www.revistageintec.net
ISSN: 2237-0722
revista GEINTEC Gestão, Inovação e Tecnologias

SIMULAÇÃO COMPUTACIONAL PARA OTIMIZAÇÃO DE FILAS EM PROCESSOS

COMPUTER SIMULATION FOR THE OPTIMIZATION OF PROCESS QUEUES

Guilherme Tonini Botassoli[1]; Rafael Alvise Alberti[2]; João Carlos Furtado[3]
[1]Programa de Pós-Graduação em Sistemas e Processos Industriais – PPGSPI
Universidade de Santa Cruz do Sul – UNISC – Santa Cruz do Sul/RS – Brasil
guibotass@live.com
[2]Programa de Pós-Graduação em Sistemas e Processos Industriais – PPGSPI
Universidade de Santa Cruz do Sul – UNISC – Santa Cruz do Sul/RS – Brasil
alberti_rafael@yahoo.com.br
[3]Programa de Pós-Graduação em Sistemas e Processos Industriais – PPGSPI
Universidade de Santa Cruz do Sul – UNISC – Santa Cruz do Sul/RS – Brasil
jcarlosf@unisc.br

Resumo

A utilização de técnicas de otimização em simulação impactam fortemente em diferentes áreas e por isso, acabam por se tornar ferramentas fundamentais na engenharia de processos. Assim, foi desenvolvido um algoritmo para otimização em simulação computacional, em linguagem C a partir do programa Code::Blocks, utilizando-se de conceitos provenientes do Método Enxame de Partículas (MEP) e Algoritmos Genéticos (AG). Partindo inicialmente de um algoritmo base com matriz de números aleatórios, buscou-se a configuração que minimizasse tempos de espera em filas frente aos processos, ou seja, um resultado Gbest. O desenvolvimento deste algoritmo e seus resultados contemplam a possibilidade de estudos e aplicações futuras em modelos reais de processos, com resultados analisados e verificados integralmente no contexto das organizações.

Palavras-chave: simulação; otimização; algoritmo; enxame de partículas; algoritmos genéticos.

Abstract

The use of optimization techniques in simulation impact heavily in different areas and, eventually to become fundamental tools in process engineering. Thus was developed an algorithm for optimization in computer simulation, in the C programming language from the program Code::Blocks, using concepts from the Particle Swarm Optimization Method and Genetic Algorithms. Starting initially from a base algorithm with array of random numbers, sought a setting that minimizes waiting times in front of queues processes, a result Gbest. The development of this algorithm and its results include the possibility of studies and future applications in actual models of processes, with results analyzed and fully verified in the context of organizations.

Key-words: simulation; optimization; algorithm; genetic algorithms; particle swarm.

1. Introdução

As organizações têm sofrido pressão crescente nos últimos anos para que realizem atualizações gerencias e de tecnologias empregadas. A forte concorrência e a crescente exigência dos consumidores gera preocupação e demanda por ferramentas para o seu desenvolvimento competitivo. Neste ambiente, identificar a melhor condição de programação para a produção aliado a otimização da produtividade é assunto de interesse das organizações.

Na medida em que o planejamento se tornou fundamental para a tomada de decisões, tecnologias de informação sobre a administração e manufatura de processos se tornaram populares, gerando incrementos na eficiência dos sistemas e possibilitando que decisões a serem tomadas pudessem ser mais assertivas possíveis (BAKHTAZE, 2004).

Assim, ferramentas de auxilio a gestão que visam aperfeiçoar processos, ampliaram sua importância e como exemplo destas ferramentas tem-se o uso da simulação computacional para a otimização de processos.

Esta integração entre simulação e otimização é tratada como um processo que testa várias combinações com diferentes valores para variáveis controláveis, na tentativa de buscar uma solução ótima (HARREL et al., 2000).

Portanto, o presente artigo procura demonstrar o desenvolvimento de um algoritmo para otimização em simulação computacional, utilizando conceitos do Método Enxame de Partículas (MEP) e Algoritmos Genéticos (AG). O objetivo do algoritmo é buscar a configuração que gere/apresente o menor tempo de espera em filas frente aos processos (recursos) de acordo com um modelo de simulação.

2 Revisão de literatura

2.1 Simulação

A simulação é uma técnica para projeto e avaliação de sistemas e suas aplicações têm crescido em todas as áreas, auxiliando gestores na tomada de decisão em problemas complexos, possibilitando um melhor conhecimento dos processos das organizações (SAKURADA e MIYAKE, 2009).

Sua utilização fornece uma visualização detalhada (planejamento, processo e controle) do funcionamento em diferentes cenários, capaz de indicar opções de decisões que contemplem menores custos e/ou maiores ganhos (SILVA, 2006; CHWIF e MEDINA, 2010; UM, HYEONJAE, LEE, 2009; MORABITO e PUREZA, 2010; COSTA, 2011).

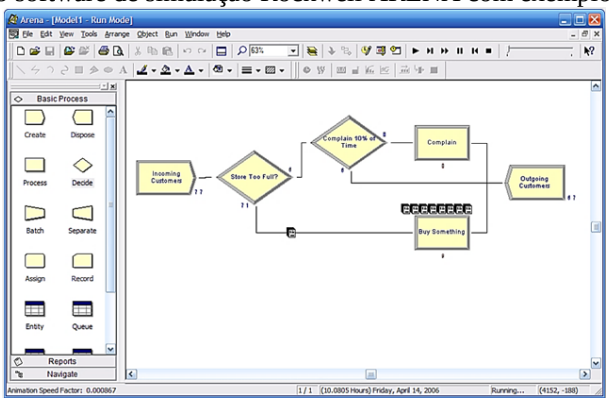
Como propósito, busca adquirir conhecimento operacional do sistema, desenvolver políticas de operação e de pesquisa para melhoria do desempenho, testar novos conceitos e/ou sistemas ante sua implementação e obter informações sem causar distúrbio no sistema atual (CHUNG, 2004).

Em questão de objetivos, Costa (2011) ressalta os seguintes: Construção de um modelo de simulação do processo implementado, que incorpore os conceitos e métodos utilizados; realização de ensaios de simulação em função de diferentes cenários; identificação e proposta de alterações que conduzam a uma maior eficiência e maior precisão de funcionamento do sistema de produção.

Banks et al. (2005) e Diehl et al. (2009), comentam que o maior benefício da utilização da simulação em ambientes manufatureiros é a possibilidade de se obter uma visão geral (macro) do efeito de uma pequena mudança (micro) no sistema.

Atualmente a simulação baseia-se no trabalho a partir de softwares simuladores (Figura 1), tornando fácil e ágil o processo de simulação como um todo, desde a implementação de modelos até a análise dos resultados a partir de relatórios gerados pelos próprios softwares.

Figura 1 - Visualização do software de simulação Rockwell ARENA com exemplo de modelo de simulação.



Fonte: Autoria própria (2014).

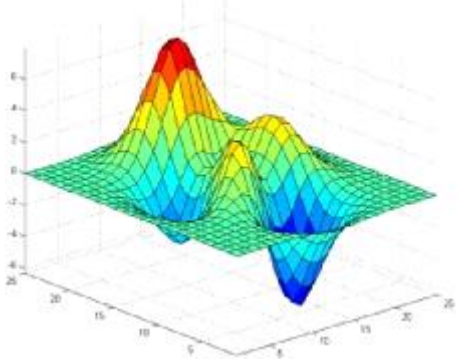
2.2 Otimização

Embora a simulação computacional apresente uma série de vantagens, ela possui algumas limitações por se tratar de uma ferramenta avaliadora e não geradora de soluções. Uma abordagem que faz com que esta situação comece a ser mudada é a utilização de técnicas de otimização na simulação computacional (CHWIF E MEDINA, 2010).

Esmin (2005) define a otimização como sendo a tarefa de determinar as "melhores" soluções para certos problemas matematicamente formulados (Figura 2). Questões de otimização são geralmente focadas em responder a questões do tipo *"how to"* (como ou o quê), visando maximizar respostas de um vetor de indicadores de interesse (BOWDEN & HALL, 1998; AZADIVAR, 1999).

Logo, toda tarefa de busca e otimização deve possuir componentes, como o espaço de busca, onde são consideradas todas as possibilidades de solução de um determinado problema e a função de avaliação, que é uma maneira de avaliar os membros do espaço de busca.

Figura 2 - Uso da otimização para encontrar os pontos máximos e/ou mínimos de uma função.



Fonte: Costa Filho e Poppi (1999)

Portanto, um dos objetivos na otimização por simulação é identificar o valor gerado pela antecipação da informação, identificando prováveis problemas/soluções futuras, assim garantindo o atendimento ao plano/planejamento de uma maneira mais efetiva e aumentando o potencial de geração de valor da empresa (CASSEL e VACCARO, 2007).

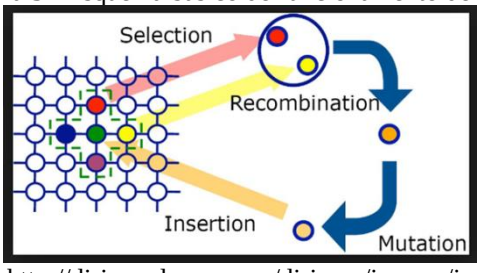
2.3 Algoritmo genético (AG)

Entre as técnicas de otimização estocástica globais, a técnica de algoritmos evolucionários, conhecida como Algoritmos Genéticos, têm encontrado muitas aplicações em vários campos da ciência e da engenharia (DAVIS, 1991).

Os Algoritmos Genéticos se inspiram no processo biológico de evolução natural de Darwin (Figura 3). Eles seguem a ideia da sobrevivência do indivíduo mais forte. Entre os animais de uma mesma espécie, aqueles que não obtêm êxito tendem provavelmente a ter um número reduzido de descendentes, tendo, portanto menor probabilidade de seus genes serem propagados ao longo de sucessivas gerações (ESMIN, 2005).

Já a combinação entre os genes dos indivíduos que perduram na espécie pode produzir um novo indivíduo muito melhor adaptado às características de seu meio ambiente, ou seja, a cada geração surgem melhores indivíduos.

Figura 3 - Esquema básico do funcionamento do AG.

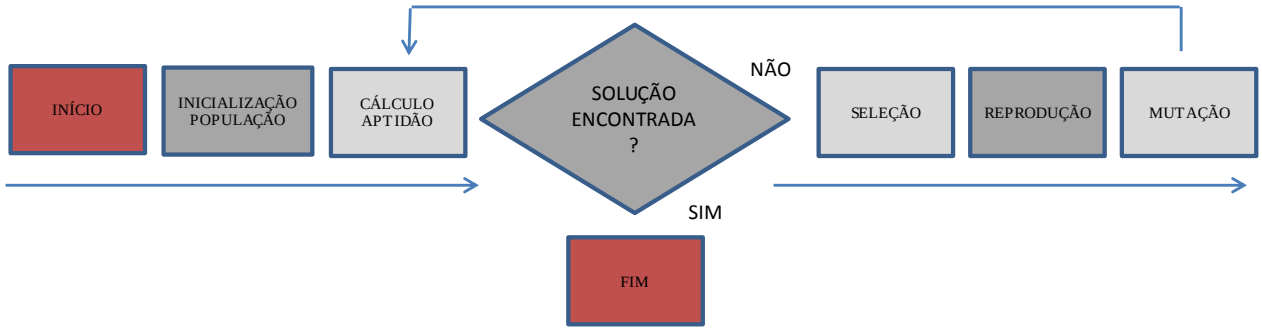


Fonte: < http://diricom.lcc.uma.es/diricom/images/jcell2.jpg>

Os Algoritmos Genéticos utilizam uma analogia direta deste fenômeno de evolução na natureza, onde cada indivíduo representa uma possível solução para um problema dado. A cada indivíduo se atribui uma pontuação de adaptação, dependendo da resposta dada ao problema por este indivíduo. Aos mais adaptados é dada uma maior oportunidade de reproduzirem-se mediante cruzamentos com outros indivíduos da população, produzindo descendentes com características de ambas as partes (PALMA-CHILLA, 2011).

Um algoritmo desenvolvido de modo adequado possui elevada probabilidade de que a população (conjunto de possíveis respostas) convergirá a uma solução ótima para o problema proposto (Figura 4). Os processos que mais contribuem para a evolução são o cruzamento, mutação e a adaptação baseada na seleção/reprodução (ESMIN, 2005).

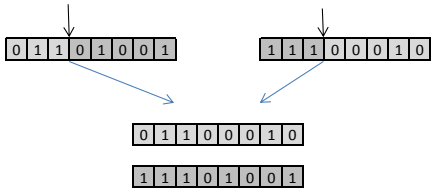
Figura 4 - Estrutura básica de funcionamento de um AG.



Fonte: Autoria própria (2014).

Em um paralelo com a natureza, os processos de Crossover (reprodução ou recombinação) envolvem dividir os *bit-strings* genômicos de dois pais em um determinado número de partes e depois emendar seções complementares de *bit-string* de cada pai para formar o genótipo do novo indivíduo (figura 5).

Figura 5 - Esquema de crossover.



Fonte: Autoria própria (2014).

Deste modo, essa ocorrência ocorre com probabilidade aleatória, envolvendo simplesmente o lançamento de pedaços na sequência (do *bit-string*) de forma estocástica.

2.4 Método enxame de partículas (MEP)

O Método Enxame de Partículas, também conhecido como Particle Swarm Optimization (PSO), simula o comportamento dos sistemas fazendo a analogia com comportamentos de uma população (um bando de pássaros, enxame de abelhas, etc.). O algoritmo mantém uma população de partículas, onde cada partícula representa uma solução potencial para um problema de otimização. A melhor posição individual da partícula representa a melhor posição que a partícula visitou e obteve a melhor avaliação (ESMIN, 2005).

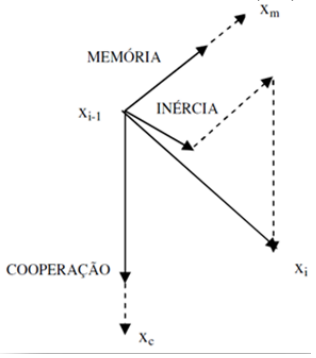
Lazzus (2009) comenta que em um sistema de PSO, cada partícula é "pilotada" por meio de um espaço de busca multidimensional, ajustando sua posição no espaço de pesquisa de acordo com sua própria experiência e a das outras partículas.

O grupo de partículas possui o conhecimento da melhor posição entre todas, que junto com a melhor posição individual é passada para as próximas gerações de partículas. Abaixo um resumo do funcionamento do método, com analogia a um bando de pássaros.

O objetivo do bando de pássaros é encontrar o melhor lugar para viver. Onda há mais alimento, por exemplo. Cada pássaro (partícula) possui seu conhecimento particular (Pbest), que é o melhor lugar que este pássaro visitou até o momento. Esse lugar é comparado com o melhor lugar dos outros pássaros. Após a comparação entre todos os pássaros define-se o melhor lugar que o bando encontrou, o Gbest.

Surge então uma nova geração de pássaros. Cada novo pássaro possui o conhecimento do Pbest, do seu pai, e também o Gbest, melhor lugar que o bando encontrou na geração passada. Esse pássaro continuará com o mesmo objetivo, encontrar o lugar onde há mais alimentos. Mas agora sua rota de voo seguirá um vetor que será a soma dos vetores que apontam para o Pbest e Gbest adquiridos da geração passada. Além disso, nesse vetor se somará outro vetor, a inércia, que impele a partícula em uma direção idêntica à que ela vinha seguindo (figura 6).

Figura 6 - Soma dos vetores Gbest (Xc), Pbest (Xm)

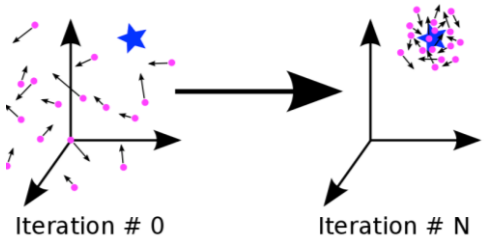


Fonte: Autoria própria (2014).

Assim o ciclo repete-se e com o passar do tempo chegamos cada vez mais próximo do melhor lugar: o resultado se torna mais preciso a cada geração.

O número de iteração (gerações de pássaros) e o número de partículas (nº de pássaros) são definidos no início da execução do algoritmo. A delimitação do espaço de busca pode ser definida no início da execução ou na implementação do algoritmo, sendo neste último caso permanente. Quanto à posição inicial das partículas, elas podem começar em lugares aleatórios ou pré-determinados, enquanto que o critério de parada é definido pelo número de iterações, como já mencionado, ou quando se atingir um resultado satisfatório (Figura 7).

Figura 7 – Resultado após N iterações as partículas chegam ao resultado satisfatório.



Fonte: Autoria própria (2014).

3 Metodologia

A primeira atividade foi verificar como seria implementado o algoritmo adaptando os métodos MEP e AG ao principal objetivo: Dado um modelo de processos encontrar a melhor configuração de tempos de forma que se consiga o menor tempo de espera entre eles, ou seja, o menor tempo possível gasto em filas.

3.1 Implementação do algoritmo de otimização

O algoritmo foi inicialmente desenvolvido em linguagem C no programa *Code::Blocks* (Anexo A), tendo como primeiro objetivo verificar a validade da combinação dos métodos adotados, o MEP e o AG. Assim, implementou-se um algoritmo base com uma matriz preenchida com números aleatórios, sendo o objetivo encontrar uma linha, entre todas as combinações de números positivos possíveis, com a menor soma. Obtendo um resultado Gbest previsível.

Na sequência são descritos os passos do funcionamento do algoritmo:

- Cria matriz com nºs aleatórios;

- Encontra Pbest - Na primeira vez serão os próprios valores de cada linha. Estes valores são armazenados em outra matriz e nas próximas iterações serão comparados com os novos valores daquela linha. A menor soma é o Pbest e se manterá;

- Encontra Gbest, a linha com a menor soma entre todas;

- Nova iteração. A matriz principal é preenchida com novos valores: cada linha é dividida em três setores, um setor receberá os números do Gbest, outro do Pbest e o último será preenchido com números aleatórios.

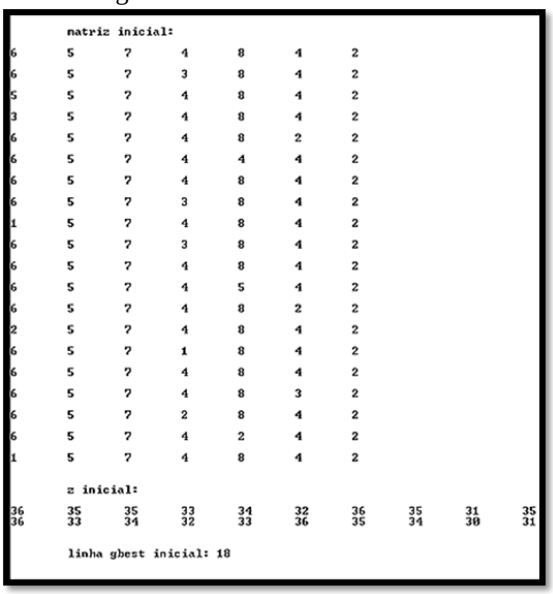
a) Cada setor é na verdade um número x de colunas da linha (varia conforme tamanho da linha). Essas colunas não precisam ser necessariamente sequenciais. Na verdade são lugares aleatórios na linha.

b) As colunas não se misturam: quando a linha receber um valor do Gbest, por exemplo, esse valor continuará na mesma coluna que estava.

- O ciclo se repete.

A figura 8, apresenta uma matriz inicial gerada aleatoriamente pelo algoritmo e o Z inicial. Cada valor de Z equivale a soma dos números da respectiva linha na matriz inicial. A linha Gbest inicialmente era a 18 (começa-se a contar do 0).

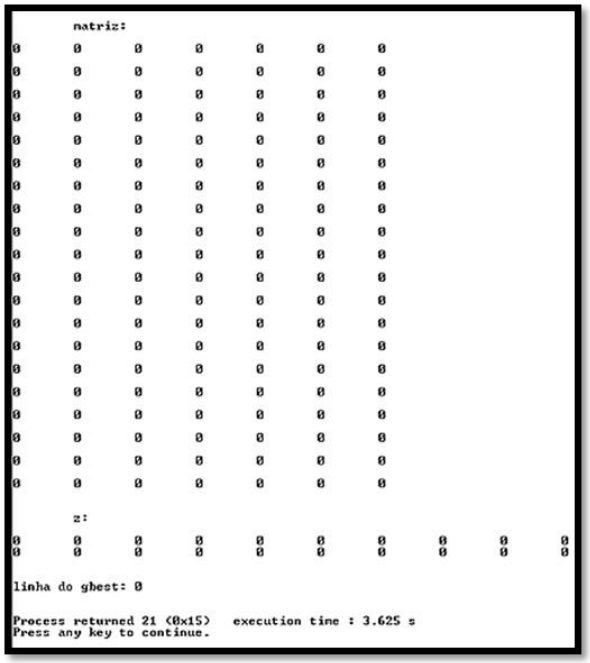
Figura 8 - Matriz inicial e z inicial.



Fonte: Autoria própria (2014).

Como esperávamos o Gbest encontrado com uma matriz 20 x 7 após 25 iterações (Figura 9) foi [0,0,0,0,0,0,0]. Vale salientar que com 20 iterações a matriz estava com apenas alguns campos não zerados.

Figura 9 - Matriz após 25 iterações.

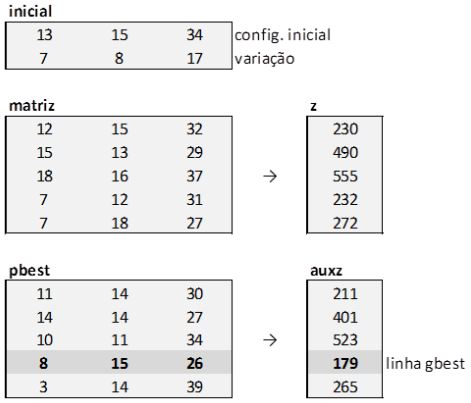


Fonte: Autoria própria (2014).

Após, começou-se as mudanças no algoritmo para atender ao objetivo de minimização de filas. A principal mudança é que, se antes o Gbest e o Pbest eram influenciados diretamente pelos valores das linhas, agora eles serão influenciados indiretamente.

Os valores de cada linha serão importados no modelo dentro do programa de simulação e este gerará um relatório onde apresentará os tempos gastos em filas em cada processo. A soma destes tempos é que será analisada para se encontrar o menor valor. A figura 10 apresenta as principais matrizes utilizadas no algoritmo.

Figura 10 - Principais matrizes utilizadas no algoritmo.



Fonte: Autoria própria (2014).

Salienta-se que agora o valor de z gerado por cada linha não tem mais relação direta com a mesma.

O código foi implementado para trabalhar apenas com nºs inteiros e também foram utilizadas duas bibliotecas especiais:

- #include <time.h>

Para melhorar a aleatoriedade dos números que é deficiente apenas com a função rand(), utilizou-se a função srand (time(NULL)). Esta função alimentanda a rand() com um número diferente a cada execução graças ao time(NULL). Este último é o tempo em segundos do computador encontrado a partir da biblioteca <time.h>.

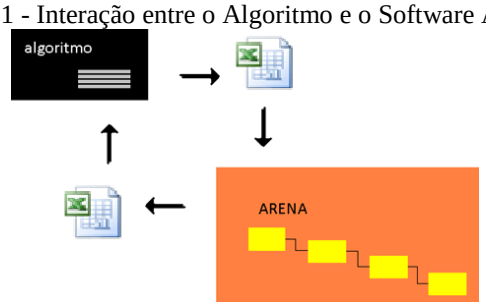
- #include "libxl.h"

Arquivos do Excel fazem a intermediação de dados entre o algoritmo e o software de simulação ARENA. A linguagem C não permite originalmente trabalhar com este tipo de arquivo. Agregou-se esta extensão ao programa através da biblioteca "libxl.h". Sua instalação e utilização são complexas mas facilitou a intermediação de dados entre os programas, além de poder abrir uma série de funcionalidades em C.

4 Resultados e discussões

Esquematicamente, o algoritmo tem seu funcionamento como demonstrado na Figura 11.

Figura 11 - Interação entre o Algoritmo e o Software ARENA.



Fonte: Autoria própria (2014).

E este funcionamento obedece a certas condições (etapas) como:

- Construção do modelo de simulação no programa ARENA onde atuará o algoritmo.

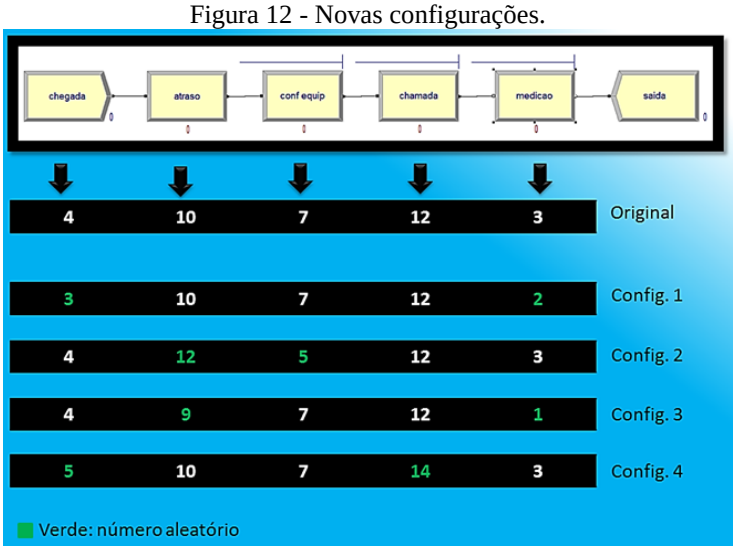
- Ajuste do algoritmo para funcionar com o modelo (define-se o nº de LINHAS e COLUNAS para dimensionar as matrizes utilizadas). Define-se também o critério de parada (nº de ITERAÇÕES) e a delimitação do espaço de busca (chamado de PROPORÇÃO, pois será a porcentagem que o valor do processo poderá variar do seu valor inteiro inicial).

- A partir do modelo de simulação no ARENA é exportado um arquivo ‘modelo.xls’ (formato Microsoft Excel) onde conterá todas as informações deste modelo. O arquivo deve ser exportado para a pasta em que se encontra o executável do algoritmo.

- Execução do algoritmo, leitura do arquivo e retirada das informações necessárias. Essas informações serão os tempos gastos em cada processo do modelo de simulação. Os tempos serão lidos a partir do modelo.xls e transferidos para a matriz dentro do algoritmo.

- Geração de novas configurações pelo algoritmo para o modelo a partir da configuração original. A configuração original é copiada para as outras linhas e em colunas aleatórias são inseridos números aleatórios (mas de acordo com a PROPORÇÃO). Este passo é visualizado na figura 12.

Figura 12 - Novas configurações.



Fonte: Autoria própria (2014).

- O algoritmo executa/abre o arquivo modelo.xls e modifica os valores dos tempos de cada processo para uma nova configuração gerada.

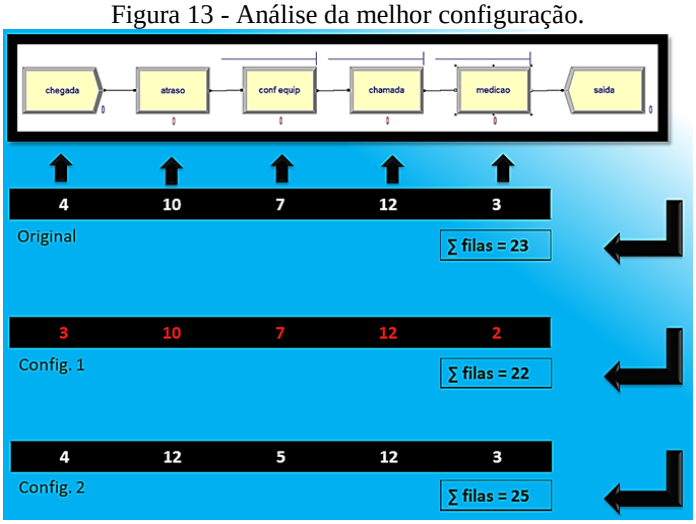
- No programa ARENA, o arquivo modelo.xls é importado e simulado gerando outro arquivo, o 'relatório.xls', ao final do processo. Neste arquivo aparecerão os tempos de espera nas filas de cada processo do modelo.

- O algoritmo lê o arquivo relatório.xls e faz o somatório dos tempos das filas. O resultado é associado à configuração que o deu origem.

- O arquivo modelo.xls é modificado com outra configuração, e o ciclo repete-se (ARENA importa, simula, gera relatório.xls, o arquivo é lido e a soma dos tempos das filas é associada à configuração).

- Após todas as configurações serem exportadas o algoritmo analisa a configuração que teve o melhor resultado, ou seja, são comparados os somatórios das filas que todas as configurações geraram em busca do menor número. A configuração que gerou este número será o Gbest. (em vermelho na figura 13).

Figura 13 - Análise da melhor configuração.



Fonte: Autoria própria (2014).

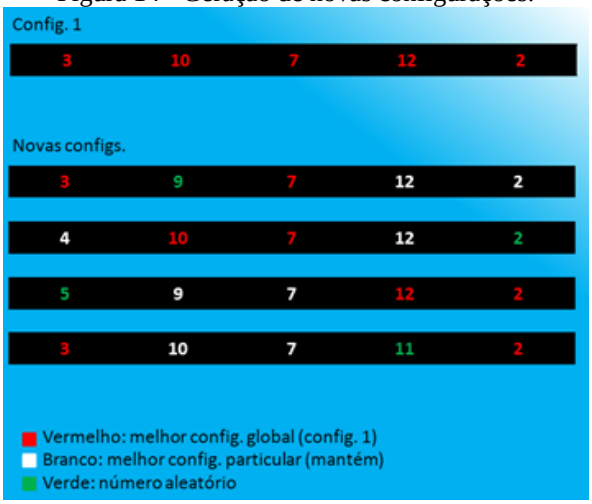
- Inicia-se outra iteração e novas configurações são criadas a partir de três bases:

a) o Gbest, encontrado na iteração anterior;

b) o Pbest, que no começo será a própria configuração da iteração anterior. Após a primeira iteração, o Pbest será a melhor configuração particular entre todas as iterações anteriores, a melhor configuração que esse vetor, linha ou partícula conseguiu até o momento;

c) e nºs aleatórios. Os campos restantes das novas configurações são preenchidos com números aleatórios (definidos pela PROPORÇÃO) para garantir que o algoritmo explore a maior variedade possível de configurações e não se estagne, gerando configurações repetidas (Figura 14).

Figura 14 - Geração de novas configurações.



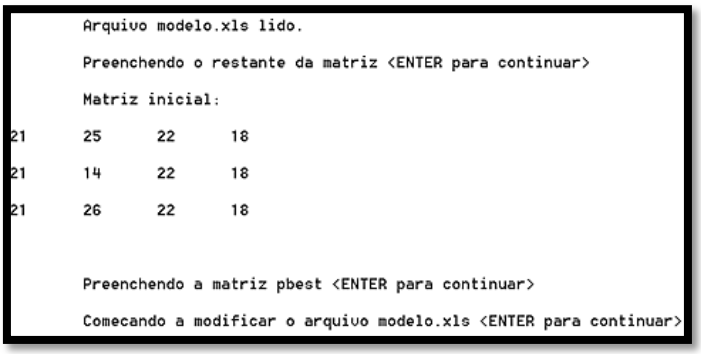
Fonte: Autoria própria (2014).

- Recomeço de todo o ciclo. Cada nova configuração é exportada e gera após a simulação um novo somatório de tempos nas filas.

- Quando atingir o número de ITERAÇÕES definidas no início da execução do algoritmo, este será finalizado e mostrará o Gbest, ou seja, a melhor configuração para o nosso modelo de simulação.

A figura 15 apresenta o layout de visualização da execução do algoritmo aplicado a um exemplo.

Figura 15 - Visualização da execução do algoritmo.



Fonte: Autoria própria (2014).

Assim, durante a elaboração e execução do algoritmo algumas dificuldades em linguagem C surgiram, como o processo de exportar e depois importar os arquivos Excel dentro do ARENA, o qual não era automático, precisando ser feito manualmente. Portanto era preciso importar cada linha da matriz a cada iteração, além de exportar o arquivo de relatório a ser lido pelo algoritmo a cada simulação. Desta maneira tornando o processo inviável para um número elevado de iterações.

A princípio pensou-se como solução criar um arquivo .bat para automatizar tarefas mas o processo se tornaria lento e complexo visto que já estaríamos trabalhando com três programas distintos. Neste caso, como o programa ARENA possui integrado um editor e copilador de linguagem *Visual Basic for Applications* (VBA) será possível trabalhar diretamente entre código e simulação, deste modo optou-se por traduzir, posteriormente, o algoritmo para VBA tornando-o melhor otimizado, pois trabalhará como complemento do software Arena.

Como resultado obteve-se um algoritmo para otimização de processos baseado no Método Enxame de Partículas e em Algoritmos Genéticos, que através de seus conceitos tornaram possível o desenvolvimento de um novo método adaptado (algoritmo) ao nosso objetivo, minimizar o tempo das filas de espera em uma cadeia de processos.

5 Conclusões

Após toda a abordagem realizada, fica evidenciado que algoritmos de otimização se caracterizam como ferramentas úteis aos gestores organizacionais, pois tornam os processos de

análise mais dinâmicos e automáticos, facilitando o trabalho dos mesmos e economizando tempo nos processos decisórios.

O desenvolvimento deste algoritmo contempla a possibilidade de estudos e aplicações futuras em modelos reais de processos, com resultados analisados e verificados integralmente no contexto das organizações. Como limitações, tem-se que a minimização de tempos em filas necessariamente não gera um *number out* (saídas do modelo computacional) maximizado, porém considera-se este o primeiro passo na elaboração de uma programação mais elaborada e abrangente.

6 Agradecimentos

Agradecimentos ao auxílio disponibilizado pela CAPES, FAPERGS e pelo PPGSPI (UNISC).

Referências

AZADIVAR, F. Simulation Optimization Methodologies. **Proceedings of the 1999 Winter Simulation Conference**, p.93-100, 1999.

BAKHTADZE, N.N. Virtual analyzers: Identification approach. **Automation and Remote Control**, v. 65, n 11, p. 1691-1709, 2004.

BANKS, J.; CARSON II, J. S.; NELSON, B. L. & NICOL, D. M. **Discrete event system simulation**. 4rd Ed. New Jersey: Pearson Prentice Hall, 2005.

BOWDEN, R.; HALL, J. Simulation Optimization Research and Development. **Proceedings of the 1998 Winter Simulation Conference**. p.1693-1698, 1998.

CASSEL, G. L.; VACCARO, G. L. R. Aplicação de Simulação Otimização para Definição do Mix Ótimo de Produção de uma Indústria Metal-Mecânica. In: XXVII Encontro Nacional de Engenharia – ENEGEG. Foz do Iguaçu. **Anais.** Foz do Iguaçu. 2007.

CHUNG, C.A. **Simulation modeling handbook: a practical approach**. Industrial and Manufacturing Engineering Series. Washington: CRC Press, 2004.

CHWIF, L.; MEDINA, A. C. **Modelagem e Simulação de Eventos Discretos: Teoria e prática**. 3. ed. São Paulo: Leonardo Chwif. 2010.

COSTA, F. M. **Construção de modelo de simulação de sistema puxado de produção para melhorias de eficiência**. Dissertação (Mestrado) - Universidade do Minho, Escola de Engenharia. Guimarães, 2011.

COSTA FILHO, P. A.; POPPI, R. J. Algoritmo genético em química. **Quím. Nova**, São Paulo , v. 22, n. 3, Jun. 1999.

DAVIS, L. **Handbook of Genetic Algorithms.** Van Nostrand Reinhold, New York, 1991.

DIEHL, F.C.; LUSA, L.P.; SECCHI A.R.; MUNIZ, L.A.R.; LONGHI, L.G.S. Simulação Operacional de uma Torre de Destilação Atmosférica via Aspen Plus e Avaliação de Modelos de Analisadores Virtuais. **Revista Controle & Automação.** v.20, n.3/Julho, Agosto e Setembro, 2009.

ESMIN, A. A. A. **Estudo de Aplicação do Algoritmo de Otimização por Enxame de Partícula na Resolução de Problemas de Otimização Ligados ao SEP**. Programa de Pós-Graduação em Engenharia Elétrica, Universidade Federal de Itajubá, Itajubá – MG.

HARREL, C. R.; GHOSH, B.K.; BOWDEN, R. **Simulation Using ProModel®.** McGraw-Hill, 2000.

LAZZUS, J.A. **J. Eng. Thermophys.** Vol 18. 2009.

MORABITO, R.; PUREZA, V. Modelagem e simulação. In: CAUCHICK MIGUEL, P.A.C. et al. **Metodologia de pesquisa em engenharia de produção e gestão de operações.** Rio de Janeiro: Elsevier, p.165-192, 2010.

PALMA-CHILLA, L.; LAZZUS, J.A.;PONCE, A.A.P. **J. Eng. Thermophys**. Vol 20. 2011.

SAKURADA, N.; MIYAKE, D. I. Aplicação de simuladores de eventos discretos no processo de modelagem de sistemas de operações de serviços. **Gestão & Produção.** São Carlos, v. 16, n. 1, p. 25-43, 2009.

SILVA, A. K. **Método para avaliação e seleção de softwares de simulação de eventos discretos aplicados à análise de sistemas logísticos.** 212p. Dissertação (Mestrado) – Escola Politécnica da Universidade de São Paulo, São Paulo, SP. 2006.

UM, I.; HYEONJAE, C.; LEE, H. The simulation design and analysis of a Flexible Manufacturing System with Automated Guided Vehicle System. **Journal of Manufacturing Systems**. v.28, p. 115-122, 2009.

Recebido: 11/04/2014

Aprovado: 17/05/2015