# MÉTODO DE GERAÇÃO DE COLUNAS PARA O PROBLEMA DE DIMENSIONAMENTO E PROGRAMAÇÃO DE LOTES EM MÁQUINA ÚNICA

**Renan Spencer Trindade**
Programa de Pós-graduação em Informática – Universidade Federal de Santa Maria
Av. Roraima nº 1000 - Cidade Universitária - Bairro Camobi - Santa Maria - RS - 97105 900
renanspencer@gmail.com

**Olinto César Bassi de Araújo**
Colégio Técnico Industrial de Santa Maria – Universidade Federal de Santa Maria
Av. Roraima nº 1000 - Cidade Universitária - Bairro Camobi - Santa Maria - RS - 97105 900
olinto@ctism.ufsm.br

**Felipe Martins Müller**
Programa de Pós-graduação em Informática – Universidade Federal de Santa Maria
Av. Roraima nº 1000 - Cidade Universitária - Bairro Camobi - Santa Maria - RS - 97105 900
felipe@inf.ufsm.br

**Marcia Helena Costa Fampa**
Instituto de Matemática – Universidade Federal do Rio de Janeiro
Av. Athos da Silveira Ramos nº 149, Centro de Tecnologia, Bloco C – Rio de Janeiro - RJ
fampa@cos.ufrj.br

## RESUMO

Este trabalho propõe um método de geração de colunas com um novo conjunto de subproblemas para o problema de programação de lotes em máquina única. São consideradas tarefas com tamanhos e tempos de processamento não necessariamente idênticos, que devem ser agrupados em lotes que satisfaçam a capacidade da máquina. O tempo de processamento de um lote é definido pelo maior tempo de processamento das tarefas a ele alocadas. O objetivo do problema é minimizar o tempo total de processamento de todos os lotes (*makespan*). Testes comparativos com outras abordagens da literatura são apresentados e discutidos, demonstrando uma sensível melhora nos tempos computacionais.

**PALAVARAS CHAVE. geração de colunas, dimensionamento de lotes, máquina de processamento em lotes.**

**Área principal: Otimização Combinatória, Programação Matemática.**

## ABSTRACT

This paper proposes a method of column generation with a new set of subproblems for single batch processing machine scheduling problem. In this problem, each job has a corresponding non-identical processing time and size, which must be grouped in batches as long as the machine capacity is not exceeded. The processing time of each batch is defined by the longest processing time among all jobs in the batch. The objective of the problem is to minimize the completion time of all batches (makespan). Comparative tests of the proposed column generation and other approaches from the literature are presented and discussed, demonstrating a significant computational time improvement.

**KEYWORDS.Column generation, batch scheduling, batch processing machine.**

**Main area: Combinatorial Optimization, Mathematical Programming**

## 1. Introdução

A indústria vem se empenhando em desenvolver e implantar processos de produção com a finalidade de fornecer produtos e serviços com tecnologia sofisticada a preços acessíveis. Com esta finalidade, muitas delas utilizam máquinas de processamento em lote (BPM – *Batch ProcessingMachines*). Estas máquinas são responsáveis por realizar uma mesma tarefa em um grupo de produtos ao mesmo tempo, visando evitar configurações em equipamentos e facilitar o manuseio de material. A indústria de semicondutores está entre aquelas que possuem as maiores demandas por utilização destas máquinas (Mathirajan e Sivakumar, 2006). Estudos sobre a utilização de BPM também reportam aplicações em indústrias de fabricação de calçados (Fanti, et. al., 1996), aeronáutica (Zee, et al., 1997), metal (Ram e Patel, 1998), fundição (Mathirajan, et. al., 2004) e fabricação de móveis (Yuan, et. al., 2004).

Na indústria de semicondutores, estudos sobre a otimização do uso de BPM são motivados por testes em lote de confiabilidade de componentes realizados em câmaras térmicas (testes de *burn-in*). Este problema envolve tarefas que devem ser agrupadas em lotes, em que cada tarefa pode possuir atributos como tempo de processamento (*Processing Time*), tempo de liberação (*Release Time* ou *JobArrivals*), tempo de entrega (*Due Date*) e tamanho (*Size*). Todas as tarefas contidas em um lote são processadas simultaneamente por uma máquina que possui restrição de capacidade. O processamento do lote não pode ser interrompido após iniciado e as tarefas não podem ser removidas até que o processamento do lote seja terminado (não-preemptivo). O tempo total de processamento de um lote é igual ao tempo da tarefa que possui o maior tempo de processamento. O tempo total de utilização da máquina é igual ao somatório do tempo de processamento de todos os lotes (*makespan*). Estudos encontrados na literatura científica consideram problemas com máquina única (*Single Machine*) ou máquinas paralelas (*ParellelMachine*).

O problema envolvendo uma única máquina de processamento em lotes foi primeiramente abordado por Uzsoy (1994), que utilizou heurísticas baseadas no procedimento *first-fit*. Ghazvini e Dupont (1998) e Dupont e Ghazvini (1998) abordam o problema utilizando a estratégia *best-fit* para compor lotes viáveis, dentro do limite de capacidade da máquina. Um algoritmo aproximativo com a redução do pior caso é apresentado por Kashan, Karimi, e Ghomi (2009). Chen, Du, e Huang (2011) demonstram em seu trabalho que, a partir de uma perspectiva diferente do problema, minimizar o *makespan* é similar a minimizar os resíduos de tempo de processamento e tamanho (*waste*) de cada lote. Seguindo esta concepção, é proposto um algoritmo de agrupamento (*clustering*) para solucionar o problema. Utilizando esta mesma perspectiva, duas novas heurísticas foram propostas por Lee e Lee (2013) que utilizam modelos de programaçãointeira mista para selecionar as tarefas que formarão os lotes.

O algoritmo *branchandbound* é aplicado por Dupont e Flipo (2002). Em Parsa, Karimi, e Kashan (2010) é proposto o método de geração de colunas e o algoritmo *branchandprice*. Em Scalcon et. al (2012) é apresentado um novo modelo matemático utilizando restrições para tratamento de simetrias e o resolvedor comercial CPLEX é utilizado para resolução.

O método *simulatedannealing* é estudado por Melouk, Damodaran, e Chang (2004) utilizando soluções geradas pela troca da tarefa com mais tempo de processamento de um lote para outro. Abordagens utilizando algoritmos genéticos são estudadas em Kashan, Karimi, e Jenabi (2008). Uma aplicação da busca tabu é apresentada em Meng e Tang (2010). Em Xu, Chen, e Li (2012) é proposto um algoritmo de colônia de formigas.

Neste trabalho é proposta uma abordagem utilizando o método de geração de colunas para resolução do problema de dimensionamento e programação de lotes em máquina única a partir do modelo matemático apresentado em Scalcon et. al (2012). O problema em estudo é fortemente NP-difícil (Uzsoy, 1994) e pode ser classificado como $1|s_j,B|C_{\max}$ (Graham et. al, 1979). Resultados computacionais da abordagem proposta neste

trabalho são comparados com Parsa, Karimi, e Kashan (2010) e demonstram que é possível obter melhoras significativas em relação aos tempos computacionais.

As demais seções deste trabalho são organizadas como segue. Na Seção 2 o problema em estudo é formalmente definido, apresentando formulações de programação inteira mista. Na Seção 3 a geração de colunas para resolução do $1|s_j,B|C_{\max}$ de Parsa, Karimi e Kashan (2012) é brevemente revisada e uma nova metodologia utilizando o modelo de Scalcon et. al (2012) é apresentada. Na Seção 4 são encontrados os resultados computacionais e discussões. As conclusões são relatadas na Seção 5.

## 2. Descrição do problema

O problema $1|s_j,B|C_{\max}$ tem como objetivo minimizar o tempo total gasto por uma única máquina para processar lotes de tarefas. Cada tarefa $j$, $j = 1, \ldots, n$, possui um tempo de processamento $p_j$ e um tamanho $s_j$ não necessariamente idênticos as demais tarefas. Cada lote tem um tempo de processamento $C_k$ definido pelo maior tempo de processamento dentre as tarefas atribuídas a ele, ou seja, $C_k = \max\{p_j : x_{jk} = 1\}$, $k = 1 \ldots m$. A definição dos lotes deve respeitar a capacidade $B$ da máquina e o processamento destes ocorre de forma não-preemptiva. Uma vez que um lote pode ser composto por uma única tarefa, considera-se que o número de lotes é igual ao número de tarefas, $m = n$. O modelo matemático é formalizado a seguir:

**Variáveis**

$x_{jk} = \begin{cases} 1 & \text{se a tarefa } j \text{ for atribuída ao lote } k \\ 0 & \text{caso contrário} \end{cases}$

$y_k = \begin{cases} 1 & \text{se o lote } k \text{ for utilizado} \\ 0 & \text{caso contrário} \end{cases}$

$C_k$ : tempo de processamento do lote $k$

**Parâmetros**

$s_j$ : tamanho da tarefa $j$

$p_j$ : tempo de processamento da tarefa $j$

$B$ : capacidade da máquina

**Modelo 1:**

$$MinC_{max} = \sum_{k=1}^{m} C_k \qquad (1)$$

S.a.

$$\sum_{k=1}^{m} x_{jk} = 1 \qquad j = 1, \ldots, n \qquad (2)$$

$$\sum_{j=1}^{n} s_j x_{jk} \leq B y_k \qquad k = 1, \ldots, m \qquad (3)$$

$$x_{jk} \leq y_k \qquad j = 1, \ldots, n, \quad k = 1, \ldots, m \qquad (4)$$

$$p_j x_{jk} \leq C_k \qquad j = 1, \ldots, n, \quad k = 1, \ldots, m \qquad (5)$$

$$x_{jk} \in \{0, 1\} \qquad j = 1, \ldots, n, \quad k = 1, \ldots, m \qquad (6)$$

$$y_k \in \{0, 1\} \qquad k = 1, \ldots, m \qquad (7)$$

$$C_k \geq 0 \qquad k = 1, \ldots, m \qquad (8)$$

A função objetivo (1) minimiza o tempo total gasto pela máquina, ou seja, o somatório do tempo de processamento dos lotes determinados na solução ótima. A restrição (2) garante que cada tarefa seja atribuída a um único lote. A restrição (3) segura que a capacidade (tamanho) da máquina seja respeitada. A restrição (4) é redundante, mas é adicionada para melhorar o desempenho de resolvedores baseados em relaxações lineares. A restrição (5) define o tempo de processamento de cada lote $k$. As restrições (6), (7) e (8) determinam o domínio das variáveis.

Em Scalcon et. al (2012) é proposto um modelo matemático mais forte. Visando reduzir o espaço de soluções, este modelo é baseado em restrições de quebra de simetrias, fornecendo melhores limitantes pela relaxação contínua do problema, não alterando a solução ótima do problema original. Para que seja utilizado, é necessário que o conjunto de tempos de processamento seja ordenado de forma crescente.

O modelo de Scalcon et. al (2012) é formalizado abaixo:

**Modelo 2:**

$$MinC_{max} = \sum_{k=1}^{m} p_k x_{jk}$$

S.a.

(3), (6)

$$\sum_{k=1}^{m} x_{jk} = 1 \qquad j = 1, \dots, n \qquad (9)$$

$$x_{jb} \leq x_{bb} \qquad j = 1, \dots, n, \quad k = 1, \dots, n \qquad (10)$$

A restrição (9) determina que cada tarefa só pode ser designada a um lote com índice maior ou igual ao índice da própria tarefa. A restrição (10) determina que um lote $j$ seja criado somente se a tarefa de índice $j$ estiver alocada a ele.

## 3. Metodologia

Nesta seção é feita uma revisão do método de geração de colunas proposto em Parsa et. al (2010) e, a seguir, é apresentada a metodologia proposta neste trabalho.

A geração de colunas (GC) é uma técnica utilizada quando um problema possui um grande número de variáveis de decisão. O melhor custo reduzido de uma variável não básica é encontrado através de um novo problema de otimização, com a finalidade de evitar a enumeração de todas as variáveis do problema. A partir da decomposição de Dantzig-Wolfe (Dantzig, 1960), o problema original é particionado em Problema Mestre (PM) e Subproblema (SP).

Para o problema em estudo, cada coluna do problema mestre correspondente a $x_k$ representa um lote criado a partir do SP. A constante $a_{jk}$ determina se uma tarefa participa ($a_{jk}$= 1)ou não ($a_{jk}$ = 0) da solução gerada pelo SP e $P_k$ indica o respectivo tempo de processamento do lote. O modelo mestre pode ser formalizado na forma que segue:

**Problema Mestre:**

$$Z_{PM}: \; Min \sum_{k=1}^{N} P_k x_k \qquad (11)$$

S.a.

$$\sum_{k=1}^{N} a_{jk} x_k = 1 \qquad j = 1, \dots, n \qquad (12)$$

$$x_k \in \{0, 1\} \qquad k = 1, \dots, m \qquad (13)$$

A função objetivo (11) minimiza o tempo total gasto pela máquina. A restrição (12) garante que cada tarefa seja designada a um único lote. A restrição (13) garante o domínio binário da variável de decisão.

Através da relaxação da variável binária (13) do $Z_{PM}$, define-se $Z_{RPM}$ com o qual é possível obter os multiplicadores duais $\pi_j$ a serem inseridos no SP.

O processo de geração de colunas é iterativo utiliza os seguintes passos:

1- Resolver o $Z_{RPM}$;
2- Utilizando os valores duais da solução ótima de $Z_{RPM}$, atualizar os coeficientes do SP;
3- Resolver o SP;
4- Inserir a nova coluna em $Z_{RPM}$ a partir da solução ótima do SP.

### 3.1. Geração de Colunas (Parsa et. al, 2010)

Na elaboração apresentada em Parsa et. al (2010) para o SP, é utilizado um único subproblema, gerando uma única coluna por iteração do método. Cada coluna pode conter qualquer formação de lote que possa melhorar a solução do problema mestre $Z_{RPM}$.

Nesta formulação, a variável $y_j$ indica se a tarefa $j$ pertence ou não ao lote que será gerado. A variável $P_k$ define o tempo de processamento deste lote. O SP é formalmente definido a seguir:

**Subproblema 1:**

$$C^*: Min P_k - \sum_{j=1}^{n} (\pi_j y_j) \quad (14)$$

S.a.

$$\sum_{j=1}^{n} s_j y_j \leq B \quad (15)$$

$$p_j x_{jk} \leq P_k \qquad j = 1, \dots, n \quad (16)$$

$$y_j \in \{0,1\} \qquad j = 1, \dots, n \quad (17)$$

A função objetivo (14) minimiza o custo reduzido. A restrição (15) garante que a capacidade da máquina não seja excedida. A restrição (16) define o tempo de processamento do lote. A restrição (17) define o domínio binário da variável.

Para que a geração de colunas comece é necessária uma solução inicial. Parsa et. al. (2010) sugere a criação de uma solução inicial utilizando soluções de duas heurísticas bastante utilizadas na literatura: BFF (*Batch firt-fit*) (Uzsoy, 1994) e BBF (*Batch best-fit*) (Dupont e Ghazvini, 1998). Cada heurística é executada utilizando quatro tipos de ordenações nas instâncias: Ordens crescente e decrescente de tempo de processamento, decrescente pelo tamanho das tarefas e crescente pela razão de $p_j/s_j$.

O procedimento utilizado adiciona no modelo $Z_{PM}$ todas as soluções encontradas nestas heurísticas. O modelo então é resolvido para encontrar uma solução inicial de melhor qualidade. Esta solução é incluída no modelo $Z_{RPM}$ onde se inicia o processo de geração de colunas.

### 3.2. Geração de Colunas baseada em Scalcon et. al (2012)

Baseando-se no método para tratar simetrias proposto por Scalcon et. al (2012), o modelo proposto é decomposto em $n$ subproblemas. Para isso, é efetuada uma ordenação em ordem não decrescente das tarefas de acordo com o tempo de processamento. Assim, assumindo $m = n$ e que um lote $i$ poderá conter apenas as primeiras $i$ tarefas, sendo a tarefa $y_i = 1$ obrigatória, infere-se que este terá sempre o tempo de processamento $p_i$. Portanto, cada subproblema é designado a gerar lotes diferentes entre si, ou seja, um subproblema $i$ nunca gerará um lote igual ao gerado no subproblema $j$, para $i \neq j$.

A Tabela 1 ilustra as possibilidades de criações de lotes para cada subproblema, mostrando o tempo de processamento (*makespan*) e os valores que cada variável pode assumir em cada subproblema.

Tabela 1 – Possibilidades de criação de lotes para o Subproblema 2.

| SP: / Tarefa: | 1 | 2 | ... | $n$ |
|---|---|---|---|---|
| $y_1$ | {1} | {0,1} | {0,1} | {0,1} |
| $y_2$ | {0} | {1} | {0,1} | {0,1} |
| ... | {0} | {0} | {1} | {0,1} |
| $y_n$ | {0} | {0} | {0} | {1} |
| ***Makespan*** | $p_1$ | $p_2$ | $p_{...}$ | $p_n$ |

A partir deste conceito, um conjunto de subproblemas é formulado sem que seja necessário considerar a restrição (16). Os subproblemas são definidos conforme segue:

**Parâmetros:**

$k$: índice do subproblema

**Subproblema 2:**

$$C^*: Min p_k - \pi_i - \sum_{j=1}^{k-1} (\pi_j y_j) \quad (18)$$

S.a.

$$\sum_{j=1}^{k-1} s_j y_j \leq (B - s_k) \quad (19)$$

$$y_j \in \{0,1\} \qquad j = 1, \dots, n \quad (20)$$

O Subproblema 2 corresponde a um conjunto de $n$ subproblemas do tipo mochila, um para cada lote $k$. A função objetivo (18) minimiza o custo reduzido da nova solução, sendo $p_k$ uma constante decorrente do fato que $y_k = 1$. A restrição (19) garante que a nova solução atenda a capacidade restante da máquina. A restrição (20) garante o domínio binário da variável.

Cada subproblema $k$ gera uma solução que pode ter um custo reduzido negativo para o PM. Para o melhor aproveitamento do esforço computacional e tentar promover uma diminuição do tempo de convergência, em cada iteração não só a solução com melhor custo reduzido é incluída no PM, mas sim todas as soluções que apresentarem um custo reduzido negativo. Isto faz com que o Subproblema 2 inclua mais colunas no problema mestre do que o Subproblema 1.

O problema mestre$Z_{RPM}$ e a geração da solução inicial utilizados em Parsa et. al (2010) também são empregados nesta proposta de geração de colunas.

## 4. Resultados computacionais

Os testes foram realizados em um computador Intel Core i5 2,4GHz com 4GB de memória RAM utilizando Windows 7 de 64bits e o programa CPLEX 12.4.

O conjunto de instâncias foi gerado de acordo a metodologia utilizada em Parsa et. al. (2010). Nesta metodologia são definidos diferentes números de tarefas, limites de tempo de processamento, limites de tamanho das tarefas e capacidade da máquina.

O tamanho das tarefas e o tempo de processamento são aleatoriamente escolhidos, com distribuição uniforme, dentro do respectivo intervalo. Diferentes instâncias são montadas conforme a tabela a seguir:

Tabela 2 – Fatores considerados na geração das instâncias aleatórias.

| Número de tarefas | Tempos de processamento | Tamanho das tarefas | Capacidade da máquina |
|---|---|---|---|
| 20, 40, 60, 80, 100 | [1,10], [1,5] | [1,10], [4,8], [1,5], [2,4] | 10 |
| | | [1,5], [2,4] | 5 |

Foram geradas 10 instâncias testes para cada combinação dos itens da Tabela 2, totalizando 600 instâncias para a realização dos testes. O tempo máximo de execução para todos os testes foi limitado arbitrariamente em 1800 segundos.

O Modelo 2 e o método de geração de colunas, com o Subproblema 1 e Subproblema 2, foram testados utilizando as instâncias geradas. Os resultados são apresentados nas tabelas 3 e 4. Nestas tabelas, as colunas representam o tipo de instância, o número de tarefas, os tempos de processamento (T(s)), a quantidade de soluções inteiras comprovadamente ótimas encontradas (#O) e o *gap* definido por : GAP = $(FOModelo\ 2 - FOGC)\ /\ (FOGC)\ *\ 100$ . A coluna (#O) é mostrada somente uma vez para o método de geração de colunas porque as geração de colunas com os dois tipos desubproblemas testados apresentaram as mesmas soluções.

Tabela 3 – Resultados computacionais para B = 10.

Resultados computacionais para P ϵ [1-10]

| Inst. | | GC 1 | GC 2 | GC | Modelo 2 | | GAP |
|---|---|---|---|---|---|---|---|
| S | T | T(s) | T(s) | #O | T(s) | #O | |
| [1,10] | 20 | 0,34 | 0,14 | 2 | 0,03 | 10 | 1,25 |
| | 40 | 0,53 | 0,19 | 3 | 0,06 | 10 | 0,75 |
| | 60 | 1,53 | 0,57 | 0 | 0,07 | 10 | 0,42 |
| | 80 | 3,85 | 1,24 | 0 | 0,87 | 10 | 0,67 |
| | 100 | 4,47 | 1,17 | 0 | 0,34 | 10 | 0,39 |
| *Média* | | *2,14* | *0,66* | *1* | *0,27* | *10* | *0,70* |
| [4,8] | 20 | 0,10 | 0,06 | 9 | 0,04 | 10 | 0,40 |
| | 40 | 0,27 | 0,14 | 3 | 0,01 | 10 | 0,46 |
| | 60 | 0,70 | 0,26 | 5 | 0,06 | 10 | 0,55 |
| | 80 | 1,66 | 0,43 | 0 | 0,12 | 10 | 0,46 |
| | 100 | 3,05 | 0,67 | 0 | 0,10 | 10 | 1,28 |
| *Média* | | *1,16* | *0,31* | *3,40* | *0,07* | *10* | *0,63* |
| [1,5] | 20 | 0,41 | 0,33 | 4 | 0,04 | 10 | 2,81 |
| | 40 | 2,70 | 1,34 | 0 | 4,93 | 10 | 2,60 |
| | 60 | 6,37 | 2,25 | 0 | 1800* | 5 | 3,84 |
| | 80 | 15,67 | 3,83 | 0 | 1800* | 0 | 3,01 |
| | 100 | 22,48 | 6,18 | 0 | 1800* | 0 | 3,67 |
| *Média* | | *9,53* | *2,79* | *0,80* | 1800* | *5* | *3,19* |
| [2,4] | 20 | 0,53 | 0,28 | 1 | 0,12 | 10 | 2,89 |
| | 40 | 2,57 | 1,06 | 0 | 69,35 | 10 | 3,19 |
| | 60 | 5,30 | 1,72 | 0 | 1800* | 1 | 2,72 |
| | 80 | 8,37 | 2,84 | 0 | 1800* | 3 | 2,78 |
| | 100 | 13,00 | 4,02 | 0 | 1800* | 0 | 2,89 |
| *Média* | | *5,95* | *1,98* | *0,20* | 1800* | *4,80* | *2,89* |

Resultados computacionais para P ϵ [1-5]

| Inst. | | GC 1 | GC 2 | GC | Modelo 2 | | GAP |
|---|---|---|---|---|---|---|---|
| S | T | T(s) | T(s) | #O | T(s) | #O | |
| [1,10] | 20 | 0,17 | 0,09 | 6 | 0,01 | 10 | 0,71 |
| | 40 | 0,44 | 0,22 | 0 | 0,03 | 10 | 0,90 |
| | 60 | 1,45 | 0,42 | 0 | 0,14 | 10 | 0,64 |
| | 80 | 3,41 | 1,21 | 0 | 0,25 | 10 | 0,67 |
| | 100 | 3,07 | 0,96 | 0 | 0,33 | 10 | 0,65 |
| *Média* | | *1,71* | *0,58* | *1,20* | *0,15* | *10* | *0,71* |
| [4,8] | 20 | 0,08 | 0,04 | 10 | 0,00 | 10 | 0,00 |
| | 40 | 0,29 | 0,12 | 3 | 0,02 | 10 | 0,50 |
| | 60 | 0,60 | 0,22 | 4 | 0,04 | 10 | 0,63 |
| | 80 | 1,31 | 0,47 | 0 | 0,09 | 10 | 1,06 |
| | 100 | 1,83 | 0,55 | 0 | 0,19 | 10 | 0,46 |
| *Média* | | *0,82* | *0,28* | *3,40* | *0,07* | *10* | *0,53* |
| [1,5] | 20 | 0,45 | 0,36 | 0 | 0,07 | 10 | 4,22 |
| | 40 | 2,07 | 1,12 | 0 | 12,02 | 10 | 5,22 |
| | 60 | 4,56 | 1,86 | 0 | 1800* | 8 | 3,19 |
| | 80 | 8,32 | 2,94 | 0 | 1800* | 5 | 3,77 |
| | 100 | 14,00 | 4,05 | 0 | 1800* | 3 | 3,09 |
| *Média* | | *5,88* | *2,07* | 0 | 1800* | *7,20* | *3,90* |
| [2,4] | 20 | 0,41 | 0,24 | 0 | 0,05 | 10 | 3,83 |
| | 40 | 1,71 | 0,77 | 0 | 158,15 | 10 | 5,16 |
| | 60 | 3,24 | 1,59 | 0 | 1800* | 6 | 3,55 |
| | 80 | 4,57 | 1,89 | 0 | 1800* | 4 | 3,65 |
| | 100 | 0,01 | 0,00 | 0 | 1800* | 4 | 2,43 |
| *Média* | | *1,99* | *0,90* | *0* | 1800* | *6,80* | *3,72* |

*Pelo menos uma instância que participou da média atingiu o limite

Tabela 4 – Resultados computacionais para B = 5.

Resultados computacionais para P ∈ [1-10]

| Inst. | | GC 1 | GC 2 | GC | Modelo 2 | | GAP |
|---|---|---|---|---|---|---|---|
| S | T | T(s) | T(s) | #O | T(s) | #O | |
| | 20 | 1,45 | 0,69 | 10 | 0,09 | 10 | 0,00 |
| | 40 | 6,37 | 2,12 | 7 | 0,34 | 10 | 0,26 |
| [1,5] | 60 | 14,32 | 3,63 | 4 | 3,46 | 10 | 0,45 |
| | 80 | 27,36 | 7,18 | 5 | 23,95 | 10 | 0,31 |
| | 100 | 51,03 | 11,97 | 0 | 9,17 | 10 | 0,51 |
| *Média* | | *20,11* | *5,12* | *5,20* | *7,4* | *100* | *0,31* |
| | 20 | 0,75 | 0,17 | 9 | 0,05 | 10 | 0,57 |
| | 40 | 3,73 | 1,30 | 7 | 0,33 | 10 | 0,25 |
| [2,4] | 60 | 9,59 | 2,37 | 9 | 0,31 | 10 | 0,07 |
| | 80 | 17,25 | 2,72 | 10 | 0,36 | 10 | 0,00 |
| | 100 | 30,53 | 5,43 | 3 | 1,14 | 10 | 0,26 |
| *Média* | | *12,37* | *2,4* | *7,60* | *0,44* | *10* | *0,23* |

Resultados computacionais para P ∈ [1-5]

| Inst. | | GC 1 | GC 2 | GC | Modelo 2 | | GAP |
|---|---|---|---|---|---|---|---|
| S | T | T(s) | T(s) | #O | T(s) | #O | |
| | 20 | 0,95 | 0,80 | 8 | 0,08 | 10 | 0,22 |
| | 40 | 5,24 | 2,45 | 8 | 0,33 | 10 | 0,19 |
| [1,5] | 60 | 10,89 | 4,06 | 4 | 1,29 | 10 | 0,62 |
| | 80 | 15,68 | 5,71 | 8 | 1,93 | 10 | 0,20 |
| | 100 | 30,14 | 8,60 | 0 | 5,46 | 10 | 0,50 |
| *Média* | | *12,58* | *4,32* | *5,60* | *1,82* | *10* | *0,35* |
| | 20 | 0,61 | 0,22 | 10 | 0,02 | 10 | 0,00 |
| | 40 | 3,89 | 0,95 | 6 | 0,30 | 10 | 0,94 |
| [2,4] | 60 | 6,13 | 2,15 | 10 | 0,17 | 10 | 0,00 |
| | 80 | 10,45 | 3,82 | 10 | 0,86 | 10 | 0,00 |
| | 100 | 18,74 | 5,04 | 3 | 1,07 | 10 | 0,30 |
| *Média* | | *7,96* | *2,44* | *7,80* | *0,48* | *100* | *0,25* |

As Tabelas 3 e 4 revelam que o método de geração de colunas, mesmo sem proporcionar soluções inteiras, possui um tempo superior ao Modelo2 para uma parcela das instâncias testadas. Este experimento demonstra que o a formulação utilizada no Modelo 2 apresenta bons resultados quando resolvido com o resolvedor CPLEX. Uma explicação possível para este bom desempenho pode ser atribuída aos cortes automáticos que o resolver possui para restrições do tipo mochila.No entanto, é fácil verificar que esse bom desempenho não se aplicada a todas as instâncias com $B$= 10. Esse resultado sugere a necessidade de reavaliar a metodologia para geração de instâncias artificiais para esse problema, de modo a construir um conjunto mais desafiador para resolvedores genéricos como o CPLEX quando utilizado em conjunto com o Modelo 2.

Comparando os resultados da geração de colunas, a utilização do Subproblema 1 apresenta uma melhora em tempos computacionais na comparação com o Subproblema 2. Na análise final, os tempos computacionais foram 67,59% menores para B = 10 e 73,06% menores para B = 5, em média. Em todos os casos, os valores de função objetivo das soluções encontradas utilizando o Subproblema 2 foram idênticas aos encontrados no Subproblema 1. O GAP médio para instâncias com B = 10 é de 2,03%, enquanto para B = 5 a média foi de 0,28%.

Depois de concluído o processamento das gerações de colunas, foram realizados testes computacionais com o problema mestre final sujeito a restrições de domínio binário. Esta transformação considera apenas as colunas inseridas ao longo do processo de geração de colunas e a resolução do problema mestre com a restrição de binariedade das variáveis (13). Obviamente esta estratégia não garante a otimalidade da nova solução obtida, no entanto, o resultado ótimo é encontrado em 91,5% dos casos testados com o método de geração de colunas proposto, contra 67,8% da geração de colunas utilizando o Subproblema 1. Esta diferença pode ser facilmente justificada devido a inserção de um número maior de colunas no problema mestre com utilização do Subproblema 2.

## 5. Conclusões

Neste trabalho é proposta uma abordagem utilizando o método de geração de colunas para resolução do problema de dimensionamento e programação de lotes em máquina única, a partir do modelo matemático apresentado em Scalcon et. al (2012).

Os testes computacionais compararam os resultados de tempo e soluções da abordagem proposta e Parsa et. al. (2012), revelando uma melhora sensível nos tempos computacionais. As soluções encontradas por ambos os métodos obtiveram o mesmo valor

de função objetivo, confirmando a consistência da geração de colunas proposta. Mesmo que o modelo do subproblema proposto seja resolvido $n$ vezes a cada iteração, os resultados comprovam que, aliado com a estratégia de adicionar todas as colunas com valor de função objetivo negativo, a utilização do mesmo é vantajosa.

Embora a geração de colunas resulte em uma solução não discreta, o ganho de tempo apresentado neste trabalho presume vantagens na aplicação de métodos como *branchandprice*, e sua utilização pode ser recomendada em instâncias para quais o desempenho do Modelo 2 não é competitivo.

## Referências

**Chen, H., Du, B., Huang, G.Q.** (2011), Scheduling a batch processing machine with non-identical job sizes: a clustering perspective, *International Journal of Production Research*, 49 (19), 5755–5778.

**Dantzig, G., Wolfe P.** (1960), Decomposition principle for linear programs, *Operation Research*, 8:101–11.

**Dupont, L., Flipo, C.D.** (2002), Minimising the makespan on a batch machine with non-identical job sizes: an exact procedure. *Computers & Operations Research*, 29, 807–819.

**Dupont, L., Ghazvini, J. F.** (1998), Minimizing makespan on a single batch processing machine with non-identical job sizes. *European Journal of Automation Systems*, 32, 431–40.

**Fanti, M. P., Maione, B., Piscitelli, G., Turchiano, B.** (1996), Heuristic scheduling of jobs on a multi-product batch processing machine.*Int J Prod Res*, 34(8), 2163–2186.

**Ghazvini, J. F., Dupont L.** (1998), Minimizing mean flow time on a single batch processing machine with non-identical job sizes. *International Journal of Production Economics*, 55, 273–80.

**Graham, R. L., Lawler, E. L., Lenstra, J.K., RinnooyKan, A.H.G.** (1979), Optimization and Approximation in Deterministic Suquencing and Scheduling: a Survey, *Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium*, Elsevier, 5, 287–326.

**Kashan, A.H., Karimi, B., Ghomi, S.M. T.F.** (2009). A note on minimizing makespan on a single batch processing machine with nonidentical job sizes.*Theoretical Computer Science*, 410, 2754–2758.

**Kashan, A.H., Karimi, B., Jenabi, M.** (2008), A hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes.*Computers & Operations Research*, 35, 1084–1098.

**Lee, Y. H., Lee, Y. H.** (2013), Minimisingmakespan heuristics for scheduling a single batch machine processing machine with non-identical job sizes, *International Journal of Production Research*.

**Mathirajan, M., Sivakumar, A. I.** (2006), A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. *International Journal of Advanced Manufacturing Technology*, 2006, 29(9–10), 990–1001.

**Mathirajan, M., Sivakumar, A.I., Chandru, V.** (2004), Scheduling algorithms for heterogeneous batch processors with incompatible job families.*J IntellManuf*, 15, 787–803

**Melouk, S., P. Damodaran, Chang, P.Y.** (2004), Minimisingmakespan for single machine batch processing with non-identical job sizes using simulated annealing. *International Journal of Production Economics*, 87, 141–147.

**Meng, Y., Tang, L.,** (2010). A tabu search heuristic to solve the scheduling problem for a batch-processing machine with non-identical job sized. *International conference on logistics systems and intelligent management*, 9-10, 1703–1707.

**Parsa, N.R., Karimi, B., Kashan, A.H.** (2010), A branch and price algorithm to minimisemakespan on a single batch processing machine with non-identical job sizes.*Computers & Operations Research*, 37, 1720–1730.
**Ram, B., Patel, G.** (1998) Modeling furnace operations using simulation and heuristics. *Proc 1998 wintersimulationconference*, 957–963.
**Scalcon, C. G, de Araújo, O. C. B., Müller, F. M.** (2012), A Um modelo de otimização para o problema de dimensionamento e programação de lotes de produção em máquina única, *XLIVSimpósio Brasileiro de Pesquisa Operacional*, anais.
**Uzsoy R.** (1994) A single batch processing machine with non-identical job sizes. International Journal of Production Research, 32, 1615–35.
**Xu, R., Chen, H., Li, X.** (2012)Makespanminimisation on single batch-processing machine via ant colony optimization,*Computers & Operations Research*, 39, 582–593.
**Yuan, J. J., Liu, Z. H., Ng, C.T., Cheng, T.C.E.** (2004), The unbounded single machine parallel batch scheduling problem with family jobs and release dates to minimize makespan. *TheorComputSci,* 320(2–3), 199–213.
**Zee, D. J. van der., Van Harten, A., Schuur P. C.** (1997), Dynamic job assignment heuristics for multi-server batch operations – A cost based approach. *Int J Prod Res*, 35(11), 3063–3093.