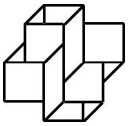


Laboratório Nacional de Computação Científica

Programa de Pós-Graduação em Modelagem Computacional

Sistemas Distribuídos para Otimização por Simulação Numérica Aplicada a Modelagem de Aquíferos

Por

Patrícia de Araújo Pereira Costa

PETRÓPOLIS, RJ - BRASIL

JULHO DE 2009

SISTEMAS DISTRIBUÍDOS PARA OTIMIZAÇÃO POR SIMULAÇÃO NUMÉRICA APLICADA A MODELAGEM DE AQUÍFEROS

Patrícia de Araújo Pereira Costa

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO LABORATÓRIO NACIONAL DE COMPUTAÇÃO CIENTÍFICA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM MODELAGEM COMPUTACIONAL

Aprovada por:

Prof. Eduardo Mendes Lúcio Garcia, D.Sc
(Presidente)

Prof. Antônio Tadeu Azevedo Gomes, D.Sc.

Prof. José Luis Drummond Alves, D.Sc.

PETRÓPOLIS, RJ - BRASIL
JULHO DE 2009

Costa, Patrícia de Araújo Pereira

C837s Sistemas distribuídos para otimização por simulação numérica aplicada a modelagem de aquíferos / Patrícia de Araújo Pereira Costa. Petrópolis, RJ. : Laboratório Nacional de Computação Científica, 2009.

xvi, 114 p. : il.; 29 cm

Orientadore(s): Eduardo Mendes Lúcio Garcia e Bruno Richard Schulze

Dissertação (M.Sc.) – Laboratório Nacional de Computação Científica, 2009.

1. Otimização matemática. 2. Modelos numéricos. 3. Sistemas distribuídos. 4. Computação paralela. 5. Remediação de aquíferos. I. Garcia, Eduardo Mendes Lúcio. II. LNCC/MCT. III. Título.

CDD 519.6

"The most beautiful system of the sun, planets, and comets could only proceed from the counsel and dominion of an intelligent and powerful Being."

Isaac Newton

Dedico este trabalho ao meu marido, Valois,

e ao meu pai, Agostinho.

Agradecimentos

Acima de tudo a Deus

Ao meu orientador, Bidu, por sua constante atenção e paciência, incentivo e estímulo ao meu trabalho, pelas conversas, apoio nos momentos difíceis, pelas idéias, explicações e desejo de sempre fazer um trabalho melhor.

Ao Pr. Bruno pela co-orientação, incentivo e ajuda na obtenção de apoio financeiro.

À minha família, de modo especial ao meu marido, Valois, pelo seu constante incentivo e apoio às minhas atividades acadêmicas e a meus queridos filhos, Mariana, Isabel, Felipe, Leonardo e Bernardo, por seu carinho e compreensão nos momentos de ausência. A meus pais, Agostinho e Maria das Dores, por suas orações, apoio e carinho.

A todos os professores do programa de Pós-Graduação do LNCC, que com seu conhecimento e dedicação à pesquisa científica, tanto me ensinaram e incentivaram.

A todos os queridos colegas que participaram desta caminhada, especialmente Jaqueline Ângelo, Natália, Carol, Gyslane, Carlos e Raquel.

Ao Gazoni, sempre prestativo e disposto não só a resolver os problemas apresentados, mas a ensinar como resolvê-los.

A todos do grupo de trabalho VCG, Thais, Fábio, Matheus e em especial ao Luís Rodrigo (Papagaio), sempre pronto a ajudar e possibilitar que eu pudesse usar mais "algumas máquinas" nos meus experimentos.

Ao Luiz Gonzaga e Marcelo Galheigo do LabInfo pela disponibilização das máquinas e ajuda na utilização destas.

Ao Pr. Hélio Barbosa pelas discussões e sugestões na área de otimização.

Aos colegas Franklin Joffly e Giacomo Valenzano pelo trabalho em equipe.

Ao LNCC pelo apoio institucional e ao CNPQ pelo suporte financeiro.

Resumo da Dissertação apresentada ao LNCC/MCT como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

SISTEMAS DISTRIBUÍDOS PARA OTIMIZAÇÃO POR SIMULAÇÃO NUMÉRICA APLICADA A MODELAGEM DE AQUÍFEROS

Patrícia de Araújo Pereira Costa

Julho, 2009

Orientador: Eduardo Mendes Lúcio Garcia, D.Sc

Co-orientador: Bruno Richard Schulze, D.Sc.

Neste trabalho, modela-se a ocorrência de contaminação de um aquífero hipotético por derramamento de substância tóxica e analisa-se a solução de descontaminação baseada na retirada do contaminante através de bombeamento feito por poços de extração. O projeto do sistema de remediação envolve a escolha do número de poços a serem instalados, suas localizações e vazões de modo a maximizar a quantidade de poluente extraída e ao mesmo tempo minimizar o custo total do sistema. A busca da solução ótima é feita de forma automática, através de um sistema paralelo de otimização por simulação numérica, composto por três subsistemas: (a) simulador numérico - resolve numericamente o modelo matemático do aquífero contaminado; (b) otimizador automático - implementa o método dos algoritmos genéticos para busca das localizações e vazões ótimas dos poços de extração; (c) sistema computacional distribuído - gerencia a distribuição e a execução paralela das simulações numéricas.

Foram feitos experimentos em vários ambientes computacionais: homogêneo, heterogêneo, em grande escala, usando máquinas não dedicadas, interligadas por rede local e ambiente de grade, e seus resultados demonstram a aplicabilidade da metodologia.

Abstract of Dissertation presented to LNCC/MCT as a partial fulfillment of the requirements for the degree of Master of Sciences (M.Sc.)

DISTRIBUTED SYSTEMS FOR NUMERICAL SIMULATION OPTIMIZATION APPLIED TO AQUIFER MODELING

Patrícia de Araújo Pereira Costa

July, 2009

Advisor: Eduardo Mendes Lúcio Garcia, D.Sc

Co-advisor: Bruno Richard Schulze, D.Sc.

In this dissertation, a hypothetical aquifer that has been contaminated by the dumping of toxic substances is modeled. The remediation strategy considered is based on withdrawal, which requires the removal of contaminated groundwater from the aquifer by pumping. The design of such a system involves the choice of the number of extracting wells to be installed, their locations and pumping rates, with the goal of maximizing the amount of contaminant extracted, while minimizing the cost of the system. To find the optimal solution, a numerical simulation optimization parallel system is used, which is composed by three subsystems: (a) numerical simulator - numerically solves the mathematical model of the contaminated aquifer; (b) optimizer - implements the genetic algorithm method to search for optimal locations and pumping rates for the extracting wells; (c) distributed computing system - manages the distribuition and parallel execution of the numerical simulations.

Experiments were done in many different computational environments: homogeneous, heterogeneous, in large scale, using non dedicated computers, connected via local network, and computational grids, and their results demonstrate the methodology's applicability.

Sumário

Lista de Figuras

Figura

Lista de Tabelas

Tabela

Capítulo 1

Introdução

Nos últimos anos, tem havido um aumento na disponibilidade de hardware computacional, bem como na sua capacidade: processadores cada vez mais rápidos e baratos, processadores com vários núcleos e redes de alta velocidade que possibilitam a integração destes recursos através de clusters, estações de trabalho ligadas via rede local, além de grades computacionais que permitem a utilização de recursos distribuídos geograficamente.

Assim, é comum que haja nas instituições de pesquisa, em particular, ciclos de máquinas disponíveis, o que motiva e possibilita a computação paralela para resolução de problemas de grande porte. Na modelagem, a resolução de modelos numéricos pode explorar essa disponibilidade. A otimização por simulação - técnica que utiliza modelos de simulação numérica para a avaliação de uma função objetivo - pode se beneficiar do uso de múltiplos recursos, pois além da necessidade de alto poder computacional, é naturalmente paralela e requer pouca comunicação.

Porém, mesmo com a oferta de ciclos, a forma de aproveitá-los não é trivial e a exploração desses recursos nunca é automática, sendo necessário o desenvolvimento de técnicas que permitam que uma aplicação possa usá-los de forma eficiente.

Neste trabalho é feita uma abordagem multidisciplinar que envolve modelo computacional de aquíferos, otimização e computação distribuída, aplicada à solução de um problema de remediação de aquíferos. Foi desenvolvido um sistema que utiliza computação paralela para realizar a busca automática baseada em soluções

numéricas.

A água doce, recurso natural valioso que tem sido objeto de atenção de toda a comunidade mundial, pode ser superficial ou subterrânea. Ambas estão sujeitas à contaminação. É importante ressaltar, no entanto, que 97,5% da água doce líquida encontra-se nas camadas subterrâneas [CETESB (2001)], onde a contaminação pode ocorrer devido a derramamento de poluentes, vazamento em tanques de armazenamento, lixões e aterros sanitários, atividades industriais, entre outras atividades humanas [Cunha (2002)].

Uma das técnicas bastante usadas para remediação é a extração da pluma de contaminante através de bombeamento da água contaminada, feito por poços de extração. Também são usadas a aspersão de ar e a contenção da pluma [J. Guan (1999)]. Nas técnicas citadas, o número de poços, sua localização e vazão são as variáveis do projeto. A remediação é um processo caro e que em geral leva muitos anos. O projeto de um sistema de remediação eficiente é essencial para a obtenção de bons resultados e a modelagem computacional deste fenômeno físico pode auxiliar de forma decisiva.

Nesta dissertação, uma evolução do trabalho apresentado em [Lima (2008)], modela-se a ocorrência de contaminação de um aquífero por derramamento de substância tóxica e analisa-se a solução de descontaminação baseada na retirada do contaminante por bombeamento feito por poços de extração.

Nesta estratégia, a escolha das variáveis do projeto - quantidade de poços, suas localizações e vazões - visa a máxima retirada de contaminante com custo mínimo. A escolha da solução ótima pode ser feita utilizando uma combinação da experiência de engenheiros com processos de tentativa e erro ou uma combinação de alguma estratégia de otimização e simulação computacional [Mayer et al. (2002)] e [Olafsson and Kim (2002)]. No primeiro método, critica-se a falta da formalização matemática necessária para garantir uma solução ótima. A metodologia de otimização por simulação é proposta para automatizar o método de tentativa e erro. O algoritmo de otimização toma decisões quanto à escolha, baseado em avaliações

de soluções candidatas que são feitas através de simulações de modelos matemáticos. Entretanto, uma das principais dificuldades é a necessidade de um grande número de simulações até que um resultado satisfatório seja obtido. As simulações são tarefas dispendiosas computacionalmente, porém podem ser realizadas independentemente, o que propicia o uso de ambientes distribuídos de computação, inclusive grades computacionais [Alba and Tomassini (2002)] e [Sayeed et al. (2007)].

O objetivo desta dissertação é desenvolver um sistema computacional que implemente o algoritmo de otimização e gerencie a distribuição e a execução das simulações para o modelo de aquífero contaminado em um ambiente de computação paralela, permitindo analisar a metodologia de otimização por simulação.

O sistema desenvolvido foi avaliado em vários ambientes: homogêneo, heterogêneo, em grande escala, usando máquinas não dedicadas, interligadas por rede local, cluster e ambiente de grade. Para cada um é analisado o desempenho quanto à utilização dos recursos e a diminuição do tempo de resposta.

Organização da dissertação

No Capítulo 2 apresenta-se o modelo do aquífero, o problema numérico estudado e a solução de remediação proposta. No Capítulo 3 é introduzida a técnica de otimização utilizada para a busca automática da solução, o algoritmo genético. São apresentados experimentos que mostram a eficiência do algoritmo implementado. O Capítulo 4 é dedicado ao estudo do sistema distribuído, sua arquitetura e características, e dos ambientes onde foram executados os experimentos que permitem analisar o desempenho do sistema e sua escalabilidade.

As escolhas feitas nesta dissertação produziram resultados satisfatórios, mas não são únicas, existindo possibilidades de melhorias, o que é revisado no Capítulo 5, onde são apresentadas as conclusões da dissertação.

Um apêndice sobre grades computacionais foi incluído.

Capítulo 2

Modelo Computacional de Aquíferos

Um modelo, num sentido amplo, pode ser definido como uma representação simplificada de um sistema real. Esta representação deve preservar uma equivalência adequada à sua utilização e permitir simular e obter a resposta do sistema a um conjunto de entradas e, assim, entender o seu funcionamento e também prever o que ocorrerá sob condições de interesse. Em muitos casos pode ser utilizado como ferramenta de gerenciamento, para facilitar o planejamento e a tomada de decisões.

A modelagem computacional de sistemas físicos usa modelos matemáticos e soluções aproximadas com este objetivo. A construção de um modelo computacional pode ser decomposta em três etapas: o modelo conceitual, o modelo matemático e o modelo numérico [Bear and Verruijt (1987)].

O modelo conceitual ou físico é a nossa compreensão da física do fenômeno. Devido à complexidade dos sistemas reais, são necessárias simplificações e estas são introduzidas como uma série de hipóteses que expressam a natureza do fenômeno e as características de seu comportamento que são relevantes para o problema investigado. Essas hipóteses descrevem, verbalmente, a composição do sistema, os processos de transporte que ocorrem no sistema, os mecanismos que governam esse transporte e as propriedades relevantes do meio [Bear et al. (1992)]. O próximo passo do processo de modelagem consiste em expressar o modelo conceitual, verbal, na forma de um modelo matemático.

O modelo matemático usa equações para representar as leis que governam o sistema, propriedades físicas como coeficientes das equações e potenciais como as variáveis do sistema [Wang and Anderson (1982)]. Na construção do modelo matemático para escoamentos em aquíferos, define-se um sistema de equações diferenciais parciais que regem o fenômeno de acordo com as hipóteses assumidas no passo anterior. Uma vez construído o modelo matemático, este deve ser resolvido para um dado conjunto de entradas, incluindo a geometria do aquífero, carregamento e condições iniciais e de contorno, que descrevem uma situação específica de interesse. Na maioria dos casos não é possível obter-se solução analítica e uma alternativa é a utilização de uma aproximação numérica.

O modelo numérico é um método aproximado para ser resolvido computacionalmente e que gera respostas aproximadas para o modelo matemático.

Existem simplificações nas três etapas da construção do modelo. A fidelidade da resposta encontrada ao problema real vai depender das simplificações introduzidas em cada uma das etapas e da precisão dos métodos numéricos.

Este capítulo tem o objetivo de ressaltar os aspectos mais importantes nas três fases da modelagem computacional de aquíferos, além de apresentar o modelo numérico do aquífero que será usado para o experimento de remediação.

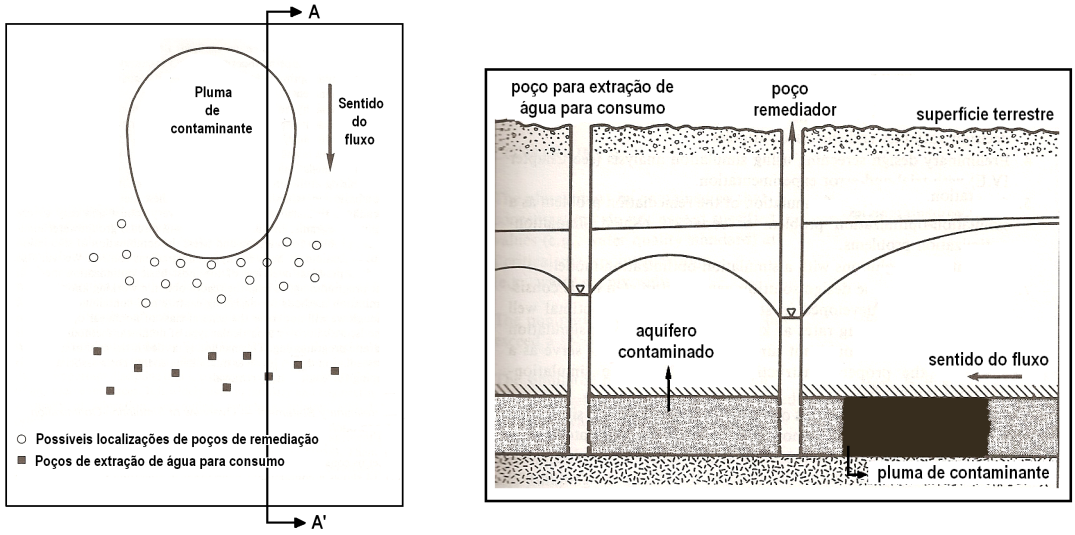
Esta dissertação não teve como objetivo resolver as dificuldades existentes nos métodos numéricos mas, conhecendo-as, saber utilizar da melhor forma um programa já existente. O modelo computacional de aquífero desenvolvido em [Garcia (1997)] será a base para o estudo de otimização por simulação feito neste trabalho.

2.1 Modelo conceitual

Aquíferos podem ser modelados usando a teoria clássica de escoamentos em meios porosos. Modelos de escoamentos em meios porosos consideram a presença de uma fase sólida e de uma fase fluida que escoa entre os interstícios da primeira. A fase sólida representa a rocha porosa que abriga o aquífero e é descrita por suas dimensões, pela porosidade e pela permeabilidade. No caso estudado, a fase fluida

é constituída pela água e pelo contaminante, e as propriedades que a descrevem são a difusividade e a dispersão física. O modelo adotado assume as hipóteses de um meio poroso homogêneo, rígido e saturado onde ocorre escoamento incompressível e monofásico.

O aquífero modelado ocupa um domínio bi-dimensional retangular. A Figura 2.1 ilustra o problema estudado. Em 2.1(a) tem-se a vista superior, onde pode-se observar uma pluma de contaminante que é transportada em direção a poços de extração de água para consumo, representados por quadrados pretos na parte inferior da figura. A seta representa o sentido do escoamento que, neste caso, é da parte superior para a inferior da figura e ocorre devido à diferença de pressão. Como estratégia de remediação, é colocada uma série de poços para a extração do contaminante, representados por círculos vazados, a fim de remover a água contaminada evitando que a pluma atinja a região de extração de água para consumo. Uma vista em corte deste mesmo esquema é mostrada na Figura 2.1(b), onde aparecem a pluma de contaminante e dois poços: um de extração de água para consumo e outro de remediação. No problema estudado, não existe influência dos poços de consumo no modelo, eles apenas representam a região que deseja-se evitar que a pluma atinja.



(a) vista superior

(b) vista em corte (AA')

Figura 2.1: Vista superior e em corte do aquífero ([Gorelick et al. (1993)]).

O contaminante é transportado na água devido ao processo de difusão e no meio devido ao processo de convecção, caracterizando um problema de convecção-difusão onde a convecção é o processo dominante. O escoamento do fluido na fase sólida ocorre devido à diferença de pressão nas fronteiras superior e inferior e por existência de poços de extração e de consumo. Nas fronteiras laterais, assume-se que não há fluxo normal. A força da gravidade é desprezada (modelo areal).

As hipóteses descritas guiam a escolha das equações do modelo matemático.

2.2 Modelo matemático

As equações diferenciais parciais que regem o fenômeno de escoamento de um fluido incompressível em um meio poroso, rígido e saturado formam um sistema elíptico obtido pela lei de Darcy, pela equação de conservação da massa e pela equação de convecção-difusão, esta última predominantemente convectiva e que expressa a conservação da massa do contaminante.

Seja Ω um domínio aberto limitado em $\mathbb{R}^2$ com fronteira Γ regular e $T \in \mathbb{R}$ um número fixo. O sistema diferencial em $\Omega \times [0, T]$, nas variáveis p, $\mathbf{u}$ e c, respectivamente, a pressão hidrostática total do sistema, a velocidade de escoamento de Darcy da fase líquida e a concentração do contaminante, normalmente usado para modelar um escoamento miscível de um fluido incompressível em um meio poroso rígido saturado é constituído pelas seguintes equações [Bear (1979)] e [Peaceman (1977)]:

lei de Darcy

$$\mathbf{u} = -\frac{K}{\mu}\nabla p, \quad \text{em} \quad \Omega \tag{2.1}$$

lei de conservação da massa

$$div\mathbf{u} = f, \quad \text{em} \quad \Omega \tag{2.2}$$

com condição de contorno

$$\mathbf{u} \cdot \mathbf{n} = 0, \quad \text{em} \quad \Gamma$$

lei de conservação da massa do contaminante

$$\phi \frac{\partial c}{\partial t} + div(\mathbf{u}c) - div(D(\mathbf{u})\nabla c) = \hat{c}f \quad \text{em} \quad \Omega \tag{2.3}$$

com condição de contorno

$$D(\mathbf{u})\nabla c \cdot \mathbf{n} = 0, \quad \text{em} \quad \Gamma$$

e condição inicial

$$c(\mathbf{x}, 0) = c_0(\mathbf{x}).$$

Em 2.1, $K = K(x)$ é a permeabilidade do meio, representado por um tensor simétrico e definido positivo que representa as direções preferenciais do fluido e $\mu = \mu(c)$ é a viscosidade da mistura dada em função da concentração do contaminante. Neste caso, assume-se a hipótese de que o contaminante tem as mesmas propriedades do fluido residente e $\mu = 1$.

Em 2.2, f representa os termos de fonte ou sumidouro associados aos poços.

Em 2.3, $\phi = \phi(x)$ é a porosidade do meio, $\hat{c}$ é o valor da concentração do solvente nos poços de injeção e concentração residual nos poços de extração e $D = D(\mathbf{u})$ é o tensor de dispersão-difusão dado por:

$$D(\mathbf{u}) = (\frac{\alpha_{mol}}{\tau} + \alpha_T \|\mathbf{u}\|)\mathbf{I} + \frac{\alpha_L - \alpha_T}{\|\mathbf{u}\|}\mathbf{u} \otimes \mathbf{u}, \tag{2.4}$$

onde τ é a tortuosidade do meio poroso, α_{mol}, α_L e α_T são coeficientes de difusão molecular e de dispersão hidrodinâmica longitudinal e transversal, respectivamente. O símbolo $\otimes$ representa o produto tensorial.

Neste sistema, a concentração é uma variavel adimensional que assume va-

lores entre 0,0 e 1,0 e representa a porcentagem volumétrica de contaminante na fase fluida.

A solução analítica deste sistema só é possível para casos particulares de geometria simples e meios homogêneos. Situações mais complexas, que representam melhor os problemas reais, são resolvidas usando métodos numéricos que fornecem soluções aproximadas para este sistema.

2.3 Modelo numérico

Entre os métodos numéricos comumente utilizados para este problema, pode-se citar o método das diferenças finitas e método de elementos finitos (MEF) [Bear (1979)], [Wang and Anderson (1982)], [Bear and Verruijt (1987)].

Nestes dois métodos, a área de interesse é subdividida através de uma malha em sub-regiões menores, chamadas células ou elementos, aos quais estão associados nós. Nas duas abordagens, a discretização no espaço e no tempo permite que o sistema de equações diferenciais seja reduzido a um sistema de equações lineares que pode ser resolvido através de métodos matriciais diretos ou iterativos.

Cada método tem suas vantagens e desvantagens. Em geral, o método das diferenças finitas é mais simples conceitualmente e matematicamente, e sua implementação computacional é mais fácil. Este método é apropriado para malhas retangulares relativamente simples, o que facilita também a entrada de dados. Já o método de elementos finitos exige um maior conhecimento matemático, e para alguns problemas gera resultados mais precisos numericamente que os produzidos por diferenças finitas. O método de elementos finitos permite flexibilidade na construção da malha, o que possibilita uma maior aproximação espacial de regiões com contorno irregulares. Em geral, no MEF e em outros métodos numéricos é comum o uso de pré-processadores de modelo que incluem um gerador de malhas e um método para numerar eficientemente os elementos e os nós da malha.

Embora esses métodos sejam frequentemente utilizados, existem outros tipos de métodos numéricos aplicáveis a esta classe de problemas, incluindo o método

dos volumes finitos, o método dos elementos analíticos e, para o problema de transporte, o método das características, *particle tracking*, os métodos Eulerianos-Lagrangianos, entre outros.

Existem várias implementações computacionais desses métodos usadas em problemas de águas subterrâneas. Destaca-se aqui o MODFLOW [1] , que determina o fluxo do escoamento usando diferenças finitas e que pode ser usado junto com o MODPATH [2] , RT3D [3] [Clement et al. (1998)] ou MT3DMS [4] [Zheng and Wang (1999)], entre outros, para a resolução da equação de transporte. O MODPATH e o RT3D usam o método numérico das diferenças finitas e o MT3DMS usa múltiplas técnicas: diferenças finitas, método das características e *total-variation-diminishing* (TVD). O SUTRA [5] [Voss (1984)] resolve o modelo de fluxo por diferenças finitas e o de transporte por elementos finitos. O FEFLOW [6] utiliza elementos finitos.

O MODFLOW acoplado com o MODPATH foi usado por [Shreedhar Maskey and Solomatine (2002)]. Já [Ren and Minsker (2005)] acoplaram o MODFLOW ao RT3D e [Kalwij and Peralta (2006)], ao MT3DMS. Para os experimentos de seu artigo [Cedeno and Vemuri (1996)] usaram o SUTRA. O modelo usado neste trabalho é implementado pelo programa AGUASUBMEF, desenvolvido por [Garcia (1997)], que utiliza o método de elementos finitos. O modelo de otimização por simulação proposto nesta dissertação não possui particularidades que impeçam o uso das outras implementações computacionais mencionadas.

2.3.1 AGUASUBMEF

O AGUASUBMEF foi inicialmente desenvolvido para modelos de escoamento em reservatórios de petróleo mas, com algumas considerações, é aplicável a águas subterrâneas. A implementação computacional foi feita utilizando a linguagem de programação Fortran e como já mencionado, este programa utiliza o método de

[1] water.usgs.gov/nrp/gwsoftware/modflow.html
[2] water.usgs.gov/software/MODPATH
[3] bioprocess.pnl.gov/rt3d.htm
[4] hydro.geo.ua.edu/mt3d
[5] water.usgs.gov/nrp/gwsoftware/sutra.html
[6] www.feflow.info

elementos finitos.

No caso estudado, resolve-se inicialmente o problema da pressão utilizando-se o método de Galerkin clássico. A velocidade é calculada por uma técnica de pós-processamento global [Loula and Garcia (1995)] e [Malta et al. (1995)]. Os métodos de pós-processamento são mais precisos do que obter a velocidade diretamente em (2.1), como gradiente da pressão. A análise de problemas que envolvem escoamentos em meios porosos, entre eles a contaminação de aquíferos e processos para sua remediação são, em geral, predominantemente convectivos e a determinação precisa da velocidade se torna essencial para a correta avaliação dos processos de transporte associados. Uma vez resolvido o subsistema elíptico, a concentração é calculada pelo SUPG (Streamline Upwind Petrov-Galerkin), que vem corrigir as dificuldades do Galerkin clássico.

Assume-se que o contaminante tem a mesma viscosidade da água. Isso implica que as propriedades do fluido não variam com a concentração do contaminante, tornando o problema linear. Assim, pode-se resolver a equação da pressão e a da velocidade uma única vez e a equação de transporte é dependente do tempo.

Os dados de entrada são fornecidos através de um arquivo-texto e incluem o tamanho da malha, o tamanho dos elementos e as coordenadas de seus nós, os nós com as condições iniciais, o número de dias da simulação e o tamanho do passo de tempo, as posições e vazões dos poços de extração, as propriedades do meio tais como porosidade, permeabilidade, dispersão transversal e longitudinal e as condições de contorno, entre outros. Dentre os arquivos com dados de saída gerados pelo AGUASUBMEF, os utilizados pelos experimentos deste trabalho são:

- o arquivo com o volume de contaminante no aquífero a cada passo de tempo;
- o arquivo que contém os valores das concentrações de contaminante nos poços de extração a cada passo de tempo.

O custo computacional das simulações, característica importante para a computação paralela, é fixo para malhas do mesmo tamanho devido a dois fatores:

- o número de sistemas de equações algébricas a ser resolvido é constante pois o modelo de remediação do aquífero é linear;

- o método utilizado para a resolução dos sistemas de equações algébricas é um método direto que, ao contrário dos métodos iterativos, faz sempre a mesma quantidade de operações.

2.4 Caso específico de contaminação

Um aquífero hipotético considerado simples, bi-dimensional, saturado, homogêneo e isotrópico foi usado como primeiro experimento, feito com o objetivo de mostrar o transporte da pluma de contaminante sem interferência de poços de extração. Os parâmetros usados nas simulações estão na Tabela (2.1) e representam uma área total de 2.000,0 por 8.000,0 metros, com uma malha de 80 X 320 elementos quadriláteros de tamanho 25,0 X 25,0 metros. As condições de contorno no lado superior e inferior do domínio são de pressão constante 100,0m e 50,0m, respectivamente, o que produz um escoamento na direção da maior para a menor pressão, com velocidade constante em toda a extensão do domínio. No lado esquerdo e direito, uma condição de fluxo normal nula é imposta. A Figura 2.2 mostra a condição inicial deste problema.

Parâmetros	
Geometria e propriedades	**Valores**
Domínio	$2.000 \times 8.000m^2$
Porosidade	$\phi = 0,20$
Volume de poluente	$2.000m^3$
Permeabilidade	$K = 100\mu D$
Difusão hidrodinâmica	$\alpha_l = 10,0m, \alpha_t = 2,0m$
Intervalo de integração no tempo	$\Delta T = 5dias$
Malha bidimensional	80×320 elementos quadriláteros

Tabela 2.1: Domínio e propriedades utilizadas na simulação do escoamento.

Esta simulação foi feita para um período total de 1.800 dias. Neste período a pluma atinge o contorno inferior do domínio. Os resultados apresentados ilustram

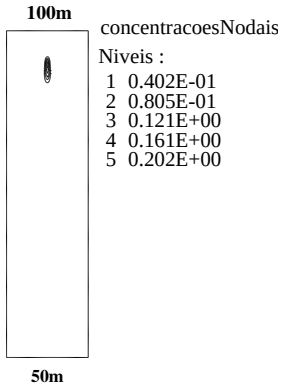
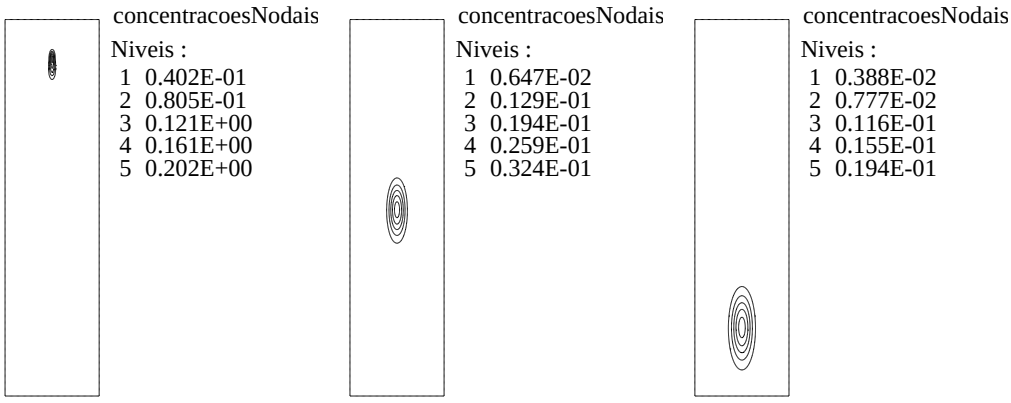


Figura 2.2: Condição inicial do aquífero.

o problema físico. As Figuras 2.3(a), 2.3(b) e 2.3(c) mostram isolinhas de concentração do que seria contaminante. Ao lado das figuras estão valores numéricos que representam as magnitudes das linhas e estão entre 0,0 e 1,0. A Figura 2.3(a) mostra a condição inicial do modelo e as Figuras 2.3(b) e 2.3(c), a pluma após 1.000 dias e 1.800 dias respectivamente. Com o passar do tempo, o contaminante se espalha devido ao processo de difusão, e as isocurvas tornam-se maiores. O valor das concentrações diminui um pouco, pois a área ocupada é maior.



(a) condição inicial (b) 1.000 dias sem remediação (c) 1.800 dias sem remediação

Figura 2.3: Isocurvas para simulação sem poços de remediação.

A Figura 2.4 mostra o volume de contaminante no aquífero com o tempo. Estes valores são obtidos pela integração espacial do campo de concentração de contaminante a cada passo de tempo. Sem remediação, este volume permanece constante, mostrando que os métodos possuem conservação da massa global. Esta curva aparecerá em outros gráficos, sendo usada para uma análise comparativa.

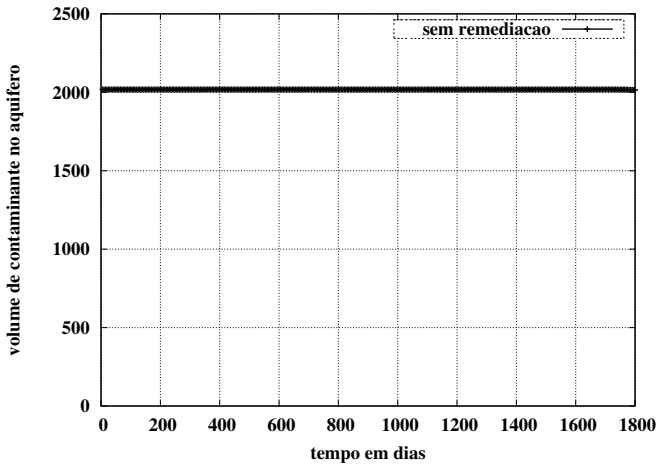


Figura 2.4: Volume de contaminante no aquífero, caso sem remediação.

Pretende-se evitar que a pluma alcance a parte inferior do domínio supondo que, a partir deste ponto, possam ocorrer problemas como a contaminação de um rio ou a extração de água contaminada por poços de consumo.

2.4.1 Modelos de Remediação

Existem várias estratégias para a remediação de águas subterrâneas contaminadas, incluindo extração do contaminante, contenção ou redirecionamento do fluxo hidráulico, tratamento químico e biorremediação [Cunha (2002)], [Gorelick et al. (1993)]. A escolha de um método é baseada nas características geológicas da região e nos objetivos da remediação, sendo a extração do contaminante uma das técnicas mais comumente utilizadas [Lehr and Nielsen (1982)].

No Brasil, alguns problemas de contaminação também têm sido remediados desta forma. Em [Abdanur (2005)], encontra-se a descrição de um problema de contaminação e remediação por extração ocorrido na Refinaria de Duque de Caxias (RJ). Em Camaçari (BA), numa área onde resíduos industriais foram dispostos por mais de 20 anos, contaminando as águas subterrâneas, foi feita a instalação e operação de um sistema de recuperação, constituído por um conjunto de poços de bombeamento, para conter o avanço das plumas em direção às águas superficiais e remover os contaminantes presentes no aquífero. No estado de São Paulo, a Companhia de Tecnologia de Saneamento Ambiental (CETESB) faz um serviço de

prevenção e monitoramento, determinando valores de concentrações de substâncias químicas que fornecem orientação sobre a condição de qualidade do solo e da água subterrânea e são utilizados como instrumentos para prevenção e controle da contaminação e gerenciamento de áreas contaminadas sob investigação [CETESB (2004)]. De acordo com a CETESB, até novembro de 2004, dos 480 casos de remediação implantados, 271 foram por bombeamento e tratamento [Nobre (2005)].

Métodos de extração exigem a retirada da pluma de contaminante do aquífero através de bombeamento feito por poços de extração. A água contaminada retirada é limpa e pode ser reinjetada no aquífero através de poços de injeção ou exportada para o local de depósito. Vários casos de sistemas de remediação através de remoção são encontrados na literatura [Ahlfeld et al. (1988)].

Neste trabalho, que estuda o método de remediação através de bombeamento feito por poços de extração, os poços são introduzidos no modelo matemático como carga pontual, prescrevendo vazão.

A forma de remediação mais favorável é a instalação de poços em regiões onde o centro da pluma passará. Neste caso, com o meio homogêneo, com a pluma localizada na região central do aquífero e com o escoamento uniforme da parte superior para a parte inferior, isto ocorre com a colocação de poços em $x = 1000$, conforme Figura 2.5, que mostra o modelo do aquífero com a linha por onde passa o centro da pluma.

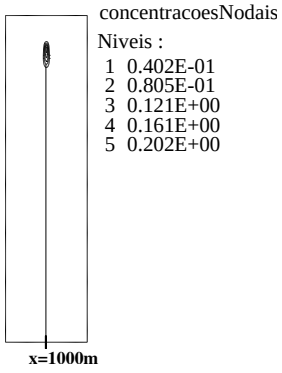
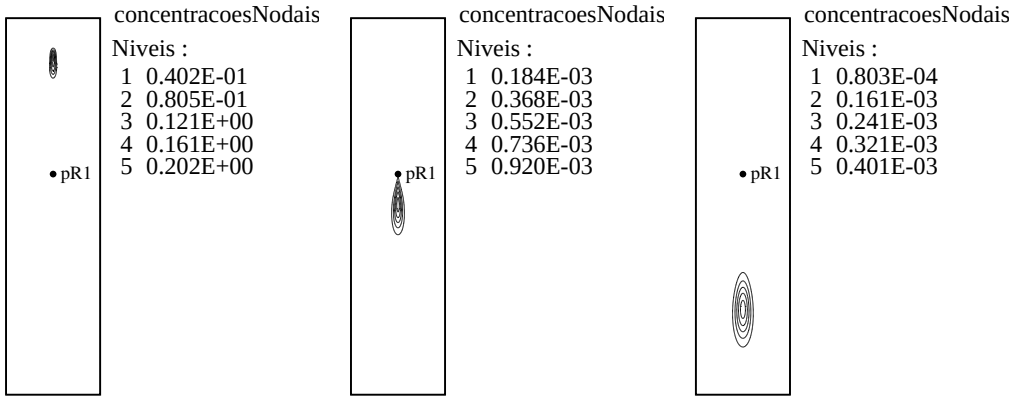


Figura 2.5: Modelo do aquífero com a linha onde passa o centro da pluma.

2.4.1.1 Remediação com um poço

Para experimento feito usando os mesmos parâmetros do experimento anterior (Tabela 2.1), mas com a colocação de um poço de extração com vazão de $400m^3/dia$, a Figura 2.6 mostra as isocurvas e o comportamento da pluma de contaminante. A coordenada y do poço é aproximadamente $4700m$, e a coordenada x encontra-se em $1000m$, ou seja na linha central do aquífero, conforme Figura 2.6(a), onde pR1 representa a localização do poço remediador.

Percebe-se que os valores dos níveis das isoconcentrações na Figura 2.6(b), que representa a pluma após 1.000 dias, já demonstram redução na quantidade de contaminante. Também pode-se observar comparando-se as Figuras 2.3(c) e 2.6(c), que devido à ação dos poços, que alteram as condições hidráulicas do aquífero, a pluma sofre um atraso e em 1.800 dias está em posição menos avançada que no experimento sem colocação de poços.



(a) condição inicial (b) 1.000 dias com remediação (c) 1.800 dias com remediação

Figura 2.6: Isocurvas para simulação com um poço de extração.

A Figura 2.7 mostra o volume de contaminante no aquífero durante o período da simulação para os dois casos: sem remediação e com remediação. Os valores que compõem estas curvas foram calculados fazendo a integração do campo de concentração do contaminante no domínio em cada instante da simulação. No caso com remediação houve retirada do contaminante pelo poço de extração e os valores diminuíram após a pluma passar pelo poço, entre os dias 400 e 900.

A Figura 2.8 mostra a concentração de contaminante na mistura retirada

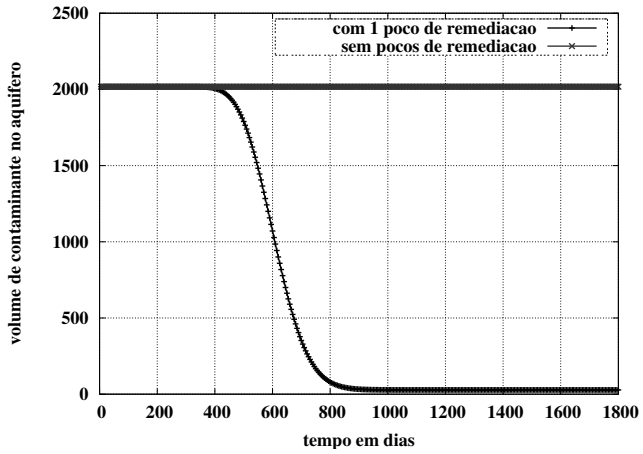


Figura 2.7: Volume de contaminante no aquífero nos casos: sem remediação e com a colocação de um poço.

pelo poço remediador com a variação do tempo. A integral desta curva multiplicada pela vazão do poço determina a quantidade de contaminante retirada pelo poço. Neste tipo de gráfico pode-se observar a influência do poço na extração do contaminante verificando: o instante em que a pluma atinge o poço (400 dias) e o período de tempo em que o poço está removendo contaminante (até 900 dias aproximadamente).

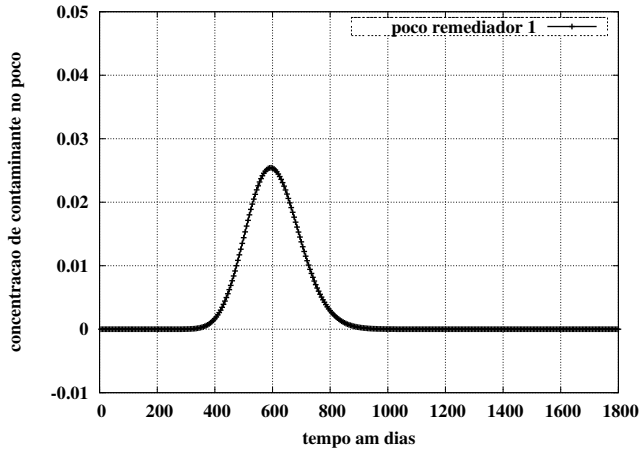


Figura 2.8: Concentração de contaminante em um poço de extração.

A solução deste caso homogêneo e sem restrições é trivial, porém pode haver casos com restrição, por exemplo, no valor da vazão e na localização dos poços. Para ilustrar soluções para problemas com restrição são apresentados dois cenários, denominados A e B.

2.4.1.2 Remediação com dois poços - cenário A

Considerando-se restrições na vazão máxima dos poços e na localização em $x = 1000m$, pode-se imaginar uma solução com dois poços com vazões de $200m^3/dia$, localizados na mesma altura que o poço do experimento anterior, porém igualmente afastados da linha central do aquífero, distantes $100m$ entre si. Esta situação é ilustrada na Figura 2.9, onde os poços remediadores são denominados pR1 e pR2.

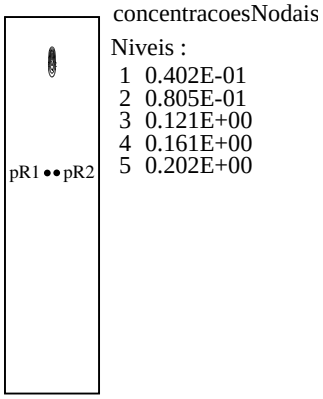


Figura 2.9: Condição inicial e posição de instalação de dois poços, cenário A.

A Figura 2.10 mostra o volume de contaminante no aquífero durante o período de simulação e se comparada com a Figura 2.7 (remediação com um poço), percebe-se que as curvas são semelhantes, o que significa que as soluções são equivalentes.

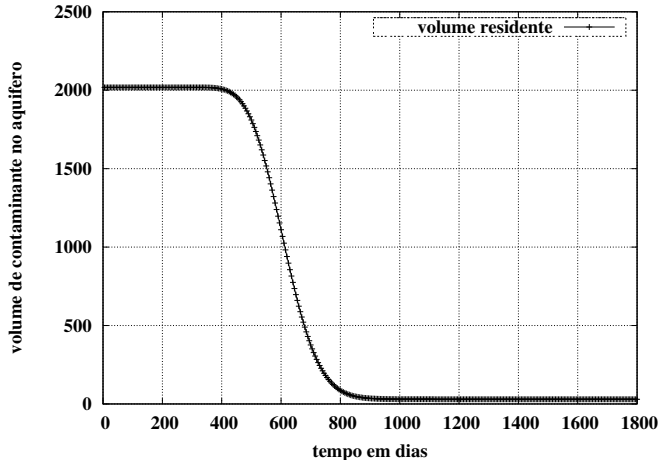


Figura 2.10: Volume de contaminante no aquífero, cenário A.

A Figura 2.11 mostra a concentração de contaminante em cada poço a cada

passo de tempo. As curvas são coincidentes devido à simetria de posição e aos poços possuírem a mesma vazão e isto significa que ambos retiram a mesma quantidade de poluente, no mesmo período de tempo.

Comparando-se as Figuras 2.8 e 2.11, pode-se entender a equivalência entre esta solução e a analisada no experimento anterior. Nota-se que as curvas são bem semelhantes, principalmente na forma, apesar dos valores máximos se diferenciarem. A soma das áreas da Figura 2.11 multiplicadas por 200 é equivalente à área da Figura 2.8 multiplicada por 400 e estes valores representam o volume de contaminante extraído pelos poços.

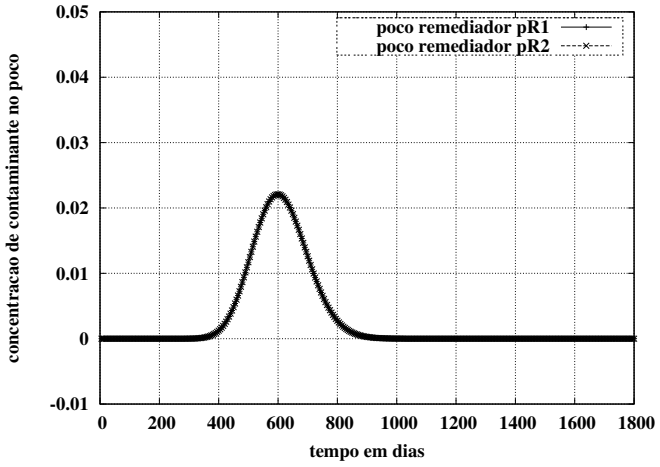


Figura 2.11: Concentração de contaminante em dois poços de extração, cenário A.

Outro aspecto observado é que com esta solução, da mesma forma que o caso anterior, a remoção ocorre entre os dias 400 e 900, o que é esperado pois a posição vertical foi mantida.

2.4.1.3 Remediação com dois poços - cenário B

Uma outra possibilidade com dois poços, cenário B, neste caso com restrição na vazão dos poços, mas sem restrição na localização, é colocar os dois poços na coluna central do aquífero ($x = 1000m$): o primeiro poço (pR1) em $y = 4200m$ e o segundo (pR2) em $y = 5200m$, conforme a Figura 2.12. Ambos têm vazão de $200m^3/dia$.

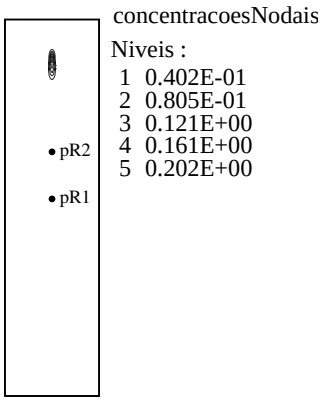


Figura 2.12: Condição inicial e a posição de instalação de dois poços, cenário B.

A Figura 2.13 mostra o volume de contaminante no aquífero com a colocação dos poços pR1 e pR2 conforme apresentado na Figura 2.12. Nota-se que o volume extraído é semelhante ao experimento com 1 poço (Figura 2.7), porém neste caso o tempo total de remoção do contaminante é maior uma vez que a distância vertical entre os poços e a pluma também é maior.

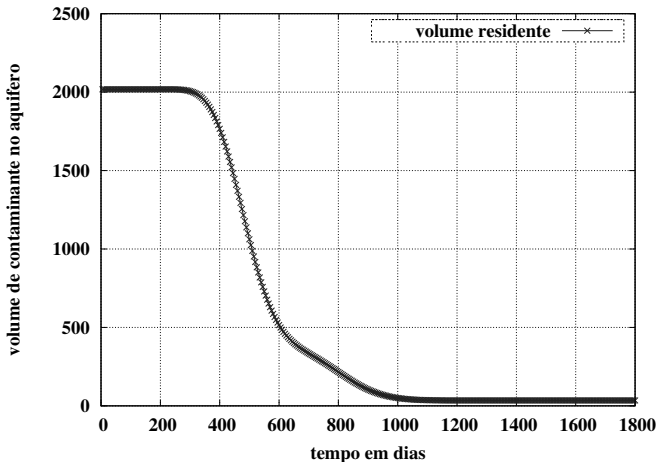


Figura 2.13: Volume de contaminante no aquífero, cenário B.

Na Figura 2.14, que mostra a concentração de contaminante em cada poço a cada passo de tempo, verifica-se que como a pluma atinge primeiro o poço pR2, este remove a maior parte do poluente. De forma semelhante ao cenário anterior, as áreas das curvas multiplicadas pelas vazões, neste caso iguais a $200m^3/dia$, determinam o volume de contaminante retirado por cada poço e ao serem somadas fornecem o volume total extraído.

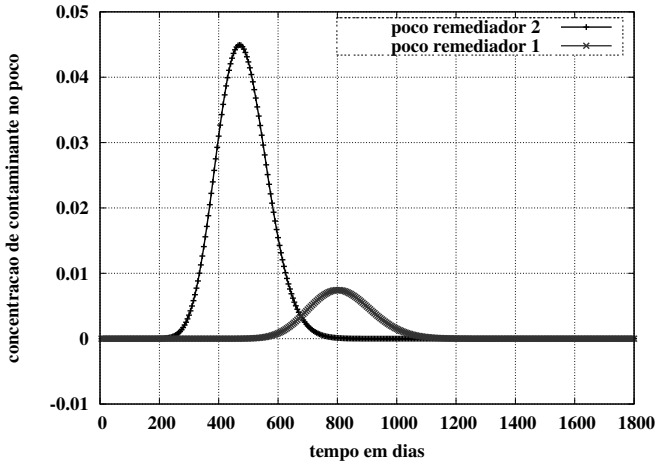


Figura 2.14: Concentração de contaminante em dois poços de extração, cenário B.

O poço pR1 retira menos contaminante e pode parecer pouco "útil". Uma simulação usando apenas com o poço pR2 foi feita e o volume retirado pode ser visto na Figura 2.15. Observa-se que nesta situação o volume retirado é menor e que o tempo de início da retirada é maior, levando a conclusão de que o poço pR1 altera as condições hidráulicas e faz com que a pluma atinja o poço pR2 mais rapidamente, produzindo uma retirada mais rápida do contaminante.

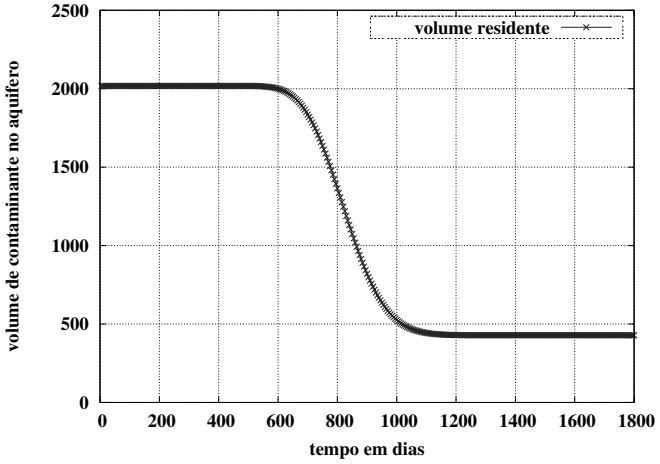


Figura 2.15: Volume de contaminante no aquífero apenas com o poço pR2, cenário B.

2.5 Busca automática da solução

Dada a habilidade de simular o transporte do contaminante usando o modelo computacional descrito, o próximo passo é incorporar este modelo a um processo

automático que facilitará a escolha das melhores soluções para a descontaminação do aquífero, de acordo com os objetivos que se quer alcançar.

Com os casos de remediação mostrados, deseja-se ilustrar algumas formas de solução do problema e como elas podem ser analisadas. Quando for utilizado o sistema automático, soluções desta natureza vão aparecer e este tipo de análise será feita.

Capítulo 3

Otimização por Simulação Numérica

Otimização por simulação é um termo usado para descrever técnicas de otimização que utilizam modelos de simulação numérica para a avaliação da função objetivo. Existe uma grande variedade de aplicações apropriadas para o uso destas técnicas, incluindo a otimização de projetos de engenharia, a otimização de sistemas estocásticos, a calibração de modelos e a solução de problemas inversos [Mirghani et al. (2005)]. Na literatura encontram-se muitos trabalhos sobre otimização por simulação numérica e, mais recentemente, rotinas de otimização têm sido incorporadas a pacotes comerciais de simulação [Olafsson and Kim (2002)].

Nas duas últimas décadas, a remediação e a monitoração da qualidade de águas subterrâneas têm sido o foco de trabalhos de otimização que podem ser encontrados na literatura de águas subterrâneas - em particular no projeto de sistemas de extração do contaminante através de poços [Mayer et al. (2002)], uma das aplicações mais consideradas - dentre eles [Ahlfeld and Sawyer (1990)], [Dougherty and Marryott (1991)], [Cedeno and Vemuri (1996)], [Huang and Mayer (1997)], [J. Guan (1999)], [Shreedhar Maskey and Solomatine (2002)], [Ren and Minsker (2005)] e [Kalwij and Peralta (2006)].

Uma abordagem comum para o problema de remoção do contaminante por bombeamento feito por poços de extração é considerar como variáveis de decisão: o número de poços, a localização e vazão de cada um deles, e definir para cada uma valores com o objetivo de minimizar o custo total do sistema [J. Guan (1999)].

Para a escolha dos melhores valores para as variáveis de decisão, existem várias técnicas de otimização. Dos trabalhos anteriormente mencionados [Huang and Mayer (1997)], [J. Guan (1999)] e [Ren and Minsker (2005)] usam Algoritmos Genéticos (AG), [Dougherty and Marryott (1991)] usa Recozimento Simulado, [Kalwij and Peralta (2006)] usa AG e Busca Tabu, [Cedeno and Vemuri (1996)] usa uma variação de AG para otimização multiobjetivo e [Shreedhar Maskey and Solomatine (2002)] testa alguns tipos de algoritmos de otimização global, entre eles AG e *adaptative cluster covering* (ACCO).

Neste trabalho utiliza-se um algoritmo genético geracional - portanto síncrono - e panmítico [1] [Lim et al. (2007)] e [Alba and Tomassini (2002)]. Neste capítulo, são apresentados conceitos sobre a otimização em geral e sobre os algoritmos genéticos, a implementação feita e os resultados de alguns experimentos computacionais para remediação de aquífero com uma pluma de contaminante e possibilidade de instalação de um ou dois poços, usando o modelo de aquífero apresentado no capítulo anterior.

3.1 Otimização

Segundo Eduardo Camponogara [Camponogara (2006)] a otimização pode ser definida como:

> *"A área da matemática aplicada que se preocupa em calcular e computar valores ótimos para variáveis de decisão que induzem desempenho ótimo, ao mesmo tempo que satisfazem restrições de um modelo matemático."*

A formulação matemática de um problema de otimização (PO) é dada por:

$$(PO) = \begin{cases} \text{minimizar} & f(x) \\ \text{sujeito a} & x \in \Omega, \end{cases} \tag{3.1}$$

[1] modelo de AG padrão, no qual a população é tratada como um único conjunto de indivíduos, ao contrário de AGs estruturados, onde são feitas divisões da população como, por exemplo, ilhas.

onde $f(x)$ é a função objetivo, Ω é a região viável e x é o vetor com as variáveis de decisão.

Os componentes de um PO são:

- variáveis de decisão: parâmetros cujos valores definem uma solução para o problema;
- função objetivo: uma função das variáveis de decisão a ser minimizada ou maximizada (otimizada);
- restrições ou conjunto viável: funções que definem o espaço factível das soluções; conjunto dos possíveis valores que as variáveis de decisão podem assumir.

Na técnica de otimização por simulação, a função objetivo é calculada com base nos resultados de um modelo de simulação, e o problema possui as componentes usuais da otimização [Olafsson and Kim (2002)] que podem ter, no caso do sistema de remediação de aquífero por poços de extração, as seguintes características: as variáveis de decisão são o número de poços de extração, as localizações e as vazões dos poços; a função objetivo, em geral, assume a forma de uma função custo, que deseja-se minimizar e que leva em consideração o custo total dos poços; o conjunto viável é definido pelas restrições impostas: os locais onde os poços podem ser instalados e os valores de vazão disponíveis.

Com relação ao espaço de busca (os valores que as variáveis de decisão podem assumir), os problemas de otimização podem ser divididos basicamente em duas categorias: os que possuem conjunto viável contínuo e os que possuem conjunto viável discreto (problemas combinatoriais). Na primeira categoria, existem infinitas soluções candidatas e, na segunda, a busca é feita num conjunto finito ou possivelmente infinito enumerável. Os métodos de resolução destas duas categorias são bastante diferenciados. Deve-se notar ainda que, na prática, podem surgir problemas onde tanto variáveis de decisão contínuas quanto discretas estão presentes.

3.1.1 Métodos de otimização

A literatura identifica três tipos principais de métodos de busca: métodos baseados em cálculo, métodos de enumeração e métodos randômicos [Goldberg (1989)], também conhecidos como não-tradicionais ou probabilísticos.

Os métodos baseados em cálculo são subdivididos em duas classes: indiretos e diretos. Os métodos indiretos, ou métodos baseados em gradiente, procuram o ótimo resolvendo um conjunto de equações não-lineares obtidas igualando-se a zero o gradiente da função objetivo, em relação às variáveis de decisão. O método do Gradiente Conjugado e o Máximo Declive (*Steepest Descent*) são exemplos deste método. Os métodos diretos, ditos métodos de ordem zero, utilizam apenas os valores da função objetivo para guiar o processo de busca, não requerendo nenhuma informação sobre o gradiente da função a otimizar. Como exemplo, podemos citar os métodos Simplex de Nelder-Mead e a busca-padrão de Hooke-Jeeves.

Métodos baseados em cálculo são criticados por serem metodologias de busca local, cujo resultado depende do ponto inicial de busca, além de dependerem da existência de derivadas, sendo apropriados apenas em casos de funções continuamente diferenciáveis.

As metodologias de enumeração resolvem problemas de otimização discreta, verificando e comparando todas as combinações possíveis das variáveis de decisão. Embora seja uma forma de pensar intrínseca do ser humano, a grande maioria dos problemas reais possui um número enorme de possíveis soluções e, neste caso, o uso deste método torna-se inviável. Para um número de possibilidades pequeno o suficiente para testarmos todas as respostas, esta metodologia certamente encontra a solução ótima.

Os métodos não-tradicionais são métodos inspirados em algum fenômeno da natureza. Essas técnicas têm ganhado popularidade devido à sua robustez e, também, devido às limitações das metodologias enumerativas e das baseadas em cálculo para a solução de problemas mais complexos. Têm como característica o uso de números pseudo-aleatórios para conduzir escolhas pseudo-aleatórias como

ferramenta de uma busca não-determinística. É importante notar que essa busca tem características aleatórias sem ser necessariamente uma caminhada aleatória [Goldberg (1989)]. Estes métodos são apropriados para conjuntos finitos, mas com grande número de possíveis combinações. Como exemplo, pode-se citar: o Recozimento Simulado (*Simulated Annealing*-SA), os Algoritmos Genéticos (AG) e a Otimização por Colônia de Formigas (ACO) [Colorni et al. (1996)].

Devido a limitações dos métodos acima e a características específicas de alguns problemas, existem também modelos híbridos que utilizam os métodos baseados em cálculo, combinados com métodos não-tradicionais. Cai et al. (2001), por exemplo, usou Programação Linear combinada com AG para resolver modelos de gerenciamento de recursos hídricos.

O problema estudado nesta dissertação é discreto e pode-se calcular o número de soluções possíveis com a seguinte expressão:

$$\sum_{i=1}^{nP} \binom{nLocal}{i} q^i,$$

onde

- $nLocal$ - o número de possíveis localizações para os poços;
- nP - o número de poços de extração instalados;
- q - as possíveis vazões para cada poço.

A Tabela 3.1 mostra o número de possibilidades para algumas das situações que serão estudadas na Seção 3.3, onde são feitos experimentos para a validação do método implementado. Para estes experimentos é usada uma malha de elementos finitos de tamanho 80X320 e, dos 26.001 nós existentes, considera-se uma região com 15.247 possíveis localizações para os poços ($nLocal$). Pode-se observar que mesmo com valores pequenos das variáveis, o número de possibilidades atinge valores muito grandes e não faz sentido pensar em avaliar todas elas. Para a escolha da solução ótima ou de boas soluções será usado o Algoritmo Genético, apresentado na seção que segue.

nP	**q**	Total de Combinações
1	1	15.247
1	4	60.988
2	4	1.859.707.084
3	4	75.602.672.329.804
1	10	152.470
2	10	11.622.940.570
3	10	1.181.262.697.230.000

Tabela 3.1: Número de possíveis combinações para casos com $nLocal = 15.247$.

3.2 Algoritmos Genéticos

Algoritmos Genéticos são uma subclasse dos Algoritmos Evolutivos (AE), inspirados na genética e na teoria evolucionista de Darwin. O uso de AGs foi proposto pela primeira vez por Holland [Holland (1975)]. AGs são técnicas heurísticas de busca global capazes de explorar simultaneamente várias possíveis regiões de pico, com o objetivo de encontrar a solução ótima; combinam a exploração sistemática de boas regiões já encontradas (*exploitation*) com a exploração de novas áreas do espaço de busca (*exploration*) [Eiben and Smith (2003)].

A idéia da técnica é a seguinte: dada uma população de indivíduos, onde cada indivíduo representa uma possível solução para o problema a ser resolvido, a pressão do ambiente causa seleção natural - sobrevivência dos mais aptos - e isso causa um aumento na aptidão da população ao longo do tempo (gerações), pois os piores indivíduos tendem a desaparecer, enquanto os melhores continuam a evoluir na busca de uma solução ótima, explorando simultaneamente as diversas áreas do espaço de busca.

A evolução acontece através da criação de novas populações cujos indivíduos são gerados aplicando-se técnicas de seleção, recombinação e mutação. Os genitores, indivíduos que darão origem a novos indivíduos, são escolhidos de acordo com um processo de seleção feito com base na aptidão de cada um. A aptidão é determinada pelo cálculo da função objetivo e quantifica a qualidade de uma solução candidata (indivíduo) para o problema em questão. Os novos indivíduos -

os filhos - são gerados aplicando-se as seguintes operações:

- recombinação: combinação da informação de dois genitores;
- mutação: modificação de uma ou mais características de um genitor.

Os novos filhos gerados poderão substituir, na população atual, os indivíduos menos aptos segundo critérios definidos. Esse processo é repetido até que uma condição de parada seja satisfeita.

Devido a sua flexibilidade, facilidade de implementação, robustez e eficácia em realizar busca global, os AGs têm ganhado popularidade desde que foram propostos, no final da década de 70, tendo obtido um grande impulso na década de 80 devido ao avanço do poder computacional.

Um AG convencional, geracional e com população única, pode ser obtido pela implementação do algoritmo a seguir:

```
1: Inicializar a população de indivíduos
2: Avaliar os indivíduos
3: Repita
4:     Selecionar os geradores dos novos indivíduos
5:     Aplicar operadores genéticos
6:     Avaliar os novos indivíduos
7:     Selecionar indivíduos para sobreviver
8: Até satisfazer critério de parada
```

Algoritmo 1: Algoritmo genético convencional.

A formulação de um AG para resolução de um determinado problema envolve, necessariamente:

- a escolha de um esquema de representação para as variáveis de decisão;
- a criação de vários conjuntos de valores iniciais para as variáveis de decisão;
- a definição de um mecanismo de avaliação capaz de atribuir um grau de desempenho para cada conjunto de variáveis de decisão;

- a especificação de mecanismos capazes de gerar, a partir de soluções existentes, novas soluções com probabilidade maior de melhor desempenho;
- a definição de uma condição de parada;
- a escolha de valores para os parâmetros do AG, como por exemplo: o tamanho da população, o número de candidatos por geração e a probabilidade de aplicação dos operadores genéticos.

As escolhas feitas nestas etapas vão definir a aplicabilidade e a eficiência do AG para o problema em questão. As seções seguintes descrevem essas etapas e como foram implementadas nesta dissertação.

3.2.1 Representação de indivíduos

O primeiro passo no desenvolvimento de um algoritmo genético para a resolução de um problema é a escolha da representação dos indivíduos. Os indivíduos "carregam" as informações sobre as variáveis do projeto e representam uma possível solução para o problema. Os tipos de representação mais comumente utilizados são binário, inteiro ou real. Os AGs tradicionais têm utilizado representação binária, independente do problema a ser resolvido. [Falkenauer (1999)] afirma que a importância do uso da representação binária foi de um certo modo exagerada no passado e conforme mostrado por [Goldberg (1989)], não há vantagens na utilização de representação binária, a não ser a facilidade da definição dos operadores. Pesquisadores sugerem que a representação de um indivíduo deve ser a mais natural possível.

Na situação considerada, cada indivíduo tem informações sobre o número de poços, suas localizações e vazões, representados como inteiros, e a aptidão - valor que quantifica quão boa é a solução - representada por um real. O indivíduo ou solução candidata tem uma "estrutura" conforme a Figura 3.1, onde:

- nP : número de poços de extração;
- lP : lista de poços, contendo nP elementos;

- localização (x_i, y_i) : coordenada do local de instalação do i-ésimo poço, $i = 1, .., nP$;
- vazão (q_i) : valor da vazão do i-ésimo poço, $i = 1, .., nP$;
- aptidão: "nota" do indivíduo.

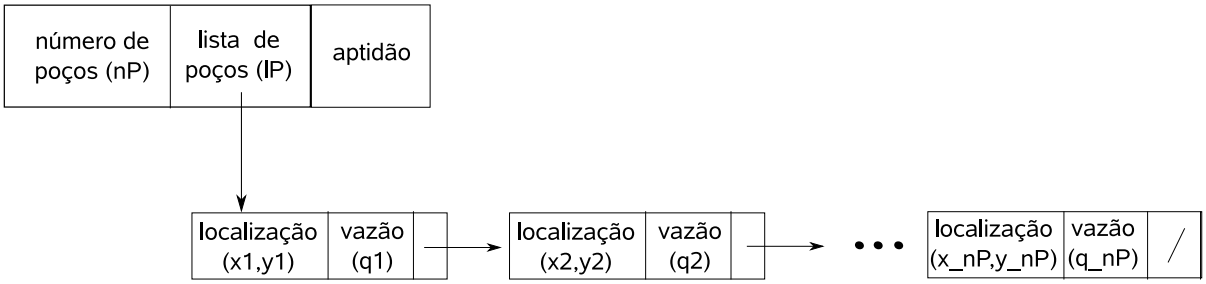


Figura 3.1: Representação do indivíduo.

3.2.2 Função aptidão

Uma vez que os AGs são baseados no princípio da sobrevivência dos mais aptos, torna-se necessária uma forma de determinar o grau de aptidão dos indivíduos da população. A função aptidão desempenha este papel de classificação, recebendo como parâmetros de entrada valores armazenados no indivíduo e valores gerados pela execução da simulação numérica, e retornando um valor que quantifica a aptidão deste indivíduo.

Um dos fatores principais para que o AG obtenha bons resultados é a escolha de uma função aptidão que avalie corretamente os indivíduos, uma vez que é a aptidão que determinará quais terão maior chance de sobreviver e de reproduzir-se, passando suas características para a próxima geração.

A função aptidão, ou função de avaliação, é particular para cada problema. Na otimização de uma função nos reais, a função aptidão pode ser a própria função objetivo que deseja-se minimizar ou maximizar, mas normalmente é preciso definir uma função aptidão adequada ao problema em questão, o que pode não ser uma tarefa muito simples.

Funções objetivo podem ser realistas e complexas ou mais simplificadas. No caso de remediação de aquíferos, Ren and Minsker (2005) apresentam um estudo comparando as duas abordagens, e mostram que uma função realista fornece resultados mais precisos, porém, requer maior esforço e mais tempo em seu desenvolvimento. Por outro lado, uma função mais simples tem desenvolvimento mais rápido, mas pode-se sacrificar a qualidade dos resultados. A função do presente modelo é simples, estando de acordo com o modelo do aquífero.

Um outro aspecto a ser considerado na construção da função objetivo é que muitos problemas de tomada de decisão de interesse prático são caracterizados por múltiplos objetivos, que são, em geral, objetivos conflitantes pois a melhoria de um leva à degradação de outro. O tomador de decisão precisa encontrar um equilíbrio entre esses objetivos conflitantes [Cedeno and Vemuri (1996)].

O problema de remediação de aquíferos através de bombeamento por poços de extração também é um problema multiobjetivo, onde em geral os objetivos são maximizar a quantidade de poluente retirada e minimizar o custo de instalação e gerenciamento dos poços e tratamento do poluente. Em alguns casos, outros fatores são considerados: pode-se, por exemplo, desejar também minimizar o tempo de remoção do poluente ou o tempo para a obtenção de um nível mínimo de poluente, considerado aceitável.

Uma das estratégias para a solução de problemas multiobjetivos é combinar, de alguma forma, os objetivos numa só função. Uma abordagem é a minimização de uma função custo que engloba o custo dos poços e penalidades cobradas sobre a quantidade de contaminante não retirada [Finsterle (2004)]. A otimização implementada neste trabalho utiliza essa estratégia, e a função aptidão a ser minimizada considera o custo dos poços de extração e um valor de penalização pela quantidade de contaminante não retirada, conforme equação (3.2).

$$f_{apt} = \sum_{i=1}^{nP} C_i + P * VR, \tag{3.2}$$

onde

- nP : número de poços de extração;
- C_i : custo do i-ésimo poço;
- P : valor da penalidade cobrada pelo volume de poluente não extraído;
- VR : volume residual.

O custo dos poços é um custo fixo que representa o custo de instalação e é calculado em função da vazão das bombas. Por simplicidade, não é considerado o custo operacional que depende da vazão das bombas e do tempo de operação. Com esses parâmetros, pode-se guiar a escolha das melhores soluções em função dos objetivos buscados. Se a multa por não descontaminação for alta, as soluções com mais poços e com vazões altas serão mais eficientes e espera-se que, com o passar das gerações, esses indivíduos predominem na população. Em um outro cenário, com alto custo dos poços de maior vazão, pode haver uma tendência para escolha de soluções com poços de vazão mais baixa.

3.2.3 População inicial

Tradicionalmente, a população inicial de um AG é gerada de forma aleatória, através de um processo computacional de geração de números pseudo-aleatórios diretamente influenciado por um número de partida denominado "semente". Devido à natureza estocástica dos AGs, os resultados de uma otimização usando os mesmos parâmetros podem ser diferentes se forem usadas sementes diferentes.

Para uma melhor exploração do espaço de busca, e consequente convergência para um ótimo global, é muito importante que os indivíduos da população inicial estejam distribuídos de forma a representar todo este espaço de busca. Uma das formas de garantir esta representatividade é a inserção de bons indivíduos, com aptidão já conhecida. Entretanto, isto deve ser usado com cautela pois a presença de indivíduos com aptidão muito melhor que os demais pode causar convergência prematura. Além disto, é preciso considerar se o esforço computacional extra gasto

para gerar estes indivíduos levará à descoberta de uma solução significativamente melhor.

O algoritmo implementado neste trabalho gera a população inicial aleatoriamente, mas também permite a inserção de indivíduos que são lidos através de um arquivo de entrada.

3.2.4 Seleção

Seguindo a idéia básica da Computação Evolucionista, os indivíduos com melhor aptidão são mais aptos à reprodução e tendem a fazê-lo com maior frequência. Entretanto, os indivíduos menos aptos não podem ser desprezados, pois estes podem ser melhorados através das gerações, além de serem fator de diversidade, o que garante a exploração das diversas áreas do espaço de busca. Para reproduzir este comportamento, existem diferentes técnicas para a seleção dos genitores.

No método de *Seleção Proporcional à Aptidão* (SPA) [Goldberg (1989)], a probabilidade de um indivíduo ser selecionado para reprodução é dada por:

$$p_i = \frac{f_i}{\Sigma_{j=1}^{N} f_j}, \tag{3.3}$$

onde f_i é a aptidão do indivíduo i e N é o tamanho da população. Neste método, também conhecido como *Seleção por Roleta*, quanto maior a aptidão do indivíduo, maior será o setor da roleta atribuído a ele. Desta forma, indivíduos com aptidão muito maior que o resto da população acabam sendo selecionados com frequência muito alta e rapidamente tomam conta da população, causando convergência prematura. Outro problema ocorre quando os valores das aptidões estão muito próximos entre si e os setores da roleta ficam com tamanhos muito parecidos, fazendo com que a seleção se transforme num processo quase aleatório (sem pressão seletiva). Além disso, é preciso que a aptidão de todos indivíduos seja positiva.

Inspirado nos pontos negativos do SPA, foi desenvolvido o método de *Seleção por Ranking* [Baker (1985)], ou *Seleção por Posto*, onde a população é ordenada de acordo com a aptidão e a probabilidade de seleção de um indivíduo é alocada de

acordo com sua posição, ao invés do valor de sua aptidão. O mapeamento entre a posição do indivíduo e sua probabilidade de seleção pode ser feito de várias formas, entre elas a linear ou a exponencial. Para o método denominado *Linear Ranking*, a probabilidade de seleção do indivíduo na i-ésima posição é dada pela fórmula:

$$P_i = \frac{MIN + (MAX - MIN)\ \dfrac{(N-i)}{(N-1)}}{N}, \tag{3.4}$$

onde i é o índice do individuo na população em ordem decrescente de valor da função aptidão (no caso de minimização da função aptidão) e N é o tamanho da população. O *Linear Ranking* requer $(1 \leq MAX \leq 2)$ e $(MIN + MAX = 2)$. Para $MIN = MAX = 1$, $P_i = \frac{1}{N}, \forall i$ e a seleção é aleatória. A Figura 3.2 mostra o mapeamento entre a posição do indivíduo num *rank* ordenado por aptidão e a sua probabilidade de seleção para dois casos. No primeiro caso $MIN = 0,9$ e $MAX = 1,1$ e a pressão de seleção é bem menor que no segundo caso, onde $MIN = 0,0$ e $MAX = 2,0$. Com o uso do *Linear Ranking* a pressão de seleção aplicável é limitada e para casos onde uma pressão maior é necessária, um esquema de *Exponential Ranking* pode ser usado [Eiben and Smith (2003)].

Uma outra metodologia é a *Seleção por Torneio* [Goldberg and Deb (1991)], onde k indivíduos são sorteados aleatoriamente e o melhor indivíduo deste grupo é selecionado. Quando $k = 2$, a seleção é denominada *Torneio Binário*.

[Goldberg and Deb (1991)] comparam diversos esquemas de seleção e concluem que, com o ajuste adequado dos parâmetros, todos os métodos apresentam desempenho equivalente, com exceção do SPA pelos motivos já mencionados.

Neste trabalho, o método de seleção usado foi o *Linear Ranking* e os valores para os parâmetros MIN e MAX podem ser escolhidos pelo usuário, através de um arquivo de entrada de parâmetros.

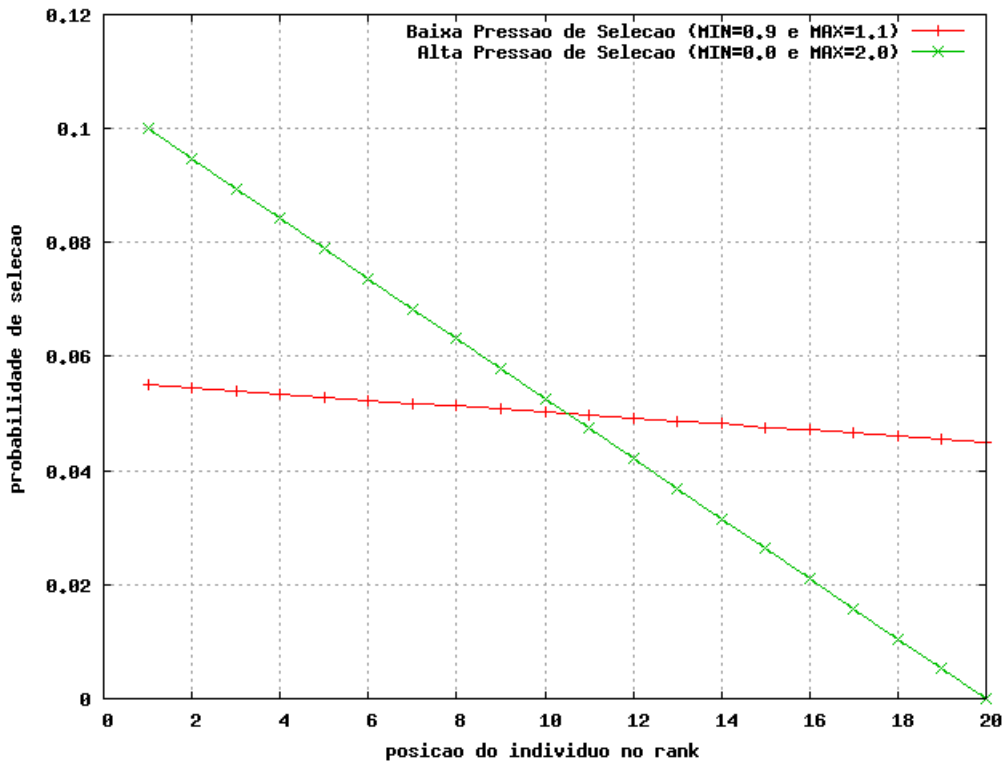


Figura 3.2: *Linear Ranking.*

3.2.5 Operadores genéticos

Os operadores genéticos são mecanismos capazes de gerar, a partir de indivíduos já existentes, novos indivíduos que tenham probabilidade maior de possuir uma melhor aptidão. A definição dos operadores genéticos em um AG depende do problema em questão e da representação escolhida para o indivíduo. Esses operadores devem ser capazes de gerar indivíduos que representem:

- soluções em regiões diversas do espaço de busca, desempenhando o papel de exploração (*exploration*);
- soluções na vizinhança de boas regiões já encontradas, sendo responsável pela intensificação da busca (*exploitation*).

Os operadores básicos dos AGs são: recombinação e mutação.

Para este problema de alocação de poços de extração foram desenvolvidos quatro operadores, descritos abaixo: um operador de recombinação e três de mutação: mutação no número de poços, mutação na localização dos poços e mutação

na vazão dos poços.

A frequência com que esses operadores são aplicados é determinada por taxas que, nesta implementação, podem ser fixas ou variar linearmente ao longo das gerações conforme parâmetros indicados no arquivo de entrada.

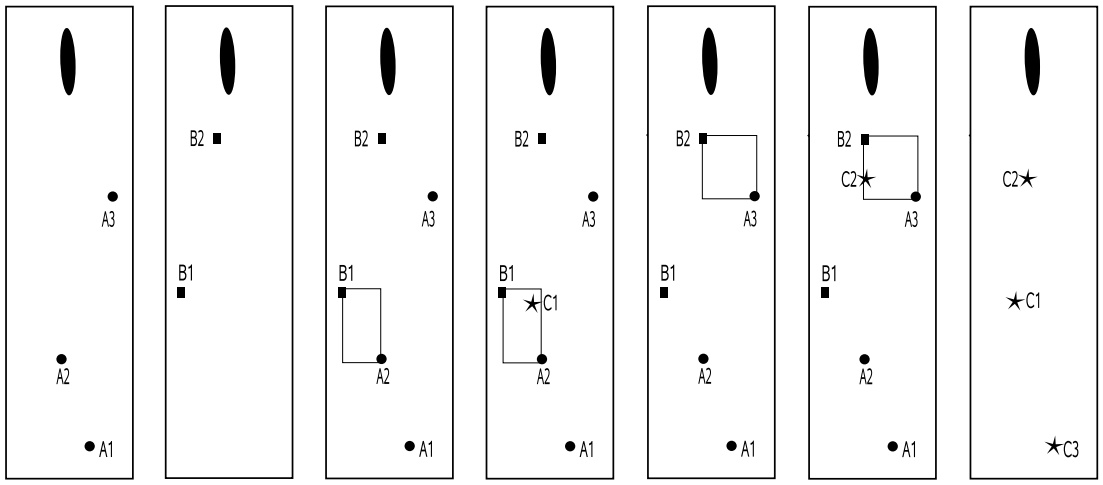
3.2.5.1 Recombinação

Na recombinação, também chamada de *crossover*, dois indivíduos da população atual são selecionados através de um dos processos de seleção, e suas características são combinadas para a formação de um novo indivíduo.

Existem vários operadores de recombinação desenvolvidos para os vários tipos de representação, mas este operador é muito específico para cada problema. Falkenauer (1999) considera o operador *crossover* um diferencial entre os AGs e outras metodologias de metaheurística iterativas, mas enfatiza que ele não funciona apropriadamente num contexto arbitrário. Uma vez que o *crossover* combina partes de soluções para formar uma nova solução, esta combinação precisa fazer sentido para o problema a ser resolvido senão o novo indivíduo terá valores inválidos, não pertencendo ao espaço de busca, ou não terá características semelhantes às de seus genitores.

O operador de recombinação desenvolvido neste trabalho é ilustrado na Figura 3.3, onde A1, A2 e A3 representam os poços de um suposto pai (paiA), B1 e B2, os poços do outro pai (paiB), e C1, C2 C3, os poços do novo indivíduo gerado (filhoC).

Inicialmente são selecionados dois genitores, (3.3(a) e 3.3(b)), e a partir desses, é gerado um novo indivíduo. As características deste novo indivíduo (número de poços, localização e vazão de cada poço) são definidas da seguinte forma: o número de poços será igual ao número de poços de um de seus pais. Esta escolha é feita aleatoriamente. Para definição da localização e da vazão dos poços do novo indivíduo, escolhe-se um poço de cada pai e cria-se uma região retangular, cujos vértices são as localizações dos poços escolhidos dos pais (3.3(c)). Em al-



(a) paiA (b) paiB (c) área p/ C1 (d) C1 (e) área p/ C2 (f) C2 (g) filhoC

Figura 3.3: Exemplo de *crossover*.

gum ponto, dentro desta região, será colocado, através de sorteio, o poço do filho (3.3(d)). A vazão deste novo poço é escolhida dentro do intervalo[2] das vazões dos pais. Repete-se este processo enquanto houver "pares" de pais (3.3(e) e 3.3(f)). Caso tenha sido sorteado que o filho terá número de poços igual ao do pai com mais poços, os poços extras são copiados do pai com mais poços para o filho, sem alterações (poço C3 em 3.3(g)).

Para a escolha dos pares de poços, um de cada pai, busca-se sempre combinar os mais próximos, numa tentativa de evitar a criação de filhos que não se assemelhem a nenhum de seus pais.

3.2.6 Mutação

O operador mutação caracteriza-se pela mudança aleatória em uma das características de um indivíduo pai, gerando um novo indivíduo.

Nesta implementação, existem três tipos de mutação:

- mutação na localização dos poços;
- mutação na vazão dos poços;

[2] Este intervalo é discreto e os valores possíveis da vazão são 100, 150, 200, 250, 300, 325, 350, 375, 400 e 425 m^3/dia

- mutação no número de poços.

O indivíduo que sofrerá a mutação será escolhido através do método de seleção *Linear Ranking*.

3.2.6.1 Mutação na localização

Quando este operador é aplicado, a posição de todos os poços do indivíduo selecionado sofre uma pequena alteração, sendo deslocada para uma das oito possíveis adjacências, conforme Figura 3.4. Este operador exerce a função de intensificação, isto é, busca na vizinhança de boas soluções.

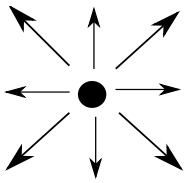


Figura 3.4: Possíveis direções na mutação da localização.

3.2.6.2 Mutação na vazão

Para a aplicação deste operador, escolhe-se da lista de poços do indivíduo selecionado o poço que retira mais poluente e altera-se sua vazão, escolhendo na lista ordenada de possíveis vazões a posição anterior (diminuindo a vazão) ou a posterior (aumentando a vazão). Escolhe-se o melhor poço produtor para ter a vazão alterada, seguindo o raciocínio de que poços que retiram pouco o fazem por estarem mal localizados, e não devido à sua vazão. A mutação na vazão, assim como na localização, tem características de intensificação na busca.

Pode-se desejar diminuir a vazão de uma bomba em casos onde um valor baixo da multa por não despoluição favoreça a escolha de um poço menos caro, uma vez que o objetivo é minimizar o custo total.

3.2.6.3 Mutação no número de poços

Nesta operação, inicialmente, realiza-se um sorteio para decidir se será colocado mais um poço ou se será retirado um dos poços existentes na lista de poços do indivíduo selecionado. Em caso de retirada, escolhe-se o pior poço produtor (o que remove menos poluente). Em caso de adição, a escolha da localização e da vazão do novo poço é feita de forma aleatória.

Este operador faz mudanças maiores e tem um papel exploratório, podendo alterar de forma radical a aptidão do indivíduo.

3.2.7 Substituição parental

Substituição parental é um mecanismo através do qual escolhe-se, entre os pais e os novos filhos gerados, quem vai sobreviver, dando origem à nova geração.

Em um AG *geracional*, para uma população de tamanho μ, a cada geração são criados λ novos indivíduos, com $\lambda > \mu$. A substituição pode ocorrer da seguinte forma:

- os μ pais são substituídos pelos μ melhores filhos;
- os μ pais são substituídos pelos μ melhores indivíduos do conjunto união de pais e filhos.

Uma estratégia muito usada é o *elitismo*, onde os K melhores indivíduos são copiados para a geração seguinte, para evitar que sejam perdidos. Quanto maior for K, maior a pressão de seleção, crescendo o risco de convergência prematura.

Em contraste, em um AG *não-geracional* ou de regime permanente (*steady-state*), cada novo indivíduo gerado é imediatamente avaliado e testado para inserção na população parental, de acordo com algum critério como, por exemplo, a substituição do pior indivíduo ou a substituição do indivíduo mais velho da população.

Neste trabalho, foi implementado um AG geracional elitista, onde a cada geração os dois melhores indivíduos são copiados para a geração seguinte. Após o

processo reprodutivo, os $\mu - 2$ melhores filhos são inseridos na nova população.

3.2.8 Condições de parada

A escolha do critério de parada é mais um fator de fundamental importância para o sucesso de um AG. Por exemplo, se o critério escolhido tornar a execução muito curta, pode-se não encontrar a solução ótima, enquanto execuções muito longas exigem alto custo computacional. Algumas das estratégias que podem ser adotadas são:

- número máximo de gerações;
- não há melhoria na função aptidão;
- não há diversidade na população.

O critério de parada adotado neste trabalho é a escolha do número de gerações. Assim pode-se determinar o orçamento, ou seja, o número máximo de avaliações da função aptidão que serão feitas, o que ajuda a calcular o custo computacional de uma execução do algoritmo.

3.3 Experimentos de validação

Os experimentos deste capítulo foram feitos com o objetivo de validar as escolhas do método de otimização e a implementação realizada. O problema de modelagem estudado é simples, as situações problemas são hipotéticas e foram criadas para permitir esta validação.

Para cada um dos experimentos, a otimização foi executada 13 vezes, variando a semente utilizada para a geração dos números aleatórios usados pelo AG. A população inicial, gerada aleatoriamente, será diferente para cada semente usada. É esperado de um AG que, partindo de populações iniciais diferentes chegue ao mesmo resultado ou a resultados semelhantes. Este tipo de teste demanda muito tempo de computação e a avaliação paralela das soluções candidatas foi fundamental para a redução do tempo de resposta.

O problema de contaminação usado nos experimentos de validação é o apresentado na Seção 2.4 do Capítulo 2, com algumas variações:

- na localização da pluma de contaminante: à esquerda e à direita da posição inicial;
- na localização dos poços: a região para colocação de poços foi restrita, conforme indicado na Figura 3.5;
- nos valores possíveis para as vazões dos poços: 100, 150, 200, 250, 300, 325, 350, 375, 400, 425 m^3/dia.

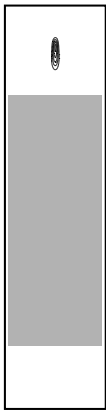


Figura 3.5: Modelo do aquífero com a pluma de contaminante e área para colocação dos poços remediadores.

Deseja-se ressaltar que, na função objetivo, os valores utilizados para a penalidade por não descontaminação e para o custo dos poços não são realistas.

3.3.1 Pluma no centro do aquífero, um (1) poço, vazão fixa

No problema deste experimento, a pluma de contaminante está localizada na parte superior do aquífero, com o centro igualmente distante dos lados esquerdo e direito do mesmo. Deseja-se retirar o contaminante com a colocação de um poço de extração, e neste experimento forçou-se a escolha da vazão mais alta - $425m^3/dia$. Assim, com as restrições impostas, para este problema de otimização existem

$$193 \text{ linhas} * 79 \text{ colunas} * 1 \text{ vazão} * 1 \text{ poço} = 15.247 \text{ possíveis soluções.}$$

Como há apenas um valor de vazão disponível, a parcela da função objetivo referente ao custo dos poços será a mesma para todas as soluções. Neste caso, a parcela correspondente à penalidade será determinante e o que se busca é a localização do poço que retira mais contaminante.

3.3.1.1 Calibração dos parâmetros

Para a escolha dos parâmetros usados na otimização, foram testados os diferentes conjuntos listados na Tabela 3.2. Essa fase de calibração dos parâmetros, mostrada apenas aqui, foi feita para todos os experimentos. A escolha dos parâmetros, por si só, pode tornar-se um problema de otimização e poderia justificar um estudo mais aprofundado, o que não foi feito.

Parâmetros da Otimização	**Exp 1.1**	**Exp 1.2**	**Exp 1.3**	**Exp 1.4**
Nº de gerações	30	40	30	30
População inicial	20	20	50	50
Nº de candidatos por geração	50	50	100	100
MIN (*Linear Ranking*)	0,7	0,3	0,7	0,3
MAX (*Linear Ranking*)	1,3	1,7	1,3	1,7
Taxa de *crossover* (inic/fin)	50% / 50%	75% / 45%	50% / 50%	75% / 45%
Taxa de mut. na loc. (inic/fin)	50% / 50%	25% / 55%	50% / 50%	25% / 55%
Orçamento (nº de sim)	1.520	2.020	3.050	3.050

Tabela 3.2: Calibração de parâmetros do modelo de otimização.

Os resultados da execução da otimização com os quatro conjuntos de parâmetros, cada um executado com 13 sementes diferentes, estão na Tabela 3.3, que apresenta duas colunas para cada conjunto: na primeira coluna apresenta-se o número da geração onde foi encontrada a melhor solução e, na segunda, o número total de avaliações realizadas. Neste problema, após algumas gerações as soluções candidatas tendem a ficar próximas umas das outras e isso faz com que o AG tenha dificuldade para gerar a quantidade pedida de novos indivíduos a cada geração, o que explica o total de avaliações feitas menor que o orçamento previsto.

A solução para este problema é aquela onde o poço fica na linha vertical que divide igualmente o aquífero e na posição mais próxima da pluma, obedecendo

	Exp 1.1		Exp 1.2		Exp 1.3		Exp 1.4	
Semente	nº de ger. até o melhor	nº de aval. total	nº de ger. até o melhor	nº de aval. total	nº de ger. até o melhor	nº de aval. total	nº de ger. até o melhor	nº de aval. total
1	18	1.181	20	1.422	8	2.491	19	2.188
2	—	1.284	–	1.546	18	2.563	11	2.265
3	—	1.248	–	1.503	21	2.488	29	2.287
4	6	1.189	–	1.458	17	2.434	23	2.297
5	—	1.229	39	1.507	–	2.490	—	2.242
6	—	1.206	–	1.454	29	2.532	—	2.257
7	19	1.200	27	1.426	22	2.532	24	2.177
8	29	1.208	36	1.455	27	2.505	26	2.300
9	—	1.234	–	1.508	–	2.467	—	2203
10	—	1.223	–	1.438	14	2.449	12	2.228
11	10	1.197	13	1.337	9	2.442	20	2.238
12	—	1.237	–	1.506	–	2.528	—	2.248
13	—	1.228	–	1.507	15	2.543	18	2.287
	5		**5**		**10**		**9**	

Tabela 3.3: Síntese de resultados para os experimentos que testam diversas combinações de parâmetros.

às restrições impostas. Estamos buscando o conjunto de parâmetros que, dentro de um orçamento razoável, encontre esta solução ótima mais vezes, entre as 13 condições iniciais geradas a partir das diferentes sementes. A última linha da Tabela 3.3 mostra quantas vezes esta solução foi obtida.

Os resultados obtidos no Exp 1.3 parecem mais adequados e serão analisados com mais detalhes na seção seguinte.

3.3.1.2 Análise dos resultados do Exp 1.3

A Tabela 3.4 mostra a localização do poço, o volume de poluente no aquífero após o processo de descontaminação e o percentual do volume de contaminante retirado. Verifica-se que todas as soluções encontradas, mesmo nos casos onde não foi obtida "a melhor", são boas soluções e removem mais de 99% do volume de contaminante.

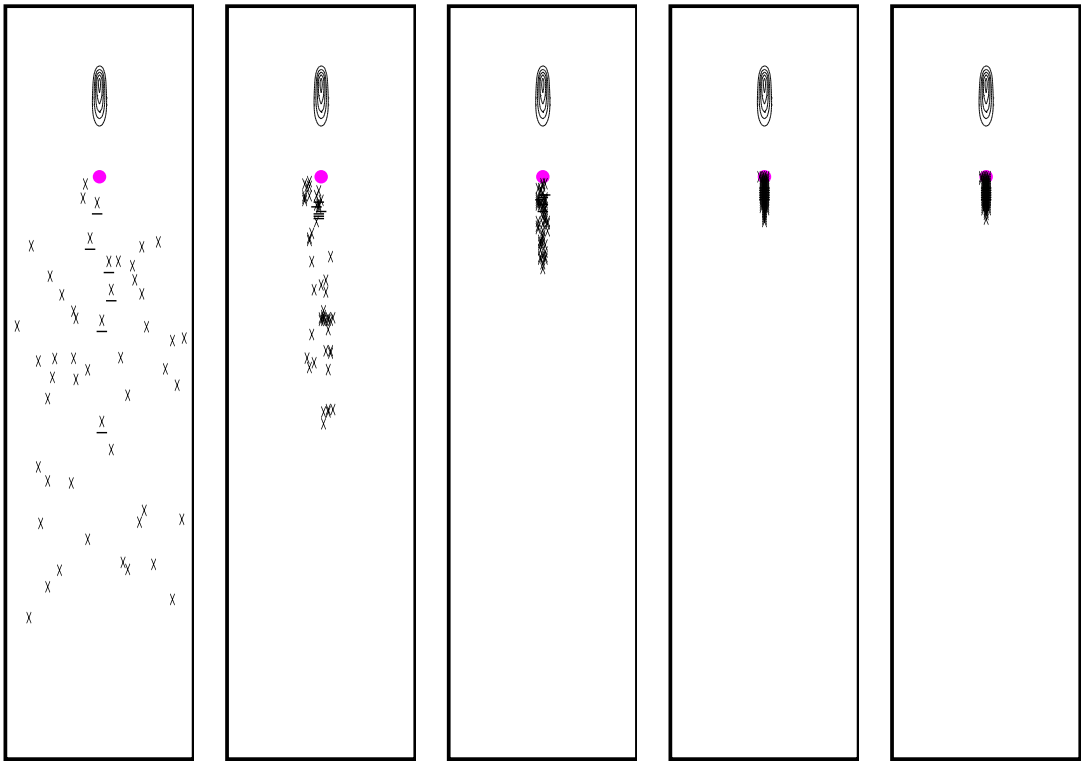
As Figuras 3.6, 3.7 e 3.8 mostram a evolução das soluções em três casos. Nessas figuras, o círculo abaixo da pluma indica o local da solução ótima e, a cada

Semente	Localização (x,y)	Volume Residual	Percentual Retirado
1	(41,249)	0,758	99,96%
2	(41,249)	0,758	99,96%
3	(41,249)	0,758	99,96%
4	(41,249)	0,758	99,96%
5	(41,238)	2,226	99,88%
6	(41,249)	0,758	99,96%
7	(41,249)	0,758	99,96%
8	(41,249)	0,758	99,96%
9	(41,227)	4,708	99,77%
10	(41,249)	0,758	99,96%
11	(41,249)	0,758	99,96%
12	(41,244)	1,334	99,93%
13	(41,249)	0,758	99,96%

Tabela 3.4: Síntese dos resultados obtidos com 13 sementes, para o Exp 1.3.

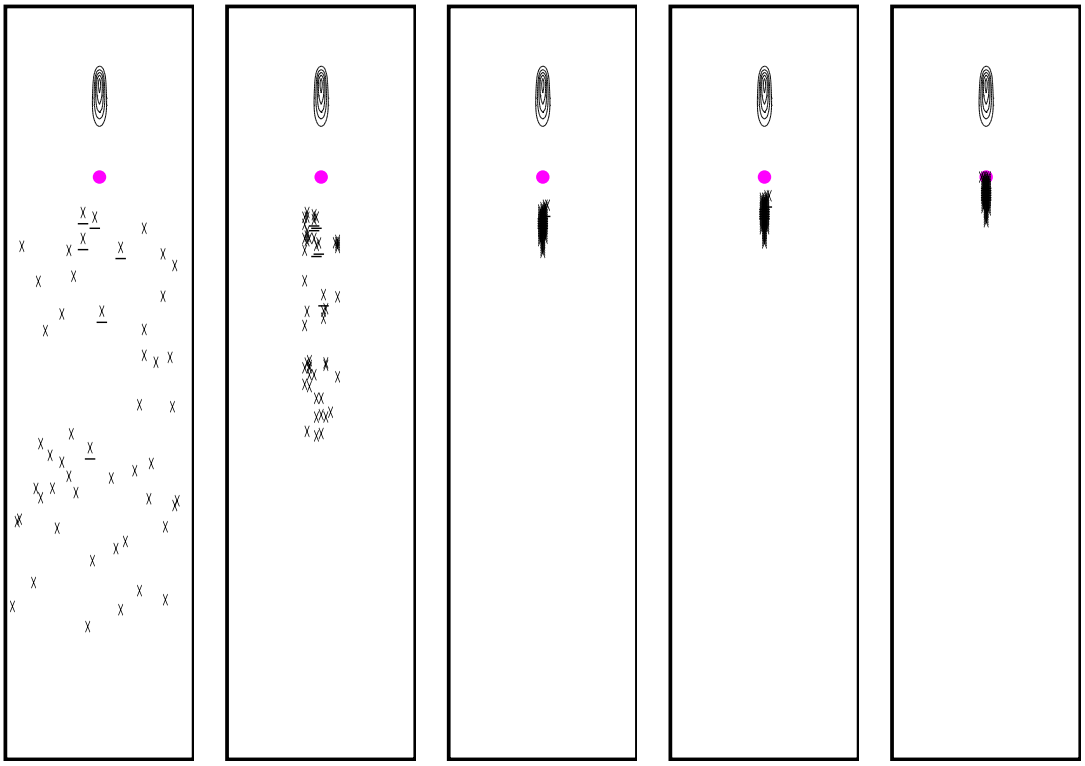
geração, as seis melhores soluções candidatas são representadas por um "X" sublinhado. Pode-se notar que a distribuição das soluções candidatas na população inicial - fator dependente da semente usada para o gerador de números aleatórios - determina quão rapidamente a solução ótima é encontrada. No primeiro caso, a melhor solução foi encontrada na oitava geração; no segundo, só no final, na penúltima geração; no terceiro caso, a melhor solução não foi encontrada, porém provavelmente covergiria se fossem executadas mais gerações. É importante ressaltar que esta é uma solução boa (retira 99,77% de contaminante) mas, para estes experimentos de validação o critério de convergência utilizado é rígido.

As Figuras 3.9(a) e 3.9(b), mostram, respectivamente, os gráficos do volume residente de contaminante no aquífero e da concentração de contaminante no poço de extração, e pode-se verificar o resultado satisfatório obtido utilizando-se a solução ótima encontrada pela otimização.



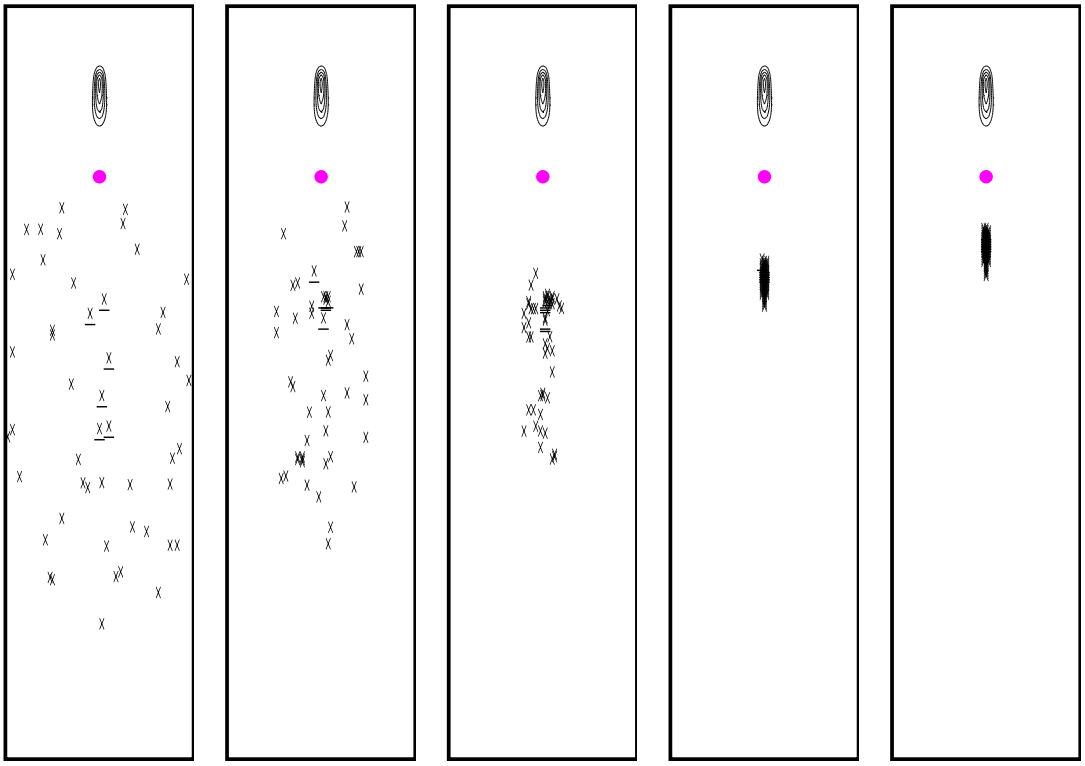
(a) pop inicial (b) ger 2 (c) ger 4 (d) ger 8 (e) pop final

Figura 3.6: Exp 1.3: evolução da população (semente 1).



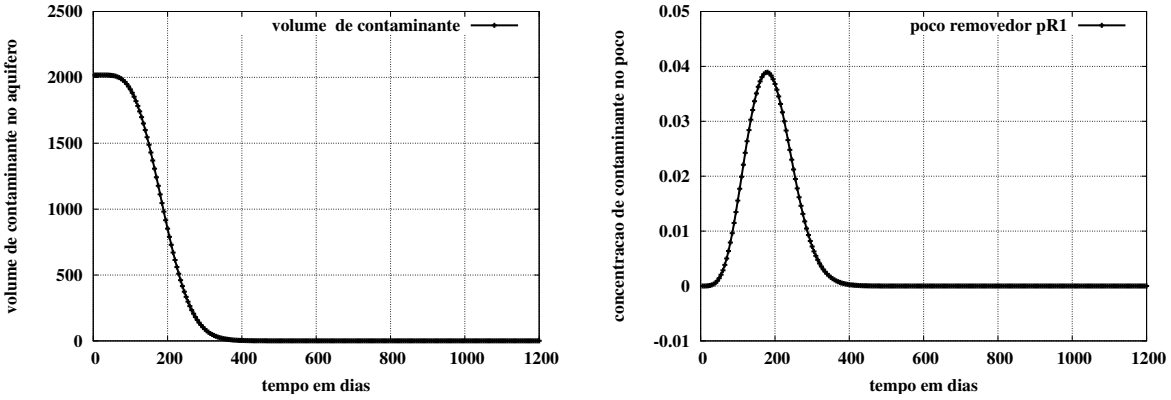
(a) pop inicial (b) ger 2 (c) ger 10 (d) ger 15 (e) pop final

Figura 3.7: Exp 1.3: evolução da população (semente 6).



(a) pop inicial (b) ger 1 (c) ger 2 (d) ger 10 (e) pop final

Figura 3.8: Exp 1.3: evolução da população (semente 9).



(a) Volume de contaminante no aquífero (b) Concentração de contaminante no poço de extração

Figura 3.9: Resultados da remediação com a melhor solução encontrada.

3.3.2 Pluma à esquerda, um (1) poço, vazão fixa

Neste experimento, o problema de otimização usa os mesmos parâmetros e tem as mesmas características apresentadas na seção anterior. A variação feita é a mudança da localização da pluma, deslocada cinco nós para a esquerda (125m).

A Tabela 3.5 mostra a síntese dos resultados obtidos, onde observa-se:

- a nova localização do poço, cinco nós para a esquerda (conforme esperado);
- 11 "sementes" encontram a solução ótima (última linha);
- como no experimento anterior, as soluções encontradas conseguem retirar mais de 99% do contaminante;
- número de avaliações total é inferior ao orçamento.

Sem.	Loc. (x,y)	Vol. Res.	Perc. Remov.	Geração onde encontrou o melhor	Nº de avaliações total
1	(36,249)	0,9138	99,96%	6	2.442
2	(36,249)	0,9138	99,96%	11	2.530
3	(36,249)	0,9138	99,96%	2	2.477
4	(36,249)	0,9138	99,96%	12	2.487
5	(36,247)	1,158	99,94%	–	2.629
6	(36,249)	0,9138	99,96%	20	2.549
7	(36,249)	0,9138	99,96%	15	2.540
8	(36,249)	0,9138	99,96%	18	2.509
9	(36,249)	0,9138	99,96%	24	2.555
10	(36,249)	0,9138	99,96%	17	2.508
11	(36,249)	0,9138	99,96%	1	2.491
12	(35,243)	1,674	99,92%	–	2.580
13	(36,249)	0,9138	99,96%	11	2.505
				11	

Tabela 3.5: Síntese dos resultados obtidos com 13 sementes, para o experimento com a pluma à esquerda, vazão fixa.

Estes resultados são considerados satisfatórios e demonstram a estabilidade do método.

3.3.3 Pluma à direita, um (1) poço, variação da vazão

Neste experimento, a pluma foi deslocada cinco nós para a direita em relação a pluma do experimento da Seção 3.3.1. Aqui também é feita uma restrição no número de poços usados (apenas um), mas a escolha da vazão é livre, podendo ser escolhidas quaisquer uma das dez disponíveis, existindo, então:

$$193 \text{ linhas} * 79 \text{ colunas} * 10 \text{ vazões} * 1 \text{ poço} = 152.470 \text{ possíveis soluções.}$$

Devido ao aumento na quantidade de soluções, o orçamento aumentou para 10.550, sendo distribuído em 50 indivíduos na população inicial, seguidos de 30 gerações com 350 indivíduos. Os parâmetros utilizados estão na Tabela 3.6.

Para avaliar a influência das parcelas (referentes ao custo dos poços e à penalidade por quantidade de contaminante não removido) na função objetivo, foram realizados experimentos com dois cenários, variando o valor da penalidade:

- cenário M1: com penalidade por não descontaminação mais alta (10^7);
- cenário M2: com penalidade reduzida (10^6).

Parâmetros da Otimização	
Nº de gerações	30
População inicial	50
Nº de candidatos por geração	350
MIN (*Linear Ranking*)	0,8
MAX (*Linear Ranking*)	1,2
Taxa de *crossover* (inic/fin)	65% / 30%
Taxa de mut. na loc. (inic/fin)	25% / 55%
Taxa de mut. na vazão (inic/fin)	15% / 15%
Orçamento (nº de sim)	10.550

Tabela 3.6: Parâmetros da otimização do experimento com a pluma à direita.

3.3.3.1 Cenário M1

A Tabela 3.7 mostra uma síntese das soluções obtidas no cenário com a penalidade mais alta, apresentando dados sobre a localização e a vazão dos poços, o volume de contaminante residual, o percentual de contaminante removido do aquífero, a geração onde foi encontrada a solução ótima e o número total de avaliações feitas nas 30 gerações.

Entre as soluções com as 13 sementes, 12 chegaram ao local esperado: na posição mais próxima da pluma, respeitando as restrições impostas, e na linha vertical correspondente à trajetória do centro da pluma que, neste experimento foi deslocada cinco nós para direita. A vazão escolhida, em todos os casos, foi a mais alta: $425m^3/dia$, sendo removido 99% da quantidade de poluente inicial.

Sem.	Loc. (x,y)	Vazão	Vol. Res.	Perc. Remov.	Geração onde encontrou o melhor	Nº de avaliações total
1	(46,249)	425	0,913	99,96%	28	4.351
2	(46,249)	425	0,913	99,96%	14	4.314
3	(46,249)	425	0,913	99,96%	11	4.142
4	(46,249)	425	0,913	99,96%	4	3.942
5	(47,232)	425	3,692	99,82%	–	4.642
6	(46,249)	425	0,913	99,96%	29	4.195
7	(46,249)	425	0,913	99,96%	22	4.244
8	(46,249)	425	0,913	99,96%	24	4.252
9	(46,249)	425	0,913	99,96%	26	4.179
10	(46,249)	425	0,913	99,96%	18	4.331
11	(46,249)	425	0,913	99,96%	9	3.994
12	(46,249)	425	0,913	99,96%	16	4.462
13	(46,249)	425	0,913	99,96%	16	4.274
					12	

Tabela 3.7: Síntese dos resultados obtidos com 13 sementes, para o experimento com a pluma à direita, cenário M1.

Na Tabela 3.8, são apresentadas para algumas soluções encontradas com a semente 1, o valor de sua aptidão e as duas parcelas usadas no cálculo desta: o custo dos poços e a penalidade por não-descontaminação. Observa-se que a utilização de uma valor alto para a penalidade (10^7) faz com que esta segunda parcela tenha peso alto forçando, desta forma, em todos os casos, a escolha da vazão mais alta. Deseja-se verificar se, com a utilização de uma penalidade menor, vazões menores são escolhidas, situação que será examinada no cenário M2.

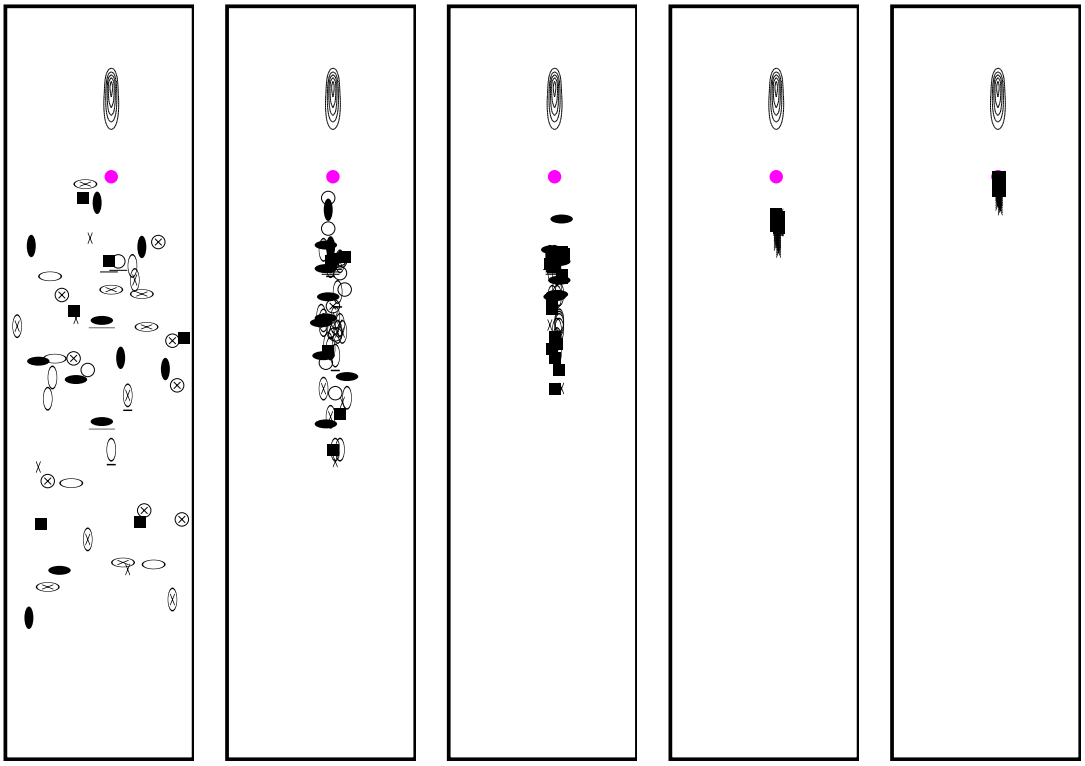
Loc. (x,y)	Vazão	Vol. Res.	Custo dos Poços	Penalidade	Aptidão
(46,249)	425	0,9138	15.000,00	45.281,00	60.281,00
(46,248)	425	0,9435	15.000,00	46.753,00	61.753,00
(47,249)	425	1,292	15.000,00	64.021,00	79.021,00
(46,249)	425	1,586	15.000,00	78.594,00	93.594,00
(46,248)	425	1,674	15.000,00	82.951,00	97.951,00

Tabela 3.8: Cálculo da função aptidão de algumas soluções, cenário M1.

As Figuras 3.10 e 3.11 mostram a evolução da população gerada com a semente 1 e 5, respectivamente. As vazões são representadas por símbolos diferentes, conforme as legendas apresentadas na Tabela 3.9. As cinco melhores soluções de cada geração têm seus símbolos sublinhados. No primeiro caso a solução ótima foi encontrada, mas no segundo não. Nesta última situação, da mesma forma que na seção 3.3.1.2, deseja-se ressaltar que a qualidade da solução obtida é boa - o percentual de contaminante retirado é 99,82% - mas o critério de convergência usado é rígido.

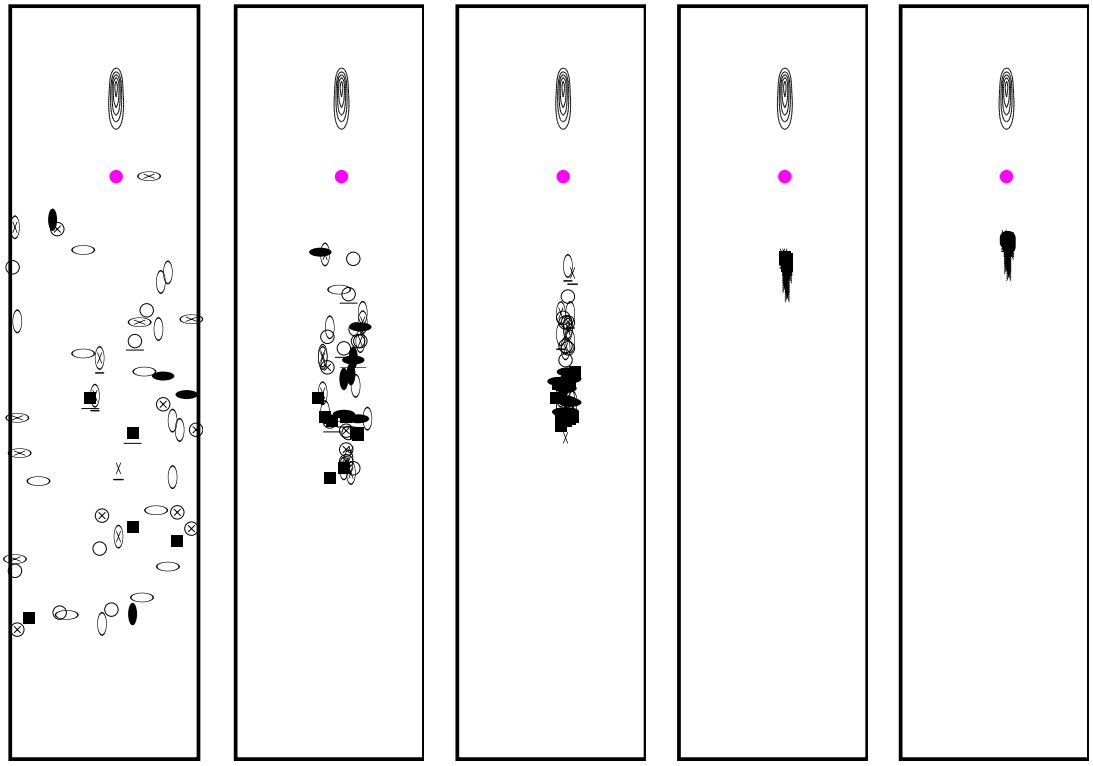
Vazão	100	150	200	250	300	325	350	375	400	425
Símbolo	⊗	⬭	▮	⊗	○	⊗	▬	○	■	x

Tabela 3.9: Legenda das vazões dos poços.



(a) pop inicial (b) ger 1 (c) ger 2 (d) ger 10 (e) pop final

Figura 3.10: Pluma à direita, cenário M1: evolução da população (semente 1).



(a) pop inicial (b) ger 1 (c) ger 2 (d) ger 10 (e) pop final

Figura 3.11: Pluma à direita, cenário M1: evolução da população (semente 5).

3.3.3.2 Cenário M2

Neste cenário a penalidade sobre o contaminante não-retirado é reduzida e deseja-se verificar se o método irá escolher uma solução com vazão menor e com menor custo.

Na Tabela 3.10, com dados semelhantes aos da Tabela 3.7, verifica-se que a localização ótima é encontrada 11 vezes, dentre as 13 sementes usadas. Com a redução no valor da penalidade sobre o volume residual, foram escolhidas soluções com vazões menores, deixando no aquífero um volume residual um pouco maior.

Apesar dos valores escolhidos tanto para o custo quanto para a penalidade não serem realísticos foi possível demonstrar que o resultado do modelo de otimização é sensível às modificações feitas e verificar que a alteração nos parâmetros da função objetivo altera a resposta encontrada. Diminuir o valor da penalidade por não retirar o contaminante leva o algoritmo genético a encontrar uma solução ótima com poços de menor custo, com valor de vazão menor que o máximo,

diferente do que ocorre no experimento do cenário M1.

Sem.	Loc. (x,y)	Vazão	Vol. Res.	Perc. Remov.	Geração onde encontrou o melhor	Nº de avaliações total
1	(46,249)	350	5,94	99,71%	26	5.355
2	(46,249)	350	5,94	99,71%	21	5.264
3	(46,249)	350	5,94	99,71%	5	5.179
4	(46,249)	350	5,94	99,71%	5	5.281
5	(46,247)	350	6,953	99,66%	—	5.485
6	(46,249)	400	1,642	99,92%	27	4.277
7	(46,249)	350	5,94	99,71%	22	5.444
8	(46,246)	350	7,49	99,63%	—	5.113
9	(46,249)	350	5,94	99,71%	30	5.263
10	(46,249)	350	5,94	99,71%	16	5.246
11	(46,249)	350	5,94	99,71%	5	5.268
12	(46,249)	350	5,94	99,71%	19	5.413
13	(46,249)	350	5,94	99,71%	13	5.562
					11	

Tabela 3.10: Síntese dos resultados obtidos com 13 sementes, para o experimento com a pluma à direita, cenário M2.

3.3.4 Pluma no centro, dois (2) poços, variação da vazão

Neste problema a pluma de contaminante está localizada no centro do aquífero, assim como no experimento da Seção 3.3.1. Os valores da lista de vazões permitidas variam entre 100 e $250 m^3/dia$, num total de quatro opções, e podem ser escolhidas soluções com um ou dois poços. Combinando esses números, chega-se a um total de:

$$\binom{15.247}{2} \times 4^2 + \binom{15.247}{1} \times 4^1 = 1.8 \times 10^6 \text{ soluções candidatas.} \qquad (3.5)$$

As 13 otimizações foram feitas usando os parâmetros da Tabela 3.11. Neste experimento aplica-se também o operador mutação no número de poços, mas com probabilidade de escolha baixa (5%). Apesar do aumento no número de soluções viáveis, manteve-se o orçamento em 10.550 avaliações, o mesmo usado nos experimentos da seção anterior.

Parâmetros da Otimização	
Nº de gerações	30
População inicial	50
Nº de candidatos por geração	350
MIN (*Linear Ranking*)	0,8
MAX (*Linear Ranking*)	1,2
Taxa de *crossover* (inic/fin)	65% / 25%
Taxa de mut. na loc. (inic/fin)	25% / 65%
Taxa de mut. na vazão (inic/fin)	5% / 5%
Taxa de mut. nº de poços (inic/fin)	5% / 5%
Orçamento (nº de sim)	10.550

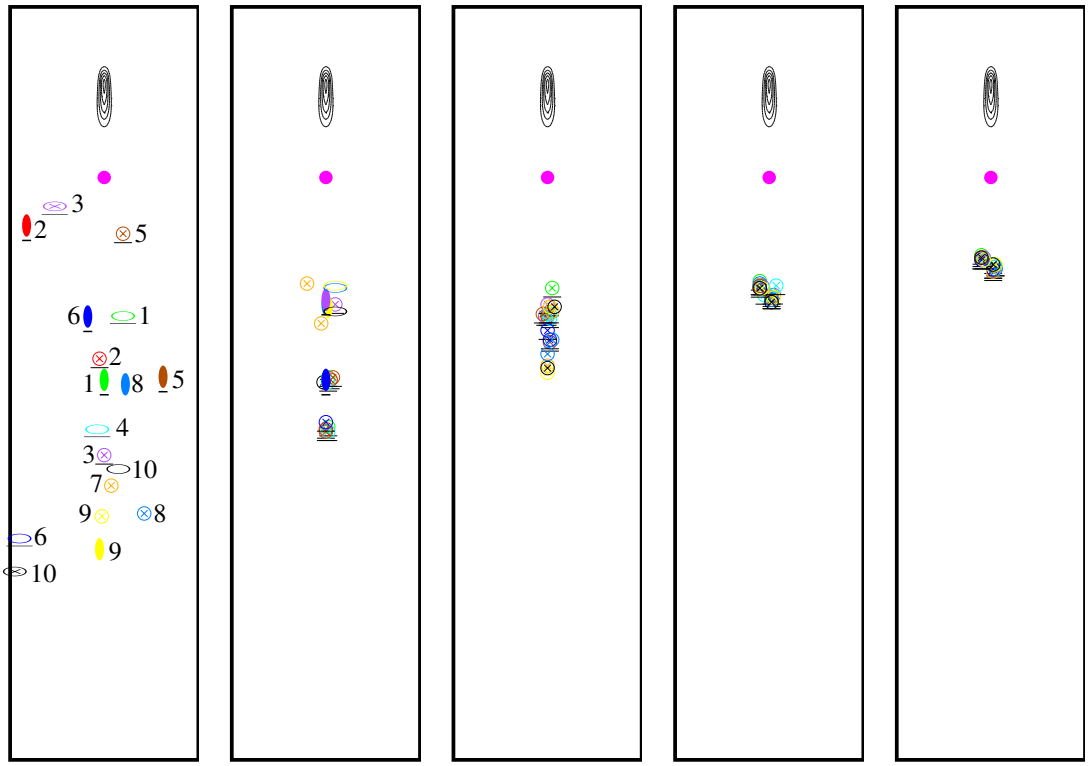
Tabela 3.11: Parâmetros da otimização do experimento com a pluma no centro, dois poços.

Uma síntese dos resultados das otimizações feitas está na Tabela 3.12, que apresenta a localização e a vazão dos poços, o volume de contaminante residual no aquífero e o percentual de contaminante removido pelos poços de extração, a geração onde foi encontrada a solução indicada e o número total de avaliações feitas. As soluções encontradas ficaram todas com dois poços. Para todos os poços foi escolhida a vazão máxima permitida ($250m^3$). As localizações dos poços ficaram no eixo longitudinal onde está localizado o centro da pluma. Isso era o esperado. As soluções obtidas conseguem retirar cerca de 99% do contaminante, o que é satisfatório, mas vale ressaltar que os poços mais próximos da pluma (em relação à altura) obtiveram melhores resultados, deixando um volume residual mais baixo. Observando-se a última coluna da tabela, nota-se que o número de candidatos avaliados foi maior que no experimento anterior, o que pode ser explicado pelo aumento no número de soluções possíveis, calculada na expressão 3.5.

Na Figura 3.12, pode-se acompanhar a evolução da população gerada com o uso da semente 5. Os poços de uma mesma solução têm cores iguais e na população inicial (3.12(a)) também são inidicados por números iguais. As legendas para as vazões estão na Tabela 3.9 e as melhores soluções de cada geração têm seus símbolos sublinhados. Verifica-se que já nas gerações iniciais a vazão dos poços converge para a mais alta permitida.

Sem.	Poço1 (x,y)	Vazão	Poço2 (x,y)	Vazão	Vol. Res.	Perc. Remov.	Geração onde achou a solução	Nº de avaliações total
1	(41,207)	250	(40,230)	250	2,063	99,898%	30	8.273
2	(42,230)	250	(38,239)	250	0,037	99,998%	27	8.800
3	(42,228)	250	(38,237)	250	0,163	99,991%	28	10.189
4	(41,184)	250	(41,248)	250	3,429	99,830%	29	9.241
5	(42,212)	250	(37,216)	250	1,724	99,914%	30	10.469
6	(41,230)	250	(38,244)	250	0,285	99,986%	30	8.539
7	(40,225)	250	(44,232)	250	0,466	99,977%	30	9.772
8	(41,222)	250	(42,247)	250	0,716	99,964%	30	9.012
9	(41,191)	250	(41,237)	250	3,405	99,831%	30	8.641
10	(42,227)	250	(38,235)	250	0,247	99,988%	30	10.378
11	(43,241)	250	(39,246)	250	0,001	99,999%	16	10.524
12	(42,232)	250	(38,238)	250	0,091	99,995%	30	10.252
13	(41,199)	250	(41,233)	250	2,718	99,865%	30	8.337

Tabela 3.12: Síntese dos resultados para a pluma no centro com dois poços.

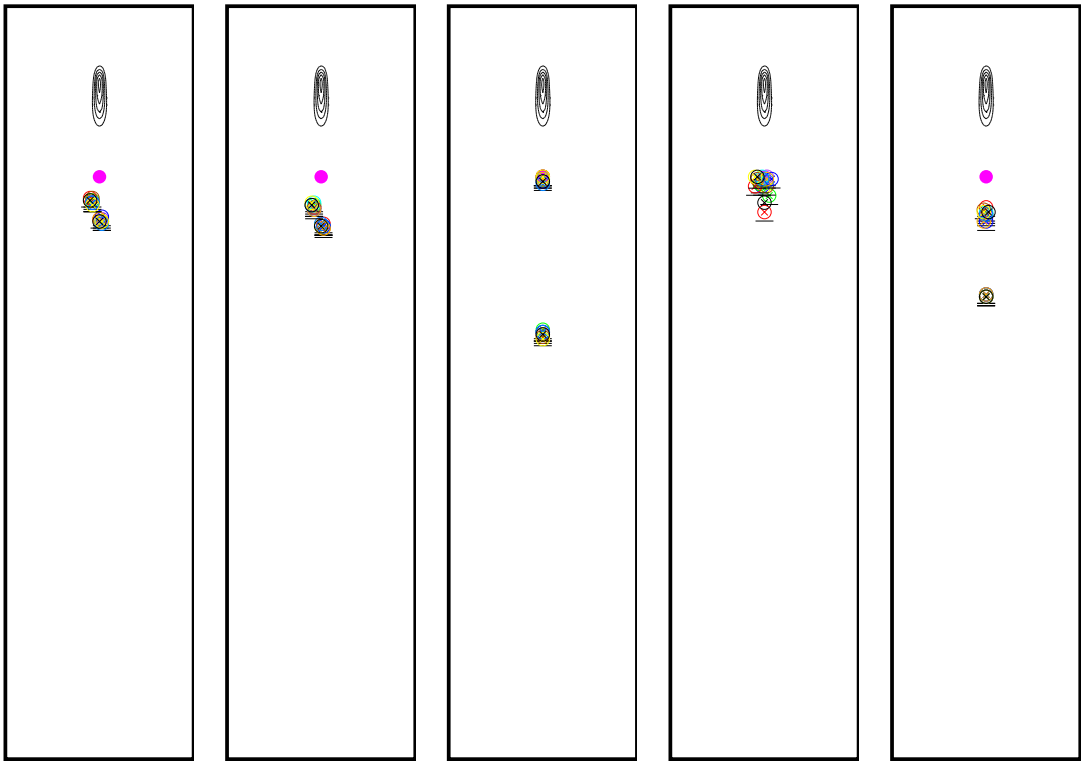


(a) pop inicial (b) ger 2 (c) ger 4 (d) ger 10 (e) pop final

Figura 3.12: Pluma no centro, dois poços: evolução da população (semente 5).

Os cinco melhores indivíduos da população final de algumas sementes são apresentados na Figura 3.13. Dentre essas, os melhores resultados foram obtidos

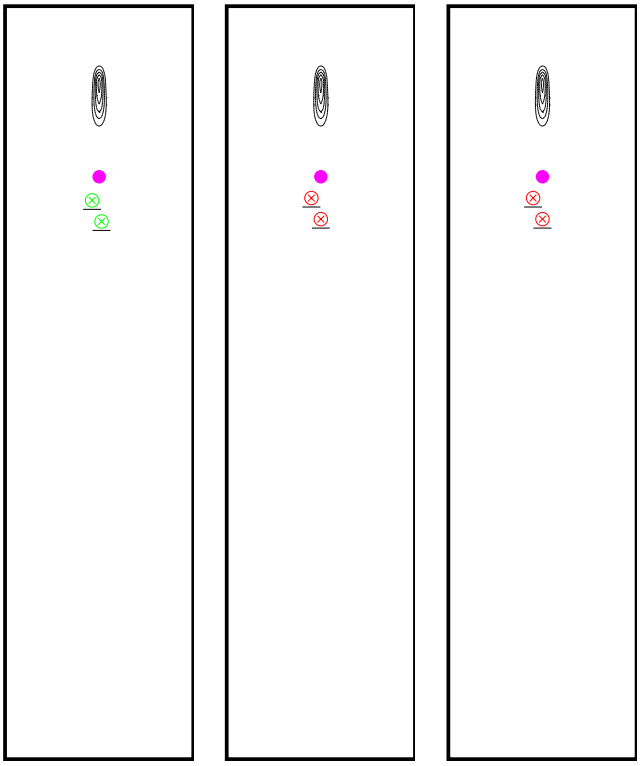
com a semente 11, onde a solução ótima deixa no aquífero um volume residual de 10^{-3}; os piores, com a 4, onde a melhor solução deixa volume residual de 3,429.



(a) sem. 2 (b) sem. 3 (c) sem. 4 (d) sem. 11 (e) sem. 13

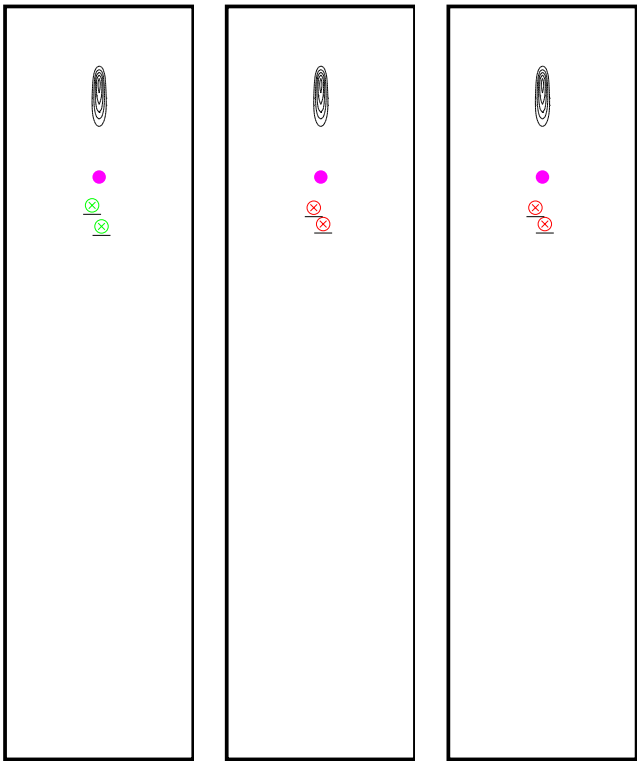
Figura 3.13: Pluma no centro, dois poços: população final (sementes variadas).

As Figuras 3.14, 3.15 e 3.16 mostram os três melhores indivíduos obtidos com as sementes 2, 3 e 11 respectivamente.



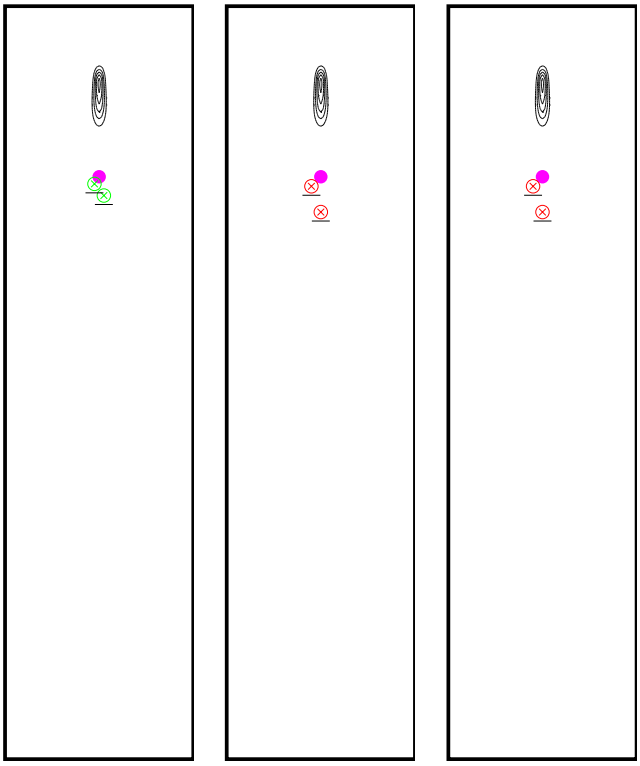
(a) indiv. 1 (b) indiv. 2 (c) indiv. 3

Figura 3.14: Pluma no centro, dois poços: três melhores indivíduos (semente 2).



(a) indiv. 1 (b) indiv. 2 (c) indiv. 3

Figura 3.15: Pluma no centro, dois poços: três melhores indivíduos (semente 3).



(a) indiv. 1 (b) indiv. 2 (c) indiv. 3

Figura 3.16: Pluma no centro, dois poços: três melhores indivíduos (semente 11).

3.4 Paralelismo dos algoritmos genéticos

Apesar dos algoritmos genéticos serem criticados pelo alto volume de avaliações necessárias, estes possuem a característica de serem naturalmente paralelizáveis; sendo assim, as muitas avaliações dos indivíduos - tarefas com custo computacional elevado - podem ser realizadas concorrentemente.

O uso do paralelismo foi fator fundamental para obtenção dos resultados. O tempo de execução dos experimentos de validação e o número de núcleos utilizados são apresentados na Tabela 3.13. A escolha da quantidade de núcleos foi feita baseada na disponibilidade dos computadores e no número de indivíduos a serem avaliados a cada geração.

Experimento	Nº de núcleos	T_{exec} (13 sementes)
pluma no centro, 1 poço (exp 1.3)	35	15h
pluma à esquerda, 1 poço	15	18h
pluma à direita, cenário M1	47	16h
pluma à direita, cenário M2	136	9h
pluma no centro, 2 poços	59	24h

Tabela 3.13: Tempos da execução paralela de alguns experimentos de validação.

No capítulo seguinte será apresentado o sistema computacional de otimização por simulação onde as avaliações são realizadas em paralelo, sendo feita uma análise de seu desempenho.

Capítulo 4

Sistema Distribuído de Simulações Numéricas

Geralmente, os métodos evolutivos de otimização por simulação demandam um número elevado de execuções do programa que implementa o modelo computacional do fenômeno em estudo. Contudo, uma execução é independente da outra e este fato permite explorar a concorrência nesta etapa, que é a de maior custo computacional desta metodologia. Realizar as simulações em paralelo permite diminuir o tempo de resposta e torna possível alguns casos mais complexos de modelagem.

Este capítulo apresenta um sistema distribuído que implementa o algoritmo evolutivo apresentado no capítulo anterior, com ênfase na maneira como a avaliação das soluções candidatas é feita: concorrentemente, utilizando computadores interligados por rede. Para tanto, o sistema deve ser capaz de realizar as seguintes tarefas comuns à computação paralela:

- definir um conjunto de tarefas que possam ser executadas concorrentemente;
- iniciar e finalizar a execução destas tarefas;
- coordenar a interação entre as tarefas.

Devido à trivialidade do paralelismo desta aplicação, a sua programação seria também "trivial" se os equipamentos usados para executá-la fossem dedicados

e homogêneos. Mas escolheu-se desenvolver um sistema preparado para utilizar computadores que formam um conjunto heterogêneo e não-dedicado. Assim sendo, as etapas de identificação dos equipamentos disponíveis e do particionamento adequado da carga para obtenção de alto desempenho merecerão uma atenção especial.

Quanto à otimização, uma vez que o algoritmo evolutivo adotado é geracional e síncrono, é esperado que o resultado obtido usando vários processadores seja idêntico ao obtido se a execução for sequencial.

Nas seções seguintes são apresentados: a organização do sistema em módulos, como o balanceamento de carga foi introduzido nesse sistema, como foi implementado um mecanismo de proteção contra falhas, os tipos de ambientes onde o sistema foi executado e, finalmente, uma análise de desempenho baseada nos experimentos realizados.

4.1 Arquitetura mestre/escravo

Na busca da solução ótima, o algoritmo de otimização precisa avaliar, via simulação numérica, um grande número de soluções candidatas. Neste sistema, essas simulações são feitas concorrentemente, utilizando diversos processadores. O modelo de programação mestre/escravo é adequado para este tipo de problema pois as simulações são tarefas independentes [Mattson et al. (2004)].

O sistema implementado é constituído por três módulos principais:

- **mestre**: escrito na linguagem Java, identifica os agentes disponíveis, executa o algoritmo de otimização e é responsável pela partição e distribuição das simulações;

- **agente**: também escrito em Java, o agente é replicado nos nós computacionais disponíveis e é responsável pela chamada ao simulador e cálculo da aptidão (função objetivo), além de retornar ao mestre informação sobre seu desempenho;

- **simulador numérico**: programa em Fortran que resolve o modelo mate-

mático do aquífero contaminado.

O agente desempenha o papel do escravo e esta denominação se aplica devido à sua autonomia: o agente é acionado de forma independente do mestre e, após o término deste, permanece em execução podendo ser utilizado em outras simulações; o agente, porém, não deve ser entendido como módulo inteligente.

Estes três programas são portáveis. Essa portabilidade é obtida pelo uso de Java e pela existência de uma versão do simulador para cada ambiente. Java foi utilizado pela sua portabilidade e o programa em Fortran já existia [Loula et al. (1999)].

O diagrama da arquitetura é apresentado na Figura 4.1.

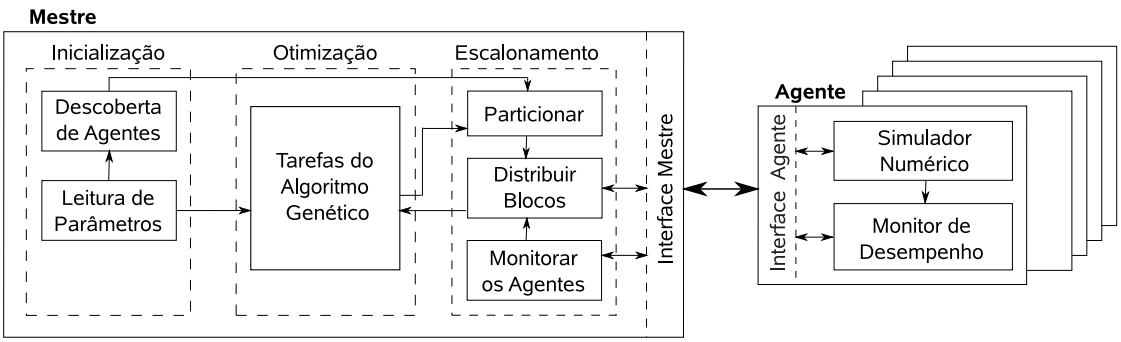


Figura 4.1: Arquitetura de implementação.

Nesta implementação, uma máquina é escolhida para executar o módulo mestre e o módulo agente é replicado nos nós computacionais disponíveis. Mestre e agente comunicam-se através de método de invocação remota (RMI), um recurso da linguagem Java que permite que uma *thread* invoque um método em um objeto remoto (os objetos são considerados remotos se residirem em uma máquina virtual Java distinta, podendo estar no mesmo computador ou num *host* remoto) [Li and Baker (2005)] e [Silberschatz et al. (2004)]. O RMI permite aos desenvolvedores manipular objetos remotos muito similarmente ao modo como são manipulados objetos locais, escondendo os detalhes da implementação.

De acordo com [Dong and Akl (2006)] são consideradas as seguintes definições:

- aplicação: uma instância do problema de otimização e sua solução;
- tarefa: a avaliação de um indivíduo;
- bloco: grupo de tarefas.
- *job*: a avaliação de uma população;

O mestre implementa os passos do método de otimização apresentados no Algoritmo 1 do Capítulo 3. No momento da avaliação das soluções candidatas - a parte com custo computacional alto - o mestre particiona o *job* em blocos de tarefas de acordo com um algoritmo de escalonamento escolhido e distribui os blocos entre os agentes. Os agentes avaliam cada indivíduo do bloco recebido executando o simulador numérico e calculando a aptidão do indivíduo de acordo com os resultados produzidos pela simulação. Quando um agente termina de avaliar o bloco recebido, retorna ao mestre os resultados obtidos e solicita um novo bloco. No final de cada geração, o mestre classifica todas as soluções candidatas de acordo com a aptidão calculada, selecionando então os indivíduos que vão sobreviver, conforme descrito na Seção 3.2.7 do Capítulo 3.

4.2 Balanceamento de carga

Para aplicações que são naturalmente paralelas, a utilização de múltiplos processadores irá certamente diminuir o tempo de resposta. Entretanto, para a obtenção de alto desempenho é necessário que os múltiplos processadores processem suas cargas em tempos semelhantes, evitando tempo ocioso provocado pela necessidade de espera em locais que exigem sincronismo, como ocorre no final da avaliação dos indivíduos de cada geração.

A distribuição de carga entre os processadores participantes da aplicação tem grande importância na computação paralela, em particular nos sistemas heterogêneos e deve considerar a capacidade de cada processador ao atribuir a carga de trabalho. Se esta premissa for ignorada, processadores mais rápidos terão alto

tempo ocioso, aguardando a finalização do trabalho pelos processadores mais lentos, o que resulta em baixa taxa de utilização.

A utilização (U) é uma métrica de desempenho de sistemas definida por [Jain (1991)]:

$$U = \frac{T_{ocupado}}{T_{total}} \quad \text{e} \quad T_{total} = T_{ocupado} + T_{ocioso} \ , \tag{4.1}$$

onde T_{total} é o tempo total, $T_{ocupado}$ é o tempo que o processador foi utilizado pela aplicação e T_{ocioso}, o tempo que o processador ficou ocioso.

Uma maneira adequada de balancear a carga é o uso de métodos de escalonamento [Cheng et al. (2004)], [Penmatsa et al. (2007)], [Yang et al. (2007)] e [Oliveira et al. (2004)].

4.2.1 Escalonamento

O termo escalonamento refere-se ao modo como as tarefas são atribuídas às CPUs disponíveis. Em geral, o escalonamento é feito com um determinado objetivo. Os dois grupos ligados à computação distribuída - os consumidores de recursos (que submetem aplicações) e os provedores de recursos (que compartilham os recursos) - têm motivações diferentes e baseado nisto os algoritmos de escalonamento podem ser classificados em dois tipos: escalonamento voltado para a aplicação e escalonamento voltado para os recursos [Zhu (2003)] e [Dong and Akl (2006)].

Os algoritmos de escalonamento voltados para a aplicação têm por objetivo otimizar o desempenho de uma aplicação individual, sem considerar o desempenho coletivo de todos os recursos do ambiente computacional. O sistema é avaliado em termos de seu impacto na aplicação considerada [Berman and Wolski (1997)]. Na maioria das aplicações deseja-se minimizar o custo da sua execução, em geral o tempo de resposta, mas nas grades computacionais, com a introdução de modelos de economia, minimizar o custo econômico [Buyya et al. (2002)] pela utilização dos recursos também pode ser uma preocupação.

Já os algoritmos de escalonamento voltados para os recursos visam otimizar o desempenho do sistema como um todo, a eficácia no uso dos recursos [Gao et al. (2005)]. O objetivo, em geral, é maximizar a taxa de utilização dos recursos e o throughput (número de resultados produzidos por unidade de tempo). Neste caso há uma visão global da infraestrutura e não somente da aplicação em si.

Escalonadores voltados para o desempenho dos recursos, em particular no caso de grades computacionais, precisam considerar múltiplos fatores, como por exemplo: recursos distribuídos em diferentes domínios administrativos, hetereogeneidade na capacidade e na utilização dos recursos, incluindo redes, computação e dados podem estar em localizações distintas, existência de políticas de reserva antecipada e qualidade de serviço (QoS). A exploração deste tipo de escalonamento não está no escopo deste trabalho, onde foram utilizados métodos de escalonamento voltados para a aplicação.

No sistema proposto, o escalonamento envolve duas etapas: a partição do *job* em blocos de tarefas e a distribuição desses blocos aos processadores disponíveis. Isto pode ser feito de forma estática ou dinâmica [Lu et al. (2007)]. No escalonamento estático, o *job* é dividido pelos processadores, antes do início da execução do laço de repetição (passos 2 e 7 do Algoritmo 1). Este tipo de escalonamento é eficiente sob duas condições: (i) as tarefas têm todas o mesmo custo e (ii) as máquinas são homogêneas e dedicadas.

No sistema apresentado neste trabalho, a primeira condição é satisfeita, mas a segunda não é, sendo mais conveniente usar o escalonamento dinâmico onde o *job* é particionado em blocos de tarefas que são distribuídos durante a execução do laço de repetição. Neste tipo de escalonamento, cria-se uma lista de M blocos de tarefas, com M maior que o número de processadores. No início da distribuição, cada processador recebe um bloco e, ao final do processamento deste bloco, outro bloco da lista é enviado até que todos os blocos tenham sido processados. A ordem em que os blocos serão distribuídos entre os processadores e a quantidade de blocos processada por cada um varia em função da sua capacidade de processamento,

sendo um processo aleatório que pode ser diferente a cada execução da aplicação, mesmo considerando o mesmo conjunto de máquinas.

4.2.1.1 Implementação

No modelo mestre/agente implementado, o mestre tem a função de particionar o *job* em blocos de tarefas e distribuir os blocos. Para o gerenciamento da distribuição destes blocos são criadas três listas com os seguintes nomes:

- **`blocosNaoAvaliados`**;
- **`blocosEmAvaliação`**;
- **`blocosAvaliados`**.

No início de cada geração, todos os blocos são colocadas na lista **`blocosNaoAvaliados`**. Quando um bloco é atribuído a um agente, ele é retirado da lista **`blocosNaoAvaliados`** e colocado na lista **`blocosEmAvaliação`**. Após a realização com sucesso da avaliação dos indivíduos do bloco, este é transferido para a lista de **`blocosAvaliados`**.

O mestre aguarda que os agentes terminem o processamento de um bloco e coloquem-se disponíveis para receber novos blocos, atendendo os primeiros a tornarem-se disponíveis. Devido a este tipo de participação dos agentes, este mecanismo é chamado auto-escalonamento [Chronopoulos et al. (2002)]. Mecanismos de auto-escalonamento podem diferenciar-se quanto à forma com a qual o mestre calcula o tamanho dos blocos de subtarefas [Hummel et al. (1992)], [Chronopoulos et al. (2002)] e [Yang et al. (2007)].

Em [Costa et al. (2008)] são apresentados e comparados resultados de experimentos que utilizam o método de escalonamento estático e alguns métodos de auto-escalonamento dinâmicos. A Tabela 4.1 mostra resultados de experimentos realizados no artigo citado, onde em cada experimento foram executadas 1.120 simulações, divididas em 120 indivíduos da população inicial e mais cinco gerações de 200 indivíduos candidatos cada, num total de seis momentos de sincronismo.

Os processadores são heterogêneos e não estavam executando em sistema dedicado. Foi instalado um agente em cada processador. Deste trabalho, pode-se concluir que o método que utiliza blocos compostos por um único indivíduo, denominado PSS (*pure self-scheduling*), apresentou comportamento mais estável em relação à distribuição mais equilibrada da carga, produzindo melhor desempenho nas otimizações que o utilizaram. A desvantagem deste método é que ele necessita de um número de comunicação entre mestre e agentes igual ao número de indivíduos, podendo haver situações em que isso é inadequado, o que ocorre quando o tempo de comunicação entre mestre e agente for relevante em relação ao tempo do processamento que o agente vai realizar.

	ESTÁTICO		**PSS**		**GSS**		**FSS**	
Nó	Nº aval	Tempo (s)	Nº aval	Tempo (s)	Nº aval	Tempo (s)	Nº aval	Tempo (s)
01	140	6339	< 74	3746	99	4330	85	3779
02	140	< 2450	> 211	3696	> 195	3419	> 207	3630
03	140	5714	88	3711	118	> 4428	95	3715
04	140	5972	86	3668	< 75	3782	84	3724
05	140	2656	200	< 3663	186	3412	197	< 3610
06	140	2690	193	3700	176	< 3380	188	3611
07	140	2680	192	3681	178	3420	189	3629
08	140	> 7645	76	> 3755	93	4166	< 75	> 3842
Tempo Total	2h7min27s		1h4min1s		1h23min12s		1h5min12s	

Tabela 4.1: Tempo dos processadores em cada método de escalonamento.

4.3 Tolerância a falhas

Uma característica que distingue sistemas distribuídos de sistemas centralizados é a noção de falha parcial. Uma falha parcial pode acontecer quando um componente de um sistema distribuído falha. Esta falha pode afetar o funcionamento correto de alguns componentes, mas ao mesmo tempo deixar outros inalterados. Já num sistema não distribuído, uma falha é geralmente total, uma vez que afeta todos os componentes e pode facilmente "derrubar" a aplicação [Tanembaum and van Steen (2002)]. Um sistema distribuído deve ser capaz de detectar automaticamente falhas parciais e recuperar-se sem comprometer o desempenho da aplicação, ou seja, o sistema deve tolerar falhas e continuar funcionando mesmo na

presença destas [Soh et al. (2006)]. Uma atenção a este tema no presente trabalho resultou em melhoria na confiabilidade do sistema.

No início da otimização, o mestre lê de um arquivo de entrada a localização de cada agente, armazenado-as numa lista. A cada geração todos os agentes da lista são testados para verificar se estão ativos. Para cada agente ativo, é criada no mestre uma *thread*, responsável pelo acesso aos blocos e pela chamada do método remoto no agente, onde os blocos de indivíduos serão avaliados. As *threads* terminam quando acessam a lista de **blocosNaoAvaliados** e verificam que ela está vazia ou em caso de falha do agente.

Quando um agente falha, duas situações podem ocorrer:

- o método remoto retorna erro;
- o método remoto não retorna erro.

No caso de retorno de erro, o bloco que não foi avaliado devido à falha é retirado da lista de **blocosEmAvaliação** e recolocado na lista de **blocosNaoAvaliados** para posteriormente ser avaliado por um dos agentes ativos e a *thread* responsável pelo agente que falhou termina. Quando o método remoto não retorna erro (isto acontece quando a máquina é desligada sem o processo de "shutdown"), a *thread* fica esperando o retorno do método remoto, que não ocorrerá, travando a execução do programa.

Para solucionar esta situação, uma vez que o RMI não oferece este tipo de recurso [Garbacki et al. (2005)], as *threads* são invocadas com um tempo máximo de execução (*time out*), que pode ser indicado pelo usuário no arquivo de entrada de dados ou pode ser calculado pelo programa através de uma estimativa. Após este tempo máximo, todas as *threads* serão terminadas e, se ainda houver blocos na lista **blocosEmAvaliação**, estes são transferidos para a lista de **blocosNaoAvaliados** e um novo processo de escalonamento recomeça para avaliação destes blocos. O cálculo do tempo máximo das *threads*, quando feito pelo programa, é baseado no tempo médio de cada simulação, no número de agentes disponíveis e na quantidade de simulações por geração. Devido à heterogeneidade

dos processadores e ao fato de que vários agentes podem falhar, esta estimativa é imprecisa e, para evitar o término prematuro de uma *thread* que esteja gerenciando um agente lento, acaba-se utilizando um tempo de espera que pode ser bem maior que o necessário.

4.4 Ambientes de execução do sistema

As aplicações científicas, em geral, demandam muita computação. O acesso à computação de alto desempenho é possível através da utilização de máquinas paralelas (supercomputadores) e de sistemas distribuídos. As máquinas paralelas, equipamentos de alto custo, são sistemas fortemente acoplados e, por isso, apropriados para a execução de aplicações que exigem muita comunicação entre as tarefas que serão executadas em paralelo. Até o final da década de 80, eram a única forma de acesso à computação de alto desempenho. Os anos 90 verificaram o surgimento de um novo ambiente de processamento: os sistemas distribuídos, que podem apresentar-se como:

- um conjunto de estações de trabalho conectadas por uma rede local;
- um *cluster* específico para computação científica;
- uma grade computacional.

Os sistemas distribuídos são considerados sistemas fracamente acoplados e apresentam-se como um excelente recurso para a resolução de problemas cujas tarefas necessitam pouca comunicação entre as máquinas, em particular aquelas que usam otimização por simulação, uma vez que as simulações, tarefas com alto custo computacional, são tarefas independentes e que podem ser realizadas concorrentemente [Yang et al. (2007)] e [Lim et al. (2007)].

Neste trabalho foram realizados experimentos em máquinas ligadas por rede local e em ambiente de grade computacional. Em ambas as situações, os agentes, diretórios e arquivos necessários para a execução de uma otimização são instalados

em disco local, em cada uma das máquinas participantes do experimento. Essa instalação, bem como a iniciação dos agentes remotos, é feita através de *shell scripts*, que utilizam o comando "ssh" para acessar máquinas remotas nos experimentos em rede local, e comandos do **Globus**, no caso dos experimentos realizados em ambiente de grade. Todas as máquinas usadas utilizam como sistema operacional alguma distribuição do Linux, entre elas: Suse, Ubuntu e Mandriva.

4.4.1 Ambiente local

A utilização das estações de trabalho ligadas via rede local foi motivada pela queda no custo dos PCs, pelo crescimento do poder computacional dessas máquinas e pelo desenvolvimento de redes locais de alta velocidade. Essas redes permitiram a interligação dos recursos, criando uma alternativa para obtenção de desempenho que, além de ter um custo menor, apresenta flexibilidade, permitindo crescimento incremental.

As máquinas utilizadas para os experimentos em ambiente de rede local estão localizadas em laboratórios e salas de uso público do Laboratório Nacional de Computação Científica (LNCC), tendo sido possível o acesso a cerca de 80 máquinas, num total de aproximadamente 200 núcleos. Assim foi possível fazer experimentos em larga escala, possibilitando uma melhor análise da escalabilidade da aplicação. A Tabela 4.2 mostra a configuração das máquinas utilizadas e na subseção referente a cada experimento será explicitado o subconjunto de máquinas utilizado.

Localização	**Processador**	**Clock**	**RAM**	**Nº de máquinas disponíveis**
Laboratório 4	Intel Core 2	1.86 GHz	3 GB	16
Laboratório 5	Intel Pentium 4	1.86 GHz	512 MB	13
Laboratório 6	Intel Pentium D	2.8 GHz	1 GB	17
Salas públicas	Intel Core 2 Quad	2.5 GHz	8 GB	6
Lab. Info. A	Dual Core AMD Opteron	2.6 GHz	64 GB	1
Lab. Info. B	Intel Xeon	2.93 GHz	132 GB	1
altix	Intel Itanium 2	1.8 GHz	64 GB	1
ComCiDis	heterogêneos	—	—	20

Tabela 4.2: Tabela de máquinas usadas para experimentos em rede local.

4.4.2 Grade computacional

Em meados dos anos 90, quando a Internet disponibilizava conexões estáveis de vários *megabits* por segundo, veio a proposta de utilizar estas linhas de comunicação de grande largura de banda para interconectar *clusters* e supercomputadores, além de outros recursos, localizados em diferentes instituições, criando uma maneira inovadora de compartilhar recursos computacionais [Kon and Goldman (2008)]. Esta nova infra-estrutura foi denominada Grade Computacional.

Segundo Viktor Berstis, computação em grade, de forma simplificada, é o próximo estágio *evolucionista* da computação distribuída. O objetivo de uma grade é: a partir de vários sistemas heterogêneos conectados e que compartilham recursos, criar a impressão de um único computador virtual, simples, porém com alta capacidade computacional [Berstis (2002)].

Uma grade computacional permite o compartilhamento de recursos computacionais, não apenas CPUs, distribuídos geograficamente e gerenciados por diferentes organizações. As grades são importantes não apenas por disponibilizarem alta capacidade de processamento, mas também por possibilitarem a utilização de um poder computacional que não poderia ser acessado localmente, ou, pelo menos, não de maneira econômica. Grades computacionais são uma tendência na área científica e têm assumido papel importante até mesmo na indústria.

A grade utilizada nos experimentos deste trabalho foi a grade VCG, uma grade oportunista, desenvolvida pelo LNCC com apoio da Rede Nacional de Ensino e Pesquisa (RNP) e em parceria com várias instituições, entre elas Universidade Federal Fluminense (UFF), Universidade Federal do Paraná (UFPR), Universidade Federal de São Carlos (UFSCAR) e Centro Brasileiro de Pesquisas Físicas (CBPF). Seu objetivo é oferecer ao sistema nacional de pesquisa um ambiente para execução distribuída de tarefas computacionais que aproveite recursos ociosos de máquinas ligadas à rede da RNP. Sua arquitetura é baseada no *middleware* **Globus Toolkit** (versões 2.4 e 4.0.5).

A grade VCG vem disponibilizando aproximadamente 30 máquinas locali-

zadas no LNCC (Petrópolis-RJ) e em outros domínios. As características das máquinas que compõem este ambiente estão listadas na Tabela 4.3.

Localização	**Processador**	**Clock**	**RAM**	**Nº de máquinas disponíveis**
LNCC	Intel Pentium 4	3.2 GHz	1.5 GB	7
LNCC	Quad-Core AMD Opteron	2.0 GHz	4 GB	2
LNCC	Intel Xeon	1.86 GHz	1 GB	7
LNCC	Intel Dual Core	1.8 GHz	1 GB	4
LNCC	Intel Pentium D	3.2 GHz	2 GB	2
UFSCAR	Intel Xeon	1.86 GHz	1 GB	1
CENAPADNE	Intel Xeon	1.86 GHz	1 GB	1
UFF	Intel Xeon	1.86 GHz	1 GB	1
CBPF	Intel Pentium 4	1.8 GHz	1 GB	2

Tabela 4.3: Tabela de máquinas usadas para experimentos em ambiente de grade.

No Apêndice A são apresentados mais conceitos sobre grades e alguns exemplos de grades no Brasil e no mundo.

4.5 Análise de desempenho experimental

Nesta seção são apresentados conjuntos de experimentos feitos com o objetivo de permitir uma análise de desempenho e de escalabilidade da aplicação, com máquinas ligadas em rede local e em ambiente de grade.

Um dos principais objetivos do desenvolvimento de aplicações paralelas é a redução do tempo total de execução da aplicação. Contudo, não se deve buscar simplesmente aumentar este ganho, mas também considerar fatores tais como a eficiência e a escalabilidade. Para a nossa análise, as seguinte métricas foram utilizadas:

Speedup: valor adimensional que mede o ganho obtido pela implementação paralela de um algoritmo, calculado pela fórmula:

$$S = \frac{T_1}{T_P}, \tag{4.2}$$

onde T_1 é o tempo de execução do sistema usando um processador e T_P, o tempo com P processadores.

Eficiência: valor da fração de tempo que os processadores estão ocupados, caracterizando sua utilização, calculada pela expressão:

$$E = \frac{S}{NP}, \tag{4.3}$$

onde S é o speedup e NP o número de processadores.

Um sistema paralelo ideal, utilizando NP processadores[1] , teria *speed-up* de NP e eficiência igual a 1. Na prática, este comportamento ideal não é alcançado porque na execução de um algoritmo paralelo, os processadores gastam tempo realizando tarefas periféricas ao processamento de interesse como, por exemplo: sincronização, comunicação, escalonamento das tarefas, entre outras. O gasto com esse tipo de tarefa é chamado de *overhead*. Nesta aplicação são possíveis causas de *overhead*:

- tarefas do AG;
- escalonamento:
  - particionamento do *job* em blocos de tarefas;
  - acesso sequencial à lista de blocos (seção crítica);
  - distribuição dos blocos de tarefas para os agentes;
  - gerenciamento das *threads*.
- comunicação entre mestre e agente;
- desbalanceamento de carga.

Dependendo dos valores do *speedup* e da eficiência, pode-se considerar uma aplicação escalável ou não. A escalabilidade é uma medida qualitativa que representa a capacidade de uma aplicação paralela resolver problemas de tamanhos variados sem perda de eficiência, usando uma quantidade de recursos fixos. De forma semelhante, é a capacidade de manter eficiência na resolução de problemas de tamanho fixo, com o uso de maior quantidade de recursos.

[1] iguais ao processador usado para cálculo do T_1

4.5.1 Medidas de tempo utilizadas

Para calcular as métricas e verificar a escabilidade, são utilizadas diferentes medidas de tempo que permitirão analisar o desempenho de diversas fases de execução do sistema. Entre elas as mais importantes são:

- Tempo de execução do simulador: tempo de CPU gasto com o simulador numérico. Este tempo vai variar em função das diferentes configurações e capacidades das máquinas utilizadas. Esta medida não considera o tempo gasto pelo processador atendendo a outros processos, de outras aplicações. Medido com a função **`cpu_time`** do Fortran 90.

- Tempo de serviço: é o tempo gasto pelo agente para a execução do método remoto, que inclui:

  (1) a atualização do arquivo de entrada do modelo (escrita no arquivo do local e vazão dos poços);

  (2) a execução do simulador numérico;

  (3) o cálculo da aptidão do indivíduo, que é baseada nos resultados produzidos pelo simulador.

  Esta medida é feita usando tempo real e é obtida na máquina do agente, através do método **`System.currentTimeMillis`** do Java, que utiliza o relógio do sistema.

- Tempo da geração: tempo total de cada geração. Considera o tempo das *threads*, do particionamento do *job* e o tempo do AG, e é obtido com o uso do método Java já citado.

- Tempo de comunicação: tempo gasto na chamada do método remoto, excluindo o tempo de serviço. Obtido com o mesmo método do tempo anterior.

- Tempo de execução do sistema de otimização - é o tempo de parede gasto na execução do sistema de otimização. Medido externamente, no mestre, usando a função **`time`** do sistema operacional.

4.5.2 Experimentos

Serão apresentados resultados de medidas de desempenho do sistema em funcionamento nas seguintes condições:

- grupo de máquinas homogêneo e heterogêneo;
- escalonamento estático e dinâmico;
- usando dezenas de núcleos;
- usando 200 núcleos;
- rede local e distribuída geograficamente (grade).

Para cada conjunto de experimentos são apresentadas duas tabelas: uma tabela principal e uma tabela complementar. A tabela principal mostra: o tempo total de execução do sistema, o tempo total de serviço dos agentes e as métricas *speedup* e eficiência. Na tabela complementar são apresentados: o tempo total de CPU das simulações, o número máximo e mínimo de simulações por agente, o tempo total de serviço máximo e mínimo por agente, o tempo de serviço mínimo médio e o tempo de serviço máximo médio, e a média dos tempos de serviço considerando-se todas as simulações feitas e chamada na tabela de tempo de serviço médio. Os dados complementares ajudam a entender alguns detalhes do funcionamento do sistema e a explicar como foram obtidos os valores da tabela principal.

Em alguns experimentos foi considerada a presença de processadores multi-core, sendo instalado mais de um agente em algumas máquinas. O reconhecimento de processadores multi-core não é feita automaticamente pela aplicação, mas através de um arquivo de entrada que contém a localização de cada agente. O número de agentes instalado em cada máquina limitou-se ao número de núcleos disponível no seu processador.

4.5.2.1 Ambiente homogêneo

Os experimentos apresentados nesta seção foram feitos com o objetivo de analisar a escalabilidade da aplicação com o aumento do número de processadores em ambiente homogêneo.

Para isto foram realizados quatro experimentos, que utilizaram 1, 10, 20 e 30 núcleos de máquinas com processador Intel Pentium D, 2.8 GHz e 1 GB de memória RAM. Essas máquinas estão conectadas por rede local de 100MB/s, através de um mesmo *switch*. Nos experimentos com 20 e 30 núcleos, foram instalados 2 agentes em algumas máquinas.

Cada um dos experimentos executou 930 simulações, divididas em 30 indivíduos da população inicial e mais 10 gerações de 90 indivíduos candidatos cada, conforme Tabela 4.4, num total de 11 momentos de sincronismo.

Dados da Otimização	
Nº de gerações	10
População inicial	30
Nº de candidatos por geração	90
Orçamento (nº de sim)	930

Tabela 4.4: Parâmetros para o algoritmo genético.

Inicialmente, a otimização foi feita utilizando um único núcleo e o tempo total da execução do sistema de otimização foi de $580min$. Este é o valor utilizado para T_1 na Fórmula 4.2, usada para a obtenção do *speedup*. A Tabela 4.5 apresenta dados para os experimentos com 10, 20 e 30 núcleos e observa-se a redução no tempo de execução do sistema com a utilização de mais núcleos. O *speedup* mostra o ganho com o aumento no número de núcleos utilizados e a eficiência, alta quando usados 10 e 30 núcleos, cai no caso onde são utilizados 20 núcleos, o que, a princípio, não era o esperado. Em aplicações paralelas é natural que haja redução na eficiência com o uso de um número maior de processadores, pois o gerenciamento destes aumenta o *overhead* mas, neste caso, o que mais influencia é o encaixe entre o tamanho do problema e o número de núcleos, que propicia ou não um bom balanço de carga.

Quantidade de núcleos usados	10	20	30
Tempo de execução do sist. de otim. (T_P)	59min	33min	20min
Σ do tempo total de serviço dos agentes	586min	586min	587min
Speed-up	9,83	17,57	29
Eficiência	0,98	0,87	0,97

Tabela 4.5: Medidas de desempenho para experimento de escalabilidade em ambiente homogêneo.

Observando os valores da Tabela 4.6 e analisando com detalhes o que ocorreu, pode-se verificar que com 10 núcleos, cada um deles realiza três tarefas na avaliação da população inicial e mais nove tarefas a cada geração, num total de 93 tarefas; com 30 núcleos, cada núcleo avalia um indivíduo na população inicial e mais três a cada geração, totalizando 31 tarefas. O tempo de serviço mínimo e máximo por agente é bem próximo: $3.516s$ e $3.520s$ para 10 núcleos, $1.171s$ e $1.178s$ para 30 núcleos. No experimento com 20 núcleos, cada núcleo processa por geração quatro indivíduos para a avaliação dos 80 primeiros indivíduos e, estando em ambiente homogêneo, todos gastam tempo semelhante. Para completar os 90 indivíduos, 10 ainda precisam ser avaliados e assim 10 núcleos avaliam mais um indivíduo, enquanto outros 10 ficam ociosos. Isto se repete a cada geração. A mesma divisão inexata ocorre na avaliação da população inicial, com 30 indivíduos. Na Tabela 4.6 verifica-se também que esse desbalanceamento fez com que um agente trabalhasse $1.924s$ (tempo máximo) enquanto outro, $1.631s$ (tempo mínimo).

Quantidade de núcleos usados	10	20	30
Tempo total de cpu das simulações	582min	582min	583min
Nº de simulações MAX por agente	93	51	31
Nº de simulações MIN por agente	93	43	31
Razão entre MAX e MIN	1	1,19	1
Tempo de serviço total MAX por agente (nº sim)	3520s (93)	1924s (51)	1178s (31)
Tempo de serviço total MIN por agente (nº sim)	3516s (93)	1631s (43)	1171s (31)
Razão entre MAX e MIN	1,001	1,17	1,005
Tempo de serviço médio MIN	37,81s	37,73s	37,76s
Tempo de serviço médio MAX	37,85s	37,95s	37,99s
Tempo de serviço médio	37,82s	37,84s	37,89s

Tabela 4.6: Medidas complementares de desempenho para experimento de escalabilidade em ambiente homogêneo.

Pode-se dizer que se fosse aumentado o número de indivíduos por etapa geracional, de modo que todos os processadores recebessem a mesma quantidade de avaliações, ou seja, mais 10 indivíduos por geração, seriam avaliados 1.040 indivíduos, ao invés de 930, e o custo computacional seria o mesmo, o que ocorre devido a diminuição do tempo ocioso ou, da mesma forma, o aumento da utilização dos núcleos.

Nestes experimentos o fator que mais contribuiu para o *overhead* e consequente redução na eficiência da aplicação foi o tempo gasto com o sincronismo, em particular, no experimento com 20 núcleos, devido à divisão inexata do número de avaliações pelo número de núcleos, que provocou, a cada geração, a ociosidade de 10 núcleos. Isto não está associado ao escalonamento usado, ou seja, qualquer escalonamento produziria resultado semelhante. Porém, em ambiente heterogêneo, esta escolha irá influenciar o desempenho, como será visto no próximo experimento.

4.5.2.2 Escalonamento estático × dinâmico

Nos experimentos desta seção são avaliados dois métodos de escalonamento diferentes, mencionados na Seção 4.2: o escalonamento estático e o escalonamento dinâmico puro (PSS), para resolver o problema de otimização com os mesmos parâmetros dos experimentos da seção anterior, conforme Tabela 4.4. A utilização destes dois métodos permite a comparação do desempenho com e sem carga balanceada.

Foram usados 10 núcleos com características diferentes, de máquinas distribuídas em laboratórios e salas de aula do LNCC, conforme listado na Tabela 4.7.

Neste experimento e nos das seções seguintes, também em ambiente heterogêneo, o cálculo do *speedup* e da eficiência é feito usando uma aproximação para T_1 que é calculado pela soma do tempo total de serviço dos agentes que, como visto, inclui: a execução da simulação numérica, a atualização do arquivo com os dados do modelo e cálculo da aptidão do indivíduo. Esta medida leva em consideração a

Localização	Processador	Clock	RAM
sala pública	Intel Core 2 Quad	2.5 GHz	8 GB
laboratório 4	Intel Core 2	1.86 GHz	3 GB
laboratório 5	Intel Pentium 4	1.86 GHz	512 MB
laboratório 6	Intel Pentium D	2.8 GHz	1 GB
lab info A	Dual Core AMD Opteron	2.6 GHz	64 GB
ComCiDis	Intel Pentium 4	3.2 GHz	1.5 GB
ComCiDis	Quad-Core AMD Opteron	2.0 GHz	4 GB
ComCiDis	Intel Xeon (8 cores)	1.86 GHz	1 GB
ComCiDis	Intel Dual Core	1.8 GHz	1 GB
ComCiDis	Intel Pentium D	3.2 GHz	2 GB

Tabela 4.7: Características de 10 máquinas diferentes.

heterogeneidade do ambiente e mede a capacidade de processamento das máquinas como um conjunto.

A Tabela 4.8 é usada para comparar o desempenho dos métodos de escalonamento usados. Com o mesmo número de núcleos (10), a utilização do método dinâmico garantiu um tempo 40% menor, executando o sistema em $56min$, enquanto foram necessários $94min$ utilizando o método estático. A diferença entre as medidas do tempo total de serviço dos agentes - $535min$ no dinâmico e $563min$ no estático - reflete a distribuição da carga: no dinâmico, os processadores mais rápidos recebem mais carga que os mais lentos, enquanto no estático a carga é distribuída igualmente. Entretanto, como este tempo não considera o tempo ocioso dos processadores, esta diferença não é tão grande quanto a diferença entre os tempos de execução do sistema.

Escalonamento	Estático	Dinâmico
Tempo de execução do sist. de otim. (T_P)	94min	56min
Σ do tempo total de serviço dos agentes (T_1)	563min	535min
Speedup	5,98	9,49
Eficiência	0,59	0,94

Tabela 4.8: Medidas de desempenho para experimento com 10 núcleos, usando escalonamento estático e dinâmico.

Na Tabela 4.9 observa-se que no escalonamento estático o número de tarefas distribuídas pelos agentes é igual, cada um processa 93 avaliações. Como os pro-

cessadores têm capacidades diferentes, a diferença entre o tempo de serviço mínimo dos agentes ($2.169,41s$) e o tempo máximo ($5.671,31s$) é grande. O escalonamento dinâmico atribui mais carga para os processadores mais rápidos, diminuindo a diferença entre os tempo de serviço (mínimo de $3.078,09s$ e máximo de $3.359,49s$) e obtendo maior eficiência.

Escalonamento	**Estático**	**Dinâmico**
Tempo total de cpu das simulações	545min	530min
Tempo de serviço total MAX por agente (nº sim)	5671,31s	3359,49s (103)
Tempo de serviço total MIN por agente (nº sim)	2169,41s	3077,79s (93)
Razão entre MAX e MIN	2,6	1,09
Nº de simulações MAX por agente (sala pública)	93	134
Nº de simulações MIN por agente (lab5)	93	52
Razão entre MAX e MIN	1	2,6
Tempo de serviço médio MIN	23,33s (93)	24,07s (134)
Tempo de serviço médio MAX	60,98s (93)	60,05s (52)
Tempo de serviço médio	36,15s	34,54s

Tabela 4.9: Medidas de desempenho complementares para experimento com 10 núcleos, usando escalonamento estático e dinâmico.

A taxa de utilização de alguns processadores, calculada de acordo com a Equação 4.1, está listada na Tabela 4.10. Com o escalonamento estático, o processador mais rápido teve utilização de 38% e o mais lento de 99,5%. Com o uso do escalonamento dinâmico, o mais rápido teve utilização de 95,34% e o mais lento de 92,29%.

	Estático	**Dinâmico**
Processador mais rápido	38,06%	95,34%
Processador mais lento	99,5%	92,29%
Processador com maior tempo de serviço	99,5%	99,28%
Processador com menor tempo de serviço	38,06%	90,95%

Tabela 4.10: Taxa de utilização dos processadores no experimento com escalonamento estático e dinâmico.

Para obtenção de alto desempenho num sistema heterogêneo, o balanceamento de carga é extremamente importante e diminui o tempo ocioso ou, da mesma

forma, aumenta o tempo de utilização. Nos próximos experimentos, o método de escalonamento puro (PSS) será utilizado.

4.5.2.3 Escalonamento dinâmico em ambiente heterogêneo

Nesta seção são apresentados experimentos realizados em ambiente heterogêneo, usando 10, 20 e 30 núcleos, e que permitem a análise da escalabilidade do sistema. Os parâmetros usados para o problema de otimização foram os mesmos da seção anterior e estão listados na Tabela 4.4.

Para estes experimentos foi formado um conjunto com 10 núcleos, todos diferentes entre si, um de cada uma das máquinas descritas na Tabela 4.7. Este conjunto foi utilizado no experimento com 10 núcleos. No experimento com 20 núcleos foram utilizados dois conjuntos idênticos ao usado no experimento com 10 núcleos e da mesma forma, no experimento com 30 núcleos foram utilizados três conjuntos idênticos ao primeiro. No caso de usar mais de um núcleo na mesma máquina, mais de um agente foi inicializado, permitindo que os núcleos fossem utilizados concorrentemente. A escolha deste ambiente controlado foi feita para que a capacidade de processamento fosse proporcional nos três casos, mesmo em ambiente heterogêneo.

Os resultados listados na Tabela 4.11 mostram que há ganho com o uso de mais processadores, mas apresentam redução na eficiência. Neste sistema considera-se o tempo ocioso dos processadores causado pelo desbalanceamento de carga como o fator que mais influencia no aumento do *overhead* e na consequente perda de eficiência. Este desbalanceamento é minimizado pelo uso de método de escalonamento dinâmico, como mostrado no experimento anterior, mas por haver vários momentos de sincronismo (neste experimento 11), este ainda é fator determinante.

Analisando os dados da Tabela 4.12, pode-se observar que no experimento com 10 núcleos, foi atribuída uma carga de 134 avaliações para o processador mais rápido, mas no experimento com 20 núcleos, apesar da capacidade computacional

Quantidade de núcleos usados	10	20	30
Tempo de execução do sist. de otim. (T_P)	56min	31min	21min
Σ do tempo total de serviço dos agentes (T_1)	535min	533min	536min
Speedup	9,49	16,89	25,01
Eficiência	0,95	0,84	0,83

Tabela 4.11: Medidas de desempenho para experimento de escalabilidade em ambiente heterogêneo.

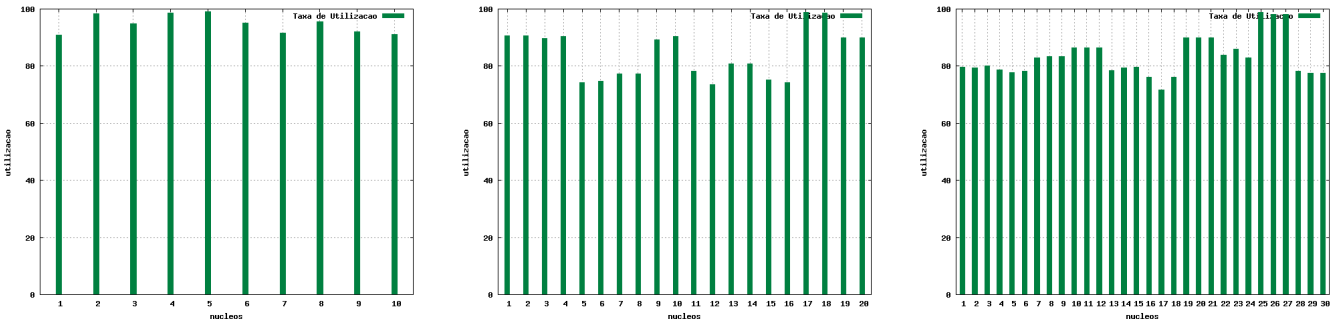
ter sido aumentada proporcionalmente, o sistema não atribuiu a metade desta carga para este processador, mas uma carga menor (62). O processador mais lento no experimento com 10 núcleos faz 52 avaliações, e no experimento com 20 núcleos processa mais do que a metade desta carga fazendo 31 avaliações. Comparação semelhante pode ser feita entre os experimentos de 10 e 30 núcleos.

Quantidade de núcleos usados	10	20	30
Tempo total de cpu das simulações	530min	528min	531min
Tempo de serv. total MAX por agente (nº sim)	3359,49s (103)	1875,63s (31)	1272,43s (21)
Tempo de serv. total MIN por agente (nº sim)	3077,79s (93)	1398,39s (62)	923,61s (41)
Razão entre MAX e MIN	1,09	1,34	1,37
Nº de sim. MAX por agente (sala pública)	134	62	41
Nº de sim. MIN por agente (lab5)	52	31	21
Razão entre MAX e MIN	2,6	2	1,95
Tempo de serviço médio MIN	24,07s (134)	22,55s (62)	22,53s (41)
Tempo de serviço médio MAX	60,05s (52)	60,50s (31)	60,59s (21)
Tempo de serviço médio	34,54s	34,41s	34,58s

Tabela 4.12: Medidas de desempenho complementares para experimento de escalabilidade em ambiente heterogêneo.

As figuras 4.2(a), 4.2(b) e 4.2(c) mostram a taxa de utilização dos processadores nos experimentos com 10, 20 e 30 núcleos respectivamente. Verifica-se utilização alta dos núcleos no primeiro e ocorrência de ociosidade nos outros dois. Destaca-se também, no experimento com 20 núcleos, onde foram usados dois núcleos de cada tipo, a utilização semelhante de cada par e, no experimento com 30 núcleos, o mesmo comportamento para cada grupo três núcleos.

A Tabela 4.13 mostra os valores da utilização dos processadores mais rápidos e mais lentos de cada experimento e pode-se observar que nos casos com 20 e 30 núcleos, a utilização dos mais rápidos cai.



(a) 10 núcleos (b) 20 núcleos (c) 30 núcleos

Figura 4.2: Taxa de utilização, ambiente heterogêneo em rede local.

	10 núcleos	**20 núcleos**	**30 núcleos**
Processador mais rápido	95,34%	73,79%	71,82%
Processador mais lento	92,29%	98,97%	98,94%
Processador com mais tempo de serviço	99,28%	98,97%	98,94%
Processador com menos tempo de serviço	90,95%	73,79%	71,82%

Tabela 4.13: Taxa de utilização dos processadores em experimento heterogêneo com 10, 20 e 30 núcleos.

Para confirmar a hipótese de que mesmo com o uso de escalonamento dinâmico, o tempo ocioso provocado pela necessidade de espera no final de cada geração é o fator determinante na redução da eficiência do sistema, foi realizado um experimento, com 30 núcleos, mantendo o número de avaliações dos experimentos anteriores (930), porém realizando-as como população inicial, ou seja, todas numa mesma iteração, produzindo então um único momento de sincronismo, ao invés dos 11 dos casos anteriores. A eficiência de 98% foi obtida neste caso, conforme Tabela 4.14, demonstrando que o sistema distribuído não introduz custos adicionais excessivos e a aplicação, sendo naturalmente paralelizável, é altamente escalável.

Quantidade de núcleos usados	**30**
Tempo de execução do sist. de otim. (T_P)	17min
Σ do tempo total de serviço dos agentes (T_1)	510min
Speedup	29,44
Eficiência	0,98

Tabela 4.14: Medidas de desempenho para experimento com apenas uma iteração.

Mais uma vez verifica-se que o balanceamento de carga é extremamente importante para a obtenção de desempenho num sistema paralelo e, particularmente neste sistema, confirma-se que é o fator que mais aumenta o *overhead* e a perda de eficiência. Uma alternativa para melhorar o desempenho é a implementação de um balanceamento inteligente que use informação dos tempos de avaliações anteriores e escolha se é ou não conveniente atribuir uma nova avaliação aos processadores mais lentos quando faltarem poucos indivíduos a serem avaliados. Esta implementação não está no escopo deste trabalho, mas é mencionada como trabalho futuro.

4.5.2.4 Experimento em grande escala

Nesta seção são apresentados os resultados de um experimento onde foram utilizados 200 núcleos de cerca de 75 máquinas, a maioria com processadores multicore, ligadas em rede local e distribuídas em diversos laboratórios do LNCC, conforme listados na Tabela 4.2. A realização desse experimento foi considerada como um desafio devido ao número de núcleos utilizados e também porque as máquinas estão em ambientes diferentes e sob diferentes administrações.

Foram feitas um total de 40.500 simulações distribuídas em 20 gerações de 2.000 indivíduos e mais 500 da população inicial, conforme listado na Tabela 4.15.

Dados da Otimização	
Nº de gerações	20
População inicial	500
Nº de candidatos por geração	2000
Orçamento (nº de sim)	40500
Nº de núcleos	200

Tabela 4.15: Parâmetros usados na otimização com 200 núcleos.

A Tabela 4.16 mostra valores de tempo de execução e de serviço de agentes, além do *speedup* e eficiência aproximados. O tempo total desta otimização foi de 2h24m e, se fosse considerada a mesma otimização usando um único núcleo, o tempo de execução seria de:

- 11 dias, com o núcleo mais rápido;
- 17 dias, com um núcleo que tivesse capacidade igual à média aritmética dos 200 núcleos usados;
- 41 dias, com o núcleo mais lento.

Quantidade de núcleos usados	**200**
Tempo de execução do sist. de otimização (T_P)	2h24min (144min)
Σ do tempo total de serviço dos agentes (T_1)	424h35min (25475min)
Speedup	176,91
Eficiência	0,884
Tempo de serviço total MAX por agente (nº sim)	8195,01s (141)
Tempo de serviço total MIN por agente (nº sim)	7276,66s (182)

Tabela 4.16: Resultados dos experimentos com 200 núcleos, 40.500 simulações.

Para entender melhor a eficiência obtida nesta otimização, pode-se fazer uma análise do pior caso, que ocorreria se a cada geração todos os agentes ficassem esperando o agente mais lento terminar de avaliar o último indivíduo da lista de soluções candidatas. O agente mais lento levou um tempo médio de $87,5s$ para fazer cada avaliação e se a espera ocorresse nas 21 gerações, o tempo total de espera seria de $1.837,5s$. Assim, o tempo mínimo de serviço de um agente, neste experimento, seria de $6.802,5s$, resultado da subtração do tempo máximo de espera $1.837,5s$ do tempo de parede da aplicação $8.640s$, gerando uma taxa de utilização mínima de 78,73%, que pode ser considerada uma aproximação para o limite inferior da eficiência.

O resultado de 88,4% obtido está dentro do esperado e, se fosse encontrada uma eficiência menor que 78,73%, esta seria devido a *overhead* causado por outro motivo, não pelo desbalanceamento de carga. Os resultados obtidos são satisfatórios e nos encorajam a realizar modelagens mais complexas e que demandem a utilização de grande quantidade de núcleos.

4.5.2.5 Ambiente de grade - máquinas locais

Nesta seção foram feitos experimentos com 10, 20 e 30 núcleos de processadores de máquinas da grade VCG, localizadas no LNCC, cujas características encontram-se na Tabela 4.3. O aumento no número de núcleos usados neste conjunto de experimentos não foi feito de forma controlada como na Seção 4.5.2.3. Estes experimentos objetivam, além de analisar o desempenho e a escalabilidade da aplicação em ambiente de grade, perceber as diferenças entre este ambiente e o de rede local, na forma de utilização da aplicação.

Na grade não há acesso direto às máquinas, isto é, não se faz *login*, e os arquivos do sistema são transferidos e acessados através de comandos do **Globus**: **`globus-job-run`** e **`globus-url-copy`**. Os arquivos necessários são compactados e transferidos para as máquinas, juntamente com um *script* de instalação.

O Programa 1 copia e instala os arquivos do sistema (mestre, agente e simulador numérico), além de iniciar os agentes. Os arquivos com os parâmetros da otimização, lista de máquinas e com os dados do modelo numérico são transferidos e instalados pelo Programa 2. O mestre é iniciado pelo Programa 3 e após sua execução os arquivos com os resultados obtidos são transferidos para a máquina local. Uma vez iniciados agentes e mestre, a comunicação entre eles é feita via RMI, da mesma forma que ocorre no ambiente de rede local.

```
1: #instalar sistema e iniciar agentes

2: dirOrigem="/homeGlobus/pcosta/bin"
3: dirDestino="/tmp/pcosta/bin"
4: maquina="grade02@lncc.br"
5: numPorta="4000"

6: globus-url-copy file:${dirOrigem}/sistema.tar.bz2   gsiftp://${maquina}/${dirDestino}/sistema.tar.bz2
7: globus-url-copy file:${dirOrigem}/instalarSist.sh   gsiftp://${maquina}/${dirDestino}/instalarSist.sh
8: globus-job-run  ${maquina}                          ${dirDestino}/instalarSist.sh
9: globus-job-run  ${maquina}                          ${dirDestino}/iniciarAgente.sh ${numPorta}
```

Programa 1: *Shell script* de instalação do sistema via **Globus**.

```
1: #instalar otimização e modelo numérico

2: dirOrigem="/homeGlobus/pcosta/otimizacao/expA"
3: dirDestino="/tmp/pcosta/expA"
4: maquina="grade02@lncc.br"

5: globus-url-copy  file:${dirOrigem}/otim.tar.bz2       gsiftp://${maquina}/${dirDestino}/otim.tar.bz2
6: globus-url-copy  file:${dirOrigem}/instalarOtim.sh    gsiftp://${maquina}/${dirDestino}/instalarOtim.sh
7: globus-job-run   ${maquina}                           ${dirDestino}/instalarOtim.sh
```

Programa 2: *Shell script* de instalação de uma otimização via **Globus**.

```
1: #executar mestre e copiar resultados

2: dirMestre="/tmp/pcosta/bin"
3: dirOtimizacao="/tmp/pcosta/expA"
4: dirLocal="/homeGlobus/pcosta/otimizacao/expA"
5: maquina="grade02@lncc.br"

6: globus-job-run   ${maquina}    ${dirMestre}/iniciarMestre.sh ${dirOtimizacao}
7: globus-url-copy  gsiftp://${maquina}/${dirOtimizacao}/mestre.out        file:${dirLocal}/mestre.out
8: globus-url-copy  gsiftp://${maquina}/${dirOtimizacao}/estatisticas.txt  file:${dirLocal}/estatisticas.txt
```

Programa 3: *Shell script* de execução do mestre via **Globus**.

A Tabela 4.17 apresenta os resultados dos experimentos feitos com os parâmetros de otimização listados na Tabela 4.4. Verifica-se, analisando os tempos de execução, que houve ganho com o aumento do número de núcleos. Porém, destaca-se a redução acentuada na eficiência da aplicação com o uso de uma quantidade maior de núcleos.

Quantidade de núcleos usados	**10**	**20**	**30**
Tempo de execução do sist. de otim. (T_P)	57min	32min	24min
Σ do tempo total de serviço dos agentes (T_1)	548min	563min	549min
Speedup	9,544	17,45	22,79
Eficiência	0,95	0,87	0,76

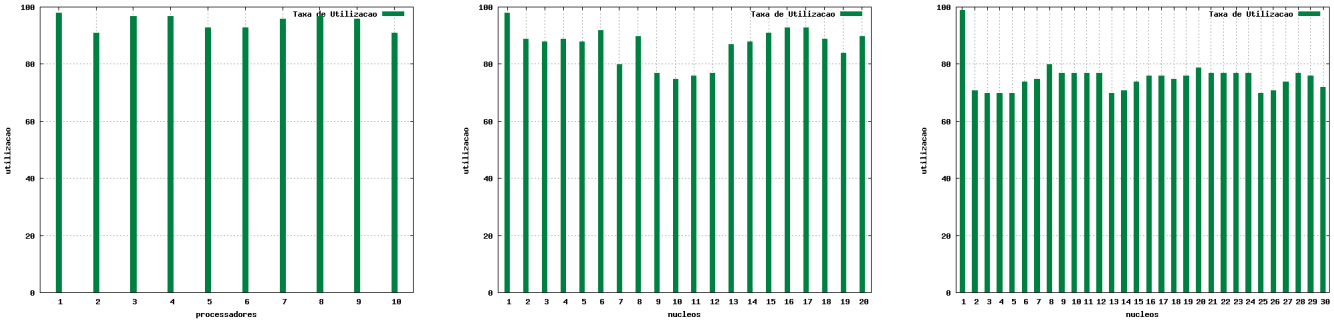
Tabela 4.17: Medidas de desempenho para experimento de escalabilidade em grade usando apenas máquinas internas.

Os dados da Tabela 4.18 mostram o funcionamento do escalonamento para obtenção do balanceamento e ajudam no entendimento desta redução. No experimento com 10 núcleos, os tempos totais de serviço dos agentes são próximos e, para conseguir isto, o número de avaliações feitas pelo agente mais rápido foi 40% maior que a quantidade do mais lento. Devido à particularidade dos parâmetros de otimização e às características das máquinas, o balanceamento no experimento com 30 núcleos foi deficiente. Apesar de haver máquinas com capacidades diferentes, o agente mais lento fez o mesmo número de avaliações que o mais rápido - 31

avaliações - prejudicando o desempenho. A cada geração todos os agentes fizeram três avaliações e o mais lento, por terminar sempre por último, fez os demais ficarem ociosos, o que pode ser verificado na Figura 4.3(c) que apresenta as utilizações dos núcleos. O mais lento tem uma utilização alta enquanto as demais ficam com valores mais baixos. Este problema é menor quando usa-se 10 e 20 núcleos, o que pode ser observado nas Figuras 4.3(a) e 4.3(b).

Quantidade de núcleos usados	**10**	**20**	**30**
Tempo total de cpu das simulações	9h4min	9h18min	9h1min
Tempo de serv. total MAX por agente (nº sim)	3397,96s (102)	1913,23s (41)	1446,89s (31)
Tempo de serv. total MIN por agente (nº sim)	3142,69s (93)	1467,21s (41)	1022,65s (31)
Razão entre MAX e MIN	1,08	1,30	1,41
Nº de simulações MAX por agente	102	52	31
Nº de simulações MIN por agente	73	31	31
Razão entre MAX e MIN	1,4	1,68	1
Tempo de serviço médio MIN	33,05s (102)	32,66s (52)	32,99s (31)
Tempo de serviço médio MAX	46,55s (73)	55,68s (31)	46,67s (31)
Tempo de serviço médio	35,35s	36,31s	35,45s

Tabela 4.18: Medidas de desempenho complementares para experimento de escalabilidade em grade usando apenas máquinas internas.



(a) 10 núcleos (b) 20 núcleos (c) 30 núcleos

Figura 4.3: Taxa de utilização, ambiente de grade.

Os resultados obtidos não sofrem influência do ambiente de grade, mas como no experimento da Seção 4.5.2.3, a redução na eficiência é resultado do tempo de espera no fim de cada geração, que tende a ser maior com o uso de mais processadores.

4.5.2.6 Ambiente de grade com utilização de máquinas externas

Neste experimento foram utilizados núcleos nas máquinas da grade VCG, descritas na Tabela 4.3, sendo sete agentes instalados em quatro máquinas externas. Os resultados apresentados para este experimento têm como objetivo analisar o comportamento do sistema em situações onde a influência de maior custo de comunicação pode trazer degradação no desempenho.

A Tabela 4.19 mostra o tempo da execução do sistema e o ganho obtido com o paralelismo e a Figura 4.4 mostra as taxas de utilização dos núcleos utilizados.

Quantidade de núcleos usados	**37**
Tempo de execução do sist. de otim. (T_P)	19min
Σ do tempo total de serviço dos agentes (T_1)	549min
Speedup	29,06
Eficiência	0,79%

Tabela 4.19: Medidas de desempenho para experimento de escalabilidade em ambiente de grade (incluindo máquinas externas).

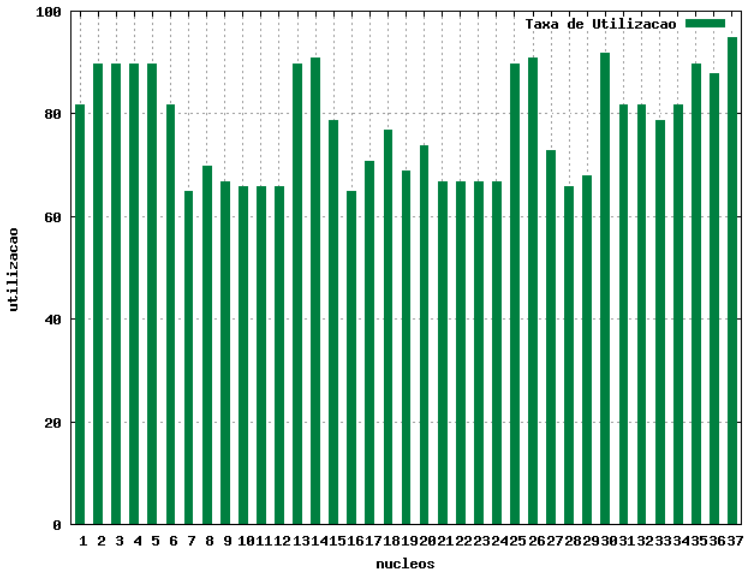


Figura 4.4: Taxa de utilização dos processadores para o experimento em ambiente de grade (incluindo máquinas externas).

Nos experimentos anteriores não houve influência do tempo gasto com a comunicação entre o mestre e os agentes. Neste ambiente, que usa máquinas externas,

este fator pode ser relevante e justificaria a utilização de um outro método de escalonamento que distribuísse blocos de tarefas maiores, diminuindo o tempo gasto com a comunicação como, por exemplo o FSS (*factorial self-scheduling*), analisado em [Costa et al. (2008)]. Porém, mesmo aqui o tempo de comunicação não foi relevante, conforme mostram os tempos da Tabela 4.20.

Máquina	**Tempo médio de comunicação**	**Tempo médio de computação**
LNCC (mais rápido)	0,0025s	36,32s
LNCC (mais lento)	0,0091s	35,53s
UFF	0,0090s	34,38s
CBPF	0,0155s	54,40s
UFSCAR	0,0301s	34,72s
UFC	0,0428s	34,68s

Tabela 4.20: Tempos médios de comunicação e computação em ambiente de grade com máquinas externas.

Capítulo 5

Conclusões e Perspectivas

Nesta dissertação foi utilizada uma metodologia multidisciplinar para otimização por simulação numérica que utiliza:

- modelos numéricos de aquíferos;
- algoritmos genéticos para otimização;
- sistemas distribuídos para computação paralela das simulações numéricas.

Essa abordagem é utilizada na busca automática de solução para um problema de grande relevância ambiental: o de contaminação de águas subterrâneas. A estratégia de remediação estudada foi a extração do contaminante através do bombeamento feito por poços de extração.

No Capítulo 2 foi apresentado o sistema de equações diferenciais usado para a modelagem do escoamento miscível, onde os poços são introduzidos como fontes nodais. Este modelo matemático é resolvido pelo método de elementos finitos. Utilizou-se um modelo de aquífero hipotético, bi-dimensional que, embora simples, combinado com o procedimento de otimização foi ferramenta útil para validação da metodologia apresentada.

No Capítulo 3 foram apresentados os conceitos fundamentais da ferramenta de otimização utilizada, o algoritmo genético, bem como o modelo geracional implementado, que por ser naturalmente paralelo é apropriado para a realização concorrente da avaliação das soluções candidatas, facilitando a implementação do sistema

distribuído. Neste capítulo foram realizados experimentos com o objetivo de validar a implementação feita e as escolhas do método de otimização. Para cada experimento foram testados diversos conjuntos de parâmetros do AG, numa etapa de calibração. O sistema paralelo, com uso de cerca de 80 máquinas, foi utilizado em todas as otimizações feitas, inclusive na calibração dos parâmetros, demonstrando ser ferramenta extremamente útil e fundamental para a redução do tempo de resposta.

O sistema distribuído que implementa os passos do AG e gerencia a avaliação paralela das simulações numéricas foi introduzido no Capítulo 4. Foram analisados os principais aspectos considerados na implementação, entre eles: a organização em módulos do sistema, o uso do modelo mestre/escravo com comunicação entre as máquinas feita via RMI, o balanceamento de carga e a tolerância a falhas. Os experimentos deste capítulo foram realizados com o objetivo de permitir uma análise de desempenho e escalabilidade do sistema distribuído implementado e foram feitos em rede interna, em ambiente homogêneo e heterogêneo, em pequena e larga escala e em ambiente de grade computacional.

Baseando-se na metodologia exposta e nos resultados dos experimentos apresentados nos Capítulos 3 e 4, pode-se enunciar as seguintes conclusões:

- O uso de computação distribuída é possível e tem importância fundamental no tempo de resposta da otimização, possibilitando que sejam realizados experimentos cuja execução serial não seria factível, devido ao longo tempo de computação necessária;

- Para a obtenção de alto desempenho e de eficiência do sistema, a distribuição equilibrada da carga é fator essencial. Esta foi obtida através do uso de métodos de escalonamento de tarefas, que reduzem o tempo de ociosidade dos processadores utilizados. Aos processadores com melhor desempenho ou com maior disponibilidade foram atribuídos maior número de simulações. A atribuição é realizada em tempo de execução, dinamicamente;

- A implementação de uma técnica de tolerância a falhas, mesmo simples, foi indispensável para a confiabilidade do sistema de simulações distribuídas;
- O uso de grades computacionais, incluindo máquinas geograficamente distribuídas, e pertencentes a diferentes domínios, amplia a possibilidade de exploração de recursos computacionais, sendo de grande importância para a computação científica.

As perspectivas para a continuação deste trabalho incluem vários aspectos. Primeiro, na área dos modelos numéricos seria interessante fazer um estudo mais aprofundado do modelo utilizado e das dificuldades numéricas encontradas, além de introduzir regiões de heterogeneidade, tornando o problema de otimização menos trivial. A implementação de paralelismo no AGUASUBMEF também pode ser explorada, principalmente em modelos tri-dimensionais. As unidades devem ser melhor avaliadas e comparadas com valores mais próximos da realidade, o que não invalida a confiança na metodologia desenvolvida.

Na área de otimização, desenvolvimentos futuros incluem o aperfeiçoamento da função objetivo, considerando o tempo de descontaminação, o custo operacional dos poços e o uso de valores mais realísticos para o custo e a penalidade, o desenvolvimento de novos operadores genéticos e o uso de modelos adaptativos para a probabilidade de aplicação destes.

No tópico da computação distribuída, um próximo passo na tentativa de gerar melhores resultados seria o uso de métodos de escalonamento que considerem a informação sobre o desempenho dos agentes, principalmente na distribuição das últimas tarefas de cada geração, momento onde ocorre o desbalanceamento de carga. Ainda com relação aos métodos de escalonamento, um aspecto importante que deve ser ressaltado é a importância do uso de métodos hierárquicos (descentralizados), essenciais para a obtenção de escalabilidade quando o número de núcleos a serem gerenciados cresce. Outra melhoria seria a implementação de agentes que atendam à solicitação de mais de um mestre e que sejam capazes de identificar processadores com mais de um núcleo.

Referências Bibliográficas

Abdanur, A. (2005). Remediação de solo e água subterrânea contaminados por hidrocarbonetos de petróleo: Estudo de caso na Refinaria de Duque de Caxias (RJ). Master's thesis, Universidade Federal do Paraná.

Ahlfeld, D. P., Mulvey, J. M., Pinder, G. F., and Wood, E. F. (1988). Contaminated groundwater remediation design using simulation, optimization, and sensitivity theory 1. Model development. **Water Resources Research**, 24:431–441.

Ahlfeld, D. P. and Sawyer, C. S. (1990). Well location in capture zone design using simulation optimization techniques. **Groundwater**, 28:507–512.

Alba, E. and Tomassini, M. (2002). Parallelism and evolutionary algorithms. **IEEE Transactions on Evolutionary Computation**, 6:443–462.

Andrade, N., Costa, L., Germoglio, G., and Cirne, W. (2005). Peer-to-peer grid computing with the OurGrid community. In **23rd Brazilian Symposium on Computer Networks (SBRC 2005) - 4th Special Tools Session**.

Araújo, E., Cirne, W., Wagner, G., Oliveira, N., Souza, E. P., and ao, C. O. G. (2005). The SegHidro experience: Using the grid to empower a hydro-meteorological scientific network. In **Proceedings of 1st IEEE Conference on e-Science and Grid Computing**.

Baker, J. E. (1985). Adaptative selection methods for genetic algorithms. In **Proceedings of the 1st International Conference on Genetic Algorithms**, páginas 101–111. Lawrence Erlbaum Associates, Inc.

Bear, J. (1979). **Hydraulics of Groundwater**. Dover Publications.

Bear, J., Beljin, M., and Ross, R. (1992). Fundamentals of ground-water modeling. **EPA - Ground Water Issue**.

Bear, J. and Verruijt, A. (1987). **Modeling Groundwater Flow and Pollution**. D. Reidel Publishing Company.

Berman, F. and Wolski, R. (1997). The AppLes Project: A status report. In **Proceedings of the 8th. NEC Research Symposium, Berlim, Germany**.

Bernholdt, D., Bharathi, S., Brown, D., Chanchio, K., Chen, M., Chervenak, A., Cinquini, L., Drach, B., Foster, I., Fox, P., Garcia, J., Kesselman, C., Markel, R., Middleton, D., Nefedova, V., Pouchard, L., Shoshani, A., Sim, A., Strand, G., and Williams, D. (2005). The earth system grid: Supporting the next generation of climate modeling research. In **Proceedings of the IEEE**.

Berstis, V. (2002). Fundamentals of grid computing. Technical report, IBM.

Buyya, R., Abramson, D., Giddy, J., and Stockinger, H. (2002). Economic models for resource management and scheduling in grid computing. **Journal of Concurrency and Computation: Practice and Experience**, 14:1507–1542.

Cai, X., McKinney, D. C., and Lasdon, L. S. (2001). Solving nonlinear water management models using a combined genetic algorithm and linear programming approach. **Advances in Water Resources**, 24:667–676.

Camponogara, E. (2006). Métodos de otimização: Teoria e prática. UFSC, Florianópolis.

Cedeno, W. and Vemuri, V. R. (1996). Genetic algorithms in aquifer management. **Journal of Network and Computer Applications**, 19:171–187.

CETESB, C. d. T. d. S. A. (2004). Relatório de qualidade de águas subterrâneas. Technical report, CETESB. Disponível em www.cetesb.sp.gov.br, último acesso em 20 de maio de 2009.

Cheng, K., Yang, C., Lai, C., and Chang, S. (2004). A parallel loop self-scheduling on grid computing environments. **Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks**.

Chronopoulos, A., S.P., and Yu, N. (2002). Scalable loop self-scheduling schemes for heterogeneous clusters. **Proceedings of the IEEE International Conference on Cluster Computing**.

Clement, T., Sun, Y., Hooker, B., and Peterson, J. (1998). Modeling multispecies reactive transport in groundwater. **Groundwater Monitoring & Remediation**, 18:79–92.

Colorni, A., Dorigo, M., Maffioli, F., Maniezzo, V., Righini, G., and Trubian, M. (1996). Heuristics from nature for hard combinatorial optimization problems. **International Transactions in Operational Research**, 3:1–21.

Costa, P. A. P., Lima, F. J., Garcia, E. L. M., Barbosa, H. J., and Schulze, B. R. (2008). Métodos de escalonamento de tarefas para otimização por simulação em grade computacional. In **Anais do VI Workshop de Computação em Grid e Aplicações**, páginas 61–72.

Council, N. R. (1994). **Realizing the Information Future: the Internet and Beyond**. National Academic Press.

Cunha, M. C. (2002). Groundwater cleanup: The optimization perspective (a literature review). **Eng. Opt**, 34:389–702.

Dong, F. and Akl, S. G. (2006). Scheduling algorithms for grid computing: State of art and open problems. Technical report, School of Computing, Queen's University, Ontario.

Dougherty, D. E. and Marryott, R. A. (1991). Optimal groudwater management 1. Simulated annealing. **Water Resources Research**, 27:2493–2508.

Eiben, A. and Smith, J. (2003). **Introduction to Evolutionary Computing**. Springer-Verlag.

Falkenauer, E. (1999). **Evolutionary Algorithms**, Capítulo Applying Genetic Algorithms To Real-World Problems, páginas 65–89. Springer.

Finsterle, S. (2004). Demonstration of optimization techniques for groundwater plume remediation using iTOUGH2. Technical report, Earth Science Division, Lawrence Bekerley National Laboratory.

Foster, I. and Kesselman, C. (1999). **The Grid: blueprint for a new computing infrastructure**. Morgan Kaufmann Publishers Inc.

Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the grid: enabling scalable virtual organizations. **Intl J. Supercomputer Applications**.

Gao, Y., Rong, H., and Huang, J. Z. (2005). Adaptative grid job scheduling with genetic algorithms. **Future Generation Computer Systems**, 21:151–161.

Garbacki, P., Biskupski, B., and Bal, H. E. (2005). Transparent fault tolerance for grid applications. In **EGC**, páginas 671–680.

Garcia, E. (1997). **Formulações de Elementos Finitos Bi e Tridimensionais para Simulação de Reservatórios de Petróleo**. PhD thesis, COPPE/UFRJ.

Goldberg, D. E. (1989). **Genetic Algorithms in search, optimization and machine learning**. Addison Wesley Publishing Company.

Goldberg, D. E. and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In **Foundations of Genetic Algorithms**. Morgan Kaufmann Publishers.

Goldchleger, A. (2004). Integrade: Um sistema de middleware para computação oportunista. Master's thesis, IME/USP.

Goldchleger, A., Kon, F., Goldman, A., Finger, M., and Bezerra, G. C. (2000). Integrade: object-oriented grid middleware leveraging idle computing power of desktop machines. **Concurrency and Computation: Practice and Experience**, 1.

Gorelick, S., Freeze, R. A., Donohue, D., and Keely, J. (1993). **Groundwater Contamination, Optima Capture and Containment**. Lewis Publishers.

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

Huang, C. and Mayer, A. S. (1997). Pump-and-treat optimization using well locations and pumping rates as decision variables. **Water Resources Research**, 33:1001–1012.

Hummel, S., E.Schonberg, and Flynn, L. (1992). Factoring, a method for scheduling parallel loops. **Communications os the ACM**, 35(8).

J. Guan, M. A. (1999). Optimal remediation with well locations and pumping rates selected as continuous decision variables. **Journal of Hydrology**, 221:20–42.

Jain, R. (1991). **The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling**. Wiley and Sons.

Junior, J. P., de Camargo, R., Goldchleger, A., and Kon, F. (2005). Integrade: a tool for executing parallel applications on a grid for opportunistic computing. In **Proceedings of the 23rd Brazilian Symposium on Computer Networks**.

Kalwij, I. M. and Peralta, R. C. (2006). Simulation/optimization modeling for robust pumping strategy design. **Groundwater**, 44:574–582.

Kon, F. and Goldman, A. (2008). **Atualizações em Informática 2008**, Capítulo Grades Computacionais: Conceitos Fundamentais e Casos Concretos, páginas 55–104. Editora PUC-Rio e SBC.

Krauter, K., Buyya, R., and Maheswaran, M. (2002). A taxonomy and survey of grid resource management systems for distributed computing. **Software - Practice and Experience**, 32:135–164.

Lehr, J. H. and Nielsen, D. (1982). Aquifer restoration and ground-water rehabilitation. **Groundwater**, 6:650–656.

Li, M. and Baker, M. (2005). **The Grid Core Technologies**. John Wiley And Sons.

Lim, D., Ong, Y., Jin, Y., Sendhoff, B., and Lee, B. (2007). Efficient hierarchical parallel genetic algorithms using grid computing. **Future Generation Computer Systems**, 23:358–370.

Lima, F. J. (2008). Modelagem computacional para análise e otimização de processos de remediação de aquíferos contaminados. Master's thesis, LNCC.

Livny, M., Basney, J., Raman, R., and Tannenbaum, T. (1997). Mechanisms for high throughput computing. **Speedup Journal of Grid Computing**, 11(1).

Loula, A., Garcia, E., and Coutinho, A. (1999). Miscible displacement simulation by finite elements method in distributed memory machines. **Computer methods in applied mechanics and engeneering**, 174:339–354.

Loula, A. F. D. and Garcia, E. L. M. (1995). Simulação numérica de injeção de traçadores em reservatórios heterogêneos em 2D e 3D. In **I Workshop sobre Caracterização e Engenharia de Reservatórios**, volume UNICAMP-SP.

Lu, K., Subrata, R., and Zomaya, A. Y. (2007). On the performance-driven load distribuition for heterogeneous computational grids. **Journal of Computer and System Sciences**, 73:1191–1206.

Malta, S. M., Loula, A. A., and Garcia, E. L. (1995). A post-processing technique to approximate the velocity field in miscible displacements simulations. **Revista**

Matemática Contemporânea da Sociedade Brasileira de Matemática, 8:239–268.

Mattson, T. G., Sanders, B. A., and Massingill, B. L. (2004). **Patterns for Parallel Programming**. Addison Wesley.

Mayer, A. S., Kelley, C., and Miller, C. (2002). Optimal design for problems involving flow and transport phenomena in saturated subsurface systems. **Advances in Water Resources**, 25:1223–1256.

Mirghani, B., Tryby, M., Baessler, D., Karonis, N., Ranjthan, R., and Mahinthakumar, K. (2005). Development and performance analysis of a simulation-optimization framework on TeraGrid linux clusters. In **Proceedings of LCI International Conference on Linux Clusters: The HPC Revolution 2005**.

Nobre, M. M. (2005). Remediação de solo. **Revista Química e Derivados**. Disponível on-line, último acesso em 21 abril 2009.

Olafsson, S. and Kim, J. (2002). Simulation optimization. **Proceedings of the 2002 Winter Simulation Conference**.

Oliveira, A., Argolo, G., Martins, S., and Plastino, A. (2004). Avaliação de estratégia de balanceamento de carga do tipo mestre/escravo em um grid computacional. **Anais do II Workshop on Grid Computing Applications**.

Peaceman, D. W. (1977). **Fundamentals of Numerical Reservoir Simulation**. Elsevier Scientific Publishing Company.

Penmatsa, S., Chronopoulos, A., Karonis, N., and Toonen, B. (2007). Implementation of distributed loop scheduling schemes on the TeraGrid. **Parallel and Distributed Processing Symposium, IEEE**, páginas 1–8.

Ren, X. and Minsker, B. (2005). Which groundwater remediation objective is

better, a realistic one or a simple one? **Journal of water resources planning and management**, 131(5):351–361.

Sayeed, M., Mahinthakumar, K., and Karonis, N. (2007). Grid-enabled solution of groundwater inverse problems on the TeraGrid network. **Simulation**, 83(6):437–448.

Shreedhar Maskey, A. J. and Solomatine, D. P. (2002). Groundwater remediation strategies using global optimization algorithms. **Journal of Water Resources Planning and Management**, 128:431–440.

Silberschatz, A., Gakvin, P. B., and Gagne, G. (2004). **Operating System Concepts**. John Wiley and Sons, Inc.

Smarr, L. (2004). **The Grid 2: Blueprint for a new Computing Infrastructure**, Capítulo Grids in Context, páginas 1–12. Elsevier, Morgan Kaufmann.

Smarr, L. and Catlett, C. (1192). Metacomputing. **Comunication of the ACM**, 35:44–52.

Soh, H., Haque, S., Liao, W., and Buyya, R. (2006). **Grid Programming Models and Environments**, Capítulo 8, páginas 141–173. Nova Science Publishers, Inc.

Sotomayor, B. and Childers, L. (2006). **Globus Toolkit 4 Programming Java Services**. Morgan Kaufmann Publishers.

Tanembaum, A. and van Steen, M. (2002). **Distributed Systems**. Prentice Hall.

Voorsluys, W., Araujo, E., Cirne, W., ao, C. O. G., Souza, E. P., and Cavalcanti, E. P. (2007). Forecasting collaboration to better manage water resources. **Concurrency and Computation: Practice and Experience**.

Voss, C. (1984). Sutra: A finite-element simulation model for saturated-unsaturated, fluid-density-dependent ground-water flow with energy transport or chemically reactive single-species solute transport. Technical report, U.S. Geological Survey Water-Resources Investigation Report 84-4369, 409 p.

Wang, H. and Anderson, M. (1982). **Introduction to Groundwater Modeling**. Academic Press.

Yang, C., Cheng, K., and Shih, W. (2007). On development of an efficient parallel loop self-scheduling for grid computing environments. **Parallel Computing**, 33:467–487.

Zheng, C. and Wang, P. P. (1999). Mt3dms: A modular three-dimensional multispecies model for simulation of advection, dispersion and chemical reactions of contaminants in groundwater systems: Documentation and users guide. Technical report, U.S. Army Engineer Research and Development Center Contract Report SERDP-99-1.

Zhu, Y. (2003). A survey on grid scheduling systems. Technical report, Department of Computer Science, Hong Kong University of Science and Technology.

Apêndice A

Grade computacional

O termo grade foi formulado em meados da década de 90 [Foster et al. (2001)]. Antes disso, esforços para orquestrar recursos distribuídos geograficamente eram conhecidos como metacomputação [Smarr and Catlett (1192)]. A origem do termo grade (*grid*) vem de uma analogia à rede elétrica (*Power Grid*), na qual recursos computacionais poderiam ser consumidos como energia elétrica. Desta forma, quando fosse necessária alguma quantidade de poder computacional, poderia-se simplesmente conectar um computador pessoal a uma grade e esta ofereceria os recursos computacionais necessários. O acesso a computação "sob-demanda" já havia sido mencionado, por Leonard Kleinrock que, na instalação do primeiro nó da Arpanet (a Internet atual), em 1969, sugeriu [Kleinrock(1969)]:

> *"As of now, computer networks are in their infance. But as they grow up and become more sophisticated, we will probably see the spread of computer utilities, which, like present electric and power utilities, will service individual homes and offices accros the country."*

Larry Smarr analisa o aparecimento e a evolução de grades computacionais, comparando-as a outras infra-estruturas tais como eletricidade, telefonia, bancos, estradas de ferro, entre outras [Smarr (2004)].

Ian Foster e Carl Kesselman propuseram, em 1998, que uma grade computacional é uma infra-estrutura de hardware e software que oferece acesso confiável, consistente, altamente disponível e de baixo custo, a capacidades computacionais

[Foster and Kesselman (1999)]. Já em 2000, no artigo *The Anatomy of the Grid*, escrito em co-autoria com Steven Tuecke, os autores refinaram esta definição, enfatizando que computação em grade envolve o compartilhamento coordenado de recursos e a solução de problemas em organizações virtuais dinâmicas e multi-institucionais. Esse compartilhamento de recursos, que não se restringe a troca de arquivos, mas engloba acesso direto a computadores, softwares, dados, instrumentos científicos, dispositivos de visualização, entre outros, requer um controle sistemático, isto é, altamente controlado. Provedores e consumidores de recursos definem clara e cuidadosamente o que é compartilhado, quem tem permissão de compartilhar e quais são as codições de compartilhamento. Um grupo de indivídos e/ou instituções definidos por tais regras é uma organização virtual [Foster et al. (2001)].

A construção de uma grade computacional inclui: a integração de componentes individuais de hardware e software através de uma rede, a implementação de um *middleware* que ofereça transparência na utilização dos recursos disponíveis, o desenvolvimento de ferramentas que permitam o gerenciamento da infraestrutura e das aplicações de grade e o desenvolvimento de aplicações distribuídas que utilizem os recursos oferecidos.

A.1 Arquitetura de grade

Em 2004, foi proposta, por Foster e Kesselman, uma arquitetura que identifica os componentes fundamentais de uma grade, especifica o objetivo e a função desses componentes e indica como eles interagem entre si. Essa arquitetura organiza os componentes em camadas e segue os princípios do modelo ampulheta [Council (1994)]. A idéia deste modelo é afunilar toda comunicação através do uso de um pequeno conjunto de protocolos comuns. Assim, múltiplas aplicações (topo da ampulheta) podem ser escritas, utilizando como ferramenta de comunicação estes protocolos que, por sua vez, podem ser mapeados em diferentes tecnologias fundamentais - os recursos (base da ampulheta).

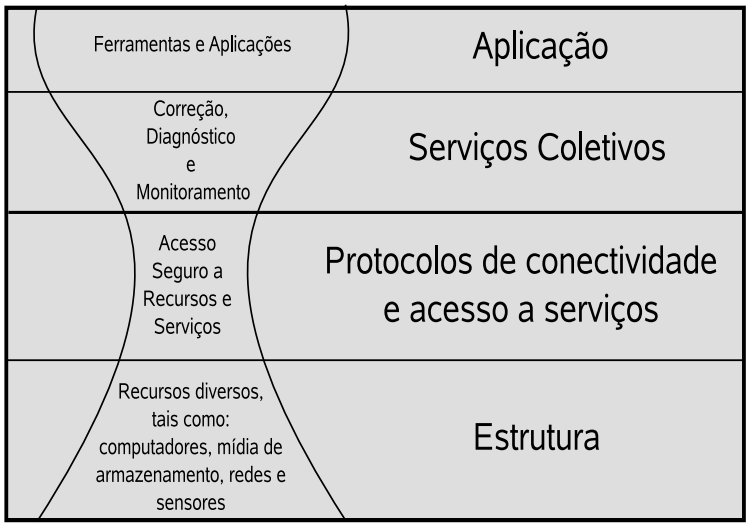


Figura A.1: Organização em camadas da arquitetura de grade.

A figura A.1 mostra as camadas da arquitetura de grade e seus componentes, descritos a seguir:

Camada estrutura

Esta camada encontra-se no nível mais baixo e é composta por todos os componentes de software e de hardware integrados por uma rede física como, por exemplo, PCs, clusters, unidades de disco, licenças de software, sistemas operacionais, sistemas de arquivos, intrumentos científicos e redes, entre outros.

Camada protocolos de conectividade e acesso a recursos

A camada protocolos de conectividade e acesso a recursos contém um conjunto básico de protocolos, necessários para efetuar operações entre os recursos da grade. Estes podem são subdivididos em:

- protocolos de comunicação: permitem a troca de dados entre os componentes da camada estrutura;
- protocolos de autenticação: fornecem mecanismos seguros para a verificação da identidade do usuário e dos recursos;

- protocolos de acesso aos recursos remotos: são definidos com base nos protocolos de comunicação e segurança (autenticação) e permitem a inicialização, o monitoramento e o controle de operações em recursos compartilhados.

Esses componentes são desenvolvidos pelos chamados *middleware* de Grade.

Camada serviços coletivos

Nesta camada encontram-se os protocolos, APIs e serviços que não estão associados a um recurso específico, mas que exploram a interação entre os grupos de recursos, combinando os componentes da camada abaixo. Exemplos são: serviços de diretórios, serviços de alocação e monitoramento e serviços de replicação de dados, entre outros.

Camada de aplicação

É a camada final da arquitetura. Aqui encontram-se as aplicações dos usuários que farão uso dos recursos disponíveis na camada estrutura, podendo para isto acessar os componentes das duas camadas abaixo.

A.2 *Middleware*

Um *middleware* é uma coleção de componentes de software que fornecem muitos dos "*building blocks*" para a criação de aplicações baseadas em Grade.

Entre os *middleware*s de Grade existentes, o Globus[1] , [Sotomayor and Childers (2006)] é, atualmente, o mais utilizado em todo o mundo, e vem tornado-se um padrão. O projeto é desenvolvido em conjunto por várias universidades e laboratórios de pesquisa, localizados em sua maioria nos Estados Unidos, dentre os quais o Argonne National Laboratory (ANL) na Universidade de Chicago, o National Center for Supercomputing Applications (NCSA) na Universidade de Illinois, e a Universidade do Sul da Califórnia.

[1] www.globus.org

A versão 1.0 foi lançada em 1998 e era utilizada pelos desenvolvedores do projeto. O Globus Toolkit 2 (GT2), lançado em 2002, trouxe uma série de novas funcionalidades e facilidades de instalação e uso, o que disseminou o uso do Globus por instituições de pesquisa por todo o mundo. Porém, os sistemas de Grade baseados no GT2 eram desenvolvidos com protocolos heterogêneos e isto tornava difícil a interoperabilidade entre eles.

Para resolver o problema da interoperabilidade um esforço grande vem sendo feito no estabelecimento de padrões. O Open Grid Forum (OGF) - que surgiu em 2006 através da união do Global Grid Forum (GGF) e da Enterprise Grid Alliance - é uma comunidade aberta dedicada à padronização de tecnologias relacionadas a grades computacionais. O GGF propôs, em 2002, a especificação de uma nova arquitetura denominada Open Grid Services Archictecture (OGSA), baseada em serviços Web que utilizam protocolos padrão, tais como XML e HTTP, para construir sistemas distribuídos orientados a serviços. O OGSA introduziu o conceito de serviços de Grade (*Grid services*), que são serviços Web com algumas extensões, incluídas para atender às necessidades da grade [Li and Baker (2005)].

O OGSA define quais são as interfaces necessárias, mas não especifica-as. Esta especificação é feita pelo Open Grid Services Infrastructure (OGSI), que descreve detalhadamente a arquitetura esboçada pela OGSA e define em detalhes como deve se comportar um serviço de Grade. Em 2003, baseada na especificação técnica do OGSI, foi lançada a versão 3.0 do Globus, um conjunto de ferramentas para a construção de sistemas de grade orientados a serviços, de acordo com a arquitetura OGSA. Com a evolução da tecnologia de Serviços Web, alguns aspectos da OGSI precisaram ser analisados novamente, dando origem ao Web Services Resource Framework (WSRF) [Sotomayor and Childers (2006)]. O Globus Toolkit encontra-se atualmente na versão 4.2.

Dentre outros *middlewares*, encontram-se o Condor[2] , o Legion[3] e no Brasil o

[2] www.cs.wisc.edu/condor
[3] legion.virginia.edu

OurGrid[4] e o InteGrade[5] . O Condor, desenvolvido pela University of Wisconsin-Madison, é um sistema de gerenciamento de recursos e escalonamento de tarefas, que tem como função principal permitir a utilização de máquinas ociosas. Com a versão Condor-G é possível a submissão e o monitoramento de tarefas em recursos de grades que utilizam o Globus como *middleware* [Li and Baker (2005)]. O Legion, desenvolvido pela University of Virginia, é um sistema baseado em objetos projetado para permitir a execução de aplicacões com alto grau de paralelismo, gerenciando as complexidades físicas para o usuário.

A.3 Tipos de grade

O desenvolvimento de *middleware* de grades e de redes de longa distância de alta velocidade e baixo custo possibilitaram o desenvolvimento de grades computacionais por todo o mundo.

As grades são usadas nas mais variadas aplicações e para cada aplicação há um tipo mais adequado de grade. Krauter, Buyya e Maheswaran propuseram, em 2002, uma taxonomia, identificando diferentes tipo de grades [Krauter et al. (2002)] de acordo com a atividade principal a que se destinam, conforme a figura A.2.

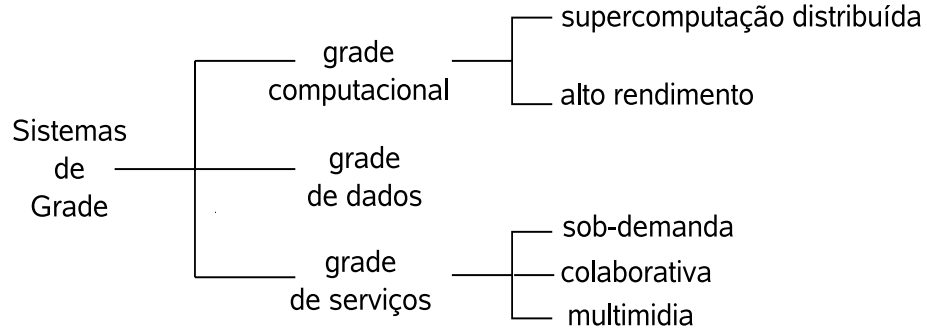


Figura A.2: Tipos de Grades.

Na categoria **grade computacional** encontram-se as grades que visam a

[4] www.ourgrid.org
[5] www.integrade.org.br

integração de recursos computacionais dispersos a fim de oferecer uma maior capacidade combinada de processamento aos seus usuários. Nesse tipo de grade o que é compartilhado são os ciclos de processamento. Este tipo pode ser subdividido nas categorias **grade de supercomputação distribuída** e **grade alto rendimento**. Uma grade de supercomputação distribuída agrega recursos computacionais afim de viabilizar a execução de aplicações que não poderiam ser executadas em um único sistema, as chamadas aplicações "*grand challenge*", como por exemplo a resolução de modelos numéricos de previsão de tempo. Já uma grade de alto rendimento é usada para escalonar um grande número de tarefas independentes com o objetivo de aproveitar ciclos de máquinas ociosas, sendo adequada por exemplo, para aplicações do tipo varredura de parâmetros. Esse subtipo de grade foi denominado "*scavenging grid*" [Livny et al. (1997)] ou grade computacional oportunista [Goldchleger (2004)].

A categoria **grade de dados** engloba os sistemas que fornecem infraestrutura para o armazenamento e o processamento de um grande volume de dados, potencialmente armazenados em repositórios, bibliotecas digitais e banco de dados, distribuídos geograficamente e conectados por uma rede de grande área.

A categoria **grade de serviços** inclui os sistemas que oferecem serviços viabilizados pela integração de diversos recursos computacionais. Esta categoria é subdividida em grade sob-demanda, grade colaborativa e grade multimidia. Uma grade sob-demanda oferece serviços que não podem ser encontrados localmente de forma conveniente e sem custo altos. Uma grade colaborativa permite a interação em tempo real de usuários e aplicações, oferecendo um ambiente de trabalho virtual. Uma grade multimidia oferece infra-estrutura para aplicações multimidia em tempo-real.

Podem haver grades com combinação de dois ou mais desses tipos e a tendência, a médio prazo, é que os diversos tipos de serviços e usos estejam presentes em todas as grades computacionais.

A.4 Grades em funcionamento

Entre as grades de supercomputação distribuída destacam-se: o TeraGrid[6] (TG), o Distributed European Infrastructure for Supercomputing Applications[7] (DEISA) e no Brasil, o Sistema Nacional de Processamento de Alto Desempenho[8] (SINAPAD). O TG é custeado pelo National Science Fundation e integra computadores de alto desempenho, recursos e ferramentas de dados de 11 parceiros nos EUA, atingindo atualmente mais de 750 teraflops de capacidade computacional e mais de 30 petabytes de armazenamento de dados, com acesso rápido através de redes de alto desempenho. O DEISA interliga 11 centros de supercomputação localizados em 7 países da Europa. No Brasil, o SINAPAD é formado por recursos de 8 centros de computação de alto desempenho, os CENAPADs, entre eles o LNCC. Grades desse tipo, que interligam supercomputadores e onde os usuários não podem adicionar recursos dinamicamente, são também denominadas grades infraestruturadas e em geral recebem apoio finaceiro de organizações governamentais. Em geral seus usuários são universidades que desenvolvem projetos de pesquisa nas mais diversas áreas, entre elas, ciências astronômicas, computação científica avançada, biofísica, química, ciências atmosféricas, geofísica, biociências moleculares e genética.

Como exemplo de grades oportunistas pode-se mencionar o Open Science Grid[9] (OSG), o World Comunity Grid[10] (WCG), o Enabling Grids for E-sciencE[11] (EGEE) e o E-Infrastructure shared between Europe and Latin America[12] (EELA); no Brasil, o InteGrade, o OurGrid e no LNCC, a grade VCG[13] . O OSG é uma grade orientada a pesquisas científicas com grande volume de dados. Nesta grade são desenvolvidos inúmeros projetos em áreas como energia negra, modelos de clima,

[6] www.teragrid.org
[7] www.deisa.org
[8] www.lncc.br/sinapad
[9] www.opensciencegrid.org
[10] www.worldcommunitygrid.org
[11] www.eu-egee.org
[12] www.eu-eela.org
[13] lncc.vcg.br

nanociência e várias outras. O OSG é baseado na Universidade de Indiana e seus coloboradores são universidades e centros de pesquisa, em sua grande maioria nos EUA, além de outros países, como no Brasil, a Universidade de São Paulo (USP) e a Universidade Estadual do Rio de Janeiro (UERJ). Na Europa o projeto EGEE, que já está na sua segunda fase (EGEE-II), é custeado pela Comissão Européia e envolve 140 instituições, com 300 sites distribuídos por 50 países num total de 80.000 núcleos de CPU. O EELA é um braço da EGEE para a América Latina. Lançado em 2006, inclui 41 centros de recursos, 13 na Europa e 28 na America Latina, incluindo o Brasil, com centros no LNCC, UFRJ, UFF, entre outros.

O OurGrid é coordenado pela Universidade Federal de Campina Grande e permite a execução de aplicações "bag-of-tasks" (BOT), isto é, aplicações com tarefas independentes, que não necessitam de comunicação durante a sua execução. Sua arquitetura é baseada no modelo par-a-par (peer-to-peer). Está em desenvolvimento desde dezembro de 2004 e reúne dezenas de laboratórios no Brasil e no exterior [Andrade et al. (2005)] e [Voorsluys et al. (2007)]. Uma das aplicações desenvolvida nesta grade é o projeto SegHidro, que pode ser acessado via portal e permite que cientistas e gerenciadores de projetos possam unir seus esforços na busca de soluções para problemas envolvendo a água na região Nordeste do Brasil. O portal oferece modelos de simulação numérica para a previsão do tempo que permitem analisar uma variedade de cenários a fim de auxiliar nas decisões relativas à gerenciamento de reservatório, planejamento da agricultura e controle de inundações [Araújo et al. (2005)]. Em 2005, o projeto Biopauá foi desenvolvido no LNCC em colaboração com o Instituto de Biofísica Carlos Chagas Filho (IBCCF) e a Universidade Federal do Rio de Janeiro (UFRJ), disponibilizando um ambiente Web para o desenvolvimento de pesquisas na área de Dinâmica Molecular por meio de plataforma de grade computacional, utilizando o *middleware* OurGrid.

O projeto InteGrade é uma iniciativa da USP, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), Universidade Federal do Mato Grosso do Sul (UFMS), Universidade Federal de Goiás (UFG), Universidade Federal do Mara-

nhão (UFMA) e a Universidade de Pernambuco (UPE). O *middleware* trabalha sobre arquitetura orientada a objetos distribuídos baseada em CORBA, e ao contrário do OurGrid, dá suporte a aplicações onde há comunicação entre as tarefas executadas em paralelo nos diversos nós - tarefas fortemente acopladas [Goldchleger et al. (2000)], [Junior et al. (2005)] e [Kon and Goldman (2008)].

O OurGrid e a InteGrade são também citados como *middleware*, uma vez que essas duas grades não utilizam um dos *middlewares* existentes, mas criaram seu próprio conjunto de ferramentas. Ambos trabalham com nível de seguraça fraca. O OurGrid não oferece suporte a privacidade dos dados e a criptografia dos dados. Já o Globus usa segurança forte, baseada na certificação de máquinas e usuários.

A Grade VCG foi utilizada para os experimentos desta dissertação e será descrita no próximo tópico.

O Earth System Grid[14] (ESG), custeado pelo U.S. Departament of Energy, é um projeto interdisciplinar colaborativo que tem o objetivo de permitir o gerenciamento, a descoberta, o acesso e a análise de cerca de 100TB de dados, crescendo rapidamente, referentes a simulações de modelos de clima global. O ESG é um exemplo de grade de dados que engloba recursos de sete parceiros, entre universidades e centros de pesquisa americanos [Bernholdt et al. (2005)].

O National Research Grid Initiative[15] (NAREGI), no Japão, foi criado em 2003 num projeto envolvendo a indústria, o setor acadêmico e o governo. Seu objetivo é pesquisar e desenvolver tecnologia de ponta para grades, como *middleware* e tecnologias de rede, afim de criar ambientes de grade que permitam a criação de novos produtos na área de nanotecnologia e biotecnologia.

A.4.1 Grade VCG

O LNCC desenvolveu, com apoio da RNP (Rede Nacional de Ensino e Pesquisa) e parceria de várias instituições (UFF, UFPR, UFSCAR, CBPF, entre outras), uma grade computacional, que tem por objetivo oferecer ao sistema nacional

[14] www.earthsystemgrid.org
[15] www.naregi.org

de pesquisa um ambiente para execução distribuída de tarefas computacionais que aproveite recursos ociosos de máquinas ligadas à rede da RNP. A grade VCG é uma grade oportunista, que permite via Internet, a inclusão de grupos de pesquisa e de recursos computacionais. O consumo e a provisão de recursos são geridos por um modelo de economia baseado em trocas de créditos entre os grupos de pesquisa. Neste ambiente, pode-se executar aplicações tipo BOT ou aplicações que exijam comunicação entre os nós. Entre os projetos em desenvolvimento podemos citar o projeto de simulação de próteses para reconstrução crânio-facial, desenvolvido pelo Prof. Gilson Giraldi, e o projeto para portar aplicações de modelagem de sistemas cardio-vasculares, escritas em MPI e executadas em cluster, para ambientes de grade, que vem sendo desenvolvido pelo HemoLab em parceria com a Universidade de São Carlos.

A Grade VCG vem disponibilizando cerca de 30 máquinas, entre Core 2, Pentium 4, Pentium D, Xeon Core 2. O sistema operacional utilizado é o Debian Linux Versão 4.0 e a versão de Java instalada nas máquinas é a 1.5.0.05.

Visando transformar a grade VCG em um ambiente facilmente utilizável por todos os usuários, foi desenvolvido um portalWeb, acessado através do endereço: http://portalvcg.lncc.br, a partir do qual é fornecida uma interface que permite a submissão de tarefas na Grade. Além disso, devido à infraestrutura de comunicação provida pela Internet, o portal permite que a grade seja acessada de qualquer lugar no mundo.

O desenvolvimento do portal foi feito utilizando o sistema operacional Linux e a tecnologia de *Portlets*, uma tecnologia Java. *Portlets* são componentes Web independentes que podem ser facilmente adicionados ou removidos do portal pelo desenvolvedor, possibilitando a inserção de *portlets* personalizados, ou seja, dedicados à submissão de aplicações específicas a um determinado projeto e que só aparecem para os usuários relacionados ao projeto. Sua arquitetura é baseada no *middleware* Globus Toolkit (versões 2.4 e 4.0.5)

No decorrer da dissertação, além do acesso via portal, a grade foi utilizada

também invocando diretamente os comandos do Globus, através de *scripts* que instalam e executam os aplicativos do sistema de otimização nas máquinas escolhidas.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo