0,2_50_30_1	22444	20359
I_0,2_0,2_50_50_1	39447	36355
I_0,2_0,2_150_10_1	14495	14217
I_0,2_0,2_150_30_1	59769	56319
I_0,2_0,2_150_50_1	131330	125946
I_0,2_0,2_250_10_1	32526	27798
I_0,2_0,2_250_30_1	104436	108626
I_0,2_0,2_250_50_1	215352	207203
I_0,2_0,2_350_10_1	50794	50817
I_0,2_0,2_350_30_1	173336	167091
I_0,2_0,2_350_50_1	329143	314149

Tabela 4: Resultados para os dois algoritmos de Busca Tabu implementados.

Instância	BRKGA1000	BRKGA2000	BRKGA 100 NHE sempre + Busca Local condicional	BRKGA 100 Busca Local condicinal
I_0,2_0,2_50_10_1	2066	2142	2232	2240
I_0,2_0,2_50_30_1	21894	20821	22360	22499
I_0,2_0,2_50_50_1	36859	36724	38527	38387
I_0,2_0,2_150_10_1	13401	13250	14163	14019
I_0,2_0,2_150_30_1	57549	51483	90333	61085
I_0,2_0,2_150_50_1	117246	110945	147977	132095
I_0,2_0,2_250_10_1	30395	27547	30207	28286
I_0,2_0,2_250_30_1	97832	93226	170274	117492
I_0,2_0,2_250_50_1	206169	197899	292622	238840
I_0,2_0,2_350_10_1	50867	50973	119982	48059
I_0,2_0,2_350_30_1	154067	145984	292940	178892
I_0,2_0,2_350_50_1	282044	265819	442154	349375

Tabela 5: Resultados para as variações dos BRKGAs implementados neste trabalho. BRKGA 1000, BRKGA 2000 e BRKGA 100, representam as variações que têm tamanho de população 1000, 2000 e 100, respectivamente.

Instância	Algoritmo	Melhor solução encontrada	Melhor solução de referência	Desvio
I_0,2_0,2_50_10_1	Tabu Search Melhorado	2031	1876	1,083
I_0,2_0,2_50_30_1	Tabu Search Melhorado	20359	19475	1,045
I_0,2_0,2_50_50_1	Tabu Search Melhorado	36355	34368	1,058
I_0,2_0,2_150_10_1	BRKGA2000	13250	11797	1,123
I_0,2_0,2_150_30_1	BRKGA2000	51483	47978	1,073
I_0,2_0,2_150_50_1	BRKGA2000	110945	105825	1,048
I_0,2_0,2_250_10_1	BRKGA2000	27547	25342	1,087
I_0,2_0,2_250_30_1	BRKGA2000	93226	86952	1,072
I_0,2_0,2_250_50_1	BRKGA2000	197899	185323	1,068
I_0,2_0,2_350_10_1	BRKGA 100 Busca Local condicinal	48059	46659	1,030
I_0,2_0,2_350_30_1	BRKGA2000	145984	144749	1,009
I_0,2_0,2_350_50_1	BRKGA2000	265819	262910	1,011

Tabela 6: Melhores resultados obtidos neste trabalho em comparação com os melhores resultados encontrados por [5] para o conjunto de instâncias selecionado.

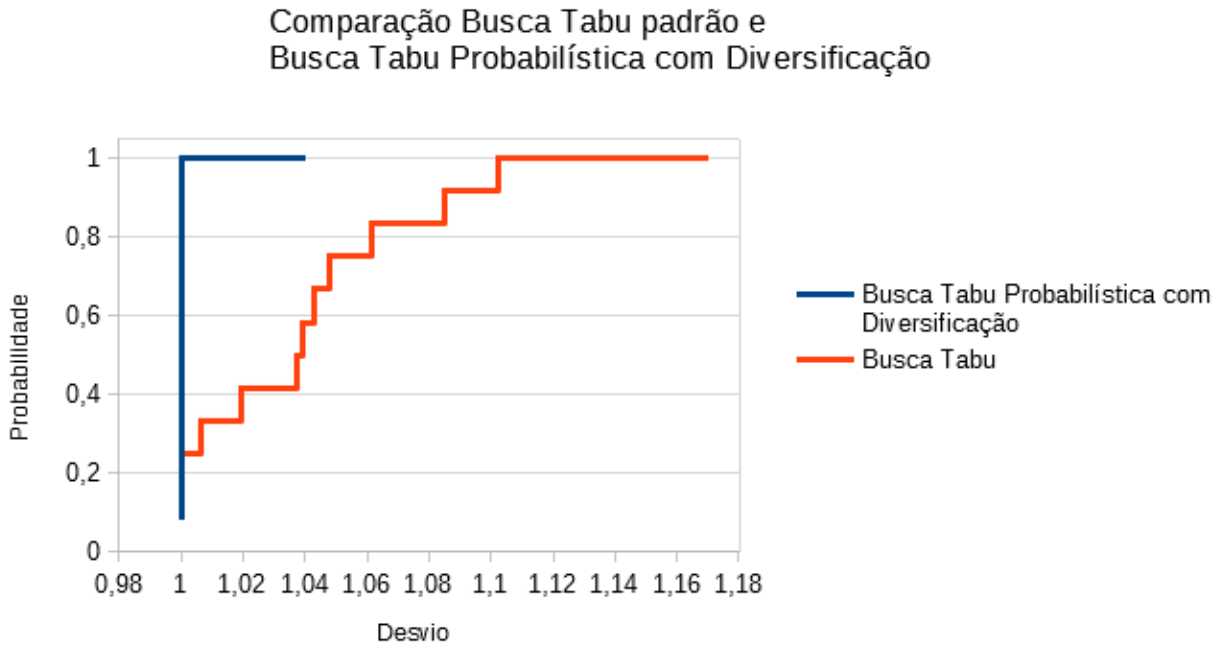

Figura 3: Performance Profile para variações da Busca Tabu.

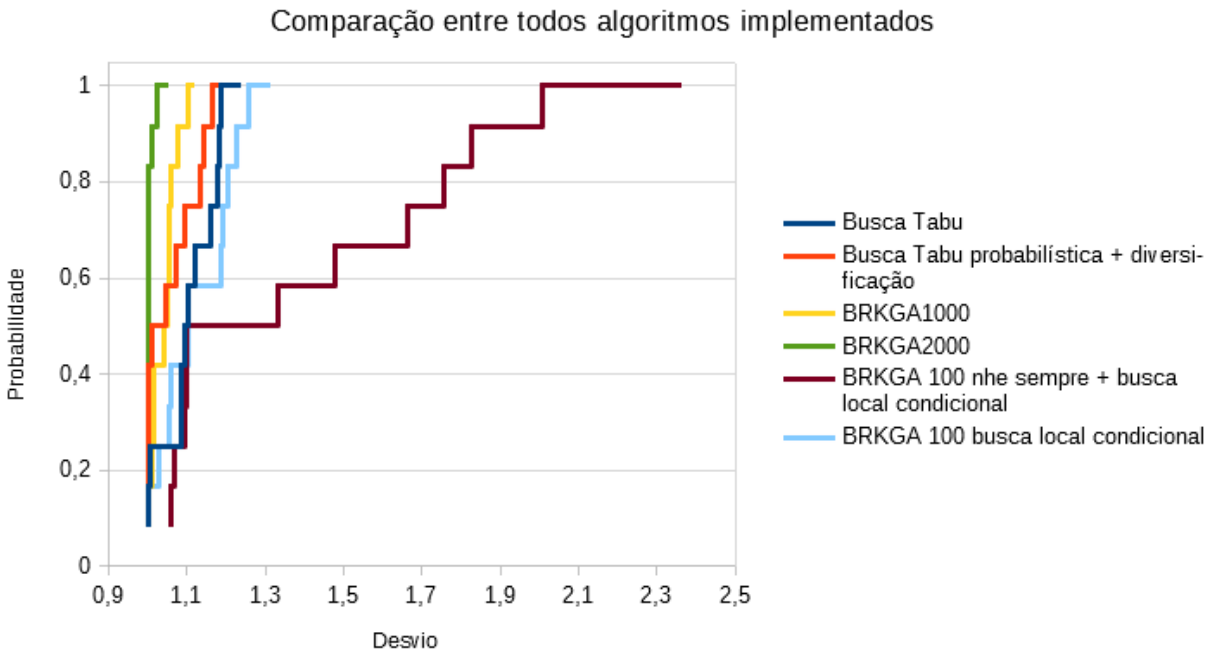

Figura 4: Performance Profile para comparação dos algoritmos e suas variações.

5 Análise de dados

Inicialmente podemos destacar que os algoritmos selecionados para esse trabalho obtiveram sucesso ao resolver o problema do *flowshop* abordado, visto que os resultados encontrados tiveram um desvio percentual, de no máximo, 12,3% em relação aos encontrados na referência [5]. Podemos apontar bons resultados com desvios de apenas 0,9% e 1,1% pelo BRKGA padrão, 5,8%, 4,5% pela Busca Tabu Melhorada e 3,0% pela variação do BRKGA. Vale-se notar que os experimentos feitos nesse trabalho foram limitados baseado em uma fórmula que em alguns casos resulta em um tempo pequeno de execução (menos de um minuto em vários casos). Em tais casos, como para as três menores instâncias, a variação de Busca Tabu implementada consegue superar os outros algoritmos, mostrando que o algoritmo apresenta uma rápida conversão para bons resultados.

De modo mais amplo, a implementação que gerou melhores soluções foi a implementação padrão do BRKGA com 2000 indivíduos na população. Como pode-se notar da figura 2 não foi muito vantajoso gastar um tempo computacional extra para se aplicar outros métodos durante a decodificação do cromossomo, visto que as duas variações comparadas não apresentaram um desempenho melhor como esperado. Nota-se também que no contexto deste problema, executar o BRKGA com uma população maior gera melhores resultados, visto que aumenta a abrangência do espaço de soluções explorado e também é muito pouco custo decodificar os cromossomos apenas ordenando suas chaves aleatórias. Mesmo com a

redução do tamanho da população em uma ordem de 10, aplicar busca local e a heurística NHE mostrou-se muito custoso, prejudicando o tempo de processamento de cada iteração do algoritmo, resultando em menos gerações sendo passadas e consequentemente uma menor melhora da solução com a passagem do tempo.

Para a variação da Busca Tabu implementada fica claro as vantagens dos métodos implementados. A técnica de diversificação da exploração da vizinhança juntamente com a Busca Tabu probabilística que proporciona maior aleatoriedade ao algoritmo, apresentaram melhora na qualidade das soluções encontradas no mesmo tempo de execução. Na realização dos experimentos das implementações que não utilizavam o método de diversificação, percebia-se uma estagnação no valor de solução em pouco tempo de execução. Isso indica que o algoritmo permanecia procurando soluções em uma parte do espaço de busca pouco promissora. Pensa-se que as melhorias significativas proporcionadas pelo método de diversificação sejam provenientes da mitigação deste problema, incentivando a busca de vizinhança a buscar soluções em regiões pouco exploradas do espaço.

Apesar de que o algoritmo que teve a melhor performance em geral ter sido o BRKGA padrão, os algoritmos implementados neste trabalho mostraram resultados promissores para o problema em questão. Analisando-se a figura 4 e a tabela 6, percebemos que o BRKGA com busca local condicional e a Busca Tabu melhorada apresentaram a melhor performance entre todos os algoritmos. Isso indica uma complementariedade de desempenho das metodologias. Tendo isso em mente poderia-se pensar no desenvolvimento de uma nova metodologia que incorporasse as características de melhor impacto de cada metodologia que as fazem obter bom desempenho em dada instância.

Para tentar aprimorar os resultados obtidos poderiam ser testados outros tamanhos de populações, visto que esse parâmetro influencia fortemente nos resultados, explorar mais métodos de intensificação durante a decodificação, métodos que não sejam custosos computacionalmente mas aprimorem de alguma maneira o indivíduo sendo avaliado, e também aumentar o tempo de execução das instâncias que tiveram limites baixos, para possibilitar talvez uma convergência tardia das soluções.

6 Conclusão

Os algoritmos propostos foram implementados com sucesso encontrando bons resultados para instâncias grandes do problemas do *flowshop*. Para métodos exatos como *branch and bound* levaria-se um tempo computacional muito grande na busca de soluções, visto que as possibilidades de soluções são $n!$ para n tarefas. E também quanto maior o número de máquinas, maior o tempo gasto calculando-se os limites primais e duais.

A partir dos resultados foi possível perceber que o BRKGA se beneficia amplamente de uma rápida e eficiente função de decodificação. Ao contrário de se gastar tempo de execução tentando melhorar cada indivíduo, é preferível uma população grande e um elevado número de gerações durante a execução, permitindo cromossomos com valores de solução cada vez melhores.

Outra conclusão importante é a necessidade da implementação de métodos, além da Busca Tabu padrão, para o aprimoramento do algoritmo, destacando-se métodos que aju-

dam na busca da solução ótima como diversificação. É de suma importância a implementação de métodos deste tipo e a calibração dos parâmetros para a utilização dos mesmos, para que o algoritmo seja guiado de forma mais eficiente na exploração do espaço de busca.

Conclui-se também que para o problema em questão, diversidade em geral gera melhores resultados. Percebe-se isso pela diversidade resultante de um número alto de indivíduos na população do BRKGA e também pela aplicação da técnica de diversificação na Busca Tabu. Tendo isso em vista recomenda-se a experimentação com outros esquemas de diversificação para a Busca Tabu visando explorar várias áreas diferentes do espaço de busca. Como exemplo temos as técnicas *Scatter Search* e *Path Relinking* propostas em [9].

Referências

[1] G. K. Rand, "Machine scheduling problems: Classification, complexity and computations: A.h.g. rinnooy kan, martinus nijhoff, the hague, 1976, ix + 180 pages," *European Journal of Operational Research*, vol. 1, no. 3, p. 206, 1977.

[2] J. Du and J. Y.-T. Leung, "Minimizing total tardiness on one machine is np-hard," *Mathematics of Operations Research*, vol. 15, no. 3, pp. 483–495, 1990.

[3] K. Karabulut, "A hybrid iterated greedy algorithm for total tardiness minimization in permutation flowshops," *Computers Industrial Engineering*, vol. 98, pp. 300 – 307, 2016.

[4] F. Pagnozzi and T. Stützle, "Automatic design of hybrid stochastic local search algorithms for permutation flowshop problems," *European Journal of Operational Research*, vol. 276, no. 2, pp. 409 – 421, 2019.

[5] G. M. Eva Vallada, Rubén Ruiz, "Minimising total tardiness in the m-machine flowshop problem: A review and evaluation of heuristics and metaheuristics," *Computers & Operations Research*, 2008.

[6] M. Gendreau, "Chapter 2 an introduction to tabu search," 04 2019.

[7] F. Pagnozzi and T. Stützle, "Speeding up local search for the insert neighborhood in the weighted tardiness permutation flowshop problem," *Optimization Letters*, vol. 11, pp. 1283–1292, Oct 2017.

[8] G. B. Mainieri, "Meta-heurística brkga aplicada a um problema de programação de taregas no ambiente flowshop híbrido," 2014.

[9] F. Glover, "A template for scatter search and path relinking," pp. 1–51, 01 1997.