Otimização baseada em Simulação para balanceamento de *Flow Shop* híbrido: Implementação e aplicação em R

Ícaro Romolo Sousa Agostino (UFSC)
icaroagostino@gmail.com

Pablo Medeiros Penna (IFSC)
pablomedeirospenna@gmail.com

Satie Ledoux Takeda-Berger (UFSC)
satietakeda@hotmail.com

Enzo Morosini Frazzon (UFSC)
enzo.frazzon@ufsc.br

Otimização baseada em simulação (SBO) é uma abordagem quantitativa que combina meta-heurísticas para solução de problemas de otimização com modelos de simulação para avaliar possíveis soluções iterativamente, sendo capaz de combinar vantagens das abordagens de otimização em conjunto com a capacidade estocástica da simulação. Nesse estudo é implementado uma abordagem SBO utilizando o modelo de otimização de Algoritmo Genético em conjunto com a Simulação de Eventos Discretos. O modelo foi sistematizado e implementado em linguagem R, que é uma linguagem de programação de código aberto para a comunidade, em conjunto com os pacotes 'Simmer' e 'GA'. Para avaliação da abordagem foi realizado uma aplicação para balanceamento de flow shop híbrido com o objetivo de maximizar a utilização dos equipamentos, utilizando tempos de produção e demanda de forma estocástica. O modelo apresentou resultados promissores, sendo capaz de encontrar soluções ótimas com boa performance computacional dentro das restrições definidas, se mostrando adequado para aplicações futuras em ambientes de produção e logística mais complexos.

Palavras-chave: Otimização baseada em Simulação, Flow Shop híbrido, Simulação de Eventos Discretos, Algoritmo Genético, linguagem R.

1. Introdução

O balanceamento de linha é uma tarefa essencial para as empresas de manufatura, uma vez que auxilia no aumento da produtividade e consequentemente minimiza os custos de produção (SIME; JANA; PANGHAL, 2019). No balanceamento de linha ocorre a alocação e sequenciamento das atividades nas estações de trabalho, de modo a alcançar a melhor utilização da mão-de-obra e equipamentos, minimizando assim o tempo ocioso. Além disso, proporciona a equalização da carga de trabalho entre as estações de trabalho, evitando a sobrecarga ou a subcarga (BAGSHAW, 2020).

Em um ambiente de manufatura de *flow shop* híbrido, torna-se essencial o balanceamento de linha. Diferentemente dos sistemas de manufatura em *flow shop* simples, o qual possuem a programação numa única linha de produção utilizando apenas uma máquina para cada etapa, o *flow shop* híbrido possui linhas de produção paralelas, ou seja, pelo menos uma etapa do processo produtivo tem duas ou mais máquinas paralelas idênticas (PAN; GAO; LI; GAO, 2017; PRATIWI; KUSBUDIONO; RISKI; HADI, 2020). Assim, as pesquisas que abordam problemas no *flow shop* híbrido, geralmente buscam distribuir os postos de trabalho entre as máquinas paralelas da forma mais equitativa possível de modo equilibrar a carga de trabalho (ZHAN; QIU; XUE, 2009).

Diferentes métodos analíticos e heurísticos/metaheurísticos são apresentados na literatura para resolver problemas de balanceamento de linha (BABAZADEH; JAVADIAN, 2019; REN; YU; ZHANG; TIAN *et al.*, 2017; YE; LI; NAULT, 2019). No entanto, a simulação computacional é uma técnica que tem sido amplamente utilizada em pesquisas, pois permite gerir a natureza estocástica das variáveis do sistema (SIME; JANA; PANGHAL, 2019). Adicionalmente, a otimização baseada em simulação (SBO, *simulation-based optmization*) combina os pontos fortes de ambos, simulação e otimização, tornando-se uma ferramenta poderosa para resolver problemas estocásticos complexos, inserindo um modelo de simulação na função objetivo da otimização (KÜCK; EHM; HILDEBRANDT; FREITAG *et al.*, 2016) .

Nesse contexto, o presente artigo tem por objetivo propor uma abordagem de otimização baseada em simulação para o balanceamento de linha produção de um sistema de *flow shop* híbrido. Na abordagem proposta realiza-se a integração do modelo de otimização utilizando o algoritmo genético (GA, *genetic algorithm*) e a simulação de eventos discretos (DES, *discrete event simulation*). Os modelos de simulação e otimização, bem como sua integração foram desenvolvidos utilizando a linguagem R, que é uma linguagem de programação de código

aberto para a comunidade. Para melhor entendimento, o artigo está estruturado da seguinte forma: na Seção 2, apresenta-se a revisão da literatura; na Seção 3, descreve-se a metodologia proposta para a abordagem; na Seção 4, apresenta-se os resultados da aplicação do SBO, e finalmente, na Seção 5, são expostas as considerações finais da pesquisa.

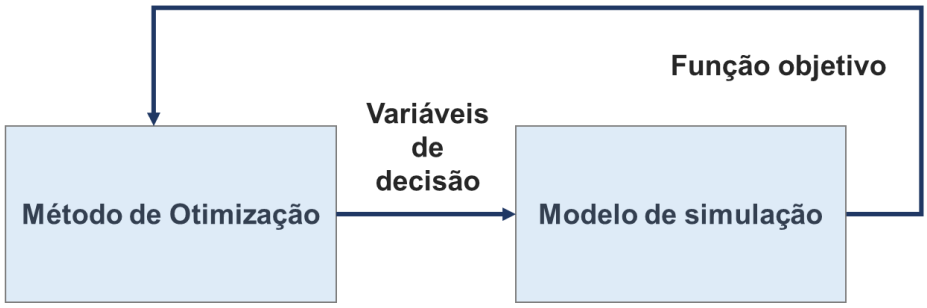
2. Revisão de Literatura

2.1 Otimização baseada em Simulação

O método de modelagem e simulação é uma referência para a análise de sistemas complexos. Por meio da modelagem é possível criar uma representação simplificada de um sistema em estudo e, através da simulação, pode-se realizar experimentos deste sistema, orientado por um conjunto de objetivos, foco do estudo (YEMANE; GEBREMICHEAL; HAILEMICHEAL; MERAHA, 2020). No entanto, a simulação não pode garantir a otimização desses sistemas perante os indicadores de desempenho analisados (FRAZZON; ALBRECHT; HURTADO; DE SOUZA SILVA *et al.*, 2015). Nesse sentido, os métodos de otimização podem ser utilizados principalmente se um sistema complexo for modelado de uma forma simplista (KÜCK; EHM; HILDEBRANDT; FREITAG *et al.*, 2016). Entretanto, os métodos de otimização também possuem limitações, como por exemplo, a avaliação do impacto de incertezas, uma vez que se torna demasiadamente complicado ou exige elevados recursos computacionais (FRAZZON; ALBRECHT; HURTADO, 2016).

Assim, Kück, Ehm, Hildebrandt, Freitag *et al.* (2016) argumentam que uma abordagem promissora com o objetivo de combinar os pontos fortes de ambos simulação e otimização, é a otimização baseada em simulação (SBO). Essa abordagem é adequada para resolver problemas em ambientes dinâmicos. A Figura 1 ilustra o funcionamento genérico da abordagem.

Figura 1 - Otimização baseada em simulação



Fonte: Adaptado de Alrabghi e Tiwari (2015)

Na abordagem SBO o método de otimização estocástica estima o valor da função objetivo por meio da simulação (FRAZZON; KÜCK; FREITAG, 2018). O método computa soluções iterativamente possíveis através de uma meta-heurística, avaliando-as com base em critérios de qualidade relevantes num modelo de simulação do sistema de produção (ou logística) e utiliza esta informação para calcular melhores soluções na próxima fase de iteração. Nesse estudo, o modelo de simulação adotado será de eventos discretos, e como método de otimização será utilizado o algoritmo genético.

2.2 Simulação de Eventos Discretos

A simulação de eventos discretos é um modelo de simulação com foco em eventos, tendo em consideração como o estado do sistema muda ao longo do tempo (CHWIF; MEDINA, 2014). Neste contexto, os sistemas de produção apresentam um amplo campo de aplicação dos modelos de simulação, a partir de alterações de estados em sistemas operacionais, tais como a expansão devido a troca ou adição de equipamentos que afetam o processo dinamicamente, a antecipação de gargalos, o planejamento de novos setores de produção para a obtenção de melhores alternativas, dentre outras. A simulação de eventos discretos se baseia em representar a realidade, incluindo eventos inesperados, hipóteses, possibilidades e incertezas, oferecendo a alternativa de "testar" soluções por meio de modelos computacionais (AGOSTINO; SOUSA; FROTA; DAHER *et al.*, 2019).

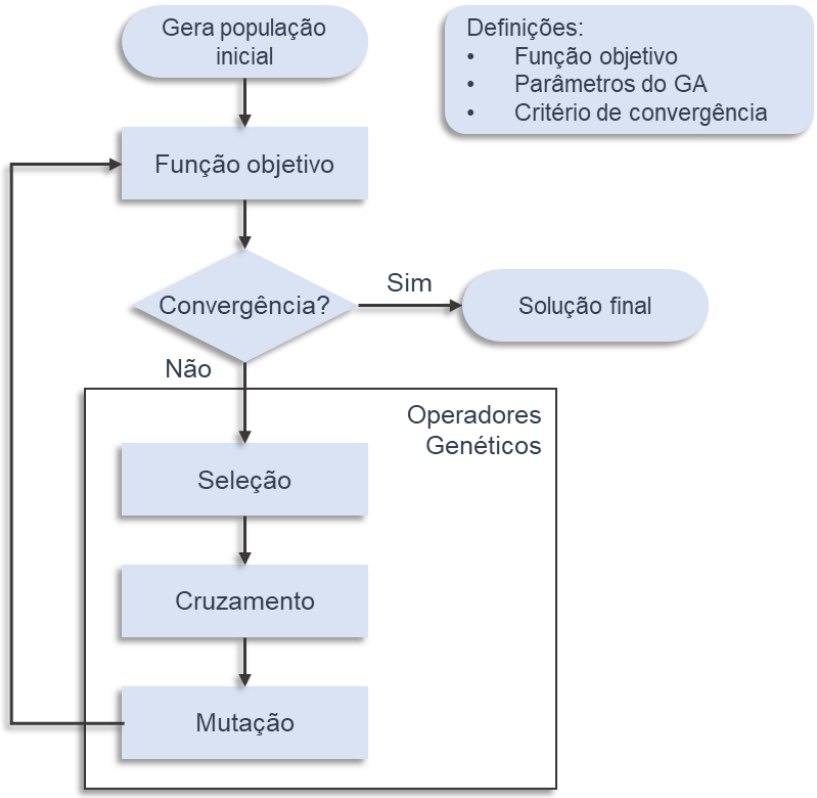
2.3 Algoritmo Genético

O Algoritmo Genético (GA) é uma heurística de otimização estocástica inspirada nos princípios básicos da evolução biológica e da seleção natural (SCRUCCA, 2013). As variáveis de decisão são combinadas na forma de indivíduos representando potenciais soluções para um problema de otimização. A estratégia de otimização adotada pelo GA gera populações de soluções de forma iterativa, evoluindo as soluções possíveis por meio de seleção, avaliando cada indivíduo e selecionando os mais aptos a se reproduzirem. Assim, com o operador de seleção, os GA's imitam o comportamento dos organismos naturais num ambiente competitivo, no qual apenas os mais qualificados e a sua descendência sobrevivem (SIVANANDAM; DEEPA, 2007).

A adequação de cada indivíduo é avaliada por uma função objetivo e apenas os indivíduos mais aptos se reproduzem, passando a sua informação genética aos seus descendentes por meios dos operadores de mutação e do cruzamento. O cruzamento forma novos resultados de duas soluções "progenitoras" através da combinação de parte da informação genética de cada um.

A mutação é um operador genético que altera, de forma geral e aleatória, os valores dos genes de uma solução, permitindo escapar de ótimos locais (SCRUCCA, 2013). A Figura 2 ilustra o funcionamento do GA.

Figura 2 - Fluxo do algoritmo genético



Fonte: Adaptado de Scrucca (2013)

O processo de evolução termina com base em alguns critérios de convergência. Alternativamente, o GA é interrompido quando um número suficientemente grande de gerações passou sem qualquer melhoria no valor da função objetivo, ou quando uma estatística populacional atinge um limite pré-definido (HAUPT; HAUPT, 2004).

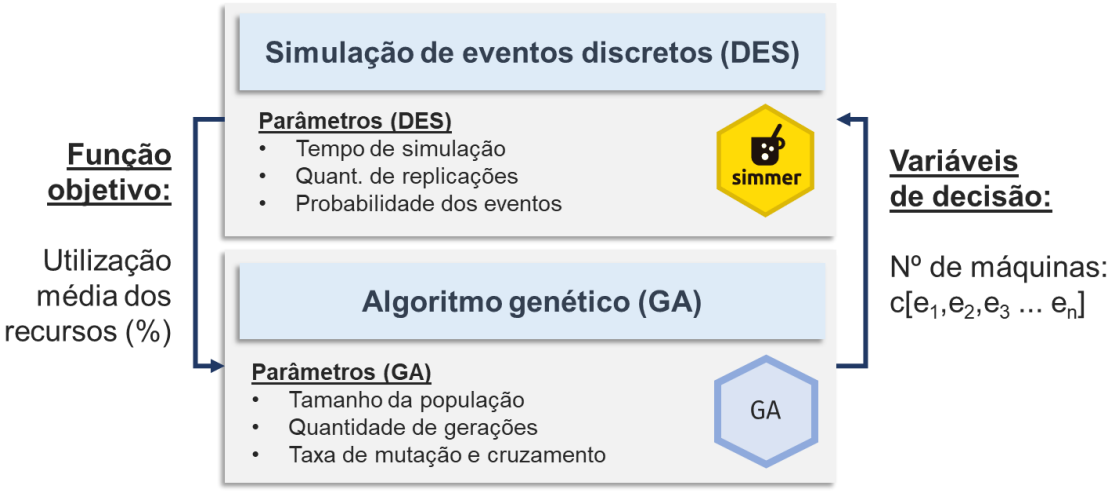
3. Metodologia da pesquisa

3.1 Abordagem proposta

A abordagem proposta nessa pesquisa objetivou sistematizar a aplicação da otimização baseada em simulação para balanceamento de *flow shop* híbrido, propondo a integração do modelo de otimização utilizando o algoritmo genético e a simulação de eventos discretos para a resolução

de problemas práticos de gestão da produção. A abordagem metodológica proposta é ilustrada na Figura 3.

Figura 3 - Otimização baseada em simulação para balanceamento de linha de produção



Fonte: Autores (2020)

A integração entre o modelo de otimização e a simulação foi realizada de forma que cada solução única gerada pelo algoritmo genético fosse simulada *n* vezes pelo modelo de simulação de eventos discretos. As variáveis de decisão representam um vetor com o número de máquinas paralelas e idênticas em cada etapa de produção. A função objetivo é o próprio modelo de simulação que fornece ao fim de *n* rodadas de simulação o valor de um indicador chave de desempenho (KPI, *key performance indicator*). Para o experimento proposto, o KPI considerado foi a utilização média dos equipamentos, conforme descrito na Equação 1.

$$Utilização\ média = \frac{Total\ de\ horas\ em\ operação}{Total\ de\ horas\ disponíveis} \quad (1)$$

Assim, de forma iterativa, o método computa soluções possíveis através do algoritmo genético, sendo que a quantidade de solução é determinada pelo tamanho da população de soluções e da quantidade de gerações. As soluções são avaliadas com base em critérios de qualidade, relevantes num modelo de simulação do sistema de produção. Estas informações são utilizadas para calcular as melhores soluções na próxima fase de iteração, aplicando a heurística evolutiva baseada em mutação e cruzamento.

A implementação da abordagem de otimização baseada em simulação foi realizada em linguagem R (R CORE TEAM, 2020), utilizando os pacotes 'GA' (SCRUCCA, 2013) para a otimização evolutiva com algoritmos genéticos e 'Simmer' (UCAR; SMEETS; AZCORRA, 2019) para a simulação de eventos discretos. Todas as ferramentas computacionais utilizadas são de código aberto e disponíveis para reprodução dos experimentos. O código completo da implementação está disponível em: https://github.com/icaroagostino/SBO.

3.2 Descrição do experimento

Para a avaliação da abordagem proposta, o experimento foi realizado com os seguintes parâmetros definidos para o algoritmo genético: tamanho da população de soluções igual a 30; quantidade máxima de gerações igual a 500; regra de parada igual a 10 (nesse caso após 10 gerações sem melhoria na solução o algoritmo para); taxa de cruzamento igual a 80% e taxa de mutação igual a 10%.

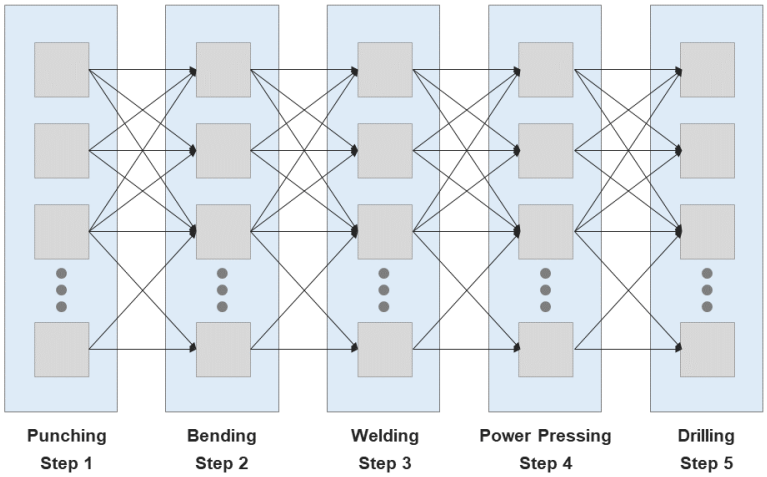
Cada solução compreende um vetor contendo o número de máquinas paralelas e idênticas a serem avaliadas na simulação. Para reduzir o espaço de soluções possíveis, foi definido o valor mínimo de uma máquina para cada etapa e cinco máquinas como quantidade máxima. Por fim, na simulação de eventos discretos foi definido um número de replicações de 30, onde cada solução possível será simulada 30 vezes para lidar com a variação dos resultados gerados pela estocasticidade dos processos e eventos considerados (HOAD; ROBINSON; DAVIES, 2010).

4. Resultados

4.1 Criando um ambiente de simulação com 'Simmer'

Para ilustrar a abordagem proposta, foi implementado como caso teste o cenário relatado no estudo de Marichelvam, Prabaharan e Yang (2014), o qual considerou um sistema de *flow shop* híbrido para a produção de peças metálicas. O processo de produção possui cinco etapas: *Punching*, *Bending*, *Welding*, *Power Pressing* e *Drilling*. Para efeito de simulação, utilizou-se o período de produção de uma semana (7 dias), considerando dois turnos de trabalho de 8 horas por dia (6.720 segundos) e os tempos de execução de cada processo foram definidos de forma empírica baseada no estudo consultado. A Figura 4 ilustra a sequência dos processos de fabricação sem um numero especifico de máquinas definidas.

Figura 4 - Representação do processo de *flow shop* híbrido



Fonte: Autores (2020)

Na definição da trajetória, cada processo produtivo é definido através das funções *Seize*, *Timeout* e *Release*, que representam a entrada, tempo de duração e saída, respectivamente. Para adicionar estocasticidade ao processo, empregou-se distribuição normal nos tempos de processamento, adotando se 10, 20, 15, 12 e 6 como média para cada processo respectivamente com desvio padrão de uma unidade ao *Timeout*, e distribuição exponencial para a chegada das entidades no sistema, representados pela geração da trajetória. A seguir é apresentado a descrição do código em R para a definição da trajetória.

```
# Definindo a trajetoria
  flowShop <- trajectory("flowShop") %>%
    ## add a Punching activity
    seize("Punching", 1) %>%
    timeout(function() rnorm(1, 10)) %>%
    release("Punching", 1) %>%
    ## add a Bending activity
    seize("Bending", 1) %>%
    timeout(function() rnorm(1, 20)) %>%
    release("Bending", 1) %>%
    ## add a Welding activity
    seize("Welding", 1) %>%
    timeout(function() rnorm(1, 15)) %>%
    release("Welding", 1) %>%
    ## add a Pressing activity
    seize("Pressing", 1) %>%
    timeout(function() rnorm(1, 12)) %>%
    release("Pressing", 1) %>%
    ## add a Drilling activity
    seize("Drilling", 1) %>%
    timeout(function() rnorm(1, 6)) %>%
    release("Drilling", 1)
```

A função **add_resource()** é responsável por definir ao sistema quais são os recursos do processo, para fins deste estudo considerou-se as quantidade de máquinas em cada etapa de

fabricação. Já a função **add_generator()**, tem o objetivo de executar a geração de novos trabalhos ou *Jobs*, representados aqui como sendo uma ordem de produção a serem executadas, com intervalo de tempo definido exponencialmente utilizando a função **rexp().**

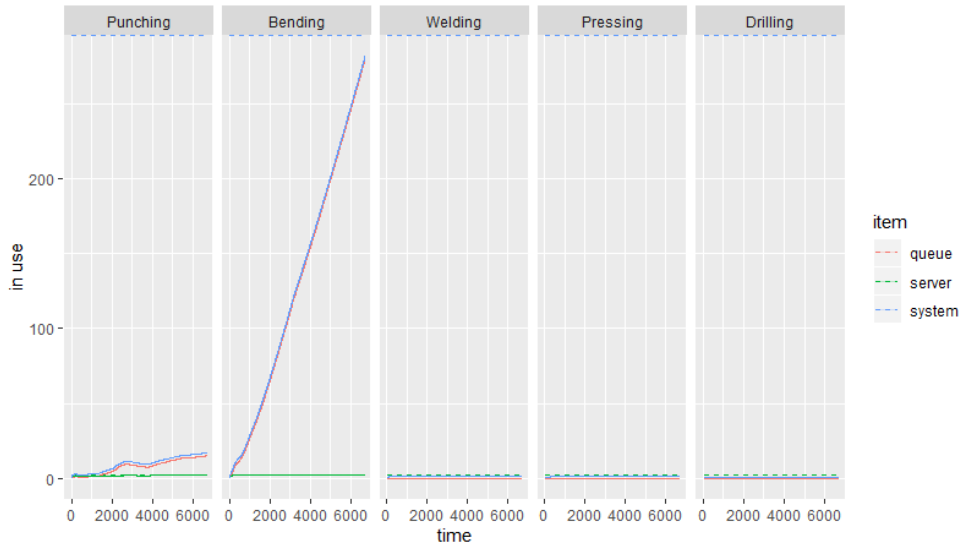
```
# Adicionando recursos
  env %>%
    add_resource("Punching", 2) %>%
    add_resource("Bending",  2) %>%
    add_resource("Welding",  2) %>%
    add_resource("Pressing", 2) %>%
    add_resource("Drilling", 2) %>%
    add_generator("flowShop", flowShop, function() rexp(1, 1/5))
```

Após a criação do modelo de simulação é possível testar soluções para análises. Como solução inicial, duas máquinas paralelas foram adicionadas em cada etapa, como pode ser vista na função **add_resource()**. Além disso, foi adicionado uma demanda estocástica com intervalo médio de 5 segundos para cada ordem de produção (representado um produto único a ser processado). Para obter a análise média dos tempos, a simulação foi replicada 30 vezes utilizando a função **lapply().**

```
envs <<- lapply(1:30, function(i) {
    reset(env) ; run(env, 960*7) # Tempo de simulação
  })
```

A utilização média dos recursos pode ser calculada utilizando o pacote 'Simmer.plot' que é uma extensão do 'Simmer' para plotagem de estatísticas e trajetórias.

Figura 5 - Resultado da simulação com solução ingênua



Fonte: Autores (2020)

Conforme apresentado na Figura 5, a solução inicial arbitrária com 2 máquinas paralelas em cada etapa não demonstrou ser adequada para a demanda atual. Esta configuração formou fila excessiva no processo *Bending* criando uma sobrecarga no sistema maior que 200 máquinas e subutilizou os demais processos. A utilização média dos recursos para esse cenário foi de 72,71%. Na seção seguinte é apresentado a implementação do GA para maximizar o resultado de utilização.

4.2 Otimização com algoritmo genético com o pacote 'GA'

O pacote 'GA' implementado em R fornece uma coleção de funções de propósito geral para a otimização usando algoritmos genéticos. A otimização utilizando GA pode ser executada sequencialmente ou em paralelo para obter um maior desempenho. A fim de solucionar o problema de balanceamento da linha de produção, utilizou-se o SBO integrando o pacote do 'GA' ao ambiente da simulação criada.

Por tratar-se da mesma plataforma responsável por todo o desenvolvimento da abordagem proposta, tem-se a vantagem de acessar os dados de utilização de recursos, sem maiores tratativas. Os parâmetros configurados para o GA foram utilizados conforme definidos na subseção 3.2.

```
popSize    <- 30     # Tamanho da população
maxiter    <- 500    # Max de iteração
run        <- 10     # Número de iterações iguais que para a otimização
pcrossover <- 0.8    # Cruzamento
pmutation  <- 0.1    # Mutação
```

A integração entre a otimização e a simulação acontece da seguinte forma: cada conjunto de soluções é testado em um ambiente de simulação que representa um estado possível do cenário real e retorna à utilização média dos recursos como KPI a ser maximizado.

Pelo tempo correspondente a uma semana de uma fábrica ativa em dois turnos, são gerados os novos *jobs*, com intervalo entre eles reproduzidos estocasticamente pela da distribuição exponencial. O conjunto de *jobs* para esse período é simulado 30 vezes, calcula-se a média do tempo e utilização de cada processo. Após atingir o tamanho da população, a otimização recombina 80% dos *jobs* buscando uma solução melhor, os 10% melhores resultados são armazenados. O novo número de máquinas é estipulado e todo o ciclo ocorre novamente, até

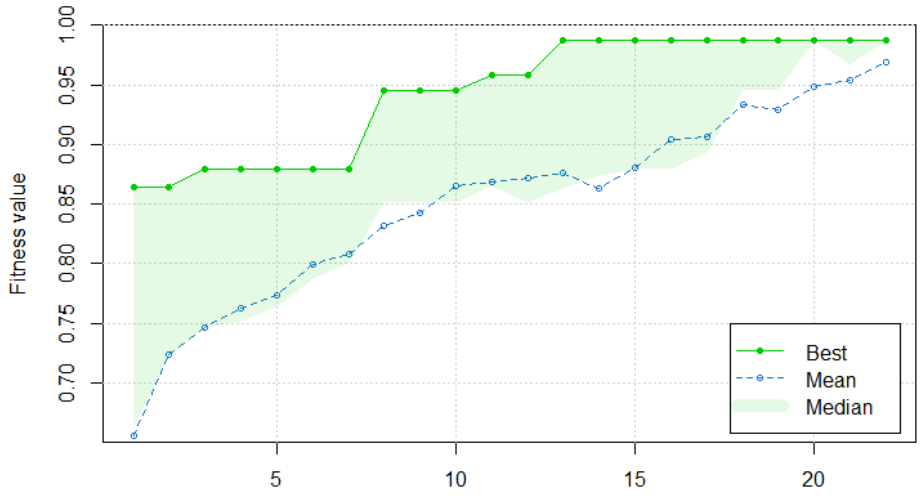
que se obtenha 10 gerações sem melhorias ou 500 gerações. Ao fim deste processo a melhor solução é encontrada.

Para a otimização, decidiu-se utilizar a capacidade de paralelização da implementação do algoritmo genético disponível no pacote 'GA' (SCRUCCA, 2017). O paralelismo distribui o processamento entre todos os núcleos do processador do computador, reduzindo o tempo de execução. Em termos práticos, cada solução do algoritmo genético será avaliada paralelamente em cada núcleo, gerando simulações de diferentes cenários possíveis rodando em diferentes núcleos simultaneamente. O paralelismo pode ser utilizado para o aumento da eficiência computacional, principalmente em projetos que apresentam longa duração e complexidade de simulação e otimização.

```
GA <- ga(type = "real-valued",
         fitness = function(x) simular(x[1],x[2],x[3],x[4],x[5]),
         lower = lower,
         upper = upper,
         popSize = popSize,
         maxiter = maxiter,
         run = run,
         pcrossover = pcrossover,
         pmutation = pmutation,
         parallel = TRUE)
```

A Figura 6 apresenta o resultado da otimização ao longo das gerações. Pode-se observar a evolução das soluções em direção a maximização da utilização média dos recursos. A partir de uma solução aleatória, o algoritmo, conforme os parâmetros estabelecidos, começa a encontrar melhores soluções e interrompe o processo após identificar 10 iterações iguais (13 até 22).

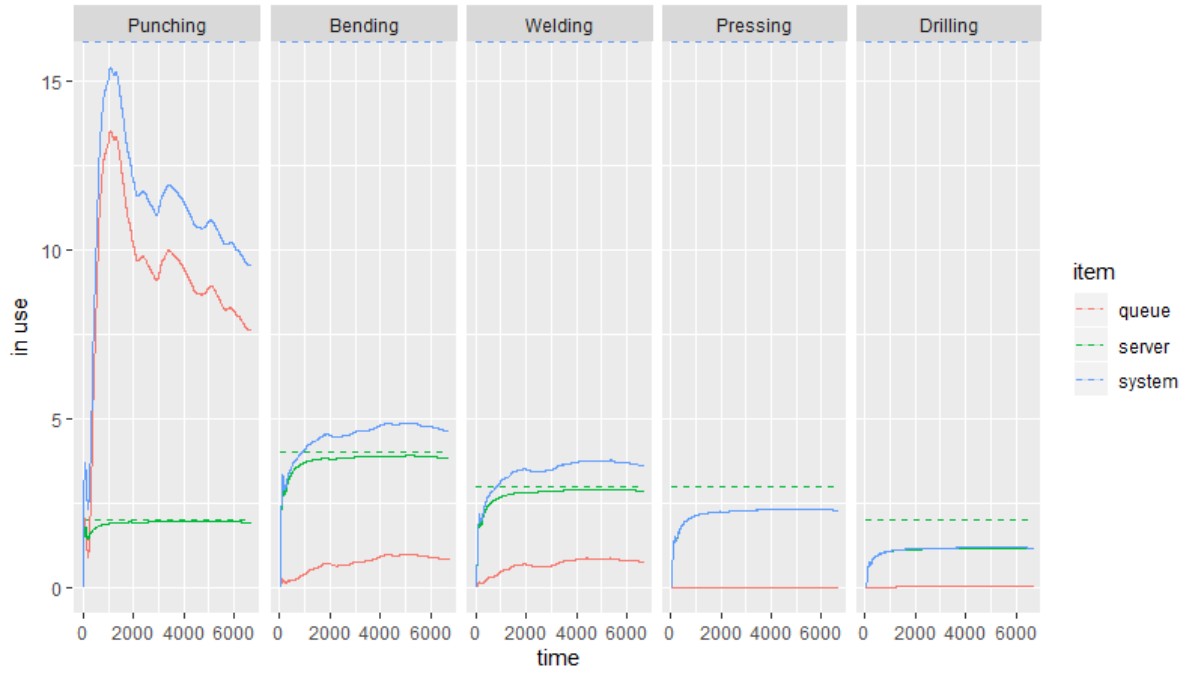
Figura 6 - Evolução das soluções com algoritmo genético



Fonte: Autores (2020)

Com a implementação da abordagem proposta, utilizando o SBO, foi possível alcançar uma utilização média dos recursos de 98,30%. A Figura 7 ilustra o balanceamento dos processos com base no nível de utilização dos mesmos. As distorções demonstradas na etapa de *Punching* são justificáveis, visto que se trata da primeira etapa do processo produtivo com o gerador de entrada exponencial. Em relação a solução ingênua anterior, é possível perceber uma significativa maximização do KPI analisado, com aumento de 25,59%.

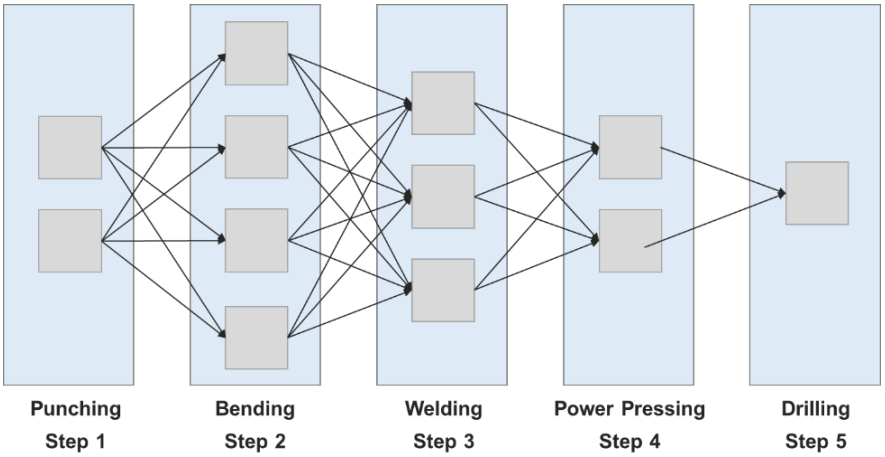
Figura 7 - Resultado da simulação com solução otimizada



Fonte: Autores (2020)

Com relação a demanda considerada, o sistema produtivo encontrou como números ideais de máquinas para o balanceamento do *flow shop* híbrido os seguintes valores: duas para o processo de *Punching*, quatro para o *Bending*, três para o processo de *Welding*, duas para o *Power Pressing* e o processo de *Drilling* permaneceu com apenas uma máquina. A Figura 8 ilustra a solução final.

Figura 8 - *Flow shop* híbrido balanceado utilizando a abordagem proposta de SBO



Fonte: Autores (2020)

O tempo computacional despendido para a otimização através do método SBO implementado na linguagem em R com as bibliotecas 'Simmer' e 'GA' foi de aproximadamente 10,06 minutos, sem o recurso de paralelização. Considerando a paralelização para a avaliação das soluções por meio da simulação, o tempo computacional gasto foi de 2,62 minutos, reduzindo cerca de 74% do tempo total. Os experimentos foram realizados utilizando um computador com processador 6-core Intel i7-9750HF.

5. Conclusões e trabalhos futuros

Esse estudo apresentou uma aplicação da abordagem de otimização baseada em simulação para o balanceamento de linhas de produção em um sistema de *flow shop* híbrido, adotando a combinação de simulação de eventos discretos e otimização com algoritmo genético. A abordagem proposta foi implementada em linguagem R, sendo totalmente em código aberto e livre para reprodução.

A abordagem implementada foi capaz de lidar com um problema complexo de balanceamento de linhas de produção, considerando características estocásticas e dinâmicas, avaliando as soluções em um ambiente que emula a realidade e representa possíveis estados do sistema. Adicionalmente, o método apresentou bom desempenho computacional, convergindo uma solução final em um tempo computacional viável para implementações reais com potencial de escalabilidade.

Como trabalhos futuros, pretende-se ampliar a aplicação do método de forma dinâmica, sendo possível periodicamente reajustar o sistema produtivo frente as variações de demandas e de

estados operacionais do sistema. Além disso, estudos futuros poderão incluir outras variáveis de produção associadas a capacidade produtiva, assim como avaliar diferentes KPIs de produção e logística.

6. Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

REFERÊNCIAS

AGOSTINO, I.; SOUSA, S.; FROTA, P.; DAHER, R. *et al.* Modeling and Simulation of Operations: A Case Study in a Port Terminal of Vale S/A. *In*: **New Global Perspectives on Industrial Engineering and Management**: Springer, 2019. p. 91-99.

ALRABGHI, A.; TIWARI, A. State of the art in simulation-based optimisation for maintenance systems. **Computers & Industrial Engineering**, 82, p. 167-182, 2015.

BABAZADEH, H.; JAVADIAN, N. A novel meta-heuristic approach to solve fuzzy multi-objective straight and U-shaped assembly line balancing problems. **Soft Computing**, 23, n. 17, p. 8217-8245, 2019.

BAGSHAW, K. B. Work Line Balancing and Production Efficiency of Manufacturing Firms in Rivers State, Nigeria. **American Journal of Industrial and Business Management**, 10, n. 01, p. 45, 2020.

CHWIF, L.; MEDINA, A. **Modelagem e simulação de eventos discretos: Teoria e aplicações**. 4a ed. Elsevier Brasil, 2014.

FRAZZON, E. M.; ALBRECHT, A.; HURTADO, P. A. Simulation-based optimization for the integrated scheduling of production and logistic systems. **IFAC-PapersOnLine**, 49, n. 12, p. 1050-1055, 2016.

FRAZZON, E. M.; ALBRECHT, A.; HURTADO, P. A.; DE SOUZA SILVA, L. *et al.* Hybrid modelling approach for the scheduling and control of integrated production and logistic processes along export supply chains. **IFAC-PapersOnLine**, 48, n. 3, p. 1521-1526, 2015.

FRAZZON, E. M.; KÜCK, M.; FREITAG, M. Data-driven production control for complex and dynamic manufacturing systems. **CIRP Annals**, 67, n. 1, p. 515-518, 2018.

HAUPT, R. L.; HAUPT, S. E. **Practical genetic algorithms**. 2nd ed. John Wiley & Sons, 2004.

HOAD, K.; ROBINSON, S.; DAVIES, R. Automated selection of the number of replications for a discrete-event simulation. **Journal of the Operational Research Society**, 61, n. 11, p. 1632-1644, 2010.

KÜCK, M.; EHM, J.; HILDEBRANDT, T.; FREITAG, M. *et al.*, 2016, **Potential of data-driven simulation-based optimization for adaptive scheduling and control of dynamic manufacturing systems**. IEEE. 2820-2831.

MARICHELVAM, M. K.; PRABAHARAN, T.; YANG, X.-S. Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan. **Applied Soft Computing**, 19, p. 93-101, 2014.

PAN, Q.-K.; GAO, L.; LI, X.-Y.; GAO, K.-Z. Effective metaheuristics for scheduling a hybrid flowshop with sequence-dependent setup times. **Applied Mathematics and Computation**, 303, p. 89-112, 2017.

PRATIWI, Y. E.; KUSBUDIONO; RISKI, A.; HADI, A. F. Hybrid flow-shop scheduling (HFS) problem solving with migrating birds optimization (mbo) algorithm. **Mathematics and Statistics**, 8, n. 2, p. 58-62, 2020. Article.

R CORE TEAM. **R: A language and environment for statistical computing**. Vienna, Austria, 2020. Disponível em: https://www.R-project.org/. Acesso em: 27 abril 2020.

REN, Y.; YU, D.; ZHANG, C.; TIAN, G. *et al.* An improved gravitational search algorithm for profit-oriented partial disassembly line balancing problem. **International Journal of Production Research**, 55, n. 24, p. 7302-7316, 2017.

SCRUCCA, L. GA: A Package for Genetic Algorithms in R. Journal of Statistical Software. **Foundation for Open Access Statistics**, 53, n. 4, 2013.

SCRUCCA, L. On some extensions to GA package: hybrid optimisation, parallelisation and islands evolution. **The R Journal**, 1, n. 9, p. 187-206, 2017.

SIME, H.; JANA, P.; PANGHAL, D. Feasibility of using simulation technique for line balancing in apparel industry. **Procedia Manufacturing**, 30, p. 300-307, 2019.

SIVANANDAM, S. N.; DEEPA, S. N. **Introduction to genetic algorithms**. Springer Science & Business Media, 2007.

UCAR, I.; SMEETS, B.; AZCORRA, A. Simmer: Discrete-event simulation for R. **Journal of Statistical Software**, 90, n. 2, p. 1-30, 2019.

YE, H.; LI, W.; NAULT, B. R. Trade-off balancing between maximum and total completion times for no-wait flow shop production. **International Journal of Production Research**, p. 1-17, 2019.

YEMANE, A.; GEBREMICHEAL, G.; HAILEMICHEAL, M.; MERAHA, T. Productivity Improvement through Line Balancing by Using Simulation Modeling. **Journal of Optimization in Industrial Engineering**, 13, n. 1, p. 153-165, 2020.

ZHAN, Y.; QIU, C. H.; XUE, K., 2009, **A hybrid genetic algorithm for hybrid flow shop scheduling with load balancing**. Trans Tech Publ. 250-255.