A Hybrid Genetic Algorithm for solving the Unrelated Parallel Machine Scheduling problem with Sequence Dependent Setup Times

L.R. Abreu and B.A. Prata

***Abstract*— The parallel machine scheduling problem is a well known combinatorial optimization problem with several applications in computer science, telecommunications and operations management. Several studies have shown that exact approaches for this problem are not useful in practical situations due to the high computational costs involved. This paper describes a hybrid genetic algorithm for solving the unrelated parallel ma- chine scheduling problem with sequence dependent setup times. Computational results with simulated data are presented and discussed. A case study on the granite industry is presented.The proposed approach outperformed three traditional dispatch rules presented in the current literature.**

***Keywords*— Evolutionary algorithms, hybrid meta-heuristics, production scheduling.**

I. INTRODUÇÃO

O sequenciamento da produção, parte substancial de qualquer planejamento industrial, tem passado por diversas alterações ao longo dos últimos anos.

O problema clássico de máquinas paralelas foi fundamentado inicialmente por Root [1], o qual discorria sobre os prazos de términos das tarefas em um ambiente com múltiplas máquinas trabalhando em paralelo. Esse problema foi abordado inúmeras vezes na literatura. Os primeiros artigos propunham, como método de resolução, regras de despacho e teoria dos grafos [2] e [3].

Tradicionalmente, o tempo de preparação (*setup*) era considerado como uma parcela dos tempos de processamento. Tal premissa é uma forte simplificação do problema real que pode conduzir a problemas no processo de sequenciamento. Por exemplo, ao se considerar os tempos de processamento e de preparação conjuntamente, perdemos a flexibilidade de preparar uma máquina previamente e executar antes uma dada tarefa. Portanto, é de suma importância a consideração explícita dos tempos de preparação [4]. Alguns trabalhos têm abordado o problema de máquinas paralelas considerando explicitamente os tempos de *setup* [5] e [6].

Os métodos de otimização com a utilização de heurísticas para a resolução desse problema foram apresentados inicialmente por [7] que utilizou a meta-heurística *Simulated Annealing* (SA), [8] com a utilização da meta heurística *Variable Neighbourhood Search* (VNS), [9] que utilizou a Busca Tabu em concomitância com regras de despacho e [10, 11, 12] e [13] que implementaram algoritmos genéticos (AGs).

Com o avanço das pesquisas as meta-heurísticas híbridas se mostraram mais eficientes para a resolução do problema. Os principais autores a retratá-las são: [14] com uma análise múltipla sobre várias meta-heurísticas, [15] que utilizam busca locais em soluções iniciais gulosas, [16] que utilizam estratégias evolucionárias híbridas, [17] e [18] que propõem uma resolução por busca local estocástica.

Na literatura pouco foi reportado sobre a resolução desse problema por algoritmos evolucionários com a utilização mútua de outras meta-heurísticas, como o SA e buscas locais. Também existe a carência da comparação desses métodos heurísticos com as clássicas regras de despacho, utilizadas largamente nas indústrias. É importante a avaliação de desempenho das regras de despacho, pois elas são técnicas utilizadas no cotidiano das indústrias. Além disso, as implementações de métodos meta-heurísticos nos ambientes industriais complexos, através de estudos de casos, são essenciais para a validação dos mesmos.

Este trabalho tem como objetivo reportar a elaboração de um algoritmo genético, com características de outras meta-heurísticas, para a resolução do problema de sequenciamento da produção no ambiente de máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência. O algoritmo genético híbrido proposto foi testado em instâncias geradas aleatoriamente e na resolução de um problema real, numa indústria de granito.

O trabalho é composto por mais seis seções. Na próxima é apresentada a definição do problema. Na terceira seção consiste na descrição dos algoritmos propostos. Na quarta seção estão descritos os resultados computacionais em instâncias aleatórias. Na quinta seção discorre-se sobre um estudo de caso. Na sexta seção são apresentadas as conclusões e propostas para estudos futuros.

II. DEFINIÇÃO DO PROBLEMA

O problema de máquinas paralelas não relacionadas com tempos de setup dependentes da sequência, em inglês *Unrelated Parallel Machine Scheduling problem with Sequence Dependent Setup Times* (UPMSPST), $Rm|S_{ij}|C_{max}$ na notação da literatura, diferente de outros problemas de sequenciamento como o *flow shop* e job *shop*, aborda apenas um estágio de produção, ou seja, cada trabalho precisa ser processado uma única vez e cada máquina processa apenas um trabalho por momento. As máquinas não relacionadas possuem os tempos de processamento distintos, não possuindo nenhum padrão entre eles (Rm) [19].

Entre dois trabalhos que são processados em sequência, existe um tempo de preparação que depende tanto da sequência de trabalhos quando da máquina que eles estão alocados. Porém, nesse artigo o tempo de preparação depende apenas da sequência de trabalhos alocados na máquina (S_{ij}).

L. R. Abreu, Universidade Federal do Ceará, Fortaleza, Ceará, Brasil, leviribeiro@alu.ufc.br

B. A. Prata, Universidade Federal do Ceará, Fortaleza, Ceará, Brasil, baprata@ufc.br

Corresponding author: Levi Ribeiro de Abreu

O objetivo do problema é encontrar uma sequência de processamento que reduza o tempo de término do último trabalho alocado na máquina que concluir esse trabalho, chamada de máquina gargalo (C_{max}).

Uma instância para o problema de máquinas paralelas não-relacionadas com tempos de *setup* dependentes da sequência é caraterizada por: um número de máquinas (m), um número de trabalhos (n), uma matriz de tempos de processamento (com m linhas e n colunas) e uma matriz de tempos de *setups* (com n linhas e n colunas). A seguir, nas Tabelas I e II, é ilustrada uma instância para o problema.

TABELA I
TEMPO DE PROCESSAMENTO DAS MÁQUINAS

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
Máquina 1	7	19	14	11	8	12	5	18	8	12
Máquina 2	9	9	20	5	8	20	18	16	14	16

TABELA II
TEMPO DE SETUP ENTRE AS TAREFAS

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
J_1	-	9	15	7	5	1	9	1	9	1
J_2	9	-	9	11	1	0	13	1	3	4
J_3	15	9	-	2	0	5	15	13	12	2
J_4	7	11	2	-	5	9	8	7	14	13
J_5	5	1	0	5	-	2	15	11	4	9
J_6	1	0	5	9	2	-	14	9	0	1
J_7	9	13	15	8	15	14	-	0	15	13
J_8	1	1	13	7	11	9	0	-	8	6
J_9	9	3	12	14	4	0	15	8	-	10
J_{10}	1	4	2	13	9	1	13	6	10	-

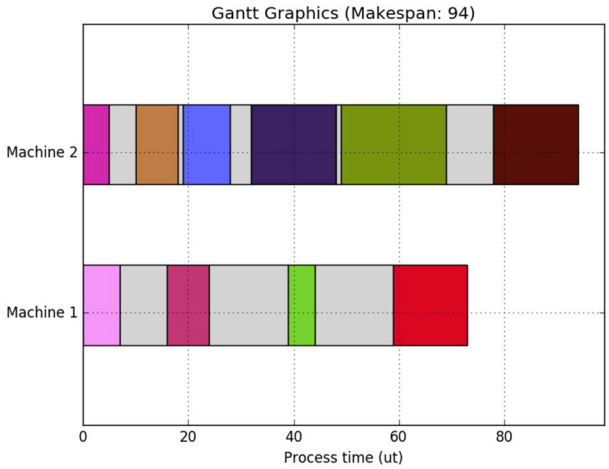


Figura 1. Exemplo gráfico de Gantt.

Na Fig. 1, é ilustrado o diagrama de Gantt, temos um exemplo de alocação de 10 trabalhos em 2 máquinas paralelas não relacionadas, com uma sequência S = {1,4,5,9,7,2,10,6,3,8}, com uma alocação: {M1: [1, 9, 7, 3] M2: [4, 5, 2, 10, 6, 8]}. Em cores são os tempos de processamento e em cinza os tempos de preparação. Nesse caso é perceptível a máquina gargalo do problema, no caso a máquina 2, no qual o último trabalho terminou de ser processado totalizando um *Makespan* de 94 unidades de tempo. A alocação é dinâmica, ou seja, o próximo trabalho a ser processado sempre é alocado na máquina com o menor tempo acumulado (tempo da máquina + *setup* + processamento).

III. ALGORITMOS PROPOSTOS

A metodologia proposta no presente artigo para a resolução do problema é a utilização de modelos holísticos como regras de prioridades e uma meta-heurística híbrida: um algoritmo genético com operadores de *Simulated Annealing* e busca local.

As regras de prioridades em ambientes de sequenciamento da produção são alternativas simples e de fácil implementação e, com um esforço computacional reduzido, garantem resultados expressivos [20].

As meta-heurísticas consistem em soluções gerais para os mais variados problemas de otimização. Baseiam-se em métodos determinísticos e na aleatoriedade, conseguindo encontrar soluções ótimas ou próximas destas [21].

Em resumo, os métodos utilizados são: SAPT (*Shortest average processing time*), LAPT (*Longest Average Processing Time*) e RAND (*Random sequence*), como regra de prioridades e o Algoritmo Genético com SA e busca local, como meta-heurística.

Regras de prioridades

(i) *Shortest Average Processing Time*

A (SAPT) é uma regra de prioridade que busca encontrar o tempo médio de processamento dos trabalhos, que consiste na soma do tempo de preparação e o tempo de processamento médios dos trabalhos, devendo eles serem processados de forma crescente. A regra garante que os trabalhos com um tempo total de processamento médio reduzido serão processados primeiro em relação aos demais trabalhos.

(ii) Longest Average Processing Time

A (LAPT) é bastante semelhante a regra de prioridade anteriormente citada, diferindo desta pela ordenação dos tempos médios de processamento em ordem decrescente. Essa regra faz com que os trabalhos com os tempos de processamento médios maiores serão processados primeiro em relação aos outros trabalhos.

(iii) *Random Sequence*

(RAND) é uma sequência gerada aleatoriamente apenas para fim de comparação.

Algoritmo genético híbrido

O Algoritmo Genético (AG) foi proposto inicialmente por Holland [22]. A meta-heurística consiste em simular o processo de evolução das espécies em um processo iterativo de geração de soluções para a resolução de problemas de otimização.

Para alcançar boas soluções, um AG parte de um conjunto inicial de indivíduos (população), no qual são representadas soluções para o problema de otimização. Em seguida, um operador de seleção é utilizado para selecionar um conjunto de pais para gerar, por meio de um operador de recombinação (cruzamento), novas soluções para o problema. Esse processo é repetido por um determinado número de iterações (gerações) até que a melhor solução obtida é retornada [23].

O AG proposto neste trabalho consiste numa hibridização com características da meta-heurística *Simulated Annealing* que realiza busca locais nas soluções geradas. A seguir, no

Algoritmo 1, é apresentado o pseudo-código do AG híbrido proposto.

Algoritmo 1 Algoritmo Genético com *Simulated Annealing* e busca local

Entrada: Parâmetros exigidos
Saída: Sequência de Processamento dos Trabalhos

```
1: Gera_População_Inicial()
2: Busca_Local_População()      ▷ 2-opt best improvemment
3: Calcula_Aptidão()
4: k ← 0
5: Enquanto k <= Número Máximo de Gerações Faça
6:     pai1,pai2 ← Seleção_pop()
7:     filho ← Cruzamento(pai1,pai2)      ▷ Cruzamento CX
8:     Reposição_SA()          ▷ Inserindo filho na População
9:     Critério_Restart()
10:    Busca_Local_População() ▷ 2-opt first improvemment
11:    temp_atual = α×temp_atual          ▷ Reduzindo a Temperatura
12:    k ← k + 1
13: Fim Enquanto
14: melhor_filho ← melhor filho gerado pelo A.G.
15: Simulated_Annealing(melhor_filho)     ▷ Realizando o Simulated Annealing final
```

A população inicial é gerada aleatoriamente, sendo em seguida, refinada por uma busca local *2-opt best improvement*. Na busca local *best improvment*, toda a vizinhança é explorada e a melhor solução obtida é retornada. Dessa forma, a população inicial já é formada por ótimos locais de boa qualidade, possibilitando o AG convergir mais rapidamente para regiões promissoras do espaço de busca.

Os pais são selecionados por torneio binário: são selecionados aleatoriamente quatro pais, os quais são comparados par a par. Os dois melhores pais serão escolhidos para a aplicação do operador de cruzamento.

Para a realização do cruzamento foram realizados testes com três tipos de operadores:

(i) *Order Crossover (OX)*

O operador seleciona dois pontos de cortes aleatórios do pai 1 e os adiciona no filho, os elementos restantes são adicionados na ordem que aparecem no pai 2.

(ii) Partially Matched Crossover (PMX)

O operador seleciona dois pontos de corte aleatórios do pai 1 e o filho herda integralmente a sequência parcial gerada, por fim cada elemento faltante é preenchido do pai 2, contudo quando surgir uma sequência inviável é realizado algumas trocas para correção.

(iii) Cycle Crossover} (CX)

Consiste em realizar o cruzamento com base no ciclo de trabalho entre os pais, gerando filhos que preservam a posição absoluta desses. Com o ciclo gerado, o filho recebe esses elementos do pai 1 e o restante do pai 2.

O operador de cruzamento escolhido, após vários testes computacionais preliminares, foi o CX que obteve resultados melhores que o PMX e o OX. Na Tabela III são apresentados os testes de calibração do algoritmo com os operadores de cruzamento tanto no pior caso quanto no melhor para um AG executado dez vezes. Na Fig. 2, é ilustrado um exemplo da aplicação dos cruzamentos propostos.

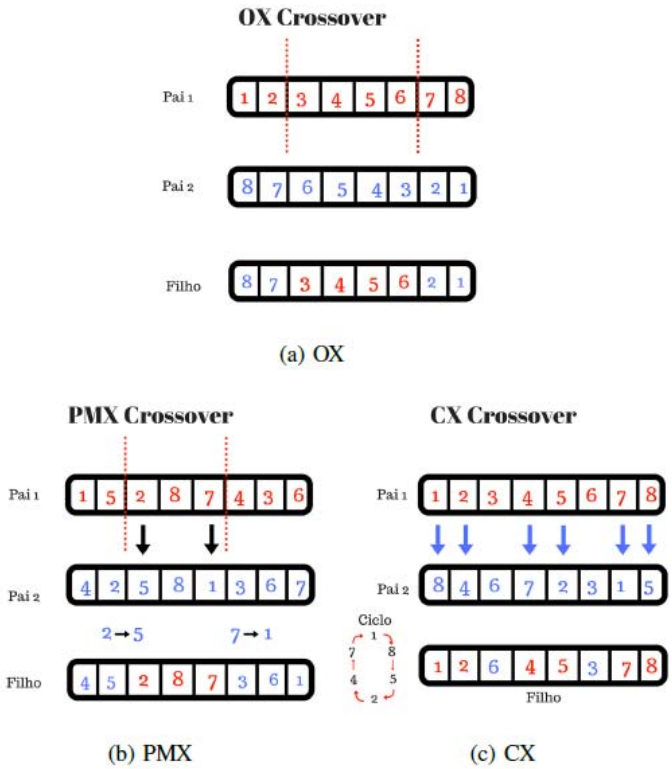


Figura 2. Operadores de Cruzamento Propostos.

TABELA III
MAKESPAN MÉDIO PARA CADA CONJUNTO DE INSTÂNCIA UTILIZANDO OS OPERADORES DE CRUZAMENTO PROPOSTOS

Instâncias	Worse OX/PMX/CX	Best OX/PMX/CX
MP - 10	79.5/72.7/**70.4**	**66.7**/**66.7**/**66.7**
MP - 25	84.4/85.6/**83.5**	**74.8**/74.5/75.8
MP - 50	**247.8**/249.8/**247.8**	237.8/238.2/**235.7**
MP - 100	343.6/344.8/**341.8**	331.1/333.8/**330.1**

Na mutação do filho ocorre um movimento do tipo *swap*. É gerado um número aleatório entre 0 e 1 e se esse número for menor ou igual a uma certa probabilidade, definida nos parâmetros do AG, ocorre uma troca na ordem de processamento de dois trabalhos.

No AG padrão, usualmente o filho gerado é inserido na população, substituindo-se o pior elemento da mesma, ou seja, no caso de um problema de minimização, o elemento da população com maior valor da função objetivo. Tal estratégia pode conduzir a busca para ótimos locais [24], outra estratégia seria substituir o filho apenas se ele for melhor que os pais, como na meta-heurística *Hill Climbing* (HC), pois assim a diversidade é mantida na população, no qual os filhos substituem seres semelhantes a eles [25].

No algoritmo proposto, o operador de reposição (atualização da população) possui um critério de aceitação baseado na meta-heurística *Simulated Annealing*. Se o filho for melhor que os pais ele é adicionado na população, mas se ele não for melhor existe uma certa probabilidade de aceitação que depende de uma temperatura que decai à medida que as gerações ocorrem.

É importante salientar que a aceitação do filho, como no *Simulated Annealing*, segue uma distribuição de *Boltzman*:

$$Critério = e^{\frac{-\Delta}{Temp\_Atual}} \quad (1)$$

Em que Δ significa a diferença do *makespan* entre o filho gerado e o melhor pai:

$$\Delta = makespan(filho) - makespan(pai) \quad (2)$$

É importante enfatizar, analisando a variável critério, que as chances de serem aceitos filhos ruins são altas nas gerações iniciais para uma maior variabilidade genética, já nas gerações finais essa chance diminui fazendo que apenas filhos bons sejam colocados na população e a busca seja conduzida para regiões promissoras. Com isso, é garantido um mecanismo de substituição com aleatoriedade, mas possuindo também elitismo, para que o AG não fique preso em ótimos locais [24].

Na Tabela IV é apresentado o resultado da calibração do AG comparando os tipos de reposição mencionados. Na tabela é ilustrado o *makespan* médio em cada conjunto de instâncias, tanto no pior caso quanto no melhor para um AG testado dez vezes, os parâmetros restantes do algoritmo estão listados na Tabela V e o operador de cruzamento utilizado foi o CX. Pode-se observar que a reposição com SA se mostrou superior aos demais operadores de reposição, por isso foi utilizada no teste final das instâncias e no estudo de caso.

TABELA IV
TESTES COM OS OPERADORES DE REPOSIÇÃO

Instancias	Worse AG-Padrão/HC/SA	Best AG-Padrão/HC/SA
MP - 10	71/**67**/72	**66.7**/**66.7**/**66.7**
MP - 25	85.7/**82.6**/86.2	76.2/76.1/**75.4**
MP - 50	249.8/249.5/**249.3**	237.7/242.2/**236.2**
MP - 100	343.9/345.2/**342.1**	331.8/337.7/**331.5**

Existe também um operador de *restart* na solução, no qual busca-se restringir a convergência prematura da população, garantindo a diversidade no processo de busca [26]. O parâmetro utilizado verifica se após um certo número de gerações não ocorrer nenhuma melhoria, a população e a temperatura são recomeçadas, guardando, ainda, o melhor filho. Assim, como a nova população terá esse filho com boa aptidão, ele vai distribuir material genético com os demais elementos da população, atingindo, por conseguinte, melhores resultados.

Por fim, é realizada em certas gerações, pelo menos três a cinco vezes ao longo do AG sempre proporcional ao número de gerações, uma busca local *2-opt first improvement* na população a fim de se manter uma melhoria contínua no processo. Na busca local *first improvement*, a vizinhança é explorada até que se obtenha uma solução melhor do que a solução que está sendo investigada. Caso se obtenha melhoria, a busca local é encerrada. Além disso, após a execução do AG é realizado um processo de SA clássico no melhor filho gerado pelo algoritmo, processo esse que utiliza a perturbação *insertion*, na qual promove, em cada interação, realocações aleatórias na ordem de produção de um trabalho no melhor filho gerado pelo AG. Os operadores genéticos do AG híbrido desenvolvido são apresentados nos Algoritmos 2, 3 e 4.

TABELA V
TABELA COM OS PARÂMETROS UTILIZADOS

Parâmetros do algoritmos genético	
Tamanho da população:	100
Quantidade de gerações	15000
Probabilidade de mutação	0.5
Temperatura inicial	1000000.00
Taxa de decrescimento (α)	0.9995

Algoritmo 2 Mutação

Entrada: filho e probabilidade de mutação (pmut).
Saída: filho modificado ou não.

```
Se aleatorio[0 : 1] <= pmut Então
    pos1,pos2 ← posições no filho aleatórias e distintas.
    filho[pos1],filho[pos2] = filho[pos2],filho[pos1]
Fim Se
```

Algoritmo 3 Reposição com *Simulated_Annealing*.

Entrada: População, filho e temperatura atual
Saída: A população com o filho inserido

```
melhor_pai ← melhor pai do cruzamento
Δ ← makespan(filho)-makespan(melhor_pai)
Se Δ <= 0 ou e^(−Δ/tem_atual) >= aleatorio[0 : 1] Então
    Insira o filho na posição do pior elemento da população
    e calcule sua aptidão.
Fim Se
```

Algoritmo 4 Restart.

Entrada: Parâmetros Propostos
Saída: Uma nova população ainda contendo o melhor filho até então inserido

```
Se Ocorrerem k gerações sem melhoria Então
    melhor_filho ← melhor elemento da população
    Gera_População_Inicial()
    Busca_Local_População() ▷ 2-opt first improvemment
    Adiciona-se o melhor_filho na população
    temp_atual ← 1000000.0          ▷ A temperatura é
recomeçada
Fim Se
```

IV. RESULTADOS COMPUTACIONAIS

As configurações do computador onde foram realizados os testes são as seguintes: Processador: Intel Celeron (U) 2957@1.40Ghz, Memória: 4.00 Gb e todas as heurísticas foram implementadas na linguagem de programação Python 3.5.2 (IDE Spyder).

Foram geradas, aleatoriamente, instâncias para a calibração e teste do algoritmo. Elas derivam de conjuntos de máquinas M = {2,4,8,12} e de trabalhos N = {10,25,50,100}. Combinando par a par cada item dos conjuntos, existem 4 tipos de problema, para os quais foram geradas 10 instâncias diferentes totalizando 40 instâncias para o teste. As instâncias estão disponíveis para download no sítio: https://www.researchgate.net/publication/315771587_Instances_Tested.

Em resumo, para cada conjunto existem 10 instâncias a serem testadas: 2 máquinas e 10 trabalhos, 4 máquinas e 25 trabalhos, 8 máquinas e 50 trabalhos e 12 máquinas e 100 trabalhos.

Os tempos de processamento e de preparação dos trabalhos foram gerados aleatoriamente através de uma distribuição uniforme U[5:50] para os tempos de processamento e U[0:25] para os *setups*. Foram realizados testes das regras de despacho e da meta heurística proposta. A regra RAND e o AG híbrido foram testados 10 vezes para cada instância, afim de uma melhor verificação e comparação. Os tempos computacionais das regras de prioridade não foram computados por que são desprezíveis. As buscas locais e o mecanismo de *restart* foram

utilizados em todas as execuções do AG. Foi calculado o desvio percentual médio, das regras de prioridade em relação ao A.G, para uma verificação da distância percentual do *makespan* das soluções obtidas por tais regras.

Na Tabela VI são apresentados os resultados do *makespan* obtido pelas regras de despacho para cada instância. Na Tabela VII são apresentados os resultados do AG, juntamente com o desvio percentual de cada regra.

TABELA VI
RESULTADOS OBTIDOS

Instâncias	SAPT	LAPT	Best	RAND Average	Worse
MP-10x01	107	96	74	86	98
MP-10x02	100	99	88	102	135
MP-10x03	111	109	90	100	112
MP-10x04	122	91	86	97	106
MP-10x05	117	109	84	104	124
MP-10x06	123	114	104	121	150
MP-10x07	83	97	95	111	127
MP-10x08	107	95	90	102	124
MP-10x09	100	83	74	90	108
MP-10x10	123	109	81	98	105
MP-25x01	110	93	109	103	109
MP-25x02	104	111	103	117	133
MP-25x03	119	112	98	114	128
MP-25x04	95	120	100	109	127
MP-25x05	114	141	110	117	124
MP-25x06	122	112	112	127	141
MP-25x07	115	112	99	116	134
MP-25x08	108	114	106	112	122
MP-25x09	115	116	112	124	136
MP-25x10	104	128	105	118	132
MP-50x01	191	164	182	179	189
MP-50x02	300	274	279	294	311
MP-50x03	309	297	286	295	314
MP-50x04	295	276	282	293	309
MP-50x05	301	282	281	289	298
MP-50x06	278	299	276	286	300
MP-50x07	291	292	279	294	308
MP-50x08	291	306	283	294	301
MP-50x09	298	301	278	292	299
MP-50x10	315	296	277	289	305
MP-100x01	379	365	339	370	383
MP-100x02	382	378	370	377	389
MP-100x03	369	366	370	378	390
MP-100x04	388	367	363	370	378
MP-100x05	367	372	356	370	380
MP-100x06	391	373	367	374	383
MP-100x07	368	366	358	372	391
MP-100x08	381	377	368	373	381
MP-100x09	387	376	367	379	386
MP-100x10	357	366	356	370	381

Na Fig. 3 é apresentado o desvio percentual médio de todas as regras de prioridades testadas em relação ao AG em cada conjunto de instâncias propostas. Na Fig. 4 é ilustrado o *makespan* médio em cada conjunto de instância.

Com base nos resultados obtidos, o AG conseguiu sobrepujar todas as regras de prioridades retratadas, apresentando tempos computacionais admissíveis, mesmo para as instâncias de grande porte como 50 e 100 trabalhos.

TABELA VII
ALGORITMO GENÉTICO

AG Best	AG Avg.	AG Worse	Time (s)	DP (%) SAPT	DP (%) LAPT	DP (%) RAND
62	89	116.0	3.53	73	54.8	**19.4**
72	106	141.0	3.56	39	37.5	**22.2**
61	105	150.0	3.54	82	78.7	**47.5**
61	100	140.0	3.54	100	49.2	**41.0**
68	106	146.0	3.51	72	60.3	**23.5**
80	120	161.0	3.56	54	42.5	**30.0**
73	113	153.0	3.53	**14**	32.9	30.1
65	103	142.0	3.52	65	46.2	**38.5**
59	96	134.0	3.49	69	40.7	**25.4**
67	108	150.0	3.54	84	62.7	**20.9**
68	100	130.0	8.07	62	**36.8**	60.3
66	113	153.0	8.13	58	68.2	**56.1**
75	115	152.0	8.14	59	49.3	**30.7**
69	111	150.0	8.1	38	73.9	**44.9**
72	116	155.0	8.07	58	95.8	**52.8**
73	117	158.0	8.14	67	**53.4**	**53.4**
68	113	151.0	8.12	69	64.7	**45.6**
70	115	155.0	8.14	54	62.9	**51.4**
73	119	161.0	8.07	58	58.9	**53.4**
72	116	157.0	8.06	44	77.8	**45.8**
120	176	216.0	21.17	59	**36.7**	51.7
237	288	329.0	21.4	27	**15.6**	17.7
235	292	332.0	21.27	31	26.4	**21.7**
236	290	327.0	21.39	25	**16.9**	19.5
231	287	325.0	21.27	30	22.1	**21.6**
227	284	325.0	21.97	22	31.7	**21.6**
232	289	326.0	21.19	25	25.9	**20.3**
235	287	327.0	21.31	24	30.2	**20.4**
235	286	326.0	21.62	27	28.1	**18.3**
233	288	329.0	21.43	35	27.0	**18.9**
306	368	402.0	59.04	24	19.3	**10.8**
308	366	404.0	59.11	24	22.7	**20.1**
312	371	407.0	61.24	18	17.3	**18.6**
307	369	402.0	58.94	26	19.5	**18.2**
319	366	399.0	59.0	15	16.6	**11.6**
327	370	405.0	59.18	20	14.1	**12.2**
319	368	402.0	58.94	15	14.7	**12.2**
329	370	407.0	59.13	16	14.6	**11.9**
317	370	405.0	58.92	22	18.6	**15.8**
335	367	399.0	59.26	7	9.3	**6.3**

Com base na Fig. 3, pode-se constatar que, na medida em que o tamanho das instâncias aumenta, as soluções geradas pela regra de prioridade se aproximam mais das geradas pela meta

heurística. Isto se dá pela dificuldade de atingir boas soluções em ambientes de busca grandes como em 100 trabalhos, além do tempo computacional ser alto. Das regras de prioridades testadas a que mais se aproximou do AG foi a RAND, visto que ela foi executada várias vezes, devido sua natureza aleatória.

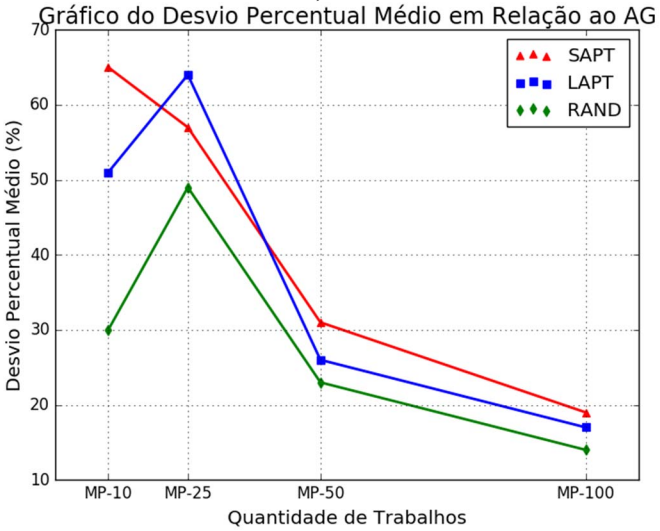


Figura 3. Desvio percentual médio.

Das demais regras a que mais se aproxima, na maioria dos testes, é a LAPT. Mesmo assim, a meta heurística consegue se sobressair pois, mesmo com o aumento da quantidade de trabalhos, o desvio percentual médio não chega a 10 %, caracterizando uma lacuna considerável.

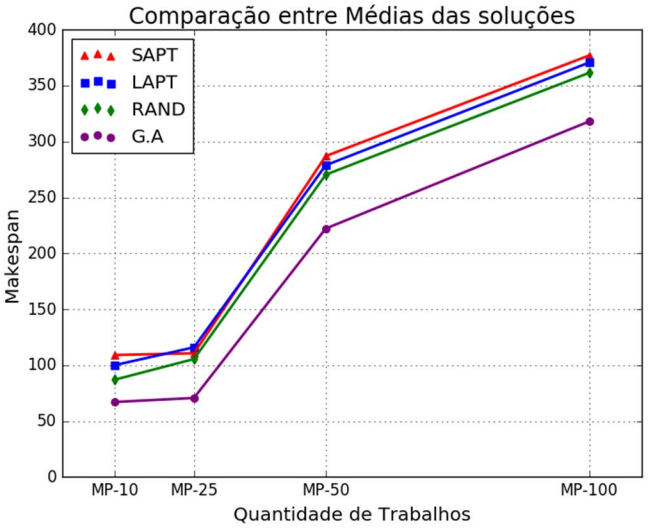


Figura 4. Resultados médios das instâncias.

V. ESTUDO DE CASO

A empresa em questão tem o objetivo de industrializar rochas ornamentais a fim de transforma-las em peças de decoração, peitoris, soleiras ou revestimento de pisos e fachadas, para assim serem comercializados como pisos, azulejos e peças decorativas diversas.

O processo de transformação consiste em cortar blocos de rochas maciços que possuem um peso médio de 1600 kg e transformá-los em peças retangulares de rochas com cerca de 5m$_2$. O processo de corte é realizado por máquinas de multi-fios, que realizam cortes verticais nas rochas, e teares mecânicos, que desempenhavam a mesma função das máquinas de multi-fios, todavia com um tempo de processamento maior. Os tempos de corte de cada rocha são computados em minutos. Os tempos de *setup* são baseados na dureza de cada rocha, por exemplo o *setup* de uma rocha leve após uma rocha dura é bem maior comparado ao *setup* se uma outra rocha dura fosse processada novamente. Essa diferença nos tempos de preparação decorre devido a troca dos fios que realizam os cortes que são diferentes para cada tipo de rocha.

O problema abordado no estudo de caso consiste em sequenciar o processamento de 50 trabalhos com tempos de processamentos distintos em 13 máquinas que trabalham em paralelo, no qual 2 são de multi-fios e o restante são teares mecânicos, com o objetivo de encontrar o menor *makespan*, para assim, reduzir o tempo de processamento do pedido.

O resultado dos experimentos computacionais é apresentado na Tabela VIII. Como a empresa do presente estudo não possui nenhuma técnica de sequenciamento da produção vigente, a ordem de processamento utilizada para comparação é a ordem inicial que os produtos aparecem na linha de produção (PEPS) sem um ordenamento prévio, ou seja, a primeira rocha que entra é a primeira que sai. S = {1,2,3, ...,59,50}.

Foram realizados testes das regras de prioridades a fim de compara-las com o resultado obtido pelo algoritmo genético. Foram calculados os desvios percentuais de cada regra de despacho em relação ao AG com o objetivo de mensurar a distância percentual de cada em relação a meta heurística. Os tempos de execução das regras de prioridades são desprezíveis, por isso não foram computados, o tempo de execução do AG foi de 22.97s.

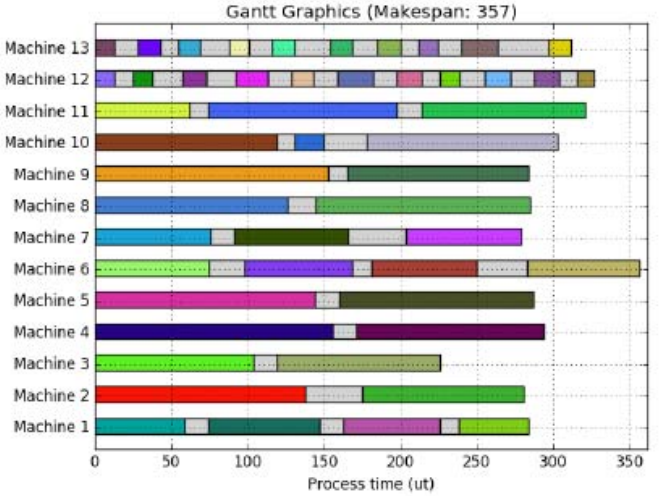
TABELA VIII
RESULTADOS DOS TESTES DO ESTUDO DE CASO

Instância	PEPS	SAPT	LAPT	RAND	GA Híbrido
50 rochas/13 máquinas	357	330	330	347	**278**
Desvios – AG (%)	28.4	**18.7**	**18.7**	24.8	-

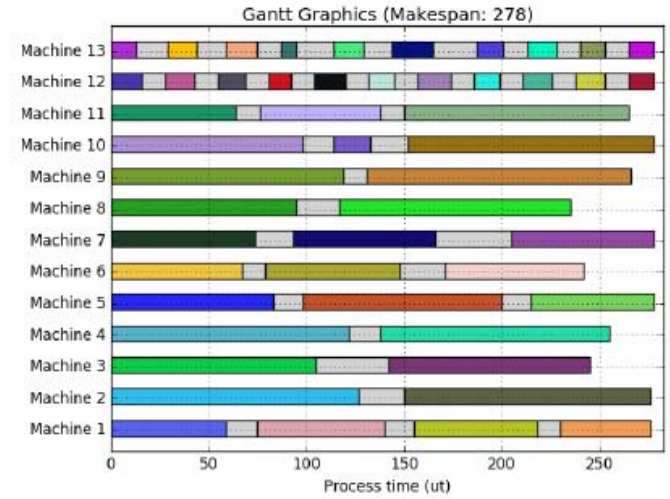
O melhor sequenciamento gerado e o menor desvio percentual estão em negrito para uma melhor visualização. Com base nos resultados obtidos, pode-se observar que a regra de sequenciamento adotada na empresa apresenta resultados de qualidade inferior a todas as heurísticas apresentadas. O desvio percentual da PEPS em relação ao AG é 28.4 %, convertendo a diferença do tempo de processamento entre a PEPS e o AG em horas, o algoritmo genético termina o processamento em 1,32 horas a menos do que a PEPS. As regras de prioridades que mais se aproximaram do AG foram a LAPT e SAPT empatadas.

Com base na Fig. 5, como já mencionado, as máquinas de multi-fios (12 e 13) são mais velozes do que as demais, pois as quantidades de trabalhos alocados nelas é bem maior do que nas outras, denotando que o algoritmo proposto busca uma alocação mais dinâmica, sempre se baseando na distribuição de trabalhos ao longo das máquinas e priorizando a alocação dos trabalhos nas máquinas com os menores tempos de processamento. É possível verificar, também, que a máquina gargalo do problema (máquina 6) obteve uma redução significante no *makespan*. Comparando o AG com a regra usual da empresa, no sequenciamento gerado pela meta-heurística,

grande parte das máquinas de corte terminam o processo em tempos próximos, denotando um certo equilíbrio na solução.



(a) PEPS



(b) AG Híbrido

Figura 5. Comparação entre a meta-heurística proposta e a alocação PEPS da empresa.

VI. CONCLUSÃO E ESTUDOS FUTUROS

Nesse Trabalho foram apresentadas novas formas de resolver o problema de sequenciamento da produção em ambiente de máquinas paralelas não relacionadas com tempos de preparação dependentes da sequência de processamento.

É notório que a utilização dos processos de *restart*, *Simulated Annealing* e busca local incorreram em melhorias consideráveis no algoritmo. Percebe-se, portanto, que o AG se mostrou superior em relação aos outros modelos de solução. Como os setups variavam dependendo da sequência de processamento, verificou-se que o AG buscava uma sequência que priorizava os *setups* menores promovendo reduções consideráveis no *makespan*. A regra de prioridade LAPT se mostrou melhor que a SAPT e a RAND se mostrou melhor que as outras duas, devido a quantidade de vezes que foi testada.

A abordagem proposta se mostrou eficiente para a resolução prática do problema abordado no estudo de caso, podendo ser aplicada rotineiramente no sequenciamento da produção no ambiente retratado.

Como sugestão para estudos futuros, propõe-se testar o algoritmo genético híbrido proposto em um conjunto maior de instâncias, objetivando avaliar o comportamento do mesmo em problemas de maiores dimensões. Outra proposição seria implementar regras de prioridade que analisassem os tempos de processamento e de preparação não de uma forma média, mas sim ponderada, como a construção de um algoritmo guloso que analisa a sequência de processamento de uma forma mais geral para, assim, verificar se essa nova regra de prioridade possui um desvio percentual menor comparado as outras em relação ao AG.

Outra proposta seria implementar mecanismos de buscas mais dinâmicos para atuarem diretamente na máquina gargalo reduzindo o *makespan* sem modificar as ordenações das demais máquinas.

No que se refere ao estudo de caso, pode ocorrer, na prática, que os planejadores do sistema desejam restringir o processamento de alguns pedidos em determinadas máquinas. Assim, os algoritmos propostos poderiam incorporar restrições de precedência, consubstanciando em uma outra variante do problema de máquinas paralelas.

AGRADECIMENTOS

O primeiro autor agradece ao Programa de Educação Tutorial (PET) do curso de engenharia de produção mecânica da Universidade Federal do Ceará (UFC) e à Coordenadoria de Apoio ao Discente da UFC (CAD), pela bolsa de estudos. O segundo autor agradece ao Conselho Nacional de Pesquisa (CNPQ).

REFERÊNCIAS

[1] J. G. Root, “Scheduling with deadlines and loss functions on k parallel machines,” Management Science, vol. 11, no. 3, pp. 460–475, 1965.
[2] C. Moodie and S. Roberts, “Experiments with priority dispatching rules in a parallel processor shop,” International Journal of Production Research, vol. 6, no. 4, pp. 303–312, 1967.
[3] R. R. Muntz and E. Coffman, “Optimal preemptive scheduling on twoprocessor systems,” IEEE Transactions on Computers, vol. 100, no. 11, pp. 1014–1020, 1969.
[4] A. Allahverdi, J. N. Gupta, and T. Aldowaisan, “A review of scheduling research involving setup considerations,” Omega, vol. 27, no. 2, pp.219 239, 1999.
[5] R. J. Wittrock, “Scheduling parallel machines with major and minor setup times,” International Journal of Flexible Manufacturing Systems, vol. 2, no. 4, pp. 329–341, 1990.
[6] Y. H. Lee and M. Pinedo, “Scheduling jobs on parallel machines with sequence-dependent setup times,” European Journal of Operational Research, vol. 100, no. 3, pp. 464–474, 1997.
[7] D.-W. Kim, K.-H. Kim, W. Jang, and F. F. Chen, “Unrelated parallel machine scheduling with setup times using simulated annealing,” Robotics and Computer-Integrated Manufacturing, vol. 18, no. 3, pp. 223–231, 2002.
[8] M. R. De Paula, M. G. Ravetti, G. R. Mateus, and P. M. Pardalos, “Solving parallel machines scheduling problems with sequence-dependent setup times using variable neighbourhood search,” IMA Journal of Management Mathematics, vol. 18, no. 2, pp. 101–115, 2007.
[9] V. c. A. Armentano and D. S. Yamashita, “Tabu search for scheduling on identical parallel machines to minimize mean tardiness,” Journal of intelligent manufacturing, vol. 11, no. 5, pp. 453–460, 2000.

[10] E. Vallada and R. Ruiz, "A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times," European Journal of Operational Research, vol. 211, no. 3, pp. 612– 622, 2011.
[11] D. Yilmaz Eroglu, H. C. Ozmutlu, and S. Ozmutlu, "Genetic algorithm with local search for the unrelated parallel machine scheduling problem with sequence-dependent set-up times," International Journal of Production Research, vol. 52, no. 19, pp. 5841–5856, 2014.
[12] A. Sadati, R. Tavakkoli-Moghaddam, B. Naderi, and M. Mohammadi, "Solving a new multi-objective unrelated parallel machines scheduling problem by hybrid teaching-learning based optimization," International Journal of Engineering-Transactions B: Applications, vol. 30, no. 2, p. 224, 2017.
[13] J. R. Zeidi and S. MohammadHosseini, "Scheduling unrelated parallel machines with sequence-dependent setup times," The International Journal of Advanced Manufacturing Technology, vol. 81, no. 9-12, pp. 1487–1496, 2015.
[14] M. N. Haddad, I. M. Coelho, M. J. F. Souza, L. S. Ochi, H. G. Santos, and A. X. Martins, "Garp: A new genetic algorithm for the unrelated parallel machine scheduling problem with setup times," in Chilean Computer Science Society (SCCC), 2012 31st International Conference of the. IEEE, 2012, pp. 152–160.
[15] L. P. Cota, M. N. Haddad, M. J. F. Souza, and A. X. Martins, "Um algoritmo heurístico para resolver o problema de sequenciamento em máquinas paralelas não-relacionadas com tempos de preparação dependentes da sequência," Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, vol. 3, no. 1, 2015.
[16] M. Nikabadi and R. Naderi, "A hybrid algorithm for unrelated parallel machines scheduling," International Journal of Industrial Engineering Computations, vol. 7, no. 4, pp. 681–702, 2016.
[17] J. P. de CM Nogueira, J. E. C. Arroyo, H. M. M. Villadiego, and L. B. Gonçalves, "Hybrid grasp heuristics to solve an unrelated parallel machine scheduling problem with earliness and tardiness penalties," Electronic Notes in Theoretical Computer Science, vol. 302, pp. 53–72, 2014.
[18] H. G. Santos, T. A. Toffolo, C. L. Silva, and G. Vanden Berghe, "Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem," International Transactions in Operational Research, 2016.
[19] T. Cheng and C. Sin, "A state-of-the-art review of parallel-machine scheduling research," European Journal of Operational Research, vol. 47, no. 3, pp. 271–292, 1990.
[20] H. Y. Fuchigami, J. V. Moccellin, and R. Ruiz, "Novas regras de prioridade para programação em flexible flow line com tempos de setup explícitos," in Revista Produção Online, vol. 25, no. 4. SciELO Brasil, 2015, pp. 779–790.
[21] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," ACM Computing Surveys (CSUR), vol. 35, no. 3, pp. 268–308, 2003.
[22] J. H. Holland, "Adaptation in natural and artificial systems. an introductory analysis with application to biology, control, and artificial intelligence," Ann Arbor, MI: University of Michigan Press, 1975.
[23] F. W. Glover and G. A. Kochenberger, Handbook of metaheuristics. Springer Science & Business Media, 2006, vol. 57.
[24] M. Gendreau and J.-Y. Potvin, Handbook of metaheuristics. Springer, 2010, vol. 2.
[25] D. Cavicchio, Adaptive Search Using Simulated Evolution. University of Michigan, Computer and Communication Siences Department., 1970.
[26] B. A. Prata, "A hybrid genetic algorithm for the vehicle and crew scheduling in mass transit systems," IEEE Latin America Transactions, vol. 13, no. 9, pp. 3020–3025, 2015.

Levi Ribeiro de Abreu é graduando em Engenharia de Produção Mecânica pela Universidade Federal do Ceará (UFC), Fortaleza, Ceará, Brasil. Atualmente suas pesquisas se concentram na área de Sequenciamento de Produção com a utilização de métodos heurísticos e Problemas de Corte e Empacotamento, com aplicações industriais em geral.

Bruno de Athayde Prata é graduado em Engenharia Civil pela Universidade Federal do Ceará (UFC), Fortaleza, Ceará, Brasil, em 2005. Obteve o título de mestre em Logística e Pesquisa Operacional pela UFC em 2007 e de Doutor, em Engenharia Industrial e Gestão pela Universidade do Porto (UP), Porto, Portugal, em 2011. É professor Adjunto do Departamento de Engenharia de Produção da UFC. Atualmente suas pesquisas se concentram em problemas de sequenciamento, com aplicações em Logística e Gestão de Operações.