PROPOSTA DE OTIMIZAÇÃO ROBUSTA PARA O PROBLEMA DE AGENDAMENTO JOB SHOP COM O USO DE SIMULAÇÃO E ALGORITMO GENÉTICO

Marilda Fatima de Souza da Silva

marildafssilva@gmail.com

Luiz Lycurgo Leite Neto

lneto@uninove.edu.br

Leila Aparecida Martins

leilamartinscarreiro@gmail.com

Fabio Henrique Pereira

fabiohp@uni9.pro.br

Universidade Nove de Julho, Programa de Pós-graduação em Informática e Gestão do Conhecimento, Rua Vergueiro, 235/249 – 12º Andar Liberdade – São Paulo/SP - CEP 01504-001, 55 (0xx11) 3385-9078

Resumo. O problema de agendamento da produção tem sido por mais de meio século, objeto de estudo de vários pesquisadores. Entretanto, a maior parte dos documentos aborda o problema determinístico e/ou estático, que não considera os tempos de processamento estocásticos e/ou a chegada dinâmica de ordens, que representam mais apropriadamente as situações da indústria do mundo real. Neste contexto, realizou-se a integração de um ambiente dinâmico de produção desenvolvido e acoplado num modelo de simulação-otimização. Este artigo aborda a otimização robusta para este problema. Essa robustez é caracterizada pela capacidade de permanecer estável mesmo na ocorrência de variações da demanda. Objetivou-se determinar a sequência de ordens fornecida pelo Algoritmo Genético e validar a sua capacidade de permanecer estável quando submetido a essas variações. Analisou-se os efeitos estocásticos dessa variação em diferentes regras de despacho e parâmetros de otimização. O modelo desenvolvido mostra que a integração proposta permite avaliar diferentes soluções de agendamento para aumentar a produtividade do ambiente de produção. A abordagem proposta é avaliada em relação ao makespan médio, atraso total e número de trabalhos atrasados. Os resultados indicam que a abordagem funciona bem e é robusta em relação à demanda variável estocástica e ao ambiente de produção dinâmico.

Palavras-chave: *Simulação, Algoritmo Genético, Job Shop, Agendamento*

Proceedings of the XXXVIII Iberian Latin-American Congress on Computational Methods in Engineering P.O. Faria, R.H. Lopez, L.F.F. Miguel, W.J.S. Gomes, M. Noronha (Editores), ABMEC, Florianópolis, SC, Brazil, November 5-8, 2017.

INTRODUCÃO

As empresas trabalham num ambiente globalizado e altamente competitivo, portanto aquela que melhor atender os requisitos dos clientes em termos de preço, qualidade e prazo estão mais propensas a permanecerem vivas. Dentre os diversos problemas que estas empresas encontram, um deles tem se destacado em função da sua dificuldade e é conhecido por problema de sequenciamento ou agendamento de produção ou no inglês, *Job Shop Scheduling Problem* (JSSP).

De acordo com Lukaszewicz (2005), o sequenciamento de produção pode ser entendido como um método de atribuir um ou mais recursos para a realização de determinadas tarefas que são executadas em uma certa quantidade de tempo. Neste ambiente de fabricação, os recursos são representados como máquinas e uma tarefa ou ordem é definida como um conjunto de uma ou mais atividades ou operações.

Tendo por objetivo a minimização de uma ou mais medidas de desempenho, o JSSP pode ser formulado matematicamente como: dado um conjunto S de variáveis discretas, chamadas soluções, e uma função objetiva f: S $\rightarrow$ R que associe um valor real a cada solução s em S, encontre uma solução s* tal f (s*) $\leq$ f (s), para todos os s em S.

Devido à sua importância prática e complexidade computacional difícil, o JSSP tem sido amplamente estudado. Durante mais de meio século, foram desenvolvidas diferentes técnicas de otimização para resolver esses problemas de agendamento.

Inicialmente, o JSSP foi tratado por métodos exatos, como o *Branch and Bound* (Jain e Meeran, 1999) e o deslocamento do gargalo (Adams, Balas e Zawack, 1988). Nos últimos anos, no entanto, os métodos metaheurísticos têm sido amplamente utilizados na solução deste tipo de problema. Entre esses métodos, aqueles com maior ênfase para a solução do JSSP são *Tabu Search* (Nowicki e Smutnicki, 1996), Recozimento Simulado (Aydin e Fogarty 2004; Zandieh, Khatami, e Rahmati, 2017), Algoritmos Genéticos (Gonçalves, Mendes e Resende, 2005) e Otimização da Colônia de Formigas (Huang e Liao, 2008).

Entretanto, a maior parte dos documentos aborda o problema determinístico e/ou estático, que não considera os tempos de processamento estocásticos e/ou a chegada dinâmica de ordens que representam mais apropriadamente as situações da indústria do mundo real. Desta forma, é possível ver uma grande lacuna entre a teoria e a prática de agendamento de trabalhos (Ouelhadj e Petrifica, 2009).

Recentemente, alguns estudos buscaram preencher esta lacuna. Zhang, Yi e Xiao (2013) apresentam uma política híbrida para lidar com a característica dinâmica do problema de agendamento. O método proposto foi comparado com algumas regras comuns de despacho e os resultados demonstraram a eficácia da abordagem em várias condições do chão de fábrica. Mas, além de apresentar um alto custo computacional, que pode não ser aceitável para sistemas de fabricação reais, o estudo não abrange outras combinações de fatores experimentais, como as variações do tempo de processamento.

O problema de agendamento estocástico também foi abordado por (Haddadzade, Razfar e Fazel Zarandi, 2014; Salch, Gayon e Lemaire, 2013; Lei, 2011; Tavakkoli-Moghaddamet. al, 2005). Nesses artigos, os tempos de processamento e as datas de vencimento são variáveis aleatórias, mas os problemas de agendamento são considerados em um ambiente estático.

Proceedings of the XXXVIII Iberian Latin-American Congress on Computational Methods in Engineering
P.O. Faria, R.H. Lopez, L.F.F. Miguel, W.J.S. Gomes, M. Noronha (Editores), ABMEC, Florianópolis, SC, Brazil, November 5-8, 2017.

Resumidamente, pode se dizer que nas últimas décadas, uma grande quantidade de pesquisa foi realizada em problemas de agendamento. Mas, poucos consideraram o agendamento estocástico e dinâmico. Os modelos de planejamento de produção que não reconhecem essa importância podem gerar decisões de planejamento inferiores em relação àquelas que explicitamente explicam a incerteza.

Por outro lado, a aplicação de tais técnicas de otimização requer uma representação do ambiente de produção real e suas restrições. Devido aos recursos de ambientes reais, como restrições estocásticas e fatores incertos, eles são muito complexos para serem modelados analiticamente.

Uma alternativa para superar essas dificuldades seria a integração da teoria da fila e das abordagens de agendamento dinâmico, uma vez que esses dois campos frequentemente lidam com as mesmas áreas de aplicação que, por exemplo, ambientes de fabricação. No entanto, poucos trabalhos sobre o agendamento dinâmico aproveitam os desenvolvimentos no campo da teoria da fila (Terekhov, et al, 2014). Na verdade, de acordo com esses autores, o primeiro desenvolvimento de uma abordagem híbrida de programação de enfileiramento foi proposto em 2012.

Assim, quando a abordagem está relacionada com restrições dinâmicas e estocásticas e funções objetivas, a avaliação de uma solução candidata deve ser realizada por meio da simulação. Nestes casos, a simulação de eventos discretos provou ser uma escolha adequada, pois pode enfrentar aleatoriedade e dinamização de forma eficiente.

No entanto, a partir de alguns trabalhos que abordam as características estocásticas e dinâmicas do problema, é possível ver que o desenvolvimento do modelo de simulação e, principalmente, a integração do modelo de simulação com o método de otimização não estão bem estabelecidos e nem sempre é uma tarefa simples (Terekhov, at al., 2014, Azadivar, 1992). Como, em geral, a integração de cada aplicativo específico usa sua própria representação do sistema e além dos diversos tipos de linguagens de simulação e programação, o que dificulta a troca de dados e pode exigir o desenvolvimento de interfaces dedicadas (Iassinovski, et al., 2003).

Além disso, as integrações de otimização e simulação às vezes são baseadas em métodos muito gerais a partir da literatura metaheurística determinista que, como já discutido, pode não ser adequado para problemas do mundo real (Fu, 2002). Como a simulação computacional é mais dispendiosa do que avaliar a função analítica, a eficiência dos algoritmos de otimização é crucial (Fu, 2002). Portanto, é importante uma abordagem de integração que permita o uso de qualquer método de otimização.

Este trabalho apresenta uma versão resumida da integração de um modelo de simulação com um método de otimização para resolver o problema dinâmico de agendamento do *Job Shop* com demanda estocástica e variável. Utilizando as instalações de controle dentro e entre aplicativos, através da tecnologia ActiveX Automation e do Visual Basic for Application (VBA), o modelo é integrado com um método de otimização. A integração do modelo proposto é realizada usando componentes fora do processo, nos quais o método de otimização é executado como um aplicativo independente. Como exemplo, o modelo é integrado ao Algoritmo Genético (AG) com um aplicativo C ++ e é usado como *fitness* para avaliar as soluções candidatas.

Proceedings of the XXXVIII Iberian Latin-American Congress on Computational Methods in Engineering P.O. Faria, R.H. Lopez, L.F.F. Miguel, W.J.S. Gomes, M. Noronha (Editores), ABMEC, Florianópolis, SC, Brazil, November 5-8, 2017.

Avaliou-se o resultado da função multiobjectivo em que considerou-se a mesma importância percentual das medidas de desempenho: *makespan* médio, número de ordens atrasadas e tempo total de atraso e comparado com algumas regras de lançamento de ordens no sistema como: Menor tempo de processamento (SPT-Shortest Process Time), Maior tempo de processamento (LPT-Longest Process Time), a data de vencimento mais próxima (EDD-Earlest Due Data), primeiro a chegar, primeiro a ser processado (FIFO- First-In-First-Out) e último a chegar, primeiro a ser processado (LIFO-Last-In-First-Out).

1 FUNDAMENTAÇÃO TEÓRICA

Neste item, será realizada uma breve apresentação de alguns conceitos sobre problemas de agendamento estocástico e dinâmico, do AG, modelagem e simulação, necessários a compreensão do artigo.

1.1 Problemas de agendamento estocástico e dinâmico

O problema clássico de agendamento *Job Shop* abordado na literatura apresenta as seguintes características: um conjunto de n ordens $J = \{J_1, J_2, J_3, ..., J_n\}$ que deve ser processado por m máquinas $M = \{M_1, M_2, M_3, ..., M_m\}$ de acordo com os processos $P = \{P_1, P_2, P_3 ... P_p\}$ e algumas dessas restrições, como: quando uma máquina possui um processo agendado ela, deve processá-lo do início ao fim. Isso significa que a ordem de operação de cada trabalho deve ser levada em consideração. O processamento da i_n operação do trabalho j (O_{ij}) deve ser realizado em uma máquina pré-atribuída $M_k \in M$. Os tempos de processamento podem ser fixos ou variáveis e a data de vencimento pode ser diferente (Baptiste, Flamini e Sourd, 2008).

O JSSP é considerado como sendo *np-hard* que, na teoria de complexidade computacional são problemas difíceis de resolver. Uma vez que em relação ao seu tamanho, o consumo de tempo para resolver essa classe de problemas não pode ser reduzido a uma função polinomial (Fan, Zhang, 2010).

O problema de agendamento dinâmico e estocástico do *Job Shop* é definido com base nos tempos de lançamento e nos tempos de processamento dos trabalhos: os tempos de liberação não são corrigidos em um único ponto e os tempos de lançamento e processamento são variáveis aleatórias descritas pelas distribuições de probabilidade. Em contraste com o problema de agendamento estático, em que todos os trabalhos estão prontos para começar no instante zero, no agendamento dinâmico, os trabalhos podem chegar em qualquer horário futuro, conhecido ou desconhecido. Além disso, no problema estocástico, os tempos de processamento são aleatórios. Assim, o problema de agendamento estocástico e dinâmico pode ser definido de acordo com as seguintes características (Zhang, Yi e Xiao, 2005):

a) O tempo entre as chegadas das ordens é considerado como uma variável aleatória, o que significa que os trabalhos chegam ao sistema dinamicamente;

b) Os tempos de processamento dos trabalhos variam de forma estocástica em cada máquina.

1.2 Algoritmo Genético

O embasamento teórico do AG foi apresentado por Holland (1975), com a ideia de imitar o processo evolutivo que ocorre com os organismos biológicos na natureza. O algoritmo baseia-se no processo de seleção natural, em que os indivíduos mais adaptados ao seu ambiente podem se reproduzir com mais frequência, passando seus traços genéticos para a próxima geração. Assim, a AG usa modelos computacionais do processo natural de evolução dos seres vivos como uma ferramenta para resolver problemas.

Embora existam vários modelos nesta área, todos eles compartilham os mesmos fundamentos baseados em conceitos de seleção, mutação e reprodução, que normalmente são chamados de operadores genéticos. No entanto, o AG apresenta aspectos peculiares em relação a outros métodos de otimização, tais como: (i) trabalhar com codificação de parâmetros em vez de variáveis originais do problema, (ii) procurar as soluções ótimas de um conjunto de soluções potenciais e não de apenas um ponto de partida, (iii) usar uma função de avaliação para as diferentes soluções e (iv) usar regras probabilísticas para encontrar novas soluções. Assim, o AG é simples, flexível, robusto e particularmente útil para resolver problemas em que outras técnicas de otimização enfrentam dificuldades (Goldberg, 1989).

A população e sua representação*:* - Em vez de trabalhar com uma única solução por iteração, o AG trabalha com um conjunto de soluções potenciais conhecido como população, que geralmente é escolhida estocasticamente na inicialização (Goldberg, 1989). Essas soluções potenciais são codificadas como uma codificação de *string* de comprimento fixo chamado cromossomo, que pode ser definido de várias maneiras para representar as soluções: cadeias binárias, cadeias de caracteres, números de ponto flutuante, lista de números inteiros e assim por diante (Michalewicz, 1996).

Geralmente, a representação da solução baseada em listas de números inteiros é aplicada para resolver problemas de otimização combinatória, como a programação de *Job Shop*. Desta forma, a lista de números inteiros pode representar uma lista de operação ordenada (Figura 1a), ou uma lista de valores de prioridade do trabalho nas máquinas (Figura 1b), por exemplo (Yamada, 2003). Um exemplo de codificação de cromossomos de números inteiros para nove operações (3 trabalhos e 3 máquinas) de *Job Shop*, é apresentado na Figura 1.

Cromossomo = [1,2,2,3,2,1,3,3,1]	Cromossomo = [1,2,5,8,7,9,3,4,6]
(a)	(b)

Figure 1. Representação de números inteiros de um cromossomo: (a) uma lista de operação ordenada e (b) uma lista de valores de prioridade do trabalho nas máquinas

Função de avaliação: - Para cada iteração, também chamada de geração, uma avaliação dos cromossomos é realizada usando uma função de *fitness* para alocar um valor de aptidão para cada solução potencial. O valor da aptidão deve representar a bondade de uma solução, que é tipicamente definida em relação à população atual, e é usada para classificar os membros da população para fornecer um tipo de limite de probabilidade para a operação de reprodução. Se, por exemplo, a otimização do AG está maximizando alguma função, o membro com um valor de aptidão relativamente maior deve ter maior probabilidade de reprodução.

A função de *fitness* pode ser uma função matemática, um experimento, um modelo de simulação ou um metamodelo.

Operadores genéticos: - A reprodução é o ponto crucial, em que a evolução dos cromossomos ocorre. O *crossover*, isto é: a recombinação dos ancestrais mais aptos selecionados, produzem novos cromossomos que eventualmente passam por um processo chamado mutação. A mutação consiste em alterar alguma característica do cromossomo.

A operação de seleção permite que os melhores cromossomos, dentro da população, tenham mais oportunidades de gerar novas soluções possíveis (descendentes) para o problema do que elementos que não foram classificados como os mais aptos. Além disso, o estágio de seleção precisa ser planejado adequadamente, porque é importante ter poucos descendentes que não foram classificados, mas que de alguma forma, podem ter algum recurso interessante para descobrir a melhor solução para o problema. Portanto a seleção, o crossover e a mutação são formas de manter a diversidade da população.

Os operadores genéticos de *crossover* e mutação têm, respectivamente, a tarefa de permitir a interação entre os cromossomos da população e a permanência de diversidade dentro do conjunto de soluções potenciais para o problema.

Com esses processos, os descendentes provavelmente terão características distintas de seus antepassados e, eventualmente, esses recursos permitem que o indivíduo gerado tenha maior capacidade de se adaptar ao ambiente em que vivem (Mitchell, 1997).

Os principais passos do algoritmo genético podem ser definidos da seguinte forma:

1. Gerar uma população inicial
2. Avaliar a capacidade dos indivíduos na população
3. Repita ...
    a) Selecione um subconjunto das soluções mais aptas da população
    b) Faça recombinação (crossover) entre os antepassados selecionados
    c) Realize a mutação em alguns descendentes gerados
    d) Avalie a aptidão dos descendentes gerados
    e) Substitua os indivíduos da população pela nova geração
4. ... até que uma solução satisfatória tenha sido encontrada

1.3 Modelagem e simulação

Conforme Banks (2000), a simulação é a imitação da operação de um sistema real ao longo do tempo. A simulação computacional pode ser definida como métodos para criar modelos reais ou artificiais, a fim de minimizar um sistema de operação ao longo de um período de tempo. Isto é: criar um modelo no computador destinado a realizar ensaios numéricos para estudar um sistema sujeito a um conjunto de restrições (Kelton, Sadowski e Sadowski, 2000).

Uma das principais vantagens dessa técnica é poder integrar diversas restrições do sistema e que pode ser acoplada ao modelo de simulação. Existem muitos métodos diferentes para modelagem e simulação de sistemas complexos, um dos quais é a simulação de evento

discreto. Nesse caso, a representação do sistema é definida para levar em consideração os pontos discretos no tempo em que ocorrem as mudanças do estado do sistema.

Em geral, a simulação é uma escolha adequada para o estudo de problemas estocásticos e dinâmicos, pois produz uma medida de desempenho semelhante ao desempenho real do sistema em comparação com funções matemáticas (Phanden, Jain e Verma, 2012).

2 DETALHAMENTO DA ABORDAGEM PROPOSTA

A abordagem de integração proposta neste trabalho baseia-se em um modelo de simulação de eventos discretos e em qualquer método de otimização, que é executado como um aplicativo independente. Nesse artigo, o modelo de simulação é integrado a um método de AG desenvolvido em linguagem C++. Portanto, a avaliação da função de *fitness* é realizada usando o modelo de simulação que também faz o cálculo para as medidas de desempenho, *makespan* médio (tempo médio de atravessamento das ordens no sistema), o atraso total e o número de trabalhos atrasados. Os módulos deste *framework* são explicados a seguir.

Módulo de simulação: - Este módulo foi desenvolvido utilizando o software de simulação ARENA 15 e simulação de eventos discretos. O módulo de simulação é usado para (i) avaliar a solução candidata ao problema de sequenciamento gerado pelo módulo de otimização, (ii) construir um agendamento tendo em conta todas as restrições, (iii) avaliar o desempenho das regras comuns de despacho em condições estocásticas e dinâmicas do chão de fábrica (para fins de comparação).

O modelo de simulação (Figura 2) representa um cenário de produção estocástica e dinâmica, constituído por 8 máquinas e 10 *jobs* com rotas pré-estabelecidas e o tempo total de produção estimado, de acordo com a Tabela 1. Neste tipo de ambiente, o modelo considera o comportamento aleatório da chegada das ordens e dos tempos estocásticos de produção. Os detalhes de seu desenvolvimento e validação são apresentados em (Silva et al.; Silva, Grassi e Pereira, 2014). É possível ver que o fluxo de trabalho não é unidirecional, conforme definido em um *job shop* (Phanden, Jain e Verma, 2012).

Tabela 1. Rota e tempo total para produção job shop

Ordem	Rota	Tempo (min)	Ordem	Rota	Tempo (min)
1	1,2,3,6	16.9	6	1,2,4,3,1,2,4,5,7	36.0
2	1,3,7,8	16.4	7	2,6,5,6,8	20.7
3	1,5,8	11.6	8	1,2,4,5,7	19.7
4	1,2,3,5,6,7,8	27.3	9	2,3,4,6,7	21.0
5	3,4,6,7	16.3	10	1,2,4,5,8	20.6

Proceedings of the XXXVIII Iberian Latin-American Congress on Computational Methods in Engineering P.O. Faria, R.H. Lopez, L.F.F. Miguel, W.J.S. Gomes, M. Noronha (Editores), ABMEC, Florianópolis, SC, Brazil, November 5-8, 2017.

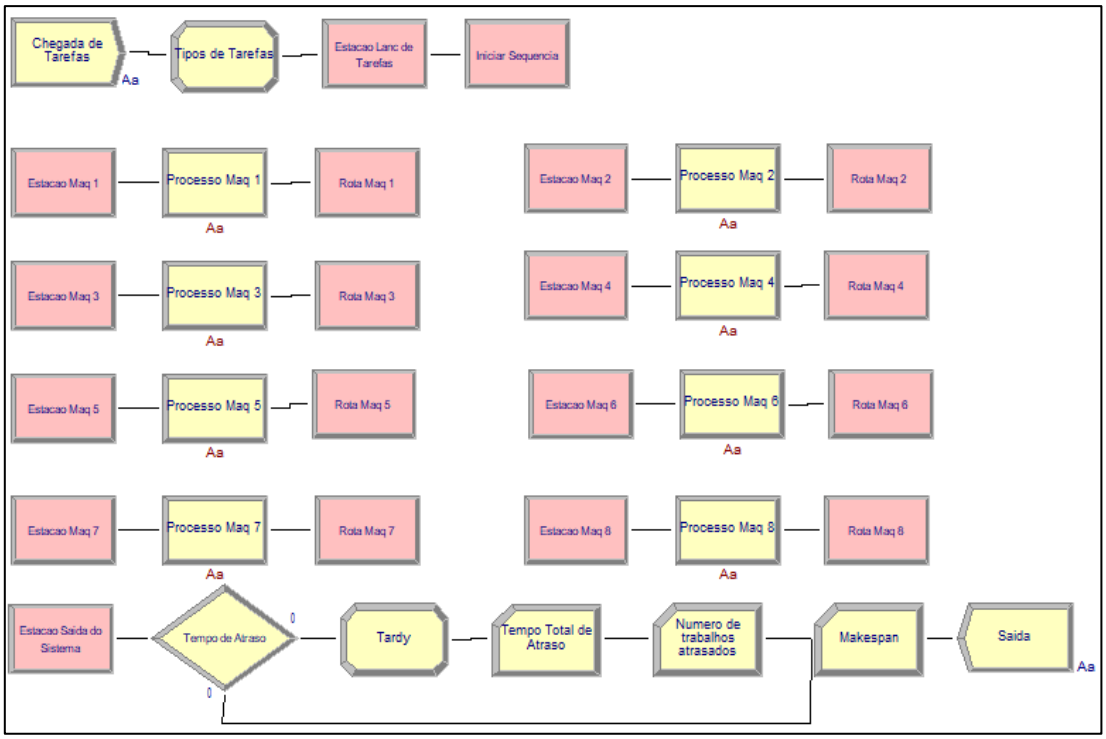

Figura 2. Modelo de simulação do cenário de produção estocástica e dinâmica abordado neste trabalho (Silva et al., 2014)

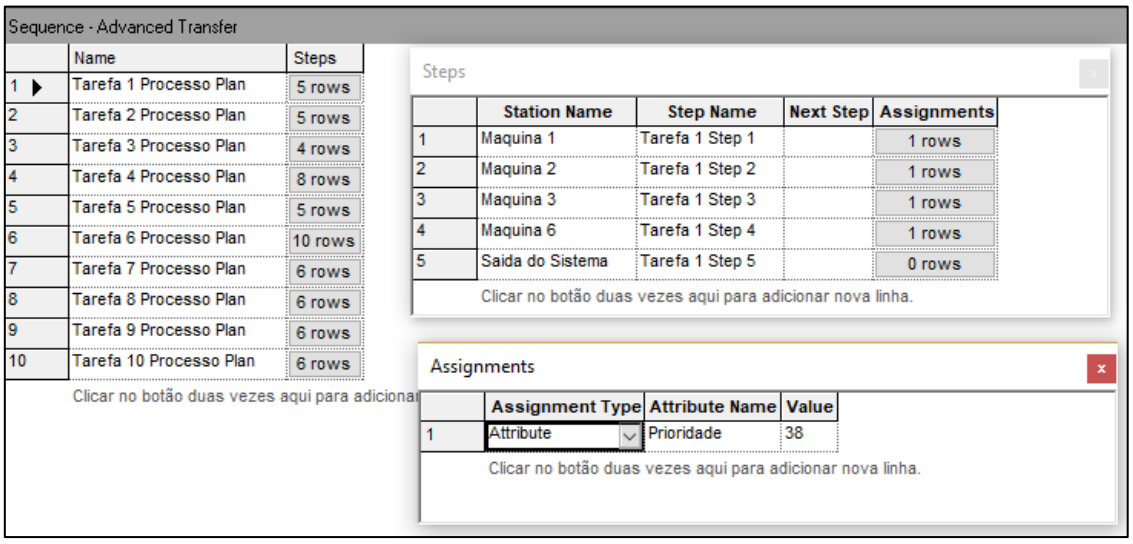

Figura 3. Programação interna pré-estabelecida no ARENA módulo *Sequence*. A Tarefa 1, possui uma sequencia de produção em 4 máquinas definidas pelos *steps* e a cada *step* há uma atribuição de prioridade, exceto para o *step* de saída

Os principais parâmetros do modelo de simulação são apresentados na Tabela 2. A média da distribuição exponencial do tempo entre as chegadas de pedidos deve ser escolhida para atingir o nível de utilização desejado em todas as máquinas. A validação operacional foi realizada de acordo com (Leal, et al., 2011).

Tabela 2: Parâmetros propostos para o modelo de simulação

Parâmetros	Expressão	Descrição
Tempo entre a chegada de ordens	**EXPO(*tba*)**	**Distribuição Exponencial da média *tba* (Hildebrandt, Heger, and Scholz-Reiter, 2010)**
Tempo de processamneto	**NORMAL(m, std_p)**	**Distribuição Normal da média m and desvio padrão = std_p**
Fator de confiança	**k**	**Confiança relacionada ao tempo de entrega**
Tempo estimado de produção total	**t0**	**Estimativa dada pelo tempo médio de processo x número de operações de cada job (Santoro e Mesquita, 2008)**
Data de entrega	**NORMAL((1+k) *t0, 0.1*(1+k)*t0)**	**Distribuição Normal da média entre (1+k)*t0 e desvio padrão 0,1*(1+k*t0(Santoro e Mesquita, 2008)**

***O formato das soluções de agendamento*:** - O módulo de simulação desenvolvido reconhece uma solução do problema de agendamento através de uma sequência de prioridades dos trabalhos em cada uma das máquinas, conforme ilustrado na Figura 3. Assim, uma solução é representada dentro do módulo como uma lista de número inteiro k, onde K é o número de operações. É importante destacar que é independente da representação adotada pelo módulo de otimização, na qual se pode usar qualquer abordagem disponível.

***A lógica do modelo**:* - Um dos principais objetivos do módulo de simulação é avaliar a solução a partir de um módulo de otimização. Aqui, as soluções criadas pelo módulo de otimização são inseridas no modelo usando o Visual Basic for Application (VBA) em ARENA. O projeto VBA tem acesso ao modelo de simulação através do objeto ThisDocument que contém os eventos VBA internos no qual o código está ativado. Esses eventos incorporados se enquadram em três categorias que definem o momento exato durante a simulação em que o código VBA é executado (Kelton, Sadowski e Sadowski, 2000): (1) Eventos pré-executados (por exemplo, DocumentOpen); (2) eventos de execução iniciados pela ARENA (por exemplo, RunBegin, RunEndReplication); (3) Eventos de execução iniciados pelo modelo / usuário (por exemplo, UserFunction, VBA_Block_Fire).

As principais informações da sequência desses eventos VBA, com instruções para inserir uma sequência de prioridades no modelo, pode ser:

1. Inicio do processamento

2. Arena chama o modelo e inicia a simulação

3. Início da simulação

   a) Define no ARENA o modelo de variáveis:
   b) Define no ARENA o módulo de variáveis:
   c) Define o Arquivo de entrada (*InputFile*):
   d) Salva numa matriz os valores de prioridades gerado pelo módulo de otimização:
   e) Carrega essa matriz para o modelo corrente:
   f) Encontre o módulo no qual esses valores foram armazenados:
   g) Carrega os valores de prioridade para cada tarefa dos J*obs*

4. Início da replicação

5. Fim da replicação

   a) Define um objeto no SIMAN onde cada um desses resultados está disponível
   b) Obtenha os resultados do SIMAN (*makespan*, tempo total de atraso e número de ordens atrasadas):
   c) Salva os resultados num arquivo:

6. Fim da simulação

7. Fim do processamento

Nessa sequência de eventos VBA, os valores de prioridade do módulo de otimização (arquivo Priotities.txt) são inseridos no campo do modelo apropriado antes da simulação e os resultados são salvos após cada replicação do modelo. Os valores de prioridade vêm do arquivo ('Priorities.txt') criado pelo módulo de otimização que será descrito oportunamente, na próxima seção

O modelo de otimização: - A simulação de eventos discretos é uma ferramenta útil para avaliar um determinado *design*, mas precisa de um método de otimização para encontrar uma solução melhorada. Devido à complexidade dos sistemas frequentemente abordados pela simulação, abordagens metaheurísticas como o AG, foram a escolha mais comum para resolver problemas práticos. Assim, o módulo de otimização é baseado em um método de AG desenvolvido em linguagem de programação C++.

O primeiro passo de AG é definir a representação/codificação da solução. Uma lista de valores de prioridade de trabalhos nas máquinas é usada aqui. Um gene de um cromossomo representa um valor de prioridade, tal como definido anteriormente na Figura 3 para cada trabalho em cada máquina, ordenado por máquinas. Assim, o valor do primeiro gene do cromossomo representa a prioridade do trabalho J_1 na máquina M_1, o valor do segundo gene representa a prioridade do trabalho J_2 na máquina M_1, e assim por diante. Além disso, cada trabalho tem um valor de prioridade diferente e sempre que um trabalho retorna a uma máquina já visitada, ele obtém o mesmo valor de prioridade que antes. Como resultado, para o problema abordado neste trabalho, o cromossomo é definido como uma lista de 47 valores de prioridade como ilustrado na Figura 5. Esses valores de prioridade do cromossomo são escritos em um arquivo de saída ('Priorities.txt') que é lido pelo código VBA dentro do modelo de simulação.

À medida que o módulo de otimização é executado como uma aplicação independente, qualquer outro tipo de codificação e método numérico poderia ser aplicado.

Cromossomo =	38 13 41 2 12 26 45 8 19 5 17 37 28 3 42 34 40 36 24 39 46 21 0 31 6 32 16 15 1 22 7 20 11 33 23 35 44 18 29 27 25 43 14 9 4 30 10

Figura 4 – Representação codificada da solução utilizada neste trabalho: lista de valores de prioridade dos jobs nas máquinas, onde cada gene do cromossomo representa o valor de prioridades ordenado por máquina

A integração foi realizada usando as instalações de controle interno e entre aplicativos, através da tecnologia ActiveX Automation e da VBA. O ActiveX é um modelo de objeto

componente (COM) que permite uma comunicação entre aplicações independentemente da linguagem de programação (Kaushal, 2011).

Ressalta-se que o controle do módulo de simulação é executado pela aplicação C++. Além de realizar as operações usuais do método de otimização, como os operadores genéticos, o aplicativo C++ controla a execução do modelo sempre que uma nova solução candidata (cromossomo) precisa ser avaliada. Sempre que uma solução deve ser avaliada pela função de *fitness*, seus valores de prioridade são escritos em um arquivo e um método é chamado para invocar o modelo de simulação em ARENA passando para ele o controle da aplicação. No final da simulação, o controle retorna ao aplicativo C++ que lê os resultados de *makespan* médio, atraso total e número de trabalhos atrasados para calcular um valor de aptidão para a solução candidata.

O método que chama o modelo de simulação ARENA (ArenaSample), que é criado com base na biblioteca COM DispHelper,

Vale ressaltar que os critérios de parada são: o número de gerações e o número de replicação para os modelos de otimização e simulação respectivamente.

Um diagrama de fluxo completo do ambiente de integração proposto é mostrado na Figura 5.

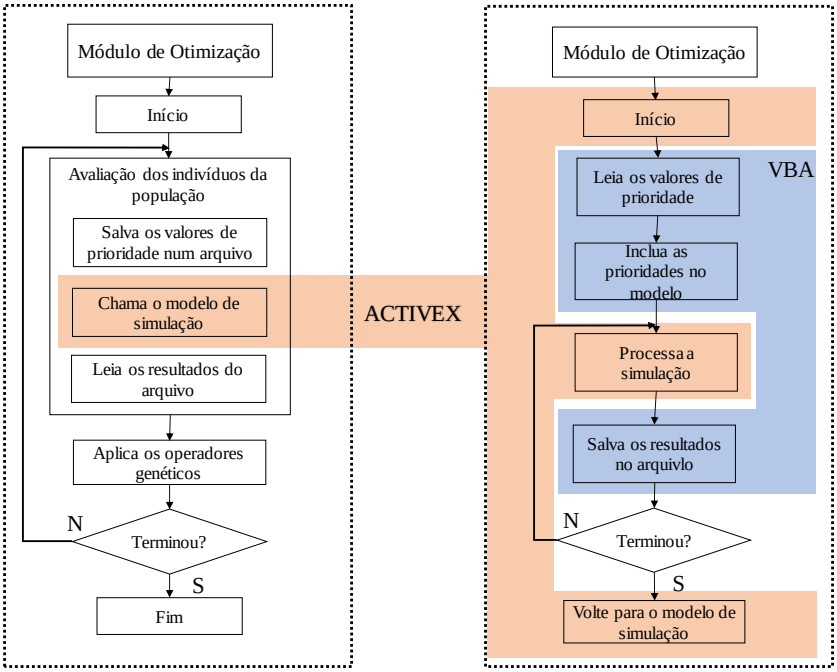

Figura 5. Fluxograma do ambiente de integração

Experimentos numéricos: - Neste trabalho, o módulo de otimização é desenvolvido com base na biblioteca C++ de Algoritmo Genético Componentes GAlib (Wall, 1996). No entanto, à medida que o método de otimização é executado como um aplicativo independente, o módulo de simulação pode ser facilmente integrado com quaisquer outros aplicativos desenvolvidos em qualquer linguagem de programação como C++, Visual Basic ou Java, apenas sem nome algum. A Tabela 3 resume os parâmetros adotados no módulo de otimização (AG).

Tabela 3. Parâmetros propostos para o modelo de otimização AG

Parâmetros	Valores adotados	Justificativa
Representação do cromossomo	**Operação ordenada**	**Lei (2011)**
Seleção	**Estado estacionário**	**Yamada (2003)**
Taxa de substituição	**80%**	**Mitchell (1997**
Tamaho da população	**20**	**Experimentos**
Crossover	**90%**	**Mitchell (1997**
Mutação	**5%**	**Mitchell (1997**
Total de geraçõres	**100**	**Experimentos**
Critério de parada	**Número de gerações**	**Mitchell (1997**
Função de Avaliação	**Modelo de simulação**	**Proposta de integração**

***A demanda variável estocástica*: -** Os efeitos da demanda variável estocástica em diferentes regras de despacho e nos parâmetros de otimização também foram avaliados. Para isso, a média da distribuição exponencial do tempo entre as chegadas de pedidos é escolhida para atingir diferentes níveis de utilização em todas as máquinas. Seis diferentes níveis de demanda foram testados para avaliar seus efeitos na solução de acordo com as medidas de desempenho, o atraso total e o número de trabalhos atrasados. Os valores utilizados nos principais parâmetros do modelo de simulação são apresentados na Tabela 4. É importante ressaltar que o nível de demanda aumenta à medida que o tempo entre as chegadas diminui.

Tabela 4. Principais parâmetros do modelo de simulação

Parâmetro	Descrição	Valores
tba	**Tempo médio entre chegadas**	**6, 7, 8, 9, 10, 11, 12, 13 e 14**
mp	**Tempo médio de processamento**	**4**
$\mathbf{std_p}$	**Desvio médio do tempo de processamento**	**0,4**
k	**Fator de confiança em relação à data de vencimento**	**0,3**
t0	**Estimativa do tempo total de produção (tempo médio de processamento x número de operação)**	**Valor de acordo com a Tabela 1**
md	**Valor médio da data de entrega**	**(1+k) *t0**
$\mathbf{std_d}$	**Desvio médio da data de entrega**	**0.1* (1+k)*t0**

***Desempenho relacionado à qualidade da solução*:-** No modelo de simulação proposto a otimização multiobjetiva, tratou de três medidas de desempenho (*makepan* médio, atraso total e número de trabalhos atrasados), onde foi considerado a mesma importância percentual para as mesmas. Os resultados do método proposto foram comparados com algumas regras comuns de despacho, a saber: Menor tempo de processamento (SPT-Shortest Process Time), Maior tempo de processamento (LPT-Longest Process Time), a data de vencimento mais

próxima (EDD-Earlest Due Data), primeiro a chegar, primeiro a ser processado (FIFO- First-In-First-Out) e último a chegar, primeiro a ser processado (LIFO-Last-In-First-Out).

3 APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

Os experimentos utilizando o AG foram realizados para ilustrar a abordagem proposta: o uso de um modelo de simulação como função *fitness* de um AG e a validação de que o resultado obtido pode ser robusto na variação da demanda, quando esta pode se afastar entre mais ou menos (+-) 40% em relação a média considerada, conforme pode ser visto na Tabela 4, apresentada anteriormente.

Considere que o processamento completo foi realizado para o intervalo de recebimento de ordens EXPO(10), considerado como frequente. A partir deste processamento com os valores de prioridades devidamente incluídos no ARENA foram realizadas as simulações para os demais valores de intervalos de chegada, isto é para: 14, 13, 12, 11, 9. 8. 7 e 6.

Na Figura 6, tem-se o gráfico onde pode ser visto o distanciamento em percentual deste valor habitual, pode-se notar que a curva que sofre menor desvio é da prioridade, embora todas as demais curvas obedeçam uma certa semelhança de afastamento, sendo mais acentuado quando o ritmo de chegada é menor e uma maior regularidade quando o ritmo é menor.

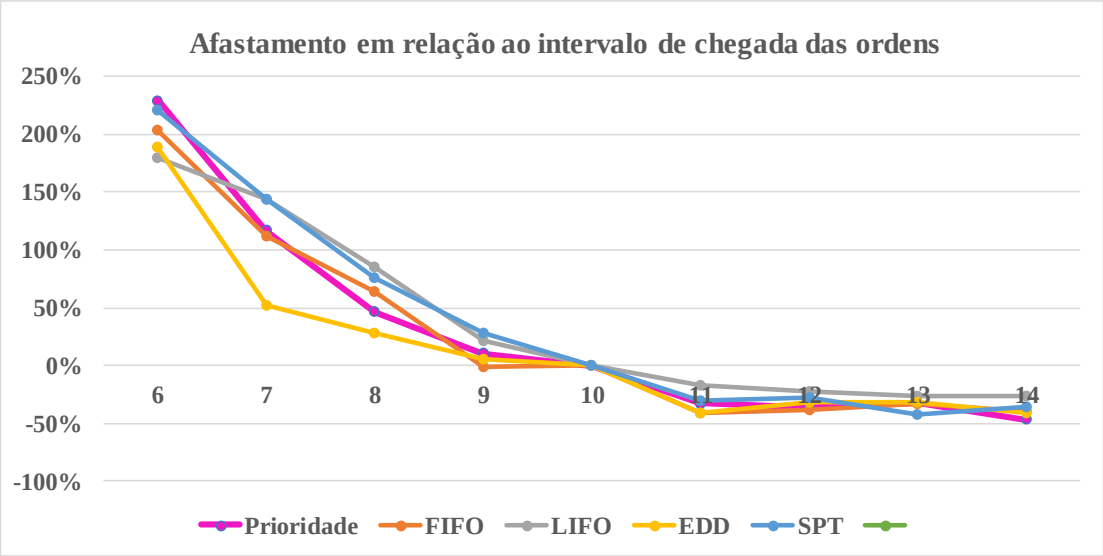

Figura 6. Comparação percentual entre o afastamento relacionado ao intervalo entre a chegada das ordens

Para melhor visualização e análise do comportamento do sistema, com relação aos resultados encontrados, optou-se em separar por maior e menor intervalo de tempo entre a chegada das ordens conforme as Figuras 7 e 8.

Para as ordens com maior intervalo de tempo entre chegadas, como pode ser observado na Figura 7, as variações são pequenas na comparação entre as diversas regras de lançamento.

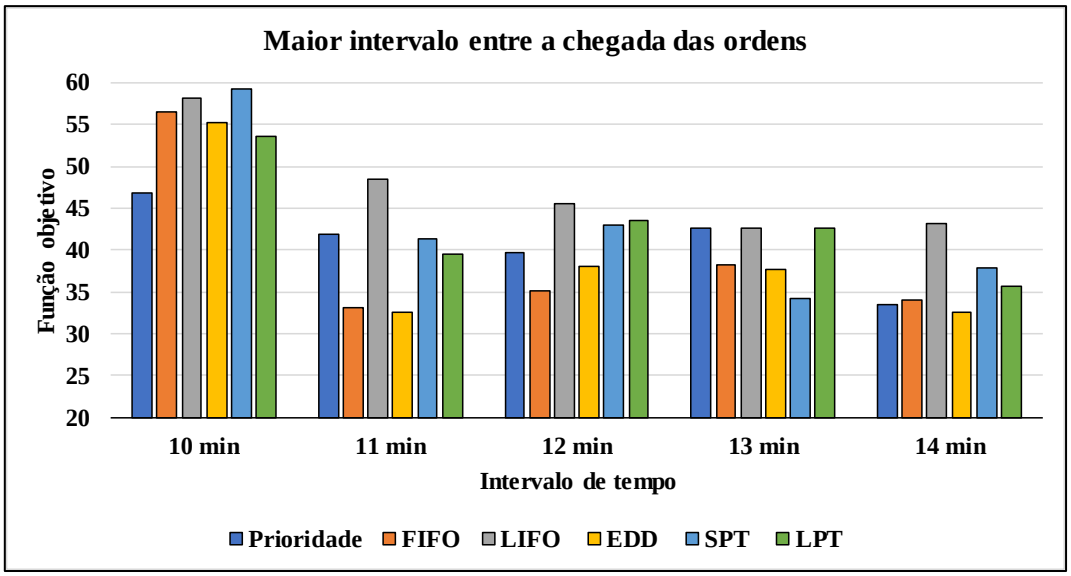

Figura 7. Comparação entre os intervalos de chegada por regra de lançamento (maior intervalo)

Com relação ao menor intervalo de tempo entre a chegada das ordens, tem-se que a diferença é bem mais acentuada, principalmente quanto menor for esse valor (Figura 8).

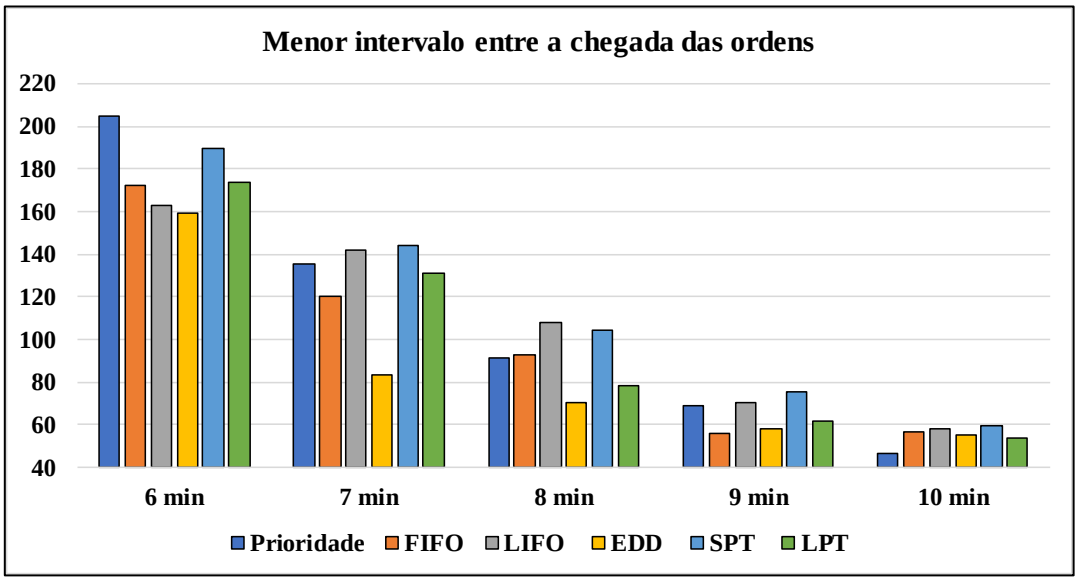

Figura 8. Comparação entre os intervalos de chegada por regra de lançamento (menor intervalo)

Em complemento a Figura 8, observe os valores disponibilizados nas Tabelas 5 e 6.

Na Tabela 5 vê-se que quanto menor intervalo de tempo, maior a distância entre os valores estabelecidos como habitual, e na Tabela 6 pode-se notar o desvio relacionado é cerca de 1,5 vezes o valor predecessor, para o AG enquanto que o pior desvio são paras regras de lançamento LIFO e EDD.

Tabela 5. Valores da função objetivo em função do menor intervalo entre a chegada de ordens

Intervalos entre chegada das ordens					
	6 min	7 min	8 min	9 min	10 min
Prioridade	205	135	91	69	47
FIFO	172	120	93	56	57
LIFO	163	142	108	70	58
EDD	160	84	70	58	55
SPT	190	144	104	76	59
LPT	174	131	79	61	54

Foi observado que a regra de lançamento que utiliza as prioridades obtidas pelo AG, nesse experimento, não foi sempre a melhor opção e em alguns casos chegou mesmo a ser superada pelas demais regras. Ressalva-se que o AG foi processado apenas para o intervalo de 10 min.

Conforme comentado anteriormente, na Tabela 6 pode ser observado o desvio relacionado aos valores predecessores, isto é: a cada redução do intervalo de tempo entre as entregas os valores da função objetivo ficam cada vez maiores e somente o AG mantém uma certa regularidade nesse afastamento em torno de 1,5 vezes o valor anterior.

Tabela 6. Desvio relacionado aos valores predecessores

Desvio em relação ao predecessor relativo ao menor intervalo entre as chegadas					
	6 min	7 min	8 min	9 min	10 min
Prioridade	1,5	1,5	1,3	1,5	1,0
FIFO	1,4	1,3	1,7	1,0	1,0
LIFO	1,1	1,3	1,5	1,2	1,0
EDD	1,9	1,2	1,2	1,1	1,0
SPT	1,3	1,4	1,4	1,3	1,0
LPT	1,3	1,7	1,3	1,1	1,0

Para esse processamento foi utilizado uma Workstation Dell Precision T7610 com um processador Intel I7 com 1,8 Ghz, memória de 32 Mb e o Windows 7.

4 CONCLUSÃO

A integração de um modelo de simulação e uma técnica de otimização foi bem sucedida, e ainda o mesmo modelo de simulação serviu como avaliação da função de aptidão do algoritmo genético.

A aplicação de 6 regras de lançamento das ordens de produção (Prioridade, FIFO, LIFO, EDD, SPT e LPT) foram executadas e comparadas entre si,

Com relação ao objetivo de demonstrar que o agendamento fornecido pelo AG poderia ser robusto, isto é: sem necessitar que um novo processamento fosse realizado a cada variação no intervalo de tempo entre a chegada das ordens de produção foi alcançado, embora os valores da função objetivo não se mantenham ela permite inferir que a cada afastamento há um aumento aproximado de 1,5 vezes o valor apresentado anteriormente. Então, pode-se considerar que a condição mais estável em função da variação entre os intervalos de chegadas das ordens foi obtida pelo AG.

Pode-se chegar a pelo menos duas outras inferências: primeiro, quanto maior o intervalo entre a chegada das ordens no sistema, menor é a variação comparada aos valores habituais, uma vez que embora tecnicamente exista uma disponibilidade maior dos recursos, as ordens podem chegar todas no último momento provocando filas e alterando os valores das medidas de desempenho que provocam a alteração na função objetivo. Vale ressaltar que quando não há fila a regra FIFO é acionada automaticamente pelo sistema de simulação.

Em segundo, que quanto menor o intervalo entre as chegadas maior é a variação comparada aos valores habituais e que os valores da função objetivo, percentualmente, aumentam muito chegando a 1,5 vezes maior que o valor obtido no intervalo anterior.

A principal dificuldade apresentada pela utilização simultânea de ambiente dinâmico e estocástico com a chegada de ordens baseadas em distribuição de probabilidades tornou o experimento muito demorado (cerca de 2,5 horas de processamento).

Serão necessários novos experimentos que busquem reduzir o tempo de processamento.

AGRADECIMENTOS

Os autores agradecem a UNINOVE e a CAPES pela bolsa de estudos e pelo apoio financeiro para participar desta Conferência.

REFERENCIAS

Adams, J., Balas, E., & Zawack, D. (s.d.). The Shifting Bottleneck Procedure for Job Shop Scheduling. *Management Science*, pp. 391-401.

Aydin, M. E., & Fogarty, T. C. (2004). A Distributed Evolutionary Simulated Annealing Algorithm for Combinatorial Optimisation Problems. *Journal of Heuristics, 10*, pp. 269-292.

Azadivar, F. A. (1992). A Tutorial on Simulation Optimization. Em D. G. J.J. Swain (Ed.), *Proceedings of the 1992 Winter Simulation.*

Banks, J. (2000). Introduction to Simulation. *Proceedind of Winter Simulation Conference.* Orlando, FLórida, USA.

Baptiste, P., Flamini, M., & Sourd, F. (2008). Lagrangian bounds for just-in-time job-shop scheduling. *Computers & Operations Research, 35, n.3*, pp. 906-915.

Fan, K., & Zhang, R. (2010). An analysis of reseaerch in job shop scheduling problem (2000-2009). *IEEE International Conference on Advanced Management Science, 2*, pp. 282-288.

Fu, M. C. (2002). Otimization for Simulation: Theory vs Practice. *INFORMS Journal of Computing*(14(3)), pp. 192-215.

Goldberg, D. (1989). *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, Reading, MA.

Gonçalves, J. F., Mendes, J. J., & Rezende, M. G. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European Journal of Operational Research,, 167*, pp. 77-95.

Haddadzade, M., Razfar, M. R., & Fazel Zarandi, M. H. (2014). Integration of process planning and job shop scheduling with stochastic processing time. *International Journal Advanced Manufactoring Technology*(71), pp. 241-252.

Hildebrandt, T., Heger, J., & Scholz-Reiter, B. (2010). Towards improved dispatching rules for complex shop floor scenarios—a genetic programming approach. *Genetic and Evolutionary Computation Conference GECCO, 10*, pp. 257-264.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* The University of Michigan Press. Ann Harbor.

Huang, K. L., & Liao, C. J. (2008). Ant colony optimization combined with taboo search for the job shop scheduling problem. *Computers & Operations Research, 4*, pp. 1030-1046.

Iassinovski, S., Artiba, A., Bachelet, V., & Riane, F. (2003). Integration of Simulation and optimization for solving complex decision making problems. *International Journal of Prtoduction Economics*, pp. 3-10.

Jain, A. S., & Meeran, S. (1999). Deterministic Job-Shop Scheduling: Past Present and Future. *European Operational Research, 18*, p. 2.

Kaushak, R. K. (2011). Component Object Model Communication Fundamentals and its Components. *International Journal of Computer Science & Technology, 2*(4). Fonte: http://www.ijcst.com/

Kelton, W. D., Sadowiski, R. P., & Sadowiski, D. A. (2000). *Simulation With Arena* (2ª ed.). McGraw-Hill.

Leal, F., Costa, R. F., & Montevechi, J. A. (2011). A practical guide for operational validation of discrete simulation models. *Pesquisa Operacional*(31 (1)), pp. 57-77.

Lei, D. (2011). Simplified multi-objective genetic algorithms for stochastic job shop scheduling. *Applied Soft Computing Journal*(11 (8)), pp. 4991-4996.

Lukaszewicz, P. P. (2005). Metaheuristics for job shop scheduling problem comparison of effective methods. *Master Thesis,123 p.* Aarhus School of Business.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs* (3 ed.). Berlim: Springer-Verlag.

Mitchell, T. M. (1997). *Machine Learning.* New York: McGraw-Hill.

Nowicki, E., & Smutinicki, C. (1996). A Fast Taboo Serarch Algorithm for the Job Shop Problem. *Management Science, 42 (6)*, pp. 797-813.

Ouelhadj, D., & Petrovic, S. (2009). A Survey of Dynamic Scheduling in Manufacturing Systems. *Journal of Scheduling*(12), pp. 417-431.

Phanden, R. K., Jain, A., & Verma, R. (2012). Genetic Algorithm-based approach for jo shop scheduling. *Journal of Manufactoring Technology Management, 23, No. 7*, pp. 937-946.

Salch, A., Gayon, J. P., & Lemaire, P. (2013). Optimal static priority rules for stochastic scheduling with impatience. *Operations Research Letters, 41*, pp. 81-85.

Santoro, M. C., & Mesquita, M. A. (2008). The effect of the workload on due-data performance in job shop scheduling. *Brasilian Journal of Productions and Production Management*(5 (1)), pp. 75-88.

Silva, E. B., Costa, M., Silva, M. F. S., & Pereira, F. H. (2014). Simulation study of dispatching rules in stochastic job shop dynamic scheduling. *World Journal of Modelling and Simulation, 10*, pp. 231-240.

Silva, M. F. S., Grassi, F., & Pereira, F. H. (2014). Integration of Discrete-event Simulation Model and Optimisation Method for Solving Stochastic Job Shop Dynamic Scheduling Problem. *International Conference on Industrial Engineering and Operations Management. 1*, p. 64. Málaga: Book of Abstracts of the 8th CIO and 20th ICIEOM. Málaga: Andalucía Tech.

Tavakkoli-Moghaddam, R., Jolai, F., Vaziri, F., Ahmed, P. K., & Azaron, A. (2005). A hybrid method for solving stochastic job shop scheduling problems. *Applied Mathematics and Computation, 170*, pp. 185-206.

Terekhov, D., Tran, T., Down, D. G., & Beck, J. (2014). Integrating Queueing Theory and Scheduling for Dynamic Scheduling Problems. *Journal of Artificial Intelligence Research, 50*, pp. 535-572.

Wall, M. (1996). GALIB: A C++ library of genetic algorithm components. *Mechanical Engineering Departament, Massachussetts Institute of Technology*. Fonte: http://lancet.mit.edu/ga/dist/

Yamada, T. (2003). Studies on metaheuristics for jobshop and flowshop scheduling problems. *PhD dissertação Kyoto University*.

Zandieh, M., Khatami, A. R., & Rahmatib, S. H. (2017). Flexible job shop scheduling under condition-based maintenance: Improved version of imperialist competitive algorithm. *Applied Soft Computing, 58*, pp. 449-464.

Zhang, B. X., Yi, L. X., & Xiao, S. (2005). Study of stochastic job shop dynamic scheduling. Em P. o. Cybernetics (Ed.). Guangzhou, 18-21 August.