IMPLEMENTAÇÃO DE UM ALGORITMO GENÉTICO BASEADO EM XML PARA A CALIBRAÇÃO DO *TANK MODEL*

Amílcar Soares Júnior[1]; Camilo A. S. de Farias[2]; Celso Augusto Guimarães Santos[3] & Koichi Suzuki[4]

RESUMO --- Problemas de otimização são comuns em diversos campos da ciência. Varias técnicas já foram propostas para resolver tais problemas, incluindo o uso de Algoritmos Genéticos (AG). O objetivo deste trabalho é apresentar a implementação de um AG configurável e portável que utiliza XML (*eXtensible Markup Language*) para descrever dados de entrada e saída em um problema de otimização. A ferramenta é aplicada para a calibração do modelo hidrológico chuva-vazão conhecido como *tank model* com o objetivo de estimar vazões diárias para o reservatório do rio Ishite, que abastece a cidade Matsuyama, Japão. O principio básico do *tank model* consiste em representar uma bacia hidrográfica como uma série de tanques nas quais as vazões de saída de cada tanque são proporcionais à altura da água no respectivo reservatório. A calibração e validação dos resultados mostram que a implementação baseada em XML do AG foi bastante eficiente para definir os parâmetros do *tank model*. Uma vez que qualquer aplicação ou plataforma seja capaz de processar dados em XML, essa ferramenta pode ser uma alternativa importante para a resolução de problemas relacionados a recursos hídricos.

ABSTRACT --- In many different fields of science, we are faced with optimization problems. Several approaches have been proposed to solve such difficulties, including the use of Genetic Algorithms (GA). This paper aims at presenting the implementation of a configurable and portable GA that uses the eXtensible Markup Language (XML) to describe input and output data of an optimization problem. The tool is applied for the calibration of the rainfall-runoff tank model in order to estimate daily inflows to the Ishite river Dam, which is the reservoir that supplies water to Matsuyama city, Japan. The basic principle of the tank model consists of representing the river basin as a set of tanks in which the outflows of each tank are proportional to the water height from the respective outlets. The calibration and validation results show that the XML-based GA tool was very efficient for defining the tank model parameters. Since any application or platform capable of processing XML can utilize this tool, it may be an important alternative for solving water resources problems.

Palavras-chave: XML, algoritmos genéticos, *tank model*.

1) Ehime University – 790-8577 Matsuyama, Ehime, Japão. Tel: +81-89-9279831 – e-mail: amilcarsj@gmail.com
2) Ehime University – e-mail: camiloallyson@yahoo.com.br
3) Universidade Federal da Paraíba – Centro de Tecnologia/DECA – 58051-900 João Pessoa – PB - e-mail: celso@ct.ufpb.br
4) Ehime University – e-mail: ksuzuki@dpc.ehime-u.ac.jp

1 – INTRODUÇÃO

Otimização é um problema bastante comum em muitos campos da ciência. Otimizar uma função matemática corresponde à procura do valor máximo ou mínimo que essa função pode assumir. Essas funções podem também possuir uma serie de restrições às variáveis a serem otimizadas. Muitas técnicas já foram propostas para encontrar tais valores. Entretanto, as maiorias dessas técnicas tradicionais não são muito eficientes para resolver problemas de otimização não-lineares. Os Algoritmos Evolucionários (AEs), definidos como um conjunto de métodos probabilísticos baseados na teoria da evolução de Charles Darwin aparecem para lidar com problemas complexos de otimização. O comportamento das espécies observado por Darwin é simulado computacionalmente com o objetivo de obter valores otimizados para parâmetros. Uma das implementações de AEs, chamada de Algoritmos Genéticos (AGs), foi desenvolvida por John Holland (Holland, 1975). Desde esse primeiro trabalho, muitas implementações de GAs foram desenvolvidas. Entretanto a maioria delas possui objetivos muito específicos (Celeste *et al.*, 2004; Franchini, 1996; Santos, 2003a; Wang, 1997). No presente trabalho, uma implementação de um AG é feita para resolver qualquer tipo de problema no qual tal técnica possa ser aplicada.

A adoção da linguagem de programação Java® (*Sun Microsystems*), do eXtensible Markup Language (XML) e de algumas técnicas de programação permite o desenvolvimento de uma implementação de um AG portável, configurável e de propósito geral. Uma simples aplicação dessa ferramenta baseada em XML pode ser vista em Soares Junior (2009). Entretanto, para mostrar o quão flexível essa ferramenta é, este trabalho utiliza o modelo chuva-vazão *tank model* (Sugawara, 1979) para estimar vazões diárias no reservatório do rio Ishite, o qual abastece a cidade de Matsuyama, capital do Estado de Ehime, Japão. Esse artigo é dividido em sete seções. Na primeira foi feita uma breve introdução do que se trata o presente trabalho. Na segunda seção é feita uma revisão a respeito dos AGs e aborda alguns tópicos da implementação do AG baseado em XML. A terceira seção possui informações a respeito das ferramentas utilizadas no desenvolvimento da implementação do AG. Na quarta seção, o funcionamento do *tank model* é explicado. A quinta seção contém detalhes a respeito de como a otimização foi realizada. Na sexta seção, o resultado da otimização é mostrado, e na sétima e última seção, são mostradas algumas conclusões a respeito deste trabalho.

2 – ALGORITMOS GENÉTICOS

Os AGs são métodos de otimização baseados na teoria da evolução de Charles Darwin de 1859 no livro *A Origem das Espécies*. Isso significa que utilizando tal teoria, quanto mais um individual consegue se adaptar ao ambiente, maior é a possibilidade deste sobreviver e gerar descendentes.

De acordo com Michalewicz (1999), um AG deve conter:

- Uma representação genética para as soluções.
- Uma forma de criar a população inicial de soluções.
- Uma função de avaliação para que seja atribuída uma aptidão a cada solução.
- Operadores genéticos que alterem a composição genética dos descendentes gerados.
- Valores para os vários parâmetros usados nos algoritmos genéticos (tamanho da população, taxa de *crossover*, taxa de mutação, etc).

Um conjunto de soluções é denominado população. Indivíduos chamados de cromossomos, que representam cada solução possível para o problema, compõem uma população. Cada solução é avaliada gerando um valor de aptidão que é usado para selecionar os melhores indivíduos.

Como pode ser visto na Figura 1, um AG deve gerar inicialmente uma população de indivíduos, e depois disto, a primeira população de indivíduos é analisada. A aptidão de cada indivíduo é determinada, ou seja, o quanto tal solução proposta é relativamente melhor quando comparada com os outros indivíduos. Assim como na teoria de Darwin, os indivíduos mais fortes possuem uma maior chance de serem selecionados para gerarem descendentes. Enquanto a condição de parada não for atingida, o algoritmo cria novas populações de indivíduos. Através da função *Alterar População*, as técnicas de *crossover* e mutação são aplicadas nos indivíduos selecionados da população. Essas técnicas alteram os valores dos parâmetros e são os conceitos chave que fazem dos AGs uma técnica de otimização bastante poderosa.

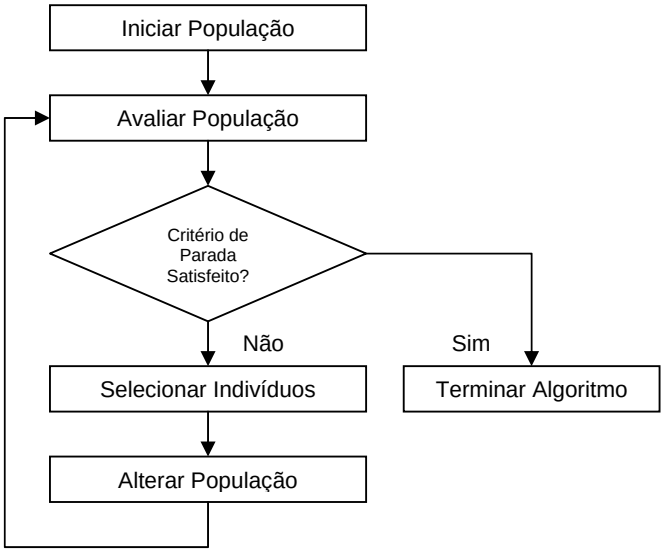

Figura 1 – Descrição geral de um AG

O primeiro passo para a implementação de um AG é gerar uma população inicial de indivíduos. Nessa implementação, duas técnicas podem ser empregadas. O usuário pode escolher uma inicialização aleatória de valores ou em grade para os parâmetros. Quando a geração aleatória de valores é escolhida, assim como o nome sugere, uma série de valores aleatórios para os parâmetros é gerado pela ferramenta. Quando é escolhida uma inicialização em grade como forma de iniciar o algoritmo, os valores gerados são igualmente espaçados no espaço de busca.

Após a atribuição de valores para a população inicial, essa primeira população é avaliada. Uma das principais vantagens dessa implementação de AG é a de que várias formas de avaliação podem ser criadas. Para gerar uma forma de avaliação, usuário precisa apenas estender a *superclasse* PopulationEvaluationMethod. Essa classe possui métodos abstratos que devem ser implementados novamente quando uma nova forma de avaliação é criada. A **Figura 2** mostra um diagrama UML (*Unified Modeling Language*). Usando XML, pode-se construir diagramas que representam conceitos do sistema, atividades e interações com o usuário. Classes do sistema e objetos podem ser representados também por tal linguagem de modelagem (Larman, 2007).

Como pode ser visto, se o usuário não puder descrever o problema como uma função matemática, apenas é necessária a criação de uma nova forma de avaliação a partir da extensão da classe PopulationEvaluationMethod. Essa técnica foi utilizada na otimização do *tank model* aqui relatada. As operações evaluatePopulation e

isAValidSolution devem então ser implementadas novamente da forma que o usuário precise utilizá-las na otimização. O primeiro método consiste em avaliar cada indivíduo e atribuir uma aptidão para este. O segundo método consiste em verificar se os valores para os parâmetros de cada indivíduo estão de acordo com as restrições do problema. Entretanto, se o problema puder ser descrito como uma única função matemática, o usuário pode utilizar a classe JEPEvaluationMethod. Essa classe avalia a função matemática configurada no arquivo XML e calcula o seu resultado. Mais detalhes sobre a biblioteca Java Math Expression Parser (JEP) pode ser vista na seção 3.4.

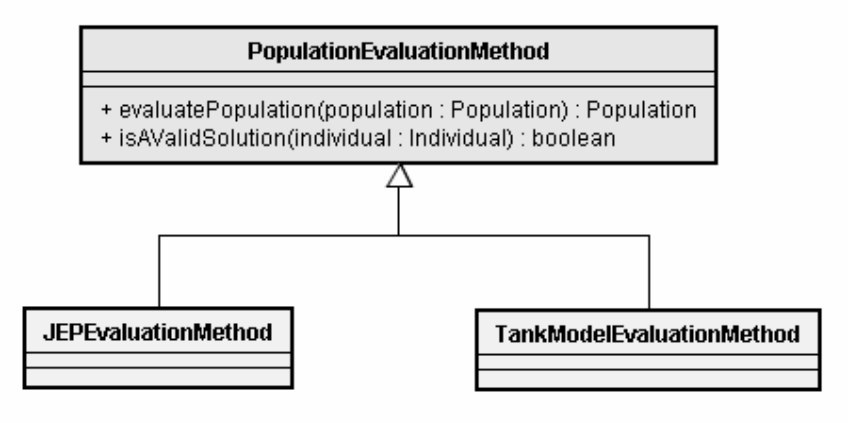

Figura 2 – Diagrama UML para a classe de avaliação do AG

A cada geração é necessária a avaliação de um critério de parada. Na ferramenta aqui exposta existem três formas de parar o AG: parar quando o valor ótimo for atingido; parar após um número determinado de gerações; parar quando os critérios de convergência forem satisfeitos. Neste último caso, o algoritmo pode ser parado quando existir uma grande quantidade de indivíduos iguais (representando a melhor solução) ou quando após certa quantidade de gerações não houver mudança no melhor indivíduo.

Outro fator que deve ser levado em questão na implementação de um AG é a forma de seleção do melhores indivíduos de uma dada geração. Nesta implementação, o usuário pode escolher entre o algoritmo da roleta ou uma seleção por torneio. Na primeira, o indivíduo que possuir a melhor aptidão recebe a maior probabilidade de ser selecionado e gerar descendentes. O segundo melhor individuo receberá uma probabilidade menor que o primeiro e assim sucessivamente. Na seleção por torneio, três indivíduos são sorteados aleatoriamente e em seguida o que possuir a melhor aptidão é então escolhido para gerar descendentes.

Como já mencionado, as operações de *crossover* e mutação são os conceitos chave de um AG. Essas operações possuem uma taxa de ocorrência que deve ser levada em consideração antes de serem aplicadas. Quando essas operações são utilizadas na execução de uma otimização, é gerado um valor aleatório a ser comparado com essa taxa. Se esse valor for menor ou igual a essa taxa, tal operação é então executada. Entretanto, se esse número for maior que a taxa, a operação não é aplicada ao indivíduo.

Em muitas aplicações que utilizam AGs, a forma mais comum de codificação é a utilização de *strings* binárias de comprimento fixo. A razão para esta utilização é que na teoria em que os AGs foram criados esta representação foi utilizada.

Na operação de *crossover*, assim como na natureza, partes do número (*bits*) são trocados entre dois indivíduos gerando um descendente diferente dos originais. A Figura 3 mostra como essa operação é realizada. Primeiro, uma quantidade de *bits* é escolhida para que eles sejam trocados entre as soluções. Neste caso, quatro *bits* foram escolhidos para serem trocados entre os indivíduos. Assim como na genética, as partes trocadas geram dois indivíduos diferentes dos originais.

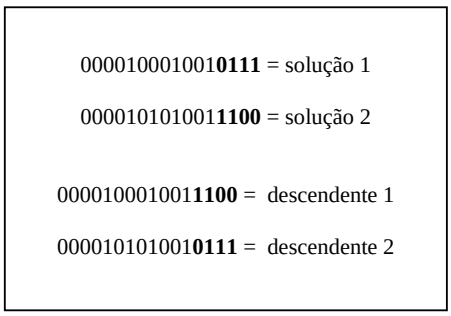

Figura 3 - Operação de *crossover*

Na operação de mutação, cada *bit* do descendente gerado pelo *crossover* possui uma pequena probabilidade de ser mudado. Isso significa que se a mutação ocorrer em um *bit*, um valor que era 0 (zero) se transformará em 1 (um) e vice-versa. A Figura 4 mostra como essa operação é executada. Essa operação é aplicada a cada *bit* do descendente gerado pelo *crossover* quando esta operação estiver ativada.

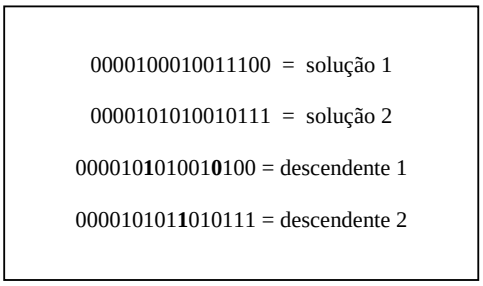

Figura 4 - Operação de mutação

A utilização da representação binária, como pôde ser visto, é muito simples para que a teoria dos AGs seja analisada. Entretanto, se o problema a ser resolvido possui parâmetros contínuos e o usuário procura uma boa precisão numérica, essa abordagem possui alguns problemas. Para adicionar mais um número de precisão é necessária a adição de 3,3 *bits* na solução. Quando existem muitos parâmetros para serem otimizados, a convergência do AG se torna um problema porque o algoritmo começa a convergir lentamente.

Para resolver esse tipo de problema, a codificação de números de ponto flutuante foi utilizada. Essa abordagem gera um número menor de cromossomos, além de ser mais facilmente legível. Cada cromossomo é representado como um número real (e.g., 34,15). Alguns experimentos comparando essas duas abordagens são mostrados em Janikow *et al.* (1991) e na maioria dos casos a codificação com números de ponto flutuante foi mais efetiva e rápida.

3 – FERRAMENTAS UTILIZADAS

Atualmente a utilização de AGs como um método de otimização é muito comum na engenharia. Entretanto, muitas dessas implementações de AGs são aplicadas a casos específicos. O sistema aqui proposto foi modelado para ser genérico e resolver qualquer problema onde os AGs podem ser aplicados.

A Figura 5 mostra como o sistema funciona. O usuário primeiro seleciona as opções que ele deseja utilizar na otimização (parâmetros, taxa de *crossover*, quantidade de indivíduos em cada geração, etc.). O sistema carrega essas opções desejadas pelo usuário (configuração), executa a otimização e cria um arquivo com os resultados.

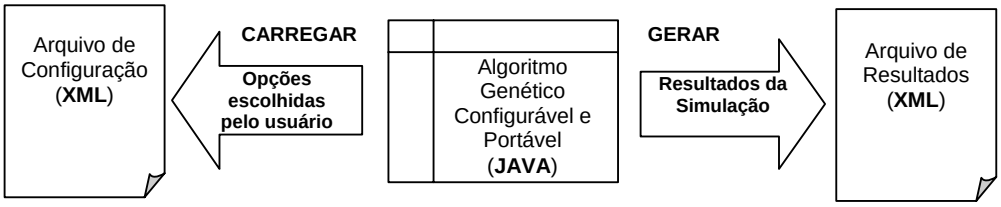

Figura 5- Sistema proposto

3.1 - Linguagem de programação Java

A linguagem de programação Java foi desenvolvida nos anos 90 pela *Sun Microsystems*. Ela é uma linguagem de programação orientada a objetos diferente das linguagens de programação comuns devido a utilização de *bytecodes* (linguagem intermediária).

Numa linguagem de programação convencional, o código fonte é compilado para o código da máquina nativa. Diferente dessa abordagem, o código de fonte da aplicação Java é compilado para esses *bytecodes* e são posteriormente executados em uma máquina virtual. A máquina virtual Java é um programa que carrega e executa o aplicativo Java, convertendo esses *bytecodes* para a linguagem de máquina que está rodando o aplicativo. Essa abordagem permite a portabilidade de qualquer aplicativo desenvolvido nessa plataforma, sendo assim independente do Sistema Operacional que está sendo utilizado pelo usuário.

3.2 - eXtensible Markup Language (XML)

A linguagem XML foi desenvolvida para representar dados flexíveis e extensíveis (Graves, 2003). Os elementos e atributos descritos nessa linguagem proporcionam informações a respeito dos dados, podendo então ser manipulados por aplicativos diferentes e modificados de acordo com as necessidades emergentes.

Essa tecnologia permite que o sistema desenvolvido possa ser configurado antes de uma execução. Ao realizar uma otimização, como pode ser visto na Figura 5, o usuário deve criar um arquivo de configuração com as opções que devem ser utilizadas. Outra vantagem desta ferramenta é a utilização de XML nos dados de saída. O desenvolvimento da ferramenta usando tal abordagem permite que os dados de saída sejam facilmente lidos pelo usuário ou qualquer outra entidade. Um exemplo de outra entidade seria um software escrito em uma linguagem de programação diferente (sistema heterogêneo).

3.3 - XStream

O XStream (*http://xstream.codehaus.org/index.html*) é uma biblioteca para serialização de objetos para XML, que faz operações inversas, gerando *tags* XML de fácil interpretação. Neste trabalho, os arquivos de entrada e saída são documentos XML. O propósito de usar tal biblioteca nesta implementação foi o fato de que não foi necessária a implementação de um tradutor para XML.

3.4 - Java Math Expression Parser 2.4.1 (JEP)

O JEP é uma biblioteca desenvolvida pela *Singular Systems* (*http://www.singularsys.com/jep/*) para traduzir e avaliar funções matemáticas. Essa biblioteca suporta variáveis, restrições e funções. Um número bastante expressivo de funções matemáticas e constantes é incluído, *e.g.* seno, cosseno e tangente.

4 – O MODELO *TANK MODEL*

O processo cíclico envolvido na transferência do conteúdo úmido do mar para a terra é conhecido como ciclo hidrológico. Os modelos chuva-vazão são usados freqüentemente por profissionais da área de recursos hídricos para gerar séries sintéticas de vazão em pontos de uma bacia hidrográfica. Esses modelos são usados também para estender séries observadas, comumente interrompidas ou muito pequenas para permitir estudos estatísticos.

O *tank model* pode ser chamado de um sistema de vazão com armazenamento, composto por uma série de tanques. A bacia é representada por um conjunto de tanques, e a saída ou escoamento de cada tanque é assumido como sendo proporcional à altura da água a partir da posição da descarga ou infiltração. A profundidade do tanque é assumida como sendo o armazenamento na bacia.

Neste trabalho, o *tank model* foi implementado com três tanques, e a **Figura 6** mostra como o mesmo foi implementado.

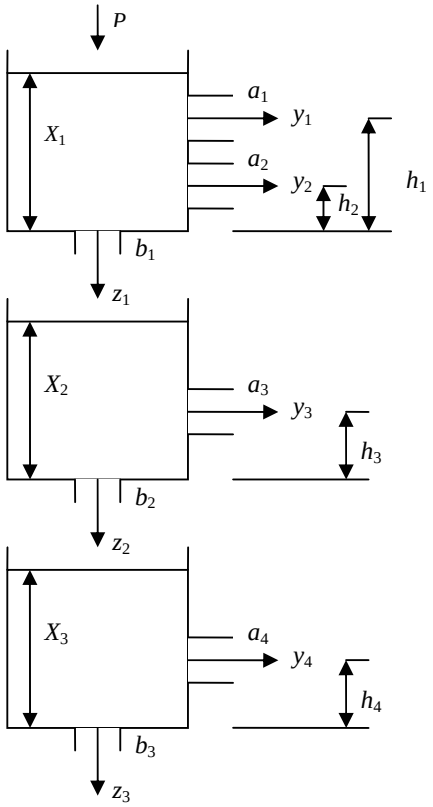

Figura 6 – O modelo *tank model* com três tanques

Os parâmetros a serem otimizados são as vazões a_1, a_2, a_3 e a_4; as infiltrações b_1, b_2 e b_3; e a altura do escoamento h_1, h_2, h_3 e h_4. Esses valores e o *tank model* são definidos pelas seguintes expressões:

$$y_1(t) = a_1[X_1(t) - h_1] \quad (1)$$

$$y_2(t) = a_2[X_1(t) - h_2] \quad (2)$$

$$y_3(t) = a_3[X_2(t) - h_3] \quad (3)$$

$$y_4(t) = a_4[X_3(t) - h_4] \quad (4)$$

$$z_1(t) = b_1 X_1(t) \quad (5)$$

$$z_2(t) = b_2 X_2(t) \quad (6)$$

$$z_3(t) = b_3 X_3(t) \quad (7)$$

$$Q(t) = y_1(t) + y_2(t) + y_3(t) + y_4(t) \quad (8)$$

$$X_1(t) = X_1(t-1) + P(t) - y_1(t) - y_2(t) - z_1(t) \quad (9)$$

$$X_2(t) = X_2(t-1) + z_1(t) - y_3(t) - z_2(t) \quad (10)$$

$$X_3(t) = X_3(t-1) + z_2(t) - y_4(t) - z_3(t) \quad (11)$$

onde t é dado em dias; $y_1(t)$, $y_2(t)$, $y_3(t)$ e $y_4(t)$ são as vazões estabelecidas no dia t; $z_1(t)$, $z_2(t)$ e $z_3(t)$ são os valores de infiltração em cada tanque no dia t; $X_1(t)$, $X_2(t)$ e $X_3(t)$ são a profundidade de armazenamento no dia t; $Q(t)$ é o total da vazão no dia t; e $P(t)$ é a precipitação no dia t.

5 – EXECUÇÃO DA OTIMIZAÇÃO

O processo de calibração consiste em achar um conjunto de valores para os parâmetros de forma que estes valores possam atingir o ótimo global. Os parâmetros a serem otimizados nesse problema são a_1, a_2, a_3, a_4, b_1, b_2, b_3, h_1, h_2, h_3 e h_4. A faixa de variação dos valores que essas variáveis podem assumir é mostrada na Tabela 1.

Tabela 1 - Faixa de variação dos parâmetros para a simulação.

	a_1	a_2	a_3	a_4	b_1	b_2	b_3	h_1	h_2	h_3	h_4
Limite inferior	0,001	0,001	0,01	0,01	0,15	0,01	0,0	10	10	10	10
Limite superior	0,1	0,1	0,3	0,7	0,9	0,04	0,3	90	120	120	120

Com o objetivo de executar a otimização do modelo, foi necessária a seleção de uma função objetivo, que para o presente estudo foi escolhida a seguinte função:

$$minimizar \sum_{t=1}^{N} \frac{|Q_o - Q_c|}{Q_o} \tag{12}$$

onde Q_o é a vazão observada, Q_c é a vazão calculada pelo modelo hidrológico, e N é o número de dias utilizado na calibração dos parâmetros para o modelo.

A **Figura 7** mostra como a ferramenta foi utilizada com o objetivo de otimizar os parâmetros. Como mencionado na seção 2, para avaliar o conjunto de parâmetros que estão sendo otimizados, a classe PopulationEvaluationMethod foi estendida. Nessa nova classe gerada, primeiramente os dados de precipitação para calibração foram lidos. Depois, a implementação do AG gera valores para os parâmetros a serem otimizados. Com esses dois conjuntos de valores, o modelo pode ser executado. Após a execução do modelo hidrológico, as vazões são geradas e a função objetivo pode ser calculada, gerando assim a aptidão desta possível solução. Esse procedimento é repetido para cada indivíduo da população no AG. Esse procedimento aqui mencionado é similar ao mostrado em Santos *et al.* (2003b).

Nessa simulação, os dados de vazão do reservatório do rio Ishite foram utilizados. Esse reservatório abastece a cidade de Matsuyama no Japão. O período utilizado para a calibração foi de 1992 a 1993, enquanto os anos de 1994 a 1995 foram utilizados para a validação do *tank model* calibrado.

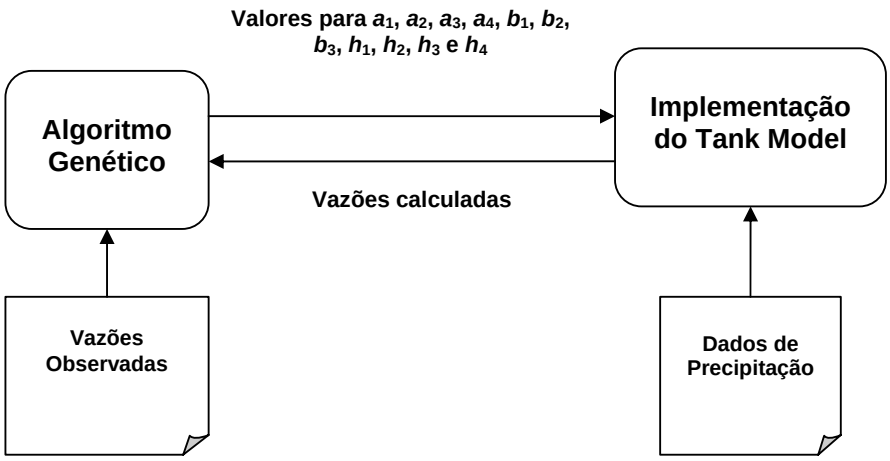

Figura 7 – Execução da simulação

6 – RESULTADOS

Os índices estatísticos de correlação (r) e erro do tipo *bias* (B) foram utilizados como critério para avaliação do desempenho da calibração do *tank model*. A correlação calcula a variabilidade de um número de previsões ao redor do valor verdadeiro. O erro do tipo *bias*, por outro lado, é a medida do erro sistemático e, assim, calcula o grau com que a previsão está consistentemente acima ou abaixo do valor real. Obter apenas uma alta correlação não significa necessariamente obter uma alta precisão. Por exemplo, um significante constante erro do tipo *bias* nas previsões forneceria uma correlação alta (r = 1), mas de fraca precisão. Como resultado, a precisão de previsões é mais bem analisada pelo uso de ambas. Uma previsão perfeita, o que é praticamente impossível de acontecer, deve possuir r = 1 e B = 0. Em Salas (1993), o autor fornece as equações para o cálculo desses índices.

As Figuras 8 e 9 apresentam uma comparação entre as vazões observadas e calculadas para a calibração e validação dos dados, respectivamente. Esses gráficos mostram que o *tank model* calibrado foi bem eficiente para estimar as vazões de entrada no reservatório. As altas correlações e valores baixos para o erro *bias* entre os valores observados e calculados indicam que a implementação do AG proposto foi capaz de calibrar o *tank model* eficientemente e, portanto, pode ser bastante útil para a calibração de outros modelos hidrológicos aplicados à área de recursos hídricos. Um exemplo prático seria a integração desta ferramenta aos modelos hidrológicos descritos por Almeida *et al.* (2007), Almeida *et al.* (2008) e Santos *et al.* (2008) do Sistema Nacional de Informação sobre Recursos Hídricos (SNIRH).

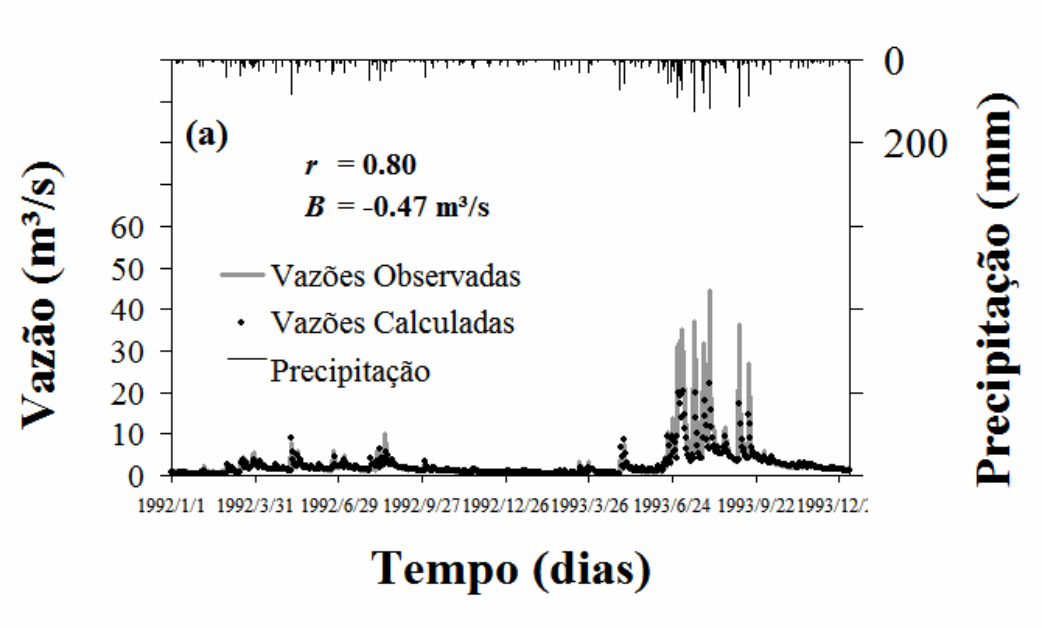

Figura 8 – Comparação entre vazões observadas e calculadas para calibração do modelo hidrológico *tank model*

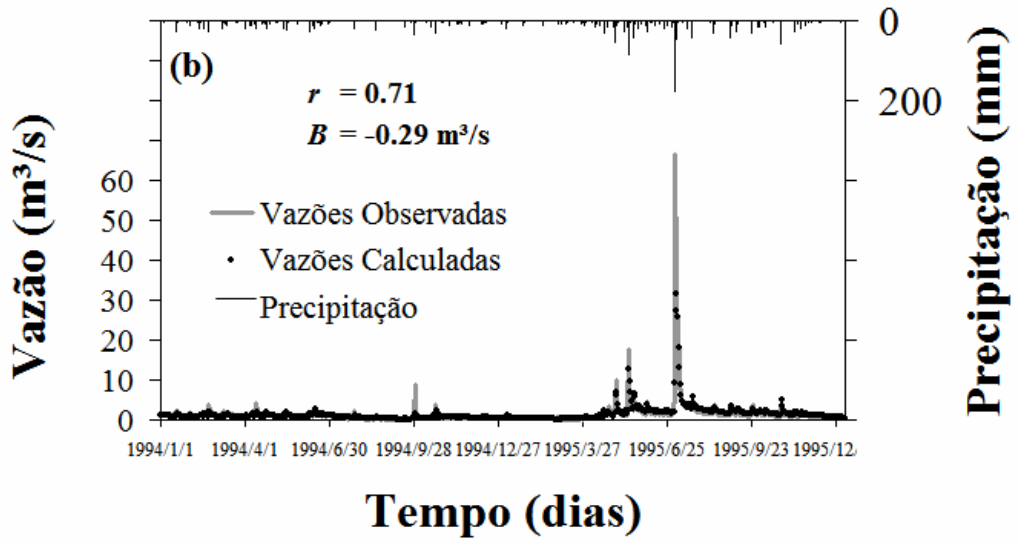

Figura 9 - Comparação entre vazões observadas e calculadas para validação do modelo hidrológico *tank model*

7 – CONCLUSÕES

Neste trabalho, foi apresentada uma implementação de um AG que pode ser utilizado na otimização de qualquer problema em que tal técnica possa ser aplicada. A utilização de XML e Java faz desta implementação uma ferramenta portável (Sistema

Operacional), configurável (dados de entrada e saída) e de propósito geral. O propósito geral aqui mencionado é possível porque a classe de avaliação de soluções pode ser estendida, fazendo com que o usuário possa implementar novas formas de avaliação das soluções.

Com o objetivo de mostrar o potencial desta implementação, o modelo hidrológico *tank model* (Sugawara, 1979) com três tanques foi calibrado, no qual onze parâmetros tiveram de ser otimizados.

Os resultados do *tank model* calibrado sugerem que a geração de vazões para o reservatório do rio Ishite em Matsuyama, Japão, foram bem confiáveis. Os resultados de calibração e validação também mostraram que esta implementação foi bem eficiente em definir os parâmetros do modelo. Uma vez que qualquer aplicação ou plataforma seja capaz de processar dados em XML, essa ferramenta pode ser uma alternativa importante para a resolução de problemas relacionados a recursos hídricos. Por exemplo, poderia ser uma ferramenta de grande valia para a otimização dos modelos hidrológicos do Sistema Nacional de Informação sobre Recursos Hídricos (SNIRH).

BIBLIOGRAFIA

ALMEIDA, C.N.; SANTOS, C.A.G.; BARBOSA, F.A.R.; SOARES JÚNIOR, A. (2007) *Integração de modelos chuva-vazão ao Sistema Nacional de Informações sobre Recursos Hídricos concepção do sistema.* In: XVII Simpósio Brasileiro de Recursos Hídricos e do 8º Simpósio de Hidráulica e Recursos Hídricos dos Países de Língua Oficial Portuguesa, 2007, São Paulo. Anais do XVII Simpósio Brasileiro de Recursos Hídricos e do 8º Simpósio de Hidráulica e Recursos Hídricos dos Países de Língua Oficial Portuguesa. Porto Alegre: ABRH, 2007. 17pp.

ALMEIDA, C.N.; SOARES JÚNIOR, A.; BARBOSA, F.A.R.; SANTOS, C.A.G. (2008) *Integração de modelos chuva-vazão ao sistema nacional de informações sobre recursos hídricos: Arquitetura e implementação.* In: I Encontro Nacional de Hidroinformática, 2008, Fortaleza. Anais do I Encontro Nacional de Hidroinformática. Fortaleza: Universidade de Fortaleza. 10pp.

CELESTE, A.B.; SUZUKI K.; KADOTA A. (2004) "*Genetic algorithms for real-time operation of multipurpose water resource systems"*, in Journal of Hydroinformatics, Vol 6; Part 1, pages 19-38.

FRANCHINI, M. (1996) "*Use of a genetic algorithm combined with a local search method for the automatic calibration of conceptual rainfall-runoff models*". Hydrological Sciences Journal **41**, pp. 21–40.

GRAVES, M. (2003) *Projeto de Banco de Dados com XML*. Makron Books, (518p.)

HOLLAND, J.H. (1975) "*Adaptation in Natural and Artificial Systems*", MIT Press.

JANIKOW, C.Z.; MICHALEWICZ, Z. (1991) "*An Experimental Comparison of Binary and Floating point Representations in Genetics Algorithms*". In ICGA.

LARMAN, C. (2007) *Utilizando UML e Padrões*. Ed. Bookman, Porto Alegre – RS, (696 p.)

MICHALEWICZ, Z. (1999) *Genetics Algorithms + Data Structures = Evolution Programs*. Pag. 17-18 (387 p) Springer, 3th edition.

SALAS, J.D. (1993) *Analysis and modeling of hydrologic time series*, in Handbook of Hydrology, edited by D. R. Maidment, Chap. 19, pp. 19.1-19.72, McGraw-Hill Inc., New York, USA.

SANTOS, C.A.G.; SRINIVASAN, V.S.; SUZUKI, K.; WATANABE, M. (2003a). "*Application of an optimization technique to a physically based erosion model*". Hydrol. Processes 47, 989–1003, doi: 10.1002/hyp.1176.

SANTOS, C.A.G.; SUZUKI, K.; WATANABE, M. (2003b). "*Modificação no Algoritmo Genético SCE-UA e sua Aplicação a um Modelo Hidrossedimentológico*". Revista Brasileira de Recursos Hídricos, Porto Alegre, v. 8, n. 1, p. 137–146.

SANTOS, C.A.G.; ALMEIDA, C.N.; BARBOSA, F.A.R.; SOARES JÚNIOR, A.; SOUZA, T.F.; MORENO, B.N. (2008) *Integração de um modelo hidrossedimentológico ao sistema nacional de informações sobre recursos hídricos.* In: VIII Encontro Nacional de Engenharia de Sedimentos, 2008, Campo Grande. Anais do VIII Encontro Nacional de Engenharia de Sedimentos. Porto Alegre: ABRH, 10pp.

SOARES JUNIOR, A; FARIAS, C.A.S.; SANTOS, C.A.G.; SUZUKI, K. (2009) "*An XML-based Genetic Algorithm for Nonlinear Optimization Problems*". In: 15th JSCE Congress - Shikoku Division, Matsuyama - Japan. v. II. p. 117-118.

SUGAWARA, M. (1979) "*Automatic calibration of the tank model*". Hydrological Science Bulletin, Vol. 24 (3). 1979

WANG, Q.J. (1997) "*Using genetic algorithms to optimize model parameters*". Environmental Modelling and Software 12, pp. 27–34.