Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Ajuste Automático de Parâmetros para Aplicações de Segmentação Nuclear em Imagens Médicas

Luís Felipe Rabello Taveira

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Orientador
Prof. Dr. George Luiz Medeiros Teodoro

Brasília
2017

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Programa de pós-graduação em Informática

Coordenador: Prof. Dr. Bruno Luiggi Macchiavello Espinoza

Banca examinadora composta por:

Prof. Dr. George Luiz Medeiros Teodoro (Orientador) — CIC/UnB
Prof.ª Dr.ª Alba Cristina Magalhães Alves de Melo — CIC/UnB
Prof. Dr. Renato Antônio Celso Ferreira — DCC/UFMG

CIP — Catalogação Internacional na Publicação

Rabello Taveira, Luís Felipe.

Ajuste Automático de Parâmetros para Aplicações de Segmentação Nuclear em Imagens Médicas / Luís Felipe Rabello Taveira. Brasília : UnB, 2017.
98 p. : il. ; 29,5 cm.

Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2017.

1. Bioinformática, 2. Segmentação Nuclear, 3. Ajuste Automático de Parâmetros, 4. Otimização Multiobjetivo, 5. Algoritmos Genéticos, 6. Consultas Espaciais, 7. Region Templates

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Ajuste Automático de Parâmetros para Aplicações de Segmentação Nuclear em Imagens Médicas

Luís Felipe Rabello Taveira

Dissertação apresentada como requisito parcial
para conclusão do Mestrado em Informática

Prof. Dr. George Luiz Medeiros Teodoro (Orientador)
CIC/UnB

Prof.ª Dr.ª Alba Cristina Magalhães Alves de Melo
CIC/UnB

Prof. Dr. Renato Antônio Celso Ferreira
DCC/UFMG

Prof. Dr. Bruno Luiggi Macchiavello Espinoza
Coordenador do Programa de pós-graduação em Informática

Brasília, 22 de Junho de 2017

Agradecimentos

Agradeço a Deus.

Agradeço a minha família, em especial ao meu pai Fernando, minha mãe Eneide, meu irmão Luís Fernando e à Giselle, minha gata.

Agradeço ao professor George, meu orientador e mestre ao longos desses anos pelos ensinamentos, conselhos e por todo tempo e esforço investido.

Agradeço à professora Alba e ao professor Renato por terem aceitado o convite de participar da banca e pela revisão do trabalho.

Agradeço aos meus amigos de laboratório e de sala de aula pelos momentos de alegria e superação ao longo dos problemas e desafios encontrados pelo caminho.

Agradeço aos meus outros professores, tanto os de sala de aula quanto os da vida, que contribuíram com a minha formação até aqui.

Agradeço aos meus avós que tanto lutaram para criar os seus filhos. Especialmente a minha avó Raquel que dedicou a sua vida para cuidar dos seus netos.

Agradeço, em especial, aos meus amigos Jeremias e Alexandre que foram os principais companheiros dessa jornada.

Obrigado!

Resumo

Imagens em alta resolução de microscopia são muito importantes no estudo de doenças em níveis celulares e sub-celulares. Os efeitos causados por muitas doenças, como o câncer por exemplo, geralmente manifestam-se como alterações na morfologia das células em escala microscópica. Investigar essas mudanças e suas correlações com dados moleculares e resultados clínicos podem levar a uma melhor compreensão dos mecanismos da doença, e permitir o desenvolvimento de novas formas de tratamento. Existem aplicações de bioinformática capazes de realizar análises qualitativas em amostras de tecidos humanos por meio do processamento desse tipo de imagem. Essas aplicações são parametrizadas e alterações nos valores de seus parâmetros de configuração podem causar impactos significativos na qualidade do resultado. Além disso, elas são pré-configuradas por um conjunto de parâmetros padrão que não são os ideais para todos os tipos de imagens que poderão ser analisadas. Dependendo do tamanho da imagem a ser processada, cada execução dessas aplicações pode levar horas em uma estação de trabalho comum. Neste trabalho foram utilizadas duas aplicações exemplo que, conforme os valores de parâmetros utilizados, podem ser ajustadas em bilhões de maneiras diferentes. Para encontrar combinações de parâmetros que melhorem a qualidade do resultado e reduzam o tempo de execução destas aplicações de maneira eficiente, foi proposto um sistema de ajuste automático de parâmetros multiobjetivo capaz de melhorar a qualidade da análise dessas aplicações em até 8,35× e de reduzir o tempo de execução em até 16,05×, testando-se apenas 100 pontos do espaço de busca. A fim de avaliar a capacidade de generalização do sistema de otimização em encontrar uma combinação de parâmetros que fosse capaz de otimizar múltiplas imagens ao mesmo tempo, realizou-se experimentos de validação cruzada em que foi possível atingir uma melhoria de até 1,15× na qualidade do resultado e de redução no tempo de execução médio em 10,25×. Para quantificar essas melhorias e as alterações na morfologia das células e tecidos em escala micro-anatômica foram desenvolvidas múltiplas métricas e mecanismos de consultas espaciais a fim de tornar essas análises mais precisas e eficientes.

Palavras-chave: Bioinformática, Segmentação Nuclear, Ajuste Automático de Parâmetros, Otimização Multiobjetivo, Algoritmos Genéticos, Consultas Espaciais, Region Templates

Abstract

High resolution microscopy images may greatly help the study of diseases at cellular and subcellular levels. The effects caused by many diseases, such as cancer, usually manifest themselves as changes in the morphology of cells on a microscopic scale. Investigating these changes and their correlations with molecular data and clinical outcomes may lead to a better understanding of the mechanisms of the disease, and allow the development of new forms of treatment. There are applications of bioinformatics capable of performing qualitative analyzes on human tissue samples through the processing of this type of image. These applications are parameterized and changes in the values of their configuration parameters can cause significant impacts on the quality of the result. In addition, they are preconfigured by a set of default parameters that are not ideal for all types of images that can be processed. Depending on the size of the image being analyzed, each execution of these applications can take hours on a regular workstation. In this work, two example applications were used. Each of them can be adjusted in billions of different ways according to the parameter values used. To find combinations of parameters that improve the quality of the result and reduce the execution time of these applications efficiently, we propose a multi-objective auto tuning system. This system is able to improve the quality of the analysis of these applications by up to 8.35× and also able to reduce the execution time by up to 16.05× by testing only 100 search-space points. In order to evaluate the generalization ability of the optimization framework to find a combination of parameters that is able to optimize multiple images at the same time, cross-validation experiments were performed in which it was possible to achieve an improvement of up to 1.15× on quality and reduction of the average execution time by 10.25×. To quantify these improvements and changes in the morphology of cells and tissues at the micro-anatomical scale, multiple metrics and spatial query mechanisms were developed to make these analyzes more accurate and efficient.

Keywords: Bioinformatics, Cell Nuclei Segmentation, Auto-Tuning, Multi-objective Tuning, Genetic Algorithms, Spatial Queries, Region Templates

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O processamento de imagens em Patologia é uma das áreas da Ciência da Computação que tem proporcionado inúmeros avanços científicos em conjunto com a Medicina. A análise de imagens digitais em alta resolução de slides de tecidos (WSI[1]) permite o estudo de doenças em níveis celulares e sub-celulares [15]. Esses estudos são realizados utilizando-se imagens digitais de microscopia médica em vez de lâminas físicas. Muitas doenças, como o câncer por exemplo, manifestam-se por meio de alterações na morfologia das células em escala micro-anatômica. Investigar essas mudanças, quantificar os seus efeitos e efetuar correlações com dados moleculares e resultados clínicos podem levar a uma melhor compreensão dos mecanismos dessas doenças, e permitir o desenvolvimento de novas formas de tratamento. Um exemplo de alteração causada por tumores cerebrais é a mudança na forma e textura dos núcleos das células comprometidas. Nesse caso, as células comprometidas passam, por exemplo, a se tornar mais alongadas, além de sofrerem alterações em sua textura [15, 67].

Os instrumentos de captura de imagens médicas progrediram significativamente nos últimos anos, de maneira que atualmente eles podem capturar, a partir de uma amostra de tecido, imagens de resolução de até 120K x 120K *pixels* em um *scanner* estado da arte [71]. Esses instrumentos são capazes de realizar a digitalização de amostras de tecidos humanos em poucos minutos [37]. Essas imagens sem compressão podem atingir aproximadamente 50 GB de tamanho.

Além disso, esses dispositivos de digitalização de tecidos estão se tornando cada vez mais populares nas unidades médicas devido à queda nos preços. Em decorrência da popularização dessas tecnologias, houve um crescimento significativo no tamanho das bases de dados de WSIs e, como consequência, existe uma grande demanda por tarefas eficientes relacionadas à compressão, armazenamento, visualização e análise desses dados.

[1]WSI - Whole Slide Imaging (Imagens digitais de slides de tecidos).

Existem aplicações de bioinformática capazes de processarem essas imagens WSI e extraírem informações a respeito do estado de saúde de um paciente por meio da análise das estruturas encontradas nessas imagens, como por exemplo, o tamanho e o formato do núcleo das células presentes na amostra de tecido estudada [11, 37, 43, 44, 79**?**]. No entanto, essas aplicações são parametrizadas, e os valores desses parâmetros influenciam significativamente os resultados da análise. Encontrar o ajuste ideal dessas aplicações, ou seja, uma combinação de parâmetros de entrada que maximize a qualidade do resultado é uma tarefa complexa que envolve uma grande quantidade de combinações parâmetros a serem testados e de dados a serem processados. Normalmente essas aplicações vêm pré-configuradas com uma combinação de parâmetros que não é adequada para todos os tipos de imagens a serem processadas pela aplicação [70].

Para realizar esse ajuste de parâmetros, foi desenvolvido neste trabalho um sistema para otimizar a precisão e/ou minimizar o tempo de execução dessas aplicações de bioinformática. Para isso, foi desenvolvido um mecanismo de otimização multiobjetivo capaz de encontrar combinações de parâmetros melhores do que a configuração padrão oferecida por essas aplicações. Além disso, foram desenvolvidas múltiplas métricas para quantificar as alterações na morfologia das células e tecidos presentes nas imagens processadas, a fim de permitir análises mais precisas e eficientes.

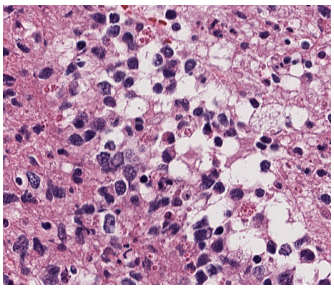
Foram utilizadas duas aplicações de bioinformática como caso de uso para este trabalho. Essas aplicações são parametrizadas e consistem em fluxos de operações e transformações que se dividem nos estágios de normalização, segmentação e extração de características. O primeiro estágio é o estágio da normalização que é responsável por corrigir e normalizar as cores da imagem que será processada. O segundo estágio é o da segmentação no qual ocorre a extração dos objetos e estruturas (ex.: núcleo das células) das imagens. O terceiro e último estágio, extração de características, dessas aplicações realiza a computação das características desses objetos, como por exemplo o perímetro e a forma predominante das estruturas, entre outras características. O estágio de segmentação, principal alvo deste trabalho, possui três funções principais: i) identificar os núcleos das células presentes nas imagens de tecidos, ii) extraí-los e iii) produzir uma máscara como saída. Uma máscara, por exemplo, pode ser uma imagem que atribui a cor branca aos objetos (células) identificados, e preto para o fundo (Figura 1.1c). Após a identificação dessas células, são realizadas uma série de análises a respeito da sua morfologia, comprimento, área, formato, a fim de se extrair características das mesmas.

Ambas aplicações são executadas sobre a plataforma de execução distribuída Region Templates [71]. A primeira aplicação possui 15 parâmetros de entrada que podem assumir múltiplos valores, de maneira que o espaço de busca total seja de 21,4 trilhões de pontos ao se considerar todas as combinações de valores de parâmetros possíveis. Já a segunda

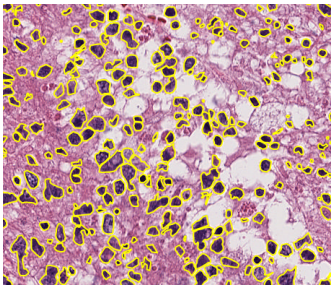
aplicação utiliza 7 parâmetros de entrada e possui um espaço de busca de 2,8 bilhões combinações de parâmetros possíveis. Além disso, cada execução dessas aplicações pode levar várias horas de processamento em uma estação de trabalho comum quando imagens em alta resolução são utilizadas.

O estágio de segmentação dessas aplicações de bioinformática é responsável por identificar e extrair as células presentes nas amostras de tecidos. É essencial que essa identificação seja precisa, de maneira que seja encontrado o maior número de células possível e que a extração delas seja realizada da maneira fidedigna. O grau de precisão da segmentação é afetado pelos parâmetros de entrada da aplicação. Dessa forma, é muito importante que essas aplicações sejam configuradas com combinações de parâmetros que produzam resultados com alta acurácia. Dado o tamanho do espaço de busca e o alto custo associado ao testar cada combinação de parâmetros, faz-se necessária uma estratégia de otimização para encontrar boas combinações utilizando-se uma quantidade pequena de testes, uma vez que a busca exaustiva é inviável. Deste modo, foi proposto neste trabalho um sistema de ajuste automático de parâmetros, com suporte a múltiplos algoritmos de otimização, para otimizar a escolha dos valores dos parâmetros de aplicações de bioinformática responsáveis pela segmentação nuclear em imagens WSI. O ajuste automático de parâmetros tem como objetivo melhorar a precisão e diminuir o tempo de execução dos algoritmos de segmentação (otimização multiobjetivo) das duas aplicações de bioinformática utilizadas como caso de uso neste trabalho.

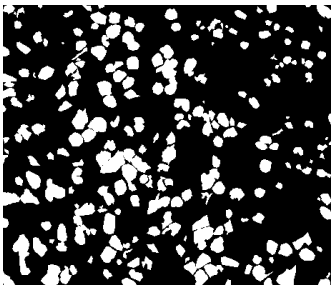
Para medir a qualidade das segmentações geradas, nós introduzimos um módulo de análises comparativas à plataforma Region Templates, a fim de oferecer suporte a várias métricas de avaliação qualitativa para os múltiplos algoritmos de otimização suportados. O objetivo é que as aplicações exemplo sejam otimizadas a fim de gerarem máscaras o mais semelhantes possível à máscara de referência anotada manualmente pelo patologista, conforme exemplo na Figura 1.1c. Uma vez ajustada, as aplicações exemplo serão capazes de reproduzir essa tarefa de identificação de células de maneira precisa e escalável.



(a) Exemplo de imagem de tecido humano utilizada como entrada para o estágio de segmentação das aplicações exemplo [21].



(b) Anotações manuais do patologista especialista na imagem (a). As partes circuladas são as células identificadas por ele. Elas serão pintadas de branco na máscara de referência (c) [21].



(c) Exemplo de máscara gerada a partir das anotações manuais do patologista. A saída do estágio de segmentação das aplicações exemplo produzem máscaras semelhantes a esta [21].

Figura 1.1: Exemplo de uma imagem de tecido humano que pode ser analisada pelas aplicações de bioinformática utilizadas como exemplo neste trabalho.

O sistema de ajuste automático proposto nesta dissertação possui atualmente suporte para quatro algoritmos de otimização, sendo que o primeiro deles foi implementado inteiramente neste trabalho e os outros três foram implementados com o auxílio de bibliotecas elaboradas por terceiros. O algoritmo de otimização desenvolvido neste trabalho é um algoritmo genético (GA) projetado para imitar os princípios evolutivos da seleção natural [19, 48]. Cada indivíduo da população GA foi modelado para representar um conjunto de parâmetros da aplicação, sendo assim uma potencial solução para o problema. Os outros três algoritmos de otimização suportados são o Nelder-Mead (NM)[53], o Parallel Rank Order (PRO) [76] e Otimização Bayesiana (Spearmint) [62]. Eles serão descritos em detalhes no Capítulo 3.

1.1 Problema

Aplicações de processamento de imagens são inerentemente parametrizadas e os valores desses parâmetros podem afetar os resultados significativamente. Ajustar esses parâmetros conforme a métrica de interesse é complexo devido à grande quantidade de combinações parâmetros a serem processados.

1.2 Principais Contribuições deste Trabalho

A principal contribuição dessa dissertação é o desenvolvimento de uma solução eficiente e eficaz capaz de otimizar a precisão e/ou minimizar o tempo de execução do estágio de segmentação nuclear de aplicações médicas. Para isso foi desenvolvido um sistema de otimização multiobjetivo que ajusta automaticamente os parâmetros dessas aplicações. Além disso, utilizou-se múltiplas métricas e mecanismos de consultas espaciais para quantificar as alterações na morfologia das células e tecidos em escala micro-anatômica a fim de realizar análises mais precisas e eficientes.

1.3 Contribuições Específicas

- Foi proposto e desenvolvido um sistema de otimização multiobjetivo com suporte a múltiplos algoritmos de otimização para ajuste de parâmetros de aplicações de bioinformática. Além disso, esse sistema de otimização multiobjetivo foi integrado ao ambiente de execução Region Templates como um módulo adicional para permitir que ele fosse utilizado em outros casos de uso.
- Foram desenvolvidas múltiplas métricas baseadas em consultas comparativas espaciais que são capazes de quantificar as alterações na morfologia das células e tecidos em escala micro-anatômica. Essas métricas e consultas espaciais foram adicionadas à plataforma Region Templates de forma que possam ser integradas à diferentes aplicações no futuro;
- Entre os algoritmos de otimização suportados, desenvolveu-se integralmente um algoritmo genético (GA) capaz de otimizar a qualidade e o tempo de execução. Foram executados múltiplos testes comparativos de precisão entre o GA e os outros algoritmos de otimização suportados pelo Region Templates. Este algoritmo mostrou desempenho igual ou superior, na maioria dos casos, em relação aos outros algoritmos de otimização suportados;
- Otimizou-se a qualidade e reduziu-se o tempo de execução das aplicações de segmentação nuclear de tecidos utilizadas como caso de uso neste trabalho.

Algumas dessas contribuições já foram publicadas no periódico *Bioinformatics* [70]. Entre elas estão o sistema de ajuste automático de parâmetros mono-objetivo, quatro das seis métricas atualmente suportadas pelo sistema e o algoritmo genético. O restante das contribuições, como por exemplo, o sistema de ajuste automático multiobjetivo, as novas métricas e os resultados experimentais, parte mais recente do trabalho, serão publicadas em um outro artigo que está em elaboração.

1.4 Organização do Texto

O restante deste trabalho está organizado da seguinte forma. No Capítulo 2, Referencial Teórico, são apresentados os conceitos fundamentais a respeito da análise de imagens em patologia, os casos de uso utilizados neste trabalho, o funcionamento do ambiente de execução Region Templates e suas principais características, os conceitos da otimização multiobjetivo e o funcionamento dos Algoritmos Genéticos clássicos. No Capítulo 3, explica-se a visão geral da solução proposta neste trabalho através do detalhamento do módulo de ajuste automático de parâmetros e do módulo de Análises Comparativas do Region Templates, os algoritmos de otimização suportados e as métricas propostas e implementadas para avaliar a qualidade das segmentações. O Capítulo 4 discute os procedimentos experimentais utilizados para o sistema de ajuste automático de parâmetros e compara os resultados dos algoritmos de otimização suportados utilizando as métricas desenvolvidas. No Capítulo 5 são apresentadas as conclusões, as contribuições, e as perspectivas para a continuação do trabalho.

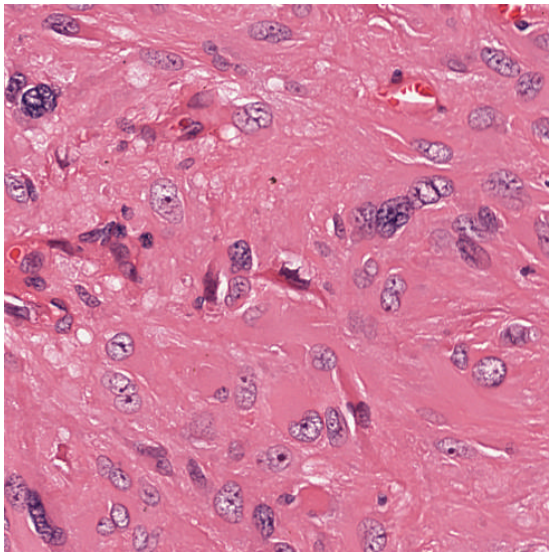
Capítulo 2

Referencial Teórico

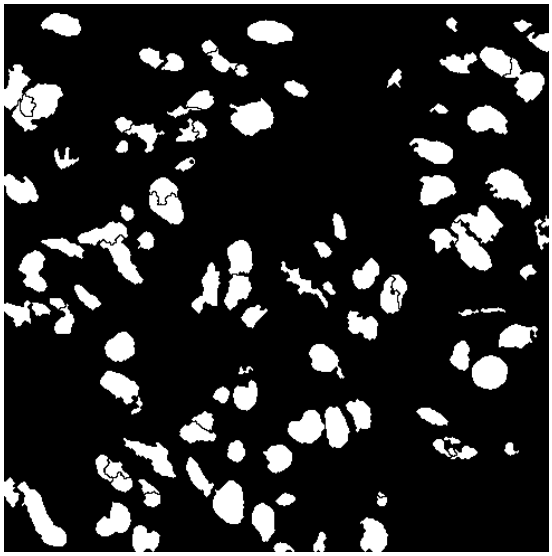
Este capítulo descreve os conceitos fundamentais e as ferramentas utilizadas nesse trabalho. A primeira seção deste capítulo explica o contexto das análises de imagens em Patologia e os desafios relacionados. A Seção 2.2 detalha as duas aplicações utilizadas como caso de uso deste trabalho. Na Seção 2.3, descreve-se o Region Templates, uma plataforma de execução distribuída de aplicações médicas com suporte a diversas bibliotecas para processamento de imagens. Em seguida, a Seção 2.4 apresenta os conceitos da otimização multiobjetivo. A Seção 2.5 detalha as características gerais dos algoritmos genéticos e a motivação por utilizá-los neste trabalho. E por último, a Seção 2.6 descreve trabalhos da literatura relacionados ao ajuste de parâmetros em aplicações de segmentação e os compara com esta dissertação.

2.1 Análise de Imagens em Patologia

Os avanços nos instrumentos de captura de imagens microscópicas têm tornado cada vez mais viável capturar imagens digitais de slides de tecidos (WSI) extraídos de seres humanos ou animais. As alterações morfológicas nesses tecidos, quando examinados em escalas celulares e sub-celulares, fornecem informações valiosas sobre o estado de saúde do paciente, complementando as informações clínicas [79]. Essas análises permitem ao patologista realizar diagnósticos mais precisos, avaliar a resposta do paciente a medicamentos e orientar a terapia. A caracterização quantitativa dos dados dessas imagens microscópicas, conforme observado nas Figuras 2.2 e 2.3, envolvem um processo de [15, 55, 71]: (1) corrigir a coloração dos artefatos da imagem (estágio da normalização); (2) detectar e extrair objetos micro-anatômicos, tais como os núcleos e as células (estágio da segmentação) [11, 22, 38, 39, 42, 82]; (3) computar as características morfológicas e moleculares destes objetos (estágio de computação de características); e (4) monitorar e quantificar as alterações destes objetos ao longo do tempo.



(a) Imagem antes da segmentação.



(b) Máscara gerada após a segmentação. As máscaras geradas pelo processo de segmentação são compostas de duas cores: branco e preto, sendo que a primeira serve para identificar os objetos (células nesse caso) e a segunda para identificar o fundo.

Figura 2.1: Recorte de uma imagem digital de slides de tecido humano (WSI) [15].

O processamento de uma única imagem de 120K x 120K *pixels* de resolução envolve a extração de milhões de estruturas micro-anatômicas e cálculo de dezenas de características (*features*) por estrutura. Este processo leva várias horas em uma estação de trabalho comum, quando executado sequencialmente. Instrumentos de captura de imagens WSI modernos podem gerar centenas dessas imagens por dia. Uma imagem não-comprimida, pode atingir cerca de 50GB [71]. Dessa forma, verifica-se a necessidade de uma abordagem de alto desempenho para realizar o processamento eficiente dessas imagens. Nesse contexto, foi desenvolvido em trabalhos anteriores, uma plataforma de execução distribuída, denominada Region Templates [67, 71], para realizar o processamento distribuído dessas imagens médicas [69]. Esta plataforma é capaz de executar aplicações de bioinformática que tratam essa imensa quantidade de imagens médicas de maneira eficiente e distribuída.

Um exemplo de banco de imagens WSI é o TCGA[1]. Este e outros bancos de WSIs, têm servido de base para múltiplos trabalhos científicos [36, 52] que objetivam melhorar a capacidade de diagnosticar, tratar e prevenir variados tipos de câncer através de uma melhor compreensão da base genética desta doença. No Brasil, também existem iniciativas tanto privadas quanto públicas de bancos de imagens de tumores. Entre elas, algumas iniciativas se destacam: o *Banco Nacional de Tumores*, do Instituto Nacional do Câncer

[1]TCGA - The Cancer Genome Atlas - https://cancergenome.nih.gov

(INCA)[28]; o *A. C. Camargo Biobank*, mantido pelo hospital A. C. Camargo[9]; e o *Banco de Tumores Ricardo Renzo Brentani* localizado no Hospital do Câncer de Barretos [8]. Estes bancos de tumores brasileiros são muito importantes para o avanço do tratamento do câncer no Brasil, uma vez que eles armazenam amostras de tecidos da população local. Esses dados servem por exemplo, para se fazer uma análise mais aprofundada e desenvolver medicamentos e terapias voltadas para os perfis genéticos da população brasileira, os quais diferem dos perfis genéticos encontrados nos bancos de tumores do hemisfério norte [8], como por exemplo o TCGA.

2.2 Casos de Uso

Foram utilizadas duas aplicações de bioinformática como exemplo neste trabalho. Ambas realizam uma caracterização quantitativa das imagens WSI recebidas como entrada. As duas aplicações são baseadas em um fluxo de operações que se dividem em três estágios: normalização, segmentação e extração de características. Por fim, a saída consiste em múltiplos vetores de características que podem ser utilizados para cruzar informações com os dados clínicos do paciente ou com bases de dados de patologias.

Esses dois fluxos de trabalho compartilham a mesma estrutura de alto nível, mas implementam o estágio de segmentação usando estratégias diferentes. A primeira aplicação (Tabela 2.1 e Figura 2.2) utiliza Operações Morfológicas e *watershed* na segmentação [37], enquanto a segunda (Tabela 2.2 e Figura 2.3) realiza a segmentação utilizando operações baseadas em *Level Set* [25] e possui duas estratégias de *declumping* (desaglomeração de estruturas, serve para separar objetos muito próximos ou justapostos), a serem escolhidas pelo usuário no momento da execução: i) *Mean Shift*, ii) *Watershed Declumping*. Como essas estratégias de *declumping* são métodos comuns encontrados na literatura para separar objetos justapostos, nós realizamos experimentos neste trabalho para medir o impacto dessas estratégias na qualidade do resultado e desempenho no processo de segmentação.

As cascatas de operações empregadas por cada um dos fluxos de trabalho são apresentadas nas respectivas figuras, 2.2 e 2.3. As duas aplicações se dividem em três estágios: *i*) normalização, *ii*) segmentação e *iii*) computação de características. Os estágios *i* e *iii* são iguais em ambas aplicações. Já o estágio *ii*, da segmentação, é implementado de maneira diferente em cada caso de uso.

Tabela 2.1: Lista dos 15 parâmetros do estágio de segmentação da aplicação baseada em Operações Morfológicas. O espaço de busca desse estágio é de aproximadamente 21,4 trilhões de pontos.

Parâmetro	Descrição	Escopo da variação dos parâmetros
B/G/R	Cores de detecção para o fundo da imagem	B, G, R $\in [210, 220, ..., 240]$
T1/T2	Limiar de detecção das células vermelhas no sangue	T1,T2 $\in [2.5, 3.0, ..., 7.5]$
G1/G2	Limites para identificar o conjunto inicial de possíveis núcleos	G1 $\in [5, 10, ..., 80]$ G2 $\in [2, 4, ..., 40]$
MinSize	Descarta objetos cuja área (pixels) < seja menor do que MinSize	MinSize $\in [2, 4, ..., 40]$
MaxSize	Descarta objetos cuja área (pixels) > seja maior do que MaxSize	MaxSize $\in [900, 950, ..., 1500]$
MinSizePl	Filtra objetos cuja área seja menor do que MinSizePl	MinSizePl $\in [5, 10, ..., 80]$
MinSizeSeg	Filtra objetos cuja área seja menor do que MinSizeSeg	MinSizeSeg $\in [2, 4, ..., 40]$
MaxSizeSeg	Filtra objetos cuja área seja maior do que MaxSizeSeg	MaxSizeSeg $\in [900, 950, ..., 1500]$
FillHoles Structure	Elemento estrutural que define a região de propagação	FillHoles $\in$ [4-conn, 8-conn]
MorphRecon Structure	Elemento estrutural que define a região de propagação	MorphRecon $\in$ [4-conn, 8-conn]
Watershed Structure	Elemento estrutural que define a região de propagação	Watershed $\in$ [4-conn, 8-conn]

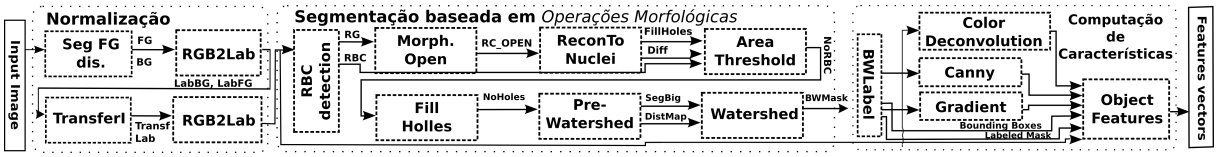


Figura 2.2: Fluxo de operações baseado em Operações Morfológicas. Aplicação exemplo para realizar análises em imagens de microscopia que utiliza a técnica de *Watershed* para separar e delinear as células segmentadas.

Tabela 2.2: Lista dos 7 parâmetros do estágio de segmentação da segunda aplicação, baseada em *Level Set*. O espaço de busca desse estágio é de aproximadamente 2,8 bilhões de pontos.

Parameter	Description	Range Value
OTSU	Valor de peso atribuído ao limiar OTSU	OTSU $\in [0.3, 0.2, ..., 1.3]$
Curvature Weight	Peso de curvatura das funções *level set*	CW $\in [0.0, 0.05, ..., 1.0]$
MinSize	Tamanho mínimo dos objetos segmentos em micron por dimensão	MinSize $\in [1, 2, ..., 20]$
MaxSize	Tamanho máximo dos objetos segmentos em micron por dimensão	MaxSize $\in [50, 55, ..., 400]$
Mpp	Controla a variabilidade dos resultados	Mpp $\in [0.25]$
MsKernel	Raio espacial do cálculo de *Mean Shift*	MsKernel $\in [5, 6, ..., 30]$
LevetSetIt	Número de iterações da computação *Level Set*	LevetSetIt $\in [5, 6, ..., 150]$

Tipicamente segmenta-se (extrai-se) os núcleos das células em cada imagem de tecido, definindo-se um espaço citoplasmático em torno de cada um dos núcleos. As características de cada objeto (células ou núcleos) são calculadas a fim de descrever a sua forma e textura na fase seguinte, denominada de estágio de computação de características. As propriedades das estruturas micro-anatômicas de cada paciente são calculadas para gerar um perfil morfológico do paciente. Os perfis morfológicos de múltiplos pacientes são agrupados e classificados utilizando-se algoritmos de aprendizado de máquina na fase de classificação.

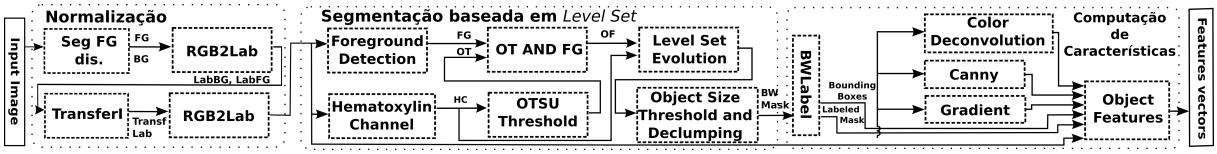


Figura 2.3: Fluxo de operações baseado em *Level Set*. Segunda aplicação exemplo para realizar análises em imagens de microscopia. Esta aplicação suporta 2 tipos de métodos de *declumping*: i) *Mean Shift*, ii) *Watershed Declumping*.

Normalmente, as aplicações de análise de imagens de Patologia são sensíveis a variações nos parâmetros de entrada. Uma aplicação otimizada para um grupo de imagens ou para um tipo de tecido pode não funcionar bem para outros tipos de tecidos ou imagens. Por exemplo, o estágio de segmentação identifica as células presentes em um tecido [4], e os limites dessas células segmentadas dependem do algoritmo utilizado e de seus parâmetros [54]. Os parâmetros determinam, por exemplo, os limiares de intensidade para detecção e separação de núcleos agrupados. Ou seja, os resultados da segmentação (o número e localização das células detectadas, a forma e os limites de um núcleo, etc.) são afetados pelos parâmetros de entrada.

A Figura 2.4 mostra os resultados da etapa de segmentação nuclear de duas aplicações utilizadas como caso de uso deste trabalho para uma mesma imagem de tecido. Os dois fluxos de operações em análise têm um bom acordo em algumas regiões (os limites das células segmentadas se sobrepõem estreitamente) e grande desacordo em outras regiões, onde um algoritmo não tem núcleos segmentados enquanto o outro tem, ou há grandes diferenças entre os limites de um núcleo segmentado ao se comparar os resultados dos dois algoritmos (anotações em azul claro e azul escuro na Figura 2.4).

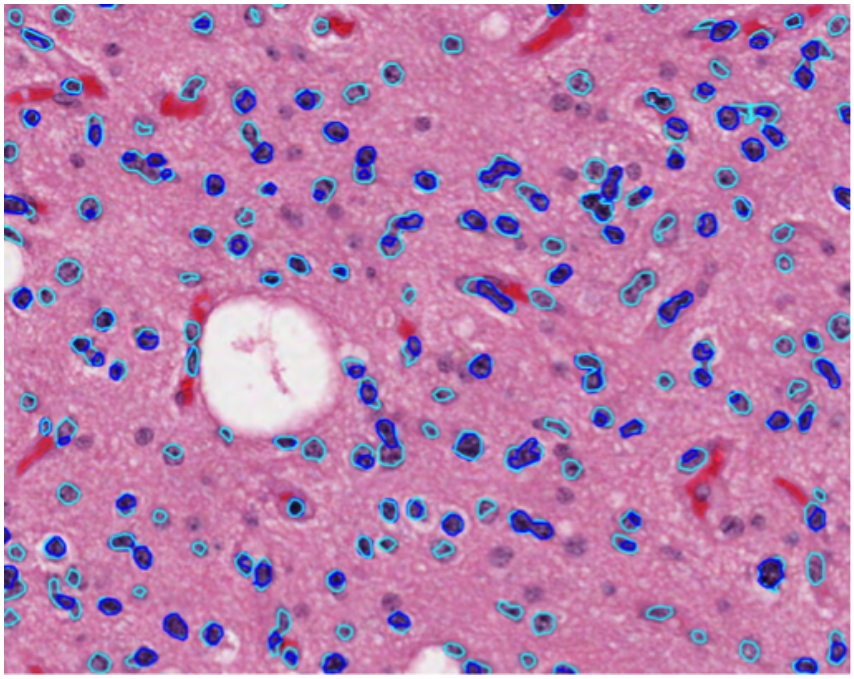


Figura 2.4: Diferença de segmentação entre os dois casos de uso (anotações em azul claro e azul escuro) [21].

Como o espaço de busca das possíveis combinações de parâmetros das aplicações de segmentação é muito grande e cada execução dessas aplicações podem levar várias horas para serem executadas, não é possível realizar uma busca exaustiva de todas as combinações de parâmetros a fim de se encontrar a melhor delas. Logo, se faz necessário utilizar metodologias mais eficientes para se encontrar boas combinações de parâmetros utilizando-se apenas poucos testes. Por isso, neste trabalho é proposto um sistema de ajuste automático de parâmetros atualmente com suporte a quatro tipos de algoritmos de otimização para realizar essa busca de maneira eficiente. Em estudos desta escala é imperativo utilizar sistemas de computação de alto desempenho para acelerar os processos de estudo de parâmetros.

2.3 Region Templates

O Region Templates é um sistema de execução de alto desempenho para sistemas distribuídos híbridos. Ele oferece estruturas de dados otimizadas, que podem ser utilizadas por desenvolvedores e usuários de aplicações médicas para processar, sob-demanda, grandes conjuntos de imagens de microscopia em um sistema de computação híbrido [71].

A plataforma de execução possui suporte para fluxos de trabalhos hierárquicos. Um fluxo de trabalho hierárquico permite que uma aplicação seja descrita como um fluxo

de dados entre macro componentes (estágios), nos quais cada macro componente também pode ser implementado como um fluxo de dados entre tarefas (Figura 2.5). Essa representação permite flexibilidade e melhor desempenho na distribuição de tarefas em sistemas híbridos. O ambiente de execução instancia os componentes/estágios da aplicação e executa as tarefas de cada estágio respeitando as dependências entre os estágios para garantir a execução correta da aplicação. O gerenciamento da execução dos componentes da aplicação é feito nas máquinas de memória distribuída e em cada nó de computação individualmente. Além disso, a plataforma de execução implementa otimizações para execução eficiente de tarefas em sistemas equipados com CPU-GPU[2]-MIC[3] híbrido.

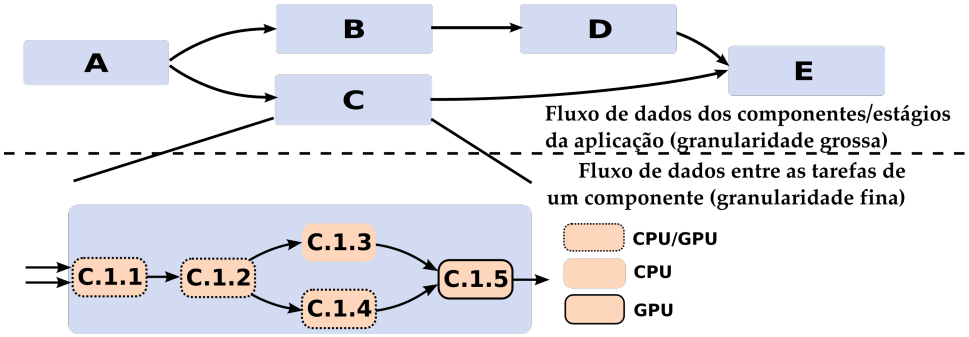


Figura 2.5: Modelo de aplicação de fluxo de trabalho hierárquico com dois níveis que pode ser executada sobre a plataforma Region Templates. Cada estágio de um fluxo em análise pode ser expresso como um outro grafo de operações de grão fino. Isso resulta em um fluxo de computação hierárquico (dois níveis). Durante a execução, os estágios podem ser mapeados para nós de computação diferentes. As operações de grão fino são despachadas como tarefas e agendadas para execução com CPUs e GPUs nesse nó.

As dependências entre os componentes e os dados de E/S dos estágios da aplicação são fornecidas à plataforma em tempo de execução pela própria aplicação. O ambiente de execução coordena as transferências de dados enquanto analisa as dependências de dados e tarefas entre os componentes da aplicação. As transferências de dados são realizadas em segundo plano por *threads* de E/S. Elas interagem com as implementações apropriadas do armazenamento de dados para recuperar ou gravar dados de acordo com a hierarquia de memória definida pela aplicação. São suportados diversos níveis de persistência de dados (Figura 2.6), variando-se a capacidade e a velocidade dos dispositivos.

[2] *Graphics Processing Units* (GPU), conhecidas também como placas de vídeo.

[3] *Many Integrated Core* (MIC), arquitetura altamente escalável com múltiplos núcleos em um único processador [16, 68], suas características são semelhantes às unidades de processamento gráfico (GPUs) além de compatibilidade com a arquitetura x86. Um exemplo de dispositivo que possui essa arquitetura é o coprocessador Intel® Xeon Phi ™.

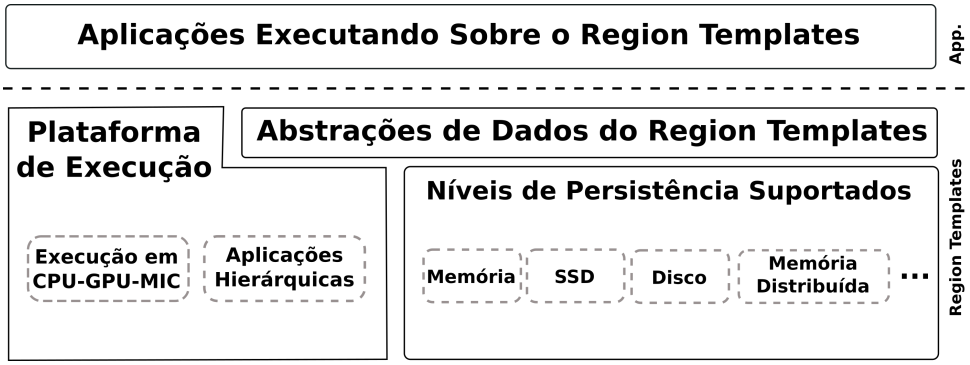


Figura 2.6: Arquitetura do Region Templates.

2.3.1 Estruturas de Dados Otimizadas

A plataforma oferece um modelo de contêiner genérico para estruturas de dados comuns nesse domínio de aplicação, como pontos, matrizes, regiões e conjuntos de objetos, dentro de uma caixa delimitadora espacial e temporal. Além disso ela provê implementações para diferentes mecanismos de armazenamento, transferência e gerenciamento de dados. O Region Templates oferece duas abstrações para o desenvolvedor de aplicações utilizando esses tipos, que são: os modelos de região (*region templates*) e as regiões de dados (*data regions*).

A abstração **modelo de região** (*region templates*) permite estratégias diferentes de gerenciamento de dados e operações de E/S, além de fornecer uma interface unificada e homogênea às aplicações para armazenar e recuperar dados. Os dados consumidos e produzidos pelos componentes das aplicações, que estão sendo executadas sobre a plataforma, são gerenciados em contêineres de armazenamento fornecidos pela abstração de dados do modelo de região. Os tipos de dados suportados nesta abstração incluem estruturas de dados que descrevem espaços de baixa dimensionalidade (espaços 1D, 2D ou 3D) com uma componente temporal, como por exemplo, pixels, pontos, vetores, matrizes, volumes 3D, polígonos e superfícies que podem representar objetos e regiões segmentadas de uma imagem WSI. A abstração de dados de modelo de região implementa mecanismos de transporte de dados eficientes para mover dados entre estágios/componentes da aplicação, que podem ser executados em nós diferentes de uma máquina de memória distribuída. Em vez de gravar dados através de fluxos de dados (*streams*) como em uma aplicação de fluxo de dados típica, os componentes da aplicação inserem dados no modelo de região que então poderão ser consumidos por outros componentes/estágios da aplicação.

Um objeto de uma **região de dados** (*data region*) é uma materialização do armazenamento dos elementos de dados de uma região contida em um modelo de região. Uma instância de um modelo de região pode conter múltiplas regiões de dados referentes a

mesma região no espaço e tempo. Por exemplo, um modelo de região pode conter a imagem original de um tecido e diferentes máscaras dessa mesma região computadas a partir de diferentes combinações de parâmetros. As aplicações interagem com as regiões de dados para ler e escrever dados que serão compartilhados entre os diversos estágios de processamento. Os tipos de regiões de dados atualmente suportados pelo sistema são implementados utilizando a biblioteca OpenCV[4]. Esta biblioteca suporta vários tipos de dados que incluem Pontos, Vetores, Matrizes, Matrizes Esparsas, etc. Uma característica adicional do OpenCV é o suporte a GPUs, que inclui algumas das estruturas de dados e métodos de processamento utilizados nas aplicações de bioinformática.

Os modelos de região e as regiões de dados podem estar relacionados a outros modelos de regiões e regiões de dados. Regiões de dados correspondentes a uma mesma área espacial podem conter diferentes tipos de dados e produtos de dados. Por exemplo, as regiões de dados podem estar relacionadas entre si para expressar estruturas em diferentes escalas que ocupam o mesmo espaço. As regiões de dados também podem estar relacionadas entre si para expressar a evolução de estruturas e características ao longo do tempo. As informações de relação espacial e temporal podem ser usadas pela plataforma de execução para tomar decisões sobre a distribuição e o gerenciamento das regiões de dados em um ambiente de computação distribuído. As regiões de dados são identificadas por uma tupla *(chave, tipo do dado, tempo, número de versão)*, conforme a Figura 2.7b. Esta tupla permite a identificação e diferenciação de relações temporais entre regiões de dados relativas a uma mesma área espacial.

Essas abstrações foram projetadas para permitir realizar análises em grandes conjuntos de imagens de alta resolução sobre *clusters* de nós computacionais híbridos (com suporte a GPUs ou MICs) de maneira eficiente e distribuída.

O Region Templates é utilizado neste trabalho para processar imagens em Patologia [27, 41, 70, 73], no entanto, ele pode ser aplicado em outros contextos, uma vez que existe uma classe ampla de casos de uso que utilizam as mesmas estruturas de dados (matrizes, pontos, regiões espaciais). Por exemplo, aplicações para processamento de imagens de satélite, subsolo e caracterização de reservatórios de petróleo, e imagens de telescópios [10, 35, 40, 59].

2.3.2 Plataforma de Execução Distribuída

A plataforma de execução apoia a definição, análise e o gerenciamento estruturas de dados espaciais e temporais em máquinas híbridas de memória distribuída. As aplicações executando sobre a plataforma Region Templates, são estruturadas no formato de um fluxo de dados. Elas interagem com as regiões de dados e os modelos de região para

[4]OpenCV - Biblioteca de visão computacional - http://opencv.org

```
class RegionTemplate {
private:
  // Data regions: 1D/2D/3D regular
  // or irregular, and polygons
  std::map<std::string,
     std::list<DataRegion*> >
     templateRegions;

  // Region template name identifier
  std::string name;

  // Region template coordinates
  BoundingBox bb;

  // If false, data are automatically
  // read during component creation
  bool lazyRead;
          ...
public:
  bool insertDataRegion(
    DataRegion *dataRegion);
  DataRegion *getDataRegion(
    std::string namespace,
    int timestamp=0, int version=0,
    int type);

  int getNumDataRegions();
  DataRegion *getDataRegion(int idx);
  void instantiateRegion();
          ...
};
```

(a) Classe simplificada do Modelo de Região.

```
class DataRegion {
private:
  // Type of each data element CHAR, UCHAR,...
  int elementType;
  // Dense, sparse, (2D/3D),...
  int regionType;
  int version, timestamp;
  // Name and instance identifier
  std::string name, id;
  // Resolution of the region
  int resolution;
  // BB surrounding domain and ROI
  BoundingBox bb, ROI;

  // Associates a given bounding box to an id,
  // for data that may be split into pieces.
  std::vector<std::pair<BoundingBox,
      std::string> > bb2Id;

  // Type of storage used to read: distributed
  //shared memory and high performance disk
  int inputDataSourceType, outpuDataSourcetType;
          ...
public:
  DataRegion();
  virtual ~DataRegion();
  virtual bool instantiateRegion();
  virtual bool write();
  virtual bool empty();
          ...
};
```

(b) Classe simplificada da Região de Dados.

Figura 2.7: Código fonte das classes do Modelo de Região e da Região de Dados. Uma estrutura de dados de modelo de região pode armazenar várias regiões de dados, que são distinguidas pelo seu identificador de tupla. A região de dados define uma classe base que é herdada por implementações concretas de diferentes tipos de região de dados. Cada aplicação pode implementar os seus próprios tipos de dados dessa forma. O sistema provê por padrão implementações dos seguintes tipos de região de dados: regiões densas ou esparsas 1D, 2D ou 3D, polígonos [71].

armazenar e recuperar elementos de dados, em vez de manipularem explicitamente o armazenamento, e a distribuição das estruturas de dados entre as unidades de processamento. Essas abstrações permitem que desenvolvedores de aplicações de bioinformática possam implementar versões distribuídas de suas aplicações de maneira simples.

A representação hierárquica de fluxo de dados permite que diferentes estratégias de escalonamento de tarefas possam ser utilizadas em cada nível do fluxo. Tarefas de grão fino podem ser despachadas para execução em um nó de computação com um escalonador de tarefas adequado para os dispositivos presentes naquela máquina. Com esta estratégia, é possível explorar a variabilidade desempenho em tarefas de grão fino para melhor utilizar os dispositivos disponíveis (CPU, GPU e MIC) nas múltiplas máquinas presentes em um ambiente distribuído.

A plataforma de execução distribuída implementa um modelo de execução de *Manager-Worker*. O *Manager* da aplicação cria as instâncias dos estágios/componentes da aplicação (granularidade grossa). A instanciação de cada estágio inclui a preparação das regiões de dados de entrada e exportação das dependências daquele estágio em relação aos outros estágios. O grafo de dependência entre os estágios da aplicação não é necessariamente conhecido antes da execução. Ele pode ser construído incrementalmente, em tempo de execução, uma vez que um estágio pode criar outras instâncias de estágios. A atribuição de trabalho do *Manager* para os nós *Workers* é feita na escala dos componentes da aplicação, e não em nível de tarefas. O *Manager* atribui instâncias de estágios aos nós *Workers* à medida que eles solicitam trabalho, repetidamente, até que todas as instâncias dos estágios da aplicação tenham sido executadas (Figura 2.8).

Cada nó *Worker* pode executar várias instâncias de um mesmo estágio ao mesmo tempo, a fim de tirar vantagem de vários dispositivos de computação disponíveis em um nó. A comunicação entre o *Manager* e os nós *Workers* é implementada usando MPI[5]. Os dados de entrada e saída da aplicação são lidos e escritos pelos componentes da aplicação usando regiões de dados globais, que são implementadas a partir de modelos de região, que permitem a comunicação entre estágios. Uma vez que um estágio é recebido e instanciado em um nó *Worker*, ele identifica todos os modelos de região usados por esse estágio, aloca memória localmente nesse nó para armazenar as regiões de dados associadas e comunica com a implementação de armazenamento de dados apropriada para recuperar esses dados (Seção 2.3.3). Somente após os dados estarem prontos na memória local do nó, a instância do estágio pode começar a executar.

Cada estágio da aplicação é capaz de usar vários dispositivos de computação em um nó *Worker*. Os dispositivos são usados cooperativamente enviando tarefas de grão fino

[5]Message Passing Interface (MPI) é um padrão para comunicação de dados em computação paralela - https://www.open-mpi.org

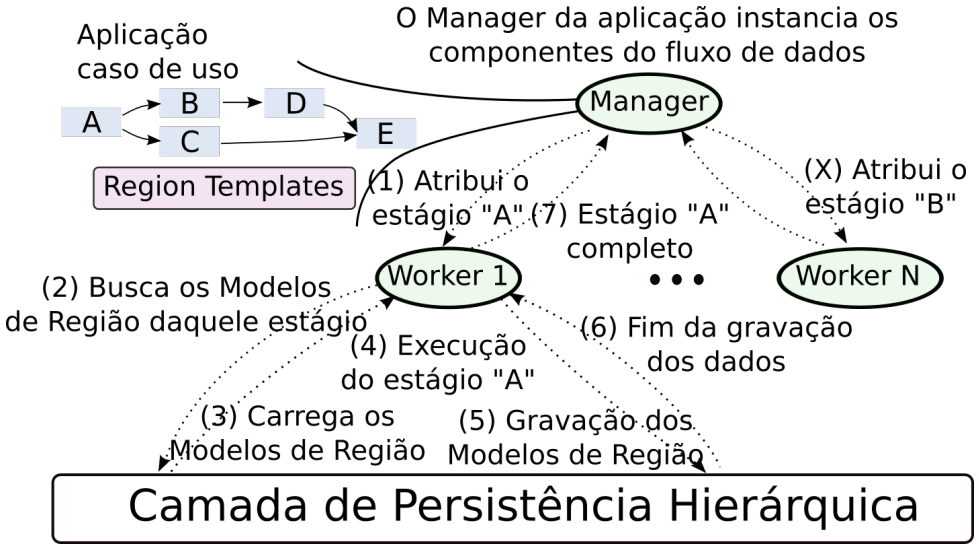


Figura 2.8: Visão geral do modelo de execução. O modelo de execução é construído em cima de um modelo *Manager-Worker*. O desenvolvedor da aplicação implementa uma parte do módulo *Manager* que instancia o fluxo de trabalho do aplicativo. O *Manager* cria quantas instâncias de cada estágio forem necessárias e define as dependências entre elas. Durante a execução, o nó *Manager* atribui instâncias dos estágios da aplicação para serem computados nos nós *Workers* sob demanda.

para execução em um núcleo de CPU ou um co-processador (GPU ou MIC). Como várias instâncias de estágio podem estar ativas no mesmo nó, as tarefas podem ter sido criadas por estágios diferentes.

As tarefas de grão fino criadas por um estágio são despachadas para execução pelo WRM (*Worker Resource Manager*) em cada nó. O WRM instancia uma *thread* de computação para cada núcleo de CPU e cada co-processador. Sempre que ociosas, as *threads* informam o WRM, que seleciona uma das tarefas prontas para execução e as atribuem a essa *thread*.

Coordenar de maneira eficiente as interações e as operações entre as diferentes unidades de processamento em sistemas híbridos, equipados com várias CPUs; GPUs; e MICs, é uma tarefa complexa. Para gerir esta complexidade, utilizou-se mecanismos de adaptações dinâmicas para dividir as tarefas entre as unidades de processamento de maneira eficiente durante a execução da aplicação, entre elas, destaca-se o escalonador de tarefas *PATS* responsável por coordenar o uso de CPUs, GPUs e MICs [66, 69, 72, 74, 80]. O escalonador *PATS* mapeia funções para as CPUs, GPUs e MICs baseado em valores estimados de *speedup* de acordo com o dispositivo, a fim de utilizar o poder de computação das diferentes unidades de processamento de maneira mais eficiente. A plataforma de execução pode restringir a execução de uma operação a um dispositivo específico quando apenas uma unidade de processamento estiver disponível.

2.3.3 Gerenciamento do Armazenamento de Dados e Otimizações

O Region Templates possui suporte para armazenamento de dados hierárquico (HDS - *Hierarchical Data Storage*). O HDS é um mecanismo de gerenciamento de dados distribuídos com suporte a um número arbitrário de níveis de armazenamento. A hierarquia de armazenamento é definida em um arquivo de configuração externo à aplicação que inclui o número de níveis de armazenamento, a descrição e posição de cada nível na hierarquia, o tipo dos dispositivos de armazenamento (por exemplo, RAM, SSD, HDD), a sua capacidade e etc.

O HDS armazena e recupera as regiões de dados produzidas ou solicitadas por uma aplicação. As regiões de dados de saída são armazenadas automaticamente pelo ambiente de execução no nível mais alto da hierarquia que possua capacidade suficiente. Quando um nível de armazenamento está cheio, uma estratégia de substituição de dados seleciona as regiões de dados que devem ser movidas para um nível de armazenamento mais baixo. As políticas de substituição de dados suportadas são: *First-In, First-Out* (FIFO) e *Least Recently Used* (LRU).

O Region Templates possui um escalonador de tarefas (DLAS) que utiliza conhecimento da localidade dos dados entre os nós de processamento e as suas respectivas hierarquias de memória. Este escalonador considera a localização dos dados ao mapear as instâncias de estágios do fluxo de trabalho da aplicação, por exemplo, quando uma tarefa de segmentação termina, o escalonador leva em consideração a localidade dos dados produzidos por essa operação para determinar o nó no qual as operações de análise, que irão consumir esses dados, serão executadas. O DLAS calcula a quantidade de reutilização de dados que essas operações de análise terão em cada nó e insere essas tarefas de execução em uma fila de operações preferenciais para o nó onde a região de dados foi produzida. Uma fila de operações preferenciais é mantida para cada nó em ordem decrescente da quantidade de reutilização de dados. Quando um nó solicita trabalho, o ambiente de execução atribui a operação com a máxima reutilização de dados para esse nó.

2.4 Otimização Multiobjetivo de Parâmetros

Na otimização de um único objetivo, também chamada de mono-objetivo, tem-se como objetivo encontrar o mínimo ou o máximo da função-objetivo do problema a ser otimizado em questão. Já no caso da otimização multiobjetivo (OMO), os problemas de otimização são caracterizados pela existência de mais de um critério a ser otimizado.

A otimização multiobjetivo trata problemas que possuem objetivos que são conflitantes entre si, de maneira que soluções próximas a um ótimo de um único objetivo exigem um compromisso nos demais (cenários conflitantes). Esses problemas de otimização que

requerem a otimização de mais de um objetivo são chamados problemas de otimização de vetores ou multiobjetivo [12]. As soluções desses problemas podem ser modeladas em um vetor de variáveis de decisão. Cada dimensão do vetor representa um objetivo da função a ser otimizada.

As soluções que satisfazem as funções objetivo constituem o que se chama espaço de busca das variáveis, que corresponde ao local onde se faz a busca pelas soluções do problema, ou seja, é o domínio das variáveis do problema. Este espaço de busca corresponde ao domínio de variáveis do problema. Para cada possível solução x neste espaço de busca, há um ponto y no espaço de objetivos. Uma solução é referida como um vetor de variáveis e um "ponto" como o correspondente vetor objetivo [20] (Figura 2.9).

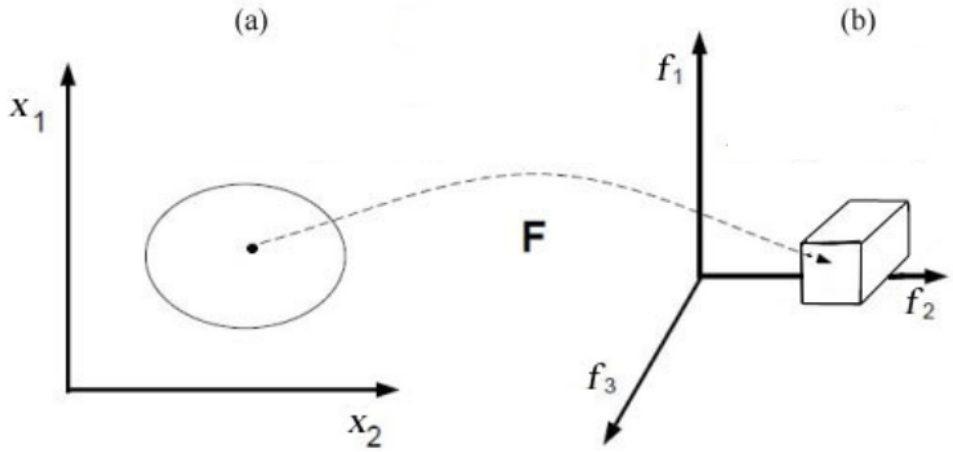


Figura 2.9: Existe um mapeamento de todas as variáveis (pontos) do domínio da função para o espaço dos objetivos (b) (vetor de objetivos). (Imagem adapatada de [56]).

2.4.1 Dominância, Otimalidade e Frente de Pareto

Nos problemas de otimização multiobjetivo, otimizar significa encontrar todos os valores ótimos para as funções objetivo com vistas a uma tomada de decisão. As soluções de problemas de otimização multiobjetivo recebem nomes diferentes, quando representadas nos espaços de variáveis e de objetivos. No espaço das variáveis (domínio da função) elas são chamadas de Conjunto de Pareto. No espaço dos objetivos (imagem da função) de Fronteira de Pareto.

Uma determinada solução x pertence ao conjunto (fronteira) de Pareto se não existir nenhuma outra solução x', capaz de melhorar algum dos objetivos (em relação a x) sem simultaneamente piorar pelo menos um dos demais.

Todas estas soluções que pertencem ao conjunto (fronteira) de Pareto são ditas **não-dominadas**. Ou seja, a solução x dominará quaisquer outras soluções que sejam piores do que ela em pelo menos um dos objetivos e que seja ao mesmo tempo, igual ou pior do que ela nos outros objetivos (Figura 2.10)..

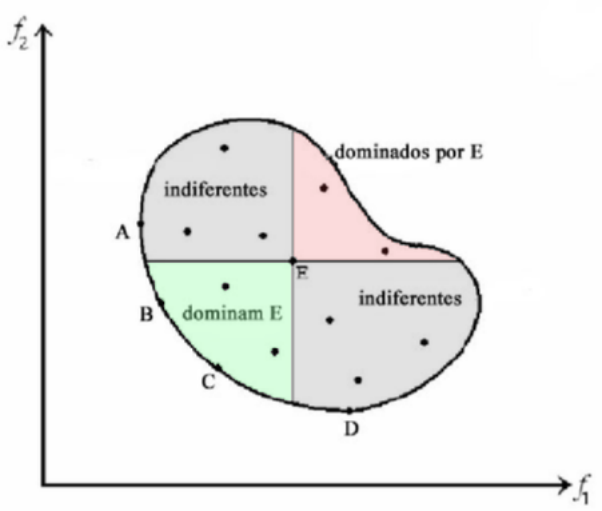


Figura 2.10: A dominância de uma solução em relação à outra permite comparar a qualidade de duas soluções em um problema de otimização multiobjetivo. O Conjunto Imagem do mapeamento das soluções de um problema de otimização de dois objetivos foi dividido em 4 quadrantes relação a uma solução arbitrária E. O quadrante inferior-esquerdo corresponde aos mapeamentos que dominam E, ou seja, que são melhores do que ele em pelo menos um dos objetivos. Os quadrantes superior-esquerdo e inferior-direito são considerados indiferentes em relação a E pois podem ser melhores em um objetivo mas piores no outro. E por último, o quadrante superior-direito, corresponde aos mapeamentos dominados por E. (Imagem adapatada de [56]).

Todos as soluções que não são dominadas por nenhuma outra solução (vetores de decisão do espaço das variáveis) são chamadas de não-dominadas ou conjuntos ótimos de Pareto. Já o mapeamento correspondente dessas soluções são denominados de fronteira de Pareto, ou frente global de Pareto (Figura 2.11). A forma e o tamanho da frente de Pareto dependem, normalmente, do número de objetivos e da função objetivo.

A otimização multiobjetivo normalmente caracteriza-se por preservar as soluções até então não-dominadas encontradas ao longo da busca (mínimos locais) para então continuar procurando novas soluções a fim de progredir continuamente em direção à Fronteira de Pareto. É importante que os algoritmos de otimização sejam capazes de manter a diversidade das soluções tanto no espaço de objetivos (fronteira de Pareto) quanto no espaço de variáveis, para que ao final da otimização eles possam retornar ao usuário uma quantidade suficiente (mas ao mesmo tempo limitada) de soluções e um conjunto de Pareto mais completo.

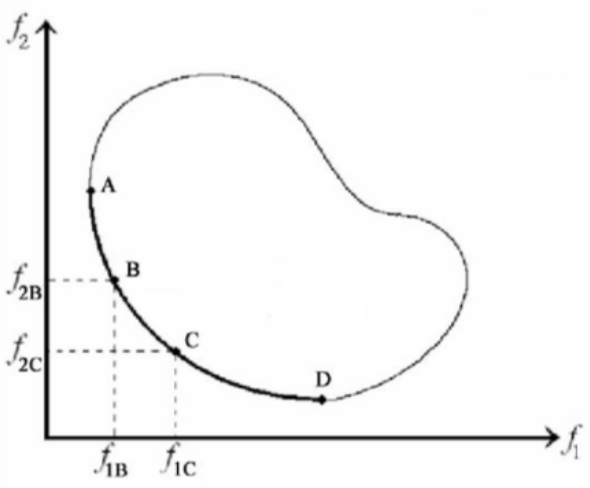


Figura 2.11: Representação gráfica da frente global de Pareto, denominada também de mapeamento do Conjunto dos ótimos de Pareto. Todos os pontos nesse segmento correspondem a mapeamentos de soluções não dominadas no espaço das variáveis, ou seja, que não existem nenhuma outra solução que sejam melhores do que elas em pelo menos um objetivo sem piorar algum outro objetivo.(Imagem adapatada de [56]).

2.4.2 Abordagens Para Resolução de Problemas Multiobjetivo

Geralmente existem múltiplas soluções ótimas para os problemas de otimização multiobjetivo (Frente de Pareto). Isso significa que resolver um problema desse tipo não é a mesma coisa que resolver um problema de otimização mono-objetivo convencional. Por isso, diferentes pesquisadores definiram o termo "Resolução de Problemas Multiobjetivo"de maneiras distintas [14].

Na literatura, tipicamente se resolve problemas de otimização multiobjetivo através de uma de três abordagens fundamentais [13]: (i) **Inserção de preferências *a posteriori*** na qual primeiramente busca-se encontrar o maior número de soluções possível, para só depois selecionar a mais adequada ao problema; (ii) **Inserção de preferências *a priori*** que é uma estratégia em que se tem de antemão alguma prefrência sobre o tipo de solução mais adequada ao problema, então a busca é direcionada para encontrar este tipo de soluções; (iii) **Inserção progressiva de preferências** é feito um direcionamento da busca por meio das escolhas de um Tomador de Decisão (uma pessoa especializada no domínio do problema) durante a otimização, para regiões que contenham soluções mais adequadas.

Este trabalho emprega uma abordagem de inserção de preferências *a priori*. As principais razões para a sua escolha baseiam-se no fato de que (a) no nosso problema, é inviável testar todas as combinações de parâmetros das aplicações que estão sendo otimizadas de antemão, de modo que calcular a frente de Pareto completa seria muito caro devido ao

alto custo da função de teste (segmentação), inviabilizando assim as estratégias *a posteriori*; (b) o uso de uma estratégia de inserção progressiva de preferências ao longo da busca não é possível porque nós desejamos realizar a otimização de maneira automática, sem precisar da intervenção de um especialista do domínio durante a execução da otimização.

Dentre as possíveis estratégias de inserção de preferências *a priori*, escolheu-se utilizar a escalarização [49]. Esta é uma abordagem que permite solucionar eficientemente problemas de otimização multiobjetivo com a adaptação de métodos da otimização mono-objetivo.

2.4.3 Escalarização

Os problemas de otimização multiobjetivo podem ser convertidos em um problema de otimização mono-objetivo pelo agrupamento dos objetivos em uma função escalar [78, 83]. A este processo denomina-se escalarização. Nem sempre é recomendável realizar a escalarização do resultado (pontuação numérica), uma vez que existem problemas de otimização multiobjetivo que possuem critérios em diferentes sistemas de valores, como por exemplo o grau de preservação de espécies vegetais em perigo e a promoção do desenvolvimento econômico de uma região [56].

O método de escalarização dos resultados utilizado nesta dissertação de metrado foi uma soma ponderada (combinação linear) dos objetivos. Para realizar essa combinação, é necessário primeiro escalonar (normalizar) todos os objetivos para a mesma faixa.

$$f(x) = \sum_{i=1}^{N} w_i * f_i(x)$$

Este método consiste em atribuir pesos (w) para cada objetivo (i), de modo que a soma dos pesos seja igual a 1. A função de otimização a ser minimizada é a soma ponderada dos pesos atribuídos a cada objetivo multiplicados pelo valor de cada objetivo. O valor de N corresponde ao número total de objetivos.

2.4.4 Teorema da Inexistência de Almoço Grátis

O Teorema da Inexistência de Almoço Grátis (*No Free Lunch Teorem* ou NFL) [81] afirma que todos os algoritmos de otimização que não utilizam conhecimento prévio a respeito da função de otimização, possuem exatamente o mesmo desempenho médio ao se considerar todos os infinitos problemas de otimização existentes [45]. Ou seja, se um algoritmo x é melhor que algum outro algoritmo y para uma série de n problemas, então deve haver uma outra série de n problemas que o algoritmo y é melhor do que o algoritmo x.

No entanto, um algoritmo desenvolvido especificamente para um problema x, ou seja, que utiliza algum conhecimento prévio a respeito da função que será otimizada, incluindo suas restrições e mapeamentos especiais para benefício da solução, terá um desempenho superior se comparado a outros algoritmos para os quais não foram introduzidos nenhuma espécie de conhecimento prévio do domínio do problema. Em contrapartida, o seu desempenho se deteriorará rapidamente para problemas diferentes de x. Um algoritmo genérico, sem adição de conhecimento prévio da função de otimização, terá sempre um desempenho razoável, mas não irá superar o algoritmo específico na classe de problemas para as quais este foi desenvolvido [56].

Os algoritmos genéticos são bons exemplos de algoritmos de otimização em que é possível embutir conhecimento específico da natureza do problema a fim de melhorar o seu desempenho. Por exemplo, é possível inicializar a população do algoritmo genético com indivíduos com combinações de parâmetros que já são sabidamente bons a fim de tentar agilizar o processo de convergência. Além disso, também é possível modelar os operadores de mutação e de cruzamento para privilegiarem as variáveis do domínio do problema que mais possuem influência no resultado.

2.5 Algoritmos Genéticos

Um algoritmo genético (GA) é um tipo de algoritmo de otimização baseado no processo de seleção natural presente na natureza. Os algoritmos genéticos são uma subclasse dos algoritmos evolucionários (EA), suas otimizações são feitas por meio de sucessivas aproximações utilizando-se técnicas baseadas em mecanismos naturais como a evolução, mutação, herança, seleção e cruzamento de indivíduos [26].

Os algoritmos genéticos possuem diferenças fundamentais em relação a outros algoritmos de otimização tradicionais, entre elas destacam-se as seguintes: não necessitam de nenhuma informação prévia do problema, mas apenas de uma forma de avaliação do resultado e de um mapeamento do domínio do problema para as estruturas de dados do algoritmo; os resultados da otimização são apresentados como uma população das melhores soluções encontradas e não como uma resposta única; diversos estágios do algoritmo utilizam transições probabilísticas ao invés de regras determinísticas, como por exemplo na inicialização e mutação aleatória da população.

Essa classe de algoritmos evolutivos foi escolhida para ser um dos algoritmos de otimização da ferramenta Region Templates devido às seguintes características [46]:

- **Busca Codificada:** Para resolver problemas utilizando-se um algoritmo genético (GA) é necessário que o conjunto de soluções viáveis para o problema possa ser codificado em uma população de indivíduos. No caso dessa dissertação, as entradas das

funções de segmentação utilizadas como caso de uso requerem 15 ou 7 parâmetros cada uma. Dessa forma, cada conjunto de parâmetros que formam uma possível entrada da função de segmentação foi modelada em um indivíduo, de maneira que cada parâmetro corresponda a um gene do indivíduo. Ao longo das gerações (iterações do algoritmo) esses indivíduos sofrerão mutações e recombinações de maneira similar ao Evolucionismo proposto por Darwin [17].

- **Paralelismo Explícito:** Cada indivíduo da população existe como um ente isolado e por consequência, eles podem ser avaliados de forma independente. Essa independência entre as tarefas permite um alto grau de paralelismo durante o processo evolutivo de uma geração.

- **Paralelismo Implícito:** Ao fazer uma busca por populações mais adaptadas (parâmetros que produzem resultados melhores) a evolução de um algoritmo genético tende a privilegiar indivíduos que compartilham determinadas características, sendo capaz de detectar implicitamente combinações de genes (parâmetros) que são mais ou menos desejáveis. Esse efeito é denominado de busca por hiperplanos, de natureza paralela.

- **Busca Guiada:** Um GA convencional ignora o significado das estruturas que ele manipula e não sabe qual a maneira mais eficiente de se trabalhar com elas (busca cega). Essa característica traz generalidade à solução, porém limita a eficiência na descoberta dos resultados. Para resolver este problema, é possível utilizar heurísticas para guiar o processo de mutação dos genes (parâmetros) a fim de alcançar os resultados desejados mais rapidamente. Por exemplo, é possível realizar um estudo no grau de influência que cada parâmetro, quando variado isoladamente, possui no resultado de uma determinada função, e a partir disso, priorizar as variações em parâmetros mais importantes a fim de aumentar a velocidade de convergência do algoritmo.

Nesta seção será descrito o funcionamento dos algoritmos genéticos de maneira geral, conforme são apresentados na literatura [26, 47]. Na Seção 3.2.4 é que será explicado o algoritmo genético que de fato foi integrado à plataforma Region Templates e a maneira como ele foi implementado. Além disso, será mostrado como cada fase do algoritmo foi configurada dada as diversas possibilidades de escolha de cada fase do algoritmo genético.

Como primeira etapa do algoritmo genético clássico (Figura 2.12), são selecionados um conjunto de soluções para compor a população inicial do algoritmo (Inicialização da População). Após isso, na etapa *Avaliação*, aplica-se a função a ser otimizada utilizando os parâmetros codificados nos genes de cada indivíduo da população. Em seguida, na etapa *Seleciona Reprodutores*, seleciona-se um conjunto de indivíduos da população para reproduzirem entre si. Estes indivíduos são então submetidos aos processos de *Cruzamento* (*crossover*) e *Mutação*. Feito isso, novos indivíduos são gerados. Estes indivíduos são avaliados na etapa *Avaliação* e então substituem a população anterior no estágio *Substituição*, iniciando-se uma nova geração. Esse processo é repetido de acordo com o número de iterações (gerações) pré-definidas ou caso algum critério de convergência pré-definido seja alcançado.

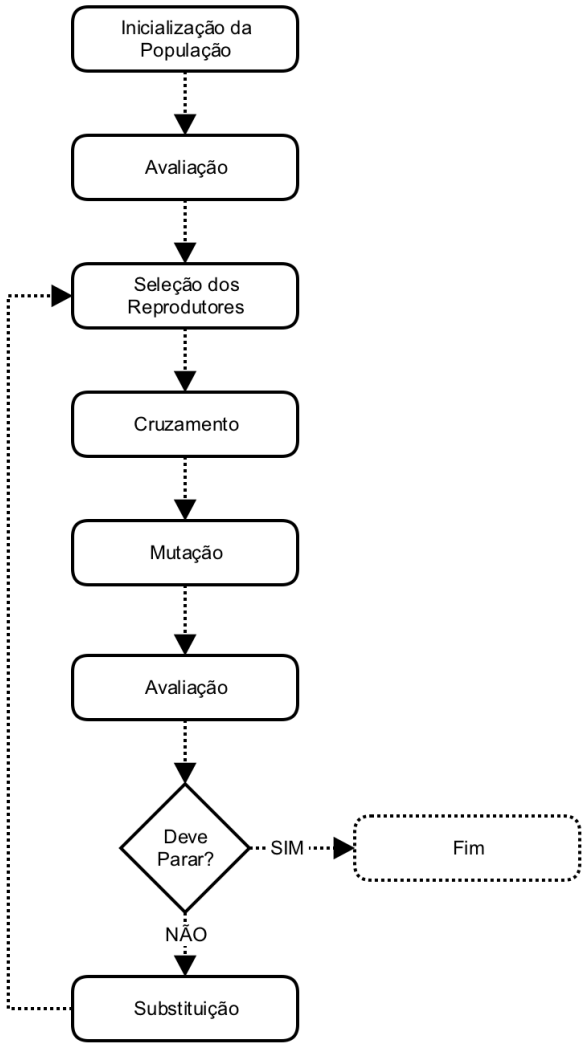


Figura 2.12: Fluxograma de um algoritmo genético típico [48].

Inicialização da População Após modelar o problema para as estruturas de dados do algoritmo genético, deve-se gerar a população inicial. Essa modelagem pode ser feita de diversas maneiras, como por exemplo, modelar as soluções em strings binárias, simulando o cromossomo de um indivíduo ou modelando cada parâmetro do problema em uma posição de um vetor de números reais. Após a modelagem, pode-se iniciar a execução do algoritmo. Para definir a população inicial do algoritmo, é necessário considerar o seu tamanho e como os seus indivíduos serão escolhidos. Esses valores são definidos com base no domínio do problema e do ambiente de execução do algoritmo, devendo-se levar em conta as limitações de tempo e espaço da memória da máquina que irá executar o algoritmo.

Avaliação Na Teoria da Evolução, o termo *fitness* está relacionado com o grau de adaptação dos indivíduos ao ambiente em que se encontram, ou seja, com a capacidade deles de sobreviverem e se reproduzirem a fim de perpetuarem a espécie [17]. Os indivíduos mais adaptados (com maior *fitness*) conseguem contribuir com seus genes para a geração seguinte. Ou seja, as características desses indivíduos vão ter uma maior probabilidade de se perpetuarem. Os algoritmos genéticos simulam este mesmo processo, de modo que as soluções que apresentam os melhores resultados transmitem os seus genes para a geração seguinte. O valor da *fitness* de um indivíduo é obtido através medição do resultado da aplicação dos parâmetros codificados no material genético do indivíduo na função que está sendo otimizada pelo algoritmo genético.

Seleção dos Reprodutores Esta etapa serve para selecionar os membros da população que serão escolhidos para a etapa de cruzamento. Existem diversos métodos de seleção de indivíduos que podem ser aplicados [26] [47], como por exemplo, a seleção pelo método da roleta em que cada indivíduo recebe um espaço numa roleta proporcional ao seu valor de *fitness*, de maneira que os indivíduos mais adaptados terão mais chances de serem selecionados para a geração seguinte ou para a seleção. Um outro método bastante similar ao método da roleta é o método de *ranking* em que ao invés da probalidade de seleção de um indivíduo estar relacionado ao valor do seu *fitness*, ela está relacionada a sua posição na ordenação dos indivíduos da população. Outra forma de seleção comum de ser utilizada é o método de torneio. Neste método divide-se a população de indivíduos em pequenos sub-grupos (normalmente de tamanho 2), a fim de se criar um torneio entre os indivíduos de cada sub-grupo e selecionar os melhores indivíduos de cada um deles. Os indivíduos são selecionados para os torneios com igual probabilidade. Os campeões dos torneios são então submetidos à etapa de cruzamento..

Cruzamento (*Crossover*) O cruzamento, também conhecido como *crossover*, serve para combinar o material genético (genes) de dois indivíduos (genitores) para produzir dois novos indivíduos (filhos) para uma nova população potencial. O operador de *crossover* pode ser aplicado ou não em um par de indivíduos selecionados para a reprodução, de acordo com a taxa de *crossover* selecionada pelo usuário do GA. Esta taxa normalmente é um valor entre 50% a 90% de chances de se aplicar o operador. Dado um par de indivíduos cujo o tamanho do material genético (vetor de genes) seja l, os tipos mais comuns de crossover são:

- *Crossover* em um ponto: Neste *crossover* escolhe-se um ponto aleatório no material genético de dois indivíduos da população do GA, k, entre 0 e l-1, onde l é o tamanho do vetor de genes. Troca-se todos os genes após este ponto k no material genético dos genitores para gerar os novos indivíduos. A representação gráfica desse processo está na Figura 2.13.

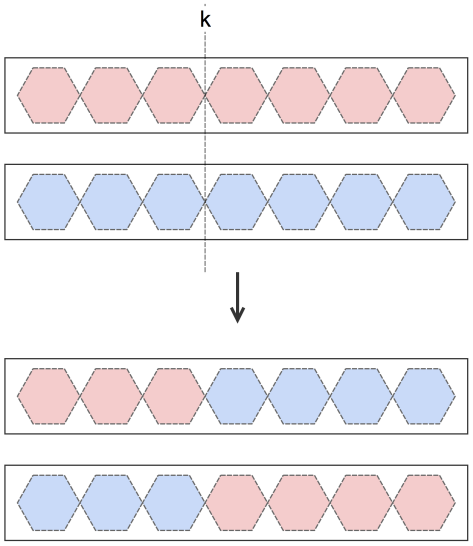


Figura 2.13: Representação gráfica do *crossover* de 1 ponto.

- *Crossover* em dois pontos: Escolhe-se dois pontos aleatórios, k e i, sendo que $k \neq i$, e ambos estão no intervalo entre 0 e l-1, onde l é o tamanho do vetor de genes. Troca-se todos os genes entre os pontos k e i no material genético dos genitores. A representação gráfica desse processo está na Figura 2.14.

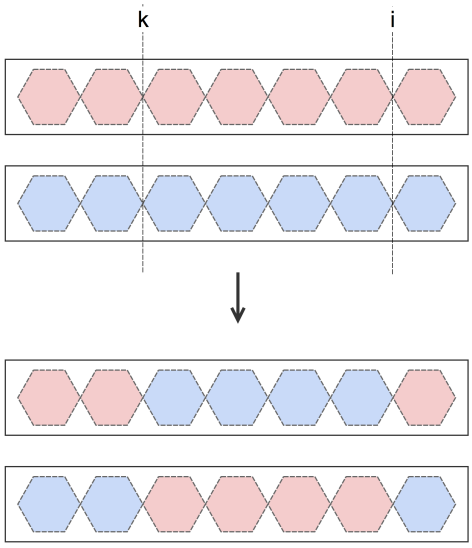


Figura 2.14: Representação gráfica do *crossover* de 2 pontos.

- *Crossover* uniforme: Todos os genes dos genitores podem ser trocados aleatoriamente com uma certa probabilidade p, chamada de probabilidade de troca. Normalmente o valor adotado para esta probabilidade é entre 50% e 90%.

Mutação O operador de mutação é aplicado individualmente ao membros da população, diferentemente do operador de cruzamento que é aplicado sempre sobre um par de indivíduos. Quando dois indivíduos possuem o mesmo gene e são cruzados entre si, os indivíduos filhos desse par também terão o mesmo gene. Para solucionar casos como esses, o operador de mutação permite adicionar uma diversidade adicional ao alterar o valor de um gene aleatoriamente com uma pequena probabilidade p, denominada taxa de mutação. Esta probabilidade normalmente é um valor pequeno, para evitar que o GA se transforme em uma busca aleatória. Ela é definida de acordo com o domínio do problema e a função de otimização em questão.

Substituição Após aplicar os operadores de mutação e *crossover* na população de novos indivíduos, esta nova população poderá substituir a anterior. Múltiplas técnicas podem ser utilizadas para realizar essa substituição. Entre elas, é possível simplesmente substituir todos os n elementos da população antiga pelos elementos da nova população. Outra opção é escolher os n indivíduos mais adaptados, levando-se em consideração a população anterior e a população dos novos indivíduos, para fazer parte da população da próxima geração do algoritmo.

Assim como os algoritmos genéticos, os algoritmos *Simulated Annealing* e *Memetic Algorithms* são baseados em meta-heurísticas para realizarem suas buscas pelo espaço de

parâmetros. Eles utilizam uma combinação de escolhas probabilísticas e conhecimento histórico dos resultados anteriores para guiarem o seu processo de otimização.

Arrefecimento Simulado (*Simulated Annealing*) É uma técnica de otimização, que ao invés de guardar uma população de indivíduos igual os algoritmos genéticos, armazena apenas uma única solução. O espaço de busca é percorrido por meio de mutações aleatórias sobre essa solução. O grau de mutação é uma função de um parâmetro T, denominado temperatura em alusão às técnicas de metalurgia nas quais este algoritmo foi inspirado. Diminui-se esse parâmetro T sistematicamente para restringir progressivamente a variabilidade da solução individual. Quando uma solução melhor é encontrada, subtitui-se a antiga pela nova, e aplica-se a mutação nessa nova solução. O algoritmo termina quando a temperatura chega a um valor próximo de zero ou quando o sistema estiver estável. Ele garante apenas que um máximo/mínimo local seja encontrado, mas não o global [29, 34].

Algoritmos Meméticos (*Memetic Algorithms*) Os algoritmos meméticos [51] são considerados uma adaptação dos algoritmos genéticos , uma vez que também apresentam um mapeamento dos parâmetros da função a ser otimizada para um abordagem populacional. A diferença para os GA consiste que os indivíduos de um algoritmo memético passam por procedimentos individuais de aprendizagem a fim de acelerar a melhoria local. Os algoritmos meméticos foram inspirados pelos memes [18], que ao contrário dos genes, podem adaptar-se. Essa adaptação individual de um meme seriam os procedimentos individuais de aprendizagem que um indivíduo da população passaria a fim de encontrar vizinhos próximos melhores do que ele.

2.6 Ajuste de Parâmetros em Aplicações de Segmentação

Existem na literatura múltiplas iniciativas de pesquisa que realizam o ajuste de parâmetros em aplicações de segmentação de imagens médicas. Entre elas, destaca-se o Tuner, trabalho de Torsney-Weir et al.[77], que utiliza mecanismos visuais para o usuário escolher as melhores combinações de parâmetros e visualizar o grau de impacto da variação de cada parâmetro em aplicações de segmentação de ressonâncias cerebrais. Nesse trabalho, eles também aferem a qualidade do resultado a partir da métrica Dice Coefficient [23]. No entanto, o ajuste de parâmetros não é realizado de maneira automática como em nosso trabalho. Outros trabalhos também utilizam mecanismos visuais para ajustar

os parâmetros de maneira não automática de diversos tipos de aplicações [58], como por exemplo na análise de tecidos [54] e de imagens de ressonâncias cerebrais em 3D [57].

No artigo de Held-Nattkemper et al.[30] é proposto um sistema de ajuste automático de parâmetros para aplicações de segmentação nuclear. Eles utilizam a métrica Jaccard [32] para avaliar a qualidade do resultado e a acurácia da segmentação ao quantificar a quantidade de células identificadas corretamente. O ajuste automático de parâmetros desse trabalho possui suporte a quatro algoritmos de otimização: *Hill Climbing*, *Steepest Ascent Hill Climbing*, *Coordinate Descent* e um Algoritmo Genético (GA). Dentre os resultados obtidos, o algoritmo genético foi o algoritmo de otimização que apresentou os melhores resultados. Diferentemente desse artigo, o nosso trabalho permite a otimização multiobjetivo das aplicações de segmentação nuclear, suporte a múltiplas métricas comparativas e a utilização de mecanismos de execução distribuída e eficiente. Além deste, existem outros trabalhos [31] que demonstram que a utilização de algoritmos genéticos se mostra uma estratégia adequada para ajustar os parâmetros de aplicações de segmentação nuclear.

Uma iniciativa brasileira, apresentada no artigo de Feitosa-Costa et al.[24], propõe um método de ajuste automático de parâmetros de uma aplicação de segmentação de imagens geográficas com base em Algoritmos Genéticos. A qualidade do resultado da segmentação é avaliada com base na similaridade da segmentação produzida pela aplicação com uma máscara de referência fornecida pelo usuário. Realizou-se um conjunto de experimentos em um conjunto de imagens de satélite e o método foi capaz, na maioria dos casos, de aproximar-se da solução ideal.

Utilizar algoritmos genéticos em problemas de otimização relacionados à area médica é uma estratégia comum na literatura. No trabalho de Mookiah-Acharya [50] eles foram utilizados para otimizar a precisão de classificadores de algoritmos de aprendizagem de máquina. Neste artigo eles demonstram a utilização de algoritmos de processamento de imagens e de aprendizado de máquina para pré-processar imagens digitais da retina ocular de pacientes com diabetes e glaucoma, além de extrair características e classificar essas imagens.

Tabela 2.3: Quadro comparativo dos artigos mencionados nesta seção

Quadro Comparativo da Literatura						
Paper	**Aplicação Alvo**	**Métrica**	**Técnica**	**Ajuste Automático**	**Ajuste Multiobjetivo**	**Execução Distribuída**
Torsney-Weir et al.[77]	Segmentação Nuclear	Dice	Ajuste Visual e Manual de Parâmetros	Não	Não	Não
Sedlmair-Heinzl et al.[58]	Múltiplas (Simulações Climáticas, Proliferação de Doenças, entre outras)	MDS Plots	Ajuste Visual e Manual de Parâmetros, Predição de Resultados por meio de Surrogate Models	Não	Não	Não
Pretorius-Zhou et al.[54]	Segmentação Nuclear	Múltiplas (Diagrama de dispersão, distribuição, entre outras)	Ajuste Visual e Manual de Parâmetros	Não	Não	Não
Schultz-Kindlmann et al.[57]	Segmentação de Imagens Médicas (Ressonâncias Cerebrais e Pulmonares)	Dice	Ajuste Visual e Manual de Parâmetros (Spectral Clustering)	Não	Não	Não
Held-Nattkemper et al.[30]	Segmentação Nuclear	Jaccard	Ajuste Automático (Hill Climbing, Steepest Ascent H C, Coordinate Descent, Algoritmo Genético)	**Sim**	Não	Não
Feitosa-Costa et al.[24]	Segmentação de Imagens Geográficas	Área de Intersecção	Ajuste Automático (Algoritmo Genético)	**Sim**	Não	Não
Teodoro-Kurc-Taveira et al.[70]	Segmentação Nuclear	Dice, Jaccard, Intersecção e Diff Pixels	Ajuste Automático (GA, NM, PRO, Bayesiana)	**Sim**	Não	**Sim**
Esta Dissertação	Segmentação Nuclear	General Dice, Individual Dice, Average Dice, Jaccard, Intersecção, Diff Pixels	Ajuste Automático (GA, NM, PRO, Bayesiana)	**Sim**	**Sim**	**Sim**

Em resumo, um dos principais diferenciais do nosso trabalho em relação à literatura (Tabela 2.3) é permitir que o ajuste automático de parâmetros das aplicações de segmentação nuclear seja realizado de maneira eficiente e distribuída em ambientes de alto desempenho e com suportes a múltiplas métricas de comparação [70]. Além disso, o nosso sistema permite uma otimização multiobjetivo dessas aplicações, levando em conta não apenas o ganho na qualidade do resultado como também na velocidade de execução do algoritmo de segmentação.

Capítulo 3

Ajuste Automático de Parâmetros

A primeira seção deste capítulo descreve o sistema de Ajuste Automático de Parâmetros (*auto-tuning*) integrado ao Region Templates capaz de selecionar um ou mais conjuntos de valores de parâmetros que produzam resultados significativamente melhores para uma determinada aplicação de segmentação nuclear de imagens médicas ou que reduzam o tempo de execução dela quando comparados com os resultados obtidos ao utilizar os parâmetros padrão dessa aplicação. A segunda seção descreve os mecanismos de Análises Comparativas propostos para quantificar e medir as alterações das estruturas segmentadas das imagens WSI, como veias, células, e os seus respectivos tecidos.

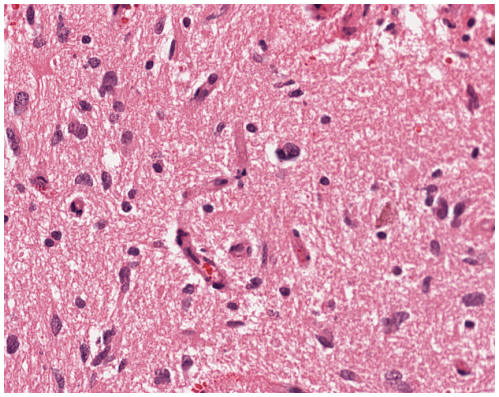
Tanto o sistema de Ajuste Automático de Parâmetros como os mecanismos de Análises Comparativas foram implementados como módulos adicionais à plataforma Region Templates (Figura 3.1). O primeiro módulo é responsável por selecionar progressivamente um ou mais conjuntos de valores de parâmetros que produzam resultados cada vez melhores para uma determinada função a ser otimizada. E o segundo módulo é capaz de calcular diversas métricas e realizar processamentos espaciais sobre os objetos extraídos das imagens médicas ou outro tipo de fonte de informações espaciais.

Figura 3.1: Em destaque azul estão os módulos desenvolvidos neste trabalho para a plataforma Region Templates.

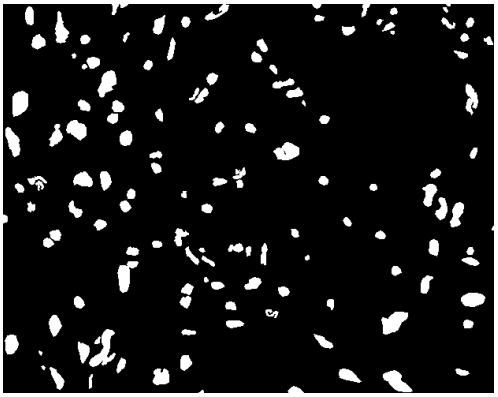
3.1 Sistema de Ajuste Automático de Parâmetros

Tipicamente, as aplicações de bioinformática vêm pré-configuradas com valores padrão de parâmetros que são utilizados independentemente do tipo de imagem que está sendo processada. Esses parâmetros não são otimizados para todos os tipos de imagens [54], e por isso, as aplicações acabam produzindo resultados piores do que poderiam produzir se estivessem configuradas corretamente.

A Figura 3.2 ilustra os efeitos que a otimização de parâmetros de uma aplicação pode proporcionar na precisão dos algoritmos de segmentação. Os objetos pintados de verde nas Figuras 3.2e e 3.2f são os objetos que estão presentes na máscara em análise e na máscara de referência (Fig.3.2b). Os objetos pintados de azul estão na máscara anotada pelo patologista (máscara de referência) mas não estão na máscara gerada pela aplicação (falso negativo). E os objetos vermelhos estão presentes na máscara computada mas não estão presentes na máscara de referência (falso positivo). Quanto mais objetos pintados de verde, melhor. Após a otimização, a máscara gerada pela aplicação fica mais similar à máscara anotada pelo patologista do que a máscara gerada com os parâmetros padrão.

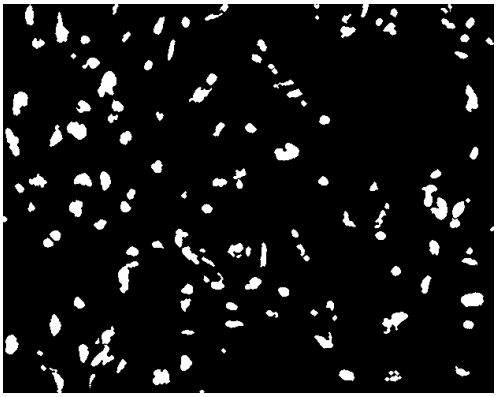


(a) Imagem do tecido antes da segmentação.

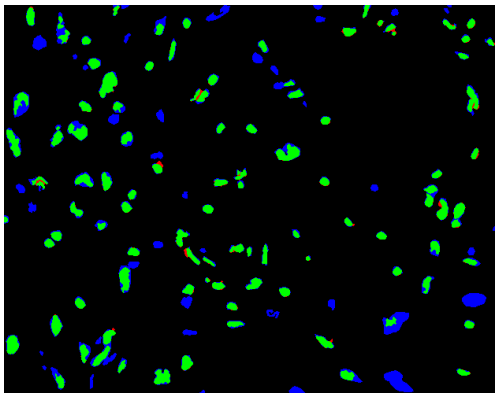


(b) Máscara gerada pelas anotações manuais do patologista especialista na área.

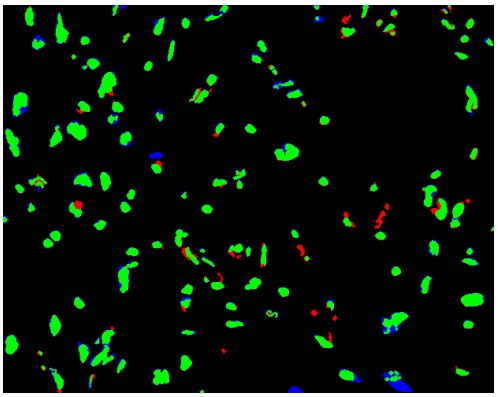
(c) Máscara gerada pela aplicação utilizando os parâmetros padrão da aplicação.



(d) Máscara gerada após a otimização dos parâmetros da aplicação.



(e) Nesta imagem, a Fig. 3.2c foi colorida para demonstrar a diferença dela com a máscara de referência (Fig. 3.2b).



(f) A máscara gerada após a otimização dos parâmetros da aplicação (Fig. 3.2d) foi colorida para demonstrar a diferença dela com a máscara de referência (Fig. 3.2b).

Figura 3.2: Este conjunto de imagens demonstra o aprimoramento da qualidade das máscaras após a aplicação do algoritmo de otimização [70]. Os objetos pintados de verde estão presentes na máscara em análise e na máscara de referência, os objetos em azul correspondem aos falsos negativos e os objetos vermelhos aos falsos positivos.

Com base nesse contexto, foi desenvolvido um sistema de Ajuste Automático de Parâmetros mono-objetivo e multiobjetivo para encontrar combinações de parâmetros que otimizem a acurácia ou a acurácia e o tempo de execução das aplicações de segmentação. O processo de otimização dessas aplicações está ilustrado na Figura 3.3. Inicialmente, a plataforma de execução do Region Templates instancia a aplicação médica a ser otimizada para todos os conjuntos de parâmetros obtidos a partir do algoritmo de otimização selecionado. São suportados quatro algoritmos de otimização atualmente: o Nelder-Mead simplex (NM) [53], o Paralel Rank Order (PRO) [76], o Spearmint (Otimização Bayesiana) [62] e o algoritmo genético (GA). A máscara de segmentação resultante de cada conjunto de parâmetros é comparada com um conjunto de dados de referência (por exemplo, uma máscara de segmentação de imagens anotada por um patologista) usando uma métrica de qualidade ou tempo, ou uma combinação delas, selecionada pelo usuário. O valor dessa métrica, junto com o tempo de execução da função de segmentação, é então retroalimentado para o sistema de Ajuste Automático de Parâmetros que irá então avaliar qual é a próxima combinação de parâmetros que deverá ser testada. Esse processo se repete até que o número de testes máximo de avaliações da função objetivo seja atingido, ou até que um critério de qualidade seja alcançado.

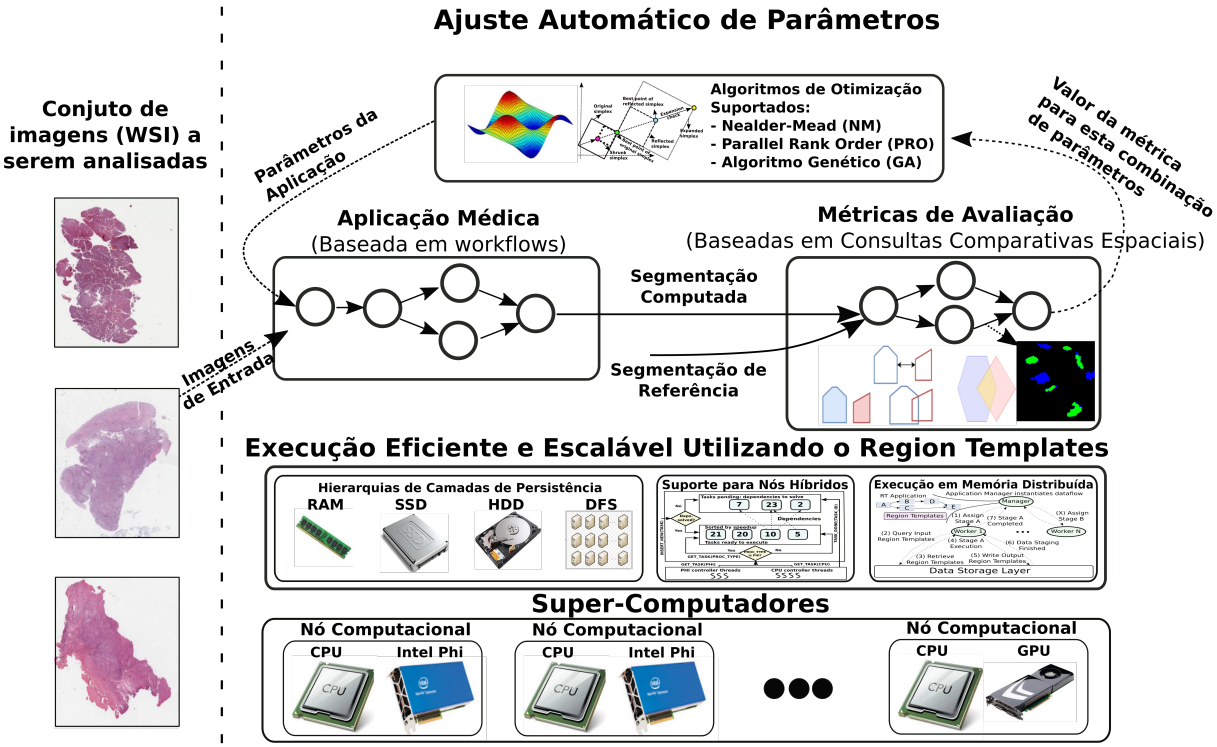


Figura 3.3: Demonstração do fluxo de análise das aplicações exemplo sendo otimizadas pelo módulo de ajuste automático de parâmetros e sendo executadas sobre a plataforma region templates [70].

A avaliação do resultado de uma segmentação, produzida a partir de uma combinação de parâmetros sugerida pelo sistema de Ajuste Automático de Parâmetros, é feita através de uma das 6 métricas suportadas pelo sistema. Uma métrica pode ser, por exemplo, a diferença no número de *pixels* entre a máscara de segmentação gerada pela aplicação em relação à máscara de referência anotada por um especialista. Outro exemplo de métrica suportada é o cálculo da área de intersecção entre os objetos da máscara computada e os objetos da máscara produzida pelo patologista. Quanto maior for essa área de intersecção, melhores são os resultados. Esta métrica, por exemplo, envolve cálculos espaciais e geométricos complexos. O cálculo da métrica é feito no módulo de Análises Comparativas do *Region Templates* que recebe a máscara computada e a compara com a máscara de referência. O valor resultante da comparação é retro-alimentado para o algoritmo de otimização, que computa outro conjunto s de valores de parâmetros para serem avaliados (Figura 3.4). Este processo iterativo continua até que o algoritmo convirja para um determinado critério de parada ou um número máximo de iterações seja alcançado. Mais detalhes sobre as métricas suportadas podem ser encontrados na Seção 3.3.

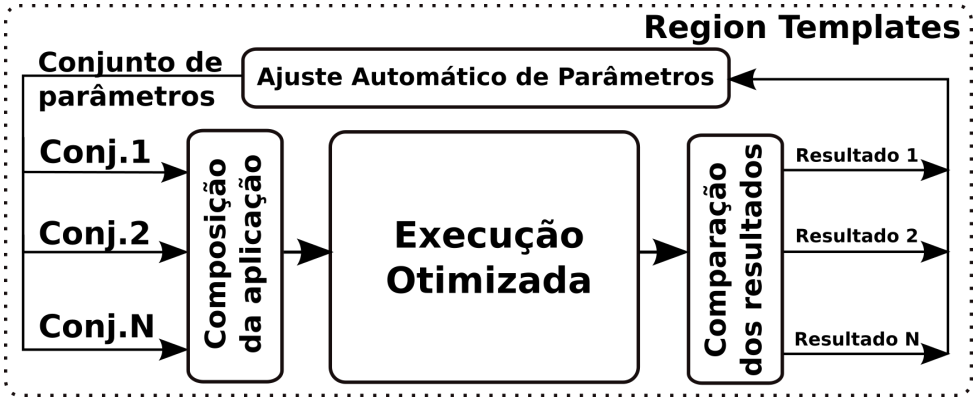


Figura 3.4: Arquitetura do processo automático de ajuste de parâmetros do Region Templates [70]. A aplicação que está sendo otimizada e o sistema de ajuste automático são executados em cima da plataforma Region Templates. Esta plataforma possui diversas otimizações para execução eficiente em sistemas híbridos (GPU e/ou MIC), em sistemas com diversas hierarquias de memória, entre outras otimizações.

3.1.1 Estratégia de Generalização da Otimização

O sistema de Ajuste Automático de Parâmetros avalia a qualidade dos parâmetros sugeridos através da comparação da máscara gerada pela aplicação com uma máscara de referência, que neste caso é uma máscara anotada manualmente por um patologista especialista na área.

No cenário de uso real, raramente o usuário possuirá uma máscara de referência para cada imagem que ele desejará processar. Por isso, é necessário que o sistema de otimização também seja capaz de generalizar as otimizações realizadas em um grupo de imagens para outros conjuntos de imagens.

Neste trabalho é proposto um mecanismo de validação cruzada para avaliar a capacidade de generalização do sistema em encontrar uma combinação de parâmetros que seja capaz de otimizar um conjunto de imagens presentes em um grupo de treinamento e ao mesmo tempo seja capaz de otimizar um conjunto distinto de imagens (grupo de teste).

Em nossos experimentos as imagens foram divididas em conjuntos de treinamento (20% das imagens) e teste (80% restante) a fim de encontrar um determinado conjunto de parâmetros que maximizasse a relação qualidade-tempo selecionada sobre um conjunto de imagens. Mais detalhes a respeito destes experimentos estão no Capítulo 4.

3.2 Algoritmos de Otimização Suportados

Dentre os algoritmos de otimização suportados pelo Region Templates, o GA foi o único implementado inteiramente neste trabalho. O NM e o PRO foram implementados

através de uma biblioteca externa chamada Active Harmony (AH) [64]. Já o Spearmint é o nome da ferramenta que implementa um algoritmo de otimização baseado em uma função probabilística bayesiana.

3.2.1 Método Nelder-Mead (NM)

Este método utiliza um *simplex*, que é um polítopo de $N + 1$ vértices, para percorrer um espaço de busca N-dimensional [53]. Por exemplo, um triângulo sobre um plano bidimensional (*2-simplex*), ou um tetraedro sobre um espaço tridimensional (*3-simplex*) são exemplos de polítopos. Cada vértice do simplex corresponde a um resultado obtido a partir do retorno da chamada da função que está sendo otimizada. O simplex é atualizado a cada iteração substituindo-se o vértice com o pior valor (v_r), por um novo vértice. Essa operação envolve a computação do baricentro c dos restantes vértices simplex para substituir v_r com um ponto na linha $v_r + \alpha(c - v_r)$. Valores típicos para o α são de 0,5; 2; e 3, no entanto eles podem variar dependendo da implementação. α é um coeficiente do método que define se a transformação do simplex é uma reflexão ($\alpha = 2$), uma expansão ($\alpha = 3$), ou uma contracção ($\alpha = 0,5$). Geralmente, o método de Nelder-Mead inicia com $\alpha = 1$ o que significa que ele realiza uma reflexão em primeiro lugar e, dependendo dos resultados, segue a reflexão com uma expansão ou contração. A Tabela 3.1 lista os polítopos de dimensionalidade até 3. Esse algoritmo é inerentemente sequencial e portanto, não é capaz de se aproveitar completamente das infraestruturas paralelas atuais. A fim de solucionar este problema, foi desenvolvido um outro algoritmo de otimização, baseado no NM, capaz de efetuar múltiplos testes concorrentemente. Este algoritmo é o Parallel Ranking Order [76] que será descrito em detalhes na Seção 3.2.2.

Tabela 3.1: Esta tabela lista os polítopos (*simplex*) de dimensionalidade 0 até 3.

Dimensionalidade	**Simplex Regular Correspondente**
0-simplex	Ponto
1-simplex	Segmento de Reta
2-simplex	Triângulo Equilátero
3-simplex	Tetraedro Regular

3.2.2 Método Parallel Rank Order (PRO)

Este método de busca também é baseado em polítopos (*simplex*), similarmente ao algoritmo Nelder-Mead. Como vantagem ele permite a avaliação simultânea de todos os vértices do simplex em cada iteração do algoritmo. Dessa forma, este método, é ideal para uma busca paralela [76].

Este algoritmo utiliza um conjunto de pontos K a partir de um simplex (K $\geq$ N + 1) dado um espaço N-dimensional. A cada iteração do algoritmo calcula-se até $K-1$ novos vértices, que são computados por reflexão, expansão ou contração do simplex em torno de seu vértice com o valor ideal. A reflexão é bem sucedida se pelo menos um dos vértices avaliados conduz a uma melhoria dos resultados de otimização. Se nenhum ponto melhora durante a reflexão, o simplex então contrai-se em torno do melhor vértice. A verificação de expansão ocorre após uma reflexão bem sucedida e é executada para aceitar o novo simplex ou não. Quando os novos pontos são aceitos, o simplex é expandido. O algoritmo para quando ele converge para um ponto pré-definido ou depois de um número limite de iterações ser alcançado. A Figura 3.5 ilustra o processo de busca do algoritmo PRO.

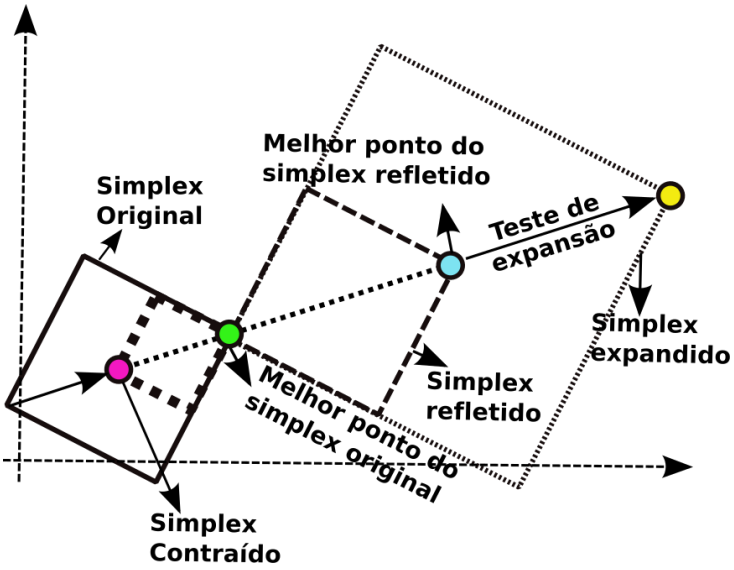


Figura 3.5: Processo de busca do algoritmo Parallel Rank Order (PRO) [76].

3.2.3 Método de Otimização Bayesiana (Spearmint)

Este algoritmo de otimização constrói e explora um modelo probabilístico da função que está sendo otimizada para selecionar pontos no espaço de pesquisa a serem avaliados [62]. A decisão de busca não se baseia apenas em gradientes ou aproximações locais. Ele usa um processo gaussiano para expressar suposições sobre a função que está sendo otimizada, devido à sua flexibilidade e traçabilidade. Ele também usa uma função de utilidade com base neste modelo que lhe permite determinar o próximo ponto a avaliar. Essa função é implementada pela própria ferramenta Spearmint. Isso pode beneficiar a otimização dos parâmetros de funções complexas. No entanto, o custo de computação do próximo conjunto de pontos a serem avaliados pode ser muito elevado com este método. À medida que são executadas iterações da otimização, o Spearmint reajusta os parâmetros

e a forma da função bayesiana de referência para que ela se assemelhe à função que está sendo otimizada.

A filosofia essencial é usar todas as informações disponíveis de avaliações anteriores de f(x) e não simplesmente depender de um gradiente local ou aproximações. Isso resulta em um procedimento que pode encontrar o mínimo de funções não convexas com relativamente poucas avaliações, ao custo de realizar mais computação para determinar o próximo ponto a tentar. O Spearmint é adequado quando as avaliações de f(x) são caras para executar - como é o caso quando se quer treinar um algoritmo de aprendizagem de máquina - pois esse custo de computação adicional que ele leva ao determinar o próximo ponto a ser avaliado é facilmente justificado ao levar em conta o tempo total da otimização [61].

3.2.4 Algoritmo Genético Implementado (GA)

Em linhas gerais, o funcionamento do algoritmo genético implementado é o seguinte: inicialmente são selecionados de maneira aleatória um conjunto de indivíduos para compor a população inicial do algoritmo (Figura 3.6), após isso, é feito o cálculo em paralelo da função de segmentação para cada indivíduo da população. Em seguida, na etapa *Cruzamento* é feito o cruzamento entre os indivíduos da população para gerar os indivíduos da nova população em potencial. Logo em seguida, os indivíduos dessa nova população passam pela etapa de *Mutação* na qual alguns de seus genes podem sofrer ligeiras modificações. Na etapa *Avaliação* estes novos indivíduos passam pela função de segmentação e pelo processo de cálculo de *fitness* (pontuação de um indivíduo utilizando uma métrica pré-determinada para avaliar a segmentação gerada ao utilizar os parâmetros codificados nos genes daquele indivíduo) de cada indivíduo da nova população potencial. Quanto melhor for o valor da *fitness* mais adaptado será o indivíduo. Baseado nos valores da *fitness* escolhe-se os n indivíduos mais adaptados entre os indivíduos dessa nova população potencial e os da população antiga na etapa de *Substituição*. A etapa *Propagação de Membros da Elite* é uma otimização opcional para aumentar a velocidade de convergência do algoritmo. Caso o número máximo de gerações tenha atingido o limite, o algoritmo termina a sua execução retornando o melhor indivíduo. Caso contrário, a população resultante da etapa de *Seleção dos Reprodutores* passa pelas etapas de *Cruzamento*, *Mutação* e *Avaliação* novamente. Uma etapa não pode iniciar antes que a anterior tenha terminado.

A Figura 3.6 demonstra o fluxograma e o funcionamento do algoritmo genético desenvolvido.

Será descrito a seguir cada um dos estágios do algoritmo genético desenvolvido:

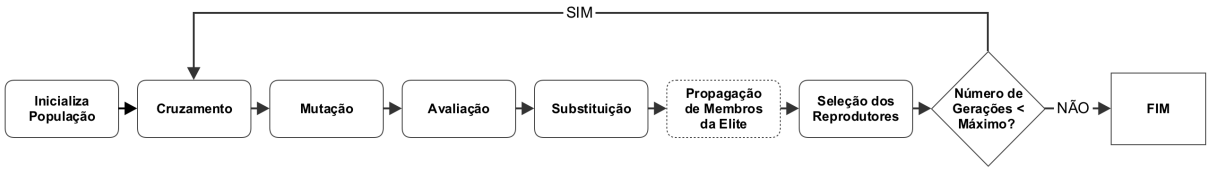


Figura 3.6: Fluxograma do algoritmo genético desenvolvido. O estágio *Propagação de Membros da Elite* é opcional. O tamanho da população do algoritmo, a taxa de *crossover*, a chance de mutação, a quantidade de indivíduos que são parte da elite e o número de gerações que o algoritmo executará são todos parâmetros do algoritmo.

Inicialização da População Para a inicialização dos indivíduos do GA, o mais comum é que a população seja gerada de forma aleatória, mas também é possível "semear"uma solução conhecida que seja uma boa solução para a função que está sendo otimizada. Isto pode ajudar o algoritmo genético a encontrar soluções melhores de maneira mais rápida. Foram realizados testes experimentais para avaliar o impacto dessa otimização no desempenho do GA. "Semeou-se"um indivíduo com os parâmetros padrão das aplicações exemplo, no entanto não houve melhorias em relação à inicialização aleatória dos indivíduos. Dessa forma, a inicialização dos indivíduos foi realizada de maneira aleatória nos outros experimentos conduzidos neste trabalho. Os valores de cada um dos genes de cada indivíduo da população são escolhidos aleatoriamente observando os valores limites permitidos para cada parâmetro.

A codificação dos parâmetros do estágio de segmentação das duas aplicações utilizadas como caso de uso neste trabalho, ficou conforme as Figuras 3.7 e 3.8. As Tabelas 3.2 e 3.3 correspondem à lista de parâmetros do estágio de segmentação dessas aplicações.

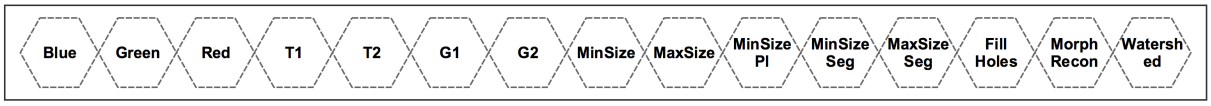


Figura 3.7: Codificação de um indivíduo do GA para a aplicação baseada em Operações Morfológicas. Cada hexágono corresponde à um gene do indivíduo. O indivíduo apresentado nessa imagem possui 15 genes, que correspondem aos valores dos parâmetros da aplicação, vide Tabela 3.2.

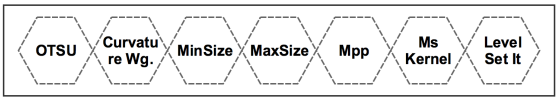


Figura 3.8: Codificação de um indivíduo do GA para a aplicação baseada em *Level Set*, vide Tabela 3.3.

Tabela 3.2: Lista dos 15 parâmetros do estágio de segmentação da aplicação baseada em Operações Morfológicas. O espaço de busca desse estágio é de aproximadamente 21,4 trilhões de pontos.

Parâmetro	Descrição	Escopo da variação dos parâmetros
B/G/R	Cores de detecção para o fundo da imagem	B, G, R $\in [210, 220, ..., 240]$
T1/T2	Limiar de detecção das células vermelhas no sangue	T1,T2 $\in [2.5, 3.0, ..., 7.5]$
G1/G2	Limites para identificar o conjunto inicial de possíveis núcleos	G1 $\in [5, 10, ..., 80]$ G2 $\in [2, 4, ..., 40]$
MinSize	Descarta objetos cuja área (pixels) < seja menor do que MinSize	MinSize $\in [2, 4, ..., 40]$
MaxSize	Descarta objetos cuja área (pixels) > seja maior do que MaxSize	MaxSize $\in [900, 950, ..., 1500]$
MinSizePl	Filtra objetos cuja área seja menor do que MinSizePl	MinSizePl $\in [5, 10, ..., 80]$
MinSizeSeg	Filtra objetos cuja área seja menor do que MinSizeSeg	MinSizeSeg $\in [2, 4, ..., 40]$
MaxSizeSeg	Filtra objetos cuja área seja maior do que MaxSizeSeg	MaxSizeSeg $\in [900, 950, ..., 1500]$
FillHoles Structure	Elemento estrutural que define a região de propagação	FillHoles $\in$ [4-conn, 8-conn]
MorphRecon Structure	Elemento estrutural que define a região de propagação	MorphRecon $\in$ [4-conn, 8-conn]
Watershed Structure	Elemento estrutural que define a região de propagação	Watershed $\in$ [4-conn, 8-conn]

Tabela 3.3: Lista dos 7 parâmetros do estágio de segmentação da segunda aplicação, baseada em *Level Set*. O espaço de busca desse estágio é de aproximadamente 2,8 bilhões de pontos.

Parameter	Description	Range Value
OTSU	Valor de peso atribuído ao limiar OTSU	OTSU $\in [0.3, 0.2, ..., 1.3]$
Curvature Weight	Peso de curvatura das funções *level set*	CW $\in [0.0, 0.05, ..., 1.0]$
MinSize	Tamanho mínimo dos objetos segmentos em micron por dimensão	MinSize $\in [1, 2, ..., 20]$
MaxSize	Tamanho máximo dos objetos segmentos em micron por dimensão	MaxSize $\in [50, 55, ..., 400]$
Mpp	Controla a variabilidade dos resultados	Mpp $\in [0.25]$
MsKernel	Raio espacial do cálculo de *Mean Shift*	MsKernel $\in [5, 6, ..., 30]$
LevetSetIt	Número de iterações da computação *Level Set*	LevetSetIt $\in [5, 6, ..., 150]$

Após definir os genes de cada indivíduo, aplica-se o cálculo em paralelo da função de segmentação para cada membro da população. Após a segmentação, aplica-se uma das métricas de qualidade ou tempo, ou combinação de ambas, suportadas pelo Region Templates para avaliar a *fitness* de cada indivíduo, atribuindo assim um valor numérico para cada indivíduo. As métricas suportadas pelo Region Templates estão descritas na Seção 3.3.2.

Cruzamento Neste trabalho foi aplicado o método de *crossover* de um ponto, de maneira que a população de indivíduos ordenada pelo valor de *fitness* é agrupada em pares, de acordo com a sua posição na ordenação, e ocorre uma escolha aleatória de um gene para cada par de indivíduos. O cruzamento da população ocorre da seguinte maneira: De acordo com uma taxa de *crossover* t, todos os genes cujo índice seja maior que o do gene sorteado são trocados com o do par e todos os genes cujo índice seja menor que o sorteado são mantidos no indivíduo.

Ao final dessa etapa, o algoritmo possui duas populações de indivíduos. A população da geração atual de indivíduos, que eram os genitores dessa etapa, e a nova população

potencial gerada nessa etapa.

Mutação O operador de mutação permite adicionar mais diversidade ao alterar o valor de um gene aleatoriamente com uma pequena probabilidade p, denominada taxa de mutação. Para cada gene de cada indivíduo da nova população potencial gerada pela etapa de cruzamento, calcula-se um número real aleatório r cujo valor esteja entre 0 e 1 e aplica-se o seguinte teste: se r $<$ taxa de mutação, então o operador é aplicado. Senão, aquele gene é preservado.

Caso um parâmetro seja sorteado para sofrer mutação, o novo valor do parâmetro é gerado aleatoriamente e seu valor é modificado. Após a etapa de mutação, a nova população potencial de indivíduos é submetida à etapa de *Avaliação* para que os novos indivíduos sejam submetidos à função de segmentação da imagem e tenham o seu valor de *fitness* calculado.

Avaliação Nesta etapa, as características dos indivíduos mais adaptados (com maior *fitness*) vão ter uma maior probabilidade de se perpetuarem. Para cada indivíduo dessa nova população potencial aplica-se a função de segmentação utilizando-se os novos parâmetros codificados nos genes de cada indivíduo e armazena-se a máscara resultante. Nesta etapa, a máscara produzida a partir dos parâmetros codificados nos genes de cada indivíduo da população é comparada com a máscara base fornecida pelo patologista. O valor de *fitness* atribuído a cada indivíduo pode ser obtido a partir de qualquer uma das métricas suportadas pelo módulo de Análise Comparativa do Region Templates, ou então um valor numérico que seja derivado da combinação de uma métrica de qualidade com uma métrica de tempo de execução. Após o cálculo da *fitness* de cada indivíduo, ordena-se a população em ordem decrescente, uma vez que os primeiros indivíduos (os mais adaptados) deverão ser aqueles que possuem os melhores valores de *fitness*.

Substituição Nesta etapa escolhe-se os indivíduos mais adaptados entre as duas populações (a atual e a população potencial) para compor a população da geração seguinte. O método de substituição utilizado é denominado truncamento [45], ordena-se todos os indivíduos de acordo com a *fitness* e são descatados todos os indivíduos cuja posição na lista ordenada ultrapasse o tamanho de uma população. Denomina-se elite o subconjunto dos indivíduos mais adaptados de cada geração. O tamanho dessa sub-população é um dos parâmetros deste algoritmo. A partir desse estágio, o algoritmo volta a ter uma única população.

Propagação de Membros da Elite (Estágio Opcional) Algoritmos genéticos podem ter uma baixa velocidade de convergência [60]. A fim de contornar esse problema,

desenvolveu-se este estágio (opcional) para aumentar a velocidade de convergência do mesmo. Basicamente, a técnica empregada nesse estágio consiste em substituir um número pré-definido dos piores indivíduos da população por cópias (mínimos locais) dos melhores indivíduos (elite) e colocá-los ao final do ranking junto do restante dos indivíduos menos adaptados [70]. Após ocorrer a substituição, a população passa pela etapas de cruzamento e mutação novamente para que eles produzam novos indivíduos, de forma que as cópias produzirão filhos que serão diferentes dos filhos dos membros dos quais eles foram copiados, iniciando-se assim uma nova geração no algoritmo. Essa medida pode acabar diminuindo a variabilidade genética do algoritmo, e por isso foi colocada como um estágio opcional no algoritmo.

Essa técnica foi modelada como um novo estágio no algoritmo genético (*Propagação de Membros da Elite*), conforme pode ser observado na Figura 3.6. A Figura 3.9 ilustra esse processo de substituição de indivíduos.

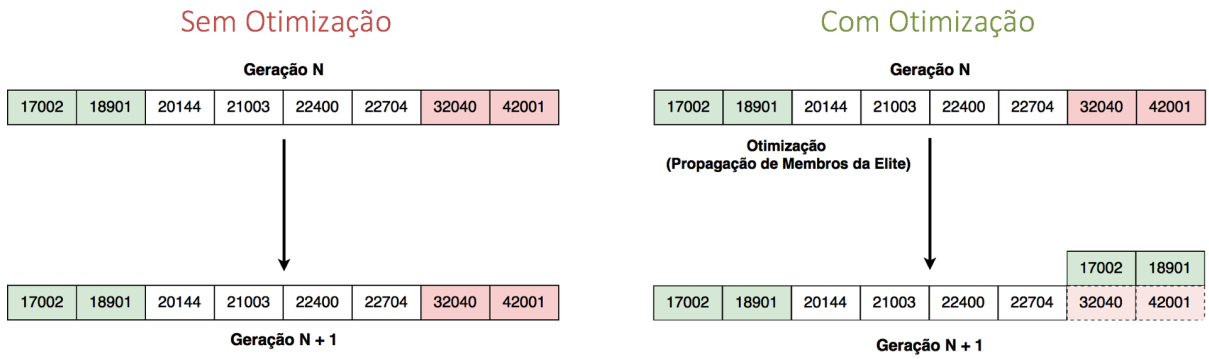


Figura 3.9: Demonstração gráfica do estágio *Propagação de Membros da Elite* em uma população de 8 indivíduos, com uma taxa de elitismo de 25% (2 indivíduos). Cada retângulo corresponde a um indivíduo da população, e o valor contido nele corresponde à *fitness* daquele indivíduo. Neste exemplo, quanto menor o valor dessa métrica, melhor será o resultado.

Seleção dos Reprodutores O método de seleção de indivíduos a serem submetidos à etapa de Cruzamento, foi o seguinte: a população de indivíduos é ordenada pelo valor de *fitness* antes da Etapa de Propagação de Membros da Elite e, depois dela, são agrupados em pares de acordo com a sua posição no *ranking*. Ou seja, todos os membros da população de indivíduos são selecionados para passarem pelo estágio de *Cruzamento*.

O algoritmo terminará a sua execução após essa etapa caso o número máximo de gerações do algoritmo seja alcançado. Caso contrário, o algoritmo volta para a etapa de *Cruzamento*, iniciando-se uma nova geração (iteração) do algoritmo.

Paralelismo O algoritmo genético implementado executa sobre a plataforma de execução distribuída Region Templates. Os operadores de *crossover*, mutação e seleção são todos realizados no nó *Master* do ambiente de execução. A etapa de *Avalição* é executada em paralelo no modelo mestre-escravo [45]. A função *fitness* de cada indivíduo, composta pela segmentação da imagem e pelo cálculo da métrica, é calculada nos nós escravos em paralelo. Após o cálculo da função *fitness*, o nó *Master* recebe os valores e prossegue para a etapa de *Substituição* do algoritmo genético.

Outras formas de se paralelizar um algoritmo genético incluem os modelos em ilha (granularidade grossa) e modelo de vizinhança (granularidade fina) [45]. No modelo em ilha, o algoritmo é constituído de várias sub-populações, que trocam indivíduos ocasionalmente através da operação de migração de indivíduos de acordo com um intervalo e uma taxa de migração pré-definida. Cada uma das subpopulações é atribuída a um nó de processamento diferente, de maneira que cada população é evoluída de forma concorrente. Já no modelo de vizinhança, ou de granularidade fina, evolui-se apenas uma única população. Os indivíduos são organizados logicamente em uma estrutura de *array* (1 dimensional ou n dimensional), de modo que as operações de cruzamento e seleção de um indivíduo se restringem a interagir com os indivíduos vizinhos. Cada indivíduo pode ser alocado para um processador e interagir apenas com a sua vizinhança (também chamada de *deme*). Essa arrumação espacial dos indivíduos em *array* proporciona o uso natural de vizinhanças locais. Vale ressaltar que neste caso é importante que exista algum mecanismo de intersecção entre as várias vizinhanças a fim permitir que a informação genética possa fluir por toda a população.

Outros projetos de pesquisa têm usado algoritmos genéticos para modelar sistemas de ajuste automático de parâmetros (*auto-tuning*), incluindo a otimização de desempenho da pilha de E/S paralela em sistemas distribuídos [6, 75], sistemas de controle de níveis de líquido [65] e sistemas Multi-agentes [7].

3.3 Análises Comparativas

Esta seção descreve o módulo de Análises Comparativas introduzido à plataforma Region Templates. Ele é capaz de calcular diversas métricas e realizar processamentos espaciais sobre as imagens médicas ou outro tipo de fonte de informações espaciais.

Foram adicionadas seis métricas para avaliarem a qualidade dos resultados dos algoritmos de segmentação. Essas métricas são: *Diff Pixels*, *General Dice Coefficient*, *Individual Dice Coefficient*, *Average Dice Coefficient*, *Intersection Overlap Area* e *Jaccard Index*. Essas métricas são utilizadas para comparar e avaliar os resultados dos algoritmos de segmentação através do cálculo de relações entre objetos espaciais (células, veias, etc) obtidos

de duas máscaras distintas. Além disso, essas métricas são utilizadas no nosso sistema de ajuste automático de parâmetros (*auto-tuning*) para iterativamente melhorar a qualidade das máscaras produzidas pelos algoritmos de segmentação e consolidar combinações de parâmetros.

3.3.1 Consultas Espaciais (*Spatial Queries*)

Consultas espaciais são um tipo de consulta para banco de dados que suportam dados espaciais como pontos, linhas e polígonos. Estas consultas diferem das consultas SQL padrão pois lidam com dados espaciais e, como consequência, necessitam de operações diferentes para manipulá-los. Por exemplo, entre as operações suportadas por esse tipo de sistema estão as operações que consideram a relação espacial entre múltiplos dados geométricos, como a área de intersecção entre dois polígonos, o tamanho de um determinado segmento de reta, a distância entre dois pontos, a localização do baricentro de uma determinada forma geométrica, entre várias outras relações. Os tipos mais comuns de consultas espaciais estão exemplificados na Tabela 3.4 e ilustradas na Figura 3.10.

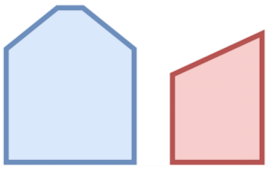
Tabela 3.4: Tipos de consultas espaciais mais comuns suportadas pelos bancos de dados espaciais [2]. O termo geometria se refere a qualquer tipo de forma espacial, por exemplo: um ponto, um segmento de reta ou um polígono.

Tipo de Consulta Espacial (parâmetro(s) de entrada)	**Saída**
Área (geometria a)	Valor numérico da área da geometria
Intersecção (geometria a, geometria b)	Valor numérico da área de intersecção
Distância (geometria a, geometria b)	Menor valor da distância entre as duas geometrias
Contém (geometria a, geometria b)	Verdadeiro se a geometria a contém a geometria b. Falso caso não contenha
Baricentro (geometria a)	Retorna uma geometria b correspondente ao baricentro da geometria a

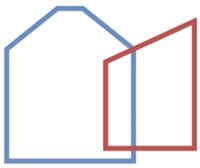
As consultas espaciais foram utilizadas neste trabalho para aprimorar a análise sobre as imagens de tecidos afetados por diversas patologias. Isso ocorre da seguinte forma: uma vez que as estruturas micro-anatômicas (células, núcleos, entre outros) foram extraídas (segmentadas) das imagens em análise, elas são convertidas em polígonos e então processadas pelo módulo de análises comparativas do Region Templates, que é implementado com o auxílio de algumas das funcionalidades de uma biblioteca espacial chamada Hadoop-GIS. Ela auxilia no processamento das relações espaciais entre estruturas micro-anatômicas.

Biblioteca de Análise Espacial: Hadoop-GIS O Hadoop-GIS é uma biblioteca capaz de processar de forma paralela vários tipos de consultas espaciais [1, 3]. O proces-

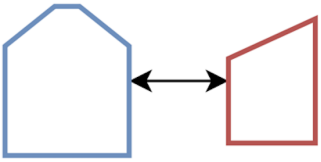
samento espacial realizado dentro do Region Templates é realizado com o auxílio desta biblioteca que possui um módulo de processamento de consultas espaciais em tempo-real chamada RESQUE[1]. Este módulo de processamento é capaz de desempenhar diversos tipos de consultas espaciais, fornecer operadores espaciais e medições geométricas. Para processar os polígonos de forma eficiente, a biblioteca fornece uma indexação espacial dos polígonos através de uma Hilbert R*-Tree [5] e de um módulo de processamento de consulta espacial para processar vários tipos de processamentos geométricos. As R*-Trees são estruturas de dados otimizadas para indexar o acesso à estruturas espaciais como polígonos e pontos.



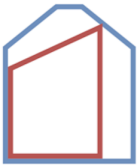
(a) A consulta espacial *Área* mede a área de um determinado polígono.



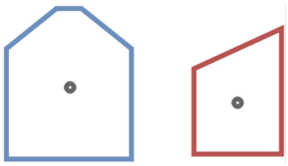
(b) A consulta espacial *intersecção* mede a área da intersecção entre duas geometrias.



(c) A consulta espacial *Distância* mede a distância entre dois polígonos.



(d) A consulta espacial *Contém* retorna verdadeiro caso todos os vértices de um polígono convexo estejam dentro de outro polígono convexo. Caso contrário, retorna falso.



(e) A consulta espacial *Baricentro* calcula o baricentro de uma geometria.

Figura 3.10: Este conjunto de imagens ilustra o funcionamento das consultas espaciais da Tabela 3.4 [3].

As consultas espaciais demandadas pelo sistema de Ajuste Automático de Parâmetros são executadas múltiplas vezes em cada rodada do ajuste automático. Os resultados dos processamentos espaciais realizados pelo módulo de processamento RESQUE são retro-alimentados ao Region Templates. O Hadoop-GIS por si só executa apenas uma consulta espacial por vez. Para solucionar essa limitação, ele foi integrado como uma biblioteca ao Region Templates, para que parte do processamento da consulta espacial fosse realizado pelo módulo de análises comparativas espaciais desta plataforma de execução.

A versão do Hadoop-GIS utilizada na presente dissertação contém todas as funcionalidades necessárias para realizar o processamento espacial de maneira eficiente em um

[1]Realtime Spatial Query Engine (RESQUE) - *Engine* de processamento de consultas espaciais em tempo-real.

único nó, como por exemplo a indexação dos polígonos por meio de estruturas de dados otimizadas (R*-Tree), particionamento dos dados de entrada em conjunto de dados menores e o módulo de processamento de processamento espacial. O paralelismo e a execução eficiente e distribuída de múltiplas consultas simultâneas ficou sob a responsabilidade do Region Templates.

A aplicação que está sendo executada sobre o Region Templates pode utilizar uma das métricas ou consultas suportadas para realizar o processamento espacial. A partir disso, é possível calcular a área da intersecção entre dois conjuntos de objetos (polígonos) ou calcular sua proximidade espacial por exemplo. Depois de receber os resultados da unidade de processamento espacial, o módulo de análises comparativas pode calcular o coeficiente Dice [63], o índice de Jaccard [32] ou algum outro cálculo, a fim de completar a tarefa da consulta.

O fluxo de trabalho é o seguinte (ilustrado pela Figura 3.11): a aplicação executando sobre a plataforma Region Templates chama uma das consultas espaciais providas pela plataforma de execução passando como parâmetro a máscara computada e a máscara de referência. O módulo de análises comparativas do Region Templates interpreta qual é o tipo de consulta espacial a ser realizada: se for uma consulta que não necessita de processamento espacial, o módulo realiza o processamento da métrica e provê o resultado para a aplicação. Caso seja uma consulta que necessite de processamento espacial, ele extrai os objetos das máscaras (núcleos, células e veias) e os convertem em polígonos. A partir daí, essas duas listas de polígonos são indexadas utilizando-se estruturas de dados otimizadas para indexação espacial (R*-Tree), filtra-se os polígonos das máscaras que não possuem relações com os polígonos da outra máscara, e por fim realiza-se o processamento espacial na unidade de processamento RESQUE. Após o processamento espacial, o resultado é retro-alimentado para o Region Templates, que realiza computações adicionais para consolidar o valor da métrica. De posse da métrica calculada, o módulo de análises comparativas do Region Templates fornece o resultado para a aplicação. Diversos tipos de consultas espaciais são suportadas, como por exemplo, a realização da junção espacial entre polígonos e o cálculo da área da união entre eles. A partir dessas consultas foram desenvolvidas as seis métricas discutidas anteriormente.

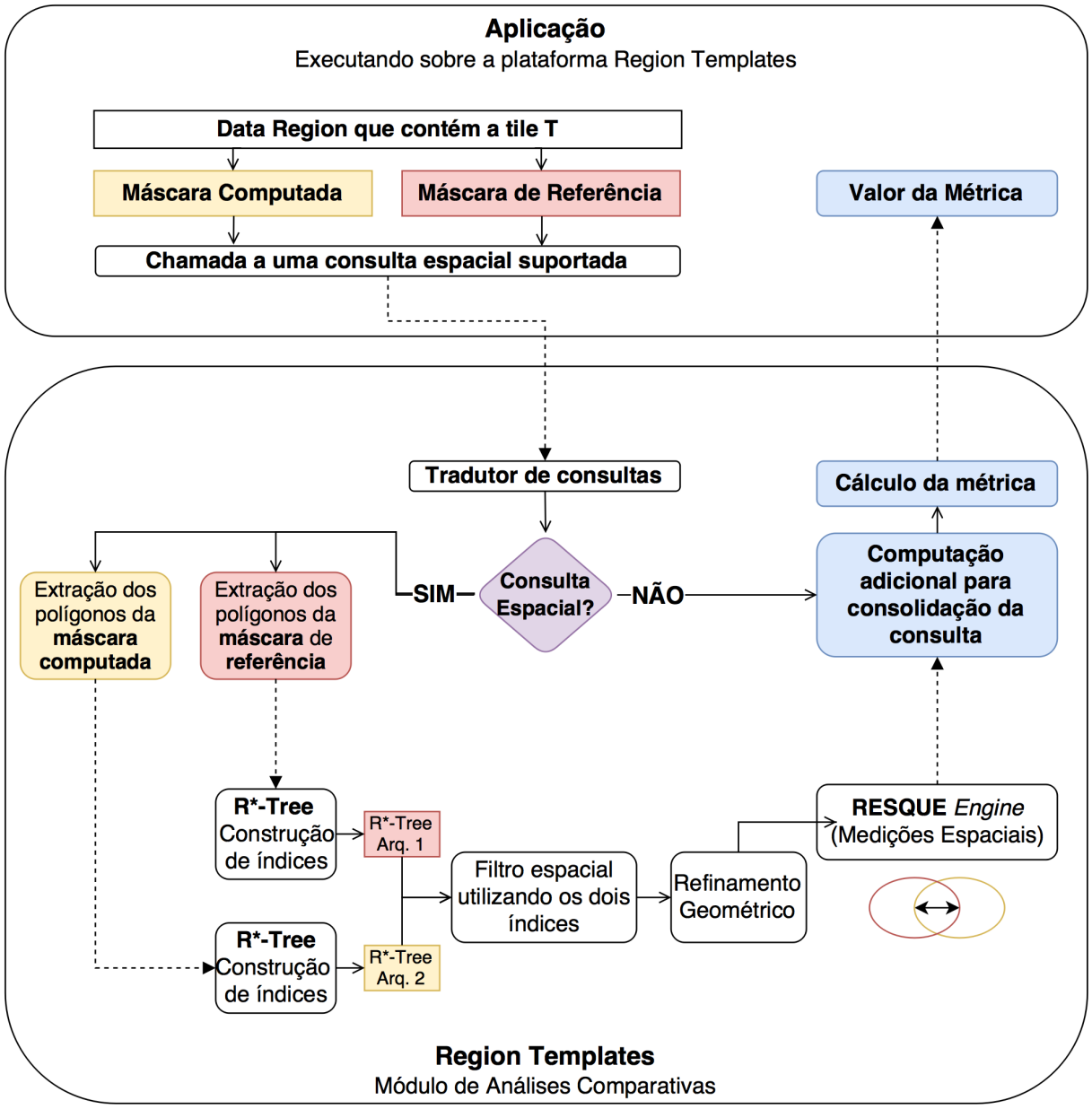


Figura 3.11: Fluxo de trabalho do processo de computação das métricas e consultas espaciais suportadas pela plataforma Region Templates.

3.3.2 Métricas Implementadas

Essa seção explica em detalhes as métricas implementadas neste trabalho e suportadas pelo módulo de análises comparativas da plataforma de execução distribuída Region Templates. Parte dessas métricas são baseados em coeficientes estatísticos que são usualmente utilizados na literatura para comparar a similaridade entre duas amostras [23, 32].

Métrica: *Diff Pixels* A métrica *Diff Pixels* conta a quantidade de *pixels* diferentes, pertencentes ao *foreground*, da máscara produzida pela aplicação em relação à máscara produzida pelo patologista. Isso ocorre porque as máscaras são compostas basicamente de duas cores: branco e preto, sendo que a primeira serve para identificar os objetos encontrados (células e etc) e a segunda para representar o plano fundo. Dentre as seis métricas suportadas, esta é a única que não utiliza a unidade de processamento espacial RESQUE.

Métrica: *General Dice Coefficient* Este coeficiente é uma medida estatística que varia de 0 a 1 e é utilizado para medir a semelhança entre duas amostras. Ela também é conhecida como índice de Sørensen-Dice [63]. Quanto mais próximo de 1 for o valor da métrica, mais semelhantes as amostras são. Esta métrica pode ser calculada através da divisão do dobro da área de intersecção das duas amostras pela soma das respectivas áreas, conforme a Fórmula 3.1.

$$Dice_{gen} = \frac{2|A \cap B|}{|A| + |B|} \tag{3.1}$$

Ou seja, é calculada a área de interseção das imagens, baseada na área de interseção dos polígonos provenientes das máscaras em análise. E após somar todas as áreas das respectivas intercessões, divide-se pelo valor total da soma das áreas dos polígonos de ambos conjuntos de dados.

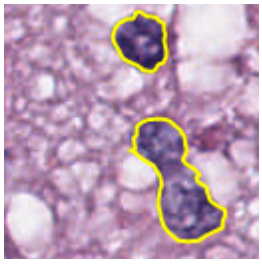
Métrica: *Individual Dice Coefficient* Essa métrica utiliza a mesma Fórmula do coeficiente de Sørensen-Dice [23], no entanto, ao invés de considerar A e B como a área total de interseção das máscaras de segmentação e de referência, considera-se A e B como um par de células encontradas em ambas as máscaras (Fórmula 3.2).

$$Dice_{ind} = \frac{1}{N} \sum_{n=0}^{N} \frac{2|A \cap B|}{|A| + |B|} \tag{3.2}$$

O A é a célula identificada na máscara gerada pela segmentação e o B é a célula equivalente encontrada na máscara de referência. Dessa forma são avaliados o graus de similaridades

de cada par de células (n) individualmente. Após calcular o valor do coeficiente Dice de cada par, calcula-se o valor médio entre todos eles (N = Número total de pares). O valor dessa métrica também varia de 0 a 1. Quanto mais próximo de 1, melhor é o resultado.

Métrica: *Average Dice Coefficient* Essa métrica é a média aritmética das métricas *General Dice Coefficient* e *Individual Dice Coefficient*. Embora ambas as métricas sejam semelhantes, elas contêm algumas desvantagens quando utilizadas de maneira isolada, daí a necessidade de criação desta métrica. A métrica *General Dice Coefficient* não atribui valores diferentes ao comparar uma única célula da máscara de referência (Figura 3.12b) à duas células com metade do tamanho correto e que estejam justapostas na máscara de segmentação (Figura 3.12c), uma vez que ela apenas considera a soma da área total de todos objetos encontrados no processo de segmentação.



(a) Exemplo de imagem de tecido anotada por um patologista.

(b) Exemplo segmentação correta. Duas células identificadas.

(c) Exemplo de segmentação incorreta. Cinco células identificadas, sendo que deveriam ser identificadas apenas duas.

Figura 3.12: Comparação de uma máscara com uma célula segmentada do tamanho correto com uma máscara que possui duas métricas com metade do tamanho e justapostas. [21].

Essa limitação pode acabar beneficiando segmentações que identifiquem mais células do que o correto ao comparar com a máscara de referência. Por outro lado, a métrica *Individual Dice Coefficient* é capaz de diferenciar qual dessas duas situações é mais semelhante à segmentação de referência porque ela considera cada par de células por vez, e, portanto, dá-lhe uma pontuação mais ou menos elevada dependendo do caso. Mesmo que esta métrica seja superior nesta situação, isso não significa que ela seja adequada a todos os casos. Por exemplo, a métrica *Individual Dice Coefficient* considera apenas os pares de células que estejam presentes em ambas segmentações e que se interceptam. Isso significa que se a segmentação gerada pela aplicação contém apenas algumas poucas células da

segmentação de referência, de modo que elas apresentem formato muito semelhante a sua célula correspondente, a métrica irá atribuir um valor alto para a segmentação.

$$Dice_{avg} = \frac{Dice_{general} + Dice_{individual}}{2} \tag{3.3}$$

A métrica *Average Dice Coefficient* (Fórmula 3.3) foi criada para tirar proveito das vantagens de ambas métricas e minimizar as suas limitações. Isso acontece porque quando ocorre a situação em que se demonstra a limitação de uma das métricas, a outra métrica é capaz de avaliar o resultado aplicando uma penalização ou uma pontuação mais elevada dependendo do caso. Essa métrica foi escolhida para ser utilizada na Avaliação Experimental (Capítulo 4) dessa dissertação, uma vez que foi observado que ela reflete de maneira precisa a qualidade de uma segmentação dentre as métricas suportadas.

Métrica: *Intersection Overlap Area* Esta métrica foi desenvolvida para representar a relação percentual entre a área total de intersecção entre as duas amostras dividida pela soma das áreas de todos os polígonos apenas da máscara de referência, de acordo com a Fórmula 3.4.

$$IOA = \frac{|A \cap B|}{|B|} \tag{3.4}$$

Quanto maior for essa porcentagem de intersecção, mais similar é a máscara computada em relação à máscara de referência [70].

Métrica: *Jaccard Index* A métrica *Jaccard Index* também varia de 0 a 1. Ela é calculada ao dividir a área total de intersecção entre duas amostras pela área total da união dessas mesmas amostras [32], conforme a Fórmula 3.5.

$$Jaccard = \frac{|A \cap B|}{|A \cup B|} \tag{3.5}$$

Uma expressão SQL-like similar a da Figura 3.13 poderia ser utilizada em um banco de dados espacial para calcular a métrica *Jaccard Index*.

```
1  SELECT
2   ST_AREA (ST_INTERSECTION (ta.polygon, tb.polygon))/
3   ST_AREA (ST_UNION (ta.polygon, tb.polygon))
4    AS JaccardIndex,
5  FROM markup_polygon ta JOIN markup_polygon tb ON
6   ST_INTERSECTS( ta.polygon, tb.polygon) = TRUE
7  WHERE ta.provenance='MASK1' AND tb.provenance='MASK2';
```

Figura 3.13: Uma consulta SQL-like equivalente à métrica *Jaccard Index* utilizada em banco de dados espaciais.

3.4 Ajuste Multiobjetivo

O Sistema de Ajuste Automático de Parâmetros permite a escolha de um número arbitrário de objetivos para realizar a otimização. Nos casos-de-uso utilizados nesta dissertação, foram escolhidos dois objetivos: a maximização da **precisão do resultado da segmentação**, e a minimização do **tempo de execução da função de segmentação**.

A abordagem para resolução de problemas multiobjetivo utilizada no sistema foi a Escalarização pois é uma abordagem que permite solucionar eficientemente problemas de otimização multiobjetivo com a adaptação de métodos da otimização mono-objetivo. A modelagem do resultado é feita por meio da escalarização do vetor que representa a solução, de modo que os diferentes objetivos são tratados como se fossem um – agregando-os em uma única função escalar.

Para agrupar os objetivos, utilizou-se uma soma ponderada. Para isso, foi necessário primeiro escalonar todos os objetivos para a mesma faixa de valores. A faixa utilizada para escalonar os dois objetivos foi o intervalo de números reais entre 0 e 1. A métrica utilizada nos experimentos deste capítulo foi a *Average Dice Coefficient* que varia de 0 a 1, de modo que quanto mais próximo de 1 for o resultado, melhor será a sua qualidade. Já o tempo de execução ($texec\_medido$) é medido em milissegundos. Como o tempo de execução a princípio não possui um limite superior, foi executado um *profiling* da função de segmentação nas máquinas de teste a fim de se extrair um provável limite máximo ($tslowest$) e mínimo ($tfastest$) para o tempo de execução dessa função. De posse desses valores máximos e mínimos, normalizou-se o tempo de execução medido utilizando-se a Fórmula 3.6.

$$tnormalizado = \frac{tslowest - texec\_medido}{tslowest - tfastest} \tag{3.6}$$

Dessa forma o tempo de execução normalizado ($tnormalizado$) também foi escalonado para a faixa entre 0 e 1. Caso o tempo de execução medido não se encaixe dentro das previsões do menor tempo de execução previsto ($tfastest$) ou acima do maior tempo pre-

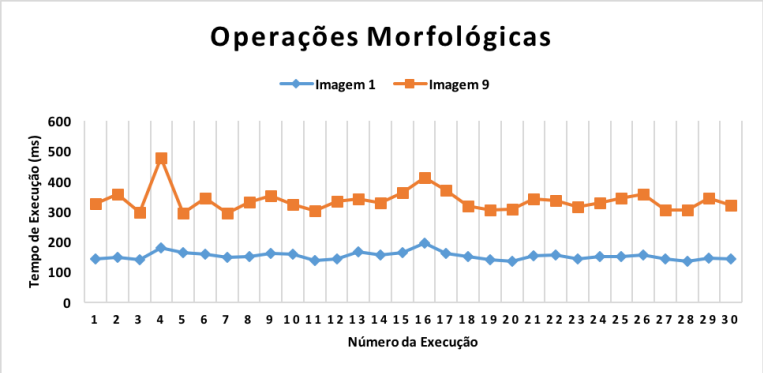
visto (*tslowest*), o seu valor é limitado a 1 ou a 0 respectivamente. Nessa normalização, quanto menor for o tempo de execução da função, mais próximo de 1 será o seu valor.

A função objetivo resultante da combinação desses dois objetivos normalizados (qualidade e tempo de execução) e dos seus respectivos pesos é apresentado na Fórmula 3.7.

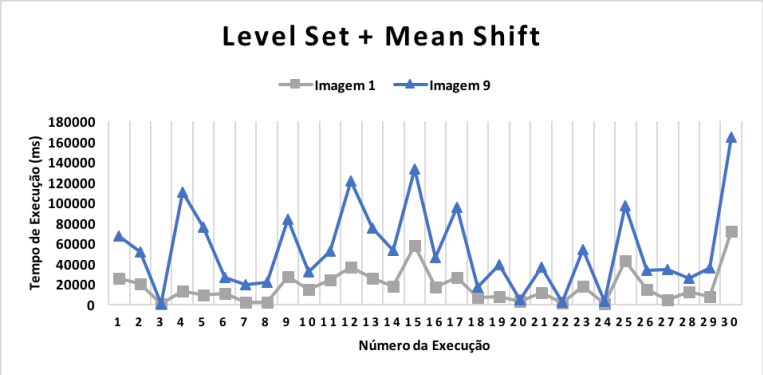
$$f(x) = \sum_{i=1}^{N} w_i * f_i(x) \tag{3.7}$$

Neste caso o valor de N é igual à 2, uma vez que essa otimização multiobjetivo possui apenas dois objetivos. É importante notar que neste caso, ambos os objetivos foram modelados para que valores mais próximos de 1 signifiquem resultados melhores, e portanto, a fim de otimizar essa função escalar é necessário maximizá-la.

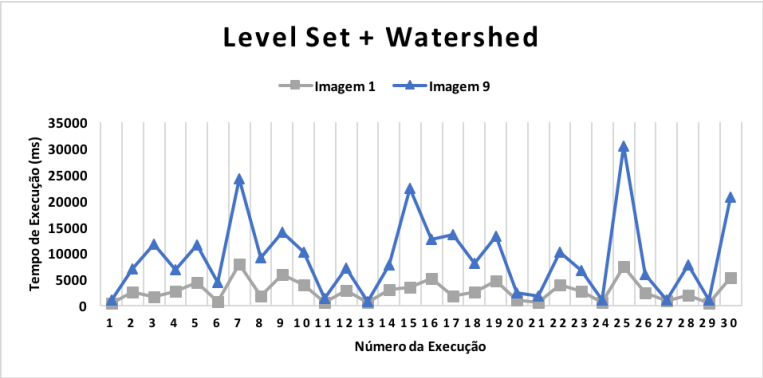
As Tabelas 3.5, 3.6, 3.7 e a Figura 3.14 ilustram o impacto da variação dos parâmetros no tempo de execução da função de segmentação e na qualidade do resultado para as duas aplicações caso-de-uso. A título de exemplo, foram testadas 30 combinações de parâmetros para a aplicação baseada em Operações Morfológicas e para a aplicação baseada em *Level Set*, variando-se os parâmetros de maneira aleatória. Foram utilizadas duas partições de 1K×1K de duas imagens WSI do nosso conjunto de testes (Imagem nº 1 e nº 9). Os resultados apresentados nessas tabelas demonstram que a variação dos parâmetros é capaz de impactar o tempo de execução da aplicação e a precisão do resultado em mais de 160× (Imagem 9 da Tabela 3.6) e 10.25× (Imagem 9 da Tabela 3.7), respectivamente.



(a) Representação gráfica da variação do tempo de execução de 30 execuções aleatórias da aplicação baseada em Operações Morfológicas (dados da Tabela 3.5).



(b) Representação gráfica da variação do tempo de execução de 30 execuções aleatórias da aplicação baseada em Level Set com Mean Shift (dados da Tabela 3.6).



(c) Representação gráfica da variação do tempo de execução de 30 execuções aleatórias da aplicação baseada em Level Set com Watershed (dados da Tabela 3.7).

Figura 3.14: Representações gráficas de 30 execuções das aplicações caso-de-uso, escolhendo-se os parâmetros de maneira aleatória, para duas imagens de 1K×1K (imagens 1 e 9 do conjunto de testes).

Tabela 3.5: Exemplo do impacto que a variação dos parâmetros das aplicações de segmentação causam na precisão do resultado e no tempo de execução da aplicação baseada em Operações Morfológicas. Ao todo foram 30 execuções, escolhendo-se os parâmetros de maneira aleatória, para duas imagens de 1K×1K (imagens 1 e 9 do conjunto de testes).

	Exemplo de Parâmetros da Aplicação Baseada em Operações Morfológicas															**Imagem 1**		**Imagem 9**	
	B	G	R	T1	T2	G1	G2	Min Size	Max Size	Min SizePl	Min SizeSeg	Max SizeSeg	Fill Holes	Morph Recon	Water-shed	**Avg. Dice Médio**	**Tempo de Execução(ms)**	**Avg. Dice Médio**	**Tempo de Execução(ms)**
Conjuntos de Parâmetros	220	230	230	6	7	30	28	4	1200	40	38	1050	8	8	8	0,67	142	0,62	325
	220	240	230	5,5	3	35	34	18	1300	45	2	1500	4	4	8	0,65	148	0,58	357
	230	230	220	5	4	20	28	34	1150	45	34	1300	8	8	4	0,62	141	0,55	296
	220	210	220	3,5	6	35	6	26	1000	40	30	950	4	4	4	0,55	179	0,35	477
	210	240	210	5	5	40	32	28	1100	35	32	1300	4	8	4	0,66	165	0,60	294
	220	210	220	6	4	20	16	6	1450	20	14	950	4	8	4	0,66	159	0,44	345
	230	240	220	7	4,5	55	38	14	950	15	20	1350	8	8	4	0,63	149	0,62	295
	210	240	220	5,5	5,5	50	20	32	1300	15	34	1400	4	4	8	0,71	151	0,58	332
	220	240	220	6	6,5	65	36	16	950	30	12	1100	4	4	8	0,70	162	0,65	352
	220	210	230	6	6	35	6	12	1200	60	28	1500	8	8	4	0,55	159	0,45	323
	210	220	210	5	3	70	38	14	1150	20	30	1050	8	8	4	0,65	138	0,63	303
	220	210	220	3,5	6,5	15	28	26	950	60	32	1150	8	4	4	0,55	142	0,47	335
	230	220	230	7	6,5	60	8	30	1000	55	14	1450	8	8	8	0,63	166	0,55	342
	220	220	240	5,5	5	55	30	32	950	40	34	1250	8	8	8	0,65	155	0,65	328
	210	220	220	6,5	3,5	10	10	22	1500	55	28	1250	8	8	4	0,60	164	0,43	363
	220	230	230	3	4,5	15	10	32	1000	45	26	900	4	4	8	0,55	194	0,34	412
	240	230	240	3,5	4,5	15	6	12	1400	45	22	1350	4	4	4	0,54	160	0,37	370
	220	220	210	3	4,5	45	28	10	1000	35	28	1100	4	8	8	0,68	150	0,62	319
	240	230	220	6,5	7,5	55	22	40	1050	10	30	950	8	8	4	0,66	140	0,62	305
	220	220	220	4,5	4,5	65	40	36	1250	50	2	1300	8	4	4	0,66	134	0,64	307
	230	220	220	2,5	3,5	20	20	28	1050	25	14	1050	4	4	4	0,61	154	0,43	341
	220	220	220	6,5	6,5	60	14	26	1300	50	12	1100	4	4	8	0,66	157	0,56	337
	230	230	210	2,5	6,5	35	26	12	1200	60	32	1450	8	4	4	0,68	144	0,59	316
	240	220	230	4	4	25	38	38	1450	25	24	1050	8	8	4	0,67	152	0,56	329
	220	230	220	5,5	6	10	26	40	1450	50	26	1100	8	8	4	0,63	150	0,54	343
	230	220	240	3	6	20	8	18	1300	35	40	1350	8	8	8	0,59	156	0,39	358
	210	210	220	4	6	70	32	16	1300	80	12	1150	4	8	8	0,65	142	0,64	304
	230	220	220	6	5	15	32	38	1450	15	12	1400	8	8	4	0,64	136	0,51	305
	220	240	230	5,5	3,5	20	34	22	1250	25	24	950	8	8	8	0,66	146	0,53	345
	230	230	220	2,5	4,5	65	30	6	1150	75	20	900	8	4	8	0,64	142	0,64	321
															Mínimo	0,54	134	0,34	294
															Máximo	0,71	194	0,65	477
															Variação	1,31×	1,45×	1,91×	1,62×

Tabela 3.6: Exemplo do impacto que a variação dos parâmetros das aplicações de segmentação causam na precisão do resultado e no tempo de execução da aplicação baseada em *Level Set com Mean Shift*. Ao todo foram 30 execuções, escolhendo-se os parâmetros de maneira aleatória, para duas imagens de 1K×1K (imagens 1 e 9 do conjunto de testes).

	Exemplo de Parâmetros da Aplicação Baseada em Level Set com Watershed							**Imagem 1**		**Imagem 9**	
	otsu Ratio	curvature Weight	size Thld	sizeUpper Thld	mpp	mskernel	levelSet Iteration	**Avg. Dice Médio**	**Tempo de Execução(ms)**	**Avg. Dice Médio**	**Tempo de Execução(ms)**
Conjuntos de Parâmetros	2,1	0,1	11	370	0,25	5	117	0,49	25615	0,34	67098
	1,9	0,25	9	260	0,25	5	92	0,40	20580	0,26	51659
	0,2	0,2	3	360	0,25	12	75	0,20	841	0,10	1031
	1,3	0,7	5	180	0,25	25	86	0,57	13013	0,41	110529
	1,3	1	17	180	0,25	14	48	0,35	9330	0,44	75862
	0,8	0,35	2	90	0,25	17	108	0,59	11268	0,82	26548
	0,5	0,5	7	135	0,25	15	23	0,20	2606	0,46	19574
	0,5	0,7	13	300	0,25	25	17	0,13	2630	0,43	21917
	1,8	0,6	13	145	0,25	17	98	0,38	27052	0,42	83890
	1,6	0,8	19	395	0,25	5	19	0,26	15127	0,15	32024
	1,9	0,5	16	390	0,25	8	36	0,39	24215	0,36	52271
	1,8	0,5	7	370	0,25	29	96	0,35	37117	0,36	121474
	1,5	0,75	14	170	0,25	14	93	0,38	25817	0,42	75444
	2,3	0,8	6	265	0,25	10	92	0,38	18300	0,39	53506
	1,9	0,2	14	250	0,25	18	84	0,38	58052	0,33	133364
	1,7	0,65	15	200	0,25	7	55	0,35	16989	0,31	46593
	0,9	0,05	19	380	0,25	22	47	0,48	26634	0,68	95966
	0,4	0,05	18	85	0,25	9	110	0,35	7060	0,51	17291
	1,2	0,35	7	145	0,25	6	128	0,52	7578	0,39	39055
	0,6	0,55	8	370	0,25	9	56	0,23	2710	0,57	4680
	0,8	0,25	14	280	0,25	22	93	0,40	11541	0,82	36623
	0,6	0,95	2	295	0,25	15	146	0,20	1656	0,50	2315
	1,9	0,4	17	395	0,25	10	135	0,41	18085	0,40	53817
	0,3	0,35	6	190	0,25	29	14	0,06	593	0,20	3193
	2,2	0,6	14	390	0,25	19	34	0,38	42847	0,39	97440
	0,9	0,35	11	245	0,25	20	43	0,47	14577	0,82	33728
	1,3	0,95	6	155	0,25	6	46	0,34	4939	0,24	34463
	0,8	0,25	4	195	0,25	16	123	0,60	12663	0,83	25542
	0,7	0,35	18	95	0,25	19	20	0,33	8054	0,77	35780
	2,2	0,1	7	140	0,25	26	124	0,34	72203	0,23	164961
							Mínimo	0,06	593	0,10	1031
							Máximo	0,60	72203	0,83	164961
							Variação	10,00×	121,76×	8,30×	160,00×

Tabela 3.7: Exemplo do impacto que a variação dos parâmetros das aplicações de segmentação causam na precisão do resultado e no tempo de execução da aplicação baseada em *Level Set com Watershed*. Ao todo foram 30 execuções, escolhendo-se os parâmetros de maneira aleatória, para duas imagens de 1K×1K (imagens 1 e 9 do conjunto de testes).

	Exemplo de Parâmetros da Aplicação Baseada em Level Set com Watershed							**Imagem 1**		**Imagem 9**	
	otsu Ratio	curvature Weight	size Thld	sizeUpper Thld	mpp	mskernel	levelSet Iteration	**Avg. Dice Médio**	**Tempo de Execução(ms)**	**Avg. Dice Médio**	**Tempo de Execução(ms)**
Conjuntos de Parâmetros	0,6	0,1	18	205	0,25	16	13	0,26	436	0,64	1126
	1,5	0,6	7	110	0,25	28	32	0,36	2620	0,41	7002
	1,2	0,7	10	360	0,25	26	131	0,41	1710	0,57	11770
	0,6	0	19	195	0,25	13	121	0,42	2793	0,72	6870
	1,8	0,5	15	175	0,25	6	81	0,36	4393	0,43	11510
	1	0,05	18	260	0,25	26	8	0,50	778	0,47	4383
	1,6	0,15	17	290	0,25	29	138	0,36	7916	0,34	24276
	0,8	0	12	65	0,25	13	62	0,56	1863	0,58	9102
	2,3	0,8	9	165	0,25	13	141	0,35	5962	0,44	14025
	2,2	0,2	19	255	0,25	24	45	0,35	3941	0,37	10186
	0,6	0,1	16	340	0,25	23	18	0,29	547	0,67	1354
	1,9	0,35	10	310	0,25	28	17	0,36	2864	0,38	7200
	0,2	0,25	1	390	0,25	29	87	0,18	616	0,08	821
	2	0,75	17	360	0,25	6	40	0,35	3081	0,44	7804
	1,1	0,15	8	130	0,25	26	147	0,62	3428	0,51	22410
	1,6	0,8	2	300	0,25	7	105	0,35	5168	0,43	12597
	1,3	0,05	6	170	0,25	8	37	0,74	1846	0,23	13604
	0,8	0,15	10	225	0,25	8	143	0,48	2534	0,82	8144
	2,3	0,55	5	190	0,25	19	97	0,36	4725	0,42	13310
	0,6	0,45	13	355	0,25	22	90	0,20	1123	0,59	2500
	0,7	0,75	6	320	0,25	18	22	0,27	541	0,71	1744
	1,5	0,65	7	365	0,25	25	63	0,36	3954	0,43	10263
	1,9	0,9	8	305	0,25	19	25	0,36	2717	0,42	6768
	0,4	0,15	16	175	0,25	15	38	0,19	657	0,36	1081
	1,9	0,05	16	65	0,25	7	123	0,36	7529	0,24	30491
	1,5	0,6	19	125	0,25	12	21	0,36	2444	0,43	5873
	0,2	0,05	11	335	0,25	26	84	0,17	854	0,23	1057
	1	0,4	14	240	0,25	16	135	0,36	2008	0,81	7793
	0,4	0,85	3	365	0,25	28	140	0,08	489	0,25	1075
	1,7	0,05	15	155	0,25	22	79	0,36	5375	0,23	20679
							Mínimo	0,08	436	0,08	821
							Máximo	0,74	7916	0,82	30491
							Variação	9,25×	18,16×	10,25×	37,14×

Capítulo 4

Avaliação Experimental

Este capítulo examina a capacidade do sistema de Ajuste Automático de Parâmetros na seleção de valores de parâmetros para as aplicações de bioinformática que maximizem a qualidade da segmentação nuclear e/ou minimizem o tempo de execução das mesmas. A etapa de segmentação é responsável por identificar as células e os seus respectivos núcleos da imagem que está sendo processada. Esse estágio recebe como entrada uma imagem normalizada e retorna uma máscara com as células que foram identificadas naquela imagem. A plataforma de execução Region Templates então calcula a diferença entre a máscara gerada pela aplicação e a máscara anotada manualmente por um patologista especialista na área, referida neste trabalho como máscara de referência. Para realizar esta avaliação experimental, foram utilizadas um conjunto de 15 imagens de tecido de tumor cerebral de Glioblastoma [37].

As avaliações experimentais foram conduzidas no computador *TACC Stampede*[1]. Cada nó tem um par de processadores Intel Xeon E5-2680 e 32GB RAM. Os nós estão interligados através de switches Mellanox FDR Infiniband.

Neste trabalho, foi realizado um ajuste multiobjetivo de dois objetivos. Os dois objetivos de otimização da função de segmentação eram a **qualidade do resultado da segmentação**, medido através da métrica *Average Dice Coefficient* (Capítulo 3), e o **tempo de execução da função**.

Para agrupar os objetivos em uma única função escalar, utilizou-se uma soma ponderada dos objetivos (escalarização) conforme descrito pela Seção 3.4. Durante os experimentos do sistema de ajuste automático, variou-se os pesos atribuídos tanto para a métrica quanto para o tempo de execução. Foram testados 4 pares de pesos para a métrica (*Average Dice Coefficient*) e para o tempo de execução da função de segmentação a fim de se obter a melhor relação de qualidade e desempenho para cada fluxo de trabalho. Os pares de pesos (w) métrica-tempo utilizados em nossos experimentos foram: (1,0), (1,1),

[1]https://portal.xsede.org/tacc-stampede#overview

(2,1) e (4,1). Além disso, foram realizados testes comparativos entre os 4 algoritmos de otimização suportados pela plataforma Region Templates, o NM, o PRO, o Spearmint e o GA, que foi desenvolvido nesta dissertação.

Na escalarização, a soma dos pesos deve ser sempre igual a 1. Portanto, para cada combinação de peso métrica-tempo, o valor do peso é dividido pela soma total dos pesos de cada objetivo. Por exemplo, no caso em que se atribui peso 2 à métrica e peso 1 ao tempo de execução, o peso da métrica na verdade corresponde a 2/3 e o do tempo a 1/3.

Os testes foram executados para os dois casos de uso: fluxo de trabalho baseado em Operações Morfológicas e o fluxo de trabalho baseado em *Level Set* (Seção 2.2).

Ao todo foram realizados 4 experimentos. O primeiro deles, descrito na Seção 4.1, serviu para avaliar o impacto dos parâmetros de configuração do GA na otimização. A partir dos testes realizados nessa seção, escolheu-se a combinação de parâmetros do GA que seria utilizada nos outros experimentos. Os outros três experimentos foram realizados para comparar a eficiência dos 4 algoritmos de otimização suportados pelo Region Templates (GA, NM, PRO e Spearmint).Dessa forma, no segundo experimento (Seção 4.2), variou-se os pesos métrica-tempo na execução de cada aplicação exemplo, a fim de quantificar a potencial melhoria na qualidade da máscara de saída e a melhoria no tempo de execução. No terceiro (Seção 4.3) e quarto experimentos (Seção 4.4), realizou-se uma validação cruzada aleatória por 10 vezes, separando as imagens em conjuntos de treinamento (20% das imagens) e teste (80% restante) a fim de encontrar um determinado conjunto de parâmetros que maximizasse a relação qualidade-tempo selecionada sobre um conjunto de imagens. A validação cruzada serve para avaliar a capacidade de generalização do conjunto de parâmetros sugerido pelo sistema de auto ajuste.

Em todos os experimentos variou-se os valores dos parâmetros no estágio de segmentação, conforme descrito nas Tabelas 2.1 (fluxo de trabalho baseado em Operações Morfológicas) e 2.2 (fluxo de trabalho baseado em *Level Set*).

4.1 Configuração e Parâmetros do GA

O algoritmo genético (GA), implementado neste trabalho e utilizado nos experimentos deste capítulo, foi configurado para executar 10 gerações com 10 indivíduos em cada uma delas. Ele foi configurado com uma taxa de mutação de 30%, com *crossover* de 1 ponto para cada par indivíduos a uma taxa de *crossover* de 50% e uma taxa de propagação de membros da elite de 20% (2 indivíduos) por geração. A seguir, serão apresentadas as justificativas para a escolha desses parâmetros de configuração do GA, incluindo testes comparativos de desempenho das taxas de mutação, de *crossover* e de elitismo. Estes

testes foram realizados utilizando-se o fluxo de trabalho baseado em Operações Morfológicas.

Tamanho da População e Quantidade de Gerações Escolheu-se evoluir uma população de 10 indivíduos por 10 gerações para que o número total de avaliações na função a ser otimizada fosse de exatamente 100 testes. O número de 100 testes foi escolhido ao se analisar os múltiplos algoritmos de otimização suportados pela plataforma Region Templates. Percebeu-se que o número de 100 testes era suficiente para que todos os algoritmos de otimização pudessem otimizar a função de maneira consistente na maioria dos casos. A Figura 4.1 mostra um exemplo de otimização da aplicação baseada em Operações Morfológicas em que a melhor solução foi encontrada no indivíduo nº9 da 7ª geração do algoritmo.

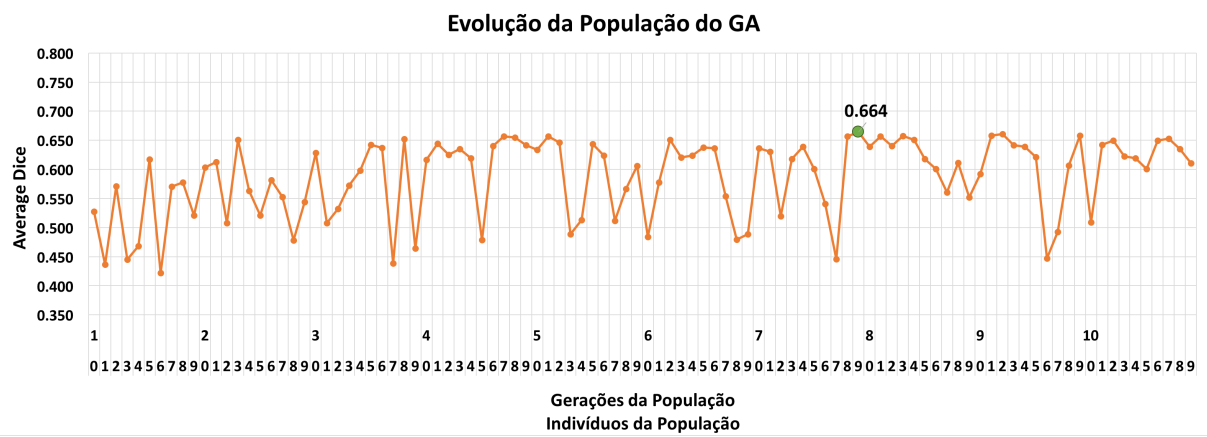


Figura 4.1: Visão geral da função *fitness* (medida pela métrica Average Dice) da população de indivíduos do GA ao longo de uma otimização da aplicação baseada em Operações Morfológicas.

Taxa de Mutação A taxa de mutação do algoritmo foi escolhida empiricamente. Para isso, nós variamos a probabilidade de mutação dos genes de cada indivíduo em valores que variaram de 10% a 50% de chance de ocorrer uma mutação em cada gene de um indivíduo. Os testes foram repetidos 10 vezes. A taxa de *crossover* foi fixada em 50%, a população e o número de gerações em 10, e a taxa de elitismo em 20% (2 indivíduos). Os resultados obtidos estão na Figura 4.2.

A partir dos resultados apresentados na Figura 4.2, a taxa de mutação escolhida para o restante dos testes foi de 30% porque foi a que apresentou o maior valor médio.

Taxa de *Crossover* Assim como a taxa de mutação, a taxa de *crossover* também foi escolhida empiricamente. O operador de *crossover* implementado foi o de um ponto.

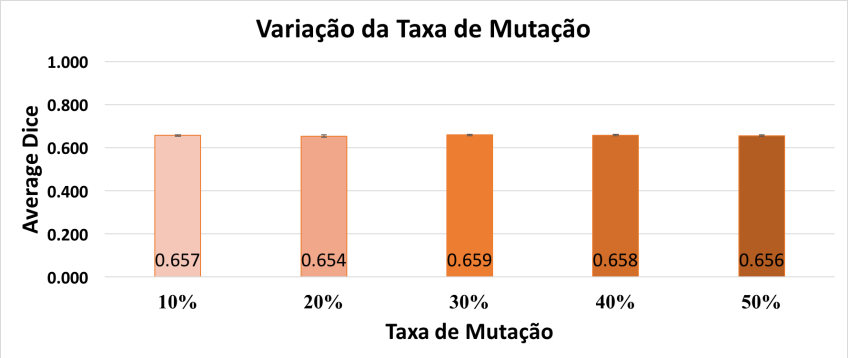


Figura 4.2: Teste de Variação da Taxa de Mutação. As barras de erro correspondem ao desvio padrão do teste.

Variou-se a taxa de *crossover* entre valores de 50% a 90% de chance de se aplicar o operador. A Figura 4.3 mostra os resultados dos testes. Estes testes também foram repetidos 10 vezes. A taxa de mutação foi fixada em 30%, a população e o número de gerações em 10, e a taxa de elitismo em 20% (2 indivíduos). A partir dos resultados obtidos neste teste, escolheu-se utilizar a taxa de *crossover* de 50% para os outros testes apresentados nesta seção porque esta taxa apresentou o melhor desempenho. Os resultados das taxas de 60% e 90% também apresentaram valores bastante semelhantes à taxa escolhida.

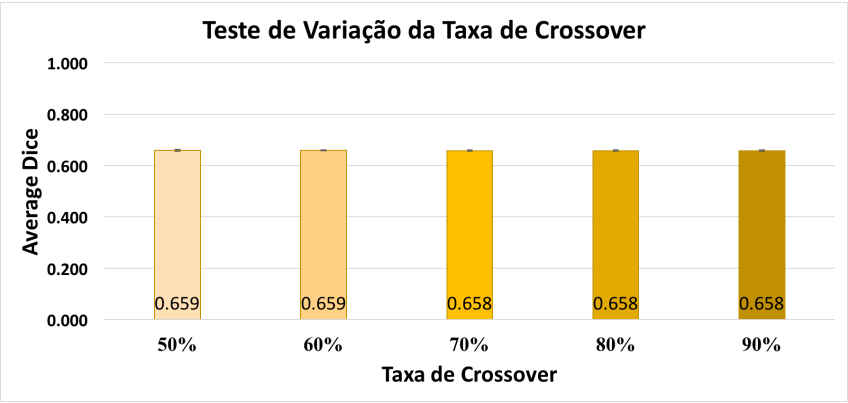


Figura 4.3: Teste de Variação da Taxa de *Crossover*. As barras de erro correspondem ao desvio padrão do teste.

Taxa de Elitismo A etapa Propagação de Membros da Elite é um estágio opcional do algoritmo genético (GA) implementado. Ela foi desenvolvida para aumentar a velocidade de convergência do algoritmo em troca de uma perda controlada na variabilidade genética da população de indivíduos do algoritmo. A otimização consiste em substituir, a cada iteração, um número pré-definido dos piores indivíduos da população por cópias dos melhores indivíduos (elite) da população (Seção 3.2.4).

Foram testadas diversas formas de propagação dos indivíduos. No primeiro caso, não há replicação de membros. Nos outros casos, as formas de propagação consistem na replicação de uma quantidade pré-determinada de membros da elite para substituir os piores membros da população naquela geração. A Figura 4.4 ilustra o teste realizado em uma imagem de tecido cerebral humano. Para este teste foi utilizada a métrica *Diff Pixels* (Seção 3.3.2), ou seja, os valores das medições no teste correspondem à diferença em *pixels* da máscara produzida pela etapa de segmentação da aplicação e a máscara de referência usada para comparação. Para essa métrica, quanto menor for o valor, menor será a diferença em relação à máscara de referência, portanto melhor será o resultado. Utilizou-se nesse experimento o estágio de segmentação do fluxo de trabalho baseado em Operações Morfológicas (Tabela 2.1). Neste teste em específico, o algoritmo executou exatamente 10 gerações, e as chances de ocorrer uma mutação em um gene de um indivíduo era de 15% e a taxa de *crossover* era de 50%. Para garantir a validade estatística deste teste, cada medição foi repetida 50 vezes. As barras de erro na Figura 4.4 correspondem ao desvio padrão das 50 execuções do procedimento de teste em questão.

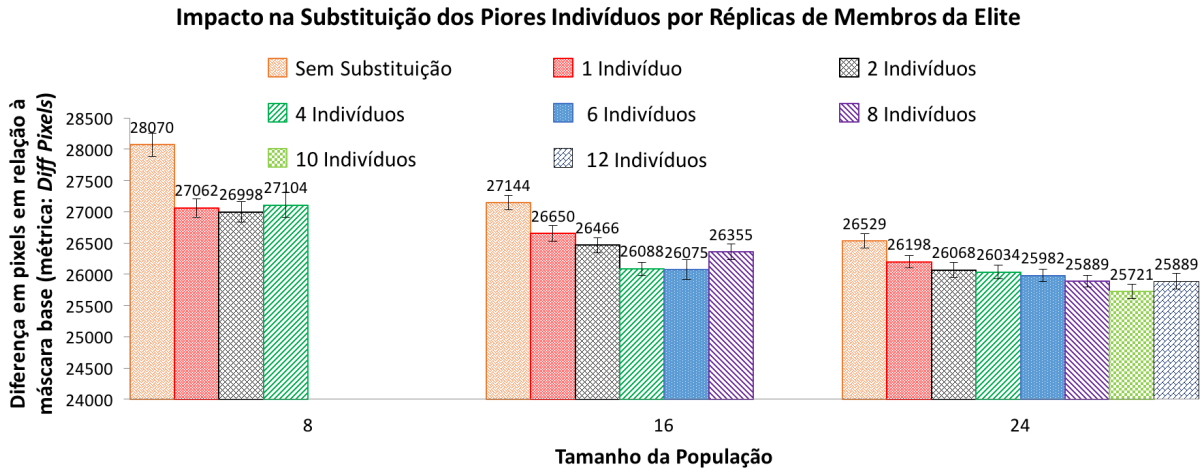


Figura 4.4: Análise comparativa da substituição dos piores indivíduos da população por réplicas de membros da elite na geração seguinte do algoritmo. Quanto menor for a diferença em *pixels* em relação à máscara base, melhor é o resultado. O tamanho da população se refere à quantidade de indivíduos avaliados por geração (iteração) do algoritmo. O eixo das ordenadas representa a diferença em *pixels* da máscara produzida em relação à máscara base.

Este experimento demonstra o impacto na qualidade do resultado proporcionado pelo aumento da população do algoritmo. Ao aumentar o tamanho da população, o algoritmo convergiu para um resultado melhor mais rapidamente. Outro fator evidenciado por este teste é que a quantidade de membros da elite que devem ser replicados ao final de cada geração é uma função do tamanho da população. Por exemplo, quando o tamanho da

população era 8, o melhor valor encontrado ocorreu com a replicação de apenas 2 dos melhores indivíduos a cada geração. Quando o tamanho da população aumentou para 16, os melhores resultados aconteceram ao replicar 4 a 6 dos melhores indivíduos por geração. Um efeito similar ocorreu quando o tamanho da população era 24, ao utilizar 8 a 10 indivíduos para fazer a replicação.

É importante ressaltar que ao substituir indivíduos pouco adaptados por réplicas de indivíduos da elite ocorre uma diminuição na variabilidade genética do algoritmo. Isso pode evitar que soluções que poderiam alcançar valores melhores sejam eliminadas precocemente. Esse efeito foi ilustrado no teste quando o número de indivíduos replicados foi 50% do tamanho da população. Para evitar isso sugere-se que a taxa de replicação, no caso de uso em análise e para esse tamanho de população e número de gerações, seja um valor próximo de 20% a 30% do tamanho da população.

4.2 Otimização de Cada Imagem Individualmente

Para todos os experimentos desta seção e das Seções 4.3 e 4.4 os algoritmos de otimização do sistema de ajuste automático de parâmetros do Region Templates foram configurados para inicializarem de maneira aleatória. O NM, o PRO e o Spearmint foram configurados para convergir com um máximo de 100 iterações. Já o GA foi configurado para evoluir uma população de 10 indivíduos por 10 gerações (conforme a Seção 4.1), com uma taxa de mutação de 30%, com *crossover* de 1 ponto a uma taxa de 50% e uma taxa de propagação de membros da elite de 20% (2 indivíduos) por geração.

Os experimentos desta seção foram conduzidos para os dois fluxos de trabalho de casos de uso: i) Fluxo de trabalho de segmentação baseado em Operações Morfológicas e ii) Fluxo de trabalho de segmentação baseado em *Level Set*.

Os resultados dos algoritmos de otimização estão na métrica *Average Dice*. Repetimos todos os experimentos 10 vezes para medir o desvio padrão e a variabilidade estatística dos resultados. O desvio padrão médio dos resultados é menor que 1% para o fluxo de trabalho de Operações Morfológicas e de 3% para o fluxo de trabalho de segmentação baseada em *Level Set*.

4.2.1 Otimização Mono-Objetivo

O foco desse experimento foi maximizar a qualidade dos resultados sem levar em conta o tempo de execução da segmentação. Os resultados apresentados na Tabela 4.1 mostram que todos os algoritmos de otimização melhoraram a qualidade das máscaras de saída das aplicações de bioinformática. Os resultados otimizados estão sendo comparados com os resultados obtidos ao utilizar os parâmetros padrão dessas aplicações.

Tabela 4.1: Comparação dos resultados utilizando-se os parâmetros padrão das aplicações de segmentação e os selecionados pelos algoritmos de otimização GA, NM, PRO e Spearmint. Os experimentos foram conduzidos usando a métrica *Average Dice* para ambos os fluxos de trabalho de segmentação. Foi dado peso 1 para a métrica e peso 0 para o tempo de segmentação (ajuste de objetivo único)

(a) Resultados da otimização mono-objetivo para o fluxo de trabalho baseado em Operações Morfológicas.

Imagem	Operações Morfológicas						
	Watershed						
	Padrão	GA	NM	PRO	Spearmint	Melhor	**Melhoria**
1	0,65	0,74	0,74	0,74	**0,75**	0,75	1,15×
2	0,59	0,77	0,76	0,76	**0,79**	0,79	1,34×
3	0,61	0,74	0,73	0,74	**0,77**	0,77	1,25×
4	0,79	0,79	0,78	0,78	**0,80**	0,80	1,01×
5	0,65	0,71	0,71	**0,72**	**0,72**	0,72	1,11×
6	0,77	**0,83**	**0,83**	**0,83**	**0,83**	0,83	1,07×
7	0,76	**0,81**	**0,81**	**0,81**	**0,81**	0,81	1,07×
8	0,59	**0,75**	**0,75**	**0,75**	**0,75**	0,75	1,27×
9	0,60	0,67	0,67	0,67	**0,68**	0,68	1,14×
10	0,55	0,77	0,77	0,78	**0,80**	0,80	1,46×
11	0,58	0,70	0,68	0,66	**0,71**	0,71	1,23×
12	0,59	0,76	0,75	0,75	**0,77**	0,77	1,31×
13	0,67	0,82	0,82	0,82	**0,83**	0,83	1,24×
14	0,72	0,82	0,82	0,82	**0,83**	0,83	1,15×
15	0,71	**0,83**	0,82	0,82	**0,83**	0,83	1,16×
Soma	9,84	11,50	11,46	11,45	**11,66**	11,66	1,18×
Média	0,66	0,77	0,76	0,76	**0,78**	0,78	1,18×
Melhoria da Métrica	-	1,17	1,16	1,16	**1,18**	1,18×	-

(b) Resultados para o fluxo de trabalho baseado em *Level Set*.

Img.	Level Set													
	Mean Shift Declumping							Watershed Declumping						
	Padrão	GA	NM	PRO	Spearmint	Melhor	**Melhoria**	*Padrão*	GA	NM	PRO	Spearmint	Melhor	**Melhoria**
1	0,44	**0,80**	0,79	0,70	0,79	0,80	1,83×	0,40	**0,78**	0,71	0,73	0,75	0,78	1,95×
2	0,13	**0,67**	0,44	0,42	0,65	0,67	5,26×	0,11	0,57	0,65	0,47	**0,69**	0,69	6,37×
3	0,09	0,56	0,30	0,29	**0,70**	0,70	7,67×	0,08	0,61	0,57	0,38	**0,63**	0,63	8,35×
4	0,36	**0,83**	0,68	0,71	**0,83**	0,83	2,28×	0,35	**0,82**	0,77	0,72	**0,85**	0,85	2,47×
5	0,23	0,71	0,64	0,61	**0,75**	0,75	3,22×	0,21	0,67	0,69	0,59	**0,70**	0,70	3,30×
6	0,73	0,82	**0,83**	0,82	0,81	0,83	1,13×	0,72	**0,83**	**0,83**	**0,83**	0,82	0,83	1,16×
7	0,71	**0,83**	0,82	0,80	0,81	0,83	1,17×	0,69	**0,82**	**0,82**	0,80	0,81	0,82	1,20×
8	0,76	**0,82**	**0,82**	**0,82**	**0,82**	0,82	1,08×	0,75	**0,82**	**0,82**	**0,82**	0,81	0,82	1,10×
9	0,80	0,83	**0,84**	0,83	0,80	0,84	1,04×	0,83	0,83	**0,84**	0,83	0,80	0,84	1,01×
10	0,85	**0,86**	**0,86**	**0,86**	0,78	0,86	1,01×	0,85	**0,87**	**0,87**	0,87	0,85	0,87	1,02×
11	0,75	**0,78**	**0,78**	0,77	0,70	0,78	1,04×	0,74	**0,78**	0,77	**0,78**	0,72	0,78	1,06×
12	0,82	0,82	**0,83**	**0,83**	0,78	0,83	1,01×	**0,82**	**0,82**	**0,82**	**0,82**	0,79	0,82	1,00×
13	0,88	0,87	**0,89**	0,88	0,82	0,89	1,01×	0,88	**0,89**	**0,89**	0,88	0,87	0,89	1,00×
14	0,83	**0,84**	**0,84**	**0,84**	0,80	0,84	1,01×	**0,84**	0,83	**0,84**	**0,84**	0,81	0,84	1,00×
15	0,86	**0,87**	**0,87**	**0,87**	0,85	0,87	1,01×	0,86	**0,87**	**0,87**	**0,87**	0,84	0,87	1,01×
Soma	9,27	**11,91**	11,24	11,05	11,67	12,14	1,31×	9,12	**11,82**	11,74	11,22	11,73	12,05	1,32×
Média	0,62	**0,79**	0,75	0,74	0,78	0,81	1,31×	0,61	**0,79**	0,78	0,75	0,78	0,80	1,32×
Melhoria	-	**1,28**×	1,21×	1,19×	1,26×	1,31×	-	-	**1,30**×	1,29×	1,23×	1,29×	1,32×	-

No fluxo de trabalho baseado em Operações Morfológicas, os algoritmos de otimização alcançaram resultados semelhantes, porém com vantagem para a abordagem bayesiana (Spearmint). A melhoria média de qualidade após otimizar a aplicação é da ordem de 1,18 vezes maior do que ao utilizar os parâmetros padrão, quando se usa o algoritmo Spearmint (Tabela 4.1a). O GA, o NM e o PRO melhoraram em 1,17×; 1,16× e 1,16× respectivamente.

Já no fluxo de trabalho baseado em *Level Set* (Tabela 4.1b), a melhoria média para o

método de *declumping* Watershed é de 1,30× ao utilizar o algoritmo GA. Também houve melhoria no outro método de *declumping* dessa aplicação, no caso do método baseado em *Mean Shift* o algoritmo que apresentou o melhor desempenho também foi o GA com uma melhoria de 1,28×.

A melhoria individual de uma imagem pode atingir até 8,35× dependendo da imagem utilizada (ex.: imagem 3 da Tabela 4.1b). Diferenças maiores são observadas no ajuste do fluxo de trabalho baseado em *Level Set* em que os algoritmos GA e Spearmint obtiveram os melhores resultados.

4.2.2 Otimização Multiobjetivo

Nesta seção, nós examinamos a capacidade do sistema de ajuste automático na seleção de valores de parâmetros que **maximizem** a métrica de interesse e **minimizem** o tempo de execução da segmentação *ao mesmo tempo*. Estes dois objetivos, na maioria dos casos, conflitam um com o outro. Portanto, foi necessário criar um mecanismo para selecionar o peso desejado para cada objetivo em cada esforço de otimização. Foram selecionados 3 combinações de pesos para a métrica e o tempo a fim de avaliar a capacidade do sistema em ajustar ambos os objetivos ao mesmo tempo. Na Tabela 4.2 são apresentados os resultados da otimização multiobjetivo juntamente com os resultados da otimização mono-objetivo para facilitar a comparação dos resultados.

Tabela 4.2: Comparação dos resultados das segmentação utilizando-se os parâmetros padrão das aplicações exemplo com aqueles selecionados pelos algoritmos de otimização (GA, NM, PRO e Spearmint). Os experimentos foram conduzidos usando a métrica *Average Dice* para ambos os fluxos de trabalho de segmentação. Os valores apresentados correspondem aos valores médios dos resultados das 15 imagens testadas.

	Pesos		Average Dice (0-1)							Speedup				
Aplicação de Segmentação	Métrica	Tempo	*Padrão*	GA	NM	PRO	Spearmint	Melhor	Melhoria	GA	NM	PRO	Spearmint	Melhor
Operações	1	0	0,66	0,77	0,76	0,76	**0,78**	0,78	1,18	-	-	-	-	-
Morfológicas	1	1	0,66	0,70	0,72	**0,73**	0,70	0,73	1,11	1,09	1,08	**1,07**	1,07	1,07
+	2	1	0,66	**0,74**	0,74	0,74	0,72	0,74	1,12	**1,08**	1,07	1,06	1,06	1,08
Watershed	4	1	0,66	**0,75**	**0,75**	0,74	0,75	0,75	1,14	**1,07**	**1,07**	1,05	1,04	1,07
Level Set	1	0	0,62	**0,79**	0,75	0,74	0,78	0,79	1,28	-	-	-	-	-
+	1	1	0,62	0,70	**0,66**	0,63	0,71	0,66	1,06	3,99	**4,91**	5,09	0,67	4,91
Mean Shift	2	1	0,62	0,73	**0,69**	0,68	0,72	0,69	1,12	2,41	**3,74**	2,85	0,61	3,74
Declumping	4	1	0,62	0,74	**0,71**	0,71	0,75	0,71	1,15	1,68	**2,61**	2,38	0,48	2,61
Level Set	1	0	0,61	**0,79**	0,78	0,75	0,78	0,79	1,30	-	-	-	-	-
+	1	1	0,61	0,74	**0,69**	0,69	0,73	0,69	1,13	14,26	**16,05**	13,01	1,38	16,05
Watershed	2	1	0,61	**0,77**	0,73	0,72	0,72	0,77	1,26	**12,68**	13,99	11,85	1,42	12,68
Declumping	4	1	0,61	**0,78**	0,75	0,71	0,76	0,78	1,28	**11,79**	11,61	10,96	1,27	11,79

Os resultados da otimização experimental multiobjetivo para o primeiro fluxo de trabalho, baseado em Operações Morfológicas, mostram que os algoritmos de otimização foram capazes de melhorar a qualidade dos resultados de segmentação em até 1,14× e acelerar a execução em 1,07× em comparação com a qualidade da segmentação e os tempos de execução usando os parâmetros padrão do fluxo de trabalho. Além disso, os resultados

mostram uma consistência na melhoria da qualidade da segmentação à medida que o peso desse componente de otimização é aumentado.

Os resultados para o fluxo de trabalho de segmentação baseado em Level Set e suas duas possíveis estratégias de *declumping* também são apresentados na Tabela 4.2. Conforme apresentado, o GA e o NM obtiveram os melhores valores de otimização agregado (marcado em negrito) quando a qualidade e os tempos de execução são considerados para todas as configurações. Os melhores resultados de qualidade foram obtidos na estratégia de *Watershed Declumping*, atingindo até 1,28× de melhoria, valor muito próximo ao encontrado no ajuste mono-objetivo quando se utilizou os mesmos algoritmos de otimização (1,30×). Nestes resultados, além de melhorar a qualidade da segmentação também foi possível melhorar o tempo de execução da segmentação em 11,79× quando comparado com a segmentação gerada com os parâmetros padrão. Todas as configurações da estratégia baseada em *Mean Shift Declumping* mostram que a qualidade da segmentação da aplicação pôde ser significativamente melhorada (vs. parâmetros padrão) ao mesmo tempo que o tempo de execução da aplicação foi acelerado no intervalo de 2,61×-4,91×. No caso de uso real, essas aplicações processarão centenas a milhares de WSIs, de maneira que essas acelerações implicarão um ganho de tempo nesses estudos na ordem de dias.

O algoritmo PRO também pôde encontrar configurações nas quais os tempos de execução melhoraram significativamente, mas isso veio acompanhado de uma penalidade maior para a métrica de qualidade em comparação com o GA e o NM. O Spearmint, por outro lado, não conseguiu encontrar configurações que reduziriam os tempos de execução em comparação com outros algoritmos. Na verdade, para o *Level Set* com *Mean Shift*, os parâmetros selecionados pela otimização Bayesiana resultaram em uma desaceleração no tempo de execução do aplicativo em comparação com os parâmetros padrão.

Notamos que a abordagem de Otimização Bayesiana é muito adequada para o ajuste mono-objetivo do fluxo de trabalho baseado em Operações Morfológicas, enquanto que o GA e o NM foram os algoritmos de otimização com melhor desempenho na otimização multiobjetivo.

Nós também analisamos as razões pelas quais os ganhos com o fluxo de trabalho baseado em Operações Morfológicas não foram tão impactantes quanto os observados no caso de uso baseado em *Level Set*. Nesta investigação, descobrimos que, na prática, a variação dos parâmetros tem um pequeno impacto no tempo de execução da aplicação baseada em Operações Morfológicas e, como tal, os pequenos ganhos obtidos não foram uma falha dos algoritmos de otimização em si, mas sim uma característica da própria aplicação.

4.3 Validação Cruzada

Esta seção descreve um experimento de validação de subamostragem que aleatoriamente divide o conjunto total de imagens em um conjunto para treinamento e um outro conjunto para validação do modelo. Para cada divisão, as aplicações exemplos são otimizadas utilizando as imagens do grupo de treinamento e a otimização é avaliada usando as imagens do grupo de validação. Esse procedimento foi repetido 10 vezes. Os resultados apresentados são calculados a partir da média dos valores de validação dessas execuções.

A otimização deve selecionar um conjunto de parâmetros que maximize a qualidade da segmentação e minimize o seu tempo de execução. Este é um experimento de validação de subamostragem aleatória em que usamos 20% das imagens (3 imagens), selecionadas aleatoriamente, para treinar os valores dos parâmetros e as 80% imagens (12 imagens) restantes para testar os parâmetros aprendidos.

O algoritmo de otimização utilizado nesta Seção foi o GA implementado nesta dissertação. Ele foi selecionado porque foi um dos algoritmos que mostrou resultados mais consistentes tanto nas otimizações multiobjetivo quanto na mono-objetivo, conforme pode ser observado na Seção 4.2.

O desvio padrão médio nos resultados é menor que 2% para o fluxo de trabalho baseado em Operações Morfológicas e 11% para o fluxo de trabalho de segmentação baseado em *Level Set*. Nós realizamos este experimento para os mesmos 4 pesos métrica-tempo avaliados anteriormente e para ambos os casos de uso. Os resultados são apresentados na Tabela 4.3. O valor da métrica *Average Dice* em cada linha da tabela corresponde à média dos valores das métricas das 10 execuções de cada aplicação de segmentação nas 12 imagens presentes no grupo de validação. Em cada execução, diferentes imagens são selecionadas para serem treinadas e testadas. Por isso, uma linha da tabela não deve ser diretamente comparada à outra.

Para o fluxo de trabalho baseado em Operações Morfológicas, o sistema de ajuste pôde encontrar conjuntos de parâmetros a partir da otimização mono-objetivo das imagens presentes no grupo de teste que melhorassem, em média, a qualidade em mais de 1,11× para as 12 imagens do grupo de validação. Na otimização multiobjetivo para essa mesma aplicação, o sistema foi capaz de encontrar combinações de parâmetros que diminuíssem o tempo de segmentação em até 1,09× mantendo um ganho marginal de qualidade.

Esse padrão do sistema de ajuste automático de parâmetros de ser capaz de otimizar a qualidade e o tempo de execução da aplicação baseada em Operações Morfológicas se repetiu em todos os testes desta seção.

Para o fluxo de trabalho baseado em *Level Set*, o sistema de ajuste não conseguiu encontrar um conjunto de parâmetros que melhorasse a qualidade de segmentação, na média, para todas as imagens de teste. Nem para os esforços mono-objetivo ou multiobjetivo.

Tabela 4.3: Resultados médio das 10 execuções da validação cruzada de subamostragem aleatória. Os valores apresentados correspondem à média da soma das qualidades individuais das imagens presentes em cada conjunto de testes das 10 execuções da validação cruzada.

	Pesos		Average Dice (0-1)		
Aplicação de Segmentação	Métrica	Tempo	*Padrão*	**Otimizado (GA)**	Speedup
Operações Morfológicas + Watershed	1	0	0,65	**0,72**	-
	1	1	0,66	**0,68**	1,09
	2	1	0,65	**0,71**	1,07
	4	1	0,66	**0,73**	1,06
Level Set + Mean Shift Declumping	1	0	**0,63**	0,60	-
	1	1	**0,59**	0,57	3,10
	2	1	**0,62**	0,58	2,32
	4	1	**0,62**	0,60	1,64
Level Set + Watershed Declumping	1	0	0,60	0,60	-
	1	1	**0,60**	0,58	8,97
	2	1	0,59	0,59	10,50
	4	1	**0,62**	0,60	7,37

No caso desta aplicação, o sistema foi sempre capaz de encontrar combinações de parâmetros que otimizassem apenas o tempo de execução da aplicação em detrimento de uma pequena perda na qualidade do resultado. Por exemplo, na otimização que atribui peso 1 para a métrica e peso 1 para o tempo na estratégia de *declumping Watershed* foi possível reduzir, em média, o tempo de execução da aplicação em 8,97× perdendo apenas 3% de qualidade no resultado.

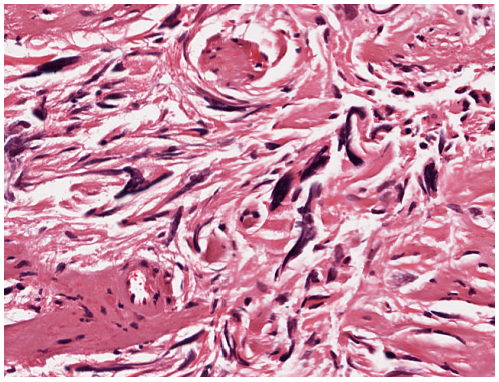
Após inspecionar os resultados, e os possíveis motivos pelos quais o sistema de ajuste automático não conseguiu encontrar uma combinação de parâmetros que melhorasse ao mesmo tempo a qualidade e o tempo de execução do fluxo de trabalho baseado em *Level Set*, nós percebemos que esta aplicação é muito sensível à forma geral das células da imagem de entrada. E por isso, o sistema de ajuste automático não conseguiu encontrar combinações de parâmetros que melhorassem os resultados consistentemente para todas as 12 imagens do grupo de validação simultaneamente.

Para testar o impacto dessa sensibilidade e abordar esta questão, separamos as 15 imagens em dois novos grupos. O primeiro grupo são para imagens que contêm células mais alongadas e esticadas, e o segundo grupo contém as imagens cujas células são mais arredondadas e compactas. Nós classificamos manualmente 5 imagens das 15 para o primeiro grupo e as outras 10 para o segundo grupo. Em seguida, realizamos a experiência de Validação Cruzada para cada grupo separadamente, conforme descrito na Seção 4.4.

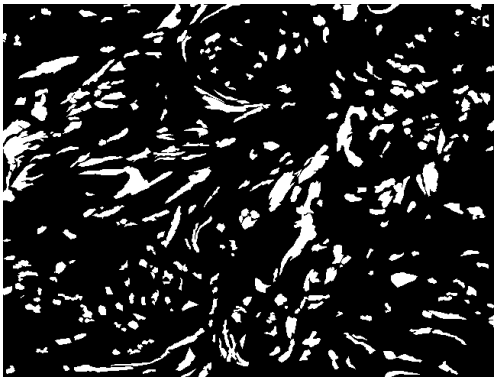
4.4 Validação Cruzada em Dois Grupos

Os parâmetros de segmentação da aplicação *Level Set* são muito sensíveis à forma geral das células encontradas na imagem de entrada. Isso significa que é muito difícil encontrar um único conjunto de parâmetros que produza bons resultados de segmentação

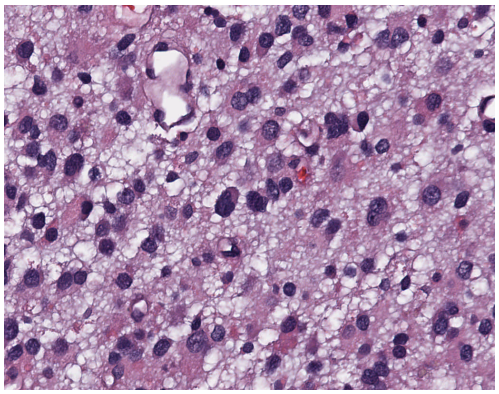
para todos os tipos de imagens de entrada. Com uma inspeção adicional em relação à forma geral das células encontradas nas 15 imagens de entrada que usamos na experiência de validação cruzada da Seção 4.3, notamos que poderíamos separar essas 15 imagens em dois grupos. O primeiro grupo é para imagens que contêm células alongadas (Figura 4.5a), e o segundo grupo contém imagens que possuem células mais arredondadas e compactas (Figura 4.5c).



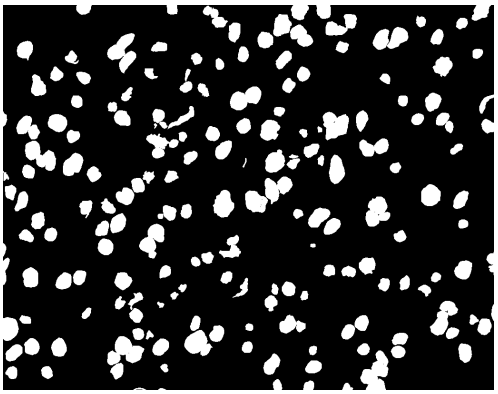
(a) Exemplo de imagem de tecido humano com células mais alongadas atribuída ao primeiro grupo.



(b) Máscara gerada pelas anotações manuais do patologista especialista na área em relação à imagem (a). As partes em branco correspondem às células identificadas.



(c) Exemplo de imagem de tecido humano com células mais arrendondadas atribuída ao segundo grupo.



(d) Máscara anotada manualmente pelo patologista especialista.

Figura 4.5: Exemplos de duas imagens de tecidos humanos com células de características e formas diferentes que foram classificadas em grupos distintos. As máscaras anotadas pelos patologistas são utilizadas como referência pelo algoritmo de otimização para avaliar a qualidade das máscaras produzidas pela aplicação.

Nós classificamos as primeiras 5 imagens para o primeiro grupo e as outras 10 para o segundo grupo. Em seguida, realizamos 10 vezes uma experiência de validação cruzada em cada grupo e somamos o resultado geral do subgrupo de validação de cada grupo para fins de comparação. O algoritmo de otimização neste experimento foi o GA e ele foi configurado da mesma forma que o experimento da Seção 4.3. Para o primeiro grupo, selecionamos aleatoriamente 1 imagem para treinar e as outras 4 imagens para testar. No segundo grupo, usamos 2 imagens para treinar (encontrar o melhor conjunto de parâmetros) e as outras 8 imagens para testar, também de maneira aleatória para cada uma das 10 execuções.

Esta estratégia de classificar as imagens em dois grupos possibilitou uma melhoria na qualidade de segmentação média de até 1,15× e uma melhoria no tempo de execução de até 10,25×.

Na otimização mono-objetivo, o sistema conseguiu encontrar conjuntos de parâmetros para cada estratégia de *declumping* que melhorasse a qualidade média da segmentação conforme mostra a Tabela 4.4.

Tabela 4.4: Resultados médio das 10 execuções da validação cruzada de subamostragem aleatória para o caso de uso baseado em *Level Set*. Os valores apresentados correspondem à média da soma das qualidades individuais das imagens presentes em cada conjunto de testes das 10 execuções da validação cruzada de cada grupo de imagens (4+8 imagens em cada conjunto de teste).

	Pesos		Average Dice (0-1)		
Aplicação de Segmentação	Métrica	Tempo	*Padrão*	**Otimizado (GA)**	Speedup
Level Set + Mean Shift Declumping	1	0	0,61	**0,69**	-
	1	1	0,62	0,62	8,56
	2	1	0,61	**0,62**	1,72
	4	1	0,62	**0,69**	1,37
Level Set + Watershed Declumping	1	0	0,62	**0,70**	-
	1	1	0,61	**0,62**	14,55
	2	1	0,61	**0,66**	13,26
	4	1	0,61	**0,70**	10,25

Para todos os casos da otimização multiobjetivo desta seção o sistema foi capaz de acelerar o tempo de segmentação. A redução no tempo de execução variou de 1,37× a 14,55×.

Em alguns casos, esta aceleração vem a um pequeno custo de qualidade, como no caso do peso 1 para a métrica e peso 1 para o tempo em ambas estratégias de *declumping* da aplicação baseada em *Level Set*. Nas outras combinações de pesos, foi sempre possível melhorar tanto a velocidade quanto a qualidade, especialmente para a estratégia *Watershed Declumping*.

A classificação das imagens em dois grupos resultou em uma melhoria média da qualidade e do tempo de execução quando comparado com a estratégia anterior para a aplicação baseada em *Level Set*. Isso sugere que existem aplicações onde não é possível encontrar

Tabela 4.5: Exemplo de combinações de parâmetros padrão e otimizada, obtida nesses experimentos, para a aplicação baseada em *Level-Set*.

		Exemplo de Parâmetros da Aplicação Baseada em Level Set com Mean Shift								
		otsuRatio	curvatureWeight	sizeThld	sizeUpperThld	mpp	mskernel	levelSetIteration	**Tempo de Execução da Segmentação (ms)**	**Avg. Dice Médio**
Conjuntos de Parâmetros	Padrão	1.0	0.8	3	200	0.25	20	100	854762	0.62
	Otimizado	0.8	0.9	16	170	0.25	11	47	267676	0.72

uma combinação única de parâmetros que seja adequada para todos os tipos de imagens. A Tabela 4.5 exemplifica uma combinação de parâmetros obtida capaz de otimizar as 12 imagens do grupo de validação simultâneamente.

Esses testes demonstram a capacidade de generalização do sistema de ajuste automático de parâmetros em encontrar conjuntos de parâmetros que melhoram tanto a qualidade do resultado quanto reduzem o tempo de execução das aplicações exemplo a partir da otimização de um pequeno número de imagens. Após realizar a otimização no grupo de imagens de treinamento, o sistema é capaz de replicar a otimização da aplicação para outras imagens sem que seja necessário otimizar cada imagem individualmente.

Capítulo 5

Conclusão

Nesta dissertação foi proposta uma solução eficiente e eficaz capaz de otimizar a qualidade do resultado e minimizar o tempo de execução de múltiplos algoritmos de segmentação de duas aplicações médicas utilizadas como caso de uso. Para isso desenvolveu-se um sistema de otimização multiobjetivo com suporte a 4 algoritmos de otimização (GA, NM, PRO e Spearmint) para efetuar o ajuste automático de parâmetros dessas aplicações. Além disso, implementou-se múltiplas métricas e mecanismos de consultas espaciais que são capazes de quantificar as alterações na morfologia das células e tecidos em escala microscópica.

Os casos de uso utilizados neste trabalho foram duas aplicações médicas que realizam a normalização, a segmentação e a comparação de estruturas obtidas a partir de imagens digitais de microscopia em alta resolução. A otimização dessas aplicações é uma tarefa desafiadora uma vez que existem 21,4 trilhões de combinações para configurar uma delas e 2,8 bilhões para configurar a outra.

Para avaliar o desempenho do sistema de ajuste automático de parâmetros foram utilizadas múltiplas métricas. Em quase todos os testes realizados, independentemente da métrica escolhida, o sistema de ajuste automático mostrou melhorias significativas nos resultados quando comparado com a aplicação padrão.

O sistema de otimização multiobjetivo foi integrado à plataforma Region Templates como um módulo adicional denominado Módulo de Ajuste Automático de Parâmetros a fim de otimizar a qualidade e o tempo de execução das aplicações de bioinformática. Dos quatro algoritmos de otimização suportados, apenas o GA foi implementado completamente neste trabalho. Os outros três algoritmos foram integrados ao sistema por meio de bibliotecas, de maneira que foi necessário criar uma interface única para todos os algoritmos de otimização para que eles pudessem ser utilizados por qualquer aplicação que estivesse sendo executada pela plataforma Region Templates. Esse trabalho de criar uma interface única para os algoritmos de otimização permite que novos algoritmos de

otimização sejam adicionados à plataforma de maneira fácil e simples. Para que novos algoritmos sejam adicionados, basta que eles implementem essa interface provida pelo sistema.

Nota-se que os algoritmos genéticos (GA) são uma técnica de otimização complexa com diversas possibilidades de configuração do próprio algoritmo, que influenciam tanto a taxa de convergência ou a velocidade de execução do mesmo. Neste trabalho foi implementada uma técnica de otimização da velocidade de convergência do algoritmo ao substituir os elementos menos adaptados da população por cópias dos melhores indivíduos. Estes indivíduos após serem copiados são submetidos às etapas de recombinação e mutação para garantir que eles produzirão resultados diferentes daqueles membros dos quais eles foram copiados.

Os resultados obtidos com o algoritmo genético demonstram que essa técnica pode produzir bons resultados mesmo quando configurada para executar por poucas iterações. Na maioria dos testes, o GA apresentou desempenho superior ou igual a outros algoritmos (Seção 4.2, 4.3, 4.4) consolidados na literatura, como o NM [53], o PRO [76] e o Spearmint [62].

Ao todo foram testadas quatro combinações de peso métrica-tempo na otimização multiobjetivo. Os algoritmos de otimização necessitaram testar apenas 100 pontos em um espaço de busca de 21,4 trilhões de pontos no primeiro caso de uso e 8,2 bilhões de pontos no segundo, para gerar resultados até 8,35× melhores (Seção 4.2) do que os parâmetros padrão da aplicação. O GA na otimização multiobjetivo por exemplo, foi capaz de melhorar a qualidade média das 15 imagens utilizadas como exemplo em 1,28× e ao mesmo tempo diminuir o tempo de execução da segmentação em 11,79× (Tabela 4.2, *Level Set + Watershed*, peso 4 para a métrica e peso 1 para o tempo).

A fim de avaliar a capacidade de generalização do sistema de otimização em encontrar uma combinação de parâmetros que fosse capaz de otimizar múltiplas imagens a partir do treinamento de um pequeno número de imagens, realizou-se dois experimentos de validação cruzada, uma vez que no caso de uso real, o usuário muito provavelmente não possuirá as máscaras de referência para avaliar a qualidade de uma segmentação produzida pela aplicação. Esta validação cruzada aleatória foi executada 10 vezes, separando as imagens em conjuntos de treinamento (20% das imagens) e teste (80% restante) a fim de encontrar um determinado conjunto de parâmetros que maximizasse a relação qualidade-tempo selecionada sobre um conjunto de imagens. Os resultados do primeiro esforço de validação cruzada mostraram que uma das aplicações era muito sensível ao formato da células presentes nas imagens em análise, e que para solucionar esse problema, foi necessário dividir as imagens em dois grupos, um com imagens que possuíam células mais alongadas e o outro com imagens que possuíam células mais arredondadas. No primeiro

esforço de validação foi possível observar uma melhoria média de até 1,11× na qualidade no caso da aplicação baseada em Operações Morfológicas e 10,50× no tempo de execução da aplicação baseada em *Level Set* ao utilizar os parâmetros sugeridos pelo sistema de otimização no conjunto de imagens teste. No segundo esforço de validação cruzada, ao utilizar dois grupos de imagens, foi possível contornar a sensibilidade da aplicação baseada em operações de *Level Set* em relação à forma das células. Nesse esforço foi possível atingir uma melhoria de 1,15× na qualidade e de redução do tempo de execução médio em até 10,25× (Tabela 4.4). A continuação deste trabalho permitirá realizar uma pré-classificação automática das imagens a fim de identificar se ela deve ser atribuída ao grupo de imagens alongadas ou arredondadas. Diversas estratégias estão sendo avaliadas para realizar isso, entre elas, podemos utilizar combinações de parâmetros sabidamente boas para cada grupo de imagens e aplicar à imagem a fim de avaliar qual combinação de parâmetros produz um resultado melhor.

Entre os trabalhos futuros, objetiva-se adicionar suporte a mecanismos visuais de interface gráfica para facilitar a interação de outros pesquisadores com o sistema de ajuste automático. Através de uma interface web, será possível realizar o envio das imagens a serem processadas, executar e otimizar as aplicações de segmentação nuclear remotamente.

Além disso, planeja-se realizar otimizações no funcionamento do algoritmo genético a fim de inserir nele informações específicas do domínio do problema em questão para aumentar a sua velocidade de convergência. Planeja-se realizar essa otimização por meio da adição de critérios de priorização dos operadores de mutação e *crossover* nos parâmetros das aplicações que mais tiverem influência no resultado da aplicação. Espera-se que, com isso, os parâmetros que mais afetam o resultado das aplicações que estão sendo otimizadas sofram mais modificações a fim de se encontrar soluções melhores mais rapidamente. De acordo com o Teorema da Inexistência de Almoço Grátis (Seção 2.4.4), algoritmos de otimização desenvolvidos especificamente para um problema e que utilizem algum conhecimento prévio a respeito da função que será otimizada, terão um desempenho superior se comparado a outros algoritmos para os quais não foram introduzidos nenhuma espécie de conhecimento prévio do domínio do problema.

Objetiva-se também reduzir a quantidade de parâmetros do ajuste automático por meio da análise de sensibilidade das aplicações à variação dos parâmetros. Para isso, é necessário realizar um estudo prévio do grau de impacto de cada parâmetro da aplicação na precisão do resultado e no tempo de execução da segmentação. Estudos a respeito da sensibilidade das aplicações já foram realizados em trabalhos anteriores do grupo de pesquisa no qual este trabalho está incluído [70]. Ao reduzir a quantidade de parâmetros do ajuste, será possível aumentar a velocidade de convergência dos algoritmos de otimização por exemplo.

Outro trabalho futuro é a paralelização das consultas espaciais utilizando o co-processador Intel® Xeon Phi™. Acredita-se que ao adicionar essa funcionalidade, as consultas espaciais poderão ser executadas de maneira ainda mais eficiente, uma vez que este coprocessador suporta a execução simultânea de até 244 *threads* em seus 61 núcleos de processamento, fornecendo até 1,2 *teraflops* além do suporte a instruções vetoriais [33].

Além disso, planeja-se também aplicar os sistemas desenvolvidos neste trabalho em bancos de imagens de hospitais e outros centros clínicos. Pretende-se expandir esse conjunto de ferramentas para realizar minerações de dados e classificações a partir dessas imagens. Espera-se que este trabalho possa contribuir na investigação de alterações na morfologia de células em escala microscópica, na quantificação dos efeitos dessas alterações e no cálculo de correlações com dados moleculares e outros resultados clínicos a fim de levar a uma melhor compreensão dos mecanismos de doenças graves, como o câncer por exemplo, e permitir o desenvolvimento de novas formas de tratamento.

Referências

[1] Ablimit Aji, George Teodoro, and Fusheng Wang. Haggis: turbocharge a MapReduce based spatial data warehousing system with GPU engine. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, pages 15–20, 2014. 47

[2] Ablimit Aji, Fusheng Wang, and Joel H Saltz. Towards building a high performance spatial query system for large scale medical imaging data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 309–318. ACM, 2012. 47

[3] Ablimit Aji, Fusheng Wang, Hoang Vo, Rubao Lee, Qiaoling Liu, Xiaodong Zhang, and Joel Saltz. Hadoop gis: a high performance spatial data warehousing system over mapreduce. *Proceedings of the VLDB Endowment*, 6(11):1009–1020, 2013. xi, 47, 48

[4] Yousef Al-Kofahi, Wiem Lassoued, William Lee, and Badrinath Roysam. Improved automatic detection and segmentation of cell nuclei in histopathology images. *IEEE Transactions on Biomedical Engineering*, 57(4):841–852, 2010. 11

[5] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. *The R*-tree: an efficient and robust access method for points and rectangles*, volume 19. ACM, 1990. 48

[6] Babak Behzad, Huong Vu Thanh Luu, Joseph Huchette, Surendra Byna, Ruth Aydt, Quincey Koziol, Marc Snir, et al. Taming parallel i/o complexity with auto-tuning. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, page 68. ACM, 2013. 46

[7] Cássio G C Coelho, Carolina G Abreu, Rafael M Ramos, Aldo H D Mendes, George Teodoro, and Célia G Ralha. MASE-BDI: agent-based simulator for environmental land change with efficient and parallel auto-tuning. *Applied Intelligence*, 45(3):904–922, 2016. 46

[8] Antonio Hugo Jose Froes Marques Campos, Dirce Maria Carraro, and Fernando Augusto Soares. Tumor banking for health research in brazil and latin america: time to leave the cradle. *Applied Cancer Research*, 37(1):6, 2017. 9

[9] Antonio Hugo Jose Froes Marques Campos, Andre Abreu Silva, Louise Danielle De Carvalho Mota, Eloisa Ribeiro Olivieri, Vera Cristina Prescinoti, Diogo Patrao,

Luiz Paulo Camargo, Helena Brentani, Dirce Maria Carraro, Ricardo Renzo Brentani, et al. The value of a tumor bank in the development of cancer research in brazil: 13 years of experience at the ac camargo hospital. *Biopreservation and biobanking*, 10(2):168–173, 2012. 9

[10] Chialin Chang, Bongki Moon, Anurag Acharya, Carter Shock, Alan Sussman, and Joel Saltz. Titan: a high-performance remote-sensing database. In *Data Engineering, 1997. Proceedings. 13th International Conference on*, pages 375–384. IEEE, 1997. 15

[11] Luís Pedro Coelho, Aabid Shariff, and Robert F Murphy. Nuclear segmentation in microscope cell images: a hand-segmented dataset and comparison of algorithms. In *Biomedical Imaging: From Nano to Macro, 2009. ISBI'09. IEEE International Symposium on*, pages 518–521. IEEE, 2009. 2, 7

[12] CA Coello Coello. Evolutionary multi-objective optimization: a historical view of the field. *IEEE computational intelligence magazine*, 1(1):28–36, 2006. 20

[13] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007. 22

[14] Jared L Cohon and David H Marks. A review and evaluation of multiobjective programing techniques. *Water Resources Research*, 11(2):208–220, 1975. 22

[15] Lee AD Cooper, Jun Kong, David A Gutman, Fusheng Wang, Jingjing Gao, Christina Appin, Sharath Cholleti, Tony Pan, Ashish Sharma, Lisa Scarpace, et al. Integrated morphologic analysis for the identification and characterization of disease subtypes. *Journal of the American Medical Informatics Association*, 19(2):317–323, 2012. x, 1, 7, 8

[16] Intel Corporation. Intel® many integrated core architecture (intel® mic architecture) – advanced, 2014. 13

[17] Charles Darwin. On the origins of species by means of natural selection. *London: Murray*, page 247, 1859. 25, 27

[18] Richard Dawkins. *The selfish gene*. Number 199. Oxford university press, 2006. 30

[19] Kalyanmoy Deb. Genetic algorithm in search and optimization: the technique and applications. In *Proceedings of International Workshop on Soft Computing and Intelligent Systems,(ISI, Calcutta, India)*, pages 58–87. Proceedings of International Workshop on Soft Computing and Intelligent Systems,(ISI, Calcutta, India), 1998. 4

[20] Kalyanmoy Deb and Kaisa Miettinen. *Multiobjective optimization: interactive and evolutionary approaches*, volume 5252. Springer Science & Business Media, 2008. 20

[21] Stony Brook University Department of Biomedical Informatics, 2015. xi, 4, 12, 52

[22] Santa Di Cataldo, Elisa Ficarra, Andrea Acquaviva, and Enrico Macii. Automated segmentation of tissue images for computerized ihc analysis. *Computer methods and programs in biomedicine*, 100(1):1–15, 2010. 7

[23] Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945. 30, 51

[24] RQ Feitosa, GAOP Costa, TB Cazes, and B Feijo. A genetic approach for the automatic adaptation of segmentation parameters. In *Proceedings of the First International Conference on Object-Based Image Analysis*, volume 4, 2006. 31, 32

[25] Yi Gao, Vadim Ratner, Liangjia Zhu, Tammy Diprima, Tahsin Kurc, Allen Tannenbaum, and Joel Saltz. Hierarchical nucleus segmentation in digital pathology images. *Proc. SPIE*, 9791:979117–979117–6, 2016. 9

[26] David E Goldberg and John H Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988. 24, 25, 27

[27] Jeremias M Gomes, George Teodoro, Alba de Melo, Jun Kong, Tahsin Kurc, and Joel H Saltz. Efficient irregular wavefront propagation algorithms on Intel (r) Xeon Phi (tm). In *Computer Architecture and High Performance Computing (SBAC-PAD), 2015 27th International Symposium on*, pages 25–32, 2015. 15

[28] Antonio Augusto Gonçalves, Claudio Pitassi, and Valter Moreno de Assis Jr. The case of inca´ s national tumor bank management system in brazil. *JISTEM-Journal of Information Systems and Technology Management*, 11(3):549–568, 2014. 9

[29] Vincent Granville, Mirko Křivánek, and Jean-Paul Rasson. Simulated annealing: A proof of convergence. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(6):652–656, 1994. 30

[30] Christian Held, Tim Nattkemper, Ralf Palmisano, and Thomas Wittenberg. Approaches to automatic parameter fitting in a microscopy image segmentation pipeline: An exploratory parameter space analysis. *Journal of pathology informatics*, 4(Suppl), 2013. 31, 32

[31] Christian Hans Held. Towards increased efficiency and automation in fluorescence micrograph analysis based on hand-labeled data. 2013. 31

[32] Paul Jaccard. *Etude comparative de la distribution florale dans une portion des Alpes et du Jura*. Impr. Corbaz, 1901. 31, 49, 51, 53

[33] James Jeffers and James Reinders. *Intel Xeon Phi coprocessor high performance programming*. Newnes, 2013. 78

[34] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983. 30

[35] Hector Klie, Wolfgang Bangerth, X Gai, Mary F Wheeler, Paul L Stoffa, Mrinal Sen, Manish Parashar, U Catalyurek, J Saltz, and T Kurc. Models, methods and middleware for grid-enabled multiphysics oil reservoir management. *Engineering with Computers*, 22(3-4):349–370, 2006. 15

[36] Jun Kong, Lee Cooper, Carlos Moreno, Fusheng Wang, Tahsin Kurc, Joel Saltz, and Daniel Brat. In silico analysis of nuclei in glioblastoma using large-scale microscopy images improves prediction of treatment response. In *Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE*, pages 87–90. IEEE, 2011. 8

[37] Jun Kong, Lee A. D. Cooper, Fusheng Wang, Jingjing Gao, George Teodoro, Tom Mikkelsen, Matthew J. Schniederjan, Carlos S. Moreno, Joel H. Saltz, and Daniel J. Brat. Machine-based morphologic analysis of glioblastoma using whole-slide pathology images uncovers clinically relevant molecular correlates. *PLoS ONE*, 2013. 1, 2, 9, 60

[38] Jun Kong, Fusheng Wang, George Teodoro, Yanhui Liang, Yangyang Zhu, Carol Tucker-Burden, and Daniel J Brat. Automated cell segmentation with 3D fluorescence microscopy images. In *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*, pages 1212–1215, 2015. 7

[39] Jun Kong, Pengyue Zhang, Yanhui Liang, George Teodoro, Daniel J Brat, and Fusheng Wang. Robust cell segmentation for histological images of Glioblastoma. In *Biomedical Imaging (ISBI), 2016 IEEE 13th International Symposium on*, pages 1041–1045, 2016. 7

[40] Tahsin Kurc, Umit Catalyurek, Xi Zhang, Joel Saltz, Ryan Martino, Mary Wheeler, Małgorzata Peszyńska, Alan Sussman, Christian Hansen, Mrinal Sen, et al. A simulation and data analysis system for large-scale, data-driven oil reservoir simulation studies. *Concurrency and Computation: Practice and Experience*, 17(11):1441–1467, 2005. 15

[41] Tahsin Kurc, Xin Qi, Daihou Wang, Fusheng Wang, George Teodoro, Lee Cooper, Michael Nalisnik, Lin Yang, Joel Saltz, and David J Foran. Scalable analysis of Big pathology image data cohorts using efficient methods and high-performance computing strategies. *BMC bioinformatics*, 16(1):399, 2015. 15

[42] Yanhui Liang, Fusheng Wang, Darren Treanor, Derek Magee, Nick Roberts, George Teodoro, Yangyang Zhu, and Jun Kong. A framework for 3D vessel analysis using whole slide images of liver tissue sections. *International journal of computational biology and drug design*, 9(1-2):102–119, 2016. 7

[43] Yanhui Liang, Fusheng Wang, Darren Treanor, Derek Magee, George Teodoro, Yangyang Zhu, and Jun Kong. A 3d primary vessel reconstruction framework with serial microscopy images. In *Medical image computing and computer-assisted intervention: MICCAI... International Conference on Medical Image Computing and Computer-Assisted Intervention*, volume 9351, page 251, 2015. 2

[44] Yanhui Liang, Fusheng Wang, Darren Treanor, Derek Magee, George Teodoro, Yangyang Zhu, and Jun Kong. Liver whole slide image analysis for 3d vessel reconstruction. In *Biomedical Imaging (ISBI), 2015 IEEE 12th International Symposium on*, pages 182–185, 2015. 2

[45] Ricardo Linden. *Algoritmos genéticos (2a ediçao)*. Brasport, 2008. 23, 44, 46

[46] Diogo C. Lucas and Luis O. Alvares. Algoritmos genéticos: uma introdução. Universidade Federal do Rio Grande do Sul, 2002. 24

[47] John McCall. Genetic algorithms for modelling and optimisation. *Journal of Computational and Applied Mathematics*, 184(1):205–222, 2005. 25, 27

[48] Zbigniew Michalewicz. *Genetic algorithms+ data structures= evolution programs*. Springer Science & Business Media, 2013. 4, 26

[49] Kaisa Miettinen and Marko M Mäkelä. On scalarizing functions in multiobjective optimization. *OR spectrum*, 24(2):193–213, 2002. 23

[50] Muthu Rama Krishnan Mookiah, U Rajendra Acharya, Roshan Joy Martis, Chua Kuang Chua, Choo Min Lim, EYK Ng, and Augustinus Laude. Evolutionary algorithm based classifier parameter tuning for automatic diabetic retinopathy grading: A hybrid feature extraction approach. *Knowledge-based systems*, 39:9–22, 2013. 31

[51] Pablo Moscato et al. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826:1989, 1989. 30

[52] National Human Genome Research Institute National Cancer Institute. TCGA the cancer genome atlas, 2017. 8

[53] J. A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4), 1965. 4, 36, 39, 76

[54] AJ Pretorius, Y Zhou, and RA Ruddle. Visual parameter optimisation for biomedical image processing. *BMC bioinformatics*, 16(Suppl 11):S9, 2015. 11, 31, 32, 34

[55] Blair J Rossetti, Fusheng Wang, Pengyue Zhang, George Teodoro, Daniel J Brat, and Jun Kong. Dynamic registration for gigapixel serial whole slide images. In *Biomedical Imaging (ISBI 2017), 2017 IEEE 14th International Symposium on*, pages 424–428, 2017. 7

[56] Shana Schlottfeldt Santos. Otimização multiobjetivo aplicada ao planejamento sistemático de conservação para espécies de plantas do cerrado brasileiro. 2016. 20, 21, 22, 23, 24

[57] Thomas Schultz and Gordon L Kindlmann. Open-box spectral clustering: applications to medical image analysis. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2100–2108, 2013. 31, 32

[58] Michael Sedlmair, Christoph Heinzl, Stefan Bruckner, Harald Piringer, and Torsten Möller. Visual parameter space analysis: A conceptual framework. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2161–2170, 2014. 31, 32

[59] Carter T Shock, Chialin Chang, Bongki Moon, Anurag Acharya, Larry Davis, Joel Saltz, and Alan Sussman. The design and evaluation of a high-performance earth science database. *Parallel Computing*, 24(1):65–89, 1998. 15

[60] Steven S Skiena. *The algorithm design manual: Text*, volume 1. Springer Science & Business Media, 1998. 44

[61] Jasper Snoek. *Bayesian optimization and semiparametric models with applications to assistive technology*. PhD thesis, Citeseer, 2013. 41

[62] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012. 4, 36, 40, 76

[63] Thorvald Sørensen. {A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons}. *Biol. Skr.*, 5:1–34, 1948. 49, 51

[64] Cristian Ţăpuş, I-Hsin Chung, Jeffrey K Hollingsworth, et al. Active harmony: Towards automated performance tuning. In *Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–11. IEEE Computer Society Press, 2002. 39

[65] TK Teng, JS Shieh, and CS Chen. Genetic algorithms applied in online autotuning pid parameters of a liquid-level control system. *Transactions of the Institute of Measurement and Control*, 25(5):433–450, 2003. 46

[66] George Teodoro, Timothy DR Hartley, Umit Catalyurek, and Renato Ferreira. Runtime optimizations for replicated dataflows on heterogeneous environments. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 13–24, 2010. 18

[67] George Teodoro, Tahsin Kurc, Guilherme Andrade, Jun Kong, Renato Ferreira, and Joel Saltz. Application performance analysis and efficient execution on systems with multi-core CPUs, GPUs and MICs: a case study with microscopy image analysis. *International Journal of High Performance Computing Applications*, page 1094342015594519, 2015. 1, 8

[68] George Teodoro, Tahsin Kurc, Jun Kong, Lee Cooper, and Joel Saltz. Comparative Performance Analysis of Intel (R) Xeon Phi (TM), GPU, and CPU: A Case Study from Microscopy Image Analysis. In *28th IEEE Int. Parallel and Distributed Processing Symposium (IPDPS)*, pages 1063–1072, 2014. 13

[69] George Teodoro, Tahsin M. Kurc, Tony Pan, Lee A.D. Cooper, Jun Kong, Patrick Widener, and Joel H. Saltz. Accelerating Large Scale Image Analyses on Parallel, CPU-GPU Equipped Systems. In *26th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1093–1104, 2012. 8, 18

[70] George Teodoro, Tahsin M Kurç, Luís FR Taveira, Alba CMA Melo, Yi Gao, Jun Kong, and Joel H Saltz. Algorithm sensitivity analysis and parameter tuning for tissue image segmentation pipelines. *Bioinformatics*, page btw749, 2017. xi, 2, 5, 15, 32, 35, 37, 38, 45, 53, 77

[71] George Teodoro, Tony Pan, Tahsin Kurc, Jun Kong, Lee Cooper, Scott Klasky, and Joel Saltz. Region templates: Data representation and management for high-throughput image analysis. *Parallel Computing*, 40(10):589 – 610, 2014. x, 1, 2, 7, 8, 12, 16

[72] George Teodoro, Tony Pan, Tahsin M. Kurc, Jun Kong, Lee A.D. Cooper, Norbert Podhorszki, Scott Klasky, and Joel H. Saltz. High-throughput Analysis of Large Microscopy Image Datasets on CPU-GPU Cluster Platforms. In *27th IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, 2013. 18

[73] George Teodoro, Tony Pan, Tahsin M Kurc, Jun Kong, Lee AD Cooper, and Joel H Saltz. Efficient irregular wavefront propagation algorithms on hybrid CPU–GPU machines. *Parallel computing*, 39(4):189–211, 2013. 15

[74] George Teodoro, Rafael Sachetto, Olcay Sertel, Metin N Gurcan, Wagner Meira, Umit Catalyurek, and Renato Ferreira. Coordinating the use of GPU and CPU for improving performance of compute intensive applications. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–10, 2009. 18

[75] George Teodoro and Alan Sussman. AARTS: low overhead online adaptive auto-tuning. In *Proceedings of the 1st International Workshop on Adaptive Self-Tuning Computing Systems for the Exaflop Era*, pages 1–11, 2011. 46

[76] Ananta Tiwari and Jeffrey K Hollingsworth. Online adaptive code generation and tuning. In *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pages 879–892. IEEE, 2011. xi, 4, 36, 39, 40, 76

[77] Thomas Torsney-Weir, Ahmed Saad, Torsten Moller, Hans-Christian Hege, Britta Weber, Jean-Marc Verbavatz, and Steven Bergner. Tuner: Principled parameter finding for image segmentation algorithms using visual response surface exploration. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1892–1901, 2011. 30, 32

[78] David A Van Veldhuizen and Gary B Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary computation*, 8(2):125–147, 2000. 23

[79] Mitko Veta, Paul J van Diest, Robert Kornegoor, André Huisman, Max A Viergever, and Josien PW Pluim. Automatic nuclei segmentation in h&e stained breast cancer histopathology images. *PloS one*, 8(7):e70221, 2013. 2, 7

[80] Hoang Vo, Jun Kong, Dejun Teng, Yanhui Liang, Ablimit Aji, George Teodoro, and Fusheng Wang. Cloud-based whole slide image analysis using mapreduce. In *VLDB Workshop on Data Management and Analytics for Medicine and Healthcare*, pages 62–77. Springer, Cham, 2016. 18

[81] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997. 23

[82] Pengyue Zhang, Fusheng Wang, George Teodoro, Yanhui Liang, Daniel Brat, and Jun Kong. Automated level set segmentation of histopathologic cells with sparse shape prior support and dynamic occlusion constraint. In *Biomedical Imaging (ISBI 2017), 2017 IEEE 14th International Symposium on*, pages 718–722, 2017. 7

[83] Eckart Zitzler, Lothar Thiele, et al. An evolutionary algorithm for multiobjective optimization: The strength pareto approach, 1998. 23