UNIVERSIDADE FEDERAL DO PAMPA

ALEX SILVA DA ROCHA

PROPOSTA DE UM MODELO DE OTIMIZAÇÃO PARA O APOIO AO PROJETO DE SISTEMAS DE IRRIGAÇÃO EM LAVOURAS DE ARROZ

Bage
2014

ALEX SILVA DA ROCHA

PROPOSTA DE UM MODELO DE OTIMIZAÇÃO PARA O APOIO AO PROJETO DE SISTEMAS DE IRRIGAÇÃO EM LAVOURAS DE ARROZ

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Orientadora: Ana Paula Lüdtke Ferreira

Bagé
2014

ALEX SILVA DA ROCHA

PROPOSTA DE UM MODELO DE OTIMIZAÇÃO PARA O APOIO AO PROJETO DE SISTEMAS DE IRRIGAÇÃO EM LAVOURAS DE ARROZ

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 28 de Agosto de 2014.

Banca examinadora:

_____

Profa. Dra. Ana Paula L. Ferreira
Orientadora
(UNIPAMPA)

_____

Prof. Dr. Sandro da Silva Camargo
(UNIPAMPA)

_____

Dr. José Pedro Trindade
(EMBRAPA)

Ofereço este trabalho primeiramente a minha família, Fabio, Cledi e Andréia. Assim como ao meu padrinho Cléber, minha namorada Carine e a todos os meus amigos pelo apoio e compreensão prestados ao decorrer desta etapa. Por final, gostaria também de dedicar à professora Ana Paula pela ajuda e confiança depositadas em mim durante o desenvolvimento deste trabalho.

“Se você quer ser bem sucedido, precisa ter dedicação total , buscar o seu último limite e dar o melhor de si.”

Ayrton Senna

RESUMO

A cultura do arroz é considerada peça chave do agronegócio brasileiro, contribuindo para a balança comercial com crescentes exportações. O Rio Grande do Sul, que emprega o sistema de cultivo irrigado, entra neste quadro como o maior produtor nacional. A cultura do arroz faz uso do método de irrigação por inundação, onde a água é aplicada sobre a superfície do solo de modo a formar uma lâmina d'água, que é mantida durante todo o desenvolvimento da planta. Para que a água permaneça na lavoura, é necessária a construção de um sistema de irrigação, que possui entre os seus principais componentes: reservatório, levantes, quadros e canais de irrigação. O presente trabalho propõe um modelo de otimização, o qual visa apoiar o projeto de um sistema de irrigação por inundação em lavouras de arroz, buscando encontrar os valores, dentre o conjunto das principais variáveis envolvidas no problema, que maximizem a eficiência do sistema, considerando os diferenciais topográficos da área onde este será introduzido. O desenvolvimento consiste na seleção de um *modelo matemático* que contemple os aspectos hidrológicos envolvidos na irrigação por inundação e na elaboração de um software de otimização baseado em a*lgoritmos genéticos.* A análise de sensibilidade dos experimentos apontou que o comportamento do algoritmo foi satisfatório, uma vez que os resultados obtidos foram coincidentes com os esperados de um sistema de irrigação por inundação real.

Palavras-Chave: Algoritmos Genéticos, Irrigação por Inundação, Modelo Hidrológico e Otimização.

ABSTRACT

Rice cultivation is considered a key part of Brazilian agribusiness, contributing to the trade balance with exports growing. The Rio Grande do Sul, which employs the irrigated cultivation system, enter this frame as the largest domestic producer. The rice cultivation method makes use of flood irrigation, in which water is applied to the soil surface to form a sheet of water, which is maintained throughout plant development. For water to remain in the field, it is necessary to build an irrigation system, this which has among its main components: shell, uprisings, frames and irrigation canals. This paper proposes an optimization model, which aims to support the design of a system of flood irrigation in rice fields, trying to find among the set values of the main variables involved in the problem, that maximize system efficiency, considering the differential surveying the area where it is introduced. The development consists of selecting a mathematical model that considers the hydrological aspects involved in flood irrigation and the development of a software optimization based on Genetic Algorithms. A sensitivity analysis of the experiments showed that the behavior of the algorithm is satisfactory, since the results obtained were mostly coincide with the expected of a system for real flood irrigation.

Keywords: Flood Irrigation, Genetic Algorithms, Hydrological Model and Optimization.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AG – Algoritmo Genético

D8 – Método Determinístico de Oito Células

GPS – *Global Positioning System*

IRGA – Instituto Rio Grandense do Arroz

MDE – Modelo Digital de Elevação

PI – Projeto de Irrigação

SIG – Sistema de Informação Geográfica

TCC-I – Trabalho de Conclusão de Curso I

TCC-II – Trabalho de Conclusão de Curso II

SUMÁRIO

1 INTRODUÇÃO

1.1 Cultura do arroz

De grande importância econômica e social, a cultura do arroz (orizicultura) se destaca principalmente pelo seu volume de produção e pelo tamanho da área de cultivo. Segundo EMBRAPA (2013), cerca de 150 milhões de hectares de arroz são cultivados anualmente no mundo todo gerando 590 milhões de toneladas, sendo mais de 90% produzido e consumido na Ásia. A produção mundial de arroz não tem acompanhado o crescimento do consumo, fato que preocupa, pois se trata de um alimento básico na dieta da população. Em 2006 o cereal atendia a cerca de 2,4 bilhões de pessoas, e estima-se que este número dobre até 2050 (INFOBIBOS, 2013).

O Brasil é o maior produtor de arroz fora do continente asiático (EMBRAPA, 2013), sendo considerado referência em produção e comercialização do alimento. Isto ocorre devido a tecnologias empregadas que asseguram as elevadas produtividade e qualidade do cereal, características exigidas pelo mercado nacional. Atualmente, a cultura do arroz é considerada peça chave do agronegócio brasileiro, contribuindo para a balança comercial com crescentes exportações. Além disso, a orizicultura conta ainda com a possibilidade da transformação de grãos de menor valor comercial em etanol, o que leva a cultura ao mercado da geração energética (ANUÁRIO BRASILEIRO DO ARROZ, 2013).

A produção do arroz emprega basicamente dois sistemas de cultivo: irrigado e de sequeiro (CASTRO, 2003). O primeiro é predominante na região Sul do Brasil e o segundo nas demais regiões do país. De acordo com Anuário Brasileiro do Arroz (2013) a colheita brasileira de arroz na safra 2011/2012 alcançou um volume total de 11,59 milhões de toneladas produzidas, com uma área de superfície semeada de 2,43 milhões de hectares a uma produtividade média de 4,78 mil quilos por hectare. Estima-se que os estados que produzem arroz irrigado (Rio Grande do Sul e Santa Catarina) no ciclo 2012/2013 sejam responsáveis por 75,4% de todo volume colhido no Brasil.

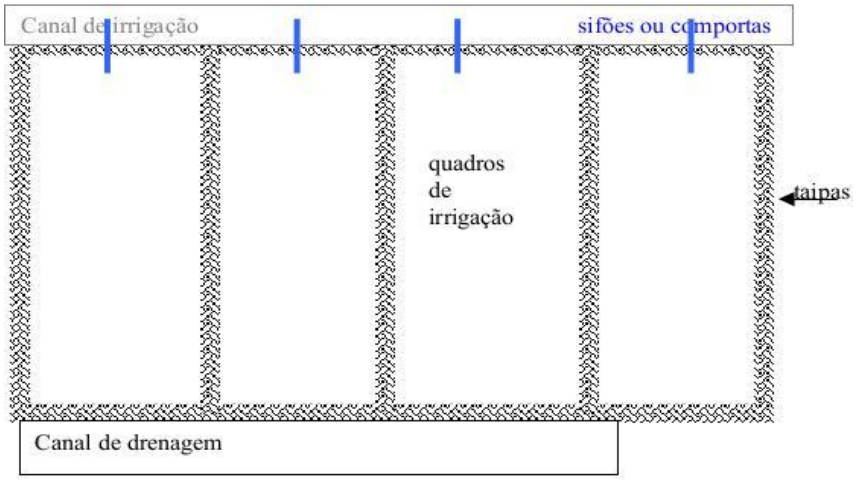
O Rio Grande do Sul (RS) entra neste quadro como o maior produtor nacional, ocupando uma parcela de cerca de 65% da produção Brasileira de arroz segundo Castro (2003). Para a safra 2012/2013, foi cultivada uma área de 1,07

milhão de hectares com uma previsão de rendimento médio de 7,53 mil quilos por hectare. Segundo (SOSBAI, 2010 apud FINGER, 2012), a cultura em 2009 representou mais de 3% do Imposto Sobre Circulação de Mercadorias e Serviços – ICMS e 2,74% do Produto Interno Bruto – PIB, chegando em alguns municípios a compor mais de 50% do valor bruto arrecadado.

De acordo com Castro (2003), a cultura do arroz faz uso do método de irrigação por inundação, onde a água é aplicada sobre a superfície do solo de modo a formar uma lâmina d'água. A irrigação no arroz dá-se de forma contínua, ou seja, a água é aplicada à lavoura alguns dias depois do plantio e mantida durante o desenvolvimento da cultura até os dias que antecedem a colheita.

Para que a lâmina d'água permaneça na lavoura é necessária à construção de quadros de irrigação (tabuleiros), os quais são delimitados por taipas (NILZA, 2003). O reservatório de água é interligado à lavoura por canais de irrigação. Se o reservatório estiver localizado em uma cota (altitude) inferior a dos quadros é necessária a utilização de um ou mais sistemas de levantes hidráulicos (também conhecidos como bombas, moto-bombas ou recalques) para levar a água do reservatório para os canais de irrigação. Então os principais componentes do sistema são (Figura 1): reservatório, bomba, quadros, taipas, canais e sifões ou comportas.

Figura 1 – Componentes de um sistema de irrigação por inundação.



Fonte: Nilza, 2003, p. 49

Segundo Castro (2003), os tabuleiros são paralelos às curvas de nível do terreno (Figura 2), e sua dimensão pode variar de alguns metros quadrados a vários hectares sendo inversamente proporcional à inclinação da área. A diferença de elevação entre as taipas à montante (superior) e à jusante (inferior) do quadro de irrigação não deve superar 2/3 da lâmina d'água média, a qual atualmente é da ordem de 10 cm. Número que não foi reduzido ainda mais devido principalmente à dificuldade de gestão da irrigação, pois pesquisas mostram que lâminas d'água menores (que reduziriam significativamente o consumo de água) não influenciam no rendimento da cultura. A distribuição da água pode ser feita de duas maneiras. Na primeira, a água passa de um tabuleiro para o outro, já na segunda a água que entra no quadro vem diretamente do canal de irrigação que alimenta a lavoura. A transição de um meio para o outro pode se dar de várias formas, entre estas estão o uso de sifões, comportas, tubos de PVC ou simplesmente é introduzida uma abertura na taipa com o auxílio de uma pá (situação mais comum).

Figura 2 – Lavoura de arroz sistematizada com curvas de nível.

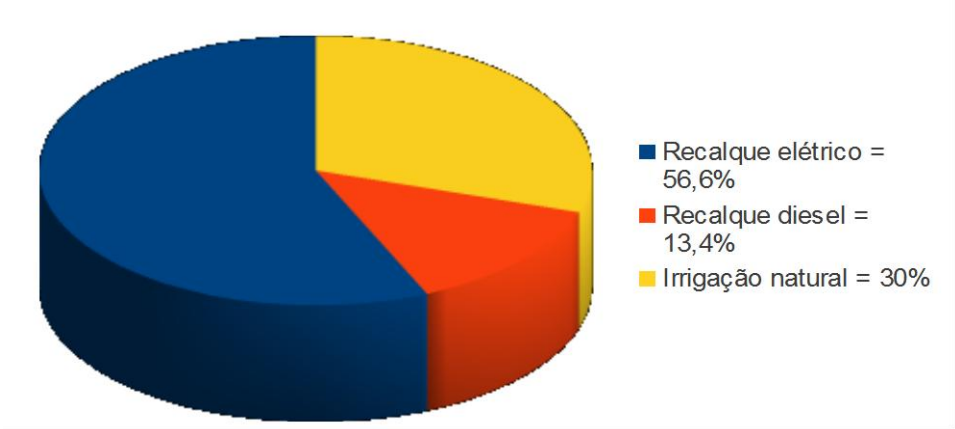
Fonte: Argosfoto, 2012

A técnica de cultivo por inundação consome aproximadamente 2 litros de água para produzir 1 kg de arroz com casca, o que coloca o arroz irrigado dentre as culturas mais exigentes em termos de recursos hídricos (EMBRAPA, 2013). O volume de água cada vez menor disponível para a irrigação e a extensa área

cultivada no RS posicionam a água como um fator limitante para o crescimento da cultura e, consecutivamente, do estado.

Em mais da metade das lavouras arrozeiras do RS, a água que as irriga se encontra em uma cota (nível altimétrico) inferior à das lavouras, necessitando a utilização de sistemas de levante por meio de bombas. Castro (2003) salienta que a maioria dos sistemas faz uso de energia elétrica, pois traz vantagens econômicas e ambientais. Mas a energia elétrica não está disponível em todas as localidades, nestes casos o diesel é utilizado como fonte de energia alternativa. A Figura 3 apresenta o cenário.

Figura 3 – Utilização de sistemas de levante no Rio Grande do Sul.



Fonte: IBGE, 2006

Mesmo com a utilização da energia elétrica, que segundo Castro (2003) pode gerar até 42% de economia na irrigação, a atividade está entre os itens de maior custo na cultura do arroz (RIGHES, 2006). Uma pesquisa realizada em 2009 pelo IRGA indicou que a irrigação representa em média 13% do custo total de produção da lavoura orizícola no Rio Grande do Sul (CASTRO, 2003).

De acordo com Corrêa (2007) a agricultura irrigada representa importantes custos adicionais a produção, uma vez que demanda investimentos em obras e aquisição de equipamentos para captação, transporte, controle e distribuição de água, energia elétrica e mão de obra. No caso específico da irrigação por inundação os custos relacionados às estruturas e operações exigidas para a execução desta atividade compreendem: canais, condutos, taipas, terraplanagem, operário

(aguador), drenagem e custos de bombeamento. Atualmente este último passou a ter influência maior no custo de produção do arroz, já que o preço da energia elétrica e do combustível tem aumentado consideravelmente.

Para Guimarães (2007), a agricultura de precisão deixou de ser futuro, passando a ser uma realidade no mundo inteiro. O termo engloba o uso de tecnologias atuais objetivando o aumento da eficiência da agricultura através de um manejo diferenciado. Basicamente, consistem em sistemas baseados em sensores interagindo com placas de hardware, transmissores e receptores de dados via satélite, inseridos na instrumentação de composições que envolvem trator e implemento. Estes sistemas coletam dados que são utilizados na administração rural como um todo através de softwares de gerenciamento agrícola, auxiliando principalmente no manejo do solo.

Contudo, o desenvolvimento de tecnologias voltadas à irrigação tem recebido pouca atenção, situação que não condiz com a relevância econômica e ambiental do tema. Guimarães (2007) ressalta que o baixo grau de controle da distribuição da água é um fator que contribui significativamente na redução da eficiência na irrigação. Neste sentido o presente trabalho propõe a utilização uma estratégia focada no projeto do sistema de irrigação para maximizar a eficiência deste processo em lavouras de arroz. É apresentado no decorrer desta monografia o desenvolvimento de um modelo de otimização capaz de oferecer apoio à tomada de decisões envoltas no projeto de sistemas de irrigação por inundação, de maneira a ampliar a eficiência e auxiliar no manejo desta atividade. A estratégia utilizada visa o aperfeiçoamento da topologia do canal de irrigação, buscando encontrar a configuração dos componentes do sistema que melhor se aplica aos diferenciais topográficos da área em que se deseja implantar método de irrigação em questão.

O desenvolvimento consiste na seleção de um modelo matemático que contemple os aspectos hidrológicos envolvidos na irrigação por inundação e na elaboração de um software de otimização baseado em a*lgoritmos genéticos* (PACHECO, 1999). Segundo Vieira (2007), modelos de otimização almejam encontrar os valores para um determinado conjunto de variáveis de decisão que maximizem ou minimizem uma função objetivo. Para Louzada (2004), no caso específico de sistemas de irrigação, é praticamente impossível aperfeiçoar tais processos através de procedimentos experimentais, e modelos computacionais

viabilizam otimizações sobre um grande número de cenários de forma rápida, barata e precisa.

De acordo com Moreira (2005), um modelo matemático hidrológico representa um protótipo do sistema de distribuição e movimentação de água no solo por meio de equações matemáticas, incluindo expressões lógicas e relações entre variáveis e parâmetros. Louzada (2004) salienta que vários modelos de sistemas hidrológicos foram desenvolvidos ao longo das últimas décadas, mas que seu uso rotineiro é limitado pelo grande número de parâmetros de entrada. O modelo de otimização proposto utiliza uma base de dados oriunda de um sistema de informação geográfico (SIG), ferramenta capacitada a armazenar e manipular informações cujo significado contém associações ou relações de natureza espacial (BOAS, 2008, p. 43).

Baseada no princípio da evolução das espécies darwiniano, a técnica dos algoritmos genéticos tem por objetivo desenvolver sistemas computacionais capazes de resolver problemas que envolvam um espaço de estados de tamanho intratável computacionalmente, através de um mecanismo de busca paralela e adaptativa. A teoria de Darwin privilegia os indivíduos mais aptos, os quais têm maior probabilidade de perpetuarem seus códigos genéticos (cromossomos) nas próximas gerações. Cada *cromossomo* consiste em uma *estrutura de dados*, que representa uma das possíveis soluções do problema. Portanto, os algoritmos genéticos buscam, por meio da evolução da população de soluções (codificadas nos cromossomos), a melhor resposta para o problema em estudo (PACHECO, 1999). Esta técnica é geralmente aplicada em problemas complexos de otimização, o que justifica sua utilização no problema de se projetar um sistema de irrigação por inundação, pois é elevado o número de parâmetros que precisam ser combinados para chegar à melhor solução.

1.2 Objetivos

Este trabalho tem como objetivo apresentar o desenvolvimento de um software de otimização baseado na técnica de algoritmo genéticos, que auxilie na implantação de um sistema de irrigação por inundação por meio da otimização da rede de distribuição de água.

Os objetivos específicos são:

- A redução dos custos relacionados à implantação do sistema, como construção de diques e instalação de sistemas de recalque;
- A redução dos custos de operação como consumo de energia e manutenção do sistema;
- Auxiliar no manejo da irrigação.

1.3 Metodologia

Para que os objetivos deste trabalho fossem alcançados, uma estratégia dividida nas etapas de modelagem e desenvolvimento foi seguida. A primeira parte (modelagem) buscou definir a estrutura do algoritmo, bem como apresentá-lo como ferramenta capaz de auxiliar no projeto de sistemas de irrigação por inundação aplicada a lavouras de arroz. O algoritmo elaborado durante a modelagem foi implementado na segunda parte do trabalho (desenvolvimento), onde os resultados obtidos por meio de simulações realizadas com o algoritmo foram avaliados e discutidos. A Tabela 1 contém o cronograma de desenvolvimento deste trabalho, onde é possível visualizar o conjunto de atividades realizadas ao decorrer deste processo. As atividades pertencentes à etapa de modelagem foram realizadas durante a disciplina de Trabalho de Conclusão de Curso I (TCC-I), e vão desde o levantamento bibliográfico até a atividade de escrita da monografia. As demais atividades pertencem à etapa de desenvolvimento, que foi realizada durante o semestre dois (disciplina de TCC-II).

A atividade de levantamento bibliográfico compreendeu uma pesquisa sobre modelos hidrológicos de otimização, algoritmos genéticos e banco de dados SIG. Durante a pesquisa, observou-se a carência na utilização de técnicas de otimização aplicadas à irrigação em lavouras de arroz, principalmente utilizando a topologia do canal de irrigação como variável principal. Também foi possível perceber a diversidade de níveis de abstração adotadas na modelagem de fenômenos físicos, assim como o fato de que problemas que em um primeiro instante parecem próximos, levam a construção de modelos completamente distintos devido a discrepâncias existentes entre os objetivos.

Tabela 1 – Cronograma de desenvolvimento.

Atividade	Semestre 1					Semestre 2				
	1	2	3	4	5	1	2	3	4	5
Levantamento bibliográfico	x	x								
Seleção do modelo de escoamento		x	x	x						
Modelagem do cromossomo				x						
Cálculo da função de adaptação				x	x					
Escrita da monografia de TCC-I				x	x					
Desenvolvimento						x	x			
Validação							x	x		
Avaliação dos resultados								x	x	
Escrita da monografia de TCC-II								x	x	x

Fonte: O autor

A segunda atividade realizada foi a seleção do modelo hidrológico de escoamento. Inicialmente o rascunho de um modelo foi concebido, baseando-se apenas nos formatos de dados manipulados por sistemas de informação geográfica. Ficou claro neste instante, que o conhecimento obtido até então não seria suficiente para elaboração de um modelo consistente, levando à necessidade de se compreender o princípio de funcionamento das principais ferramentas de modelagem hidrológica existentes. Durante esta pesquisa, uma ferramenta SIG uma chamou a atenção foi o software *PCRaster*[1], o qual permite simulação e modelagem dinâmica de processos físicos, descrevendo a distribuição, fluxo e transporte de material no solo (SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 2003). A utilização deste instrumento contribuiu para o esclarecimento de quais aspectos deveriam ser considerados pelo modelo hidrológico em desenvolvimento. A compreensão do problema obtida com o estudo desta ferramenta possibilitou a elaboração do modelo utilizado pelo algoritmo genético.

A atividade três foi a mais complexa do trabalho, onde se definiu a estrutura do cromossomo do algoritmo genético. A maior dificuldade encontrada neste ponto foi a de definir quais seriam as variáveis consideradas pelo modelo, ou seja, quais são os componentes que influenciam na eficiência do sistema de irrigação por

[1] O site da ferramenta é < http://pcraster.geo.uu.nl/ >

inundação. As restrições envolvidas na criação de indivíduos (cromossomos) também exigiram grande atenção, uma vez que estão diretamente ligadas ao desempenho do algoritmo genético.

A elaboração da função de adaptação dos cromossomos corresponde à quarta atividade do processo de modelagem. A primeira dificuldade enfrentada nesta etapa foi a de se encontrar a granularidade adequada à descrição o problema, de maneira que a definição da função objetivo obriga a formalização dos aspectos hidrológicos. Outro obstáculo encontrado foi o do estabelecimento de extremos para os valores alcançados pelas equações, permitindo a transformação da função objetivo em equação de aptidão. É necessário se estabelecer limites aos valores atribuídos pelas métricas aos resultados alcançados pelas soluções, de maneira que o ajuste destes pode alterar a importância que as variáveis têm para o problema.

A escrita da monografia do TCC-I encerra a etapa de modelagem do algoritmo. Este processo buscou, por meio da definição das principais referências consultadas, expor o problema e a teoria envolvida na solução proposta para este.

Conforme apresentado na Tabela 1, a segunda etapa do trabalho têm início com a atividade de desenvolvimento do algoritmo de otimização. Inicialmente se pensou em uma forma de desenvolvimento sequencial, isto é, com um fluxo unidirecional de desenvolvimento onde uma etapa concluída não é repetida. Entretanto, conforme o processo de implementação convergia para a conclusão desta atividade o conhecimento referente ao tema foi se ampliando, e consequentemente incoerências na modelagem inicial foram observadas levando ao surgimento de novas soluções. Portanto, o desenvolvimento do algoritmo foi um processo iterativo, onde os procedimentos de implementação, testes e modelagem foram repetidos. Um volume considerável de testes baseados na experiência do desenvolvedor foram realizados ao decorrer desta atividade, orientados pelo objetivo de garantir o funcionamento correto do algoritmo. A maior dificuldade enfrentada nos testes foi a de gerar saídas que permitissem verificar o desempenho do software, já que este abriga uma quantidade relevante de estruturas de dados no formato de matrizes e listas encadeadas. De um modo geral, as modificações realizadas no projeto inicial foram uma consequência das alterações realizadas na estratégia de modelagem do canal, uma vez que modificações nesta estrutura afetam o formato do cromossomo, o mecanismo de avaliação e o processo de propagação da água entre componentes. Várias dificuldades foram enfrentadas durante definição da

representação do processo de propagação da água entre os quadros de irrigação. E embora, inicialmente, este parecesse um processo simples, o desenvolvimento do software mostrou que não é.

A distinção de objetivos existente entre o modelo desenvolvido e os encontrados durante a pesquisa, dificultou a atividade de validação do algoritmo de otimização por meio da estratégia de comparação com outro modelo já existente. Desta forma a validação se restringiu ao comportamento do algoritmo de otimização, confrontando a conduta do modelo com a teoria.

A atividade de avaliação dos resultados seguiu um plano de simulações elaborado pelo desenvolvedor, onde um conjunto de configurações para as variáveis de entrada do algoritmo foi estabelecido, variando apenas uma variável de cada vez. Inicialmente o número total de configurações era inferior ao apresentado no Capítulo 4, porém o procedimento de análise demonstrou que para que conclusões concretas fossem retiradas, as variáveis deveriam ser submetidas a um número maior de valores de entrada. A maior dificuldade nesta etapa foi o tempo demandado, já que os experimentos foram realizados em triplicata e algumas simulações demoram até 8 horas (dependendo dos valores assumidos pelas variáveis de entrada). A conclusão do projeto ficou a cargo da atividade de escrita da monografia final.

1.4 Organização do Trabalho

No Capítulo 1 uma visão geral sobre o assunto é apresentada. O problema é exposto de forma a justificar a escolha do tema que posteriormente é delimitado. Objetivos gerais e específicos são descritos.

Uma revisão literária focada em algoritmos genéticos e modelos de otimização é exibida no Capítulo 2. Aspectos como a teoria envolvida no método de irrigação em questão, sua modelagem e os principais formatos de dados SIG são descritos. Assim, levou-se de forma sequencial o leitor a compreender os principais aspectos de complexidade envoltos na solução do problema para o qual o projeto propõe contribuir.

O Capitulo 3 é a parte central do trabalho, abrigando a solução desenvolvida e discussões sobre as escolhas feitas e resultados obtidos ao longo do trabalho. O algoritmo de otimização criado é apresentado, exibindo as principais abstrações e

sua relevância para com o problema de implantação do sistema de irrigação por inundação.

O fechamento do trabalho é feito no Capítulo 4. Nele são apresentadas as principais deduções obtidas ao decorrer do projeto e uma discussão sobre o modelo criado. Também nesse capítulo são discutidos os trabalhos que ainda poderão ser efetuados sobre o modelo proposto, visando a qualificação do mesmo.

2 REFERENCIAL TEÓRICO

2.1 Irrigação por inundação

Conforme apresentado na introdução, a cultura do arroz emprega o método de irrigação por inundação, o qual se caracteriza pela manutenção de lâmina d'água sobre a superfície do solo. Nesta seção estão expostos os aspectos técnicos envolvidos neste processo, juntamente com a descrição dos principais parâmetros associados aos componentes do sistema.

Áreas de topografia plana e de difícil drenagem são próprias para o cultivo do arroz irrigado. Tais características facilitam a manutenção da lâmina d'água e dificultam a perda de nutrientes (REUNIÃO TÉCNICA DA CULTURA DO ARROZ IRRIGADO, 2010). De acordo com Castro (2003), existem várias técnicas de preparo de solo, sendo o *preparo convencional* e o *plantio direto* os tipos predominantes no Estado do Rio Grande do Sul. Após o preparo do solo é feito o plantio, que é seguido do processo de irrigação. Este último tem início cerca de 10 a 15 dias após a emergência de 80% das plantas e término entre 15 e 20 dias antes da colheita da safra (etapa final do cultivo).

A água utilizada na irrigação é proveniente de barragens ou rios, sendo o primeiro tipo predominante no RS. Segundo Righes (2006) três operações contemplam o movimento da água em um sistema de irrigação, e para cada uma delas existem métricas quanto à eficiência do uso da água. São elas: *condução*, *distribuição* e *aplicação da água na lavoura*. A condução reflete os deslocamentos de água através de canais, condutores e estações de bombeamento, que vão desde a fonte do recurso até o sistema de distribuição na lavoura. A operação de distribuição representa o movimento da água pela rede de canais, alimentando as parcelas individuais da área cultivada. E a aplicação da água na lavoura, como o próprio nome denuncia, exprime o fluxo interno dos tabuleiros. No algoritmo de otimização desenvolvido, apenas as formas de distribuição e aplicação da água na lavoura são consideradas.

Na irrigação por inundação a eficiência no uso da água depende das características físicas e topográficas, cuidados operacionais e de um planejamento apropriado do sistema por meio da alocação de drenos e canais de irrigação (RIGHES, 2006). De acordo com EMBRAPA (2005), a eficiência de irrigação das

lavouras arrozeiras do RS está entre 40 e 45%, podendo chegar a 60% em condições adequadas de solo, relevo e manejo.

O canal principal que abastece a lavoura deve estar localizado na parte mais elevada do terreno, não necessitando ser retilíneo e nem de seguir as cotas mais elevadas em todo o seu traçado, devendo este buscar sempre o menor volume de aterro. Os canais de derivação devem estar preferencialmente a uma distância de cerca de 500 metros, com uma área de abrangência de 250 metros. De acordo com Righes (2006), a condução da água em canais representa perda considerável ao sistema de irrigação, e esta diminuição na eficiência deve-se principalmente aos fenômenos de percolação e infiltração lateral (fenômenos que serão explicados logo abaixo). A evaporação não representa perdas significativas, principalmente em canais de pequena largura. A maioria dos sistemas de levante utiliza energia elétrica (que apresenta tarifas diferenciadas em determinados horários). Com isso, as motobombas passam muito tempo desativadas e quando retoma o seu funcionamento uma vazão adicional é requerida para elevar o nível de água novamente nos canais principais. Devido à extensão dos canais, o nível da água cresce muito nas regiões próximas aos levantes, gerando assim desperdício.

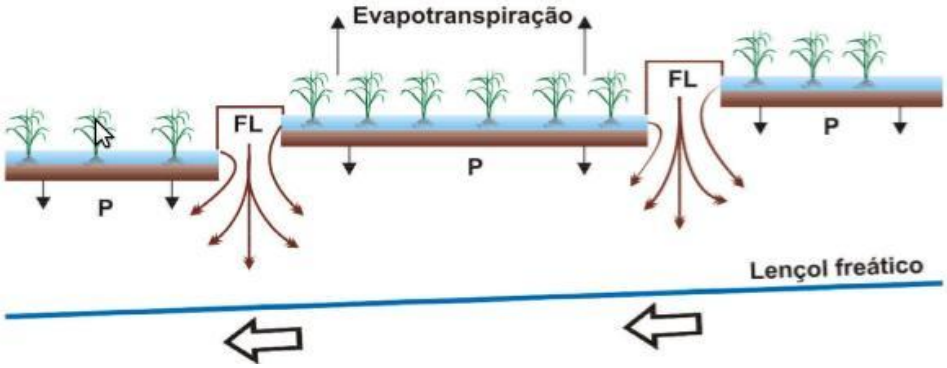
Conforme Righes (2006) o dimensionamento das estações de bombeamento está diretamente ligado às condições locais e um projeto implantado em um contexto não pode ser utilizado como base para outro. Desta forma, as principais recomendações são: minimizar a perda de carga nas tubulações, mantendo a velocidade na tubulação de recalque[2] (saída da bomba) inferior a 2 $m.s^{-1}$, e abaixo de 1,5 $m.s^{-1}$ na de sucção[3]; definir o conjunto moto-bomba com base nas curvas características das bombas considerando a altura manométrica total, vazão, diâmetro do rotor e rotação; utilizar a forma de acoplamento que apresente maior eficiência para interligar o motor e a bomba.

De acordo com Stone (2005) a quantidade de água que a cultura do arroz realmente necessita é a que a planta utiliza para crescer e transpirar. Mas para que essa demanda seja atendida é necessário uma quantidade de água adicional, esta que é perdida através de *evaporação*, *fluxo lateral* e *percolação* (Figura 4). Perdas através destes fenômenos são vistas como requerimento, pois não podem ser

[2] Tubulação que interliga a motobomba ao canal a montante
[3] Tubulação que interliga a motobomba ao canal à jusante

eliminadas do processo. Porém, é possível que estas sejam reduzidas com um manejo adequado da irrigação.

Figura 4 – Fenômenos evaporação, fluxo lateral (FL) e percolação (P) em lavouras de arroz.



Fonte: Stone, 2005, p. 20

Segundo Stone (2005) o movimento ascendente da água do solo para a atmosfera não ocorre apenas por meio da evaporação da água na superfície do solo, mas também pela transpiração das plantas, sendo denominada evapotranspiração a união destes dois eventos. A evapotranspiração independe da fase evolutiva da planta, e sim da demanda evaporativa da atmosfera.

Os fenômenos de percolação e fluxo lateral são ambos movimentos subterrâneos e de difícil separação (STONE, 2005). O primeiro pode ser descrito como o movimento vertical da água além da zona radicular da planta em direção ao lençol freático, e o fluxo lateral, como o movimento lateral da água subsuperficial. A água que percola se destina ao lençol freático, enquanto a proveniente de fluxo lateral geralmente está direcionada a um dreno ou rio. As perdas líquidas por fluxo lateral ocorrem apenas no quadro mais baixo da lavoura, pois a perda ocorrida em um tabuleiro é compensada pelo fluxo lateral do tabuleiro mais alto, funcionando o ultimo quadro (mais baixo) como dreno para o sistema como um todo. Dentre os aspectos que influenciam os níveis de percolação e fluxo lateral, estão as propriedades do solo e o manejo, este último representado principalmente pela altura da lâmina d'água.

Stone (2005) salienta que o volume de água necessário na saturação do solo é diretamente proporcional à porosidade, à profundidade a ser saturada e ao grau de saturação inicial do solo. Já o processo de formação da lâmina depende do tempo

gasto na sua formação, da velocidade de infiltração e da evapotranspiração durante o período de formação da lâmina. Este é o período em que o sistema de irrigação é submetido aos maiores valores de demanda hídrica, requerimento que tende a diminuir durante o período contínuo de irrigação. A vazão necessária durante o período contínuo é de aproximadamente 1,15 a 1,76 $l.s^{-1}.ha^{-1}$, não considerando precipitações pluviais e perdas nos canais durante o processo de condução da água.

Finger (2012) destaca que a área cultivada com arroz no RS está diretamente ligada à disponibilidade de recursos hídricos, sendo este considerado como um fator limitante para o crescimento da produção gaúcha do cereal. Conforme Righes (2006) os orizicultores gaúchos sofrem problemas de escassez de água para irrigação, situação que é agravada pelos fenômenos de estiagem. A realidade é que, com o passar dos anos, os recursos hídricos têm ficado cada vez mais escassos e os solos mais impermeáveis, fato que reduz a água das vertentes e as vazões dos rios. Para que esta situação seja contornada é necessário ampliar a capacidade de armazenamento de água, aumentar as eficiências de: captação, condução, aplicação e de uso de água na cultura do arroz.

2.2 Modelos de otimização

Conforme Rennó (2004), um modelo é uma representação simplificada dos processos envolvidos em um sistema real. O seu uso cada vez maior em estudos ambientais é justificado pelo auxilio por ele prestado no entendimento do impacto que alterações na cobertura da terra promovem nos ecossistemas. De forma geral, processos ambientais são bastante complexos e nem sempre é possível descrevê-los de forma detalhada por um conjunto de equações matemáticas, já que estas podem ter elevada complexidade exigindo simplificações para sua utilização. É possível se descrever um processo de várias maneiras, não existindo um modelo que possa ser considerado o melhor, mas sim, o mais adequado para a descrição de determinando fenômeno.

Um modelo é constituído de quatro grupos básicos de informações (RENNÓ, 2004). O primeiro contém um conjunto de variáveis que correspondem ao estado ou condição do sistema. Um exemplo é uma variável de estado pode armazenar o volume de água no solo em um determinado instante. As equações que compõem o modelo são denominadas *funções de transferência*. O fluxo e as interações que

acontecem entre estas funções formam o segundo grupo. No terceiro, estão as entradas do modelo, que compõem fatores que afetam os componentes do mesmo e são representados por meio de equações chamadas *funções forçantes*. Estas funções funcionam como um controlador de fluxos, ou atuam como um propulsor para o modelo, representando um fluxo ou estoque externo que o alimenta. No quarto e último grupo estão os parâmetros do modelo, que são constantes utilizadas pelas equações matemáticas.

Abaixo estão os principais aspectos considerados ao se classificar modelos (RENNÓ, 2004):

- variáveis utilizadas (estocástico ou determinístico);
- relacionamento entre as variáveis (empíricos ou baseados em processos);
- representação dos dados (discretos ou contínuos);
- existência de dependência temporal (estáticos ou dinâmicos).

De acordo com Rennó (2004), quando pelo menos uma das variáveis envolvidas no modelo apresenta comportamento aleatório, este é dito estocástico. Em outras palavras, dado um determinado valor de entrada o modelo pode produzir mais de um valor de saída, caso contrário, este é chamado determinístico. Modelos empíricos utilizam relações baseadas em observações, sendo muito simples e úteis, mas pouco robustos, não permitindo simulação de condições não previstas. Já os modelos baseados em processo, são mais complexos e buscam descrever por meio de equações todos os processos envoltos no fenômeno. Apesar dos fenômenos naturais ocorrerem de forma contínua no tempo, modelos computacionais tem a capacidade de manipular apenas um conjunto discreto de informações. Sendo assim, a chave para driblar esta restrição está na definição da precisão do resultado, pois é ela que dita o intervalo de tempo que o modelo deve considerar. O último aspecto citado acima classifica modelos em dinâmico e estático, sendo o primeiro caracterizado pela iteratividade, utilizando como entrada para uma repetição o resultado da iteração anterior. No estático, os dados de entrada são tomados como base e o resultado é produzido em um único passo.

Processos podem ser modelados com diferentes escalas de tempo e detalhamento espacial (RENNÓ, 2004). Em escalas temporais menores a conexão entre atmosfera e vegetação ocorre por meio da radiação solar incidente, que promove circulação de umidade e calor. Ao se ampliar o ciclo de resposta este acoplamento se dá mediante ciclos bioquímicos, hidrológicos e com o solo.

Predominam na atmosfera processos da ordem entre segundos e dias, dentre estes estão os processos de troca de energia, descritos em função de características da superfície, tais como vegetação e rugosidade. Sendo assim, as escalas espacial e temporal selecionadas para o modelo, por meio de um nível de conceituação dos processos hidrológicos, devem ser compatíveis com o fenômeno observado.

Para Lima (2008), *ciclo hidrológico* é o nome dado ao conjunto de fases e processos envolvidos na circulação da água no meio ambiente. O termo ciclo vem do fato de não haver perdas significativas no volume de água disponível no planeta, sofrendo apenas alterações em seu estado e localização. Um modelo hidrológico contém representações matemáticas capacitadas a simular processos relativos ao ciclo hidrológico. Rennó (2004) define modelo hidrológico como uma representação matemática do fluxo de água e seus constituintes sobre alguma parte da superfície e/ou subsuperfície terrestre. Modelos atmosféricos e hidrológicos não são totalmente disjuntos, pois a hidrologia é fortemente relacionada às condições climáticas.

Rennó (2004) ressalta que as bacias hidrográficas são objeto de estudo da maioria dos modelos hidrológicos. Elas são compreendidas por superfícies que captam e drenam água sobre canais de escoamento direcionados a uma única saída. Modelos hidrológicos oferecem uma melhor compreensão dos fenômenos hidrológicos em uma bacia hidrográfica, assim como apresentam o comportamento da bacia sob condições diversas. Vale salientar que na maioria dos modelos hidrológicos, nem todos os processos envolvidos em uma bacia hidrográfica são representados, o que varia com os objetivos do modelo desenvolvido (LIMA, 2008).

O solo, do ponto de vista hidrológico, basicamente consiste em um reservatório onde o volume de água armazenado varia em função do tempo (RENNÓ, 2004). Esta variação é chamada de balanço de água, e pode ser calculada por meio de todas as entradas e saídas do sistema. A precipitação é a principal entrada de água no sistema, parte desta é interceptada e outra atinge o solo. Uma parcela da água que chega ao solo é infiltrada e outra escoa superficialmente, sendo que durante o período em que existe água na superfície da terra, ela está sendo evaporada.

A utilização de informações com associação espacial como entrada para modelos hidrológicos possibilita o incremento das unidades espaciais de estudo em número e em detalhes (FERRAZ, 1999). O uso de ferramentas SIG (sistema de informação geográfica) têm se destacado nos últimos anos em aplicações de

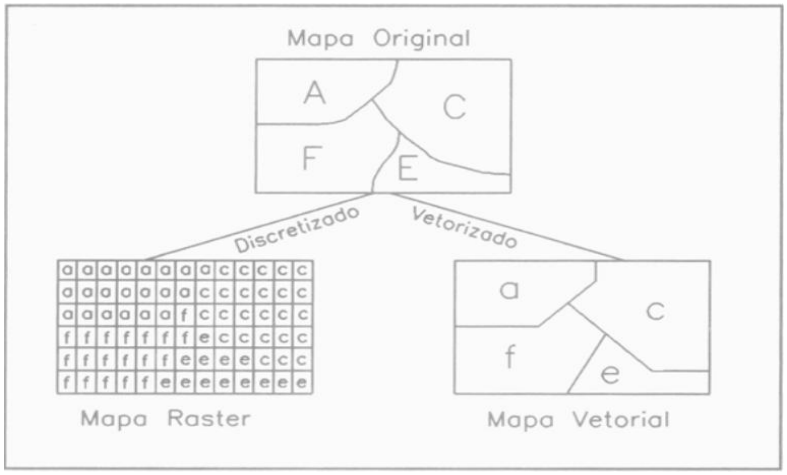
modelagem hidrológica. Estes possibilitam, por meio de análise topográfica de vários aspectos do solo a determinação de parâmetros hidrológicos.

Segundo Ferraz (1999), uma ferramenta SIG é um software desenvolvido para armazenar, apresentar e manipular informações espacialmente distribuídas. Um sistema de informação geográfica possibilita que informações provenientes das mais distintas fontes sejam integradas em uma única base de dados. Para Fundamento de Geoprocessamento (2013), a capacidade que um SIG tem de estabelecer relações espaciais entre elementos gráficos é o que o distingue dos demais sistemas e o coloca como o mais adequado para a análise espacial de dados geográficos. Além do banco de dados contendo informações geométricas e espaciais, outras bases de dados podem ser utilizadas para abrigar atributos alfanuméricos e gráficos. O primeiro geralmente é associado ao segundo e contém informações descritivas sobre este. Ferramentas SIG permitem o desenvolvimento de programas que têm poder de executar exames nas bases geográficas e numéricas simultaneamente, pesquisando informações e associando a entidades gráficas, assim como o contrário também é possível.

Conforme Fundamento de Geoprocessamento (2013), as formas *vetorial* e *matricial* (Figura 5), são os dois principais modos de representação de dados espaciais em um SIG. No formato vetorial, mapas são abstrações gráficas compostas por pontos, linhas e polígonos. Elementos estes representados como um conjunto de pares de coordenadas (x,y). Pontos são descritos com apenas um par coordenado, e as linhas e polígonos por uma sequência de pares de pontos. Também conhecido como *raster,* o formato matricial descreve o mapa por meio de uma matriz bidimensional composta por células, também chamadas de *pixel.* A cada um dos pixels, valores que representam a medição de alguma grandeza física são associados.

O modelo *raster* possui uma estrutura de dados mais simples, o que facilita a visualização e manipulação de dados. Porém, apresenta como desvantagens o elevado espaço de armazenamento consumido e a exigência de uma resolução elevada para se obter um resultado satisfatório. O formato vetorial explora muito bem o espaço de memória do computador, e representa com maior facilidade relações topológicas. Em contrapartida, sua estrutura mais complexa geralmente exige software e hardware mais caros (FUNDAMENTO DE GEOPROCESSAMENTO, 2013).

Figura 5 – Formatos de representação de informações espaciais.



Fonte: Fundamento de Geoprocessamento, 2013, p. 8

De acordo com Chaves (2002), o relevo é o principal responsável pelo controle dos fluxos de água e energia em uma bacia hidrográfica, e o formato digital que o descreve é denominado modelo digital de elevação (MDE). Tanto a forma vetorial quanto a forma matricial podem ser utilizadas em um MDE, sendo o primeiro utilizado com maior frequência. Em modelos digitais de elevação construídos no formato *raster*, a informação armazenada em dada célula da matriz representa a altitude do solo naquele ponto, formando assim, uma representação tridimensional por meio de uma matriz bidimensional.

Nos últimos anos vários módulos foram desenvolvidos em ferramentas SIG para automatizar o processo de extração de informações a partir de MDE. A partir dos MDE, podem-se calcular volumes, áreas, desenhar perfis e seções transversais, gerar imagens, mapas de declividade e exposição, perspectivas tridimensionais, entre outros (FERRAZ, 1999). Um dos principais parâmetros hidrológicos determinados por meio da utilização de modelos digitais de elevação é a identificação da rede de drenagem de uma bacia hidrográfica. Para tanto, Chaves (2002) apresenta uma abordagem que define a rede de drenagem a partir da simulação do caminho preferencial de escoamento sobre o MDE *raster*. Neste, a direção de escoamento superficial é definida pelo método determinístico de 8 células (D8), o qual analisa apenas as 8 células vizinhas na definição da direção de drenagem, simplesmente escolhendo a célula de maior declive neste conjunto e atribuindo uma codificação numérica em um novo arquivo *raster* de mesma resolução. A Figura 6 apresenta todo o processo de extração da rede de drenagem

com limiar zero, situação onde não existe absorção, ou seja, todo o material que cai em uma célula escoa para a célula vizinha de maior desnível. O fluxo acumulado em uma célula é obtido por meio da adição das células localizadas acima da em questão.

Com base nos parâmetros descritos acima, é possível encontrar outros, como a área da bacia e o comprimento dos canais. Por exemplo, dados de precipitação associados a imagens de uso do solo permitem determinar o escoamento superficial, assim como a utilização de dados da literatura possibilite estimar o fluxo de poluentes oriundos de um rio ou represa (Fundamento de Geoprocessamento, 2013).

Existem três principais fontes de dados altimétricos, para os quais três diferentes técnicas de geração de modelos digitais de elevação são aplicáveis. São estas: pontos cotados, curvas de nível e sensoriamento remoto (FERREIRA, 2006). No primeiro, a interpolação de pontos cotados (incluindo pontos altos e baixos) forma o conjunto de dados abrigado no MDE. Um método de aquisição de dados muito utilizado nos dias de hoje é o de aquisição via GPS, denominado cinemático. Nele o terreno é percorrido com uma antena portátil, ligada a um coletor de feições. Curvas de nível são representadas por um mapa de isolinhas, podendo ser extraídas de mapas topográficos existentes ou de estereomodelos fotogramétricos, que ainda hoje, são a fonte de dados altimétricos mais comum quando se busca modelar grandes áreas. A interpretação de dados coletados por sensores em nível aéreo e orbital pode fornecer valores altimétricos para modelos digitais de elevação. A fonte de dados tradicional é a fotografia aérea, que quando não limitada pela existência de cobertura vegetal no solo, é capaz de fornecer altitudes com precisão inferior a um metro.

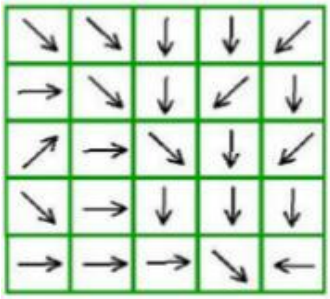
Figura 6 – Processo de extração da rede de drenagem a partir de um MDE.

67	56	49	46	50
53	44	37	38	48
58	55	22	31	24
61	47	21	16	19
53	34	12	11	12

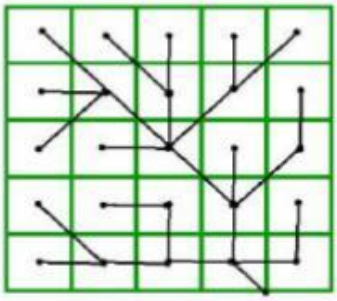
(a) MDE *raster*

2	2	4	4	8
1	2	4	8	4
128	1	2	4	8
2	1	4	4	4
1	1	1	2	16

(b) Codificação das direções de escoamento



(c) Direção do escoamento



(d) Drenagem numérica com limiar 0

Fonte: Chaves, 2002, p. 38

O modelo de otimização selecionado para o desenvolvimento do software objeto de estudo deste trabalho é do tipo hidrológico, uma vez que representa o fluxo de água sobre a área da lavoura. A área em questão limita o sistema (espaço de estudo) do modelo de otimização, de maneira que o reservatório de água (barragem) é a única fonte de água considerada neste sistema. A principal entrada do sistema é um modelo digital de elevação no formato *raster*, isto é, mapa cuja informação portada é o relevo da lavoura.

Quanto as variáveis utilizadas, o modelo selecionado é classificado como estocástico, já que é o software é baseado na técnica de algoritmos genéticos e, portanto, o mesmo conjunto de variáveis de entrada pode produzir saídas distintas. Quanto à relação entre variáveis, este é categorizado como empírico, pois algumas

relações são baseadas em observações e nem todos os processos envolvidos são descritos por meio de equações. O fato de se tarar de um modelo computacional obrigatoriamente o classifica como discreto, posto que computadores não tem capacidade de manipular um conjunto contínuo de dados. Como um algoritmo genético é formado por um processo iterativo de evolução das soluções, modelos de otimização que empregam esta técnica são ditos dinâmicos, dado que a entrada de uma iteração coincide com o resultado da iteração anterior.

2.3 Algoritmos genéticos

Para Aguiar (1998), problemas podem ser descritos como tarefas a serem resolvidas, as quais podem ser descritas como objetos matemáticos, usualmente funções. Um algoritmo consiste em uma série de procedimentos e atividades que, posicionadas de uma forma estruturada, constituem um método para solucionar um problema. Segundo Pescador (2009), problemas podem ser divididos quanto a sua natureza em dois tipos: o primeiro abriga os problemas de decisão, que são aqueles que apresentam solução binária, ou seja, o resultado para o problema consiste em uma resposta do tipo “sim ou não”; já no segundo tipo estão problemas de otimização, os quais buscam encontrar uma estrutura que satisfaça a um conjunto de propriedades de maneira a minimizar ou maximizar algum valor. De acordo com Aguiar (1998), problemas de otimização levam a soluções mais difíceis pois os elementos básicos são os mesmos presentes nos problemas de decisão, apenas acrescidos de uma função que define o quão boa é determinada solução.

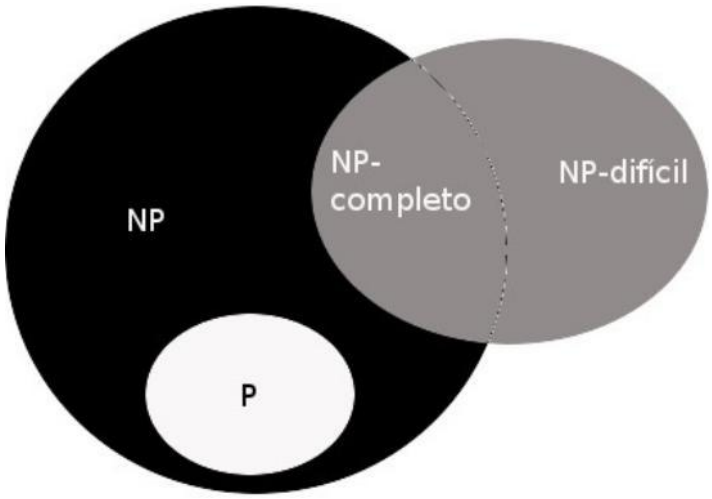
Conforme Aguiar (1998), desenvolver uma solução para o problema não basta, é necessário compreender a complexidade associada ao mesmo, estando esta ligada aos recursos necessários para que um algoritmo possa resolver o problema. A medida mais importante da complexidade é o tempo, utilizado com frequência para determinar se o algoritmo é eficiente a ponto de oferecer utilidade prática. Uma instância para um problema é constituída de um conjunto de valores particulares especificados para todos os seus parâmetros.

De acordo com Lewis (2004) uma máquina de Turing consiste em um modelo capaz de obter qualquer forma aceitável de expressões contidas em um algoritmo. Existem problemas computáveis (ou decidíveis) e não computáveis (ou indecidíveis). Um problema é dito não computável quando não é possível resolvê-lo utilizando uma

máquina de Turing determinística. Já os computáveis, são aqueles computáveis por alguma máquina de Turing. Para problemas indecidíveis não existe nenhum algoritmo que o solucione, o que não ocorre com os decidíveis (recursivos) que são subdivididos em classes de complexidade. As classes de complexidade de maior interesse prático são P, NP, NP-Completo e NP-Difícil.

P é classe em que se encontram os problemas que podem ser resolvidos por algoritmos em tempo polinomial, ou ainda, a classe de problemas que podem ser resolvidos por uma máquina de Turing determinística em tempo polinomial. A classe NP engloba problemas solúveis em tempo polinomial por uma máquina de Turing não determinística, ou seja, estes problemas podem ser verificados em tempo polinomial mesmo que não se conheça algoritmo de tempo polinomial que os resolvam. Se um dado problema X pertencente à classe NP, e todos os demais problemas contidos nesta classe (NP) podem ser reduzidos em tempo polinomial a X, então X é NP-completo. Um problema NP-completo também pertence à NP (a classe dos NP-completos está contida em NP), e é ao menos tão difícil quanto todos os demais problemas pertencentes a esta classe, de maneira que a classe dos NP-completos é a dos mais difíceis em NP. A classe dos NP-difíceis contém problemas de dificuldade no mínimo igual aos pertencentes à classe dos NP-completos, com o diferencial de que não podem ser verificados em tempo polinomial por uma máquina de Turing determinística, o que significa que esta classe não está contida em NP (Figura 7).

Figura 7 – Relacionamento entre classes de complexidade.



Fonte: Pescador (2009), p. 9

Segundo Melo (2008), problemas de otimização focam a maximização ou a minimização de um ou mais parâmetros, aparecendo com frequência em projetos técnicos. Em problemas desta natureza, o conjunto de variáveis presentes possibilita um número incontável de combinação de pontos para se chegar à solução. Solução esta que não pode ser verificada com base apenas em uma instância do problema, pois para afirmar que determinada solução é a melhor é necessário analisar todas as possibilidades. Portanto, em geral, problemas de otimização não são verificáveis em tempo polinomial, e como não pertencem à NP são classificados como NP-difíceis. Como não é possível encontrar a melhor solução, estratégias são utilizadas para o tratamento desta categoria de problemas, dentre estas se destaca o ramo da computação evolutiva.

Conforme Zumbem (2010) a seleção natural é responsável pela distinção entre os sistemas biológicos e os demais sistemas físicos e químicos. A teoria evolucionária (da seleção natural) proposta pelo inglês Charles Darwin, em 1859, buscou esclarecer a forma como se dá a adaptação dos indivíduos, fato não explicado pela teoria lamarckista, única alternativa até então. Para Lamarck as características adquiridas pelos indivíduos ao decorrer da sua vida eram herdadas pelos seus descendentes diretos, teoria que mais tarde foi considerada inviável por meio de um trabalho publicado no final do século XIX. Já Darwin sugeriu que uma associação entre seleção natural e diversidade explicaria melhor o processo evolucionário.

Segundo Zumbem (2010), a adaptação ao meio é conduzida pelas variações sucessivas ocorridas junto aos organismos de uma espécie. O princípio da seleção natural é o de que os indivíduos cujas variações se adaptam melhor ao ambiente terão maior probabilidade de sobreviver e se reproduzir. Para explicar este processo, Charles Darwin apresentou algumas hipóteses, as quais somente foram aceitas para explicar os processos adaptativos em nichos ecológicos. Para que esta teoria explique a origem das espécies, é necessário se adicionar outras hipóteses, conduzindo ao neodarwinismo. Dentre estas estão as de que existe algum processo capaz de inserir novas informações junto à carga genética dos indivíduos, e a preservação das informações se dá através da seleção natural, mecanismo que prioriza os organismos mais aptos. Sendo assim, a seleção natural é um processo probabilístico e o efeito da evolução se manifesta na espécie como um todo, embora o alvo primário seja o indivíduo.

Vários são os motivos que justificam o uso do processo evolutivo como estratégia para a resolução de problemas computacionais. Mitchell (1996) destaca o elevado número de possíveis soluções e a necessidade que alguns programas apresentam de se adaptarem a um ambiente em mudança. A computação evolutiva é uma técnica capaz de realizar uma pesquisa de forma massivamente paralela em uma população muito grande de soluções, oferecendo uma resposta de alta qualidade para problemas difíceis.

De acordo com Zumbem (2010), existem diversos métodos computacionais baseados na teoria evolutiva, sendo os algoritmos genéticos, as estratégias evolutivas e a programação evolutiva os principais. Contudo, existe uma estrutura básica comum entre as abordagens existentes dentro da computação evolutiva. Todo o algoritmo evolutivo mantém uma população de indivíduos *P(t)*, onde *t* é a iteração ou *geração*. Cada indivíduo é uma possível solução para o problema e é abrigado por uma estrutura de dados. A cada geração, a população é avaliada indivíduo a indivíduo por uma função de adaptação, também chamada de *fitness*. Uma nova população é gerada para a próxima iteração do algoritmo, processo que emprega *operadores genéticos* e privilegia os mais aptos da geração atual. Estes operadores submetem alguns indivíduos da população a um processo de alteração, através de mecanismos como *crossover* e *mutação*. O algoritmo termina quando a condição de parada for atingida, podendo isto ocorrer de duas formas. Na primeira é verificado por meio da função de adaptação que um indivíduo alcançou um nível aceitável para ser considerado portador de uma solução para o problema. A segunda consiste simplesmente em se definir um limite máximo de iterações que quando atingido, o indivíduo mais apto da população é tomado como solução.

Os algoritmos genéticos (AG) basicamente executam um procedimento de busca que se dá em um espaço que contém algumas das possíveis soluções para o problema. Um aspecto importante deste método é o equilíbrio entre o aproveitamento da melhor solução e a exploração do espaço de procura, característica ausente nas demais técnicas de busca (ZUMBEM 2010). Segundo Soares (1997), como a busca não fica restrita ao entorno da melhor solução, a chance de se encontrar um ótimo global aumenta. Soares (1997) também salienta que, os algoritmos genéticos operam com uma população de pontos de forma paralela levando ao surgimento de subpopulações, o que dá ao método capacidade

de chegar a uma solução refinada, pois é possível aprimorar a busca em torno de uma vizinhança e não só da solução global.

Conforme Soares (1997), otimização consiste na análise de cenários, com o objetivo e encontrar a melhor solução para um problema, o que geralmente envolve seleção de valores para variáveis, decisões cuja qualidade deve ser medida e avaliada. São vários os métodos de procura existentes e a escolha do método correto é imprescindível, pois nem sempre o mesmo apresenta uma solução de qualidade. O principal fator a ser analisado durante a definição do método é a capacidade de atingir mais vezes a solução global por número de execuções, e, considerando esta métrica, os algoritmos genéticos estão entre os métodos mais eficazes.

Zumbem (2010) explica que em um algoritmo genético, para cada indivíduo existe um *cromossomo* que o representa, o qual abriga a codificação (*genótipo*) de uma possível solução do problema (*fenótipo*). Os *cromossomos* são implementados na forma de estruturas de dados, geralmente como vetores ou listas encadeadas, onde cada posição (atributo) é chamada de *gene*. *Alelo* é o nome dado a cada valor que um gene pode assumir. Um AG desenvolvido para um problema particular contém os seguintes atributos (ZUBEN, 2000):

- uma forma para criação da população inicial de soluções candidatas;
- um formato de cromossomo que represente a solução;
- uma função de adaptação que avalie o quão apto uma solução é ao ambiente e retorne um valor associado a essa capacidade;
- operadores genéticos;
- valores para os parâmetros do algoritmo (tamanho da população, probabilidades de aplicação dos operadores genéticos, etc.).

Na maioria das vezes a população inicial é gerada de forma aleatória, mas Bueno (2010) salienta que existem outras formas de se definir o conjunto de indivíduos. Estratégias mais orientadas à solução podem ser utilizadas como uma forma de se acelerar o processo de convergência caso exista algum conhecimento prévio sobre o problema. O principal cuidado ao se selecionar um método é garantir que o mesmo gere apenas indivíduos válidos.

A forma mais utilizada para se representar indivíduos em algoritmos genéticos é por meio de codificação binária mapeada no formato de um vetor. Porém, existem

outros formatos que podem ser aplicados na representação de soluções, como representações baseadas em números reais, permutações de inteiros e árvores (BUENO, 2010). Zumbem (2010) ressalta que a etapa de codificação é crítica na definição de um AG, uma vez que a declaração incorreta da solução pode ocasionar uma convergência prematura do algoritmo. A estrutura do cromossomo deve contemplar todos os aspectos da solução da forma mais simples possível.

Responsável por direcionar o AG na busca pela solução ideal, o desenvolvimento da função de adaptação requer atenção. A métrica da aptidão é expressa pela função objetivo e reflete a qualidade de uma solução no contexto de um problema (BUENO, 2010). A cada rodada esta função atribui um valor a cada elemento da população, tornando possível a comparação entre estes.

Para Melo (2008) a seleção dos melhores indivíduos é a base da programação evolutiva, sendo o critério de seleção adotado de extrema importância para a otimização das soluções. A seleção é uma tarefa que a princípio é simples, pois se baseia no resultado da função de aptidão. Mas o aumento da população torna a atividade computacionalmente custosa, justificando o esforço na busca de métodos eficientes. A *roleta* é uma das principais técnicas de seleção, e sua denominação remete a gráficos do tipo torta ou a de uma roleta de cassino, onde o espaço ocupado por cada indivíduo é proporcional à sua probabilidade. A razão entre a aptidão de um indivíduo e a soma total das aptidões da população define a probabilidade deste ser selecionado. O método tem como desvantagens a necessidade da ordenação da população e a tendência de convergência prematura. Outra técnica bastante utilizada é a da *seleção por torneio*, nesta uma determinada quantidade de cromossomos é selecionada aleatoriamente, escolhendo aquele de maior aptidão dentre os que formam este conjunto. Este método não necessita de prévia ordenação.

Silva (2006) citado por Melo (2008) destaca o ordenamento e o escalonamento como os principais métodos para atribuição da aptidão dos indivíduos em algoritmos genéticos. Melo (2008) apresenta o método de ordenamento como a atividade de dispor os cromossomos em uma sequência baseada nos valores obtidos com auxílio da função de adaptação, podendo esta ser de forma crescente ou decrescente. Se o problema está em determinar o menor valor da função objetivo os cromossomos mais aptos serão aqueles de menor valor, e os cromossomos são ordenados em ordem crescente, caso contrário, a ordem

decrescente é utilizada. Já no escalonamento, uma estratégia para considerar a variabilidade dos indivíduos da população é empregada. Uma equação baseada na pressão de seleção (razão entre a aptidão máxima e a aptidão média) e função objetivo aproxima o valor da aptidão dos indivíduos, dificultando sua distinção e seleção.

Aos operadores genéticos cabem as modificações efetuadas sobre os indivíduos, as quais geram novos elementos para compor a população da próxima rodada. Os operadores genéticos mais utilizados nos algoritmos genéticos são o *crossover* e a *mutação*. De acordo com Bueno (2010) o *crossover* simula o processo de troca genética que ocorre na natureza por meio da recombinação de partes dos pais, produzindo assim novos indivíduos. O operador de *crossover* mais usado é o de um ponto, onde um ponto aleatório é selecionado nos dois cromossomos (o mesmo ponto para os dois) envolvidos na operação. Os segmentos (um de cada cromossomo) são concatenados de forma a originar um novo elemento para a população. Existem outros tipos de crossovers, dentre estes o de dois pontos, mas de maneira geral não existe um que se destaque quanto ao desempenho. Desta forma, a eficiência de um operador depende da classe de problemas ao qual ele é aplicado (ZUBEN, 2000).

Objetivando promover a diversidade genética da população, o operador de *mutação* altera aleatoriamente um ou mais genes dos indivíduos gerados pelo *crossover*. A probabilidade de sua ocorrência é denominada taxa de *mutação* e está sempre associada a valores muito baixos (cerca de 1%). O mecanismo de *mutação* está ligado à estrutura de dados do indivíduo. No caso da codificação em ponto flutuante, os operadores mais populares são a *mutação* uniforme e a *mutação* gaussiana (ZUBEN, 2000).

A população da nova geração é formada pela etapa de substituição dos cromossomos, que sucede os processos de *crossover* e *mutação* (MELO, 2008). Um dos métodos aplicados nesta fase é o de substituição geracional, onde os pais serão substituídos inteiramente por seus filhos durante a formação de uma nova geração. Essa estratégia resulta em elevado risco de perda de boas soluções para o problema, o que torna seu uso proibitivo em algoritmos com elevadas taxas de *crossover* e *mutação*. A ameaça do acréscimo de indivíduos com pior aptidão no sistema é reduzida com o uso do método *steady state*, onde apenas um determinado número de soluções pertencentes à população atual são substituídos

por novos indivíduos. Na escolha pelos que serão substituídos, os que possuírem menor aptidão e maior longevidade (gerações sem alteração) recebem prioridade.

Dentre os principais aspectos envolvidos na criação de um algoritmo genético para a otimização de um problema real Melo (2008) ressalta o conhecimento especializado sobre o problema em estudo, pois nele está a base para a definição da função objetivo e dos parâmetros do algoritmo genético. O desenvolvimento do algoritmo depende da representação apropriada dos operadores genéticos correspondentes à representação escolhida, uma vez que o que diferencia os algoritmos genéticos de outras estratégias evolutivas é o predomínio de operadores na formação de novos indivíduos.

3 DESENVOLVIMENTO

O algoritmo de otimização apresentado no decorrer deste trabalho consiste em um algoritmo genético, que se baseia nos resultados apontados pelo modelo hidrológico elaborado para direcionar sua busca pela solução ideal. O modelo hidrológico consiste em uma representação computacional dos aspectos físicos relacionados ao sistema de irrigação por inundação, aplicado especificamente a lavouras de arroz. Desta forma, o modelo busca representar as principais características relacionadas ao fluxo e acúmulo superficial de água em lavouras de arroz, bem como a relação entre os componentes do sistema e os seus respectivos dimensionamentos. O algoritmo genético realiza uma varredura ampla e direcionada em um espaço onde estão potencialmente localizadas todas as soluções (as quais são representadas em cromossomos) para o problema em questão. Durante o processo de busca, um subconjunto das soluções contidas neste espaço é avaliado com o objetivo de encontrar a que melhor se aplica ao problema. Durante o processo de avaliação, o algoritmo desenvolvido dimensiona o sistema de irrigação contido no cromossomo, e atribui ao mesmo uma nota para cada um dos aspectos por ele considerados. A equação de aptidão do algoritmo genético é composta pelo conjunto de aspectos previamente avaliados como sendo relevantes no projeto de uma lavoura de arroz, sendo que cada um destes possui um coeficiente (influência ou peso na equação) a ele associado. O somatório de todos os aspectos resulta no valor de aptidão do indivíduo para com o a solução do problema, concluindo assim o processo de avaliação da solução.

Várias são as características físicas e lógicas reproduzidas no modelo, de maneira que seu desenvolvimento ocorreu de forma iterativa, fazendo uso da linguagem de programação *python* (PYTHON, 2014). Acompanhando o progresso da atividade de implementação do software, soluções mais eficientes para a representação das características funcionamento do sistema surgiram. Neste capítulo o desenvolvimento do algoritmo de otimização será descrito, bem como as principais decisões tomadas ao decorrer deste processo e os resultados obtidos com sua utilização.

3.1 Codificação do problema

O algoritmo de otimização se restringe à otimização da fração do sistema de irrigação contido dentro da área do mapa. Isto quer dizer que os componentes do sistema de irrigação e os fenômenos físicos presentes no caminho percorrido pelo fluido entre o reservatório (fonte de água) e a lavoura, assim como o sistema de drenagem, fogem ao escopo deste trabalho. Condições de precipitação também são desconsideradas, uma vez que o foco do algoritmo não é o de estimar o consumo total da irrigação, e sim o de auxiliar no projeto do sistema de irrigação (o qual deve ser capaz de suportar situações de escassez de chuva). Visto que a precipitação usualmente ocorre em toda a lavoura, essa tem efeito na altura da lâmina d'água, mas não na distribuição da água no espaço. Como referência ao longo do texto, o canal que conecta a área cultivada ao reservatório é denominado *canal primário* e o canal interno ou adjacente à lavoura (foco do algoritmo) é chamado de *canal secundário*.

Para facilitar o entendimento sobre o funcionamento do algoritmo, é conveniente, em um primeiro instante, visualizá-lo como uma caixa preta (Figura 8), apresentando apenas suas interações com o ambiente externo. Sendo assim, um conjunto de dados é fornecido como entrada do algoritmo de otimização que, por sua vez, retorna a solução para o problema em sua saída.

Figura 8 – Interação entre usuário e o algoritmo de otimização.

Fonte: O autor

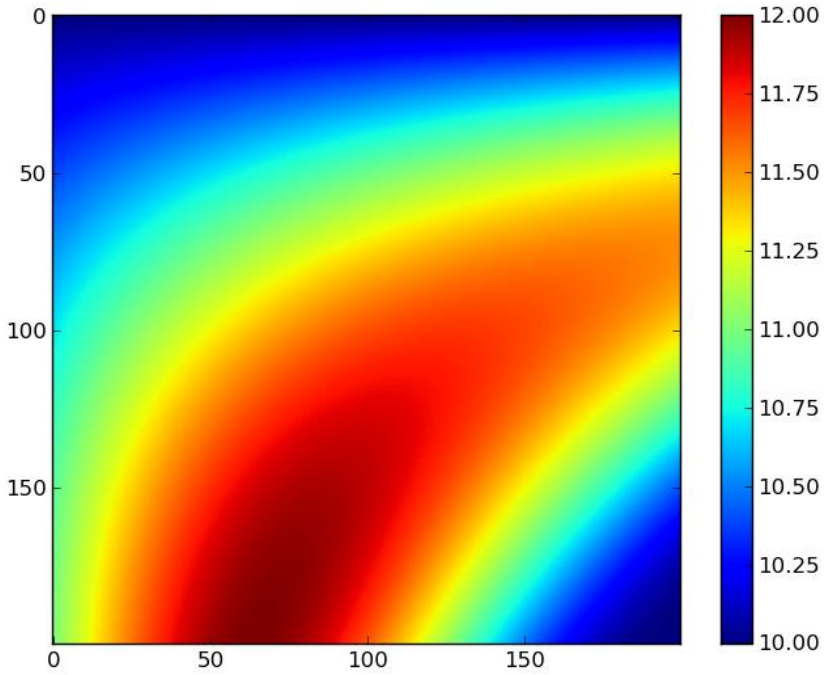
As informações que precisam ser fornecidas pelo usuário ao software, funcionando como entradas do mesmo, são:

- *mapa da área cultivada*;
- *parâmetros agronômicos*;
- *parâmetros do algoritmo genético*.

A informação contida no *mapa da área cultivada* é o relevo, abrigado em um modelo digital de elevação (MDE) no formato *raster*. Este modelo raster (Figura 9) é implementado por meio de uma matriz bidimensional, onde cada célula armazena a *cota* (nível altimétrico) em um dado ponto do terreno. São especificados

paralelamente ao MDE a variação altimétrica máxima admitida pelos tabuleiros (*dh*) e a posição no mapa do ponto de contato entre a lavoura e o canal primário.

Figura 9 – Modelo digital de elevação *raster.*



Fonte: O autor

Fazem parte dos *parâmetros agronômicos*, os períodos de formação e manutenção da lâmina d'água (características que dependem da variedade da semente cultivada), bem como a profundidade a ser mantida durante o período de manutenção da lâmina. O período de funcionamento diário das moto-bombas também é definido pelo usuário, uma vez que existem incentivos financeiros por parte das concessionárias de energia para sua utilização em horários pré-determinados. Outros aspectos descritos na Seção 2.1, como taxa de infiltração (taxa de percolação mais fluxo lateral), taxa de evapotranspiração, coeficiente de escoamento em canais e coeficiente de rendimento dos levantes também devem ser fornecidos ao modelo.

Os *parâmetros do algoritmo genético* são: taxa de *crossover*, taxa de *mutação*, tamanho da população e o número de gerações. As taxas de *crossover* e *mutação* correspondem ao percentual de indivíduos submetidos a alterações por operadores genéticos por iteração. O último parâmetro genético do modelo é o número de gerações (iterações) que o mesmo deve realizar, sendo o único critério

de parada do algoritmo. Estes valores, associados ao tamanho da população, influenciam diretamente o desempenho do algoritmo, exigindo cautela na sua seleção. As demais variáveis de entrada do software serão apresentadas posteriormente, pois estão relacionadas às soluções computacionais tomadas no decorrer do projeto e sua compreensão demanda um entendimento prévio referente ao funcionamento do software.

São entregues como saída do algoritmo as seguintes informações:

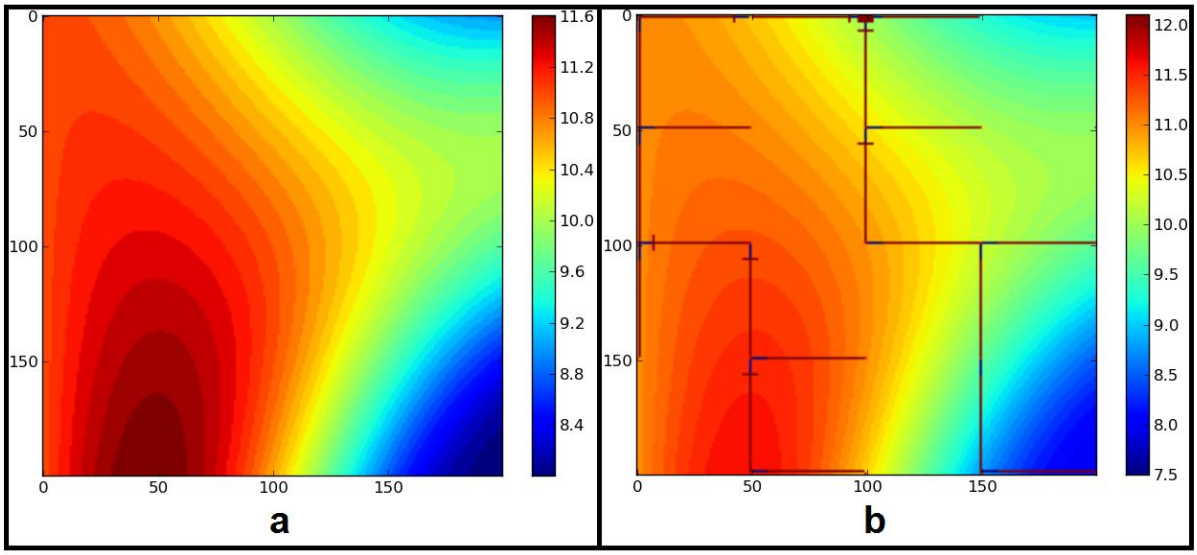
- *mapas das soluções*;
- *relatórios das soluções*;
- *relatórios evolutivos*.

Os *mapas das soluções* gerados pelo modelo são do tipo *raster* e estão divididos (de acordo com a informação portada pelos mesmos) em dois grupos. O primeiro (Figura 10-a) é criado assim que a execução do software tem início, e apresenta a área sistematizada, ou seja, dividida em quadros de irrigação. O segundo (Figura 10-b) é criado a cada repetição do processo evolutivo do programa (apenas para o indivíduo com maior valor de aptidão da iteração) e demonstra a localização espacial dos principais componentes do sistema. Neste, o *relevo* (variação das cotas) é representado pela coloração das células, a *divisão entre quadros* por meio da brusca alteração de cotas, o *ponto de entrada d'água* mediante um retângulo (localizado na parte superior da Figura 10-b) e os *segmentos de canal* (frações ou partições do canal secundário) por meio de linhas horizontais ou verticais. A *direção de escoamento* de cada segmento de canal é reproduzida partindo da linha azul (cota 7.5 na Figura 10-b) em direção à vermelha (cota 12 na Figura 10-b). A *existência de um levante em um seguimento* de canal é representada por uma linha transversal ao mesmo.

Os relatórios gerados pelo algoritmo de otimização são armazenados em planilhas eletrônicas no formato *xls (Microsoft Excel)*. Os *relatórios das soluções* encontradas apresentam descrições detalhadas sobre as soluções expostas pelos mapas (Figura 10-b), o que é feito por meio de: conexões entre elementos (ponto de entrada d'água, canais, quadros e levantes), dimensionamento dos componentes (área de secção transversal dos canais e potência dissipada pelas bombas), estimativas de vazões, estimativas de consumo energético, entre outros. Já os *relatórios evolutivos* retratam o avanço das soluções em direção a um sistema de irrigação mais eficiente, o que é descrito por meio dos valores encontrados para as

variáveis de aptidão. A fonte dos indivíduos de cada geração (população inicial, *crossover* ou *mutação*) bem como gráficos apresentando a variação dos valores calculados para as variáveis de aptidão ao decorrer do processo evolutivo, completam o conjunto de informações geradas pelo software.

Figura 10 – Mapas gerados pelo algoritmo de otimização.



Fonte: O autor

Concluindo a exibição do algoritmo por meio de sua interação com o usuário, é chegada a hora de abrir a caixa preta e descrever os componentes de software que o constituem. Sendo assim, a primeira atividade que o algoritmo realiza logo que sua execução tem início consiste na sistematização da área de cultivo, dividindo o mapa fornecido pelo usuário em quadros de irrigação. Um quadro de irrigação do ponto de vista computacional nada mais é do que um conjunto de células adjacentes que estão dentro de uma determinada faixa variação altimétrica (*dh*). A atividade de sistematização é dividida em duas etapas: *divisão de cotas* e *distinção de quadros*.

A etapa de *divisão de cotas* consiste no processo de aglutinação das células que estão dentro do mesmo intervalo de variação altimétrica, intervalos estes que são definidos a partir da cota máxima do mapa (*max*) e possuem faixa de abrangência de tamanho *dh*. Desta maneira, a célula cuja cota (*h*) estiver localizada no intervalo de ($max - dh$) < h < max passa a ter o valor médio deste intervalo ($max - dh/2$) como informação por ela armazenada. Este processo (exemplificado por meio

da Tabela 2) é repetido para todas as células da matriz, e gera ao seu final o *mapa de cotas* (localizado na parte superior direita da Figura 11).

Contudo o *mapa de cotas* não caracteriza área sistematizada, pois pode existir em um mapa mais de um quadro de irrigação com o mesmo valor de cota. De maneira que para o funcionamento correto do algoritmo deve ser possível a realização da distinção entre todo e qualquer tabuleiro presente na lavoura. Logo, o algoritmo de *distinção de quadros* efetua uma varredura no *mapa de cotas*, atribuindo um valor idêntico (referência para o quadro) a todas as células adjacentes que compartilhem do mesmo nível altimétrico (cota), chegando assim ao *Mapa da Área Sistematizada* (mapa localizado na parte inferior da Figura 11).

Tabela 2 – Cálculo da faixa de variação de cotas.

Condição	Atribuição
(*max* – **1**.*dh*) < *h* < (*max* – **0**.*dh)*	*h* = *max*- **1**.*dh*/2
(*max* - **2**.*dh*) < *h* < (max – **1**.dh)	*h* = *max*- **2**.*dh*/2
(max - **3**.dh) < h < (max - **2**.dh)	*h* = *max*- **3**.*dh*/2
...	**...**

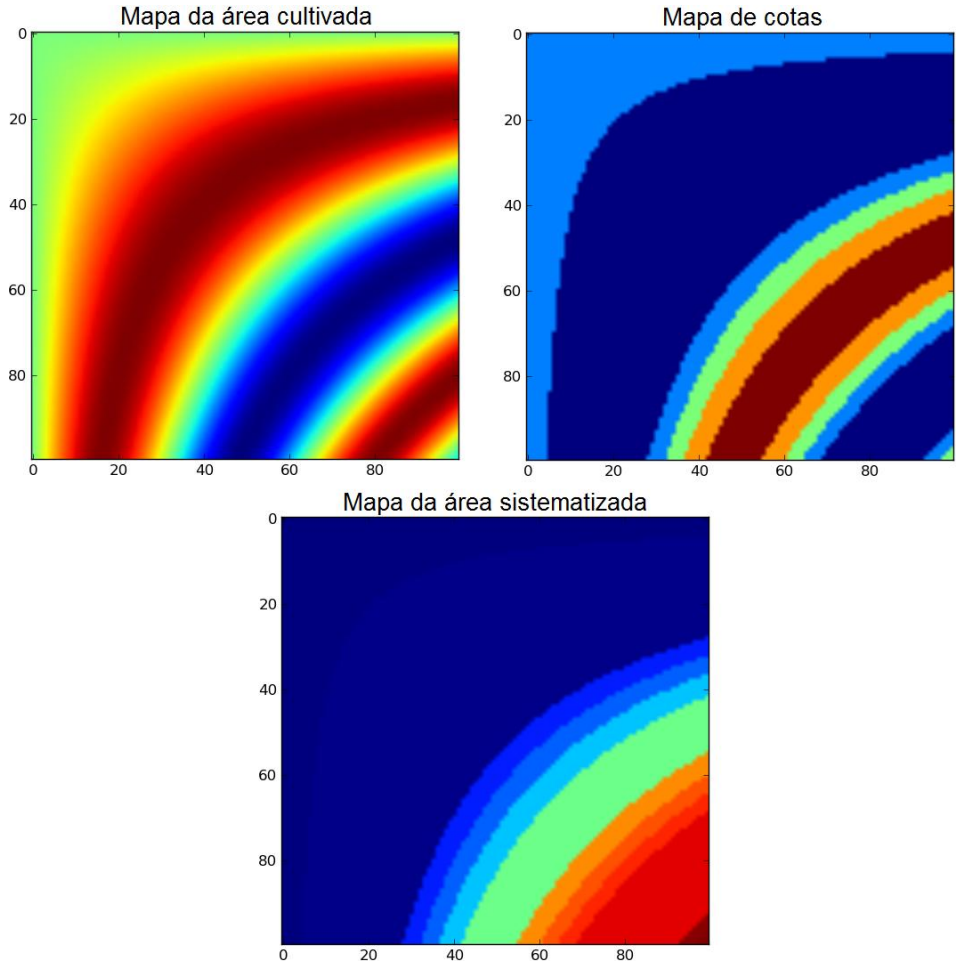
Fonte: O autor

A Figura 11 apresenta os mapas resultantes de cada uma das etapas do processo de sistematização da área de cultivo. No *mapa da área cultivada* (fornecido pelo usuário) é visível a transição suave de cotas (representado por cores na Figura 11) no relevo natural da área. Já no *Mapa de Cotas* é nítida a passagem de um nível altimétrico para outro, este que contém as curvas de nível da área e consequentemente o formato final assumido pelos tabuleiros. Porém, é possível observar neste mapa a existência de quadros com mesma cota (cor), diferenciação efetuada durante a criação do *Mapa da Área Sistematizada*, que é o produto final deste processo.

Ao analisar o mapa da solução final apresentado pela Figura 10-b é possível visualizar quadros com colorações repetidas, isto acontece por que o mapa utilizado como base para a reprodução da solução apontada pelo algoritmo genético é o *mapa de cotas*. Esta escolha é justificada pelo fato de este portar a informação de relevo, o que não ocorre com o *mapa da área sistematizada*. O mapa final da atividade de sistematização apenas é utilizado pelo algoritmo para a distinção entre

quadros e, portanto, não é o mais indicado para a apresentação ao usuário do sistema de irrigação completo.

Figura 11 – Processo de sistematização do modelo digital de elevação.



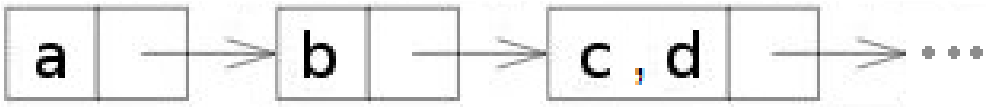
Fonte: O autor

Sem dúvida, um dos fenômenos mais importantes a serem modelados é o processo de propagação da água entre os elementos do sistema de irrigação. Basicamente, o modelo considera duas formas de transição da água: do canal secundário para um quadro de irrigação e de um quadro para outro. No caso da passagem de fluido entre quadros, existem três aspectos a contabilizar. O primeiro é a adjacência, de modo que a água só transita de um quadro para outro se estes estiverem em contato. A cota dos tabuleiros envolvidos neste processo é o segundo aspecto a ser considerado, pois para que a água localizada em um dado quadro *a* escoe para o quadro *b*, o segundo necessita obrigatoriamente estar em uma cota inferior à do primeiro. E o terceiro e último ponto a computar é a ordem de

distribuição, o que é modelado por meio da distância entre o quadro a ser irrigado e o canal secundário (fonte de água principal). Para exemplificar este aspecto, pode-se imaginar a situação onde em um tabuleiro à jusante (cota inferior) possui mais de um quadro à montante (cota superior) como adjacente. A qual quadro à montante deve ser atribuído à responsabilidade de irrigar o quadro à jusante? O modelo responde esta pergunta calculando a distância entre o quadro à montante e o canal secundário, ou seja, o quadro à montante que possuir o menor número de quadros entre ele próprio e o canal secundário (seguindo o fluxo de água e o irriga) está a uma distância menor, e consequentemente é o encarregado de irrigar o quadro à jusante.

Para que todos estes aspectos sejam considerados, após a conclusão do processo de sistematização da lavoura, o software cria uma estrutura de dados no formato de lista encadeada, a qual é utilizada durante a atividade de cálculo da propagação de água entre os componentes do sistema de irrigação. Para cada quadro de irrigação existe uma *lista de (quadros) adjacentes*, sendo que cada elemento desta lista possui uma coleção de quadros adjacentes (direta ou indiretamente dependendo da distância) à jusante. Digamos que o quadro *a* receba água diretamente do canal secundário e seja adjacente à montante do quadro *b*, que por sua vez possui dois adjacentes diretos à jusante (*c* e *d*). Neste caso o segundo elemento da lista encadeada (Figura 12) corresponde ao conjunto de adjacentes diretos de *a* (com grau de adjacência zero), sendo o terceiro o conjunto de adjacentes indiretos de *a* (com grau de adjacência um). Este encadeamento tem sequência até que o grau de adjacência máximo pré-estabelecido pelo usuário seja atingido. Desta forma os quadros de irrigação são conectados ao canal secundário por grau de adjacência, seguindo a ordem presente nas *listas de adjacentes* dos quadros. Primeiro os quadros adjacentes ao canal são conectados, segundo os adjacentes ao primeiros, e assim sucessivamente. Este processo será descrito com maiores detalhes na Seção 3.2 desta monografia.

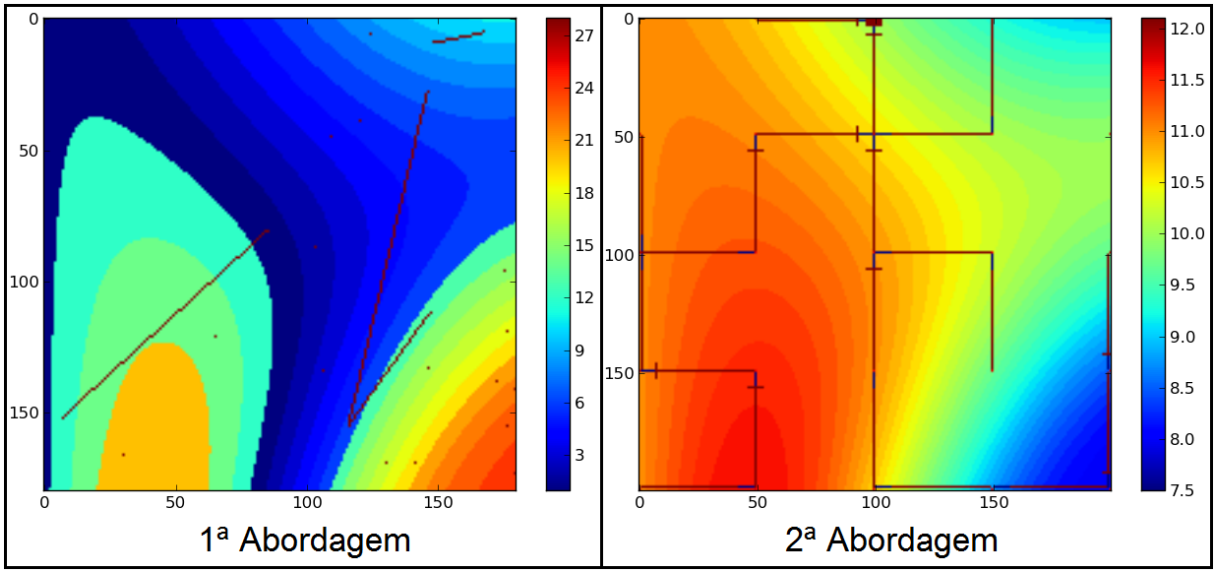
Figura 12 – Lista de quadros adjacentes ao quadro *a.*



Fonte: O autor

A forma encontrada para modelar o canal de irrigação (canal secundário) foi a de tratá-lo como um grafo, isto é, como uma coleção de vértices que são interligados por arestas (segmentos de canal). Cada vértice consiste em uma célula da matriz (ou pixel do mapa), o que exige a presença de algum critério para a seleção deste subconjunto de pontos. Primeiramente, optou-se por uma abordagem (Figura 13) onde o conjunto vértices era formado pela seleção dos pontos médios de cada quadro de irrigação (aquele sitiado o mais próximo do meio do intervalo de cotas admitido pelo quadro). Isto quer dizer que para cada quadro criado durante a etapa de sistematização, existe um vértice que pode ser conectado e fazer parte o canal de irrigação. Porém, a variabilidade espacial dos vértices intrínseca a esta abordagem propicia o surgimento de soluções onde arestas se cruzam, assim como segmentos de canais muito pequenos (ou muito grandes). Sendo assim, optou-se por uma segunda abordagem (Figura 13) onde um conjunto de pontos simetricamente distribuídos é definido, de modo que sua densidade (número total de vértices no mapa) é definida pelo usuário. A Figura 13 torna aparente outro aspecto que evoluiu juntamente com a implementação do modelo, que é a representação do sistema se irrigação por meio de mapas. Visto que elementos presentes no mapa gerado para a segunda (*relevo*, ***ponto de entrada d'água***, *direção de escoamento* e a *existência de levantes)* abordagem não podem ser visualizado no mapa da primeira.

Figura 13 – Distinção entre as duas formas de modelagem do canal de irrigação.



Fonte: O autor

Conforme descrito na Seção 2.1, a atividade de irrigação por inundação é composta por dois períodos: o período de formação e o de manutenção da lâmina d'água sobre a superfície. Segundo Mello (2009), as equações que estimam o volume e a vazão demandados pelos quadros durante o primeiro período são

$$\boldsymbol{Ve} = (\boldsymbol{Si.hl}) + (\boldsymbol{Si.ET.T1}) + (\boldsymbol{Si.Vi.T1})$$

e

$$\boldsymbol{Qe} = \frac{\boldsymbol{Ve}}{\boldsymbol{T1}}$$

onde:

Ve: volume total de água consumido pelo quadro durante o período de formação da lâmina d'água (m^3);

Qe: vazão necessária para atender a *Ve* ($m^3.s^{-1}$);

Si: área total do quadro (m^2);

hl: altura da lâmina d'água na lavoura (m);

ET: taxa de evapotranspiração média ($m.s^{-1}$);

Vi: taxa de infiltração média ($m.s^{-1}$);

T1: duração do período de formação da lâmina d'água (s).

O primeiro termo da equação que descreve o volume de água consumido (*Si.hl*) corresponde ao volume do fluído armazenado sobre a superfície do quadro. O segundo (*Si.ET.T1*) representa volume de água despendido pelo fenômeno de evapotranspiração neste período. E o terceiro (*Si.Vi.T1*) corresponde ao volume consumido pelos movimentos subterrâneos de percolação e fluxo lateral.

No período onde é feita apenas a manutenção da lâmina d'água, a equação que descreve a demanda de vazão segue o mesmo princípio do cálculo de *Ve*, porém com o diferencial da inexistência do primeiro termo da equação (desnecessário uma vez que a lâmina d'água já está formada). Deste modo as estimativas de volume e vazão são definidos por

$$\boldsymbol{Vm} = (\boldsymbol{Si.ET.T2}) + (\boldsymbol{Si.Vi.T2})$$

e

$$\boldsymbol{Qm} = \frac{\boldsymbol{Vm}}{\boldsymbol{T2}}$$

sendo:

Vm: volume total de água consumido pelo quadro durante o período de manutenção da lâmina d'água (m^3);

Qm: vazão necessária para atender a *Vm* ($m^3.s^{-1}$);

T2: duração do período de manutenção da lâmina d'água (s).

Conforme exposto na Seção 2.1, o período de formação da lâmina é o que apresenta maior vazão requerida e, consequentemente é a partir dos valores de vazão estimados para este período que o sistema de irrigação é dimensionado. É importante ressaltar que os valores de vazão (*Qi*) calculados para atender a demanda hídrica dos quadros levam em conta o período de funcionamento dos levantes (*T*), isto é, o modelo busca fornecer ao quadro todo o volume de água diário apenas no período em que as bombas estão funcionando. De modo que o valor de *Qi* é expresso por

$$\boldsymbol{Qi = Qe} \,.\frac{\boldsymbol{24}}{\boldsymbol{T}} \,,$$

considerando que *T* é dado em horas.

Muitas variáveis devem ser consideradas no problema de se implantar um sistema de irrigação por inundação em lavouras de arroz. De maneira geral, existem variáveis cujo valor expressa a aptidão que determinada solução tem para com o problema. Estas são as variáveis de saída do algoritmo e servem como métrica de desempenho, aparecendo na função de aptidão do algoritmo genético. Já as variáveis de entrada são aquelas que influenciam o resultado final e são expressas pelo cromossomo do algoritmo.

O canal de irrigação é a principal variável considerada pelo algoritmo genético, uma vez que todas as demais são derivadas da configuração do canal de irrigação. Sendo assim, nada mais óbvio que o cromossomo tenha o canal de irrigação como informação por ele carregada. Como durante o desenvolvimento do modelo duas abordagens foram elaboradas para a representação do canal, consequentemente dois formatos de cromossomo foram criados.

O cromossomo correspondente à primeira abordagem (Figura 14) é constituído de duas matrizes de adjacência, sendo que a primeira (*Canal*) contém a composição do canal secundário, e a segunda (*Levantes*) o posicionamento e a potência dos sistemas de levantes hidráulicos. O número de linhas e colunas de ambas as matrizes é exatamente igual ao número de quadros (um vértice para cada quadro), e o sentido de escoamento direcionado da linha para a coluna. A base numérica da matriz é binária, e a potência dos levantes selecionada aleatoriamente

durante o processo de criação dos indivíduos, a partir de uma fração do valor máximo de potência admitida pelo modelo.

Já na segunda abordagem (Figura 15) o cromossomo é composto apenas pela matriz de adjacência que descreve o canal posto que, durante o processo de avaliação das soluções (que será discutido na próxima seção), o software insere os levantes sempre que necessário. O formato da matriz é o mesmo da primeira abordagem, com o diferencial de possuir um número de colunas e linhas variável, obedecendo ao número de pontos (vértices) definido pelo usuário antes que a execução do software tenha início.

Figura 14 – Formato do cromossomo para a 1ª abordagem.

Canal										Levantes									
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	2.2	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	4.9	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0.6	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	7.3	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fonte: O autor

Figura 15 – Formato do cromossomo para a 2ª abordagem.

Canal									
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Fonte: O autor

No algoritmo genético, a população inicial dos cromossomos é gerada de forma aleatória, vindo a sofrer alterações por meio de dos operadores genéticos de cruzamento e *mutação*. Desta forma, inserir um elevado número de restrições quanto à criação e/ou operações genéticas aos quais os indivíduos são submetidos, além de ser uma tarefa de elevada complexidade, iria contra o princípio dos algoritmos genéticos de não limitar o espaço de busca. Sendo assim, a diversidade característica do algoritmo genético obriga a existência de indivíduos que não necessariamente correspondam a um funcionamento correto do sistema, situações que são tratadas pelo algoritmo durante o processo de avaliação do cromossomo.

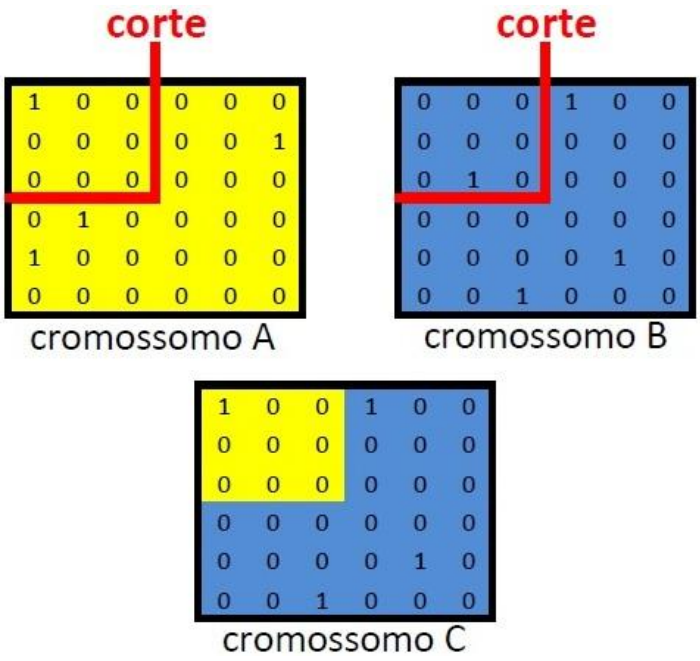
A criação da população inicial é a etapa que antecede o ciclo evolutivo do algoritmo genético. O tamanho da população obedece a um valor predefinido pelo usuário, sendo mantido constante ao decorrer da execução do algoritmo genético. A única restrição imposta neste processo é a de que somente existam arestas entre vértices adjacentes, caso contrário o processo de avaliação dos indivíduos bem como a detecção do cruzamento entre arestas seria bastante dificultado. Desta forma, cada vértice pode possuir até quatro adjacentes: a norte, a sul, a leste e a oeste. É válido ressaltar que tanto os vértices envolvidos nas arestas quanto o número de arestas criadas são definidos de forma aleatória.

A técnica de *crossover* escolhida é a de um ponto (descrita na Seção 2.3), sendo a seleção deste ponto feita de forma aleatória por meio de uma distribuição uniforme, onde todas as posições da matriz têm a mesma chance de serem selecionadas. Contudo, o fato de cromossomo ser estruturado no formato de matriz faz com que seja necessária a definição de dois pontos para o *crossover* (um vertical e outro horizontal). Para isto, o mesmo ponto de corte é utilizado nas duas dimensões, gerando então um corte quadrado. Conforme é apresentado na Figura 16, a primeira parte (fração superior esquerda) do cromossomo concebido pelo *crossover* (*cromossomo C*) é oriunda do primeiro indivíduo selecionado para a formação do par (*cromossomo A*). Logo, a segunda parte (inferior direita) é retirada do segundo cromossomo escolhido para participar do *crossover* (cromossomo B), concluindo assim o *crossover*.

O operador genético de *mutação* é responsável por promover a diversidade da população modificando de forma aleatória um ou mais genes do cromossomo, ampliando o espaço de busca e dificultando a convergência prematura do algoritmo genético. No algoritmo de otimização desenvolvido, dois genes de cada cromossomo

(células da matriz de adjacência) são selecionados de forma aleatória para serem submetidos ao processo de *mutação*, alterando então dois dos vértices do canal. Assim como ocorre na atividade de criação da população inicial, a única restrição imposta à alteração de um gene é a de que os vértices envolvidos sejam obrigatoriamente adjacentes.

Figura 16 – Processo de *crossover.*



Fonte: O autor

O método de seleção por torneio é o adotado pelo algoritmo genético. Nele, três indivíduos são selecionados aleatoriamente para participarem de um torneio, sendo o mais apto dentre estes é escolhido para compor um novo conjunto denominado *população intermediária*. O processo é repetido até a população intermediária atingir 50% do tamanho da população total. A partir desta, indivíduos aos pares são selecionados aleatoriamente para passarem por *crossover* e *mutação*, gerando novos cromossomos. A técnica de substituição selecionada é a *steady state* (descrita na Seção 2.3), onde os indivíduos gerados por operadores genéticos substituem os menos aptos dentre a população total. Esta seleção também é feita por torneio, apenas contando com o diferencial da escolha do menos apto para ser substituído. O percentual de indivíduos substituídos é obtido pela adição das taxas de *crossover* e *mutação*, que são definidos de maneira estática pelo usuário do

software. Este processo é reiterado até que o número de gerações definido pelo usuário seja alcançado.

3.2 Função de Aptidão

A função de aptidão consiste em uma equação formada pelo conjunto de notas calculadas para as variáveis de saída do modelo, que retorna um valor real entre zero e dez, diretamente proporcional à qualidade do resultado estimado para uma dada solução. O cálculo da nota correspondente a cada variável de saída é realizado durante a etapa de avaliação dos indivíduos. A equação de aptidão foi determinada de forma empírica, e multiplica cada termo (aptidão referente à variável de saída) por um fator de proporcionalidade que é definido pelo usuário. As variáveis, no caso, correspondem aos aspectos que são considerados importantes no projeto de uma lavoura de arroz e a ponderação destes fatores expressa a importância relativa entre eles.

A função de aptidão é dada pela expressão

$$\boldsymbol{APTIDÃO} = (\boldsymbol{pa}.\boldsymbol{A}) + (\boldsymbol{pe}.\boldsymbol{E}) + (\boldsymbol{pl}.\boldsymbol{L}) + (\boldsymbol{pc}.\boldsymbol{C})$$

onde cada termo corresponde:

A: eficácia do projeto de irrigação (PI);

E: eficiência energética do PI;

L: eficiência econômica dos levantes;

C: eficiência econômica do canal;

pa: coeficiente de importância de *A* na aptidão do indivíduo;

pe: coeficiente de importância de *E* na aptidão do indivíduo;

pl: coeficiente de importância de *L* na aptidão do indivíduo;

pc: coeficiente de importância de *C* na aptidão do indivíduo.

Como pode ser visto na equação de aptidão cada variável de saída é detentora de um coeficiente multiplicador, termo cuja existência é justificada pela necessidade de estabelecer prioridade aos aspectos considerados pelo modelo. Valores entre zero e um são atribuídos a cada um destes coeficientes (*pa*, *pe*, *pl* e

pc), de maneira a estabelecem o objetivo principal do sistema de irrigação, cabendo ao usuário a responsabilidade de realizar esta definição.

Dando início à fase de avaliação dos indivíduos, primeiramente o modelo necessita tratar possíveis situações de incoerência presentes nos cromossomos, calcular a demanda hídrica a que o canal secundário será submetido e, posteriormente, dimensionar o sistema de irrigação para que o mesmo possa atender à solicitação hídrica. Só então o cálculo das notas das variáveis de saída listadas acima é realizado. Contudo, este princípio de modelagem descrito acima (2ª abordagem) não foi o utilizado desde o início do desenvolvimento do software, e sim elaborado durante o processo de implementação do algoritmo de otimização.

Na primeira abordagem, a área de secção transversal canal secundário é pré-definida pelo usuário, e as potências das bombas portadas pelo cromossomo (geradas aleatoriamente). Um dos problemas de estabelecer a área de secção transversal do canal previamente é que o mesmo será obrigatoriamente superdimensionado, uma vez que conforme os segmentos de canal se distanciam do ponto de entrada d'água, quadros são irrigados e a vazão do canal tende a diminuir. Portanto, a magnitude de dois segmentos de canal teoricamente não deve ser a mesma. Outro problema é o favorecimento da instalação de levantes em declives, já que no caso de um segmento de canal com sentido de fluxo direcionado à jusante (para baixo) possuir uma área de secção transversal inferior à necessária para atender sua demanda, a única alternativa remanescente ao modelo é a de instalar uma moto-bomba neste seguimento (o que sem dúvida é um erro de projeto). A atribuição da responsabilidade de encontrar a potência correta para os sistemas de levante ao algoritmo genético (por meio do cromossomo) não tem sentido algum, pois com a utilização de uma equação (apresentada logo abaixo) é possível encontrar a potência dissipada pela moto-bomba (*pot*) com precisão e eficiência computacional superiores. Desta forma, a segunda abordagem do algoritmo não apenas encontra a melhor solução, como também dimensiona cada segmento de canal e a potência de cada bomba instalada em seu percurso.

Voltando à segunda abordagem, o canal representado no cromossomo pode conter incoerências como cruzamentos de segmentos e formação de ciclos, e mesmo assim sequer estar conectado a sua fonte de água (ponto de entrada d'água na lavoura). Buscando driblar problemas desta natureza, uma varredura é realizada no cromossomo (matriz de adjacência do canal) em busca de arestas que estão

direta ou indiretamente ligadas ao canal primário, sem que formem ciclos ou se cruzem. Somente os vértices encontrados neste percurso são considerados como pertencentes ao canal durante as próximas etapas avaliação desse indivíduo.

Uma vez definido o conjunto de vértices que fazem parte do canal secundário, é chegada a hora de atribuir a irrigação dos quadros a este concomitante de vértices. Primeiramente cada vértice do canal é marcado como montante do quadro onde ele próprio está posicionado. Em seguida, um algoritmo iterativo de atribuição de quadros (processo de marcação dos quadros como jusantes indiretos do vértice em questão) tem início, de modo que cada vértice segue a *Lista de Adjacentes* (descrita na Seção 3.1) do quadro onde ele se localiza. Como cada elemento da *Lista de Adjacentes* representa o grau de adjacência (ou distância) para o quadro possuidor da lista, a cada iteração todos os vértices são marcados como montantes dos quadros localizados em uma posição de suas respectivas listas. Desta forma, a cada iteração o processo a atribuição de quadros avança em um grau de adjacência, o que ocorre até que a distância máxima seja alcançada ou todos os quadros das listas sejam irrigados. É importante frisar que a cada vez que um quadro é atribuído a um vértice, este deve ser excluído de todas as *Listas de Adjacentes*. Pois, caso contrário mais de um vértice poderia ser marcado como montante do mesmo quadro.

Ao término do processo de atribuição de tabuleiros, um vetor contendo a demanda hídrica (vazão) de cada vértice do canal é retornado. Contudo, a relação de vazões contidas neste vetor está incorreta, uma vez que a vazão dos vértices tende a aumentar na direção do ponto de entra d'água da lavoura (sendo que um segmento de canal não deve apenas atender a demanda hídrica do seu vértice à jusante, mas também de todos os vértices posteriores seguindo a direção de escoamento do fluído). Desta forma o conjunto de vazões do canal é cumulativo, e um algoritmo elaborado utilizado o princípio da recursividade acumula as vazões dos vértices do canal das folhas em direção à raiz (considerando o canal como uma estrutura no formato de árvore, onde a raiz é o ponto de entrada d'água e as folhas são as demais extremidades do canal).

Posto que a demanda hídrica de cada vértice do canal de irrigação é conhecida, a próxima ação executada pelo algoritmo de otimização é a de dimensionar o canal secundário. Durante este procedimento, cada segmento do canal é analisado de forma independente, e dimensionado para atender a demanda do vértice à jusante da aresta. Neste contexto, duas situações podem surgir: a que o

vértice à jusante está a uma cota superior a do vértice à montante, e a que o mesmo está a uma cota inferior a do vértice à montante.

Na primeira situação a instalação de um sistema de levante hidráulico (bomba) é necessária, de maneira que o cálculo da potência da bomba é feito por meio da equação abaixo (CORRÊA, 2007).

$$\boldsymbol{pot} = \frac{\boldsymbol{H}.\boldsymbol{Q}.\boldsymbol{\gamma}}{\boldsymbol{\eta}}$$

sendo:

pot: potência dissipada pela moto-bomba ($J.s^{-1}$).

H: diferença de cota entre os dois vértices da aresta (m);

Q: vazão demanda pelo vértice à jusante da aresta ($m^3.s^{-1}$);

γ: peso específico da água ($N.m^{-3}$);

η: coeficiente de rendimento do conjunto moto-bomba (adimensional).

Vale salientar que no segmento de canal onde um recalque é instalado, apenas a equação apresentada acima é utilizada durante os cálculos de escoamento. Isto quer dizer que apenas a perda de carga localizada na bomba é contabilizada, sendo então a perda na tubulação (ou canal) percorrida pela água entre a bomba e o próximo vértice desconsiderada pelo modelo.

Já nas arestas onde o vértice à jusante está localizado a uma cota inferior a do vértice à montante, a aplicação de moto-bombas é proibitiva. Sendo a vazão demandada alcançada por meio do ajuste da área de secção transversal do segmento de canal em foco. Logo, o cálculo efetivado para dimensionar esta aresta do canal é feito com a utilização da equação de chezy-Manning (BASSO, 2012), a qual e definida como

$$\boldsymbol{Q} = \frac{\boldsymbol{1}}{\boldsymbol{m}}\ \boldsymbol{a}\ (\boldsymbol{Rh})^{\frac{2}{3}}\ (\boldsymbol{S})^{\frac{1}{2}}$$

sendo:

Q: vazão ($m^3.s^{-1}$);

a: área de secção transversal do canal (m^2);

Rh: raio hidráulico (m);

S: declividade ($m.m^{-1}$);

m: coeficiente de rugosidade de Manning (adimensional).

Onde a área (*a*) é dada por

$$\boldsymbol{a = \frac{Q\ n}{(Rh)^{\frac{2}{3}}\ (S)^{\frac{1}{2}}}}$$

A vazão (*Q*) e a declividade (*S*) são conhecidas, porém o raio hidráulico (*Rh*) é uma variável que está relacionada com a geometria do canal secundário. De acordo com Carvalho (2009) as formas geométricas mais utilizadas em canais de irrigação são as retangulares, trapezoidais, triangulares e semicirculares. A escolha pela utilização da geometria circular no modelo definiu o cálculo do raio (*Rh*) como

$$\boldsymbol{Rh = \frac{128\ a}{\pi}}$$

Como é possível observar nas equações apresentadas acima a área (*a*) e o raio hidráulico (*Rh*) são variáveis dependentes, portanto é necessário que um sistema não linear seja resolvido para que a área de secção transversal do canal (*a*) seja encontrada. Devido à elevada complexidade inerente à resolução de sistemas desta natureza, o algoritmo adota uma técnica bastante utilizada para o dimensionamento de canais, o *Método das Tentativas* (CARVALHO, 2009). Neste, um algoritmo iterativo atribui valores ao raio hidráulico (*Rh*), calculando a área (*a*) e a vazão (*Q*) logo em seguida. A cada revolução deste algoritmo o raio (*Rh*) é alterado de maneira à vazão (*Q*) se aproximar da demanda hídrica do segmento. O processo é repetido até que a vazão calculada para o segmento se encontre em uma faixa de 0 a 10% acima da requerida.

Partindo do pressuposto de que a solução abrigada pelo cromossomo foi tratada quanto a possíveis incoerências, e que o sistema de irrigação por ela descrito está corretamente dimensionado para atender a lavoura especificada, a qualidade de atendimento a cada um dos aspectos (variáveis de saída) considerados pela equação de aptidão pode ser avaliada. De forma análoga a função de aptidão, as funções objetivo também retornam notas entre zero e dez. Entretanto, para que os valores obtidos para as variáveis de saída devolvam valores contidos neste intervalo que sejam diretamente proporcionais aos objetivos do algoritmo de otimização, existe a necessidade de que limites sejam estabelecidos, já que não há como especificar (as grandezas relacionadas a estas variáveis) que valor equivale à nota máxima ou mínima. A primeira alternativa encontrada foi a de utilizar limites constantes para cada uma das grandezas, isto é, uma vez estabelecidos, estes valores eram mantido iguais até o término da execução do software. Contudo

esta saída tende a reduzir a sensibilidade das variáveis, dado que limites muito distantes fazem com que uma grande diferença medida resulte em uma modificação muito sutil na nota dada a aquele aspecto. A segunda solução encontrada (e atualmente utilizada pelo modelo) não faz uso de limites previamente estabelecidos, mas sim do melhor e pior valor encontrado para uma dada grandeza na geração. Estas medidas são utilizadas como parâmetros para as notas máxima (dez) e mínima (zero) respectivamente, o que garante a sensibilidade do modelo quanto à avaliação dos aspectos por ele considerados.

Sendo assim a equação objetivo que avaliada a eficácia da solução (*A*) é baseada no percentual da área irrigada, e obedece ao mesmo princípio de limites máximos e mínimos descritos acima. O valor de *A* é calculado pela equação

$$\boldsymbol{A = \frac{(percA - amin).10}{amax - amin}}$$

sendo:

A: eficácia da solução;

percA: percentual da área irrigada pelo PI (%);

amax: maior percentual de área irrigada para uma solução na iteração atual (%);

amin: menor percentual de área irrigada para uma solução na iteração atual (%).

O valor de *percA* é encontrado pela divisão da área atingida pela irrigação pelo total da área cultivada, cálculo descrito por

$$\boldsymbol{percA = \frac{ai}{al}\ 10}$$

onde:

ai: área irrigada (m^2);

al: área total da lavoura (m^2).

O consumo de energia é definido como o produto entre a potência dissipada e o período total de funcionamento do equipamento. Visto que o segundo é uma constante, o somatório das potências das moto-bombas (*P*) é a única variável que influência na eficiência energética do sistema. Sendo assim, a nota atribuída a este aspecto é contabilizada na função de aptidão pela variável *E,* de modo que o valor

encontrado por *E* é inversamente proporcional ao somatório das potências do conjunto de levantes (*P*). *E* é dado por

$$\boldsymbol{E = \frac{(pmax - P).10}{pmax - pmin}}$$

considerando:

E: eficiência energética da solução;

P: potência total dissipada pelo PI (w);

pmax: maior potência dissipada por uma solução na iteração atual(w);

pmin: menor potência dissipada por uma solução na iteração atual(w).

P é calculado por

$$\boldsymbol{P = \sum_{i=1}^{nº\ de\ recalques} Pi}$$

, de modo que *Pi* representa a Potência (w) de cada recalque utilizado pelo sistema de irrigação.

Três aspectos são considerados pelo modelo na análise da eficiência econômica do sistema de irrigação: os custos variáveis associados aos levantes, os custos fixos relacionados aos levantes e os custos associados ao canal. O primeiro corresponde aos custos de instalação e manutenção dos recalques que variam de acordo com seu porte (potência). Como este custo é diretamente proporcional à potência total do sistema, ele já está representado na equação de aptidão por meio da variável *E.* O segundo corresponde aos custos fixos presentes na instalação e manutenção dos levantes, isto é, aqueles que existem em todo o levante hidráulico instalado, independente de sua potência. Portando este custo depende do número de levantes instalados no sistema de irrigação, sendo esta a grandeza representada pela nota *L* na função de aptidão. A equação utilizada para o cálculo de *L* é

$$\boldsymbol{L = \frac{(nmax - N).10}{nmax - nmin}}$$

onde:

L: eficiência econômica do PI relacionados aos levantes;

N: total de levantes hidráulicos do PI;

nmax: maior número de levantes instalados em uma solução na iteração atual;

nmin: menor número de levantes instalados em uma solução na iteração atual.

Logo, o último aspecto financeiro a ser considerado na avaliação das soluções é o custo relacionado ao canal de irrigação. O canal utilizado em sistemas de irrigação por inundação é do tipo conduto livre, isto é, condutos (abertos) em que a parte superior do líquido está submetida à pressão atmosférica (CARVALHO, 2009). A construção destes condutos é concretizada com ao auxílio que equipamentos específicos, cujo custo de utilização é medido em horas de trabalho. Desta forma, quanto maior for o conduto (comprimento e área de secção transversal) maior será o tempo gasto e, consequentemente, o valor dispendido em sua construção. Com a manutenção não é diferente, canais necessitam de limpeza e podem sofrer de infiltração, problemas que geram gastos proporcionais ao porte do canal. Sendo assim a variável de saída que representa eficiência econômica do canal é *C,* estimado por

$$\boldsymbol{C = \frac{(vmax - V).10}{vmax - vmin}}$$

considerando:

C: eficiência econômica do PI relacionada aos levantes;

V: volume máximo de água comportado pelo PI (m^3);

vmax: volume máximo de água comportado em uma solução na iteração atual;

vmin: volume máximo de água comportado em uma solução na iteração atual.

Como o porte do canal é dado pelo produto da área secção transversal pelo comprimento do mesmo, a grandeza avaliada é o volume (*V*), ou mais especificamente o somatório dos volumes dos segmentos de canal. A equação que calcula esta grandeza é

$$\boldsymbol{V = \sum_{i=1}^{nº\ de\ arestas} sec\_i\ .\ compr\_i}$$

onde:

V: volume total de água comportado pelo canal;

sec_i: área de secção transversal do segmento de canal (m^3);

compr_i: distância entre o vértice à montante e à jusante do segmento de canal (m).

3.3 Resultados e Discussões

Os modelos hidrológicos encontrados durante a atividade de pesquisa utilizam o balanço de bacias hidrológicas (descrito na Seção 2.2) para descrever os fenômenos físicos relacionados ao acúmulo e fluxo de água. Em alguns modelos encontrados, o foco engloba áreas extensas da lavoura, mostrando o relacionamento entre a lavoura e o rio ou canal de drenagem que a circunda (MARCUZZO, 2008). Em outros (MARCUZZO, 2008), onde os modelos estão direcionados estreitamente à área da lavoura, não é buscado descrever a topologia de canal, mas sim gerar estimativas de consumo de água e energia elétrica, sendo geralmente aplicados a sistemas de irrigação por aspersão e não por inundação (como no caso do arroz). Já a estratégia de modelagem elaborada para o modelo hidrológico utilizado neste trabalho é baseada na interação entre os componentes do sistema de irrigação por inundação em lavouras de arroz.

Sendo assim, a distinção de objetivos existente entre o modelo utilizado e os encontrados durante a pesquisa, dificulta a validação do algoritmo de otimização por meio da estratégia de comparação com outro modelo já existente. Outra estratégia de validação seria comparar os resultados apresentados pelo algoritmo com resultados adquiridos por meio de experimentos reais, o que demandaria tempo e recursos financeiros não disponíveis para o desenvolvimento deste trabalho. Desta forma seria incorreto afirmar que os resultados encontrados durante as simulações realizadas com o algoritmo refletem com exatidão o funcionamento do sistema real. Por isso o objetivo da análise dos resultados se restringe a discutir o comportamento do modelo hidrológico e do algoritmo genético, confrontando a conduta do algoritmo de otimização com a teoria. Procura-se com isso demonstrar que as decisões tomadas durante o desenvolvimento do algoritmo constituem uma base consistente para que a continuidade dada ao desenvolvimento deste trabalho possa trazer resultados satisfatórios no auxílio ao projeto de sistemas de irrigação para lavouras de arroz.

Com o objetivo de demonstrar o comportamento do algoritmo de otimização, este foi submetido a um conjunto de simulações. No decorrer deste capítulo, os

experimentos realizados são avaliados via análise de sensibilidade, de maneira que um estudo referente aos resultados obtidos é exposto. A técnica de análise de sensibilidade foca no comportamento dos parâmetros de saída frente às variações sofridas pelos parâmetros de entrada, e objetiva verificar se o modelo produz resultados lógicos em função da alteração destes parâmetros (ANDRADE, 2008). A estratégia adotada para a avaliação do modelo foi a de alterar apenas o valor de um único parâmetro entrada de cada vez, mantendo os demais constantes durante o período de simulação.

As variáveis de saída são utilizadas como métrica para que o desempenho do modelo seja determinado. Este conjunto de variáveis é exatamente o mesmo considerado pela função de aptidão do modelo, isto é, o percentual da área irrigada, o somatório das potências dos levantes, o total de levantes e o volume de fluido comportado pelo canal.

Os parâmetros de entrada estão divididos em variáveis e constantes. Variáveis (ou variáveis de entrada) são aqueles que sofrem alterações durante os experimentos, e foram selecionados (de forma empírica) devido à necessidade de avaliar o comportamento do modelo. Já os constantes não sofrem nenhuma alteração durante as simulações e, portanto fazem parte das condições de contorno do procedimento experimental. O conjunto de parâmetros listados na Tabela 3 acrescidos do mapa contido na Figura 17 formam as condições de contorno definidas para a realização dos experimentos. O mapa utilizado foi gerado por meio de uma função criada pelo desenvolvedor e possui uma área total de 100 hectares.

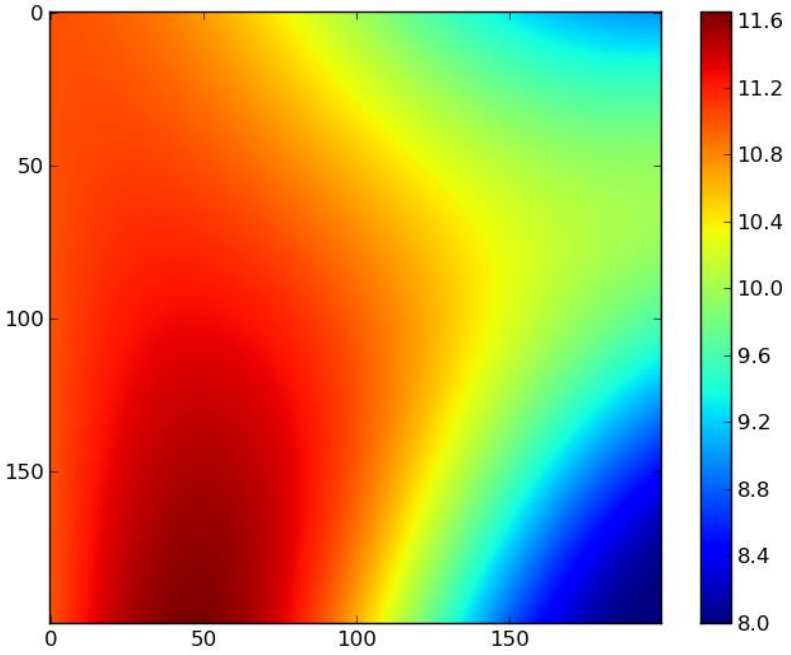
O número de iterações foi determinado de forma experimental, isto é, testes foram realizados para que o número de gerações necessário para a convergência do algoritmo fosse estabelecida. A Figura 18 apresenta o comportamento das variáveis de saída para um dos testes realizados, onde o eixo horizontal representa a iteração e o vertical a variável de saída. Nesta imagem é possível observar que o algoritmo genético apresenta facilidade de convergência e que, portanto 50 iterações são suficientes.

Tabela 3 – Condições de contorno.

Parâmetro	Símbolo	Valor	Unidade
Total de Iterações	*I*	50	–
Taxa de evapotranspiração média	*ET*	9,9	$mm.dia^{-1}$
Taxa de infiltração média	*Vi*	0,3	$m.dia^{-1}$
Período de formação da lâmina d'água	*T1*	5	dias
Período de saturação	*T2*	90	dias
Profundidade média da lâmina d'água	*hl*	7,5	cm
Diferença altimétrica entre quadros	*dh*	10	cm
Coeficiente de Chezy-Manning	*m*	0,02	–
Coeficiente de rendimento dos levantes	η	60	%
Peso específico da água	γ	9790,4	$N.m^{-3}$

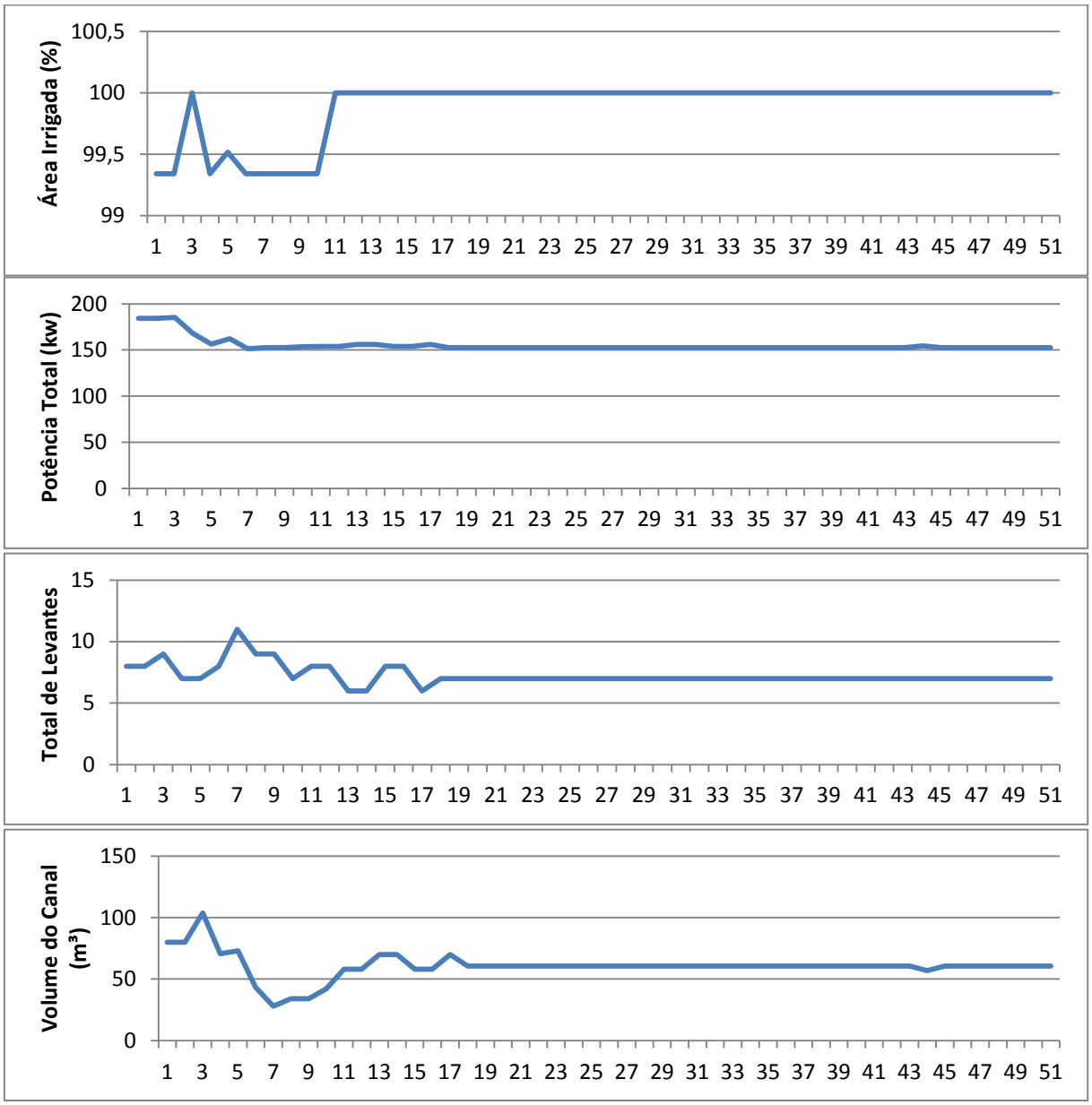
Fonte: O autor

Figura 17 – Mapa da área cultivada.



Fonte: O autor

Figura 18 – Teste de convergência do algoritmo.



Fonte: O autor

O coeficiente de rendimento dos levantes (η), a altura da lâmina d'água (*hl*) e os períodos de formação (*T1*) e manutenção (*T2*) da lâmina seguem os valores sugeridos por Corrêa (2007). A taxa de evapotranspiração média (*ET*) utilizada também é a mesma calculada por Corrêa (2007), sendo esta resultante da média ponderada das taxas das correspondentes as fazes *T1* e *T2*. A taxa de infiltração média (*Vi*) é formada pelo somatório das perdas ocorridas nos fenômenos de percolação e fluxo lateral (Seção 2.1), de maneira que o valor utilizado foi

encontrado em Stone (2005) e se refere à taxa de infiltração para canais em terreno arenoso. A diferença altimétrica admitida entre os quadros (*dh*) foi calculada a partir do principio estabelecido por Castro (2003), que diz que este valor não deve superar mais que 2/3 de *hl*. O Peso específico da água (*γ*) foi extraído de Çengel (2007), livro referência na área de mecânica de fluídos. O coeficiente associado ao atrito entre o fluído e as paredes do canal que o conduz, denominado de Chezy-Manning (*m*), selecionado a partir de uma tabela encontrada em Carvalho (2009). A tabela em questão contém uma relação de coeficientes associados aos materiais utilizado na construção do canal e ao seu formato, sendo o valor utilizado o correspondente a canais retos e uniformes com paredes de terra.

As variáveis de entrada selecionadas para sofrerem variações são: o total de indivíduos (tamanho da população), taxa de *crossover*, taxa de *mutação*, distância máxima de propagação da água a partir dos vértices do canal, ponto de entrada d'água na lavoura, total de vértices estabelecidos para o canal e a distribuição de prioridades das variáveis de aptidão. O termo *parametrização* é bastante utilizado ao decorrer deste capítulo, e se refere ao conjunto de valores (ou configuração) estabelecido para as variáveis de entrada do software. Desta maneira um conglomerado de parametrizações é estabelecido e experimentos (simulações) são realizados conforme o conjunto de configurações (parametrizações) elaborado, possibilitando assim a criação de uma coleção de resultados que servem de base para a atividade de avaliação do algoritmo de otimização.

Os experimentos foram realizados em triplicata, isto é, para cada parametrização realizou-se três simulações. Deste modo os valores considerados durante a análise não são resultantes de uma única simulação, mas sim do cálculo da média aritmética simples dos três experimentos realizados. O procedimento de análise experimental é orientado pelas variáveis de entrada do modelo, de maneira que para cada variável dois procedimentos de análise são praticados. No primeiro, o comportamento de cada uma das variáveis de saída é avaliado de forma isolada, exaltando os resultados obtidos com a simulação. Já o segundo procedimento analisa as variáveis de saída de maneira conjunta, buscando assim esclarecer a influência da variável de entrada na solução encontrada pelo modelo. Este procedimento é justificado pelo fato de conduta evolutiva do modelo estar amarrada à distribuição dos coeficientes (*pa*, *pe, pl* e *pc*) da função de aptidão, uma vez que o modelo busca atender a ordem de prioridade estabelecida por meio destas variáveis.

Sendo assim, esta distribuição de prioridades deve ser considerada pelo procedimento a análise, o que exalta a necessidade de uma equação que demonstre o comportamento do modelo frente à solicitação do usuário.

A necessidade de uma equação que considere os coeficientes de prioridade é visível na seguinte situação. Duas parametrizações, diferenciadas apenas pelo tamanho da população (variável de entrada), são elaboradas de modo que a população da segunda simulação possui o dobro de indivíduos da primeira. A análise isolada do percentual da área irrigada (primeira variável de saída analisada) apresentou uma queda de 2% da primeira simulação para a segunda. Considerando que a queda da área irrigada vai contra o intuito do sistema de irrigação, a análise individual desta variável de saída demonstrou uma piora na qualidade do resultado obtido para o aumento da população. Já a análise da potência total dos levantes (segunda variável de saída analisada) apresenta uma queda de 20% no seu valor, o que é um resultado positivo no que se refere ao aumento do tamanho da população. Ao analisar as duas variáveis de maneira conjunta fica a pergunta, o aumento do número de indivíduos na população do algoritmo genético faz com que soluções apresentadas pelo algoritmo de otimização sejam melhores? Buscando esclarecer esta dúvida elaborou-se uma equação baseada na variação percentual das variáveis de saída, onde é associado a cada percentual o mesmo coeficiente de prioridade presente na função de aptidão do modelo. Desta forma, cada resultado observado por meio das variáveis de saída (seja ele positivo ou negativo) é considerado com o mesmo percentual de importância estabelecido ao algoritmo de otimização pelo usuário. A *equação de avaliação conjunta das variáveis de saída* é

$$\boldsymbol{avaliação} = (\boldsymbol{pa} * \boldsymbol{v1}) - (\boldsymbol{pe} * \boldsymbol{v2}) - (\boldsymbol{pl} * \boldsymbol{v3}) - (\boldsymbol{pc} * \boldsymbol{v4})$$

, onde:

v1: variação percentual da percentual da área irrigada;

v2: variação percentual do somatório das potências dos levantes;

v3: variação percentual do número de levantes;

v4: variação percentual do volume de fluido comportado pelo canal;

pa: coeficiente de importância de *v1* na aptidão do indivíduo;

pe: coeficiente de importância de *v2* na aptidão do indivíduo;

pl: coeficiente de importância de *v3* na aptidão do indivíduo;

pc: coeficiente de importância de *v4* na aptidão do indivíduo.

O percentual da área irrigada é a única variável diretamente proporcional à aptidão de uma solução, de maneira que todos os demais termos da equação de análise possuem sinal negativo. Se um valor maior que zero for retornado pela equação, significa que a alteração realizada na variável de entrada surtiu efeito positivo no desempenho do algoritmo. Caso contrário, a alteração de valores têm efeito negativo sobre modelo. O cálculo das variações percentuais é feito com base nos valores de saída encontrados na simulação realizada com a primeira parametrização da variável de entrada em foco. A equação utilizada neste procedimento é

$$vi = -100 + (100.\frac{xo}{xj})$$

sendo:

vi: variação percentual da variável de saída;

xo: valor da variável de saída referente a primeira parametrização da variável de entrada;

xj: valor da variável de saída referente a última parametrização da variável de entrada.

É importante ressaltar que a *equação de avaliação conjunta das variáveis de saída* é apenas uma estratégia que vêm a auxiliar na atividade de análise dos resultados. Pois como o cálculo da variação percentual da variável de saída (*vi*) é realizado sobre o valor obtido para a primeira parametrização (*xo*), uma pequena variação da grandeza (*xj* – *xo*) pode representar uma grande variação percentual, e desta forma distorcer o resultado apontado pela equação de avalição.

Com exceção dos experimentos em que a distribuição de prioridades das variáveis de aptidão é a variável foco da análise, os valores estabelecidos para os coeficientes *pa*, *pe, pl* e *pc* foram de 75%, 15%, 2.5% e 7.5% respectivamente. As demais parametrizações das simulações são apresentadas no formato de tabela juntamente com os resultados obtidos.

A primeira variável submetida à variação da parametrização é o tamanho da população, configuração esta que é descrita pela Tabela 4. Os resultados obtidos (Figura 19) demonstram uma queda nos valores de todas as variáveis de saída analisadas. Como a área irrigada é a única variável diretamente proporcional à qualidade da solução, apenas esta apresenta resultados negativos para o aumento

do número de indivíduos. De maneira que a análise independente das demais variáveis somada a avaliação conjunta de todas as variáveis de saída (gráfico localizado na parte inferior da Figura 19) apontam que a qualidade dos resultados apresentados pelo modelo está diretamente ligada à dimensão desta variável de entrada, e portanto melhora com o aumento da população. Isto demonstra que a variedade das soluções contidas no espaço de busca explorado é proporcional ao tamanho da população, característica condizente com o princípio de funcionamento desta técnica de otimização. Como a queda no percentual da área irrigada foi muito sutil (inferior a 2%) quando comparada à melhoria das demais variáveis de saída, o modelo apontou que não é vantajoso, do ponto de vista econômico, que 100% da área seja cultivada. É importante ressaltar que esta afirmativa está condicionada a distribuição de prioridades (custos) listada acima, valores que foram estipulados de forma empírica.

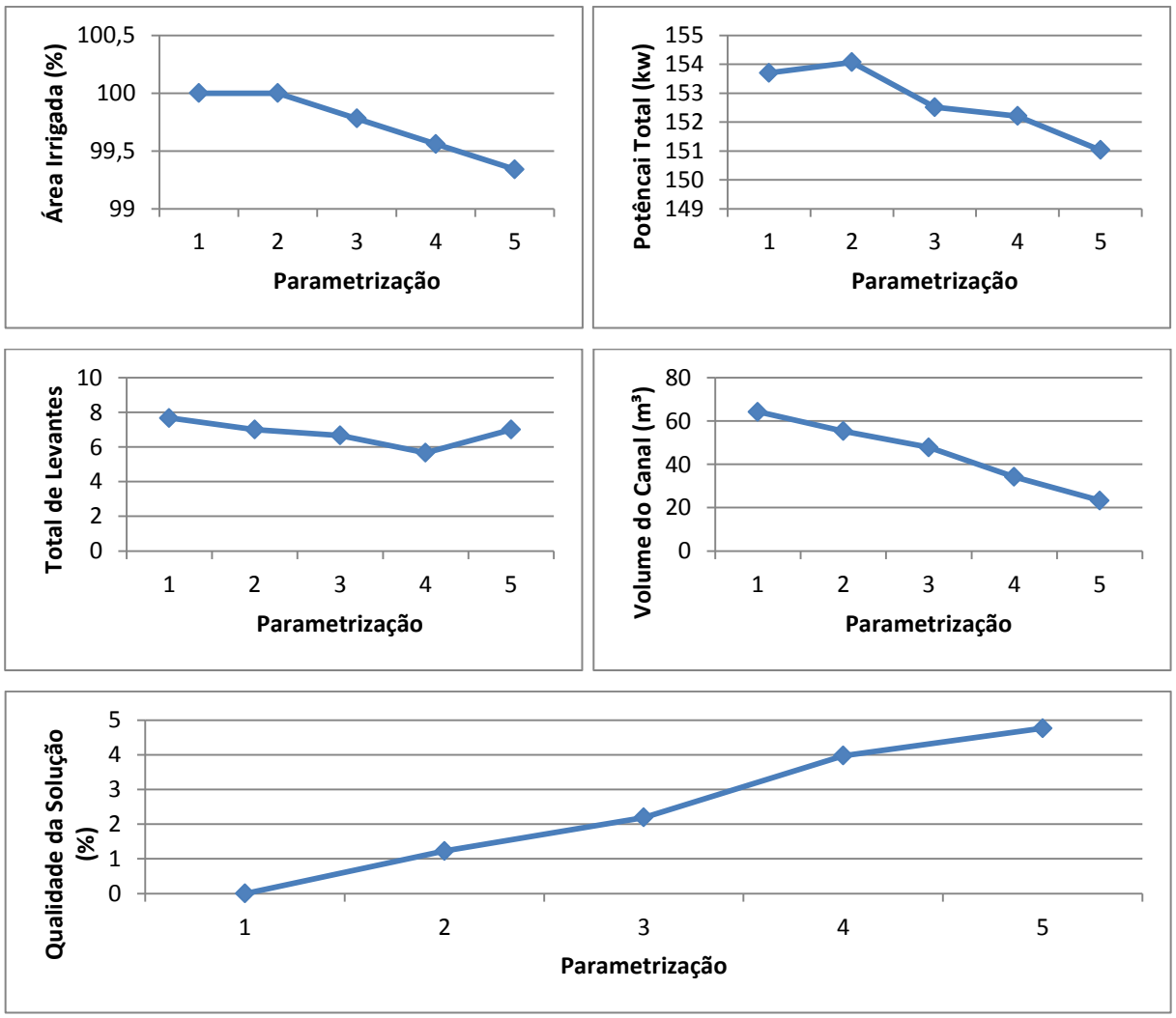
Mudando o foco da análise para a taxa de *crossover* do algoritmo genético, cinco valores próximos aos sugeridos na literatura foram estabelecidos para as simulações (Tabela 5). Ao analisar o desempenho das variáveis de saída (Figura 20) de forma independente verificou-se que a área irrigada foi a única variável que apresentou resultados positivos, ainda que com um aumento inferior a 1%. O volume de água comportado pelo canal aumentou cerca de 58% da parametrização um para a cinco, o que significa que as dimensões do canal aumentaram consideravelmente com o crescimento da taxa de *crossover*. Desta maneira, análise conjunta das variáveis de saída não poderia apontar outro resultado senão uma queda na qualidade da solução. Como o operador genético de *mutação* apresentou bons resultados (discutidos posteriormente), isto leva a crer que as técnicas de seleção e substituição escolhidas (de uso comum entre os operadores de *mutação* e *crossover*) apresentam bom desempenho e portanto não contribuíram de forma negativa para os resultado apresentados. Sendo assim, uma das hipótese existentes para este comportamento é a de que a técnica de *crossover* selecionada talvez não seja a ideal para a estrutura para o cromossomo do algoritmo. Outra hipótese que justifica o comportamento do modelo é o fato de que a elevada taxa de *crossover* pode promover a perda da informação por parte do algoritmo genético, de maneira que o desempenho do modelo pode regredir a partir de uma determinada taxa de *crossover*.

Tabela 4 – Parametrização dos experimentos em que o total de indivíduos é a variável em foco.

Parametrização	**Total de Indivíduos**	Taxa de Crossover	Taxa de Mutação	Distância de Propagação	Período de utilização dos levantes	Ponto de Entrada	Total de Vértices
1	**250**	70%	3%	6 quadros	8,5 h	médio	25
2	**500**	70%	3%	6 quadros	8,5 h	médio	25
3	**750**	70%	3%	6 quadros	8,5 h	médio	25
4	**1000**	70%	3%	6 quadros	8,5 h	médio	25
5	**1250**	70%	3%	6 quadros	8,5 h	médio	25

Fonte: O autor

Figura 19 – Resultados obtidos com o aumento do tamanho da população.



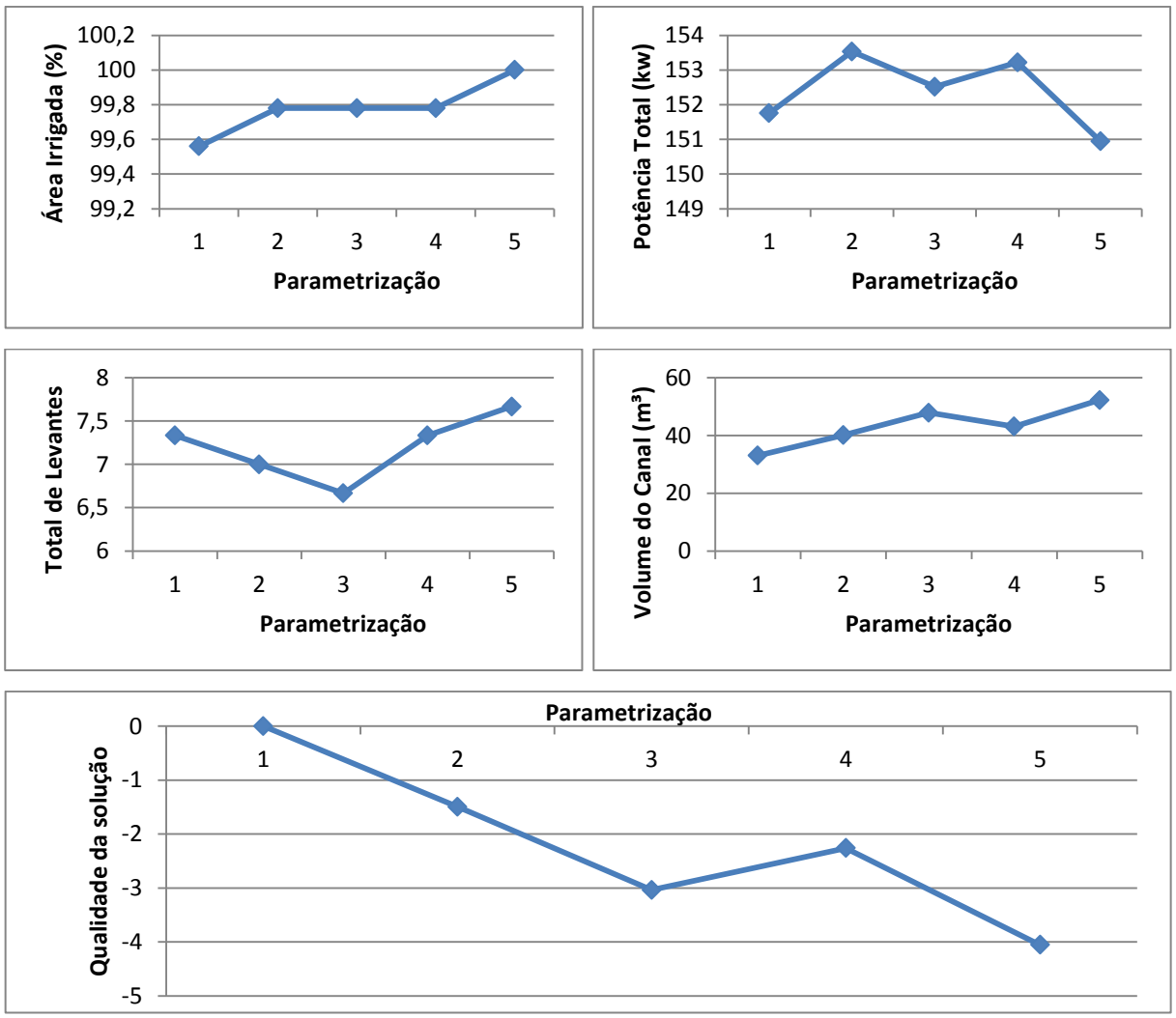
Fonte: O autor

Tabela 5 – Parametrização dos experimentos em que a taxa de *crossover* é a variável em foco.

Parametrização	Total de Indivíduos	**Taxa de Crossover**	Taxa de Mutação	Distância de Propagação	Período de utilização dos levantes	Ponto de Entrada	Total de Vértices
1	750	**50%**	3%	6 quadros	8,5 h	médio	25
2	750	**60%**	3%	6 quadros	8,5 h	médio	25
3	750	**70%**	3%	6 quadros	8,5 h	médio	25
4	750	**80%**	3%	6 quadros	8,5 h	médio	25
5	750	**90%**	3%	6 quadros	8,5 h	médio	25

Fonte: O autor

Figura 20: Resultados obtidos com o aumento da taxa de *crossover*.

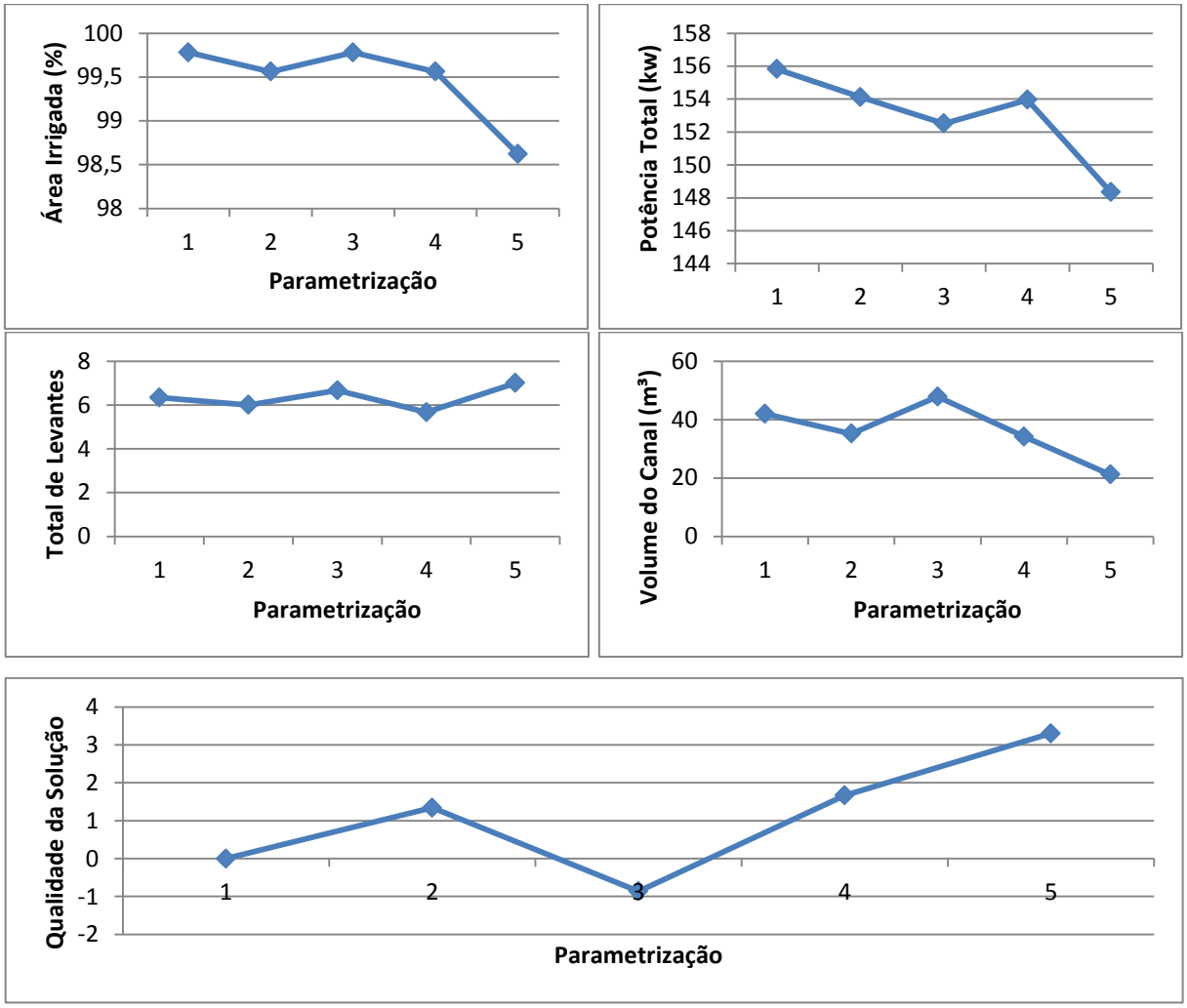


Fonte: O autor

Conforme citado anteriormente, o aumento da taxa de *mutação* descrito na Tabela 6 exibiu uma melhora nas soluções apontadas pelo algoritmo de otimização

(Figura 21). O número total de levantes manteve-se relativamente constante, e as demais variáveis apresentaram desempenho positivo – principalmente o volume de água admitido pelo canal que chegou a uma redução de 50% do seu valor inicial –. Não se observou nenhum retardo na convergência do algoritmo genético, situação possível uma vez que o operador de *mutação* promove o aumento da diversidade da população. A melhora nos resultados indicada pela análise individual das variáveis de saída apenas foi confirmada pela análise conjunta (último gráfico da Figura 21), situação esperada pela varredura mais ampla da região de busca por soluções resultante do aumento da taxa de *mutação*.

Figura 21 – Resultados obtidos com o aumento da taxa de *mutação*.



Fonte: O autor

Tabela 6 – Parametrização dos experimentos em que a taxa de *mutação* é a variável em foco.

Parametrização	Total de Indivíduos	Taxa de Crossover	**Taxa de Mutação**	Distância de Propagação	Período de utilização dos levantes	Ponto de Entrada	Total de Vértices
1	750	70%	**1%**	6 quadros	8,5 h	médio	25
2	750	70%	**2%**	6 quadros	8,5 h	médio	25
3	750	70%	**3%**	6 quadros	8,5 h	médio	25
4	750	70%	**4%**	6 quadros	8,5 h	médio	25
5	750	70%	**5%**	6 quadros	8,5 h	médio	25

Fonte: O autor

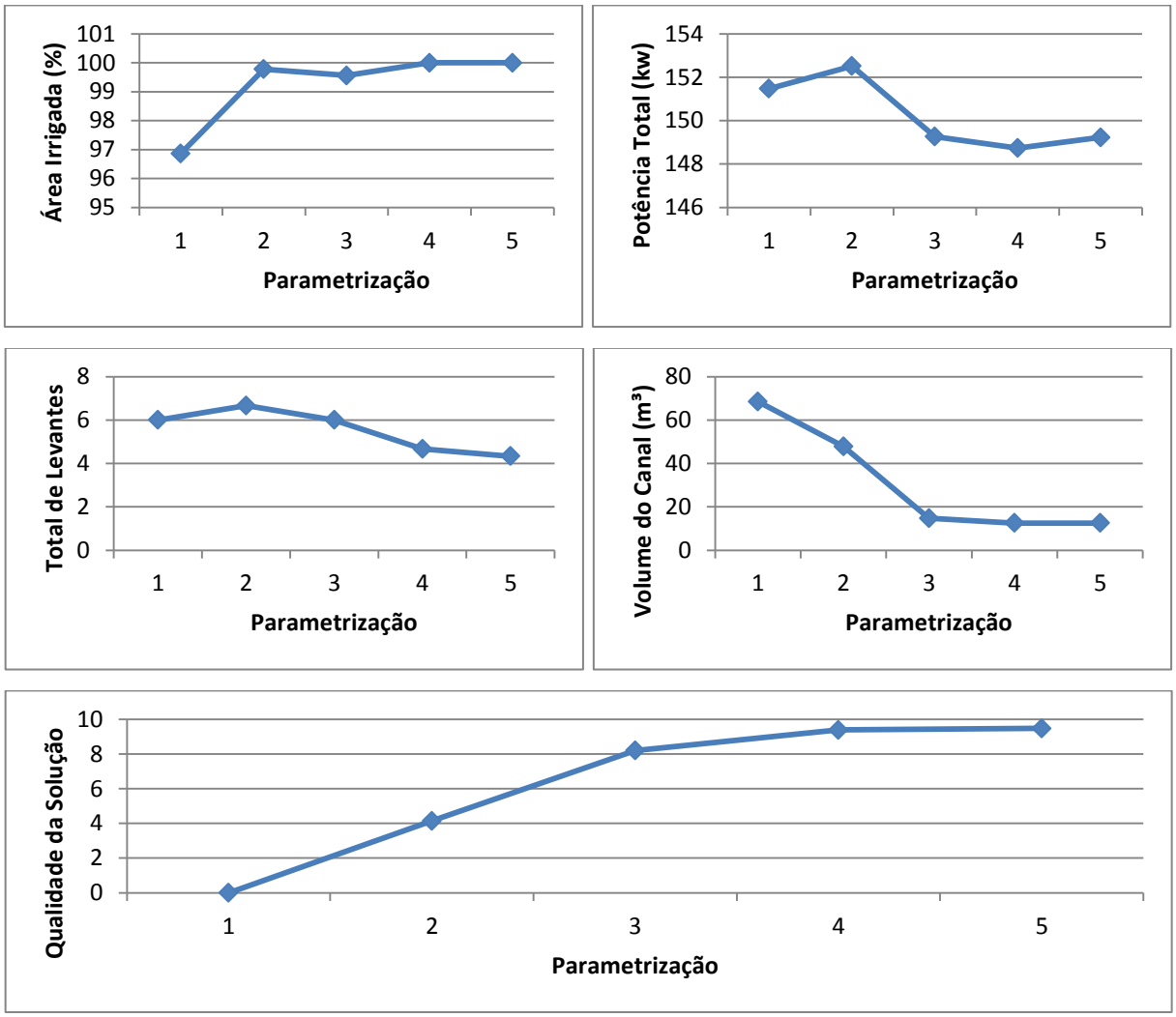
O aumento do número máximo de quadros irrigados por um único vértice (distância de propagação) definido na Tabela 7 é uma alteração que obviamente tende a trazer resultados positivos para a solução encontrada. Dado que aumentando a capacidade de irrigação dos vértices, um número menor de segmentos e canal é necessário para que os tabuleiros sejam irrigados, tem-se um impacto positivo em todas as variáveis de saída do algoritmo. Assim, os resultados apresentados pelo modelo por intermédio da Figura 22 não poderiam ser distintos, pois o comportamento esperado foi observado em todas as variáveis de saída do modelo, chegando a uma redução de até 82% do volume de água admitido pelo canal de irrigação. Desfecho que atesta a coerência do algoritmo quanto à sensibilidade para as restrições naturalmente impostas ao funcionamento de um sistema de irrigação por inundação.

Tabela 7 – Parametrização dos experimentos em que distância de propagação é a variável em foco.

Parametrização	Total de Indivíduos	Taxa de Crossover	Taxa de Mutação	**Distância de Propagação**	Período de utilização dos levantes	Ponto de Entrada	Total de Vértices
1	750	70%	1%	**3 quadros**	8,5 h	médio	25
2	750	70%	2%	**6 quadros**	8,5 h	médio	25
3	750	70%	3%	**9 quadros**	8,5 h	médio	25
4	750	70%	4%	**12 quadros**	8,5 h	médio	25
5	750	70%	5%	**15 quadros**	8,5 h	médio	25

Fonte: O autor

Figura 22 – Resultados obtidos com o aumento da distância de propagação entre quadros.



Fonte: O autor

Assim como a variável de entrada anterior, o acréscimo estabelecido para o período de utilização diária das moto-bombas comportou-se exatamente conforme esperado. Como o algoritmo busca atender a demanda hídrica diária da lavoura apenas no intervalo de tempo em que os levantes estão em funcionamento, tanto as dimensões do canal quanto a potência dissipada pelos recalques tende a diminuir com as alterações apresentadas na Tabela 8. Na análise individual das variáveis de saída (Figura 23) verificou-se que o total de potência dos levantes e o volume do canal tiveram melhoria superior a 50%, e que o percentual da área irrigada juntamente com o número de levantes mantiveram-se relativamente constantes ao decorrer dos cinco experimentos simulados. Desta forma, a análise conjunta apenas

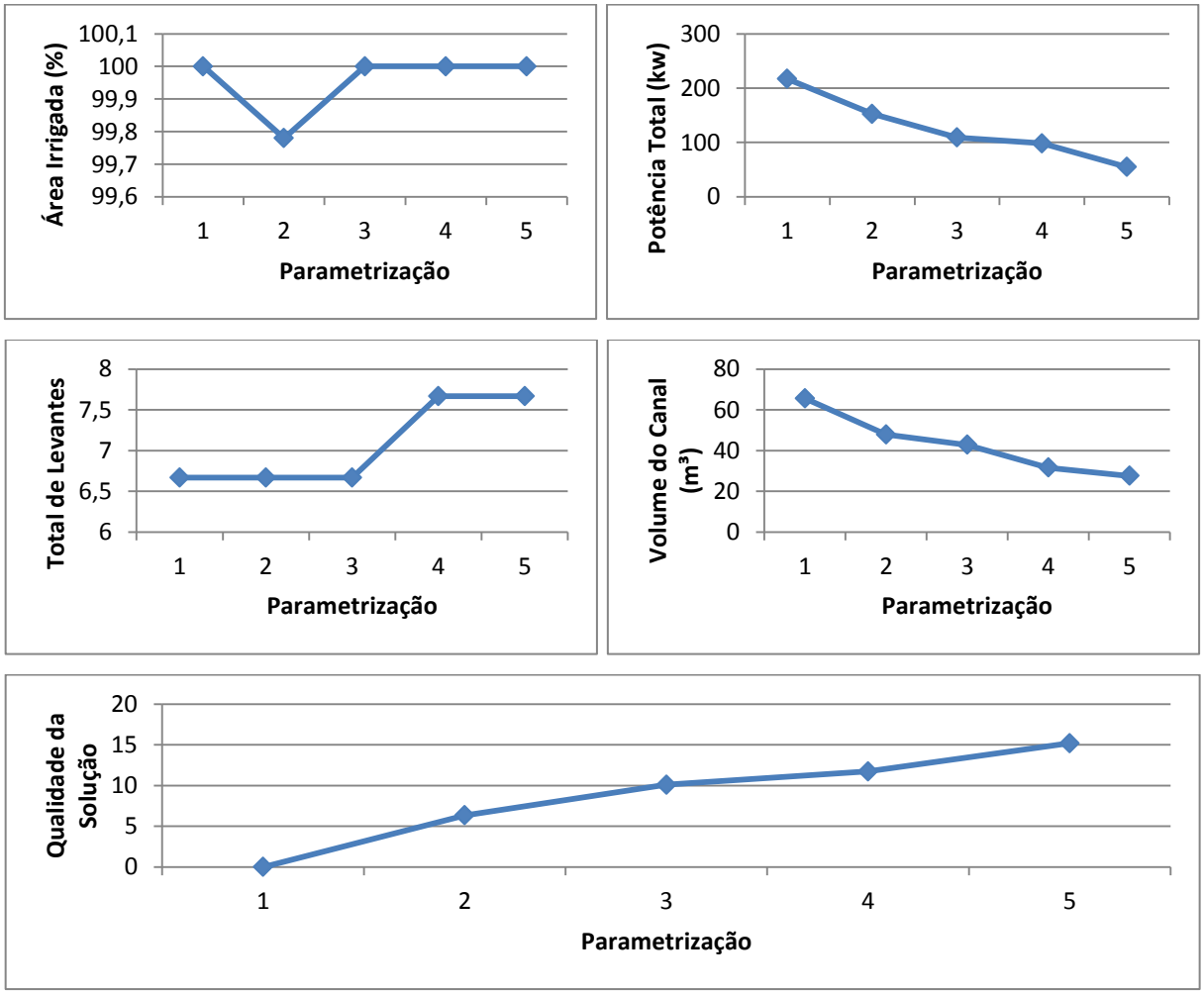
confirma a melhoria esperada na qualidade dos resultados frente ao aumento do tempo de utilização dos sistemas de recalque hidráulico.

Tabela 8 – Parametrização dos experimentos em que o período de utilização diário dos levantes é a variável em foco.

Parametrização	Total de Indivíduos	Taxa de Crossover	Taxa de Mutação	Distância de Propagação	**Período de utilização dos levantes**	Ponto de Entrada	Total de Vértices
1	750	70%	3%	6 quadros	**6 h**	médio	25
2	750	70%	3%	6 quadros	**8,5 h**	médio	25
3	750	70%	3%	6 quadros	**11 h**	médio	25
4	750	70%	3%	6 quadros	**13,5 h**	médio	25
5	750	70%	3%	6 quadros	**16 h**	médio	25

Fonte: O autor

Figura 23 – Resultados obtidos com o aumento do período de utilização diária dos levantes.



Fonte: O autor

A justificativa que vai contra o aumento do número de pontos (vértices) do canal é a de que como no modelo hidrológico um levante somente eleva a cota da água entre vértices adjacentes, o aumento do número de pontos indiretamente leva a um crescimento considerável do total de levantes. Já a justificativa que sustenta a tentativa de aumentar do total de vértices, é a de que há uma tendência maior de que somente o volume de água que necessita a ser recalcado seja elevado, pois como os vértices estão mais próximos, a probabilidade de que a situação onde um volume grande de água é recalcado a uma cota muito elevada, para depois descer novamente por meio da propagação entre quadros (o que acarreta em desperdício de energia) diminui consideravelmente.

A variação do número de vértices do canal demonstrou que para a distribuição de prioridades estabelecidas na Tabela 9 as parametrizações com 25 e 81 vértices são as ideais, possuindo a primeira um desempenho ligeiramente superior ao da segunda. Na Figura 24, é possível observar que total de levantes aumentou consideravelmente conforme esperado, exaltando o impacto da restrição no processo de inserção de levantes descrita no início do parágrafo anterior. Já o total de potência das bombas não se mostrou tão sensível como se esperava, pois acompanhou a variação da área irrigada, o que significa que ela não tende a diminuir substancialmente com o aumento do número de levantes. Como este é o principal argumento favorável ao aumento da variável em questão, os resultados tornam as simulações com um baixo número de pontos bastante atraentes. É importante ressaltar que o coeficiente justifica-se pelo conhecimento da restrição no processo de inserção de levantes imposta na modelagem.

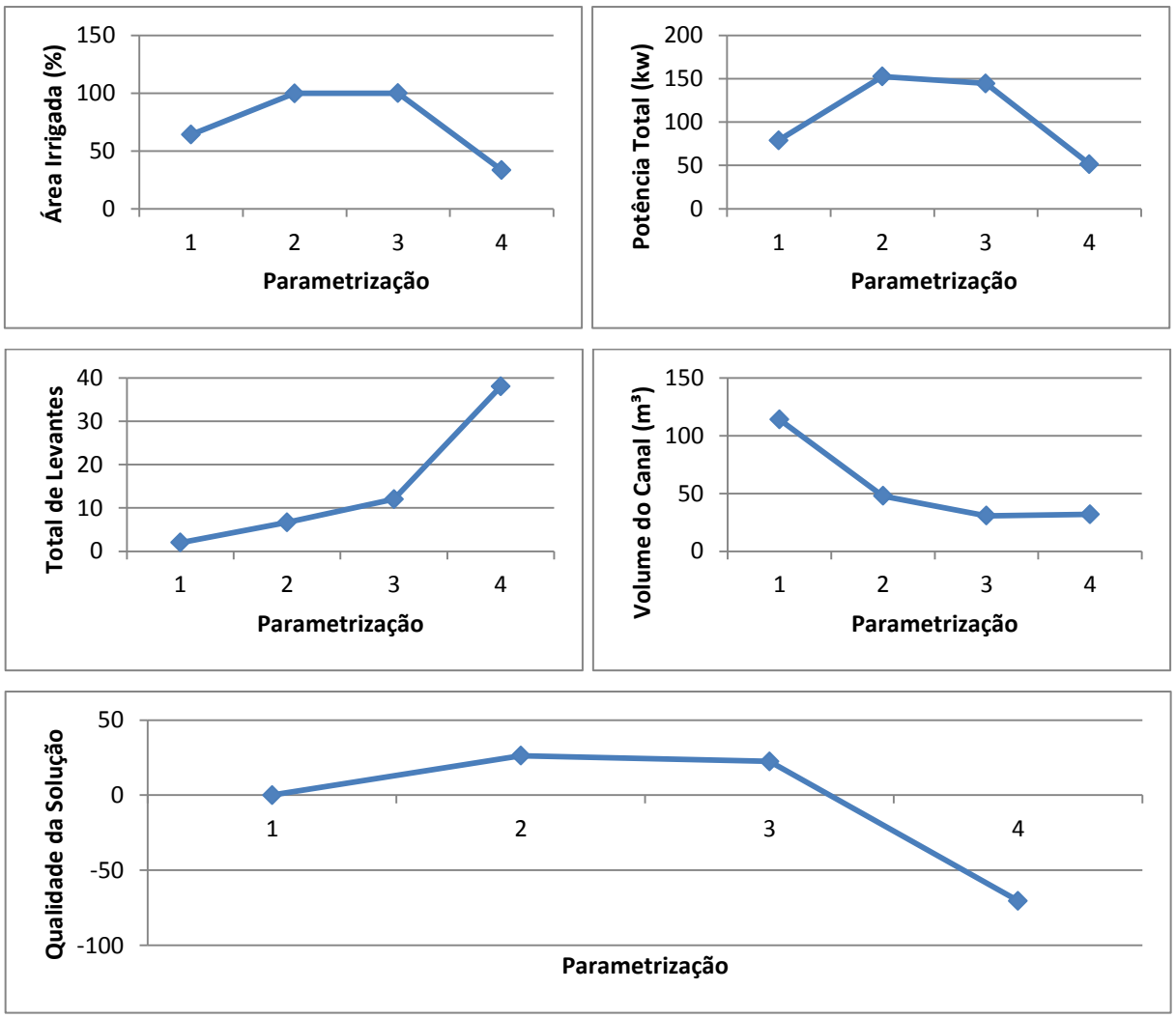
Como o desenvolvimento do modelo deu-se de maneira iterativa, à medida que este é analisado, o conhecimento sobre o problema é ampliado e novas alternativas aparecem naturalmente. Uma opção capaz de atenuar drasticamente a restrição de uma bomba apenas elevar a água entre dois pontos adjacentes e ascendentes (fluxo direcionado a uma cota superior), seria a em que ela recalcasse a água não entre dois, mas entre o conjunto de vértices ascendentes encontrados em sequência seguindo a direção de escoamento do canal. Esta sem dúvida é uma solução a ser incorporada futuramente ao modelo.

Tabela 9 – Parametrização dos experimentos em que o número de vértices do canal é a variável em foco.

Parametrização	Total de Indivíduos	Taxa de Crossover	Taxa de Mutação	Distância de Propagação	Período de utilização dos levantes	Ponto de Entrada	**Total de Vértices**
1	750	70%	3%	6 quadros	8,5 h	médio	**9**
2	750	70%	3%	6 quadros	8,5 h	médio	**25**
3	750	70%	3%	6 quadros	8,5 h	médio	**81**
4	750	70%	3%	6 quadros	8,5 h	médio	**289**

Fonte: O autor

Figura 24 – Resultados obtidos com o aumento do número de vértices do canal.



Fonte: O autor

A variação do ponto de entrada d'água (Tabela 10) exaltou outra limitação do algoritmo de otimização: a do procedimento de cálculo do volume do canal. Nos segmentos de canal onde o fluxo de água está orientado no sentido descendente

(direcionado para a conta mais baixa) o modelo contabiliza a existência de um segmento de canal. Entretanto nas situações onde a direção de escoamento do fluído é ascendente, somente o levante inserido é contabilizado, desconsiderando o canal ou tubulação existente neste intervalo. Esta restrição não é tão aparente nas situações onde o ponto de entrada d'água está localizado em um ponto a cota média (como nos demais experimentos), contudo quando comparados resultados oriundos de pontos de entrada distintos o impacto ocasionado por esta limitação no desempenho do modelo torna-se evidente.

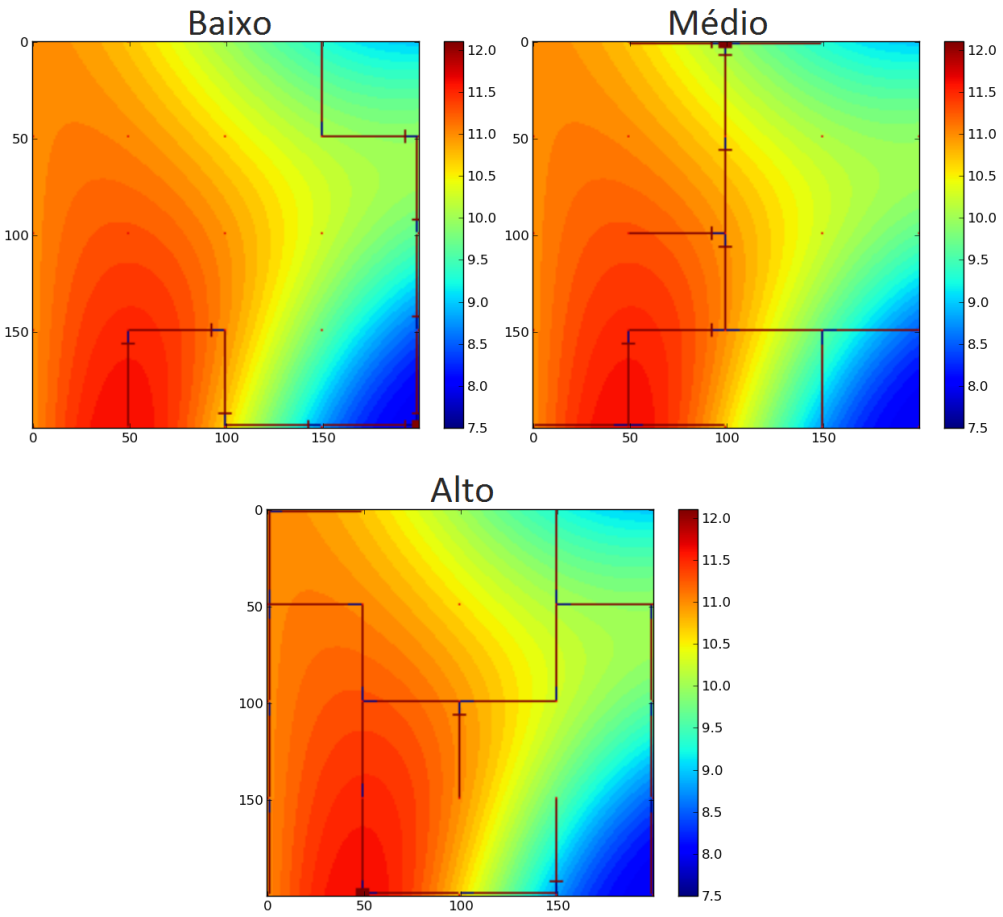
O conjunto de pontos de entrada estabelecidos na Tabela 10 pode ser visualizado nos mapas presentes na Figura 25. Os resultados obtidos por meio dos procedimentos experimentais (Figura 26) quando analisados de forma independente demonstram um comportamento coerente do modelo. A área irrigada permanece constante para as três situações (com uma variação de apenas 0,2%). A potência total e o número de levantes partem de um valor bem elevado na parametrização onde o ponto de entrada está no ponto mais baixo da lavoura (parametrização 1) e caem para zero quando o ponto de intersecção com o canal primário translada para a cota mais elevada (parametrização 3). O volume de fluído suportado pelo canal também condiz com a estratégia de modelagem, uma vez que mais segmentos de canal sem levantes (os únicos contabilizados pelo modelo) tendem a aparecer na solução onde o ponto de entrada d'água está na cota mais elevada da área. A análise conjunta das variáveis de saída demonstra uma queda na qualidade da solução conforme o ponto de intersecção da lavoura com o canal primário caminha em direção aos pontos mais elevados. Isto é justificado pela acentuada variação percentual nas dimensões do canal (volume do canal), o que diminui a sensibilidade dos resultados apontados pela equação de aptidão, situação criada pela restrição imposta ao cálculo das dimensões do canal. Problema que se tornou aparente durante a análise do modelo deve tomado como prioridade durante a continuidade da evolução do algoritmo de otimização.

Tabela 10 – Parametrização dos experimentos em que a cota do ponto de entrada é a variável em foco.

Parametrização	Total de Indivíduos	Taxa de Crossover	Taxa de Mutação	Distância de Propagação	Período de utilização dos levantes	**Ponto de Entrada**	Total de Vértices
1	750	70%	3%	6 quadros	8,5 h	**baixo**	25
2	750	70%	3%	6 quadros	8,5 h	**médio**	25
3	750	70%	3%	6 quadros	8,5 h	**alto**	25

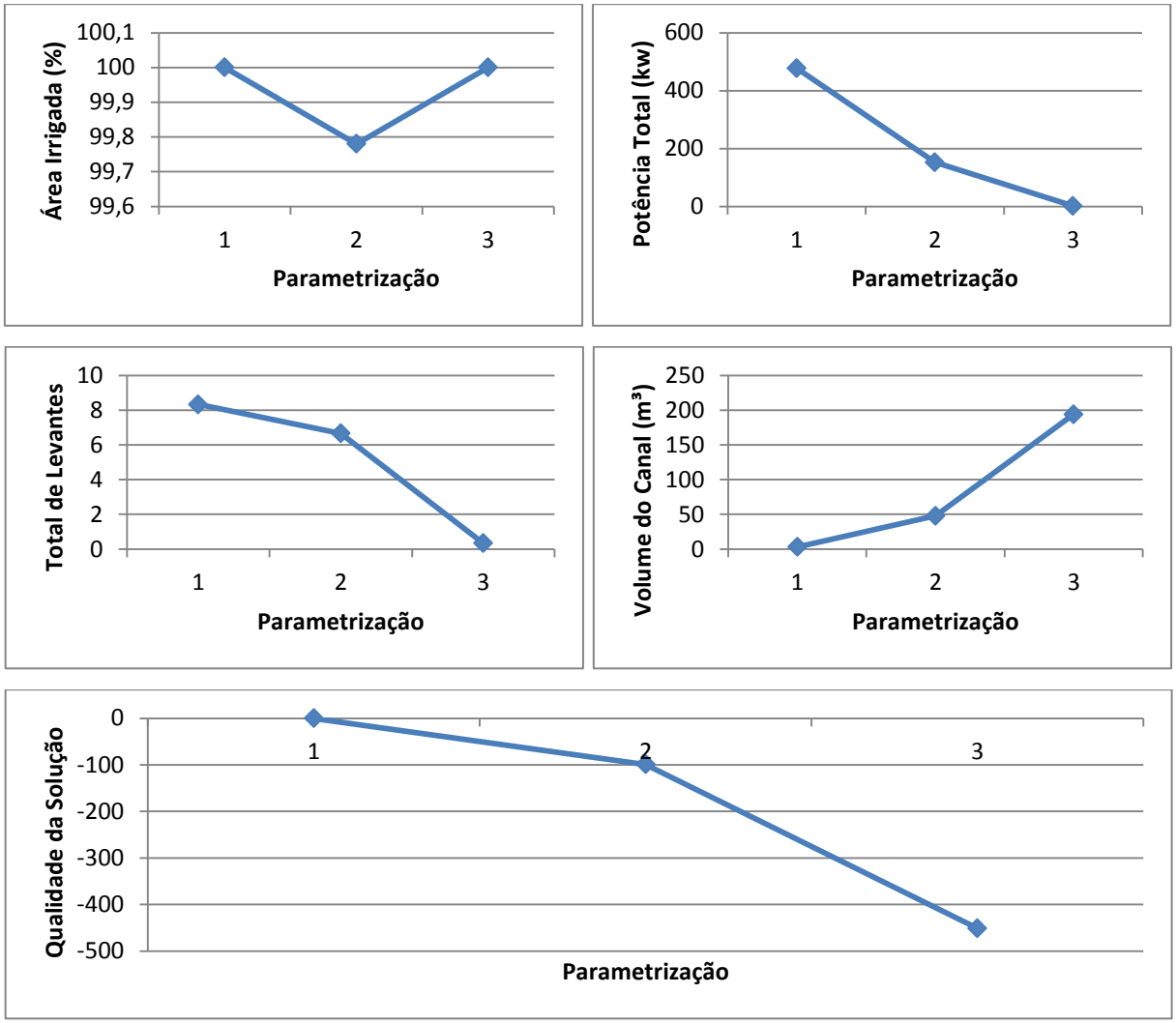
Fonte: O autor

Figura 25 – Soluções encontradas para as parametrizações descritas na Tabela 9.



Fonte: O autor

Figura 26 – Resultados obtidos com o aumento da cota do ponto de entrada d'água.



Fonte: O autor

Devido à natureza das variáveis de entrada *pa, pe, pl* e *pc* (coeficientes de prioridade da função de aptidão) três estratégias de distribuição de prioridades foram elaboradas. Na primeira estratégia (parametrizações de 1 a 12) cada um dos quatro coeficientes de prioridade assume três valores de importância (25, 55 e 85%), de maneira que os o percentual restante é distribuído uniformemente entre os demais coeficientes. Nas estratégias dois (parametrizações de 13 a 15) e três (parametrizações de 16 a 18) enquanto o coeficiente associado à área irrigada (*pa*) varia (60, 75 e 90%), o percentual de influência restante da equação de aptidão (*pe, pl* e *pc*) segue com a mesma proporção de divisão empiricamente estabelecida (e que não é equilibrada como na primeira estratégia). Desta forma por mais que o somatório destes diminua com o aumento de *pa*, a proporção de influência entre *pe,*

pl e *pc* na função para a mesma estratégia é mantida. Em todos os experimentos listados na Tabela 11, a parametrização das demais variáveis de entrada do algoritmo seguem os valores estabelecidos para a parametrização 2 da Tabela 10.

Tabela 11 – Parametrização dos experimentos em que a variação dos coeficientes de prioridade da equação de aptidão são as variáveis em foco.

Parametrização	Estratégia	*pa*	*pe*	*pl*	*pc*
1	1	**25%**	25%	25%	25%
2	1	**55%**	15%	15%	15%
3	1	**85%**	5%	5%	5%
4	1	25%	**25%**	25%	25%
5	1	15%	**55%**	15%	15%
6	1	5%	**85%**	5%	5%
7	1	25%	25%	**25%**	25%
8	1	15%	15%	**55%**	15%
9	1	5%	5%	**85%**	5%
10	1	25%	25%	25%	**25%**
11	1	15%	15%	15%	**55%**
12	1	5%	5%	5%	**85%**
13	2	**60%**	24%	4%	12%
14	2	**75%**	15%	2,50%	7,50%
15	2	**90%**	6%	1%	6%
16	3	**60%**	18%	4%	18%
17	3	**75%**	11,25%	2,50%	11,25%
18	3	**90%**	4,50%	1%	4,50%

Fonte: O autor

Em razão do fato dos coeficientes de prioridade da função de aptidão fazerem parte da *equação de avaliação conjunta das variáveis de* saída, este procedimento de análise não pode ser realizado para estas variáveis. Sendo assim a análise baseou-se apenas no procedimento individual de verificação das variáveis de saídas, de maneira que optou-se por apresentar os resultados por meio da tabela 12, já que o volume de informação a ser comparado acarretaria na análise de um conjunto elevados de gráficos, o que dificultaria a visualização dos resultados.

Do conjunto de experimentos listados na Tabela 11, as simulações que apresentaram melhor desempenho foram aquelas que o coeficiente *pa* ficou entre 55% e 60%, principalmente onde as prioridades de *pe* e *pc* foram iguais (parametrizações 2 e 16). Nas configurações onde *pa* esteve abaixo do patamar

descrito acima, a baixa importância dada ao percentual da área irrigada (parametrização 1 e parametrizações de 3 a 11) fez com que o canal secundário deixasse de existir, zerando as demais variáveis e consequentemente irrigando apenas 20% da área. Para valores de *pa* superiores a 60% observou-se que uma pequena melhora na área irrigada acarreta em impacto negativo considerável no comportamento das demais variáveis. Ao confrontar as estratégias de distribuição empíricas (2 e 3) as configurações da estratégia 3 se mostram mais sensíveis ao aumento de *pa*, piorando o resultado conforme esta alteração. Contudo, o melhor resultado dentre todas as parametrizações listadas na Tabela 11 foi alcançado com a primeira configuração pertencente a esta estratégia (parametrização 16).

Tabela 12 – Resultados obtidos com a variação dos coeficientes de prioridade da equação de aptidão.

Parametrização	Área Irrigada (%)	Potência Total (kw)	Total de Levantes	Volume do Canal (m^3)
1	20,00	0,00	0,00	0,00
2	93,99	150,11	3,33	9,56
3	99,78	163,39	5,00	49,58
4	20,00	0,00	0,00	0,00
5	20,00	0,00	0,00	0,00
6	20,00	0,00	0,00	0,00
7	20,00	0,00	0,00	0,00
8	20,00	0,00	0,00	0,00
9	20,00	0,00	0,00	0,00
10	20,00	0,00	0,00	0,00
11	20,00	0,00	0,00	0,00
12	27,77	9,62	0,33	0,00
13	98,62	155,10	5,33	26,83
14	99,78	152,51	6,67	47,82
15	100,00	155,79	7,00	48,51
16	95,33	143,50	4,67	11,71
17	98,62	157,35	5,67	25,38
18	100,00	158,30	6,33	52,04

Fonte: O autor

Conforme na Seção 2.1, a eficiência hídrica da irrigação em lavouras de arroz varia entre 40 a 60% (EMBRAPA, 2005), isto é, de todo o volume de água fornecido à lavoura apenas esta fração é aproveitada pela planta. O algoritmo de otimização apresentou um rendimento de cerca de 68%, o que é justificado pelas abstrações

contidas no modelo. As formas de perda consideradas pelo modelo hidrológico se devem aos fenômenos de infiltração (percolação e fluxo lateral), evapotranspiração e descarte do volume de água acumulado sobre a superfície da lavoura que, drenado ou não, após o período de irrigação não pode ser aproveitado pela planta. Dentre as fontes de perda não consideradas estão as perdas em canais e drenos, fatores que tendem a aumentar a taxa de fluxo lateral e, consequentemente o desperdício. Generalizações como manejo da irrigação (considerado com ideal pelo modelo) e as taxas de infiltração e evapotranspiração (tratadas como constante durante todo o ciclo de irrigação) também são fontes de perda desconsideradas pelo modelo. Sendo assim, a discrepância entre os valores práticos e os apresentados pelo algoritmo estão coerentes com a precisão do algoritmo possui no que se refere a modelagem matemática dos fenômenos físicos, bem como a etapa de desenvolvimento em que o mesmo se encontra.

Dentre todos os aspectos modelados pelo algoritmo de otimização sem dívida o que apresentou piores resultados foi o cálculo da demanda hídrica do projeto. De acordo com Stone (2005) a vazão requerida pela lavoura durante o período de manutenção da lâmina d'água desconsiderando perdas em canais e precipitações pluviais é de aproximadamente 1,15 a 1,76 $l.s^{-1}.ha^{-1}$. As equações utilizadas pelo modelo hidrológico forneceram valores de vazão cerca de dez vezes superiores a estes, o que é uma disparidade inaceitável. A fonte deste conjunto de equações foi Mello (2009), escolha justificada pelo baixo número de parâmetros requisitados. Sendo assim, a substituição destas equações faz parte das prioridades dos trabalhos a serem realizados futuramente com o algoritmo. Entretanto, esta falha de modelagem não compromete os resultados discutidos anteriormente, pois uma vez estabelecida a demanda, esta é apenas uma constante e não interfere no comportamento do algoritmo. De maneira que o superdimensionamento dos levantes e canais é o único impacto que este erro de modelagem tem sobre as soluções apresentadas acima. O que quer dizer que os sistemas de irrigações expostos acima estão projetados para atender a demanda referente a uma área de cerca de mil hectares ao invés de cem, como a área definida para as simulações.

4 CONSIDERAÇÕES FINAIS

4.1 Conclusões

O algoritmo de otimização apresentado neste trabalho integra modelagem matemática e computação evolutiva, de maneira a auxiliar na tomada de decisões de grande impacto no projeto de sistemas de irrigação por inundação de lavouras de arroz. A possibilidade de simular diferentes configurações de implantação dá flexibilidade ao projeto, permitindo a quantificação de recursos e a avaliação do impacto que determinada decisão tem no sistema como um todo. Durante a evolução da solução, o algoritmo busca a configuração de canais que satisfaz o funcionamento correto do sistema de irrigação da forma mais eficiente, considerando as características particulares (dimensões e relevo) da área de cultivo selecionada.

O algoritmo está restrito à otimização dos componentes do sistema de irrigação contidos dentro da área do mapa da lavoura. Isto quer dizer que os elementos do sistema presentes no caminho percorrido pelo fluido entre o reservatório e a lavoura, bem como os relacionados a sistema de drenagem, fogem ao escopo deste trabalho. Desta forma o foco de desenvolvimento manteve-se direcionado na topologia da rede de canais internos da lavoura, a qual é responsável pela distribuição entre os quadros.

Uma descrição completa da solução encontrada é devolvida pelo algoritmo de otimização, o que é feito por meio de relatórios e mapas. O conteúdo destas saídas além de possibilitar a implantação do sistema auxilia no manejo da irrigação, uma vez que o fluxo de água entre os componentes do sistema é detalhado.

As duas formas possíveis de propagação da água entre os elementos do sistema de irrigação, que são do canal para os quadros e de um quadro para outro são consideras pelo algoritmo. No caso da passagem de fluido entre quadros aspectos como a adjacência, a cota dos tabuleiros envolvidos e a perda de carga ocorrida durante propagação são contabilizados, de maneira que este o processo demonstra coerência com a realidade.

O canal de irrigação é modelado pelo algoritmo como um grafo, isto é, como uma coleção de vértices que podem ser interligados por arestas, que formam segmentos de canal. Os vértices consistem em um conjunto de pontos simetricamente distribuídos, cuja densidade é estabelecida pelo usuário. Esta

possibilidade de ajuste permite níveis distintos de refinamento do canal sejam testados, bem como faz com que o algoritmo possa ser utilizado para áreas de dimensões completamente diferentes (de dezenas a milhares de hectares), o que demonstra a versatilidade do algoritmo.

Na avaliação das soluções, perdas por área não irrigada e custos relacionados à operação e implantação dos componentes do sistema são considerados. Nos sistemas de levante hidráulico, tanto gastos dependentes da potência (variáveis) quanto fixos (que independem do tamanho da moto-bomba) são ponderados pelo algoritmo. Os custos associados ao canal também estão presentes na equação de aptidão, o que é feito por meio de variáveis que refletem as dimensões deste componente.

A opção por estabelecer limites variáveis para o cálculo das notas durante o processo de avaliação se mostrou uma ótima alternativa para a comparação entre soluções. Uma vez que as notas atribuídas aos indivíduos necessitam apresentar coerência apenas para uma dada geração, de maneira que a evolução das soluções verificada pela melhoria nos valores das variáveis de aptidão, e não das notas. Esta decisão ampliou a precisão na comparação entre indivíduos, e contribuiu de forma significativa para os bons resultados apresentados pelo algoritmo na evolução das soluções.

Durante o procedimento experimental realizado, um conjunto de variáveis de entrada foi selecionado para ser submetido a alterações, de modo que enquanto uma variável era alterada os valores assumidos pelas demais eram mantidos inalterados. O foco deste método era provocar reações do algoritmo que demonstrassem se comportamento deste é compatível ou não com a conduta esperada do sistema real.

Os resultados obtidos pelos experimentos onde apenas o tamanho da população foi alterado demonstraram que uma porção maior do espaço de soluções é considerada pelo algoritmo quando o valor desta variável ampliado. Já nas parametrizações onde a variação da taxa de *crossover* foi considerada, resultados inferiores foram observados para o aumento desta variável, o que leva a considerar o fato de que a técnica de *crossover* selecionada talvez não seja a ideal para a estrutura para o cromossomo do algoritmo. O aumento na taxa de *mutação* atendeu às expectativas, pois o algoritmo exibiu uma melhora clara nas soluções

encontradas, comportamento justificado pelo aumento na diversidade da população promovido por esta alteração.

O estabelecimento de um número máximo de quadros irrigados por um único vértice é a forma utilizada pelo modelo para representar a perda de carga existente no processo de propagação entre quadros. Conforme esperado o aumento do valor atribuído a esta variável de entrada resultou em uma redução significativa nas dimensões do canal e nos custos associados aos levantes.

A ampliação do período de funcionamento diário dos recalques hidráulicos não acarretou em variação significativa do percentual da área irrigada e do total de levantes, reduzindo apenas a potência total das soluções e o volume do canal. Resultado aguardado, pois com um período maior para o sistema satisfazer a demanda hídrica diária da lavoura, as vazões e, consequentemente, as dimensões dos componentes devem diminuir.

Como a variação do número de vértices acarreta em vantagens e desvantagens (descritas na Seção 2.3), era esperado que os resultados apontassem que um meio termo na parametrização traria os melhores resultados. Satisfazendo as expectativas, o algoritmo indicou que o total de 25 vértices favorece a seleção de soluções de melhor qualidade por parte do algoritmo.

Dentre o conjunto de soluções elaboradas para modelar o comportamento do sistema de irrigação por inundação, o único aspecto que demonstrou conduta negativa foi o procedimento de cálculo das dimensões do canal. Esta falha de projeto foi observada na análise dos experimentos em que apenas o ponto de entrada d'água na lavoura foi alterado, pois a acentuada variação percentual nas dimensões do canal (volume do canal) devido a uma restrição imposta ao cálculo das dimensões do canal (descrita na Seção 3.3) diminui a sensibilidade dos resultados apontados pela equação de aptidão.

A avaliação do comportamento do modelo frente ao ajuste dos coeficientes de prioridade da equação de aptidão indicou que importância dada ao percentual da área irrigada deve ser superior a 50%, já que percentuais inferiores a estes fazem com que o custo associado ao canal secundário aumente ao ponto deste não ser inserido na área pela solução. Os resultados mostraram que a parametrização destas variáveis é um processo delicado, e que várias tentativas devem ser realizadas até que o melhor resultado seja alcançado.

A análise de sensibilidade dos experimentos apontou que o comportamento do algoritmo de otimização foi satisfatório, uma vez que os resultados obtidos foram coincidentes com os esperados de um sistema de irrigação por inundação real. O desempenho evolutivo indicou grande facilidade de convergência, o que demonstra o funcionamento correto do algoritmo genético e evidencia a necessidade de que o tamanho estabelecido para a população deve ser elevado, de modo que valores superiores aos utilizados durante os experimentos são indicados para que o espaço de soluções seja mais bem explorado pelo algoritmo.

A comparação entre a demanda hídrica teórica e a calculada pelo algoritmo não apresentou um bom resultado, já que o valor estimado pelo modelo foi cerca de dez vezes superior ao teórico. Isto confirma o fato de as equações utilizadas pelo algoritmo de otimização não estão corretas, e sua substituição é necessária para que o dimensionamento dos componentes do sistema de irrigação esteja correto.

Sendo assim, o objetivo desta monografia que é o de demonstrar que o algoritmo de otimização desenvolvido apresenta comportamento compatível com o do sistema real modelado foi alcançado. De modo que a continuidade do processo de desenvolvimento deste trabalho pode trazer resultados satisfatórios no auxílio ao projeto de sistemas de irrigação para lavouras de arroz.

4.2 Trabalhos futuros

Conforme os resultados apontaram, o único procedimento que não apresentou resultados positivos foi o cálculo das dimensões do canal. Pois em segmentos de canal onde uma moto-bomba é implantada o canal (ou encanamento) que liga esta ao próximo vértice no sentido de fluxo da água é desconsiderado pelo algoritmo, favorecendo a inserção de levantes (que possuem maior custo) no lugar de segmentos de canal que escoam água por gravidade. Desta forma, a contabilização durante a avaliação do custo referente a esta fração de canal (desconsiderada até então) é prioridade na evolução do algoritmo de otimização.

Outra solução de modelagem descrita na Seção 3.1 que deve receber maior atenção é a inserção de recalques hidráulicos. Em todo o segmento de canal em que fluxo de água é ascendente (direcionado para cima) o algoritmo insere um levante para atender a demanda hídrica do vértice à jusante. Isto acarreta na inserção de um número desnecessário de moto-bombas na solução. Uma alternativa

a esta abordagem é a de que em situações onde segmentos de canal subsequentes com fluxo ascendente devem receber apenas um levante hidráulico. Contudo, outras soluções mais complexas devem ser consideradas para que esta restrição seja contornada da melhor forma possível.

O próximo passo na evolução do algoritmo é o de refinar as equações que estabelecem a demanda hídrica da lavoura, já que os resultados obtidos neste aspecto não foram positivos. A complexidade imposta na solução deste problema certamente não está em seu desenvolvimento, e sim na definição dos valores assumidos pelo conjunto de variáveis presentes nestas equações, uma vez que estes dependem da área de plantio e geralmente são geralmente por métodos de análise do terreno.

A perda de carga existente no escoamento de água entre quadros é representada no algoritmo por meio de estabelecimento de um número máximo (limite fixo) de quadros que o fluído pode percorrer a partir do canal de irrigação. Encontrar uma forma matemática de representar este fenômeno é um passo essencial que deve ser tomado na evolução do algoritmo, para que este possa efetivamente ser utilizado como ferramenta de auxílio no projeto de sistemas de irrigação por inundação.

Vários modelos de otimização hidrológica encontrados durante a etapa de levantamento bibliográfico partilhavam de uma característica em comum: eles utilizam valores expressos em moeda para quantificar os custos associados às variáveis de saída do algoritmo de otimização. Esta solução é uma ótima forma de ampliar a precisão dos resultados, bem como facilitar a atividade de definição dos coeficientes de proporcionalidade presentes na equação de aptidão.

Embora os mapas fornecidos pelo algoritmo representem os componentes do sistema de irrigação, não existe um mapa que demonstre que quadros exatamente foram atingidos durante o manejo da irrigação com a solução sugerida. Estas informações são detalhadas nos relatórios via planilhas no formato *xls*, porém esta forma de representação dificulta a visualização dos resultados. Sendo assim, é necessário que um mapa orientado a apresentação desta informação seja desenvolvido.

REFERÊNCIAS

AGUIAR, Marilton. **Análise Formal da Complexidade de Algoritmos Genéticos**. Porto Alegre, 1998.

ANDRADE, Camilo et al. **Modelo computacional para suporte à decisão em áreas irrigadas. Parte I: Desenvolvimento e análise de sensibilidade**. In: REVISTA BRASILEIRA DE ENGENHARIA AGRÍCOLA E AMBIENTA, 2008, Campina Grande.

ANUÁRIO BRASILEIRO DO ARROZ. Santa Cruz do Sul: Gazeta, 2013.

ARGOSFOTO, Disponível: http://www.images.argosfoto.com.br/. Acesso em 10 de maio de 2012.

BELISÁRIO, A. Thomé et al. **Gerenciamento Automático de Sistemas de Bombeamento Para Irrigação de Lavouras de Arroz a partir do Controle da Lâmina d'Água**. Alegrete, 2011.

BOAS, Mariana. **Modelo de Simulação de Sistemas Hídricos Complexos, Integrado Com a Avaliação de Qualidade da Água**: uma ferramenta de gestão para apoio a decisão. Rio de Janeiro, 2008.

BASSO, Luiz. **ESCOAMENTO À SUPERFÍCIE LIVRE (CANAIS)**. Cascavel, 2012.

BUENO, Marcos. **Heurística e Algoritmos Evolutivos Para Formulações Mono e Multiobjetivo do Problema do Roteamento Multicast**. Uberlândia. 2010. 23.

CARVALHO, Daniel; MELLO, Jorge; SILVA, Leonardo. **HIDROLOGIA**. Rio de Janeiro, 2009.

CASTRO, Nilza. **Apostila de Irrigação. Porto Alegre**, 2003.

ÇENGEL, Y.A. e Cimbala, J.M. 2007**. Mecânica dos Fluidos - Fundamentos e Aplicações**, McGraw-Hill Interamericana do Brasil Ltda, 819 p.

CHAVES, Marcelo. **Modelos Digitais e Elevação Hidrológicamente Consistentes para a Bacia Amazônica**. Viçosa. 2002.

CORRÊA, Henrique. **Adequação da Demanda Hídrica e da Potência Instalada em Sistemas de Recalque para o Arroz Na Depressão Central do Rio Grande do Sul**. Santa Maria, 2007.

EMBRAPA. Cultivo do arroz irrigado no Brasil Sistemas de Produção, **Embrapa Clima Temperado**. Versão Eletrônica, 2005.

FERRAZ, Fernando. **Modelos Hidrológicos Acoplados a Sistemas de Informações Geográficas:** Um Estudo de Caso. Piracicaba: Editora Unimep. Novembro, 1999.

FERREIRA, Nilson. **Apostila de Sistema de Informações Geográficas**. Goiânia. 2006.

FINGER, Maria. **Percepção e Medidas de Gestão de Riscos por Produtores de Arroz Irrigado na Fronteira-oeste do Rio Grande do Sul**. Porto Alegre, 2012.

Fundamento de Geoprocessamento, Disponível: http://www.ltc.ufes.br/. Acesso em 20 de setembro de 2013.

GUIMARÃES, José; PATTA, Juliano; SORIANO, Marcelo. A Agricultura de Informação e as Estações de Bombeamento para o Cultivo do Arroz Irrigado por Inundação. In: SIMPÓSIO BRASILEIRO DE RECURSOS HÍDRICOS, 17., 2007, São Paulo.

IBGE. **Censo Agropecuário 2006**. Rio de Janeiro, 2006.

INFOBIBOS, Disponível: http://www.infobibos.com/. Acesso em 4 de setembro de 2013.

LIMA, Letícia. **Implementação de um Modelo Hidrológico Distribuído na Plataforma de Modelagem Dinâmica Ego**. Belo Horizonte, 2008.

LOUZADA, José. **Simulação da Irrigação por Inundação e da Drenagem nos Solos de Várzea do Rio Grande do Sul**. Porto Alegre, 2004.

LEWIS, Harry R.; PAPADIMITRIOU, Christos H. **Elementos de Teoria da Computação**. 2º ed. 2004, Bookman.

MARCUZZO, Francisco. **SISTEMA DE OTIMIZAÇÃO HIDRÁULICA E ECONÔMICA DE REDE DE IRRIGAÇÃO LOCALIZADA USANDO ALGORITMOS GENÉTICOS**. São Carlos, 2008.

MELLO, Jorge; SILVA, Leonardo. **IRRIGAÇÃO**. Rio de Janeiro, 2009.

MELO, Ten. **Emprego de Algoritmos Genéticos na Generalização e Modelos Digitais de Superfície**. Rio de Janeiro. 2008.

MITCHELL, Melanie. **An Introduction to Genetic Algorithms** . Mit Press paperback. 1996. 20.

MOREIRA, Itamar. **Modelagem Hidrológica Chuva-Vazão com Dados de Radar e Pluviômetros**. Curitiba, 2005.

PACHECO, Marco. **Algoritmos Genéticos: Princípios e Aplicações**. Rio de Janeiro, 1999.

PESCADOR, Suzana. **Entendendo P Versus NP** . Florianópolis, 2009.

PYTHON, Disponível: http://www.python.org/. Acesso em 10 de maio de 2014.

RENNÓ, Camilo. **Construção de um Sistema de Análise e Simulação Hidrológica: Aplicado a Bacias Hidrográficas**. São José dos Campos. 2004.

REUNIÃO TÉCNICA DA CULTURA DO ARROZ IRRIGADO, 28., 2010, Bento Gonçalves. **Arroz Irrigado**: recomendações técnicas da pesquisa para o Sul do Brasil, Porto Alegre, IRGA 2010.

RIGHES, Afranio Almir. **Eficiência em Sistemas com Inundação o Caso do Arroz**. In: SIMPÓSIO NACIONAL SOBRE O USO DA ÁGUA NA AGRICULTURA , 2006, Passo Fundo.

SILVA, ALYSSON C.A. **Calibração Automática de Rugosidade de Tubulações em Sistemas de Distribuição de Água com Aplicação de Algoritmos Genéticos**.

Dissertação (Mestrado em Engenharia – Área de Concentração: Recursos Hídricos). Universidade Federal do Ceará. Fortaleza. 2006. 32.

SIMPÓSIO BRASILEIRO DE SENSORIAMENTO REMOTO, 2003, Belo Horizonte. **Introdução à Modelagem Dinâmica Espacial**, São José dos Campos, INPE 2003.

SOARES, Gustavo. **Algoritmos Genéticos: Estudo, Novas Técnicas e Aplicações**. Belo Horizonte. 1997.

Sociedade Sul-Brasileira de Arroz Irrigado. **Arroz Irrigado**: recomendações técnicas da pesquisa para o Sul do Brasil. Bento Gonçalves, 2010.

STONE, Luiz. **Eficiência do Uso da Água na Cultura do Arroz Irrigado** . Santo Antônio de Goiás , 2005.

VIEIRA, Allan. **Um Modelo de Simulação via Programação Linear Sequencial, Para Sistema de Recursos Hídricos**. Campina Grande, 2007.

ZUBEN, Fernando. **Computação Evolutiva**: Uma Abordagem Pragmática . 2000.

APÊNDICE A – Código fonte do algoritmo

```
import sys
sys.path.append('/home/alex/networkx-1.8.1')
import networkx as nx
import numpy
import math
import matplotlib.pyplot
import matplotlib.pylab
import random
import xlwt
import xlrd
import time

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ VARIAVEIS DE ENTRADA
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ MAPA DA AREA CULTIVADA
MDE = 'MDE3'    #equacao contida na funcao criaMde()
V = 200         #num linhas do mapa
H = 200         #num colunas do mapa
A_CELULA = 5    #area de cada celula (m quadrado)
VERT = 16       #numero de vertices da area cultivada (>=8 E MULT DE 2)
ENTRA_L = 0     #componente LINHA do pto de entrada da agua oriunda do reservatorio (0 == NORTE, V ==
SUL)
ENTRA_C = 99    #componente COLUNA do pto de entrada da agua oriunda do reservatorio  (0 == OESTE, H ==
LESTE)
DH = 0.1        #delta h de cada quadro de irrigacao

#cria o MODELO DIGITAL DE ELEVACAO
def criaMde():
        mde = numpy.empty((V,H))
        passoi = numpy.pi/(V*2)
        passoj = numpy.pi/V
        for i in range(V):
                for j in range(H):
                        xni = passoi * i
                        xnj = passoj * j
                        yn = numpy.sin(xni*xnj) + numpy.cos(xnj) + 10
                        mde[i,j] = yn

        #gera o mapa do mde
        matplotlib.pyplot.imshow(mde) #Needs to be in row,col
        matplotlib.pyplot.colorbar()
        matplotlib.pyplot.savefig("saidas/mde.png")#plt.savefig("mde.png")
        matplotlib.pyplot.clf()

        return mde

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ PARAMETROS AGRONOMICOS
M = 0.02#coeficiente de chezy-manning
HL = 0.075      #altura lamina dagua lavoura (m)
G_MAX = 6       #numero maximo de quadros q a agua pode percorrer a aprtir do vertice do canal
T1 = 5          #periodo de enchimento da lavoura (dias)
T2 = 90         #periodo de manutencao da lamina (dias)
FUNC = 8.5      #periodo diario de funcionamento recalques (hs)

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ PARAMETROS GENETICOS
POP = 1#tamanho da populacao o AG
TAXA_CROS = 70          #taxa de crossover
TAXA_MUT = 3    #taxa de mutacao
N_GENES = 2     #numero de genes atingidos durante a mutacao
N_ITER = 0      #numero de iteracoes do AG

PesoA = 0.05
PesoE = 0.05
```

```
PesoL = 0.05
PesoC = 0.85

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ CONSTANTES
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
G = 9.81#aceleracao da gravidade
Y = 9790.4        #peso especifico da agua (N/m cubico)
N = 0.6           #coeficinete de rendimento do conj moto-bomba (%)
E = 0.006         #taxa de evaporacao media (m/dia)
ET = 0.0099       #taxa de evapotranspiracao media (m/dia)
Vi = 0.2268       #taxa de infiltracao (m/dia)

#+++++++++++++++++++++++++++++++++++++++++++++++ TRATAMENTO DOS DADOS DE ENTRADA
+++++++++++++++++++++++++++++++++++++++++++++++

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++ SISTEMATIZACAO O MDE

#cria a matriz com a area cotada
def discrMde(mde):
        cotas = numpy.zeros((V,H))
        maximo = mde.max()

        for l in range(V):
                for c in range (H):
                        a = int((maximo-mde[l,c])/DH)                   #intervalo de discretizacao
                        cotas[l,c] = maximo - (a*DH) - (DH/2.0)         #valor medio do intervalo

        #gera o mapa da area cotada
        matplotlib.pyplot.imshow(cotas) #Needs to be in row,col
        matplotlib.pyplot.colorbar()
        matplotlib.pyplot.savefig("saidas/MapaCotas.png")
        matplotlib.pyplot.clf()

        return cotas

def substitui(quadros, vAnt,vNovo,x):
        for i in range(x+1):
                for j in range(H):
                        if (quadros[i,j] == vAnt):
                                quadros[i,j] = vNovo

def corrigeQuadros(lista):

        i=0
        while (i<len(lista[1])):
                if (lista[1][i] == -1):
                        #exclui da matriz
                        for j in range(V):
                                for k in range(H):
                                        if (lista[0][j,k] > i):
                                                lista[0][j,k] = lista[0][j,k]-1

                        #exclui do vetor
                        lista[1].remove(-1)
                else:
                        i=i+1

        return lista

def calcArea(quadros,vet,cotas):
        info = numpy.zeros((len(vet),2)) #info[cota,area]

        for i in range(len(vet)):
                info[i,0] = vet[i]

        for i in range(V):
                for j in range(H):
```

```
                                info[quadros[i,j],1] = info[quadros[i,j],1] + A_CELULA

        return [quadros,info,cotas]

#cria o MAPA DA AREA SISTEMATIZADA
def sistMde(mde):
        quadros = numpy.zeros((V,H))
        quadros = quadros.astype(int)

        cotas = discrMde(mde)

        for i in range(V):
                for j in range(H):
                        quadros[i,j] = -1

        for i in range(V):
                for j in range(H):
                        if (i==0 and j==0):
                                quadros[i,j] = 0
                                vet = [cotas[0,0]]

                        elif (i==0 and j>0):
                                if (cotas[i,j-1] == cotas[i,j]):
                                        quadros[i,j] = quadros[i,j-1] #recebe o mesmo nome do vizinho
esquerdo
                                else:
                                        vet += [cotas[i,j]] #adicona outra cota ao vetor de nomes
                                        quadros[i,j] = len(vet)-1 #adiciona novo nome a celula

                        elif (i>0 and j==0):
                                if (cotas[i-1,j] == cotas[i,j]):
                                        quadros[i,j] = quadros[i-1,j] #recebe o mesmo nome do vizinho
superior
                                if (cotas[i-1,j+1] == cotas[i,j]):
                                        if (quadros[i,j] == -1):
                                                quadros[i,j] = quadros[i-1,j+1] #recebe o mesmo nome do
vizinho superior direito
                                        #se o nome do quadro armazenado for diferente do encontrado
                                        elif (quadros[i,j] != quadros[i-1,j+1]):
                                                #substitui todas as celulas que tiverem o maior "nome" de
quadro pelo menor "nome"
                                                #e elimina este valor do vetor (substitui a cota por -1)
                                                if (quadros[i,j] < quadros[i-1,j+1]):
                                                        vet[quadros[i-1,j+1]] = -1.0
                                                        substitui(quadros, quadros[i-1,j+1], quadros[i,j],i)
                                                else:
                                                        vet[quadros[i,j]] = -1.0
                                                        substitui(quadros, quadros[i,j], quadros[i-1,j+1],i)
                                if(quadros[i,j] == -1):
                                        vet += [cotas[i,j]] #adiciona outra cota ao vetor de nomes
                                        quadros[i,j] = len(vet)-1 #adiciona novo nome a celula

                        elif (i>0 and j==(H-1)):
                                if (cotas[i-1,j-1] == cotas[i,j]):
                                        quadros[i,j] = quadros[i-1,j-1] #recebe o mesmo nome do vizinho
superior esquerdo
                                if(cotas[i-1,j] == cotas[i,j]):
                                        if (quadros[i,j] == -1):
                                                quadros[i,j] = quadros[i-1,j] #recebe o mesmo nome do
vizinho superior direito
                                        #se o nome do quadro armazenado for diferente do encontrado
                                        elif (quadros[i,j] != quadros[i-1,j]):
                                                #substitui todas as celulas que tiverem o maior "nome" de
quadro pelo menor "nome"
                                                #e elimina este valor do vetor (substitui a cota por -1)
                                                if (quadros[i,j] < quadros[i-1,j]):
                                                        vet[quadros[i-1,j]] = -1.0
```

```
                                        substitui(quadros, quadros[i-1,j], quadros[i,j],i)
                                    else:
                                        vet[quadros[i,j]] = -1.0
                                        substitui(quadros, quadros[i,j], quadros[i-1,j],i)

                            if (cotas[i,j-1] == cotas[i,j]):
                                if (quadros[i,j] == -1):
                                    quadros[i,j] = quadros[i,j-1] #recebe o mesmo nome do
vizinho superior direito
                                #se o nome do quadro armazenado for diferente do encontrado
                                elif (quadros[i,j] != quadros[i,j-1]):
                                    #substitui todas as celulas que tiverem o maior "nome" de
quadro pelo menor "nome"
                                    #e elimina este valor do vetor (substitui a cota por -1)
                                    if (quadros[i,j] < quadros[i,j-1]):
                                        vet[quadros[i,j-1]] = -1.0
                                        substitui(quadros, quadros[i,j-1], quadros[i,j],i)
                                    else:
                                        vet[quadros[i,j]] = -1.0
                                        substitui(quadros, quadros[i,j], quadros[i,j-1],i)

                            if(quadros[i,j] == -1):
                                vet += [cotas[i,j]] #adiciona outra cota ao vetor de nomes
                                quadros[i,j] = len(vet)-1 #adiciona novo nome a celula

                        #i>0 and j>0
                        else:
                            if (cotas[i,j-1] == cotas[i,j]):
                                quadros[i,j] = quadros[i,j-1] #recebe o mesmo nome do vizinho
esquerdo
                            if (cotas[i-1,j-1] == cotas[i,j]):
                                if (quadros[i,j] == -1):
                                    quadros[i,j] = quadros[i-1,j-1] #recebe o mesmo nome do
vizinho superior esquerdo
                                #se o nome do quadro armazenado for diferente do encontrado
                                elif (quadros[i,j] != quadros[i-1,j-1]):
                                    #substitui todas as celulas que tiverem o maior "nome" de
quadro pelo menor "nome"
                                    #e elimina este valor do vetor (substitui a cota por -1)
                                    if (quadros[i,j] < quadros[i-1,j-1]):
                                        vet[quadros[i-1,j-1]] = -1.0
                                        substitui(quadros, quadros[i-1,j-1], quadros[i,j],i)
                                    else:
                                        vet[quadros[i,j]] = -1.0
                                        substitui(quadros, quadros[i,j], quadros[i-1,j-1],i)


                            if(cotas[i-1,j] == cotas[i,j]):
                                if (quadros[i,j] == -1):
                                    quadros[i,j] = quadros[i-1,j] #recebe o mesmo nome do
vizinho superior direito
                                #se o nome do quadro armazenado for diferente do encontrado
                                elif (quadros[i,j] != quadros[i-1,j]):
                                    #substitui todas as celulas que tiverem o maior "nome" de
quadro pelo menor "nome"
                                    #e elimina este valor do vetor (substitui a cota por -1)
                                    if (quadros[i,j] < quadros[i-1,j]):
                                        vet[quadros[i-1,j]] = -1.0
                                        substitui(quadros, quadros[i-1,j], quadros[i,j],i)
                                    else:
                                        vet[quadros[i,j]] = -1.0
                                        substitui(quadros, quadros[i,j], quadros[i-1,j],i)


                            if (cotas[i-1,j+1] == cotas[i,j]):
                                if (quadros[i,j] == -1):
```

```
                                                quadros[i,j] = quadros[i-1,j+1] #recebe o mesmo nome do
vizinho superior direito
                                        #se o nome do quadro armazenado for diferente do encontrado
                                        elif (quadros[i,j] != quadros[i-1,j+1]):
                                                #substitui todas as celulas que tiverem o maior "nome" de
quadro pelo menor "nome"
                                                #e elimina este valor do vetor (substitui a cota por -1)
                                                if (quadros[i,j] < quadros[i-1,j+1]):
                                                        vet[quadros[i-1,j+1]] = -1.0
                                                        substitui(quadros, quadros[i-1,j+1], quadros[i,j],i)
                                                else:
                                                        vet[quadros[i,j]] = -1.0
                                                        substitui(quadros, quadros[i,j], quadros[i-1,j+1],i)


                                if(quadros[i,j] == -1):
                                        vet += [cotas[i,j]] #adiciona outra cota ao vetor de nomes
                                        quadros[i,j] = len(vet)-1 #adiciona novo nome a celula

        lista = [quadros,vet,cotas]
        lista = corrigeQuadros(lista)
        lista = calcArea(lista[0],lista[1],lista[2])


        #gera o mapa da area sistematizada (quadros)
        matplotlib.pyplot.imshow(quadros)
        matplotlib.pyplot.colorbar()
        matplotlib.pyplot.savefig("saidas/MapaSist.png")
        matplotlib.pyplot.clf()

        return lista

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ CRIACAO DAS EDs ADICIONAIS

def complGrafo(g,q1,q2):
        g[q1,q2] = 1
        g[q2,q1] = 1

        return g

#cria uma matriz contendo as adjacencias entre os quadros de irrigacao
def buscaAdjacQuadros(sist):
        quadros = sist[0]
        vet = sist[1]

        g = numpy.zeros((len(vet),len(vet)))
        g = g.astype(int)

        for i in range(V):
                for j in range(H):
                        if (i==0 and j==0):
                                g = complGrafo(g,quadros[i,j],quadros[i+1,j])#baixo
                                g = complGrafo(g,quadros[i,j],quadros[i,j+1])#direita
                                g = complGrafo(g,quadros[i,j],quadros[i+1,j+1])#baixo-direita
                        elif (i==0 and j>0):
                                g = complGrafo(g,quadros[i,j],quadros[i+1,j])#baixo
                                g = complGrafo(g,quadros[i,j],quadros[i+1,j-1])#baixo-esquerda
                                g = complGrafo(g,quadros[i,j],quadros[i,j-1])#esquerda
                                if (j<H-1):
                                        g = complGrafo(g,quadros[i,j],quadros[i,j+1])#direita
                                        g = complGrafo(g,quadros[i,j],quadros[i+1,j+1])#baixo-direita
                        elif (i>0 and j==0):
                                g = complGrafo(g,quadros[i,j],quadros[i-1,j])#cima
                                g = complGrafo(g,quadros[i,j],quadros[i-1,j+1])#cima-direita
                                g = complGrafo(g,quadros[i,j],quadros[i,j+1])#direita
                                if (i<V-1):
                                        g = complGrafo(g,quadros[i,j],quadros[i+1,j+1])#baixo-direita
```

```
                                        g = complGrafo(g,quadros[i,j],quadros[i+1,j])#baixo
                        #(i>0 and j>0)
                        else:
                                g = complGrafo(g,quadros[i,j],quadros[i,j-1])#esquerda
                                g = complGrafo(g,quadros[i,j],quadros[i-1,j-1])#cima-esquerda
                                g = complGrafo(g,quadros[i,j],quadros[i-1,j])#cima
                                if (j<H-1):
                                        g = complGrafo(g,quadros[i,j],quadros[i,j+1])#direita
                                        g = complGrafo(g,quadros[i,j],quadros[i-1,j+1])#cima-direita
                                if (i<V-1):
                                        g = complGrafo(g,quadros[i,j],quadros[i+1,j])#baixo
                                        g = complGrafo(g,quadros[i,j],quadros[i+1,j-1])#baixo-esquerda
                                if ((j<H-1) and (i<V-1)):
                                        g = complGrafo(g,quadros[i,j],quadros[i+1,j+1])#baixo-direita

        for i in range(len(vet)):
                g[i,i] = 0

        return g

def  geraListaComportas(g):

        for i in range(len(g)):
                lista = [i]
                for j in range(len(g)):
                        if g[i,j] != 0:
                                lista = lista + [j]
                if i == 0:
                        vet = [lista]
                else:
                        vet = vet + [lista]

        return vet

def geraComportas(grafAdjac,infoSist):

        nPts = len(grafAdjac)
        g = numpy.zeros((nPts,nPts))
        g = g.astype(int)

        #gera as comportas entre quadros considerando a altitude (matriz direcionada)
        for i in range(nPts):
                for j in range(nPts):
                        if (grafAdjac[i,j] == 1) and (infoSist[i][0] > infoSist[j][0]):
                                g[i,j] = 1

        diretos = geraListaComportas(g)
        indiretos = [diretos[:]]

        #atribui os adjacentes indiretos ao quadros
        for i in range(G_MAX):                                                  #para cada
grau de adjacencia
                for j in range(nPts):                                           #para cada
quadro
                        ultimo = indiretos[i][:]
                        if i == 0:
                                x=1
                        else:
                                x=0
                        if ((i == 0) and (len(ultimo[j]) == 1)) or ((i > 0) and (len(ultimo[j]) == 0)):
                                        adj = []
                        else:
                                #if len(ultimo[j]) == 0:
                                #       adj = []
                                for k in range(len(ultimo[j])-x):                       #para cada adjacente
fonte[grau]
                                        posk = ultimo[j][k+x]
```

```
                                        if len(diretos[posk]) == 1:
                                                adj = []
                                        for l in range(len(diretos[posk])-1):           #para cada adjacente
destino
                                                if (l == 0) and (k == 0):
                                                        adj = [diretos[posk][l+1]]
                                                else:
                                                        adj = adj + [diretos[posk][l+1]]
                        if j == 0:
                                proximo = [adj]
                        else:
                                proximo = proximo + [adj]

                indiretos = indiretos + [proximo]

        return indiretos

#retorna o conjunto de pontos pelos quais o canal de irrg pode passar, assim como o pto de entrada dgua do
reservatorio na lavoura
def criaPontos(mapaSist):

        n = (VERT/4) + 1
        tot = n*n
        pts = numpy.zeros((tot,3))                              #'i: ponto'[linha, coluna, quadro]
        pts = pts.astype(int)
        pasCol = float(H-1)/float(n-1)
        pasLin = float(V-1)/float(n-1)

        entra=0
        for i in range(n):
                for j in range(n):
                        pts[(i*n)+j,0] = int(i * pasLin)
                        pts[(i*n)+j,1] = int(j * pasCol)
                        pts[(i*n)+j,2] = mapaSist[pts[(i*n)+j,0],pts[(i*n)+j,1]]
                        if (pts[(i*n)+j,0] <= ENTRA_L) and (pts[(i*n)+j,1] <= ENTRA_C):
                                entra = (i*n)+j

        conj = [pts,entra]
        return conj

def buscaAdjacPts(pts):
        n = (VERT/4) + 1
        tot = n*n

        g = numpy.zeros((len(pts),len(pts)))
        g = g.astype(int)

        for i in range(len(pts)):
                #COLUNA ESQUERDA
                if i%n == 0:
                        #superior
                        if i == 0:
                                g[i,i+1] = 1                    #leste
                                #g[i,i+n+1] = 1                 #sudeste
                                g[i,i+n] = 1                    #sul

                        #inferior
                        elif i == (tot-n):
                                g[i,i-n] = 1                    #norte
                                #g[i,i-n+1] = 1                 #nordeste
                                g[i,i+1] = 1                    #leste

                        #meio
                        else:
                                g[i,i-n] = 1                    #norte
                                #g[i,i-n+1] = 1                 #nordeste
                                g[i,i+1] = 1                    #leste
```

```
                        #g[i,i+n+1] = 1                 #sudeste
                        g[i,i+n] = 1                    #sul

                #COLUNA DIREITA
                elif (i+1)%n == 0:
                        #superior
                        if i == (n-1):
                                g[i,i+n] = 1            #sul
                                #g[i,i+n-1] = 1         #sudoeste
                                g[i,i-1] = 1            #oeste

                        #inferior
                        elif i == (tot-1):
                                g[i,i-1] = 1            #oeste
                                #g[i,i-n-1] = 1         #noroeste
                                g[i,i-n] = 1            #norte

                        #meio
                        else:
                                g[i,i+n] = 1            #sul
                                #g[i,i+n-1] = 1         #sudoeste
                                g[i,i-1] = 1            #oeste
                                #g[i,i-n-1] = 1         #noroeste
                                g[i,i-n] = 1            #norte

                #LINHA SUPERIOR
                elif i < n:
                        g[i,i+1] = 1                    #leste
                        #g[i,i+n+1] = 1                 #sudeste
                        g[i,i+n] = 1                    #sul
                        #g[i,i+n-1] = 1                 #sudoeste
                        g[i,i-1] = 1                    #oeste

                #LINHA INFERIOR
                elif i > (tot-n):
                        g[i,i-1] = 1                    #oeste
                        #g[i,i-n-1] = 1                 #noroeste
                        g[i,i-n] = 1                    #norte
                        #g[i,i-n+1] = 1                 #nordeste
                        g[i,i+1] = 1                    #leste

                #MEIO
                else:
                        g[i,i-n] = 1                    #norte
                        #g[i,i-n+1] = 1                 #nordeste
                        g[i,i+1] = 1                    #leste
                        #g[i,i+n+1] = 1                 #sudeste
                        g[i,i+n] = 1                    #sul
                        #g[i,i+n-1] = 1                 #sudoeste
                        g[i,i-1] = 1                    #oeste
                        #g[i,i-n-1] = 1                 #noroeste

        return g

#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ AG
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

def geraLista(g):

        for i in range(len(g)):
                lista = [i]
                for j in range(len(g)):
                        if g[i,j] != 0:
                                lista = lista + [j]
                lista.remove(i)
                if i == 0:
                        vet = [lista]
```

```
                else:
                        vet = vet + [lista]

        return vet

def limpaAdj(l,x):

        for i in range(len(l)):
                aux = l[i]
                j=0
                while j < len(aux):
                        if aux[j] == x:
                                l[i].remove(x)
                        j = j+1
        return l

#gera canal a apartir do ponto de entrada
'''def  geraCanais(g,l,mont):

        l = limpaAdj(l,mont)

        #se ainda tem adjacentes
        if len(l[mont]) > 0:

                #entre 0 e o numero de adjacentes deste ponto
                tamy = random.choice(range(len(l[mont])))
                i=0
                while (i < tamy) and (len(l[mont]) != 0):

                        #seleciona jusante
                        jus = random.choice(l[mont])
                        g[mont,jus] = 1
                        geraCanais(g,l,jus)
                        i = i+1'''

#gera canal totalmente aleatorio
def  geraCanais(canal,lAdj):
        l = range(len(canal))

        tamV = random.choice(l)
        tamH = random.choice(l)

        for i in range(tamV):
                for j in range(tamH):
                        mon = random.choice(l)
                        jus = random.choice(lAdj[mon])
                        canal[mon,jus] = 1

def criaPopulacao(grafAdjac,infoSist,entra):

        for i in range(POP):
                cromossomo = numpy.zeros((len(grafAdjac),len(grafAdjac)))
                cromossomo = cromossomo.astype(int)

                #geraCanais(cromossomo,geraLista(grafAdjac),entra)
                geraCanais(cromossomo,geraLista(grafAdjac))

                if i == 0:
                        populacao = [cromossomo]
                else:
                        populacao = populacao + [cromossomo]

        return populacao

def calcComprAresta(pts,p1,p2):
        dV = V/(VERT/4)
        dH = H/(VERT/4)
```

```
		compr=-1

	#se vertical
		if pts[p1,0] != pts[p2,0]:
			d = dV
	#se horizontal
		else:
			d = dH

		return d

def copiaComportas(comp):
		for i in range(len(comp)):
			for j in range(len(comp[i])):
				if j == 0:
					conjs = [comp[i][j][:]]
				else:
					conjs = conjs + [comp[i][j][:]]
			if i == 0:
				linhas = [conjs]
			else:
				linhas = linhas + [conjs]
		return linhas

def calcDemandaQ(infoSist,fase):

		if fase == 1:
			for i in range(len(infoSist)):
				s = infoSist[i,1]
				v1 = s*HL				#vol armazenado sobre a superficie (m
cubicos)
				v2 = s*ET*T1				#vol evapotransp durante o periodo de
enchimento (m cubicos)
				v3 = s*Vi*T1				#vol infiltrado durante o periodo de
enchimento (m cubicos)
				vt = v1+v2+v3				#vol tot consumido durante o periodo de
enchimento (m cubicos)
				ql = vt/T1				#vazao diaria durante o periodo de
enchimento (m cubicos/dia)
				fator = 24/FUNC
				q = ql*fator				#vazao corrigida para o periodo de func
diario(m cubicos/dia)
				q = q/84600				#vazao corrigida para (m cubicos)/(seg de
funcionamento)
				#print ''
				#print 'q[', i,'] - ', q
				if i == 0:
					demandaQ = [q]
				else:
					demandaQ = demandaQ + [q]
		else:
			for i in range(len(infoSist)):
				s = infoSist[i,1]
				ql = (s)*(ET+Vi)			#vazao diaria durante o periodo de manut da
lamina (m3/dia)
				fator = 24/FUNC
				q = ql*fator				#vazao corrigida para o periodo de func
diario(m cubicos/dia)
				q = q/84600				#vazao corrigida para (m cubicos)/(seg de
funcionamento)
				if i == 0:
					demandaQ = [q]
				else:
					demandaQ = demandaQ + [q]

		return demandaQ
```

```
def limpaAdj2(l,pts,conj,q):

	for i in range(len(l)):
		for j in range(len(conj)):
			aux = len(l[i][pts[conj[j],2]])
			k=0
			while k < aux:
				if l[i][pts[conj[j],2]][k] == q:
					l[i][pts[conj[j],2]].remove(q)
					aux = aux-1
				k = k+1
	return l

def criaConjCanal(infoSist,pts,cromo,p1,e,conj,mont):
	#print 'p1', p1
	#time.sleep(0.5)

	mont[p1] = 0
	if p1 != e:
		conj = conj + [p1]

	for p2 in range(len(cromo)):
		if (p1 != p2) and (cromo[p1,p2] != 0) and (mont[p2] != 0):
			cromo[p1,p2] = 2                                        #marca aresta como
pertencente ao canal
			conj = criaConjCanal(infoSist,pts,cromo,p2,e,conj,mont)
	return conj

def corrigeVazoes(cromo,vazoes,p1):
	#print 'p1', p1
	#time.sleep(0.5)

	for p2 in range(len(cromo)):
		if (p1 != p2) and (cromo[p1,p2] == 2):
			vazoes[p1] = vazoes[p1] + corrigeVazoes(cromo,vazoes,p2)
	return vazoes[p1]

def calcRaioPot(infoSist,pts,vaz,p1,p2):

	p1i = pts[p1,0]
	p1j = pts[p1,1]
	p2i = pts[p2,0]
	p2j = pts[p2,1]

	a = (p1i-p2i)**2
	b = (p1j-p2j)**2
	d = math.sqrt((a+b))
	dh = infoSist[pts[p1,2]][0] - infoSist[pts[p2,2]][0]

   #se desce (bomba false / canal aberto)
	if (dh > 0.) and (vaz > 0.):
		rh = 1.0
		area = (math.pi*((rh/4.0)**2)) / 8                      #seccao semi-circular
		q = (area * math.pow(rh,(2.0/3.0)) * math.sqrt(dh/d)) / M       #chezy-maning

		i=0
		f=2
		ant = 0
	   #enquanto (i<10) e (vaz <= q < (vaz+10%))
		while ((q < vaz) or (q > (vaz*1.1))):

			if q > vaz:
				rh = rh - (rh*(1.0/f))
				direcao = 0
			else:
				rh = rh + (rh*(1.0/f))
```

```
                                direcao = 1
                        if ant != direcao:
                                f = f+1
                        ant = direcao

                        area = (math.pi*((rh/4.0)**2)) / 8
                        q = (area * math.pow(rh,(2.0/3.0)) * math.sqrt(dh/d)) / M
                        i = i+1
                print ''
                print 'vaz', vaz
                print 'area', area
                print 'vel', vaz/area
                #print 'iter', i

                d = calcComprAresta(pts,p1,p2)
                dimensao = area * d * math.sqrt(A_CELULA)

    #se sobe (bomba true)
        elif (dh < 0.) and (vaz > 0.):
                dh = dh * (-1)
                potencia = (Y*dh*vaz)/N                                         #vazao volumetrica

                dimensao = potencia
        else:
                dimensao = 0.

        return dimensao

def calcDimensoes(infoSist,cromo,pts,vazoes,p1,dimensoes):

        for p2 in range(len(cromo)):
                if (p1 != p2) and (cromo[p1,p2] == 2):
                        dimensoes[p1,p2] = calcRaioPot(infoSist,pts,vazoes[p2],p1,p2)
                        dimensoes = calcDimensoes(infoSist,cromo,pts,vazoes,p2,dimensoes)

        return dimensoes

def atribuiQuadros(infoSist,pts,adjQ,cromo,entra,demanda1,demanda2):

        alimentacao = [-1]
        for i in range(len(infoSist)-1):
                alimentacao = alimentacao + [-1]
        vazoes1 = numpy.zeros((len(pts)))
        vazoes2 = numpy.zeros((len(pts)))
        conj = [entra]
        mont = numpy.ones((len(pts)))
        conj = criaConjCanal(infoSist,pts,cromo,entra,entra,conj,mont)

    #para o proprio quadro
        i=0
        while i < len(conj):
                q = pts[conj[i],2]
                if len(adjQ[0][q]) > 0:
                        if adjQ[0][q][0] == q:
                                alimentacao[q] = conj[i]
                                limpaAdj2(adjQ,pts,conj[:],q)
                                vazoes1[conj[i]] = demanda1[q]
                                vazoes2[conj[i]] = demanda2[q]
                                i = i+1
                        else:
                                conj.remove(conj[i])
                else:
                        conj.remove(conj[i])
    #para cada geracao
        x=0
        #enquanto ainda existem adjacentes a serem inseridos
        while (len(conj) > 0) and (x<len(adjQ)):
```

```
                conj2 = conj[:]
                vet = range(len(conj))

            #para cada vertice da geracao
                for i in vet:

                        #seleciona o vertice a recerber adjacentes
                        v = random.choice(conj2)
                        conj2.remove(v)

                        quadro = pts[v,2]
                        #se nao possui mais adjacentes elimina o vertice
                        if len(adjQ[x][quadro]) == 0:
                                conj.remove(v)
                        #caso contrario atribui os adjacentes da geracao a v
                        while len(adjQ[x][quadro]) > 0:
                                q = adjQ[x][quadro][0]
                                alimentacao[q] = v
                                limpaAdj2(adjQ,pts,conj[:],q)
                                vazoes1[v] = vazoes1[v] + demanda1[q]
                                vazoes2[v] = vazoes2[v] + demanda2[q]
                x = x+1

        corrigeVazoes(cromo,vazoes1,entra)
        corrigeVazoes(cromo,vazoes2,entra)
        dimensoes1 = numpy.zeros((len(pts),len(pts)))
        dimensoes1 = calcDimensoes(infoSist,cromo,pts,vazoes1,entra,dimensoes1)
        dimensoes2 = numpy.zeros((len(pts),len(pts)))
        dimensoes2 = calcDimensoes(infoSist,cromo,pts,vazoes2,entra,dimensoes2)

        lista = [conj,alimentacao,vazoes1,vazoes2,dimensoes1,dimensoes2]
        return lista

def
calcAptidao(populacao,pts,infoSist,comportas,ite,resultado,entra,cotaSist,listaAdjcP,demanda1,demanda2,grafico
s):

        Area = (V*H*A_CELULA)
        v1 = Area*HL
        v2 = v1*ET*T1
        v3 = v1*Vi*T1
        vt = v1+v2+v3                                   #vol total consumido durante o periodo de
enchimento
        vs = (V*H*A_CELULA)*T2*(ET+Vi)                  #vol total consumido durante o periodo de manut
da lamina
        Vl = vt + vs                                    #vol TOTAL consumido pela LAVOURA (m
cubicos)

        valores = numpy.zeros((len(populacao),4))
        nPts = len(pts)

        style = xlwt.easyxf('align: horizontal center;')
        style2 = xlwt.easyxf('align: horizontal center;' 'font: bold true;')
        wb = xlrd.open_workbook('saidas/resultado.xls')
        '''s = wb.sheet_by_index(1)
        n1 = s.nrows
        s = wb.sheet_by_index(3)
        n3 = s.nrows
        s = wb.sheet_by_index(4)
        n4 = s.nrows'''
        s = wb.sheet_by_index(1)
        n3 = s.nrows
        s = wb.sheet_by_index(2)
        n4 = s.nrows

        for i in range(len(populacao)):
                print ''
```

```
                print 'Iteracao', ite
                print 'Cromossomo', i

                adjQ = copiaComportas(comportas)
                lista = atribuiQuadros(infoSist,pts,adjQ,populacao[i],entra,demanda1,demanda2)
                conjVert = lista[0]                                     #lista de vertices do canal
                alimentacao = lista[1]                                  #lista de quadros
contendo os vertices q os alim
                vazoes1 = lista[2]                                      #lista contendo as vazoes de
cada vertice no T1
                vazoes1 = lista[3]                                      #lista contendo as vazoes de
cada vertice no T1
                dimensoes1 = lista[4]                                   #arrayList contendo as
dimens das arestas no T1
                dimensoes2 = lista[5]                                   #arrayList contendo as
dimens das arestas no T2

                area = 0
                qIrrig = 0
                for j in range(len(alimentacao)):
                        if alimentacao[j] != -1:
                                area = area + infoSist[j,1]
                                qIrrig = qIrrig + 1
                A = ((area*1.0)/(V*H*A_CELULA))*10

                E=0                                     #E == somatorio das potencias
                L=0                                     #L == numero total de levantes canal
                C=0                                     #C == somatorio dos volumes dos segm canal
IRRIGADOS
                E2=0
                for j in range(nPts):
                        for k in range(nPts):
                                if populacao[i][j,k] == 2:
                                        delta = infoSist[pts[j,2]][0] - infoSist[pts[k,2]][0]
                                   #se sobe E pot>0 (tem bomba)
                                        if (delta < 0.):
                                                E = E + dimensoes1[j,k]
                                                L = L + 1

                                                E2 = E2 + dimensoes2[j,k]
                                   #se desce (tem canal)
                                        else:
                                                C = C + dimensoes1[j,k]
                if i == 0:
                        aMax = A
                        aMin = A
                        pMax = E
                        pMin = E
                        nLevMax = L
                        nLevMin = L
                        cMax = C
                        cMin = C
                else:
                        if A > aMax:
                                aMax = A
                        elif A < aMin:
                                aMin = A
                        if E > pMax:
                                pMax = E
                        elif E < pMin:
                                pMin = E
                        if L > nLevMax:
                                nLevMax = L
                        elif L < nLevMin:
                                nLevMin = L
                        if C > cMax:
                                cMax = C
```

```
                elif C < cMin:
                        cMin = C

        valores[i,0] = A
        valores[i,1] = E
        valores[i,2] = L
        valores[i,3] = C

        #-----

        '''#plan3 -> Canal
        for j in range(nPts):
                resultado[4].write(n3,0,ite,style)
                resultado[4].write(n3,1,i,style)
                resultado[4].write(n3,2,j,style2)#vertice
                resultado[4].write(n3,3,vazoes1[j],style)#vazao
                if j == entra:
                        resultado[4].write(n3,4,u'-1',style)#montante
                cont=0
                for k in range(nPts):
                        if populacao[i][j,k] == 2:
                                resultado[4].write(n3,5+cont,k,style)#jusantes
                                delta = infoSist[pts[j,2]][0] - infoSist[pts[k,2]][0]
                                if delta < 0:

    resultado[4].write(n3,5+cont+4,dimensoes1[j,k],style2)#potencia bomba
                                else:
                                        resultado[4].write(n3,5+cont+4,dimensoes1[j,k],style)#area
sec canal
                                cont = cont+1
                aux=0
                for k in range(len(infoSist)):
                        if alimentacao[k] == j:
                                resultado[4].write(n3,13+aux,k,style)#quadros irrigados
                                aux=aux+1
                n3=n3+1

        #plan4 -> Qualidade Funcionamento
        consE = E * FUNC * T1
        consM = E2 * FUNC * T2
        resultado[5].write(n4,0,ite,style)
        resultado[5].write(n4,1,i,style)
        resultado[5].write(n4,2,(consE+consM)/1000.0,style)
        resultado[5].write(n4,3,(float(area)/float(Area))*100,style)
        resultado[5].write(n4,4,(float(qIrrig)/float(len(alimentacao)))*100,style)
        resultado[5].write(n4,5,area,style)
        resultado[5].write(n4,6,qIrrig,style)
        n4=n4+1'''

        #-----

    notas = valores.copy()

    for i in range(len(populacao)):
        #A == aMin          -> nota 0
        #A == aMax          -> nota 10
        if ((aMax-aMin) == 0.):
                notas[i,0] = aMax
        else:
                notas[i,0] = (((aMax-aMin)-(aMax-notas[i,0]))*10)/(aMax-aMin)

        #E >= pMax -> nota 0
        #E == pMin -> nota 10
        if (notas[i,1] == 0) or ((pMax-pMin) == 0.):
                notas[i,1] = 10.
        else:
                notas[i,1] = (((pMax-pMin)-(notas[i,1]-pMin))*10)/(pMax-pMin)
```

```
                #L >= nLevMax -> nota 0
                #L == nLevMin -> nota 10
                if (notas[i,2] == 0) or ((nLevMax-nLevMin) == 0.):
                        notas[i,2] = 10.
                else:
                        notas[i,2] = (((nLevMax-nLevMin)-(notas[i,2]-nLevMin))*10)/(nLevMax-nLevMin)

                #C >= cMax -> nota 0
                #C == cMin -> nota 10
                if (notas[i,3] == 0) or ((cMax-cMin) == 0.):
                        notas[i,3] = 10.
                else:
                        notas[i,3] = (((cMax-cMin)-(notas[i,3]-cMin))*10)/(cMax-cMin)


                soma = (notas[i,0]*PesoA) + (notas[i,1]*PesoE) + (notas[i,2]*PesoL) + (notas[i,3]*PesoC)

                if i == 0:
                        aptidao = [soma]
                else:
                        aptidao = aptidao + [soma]

                #-----

                '''#plan1 -> Nota
                resultado[2].write(n1,0,ite,style)
                resultado[2].write(n1,1,i,style)
                resultado[2].write(n1,3,aptidao[i],style2)
                resultado[2].write(n1,4,notas[i,0],style)
                resultado[2].write(n1,5,notas[i,1],style)
                resultado[2].write(n1,6,notas[i,2],style)
                resultado[2].write(n1,7,notas[i,3],style)
                resultado[2].write(n1,8,notas[i,4],style)'''
                if i==0:
                        melhor = [i,aptidao[i]]
                elif aptidao[i] > melhor[1]:
                        melhor = [i,aptidao[i]]
#               if i == (len(populacao)-1):
#                       resultado[2].write(n1-len(populacao)+1+melhor[0],2,u'S',style)
#                       plotaCanalMapa(cotaSist,infoSist,pts,populacao[melhor[0]][:],ite,melhor[0],entra)
#               n1=n1+1

                '''#plan2 -> Evolucao-1
                resultado[3].write(i+12,ite+3,aptidao[i],style)
                if i == (len(populacao)-1):
                        resultado[3].write(i+13,ite+3,melhor[0],style2)

        #plan2 -> Evolucao-2
        resultado[3].write(2,ite+3,aMax,style)
        resultado[3].write(3,ite+3,aMin,style)
        resultado[3].write(4,ite+3,pMax,style)
        resultado[3].write(5,ite+3,pMin,style)
        resultado[3].write(6,ite+3,nLevMax,style)
        resultado[3].write(7,ite+3,nLevMin,style)
        resultado[3].write(8,ite+3,cMax,style)
        resultado[3].write(9,ite+3,cMin,style)'''

        #plan3 -> Canal
        i = melhor[0]
        for j in range(nPts):
                resultado[2].write(n3,0,ite,style)
                resultado[2].write(n3,1,i,style)
                resultado[2].write(n3,2,j,style2)#vertice
                resultado[2].write(n3,3,vazoes1[j],style)#vazao
                if j == entra:
                        resultado[2].write(n3,4,u'-1',style)#montante
```

```
                cont=0
                for k in range(nPts):
                        if populacao[i][j,k] == 2:
                                resultado[2].write(n3,5+cont,k,style)#jusantes
                                delta = infoSist[pts[j,2]][0] - infoSist[pts[k,2]][0]
                                if delta < 0:
                                        resultado[2].write(n3,5+cont+4,dimensoes1[j,k],style2)#potencia
bomba
                                else:
                                        resultado[2].write(n3,5+cont+4,dimensoes1[j,k],style)#area sec canal
                                cont = cont+1
                aux=0
                for k in range(len(infoSist)):
                        if alimentacao[k] == j:
                                resultado[2].write(n3,13+aux,k,style)#quadros irrigados
                                aux=aux+1
                n3=n3+1

        #plan4 -> Qualidade Funcionamento
        consDiaE = (E * FUNC)/1000.0            #kw.h
        consDiaM = (E2 * FUNC)/1000.0           #kw.h
        consE = E * FUNC * T1
        consM = E2 * FUNC * T2
        consumo = (consE+consM)/1000.0          #kw.h
        resultado[3].write(n4,0,ite,style)
        resultado[3].write(n4,1,i,style)
        resultado[3].write(n4,2,consDiaE,style)
        resultado[3].write(n4,3,consDiaM,style)
        resultado[3].write(n4,4,consumo,style)
        resultado[3].write(n4,5,(float(area)/float(Area))*100,style)
        resultado[3].write(n4,6,(float(qIrrig)/float(len(alimentacao)))*100,style)
        resultado[3].write(n4,7,area,style)
        resultado[3].write(n4,8,qIrrig,style)
        n4=n4+1

        #plan10 -> Melhores
        resultado[4].write(ite+2,1,melhor[0],style)
        resultado[4].write(ite+2,2,aptidao[melhor[0]],style2)
        resultado[4].write(ite+2,3,aMax,style)
        resultado[4].write(ite+2,4,aMin,style)
        resultado[4].write(ite+2,5,valores[melhor[0],0],style2)
        resultado[4].write(ite+2,6,pMax,style)
        resultado[4].write(ite+2,7,pMin,style)
        resultado[4].write(ite+2,8,valores[melhor[0],1],style2)
        resultado[4].write(ite+2,9,nLevMax,style)
        resultado[4].write(ite+2,10,nLevMin,style)
        resultado[4].write(ite+2,11,valores[melhor[0],2],style2)
        resultado[4].write(ite+2,12,cMax,style)
        resultado[4].write(ite+2,13,cMin,style)
        resultado[4].write(ite+2,14,valores[melhor[0],3],style2)

        graficos[0,ite+1] = ite
        graficos[1,ite+1] = valores[melhor[0],0]        #A
        graficos[2,ite+1] = valores[melhor[0],1]        #E
        graficos[3,ite+1] = valores[melhor[0],2]        #L
        graficos[4,ite+1] = valores[melhor[0],3]        #C
        graficos[5,ite+1] = consDiaE
        graficos[6,ite+1] = consDiaM
        graficos[7,ite+1] = consumo

        #-----

        resultado[0].save('saidas/resultado.xls')

        return aptidao

def selecao(aptidao):
```

```
	popInterm = numpy.zeros((len(aptidao)/2))
	popInterm = popInterm.astype(int)

	for i in range(len(aptidao)/2):
		maior = 0
		for j in range(3):
			sorteio = random.choice(range(len(aptidao)))
			if aptidao[sorteio] >= maior:
				maior = aptidao[sorteio]
				pos = sorteio
		popInterm[i] = pos

	return popInterm

def cruza(cromo1,cromo2):
	tam = len(cromo1[0])
	corte = random.choice(range(tam))

	canal1 = cromo1.copy()
	canal2 = cromo2.copy()

	for i in range(tam-corte):
		for j in range(tam-corte):
			canal1[i+corte,j+corte] = canal2[i+corte,j+corte]
		for j in range(corte):
			canal1[i+corte,j] = canal2[i+corte,j]
	for i in range(corte):
		for j in range(tam-corte):
			canal1[i,j+corte] = canal2[i,j+corte]

	return canal1


def crossover(populacao,popInterm):
	tam = int(len(populacao)*(float(TAXA_CROS)/float(100)))

	for i in range(tam):
		cromo1 = random.choice(popInterm)
		cromo2 = random.choice(popInterm)
		if i == 0:
			novos = [cruza(populacao[cromo1],populacao[cromo2])]
		else:
			novos = novos + [cruza(populacao[cromo1],populacao[cromo2])]
	return novos

def trocaBin(x):
	if x == 0:
		x = 1
	else:
		x = 0
	return x

def jusantes(cromo,b):
	for i in range(len(cromo)):
		if cromo[i,b] == 1:
			return 1  #ja e jusante e nao pode troca
	return 0

def excluiArestas(canal,b):
	for i in range(len(canal)):
		if canal[b,i] == 1:
			canal[b,i] = 0
			canal = excluiArestas(canal,i)
	return canal

'''def mutacao(populacao,popInterm,AdjcP):
	tam = int(math.ceil(len(populacao)*(float(TAXA_MUT)/float(100))))
```

```
	i=0
	j=0
	while i < tam:
		aux=i

	   #seleciona o individuo q sofrera mutacao
		pos = random.choice(popInterm)
		canal = populacao[pos].copy()

		conjVertices = [entra]
		conjVertices = criaConjCanal(infoSist,pts,canal,entra,entra,conjVertices)

		a = random.choice(conjVertices)
		b = random.choice(AdjcP[a])

	   #se inclui aresta E b nao e jusante ainda
		if (canal[a,b] == 0) and (jusantes(canal,b) == 0):

			#inclui aresta com adjacente
			canal[a,b] = 1
			i = i+1

	   #se exclui aresta
		elif canal[a,b] == 1:

			#exclui aresta e os segmentos de canal subsequentes
			canal[a,b] = 0
			canal = excluiArestas(canal,b)
			i = i+1

		if (j == 0) and (i != aux):
			novos = [canal]
			j=1
		elif (j == 1) and (i != aux):
			novos = novos + [canal]
	return novos"'

def mutacao(populacao,popInterm,AdjcP):
	tam = int(math.ceil(len(populacao)*(float(TAXA_MUT)/float(100))))

	pts = range(len(populacao[0]))

	for i in range(tam):

	   #seleciona o individuo q sofrera mutacao
		pos = random.choice(popInterm)
		canal = populacao[pos].copy()

		for j in range(N_GENES):

		   #seleciona a aresta q sofrera alteracao
			a = random.choice(pts)
			b = random.choice(AdjcP[a])

			if (canal[a,b] == 0):
				canal[a,b] = 1
			else:
				canal[a,b] = 0

		if (i == 0):
			novos = [canal]
		else:
			novos = novos + [canal]

	return novos
```

```
def substituiPop(populacao,aptidao,popCros,popMut,ite,resultado):
	style = xlwt.easyxf('align: horizontal center;')
	vet = range(len(populacao))

	for i in range(len(popCros)):
		menor = 11.
		for j in range(3):
			sorteio = random.choice(vet)
			if aptidao[sorteio] <= menor:
				menor = aptidao[sorteio]
				pos = sorteio
		populacao[pos] = popCros[i]
		vet.remove(pos)
#		resultado[3].write(POP+16+pos,ite+3,u'crossover',style)

	for i in range(len(popMut)):
		menor = 11.
		for j in range(3):
			sorteio = random.choice(vet)
			if aptidao[sorteio] <= menor:
				menor = aptidao[sorteio]
				pos = sorteio
		populacao[pos] = popMut[i]
		vet.remove(pos)
#		resultado[3].write(POP+16+pos,ite+3,u'mutacao',style)

#	resultado[0].save('saidas/resultado.xls')
	return populacao

def limpaCromossomos(populacao):
	for i in range(len(populacao)):
		cromo = populacao[i]
		for j in range(len(cromo)):
			for k in range(len(cromo)):
				if cromo[j,k] == 2:
					cromo[j,k] = 1

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ SAIDAS
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

def delta(y1,y2):
	if y2>y1:
		return y2-y1
	else:
		return y1-y2

#plota o canal no formato de mapa
def plotaCanalMapa(mapaSist,infoSist,pts,cromo,ite,cro,entra):
	g = mapaSist.copy()

	canal = mapaSist.max() + (DH*5)
	direcao = mapaSist.min() - (DH*5)

	#defini as dimensoes da representacao do ponto de entrada e dos levantes
	med = (V+H)/2
	if med < 150:
		X = 3
	else:
		X = ((med/50)*2)-1

	#marca pontos
	for i in range(len(pts)):
		g[pts[i,0],pts[i,1]] = mapaSist.max() + (DH)

	#marca canal e levantes
	for i in range(len(pts)):
		for j in range(len(pts)):
```

```
if (cromo[i,j] == 2) and (cromo[j,i] != 2):

    #marca canal
        aux=0
        if (j==(i+1)) or (j==(i-1)):     #se horizontal
            if pts[i,0] == 0:
                l = pts[i,0]+1
            elif pts[i,0] == (V-1):
                l = pts[i,0]-1
            else:
                l = pts[i,0]
            if j > i:
                c = pts[i,1] + 1
                while c != pts[j,1]:
                   #marca a direcao
                        if aux < X:
                            g[l,c] = direcao
                            aux = aux+1
                        else:
                            g[l,c] = canal
                        c=c+1
            elif j < i:
                c = pts[i,1] - 1
                while c != pts[j,1]:
                   #marca a direcao
                        if aux < X:
                            g[l,c] = direcao
                            aux = aux+1
                        else:
                            g[l,c] = canal
                        c=c-1

        else:                                #se vertical
            if pts[i,1] == 0:
                c = pts[i,1]+1
            elif pts[i,1] == (H-1):
                c = pts[i,1]-1
            else:
                c = pts[i,1]
            if j > i:
                l = pts[i,0] + 1
                while l != pts[j,0]:
                   #marca a direcao
                        if aux < X:
                            g[l,c] = direcao
                            aux = aux+1
                        else:
                            g[l,c] = canal
                        l=l+1
            if j < i:
                l = pts[i,0] - 1
                while l != pts[j,0]:
                   #marca a direcao
                        if aux < X:
                            g[l,c] = direcao
                            aux = aux+1
                        else:
                            g[l,c] = canal
                        l=l-1

    #marca levantes
        dlt = infoSist[pts[i,2]][0] - infoSist[pts[j,2]][0]
        if (dlt < 0.):
            for a in range(X):
                if (j==(i+1)) or (j==(i-1)):     #se horizontal
                    if j<i:
                        l = pts[i,0]-(X/2)+a
```

```
                                                                        c = pts[i,1]-X
                                                                else:
                                                                        l = pts[i,0]-(X/2)+a
                                                                        c = pts[i,1]+X
                                                        else:                                   #se vertical
                                                                if j<i:
                                                                        l = pts[i,0]-X
                                                                        c = pts[i,1]-(X/2)+a
                                                                else:
                                                                        l = pts[i,0]+X
                                                                        c = pts[i,1]-(X/2)+a
                                                        if ((l >= 0) and (l < V) and (c >= 0) and (c < H)):
                                                                g[l,c] = canal

        #marca ponto de entrada
        for i in range(X):
                for j in range(X):
                        l = pts[entra,0]-(X/2)+i
                        c = pts[entra,1]-(X/2)+j
                        if ((l >= 0) and (l < V) and (c >= 0) and (c < H)):
                                g[l,c] = mapaSist.max() + (DH*5)

        if ite == -1:
                ite = str(ite)
                cro = str(cro)
                name = 'saidas/' + 'cro_' + cro + '.png'                                # cromossomo_iteracao
        else:
                ite = str(ite)
                cro = str(cro)
                name = 'saidas/' + 'ite_' + ite + '_cro_' + cro + '.png'        # cromossomo_iteracao

        matplotlib.pyplot.imshow(g)
        matplotlib.pyplot.colorbar()
        matplotlib.pyplot.savefig(name)
        matplotlib.pyplot.clf()

#cria a planilha e insere informacoes fixas
def criaPlanilha(infoSist,pontos,grafAdjacP,grafAdjacQ,comportas):
        nPts = len(infoSist)

        style = xlwt.easyxf('align: horizontal center;')
        style2 = xlwt.easyxf('font: bold true;')

        arquivo = xlwt.Workbook()

        #Entradas
        Area = V*H*A_CELULA
        v1 = Area*HL
        v2 = Area*ET*T1
        v3 = Area*Vi*T1
        vt = v1+v2+v3                                   #vol total consumido durante o periodo de
enchimento
        vs = Area*T2*(ET+Vi)                    #vol total consumido durante o periodo de manut da lamina
        Vl = vt + vs                                    #vol TOTAL consumido pela LAVOURA (m
cubicos)
        plan0 = arquivo.add_sheet(u'Entradas')
        plan0.col(0).width = 12000
        plan0.col(1).width = 5000
#       plan0.write_merge(0,0,0,0,'MAPA',style)         #(linhaInicial, LinhaFinal5, colunaInicial,
colunaFinal5,'label',style)
        plan0.write(0,0,u'MAPA',style)
        plan0.write(1,0,u'Funcao')              #(linha, coluna, u'conteudo')
        plan0.write(2,0,u'Numero de linhas do MDE')
        plan0.write(3,0,u'Numero de colunas do MDE')
        plan0.write(4,0,u'Variacao de cota maxima por quadro (m)')
        plan0.write(5,0,u'Area total (m2)')
        plan0.write(6,0,u'Estimativa demanda hidrica alagamento (m3)')
```

```
plan0.write(7,0,u'Estimativa demanda hidrica manutencao (m3)')
plan0.write(8,0,u'Estimativa demanda hidrica total (m3)')
plan0.write(10,0,u'PARAMETROS AGRONOMICOS',style)
plan0.write(11,0,u'Coeficiente de chezy-manning')
plan0.write(12,0,u'Altura lamina dagua lavoura (m)')
plan0.write(13,0,u'Periodo de enchimento da lavoura (dias)')
plan0.write(14,0,u'Periodo de manutencao da lamina (dias)')
plan0.write(15,0,u'Periodo diario de funcionamento recalques (hs)')
plan0.write(17,0,u'PARAMETROS GENETICOS',style)
plan0.write(18,0,u'Tamanho da populacao')
plan0.write(19,0,u'Taxa de crossover (%)')
plan0.write(20,0,u'Taxa de mutacao (%)')
plan0.write(21,0,u'Numero de iteracoes')
plan0.write(22,0,u'Peso A')
plan0.write(23,0,u'Peso E')
plan0.write(24,0,u'Peso L')
plan0.write(25,0,u'Peso C')
plan0.write(27,0,u'EXECUCAO',style)
plan0.write(28,0,u'Tempo de sistematizacao (s/m)')
plan0.write(29,0,u'Tempo de criacao da populacao (s/m)')
plan0.write(30,0,u'Tempo de avaliacao da populacao (s/m)')
plan0.write(31,0,u'Tempo de evolucao da populacao (s/m)')
plan0.write(32,0,u'Tempo de execucao total (s/m)')
plan0.write(1,1,MDE,style)
plan0.write(2,1,V,style)
plan0.write(3,1,H,style)
plan0.write(4,1,DH,style)
plan0.write(5,1,Area,style)
plan0.write(6,1,vt,style)
plan0.write(7,1,vs,style)
plan0.write(8,1,Vl,style)
plan0.write(11,1,M,style)
plan0.write(12,1,HL,style)
plan0.write(13,1,T1,style)
plan0.write(14,1,T2,style)
plan0.write(15,1,FUNC,style)
plan0.write(18,1,POP,style)
plan0.write(19,1,TAXA_CROS,style)
plan0.write(20,1,TAXA_MUT,style)
plan0.write(21,1,N_ITER,style)
plan0.write(22,1,PesoA,style)
plan0.write(23,1,PesoE,style)
plan0.write(24,1,PesoL,style)
plan0.write(25,1,PesoC,style)

'''#Nota
plan1 = arquivo.add_sheet(u'Nota')
plan1.col(3).width = 4000
plan1.col(4).width = 4000
plan1.col(5).width = 4000
plan1.col(6).width = 4000
plan1.col(7).width = 4000
plan1.col(8).width = 4000
plan1.write(0,0,u'Iteracao',style)
plan1.write(0,1,u'Cromossomo',style)
plan1.write(0,2,u'MELHOR',style)
plan1.write(0,3,u'APTIDAO',style)
plan1.write(0,4,u'A',style)
plan1.write(0,5,u'E',style)
plan1.write(0,6,u'L',style)
plan1.write(0,7,u'C',style)

#Evolucao
plan2 = arquivo.add_sheet(u'Evolucao')
plan2.col(0).width = 6500
plan2.write(0,0,u'NOTA',style)
plan2.write(1,0,u'-',style)
```

```
plan2.write(2,0,u'A - max')
plan2.write(3,0,u'A -  min')
plan2.write(4,0,u'E - max')
plan2.write(5,0,u'E -  min')
plan2.write(6,0,u'L - max')
plan2.write(7,0,u'L -  min')
plan2.write(8,0,u'C - max')
plan2.write(9,0,u'C -  min')
plan2.write(10,0,u'-',style)
plan2.write(11,0,u'Cromossomo',style)
plan2.write(0,1,u'ITERACAO',style)
for i in range(13):
        plan2.write(i+1,1,u'-',style)
plan2.write(POP+12,1,u'Melhor',style)
plan2.write(POP+14,0,u'OPERADOR GENETICO',style)
plan2.write(POP+15,0,u'Cromossomo',style)
plan2.write(POP+14,1,u'ITERACAO',style)
plan2.write(POP+15,1,u'-',style)
for i in range(POP):
        plan2.write(i+12,0,i,style)
        plan2.write(i+12,1,u'-',style)
        plan2.write(i+POP+16,0,i,style)
        plan2.write(i+POP+16,1,u'-',style)
for i in range(N_ITER+1):
        plan2.col(i+2).width = 4000
        plan2.write(0,i+2,i-1,style)
        plan2.write(1,i+2,'-',style)
        plan2.write(10,i+2,'-',style)
        plan2.write(11,i+2,'-',style)
        plan2.write(POP+14,i+2,i-1,style)
        plan2.write(POP+15,i+2,'-',style)'"

#Canal
plan3 = arquivo.add_sheet(u'Canal')
plan3.col(1).width = 3500
plan3.col(2).width = 2400
plan3.col(3).width = 3500
plan3.col(4).width = 2700
plan3.write(0,0,u'Iteracao',style)
plan3.write(0,1,u'Cromossomo',style)
plan3.write(0,2,u'Vertice',style)
plan3.write(0,3,u'Vazao (m3/s)',style)
plan3.write(0,4,u'Montante',style)
plan3.write_merge(0,0,5,8,u'Jusates',style)
plan3.write_merge(0,0,9,12,u'Dimensao (m2,w)',style)
plan3.write_merge(0,0,13,12+nPts,u'QUADROS IRRIGADOS',style)
for i in range(4):
        plan3.col(5+i).width = 1400
for i in range(4):
        plan3.col(9+i).width = 1700
for i in range(nPts):
        plan3.col(13+i).width = 900

#Qualidade Funcionamento
plan4= arquivo.add_sheet(u'Qualidade Funcionamento')
plan4.col(2).width = 6500
plan4.col(3).width = 6500
plan4.col(4).width = 6500
plan4.col(5).width = 6500
plan4.col(6).width = 6500
plan4.col(7).width = 6500
plan4.col(8).width = 6500
plan4.write(0,0,u'Iteracao',style)
plan4.write(0,1,u'Cromossomo',style)
plan4.write(0,2,u'Consumo Diario Ench (kw.h)',style)
plan4.write(0,3,u'Consumo Diario Man (kw.h)',style)
plan4.write(0,4,u'Consumo Energetico (kw.h)',style)
```

```
	plan4.write(0,5,u'Area Irrigada (%)',style)
	plan4.write(0,6,u'Quadros Irrigados (%)',style)
	plan4.write(0,7,u'Area Irrigada (m2)',style)
	plan4.write(0,8,u'Quadros Irrigados (unidades)',style)

	#Melhores
	plan10 = arquivo.add_sheet(u'Melhores')
	plan10.col(1).width = 3500
	plan10.write(0,0,u'ITERACAO',style)
	plan10.write(0,1,u'Cromossomo',style)
	plan10.write(0,2,u'Nota',style)
	for i in range(N_ITER+1):
		plan10.write(i+1,0,i-1,style)
	plan10.write(0,3,u'A - max',style)
	plan10.write(0,4,u'A -  min',style)
	plan10.write(0,5,u'A',style2)
	plan10.write(0,6,u'E - max',style)
	plan10.write(0,7,u'E -  min',style)
	plan10.write(0,8,u'E',style2)
	plan10.write(0,9,u'L - max',style)
	plan10.write(0,10,u'L -  min',style)
	plan10.write(0,11,u'L',style2)
	plan10.write(0,12,u'C - max',style)
	plan10.write(0,13,u'C -  min',style)
	plan10.write(0,14,u'C',style2)

	'''#Vertices
	plan5= arquivo.add_sheet(u'Vertices')                    #'i: ponto'[linha, coluna, quadro]
	plan5.write(0,0,u'Vertice',style)
	plan5.write(0,1,u'Linha',style)
	plan5.write(0,2,u'Coluna',style)
	plan5.write(0,3,u'Quadro',style)
	plan5.write(0,4,u'Cota',style)
	for i in range(len(pontos)):
		plan5.write(i+1,0,i,style)
		plan5.write(i+1,1,pontos[i,0],style)
		plan5.write(i+1,2,pontos[i,1],style)
		plan5.write(i+1,3,pontos[i,2],style)
		plan5.write(i+1,4,infoSist[pontos[i,2]][0],style)

	#Vertices Adjacentes
	listaAdjP = geraLista(grafAdjacP)
	plan6= arquivo.add_sheet(u'Vertices Adjacentes')
	plan6.write(0,0,u'Vertice',style)
	for i in range(len(listaAdjP)):
		plan6.write(i+1,0,i,style)
		for j in range(len(listaAdjP[i])):
			plan6.write(i+1,j+1,listaAdjP[i][j],style)
	for i in range(4):
		plan6.col(1+i).width = 1400

	#Quadros
	dem = calcDemandaQ(infoSist,1)
	plan7= arquivo.add_sheet(u'Quadros')                     #'i: quadro'[cota do quadro, area do
quadro]
	plan7.col(3).width = 5800
	plan7.write(0,0,u'Quadro',style)
	plan7.write(0,1,u'Cota',style)
	plan7.write(0,2,u'Area',style)
	plan7.write(0,3,u'Vazao Requerida (m3/s)',style)
	for i in range(nPts):
		plan7.write(i+1,0,i,style)
		plan7.write(i+1,1,infoSist[i][0],style)
		plan7.write(i+1,2,infoSist[i][1],style)
		plan7.write(i+1,3,dem[i],style)

	#Quadros Adjacentes
```

```
	listaAdjQ = geraLista(grafAdjacQ)
	plan8= arquivo.add_sheet(u'Quadros Adjacentes')
	plan8.write(0,0,u'Quadro',style)
	for i in range(len(listaAdjQ)):
		plan8.write(i+1,0,i,style)
		for j in range(len(listaAdjQ[i])):
			plan8.write(i+1,j+1,listaAdjQ[i][j],style)
	for i in range(4):
		plan8.col(1+i).width = 1400

	#Comportas
	plan9= arquivo.add_sheet(u'Comportas')
	plan9.write(0,0,u'Quadro',style)
	for i in range(len(infoSist)):                                          #para cada quadro
		plan9.write(i+1,0,i,style)
		cont=0
		for j in range(G_MAX):                                                  #para cada
grau de adjacencia
			for k in range(len(comportas[j][i])):                           #para cada adjacente
				plan9.write(i+1,cont+1,comportas[j][i][k],style)
	#comportas[grau,quadro,adjacente]
				cont = cont+1
			plan9.write(i+1,cont+1,u'|',style)
			cont = cont+1
	for i in range(len(comportas)*2):
		plan9.col(1+i).width = 1100'''

	arquivo.save('saidas/resultado.xls')

	resultado = [arquivo,plan0,plan3,plan4,plan10]
	return resultado

#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ MAIN
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
style = xlwt.easyxf('align: horizontal center;')
t0 = time.time()

#-----

mde = criaMde()                                                         #MODELO DIGITAL DE ELEVACAO
sist = sistMde(mde)
mapaSist = sist[0]                                                      #MAPA DA AREA SISTEMATIZADA
infoSist = sist[1]                                                  #'i: quadro'[cota do quadro, area do quadro]
cotaSist = sist[2]
grafAdjacQ = buscaAdjacQuadros(sist)                                    #GRAFO DE ADJACENCIAS ENTRE
QUADROS
comportas = geraComportas(grafAdjacQ,infoSist)                          #MATRIZ COM AS COMPORTAS
ENTRE QUADROS: considerando a gravidade
pontos = criaPontos(mapaSist)
pts = pontos[0]
entra = pontos[1]
grafAdjacP = buscaAdjacPts(pts)
listaAdjcP = geraLista(grafAdjacP)
demanda1 = calcDemandaQ(infoSist,1)
demanda2 = calcDemandaQ(infoSist,2)

d1 = 0
d2 = 0
areaL = 0
for i in range(len(demanda1)):
	d1 = d1 + demanda1[i]
	d2 = d2 + demanda2[i]
	areaL = areaL + infoSist[i,1]
areaL = areaL / 10000
d1 = d1*1000
d1 = d1/86400
d1 = d1/areaL
```

```
print 'demanda T1 [l/s/ha]->', d1
d2 = d2*1000
d2 = d2/86400
d2 = d2/areaL
print 'demanda T2 [l/s/ha] ->', d2
print 'demanda Total [l] ->', (T1*d1) + (T2*d2)
print 'area [ha] ->', areaL*HL

resultado = criaPlanilha(infoSist,pts,grafAdjacP,grafAdjacQ,comportas)

#-----

tempo = time.time()-t0
print ' '
print '       SISTEMATIZACAO DA AREA CONCLUIDA ->', tempo, 'segundos'
resultado[1].write(28,1,tempo,style)
resultado[1].write(28,2,tempo/60.0,style)
t1 = time.time()

#-----

populacao = criaPopulacao(grafAdjacP,infoSist,entra)
tempo = time.time()-t1
print ' '
print '       POPULACAO INICIAL CRIADA         ->', tempo, 'segundos'
resultado[1].write(29,1,tempo,style)
resultado[1].write(29,2,tempo/60.0,style)
t2 = time.time()

#-----

graficos = numpy.zeros((8,N_ITER+1))
aptidao = calcAptidao(populacao[:],pts,infoSist,comportas,-
1,resultado,entra,cotaSist,listaAdjcP,demanda1,demanda2,graficos)
tempo = time.time()-t2
print ' '
print '       POPULACAO INICIAL AVALIADA       ->', tempo, 'segundos'
resultado[1].write(30,1,tempo,style)
resultado[1].write(30,2,tempo/60.0,style)
t3 = time.time()
t = t3

#-----

'''for i in range(len(populacao)):
        cromo = populacao[i]
        plotaCanalMapa(cotaSist,infoSist,pts,cromo,-1,i,entra)'''

#-----

for x in range(N_ITER):
        limpaCromossomos(populacao)

        popInterm = selecao(aptidao)                                    #posicao dos
selecionados na populacao

        popCros = crossover(populacao[:],popInterm)                     #popul nova
gerada por cruzamento
        popMut = mutacao(populacao[:],popInterm,listaAdjcP)             #popul nova
gerada por mutacao

        populacao = substituiPop(populacao[:],aptidao,popCros,popMut,x,resultado)

        aptidao =
calcAptidao(populacao[:],pts,infoSist,comportas,x,resultado,entra,cotaSist,listaAdjcP,demanda1,demanda2,grafic
os)
```

```
            '''for i in range(len(populacao)):
                        cromo = populacao[i]
                        plotaCanalMapa(cotaSist,infoSist,pts,cromo,x,i,entra)'''

            print ' '
            print '             Iteracao', x, 'concluida ->', time.time()-t, 'segundos'
            t = time.time()

#-----

tempo = time.time()-t3
resultado[1].write(31,1,tempo,style)
resultado[1].write(31,2,tempo/60.0,style)

matplotlib.pylab.plot(graficos[0],graficos[1])
matplotlib.pylab.savefig('saidas/grafico_A.png')
matplotlib.pyplot.clf()
matplotlib.pylab.plot(graficos[0],graficos[2])
matplotlib.pylab.savefig('saidas/grafico_E.png')
matplotlib.pyplot.clf()
matplotlib.pylab.plot(graficos[0],graficos[3])
matplotlib.pylab.savefig('saidas/grafico_L.png')
matplotlib.pyplot.clf()
matplotlib.pylab.plot(graficos[0],graficos[4])
matplotlib.pylab.savefig('saidas/grafico_C.png')
matplotlib.pyplot.clf()
matplotlib.pylab.plot(graficos[0],graficos[5])
matplotlib.pylab.savefig('saidas/grafico_Cons_Dia_Ench.png')
matplotlib.pyplot.clf()
matplotlib.pylab.plot(graficos[0],graficos[6])
matplotlib.pylab.savefig('saidas/grafico_Cons_Dia_Man.png')
matplotlib.pyplot.clf()
matplotlib.pylab.plot(graficos[0],graficos[7])
matplotlib.pylab.savefig('saidas/grafico_Consumo.png')
matplotlib.pyplot.clf()
tempo = time.time()-t0
print ' '
print '       TEMPO TOTAL DE EXECUCAO ->', tempo, 'segundos'
resultado[1].write(32,1,tempo,style)
resultado[1].write(32,2,tempo/60.0,style)
resultado[0].save('saidas/resultado.xls')
```