EMERSON STIILPEN BATISTA

DESENVOLVIMENTO E AVALIAÇÃO DE UM SISTEMA DE ESCALONAMENTO DE PIVÔS CENTRAIS COM USO DE METAHEURÍSTICAS - UM ESTUDO DE CASO

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS - BRASIL
2007

Dedico esta vitória a todos que me ajudaram durante estes dois anos e 7 meses de Viçosa.

AGRADECIMENTOS

- Aos meus pais, Lúcia e Sebastião, por terem entendido o excesso de isolamento e o eventual mau-humor, mesmo sem entender 'patavina' do assunto da dissertação.

- Ao Heleno, meu orientador, pela orientação neste trabalho e pela ajuda no meu processo de crescimento pessoal.

- Aos meus irmãos Pedro e Isabella pelo apoio nos momentos difíceis.

- À Ana Amélia, minha namorada e amiga, por ter sido uma amiga quando isso é o que se espera de uma namorada, e por ter sido minha namorada quando isso é o que menos se espera de uma amiga.

- Ao Brauliro, meu amigo e conselheiro, pelo apoio, conselhos e indicações.

- A todos os professores do DPI pelos anos de aprendizado e convívio.

- Aos amigos da República, pelo colchão, o chuveiro e a parceria.

- Ao prof. José Luis Braga pelos conselhos e projetos externos que viabilizaram financeiramente este trabalho.

- À Universidade Federal de Viçosa pela oportunidade de conquistar o título de mestre.

BIOGRAFIA

Emerson Stiilpen Batista, filho de Sebastião Batista Silva e Lúcia de Fátima Stiilpen Batista, brasileiro, nascido em 17 de fevereiro de 1981 no município de Coronel Fabriciano, no Estado de Minas Gerais.

Em 1998 concluiu os cursos técnicos em Processamento de Dados na Escola Técnica da UNIVALE, em Governador Valadares - MG. Atuou de 1999 até 2002 como programador. No ano 2004 concluiu o curso de Ciência da Computação na Universidade Vale do Rio Doce - UNIVALE.

Em 2005 foi para a cidade de Viçosa para cursar o mestrado em Ciência da Computação na Universidade Federal de Viçosa - UFV, onde se tornou mestre do curso de mestrado do Departamento de Informática - DPI, defendendo esta dissertação em outubro de 2007.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

RESUMO

BATISTA, Emerson Stiilpen, M.Sc., Universidade Federal de Viçosa, outubro de 2007. **Desenvolvimento e avaliação de um sistema de escalonamento de pivôs centrais com uso de metaheurísticas - Um estudo de caso**. Orientador: Heleno do Nascimento Santos. Co-Orientadores: Brauliro Gonçalves Leal e Luiz Aurélio Raggi.

O pivô central é um sistema de irrigação de grande porte, que utiliza grandes quantidades de água durante seu funcionamento. A criação de uma escala de funcionamento para os pivôs centrais que atuam compartilhando um mesmo recurso hídrico é uma solução para reduzir custos com energia elétrica e controlar o consumo de água visando evitar possíveis danos ambientais sem comprometer a eficácia na irrigação. Para encontrar tal solução, um modelo matemático que representa o problema foi criado e validado. Este modelo serviu de base para a construção de algoritmos que geram a escala de funcionamento dos pivôs centrais. Devido ao grau de complexidade do problema, o uso das metaheurísticas *Simulated Annealing* e *GRASP* foi adotado na busca de boas soluções em curto espaço de tempo. A aplicação das metaheurísticas foi feita de maneira simples, oferecendo a opção de modificar os parâmetros do problema, de acordo com as necessidades encontradas, sendo possível adaptar a forma de execução dos algoritmos à diferentes realidades. Uma interface que utiliza os algoritmos desenvolvidos foi construída com o objetivo de facilitar a manipulação das informações de entrada e saída do sistema. Os resultados dos testes obtidos com o sistema utilizando ambos os algoritmos geraram soluções que atenderam plenamente a expectativa inicial sendo, inclusive, avaliados por profissionais da área. Além disto, um estudo de caso, utilizando os algoritmos desenvolvidos, foi realizado com base em informações reais do Projeto Colonização Paracatu Entre-Ribeiros, localizado na Ba-

cia do Rio São Francisco, um dos maiores perímetros de irrigação com pivô central da América Latina.

ABSTRACT

BATISTA, Emerson Stiilpen, M.Sc., Universidade Federal de Viçosa, October of 2007. **Development and evaluation of a system scheduler of central pivot using metaheuristics - a case study**. Adviser: Heleno do Nascimento Santos. Co-Advisers: Brauliro Gonçalves Leal and Luiz Aurélio Raggi.

The central pivot is an system of great importance, which uses much water resource during its working. The development of a working scale to the central pivots that act using one same water resource is a solution to reduce costs with electric energy and to control the water consumption being aimed at to avoid possible nature damages, without compromising its efficiency. To find such solution, a mathematical model that represents the problem was created and validated. This model was the base for the construction of algorithms which solving the scale of the central pivots. Due to the complexity of the problem, the use of metaheuristic Simulated Annealing and GRASP was adopted in the searches of good solutions in very small period of time. The application of the metaheuristics made very simply, offering the options to modify the parameters of the execution, in accordance with the found necessities, being possible to adapt the form of execution of the algorithms to the different realities. An interface that uses the algorithms was constructed with the objective to facilitate to the manipulation of the input and output of the system. The obtained results of tests with the system using both the algorithms had generated solutions that had taken care of the initial expectation fully being, also, evaluated for professionals of the area. Moreover, a case study, using the developed algorithms, were carried through with basis of information of the Project Colonização Paracatu Entre-Ribeiros, located in the Basin of the Rio São Francisco, one of the biggest perimeters of irrigation with central pivot of Latin America.

Capítulo 1

Introdução

1.1 O problema e sua importância

O pivô central é um sistema de irrigação muito utilizado em grandes áreas cultivadas. Ele consiste, basicamente, de uma tubulação metálica, onde são instalados os aspersores. A tubulação recebe a água sobre pressão de um sistema de bombeamento que alimenta o centro do pino e se apóia em torres metálicas triangulares, montadas sobre rodas de pneu. As torres movem-se acionadas por dispositivos elétricos ou hidráulicos, descrevendo movimentos concêntricos ao redor do ponto do pivô. O movimento da última torre inicia uma reação de avanço em cadeia de forma progressiva para o centro.

Quanto maior é o raio de um pivô, maior é a sua área de irrigação e, consequentemente, maior a sua lucratividade e o seu consumo de água. Por se tratar de equipamento de irrigação de larga escala um volume de água muito alto se faz necessário para que um pivô central funcione (Keller and Bliesner, 1990).

Desta forma, caso haja uma região que possua muitos pivôs fazendo uso de um mesmo recurso hídrico comum, como um rio, por exemplo, existirá sérios riscos de acontecerem prejuízos ambientais caso estes pivôs funcionem simultaneamente, sem nenhum tipo de controle. Este risco gera a necessidade de se controlar todo um conjunto de pivôs centrais, a partir da construção de uma escala de funcionamento, de modo que as plantações e o ecossistema não sejam prejudicados.

Atualmente, o controle dos pivôs centrais é feito através de autorizações para a instalação, outorgadas por órgãos estaduais, como por exemplo, o Instituto Mineiro de Gestão das Águas, atuante no estado de Minas Gerais, que verifica a capacidade de um recurso hídrico poder ceder parte de sua vazão para o referido sistema de irrigação e quais os impactos ambientais que esta instalação causaria. É importante destacar que trabalha-se com uma margem de segurança muito alta, para evitar problemas graves causados por possíveis falhas. Apesar de eficaz em relação ao seu objetivo, esta margem de segurança limita significativamente a quantidade de água útil.

Este estudo não almeja substituir as práticas atualmente utilizadas para controlar os sistemas de irrigação por pivô central. Um dos objetivos deste trabalho é criar uma ferramenta que possa trabalhar em parceria com as práticas atuais, agregando alternativas e funcionalidades.

Uma pesquisa foi feita, consultando profissionais da irrigação e a rede mundial de computadores, buscando soluções ou estudos similares à proposta deste trabalho. O resultado encontrado nesta pesquisa é que não há nada semelhante, em estudo ou pronto, que tenha como meta criar um escalonamento, ou agendamento, de horários de funcionamento dos pivôs centrais, ou outras ferramentas de irrigação, visando controlar o consumo de recursos hídricos e otimizar o consumo de energia elétrica. Desta forma, podemos classificar este trabalho como pioneiro em escalonamento de pivôs centrais via *software* de computador.

Além disto, alguns fatores contribuem para o estudo desenvolvido, pois de acordo com uma análise da situação atual, caso altenativas não sejam criadas, possivelmente a instalação de novos pivôs centrais ficará extremamente restrita.

O Projeto de Colonização Paracatu-Entre Ribeiros, que é tratado como estudo de caso neste trabalho, possui uma situação complexa, na qual possui 180 pivôs centrais atualmente instalados e mais 16 projetados, aguardando autorização. Segundo Pruski et al. (2007), a região do Paracatu apresentou o maior uso da água pelo setor agropecuário, principalmente pela agricultura irrigada. Em 2006 o problema se faz mais grave e a não realização de ações em curto prazo pode gerar a insustentabilidade do referido perímetro de irrigação, um dos mais importantes da América Latina.

O problema de escalonamento de pivôs centrais consiste no desenvolvimento de uma escala de horários para o funcionamento destes pivôs. Esta escala deve buscar soluções que minimizem o consumo de energia elétrica do conjunto de pivôs centrais envolvidos na escala de funcionamento, respeitando todas as restrições existentes, que visam proteger o ecossistema sem prejudicar a produtividade das lavouras.

Cada pivô central atende a apenas um cliente fixo. Os clientes possuem áreas irrigáveis de plantações, cobertas pelo pivô e que possuem, a cada dia, um número diferente de horas de irrigação, via pivô central. Estas horas são estipuladas pelo técnico agrícola responsável pelo cliente, com o objetivo de atender as necessidades globais da plantação. O número de horas de cada pivô central pode variar em função de diversos fatores externos como clima, cultura irrigada, tipo de solo, entre outros.

Por se trabalhar, sempre, com o tempo de irrigação em horas diárias, toda solução construída estará dividida em 24 janelas de tempo, que correspondem às horas do dia. Assim, todo pivô terá o seu período de trabalho distribuído entre as 24 janelas de tempo que compõe a escala, diariamente.

A principal restrição a ser controlada no problema é o volume de água total, utilizado por todos os pivôs centrais de um grupo, em cada janela de tempo. Cada pivô central possui um volume de água fixo, necessário para seu funcionamento. Sendo assim, a soma dos volumes de todos os pivôs centrais que estiverem funcionando, em uma determinada janela de tempo, sempre deverá ser menor ou igual ao volume de água que se permite retirar do recurso hídrico. Esta restrição é fundamental para evitar danos ambientais ao ecossistema.

O problema de escalonamento de pivôs centrais tem como meta a minimização do consumo de energia de todo o problema, que pode ser obtida de duas maneiras. A primeira maneira é graças ao custo reduzido da tarifa de energia elétrica em períodos noturnos, chegando esta tarifa a ser até 60 % mais barata nestes horários. A segunda maneira é escalonando os pivôs centrais de modo que eles funcionem o maior número possivel de janelas de tempo ininterruptas, evitando-se assim o alto custo de retirar o pivô central do estado de ócio e colocá-lo em funcionamento.

1.2 Objetivos

- Desenvolver um modelo matemático que represente fielmente o problema de escalonamento de pivôs centrais e suas restrições;
- criar uma estrutura de dados que permita armazenar e acessar às informações do problema facilmente, com eficiência;
- implementar os algoritmos *Simulated Annealing* e GRASP, com base no modelo proposto e fazendo uso da estrutura de dados construída, visando encontrar boas soluções para o escalonamento dos pivôs centrais;
- criar uma ferramenta computacional que utilize os algoritmos implementados, viabilize a inserção de informações nas estruturas de dados e facilite a interpretação das soluções encontradas pelos algoritmos;
- possibilitar a análise de cenários por meio da ferramenta computacional implementada para o estudo de caso proposto.

1.3 Organização do texto

Esta dissertação está organizada da seguinte forma.

A Seção 1 mostra a importância do escalonamento de pivôs centrais e da utilização de procedimentos heurísticos na busca de solução para os problemas nessa área.

A Seção 2 descreve o conceito de pivô central, de otimização combinatória, as metaheurísticas Simulated Annealing e GRASP, de acordo com diversos conceitos apresentados na literatura, e a apresentação do problema tratado como um estudo de caso.

A Seção 3 descreve o modelo criado e utilizado para o problema de escalonamento de pivôs centrais, os algoritmos *Simulated Annealing* e GRASP implementados, o sistema construído para alimentar a estrutura de dados dos algoritmos e os problemas testes utilizados para mostrar a potencialidade de aplicação das metaheurísticas

na solução de problemas de escalonamento de pivôs centrais.

A Seção 4 faz uma análise comparativa das soluções obtidas pelos algoritmos propostos e pelo software comercial Lingo 3.0 (Schrage, 1998). Algumas considerações são feitas sobre estas análises. Também são apresentados e analisados os resultados obtidos no estudo de caso.

Finalmente, a Seção 5 apresenta algumas conclusões sob o que foi realizado e sugestões para trabalhos futuros.

Capítulo 2

Referencial Teórico

2.1 Otimização Combinatória: Problemas e Métodos de Solução

Segundo Goldbarg and Luna (2005), problemas de Otimização Combinatória aparecem quando é necessário selecionar, de um conjunto discreto e finito de dados, o melhor subconjunto que satisfaz a determinados critérios.

Existem muitas classificações possíveis para o problema de otimização, e algumas delas apresentarão métodos exatos e eficientes de resolução. Outras levarão à necessidade de métodos não-exatos, também conhecidos como heurísticos, uma vez que sua formulação e/ou resolução exatas levariam a uma complexidade intratável.

Entre os problemas de otimização combinatória clássicos podemos destacar, entre os diversos existentes 'O Problema da Mochila', 'Caminho Mais Curto' e 'Escalonamento de Máquinas'.

O problema da Mochila caracteriza-se pelo estreito relacionamento com um grande número de outros modelos de programação. Sua importância está associada exatamente a esse fato. Metaforicamente podemos entendê-lo como o desafio de encher uma mochila sem ultrapassar um determinado limite de peso, otimizando o valor do produto carregado. Constitui um marco das técnicas de programação inteira, ottimização combinatória e programação dinâmica (Cormen et al., 2002).

O problema do 'Caminho Mais Curto' consiste em encontrar a menor rota, em um conjunto de vértices, ou caminhos, possíveis, que permita sair da origem e chegar ao destino percorrendo a menor distância possível. Este problema está intimamente relacionado à solução de vários problemas combinatórios como so de roteamento, programação e seqüenciamento de tarefas etc (Goldbarg and Luna, 2005).

O problema de 'Escalonamento de Máquinas' pode ser entendido como o problema de alocar n máquinas de produção a n tarefas. Cada um das máquinas é capaz de atender à uma tarefa segundo um custo. A idéia, normalmente, é minimizar o custo total gerado pelo uso das máquinas e maximizar o lucro gerado pelos produtos criados por elas.

Todos estes problemas possuem várias possiveis formas de resolução, podendo elas serem exatas ou não. Cabe ao responsável por buscar estas soluções mensurar a viabilidade e eficácia do método a ser utilizado.

2.1.1 Métodos Exatos

Segundo Goldbarg and Luna (2005), os métodos exatos procuram a melhor solução, ou a solução ótima, para um problema, quando esta solução existe, satisfazendo todas as restrições impostas.

Através de uma modelagem matemática, transcrição do problema para linguagem matemática, o problema é resolvido pelo Método Simplex, desenvolvido por Dantzig et al. (1955) ou algum outro algoritmo exato.

Programação Linear

O Simplex trata de um algoritmo geral extremamente eficiente para solução de sistemas lineares e adaptável ao cálculo computacional, cuja compreensão funcional embasa vários outros métodos. Seu estudo é indispensável para o profissional que deseja dominar as técnicas quantitativas de análise e solução de problemas em um contexto razoavelmente avançado (Goldbarg and Luna, 2005).

Ele começa com um sistema de desigualdades lineares cuja solução é desconhecida. Em cada iteração executada, reescreve-se esse sistema em uma forma equivalente

que tem alguma estrutura adicional. Após certo número de iterações, reescrevemos o sistema de modo que a solução seja simples de se obter.

Em associação a cada iteração, há uma solução básica facilmente obtida a partir da forma relaxada do programa linear, definindo-se cada variável não básica como zero e calculando-se os valores das variáveis básicas a partir das restrições de igualdade. Uma solução básica sempre corresponde a um vértice do Simplex. Algebricamente, uma iteração transforma uma forma relaxada em uma forma relaxada equivalente. O valor de objetivo da solução básica possível associada não será menor que o da iteração anterior. Para alcançar esse aumento no valor de objetivo, escolhe-se uma variável não básica tal que, se for aumentado o valor dessa variável desde zero, então o valor de objetivo também aumentaria. A quantidade pela qual é aumentada a variável é limitada pelas outras restrições. Em particular, aumenta-se até alguma variável básica se tornar zero. Em seguida, reescreve-se a forma relaxada, trocando as funções dessa variável básica e da variável não básica escolhida. Apesar de estar sendo usada uma configuração especifica das variáveis para orientar o algoritmo, o algoritmo não mantém de forma explícita essa solução. Ele simplesmente reescreve o programa linear ate a solução ótima se tornar 'óbvia' Cormen et al. (2002).

Desde 1950, a programação linear vem causando um impacto extraordinário para empresas de todos os portes, podendo ser considerada como um dos maiores avanços científicos do século vinte. Atualmente é uma ferramenta de planejamento que poupa milhões de dólares em empresas em todo o mundo (Hillier and Lieberman, 1988).

Programação linear é uma ferramenta de planejamento utilizada para auxiliar a selecionar que atividades (variáveis de decisão) empreender, dado que essas atividades (diversas alternativas) competem entre si pela utilização de recursos escassos (restrições) ou então precisam satisfazer certos requisitos mínimos (Ehrlich, 1978).

Segundo Andrade (2000), um problema deste tipo é caracterizado pelos seguintes fatos:

- existência de um objetivo que pode ser explicitado em termos das variáveis de decisão do problema;

- existência de restrições à aplicação dos recursos, tanto com relação às quantidades disponíveis quanto com relação à forma de emprego;
- pode ser representado por um modelo de otimização, onde todas as relações matemáticas são lineares.

Segundo Cormen et al. (2002), em geral, utiliza-se programação linear para otimizar uma função linear de acordo com um conjunto de desigualdades lineares. Dado um conjunto de números reais $a_1, a_2, ..., a_n$ e um conjunto de variáveis $x_1, x_2, ..., x_n$, uma função linear f sobre essas variáveis é definida por

$$f(x_1, x_2, ..., x_n) = a_1x_1 + a_2x_2 + ... + a_nx_n = \sum_{j=1}^{n} a_jx_j.$$

.

Branch and Bound

Segundo (Puccini and Pizzolato, 1990), um problema de Programação Linear Inteira (PLI) é um problema de Programação Linear (PL) em que todas ou alguma(s) das suas variáveis são discretas, e por isso devem assumir valores inteiros. Quando todas as variáveis estão sujeitas à condição de integralidade estamos perante um problema de Programação Linear Inteira Pura (PLIP); e se apenas algumas o estão trata-se de um problema de Programação Linear Inteira Mista (PLIM). Embora a Programação Inteira (PI) inclua também a Programação Não-Linear Inteira, na maioria dos modelos da vida real se preserva a estrutura linear das funções.

Para a resolução de problemas de PLI, destacam-se dois métodos: o Método dos Planos de Cortes (*Cutting Planes*) e o Método de Partição e Avaliação Sucessivas (*Branch and Bound*), ambos utilizam o algoritmo Simplex para chegar à solução ótima de problemas de PL cuja região admissível vai sendo sucessivamente reduzida até se alcançar a solução do problema de PLI. Estes métodos são gerais, podendo ser aplicados a qualquer modelo de PLI.

O método dos Planos de Corte foi o primeiro método a ser desenvolvido e deve-se a Gomory (1958). Consiste em introduzir sucessivamente novas restrições no

problema linear do PLI, restrições essas que cortam o conjunto das soluções possíveis eliminando algumas delas e a própria solução ótima do PL (por isso se chamam planos de corte), sem contudo eliminar qualquer solução inteira (Puccini and Pizzolato, 1990).

O método *Branch and Bound*, literalmente, método de ramificação e limitação, foi apresentado inicialmente por Land and Doig (1960) e consiste na partição, ou ramificação, sucessiva do conjunto de soluções possíveis do problema de PLI em subconjuntos e na limitação do valor ótimo da função objetivo (limite inferior se tratar de maximização, ou superior se tratar de minimização), de modo a excluir os subconjuntos que não contenham a solução ótima. Se na solução ótima de um problema de PL as variáveis tomam valores inteiros, essa é a solução ótima do PLI. Então se começa a resolver o PLI como um problema de PL. Quando sua solução não atende às restrições de integralidade, divide-se o problema de PL em dois, através da introdução de restrições adicionais, eliminando tal solução do conjunto das soluções possíveis. São resolvidos sucessivos problemas de PL, estabelecendo-se limites para o valor ótimo da função objetivo e eliminando diversos subconjuntos, até se alcançar a solução ótima do PLI (Goldbarg and Luna, 2005).

Existe um caso especial PLI: os problemas onde as variáveis devem ser binárias, ou seja podem assumir os valores 0 (zero) ou 1 (um) que são os problemas de Programação Linear Binária (PPLB). Quando todas as variáveis de um modelo são binárias, o modelo diz-se de Programação Inteira Binária. As variáveis binárias são muito úteis para exprimirem situações dicotômicas (sim ou não, fazer ou não fazer, e outros) (Zionts, 1974).

2.1.2 Métodos Heurísticos

"'Heurísticas são regras que podem auxiliar a solucionar certos tipos de problemas, mas, não garantem que se chegue à solução ótima."' (PERKINS, 1981, apud CARVALHO, 2005, p. 3)

Segundo da Silva Gomes and Neto (2003), métodos heurísticos, em sua grande maioria, são utilizados para resolver problemas que não podem ser tratados por métodos exatos, tentando reproduzir, muitas vezes, o que é realizado manualmente. Desta

forma o objetivo principal de uma heurística é encontrar soluções de boa qualidade, muitas vezes melhores que as conhecidas, mas com um baixo custo computacional, ou seja, de forma rápida.

É comum que uma solução boa seja encontrada, e algumas vezes uma solução ótima é obtida. Desta forma, quanto maior for o número de informações (restrições) fornecidas às heurísticas, maiores são as chances de se obter uma solução que satisfaça o problema no começo das iterações.

Como em uma heurística não se pode em geral garantir que a solução encontrada seja a solução ótima, podemos limitar sua busca. Esta limitação pode ser realizada através de um número fixo de iterações, retirada ou relaxamento de restrições, entre outras técnicas. Desta forma pode-se estar eliminando a solução ótima para o problema.

Na década de 1970, um novo tipo de algoritmo de aproximação emergiu sendo basicamente uma tentativa para combinar métodos heurísticos básicos em um *framework* de alto nível visando explorar um espaço de busca de forma eficiente e efetiva. Estes métodos são hoje em dia chamados de metaheurísticas. O termo metaheurística, inicialmente introduzido por Glover (1986), vem da composição de duas palavras gregas: Heurística deriva do verbo grego *heuriskein* que significa 'procurar', enquanto meta significa 'em um nível superior'. Este termo foi adaptado do termo anterior, denominado Heurísticas Modernas (Reeves, 1993).

Embora não haja um consenso exato sobre a definição de metaheurística, o conceito mais aceito nos meios acadêmicos, pode ser encontrado em Glover and Kochenberger (2002):

> *"'Uma metaheurística é um conjunto de conceitos que podem ser utilizados para definir métodos heurísticos aplicáveis a um extenso conjunto de diferentes problemas. Em outras palavras, uma metaheurística pode ser vista como uma estrutura algorítmica geral que pode ser aplicada a diferentes problemas de otimização com relativamente poucas modificações que possam adaptá-las a um problema específico"'* (Glover and Kochenberger, 2002).

Metaheurísticas dividem-se em duas categorias: metaheurísticas de busca local (LSMs) e algoritmos evolucionários (EAs). Uma busca local inicia com uma solução

inicial e, a cada etapa de busca, a solução atual é substituída por outra (normalmente, a melhor) solução melhorada e identificada na vizinhança. Normalmente, LSMs dispõem de mecanismo para diversificar a busca, fugindo de ótimos locais e explorando todo o espaço de soluções. Já os EAs fazem uso de um população de soluções gerada aleatóriamente. A cada iteração do processo de busca, a população inteira ou uma parte da população são substituídas por indivíduos recentemente gerados e melhores (Alba et al., 2005). Também, aqui, mecanismos de diversificação garantem uma melhor exploração do espaço de soluções.

Alguns dos principais métodos de solução conhecidos na categoria LSMs são: *Simulated Annealing* (Kirkpatrick et al., 1983), *Tabu Search* (Glover, 1989), *Greedy Randomized Adaptative Search Procedure* (Feo and Resende, 1995), *Variable Neighborhood Search* (Mladenovic and Hansen, 1997)); enquanto na categoria dos EAs: Algoritmos Genéticos (Bäck et al., 1997), Estratégias Evolucionarias (Bäck et al., 1997), *Genetic Programming* (Bäck et al., 1997), *Ant Colonies* (Dorigo, 1992), *Estimation of Distribution Algorithms* (Mühlenbein et al., 1999), *Scatter Search* (López et al., 2006)). Outros métodos são também esboçados sob as denominações de metaheurísticas heterogêneas e otimização multiobjetivo paralela.

Simulated Annealing

Segundo Reeves (1993) o uso do *Simulated Annealing* (SA) como uma técnica para otimização discreta surgiu no início dos anos 80. A idéia que forma a base do *simulated annealing* foi primeiramente publicada por Metropolis et al. (1953) em um algoritmo para simular o resfriamento de um material em banho quente - um processo conhecido como *annealing* (recozimento, temperamento, etc). Essencialmente, o algoritmo de Metropolis et al. (1953) simula a mudança de energia do sistema quando sujeito a um processo de resfriamento, até ele convergir para um estado calmo e ordenado (*congelado*). Trinta anos mais tarde, a analogia com a otimização combinatória foi introduzida por Kirkpatrick et al. (1983) e, posteriormente, aperfeiçoada por Cerny (1985) e Reeves (1993), sugerindo que este tipo de simulação poderia ser usado para procurar soluções viáveis em um problema de otimização, com o objetivo de convergir

para uma solução ótima.

A metaheurística SA é basicamente um algoritmo de busca local, sendo que a escolha de um novo elemento na vizinhança é realizada de forma semi-aleatória. Os algoritmos de busca local apresentam o inconveniente de estacionar num ótimo local. A metaheurística SA evita este problema, utilizando uma probabilidade de aceitação de uma solução que piora a solução corrente, o que implica na possibilidade de se escapar de um ótimo local. Esta probabilidade vai diminuindo, à medida que se aproxima da solução ótima (Ibaraki et al., 2005).

Segundo Rodrigues (2001), citando Ignacio et al. (2000), uma das vantagens da SA é o fato que, em cada iteração, uma única solução é visitada, ao contrário de muitas técnicas como os Algoritmos Geneticos (AG) e a Busca Tabu (BT), onde várias soluções são visitadas em cada iteração. Desta forma, processamentos como cálculo da função objetivo e outros processamentos necessários para avaliar as soluções pesquisadas não degradam muito a eficiência do algoritmo. Uma das desvantagens da SA é o fato de a heurística utilizar poucas informações do problema. De modo geral, a SA utiliza somente a variação do valor da função objetivo nas avaliações dos movimentos, sendo este procedimento considerado miúpe por não considerar outros aspectos do problema na pesquisa do espaço de soluções, o que torna o algoritmo pouco '*inteligente*'.

Para a utilização do SA, deve-se definir a priori, um método para geração de uma solução inicial S, um método para geração das soluções vizinhas S' (estrutura de vizinhança), e uma função objetivo $f(S)$ a ser otimizada (Mauri and Lorena, 2006). O diagrama de atividades da metaheurística SA é apresentado na figura 2.1, onde S é a solução atual e S' é a solução vizinha.

Segundo Blum and Roli (2001) o algoritmo inicia por gerar uma solução inicial (construída aleatoriamente ou heuristicamente) e por inicializar um parâmetro T chamado de temperatura. Então ele repete o processo de busca até alcançar algum critério de parada. Diferentes critérios de parada são adotados: tempo máximo de processamento, número máximo de iterações, quando é encontrada uma solução S com $f(S)$ (valor da função objetivo) menor que um valor predefinido, ou um número

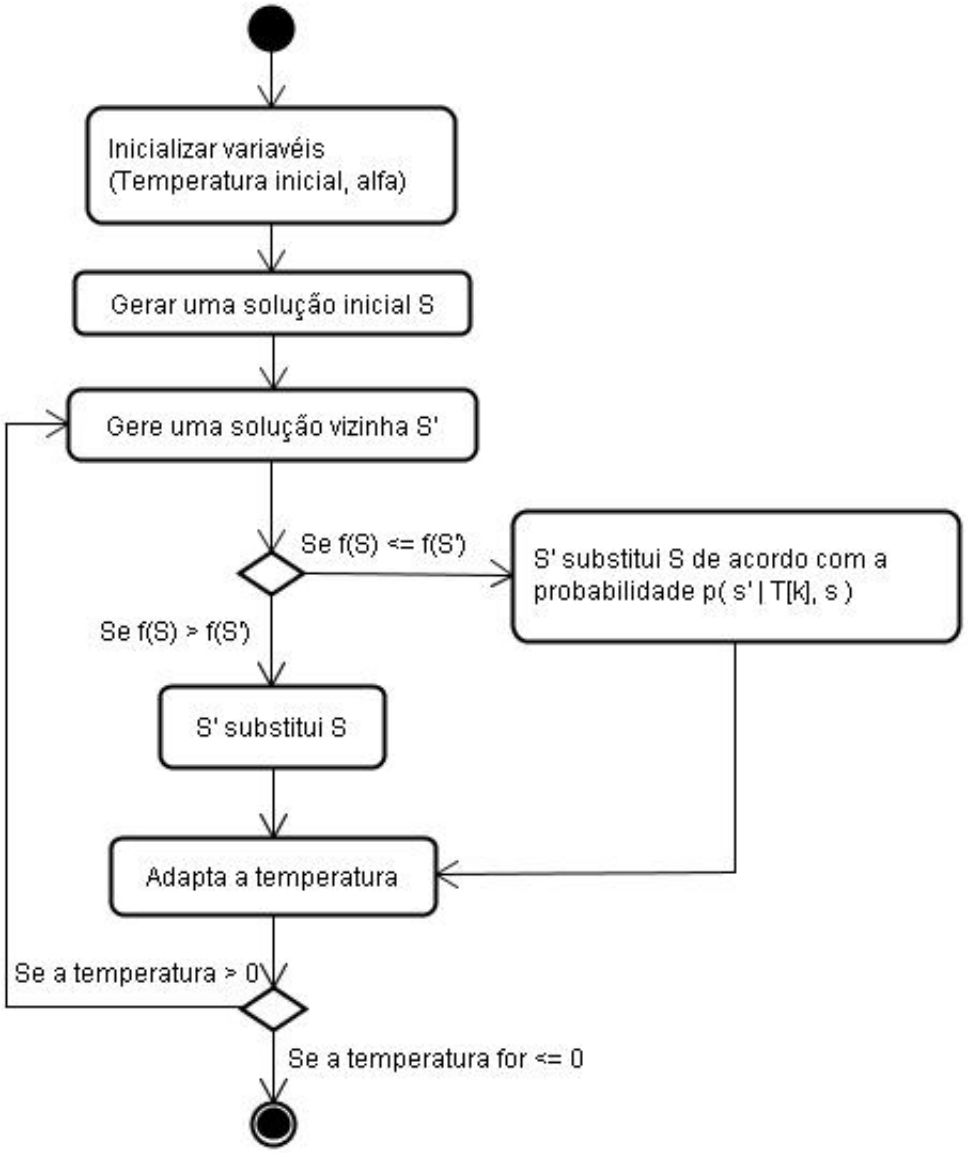

Figura 2.1: Diagrama de atividades do *Simulated Annealing*.

máximo de iterações sem alcançar um melhoramento.

De acordo com Busseti (2001), citado em da Silva Gomes (2003) a principal vantagem da SA sobre os outros métodos é uma habilidade para evitar cair em um mínimo local. O algoritmo emprega uma busca aleatória que não só aceita mudanças que decrescem o valor da função objetivo mais também mudanças que aumentam o seu valor. Soluções que melhoram o valor da função objetivo são sempre aceitas e soluções que pioram o valor da função objetivo são condicionalmente aceitas, dependendo do critério de *Metropolis*. Porém, a probabilidade de aceitar um movimento que piore o valor da função objetivo decresce com a temperatura. O que significa que quanto menor a temperatura, menor a probabilidade de movimentos degradantes serem aceitos (Noronha et al., 2001).

O critério de *Metropolis* (Metropolis et al., 1953), conforme exposto por Baskent and Jordan (2002), é baseado na seguinte função de probabilidade:

$$P(S) = e^{\frac{-(f(S')-f(S))}{T}} \tag{2.1}$$

onde:

- $P(E)$ = probabilidade (entre 0 e 1) de uma solução inferior ser aceita.
- T = é o parâmetro de controle.
- $f(S)$ = é o valor da melhor solução da função objetivo encontrada até o momento.
- $f(S')$ = é o novo valor da função objetivo encontrado.

A probabilidade de $P(S)$ é calculada e comparada com um número (r) aleatório e contínuo, uniformemente distribuído entre 0 e 1. A nova solução será aceita se $P(S) > r$, caso contrário, ignorada. O parâmetro de controle T é a temperatura que se inicia com um valor alto e é gradativamente reduzida, no final do processo é reduzida para um ponto onde somente as melhores soluções são aceitas. O algoritmo pára quando um valor para o parâmetro T seja satisfeito ou quando a função objetivo é atendida.

Segundo Pereira and do Nascimento Santos (2004), a temperatura T é considerada como um parâmetro de controle da mobilidade do sistema, ou seja, traduz a flexibilidade de aceitar novas soluções ou não. Cabe assinalar que, quanto menor a temperatura T, menor será a probabilidade de aceitar soluções que piorem a função objetivo, o que se traduz em rejeitar cada vez mais aquele tipo de soluções. Quando T é maior, a probabilidade fica próxima de um.

Não existem regras gerais para definir a redução da temperatura. Ao se considerar temperaturas altas como temperatura inicial, a qualidade da solução da SA não dependerá da solução inicial e permitirá o percurso de um espaço de soluções mais amplo. No entanto, é conveniente considerar uma temperatura inicial, dependente do problema. A velocidade de redução da temperatura irá implicar no número de soluções a serem visitadas. Se o processo de redução for lento implicará em tempos de processamento muito grandes, caso contrário pode-se acelerar o processo de redução diminuindo-se assim o tempo de processamento, implicando em menos soluções a serem visitadas podendo não percorrer todo o espaço de soluções possíveis (Pereira, 2004).

GRASP

A metaheurísticas GRASP, de *Greedy Randomized Adaptive Search Procedures*, foi proposta inicialmente em 1989 por Feo and Resende (1989). Ela pode ser vista como uma metaheurística que se utiliza das boas características dos algoritmos puramente gulosos e dos procedimentos aleatórios na fase de construção de soluções viáveis.

O GRASP é um método iterativo probabilístico, onde a cada iteração é obtida uma solução para o problema em estudo. Cada iteração GRASP é composta de duas fases: a construtiva, que determina a solução que será submetida à busca local ou fase de melhoria, segunda fase do algoritmo, cujo objetivo é tentar obter alguma melhoria na solução corrente (Feo and Resende, 1995; Resende and Ribeiro, 2003). Na figura 2.2 o diagrama de atividades do algoritmo GRASP é apresentado.

Na maioria das aplicações, o critério de parada é baseado no número máximo de iterações. Podem-se definir outros critérios, como por exemplo parar quando a solução

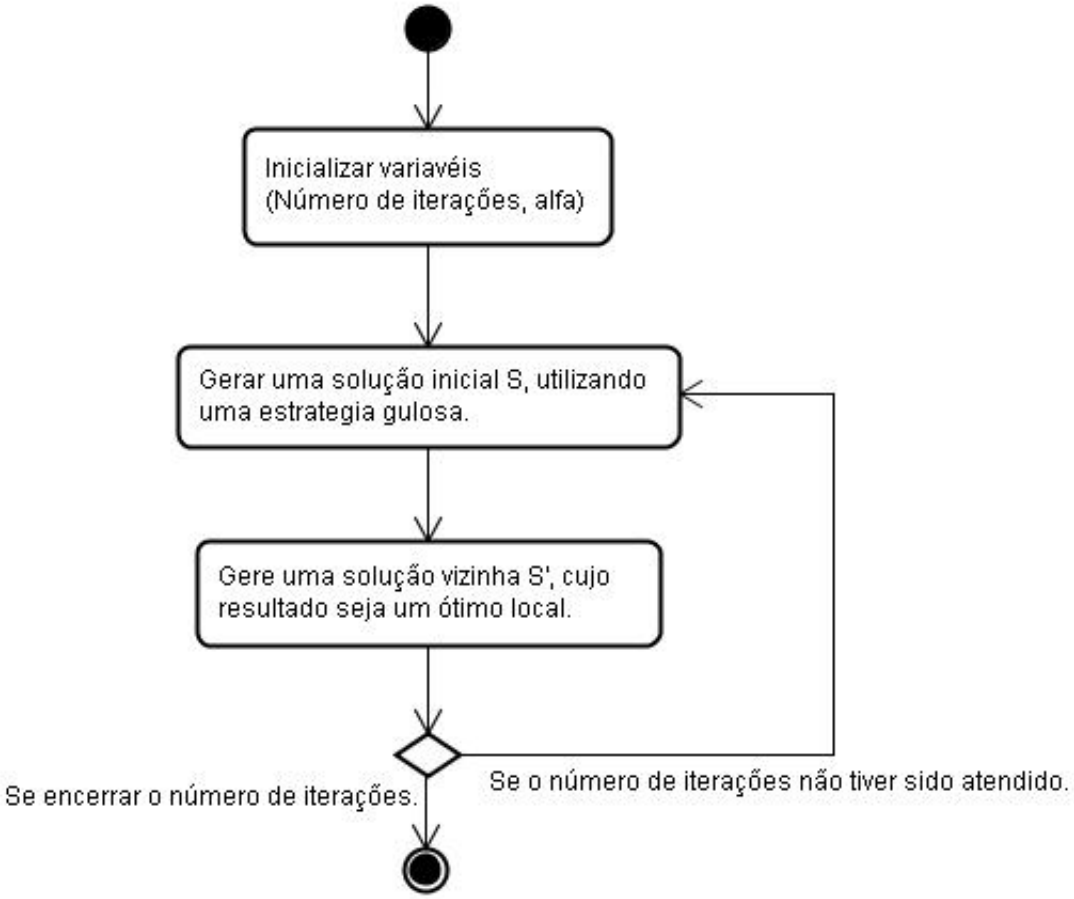

Figura 2.2: Diagrama de atividades do GRASP.

procurada for encontrada ou estabelecer um tempo máximo de execução (Rangel et al., 2000).

Segundo Melo and Martinhon (2004), na fase construtiva, a geração da solução viável e construída iterativamente, um elemento por vez. A cada iteração, o próximo elemento a ser adicionado é determinado pela ordenação de todos os elementos numa lista de candidatos C, isto é, todos que podem ser adicionados a solução, respeitando uma funcao gulosa g : $C \rightarrow \Re$. Esta função estima os benefícios na escolha de cada elemento. A heurística é adaptativa, por que os benefícios associados a cada elemento, são atualizados a cada iteração da fase de construção e refletem nas mudanças trazidas através da seleção do elemento anterior.

O componente probabilístico do GRASP é caracterizado pela escolha aleatória de um dos melhores candidatos da lista, não necessariamente do melhor. A lista dos melhores candidatos é chamada de *Restricted Candidate List* (RCL), isto é, Lista Restrita de Candidatos. A escolha do próximo elemento é dita adaptativa, pois é guiada por uma função gulosa que mede, de forma míope, o benefício que o mais recente elemento adiconado à solução concede à parte já construída. O GRASP possui uma componente probabilística, em vista da escolha aleatória na lista de candidatos (em um guloso simples seria selecionado o primeiro elemento da lista). Esta técnica de escolha permite que diferentes soluções sejam geradas a cada iteração GRASP. A figura 2.3 ilustra a fase de construção da heurística.

Como no caso de vários métodos de construção, as soluções geradas a partir da fase de construção do GRASP não garantem alcançar um ótimo local. Por isso é quase sempre benéfico aplicar uma busca local na tentativa de melhorar cada solução construída. O algoritmo de busca local trabalha de uma maneira iterativa, através de sucessivas trocas na solução corrente por uma solução melhor na sua vizinhança. Esta termina, quando nenhuma solução melhor e encontrada na vizinhança.

De acordo com Serra and Moura (2006), as RCLs de tamanho fixo apresentam bons resultados apenas quando seu tamanho representa algum percentual significativo do total de elementos, e mesmo assim com alguma degeneração ao final do processo iterativo, pois a quantidade de elementos com um bom atendimento relativo da de-

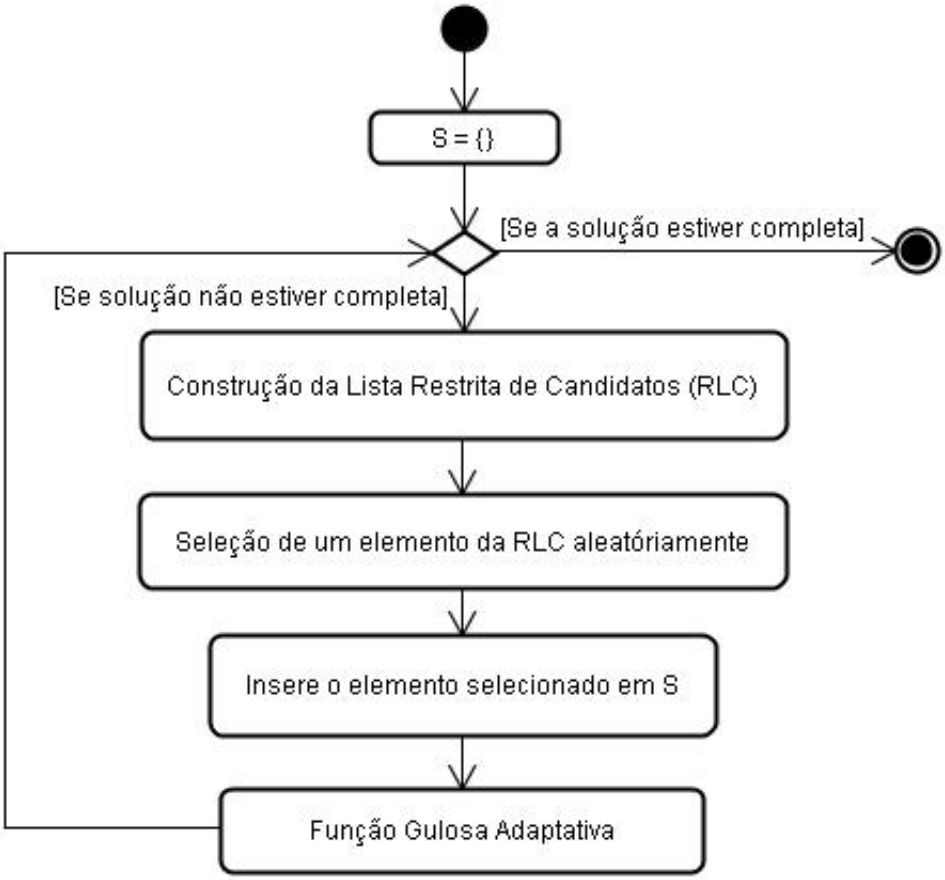

Figura 2.3: Diagrama de atividades da fase construtiva do GRASP.

manda passa a ser cada vez menor. Para contornar este problema, uma boa alternativa é o uso do parâmetro α ao inves das RCLs de tamanho fixo.

Segundo Nogueira et al. (2003), uma vez obtida uma solução S, inicia-se a fase de melhoria da solução, que consiste em uma busca local, que objetiva melhorar a solução míope aleatória construída. Assim, pela aleatoriedade das escolhas, várias rodadas do algoritmo resultam em várias soluções, possibilitando a escolha da melhor delas como a solução heurística do problema, o que implica maior probabilidade de se achar a solução ótima.

O processo de melhoria da solução tem como base uma consulta a estrutura de vizinhança $Viz(S)$ relativa à solução S. Uma solução é dita localmente ótima se não existir nenhuma solução melhor em $Viz(S)$. As soluções iniciais do GRASP não são necessariamente ótimos locais. Como consequência, faz-se necessária a aplicação de um procedimento de busca local para tentar melhorar as soluções advindas da fase construtiva. Esta busca realiza sucessivas trocas da solução corrente, sempre que uma melhor solução é encontrada na vizinhança. Este procedimento termina quando nenhuma solução melhor é encontrada.

A figura 2.4 apresenta um diagrama de atividades com a idéia de uma busca local genérica.

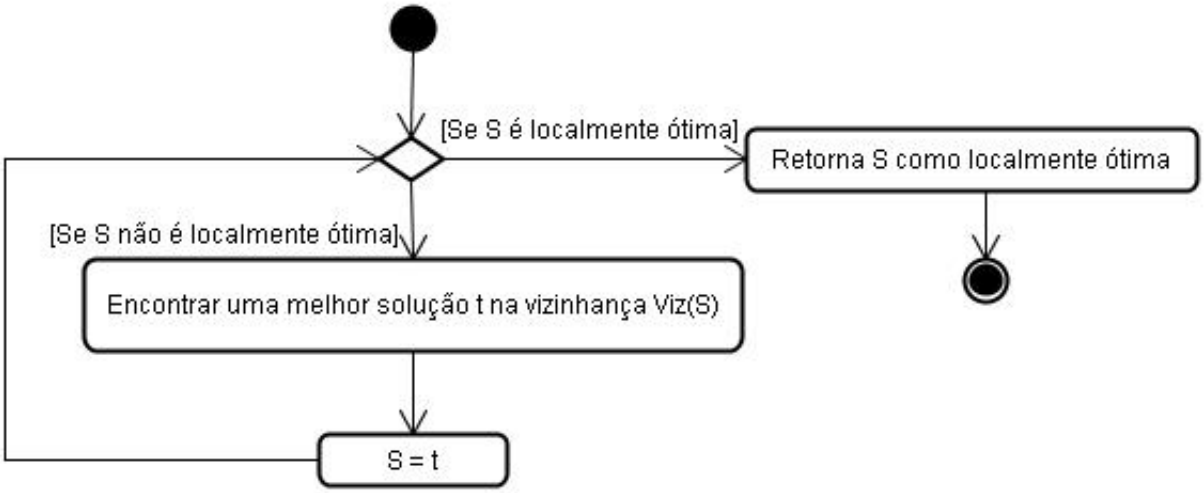

Figura 2.4: Diagrama de atividades da fase de melhoria da solução do GRASP.

O procedimento de otimização local pode exigir um tempo exponencial se a

busca partir de uma solução inicial qualquer, embora se possa constatar empiricamente a melhoria de seu desempenho de acordo com a qualidade da solução inicial. O tempo gasto pela busca local pode ser diminuído, portanto, através do uso de uma fase de construção que gere uma boa solução inicial. É claro que uma estrutura de dados eficiente e uma implementação cuidadosa são importantes (Rangel et al., 2000).

2.2 Sistema Pivô Central

Segundo (Bernardo et al., 2006), o sistema de irrigação por pivô central foi criado em 1952, no Colorado (EUA), mas teve seu uso consolidado a partir de 1961, quando começou a ser empregado com mais freqüência. Em 1973, somente nos EUA, já se irrigavam 800.000 ha por pivô central. Atualmente, seu uso já está difundido na maioria dos países, existindo mais de quatro milhões de hectares irrigados por este sistema. No Brasil, estima-se uma área de cerca de 650.000 ha irrigados por pivô central. A figuras 2.5 ilustra este sistema de irrigação, em uma vista aérea.

Figura 2.5: Visão aérea de um sistema de irrigação por pivô central.

De acordo com Keller and Bliesner (1990), o pivô central é um sistema de movimento circular, autopropelido a energia hidráulica ou elétrica. Sua composição é dada, em geral, de uma linha, conhecida como braço, ilustrada na figura 2.6, com vários aspersores, de 200 a 800 metros de comprimento, com tubos de aço de acoplamento especial, suportada por torres dotadas de rodas, nas quais operam os dispositivos de propulsão do sistema, imprimindo à linha um movimento de rotação, em torno de um ponto ou pivô, que lhe serve de ancoragem e de tomada de água (figura 2.7) por bombeamento de poços profundos ou canal, junto do pivô ou conectado a uma adutora. O sistema é dotado de recursos de ajuste de velocidade de rotação e de alinhamento das tubulações. Sua capacidade varia entre 25 e 200 ha, por unidade.

Figura 2.6: Vista parcial do braço de um sistema de pivô central com aplicação localizada de água na cultura do cafeeiro.

As distâncias entre as torres variam de 24 a 76 m, sendo mais comuns as de 30, 38, 52 e 54 m. Cada torre tem um sistema de propulsão próprio, mas existe um, central, para controle da velocidade e do alinhamento do pivô, tendo como referência a última torre. O sistema de propulsão de cada torre é elétrico, com motores de 0,5 a 1,5 cv, os quais permitem melhor controle da velocidade das torres (Keller and Bliesner, 1990).

No Brasil, são fabricados diversos modelos de pivô para atender aos mais dis-

Figura 2.7: Base do sistema de pivô central.

tintos sistemas de produção, envolvendo culturas de baixo ou alto porte, sistemas de aplicação de água com molhamento total ou limitado, sistemas fixos (mais comuns) e sistemas móveis, com raio de até 800 m e área útil de cerca de 180 ha por unidade, embora se recomenda utilizar no máximo equipamentos de até 120 ha ou menor, dependento do tipo de solos (Bernardo et al., 2006).

Segundo de Miranda and de Matos Pires (2003), dentre as vantagens do sistema pivô central destacam-se: (a) a sua trajetória circular (que permite que o equipamento termine a irrigação já posicionado para o início da próxima irrigação e faz com que os custos do equipamento sejam proporcionais ao raio irrigado enquanto a área irrigada é proporcional ao quadrado do raio), (b) a flexibilidade no manejo da irrigação, (c) o potencial para atingir elevada uniformidade de aplicação de água, (d) o reduzido, quando comparado a alguns sistemas de irrigação por aspersão, consumo de energia, (e) a redução na mão de obra necessária para efetuar a irrigação e (f) a possibilidade de automatização de todo o processo de irrigação através do acionamento remoto de uma ou mais unidades.

Dentre as desvantagens citam-se: (a) seu deslocamento circular, que limita a sua aplicação em áreas retangulares, gerando a perda de 20% da área, aproximadamente (com um raio de 400 m, irriga 50 a 54 ha de cada 64 ha), (b) faz com que a

intensidade de aplicação de água cresça no sentido da torre central para a extremidade da área irrigada, (c) por causa da alta intensidade de aplicação, na extremidade do pivô, precisa-se tomar cuidado com o escoamento superficial, sendo o manejo adequado do solo, como plantio direto, fundamental.

2.3 O estudo de caso: Projeto Paracatu-Entre Ribeiros

O Projeto Colonização Paracatu Entre-Ribeiros, localizado na Bacia do Rio São Francisco, é um dos maiores perímetros de irrigação com pivô central da América Latina. Por demandar grandes vazões dos cursos d'água da Bacia do Rio São Francisco, requer a racionalização dos seus recursos hídricos.

De acordo com o estudo desenvolvido por Pruski et al. (2007), a Região do Paracatu foi a que apresentou o maior uso da água pelo setor agropecuário, principalmente pela agricultura irrigada, o que tem gerado constantes conflitos pelo uso da água na Bacia do Rio Paracatu.

Em 2006, de acordo com relatos de lideranças locais, o problema se faz mais grave e a não realização de ações em curto prazo pode gerar a insustentabilidade do referido perímetro de irrigação, um dos mais importantes da América Latina.

O Projeto Colonização Paracatu Entre-Ribeiros está localizado às margens dos Rios Paracatu e Entre Ribeiros, no município de Paracatu no Noroeste do Estado de Minas Gerais, o projeto conta com uma área total de mais de 30.000 ha e com uma área irrigável de mais de 12.700 ha composta por mais de 80 km de canais, adutora, estações de bombeamento, elevatórias e rede de distribuição de energia elétrica. A área plantada na safra 2004 / 2005 foi de 10.886,9 ha, com uma produção estimada de 49.088,4 toneladas de grãos (soja, arroz, milho e feijão) e banana. O projeto conta com 99 produtores na área e aproximadamente 180 pivôs centrais instalados e outros 16 projetados, conforme a Figura 2.8.

O Projeto Colonização Paracatu Entre-Ribeiros faz uso da água dos Rios Paracatu e Entre Ribeiros para irrigação dos seus plantios por meio de pivôs centrais.

A água é conduzida desde o rio até o equipamento de irrigação através de canais de irrigação.

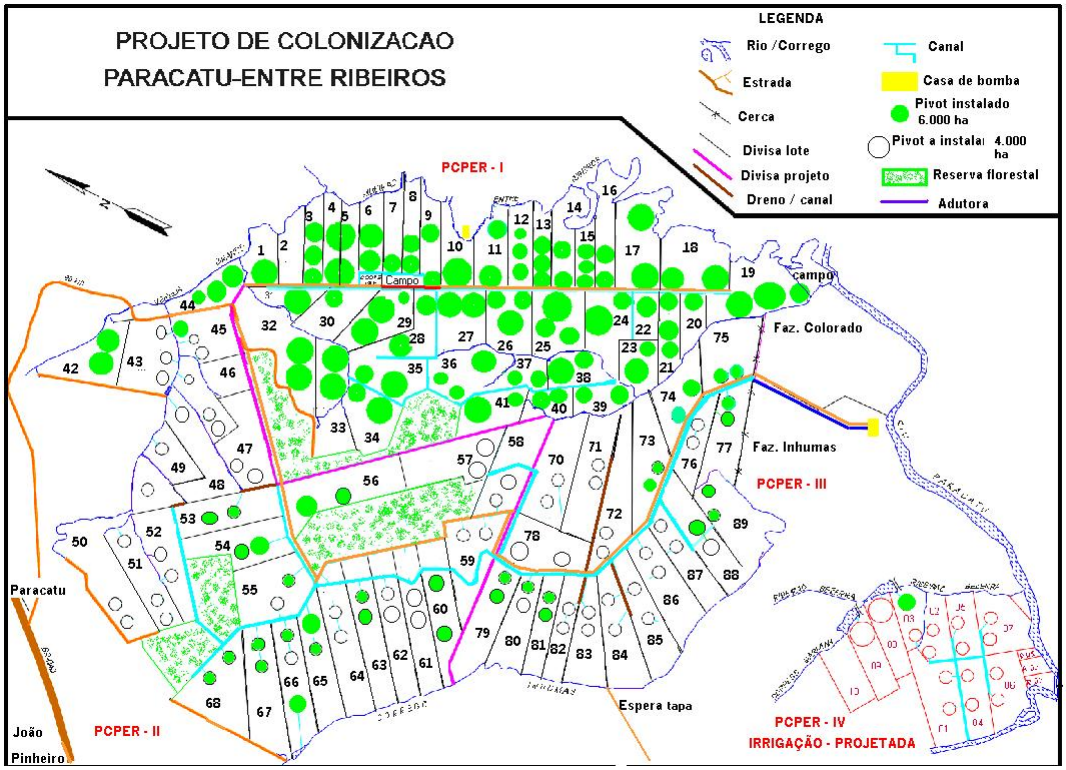

Figura 2.8: Mapa da região do projeto de Colonização Paracatu Entre-Ribeiros.

Através do balanço hídrico diário faz-se a estimativa da lâmina e o tempo de irrigação de cada pivô central do projeto. Desta forma, a cada dia tem-se um cenário de irrigação para os 180 pivôs centrais do projeto. Ligar todos os equipamentos ao mesmo tempo causa um grande impacto nos rios, podendo inviabilizar o uso da irrigação, uma vez que sistemas de irrigação que utilizam pivôs centrais demandam grandes vazões dos cursos d'água para sua operação. A alternativa ao conflito é a otimização do uso da água visando maximizar seu uso na irrigação.

Capítulo 3

Modelagem e resolução do problema

Neste capítulo serão discutidos todos os passos dados para a realização deste trabalho, desde a idealização de um modelo matemático que represente o problema apresentado até os testes feitos com base em nosso estudo de caso que faz uso de todas as idéias e produtos criados neste trabalho.

Para uma maior compreensão do problema, inicialmente foi desenvolvido um modelo matemático, discutido na seção 3.1, que foi submetido a alguns testes de validação e que, apesar de comprovar a fidelidade do modelo junto ao problema, deixou clara a necessidade de abordarmos métodos que apresentassem respostas com menor esforço computacional e em menor tempo. Com base nesta conclusão, foi adotado o uso de duas metaheurísticas, a saber, *Simulated Annealing*, ou simplesmente SA (seção 3.2), e GRASP (seção 3.3), para a busca de melhores resultados.

As duas metaheurísticas foram implementadas adotando abordagens diferentes. A SA implementada cria uma solução inicial se preocupando apenas com a viabilidade da solução, para depois otimizar a resposta, sem se prender a ótimos locais. O método GRASP implementado cria sua solução inicial tentando encontrar uma boa solução, que depois será submetida a uma busca local e que, consequentemente, apresentará uma solução ótima local.

Com ambos os algoritmos prontos, um sistema de informação, discutido na seção 3.4, foi desenvolvido com o objetivo de facilitar a manipulação dos algoritmos. Esta facilidade é focada no auxílio de inserção e retirada de informações nas estruturas

de dados dos algoritmos e na visualização das informações e das respostas emitidas pelo sistema de forma mais amigável.

Com o sistema completamente implementado, alguns testes que correspondem à região do Projeto Entre-Ribeiros/Paracatu, apresentados na seção 3.5, foram criados e executados, para que soluções para o nosso estudo de caso fossem explorados.

3.1 Modelo matemático proposto

Com o objetivo de validar as restrições encontradas no problema e, consequentemente, confirmar o domínio de todas as informações pertinentes ao mesmo, um modelo matemático foi estudado e proposto como forma de representação do problema como um todo.

Algumas pequenas instâncias do problema de escalonamento de pivôs centrais foram criadas, com dimensões bastante reduzidas, e submetidas ao modelo proposto, sendo resolvidas com o uso do software LINGO (Schrage, 1998), que utiliza Programação Inteira 0-1 (PI) e o algoritmo de Branch-and-Bound (B&B) na busca de soluções exatas. Segundo Gomes and Souza (2004), citado por Sartori et al. (2006), o LINGO é uma ferramenta simples, que utiliza o poder da programação matemática para formular problemas de grande porte com o uso de uma linguagem de modelagem, resolvendo-os e facilitando a análise da solução.

O LINGO mostrou-se eficiente para resolver problemas de pequeno porte, desde que a quantidade de água disponível para o uso não seja muito restrita, dando respostas rápidas e interessantes. Quando são testados casos com maior grau de complexidade, representada, principalmente, pela escassez de água que representa períodos de seca e/ou poucas chuvas, o LINGO não consegue apresentar soluções viáveis. Caso as dimensões do problema se aproximem da realidade, ou seja, caso exista um número de variáveis maior do que o simulado inicialmente, fica constatada a necessidade de se adotar métodos que encontrem boas soluções, exigindo menor esforço computacional e menor tempo de processamento que os conseguidos com a PI e o B&B. Porém, com um número de variáveis bastante reduzido, foram obtidos resultados bastante

satisfatórios, conseguindo assim a validação do modelo.

Neste modelo matemático, assume-se, de forma generica, a existência de n pivôs centrais que terão suas horas de trabalho distribuídas uniformemente em 24 janelas de tempo, equivalendo a 1 dia. Para representar o problema foram utilizadas as seguintes variáveis e índices:

- n: representa o número total de pivôs centrais do problema;
- i: representa cada pivô central;
- j: representa cada janela de tempo;
- C_{ij}: representa o custo de atendimento do pivô central i, na janela de tempo j.
- X_{ij}: é uma variável binária que ao assumir o valor 1 indica que o pivô central i está atuando na janela de tempo j.
- A_{ij}: é um elemento que armazena o custo de abertura de cada pivô central i, em cada janela de tempo j, de acordo com a atual escala apresentada pela matriz X_{ij}.
- D_i: representa o número de horas de água exigida pelo pivô central i.
- Q_i: representa o consumo de água do pivô central i.

Os valores de cada linha da matriz binária X_{ij} representam o escalonamento de um pivô central do problema. Desta forma, o estado de cada pivô central do problema, a cada instante do dia, é representado nesta matriz, sendo cada linha i a representação de um pivô central e cada coluna j uma janela de tempo diferente. Cada posição X_{ij} representa o estado de um pivô central i em uma janela de tempo j. Assim, indica-se que o pivô central i estará funcionando na janela de tempo j se $X_{ij} = 1$, e o pivô central i estará parado na janela de tempo j se $X_{ij} = 0$. O valor da função objetivo e o atendimento de todas as restrições do modelo dependem dos valores assumidos pelos elementos da matriz de decisão.

Os valores de cada linha do elemento A_{ij} representam as mudanças de estado de um pivô central do problema. A importância desta matriz está em tentar manter o pivô central atuando o maior número possivel de janelas de tempo continuas, já que há um gasto significativo para retirar o pivô central do estado de ócio e colocá-lo ativo. As mudanças de estado de cada pivô central do problema, a cada hora do dia, podem ser vislumbradas nesta matriz, sendo cada linha i da matriz A_{ij} a representação de um pivô central e cada coluna j da matriz A_{ij} uma janela de tempo diferente. Desta forma, cada posição A_{ij} representa o estado de um pivô central i em uma janela de tempo j, indicando que o pivô central i estará saindo do estado de ócio na janela de tempo j se $A_{ij} \neq 0$, e o pivô central i não estará iniciando suas atividades na janela de tempo j se $A_{ij} = 0$.

Como o valor de abertura de um pivô central não possui um consumo nominal, e este valor não é pratico de ser estipulado, foi considerado que a partida de um pivô central consome o equivalente a uma hora de serviço prestado. Este valor, apesar de ser acima da realidade, apresenta o efeito desejado que é de manter os pivôs centrais trabalhando o maior número de horas contíguas. Desta forma, sempre que um pivô central i sair de seu estado de ócio na janela de tempo j, A_{ij} receberá o valor C_{ij}, referente ao custo de uma hora de funcionamento do mesmo. O valor da função objetivo sofre influência direta dos valores assumidos pela matriz A_{ij}.

Cada pivô central deve ter a soma das janelas de tempo abertas igual à sua demanda horária de funcionamento, ou seja, a soma das janelas de tempo de cada pivô central deverá ser idêntica à demanda do pivô central. Assim, estas restrições garantem o total atendimento dos pivôs centrais.

Em cada janela de tempo, a soma das demandas de todos os pivôs centrais abertos deve ser menor ou igual ao volume de água disponibilizado na janela de tempo, ou seja, o total de água consumido em cada janela de tempo deverá ser igual ou inferior a um limite previamente estipulado. Esta restrição garante que o recurso hídrico não sofrerá consumo além do permitido.

A função objetivo visa minimizar os custos de funcionamento do conjunto de pivôs centrais, somando os custos de atendimento de cada um, em cada janela aberta.

Existe a tendência de haver preferência de pivôs centrais funcionarem em horários noturnos. Isto acontece devido ao fato de que o valor da energia elétrica, no período noturno, chega a ser 60% mais barata do que no período diurno.

Há também uma segunda componente na função objetivo para que visa escalonar os pivôs centrais no maior número de janelas de tempo consecutivas possível. Isto acontece devido ao custo extra atribuído à função objetivo sempre que o pivô central deixa o estado de ócio e inicia os trabalhos.

A partir dessas considerações sobre o problema de escalonamento de pivôs centrais, foi construído o seguinte modelo matemático:
Minimizar

$$\sum_{i}\sum_{j} C_{ij}X_{ij} + \sum_{i}\sum_{j} A_{ij} \tag{3.1}$$

Sujeito à:

$$\sum_{j} X_{ij} = D_i, \forall i. \tag{3.2}$$

$$\sum_{i} Q_i X_{ij} \leq vazao, \forall j. \tag{3.3}$$

$$A_{ij} = C_{ij}X_{ij}(1 - X_{ij-1}), \forall i, j \neq 1. \tag{3.4}$$

$$A_{i1} = C_{i1}X_{i1}, \forall i. \tag{3.5}$$

$$X_{ij} \in \{0, 1\}, \forall i, j. \tag{3.6}$$

$$i = 1, 2, ..., n. \tag{3.7}$$

$$j = 1, 2, ..., 24. \tag{3.8}$$

No modelo, a função objetivo fica representada pela expressão 3.1, tendo-se como objetivo a minimização do custo total gasto com energia eletrica, somado ao custo total de abertura de cada pivô central, nas janelas de tempo, respeitadas as restrições.

A restrição 3.2 garante o atendimento total das horas de funcionamento de cada pivô central, garantindo a irrigação da área associada ao pivô central i; a restrição 3.3 garante que o recurso hídrico será utilizado dentro dos limites estipulados; as restrições 3.4 e 3.5 atualizam a matriz A, definindo o custo de abertura de cada pivô i em cada janela j; a restrição 3.6 indica que a matriz é binária.

3.2 Simulated Annealing

O método SA foi escolhido para resolver o problema de escalonamento de pivôs centrais por ser um algoritmo de busca local capaz de aceitar soluções piores do que a solução atual para fugir de ótimos locais. Ele foi proposto originalmente por Kirkpatrick et al. (1983), e se fundamenta em uma analogia com a termodinâmica, ao simular o resfriamento de um conjunto de átomos aquecidos (Metropolis et al., 1953). Para a utilização do SA, foram construídos um método para geração de uma solução inicial S, um método para geração das soluções vizinhas S' (estrutura de vizinhança), e uma função objetivo $f(S)$ a ser otimizada.

O esquema de resfriamento adotado nesta implementação foi o geométrico (Biajoli et al., 2003), isto é, a temperatura T é atualizada pela formula $T \leftarrow \alpha \times T$, onde α representa a razão de resfriamento.

O pseudo-código do algoritmo implementado do SA é apresentado a seguir, onde S é a solução inicial e S' é a solução vizinha de S. A implementação utiliza, como critério de encerramento do algoritmo, um limite inferior (T_c) para a variável Temperatura (T), que sofre decremento constante durante a execução do algoritmo. Desta forma, quando T assumir um valor menor ou igual a T_c, o algoritmo é encerrado.

Algoritmo *Simulated Annealing*(T_0, T_c, α):

(T_0 é a temperatura inicial)

(T_c é a temperatura final)

(α é a razão de resfriamento)

1 $T \leftarrow T_0$;

2 $S \leftarrow$*SoluçãoInicial()*;

3 **Enquanto** $T > T_c$ **faça**

4 Gerar S' de S;

5 **Se** $f(S') < f(S)$ **então**

6 $S \leftarrow S'$;

7 **senão**

8 Gerar $r \cap U(0,1)$

9 **Se** $r < e^{\frac{-(f(S')-f(S))}{T}}$ **então**

10 $S \leftarrow S'$;

11 **fim-se**

12 **fim-se**

13 $T \leftarrow T \times \alpha$;

14 **Fim-enquanto**

15 Returne S

Fim *Simulated Annealing*

3.2.1 Gerador de soluções iniciais

O método criado para gerar uma solução inicial segue o principio de um algoritmo guloso (Cormen et al., 2002), criando uma lista de pivôs centrais ordenada em ordem decrescente de número de horas demandadas de cada pivô central, e organizando-os, desta forma, um-a-um. Assim ele sempre realiza a escolha que parece ser a melhor no momento; a escolha sempre é feita com o objetivo de encontrar uma solução viável ao término do processo, respeitando todas as restrições e sem se preocupar com valores da função objetivo ou com qualquer otimização.

Desta maneira, a solução inicial é montada de forma que, a cada pivô escalonado, o consumo de água nas janelas de tempo estarão sempre assumindo valores

próximos, evitando que uma janela de tempo atinja seu limite muito rápido e mantendo a disponibilidade de todas as janelas de tempo. Além disto, como os pivôs centrais com maior demanda são alocadaos no ínicio, quanto mais próximo do final de sua execução o método estiver, menor será a exigência dos pivôs centrais ainda não escalonados, tornando o encerramento do método pouco custoso. O pseudo-código deste método é apresentado a seguir.

Algoritmo SoluçãoInicial():

Criar uma lista L de pivôs centrais vazia;
Adicionar todos os pivôs centrais do problema na lista L;
Ordenar a lista L em ordem decrescente, pelo número de horas demandadas de irrigação;
Enquanto L não estiver vazia **faça**
    Enquanto as horas de irrigação do primeiro elemento não estão atendidas **faça**
        Em S, abra a janela de tempo com menor consumo de água até o momento;
    Fim-enquanto
    Retire o primeiro elemento da lista L;
Fim-enquanto
Returne S

Fim SoluçãoInicial

Após a construção da lista ordenada, o primeiro elemento da lista é selecionado e as suas janelas de tempo começam a ser abertas, sempre buscando abrir as janelas de tempo que possuem o menor consumo de água naquele momento. O processo de abertura das janelas é encerrado quando o número de janelas abertas é igual ao número de horas de irrigação exigidas pelo cliente. Mesmo em casos mais complexos, com um pequeno limite de vazão, a restrição 3.3 do modelo sempre é respeitada. Após o encerramento do processo de abertura das janelas de tempo do primeiro elemento, ele é excluído da lista, e todo o processo é repetido para o novo primeiro elemento da lista, até que a mesma fique vazia.

Este método construído para a obtenção de uma solução inicial, em nenhum

momento, se preocupa em gerar uma solução otimizada, tendo como único objetivo construir soluções viáveis de maneira rápida.

Desta forma, os pivôs centrais com maior número de horas de consumo, sempre serão os primeiros a serem escalonados, sendo este escalonamento feito de forma a deixar sempre o consumo de água nas janelas de tempo o mais baixo possível. Desta maneira, à medida que os pivôs centrais são excluídos da lista, pivôs centrais de menor consumo restarão na lista, e as janelas de tempo estarão sempre com os volumes de água semelhantes. Este procedimento elimina a possibilidade de encontrarmos um pivô central de alto consumo de água quando as janelas de tempo estiverem próximas de seu limite de consumo de água.

O método de construção da solução inicial demonstrou ser adequado para solucionar o atual problema, tendo gerados resultados satisfatórios, mesmo em casos críticos.

3.2.2 Regras de aceitação

Há duas regras que permitem a aceitação das soluções vizinhas encontradas pelo algoritmo. Elas são aplicadas e estruturadas seguindo as condições apresentadas a seguir em pseudo-código.

1 **Se** $f(S') < f(S)$ **então**
2 $S \leftarrow S'$; // *Solução aceita!*
3 **senão**
4 Gerar $r \in Unif[0,1]$
5 **Se** $r < e^{\frac{-(f(S')-f(S))}{T}}$ **então**
6 $S \leftarrow S'$; // *Solução aceita!*
7 **fim-se**
8 **fim-se**

A primeira regra consiste no fato que a solução vizinha S' encontrada seja uma solução melhor do que a solução atual S. Desta forma, toda solução S' encontrada e

que apresente alguma melhoria, vislumbrada através da função objetivo, em relação a S, será aceita.

A segunda regra de aceitação só é testada caso a primeira não obtenha êxito e seu funcionamento consiste no uso de um critério semi-aleatório para a aceitação da nova solução vizinha S'. Seu primeiro passo é gerar um valor r aleatório uniformente distribuído entre 0 e 1. Este valor r será comparado ao valor obtido pelo critério de Metropolis (Metropolis et al., 1953), exposto na linha 5 do pseudo-código, e caso a condição seja verdadeira, a solução S' será aceita.

A segunda regra é utilizada com a finalidade de permitir que a busca por soluções vizinhas não se prenda a ótimos locais, aceitando assim soluções consideradas piores com o intuito de permitir que novas regiões sejam analisadas. À medida que a temperatura do algoritmo diminui e que, consequentemente, o algoritmo se aproxima de seu final, a probabilidade de aceitação de soluções piores utilizada pelo critério de Metropolis também diminui, apresentando uma curva de aceitação exponencial. Isto garante ao algoritmo que, ao se aproximar do final de sua execução, soluções que prejudiquem o seu resultado final não serão aceitas.

3.2.3 Movimento e exploração da vizinhança

Foram utilizados dois tipos movimentos de troca como estruturas de vizinhança: *Modificação de Janela Simples* e *Troca Entre Pivôs Centrais*. Estes movimentos foram desenvolvidos levando-se em consideração as limitações impostas pelo problema.

É interessante destacar que estes movimentos criados não tornam, em momento algum, a solução inviável, respeitando sempre as restrições do problema, apresentadas anteriormente. Caso a solução se torne inviável, o movimento utilizado é automaticamente desfeito.

O movimento *Modificação de Janela Simples*, ilustrado na Figura 3.1, baseia-se em fazer a troca dos valores entre duas janelas de tempo que possuam valores diferentes e que pertençam a um mesmo pivô central. Desta forma, seleciona-se de forma aleatória um pivô central para sofrer a modificação e, depois, também aleatoriamente, uma janela de tempo aberta deste mesmo pivô central, para ser fechada. Assim, o

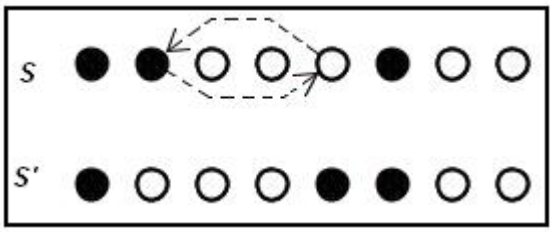

Figura 3.1: Movimento 'Modificação de Janela Simples'.

pivô central necessitará que uma janela de tempo seja aberta para que sua meta de horas seja atendida. A janela de tempo a ser aberta é preferencialmente uma janela de tempo noturna, devido ao fato de serem mais baratas, respeitando-se a restrição que limita o volume total de água a ser utilizado.

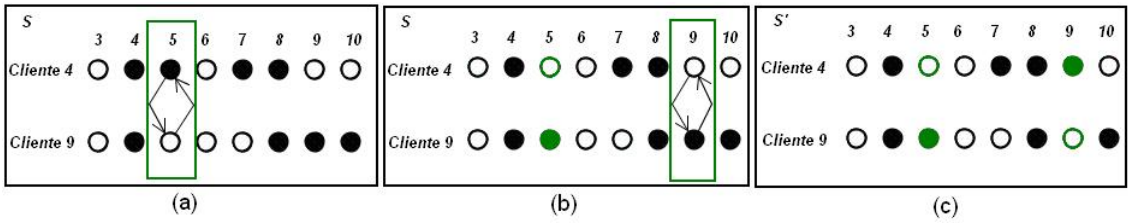

Figura 3.2: Movimento 'Troca Entre Pivôs Centrais'.

O movimento *Troca Entre Pivôs Centrais* é mais complexo do que o primeiro. Ele consiste em, inicialmente, selecionar dois pivôs centrais de maneira aleatória. Na seqüência, ele procura, aleatoriamente, uma janela de tempo onde um dos pivôs centrais selecionados apresente-se aberto e o outro fechado. Ao encontrar esta janela de tempo, substituem-se os valores nelas existentes e verifica-se a existência de algum problema em relação ao volume de água consumido (Figura 3.2a). Se esta restrição for desrespeitada, todo o processo é desfeito. Caso contrário, avança-se para a próxima etapa do movimento, visando corrigir o número de horas de irrigação de ambos os pivôs centrais, uma vez que, depois da troca, um deles ficou com uma hora a mais do que seria correto, e o outro com uma hora a menos. Chamaremos estes pivôs centrais respectivamente de $i+$ e $i-$.

O pivô central $i+$ passa a procurar entre suas janelas de tempo, uma que esteja aberta para poder fechar, sem se preocupar com as restrições, já que o fechamento

da janela de tempo estará apenas diminuindo o volume de água utilizado. Vamos chamar o índice da janela de tempo escolhida de j. Depois de acertar as horas do cliente $i+$, tentamos acertar as horas de $i-$, que precisa abrir uma de suas janelas de tempo. O próximo passo então é abrir a janela de tempo j do pivô central $i-$ (Figura 3.2b), e verificar se nenhuma restrição foi desrespeitada. Caso todas as restrições estejam sendo respeitadas, o movimento é concluído (Figura 3.2c). Caso contrário, a janela de tempo j do pivô central $i-$ será novamente fechada e a busca por uma nova janela de tempo será feita de forma aleatória, até que se encontre uma janela de tempo adequada. Se todas as janelas de tempo forem testadas e todas se mostrarem inadequadas, o movimento será totalmente cancelado.

As estruturas de vizinhança construídas foram implementadas no algoritmo do SA de maneira que cada solução vizinha é gerada por apenas um destes movimentos, sendo a sua escolha feita antes de iniciar a execução, e passada através de parâmetro.

A função objetivo $f(S)$ utilizada para avaliar as soluções é a descrita pela equação (3.1) (ver Seção 3.1), e as restrições apresentadas no modelo proposto na Seção 3.1 (equações (3.2) à (3.8)) são atendidas na função de construção de uma solução inicial e nos movimentos de troca aqui descritos.

3.3 GRASP

Segundo Rangel et al. (2000), o GRASP, ou *Greedy Randomized Adaptive Search Procedure*, é um método iterativo probabilístico, onde a cada iteração é obtida uma solução para o problema em estudo.

Para a implementação do GRASP, dois métodos que correspondem às fases do algoritmo foram construídos. O primeiro método, chamado de $ConstSolInicGulosaAleatoria()$, corresponde a fase construtiva, que cria uma solução inicial para o problema. Tal solução é construída de forma que já encontramos uma boa solução para o problema ainda na primeira fase do algoritmo. A solução criada será submetida ao segundo método do GRASP.

O segundo método, chamado de $BuscaLocal()$, corresponde à fase do algo-

ritmo, cujo objetivo é tentar obter alguma melhoria na solução corrente, encontrando, a partir de uma busca local, uma solução ótima local. A seguir o algoritmo GRASP implementado é apresentado em pseudo-código:

Procedimento GRASP(α**,** *iteracoes* **)**

(α define o tamanho da lista restrita de candidatos ou LRC)

(*iteracoes* é o número de iterações que o algoritmo irá executar)

1 $cont \leftarrow 0$;

2 **Enquanto** $cont < iteracoes$ **faça**

3 $S \leftarrow ConstSolInicGulosaAleatoria(\alpha)$;

4 $S \leftarrow BuscaLocal(S, Viz(S))$;

5 **Se** $S < MelhorS$ **então**

6 $S \leftarrow MelhorS$;

7 **fim-se**

8 $cont \leftarrow cont + 1$;

9 **Fim Enquanto**

10 Retorne($MelhorS$);

Fim GRASP

O número de iterações que o algoritmo irá executar é definido pelo argumento *iteracoes* recebido pelo procedimento e que é informado na inicialização do algoritmo. A cada iteração, uma nova solução inicial e uma nova busca local são executados.

Antes de encerrar cada iteração, o algoritmo análisa a nova solução S encontrada e a compara com a melhor solução ($MelhorS$) encontrada até o momento. A partir desta comparação, o algoritmo sempre guarda em $MelhorS$ a melhor solução obtida, baseando esta comparação em uma analise miúpe feita com base no valor obtido pela função objetivo do problema.

3.3.1 Gerador de soluções iniciais

O gerador de soluções iniciais corresponde à fase de construção do GRASP, cujo objetivo é gerar uma solução viável, contruindo-a através do escalonamento de um-a-um dos pivôs centrais.

O primeiro passo para a construção da solução é inserir todos os pivôs centrais em uma lista, onde serão organizados de acordo com o critério escolhido para aquela execução. O critério padrão adotado é o número de horas demandadas por cada pivô central em ordem decrescente.

O GRASP possui uma componente probabilistica que o diferencia de um simples algoritmo guloso. Desta forma, o primeiro pivô central a ser escalonado não será o primeiro da lista. Uma segunda lista, denominada Lista Restrita de Candidatos (LRC), recebe os α primeiros pivôs da lista principal. Com a LRC completa, um de seus pivôs é selecionado, de maneira aleatoria, para ser escalonado. Este formato de escolha permite que diferentes soluções sejam geradas a cada iteração GRASP. Sempre que um pivô central é escalonado, ele é retirado da lista principal e da LRC, passando a compor apenas a solução corrente (S) do problema. Mostramos, em pseudo-código, a fase de construção da heurística.

Procedimento ConstSolInicGulosaAleatoria(α);
(α define o tamanho da lista restrita de candidatos ou LRC)
1 $S = \{\}$;
2 **Enquanto** "S não estiver completa" **faça**
3 ConsLRC(LRC, α);
4 $pivo$ = SelecAleatElem(LRC);
5 $S = S \cup pivo$;
6 EscalonaPivo($pivo$);
7 **FimEnquanto**;
8 Retorne S;
Fim ConstSolInicGulosaAleatoria

3.3.2 Movimento e exploração da vizinhança

Após obtermos uma solução inicial S, consulta-se a estrutura de vizinhança $Viz(S)$ relativa à essa solução S. As soluções iniciais do GRASP não são necessariamente ótimos locais. Consequentemente, faz-se necessária a aplicação de um procedimento de busca local para tentar melhorar as soluções herdadas da fase construtiva.

Esta busca realiza sucessivas trocas da solução corrente, sempre que uma melhor solução é encontrada na vizinhança. Este procedimento termina quando nenhuma solução melhor é encontrada.

Acompanhando o pseudo-código a seguir, pode-se entender a idéia de uma busca local genérica.

Procedimento $BuscaLocal(S, Viz(S))$;
(S é a solução atual encontrada)
1 **Enquanto** "S nao é localmente ótima" **faça**
2 Encontrar a melhor solucao $S' \in Viz(S)$;
3 $S \leftarrow S'$;
4 **Fim enquanto**
5 Retorne(S);
Fim BuscaLocal

A vizinhança do método implementado é definida como o escalonamento de um pivô, sendo que quando uma vizinhança é selecionada estamos dizendo que um pivô central foi selecionado e sofrerá alterações em seu escalonamento. A escolha da vizinhança é feita de forma aleatória.

A otimização local aplicada consiste em escalonar a vizinhança selecionada no maior número de janelas consecutivas noturnas possível. Para conseguir êxito nesta operação, já que as janelas noturnas estarão bem próximas de seu limite máximo, outros pivôs terão seu escalonamento alterados e provavelmente serão prejudicados, pois terão que ser realocados para em janelas de tempo que não viole nenhuma restrição.

Caso o movimento de vizinhança $Viz(S)$ gere uma solução melhor que a solução S, a solução atual S assumirá os valores de $Viz(S)$. Caso contrário, S não sofrerá modificações.

3.4 O sistema gerenciador

Após a conclusão dos algoritmos, houve a necessidade de desenvolver um sistema computacional para auxiliar na manipulação das informações do problema e dos algoritmos.

Os principais objetivos deste sistema são:

- facilitar a manipulação dos dados do problema, auxiliando a inserção das informações nas estruturas de dados definidas nos algoritmos;
- oferecer opção de executar os algoritmos desenvolvidos com passagem de parâmetros; e,
- apresentar as soluções obtidas no formato de grade.

O sistema foi totalmente desenvolvido dentro do paradigma orientado a objetos (Deitel and Deitel, 2001). A programação orientada a objetos é utilizada na troca de mensagens entre as várias classes desenvolvidas, deixando encapsuladas todas as operações e dados das distintas classes e pacotes. A estrutura de dados adotada tem como principal objetivo oferecer uma fácil manipulação dos dados, para que o desempenho do algoritmo não fique comprometido por dificuldades em leitura e escrita de informações.

Basicamente, o sistema foi dividido em três pacotes denominados respectivamente como *Dados*, *Algoritmos* e *Interfaces*. A disposição das classes no sistema é apresentada na Figura 3.3.

3.4.1 Estruturas de dados

No pacote *Dados*, existem três classes que abstraem todo o problema. Estas classes compõem toda estrutura de dados do problema, ou seja, são responsáveis por

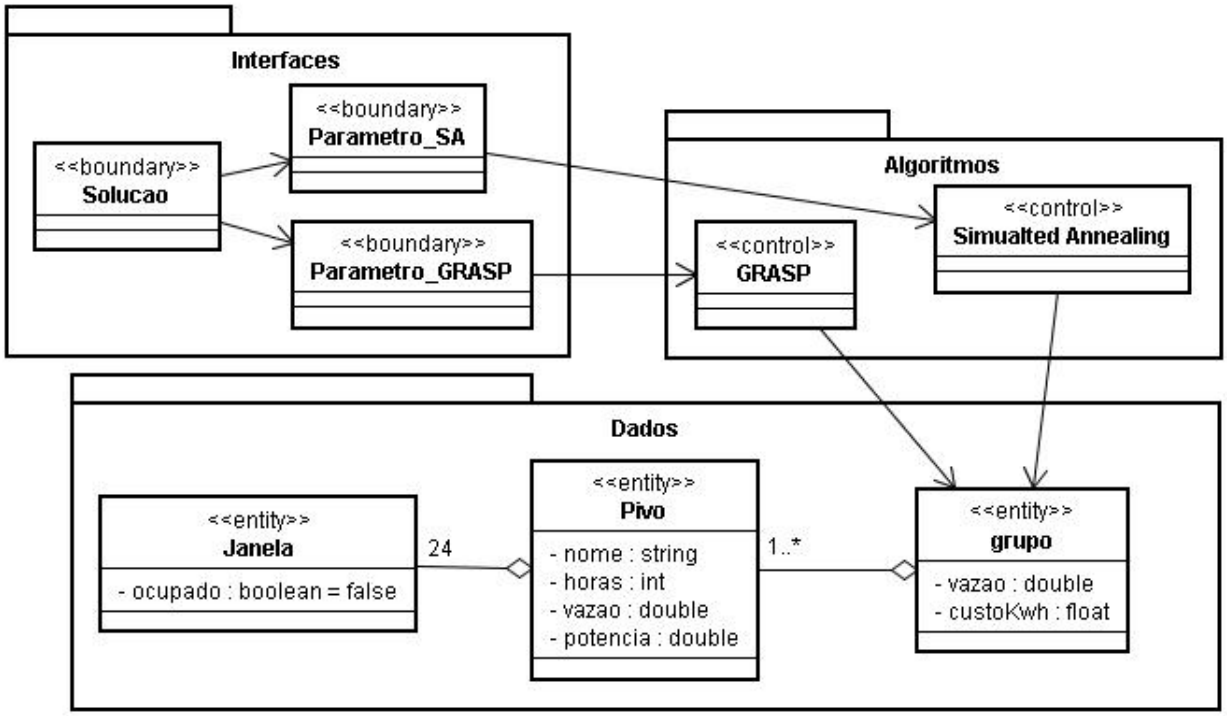

Figura 3.3: Diagrama de classes do problema.

armazenar todos os dados referentes ao problema.

Como demonstrado na Figura 3.3, possuímos uma classe, chamada *Janela*, que representa as janelas de tempo do problema, podendo ela estar no estado de ocupada ou não.

Também foi criada uma classe batizada como *Pivo*, capaz de armazenar informações relevantes aos pivôs centrais do problema, sendo cada objeto instanciado desta classe, representante de um pivô do problema. É importante destacar que, cada objeto da classe *Pivo* possui, em sua composição, 24 objetos do tipo *Janela*, de forma que cada um dos objetos representa o estado do pivô em uma das horas do dia.

A última classe deste pacote é a classe *Grupo*, composta por no mínimo um objeto da classe *Pivo* e, assim, responsável por agrupar as informações de todo o problema. As instâncias desta classe são responsáveis por representar o problema como um todo, sendo, ainda, capaz de validar os dados e as soluções encontradas, através de seus métodos implementados. Uma instância da classe *Grupo* é o principal parâmetro à ser recebido pelos algoritmos.

3.4.2 As interfaces

O sistema escalonador de pivôs centrais foi desenvolvido contendo três interfaces funcionais, tendo, uma delas, a função de apresentar todos os dados disponíveis referentes ao problema, e as outras duas, a função de informar os parâmetros e fazer a chamada dos algoritmos.

A interface inicial, apresentada na Figura 3.4, é a responsável por apresentar todas as informações do problema. Nela, as informações estão divididas em duas abas, localizadas do lado esquerdo abaixo das opções de menu, e que podem ser alternadas durante sua execução.

Na aba *Pivôs Individuais*, encontra-se do lado esquerdo uma *árvore de pivôs centrais*, que permite o acesso a todos os pivôs centrais do sistema. Todos os pivôs, inseridos nesta *árvore*, são referenciados com o nome atribuído ao mesmo durante seu cadastro. Para visualizar as informações de algum pivô central, basta colocar o foco do sistema sobre sua representação na *árvore* e, todas as informações, inclusive sua escala atual, serão apresentadas no lado direito.

Além das informações individuais de cada pivô central, é possível visualizar o escalonamento de todos os pivôs na aba *Geral*. Nela, é apresentada a solução do algoritmo, gerada em uma tabela com formatação HTML, contendo todas as janelas de tempo, de todos os pivôs centrais, indicando se o pivô central deverá ou não estar em funcionamento.

As demais interfaces são responsáveis, cada uma, por passar os parâmetros de um dos algoritmos. O fato de cada algoritmo possuir suas particularidades justifica a necessidade de uma interface específica para cada um. Além de informar os parâmetros do problema, estas interfaces também exibem os principais dados do problema durante sua execução, permitindo, assim, informar ao usuário qual o status atual do problema e alguns valores atuais e/ou parciais, como tempo de execução, número de execuções e resultado atual, por exemplo.

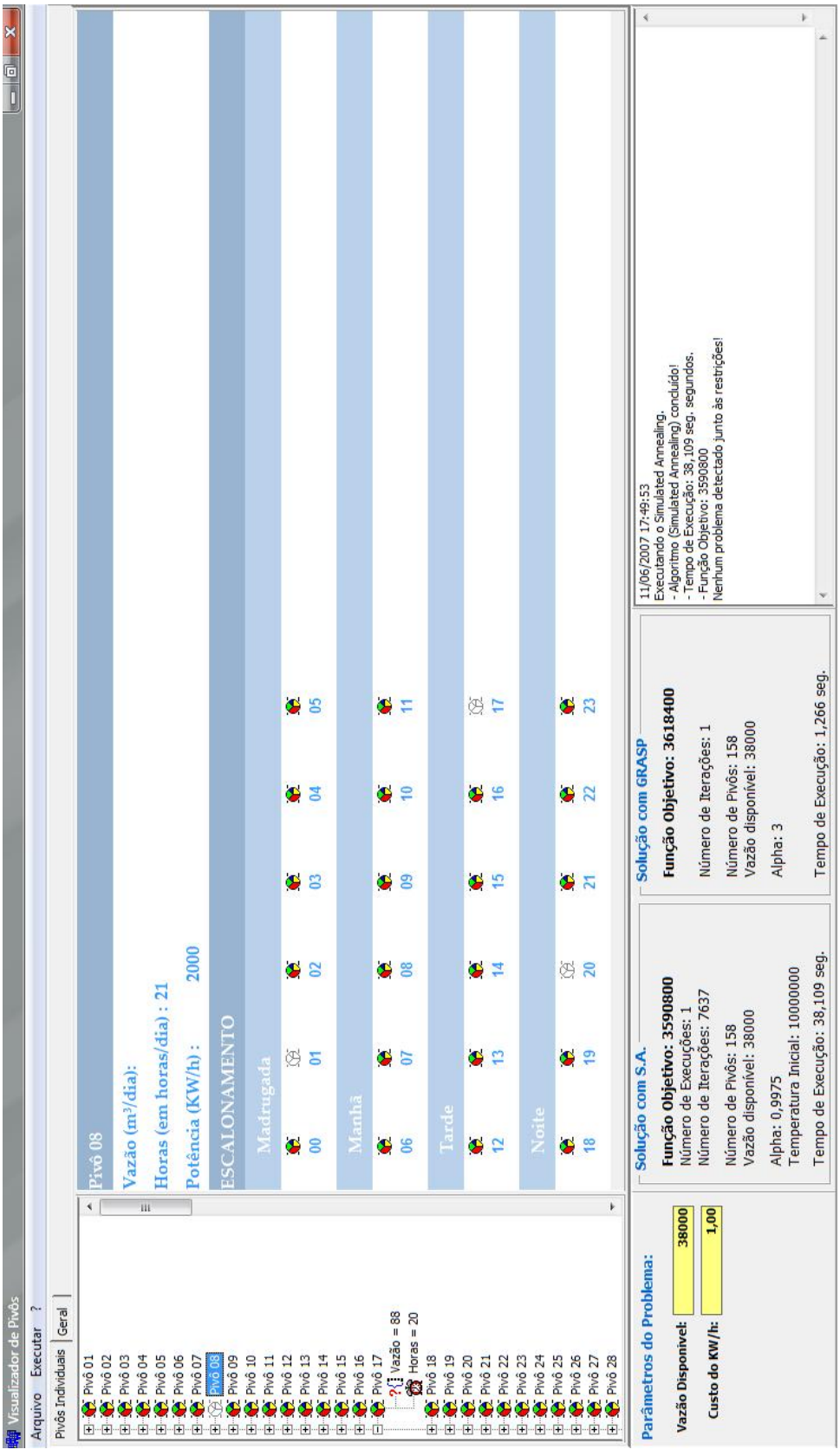

Figura 3.4: Interface principal. Apresenta dados de todos os pivôs centrais e da solução de cada método.

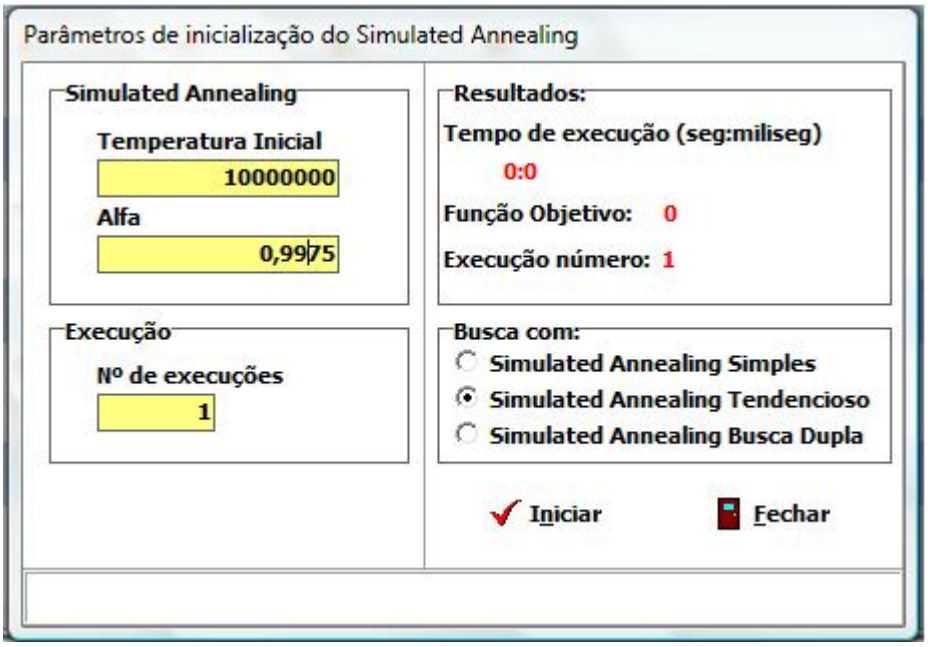

Figura 3.5: Interface de parâmetros para o método *Simulated Annealing*.

3.4.3 Aplicação dos algoritmos

No pacote *Algoritmos* existem duas classes, responsáveis pela execução das metaheurísticas, sendo cada classe responsável por um dos metodos implementados e descritos anteriormente (seções 3.2 e 3.3). Ambas as classes possuem acesso à estrutura de dados construída, através de instâncias da classe *Grupo*, contida no pacote *Dados*. Desta forma, todas as classes deste pacote podem ler e modificar o escalonamento, através dos métodos disponibilizados pelo pacote *Dados*.

Uma das classes implementadas é a *SA*, cujo nome faz referência à sigla do algoritmo que a mesma executa. Esta classe executa o método *Simulated Annealing*, sendo necessário informar, obrigatoriamente, qual o *Grupo* que será escalonado. Opcionalmente, podem ser passados alguns parâmetros de execução que influenciam no formato de busca por soluções, no tempo de execução do algoritmo e na probabilidade de aceitação de ótimos locais (Metropolis et al., 1953). Os parâmetros para o SA são informados na interface de parâmetros do *Simulated Anneling*, ilustrada na figura 3.5.

A outra classe implementada foi chamada de *GRASP*, numa referência direta ao algoritmo que a mesma executa. Assim como a *SA*, *GRASP* executa o método

proposto, sendo necessário informar, obrigatoriamente, qual o *Grupo* que será escalonado. Possui como parâmetros algumas opções que influenciam a forma de busca das soluções, o tempo de execução do algoritmo e o número de iterações a serem realizadas durante a execução. Os parâmetros para o GRASP são informados na interface de parâmetros do GRASP, ilustrada na figura 3.6.

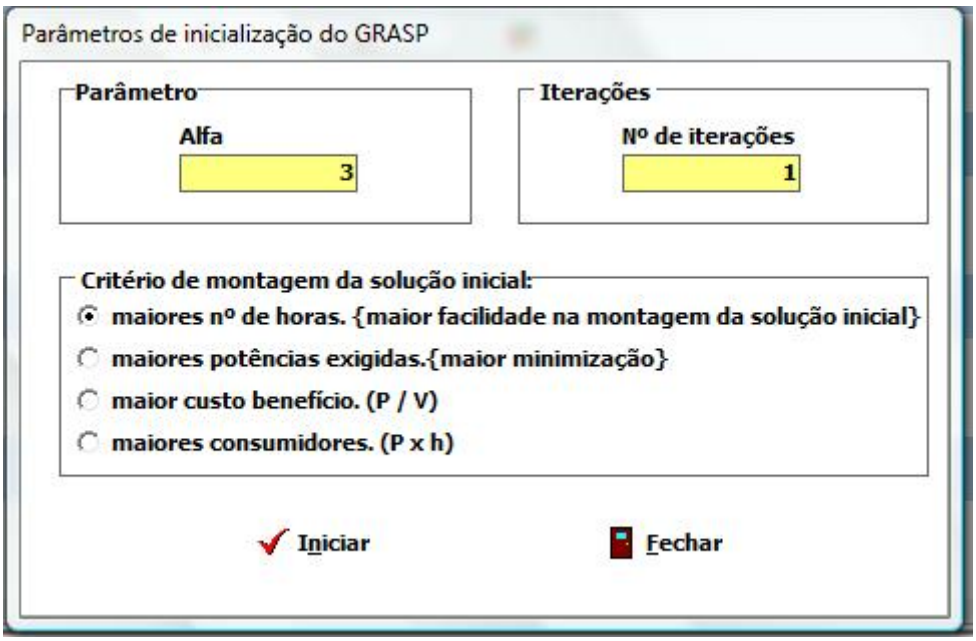

Figura 3.6: Interface de parâmetros para o método *GRASP*.

3.4.4 Entrada do sistema gerenciador

A entrada de informações no sistema é feita através de arquivos com informações em formato ASCII. Apesar de ser um arquivo de texto, a extensão padrão para arquivos deste sistema é a *.PIV*.

Dentro de um arquivo *.PIV*, cada linha armazena informações de um pivô central diferente. Todas as informações devem estar organizadas de maneira consecutiva e seguindo a sequencia pré-determinada. As informações relevantes sobre um pivô central devem estar expostas na sequencia:

- Nome do pivô central, que será o identificador deste pivô no sistema;
- Horas de irrigação, que deverá ser um valor inteiro;

- Demanda de água, que deverá ser um valor real;
- Potência consumida em KW/h, que deverá ser um valor real.

Cada item de uma linha deve ser separada pelo caracter ‘,’ (vírgula), enquanto que as casas decimais deverão ser separadas pelo caracter ‘.’ (ponto), conforme exemplo da figura 3.7.

A partir da leitura do arquivo, todas as informações são inseridas nas estruturas de dados do sistema automaticamente.

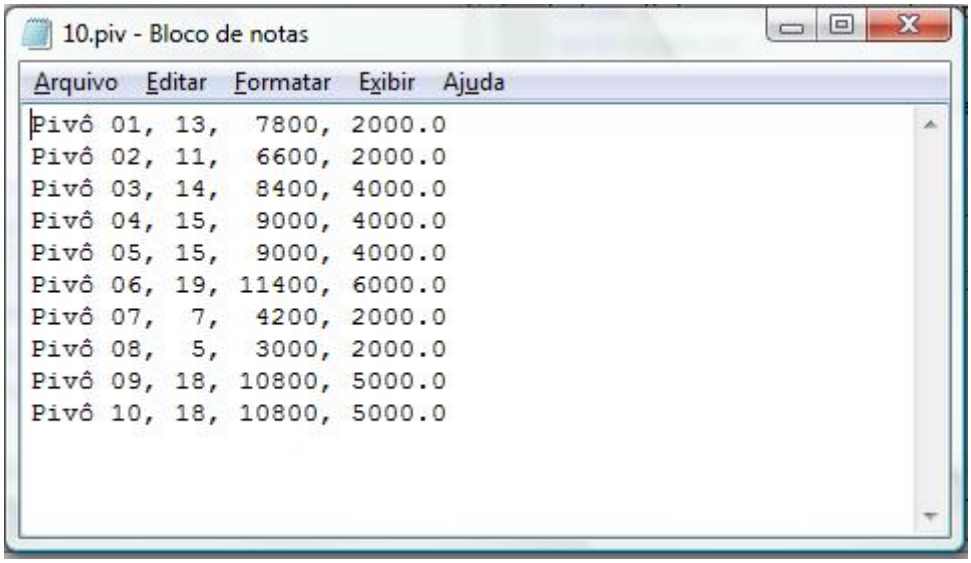

Figura 3.7: Exemplo de arquivo de entrada.

3.4.5 Saída do sistema gerenciador

Sempre que um dos algoritmos é concluído, a solução encontrada é automaticamente exportada para um arquivo com formato HTML que, imediatamente, é apresentado no sistema. Por haver um volume de informações muito alto, faz-se necessário que estes dados sejam expostos com uma formatação, visando facilitar sua análise.

Os dados estão dispostos em uma tabela, de forma que cada linha possui informações de um Pivô Central e cada coluna se refere a uma janela de tempo diferente. Onde for encontrado os caracteres ‘XXX’, quer dizer que o pivô central estará em

funcionamento na respectiva janela de tempo, onde for encontrado os caracteres '—', quer dizer que o pivô central estará ocioso. Desta forma é possível visualizar o escalonamento de todos os pivôs, simultaneamente. A Figura 3.8 apresenta um exemplo de solução para um escalonamento, envolvendo alguns dados aleatórios, com a formatação proposta.

3.5 Problemas teste

Durante a validação do modelo, pequenos grupos de pivôs foram emulados para inserção no sistema. O principal problema teste utilizado foi batizado como *'10pivos.piv'*. Nele há informações de 10 pivôs centrais reais que submetemos a diversos testes no LINGO e nos algoritmos criados. Por se tratar de um conjunto restrito de pivôs, as execuções eram rápidas e as dimensões das variáveis eram menores, facilitando a análise dos resultados. Outra vantagem é que por se tratar de uma instância pequena, os limites inferior e superior do problema são facilmente identificados. Os valores utilizados com estas instâncias são apresentados na tabela 3.1.

Tabela 3.1: Problema teste com 10 pivôs centrais.

Pivô central	**Horas de funcionamento**	**Água utilizada**	**Potência dissipada**
Pivô 01	13	7800 $m^3/hora$	2000 KW/h
Pivô 02	11	6600 $m^3/hora$	2000 KW/h
Pivô 03	14	8400 $m^3/hora$	4000 KW/h
Pivô 04	15	9000 $m^3/hora$	4000 KW/h
Pivô 05	15	9000 $m^3/hora$	4000 KW/h
Pivô 06	19	11400 $m^3/hora$	6000 KW/h
Pivô 07	7	4200 $m^3/hora$	2000 KW/h
Pivô 08	5	3000 $m^3/hora$	2000 KW/h
Pivô 09	18	10800 $m^3/hora$	5000 KW/h
Pivô 10	18	10800 $m^3/hora$	5000 KW/h

Algumas instâncias maiores foram geradas, para avaliarmos o desempenho do

	JANELAS																								
-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	TOTAL
Pivô 01	---	---	---	---	XXX	XXX	XXX	---	XXX	XXX	XXX	XXX	XXX	XXX	XXX	-	-	-	XXX	---	XXX	---	---	XXX	13
Pivô 02	XXX	XXX	XXX	XXX	XXX	---	---	---	---	---	---	---	---	---	---	-	-	-	XXX	XXX	XXX	XXX	XXX	XXX	11
Pivô 03	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	---	---	---	---	---	---	---	-	-	-	XXX	XXX	XXX	XXX	XXX	XXX	14
Pivô 04	XXX	XXX	XXX	XXX	---	XXX	XXX	XXX	XXX	XXX	---	---	---	---	---	-	-	-	XXX	XXX	XXX	XXX	XXX	XXX	15
Pivô 05	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	---	---	---	---	---	---	-	-	-	XXX	XXX	XXX	XXX	XXX	XXX	15
Pivô 06	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	-	-	-	XXX	XXX	---	XXX	XXX	---	19
Pivô 07	---	---	---	---	XXX	---	---	XXX	XXX	XXX	XXX	---	---	---	---	-	-	-	---	---	XXX	---	---	XXX	7
Pivô 08	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	-	-	-	XXX	XXX	XXX	XXX	XXX	---	5
Pivô 09	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	---	---	---	-	-	-	XXX	XXX	XXX	XXX	XXX	XXX	18
Pivô 10	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	XXX	---	---	-	-	-	---	XXX	XXX	XXX	XXX	XXX	18
TOTAL	66000	66000	66000	66000	69000	67200	67200	63600	63000	54000	45000	40800	30000	19200	19200	0	0	0	66000	69000	69600	69000	69000	66600	---

Figura 3.8: Formato da saída gerada pelo sistema. Apresenta a escala de todos os pivôs centrais durante o dia.

algoritmo. Desta forma foram geradas instâncias de testes com até 300 pivôs centrais.

Para servir como a principal instância de testes, foram levantadas as informações pertinentes ao sistema de todos os 180 pivôs centrais do Projeto Paracatu-Entre Ribeiros. Todas estas informações foram transformadas em um arquivo do tipo *'.PIV'* para futura submissão aos algoritmos gerados. As informações sobre este teste pode ser encontrada no Apêndice A.

Capítulo 4

Resultados e Discussão

4.1 Validação do Modelo

Durante a construção do modelo, surgiu a necessidade de comprovar se o modelo proposto representava realmente o problema estudado. A forma adotada para a validação foi aplicar o modelo, escrevendo-o de forma algébrica, em uma ferramenta computacional, que tenha uma linguagem de modelagem e que possua capacidade de solucionar problemas de natureza combinatória, utilizando os dados passados para ele.

A ferramenta escolhida para fazer a validação foi o software LINGO, devido a sua facilidade de uso e por atender as necessidades encontradas. Porém o LINGO possui algumas limitações baseadas nos métodos de resolução que são utilizados. Desta forma, caso o problema possua restrições muito apertadas e/ou um número de variáveis muito amplo, o LINGO pode ter dificuldades em apresentar sua solução em tempo viável, exigindo um alto esforço computacional para apresentar soluções de ótimos locais.

Visando tornar a validação do modelo e dos algoritmo construídos viável, testes de pequeno porte, com dados simulados de poucos pivôs centrais, foram executados em algumas instâncias, onde cada instância significava um diferente valor de vazão de água disponível aos pivôs centrais.

A principal instância dos testes de validação possui 10 pivôs centrais distintos, apresentados na tabela 3.1, que compuseram três diferentes instâncias, representando

diferentes níveis de dificuldade para a construção da solução do problema. A primeira instância, com uma vazão de $2.916,67m^3$ de água por hora, foi montada visando apresentar uma folga de água disponivel, permitindo que a escala de funcionamento possa ser construída com relativa facilidade. A segunda instância, com $2.500m^3$ de água por hora, possui o recurso um pouco mais escasso, mais ainda não compromete a solução. Já a terceira instância, com $2.291,67m^3$ de água por hora, apesar de possuir uma quantidade de água viável, exige um esforço muito alto do sistema e, apesar de executar em um computador por 5 dias consecutivos, não apresentou uma solução definitiva. É importante salientar que o LINGO faz a busca da solução ótima global, podendo encontrar apenas o ótimo local em alguns casos.

A validação foi feita em duas etapas visando facilitar a correção de possiveis falhas encontradas no modelo. Na primeira, o fator 'Custo de Abertura' dos pivôs centrais não foi levado em consideração, utilizando o modelo apenas parcialmente. Já na segunda, etapa o 'Custo de Abertura' foi levado em consideração, sendo utilizado o modelo na integra.

Como o 'Custo de Abertura' foi ignorado, a escala foi construída com certa facilidade, apresentando resultados viavéis em um espaço de tempo aceitável, conforme apresentado na tabela 4.1 e criou os escalonamentos apresentados nas tabelas 4.2, 4.3 e 4.4, para as instâncias de $2.916,67m^3$; $2.500m^3$ e $2.291,67m^3$ de água por hora, respectivamente.

Tabela 4.1: Resultados dos testes de validação do modelo, utilizando o software LINGO, com 10 pivôs centrais, ignorando o sequenciamento de janelas de tempo.

Água (m^3/hora)	LINGO		
	Tempo de processamento (Min:Seg)	Iterações	Função Objetivo (u.m.).
2.291,67	10:10	6.109.914	354.080
2.500	5:12	3.234.166	338.275
2.916,67	6:13	4.222.602	315.374

Tabela 4.2: Escalonamento construído pelo LINGO com 10 pivôs centrais, com $2.916,67m^3$ de água, ignorando o sequenciamento de janelas de tempo.

Janelas de Tempo	**Pivôs Centrais**										**Vazão Utilizada (m^3/hora)**
	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	
0 à 1 hora	-	X	X	X	X	X	-	X	X	X	**2.875**
1 às 2 horas	X	-	X	X	X	X	-	-	X	X	**2.800**
2 às 3 horas	-	X	X	X	X	X	-	X	X	X	**2.875**
3 às 4 horas	X	-	X	X	X	X	-	-	X	X	**2.800**
4 às 5 horas	X	-	X	X	X	X	-	-	X	X	**2.800**
5 às 6 horas	X	-	X	X	X	X	-	-	X	X	**2.800**
6 às 7 horas	-	X	-	-	-	X	X	-	-	-	**925**
7 às 8 horas	-	X	-	X	-	X	-	-	X	X	**2.025**
8 às 9 horas	X	-	-	X	X	X	-	-	-	-	**1.550**
9 às 10 horas	-	-	-	-	-	-	X	-	-	-	**175**
10 às 11 horas	-	X	X	-	-	-	-	-	X	X	**1.525**
11 às 12 horas	X	X	-	-	-	X	-	-	-	-	**1.075**
12 às 13 horas	X	-	-	-	-	-	X	-	-	-	**500**
13 às 14 horas	X	-	-	X	X	X	-	-	X	X	**2.450**
14 às 15 horas	X	-	-	-	-	X	X	-	X	X	**1.875**
15 às 16 horas	X	X	X	-	-	X	-	-	X	X	**2.325**
16 às 17 horas	X	-	-	-	X	-	X	-	X	X	**1.775**
17 às 18 horas	-	X	-	-	-	-	X	-	-	-	**450**
18 às 19 horas	X	-	X	X	X	X	-	-	X	X	**2.800**
19 às 20 horas	-	X	X	X	X	X	-	X	X	X	**2.875**
20 às 21 horas	-	X	X	X	X	X	-	X	X	X	**2.875**
21 às 22 horas	-	X	X	X	X	X	-	X	X	X	**2.875**
22 às 23 horas	-	-	X	X	X	X	X	-	X	X	**2.650**
23 às 00 horas	X	-	X	X	X	X	-	-	X	X	**2.800**
Total de Horas	**13**	**11**	**14**	**15**	**15**	**19**	**7**	**5**	**18**	**18**	

Tabela 4.3: Escalonamento construído pelo LINGO com 10 pivôs centrais, com $2.500m^3$ de água, ignorando o sequenciamento de janelas de tempo.

Janelas de Tempo	**Pivôs Centrais**										**Vazão Utilizada (m^3/hora)**
	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	
0 à 1 hora	-	X	X	X	-	X	-	X	X	X	**2.500**
1 às 2 horas	-	X	X	X	-	X	-	X	X	X	**2.500**
2 às 3 horas	-	X	X	-	X	X	-	X	X	X	**2.500**
3 às 4 horas	-	X	X	X	X	-	X	-	X	X	**2.450**
4 às 5 horas	-	-	X	X	X	X	-	-	X	X	**2.475**
5 às 6 horas	-	-	X	X	X	X	-	-	X	X	**2.475**
6 às 7 horas	X	-	-	-	-	X	-	-	X	-	**1.250**
7 às 8 horas	X	X	-	-	-	X	X	-	-	-	**1.250**
8 às 9 horas	X	X	X	-	X	-	X	-	X	-	**1.950**
9 às 10 horas	X	-	-	-	X	X	X	-	-	-	**1.350**
10 às 11 horas	X	-	X	X	X	X	-	-	-	X	**2.350**
11 às 12 horas	X	X	-	-	-	-	X	-	-	-	**775**
12 às 13 horas	X	X	-	-	-	-	X	X	-	-	**900**
13 às 14 horas	X	X	-	X	-	X	-	-	X	X	**2.350**
14 às 15 horas	X	X	-	-	X	X	-	-	X	X	**2.350**
15 às 16 horas	X	-	-	-	X	-	X	-	-	X	**1.325**
16 às 17 horas	X	-	-	X	X	X	-	-	X	X	**2.450**
17 às 18 horas	X	-	-	X	X	X	-	-	X	X	**2.450**
18 às 19 horas	-	-	X	X	X	X	-	-	X	X	**2.475**
19 às 20 horas	-	X	X	X	-	X	-	X	X	X	**2.500**
20 às 21 horas	-	-	X	X	X	X	-	-	X	X	**2.475**
21 às 22 horas	-	-	X	X	X	X	-	-	X	X	**2.475**
22 às 23 horas	X	-	X	X	-	X	-	-	X	X	**2425**
23 às 00 horas	-	-	X	X	X	X	-	-	X	X	**2.475**
Total de Horas	**13**	**11**	**14**	**15**	**15**	**19**	**7**	**5**	**18**	**18**	

Tabela 4.4: Escalonamento construído pelo LINGO com 10 pivôs centrais, com $2.291,67m^3$ de água, ignorando o sequenciamento de janelas de tempo.

Janelas de Tempo	**Pivôs Centrais**										**Vazão Utilizada (m^3/hora)**
	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	
0 à 1 hora	-	-	X	-	X	X	X	-	X	X	**2.275**
1 às 2 horas	-	-	X	-	X	X	X	-	X	X	**2.275**
2 às 3 horas	-	-	X	-	X	X	X	-	X	X	**2.275**
3 às 4 horas	-	-	X	-	X	X	-	X	X	X	**2.225**
4 às 5 horas	-	-	X	X	-	X	X	-	X	X	**2.275**
5 às 6 horas	-	-	X	X	-	X	X	-	X	X	**2.275**
6 às 7 horas	X	X	-	X	X	X	-	-	-	X	**2.275**
7 às 8 horas	X	-	-	-	X	X	-	-	X	X	**2.075**
8 às 9 horas	X	X	-	X	X	X	-	-	X	-	**2.275**
9 às 10 horas	X	X	-	X	X	-	-	-	X	X	**2.250**
10 às 11 horas	X	X	-	X	-	-	-	-	-	-	**975**
11 às 12 horas	X	-	-	-	X	X	-	-	X	X	**2.075**
12 às 13 horas	X	X	-	-	-	X	-	-	-	-	**1.075**
13 às 14 horas	X	X	-	X	X	X	-	-	X	-	**2.275**
14 às 15 horas	X	X	-	X	X	X	-	-	-	X	**2.275**
15 às 16 horas	X	X	X	-	X	X	-	-	-	-	**1.800**
16 às 17 horas	X	-	-	X	-	X	-	-	X	X	**2.075**
17 às 18 horas	X	X	X	X	X	-	-	-	-	X	**2.150**
18 às 19 horas	-	-	X	X	-	X	-	X	X	X	**2.225**
19 às 20 horas	-	-	X	X	-	X	-	X	X	X	**2.225**
20 às 21 horas	-	X	X	X	-	-	X	X	X	X	**2.200**
21 às 22 horas	-	-	X	X	-	X	X	-	X	X	**2.275**
22 às 23 horas	-	-	X	X	X	X	-	X	X	-	**2.150**
23 às 00 horas	X	X	X	-	X	-	-	-	X	X	**2.225**
Total de Horas	**13**	**11**	**14**	**15**	**15**	**19**	**7**	**5**	**18**	**18**	

Tabela 4.5: Resultados dos testes de validação do modelo com 10 pivôs centrais (‘*10pivos.piv*’).

Água (m^3/hora)	LINGO		
	Tempo (Hr:Min)	Iterações	F.O. (u.m.)
2.291,67	$>$ 72:00	$>$ 2.496.816	-
2.500	7:48	199.681	392.400
2.916,67	2:15	62.021	394.200

Com este teste, fica claro o objetivo do sistema em escalonar os pivôs centrais dando prioridade para as janelas de tempo dos horários noturnos, graças ao custo destas janelas serem mais baixos que o custo das demais janelas de tempo.

Na segunda etapa, quando o problema é avaliado em sua totalidade, o LINGO teve dificuldades em apresentar os valores, conforme apresentado na tabela 4.5, não conseguindo encontrar uma boa solução para a instância de $2.291,67m^3$ de água por hora, apesar de mais de 72 horas de processamento. As tabelas 4.6 e 4.7 apresentam as escalas de funcionamento montadas para as instâncias de $2.916,67m^3$ e $2.500m^3$ de água por hora, respectivamente.

Neste teste foi adicionado um novo objetivo. Desta forma o sistema, além de buscar o menor custo ao escalonar os pivôs centrais, priorizando as janelas de tempo dos horários noturnos, também manteve os pivôs centrais escalonados o maior número de janelas de tempo sequenciais possivel. Ambos os objetivos compõem a função objetivo do problema.

O principal item analisado nestes testes foi que nenhuma das restrições encontradas no problema foi desrespeitada, entendendo-se que as mesmas estão consistentes. A função objetivo ficou prejudicada devido ao fato de que o LINGO não conseguiu

Tabela 4.6: Escalonamento construído pelo LINGO com 10 pivôs centrais, com $2.916,67 m^3$ de água, utilizando todo o modelo.

Janelas de Tempo	**Pivôs Centrais**										**Vazão Utilizada** (m^3/**hora**)
	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	
0 à 1 hora	X	X	X	X	X	X	X	-	X	-	**2.800**
1 às 2 horas	X	X	X	X	X	X	X	-	X	-	**2.800**
2 às 3 horas	X	X	X	X	X	X	X	-	X	-	**2.800**
3 às 4 horas	X	X	X	X	X	X	X	-	X	-	**2.800**
4 às 5 horas	-	-	X	X	-	-	X	X	X	-	**1.475**
5 às 6 horas	-	-	X	X	X	-	X	X	X	X	**2.300**
6 às 7 horas	-	-	-	X	X	-	X	X	X	X	**1.950**
7 às 8 horas	-	-	-	X	X	-	-	X	X	X	**1.775**
8 às 9 horas	-	-	-	-	-	-	-	X	X	X	**1.025**
9 às 10 horas	-	X	-	-	-	X	-	-	X	X	**1.650**
10 às 11 horas	X	X	-	-	-	X	-	-	X	X	**1.975**
11 às 12 horas	X	X	-	-	-	X	-	-	-	X	**1.525**
12 às 13 horas	-	X	-	-	-	X	-	-	-	X	**1.200**
13 às 14 horas	-	X	-	-	-	X	-	-	-	X	**1.200**
14 às 15 horas	-	X	-	-	-	X	-	-	-	X	**1.200**
15 às 16 horas	-	X	-	-	-	X	-	-	-	X	**1.200**
16 às 17 horas	-	-	X	-	X	X	-	-	-	X	**1.650**
17 às 18 horas	X	-	X	X	X	X	-	-	X	X	**2.800**
18 às 19 horas	X	-	X	X	X	X	-	-	X	X	**2.800**
19 às 20 horas	X	-	X	X	X	X	-	-	X	X	**2.800**
20 às 21 horas	X	-	X	X	X	X	-	-	X	X	**2.800**
21 às 22 horas	X	-	X	X	X	X	-	-	X	X	**2.800**
22 às 23 horas	X	-	X	X	X	X	-	-	X	X	**2.800**
23 às 00 horas	X	-	X	X	X	X	-	-	X	-	**2.350**
Total de Horas	**13**	**11**	**14**	**15**	**15**	**19**	**7**	**5**	**18**	**18**	

Tabela 4.7: Escalonamento construído pelo LINGO com 10 pivôs centrais, com $2.500m^3$ de água, utilizando todo o modelo.

Janelas de Tempo	**Pivôs Centrais**										**Vazão Utilizada** (m^3/**hora**)
	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	
0 à 1 hora	X	X	-	X	X	X	X	-	-	X	**2.450**
1 às 2 horas	X	X	-	X	X	X	X	-	-	X	**2.450**
2 às 3 horas	X	X	-	X	X	X	X	-	-	X	**2.450**
3 às 4 horas	X	X	-	X	X	X	X	-	-	-	**2.000**
4 às 5 horas	-	X	-	X	X	X	X	-	X	-	**2.125**
5 às 6 horas	-	X	X	X	X	X	X	-	X	-	**2.475**
6 às 7 horas	X	-	X	X	-	X	X	-	X	-	**2.150**
7 às 8 horas	X	-	X	X	-	X	X	-	X	-	**1.975**
8 às 9 horas	X	-	X	X	-	X	X	-	X	-	**1.975**
9 às 10 horas	X	-	X	X	-	X	X	-	X	X	**2.425**
10 às 11 horas	X	-	X	X	-	X	X	-	X	X	**2.425**
11 às 12 horas	X	-	X	-	-	X	-	-	X	X	**2.050**
12 às 13 horas	X	-	X	-	-	X	-	-	X	X	**2.050**
13 às 14 horas	X	-	X	-	-	X	-	-	X	X	**2.050**
14 às 15 horas	X	-	-	-	-	-	-	-	X	X	**1.225**
15 às 16 horas	-	-	-	-	X	-	-	-	X	X	**1.275**
16 às 17 horas	-	-	-	-	X	-	-	-	X	X	**1.275**
17 às 18 horas	-	-	-	X	X	-	-	-	X	X	**1.650**
18 às 19 horas	-	-	-	X	X	-	-	X	X	X	**1.775**
19 às 20 horas	-	X	X	-	X	X	-	X	X	X	**2.500**
20 às 21 horas	-	X	X	-	X	X	-	X	X	X	**2.500**
21 às 22 horas	-	X	X	-	X	X	-	X	X	X	**2.500**
22 às 23 horas	-	X	X	X	X	X	-	X	-	X	**2.425**
23 às 00 horas	-	X	X	X	X	X	-	-	-	X	**2.300**
Total de Horas	**13**	**11**	**14**	**15**	**15**	**19**	**7**	**5**	**18**	**18**	

encontrar uma solução ótima global em nenhuma das instâncias. Mesmo com soluções que não encontraram o ótimo global, é visivel que a solução encontrada buscava minimizar a função objetivo.

Todos os resultados encontrados foram apresentados para profissionais que lidam com o problema abordado e as soluções foram validadas pelos mesmos, e os resultados foram considerados consistentes.

4.2 Validação das metaheurísticas implementadas

Para validar as metaheurísticas implementadas com base no modelo, foi utilizado o mesmo teste aplicado na validação do modelo, sendo encontradas soluções muito melhores do que as apresentadas pelo LINGO. As melhorias apresentadas dizem respeito ao tempo de processamento necessário para apresentar uma boa solução para o problema, se comparado aos resultados apresentados pelo LINGO.

Os resultados encontrados estão apresentados na tabela 4.8, para o algoritmo do *Simulated Annealing* e GRASP.

Tabela 4.8: Resultados obtidos com os algoritmos construídos utilizando 10 pivôs centrais.

	SA			GRASP		
Água (m^3/hora)	Tempo (seg.)	Iterações	F.O. (u.m.)	Tempo (seg.)	Iterações	F.O. (u.m.)
2.291,67	0,422	756	414.400	0,78	25	433.400
2.500	0,422	756	403.800	0,78	25	406.000
2.916,67	0,422	756	359.200	0,62	25	347.600

A figura 4.1 apresenta o gráfico comparativo entre os tempos gastos e a figura 4.2 aprenta o gráfico comparativo entre as funções objetivo encontradas. Ambos gráficos fazem referência aos resultados encontrados pelo LINGO, *Simulated Annealing* e GRASP, e as para as 3 instâncias do problema com 10 pivôs centrais.

É visível no gráfico apresentado na figura 4.2 uma característica encontrada em

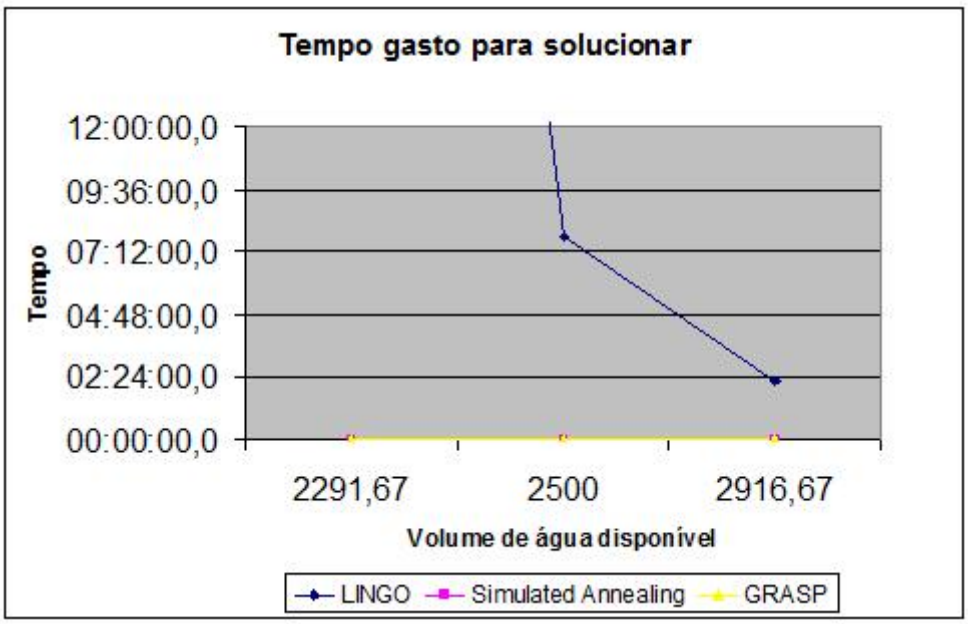

Figura 4.1: Gráfico comparativo entre os tempos de processamento dos métodos de solução, com 3 instâncias do problema com 10 pivôs centrais.

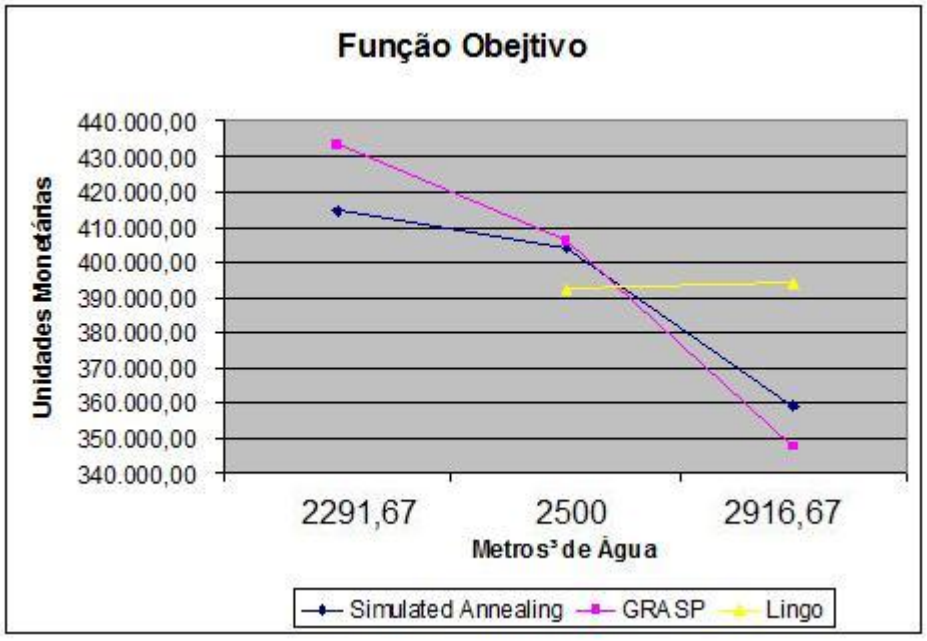

Figura 4.2: Gráfico comparativo entre as funções objetivo encontradas pelos métodos de solução, com 3 instâncias do problema com 10 pivôs centrais.

todos os testes. O algoritmo do GRASP implementado consegue encontrar solução mais baratas que o *Simulated Annealing* quando não possui um volume de água disponível restrito. Em compensação, a medida que a disponibilidade de água diminui, a *Simulated Annealing* consegue encontrar soluções cada vez melhores, em relação às encontradas pelo GRASP. Em alguns casos extremos, o *Simulated Annealing* implementado consegue encontrar soluções viáveis com volumes de água extremamente baixo, enquanto o GRASP não consegue sequer encontrar uma solução viável. Isso é perfeitamente explicado devido à forma que cada algoritmos busca sua solução inicial, conforme já apresentado nas seções 3.2.1 e 3.3.1. Esta dificuldade apresentada pelo GRASP é fruto da forma como ele foi implementado, visando otimizar o resultado, sem se preocupar tanto com as restrições quanto o *Simulated Annealing*. Esta característica negativa da implementação do GRASP é compensada pela maior qualidade dos resultados em relação ao *Simulated Annealing*, quando à uma folga maior. É importante destacar que esta característica é da implementação criada, e não do algoritmo conceitual.

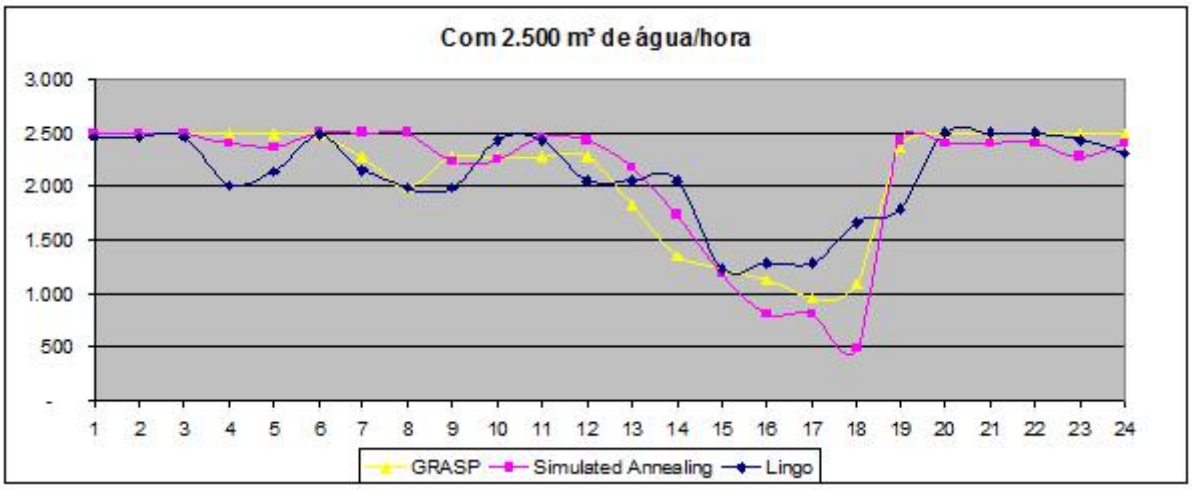

Figura 4.3: Gráfico com o volume de água consumido pelos pivôs centrais a cada janela de tempo, com a vazão disponivel de 2500 m^3 de água/hora.

As figuras 4.3 e 4.4 apresentam os volumes de água utilizados pelos pivôs centrais, a cada janela de tempo. Pode-se observar que em nenhuma janela de tempo o volume de água máximo estipulado é ultrapassado, conforme estipulado nas restrições

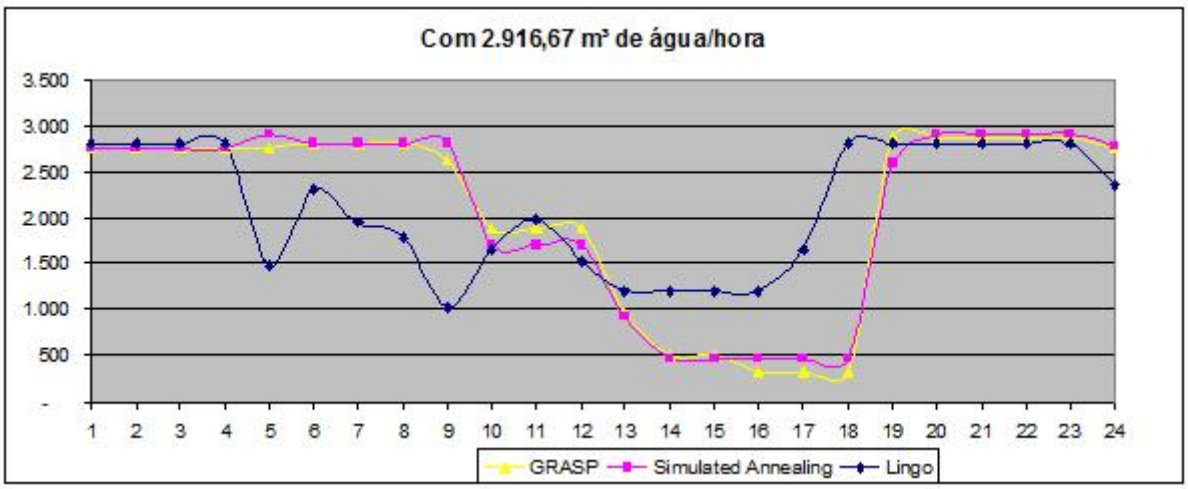

Figura 4.4: Gráfico com o volume de água consumido pelos pivôs centrais a cada janela de tempo, com a vazão disponivel de 2916,67 m^3 de água/hora.

do problema.

Todos os resultados encontrados foram apresentados para profissionais que lidam com o problema abordado e as soluções foram validadas pelos mesmos, sendo consideradas soluções muito boas. A principal característica elogiada foi o tempo total de processamento gasto para montar as soluções.

4.3 Testes de Desempenho

Os testes apresentados até o momento apresentaram soluções bastante rápidas, mas as instâncias usadas possuem um número de variáveis relativamente baixo. Para conhecermos o real desempenho dos algoritmos e da estrutura de dados a eles associados foram criados, de forma simulada, dois novos grupos de pivôs centrais com 79 e 300 pivôs centrais, respectivamente, com características de pivôs reais. As informaçoes deste dois grupos podem ser encontradas respectivamente nos apêndices B e C.

Ambos os grupos de pivôs centrais foram submetidos a duas diferentes vazões de água disponivel, compondo assim 4 instâncias teste de problemas, sendo uma vazão com folga e um pior caso aceito para cada grupo.

Nos testes efetuatuados nas 4 instâncias o *Simulated Annealing*, apresentados

Tabela 4.9: Teste de Agilidade do *Simulated Annealing*.

Nº de Pivos Centrais	Volume de Água	Tempo de Processamento	Função Objetivo
79	18.000 m^3/hora	39,656 seg.	2.048.000 u.m.
79	30.000 m^3/hora	36,890 seg.	1.850.000 u.m.
300	65.000 m^3/hora	123,828 seg.	7.368.000 u.m.
300	80.000 m^3/hora	121,750 seg.	7.172.200 u.m.

Tabela 4.10: Teste de Agilidade do GRASP.

Nº de Pivos Centrais	Volume de Água	Tempo de Processamento	Função Objetivo
79	20.000 m^3/hora	10,063 seg.	1.878.400 u.m.
79	30.000 m^3/hora	5,563 seg.	1.850.000 u.m.
300	70.000 m^3/hora	179,172 seg.	7.207.600 u.m.
300	80.000 m^3/hora	171,532 seg.	6.973.200 u.m.

na tabela 4.9 , em todas as instâncias foram executadas 7637 iterações e nos testes efetuados com as mesmas instâncias com o GRASP, apresentados na tabela 4.10 , foram executadas 25 iterações.

Nestes teste pode ser observado uma vantagem do algoritmo *Simulated Annealing* implementado sobre o GRASP. O *Simulated Annealing* consegue encontrar boas soluções mesmo com o volume de água muito baixo, como no teste de 79 pivôs centrais com apenas 18.000 m^3/hora de água. O GRASP não conseguiu encontrar solução para este problema, sendo necessário dar uma quantidade de água um pouco maior para o mesmo. Isto aconteceu devido à forma que o algoritmo foi montado. O mesmo quadro se repete para a instância de 300 pivôs centrais, conforme apresentado nas tabelas 4.9 e 4.10.

É importante ressaltar que estes teste foram extremos e os valores apresentados, ou equivalentes a eles, não refletem a realidade, já que para um pivô central ser regulamentado, a região precisa ter uma boa quantidade de água disponivel para a irrigação, e os valores utilizados não condizem com isto. A idéia deste teste é de submeter os algortimos a situações extremas e podermos analisar o comportamento obtido.

Os resultados encontrados são satisfatórios, pois se tratam de boas soluções viáveis encontradas em tempo viável, levando-se em consideração a dimensão dos problemas resolvidos. Devido ao fato de não conhecermos os valores ótimos destes problemas, não foi feito nenhum estudo comparativo. Porém, de acordo com a opinião de especialistas da área, os resultados foram muito bons, suprindo a necessidade atual de resultados bons e rápidos.

4.4 Resultados do Estudo de Caso

Com o modelo e os algortimos testados e validados, alguns testes com dados reais foram executados. Os dados utilizados pertencem ao projeto *Colonização Paracatu Entre-Ribeiros*, que atua na Bacia do Rio São Francisco e corresponde a um dos maiores perímetros de irrigação com pivô central da América Latina. Atualmente, o

projeto conta com 180 pivôs centrais já instalados e funcionando e mais 16 projetados para começar a trabalhar em breve.

O cenário apresentado neste projeto possui características interessantes para a execução de testes com a implementação do SA, já que possui um alto número de pivôs centrais e, durante os períodos de estiagem de chuvas, o volume de água disponível precisa ser controlado para evitar danos ambientais.

O volume de água normalmente disponibilizado para o uso na irrigação é de 2.916,67 $m^3/hora$ de água por hora, podendo ocilar para mais ou menos, de acordo com o clima da região. No teste feito foram utilizadas 3 instâncias do problemas, cuja diferença entre elas é o volume de água disponibilizado. A demanda de água dos pivôs centrais ficou fixada e é apresentada no Apendice A, junto com as demais características dos pivôs centrais.

Os resultados encontrados com estas três instâncias estão apresentados na tabela 4.11, para o algoritmo do *Simulated Annealing* e GRASP.

Tabela 4.11: Resultados obtidos com os algoritmos construídos utilizando os 180 pivôs centrais do estudo de caso.

	SA			GRASP		
Água (m^3/hora)	Tempo (seg.)	Iterações	F.O. (u.m.)	Tempo (seg.)	Iterações	F.O. (u.m.)
2.395,83	73	7637	32.819,53	47,875	25	31.551,35
2.708,33	86,828	7637	30.565,93	44,875	25	29.228,45
2.916,67	83,812	7637	28.713,04	49,344	25	27.908,02

Ambos os algoritmos conseguiram construir boas soluções viáveis, porém percebe-se que o *Simulated Annealing* apresentou resultados na função objetivo piores do que o GRASP, ao contrário do que vinha acontecendo nos outros testes, conforme ilustrado na figura 4.5. Com base neste resultado, foram executados alguns testes, do tipo 'tentativa-e-erro', onde a vazão disponível de água foi reduzida com a intenção de encontrar o limite inferior de cada algoritmo.

O limite inferior do GRASP já era conhecido, e é de aproximadamente 57.500

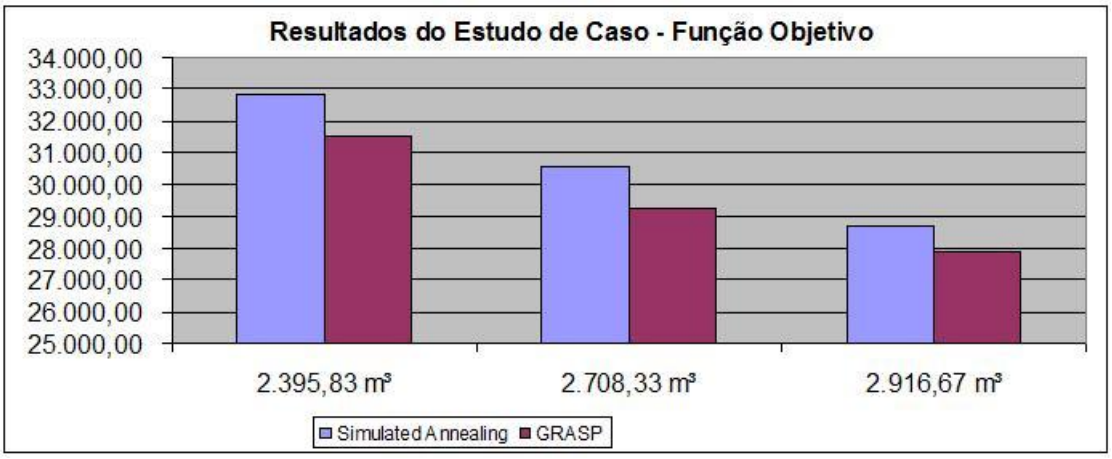

Figura 4.5: Gráfico com a função objetivo encontrada com o *Simulated Annealing* e o GRASP, aplicados junto às 3 instâncias do estudo de caso.

m^3 de água, igual à primeira instância de testes. Desta forma podemos considerar o algoritmo GRASP como a melhor alternativa para montagem de escalonamento quando tivermos um volume de água relativamente normal, sem grande escassez.

O limite inferior do *Simulated Annealing* ficou um pouco mais abaixo, em aproximadamente 52.500 m^3 de água, apresentando o algoritmo como uma excelente alternativa para situações mais críticas, pois sua capacidade de escalonamento em situações mais complexas mostrou-se satisfatória neste cenário.

Acredita-se que, com estes resultados, a montagem das escalas de funcionamento dos pivôs centrais desta região ficarão mais fáceis, ágeis e eficientes, pois serão feitas de forma automatizada, sem nenhum risco de haver alguma restrição violada. Além de fazer o controle de água utilizada pelo conjunto de pivôs centrais, o sistema ainda tenta minimizar o consumo de energia deste conjunto, tarefa que não é feita atualmente.

Os parâmetros utilizados pelo SA, em todos os experimentos, foram $T_0 = 10.000.000$, $\alpha = 0,9975$ e $T_c = 0,01$. Os parâmetros utilizados pelo GRASP, em todos os experimentos, foram $\alpha = 3$ e N^o de iterações $= 25$. Todos os testes foram realizados num laptop Sony Vaio fs-660/w com processador Intel Centrino 1.73 GHz e 1 Gb de memória RAM. Os algoritmos *Simulated Annealing* e GRASP foram desenvolvidos com a linguagem de programação C++.

Capítulo 5

Conclusões e Trabalhos Futuros

Este trabalho apresentou uma solução para resolver uma abordagem específica do problema de escalonamento de pivôs centrais. O desenvolvimento desta solução, que consiste em um programa de computador, baseou-se em quatro etapa, a saber, Levantamento e Análise de Requisitos, modelagem matemática do problema, modelagem do sistema, implementação, dos algortimos e sistema gerenciador, e testes.

A etapa de levantamento e análise dos requisitos foi baseada na experiência de profissionais da irrigação e em dados coletados do estudo de caso. Esta etapa foi feita de forma criteriosa, com a preocupação de que nenhuma falha grave poderia escapar, para não prejudicar as etapas seguintes. O maior parte do sucesso desta etapa se deve à execução da etapa de modelagem matemática do problema em paralelo. Desta forma foi possível validar as informações relativas à forma de resolução do problema, já que o modelo matemático criado era passado para o *software* LINGO, que apresentava valores de sua solução. Com base na análise destes valores, o modelo foi amadurecido, contemplando mais restrições do que foi pensado durante a concepção do projeto deste trabalho. O modelo matemático final proposto foi capaz de representar o problema em todos os seus aspectos importantes. Com o modelo matemático consolidado, a modelagem do sistema, utilizando a UML, foi iniciada. Nesta etapa o sistema foi dividido em três módulos distintos: Interface, Algoritmos e Dados.

O módulo Inteface foi projetado de forma a ser extremamente especializado, ou seja, sua única função seria executar funções para escalonar os pivôs centrais e

apresentar o resultado obtido. Em contra partida, os demais módulos do sistema foram desenvolvidos para serem totalmente portábeis, podendo ser facilmente reutilizado em outros sistemas.

O módulo Algoritmos possui as duas metaheurísticas implementadas e é dependente do módulo Estrutura de Dados, por ser o local em que ele busca os dados relevantes para a resolução do problema. A utilização da *Simulated Annealing* e do GRASP é uma boa alternativa para a obtenção de boas soluções em curtos espaços de tempo e com uma mínima exigência computacional, demonstrando ter sido uma escolha acertada para esta situação, pois atingiu os objetivos previstos. Além disto, as implementações realizadas apresentaram características distintas que faz com que se complementem, pois para cada cenário distinto, uma das implementações será mais vantajosa.

O *Simulated Annealing* mostrou-se mais eficiente quando o recurso hídrico apresenta-se mais escasso, demonstrando capacidade de contruir soluções com volumes de água mais baixos do que os suportados pelo GRASP. Já o GRASP mostrou-se mais eficiente quando o recurso hídrico não é tão escasso, apresentando resultados melhores que o *Simulated Annealing*, na maior parte das vezes. Apesar destes fatos, é interessante que, sempre que possivel, executar ambos os algoritmos e utilizar o resultado mais atraente para o tomador de decisão. É importante salientar que estas características são das implementações, e não dos algoritmos conceituais.

A estrutura de dados utilizada, na execução do algoritmo, também se mostrou eficaz, pois não exigiu muito recurso computacional e, consequentemente, não comprometeu a performance do sistema.

O sistema desenvolvido visa obter e apresentar soluções para o problema de escalonamento de pivôs centrais. A utilização de metaheurísticas é vista como a solução para se obter boas respostas em curtos intervalos de tempo e com uma menor exigência computacional, demonstrando ter sido acertada a escolha destes algoritmos para esta situação, pois foram apresentados bons resultados em curtos intervalos de tempo.

A complexidade dos problemas aumenta proporcionalmente em relação ao nú-

mero de pivôs centrais. Porém, o aumento da complexidade do problema não prejudica o desempenho dos algoritmos, já que os mesmos fazem suas buscas por soluções sem armazenar histórico e sem executar cálculos complexos.

A estrutura de dados utilizada, na execução do algoritmo, também se mostrou eficaz, pois não exige muito recurso computacional e, consequentemente, não compromete a performance do sistema. Além disto, a organização em pacotes permite que partes do sistema possam ser reaproveitadas em outros sistemas, tornando seus módulos reutilizáveis.

Atualmente estão implementados os algoritmos do *Simulated Annealing* e GRASP. Ainda são previstos, como trabalhos futuros, melhorias nos algoritmos implementados e nas interfaces, além da implementação de um Algoritmo Genético (Bäck et al., 1997), e um estudo comparativo das soluções encontradas, para a identificação das particularidades de cada caso.

Além disto, podemos citar como possíveis trabalhos futuros:

- Criar critérios para que o algoritmo do *Simulated Annealing* possa trabalhar com os três movimentos desenvolvidos em uma mesma execução.
- Criar um sistema de entrada, com cadastro dos pivôs centrais, mais interessante e de fácil usabilidade para usuários leigos.
- Desenvolver um algorítmo híbrido com a solução inicial do GRASP e o método de busca na vizinhança do *Simulated Annealing*.
- Trabalhar com períodos de tempo maiores, englobando, talvez, dias, semanas ou meses, fazendo uso de históricos de decisões anteriores do sistema.
- Trabalhar com várias regiões de pivôs centrais simultaneamente, utilizando diferentes fontes de recursos hídricos.
- Implementar heurísticas clássicas e análogicas para resolver o problema de escalonamento de pivôs centrais, uma vez que neste trabalho foram implementadas duas metaheurísticas estocásticas.

Referências Bibliográficas

Alba, E., Talbi, E.-G., Luque, G., and Melab, N. (2005). *Metaheuristics and Parallelism.* John Wiley & Sons.

Andrade, E. L. (2000). *Introdução à pesquisa operacional.* LTC, Rio de Janeiro, 2ª edition.

Baskent, E. and Jordan, G. (2002). Forest landscape management modeling using simulated annealing. *Forest Ecology and Management*, 165:29–45.

Bäck, T., Fogel, D. B., and Michalewicz, Z. (1997). *Handbook of Evolucionary Computation.* Oxford University Press.

Bernardo, S., Soares, A. A., and Mantovani, E. C. (2006). *Manual de Irrigação.* Editora UFV, 8ª edition.

Biajoli, F. L., Mine, O. M., Chaves, A. A., and Souza, M. J. F. (2003). Escala de jogos de torneios esportivos: uma abordagem via simulated annealing. *Anais do XXXV Símposio Brasileiro de Pesquisa Operacional*, pages 1295–1306.

Blum, C. and Roli, A. (2001). Metaheuristics in combinatorial optimisation: Overview and conceptual comparison. Technical Report 13, Université Libre de Bruxelles.

Busseti, F. (2001). Simulated annealing overview. http://www.geocities.com/francorbusetti/.

Cerny, V. (1985). Thermodynamical aproach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Aplication*, 45(1):41–51.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2002). *Algoritmos*. Editora Campus, 2ª edition.

da Silva Gomes, H. A. (2003). Utilização da metaheurística simulated annealing no problema de alocação de pessoal em empresas de transporte coletivo por ônibus. Master's thesis, Universidade Federal do Ceará.

da Silva Gomes, H. A. and Neto, J. F. B. (2003). Utilização de metaheuristica na programação de escala de pessoal em empresas de transporte coletivo por ônibus. *Anais do XXXV Simpósio Brasileiro de Pesquisa Operacional*, pages 894–905.

Dantzig, G., Orden, A., and Wolfe, P. (1955). The generalized simplex method for minimizing a linear form under linear inequality restraints. *Pacific Journal of Mathematics*, 5:183.

de Miranda, J. H. and de Matos Pires, R. C., editors (2003). *Irrigação*, volume 2 of *Engenharia Agrícola*. FUNEP.

Deitel, H. and Deitel, P. (2001). *C++: como programar*. Bookman, 3ª edition.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms*. Phd thesis, Politecnico di Milano, Italy.

Ehrlich, P. J. (1978). *Pesquisa Operacional - Curso introdutório*. Editora Atlas, São Paulo, 2ª edition.

Feo, T. and Resende, M. (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71.

Feo, T. A. and Resende, M. G. C. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13:533–549.

Glover, F. (1989). Tabu search - part i. *ORSA Journal of Computing*, 1:190–206.

Glover, F. and Kochenberger, G., editors (2002). *Handbook of Metaheuristics*. Kluwer Academic Publishers.

Goldbarg, M. C. and Luna, H. P. (2005). *Otimização combinatória e programação linear: modelos e algoritmos*. Editora Campus, 2ª edition.

Gomes, A. C. and Souza, M. J. F. (2004). *Softwares de Otimização: Manual de Referência*. Universidade Federal de Ouro Preto.

Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278.

Hillier, F. S. and Lieberman, G. J. (1988). *Introdução à pesquisa operacional*. Editora Campus, Rio de Janeiro, 3ª edition.

Ibaraki, T., Nonobe, K., and Yagiura, M., editors (2005). *Metaheuristics: Progress as real Problem Solvers*. Springer Science, New York, United States of America.

Ignacio, A., Ferreira Filho, V., and Galvão, R. (2000). Métodos heurísticos num entorno paralelo. *Anais do Simpósio Brasileiro de Pesquisa Operacional*, 32:769–788.

Keller, J. and Bliesner, R. D. (1990). *Sprinkle and trickle irrigation*. AVI Book, New York.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

Land, A. H. and Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28:497–520.

López, F. G., Torres, M. G., Batista, B. M., Pérez, J. A. M., and Moreno-Vega, J. M. (2006). Solving feature subset selection problem by a parallel scatter search. *European Journal of Operational Research*, 169(2):477–489.

Mauri, G. R. and Lorena, L. A. N. (2006). Simulated annealing aplicado a um problema de roteirização e programação de veículos. *Anais do XXXVIII Simpósio Brasileiro de Pesquisa Operacional*, pages 1054–1065.

Melo, V. A. and Martinhon, C. A. (2004). Metaheurísticas híbridas para o problema do caixeiro viajante com coleta de prêmios. *Anais do XXXVI Simpósio Brasileiro de Pesquisa Operacional*, pages 1295–1306.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Jornal of Chemical Physics*, 21:1087–1092.

Mühlenbein, H., Mahnig, T., and Ochoa, A. (1999). Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5(2):215–247.

Mladenovic, N. and Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, 24:1097–1100.

Nogueira, R. T., de Paula Jr., G. G., and Póvoa, C. L. R. (2003). Uma heurística grasp para o problema da mochila quadrática 0-1. *XXXV Simpósio Brasileiro de Pesquisa Operacional*, pages 1245–1254.

Noronha, T., da Silva, M. M., and Aloise, D. (2001). Uma abordagem sobre estratégias metaheurísticas. *Revista Eletrônica de Iniciação Científica (REIC)*, I(I).

Pereira, G. W. (2004). Aplicação da técnica de recozimento simulado em problemas de planejamento florestal multiobjetivo. Master's thesis, Universidade Federal de Minas Gerais.

Pereira, G. W. and do Nascimento Santos, H. (2004). Aplicação do simulated annealing na solução de problemas de planejamento florestal multiobjetivo. *Anais do XXXVI Simpósio Brasileiro de Pesquisa Operacional*, pages 9–20.

Pruski, F. F., del G. Rodriguez, R., de Novaes, L. F., da Silva, D. D., Ramos, M. M., and de F. Teixeira, A. (2007). Impacto das vazões demandadas pela irrigação e

pelos abastecimentos animal e humano, na bacia do paracatu. *Revista Brasileira de Engenharia Agrícola e Ambiental*, 11(2):199–210.

Puccini, A. d. L. and Pizzolato, N. D. (1990). *Programação Linear*. Editora Livros Técnicos e Científicos, Rio de Janeiro, 2ª edition.

Rangel, M. C., de Abreu, N. M. M., and Boaventura-netto, P. O. (2000). Grasp para o pqa: um limite de aceitação para soluções iniciais. *Pesqui. Oper.*, 20(1):45–58.

Reeves, C. R. (1993). *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Press.

Resende, M. and Ribeiro, C. (2003). *Handbook of Metaheuristics.*, chapter Greedy randomized adaptive search procedures. Kluwer.

Rodrigues, F. (2001). *Metaheurística e sistema de suporte à decisão no gerenciamento de recursos florestais*. PhD thesis, Universidade Federal de Viçosa.

Sartori, M. A., Dias, D. F., Perez, R., Batista, E. S., and Masala, R. M. (2006). Estudo locacional de uma unidade produtora de óleo vegetal de mamona no norte do estado de minas gerais. *Anais do XIII Simpósio de Engenharia de Produção*.

Schrage, L. (1998). *Optimization Modeling with Lingo*. Lindo Systems Inc., second edition.

Serra, T. and Moura, A. V. (2006). Escalonamento integrado para o transporte coletivo utilizando grasp, path-relinking e paralelismo. *Anais do XXXVIII Simpósio Brasileiro de Pesquisa Operacional*, pages 963–974.

Zionts, S. (1974). *Linear and Integer Programming*. Prentice-Hall, New Jersey.

Apêndice A

Pivôs centrais do estudo de caso

Abaixo, segue os dados relevantes de todos os pivôs centrais do projeto Paracatu-Entre Ribeiros, componentes do estudo de caso.

Pivô central	**Horas de funcionamento**	**Água utilizada** ($m^3/hora$)	**Potência gasta** ($KWs/hora$)
Pivô 001	8	704,00	24,31
Pivô 002	9	550,00	19,00
Pivô 003	9	600,00	20,72
Pivô 004	9	208,00	7,18
Pivô 005	9	340,00	11,74
Pivô 006	9	158,00	5,46
Pivô 007	9	704,00	24,31
Pivô 008	10	550,00	19,00
Pivô 009	10	600,00	20,72
Pivô 010	11	340,00	11,74
Pivô 011	12	208,00	7,18
Pivô 012	13	158,00	5,46
Pivô 013	14	704,00	24,31
Pivô 014	15	550,00	19,00
Pivô 015	16	704,00	24,31

Pivô 016	17	550,00	19,00
Pivô 017	18	640,00	22,10
Pivô 018	19	704,00	24,31
Pivô 019	20	550,00	19,00
Pivô 020	21	600,00	20,72
Pivô 021	22	340,00	11,74
Pivô 022	22	208,00	7,18
Pivô 023	23	158,00	5,46
Pivô 024	23	704,00	24,31
Pivô 025	23	550,00	19,00
Pivô 026	22	600,00	20,72
Pivô 027	21	208,00	7,18
Pivô 028	21	340,00	11,74
Pivô 029	20	158,00	5,46
Pivô 030	19	704,00	24,31
Pivô 031	18	280,00	9,67
Pivô 032	8	350,00	12,09
Pivô 033	9	704,00	24,31
Pivô 034	9	550,00	19,00
Pivô 035	9	600,00	20,72
Pivô 036	9	208,00	7,18
Pivô 037	9	340,00	11,74
Pivô 038	9	158,00	5,46
Pivô 039	10	704,00	24,31
Pivô 040	10	550,00	19,00
Pivô 041	11	600,00	20,72
Pivô 042	12	340,00	11,74
Pivô 043	13	208,00	7,18
Pivô 044	14	158,00	5,46

Pivô 045	15	704,00	24,31
Pivô 046	16	550,00	19,00
Pivô 047	17	704,00	24,31
Pivô 048	18	550,00	19,00
Pivô 049	19	640,00	22,10
Pivô 050	20	704,00	24,31
Pivô 051	21	550,00	19,00
Pivô 052	22	600,00	20,72
Pivô 053	22	340,00	11,74
Pivô 054	23	208,00	7,18
Pivô 055	23	158,00	5,46
Pivô 056	23	704,00	24,31
Pivô 057	22	550,00	19,00
Pivô 058	21	600,00	20,72
Pivô 059	21	208,00	7,18
Pivô 060	20	340,00	11,74
Pivô 061	19	158,00	5,46
Pivô 062	18	704,00	24,31
Pivô 063	8	280,00	9,67
Pivô 064	9	350,00	12,09
Pivô 065	9	704,00	24,31
Pivô 066	9	550,00	19,00
Pivô 067	9	600,00	20,72
Pivô 068	9	208,00	7,18
Pivô 069	9	340,00	11,74
Pivô 070	10	158,00	5,46
Pivô 071	10	704,00	24,31
Pivô 072	11	550,00	19,00
Pivô 073	12	600,00	20,72

Pivô 074	13	340,00	11,74
Pivô 075	14	208,00	7,18
Pivô 076	15	158,00	5,46
Pivô 077	16	704,00	24,31
Pivô 078	17	550,00	19,00
Pivô 079	18	704,00	24,31
Pivô 080	19	550,00	19,00
Pivô 081	20	640,00	22,10
Pivô 082	21	704,00	24,31
Pivô 083	22	550,00	19,00
Pivô 084	22	600,00	20,72
Pivô 085	23	340,00	11,74
Pivô 086	23	208,00	7,18
Pivô 087	23	158,00	5,46
Pivô 088	22	704,00	24,31
Pivô 089	21	550,00	19,00
Pivô 090	21	600,00	20,72
Pivô 091	20	208,00	7,18
Pivô 092	19	340,00	11,74
Pivô 093	18	158,00	5,46
Pivô 094	8	704,00	24,31
Pivô 095	9	280,00	9,67
Pivô 096	9	350,00	12,09
Pivô 097	9	704,00	24,31
Pivô 098	9	550,00	19,00
Pivô 099	9	600,00	20,72
Pivô 100	9	208,00	7,18
Pivô 101	10	340,00	11,74
Pivô 102	10	158,00	5,46

Pivô 103	11	704,00	24,31
Pivô 104	12	550,00	19,00
Pivô 105	13	600,00	20,72
Pivô 106	14	340,00	11,74
Pivô 107	15	208,00	7,18
Pivô 108	16	158,00	5,46
Pivô 109	17	704,00	24,31
Pivô 110	18	550,00	19,00
Pivô 111	19	704,00	24,31
Pivô 112	20	550,00	19,00
Pivô 113	21	640,00	22,10
Pivô 114	22	704,00	24,31
Pivô 115	22	550,00	19,00
Pivô 116	23	600,00	20,72
Pivô 117	23	340,00	11,74
Pivô 118	8	208,00	7,18
Pivô 119	9	158,00	5,46
Pivô 120	10	704,00	24,31
Pivô 121	11	550,00	19,00
Pivô 122	12	600,00	20,72
Pivô 123	13	208,00	7,18
Pivô 124	14	340,00	11,74
Pivô 125	14	158,00	5,46
Pivô 126	14	704,00	24,31
Pivô 127	14	280,00	9,67
Pivô 128	15	350,00	12,09
Pivô 129	15	704,00	24,31
Pivô 130	14	550,00	19,00
Pivô 131	14	600,00	20,72

Pivô 132	12	208,00	7,18
Pivô 133	8	340,00	11,74
Pivô 134	9	158,00	5,46
Pivô 135	10	704,00	24,31
Pivô 136	10	550,00	19,00
Pivô 137	11	600,00	20,72
Pivô 138	12	340,00	11,74
Pivô 139	13	208,00	7,18
Pivô 140	13	158,00	5,46
Pivô 141	13	704,00	24,31
Pivô 142	14	550,00	19,00
Pivô 143	14	704,00	24,31
Pivô 144	14	550,00	19,00
Pivô 145	14	640,00	22,10
Pivô 146	13	704,00	24,31
Pivô 147	13	550,00	19,00
Pivô 148	14	600,00	20,72
Pivô 149	14	340,00	11,74
Pivô 150	14	208,00	7,18
Pivô 151	13	158,00	5,46
Pivô 152	13	704,00	24,31
Pivô 153	12	550,00	19,00
Pivô 154	11	600,00	20,72
Pivô 155	10	208,00	7,18
Pivô 156	9	340,00	11,74
Pivô 157	8	158,00	5,46
Pivô 158	22	704,00	24,31
Pivô 159	22	280,00	9,67
Pivô 160	23	350,00	12,09

Pivô 161	23	704,00	24,31
Pivô 162	23	550,00	19,00
Pivô 163	22	600,00	20,72
Pivô 164	21	208,00	7,18
Pivô 165	21	340,00	11,74
Pivô 166	20	158,00	5,46
Pivô 167	19	704,00	24,31
Pivô 168	18	550,00	19,00
Pivô 169	8	600,00	20,72
Pivô 170	9	340,00	11,74
Pivô 171	9	208,00	7,18
Pivô 172	9	158,00	5,46
Pivô 173	9	704,00	24,31
Pivô 174	9	550,00	19,00
Pivô 175	9	704,00	24,31
Pivô 176	10	550,00	19,00
Pivô 177	10	640,00	22,10
Pivô 178	11	704,00	24,31
Pivô 179	12	550,00	19,00
Pivô 180	13	600,00	20,72

Apêndice B

Grupo de 79 pivôs centrais simulados para testes

Abaixo, segue os dados do grupo de pivôs centrais simulados para a instância de 79 pivôs centrais.

Pivô central	**Horas de funcionamento**	**Água utilizada** ($m^3/hora$)	**Potência gasta** ($Watts/hora$)
Pivô 01	10	81,0	2000
Pivô 02	21	58,0	2000
Pivô 03	10	58,0	2000
Pivô 04	10	58,0	2000
Pivô 05	21	58,0	2000
Pivô 06	17	390,0	2000
Pivô 07	18	427,0	2000
Pivô 08	21	310,0	2000
Pivô 09	21	214,0	2000
Pivô 10	21	160,0	2000
Pivô 11	20	176,0	2000
Pivô 12	20	134,0	2000
Pivô 13	20	148,0	2000

Pivô 14	20	200,0	2000
Pivô 15	20	40,0	2000
Pivô 16	6	128,0	2000
Pivô 17	20	88,0	2000
Pivô 18	20	308,0	2000
Pivô 19	20	303,0	2000
Pivô 20	20	333,0	2000
Pivô 21	20	258,0	2000
Pivô 22	20	320,0	2000
Pivô 23	21	247,0	2000
Pivô 24	20	352,0	2000
Pivô 25	21	216,0	2000
Pivô 26	20	602,0	2000
Pivô 27	20	632,0	2000
Pivô 28	20	483,5	2000
Pivô 29	20	481,0	2000
Pivô 30	21	164,0	2000
Pivô 31	8	235,0	2000
Pivô 32	11	448,0	2000
Pivô 33	11	560,0	2000
Pivô 34	11	503,0	2000
Pivô 35	11	503,0	2000
Pivô 36	11	508,6	2000
Pivô 37	11	491,5	2000
Pivô 38	14	236,4	2000
Pivô 39	14	242,3	2000
Pivô 40	12	326,0	2000
Pivô 41	20	148,0	2000
Pivô 42	20	399,7	2000

Pivô 43	20	393,0	2000
Pivô 44	20	171,0	2000
Pivô 45	18	324,0	2000
Pivô 46	21	58,0	2000
Pivô 47	21	400,0	2000
Pivô 48	21	400,0	2000
Pivô 49	21	400,0	2000
Pivô 50	21	400,0	2000
Pivô 51	21	69,0	2000
Pivô 52	15	115,5	2000
Pivô 53	15	111,7	2000
Pivô 54	17	224,0	2000
Pivô 55	17	224,0	2000
Pivô 56	21	294,0	2000
Pivô 57	21	355,0	2000
Pivô 58	21	207,0	2000
Pivô 59	19	300,0	2000
Pivô 60	19	121,0	2000
Pivô 61	19	166,5	2000
Pivô 62	19	130,0	2000
Pivô 63	19	216,0	2000
Pivô 64	19	200,0	2000
Pivô 65	19	233,0	2000
Pivô 66	21	410,0	2000
Pivô 67	21	360,0	2000
Pivô 68	19	106,0	2000
Pivô 69	20	371,0	2000
Pivô 70	19	425,0	2000
Pivô 71	19	215,0	2000

Pivô 72	19	202,0	2000
Pivô 73	19	230,0	2000
Pivô 74	19	248,0	2000
Pivô 75	19	190,8	2000
Pivô 76	19	293,8	2000
Pivô 77	19	123,0	2000
Pivô 78	19	126,0	2000
Pivô 79	5	106,5	2000

Apêndice C

Grupo de 300 pivôs centrais simulados para testes

Abaixo, segue os dados do grupo de pivôs centrais simulados para a instância de 300 pivôs centrais.

Pivô central	**Horas de funcionamento**	**Água utilizada** ($m^3/hora$)	**Potência gasta** ($Watts/hora$)
Pivô 01	10	81,0	2000
Pivô 02	21	58,0	2000
Pivô 03	10	58,0	2000
Pivô 04	10	58,0	2000
Pivô 05	21	58,0	2000
Pivô 06	17	390,0	2000
Pivô 07	18	427,0	2000
Pivô 08	21	310,0	2000
Pivô 09	21	214,0	2000
Pivô 10	21	160,0	2000
Pivô 11	20	176,0	2000
Pivô 12	20	134,0	2000
Pivô 13	20	148,0	2000

Pivô 14	20	200,0	2000
Pivô 15	20	40,0	2000
Pivô 16	6	128,0	2000
Pivô 17	20	88,0	2000
Pivô 18	20	308,0	2000
Pivô 19	20	303,0	2000
Pivô 20	20	333,0	2000
Pivô 21	20	258,0	2000
Pivô 22	20	320,0	2000
Pivô 23	21	247,0	2000
Pivô 24	20	352,0	2000
Pivô 25	21	216,0	2000
Pivô 26	20	602,0	2000
Pivô 27	20	632,0	2000
Pivô 28	20	483,5	2000
Pivô 29	20	481,0	2000
Pivô 30	21	164,0	2000
Pivô 31	8	235,0	2000
Pivô 32	11	448,0	2000
Pivô 33	11	560,0	2000
Pivô 34	11	503,0	2000
Pivô 35	11	503,0	2000
Pivô 36	11	508,6	2000
Pivô 37	11	491,5	2000
Pivô 38	14	236,4	2000
Pivô 39	14	242,3	2000
Pivô 40	12	326,0	2000
Pivô 41	20	148,0	2000
Pivô 42	20	399,7	2000

Pivô 43	20	393,0	2000
Pivô 44	20	171,0	2000
Pivô 45	18	324,0	2000
Pivô 46	21	58,0	2000
Pivô 47	21	400,0	2000
Pivô 48	21	400,0	2000
Pivô 49	21	400,0	2000
Pivô 50	21	400,0	2000
Pivô 51	21	69,0	2000
Pivô 52	15	115,5	2000
Pivô 53	15	111,7	2000
Pivô 54	17	224,0	2000
Pivô 55	17	224,0	2000
Pivô 56	21	294,0	2000
Pivô 57	21	355,0	2000
Pivô 58	21	207,0	2000
Pivô 59	19	300,0	2000
Pivô 60	19	121,0	2000
Pivô 61	19	166,5	2000
Pivô 62	19	130,0	2000
Pivô 63	19	216,0	2000
Pivô 64	19	200,0	2000
Pivô 65	19	233,0	2000
Pivô 66	21	410,0	2000
Pivô 67	21	360,0	2000
Pivô 68	19	106,0	2000
Pivô 69	20	371,0	2000
Pivô 70	19	425,0	2000
Pivô 71	19	215,0	2000

Pivô 72	19	202,0	2000
Pivô 73	19	230,0	2000
Pivô 74	19	248,0	2000
Pivô 75	19	190,8	2000
Pivô 76	19	293,8	2000
Pivô 77	19	123,0	2000
Pivô 78	19	126,0	2000
Pivô 79	5	106,5	2000
Pivô 80	10	81,0	2000
Pivô 81	10	81,0	2000
Pivô 82	21	58,0	2000
Pivô 83	10	58,0	2000
Pivô 84	10	58,0	2000
Pivô 85	21	58,0	2000
Pivô 86	17	390,0	2000
Pivô 87	18	427,0	2000
Pivô 88	21	310,0	2000
Pivô 89	21	214,0	2000
Pivô 90	21	160,0	2000
Pivô 91	20	176,0	2000
Pivô 92	20	134,0	2000
Pivô 93	20	148,0	2000
Pivô 94	20	200,0	2000
Pivô 95	20	40,0	2000
Pivô 96	6	128,0	2000
Pivô 97	20	88,0	2000
Pivô 98	20	308,0	2000
Pivô 99	20	303,0	2000
Pivô 100	20	333,0	2000

Pivô 101	20	258,0	2000
Pivô 102	20	320,0	2000
Pivô 103	21	247,0	2000
Pivô 104	20	352,0	2000
Pivô 105	21	216,0	2000
Pivô 106	20	602,0	2000
Pivô 107	20	632,0	2000
Pivô 108	20	483,5	2000
Pivô 109	20	481,0	2000
Pivô 110	21	164,0	2000
Pivô 111	8	235,0	2000
Pivô 112	11	448,0	2000
Pivô 113	11	560,0	2000
Pivô 114	11	503,0	2000
Pivô 115	11	503,0	2000
Pivô 116	11	508,6	2000
Pivô 117	11	491,5	2000
Pivô 118	14	236,4	2000
Pivô 119	14	242,3	2000
Pivô 120	12	326,0	2000
Pivô 121	20	148,0	2000
Pivô 122	20	399,7	2000
Pivô 123	20	393,0	2000
Pivô 124	20	171,0	2000
Pivô 125	18	324,0	2000
Pivô 126	21	58,0	2000
Pivô 127	21	400,0	2000
Pivô 128	21	400,0	2000
Pivô 129	21	400,0	2000

Pivô 130	21	400,0	2000
Pivô 131	21	69,0	2000
Pivô 132	15	115,5	2000
Pivô 133	15	111,7	2000
Pivô 134	17	224,0	2000
Pivô 135	17	224,0	2000
Pivô 136	21	294,0	2000
Pivô 137	21	355,0	2000
Pivô 138	21	207,0	2000
Pivô 139	19	300,0	2000
Pivô 140	19	425,0	2000
Pivô 141	19	215,0	2000
Pivô 142	19	202,0	2000
Pivô 143	19	230,0	2000
Pivô 144	19	248,0	2000
Pivô 145	19	190,8	2000
Pivô 146	19	293,8	2000
Pivô 147	19	123,0	2000
Pivô 148	19	126,0	2000
Pivô 149	5	106,5	2000
Pivô 150	5	106,5	2000
Pivô 151	21	69,0	2000
Pivô 152	15	115,5	2000
Pivô 153	15	111,7	2000
Pivô 154	17	224,0	2000
Pivô 155	17	224,0	2000
Pivô 156	21	294,0	2000
Pivô 157	21	355,0	2000
Pivô 158	21	207,0	2000

Pivô 159	19	300,0	2000
Pivô 160	19	121,0	2000
Pivô 161	19	166,5	2000
Pivô 162	19	130,0	2000
Pivô 163	19	216,0	2000
Pivô 164	19	200,0	2000
Pivô 165	19	233,0	2000
Pivô 166	21	410,0	2000
Pivô 167	21	360,0	2000
Pivô 168	19	106,0	2000
Pivô 169	20	371,0	2000
Pivô 170	19	425,0	2000
Pivô 171	19	215,0	2000
Pivô 172	19	202,0	2000
Pivô 173	19	230,0	2000
Pivô 174	19	248,0	2000
Pivô 175	19	190,8	2000
Pivô 176	19	293,8	2000
Pivô 177	19	123,0	2000
Pivô 178	19	126,0	2000
Pivô 179	5	106,5	2000
Pivô 180	10	81,0	2000
Pivô 181	10	81,0	2000
Pivô 182	21	58,0	2000
Pivô 183	10	58,0	2000
Pivô 184	10	58,0	2000
Pivô 185	21	58,0	2000
Pivô 186	17	390,0	2000
Pivô 187	18	427,0	2000

Pivô 188	21	310,0	2000
Pivô 189	21	214,0	2000
Pivô 190	21	160,0	2000
Pivô 191	20	176,0	2000
Pivô 192	20	134,0	2000
Pivô 193	20	148,0	2000
Pivô 194	20	200,0	2000
Pivô 195	20	40,0	2000
Pivô 196	6	128,0	2000
Pivô 197	20	88,0	2000
Pivô 198	20	308,0	2000
Pivô 199	20	303,0	2000
Pivô 200	20	333,0	2000
Pivô 201	10	81,0	2000
Pivô 202	21	58,0	2000
Pivô 203	10	58,0	2000
Pivô 204	10	58,0	2000
Pivô 205	21	58,0	2000
Pivô 206	17	390,0	2000
Pivô 207	18	427,0	2000
Pivô 208	21	310,0	2000
Pivô 209	21	214,0	2000
Pivô 210	21	160,0	2000
Pivô 211	20	176,0	2000
Pivô 212	20	134,0	2000
Pivô 213	20	148,0	2000
Pivô 214	20	200,0	2000
Pivô 215	20	40,0	2000
Pivô 216	6	128,0	2000

Pivô 217	20	88,0	2000
Pivô 218	20	308,0	2000
Pivô 219	20	303,0	2000
Pivô 220	20	333,0	2000
Pivô 221	20	258,0	2000
Pivô 222	20	320,0	2000
Pivô 223	21	247,0	2000
Pivô 224	20	352,0	2000
Pivô 225	21	216,0	2000
Pivô 226	20	602,0	2000
Pivô 227	20	632,0	2000
Pivô 228	20	483,5	2000
Pivô 229	20	481,0	2000
Pivô 230	21	164,0	2000
Pivô 231	8	235,0	2000
Pivô 232	11	448,0	2000
Pivô 233	11	560,0	2000
Pivô 234	11	503,0	2000
Pivô 235	11	503,0	2000
Pivô 236	11	508,6	2000
Pivô 237	11	491,5	2000
Pivô 238	14	236,4	2000
Pivô 239	14	242,3	2000
Pivô 240	12	326,0	2000
Pivô 241	20	148,0	2000
Pivô 242	20	399,7	2000
Pivô 243	20	393,0	2000
Pivô 244	20	171,0	2000
Pivô 245	18	324,0	2000

Pivô 246	21	58,0	2000
Pivô 247	21	400,0	2000
Pivô 248	21	400,0	2000
Pivô 249	21	400,0	2000
Pivô 250	21	400,0	2000
Pivô 251	21	69,0	2000
Pivô 252	15	115,5	2000
Pivô 253	15	111,7	2000
Pivô 254	17	224,0	2000
Pivô 255	17	224,0	2000
Pivô 256	21	294,0	2000
Pivô 257	21	355,0	2000
Pivô 258	21	207,0	2000
Pivô 259	19	300,0	2000
Pivô 260	19	121,0	2000
Pivô 261	19	166,5	2000
Pivô 262	19	130,0	2000
Pivô 263	19	216,0	2000
Pivô 264	19	200,0	2000
Pivô 265	19	233,0	2000
Pivô 266	21	410,0	2000
Pivô 267	21	360,0	2000
Pivô 268	19	106,0	2000
Pivô 269	20	371,0	2000
Pivô 270	19	425,0	2000
Pivô 271	19	215,0	2000
Pivô 272	19	202,0	2000
Pivô 273	19	230,0	2000
Pivô 274	19	248,0	2000

Pivô 275	19	190,8	2000
Pivô 276	19	293,8	2000
Pivô 277	19	123,0	2000
Pivô 278	19	126,0	2000
Pivô 279	5	106,5	2000
Pivô 280	10	81,0	2000
Pivô 281	10	81,0	2000
Pivô 282	21	58,0	2000
Pivô 283	10	58,0	2000
Pivô 284	10	58,0	2000
Pivô 285	21	58,0	2000
Pivô 286	17	390,0	2000
Pivô 287	18	427,0	2000
Pivô 288	21	310,0	2000
Pivô 289	21	214,0	2000
Pivô 290	21	160,0	2000
Pivô 291	20	176,0	2000
Pivô 292	20	134,0	2000
Pivô 293	20	148,0	2000
Pivô 294	20	200,0	2000
Pivô 295	20	40,0	2000
Pivô 296	6	128,0	2000
Pivô 297	20	88,0	2000
Pivô 298	20	308,0	2000
Pivô 299	20	303,0	2000
Pivô 300	20	333,0	2000

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