UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

EDUARDO VIEIRA QUEIROGA

ABORDAGENS META-HEURÍSTICAS PARA CLUSTERIZAÇÃO DE DADOS E SEGMENTAÇÃO DE IMAGENS

JOÃO PESSOA
2017

EDUARDO VIEIRA QUEIROGA

ABORDAGENS META-HEURÍSTICAS PARA CLUSTERIZAÇÃO DE DADOS E SEGMENTAÇÃO DE IMAGENS

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Centro de Informática da Universidade Federal da Paraíba, como requisito parcial para obtenção do grau de Mestre em Informática

Orientador: Prof. Dr. Lucídio dos Anjos Formiga Cabral

Co-orientador: Prof. Dr. Anand Subramanian

JOÃO PESSOA
2017

Q3a Queiroga, Eduardo Vieira.
Abordagens meta-heurísticas para clusterização de dados e segmentação de imagens / Eduardo Vieira Queiroga.- João Pessoa, 2017.
88 f. : il.-

Orientador: Profº. Drº. Lucídio dos Anjos Formiga Cabral.
Coorientador: Profº. Drº. Anand Subramanian,
Dissertação (Mestrado) – UFPB/CI

1. Informática. 2. Otimização - Computação. 3. Meta-heurística. 4. Clusterização Particional. 5. Segmentação - Imagens. I. Título.

UFPB/BC CDU – 004(043)

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de EDUARDO VIEIRA QUEIROGA, candidato ao título de Mestre em Informática na Área de Sistemas de Computação, realizada em 17 de fevereiro de 2017.

Aos dezessete dias do mês de fevereiro, do ano de dois mil e dezessete, às quatorze horas e trinta minutos, no Centro de Informática da Universidade Federal da Paraíba, em Mangabeira, reuniram-se os membros da Banca Examinadora constituída para julgar o Trabalho Final do Sr. Eduardo Vieira Queiroga, vinculado a esta Universidade sob a matrícula nº 2015103456, candidato ao grau de Mestre em Informática, na área de "*Sistemas de Computação*", na linha de pesquisa *"Computação Distribuída"*, do Programa de Pós-Graduação em Informática, da Universidade Federal da Paraíba. A comissão examinadora foi composta pelos professores: Lucídio Dos Anjos Formiga Cabral (PPGI-UFPB), Orientador e Presidente da Banca, Anand Subramanian (PPGI-UFPB), Examinador Interno ao Programa, Clauirton De Albuquerque Siebra (PPGI-UFPB), Examinador interno ao programa, Sandro Marden Torres (UFPB), Examinador Externo ao Programa, e Plácido Rogério Pinheiro (UNIFOR), Examinador Externo à Instituição. Dando início aos trabalhos, o Presidente da Banca cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse a exposição oral do trabalho de dissertação intitulado "Abordagens meta-heurísticas para clusterização de dados e segmentação de imagens". Concluída a exposição, o candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: "***aprovado***". Do ocorrido, eu, Ruy Alberto Pisani Altafim, Vice-Coordenador do Programa de Pós-Graduação em Informática, lavrei a presente ata que vai assinada por mim e pelos membros da banca examinadora. João Pessoa, 17 de fevereiro de 2017.

Prof. Dr. Ruy Alberto Pisani Altafim
Ruy Alberto Pisani Altafim
Vice-Coordenador do Programa de Pós-Graduação em Informática
SIAPE 1971934

Prof. Dr. Lucídio Dos Anjos Formiga Cabral
Orientador (PPGI-UFPB)

Prof. Dr. Anand Subramanian
Examinador interno (PPGI-UFPB)

Prof. Dr. Clauirton De Albuquerque Siebra
Examinador interno (PPGI-UPFB)

Prof. Dr. Sandro Marden Torres
Examinador externo ao programa (UFPB)

Prof. Dr. Plácido Rogério Pinheiro
Examinador externo à instituição (UNIFOR)

Agradecimentos

Agradeço a Deus pela saúde e proteção que me deu para enfrentar este ciclo acadêmico.

A minha família que foi extremamente importante, dando-me sempre o suporte necessário e bastante amor quando precisei. Em especial, a minha mãe Antônia Vieira Queiroga, a meu pai José Edu de Queiroga e a minha irmã Francisca Amanda Vieira Queiroga. Obrigado por tudo!

Agradeço ao meu orientador e amigo Prof. Dr. Lucídio dos Anjos Formiga Cabral pelas oportunidades, ensinamentos e imprescindíveis orientações que me direcionaram para esta área de pesquisa e contribuíram significativamente para minha formação pessoal e profissional.

Agradeço ao meu coorientador e amigo Prof. Dr. Anand Subramanian pelos ensinamentos transmitidos diariamente através de discussões sobre assuntos diversos que foram importantes para o aprimoramento do meu conhecimento sobre o processo de produção científica e formação de pessoas.

A todos os meus amigos de curso, em especial Daniel Miranda, Iron Araújo, Marcílio Lemos, Matheus Cordeiro e Rubens Soares. Aos amigos de laboratório e pesquisa Bia, Yuri, Vitor, Nailson, Carlos Magno, Felipe, Anderson, Thiago Gouveia e Teobaldo, que compartilharam comigo alguns momentos de descontração e muito trabalho. Obrigado a todos, inclusive aqueles que me deram caronas quando precisei (vários desses)!

A CAPES que financiou essa pesquisa com bolsa.

A UFPB, Centro de Informática e PPGI, que forneceram uma estrutura adequada e todo o suporte necessário para o desenvolvimento deste trabalho.

Resumo

Muitos problemas computacionais são considerados difíceis devido à sua natureza combinatória. Para esses problemas, o uso de técnicas de busca exaustiva para resolver instâncias de médio e grande porte torna-se impraticável. Quando modelados como problemas de otimização, alguns problemas de clusterização de dados e segmentação de imagens pertencem à classe $\mathcal{NP}$-Difícil e requerem um tratamento adequado por métodos heurísticos. Clusterização de dados é um vasto conjunto de problemas em reconhecimento de padrões e aprendizado de máquina não-supervisionado, cujo objetivo é encontrar grupos (ou *clusters*) de objetos similares em uma base de dados, utilizando uma medida de similaridade preestabelecida. O problema de clusterização particional consiste em separar completamente os dados em conjuntos disjuntos e não vazios. Para métodos de clusterização baseados em centros de *cluster*, minimizar a soma das distâncias *intracluster* é um dos critérios mais utilizados. Para tratar este problema, é proposta uma abordagem baseada na meta-heurística *Continuous Greedy Randomized Adaptive Search Procedure* (C-GRASP). Resultados de alta qualidade foram obtidos através de experimentos envolvendo o algoritmo proposto e outras meta-heurísticas da literatura. Em visão computacional, segmentação de imagens é o processo de particionar uma imagem em regiões de interesse (conjuntos de pixels) sem que haja sobreposição. Um dos tipos mais simples de segmentação é a *limiarização do histograma* para imagens em nível de cinza. O método de Otsu é um dos mais populares e propõe a busca pelos limiares que maximizam a variância entre os segmentos. Para imagens com grande profundidade de cinza, técnicas de busca exaustiva possuem alto custo computacional, uma vez que o número de soluções possíveis cresce exponencialmente com o aumento no número de limiares. Dessa forma, as meta-heurísticas tem desempenhado um papel importante em encontrar limiares de boa qualidade. Neste trabalho, uma abordagem baseada em *Quantum-behaved Particle Swarm Optimization* (QPSO) foi investigada para limiarização multinível de imagens disponíveis na literatura. Uma busca local baseada em *Variable Neighborhood Descent* (VND) foi proposta para acelerar a convergência da busca pelos limiares. Além disso, uma aplicação específica de segmentação de imagens de microscopia eletrônica para análise microestrutural de materiais cimentícios foi investigada, bem como a utilização de algoritmos em grafos para detecção de trincas e extração de características de interesse.

Palavras-chave: Otimização, Meta-heurísticas, Clusterização particional, Segmentação de imagens.

Abstract

Many computational problems are considered to be hard due to their combinatorial nature. In such cases, the use of exaustive search techniques for solving medium and large size instances becomes unfeasible. Some data clustering and image segmentation problems belong to $\mathcal{NP}$-Hard class, and require an adequate treatment by means of heuristic techniques such as metaheuristics. Data clustering is a set of problems in the fields of pattern recognition and unsupervised machine learning which aims at finding groups (or clusters) of similar objects in a benchmark dataset, using a predetermined measure of similarity. The partitional clustering problem aims at completely separating the data in disjont and non-empty clusters. For center-based clustering methods, the minimal intra-cluster distance criterion is one of the most employed. This work proposes an approach based on the metaheuristic *Continuous Greedy Randomized Adaptive Search Procedure* (C-GRASP). High quality results were obtained through comparative experiments between the proposed method and other metaheuristics from the literature. In the computational vision field, image segmentation is the process of partitioning an image in regions of interest (set of pixels) without allowing overlap. Histogram thresholding is one of the simplest types of segmentation for images in grayscale. Thes Otsu's method is one of the most populars and it proposes the search for the thresholds that maximize the variance between the segments. For images with deep levels of gray, exhaustive search techniques demand a high computational cost, since the number of possible solutions grows exponentially with an increase in the number of thresholds. Therefore, metaheuristics have been playing an important role in finding good quality thresholds. In this work, an approach based on *Quantum-behaved Particle Swarm Optimization* (QPSO) were investigated for multi-level thresholding of available images in the literature. A local search based on *Variable Neighborhood Descent* (VND) was proposed to improve the convergence of the search for the thresholds. An specific application of thresholding for electronic microscopy images for microstructural analysis of cementitious materials was investigated, as well as graph algorithms to crack detection and feature extraction.

Keywords: Optimization, Metaheuristics, Partitional clustering, Image segmentation.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

INTRODUÇÃO

Muitos problemas computacionais são considerados difíceis devido à sua natureza combinatória, sendo inviável a utilização de técnicas de busca exaustiva — métodos que enumeram muitas ou todas as possibilidades de solução — para instâncias de médio e grande porte. *Clusterização de dados* e *segmentação de imagens*, quando modelados como problemas de otimização, são exemplos desse tipo de problema e requerem um tratamento adequado por técnicas avançadas, tal como meta-heurísticas (veja a seção 2.2).

Neste capítulo, serão descritos os problemas tratados, as observações que justificam a pesquisa realizada, bem como os objetivos detalhados e a estrutura do trabalho.

1.1 Clusterização de dados

Em problemas do mundo real que envolvem grandes bases de dados, mecanismos de reconhecimento de padrões podem prover informações úteis para apoiar processos de tomada de decisão. Clusterização de dados ou análise de *cluster* é uma tarefa de classificação não supervisionada, cujo objetivo é encontrar grupos de objetos (ou *clusters*) mais similares dada uma métrica de similaridade preestabelecida. Esse vasto conjunto de técnicas possui aplicações em diversas áreas, incluindo recuperação da informação (RASMUSSEN, 1992; AGGARWAL; ZHAI, 2012; CAI; LI, 2011), análise de imagens (CELENK, 1990; DAS; SIL, 2010; ARBELAEZ et al., 2011) e bioinformática (LI et al., 2001; HRUSCHKA et al., 2006; BRITO et al., 2016).

Entre os tipos existentes, clusterização hierárquica e particional são os mais conhecidos e estudados. No primeiro tipo, uma hierarquia de *clusters* é construída sobre os dados e em cada nível existe uma clusterização distinta. No segundo tipo, busca-se uma separação

completa dos dados em *clusters* disjuntos e não-vazios. Em algumas situações, o número de *clusters* pode ser um requisito do problema, em contraste com clusterização automática, que não possui essa restrição. Formalmente, como definido em (HANSEN; JAUMARD, 1997), para um particionamento $P = \{C_1, C_2, \ldots, C_k\}$ do conjunto de dados D em k *clusters*, temos que:

1. $C_i \neq \emptyset$ para $i = 1, ..., k$
2. $C_i \cap C_j = \emptyset$, para $i, j = 1, ..., k$ e $i \neq j$;
3. $\bigcup_{i=1}^{k} C_i = D$;

Em clusterização baseada em centros (e.g., algoritmo k-means descrito na seção 2.3), um conjunto de pontos virtuais ou centros de *cluster* $K = \{c_1, \ldots, c_k\}$ é utilizado para definir o particionamento resultante, como ilustrado na Figura 1.1. A abordagem comumente utilizada consiste em determinar os centros de *cluster* que minimizam a soma das distâncias *intracluster* (Equação 1.1). Dado m objetos com d atributos (dimensões), a função é computada por somar a distância, dada pela norma Euclidiana (Equação 1.2), entre cada ponto (objeto da conjunto de dados) $p \in D$ e o centro mais próximo em K.

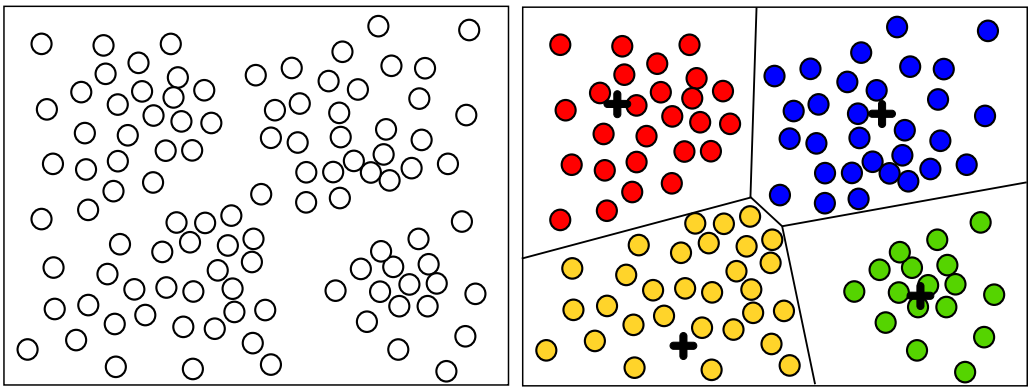


Figura 1.1: Clusterização particional baseada em centros de *cluster*. Um diagrama de Voronoi reproduz o particionamento gerado no espaço.

$$f(D, K) = \sum_{p \in D} \min_{c \in K} \{dist(p, c)\} \tag{1.1}$$

$$dist(a, b) = \|a - b\| = \sqrt{(\sum_{i=1}^{d} (a_i - b_i)^2)} \tag{1.2}$$

Problemas de clusterização geralmente possuem alta complexidade computacional. Em particular, clusterização particional é $\mathcal{NP}$-Difícil, quando modelado como um problema de otimização para algumas funções objetivo (GONZALEZ, 1982). Consequentemente, o uso de métodos heurísticos é uma boa alternativa para a resolução, sem garantia de otimalidade.

Neste trabalho, uma abordagem baseada na meta-heurística *Continuous Greedy Randomized Adaptive Search Procedure* (C-GRASP) é proposta para resolver clusterização particional. A performance do algoritmo é observada por meio de experimentos computacionais e estatísticos que foram conduzidos em comparação com meta-heurísticas conhecidas, bem como outras heurísticas do estado da arte para o problema.

1.2 Segmentação de imagens

Em visão computacional, segmentação de imagens é o processo de particionar uma imagem em algumas regiões de interesse (conjuntos de pixels), sem sobreposição, de modo que cada região é homogênea e a união de duas regiões adjacentes não é homogênea (PAL; PAL, 1993). Limiarização é uma das mais antigas, simples e populares técnicas de segmentação de imagens em nível de cinza, e geralmente é aplicada de forma global através do histograma da imagem, que armazena a frequência (número de pixels) para cada tom de cinza existente, como ilustrado na Figura 1.2. Formalmente, podemos generalizar a definição através da equação 1.3, para uma matriz de intensidade I e um conjunto de limiares $T = \{t_1, t_2, ..., t_n\}$, produzindo uma rotulação de cada pixel $I_{i,j}$.

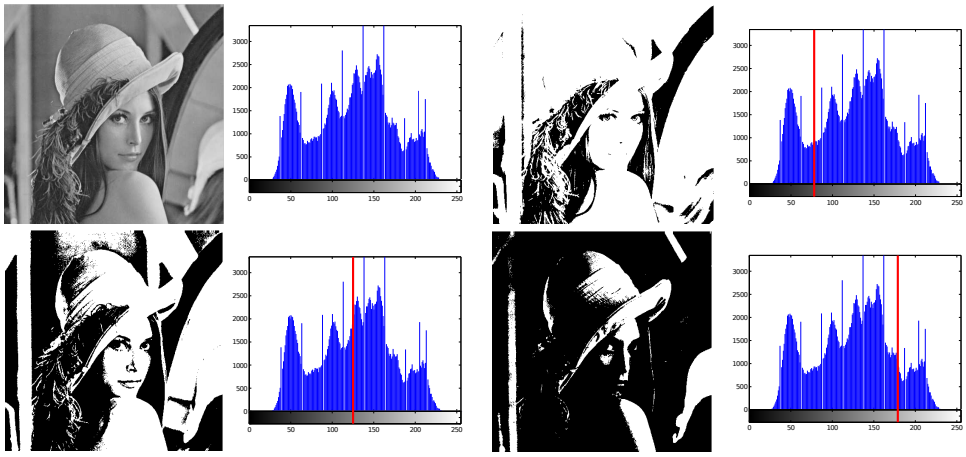


Figura 1.2: Diferentes limiarizações no histograma para uma imagem em nível de cinza.

$$I_{i,j} = \begin{cases} 0, \text{ se } I_{i,j} < t_1 \\ 1, \text{ se } t_1 \geq I_{i,j} < t_2 \\ ... \\ n, \text{ se } I_{i,j} \geq t_n \end{cases} \tag{1.3}$$

Diferentes métodos de limiarização automática se baseiam em distintos critérios de otimização para encontrar limiares adequados. O método de Otsu (OTSU, 1975), por exemplo, é um dos algoritmos mais populares e considera o critério de variância intraclasse mínima ou, de forma equivalente, a máxima variância entre classes (veja a seção 2.4). Para aplicações que demandam imagens com alta profundidade de bits, ou seja, definidas por uma grande quantidade de níveis de cinza (e.g. imagens de 16 bits podem codificar 65,536 níveis de cinza), técnicas de busca exaustiva como o método de Otsu são inviáveis, uma vez que o número de soluções possíveis cresce exponencialmente com relação ao número de limiares necessários. Dessa forma, abordagens meta-heurísticas podem encontrar limiares de boa qualidade em tempo aceitável, sem garantia de otimalidade.

Para o problema de limiarização multinível, algoritmos baseados nas meta-heurísticas *Particle Swarm Optimization* (PSO) e *Quantum-behaved Particle Swarm Optimization* (QPSO) serão investigados para imagens conhecidas da literatura. Uma hibridização do QPSO com uma busca local baseada em *Variable Neighborhood Descent* (VND) também será estudada. Além disso, uma aplicação específica de segmentação em estudos de materiais cimentícios é realizada, com a utilização dos resultados em uma etapa de detecção de trincas em amostras, para extração de características de interesse através de algoritmos em grafos.

1.3 Objetivos

- Propor uma nova abordagem baseada em C-GRASP para clusterização de dados particional com mínima distância *intracluster*.
- Propor uma abordagem híbrida que combina uma versão cooperativa do QPSO com VND para segmentação de imagens de propósito geral e imagens microscópicas de materiais cimentícios.
- Propor uma metodologia de extração de características pós-segmentação através de algoritmos em grafos.

1.4 Justificativa

A proposição de novos métodos e mecanismos de exploração do espaço de busca é importante para o avanço na área de meta-heurísticas, podendo gerar novos *insights* para técnicas futuras mais eficientes em diferentes problemas com aplicações relevantes. Pode-se ainda considerar as seguintes justificativas:

- O C-GRASP é uma meta-heurística recente e pouco explorada que ainda não foi utilizada e avaliada para tarefas de clusterização de dados, em particular, para clusterização particional com mínima distância *intracluster*.
- Abordagens híbridas baseadas em QPSO e busca local foram pouco exploradas pela literatura, principalmente para o problema de segmentação de imagens.
- O uso de limiarização multinível automática para segmentação de imagens de microscopia eletrônica de materiais baseados em cimento foi pouco explorado. Para extração de características pós-segmentação, o uso de algoritmos em grafos também foi pouco explorado nesse tipo de aplicação. Esses resultados podem induzir ferramentas importantes para estudos de propriedades em engenharia de materiais.

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, são apresentados conceitos e métodos necessários para facilitar o entendimento dos demais capítulos deste trabalho.

2.1 Otimização

Otimização ou Programação Matemática é uma área da matemática caracterizada pela maximização ou minimização de uma função objetivo sobre variáveis de decisão, geralmente modelando alguma aplicação real existente. Um conjunto de restrições limitam os valores possíveis que as variáveis de decisão podem assumir, modelando as limitações reais da aplicação. Formalmente, podemos definir genericamente um problema de minimização da seguinte forma:

$$z = \min\{f(x) \ : \ x \in S \subseteq \mathbb{R}^n\} \tag{2.1}$$

onde o conjunto S é a região viável definida pelas restrições do problema e a função objetivo $f(.)$ qualifica cada ponto da região. Dizemos que uma solução x' é um *ótimo local* se $f(x') \leq f(x)$ para $\forall x \in V \subset S$, onde V é uma vizinhança de x'. Uma solução x^* é um *ótimo global* (ou solução ótima) se $f(x^*) \leq f(x)$, $\forall x \in S \subseteq \mathbb{R}^n$.

A natureza da função objetivo e das restrições e o domínio das variáveis de decisão caracterizam o tipo de problema tratado, bem como o grau de dificuldade e o conjunto de técnicas de resolução disponíveis. Por exemplo, quando os problemas possuem função objetivo e restrições lineares sobre variáveis reais, temos um problema de *programação linear* e métodos como o *Algoritmo Simplex* (DANTZIG, 1998) e o *Algoritmo de Pontos*

Interiores (YE, 2011) possuem boa capacidade de resolução. Porém, muitos problemas possuem natureza não-linear e as técnicas de resolução existentes geralmente conseguem tratar apenas instâncias muito pequenas. Por exemplo, em otimização contínua não-linear , os métodos de resolução geralmente realizam cálculos de gradiente em larga escala, que possuem custo computacional bastante elevado. Tais problemas, quando não possuem algoritmos polinomiais e pertencem a classe de complexidade $\mathcal{NP}$-Difícil – conjunto de problemas pelo menos tão difíceis quanto os mais difíceis em $\mathcal{NP}$, ou seja, estão entre os problemas mais difíceis da computação – , podem ser tratados adequadamente por métodos heurísticos capazes de encontrar soluções de boa qualidade sem garantia de otimalidade.

Uma introdução detalhada sobre otimização pode ser encontrada em Nocedal e Wright (2006).

2.2 Heurísticas e meta-heurísticas

Para problemas de otimização que pertencem à classe de complexidade $\mathcal{NP}$-Difícil, encontrar boas soluções, que não são soluções ótimas, pode ser útil em muitas situações. Nestes casos, existe como alternativa a utilização de algoritmos que produzem soluções aproximadas, tais como *heurísticas* e *meta-heurísticas*.

Um algoritmo heurístico ou heurística é aquele que utiliza critérios ou condições inspirados no processo de intuição humana para encontrar, em tempo aceitável, boas soluções para um dado problema, mesmo que não seja garantida a otimalidade das soluções encontradas (SOUZA, 2008). As heurísticas se dividem em dois tipos principais: *heurísticas de construção* e *heurísticas de refinamento*.

Heurísticas de construção são procedimentos que produzem uma solução inicial viável para um problema de otimização através da inserção incremental de elementos, utilizando um critério específico e intuitivo. Um dos critérios mais utilizados é o da *escolha gulosa*, onde a construção considera a melhor escolha baseado em informações "locais" para determinar o próximo elemento da solução em construção.

Heurísticas de refinamento ou *busca local* são técnicas que se baseiam no conceito de vizinhança para melhorar uma solução previamente construída. Seja uma solução $s \in S$, onde S é o espaço de soluções possíveis, temos que $N(s)$ representa, através de um critério específico, a *estrutura de vizinhança* ou conjunto de soluções vizinhas de s, tal que $N(s) \subseteq S$. Um *movimento* em uma busca local é a mudança de uma solução s para

uma solução vizinha $s' \in N(s)$.

Meta-heurísticas são procedimentos genéricos ou *frameworks* de otimização global para busca de boas soluções (podendo ser a solução ótima) que se adequam a diversos problemas de otimização. Geralmente, elas utilizam em conjunto heurísticas de construção e de refinamento para escapar de ótimos locais durante a execução. Algumas das meta-heurísticas mais conhecidas são: Busca Tabu, GRASP, *Simulated Annealing*, ILS (do inglês, *Iterated Local Search*), Algoritmos Genéticos, VNS (do inglês, *Variable Neighborhood Search*) e PSO. Duas características principais em uma meta-heurística são a capacidade de *diversificação* e *intensificação*. A diversificação está relacionada a variabilidade de amostragem de soluções no espaço de busca e é importante na fuga de ótimos locais. A intensificação consiste na capacidade de busca intensa do método em uma região considerada promissora do espaço de busca. O equilíbrio desses dois fatores é fundamental para uma meta-heurística eficiente. A Figura 2.1 ilustra essas duas características através da perturbação – alteração parcialmente aleatória de uma solução para fugir do ótimo local atual – e busca local da meta-heurística ILS (LOURENÇO et al., 2003).

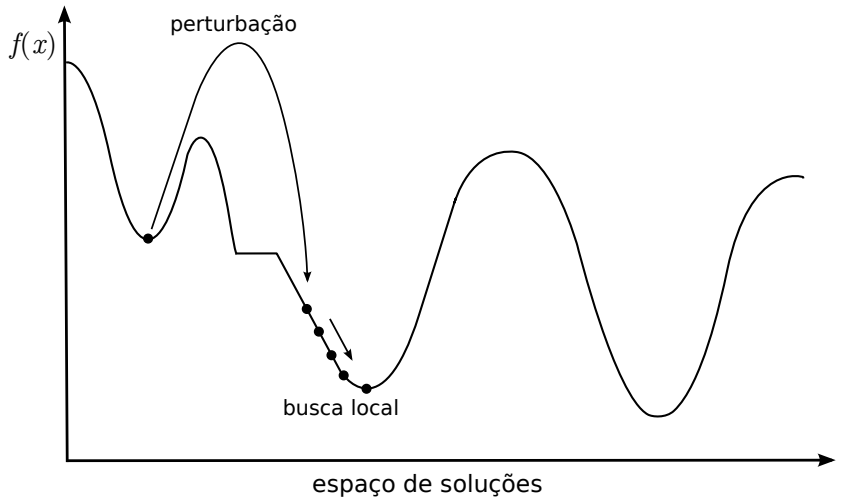


Figura 2.1: Diversificação e intensificação ilustrados pelos mecanismos de perturbação e busca local do ILS.

2.2.1 Busca Tabu

A meta-heurística *Busca Tabu* (TS, do inglês *Tabu Search*), proposta por Glover (1986) e Hansen (1986), consiste essencialmente em um procedimento iterativo que se movimenta no espaço de soluções através de uma busca local que encontra o melhor vizinho da solução corrente por explorar uma vizinhança bem definida. Além disso, um mecanismo de memória é utilizado para tentar evitar uma exploração cíclica que tende a

prender a busca em uma região de ótimo local.

A partir de uma solução inicial s, a busca local do TS avalia iterativamente um subconjunto V da vizinhança $N(s)$ e determina a próxima solução corrente como o melhor vizinho $s' \leftarrow \arg\min\{f(s'') \mid s'' \in V\}$, mesmo que $f(s') > f(s)$, considerando um problema de minimização.

Devido o critério guloso da busca local, o algoritmo pode ciclar, repetindo uma sequência de soluções já analisada. Para amenizar esse problema, um mecanismo chamado *lista tabu* é utilizado para armazenar o histórico recente de movimentos que devem ser ignorados pela busca local. A lista tabu T é uma fila de tamanho fixo que segue o conceito FIFO (do inglês, *First In First Out*), onde o movimento mais antigo deve ser o primeiro a sair, caso a fila já possua tamanho máximo. Geralmente, armazena-se os movimentos retroativos em T, com o intuito de evitar o retorno para aquela solução já visitada nas próximas $|T|$ iterações.

Movimentos *tabus* podem gerar novas soluções quando aplicados a soluções diferentes daquelas que induziram a sua inserção na lista. Dessa forma, considerar o uso desses movimentos, em algumas situações, pode ser vantajoso. A *função de aspiração* $A(.)$ é utilizada como estratégia para desprezar o *status* tabu de um movimento quando certas condições forem satisfeitas. Para uma solução s, $A(f(s))$ define um limiar de aspiração que a solução s' gerada pelo movimento tabu deve respeitar para ser aceito, ou seja, $f(s') < A(f(s))$. A abordagem mais simples e comumente utilizada consiste em aceitar movimentos tabu quando os mesmos produzem um melhoramento na melhor solução corrente s^*, ou seja, quando $A(f(s)) = f(s^*)$.

O pseudocódigo do TS é apresentado no Algoritmo 1 e possui os seguintes parâmetros de entrada: a função objetivo $f(.)$, a estrutura de vizinhança $N(.)$, a função de aspiração $A(.)$, a quantidade de vizinhos avaliada $|V|$, um limite inferior da função objetivo f_{min}, tamanho máximo da lista tabu $|T|$, número máximo de iterações sem melhora BT_{max} e a solução de partida s. O laço principal entre linhas 7-16, possui duas condições de término comumente utilizadas: quando o valor de $f(.)$ atingido é menor que um limite inferior conhecido f_{min} ou quando ocorre um número de iterações sem melhora da melhor solução corrente através de BT_{max}. Nas linhas 9 e 10 a busca local e atualização da lista tabu são realizadas, respectivamente. Entre as linhas 11-14 a melhor solução conhecida é atualizada, quando necessário. Finalmente, após a saída do laço principal, a melhor solução é retornada (linhas 18).

Mais detalhes sobre a Busca Tabu podem ser encontrados em Souza (2008) e Glover

Algoritmo 1: Pseudocódigo da Busca Tabu. Adaptado de Souza (2008).

```
1  procedimento BuscaTabu (f(.), N(.), A(.), |V|, f_min, |T|, BT_max, s)
2      s* ← s ;                                   /* melhor solução corrente */
3      Iter ← 0;
4      MelhorIter ← 0 ;              /* última iteração que atualizou s* */
5      T ← ∅ ;                                              /* lista tabu */
6      Inicialize a função de aspiração A;
7      enquanto Iter − MelhorIter ≤ BT_max faça
8          Iter ← Iter + 1;
9          Seja s' ← s ⊕ m o melhor vizinho em V ⊆ N(s) tal que m ∉ T ou
           f(s') < A(f(s));
10         Atualize a lista tabu T;
11         s ← s'; se f(s) < f(s*) então
12             s* ← s;
13             MelhorIter ← Iter;
14         fim
15         Atualize a função de aspiração A;
16     fim
17     s ← s*;
18     returne s;
19 fim
```

(1986).

2.2.1.1 Busca Tabu para clusterização de dados

Em Al-Sultan (1995) uma abordagem baseada em TS foi proposta para clusterização de dados.

Antes da descrição algoritmo, é necessário definir as seguintes notações:

- A é um arranjo com $|D|$ dimensões (número de objetos na base D) e no i-ésimo elemento A_i existe um número que representa o *cluster* a qual esse elemento pertence. Por exemplo, $A = [1\ 2\ 2\ 2\ 1\ 3\ 3\ 2\ 1\ 2]$ representa uma clusterização para $|D| = 10,\ k = 3$.
- J é o valor da função objetivo calculada pelos centróides (definido na seção 2.3) obtidos da clusterização A.
- A_t, A_c e A_b denotam os arranjos de solução experimental, atual e melhor.
- J_t, J_c e J_b denotam os valores da função objetivo para A_t, A_c e A_b, respectivamente.

Segue a descrição do algoritmo:

1. *Inicialização*: Considerando A_c uma solução inicial e J_c o valor da função objetivo dessa solução. Consequentemente temos que $A_b = A_c$ e $J_b = J_c$. Atribua valores para os seguintes parâmetros de entrada: MTLS (tamanho máximo da lista tabu) , P $\in [0, 1]$ (limiar de probabilidade), NTS (número de vizinhos a avaliar) e ITMAX (número máximo de iterações). Atribua $k = 1$ e TLL(tamanho da lista tabu)= 0 e vá para o passo 2.

2. A partir de A_c, avalie NTS vizinhos $A_t^1, A_t^2, \ldots, A_t^{NTS}$ e calcule seus valores de função objetivo $J_t^1, J_t^2, \ldots, J_t^{NTS}$. A estratégia utilizada para gerar um vizinho é a seguinte: Dado A_c e o limiar de probabilidade P, para $i = 1, 2, \ldots, |D|$, $A_t(i) = A_c(i)$ se um número aleatório uniforme $u(0, 1)$ for menor do que P, caso contrário, $A_t(i)$ recebe um identificador de *cluster* aleatório do conjunto $\{l : l = 1, 2, 3, \ldots, k,\ l \neq A_c(i)\}$. Em seguida, vá para o passo 3.

3. Após ordenar de forma crescente o arranjo $J_t^1, J_t^2, \ldots, J_t^{NTS}$, temos o arranjo $J_t^{[1]}, J_t^{[2]}$, $\ldots, J_t^{[NTS]}$. Se a solução associada a $J_t^{[1]}$ não for tabu ou se $J_t^{[1]} < J_b$ (critério de aspiração), então $A_c = A_t^{[1]}$ e $J_c = J_t^{[1]}$ e vá para o passo 4. Caso contrário, atribua a primeira solução dos vizinhos ordenados a A_c que satisfaz essas condições. Se nenhum vizinho satisfaz essas condições, vá para o passo 2.

4. Insira J_c no fim da lista tabu e faça TLL = TLL + 1 (se TLL < MTLS). Se TLL = MTLS, delete o primeiro elemento da lista, que deixa de ser tabu. Se $J_c < J_b$, então $J_b = J_c$ e $A_b = A_c$. Faça $k = k + 1$ e vá para o passo 2.

2.2.2 Descida em Vizinhança Variável

Proposto por Mladenović e Hansen (1997), o algoritmo de *Descida em Vizinhança Variável* (VND, em inglês *Variable Neighborhood Descent*) é uma técnica de busca local que explora de forma sequencial um conjunto de k estruturas de vizinhança $N = \{N_1, N_2, ..., N_k\}$ (definido na seção 2.2). Basicamente, o algoritmo compara o melhor vizinho s' gerado pela estrutura atual (inicialmente N_1) com a melhor solução corrente s. Caso s' seja pior, muda-se para a próxima estrutura da sequência. Caso s' seja melhor, reinicia-se a busca a partir da primeira estrutura N_1, dessa vez tentando refinar a solução s'. O critério de parada consiste em utilizar todas as estruturas de vizinhança sem que nenhuma melhora ocorra, ou seja, se N_k não for capaz de produzir uma melhora. No Algoritmo 2 temos o pseudocódigo genérico do VND para problemas de minimização.

Algoritmo 2: Pseudocódigo do VND.

```
1  procedimento VND (s, N, f)
2      k_max ← |N| ;          /* número máximo de estruturas de vizinhança */
3      k ← 1;
4      enquanto k ← k_max faça
5          s' ← melhorVizinho ∈ N_k(s) ;          /* captura melhor vizinho */
6          se f(s') < f(s) então
7              s ← s' ;            /* nova melhor solução global encontrada */
8              k ← 1 ;                              /* reinicia varredura em N */
9          fim
10         senão
11             k ← k + 1 ;  /* muda para a próxima estrutura de vizinhança */
12         fim
13     fim
14     retorna s ;                 /* retorna a melhor solução encontrada */
15 fim
```

Em Mladenović e Hansen (1997) pode-se encontrar informações mais detalhadas sobre este método.

2.2.3 Algoritmos Genéticos

Um *Algoritmo Genético* (GA, em inglês *Genetic Algorithm*) é uma técnica que se inspira em princípios da evolução para resolver problemas computacionais. Conceitos como população de indivíduos, cruzamento, mutação e *fitness* (aptidão) são a base destes algoritmos.

Em problemas de otimização, considera-se uma *população* como um conjunto de *cromossomos* (ou *indivíduos*) que são soluções viáveis para o problema. Cada elemento de um indivíduo é chamado de *gene*. Toda solução possui um valor de *fitness* (ou *aptidão*) que o classifica em relação ao resto da população. A cada *geração* temos mudanças na população provenientes de operações de *cruzamento* e *mutação*. A operação de cruzamento consiste na geração de indivíduos descendentes (ou filhos) a partir da mistura de genes de dois indivíduos parentais. A *seleção* é a etapa que define quais indivíduos da população serão escolhidos para realizar cruzamento através de uma estratégia específica, que geralmente considera o *fitness* dos mesmos. A operação de mutação consiste de alterações individuais que ocorrem em uma pequena parte dos genes de uma solução (cromossomo), geralmente nos dois filhos gerados após o cruzamento de dois pais. Na Figura 2.2, temos o diagrama de fluxo do algoritmo, com ilustração de uma operação de cruzamento e mutação simples. Após a ocorrência de todas as gerações, o indivíduo que em alguma das gerações

possuiu o maior valor de *fitness* obtido é considerado a melhor solução encontrada pelo algoritmo. Os principais parâmetros do GA são: tamanho da população, probabilidade de cruzamento (percentual da população que fará cruzamento) e probabilidade de mutação (percentual do cromossomo que sofrerá mutação).

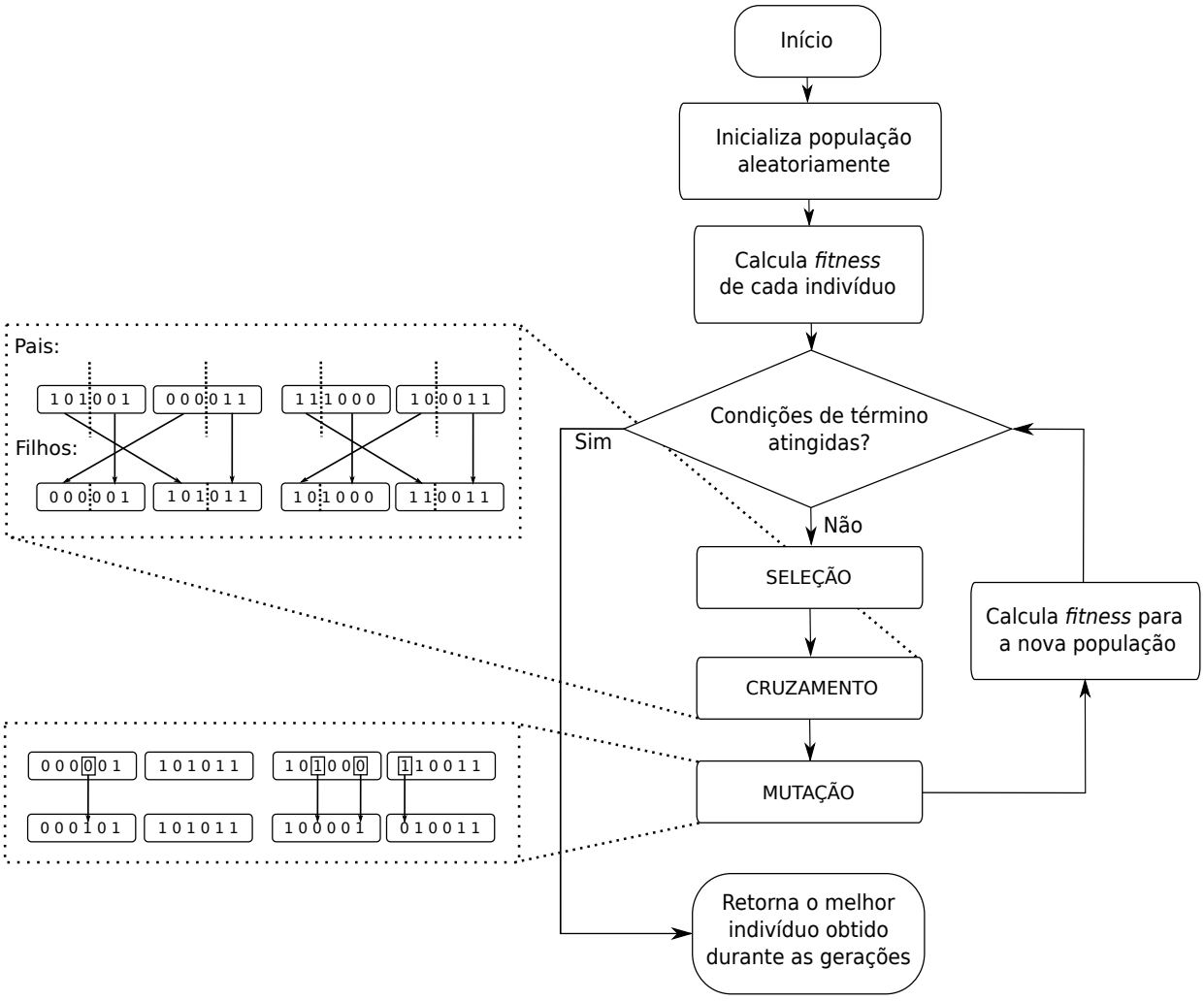


Figura 2.2: Diagrama de fluxo do Algoritmo Genético básico.

Mais detalhes sobre algoritmos genéticos podem ser encontrados em Whitley (1994).

2.2.3.1 Algoritmo Genético para clusterização de dados

Na abordagem baseada em GA proposta por Maulik e Bandyopadhyay (2000) para clusterização de dados, cada indivíduo retém uma sequência de números reais para codificar os centros de *cluster*. Por exemplo, para uma tarefa no espaço bidimensional e 3 *clusters*, o indivíduo $I = [10,2\ 33,4\ 40,3\ 63,7\ 9,4\ 90,4]$ representa os centros: (10,2; 33,4), (40,3; 63,7) e (9,4; 90,4).

Na fase inicialização da população, para cada indivíduo (cromossomo) são atribuídos os centróides obtidos da clusterização dada por k pontos dos dados escolhidos aleatoriamente como centros de *cluster*.

Na fase de seleção por roleta (estratégia utilizada), a probabilidade de escolha de um indivíduo é proporcional ao seu *fitness*. Uma implementação comum é empilhar o *fitness* de todos os candidatos e gerar um número aleatório limitado pelo valor da soma total. Maiores valores de *fitness* representam maiores "fatias" da pilha e consequentemente uma maior chance de ser escolhido. Uma analogia com fatias de pizza também é bastante utilizada para ilustrar essa estratégia.

O cruzamento de corte simples com probabilidade de cruzamento fixo μ_c foi utilizado. Basicamente um número inteiro aleatório é gerado como ponto de corte dos dois cromossomos (pais) para gerar dois filhos por unir partes de pais diferentes, preservando o tamanho dos indivíduos. Essa operação já foi ilustrada na Figura 2.2.

A operação de mutação simples é aplicada com probabilidade μ_m para cada indivíduo após o cruzamento. Por se tratar de genes codificados com números reais, um valor δ é gerado aleatoriamente entre $[0, 1]$ com distribuição uniforme e o novo valor v do gene é dado por ($\pm$ com igual probabilidade):

$$v \pm 2 \times \delta \times v, \ v \neq 0, \tag{2.2}$$

$$v \pm 2 \times \delta, \ v = 0 \tag{2.3}$$

O critério de término é atingir o número máximo de iterações (gerações). Um leve elitismo foi introduzido por preservar na população o melhor indivíduo encontrado até a iteração corrente.

2.2.4 *Continuous* GRASP

O *Continuous Greedy Randomized Adaptive Search Procedure* (C-GRASP), proposto por Hirsch et al. (2007), é uma adaptação da meta-heurística GRASP (FEO; RESENDE, 1995) para resolução de problemas de otimização contínua, uma vez que o método convencional é adequado para problemas de otimização discreta. A ideia principal do algoritmo foi herdada do GRASP e consiste em uma estratégia *multi-start* sobre uma fase de construção e busca local. A primeira fase combina aleatoriedade com um critério guloso para

prover diversificação e gerar boas soluções de partida que devem ser refinadas pela fase posterior.

No C-GRASP, a tarefa de otimização pode ser definida como encontrar o ponto ótimo $x^* \in S \subseteq \mathbb{R}^n$ para uma função $f : S \mapsto \mathbb{R}$, de modo que $S = \{x \in \mathbb{R}^n : l \leq x \leq u\}$ é um hiperetângulo limitado por $l, u \in \mathbb{R}^n$, tal que $u \geq l$. O algoritmo explora a discretização de S em uma grade com densidade crescente sem realizar cálculos de derivada, que possuem custo computacional elevado para determinadas funções.

O pseudocódigo do C-GRASP é descrito no Algoritmo 3, com os seguintes parâmetros de entrada: n é o número de dimensões do problema, l e u são os vetores *lower bound* e *upper bound* que determinam as restrições em caixa (*box constraints*) do espaço de busca, $f(.)$ é a função objetivo, *MaxIters* consiste no número de iterações no procedimento *multi-start*, *MaxNumIterNoImprov* determina o número de iterações que o método busca uma melhor solução sem reduzir o tamanho do passo h, *NumTimesToRun* determina o número de execuções do procedimento, *MaxDirToTry* o número de direções que a busca local irá analisar, e o α determina o balanceamento entre aleatoriedade e escolha gulosa na fase de construção. Para cada iteração do laço na linha 4, o método é realizado para um ponto inicializado aleatoriamente com distribuição uniforme (linha 5) e tamanho do passo h inicialmente igual a 1, sendo este último responsável pela densidade da grade a ser explorada. No *multi-start* interno (linha 6), a fase de construção `Construção` é executada sobre o x seguido pelo procedimento `BuscaLocal` que recebe o x resultante da fase anterior. Em seguida a melhor solução encontrada x^* é atualizada quando houver melhora (linhas 9 e 10). O tamanho do passo h é reduzido pela metade (aumentando a densidade da grade), sempre que houver *NumIterNoImprov* iterações sem melhora (linhas 15 e 16).

Na fase de construção do C-GRASP, como mostra o Algoritmo 4, um ponto x é tomado como entrada e uma busca linear (linha 7) é realizada para cada coordenada não fixada (conjunto C). Inicialmente todas as coordenadas não estão fixadas e para cada coordenada i é realizada uma busca linear mantendo-se as outras coordenadas inalteráveis. As soluções resultantes z_i das buscas lineares são armazenadas, bem como a melhor (min) e pior (max) soluções decorrentes dessa etapa (linhas 8-11). A Lista Restrita de Candidatos (RCL, do inglês *Restricted Candidate List*) consiste em um conjunto de soluções que respeitam um limiar definido pelo parâmetro α e as soluções min e max obtidas pela etapa anterior (linhas 14-19). Finalmente uma solução é escolhida aleatoriamente da RCL para atualizar a j-ésima coordenada de x, sendo esta coordenada fixada para as próximas iterações (linhas 20 e 21). O procedimento se repete até que todas as coordenadas sejam fixadas

Algoritmo 3: Pseudocódigo do C-GRASP. Adaptado de Hirsch et al. (2007)

```
1  procedimento C-GRASP (n, l, u, f(.), MaxIters, MaxNumIterNoImprov,
2  NumTimesToRun, MaxDirToTry, α)
3      x* ← nil;  f* ← ∞;
4      para cada j = 1, ..., NumTimesToRun fazer
5          x ← UnifRand(l, u); h ← 1; NumIterNoImprov ← 0;
6          para cada Iter = 1, ..., MaxIters fazer
7              x ← Construção(x, f(.), n, h, l, u, α);
8              x ← BuscaLocal(x, f(.), n, h, l, u, MaxDirToTry);
9              se f(x) < f* então
10                 x* ← x; f* ← f(x); MaxNumIterNoImprov ← 0;
11             fim
12             senão
13                 NumIterNoImprov ← NumIterNoImprov + 1;
14             fim
15             se NumIterNoImprov ≥ MaxNumIterNoImprov então
16                 h ← h/2; NumIterNoImprov ← 0;
17             fim
18         fim
19     fim
20     retorne x*;
21 fim
```

$(C = \emptyset)$.

Na fase de busca local, ilustrada no algoritmo 5, busca-se melhorar a solução por explorar o conjunto de direções $\Gamma = \{d \in \mathbb{R}^n : d_i = -1 \vee d_i = 0 \vee d_i = 1,\ i = 1, ..., n\} \setminus \{d \in \mathbb{R}^n : d_i = 0, i = 1, ..., n\}$ que consiste em todas as possibilidades de vetores com $\{-1, 0, 1\}$ em cada coordenada, exceto o vetor nulo. Por exemplo, para $n = 2$, temos que $\Gamma = \{(0,1), (0,-1), (1,1), (1,0), (1,-1), (-1,1), (-1,0), (-1,-1)\}$, como ilustrado no exemplo da Figura 2.3(a). A cardinalidade do conjunto de direções é dada por $3^n - 1$ para n dimensões, podendo-se realizar um mapeamento bijetivo $T : \{1, ..., 3^n - 1\} \mapsto \Gamma$. Dado um ponto x de entrada, o procedimento basicamente gera *NumDirToTry* (linha 4) direções aleatórias através do mapeamento T e movimenta x com um passo h na direção candidata (linha 10), repetindo o mecanismo sempre que houver melhora.

Algoritmo 4: Construção do C-GRASP original Hirsch et al. (2007)

```
1  procedimento Construção (x, f(.), n, h, l, u, α)
2      C ← {1, 2, ..., n};
3      enquanto C ≠ ∅ faça
4          min ← +∞; max ← −∞;
5          para cada i = 1, ..., n fazer
6              se i ∈ C então
7                  z_i ← BuscaLinear(x, h, i, n, f(.), l, u);
8                  x' ← (x_1, ..., x_{i−1}, z_i, x_{i+1}, ..., x_n);
9                  g_i ← f(x');
10                 se min > g_i então  min ← g_i ;
11                 se max < g_i então  max ← g_i ;
12             fim
13         fim
14         RCL ← ∅;
15         para cada i = 1, ..., n fazer
16             se i ∈ C and g_i ≤ (1 − α) * min + α * max então
17                 RCL ← RCL ∪{i};
18             fim
19         fim
20         j ← SelectionaElementoAleatoriamente(RCL);
21         x ← z_j; C ← C \ {j};
22     fim
23     returne x*;
24 fim
```

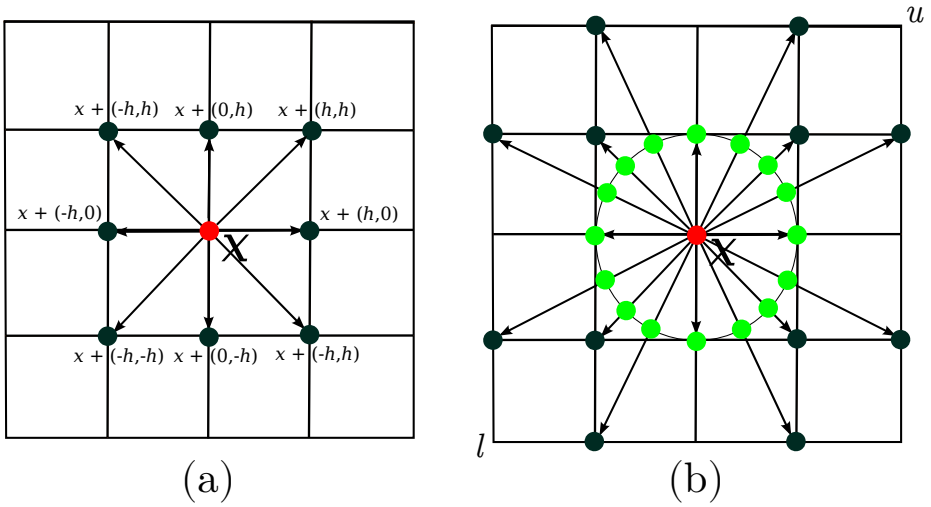


Figura 2.3: Geração de vizinhança utilizado na busca local para um problema bidimensional. a) Mecanismo de vizinhança da busca local do C-GRASP. b) Mecanismo de vizinhança da busca local do *new* C-GRASP.

Algoritmo 5: Busca local do C-GRASP original (HIRSCH et al., 2007)

```
1  procedimento BuscaLocal (x, n, l, u, f(.), MaxDirToTry)
2      Improved ← true; D ← ∅;
3      x* ← x; f* ← f(x);
4      NumDirToTry ← min{3^n − 1, MaxDirToTry};
5      enquanto Improved faça
6          Improved ← false;
7          enquanto |D| ≤ NumDirToTry ∧ ¬Improved faça
8              r ← ⌈UnifRand(1, 3^n − 1)⌉ ∉ D;
9              D ← D ∪ {r};
10             d ← T(r); x ← x* + h ∗ d;
11             se l ≤ x ≤ u então
12                 se f(x) ≤ f* então
13                     x* ← x; f* ← f(x);
14                     D ← ∅;
15                     Improved ← true;
16                 fim
17             fim
18         fim
19     fim
20     returne x*;
21 fim
```

2.2.4.1 Acelerando o *Continuous* GRASP

Em Hirsch et al. (2010), alguns melhoramentos foram propostos para acelerar o C-GRASP em tarefas de otimização contínua de funções multimodais de pequeno e médio porte. Algumas mudanças estruturais foram realizadas no procedimento principal além de melhoramentos na construção e uma nova busca local. Esta versão foi chamada de *new* C-GRASP e seu pseudocódigo é apresentado no Algoritmo 6 com os novos parâmetros de entrada: tamanho do passo inicial h_s, tamanho do passo final ou mínimo h_e e ρ_{lo} que define o percentual de direções a serem analisadas na fase de busca local. Nesta versão, para cada novo ponto de partida a busca é feita com tamanho de passo decaindo de h_s até h_e (linhas 5 e 6) ao invés de um número fixo de iterações como no C-GRASP. Além disso, h é reduzido pela metade sempre que nenhum melhoramento ocorre em x após a construção e busca local, sinalizados pelas variáveis ImprC e ImprL, respectivamente (linhas 9-13). Essa etapa difere do C-GRASP, que insiste uma quantidade máxima de iterações sem melhora até reduzir h.

Algoritmo 6: *new* C-GRASP

```
1  procedimento new C-GRASP (n, l, u, f(.), h_s, h_e, ρ_lo)
2      f* ← ∞;
3      enquanto critérios de parada não satisfeitos faça
4          x ← UnifRand(l, u);
5          h ← h_s;
6          enquanto h > h_e faça
7              ImprC ← false;
8              ImprL ← false;
9              [x, ImprC] ← Construção(x, f(.), n, h, l, u, ImprC);
10             [x, ImprL] ← BuscaLocal(x, f(.), n, h, l, u, ρ_lo, ImprL);
11             se f(x) < f* então  x* ← x; f* ← f(x); ;
12             se ImprC = false ∧ ImprL = false então
13                 h ← h/2
14             fim
15         fim
16     fim
17     retorne x*;
18 fim
```

Basicamente, os ajustes feitos na construção original foram: calcular dinamicamente o valor do parâmetro α e utilizar uma variável booleana chamada $ReUso$ para evitar buscas lineares repetidas quando x não tem sido modificado na iteração anterior. A nova versão é descrita no Algoritmo 7.

A nova busca local proposta consiste basicamente em projetar pontos da grade na

hiperesfera de raio h a partir da solução atual $\bar{x}$, como ilustrado na Figura 2.3(b). Formalmente, seguindo a definição de Hirsch et al. (2010), temos que:

$$S_h(\bar{x}) = \{x \in S \mid l \leq x \leq u, \bar{x} + r \times h, r \in \mathbb{Z}^n\} \tag{2.4}$$

é o conjunto de pontos da grade com densidade h em S gerados a partir de $\bar{x}$. Dessa forma, temos que o conjunto de pontos vizinhos pode ser definido da seguinte maneira:

$$B_h(\bar{x}) = \{x \in S \mid x = \bar{x} + h \times (x' - \bar{x})/||x' - \bar{x}||, x' \in S_h(\bar{x}) \setminus \{\bar{x}\}\} \tag{2.5}$$

onde $B_h(\bar{x})$ é gerado pela projeção de cada ponto em $S_h(\bar{x}) \setminus \{\bar{x}\}$ na hiperesfera.

O pseudocódigo da busca local é descrito no Algoritmo 8. O procedimento calcula a quantidade total de pontos na grade (linha 4) e em seguida define o número máximo de vizinhos gerados $MaxPoints$ como um percentual do total através do parâmetro p_{lo}. Finalmente, os vizinhos são amostrados aleatoriamente (linhas 7-16) e a melhor solução encontrada é retornada.

Algoritmo 7: Construção do *new* C-GRASP

```
1  procedimento ConstruçãoMelhorada (x, f(.), n, h, l, u, ImprC)
2      C ← {1, 2, ..., n};
3      α ← UnifRand(0,1);
4      ReUso ← false;
5      enquanto C ≠ ∅ faça
6          g ← +∞; ḡ ← −∞;
7          para cada i = 1, ..., n fazer
8              se i ∈ C então
9                  se ReUso = false então
10                     z_i ← BuscaLinear(x, h, i, n, f(.), l, u);
11                     x' ← (x_1, ..., x_{i−1}, z_i, x_{i+1}, ..., x_n);
12                     g_i ← f(x');
13                 fim
14                 se g > g_i então  g ← g_i ;
15                 se ḡ < g_i então  ḡ ← g_i ;
16             fim
17         fim
18         RCL ← ∅;
19         para cada i = 1, ..., n fazer
20             se i ∈ C ∧  g_i ≤ g + α(ḡ − g) então
21                 RCL ← RCL ∪{i};
22             fim
23         fim
24         j ← SelectionaElementoAleatoriamente(RCL);
25         se x_j = z_j então
26             ReUso ← true;
27         fim
28         senão
29             x ← z_j;
30             ReUso ← false;
31             ImprC ← true;
32         fim
33         C ← C \ {j};
34     fim
35     retorne [x, ImprC];
36 fim
```

Algoritmo 8: Busca local do *new* C-GRASP

procedimento `BuscaLocalMelhorada` $(x, f(.), n, h, l, u, \rho_{lo}, ImprL)$
  $x^* \leftarrow x$;
  $f^* \leftarrow f(x)$;
  $GridPoints \leftarrow \prod_{i=1}^{n} \lceil (u_i - l_i/h) \rceil$;
  $MaxPoints \leftarrow \lceil \rho_{lo} \times GridPoints \rceil$;
  $PointsExamined \leftarrow 0$;
  enquanto $PointsExamined \leq MaxPoints$ **faça**
    $PointsExamined \leftarrow PointsExamined + 1$;
    $x \leftarrow$ `SelectionaElementoAleatoriamente`$(B_h(x^*))$;
    se $l \leq x \leq u \wedge f(x) < f^*$ **então**
      $x^* \leftarrow x$;
      $f^* \leftarrow f(x)$;
      $ImprL \leftarrow true$;
      $PointsExamined \leftarrow 0$;
    fim
  fim
  retorne $[x^*, ImprL]$;
fim

2.2.5 *Particle Swarm Optimization*

Otimização por Enxame de Partículas (PSO, em inglês *Particle Swarm Optimization*) é um técnica de otimização estocástica, proposta por Eberhart e Kennedy (1995), motivada pelo comportamento social de organismos, tais como pássaros em busca de comida. No PSO, cada partícula representa uma solução para o problema de otimização subjacente e a nuvem de partículas, de forma análoga à população de indivíduos em Algoritmos Genéticos (MELANIE, 1996), constitui um conjunto de soluções candidatas. Na busca pelo ótimo global, o enxame de partículas (veja a Figura 2.4) se movimenta no hiperespaço de acordo com equações que definem a próxima posição de cada partícula baseado em estratégias que consideram a melhor posição obtida por uma partícula ou *pbest* (do inglês, *personal best*) e a melhor solução do enxame *gbest* (do inglês, *global best*) encontrada até a iteração corrente. Em alguns casos, as partículas vizinhas também podem influenciar a movimentação.

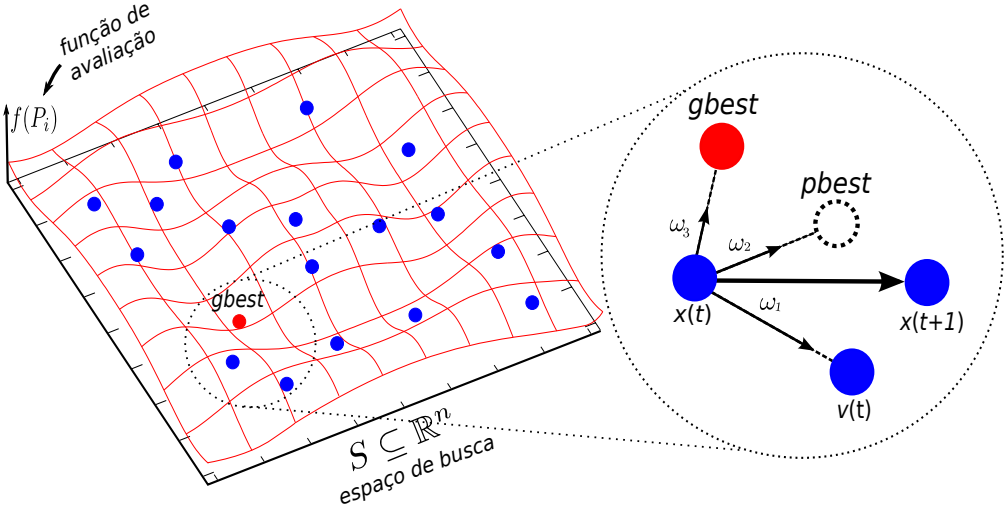


Figura 2.4: Enxame de partículas para um problema bidimensional.

No PSO básico, o cálculo das componentes do vetor velocidade de cada partícula i em cada dimensão j na iteração $t+1$ é dado pela equação 2.6. O escalar $w \in [0,1]$ consiste no peso inercial que possui o papel de freio gradual do enxame em sintonia com a convergência do método. As constantes c_1 e c_2 são coeficientes de aceleração que ponderam os termos cognitivo (que utiliza o *pbest*) e social (que utiliza o *gbest*) da partícula. Os pesos $r_{1j}, r_{2j} \sim U([0,1])$ são valores aleatórios distribuídos uniformemente e podem prover diversificação na movimentação do enxame. A posição de uma partícula i na iteração $t+1$ é calculada pela equação 2.7.

$$V_{ij}^{t+1} = w \times V_{ij}^{t} + c_1 \times r_{1j} \times (pbest_{ij} - X_{ij}^{t}) + c_2 \times r_{2j} \times (gbest_{ij} - X_{ij}^{t}) \tag{2.6}$$

$$X_{ij}^{t+1} = X_{ij}^{t} + V_{ij}^{t+1} \tag{2.7}$$

Na versão pura, o PSO inicia com vetores de velocidade nulos. Por isso, nesta iteração, o gbest possui maior poder de influência do que nas iterações seguintes. O PSO é descrito em alto nível pelo diagrama de fluxo na Figura 2.5 e pelo seguinte pseudocódigo:

Passo 1: Inicializa o enxame de partículas com posições aleatórias, calcula a função objetivo de cada partícula (*pbest* inicial) e encontra o *gbest* inicial;
Passo 2: Calcula os vetores velocidade e posições através das equações 2.6 e 2.7;
Passo 3: Atualiza, quando ocorre melhora, *pbest* e *gbest*;
Passo 4: Se os critérios de término são satisfeitos, retorne o *gbest*. Caso contrário, vá para o *Passo 2*;

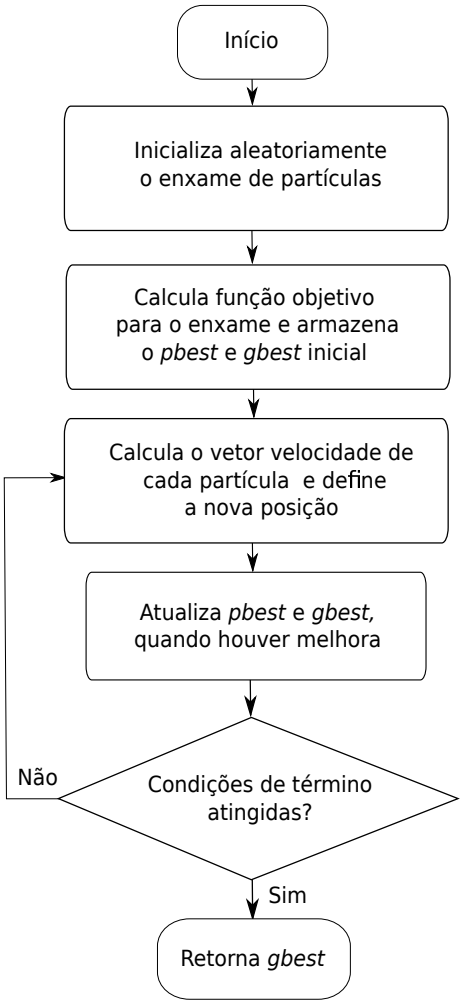


Figura 2.5: Diagrama de fluxo do PSO básico.

2.2.5.1 *Quantum-behaved* PSO

Nos últimos anos, vários pesquisadores tem tentado propor novas estratégias de movimentação das partículas, alguns deles com técnicas baseadas em conceitos da mecânica quântica (COELHO, 2010). Diferente do PSO tradicional, a Otimização por Nuvem de Partículas com Comportamento Quântico (QPSO, do inglês, Quantum-behaved Particle Swarm Optimization), proposta por Sun et al. (2004), se inspira no princípio da incerteza de Heisenberg, onde não se pode determinar simultaneamente a posição X_i e velocidade V_i de uma partícula. Segundo Li et al. (2015), o QPSO é mais eficiente que o PSO pois o mesmo modela um sistema não-linear complexo, podendo codificar mais estados que o PSO convencional, além de gerar trajetórias não-determinísticas para as partículas. No QPSO, o estado de cada partícula é descrito por uma função de onda $\Psi(x,t)$, ao invés da posição e velocidade. Dessa forma, temos que a probabilidade da partícula estar na posição X_i é definida pela função de densidade de probabilidade $|\Psi(x,t)|^2$. Por exem-

plo, em um espaço tridimensional, $|\Psi(x,t)|^2 dxdydz$ determina a probabilidade de uma partícula x estar no volume $dxdydz$ no tempo t. A partir de um desenvolvimento extenso de equações em Sun et al. (2004), temos que a movimentação das partículas é guiada pelas equações 2.8 e 2.9.

$$P_i = \varphi \times pbest_i(t) + (1-\varphi) \times gbest, \tag{2.8}$$

$$X_i(t+1) = P_i \pm \alpha \times |P_i - X_i(t)| \times ln(\frac{1}{u}) \tag{2.9}$$

onde P_i é chamado de *Ponto de Inclinação de Aprendizagem* (LIP, do inglês *Learning Inclination Point*) e combina informações do *pbest* e *gbest* para influenciar a aprendizagem de cada partícula. As constantes φ e u são números aleatórios uniformemente distribuídos entre 0 e 1. O parâmetro α é chamado de *coeficiente de criatividade*, sendo o único parâmetro a ser especificado.

Em Sun et al. (2005) foi introduzido ao QPSO o uso do *mbest* (veja a equação 2.10), que consiste na posição média entre os valores *pbest* do enxame com M partículas e foi proposto por Shi e Eberhart (1999) para uso no PSO. A equação 2.11, descreve a movimentação das partículas influenciada pelo o vetor *mbest*.

$$mbest = (\frac{1}{M}\sum_{i=1}^{M} pbest_{i1}, \frac{1}{M}\sum_{i=1}^{M} pbest_{i2}, ..., \frac{1}{M}\sum_{i=1}^{M} pbest_{id}), \tag{2.10}$$

$$X(t+1) = P_i \pm \alpha \times |mbest - X_i(t)| \times ln(\frac{1}{u}) \tag{2.11}$$

Em Li et al. (2015), os autores alegam que a amostragem única para a nova posição de uma partícula do QPSO original pode levar a perda de informação. Para resolver esse problema, a amostragem múltipla é proposta com um mecanismo de cooperação dinâmica em um algoritmo chamado *Dynamic Context Coorperative Quantum-behaved Particle Swarm Optimization* (CCQPSO). No CCQPSO, um conjunto de posições candidatas são geradas para então produzir a nova posição da partícula. Para isso, um *vetor de contexto*, que consiste na melhor partícula do conjunto, irá cooperar com as demais posições, trocando os valores de cada coordenada e verificando possíveis melhoras. A cooperação é dita dinâmica pois o vetor de contexto é atualizado no decorrer do procedimento sempre que houver melhora. No algoritmo 9, temos o pseudocódigo do CCQPSO. Após inicializar aleatoriamente o enxame (linha 2), *maxIter* iterações são realizadas (linhas 5-27),

onde para cada partícula são feitas amostragens da equação 2.11 através do procedimento `geraPartículas` (linha 7). O número de amostragem é definido pelo parâmetro γ. A cooperação é feita entre as linhas 9 e 17, onde o procedimento *troca* substitui a k-ésima coordenada do vetor de contexto X_c^k com a mesma coordenada para uma posição candidata β_j^k. Após a obtenção da nova posição da partícula, *pbest* e *gbest* são atualizados quando necessário (linhas 19-24).

Algoritmo 9: CCQPSO (LI et al., 2015)

```
1  procedimento CCQPSO
2      X_i ← posição aleatória, ∀i ∈ {1,2,...,M}; // inicialização do enxame
3      pbest_i ← X_i, ∀i ∈ {1,2,...,M}; // pbest inicial é a primeira posição gerada
4      gbest ← melhor{X_1, X_2, ..., X_M}; // captura gbest
5      se t < maxIter então
6          para cada i = 1,...,M fazer
7              β ← geraPartículas(X_i, γ); // gera posições candidatas através da
               equação 2.11
8              X_c ← melhor{β}; // vetor de contexto é a melhor solução em β
9              para cada j = 1,...,|β| fazer
10                 se β_j == X_c então continue;
11                 para cada k = 1,...,N fazer
12                     X_aux ← troca(X_c^k, β_j^k);
13                     se f(X_aux) < f(X_c) então
14                         X_c ← X_aux; // atualiza vetor de contexto
15                     fim
16                 fim
17             fim
18             X_i ← X_c; // nova posição da partícula
19             se f(X_i) < f(pbest_i) então
20                 pbest_i ← X_i;
21             fim
22             se f(X_i) < f(gbest) então
23                 gbest ← X_i;
24             fim
25         fim
26         t ← t + 1;
27     fim
28     returne gbest;
29 fim
```

2.3 Algoritmo *k*-means

O k-means é um dos algoritmos mais conhecidos de clusterização particional. Sua popularidade se justifica pela eficiência, simplicidade e facilidade de implementação. O

método é baseado em centróide, que é o ponto geométrico médio obtido pela média dos valores nas coordenadas dos pontos de um *cluster*. Formalmente, para cada *cluster* C_i, o centróide m_i é obtido através da Equação (2.12). O algoritmo implicitamente busca uma clusterização que minimiza a soma do erro quadrático entre os objetos e o seu respectivo centróide.

$$m_i = \frac{1}{|C_i|} \sum_{p_j \in C_i} p_j \tag{2.12}$$

O seguinte pseudocódigo descreve o algoritmo:

Passo 1: Encontre um particionamento inicial aleatório. (Um método comum é escolher pontos aleatórios no espaço para fazer o papel dos centróides e definir a clusterização por atribuir a cada objeto o centróide mais próximo).
Passo 2: Calcule os centróides de cada *cluster* através da Equação (2.12) e vá para o *Passo 3*;
Passo 3: Defina o novo particionamento por atribuir, para cada objeto, o *cluster* representado pelo centróide mais próximo. Se houver mudança no particionamento, volte para o *Passo 2*, caso contrário, finalize o método com a clusterização atual;

Esse algoritmo é sensível a inicialização e pode facilmente prover soluções em mínimos locais, como reportado em Bubeck et al. (2012).

2.4 Método de Otsu

O método proposto por Otsu (1975) é uma técnica de limiarização automática que produz uma imagem binária de uma imagem em escala de cinza. O algoritmo realiza uma busca exaustiva pelo limiar ótimo com relação ao critério de variância intraclasse mínima ou, equivalentemente, variância interclasse máxima. Formalmente, dado que uma imagem precisa ser dividida em dois segmentos S_1 e S_2, por um limiar no nível t, o segmento S_1 possui os pixels com tons de cinza de 0 a $t-1$ e S_2 de t a $L-1$, para imagens com L tons de cinza. A probabilidade de cada segmento é dada por:

$$w_1 = \sum_{i=1}^{t-1} p_i, \;\; w_2 = \sum_{i=t}^{L-1} p_i \tag{2.13}$$

onde $p_i = \frac{h(i)}{N}$, sendo $h(i)$ a frequência do nível i no histograma e N o número total de pixels. A média de cada segmento é calculada como:

$$\mu_1 = \sum_{i=0}^{t-1} \frac{i \times h(i)}{w_1 \times N}, \;\; \mu_2 \sum_{i=t}^{L-1} \frac{i \times h(i)}{w_2 \times N} \tag{2.14}$$

precedendo o cálculo da média total $\mu = w_1\mu_1 + w_2\mu_2$. O método se baseia em encontrar um limiar que minimize a variância intraclasse, que é equivalente a maximizar a variância interclasse por $f(t) = w_1(\mu_1 - \mu)^2 + w_2(\mu_2 - \mu)^2$, tal que:

$$t^* = \arg\max_{0 \leq t \leq L-1} f(t) \tag{2.15}$$

O método de Otsu pode ser facilmente generalizado para uma segmentação com múltiplos limiares. A descrição matemática para encontrar k limiares e os conjuntos $S_1 = [0, ..., t_1 - 1], S_2 = [t_1, ..., t_2 - 1], ..., S_{k+1} = [t_k, ..., L - 1]$, com $f(t) = w_1(\mu_1 - \mu)^2 + w_2(\mu_2 - \mu)^2 + w_{k+1}(\mu_{k+1} - \mu)^2$ é feita da seguinte forma:

$$(t_1^*, t_2^*, ..., t_k^*) = \arg\max_{0 \leq t_1 \leq ... \leq t_k \leq L-1} f(t) \tag{2.16}$$

2.5 Análise microestrutural de sistemas cimentícios

No estudo de materiais cimentícios, o uso de imagens geradas por Microscopia de Varredura Eletrônica (MEV) em análise microestrutural é bastante difundido. O MEV é um equipamento que utiliza feixes de elétrons para fornecer informações sobre a morfologia e identificação de elementos químicos de uma amostra (DEDAVID et al., 2007). Uma das perspectivas geradas pelo MEV sobre uma amostra consiste nas Imagens por Elétrons Retro Espalhados (BEI, do inglês *Backscattered Electron Image*), que são imagens em escala de cinza com intensidade determinada principalmente por uma função do número atômico médio do local na amostra (SCRIVENER, 2004). Na Figura 2.6, temos uma BEI de uma seção polida (plana) de uma amostra de pasta de cimento hidratada em am-

pliação de 136× e 20kV de tensão de aceleração do feixe de elétrons. Pode-se associar regiões ou picos do histograma com algumas fases da amostra, tais como: o conteúdo de cimento desidratado, conteúdo de cimento hidratado, agregado (grãos de , poros e trincas. Uma segmentação eficiente dessas imagens pode fornecer uma quantificação automática aproximada do percentual das fases na amostra, podendo ser uma ferramenta útil para o estudo de propriedades do material. Além disso, a segmentação resultante das trincas e agregados podem ser utilizadas posteriormente por técnicas de detecção de rachaduras e análise de formas para extração de atributos de interesse.

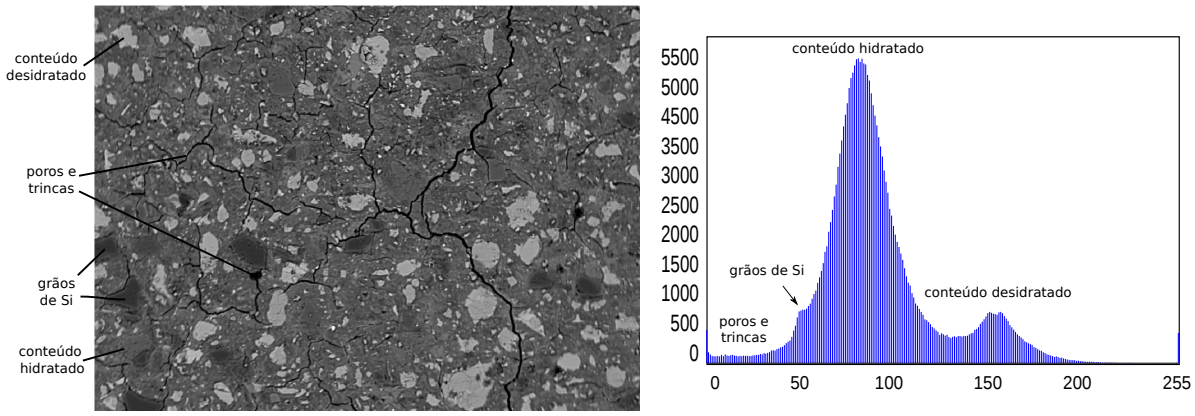


Figura 2.6: BEI de uma amostra de pasta de cimento hidratada e o seu respectivo histograma.

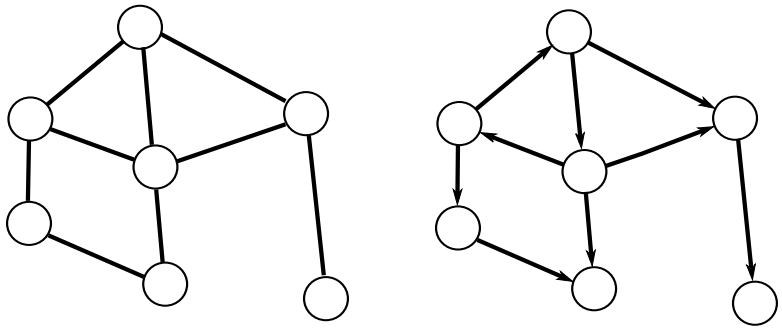
2.6 Teoria dos grafos

A Teoria dos Grafos foi primeiramente estudada por Leonhard Euler através do problema das *7 pontes de Konigsberg* e faz parte da matemática discreta, sendo uma ferramenta poderosa para a modelagem e implementação de problemas computacionais.

Um Grafo *não-orientado* (Figura 2.7(a)), é um modelo abstrato composto por nós (ou vértices) e por arestas que ligam estes nós. Ele é formalmente representado por $G = (V, E)$, onde V representa o conjunto de nós e E representa o conjunto de arestas, tal que $e = i, j \in E$ para i e $j \in V$. Em um Grafo *orientado* (Figura 2.7(b)), tem-se arcos ao invés de arestas, onde $a = (i, j) \in A$ denota um arco partindo do vértice i em direção ao vértice j.

Para um grafo $G = (V, E)$, temos que $H = (V', E')$ é um subgrafo de G se $V' \subseteq V$ e $E' \subseteq E$. O mesmo acontece para um grafo orientado.

Um grafo *completo* é aquele que possui todas as arestas possíveis para o conjunto de nós existentes. Um grafo desse tipo com cardinalidade (número de vértices) n é geralmente



(a) (b)

Figura 2.7: (a) Grafo não-orientado. (b) Grafo orientado.

chamado de K_n. A densidade de um grafo está relacionada a porcentagem de arestas existentes entre as possíveis, sendo o mesmo considerado *esparso* para uma porcentagem pequena e *denso* para uma grande porcentagem. Um grafo completo possui densidade máxima.

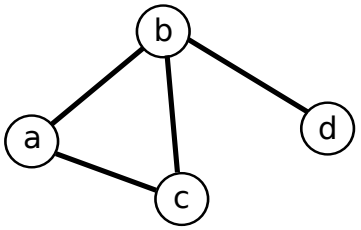
Um caminho P em um grafo é uma sequência de nós $n_0, n_1, n_2, ..., n_k$, $k \geq 1$ tal que $\{n_{i-1}, n_i\} \in E$ e nenhuma aresta ou nó é repetido na sequência. Todo subgrafo $C = (V', E')$ de um grafo $G = (V, E)$, onde existe um caminho P entre quaisquer dois nós de V' é considerado um componente conectado ou conexo de G, ou seja, C é um subgrafo onde todos os seus vértices estão conectados (direta ou indiretamente) entre si.

Ciclos são caminhos fechados sem repetição de vértices. Um grafo ou subgrafo conectado que não possui ciclos (acíclico) é denominado *árvore*. Se para uma árvore $T = (V', E')$ de um grafo $G = (V, E)$ temos que $V' = V$, então ela é uma árvore geradora de G. Um grafo acíclico que não é conectado é conhecido como *floresta*, ou seja, todos os componentes conectados são árvores.

Dois *vértices* i e j são *adjacentes* se existe uma aresta $e = \{i, j\} \in E$ entre eles. Duas *arestas* são *adjacentes* se possuem um vértice em comum, ou seja, ambas as arestas incidem em um mesmo vértice.

As maneiras mais conhecidas de se representar um Grafo são: *matriz de adjacências* e *matriz de incidências*. A matriz de adjacências, como pode ser visualizado na Figura 2.8, consiste de uma matriz de dimensões $|V| \times |V|$ que descreve a existência de uma aresta (para um grafo não-orientado) entre os vértices i e j através do elemento a_{ij}, que pode ser 1, caso a aresta exista, e 0, caso contrário.

Na matriz de incidências temos uma matriz de dimensões $|V| \times |E|$, onde cada elemento a_{ij} indica a incidência da aresta j no vértice i. Essa representação não é adequada para



	a	b	c	d
a	0	1	1	0
b	1	0	1	1
c	1	1	0	0
d	0	1	0	0

Figura 2.8: Matriz de adjacências de um grafo.

grafos com baixa densidade de arestas, devido ao desperdício de memória produzido.

Capítulo 3

REVISÃO DA LITERATURA

A literatura em clusterização de dados é extremamente ampla, devido a alta variação na definição do problema em termos de restrições, natureza dos atributos e critérios de qualidade. Portanto, nesta seção será apresentada uma breve revisão da literatura em meta-heurísticas para clusterização particional minimizando a soma das distâncias *intra-cluster*. Para segmentação de imagens, será feita uma revisão dos principais algoritmos baseados em PSO e QPSO para o critério de Otsu (apresentado na seção 2.4), além da revisão dos principais trabalhos em análise microestrutural de materiais cimentícios por segmentação de imagens retro espalhadas (veja a seção 2.5).

3.1 Clusterização de dados

Pare tratar o problema da inicialização do algoritmo k-means, muitas abordagens heurísticas tem sido propostas na literatura, com um número elevado de meta-heurísticas bio-inspiradas e populacionais. Na Tabela 3.1 é reportado, em ordem cronológica, os principais trabalhos relacionados para clusterização de dados.

Em Selim e Alsultan (1991), uma abordagem baseada em *Simulated Annealing* foi proposta para clusterização. Essa meta-heurística é uma metáfora da termodinâmica, baseada no processo de arrefecimento de sólidos na metalurgia. Um estudo da parametrização adequada do método proposto para clusterização foi realizado, porém experimentos comparativos com outros algoritmos não foram reportados.

Em Al-Sultan (1995), uma abordagem baseada em Busca Tabu foi proposta para clusterização. O algoritmo foi superior quando comparado com o k-means e o *Simulated Annealing* de Selim e Alsultan (1991) para um conjunto de teste com bases de até 89

objetos. Esse algoritmo está descrito em detalhes na seção 2.2.1.1.

Em Maulik e Bandyopadhyay (2000), um algoritmo genético (GA, do inglês *Genetic Algorithm*) é proposto para clusterizar bases de dados artificiais e do mundo real. O método se mostrou superior ao k-means em todas as bases testadas. Esse método é descrito em detalhes na seção 2.2.3.1.

Shelokar et al. (2004) propuseram uma abordagem baseada em otimização por colônia de formigas (ACO, do inglês *Ant Colony Optimization*), que é uma meta-heurística bio-inspirada no comportamento de formigas na busca por alimento. A performance do algoritmo é testada com outras meta-heurísticas populares e bons resultados foram obtidos em termos de qualidade de solução, número médio de avaliações da função objetivo e tempo de processamento requerido.

Em Kao et al. (2008), uma hibridização do PSO (veja a seção 2.2.5) com o k-means e o método Simplex de Nelder-Meadde de busca linear é proposto e comparado com outras versões do PSO e o algoritmo k-means original.

Uma abordagem baseada em otimização por colônia artificial de abelhas (ABC, do inglês *Artificial Bee Colony Optimization*) é proposta em Zhang et al. (2010). Semelhante ao ACO, o ABC é uma meta-heurística bio-inspirada no comportamento sistemático de abelhas à procura de néctar. O algoritmo utiliza o método de Deb, proposto por Goldberg e Deb (1991), para corrigir soluções inviáveis, através de regras heurísticas em um operador de seleção por torneio.

No trabalho de Niknam e Amiri (2010), um método híbrido é proposto através da combinação de uma variante do PSO chamada FAPSO (em inglês, *Fuzzy Adaptive Particle Swarm Optimization*), da ACO e do algoritmo k-means. O método demonstrou grande capacidade de resolução comparado às principais técnicas da literatura. Através da tomada de decisão inteligente proveniente da estrutura do ACO, é definido o *gbest* (veja a seção 2.2.5) de cada partícula, que é um atributo individual no FAPSO. A saída da etapa FAPSO-ACO produz os centros de *cluster* inicias para o k-means.

Uma técnica que combina CPSO (em inglês, *Chatic Map Particle Swarm Optimization*) com uma estratégia de aceleração é proposta em Chuang et al. (2011). No CPSO, a definição das variáveis aleatórias é dada por uma equação iterativa inspirada na teoria do caos, com o intuito de aumentar a capacidade do método em escapar de ótimos locais.

Cura (2012) alega que poucos trabalhos estudaram a aplicação do PSO em sua versão original para clusterização de dados. O autor propõe um algoritmo PSO que utiliza a

topologia do gbest na vizinhança, onde cada partícula move em direção a sua melhor posição e em direção a melhor posição obtida pela sua vizinhança.

Hatamlou et al. (2012) propuseram uma abordagem combinada entre o k-means e o algoritmo de busca gravitacional (GSA, do inglês *Gravitational Search Algorithm*). O mecanismo do GSA é inspirado na interação de massas no universo pela lei da gravidade, guiando a movimentação de partículas com atração mútua.

Em Hatamlou (2013), uma nova meta-heurísticas chamada Buraco negro (BH, do inglês *Black hole*) foi proposta para clusterização de dados. Esse método é inspirado no fenômeno de buracos negros, onde o método seleciona a melhor solução candidata para ser o buraco negro que puxa outros candidatos chamados de estrelas. Os experimentos demostram que o BH supera outras heurísticas tradicionais.

Em Krishnasamy et al. (2014), é proposta uma abordagem híbrida entre o k-means e inteligência de grupos (em inglês, *cohort intelligence*), que é uma método inspirado em tendências sociais onde grupos de indivíduos aprendem uns com os outros para resolução de um problema.

Recentemente, Yeh e Lai (2015) propuseram um método baseado em PSO combinado com uma busca de vibração variável (VVS, do inglês *variable vibrating search*) e uma estratégia centralizada rápida (RCS, do inglês *rapid centralized strategy*). O VVS é um esquema que busca refinar soluções por buscar valores extremos perto do *gbest*. O RCS acelera a taxa de convergência através do uso de média aritmética.

Shabanzadeh e Yusof (2015) propuseram uma heurística chamada Otimização Baseada em Biogeografia (BBO, do inglês *Biogreography-Based Optimization*) que é inspirada na distribuição geográfica natural de diferentes espécies. Além das bases de dados tradicionais, conjunto de bases médicas foram utilizadas nos experimentos.

Pakrashi e Chaudhuri (2016) propuseram uma abordagem híbrida que combina o Algoritmo heurístico de Kalman (HKA, em inglês *Heuristic Kalman Algorithm*) e o k-means. O HKA é uma meta-heurística populacional que usa a geração de número aleatórios com distribuição de probabilidade Gaussiana para encontrar pontos no espaço de busca.

Wang et al. (2016) propuseram uma abordagem chamada *Flower Pollination Algorithm with Bee Pollinator* (FPABP). Esse método é inspirado no processo de polinização de flores através de abelhas. O algoritmo obteve sucesso quando comparado com outros algoritmos de inteligência de enxame tais como PSO, *Differential Evolution* (DE) e *Artificial Bee Colony* (ABC).

Tabela 3.1: Trabalhos relacionados na literatura para clusterização de dados.

Abordagem	Autor(es)
SA	Selim e Al-Sultan (1991)
TS	Al-Sultan (1995)
GA	Murthy e Chowdhury (1996)
GA + KM	Krishna e Murty (1999)
GA	Maulik e Bandyopadhyay (2000)
ACO	Shelokar et al. (2004)
PSO	Kao et al. (2008)
ABC	Zhang et al. (2010)
PSO + ACO + KM	Niknam e Amiri (2010)
ABC	Karaboga e Ozturk (2011)
PSO	Chuang et al. (2011)
PSO	Cura (2012)
BH	Hatamlou (2013)
CI + KM	Krishnasamy et al. (2014)
PSO	Yeh e Lai (2015)
HKA	Pakrashi e Chaudhuri (2016)
FPA	Wang et al. (2016)

3.2 Segmentação de Imagens

3.2.1 Métodos baseados em PSO e QPSO para limiarização Otsu

Lin et al. (2008) propuseram um PSO melhorado para limiarização simples baseado na movimentação das partículas sem considerar uma função de custo. O algoritmo foi comparado com o método de Otsu e outras técnicas conhecidas.

Em Wei e Kangling (2008), um esquema híbrido baseado em PSO e mapeamento caótico chamado SAPSO (do inglês, *Self-adaptive Particle Swarm Optimization*), é proposto para limiarização multinível com critério de Otsu.

Djerou et al. (2009) propuseram um método chamado DMTBPSO (do inglês, *Dynamic Multilevel Thresholding Binary Particle Swarm Optimization*) que determina o número e os valores dos limiares simultaneamente. Essa versão do PSO é adaptada ao espaço binário, restringindo as coordenadas em {0,1}. Dessa forma, cada partícula armazena uma variável binária para cada nível de cinza, e que define a sua inclusão.

Hammouche et al. (2010) realizaram um estudo comparativo entre meta-heurísticas para limiarização multinível com critério de Otsu. Entre os métodos testados, o GA, PSO e DE (do inglês, *Differential Evolution*) se mostraram superiores em termos de precisão, robustez e tempo de convergência.

Duraisamy et al. (2010) propuseram um PSO simples para limiarização multinível com critério de Otsu e um outro critério. Em comparação com um GA, o PSO conseguiu resultados superiores.

Akay (2013) comparou o PSO com o algoritmo ABC, em suas versões originais. Experimentos demostraram igualdade para dois limiares e vantagem do ABC para um número maior de limiares.

Em Dey et al. (2014), seis diferentes abordagens inspiradas na mecânica quântica foram propostas para diferentes meta-heurísticas, incluindo um algoritmo QPSO. Como resultado de experimentos estatísticos sofisticados, o QPSO se mostrou superior que as demais hibridizações.

Recentemente, Liu et al. (2015) propuseram um método chamado MPSO (do inglês, *Modified Particle Swarm Optimization*) que foi proposto para superar problemas provenientes do PSO. Esse método emprega duas estratégias chamadas inércia adaptativa (IA) e população adaptativa (PA), a primeira busca variar o peso inercial, e a segunda variações

no tamanho do enxame. Os resultados confirmam a hipótese inicial, onde o MPSO consegue melhores resultados que o PSO original, além de vencer um algoritmo genético padrão (SGA).

3.2.2 Segmentação de imagens BEI em sistemas cimentícios

Scrivener (1986) foi um dos primeiros trabalhos a investigar o uso de imagens BEI para análise microestrutural de cimento hidratado, aplicando análise de imagem simples para quantificação de fases em amostras.

Yang e Buenfeld (2001) propuseram a segmentação binária de agregados por operações morfológicas e limiarização simples manual.

Para avaliar o estado de hidratação de amostras, Mouret et al. (2001) utilizou limiarização simples com maximização da entropia para quantificar a fase com partículas desidratadas de cimento.

Em Scrivener (2004), o potencial de estudos de durabilidade de concreto por imagens BEI através de análise de imagens é discutido. Também é feita uma discussão sobre os níveis de cinza relacionados as fases do material e a técnicas de quantificação baseadas em limiarização no histograma.

Igarashi et al. (2004) utilizaram limiarização dinâmica baseada em informações de contraste de pixels vizinhos para obter a fração de área das fases de cimento desidratado e porosidade.

Head e Buenfeld (2006) apresenta uma segmentação precisa de partículas de agregado em BEI de concreto endurecido, através de um conjunto semiautomático de operações em imagens, entre elas, operações de limiarização simples manual.

Em Wong et al. (2006), uma técnica de segmentação de poros em BEI para materiais baseados em cimento é proposta. Essa técnica de limiarização simples, encontra um ponto de inflexão no histograma baseado no vazamento de pixels com o aumento do valor do limiar. O método foi comparado com outras técnicas, tais como limiarização por máxima entropia e picos entre pontos de mínimo.

Deschner et al. (2013) propuseram a segmentação de BEI de pasta hidratada de cimento Portland misturado com cinza volante. Uma limiarização múltipla é realizada de forma semiautomática, através de pontos de mínimo, máximo e bases de pico no histograma.

Capítulo 4

MÉTODOS PROPOSTOS

Esta seção apresenta em detalhes os métodos propostos para resolver os problemas de clusterização de dados e segmentação de imagens. A leitura das seções 2.2.4 e 2.2.5 é recomendada para o entendimento pleno desta seção.

4.1 *Continuous* GRASP para clusterização de dados

Em suas versões originais (descritas na seção 2.2.4), o C-GRASP demonstrou alta performance quando submetido a experimentos com funções multimodais de baixa dimensionalidade. Porém, esse desempenho pode decair em tarefas de clusterização, que possuem alta variabilidade na dimensão do problema. Isso acontece devido ao valor inicial do tamanho do passo h, que possui grande influência na capacidade de exploração do espaço de busca com diferentes ordens de magnitude para cada atributo (dimensão). Dessa forma, é proposto ajustar esse valor de forma proporcional à instância. Além disso, algumas modificações semelhantes aos melhoramento incorporados no *new* C-GRASP (veja a seção 2.2.4.1), foram adicionadas. Mais precisamente, um valor mínimo para h é utilizado como critério de parada (análogo ao uso do parâmetro h_e no *new* C-GRASP) e a redução de h a cada iteração (semelhante ao *new* C-GRASP, onde isso é feito a cada iteração sem melhora). O algoritmo proposto será referido durante o restante do trabalho como `C-GRASP-Clu`.

A dimensão do problema de clusterização no `C-GRASP-Clu` é dada por $n = k \times d$, onde k é o número de *clusters* e d é a dimensão de cada elemento no conjunto de dados. Dessa forma, cada ponto $x \in S$ (S é o espaço de busca definido na seção 2.2.4) representa uma solução que armazena as coordenadas dos centros de *cluster*, como exemplificado na Figura 4.1(a), onde c_i^j é a j-ésima coordenada do i-ésimo centro de *cluster*.

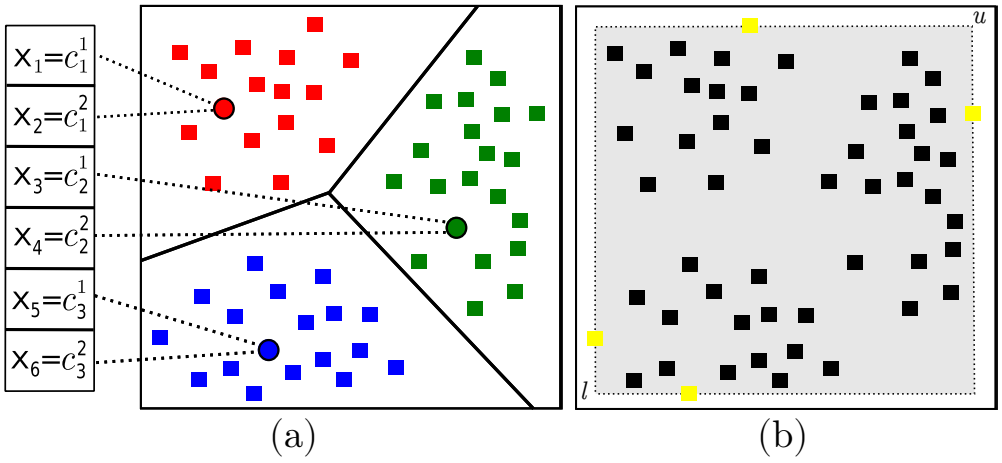


Figura 4.1: Exemplo envolvendo três *clusters* no espaço Euclidiano bidimensional. a) Representação da solução. b) Definição dos vetores limitantes l e u através dos valores extremos nos pontos.

O pseudocódigo do `C-GRASP-Clu` é apresentado no Algoritmo 10. Além dos parâmetros $f(.)$ e *MaxDirToTry* (herdados do C-GRASP), temos D como o conjunto de dados, d como a dimensão de cada elemento, k como o número de clusters, I_s como o número de tentativas na etapa de geração de uma solução inicial, bem como Δ_1 e Δ_2 que controlam um mecanismo de filtro de iterações. Os vetores l e u são definidos através dos valores extremos existentes em D para todas os eixos (linhas 3-6), como ilustrado na Figura 4.1(b). Dessa forma, cada centro de *cluster* deve respeitar a região delimitada por l e u. Finalmente, h_s e h_e determinam o valor inicial e mínimo para tamanho do passo h, respectivamente, como no *new* C-GRASP. O cálculo de h_s e h_e é baseado em $\mu = \{\bar{p}_1, \bar{p}_2, \ldots, \bar{p}_d\}$ (linha 7), onde $\bar{p}_i$ é a média aritmética dos valores no i-ésimo atributo (coordenada) de todos os elementos em D.

O algoritmo é executado até que as condições de término sejam satisfeitas (linhas 9-36). A cada iteração, a sub-rotina `SoluçãoInicial` gera uma solução inicial x (linha 10). Esta etapa é baseada no esquema de inicialização do k-means proposto por MacQueen et al. (1967) que consiste em selecionar aleatoriamente k objetos de D como centros de *cluster* e, a partir da clusterização resultante, calcular os centróides (média aritmética dos elementos em cada cluster, como definido na seção 2.3). Após I_s tentativas, os melhores centróides encontrados são retornados como solução inicial.

Em seguida, os procedimentos `ConstruçãoMelhorada` (Algoritmo 7) e `BuscaLocal` (Algoritmo 5) são chamados iterativamente. No `C-GRASP-Clu`, tais procedimentos avaliam a expressão $l \leq x \leq u$ como uma verificação individual de cada centro em x, uma vez que esses vetores possuem dimensões distintas ($l, u \in \mathbb{R}^d$ e $x \in \mathbb{R}^{k \times d}$). Na primeira etapa

(linhas 13-26), a fase de construção é executada múltiplas vezes para h decrescendo de h_s até h_e. Em caso de melhoramento, a melhor solução é atualizada (linha 16). Um mecanismo de filtro é ativado após o primeiro melhoramento local imp_l (linha 19) para evitar iterações não promissoras. Este mecanismo é baseado na observação empírica de que o percentual de melhoramento local tende a decair gradualmente a cada iteração, principalmente devido a redução de h. Dessa forma, quando o *gap* percentual, dado pela equação (4.1), entre f_{prev} (qualidade da solução anterior) e $f(x)$ for menor do que Δ_1 e o *gap* percentual de $f(x)$ para f^* for superior a Δ_2, caracteriza-se uma iteração com baixa probabilidade de melhora e a iteração atual é abortada na linha 22. Em seguida, se x^* foi atualizado, ou seja, imp_g está ativado, a busca local é realizada para refinamentos em pequena escala (linhas 29-35).

$$gap(f_1, f_2) = \left(\frac{f_1 - f_2}{f_1} \times 100 \right) \tag{4.1}$$

Um dos objetivos das modificações citadas anteriormente é evitar que a busca local seja executada com h pequeno para uma solução distante do ótimo local. Isso poderia provocar uma execução muito extensa da busca local, pois consecutivos melhoramentos de pequena escala seriam realizados sem que o ótimo local fosse atingido e as condições de término nunca seriam satisfeitas (número de iterações sem melhora).

Algoritmo 10: Pseudocódigo do algoritmo `C-GRASP-Clu`.

```
1  procedimento C-GRASP-Clu (D, d, k, f(.), I_s, MaxDirToTry, Δ_1, Δ_2)
2      n ← k × d;
3      para cada i = 1, ..., d fazer
4          l(i) ← min{p(i) | p ∈ D};
5          u(i) ← max{p(i) | p ∈ D};
6      fim
7      h_s ← max(μ); h_e ← min(μ)/10^4;
8      x* ← nil; f* ← ∞;
9      enquanto Critérios de parada não satisfeitos faça
10         x ← SoluçãoInicial(I_s, D);
11         h ← h_s;
12         f_prev ← ∞; imp_l ← false; imp_g ← false;
13         enquanto h ≥ h_e faça
14             x ← ConstruçãoMelhorada(x, f(.), n, h, l, u);
15             se f(x) < f* então
16                 x* ← x; f* ← f(x); imp_g ← true
17             fim
18             se ¬ imp_l ∧ f_prev ≠ ∞ ∧ (f_prev − f(x)) ≥ 0 então
19                 imp_l ← true
20             fim
21             se imp_l ∧ gap(f_prev, f(x)) < Δ_1 ∧ gap(f(x), f*) > Δ_2 então
22                 Go to linha 9;
23             fim
24             f_prev ← f(x);
25             h ← h/2;
26         fim
27         se ¬ imp_g então Go to linha 9;
28         h ← 1;
29         enquanto h ≥ h_e/10 faça
30             x ← BuscaLocal(x*, f(.), n, h, l, u, MaxDirToTry);
31             se f(x) < f* então
32                 x* ← x; f* ← f(x);
33             fim
34             h ← h/2;
35         fim
36     fim
37     returne x*;
38 fim
```

4.2 Abordagem híbrida baseada em CCQPSO e VND para segmentação de imagens

Para limiarização multinível com critério de Otsu, cada partícula do enxame codifica um conjunto de limiares candidatos e o valor da função objetivo ou *fitness* é calculado pela Equação (2.16) apresentada na seção 2.4. A abordagem proposta utiliza como base a versão do QPSO com cooperação dinâmica chamada `CCQPSO`, apresentada em pseudocódigo no Algoritmo 9 da seção 2.2.5.1.

Para acelerar a convergência e melhorar a capacidade de intensificação do `CCQPSO`, uma busca local baseada em *Variable Neighborhood Descent* (VND) é proposta. As estruturas de vizinhança $N = \{N^{(1)}, N^{(2)}, ..., N^{(max_n)}\}$ definem um conjunto de soluções vizinhas pela soma do valor absoluto das diferenças para os k limiares da partículas X, como descrito em (4.2). O pseudocódigo do VND é apresentado no Algoritmo 11. A abordagem híbrida proposta, chamada de `CCQPSO+VND` (Algoritmo 12), consiste basicamente em executar o VND, com $max_n = 10$, sempre que houver uma atualização na partícula *gbest* (linha 23).

$$N^{(R)} = \{T \in \{0, L-1\}^k \mid \sum_{i=1}^{k} |X(i) - T(i)| = R\} \tag{4.2}$$

Algoritmo 11: VND para limiarização multinível.

```
1  procedimento VND(X, max_n)
2      T* ← X; f* ← f(X);
3      s ← 1; // índice da estrutura de vizinhança
4      enquanto s ≤ max_n faça
5          // número de vizinhos analisados cresce por um fator de 10
6          para cada j = 1, ..., 10 × s fazer
7              T ← selecionaVizinhoAleatoriamente(N^(s));
8              se f(T) > f* então
9                  T* ← T; f* ← f(T);
10                 s ← 1; // reinicia índice da estrutura de vizinhança
11                 Vá para a linha 4;
12             fim
13         fim
14         s ← s + 1; // avança para a próxima estrutura de vizinhança
15     fim
16     returna T*;
17 fim
```

Algoritmo 12: CCQPSO+VND

```
1  procedimento CCQPSO+VND
2      X_i ← posição aleatória, ∀i ∈ {1,2,...,M}; // inicialização do enxame
3      pbest_i ← X_i, ∀i ∈ {1,2,...,M}; // pbest inicial é a primeira posição gerada
4      gbest ← melhor{X_1, X_2, ..., X_M}; // captura gbest
5      se t < maxIter então
6          para cada i = 1,...,M fazer
7              β ← geraParticulas(X_i, γ); // gera posições candidatas através da
               equação 2.11
8              X_c ← melhor{β}; // vetor de contexto é a melhor solução em β
9              para cada j = 1,...,|β| fazer
10                 se β_j == X_c então continue;
11                 para cada k = 1,...,N fazer
12                     X_aux ← troca(X_c^k, β_j^k);
13                     se f(X_aux) < f(X_c) então
14                         X_c ← X_aux; // atualiza vetor de contexto
15                     fim
16                 fim
17             fim
18             X_i ← X_c; // nova posição da partícula
19             se f(X_i) < f(pbest_i) então
20                 pbest_i ← X_i;
21             fim
22             se f(X_i) < f(gbest) então
23                 X_i ← VND(X_i, max_n); // executa busca local no novo gbest
24                 gbest ← X_i;
25             fim
26         fim
27         t ← t + 1;
28     fim
29     returne gbest;
30 fim
```

Capítulo 5

EXPERIMENTOS COMPUTACIONAIS E RESULTADOS

Nesta seção, os experimentos computacionais envolvendo os algoritmos propostos `C-GRASP-Clu` e `QPSO+VND`, bem como os resultados obtidos serão descritos e discutidos em detalhes.

5.1 `C-GRASP-Clu`

5.1.1 Configuração do ambiente de teste

Todos os algoritmos foram codificados em Matlab 7.14 (R2012a) e executados em um PC Intel Core i7-2600 com 3.40 GHz e 16 GB de RAM em sistema operacional Ubuntu 12.04 LTS (64 bits). A análise estatística foi realizada através do pacote *scmamp* (CALVO; SANTAFE, 2015) para o software R 3.2.0.

O conjunto de instâncias de teste utilizado nos experimentos é composto por 10 bases de dados de propósito geral. As principais características das bases são apresentadas na Tabela 5.1. As bases *iris*, *wine*, *cancer*, *cmc* e *glass* foram obtidas do repositório de aprendizado de máquina da University of California, Irvine (UCI) (MERZ; BLAKE, 2016). As bases *vowel* e *crudeoil* são originadas de Pal e Majumder (1977) e Gerrild e Lantz (1969), respectivamente. Duas bases artificiais, *artset1* e *artset2*, foram geradas como na literatura (NIKNAM; AMIRI, 2010; WANG et al., 2016).

Tabela 5.1: Principais características das bases de dados utilizadas nos experimentos.

Base de dados	Nº de atributos (d)	Nº de *clusters* (k)	Nº de objetos (m)
Fisher's Iris (*iris*)	4	3	150
Wine (*wine*)	13	3	178
Indian Telugu Vowel Sounds (*vowel*)	3	6	871
Contraceptive Method Choice (*cmc*)	9	3	1473
Breast Cancer Wisconsin (*cancer*)	9	2	683
Glass Identification (*glass*)	9	6	214
Crude Oil (*crudeoil*)	5	3	56
Thyroid Disease (*thyroid*)	5	2	215
Artificial Set One (*artset1*)	3	5	250
Artificial Set Two (*artset2*)	2	4	600

5.1.2 Calibração de parâmetros

Para encontrar uma configuração interessante dos parâmetros do C-GRASP-Clu, um projeto experimental fatorial foi elaborado. Neste experimento, 2^p configurações foram avaliadas para p parâmetros em níveis mínimo e máximo. Para I_s, $MaxDirToTry$ (γ), Δ_1 e Δ_2, existem $2^4 = 16$ possíveis configurações. Para cada configuração, a meta-heurística foi executada 10 vezes nas bases *glass* e *vowel*, que estão entre as mais difíceis do conjunto de teste. Foram armazenados os valores médios da função objetivo em diferentes instantes de tempo: 1, 10, 30 e 60 segundos. Os *gaps* percentuais com respeito a melhor solução conhecida obtidos por cada configuração em cada instante de tempo são reportados na Tabela 5.2. Os resultados mostram que nenhuma das configurações obtiveram vantagem significante sobre as demais, sugerindo que o C-GRASP-Clu não parece ser altamente sensível aos valores dos parâmetros de entrada. Devido a relativamente favorável performance após um segundo de execução, foi escolhida a seguinte configuração: $I_s = 100$, $\gamma = 30$, $\Delta_1 = 0,1$ e $\Delta_2 = 0,5$.

Tabela 5.2: Calibração de parâmetros para as bases *vowel* e *glass*. O *gap* médio para a melhor solução conhecida em 1, 10, 30 e 60 segundos.

Dataset	I_s	γ	$\Delta_1 = 0.05\%$		$\Delta_1 = 0.1\%$	
			$\Delta_2 = 0.5\%$	$\Delta_2 = 1\%$	$\Delta_2 = 0.5\%$	$\Delta_2 = 1\%$
vowel	20	30	18,5 0,4 0,0 0,0	18,4 0,4 0,0 0,0	18,7 0,3 0,1 0,0	18,2 0,0 0,0 0,0
		50	17,8 0,3 0,0 0,0	16,5 0,2 0,0 0,0	17,3 0,4 0,0 0,0	18,7 0,2 0,0 0,0
	100	30	17,1 0,2 0,0 0,0	18,8 0,3 0,0 0,0	18,1 0,2 0,0 0,0	17,7 0,5 0,0 0,0
		50	16,4 0,1 0,0 0,0	18,1 0,2 0,0 0,0	18,2 0,3 0,0 0,0	17,5 0,4 0,0 0,0
glass	20	30	4,1 1,2 0,5 0,1	4,0 1,0 0,5 0,1	4,4 1,5 0,6 0,0	5,2 2,0 1,2 0,1
		50	4,4 1,6 0,8 0,1	4,2 1,5 0,4 0,1	5,0 1,9 0,8 0,0	5,2 1,7 1,0 0,2
	100	30	4,2 1,6 1,0 0,2	4,3 1,5 0,8 0,0	3,7 1,2 0,5 0,0	4,3 1,4 0,8 0,1
		50	5,5 2,3 1,5 0,3	4,4 2,0 0,7 0,2	3,8 1,5 0,8 0,1	5,0 1,9 1,0 0,2

5.1.3 Análise de performance

O objetivo deste primeiro experimento é comparar a performance do `C-GRASP-Clu` com meta-heurísticas conhecidas da literatura, bem como o algoritmo k-means. Para isso, foram implementados os seguintes métodos:

- Algoritmo k-means.
- Busca Tabu (`TS`, em inglês *Tabu Search*) proposta por Al-Sultan (1995) (descrito na seção 2.2.1.1). Foi atribuído $0,99$ para o limiar de probabilidade, $|D|$ para o tamanho máximo da lista tabu e 20 para o número de vizinhos. Além disso, uma vez que os centróides são determinados, a clusterização é redefinida apenas antes de computar o valor da função objetivo, ao contrário do que é feito em Al-Sultan (1995).
- Algoritmo Genético (`GA`, em inglês *Genetic Algorithm*) proposto em Maulik e Bandyopadhyay (2000) (descrito na seção 2.2.3.1) e com os mesmo valores de parâmetros adotados pelos autores: tamanho da população $P = 100$, probabilidade de cruzamento $\mu_c = 0,8$ e probabilidade de mutação $\mu_m = 0,001$.
- Otimização por Enxame de Partículas (`PSO`, em inglês *Particle swarm optimization*) proposto em (CHEN; YE, 2004). Os autores basicamente aplicaram o PSO básico, codificando em cada partícula os centros de *cluster* candidatos. Em vez dos valores de parâmetros originais, foram utilizados: tamanho do enxame 40, coeficientes de aceleração $c_1 = c_2 = 1,5$ e peso inercial decaindo uniformemente de 0,4 até 0,1.

- C-GRASP (HIRSCH et al., 2007) com os valores originais de parâmetros: $h = 1$, $\alpha = 0,4$, MaxDirToTry = 30, MaxNumIterNoImprov = 20, MaxIters = 200, NumTimesToRun = 20.

É necessário mencionar que todas os valores de parâmetros selecionados no TS e PSO provocaram uma melhoria de desempenho em tais métodos, em comparação com os valores originalmente utilizados.

Para cada algoritmo, foram realizadas 50 execuções independentes em cada base de dados, sendo registrada a soma das distâncias *intracluster* obtida (equação (1.1)). O melhor, médio e pior valor, além do desvio padrão, são reportados na Tabela 5.3. Como pode-se notar facilmente, o C-GRASP-Clu atingiu resultados de melhor qualidade em todas as bases para quase todos os critérios considerados, com exceção apenas da melhor solução nas bases *vowel* e *artset1*. Destaca-se a robustez do algoritmo proposto, uma vez que o desvio padrão obtido é quase zero para todas as bases, em contraste com as demais abordagens. O algoritmo k-means gerou os piores resultados para alguns dos critérios, com destaque para as bases *wine*, *crudeoil* e *thyroid*. Apesar disso, o algoritmo conseguiu valores razoáveis nas demais bases. O TS obteve desempenho muito ruim para as bases *vowel*, *cmc* e *cancer*, apesar de atingir resultados competitivos com baixo desvio padrão na demais bases. O PSO conseguiu a melhor solução para 6 das 10 bases, contabilizando os empates com outros algoritmos. Contudo, o método obteve o maior desvio padrão para quase todas as bases, além de produzir soluções de baixa qualidade na média. O GA, apesar de não conseguir o melhor resultado para nenhum critério, mostrou-se bastante competitivo e robusto. Finalmente, o C-GRASP conseguiu resultados semelhantes ao C-GRASP-Clu para as bases *iris*, *cancer* e *thyroid*. Porém, nas demais instâncias, o seu desempenho é nitidamente inferior em qualidade e robustez, evidenciando a contribuição dos mecanismos incorporados na abordagem proposta.

As Figuras 5.1 e 5.2 mostram as curvas de convergência média sobre o tempo de execução. É possível verificar que o C-GRASP-Clu domina todas os outros algoritmos em quase todas os instantes de tempo nas 10 bases consideradas. Essa observação justifica a escolha do algoritmo para aplicações envolvendo diferentes restrições de tempo de execução.

Tabela 5.3: Resultados obtidos pelos algoritmos implementados nas 10 bases de dados consideradas.

Base	Critério	k-means	TS	PSO	GA	C-GRASP	C-GRASP-Clu
iris	Melhor	97,34622	96,78150	**96,65548**	96,84127	96,65552	**96,65548**
	Média	103,43531	96,88615	107,78288	96,90238	96,65560	**96,65548**
	Pior	128,40419	96,97888	128,99037	97,01209	96,65567	**96,65548**
	Desvio P.	12,32979	0,04439	12,88245	0,04304	0,00003	$< 10^{-7}$
wine	Melhor	16555,67942	16293,51381	16292,72117	16298,86886	16292,18477	**16292,18464**
	Média	16555,67942	16293,76674	16320,21929	16310,65666	17084,20347	**16292,18464**
	Pior	16555,67942	16293,96034	16602,94838	16312,76275	18598,74279	**16292,18464**
	Desvio P.	0	0,10027	49,58481	3,33347	863,74511	$< 10^{-9}$
vowel	Melhor	149433,72050	184767,35474	**148967,24081**	149252,53673	**148967,24081**	148967,24093
	Média	160223,93546	186995,35821	156485,15098	149318,20417	150763,35701	**148967,24123**
	Pior	182350,24215	189335,67270	183624,67374	149377,30589	168340,28679	**148967,24157**
	Desvio P.	8783,33816	1079,49332	7636,89499	24,99493	4076,18376	0,00014
cmc	Melhor	5542,18214	6477,33093	5532,18479	5539,20866	5532,19087	**5532,18472**
	Média	5542,78752	6583,22006	5604,21176	5540,21830	5621,34653	**5532,18472**
	Pior	5545,33337	6658,22038	5929,21048	5540,67773	7018,04747	**5532,18472**
	Desvio P.	1,20787	37,37007	88,30712	0,40513	356,45274	$< 10^{-9}$
cancer	Melhor	2986,96134	3190,97306	2974,13324	2969,70014	2964,38711	**2964,38697**
	Média	2987,07866	3213,16391	3187,30793	2971,16976	2964,38725	**2964,38697**
	Pior	2988,42780	3231,88984	3829,70040	2971,96160	2964,38743	**2964,38697**
	Desvio P.	0,40188	10,05480	227,33759	0,48490	0,00007	$< 10^{-6}$
glass	Melhor	215,47041	214,72857	236,13250	211,96856	210,42909	**210,00104**
	Média	222,49235	215,39118	278,75363	212,53390	228,04800	**210,00104**
	Pior	260,19479	219,45732	369,60045	212,99928	255,44523	**210,00104**
	Desvio P.	6,98161	0,67156	26,74067	0,18541	11,89373	$< 10^{-8}$
crudeoil	Melhor	279,27099	277,74179	**277,21073**	278,03081	277,21074	**277,21073**
	Média	281,20142	278,17657	278,26177	278,56051	277,59669	**277,21073**
	Pior	355,40633	278,77052	289,21476	278,84239	293,93768	**277,21073**
	Desvio P.	10,70904	0,23474	2,43474	0,19768	2,35900	$< 10^{-9}$
thyroid	Melhor	2334,47554	2225,14809	**2155,61791**	2171,30633	2155,61792	**2155,61791**
	Média	2335,86947	2225,17001	2267,49204	2193,91762	2155,61793	**2155,61791**
	Pior	2340,41916	2225,22310	2704,30181	2218,50331	2155,61794	**2155,61791**
	Desvio P.	2,35974	0,01305	150,51441	12,83118	$< 10^{-5}$	$< 10^{-7}$
artset1	Melhor	1836,24687	1835,91563	**1831,11531**	1835,28400	1831,11532	1831,11532
	Média	1860,00031	1836,86426	2377,92209	1835,78010	2463,81834	**1831,11534**
	Pior	2437,61251	1838,12592	3017,35924	1836,07545	3017,36758	**1831,11542**
	Desvio P.	117,55890	0,54312	294,70857	0,23767	291,61102	0,00001
artset2	Melhor	545,64185	655,8450	**545,03769**	545,19239	545,03775	**545,03769**
	Média	580,43200	702,58027	563,75709	545,31319	551,64447	**545,03769**
	Pior	893,49686	775,35296	875,20532	545,38472	875,24286	**545,03825**
	Desvio P.	105,41441	25,43743	74,91211	0,04104	46,69767	0,00007

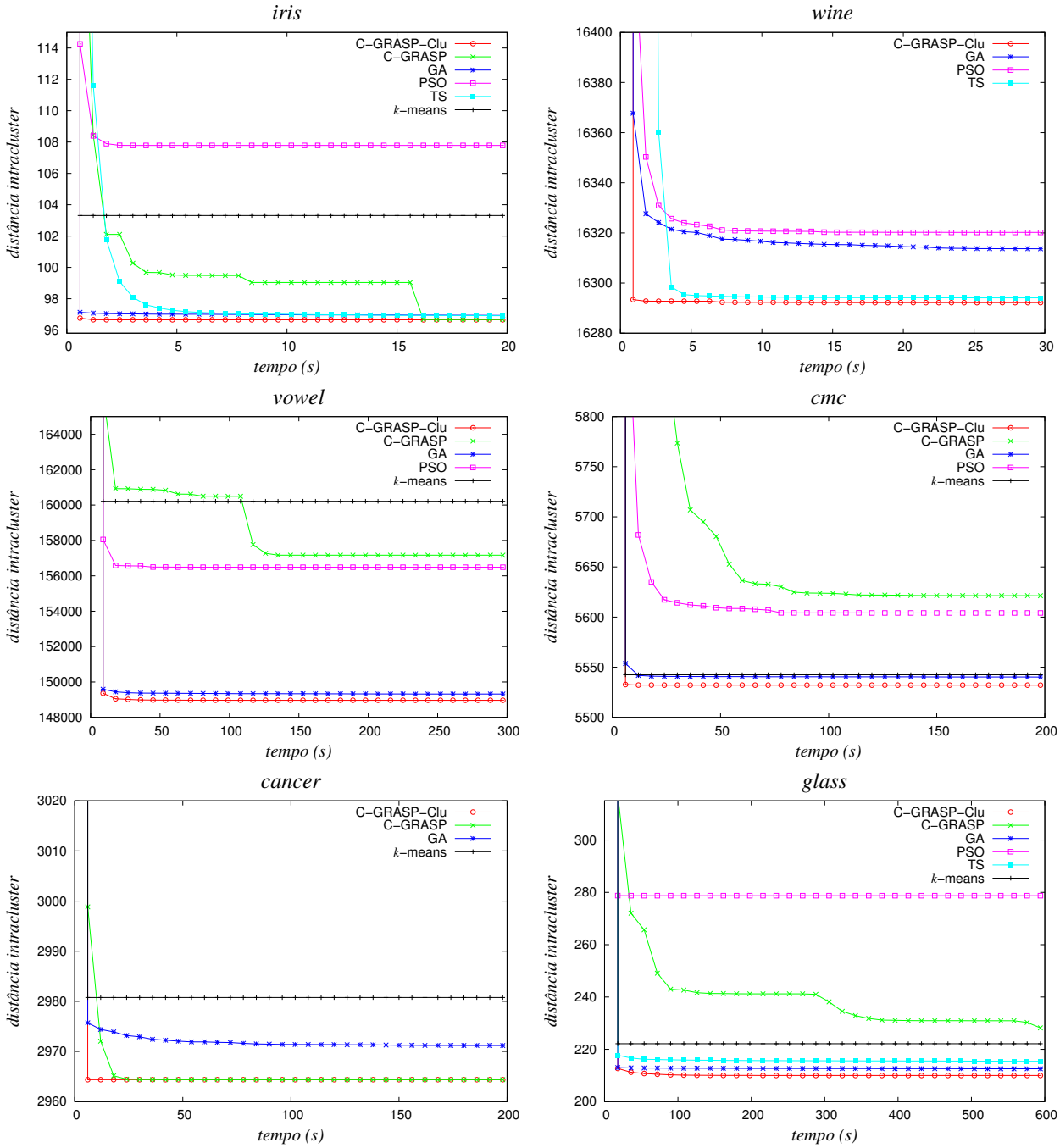


Figura 5.1: Curvas de convergência média dos algoritmos implementados para as bases *iris*, *wine*, *vowel*, *cmc*, *cancer* e *glass*. Foram omitidas as curvas com performance significantemente baixa para facilitar a análise das demais.

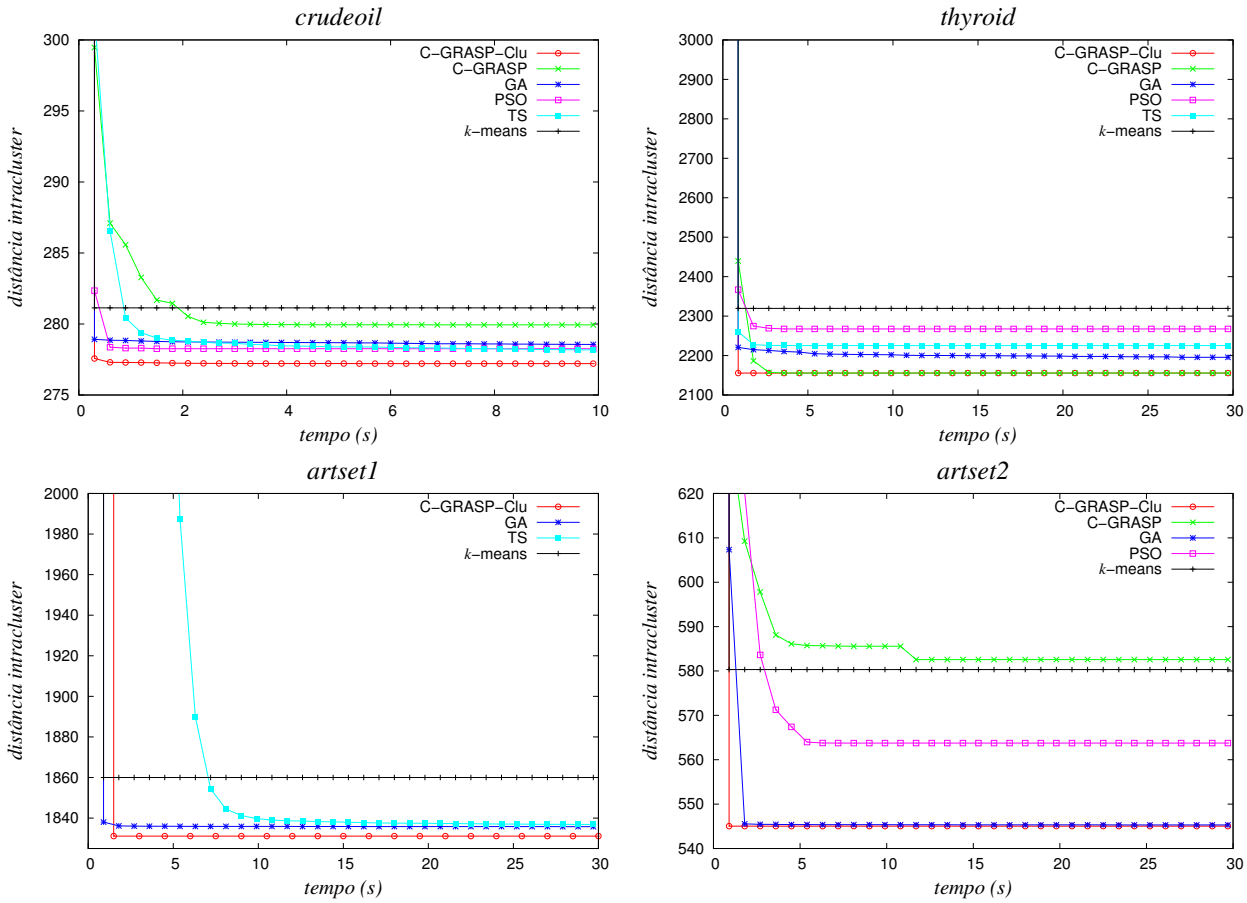


Figura 5.2: Curvas de convergência média dos algoritmos implementados para as bases *crudeoil*, *thyroid*, *artset1*, *artset2*. Foram omitidas as curvas com performance significantemente baixa para facilitar a análise das demais.

5.1.3.1 Testes estatísticos

Para fortalecer a análise de performance, foram conduzidos testes estatísticos não paramétricos. A hipótese nula assume que os algoritmos tem performance média igual e as diferenças observadas são aleatórias. O teste de *Friedman* é um procedimento não paramétrico que busca detectar se, em um conjunto de amostras, pelo menos duas delas representam populações com diferente média. Essa estatística é baseada nos *ranks* definidos pela qualidade das amostras, que no caso estudado corresponde a qualidade das soluções obtidas pelos algoritmos em cada instância do problema. A partir da performance média reportada na Tabela 5.3, os algoritmos receberam *ranks* de 1 até 5 para cada base, e o *rank* médio foi atribuído em caso de empate (por exemplo, se dois algoritmos estão empatados como melhor, é atribuído *rank* $(1 + 2)/2 = 1,5$ para ambos). Os *ranks* são apresentados na Tabela 5.4. O valor da estatística de *Friedman* obtido foi de $F_f = 28,686$ com valor-p = 0,00002, rejeitando a hipótese nula com nível de significância de $\alpha = 0,01$.

Tabela 5.4: Ranks baseados na perfomance média dos algoritmos.

Base	k-means	TS	PSO	GA	C-GRASP	C-GRASP-Clu
iris	5	3	6	4	2	1
wine	5	2	4	3	6	1
vowel	5	6	4	2	3	1
cmc	3	6	4	2	5	1
cancer	4	6	5	3	2	1
glass	4	3	6	2	5	1
crudeoil	6	4	3	5	2	1
thyroid	6	4	5	3	2	1
artset1	4	3	5	2	6	1
artset2	5	6	4	2	3	1
Média	4,7	4,3	4,6	2,8	3,6	1

Baseado nessa evidência, foi realizada uma análise *post-hoc* por meio de comparação pareada para identificar as diferenças entre os algoritmos. Em particular, foi testada a família de hipóteses com o C-GRASP-Clu como método de controle. O procedimento de *Hommel* foi utilizado para ajustar o valor-p obtido pelo teste de *Friedman* para cada hipótese. Podemos observar que o C-GRASP-Clu superou, com significância estatística, os outros algoritmos (quando considerado $\alpha = 0,01$), com exceção do GA. Maiores explicações sobre todos os procedimentos estatísticos utilizados podem ser encontrados em Derrac et al. (2011).

Tabela 5.5: Valor-p ajustado através do procedimento de *Hommel* (C-GRASP-Clu é o método de controle.)

Algoritmo	valor-p não ajustado	valor-p ajustado	Hipótese
k-means	0,00001	0,0001	Rejeitado ($\alpha = 0,01$)
TS	0,00008	0,0005	Rejeitado ($\alpha = 0,01$)
PSO	0,00001	0,0001	Rejeitado ($\alpha = 0,01$)
GA	0,03	0,23	Não Rejeitado
C-GRASP	0,002	0,013	Rejeitado ($\alpha = 0,05$)

5.1.3.2 Comparação com heurísticas do estado da arte na literatura

A performance do C-GRASP-Clu foi comparada com heurísticas do estado da arte, nomeadamente:

- *Black Hole* (BH) (HATAMLOU, 2013): 50 execuções.
- k-means *modified cohort intelligence* (K-MCI) (KRISHNASAMY et al., 2014): 20 execuções.
- k-means *Nelder-Mead simplex* PSO (K-NM-PSO) (KAO et al., 2008): 20 execuções.
- *Variable vibrating search + simplified swarm optimization + rapid centralized strategy* (VSSO-RCS) (YEH; LAI, 2015): 50 execuções.
- k-means *selective regeneration* PSO (KSRPSO) (TSAI; KAO, 2011): 50 execuções.

Todos os resultados foram obtidos diretamente dos respectivos trabalhos, o que justifica a diferença de precisão numérica e falta de valores para algumas bases, como apresentado na Tabela 5.6. Os algoritmos K-MCI, KSRPSO e C-GRASP-Clu obtiveram os melhores resultados para a base *iris*. Para a base *wine*, o C-GRASP-Clu obteve os melhores valores em média, pior solução e desvio padrão, e o K-NM-PSO obteve a melhor valor entre todas as execuções. Para a base *vowel*, novamente o C-GRASP-Clu obteve os melhores valores em média, pior solução e desvio padrão, o KSRPSO obteve o melhor valor. Para a base *cmc* o VSSO-RCS e o C-GRASP-Clu obtiveram os melhores resultados para todos os critérios. Para a base *cancer*, o K-MCI e o C-GRASP-Clu obteveram os melhores resultados para todos os critérios. O K-NM-PSO e o KSRPSO obtiveram de forma isolada os melhores resultados para as bases *glass* e *thyroid*, respectivamente.

Em geral, os resultados indicam que o C-GRASP-Clu é altamente eficiente e robusto, uma vez que atingiu os melhores resultados para a maioria dos critérios em 5 das 7

bases. Além disso, a heurística proposta obteve, em média, os menores valores para desvio padrão.

Tabela 5.6: Comparação entre o C-GRASP-Clu com heurísticas do estado da arte.

Base	Critério	BH	K-MCI	K-NM-PSO	VSSO-RCS	KSRPSO	C-GRASP-Clu
iris	Best	96,65589	**96,6554**	96,66	96,66	**96,655**	**96,65548**
	Melhor	96,65681	**96,6554**	96,67	96,66	**96,655**	**96,65548**
	Pior	96,66306	**96,6554**	–	96,66	**96,655**	**96,65548**
	Desvio P.	0,00173	0	0,008	0	0,00003	$< 10^{-7}$
wine	Best	16293,41995	16292,44	**16292,00**	16292,18	16292,180	16292,18464
	Melhor	16294,31763	16292,70	16293,00	16292,76	16292,580	**16292,18464**
	Pior	16300,22613	16292,88	–	16294,17	16292,720	**16292,18464**
	Desvio P.	1,65127	0,130	0,46	0,82	0,287	$< 10^{-9}$
vowel	Best	148985,61373	148967,24	149005,00	148967,24	**148967,200**	148967,24093
	Melhor	149848,18144	148987,55	149141,4	149148,08	149015,900	**148967,24123**
	Pior	153058,98663	149048,58	–	150139,66	149062,800	**148967,24157**
	Desvio P.	1306,95375	36,086	120,38	336,16	45,734	0,00014
cmc	Best	5532,88323	5693,73	5532,40	**5532,18**	5532,185	**5532,18472**
	Melhor	5533,63122	5693,75	5532,70	**5532,18**	5532,185	**5532,18472**
	Pior	5534,77738	5693,80	–	**5532,18**	5532,185	**5532,18472**
	Desvio P.	0,59940	0,014	0,23	0	0	$< 10^{-9}$
cancer	Best	2964,38878	**2964,38**	2964,5	2964,39	2964,387	**2964,38697**
	Melhor	2964,39539	**2964,38**	2964,7	2964,39	2964,388	**2964,38697**
	Pior	2964,45074	**2964,38**	–	2964,39	2964,388	**2964,38697**
	Desvio P.	0,00921	0	0,15	0	0	$< 10^{-6}$
glass	Best	210,51549	212,34	**199,68**	210,43	201,960	210,00104
	Melhor	211,4986	212,57	**200,5**	211,31	206,594	210,00104
	Pior	213,95689	212,80	–	214,81	211,697	210,00104
	Desvio P.	1,18230	0,135	2,26	1,78	4,385	$< 10^{-8}$
crudeoil	Best	–	–	277,15	277,21	**277,133**	277,21073
	Melhor	–	–	277,29	277,26	**277,133**	277,21073
	Pior	–	–	–	277,36	**277,133**	277,21073
	Desvio P.	–	–	0,095	0,05	0	$< 10^{-9}$

5.2 CCQPSO+VND

5.2.1 Configuração do ambiente de teste

Todos os algoritmos foram codificados em Matlab 7.14 (R2012a) e executados em um Intel Core i5-3210M com 2.5 GHz e 8 GB de RAM em sistema operacional Linux Mint 17 (64 bits).

Os experimentos foram realizados em dois conjuntos de teste composto por imagens conhecidas da literatura (Figura 5.3) e imagens retro espalhadas (BEI, em inglês *Backscattered Electron Image*) obtidas por microscopia eletrônica para análise de materiais cimentícios (Figura 5.4). O conjunto de imagens BEI foi concebido e disponibilidade no trabalho de Vieira et al. (2010).

Os parâmetros dos métodos descritos na Tabela 5.7, com exceção do número de iterações, foram obtidos em Duraisamy et al. (2010) e Li et al. (2015), respectivamente.

Tabela 5.7: Parâmetros de entrada do PSO e CCQPSO/CCQPSO+VND.

Algoritmo	Parâmetro	Valor
PSO	Tamanho do enxame	20
	Número de iterações	200
	c_1, c_2	2
	w_{max}, w_{min}	0,4, 0,1
CCQPSO/CCQPSO+VND	Tamanho do enxame	20
	Número de iterações	200
	γ	5
	$\alpha_{max}, \alpha_{min}$	1,0, 0,5

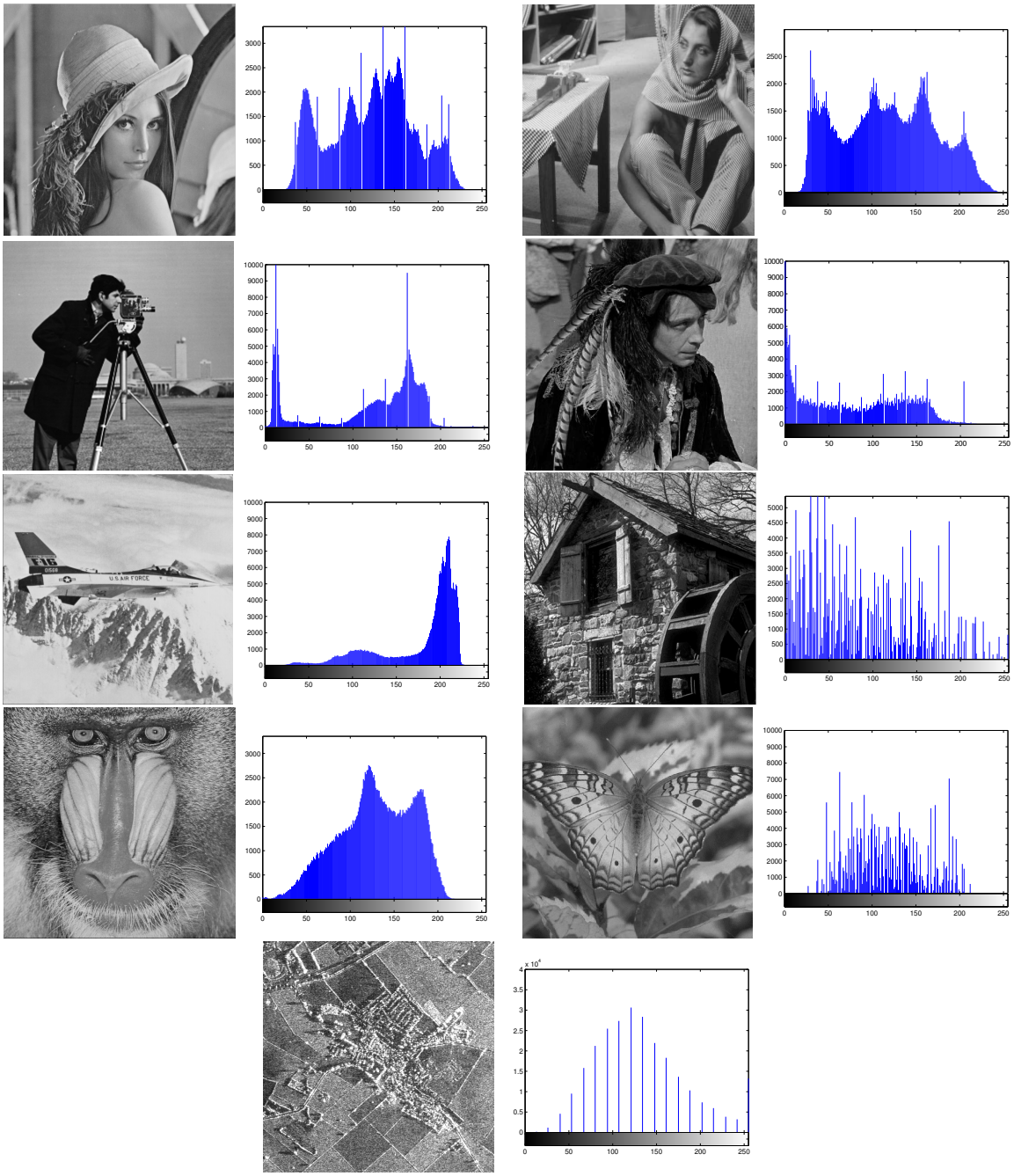


Figura 5.3: Conjunto de imagens da literatura com os seus respectivos histogramas: Lena, Barbara, Cameraman, Casa, Caçador, Avião. Babuíno, Borboleta e Mapa

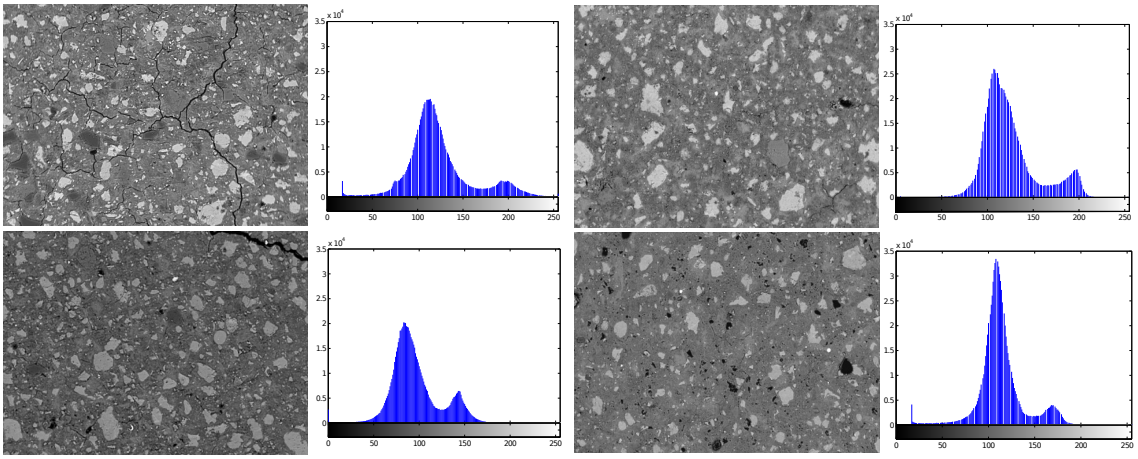


Figura 5.4: Conjunto de imagens BEI e seus respectivos histogramas: BEI01, BEI02, BEI03, BEI04.

5.2.2 Análise de desempenho

Na Tabela 5.8, são reportados os resultados obtidos pelo PSO, CCQPSO e CCQPSO+VND para o primeiro conjunto de imagens. Os melhores limiares encontrados entre 50 execuções independentes bem como o valor médio da função objetivo (critério de Otsu de máxima variância *intercluster*) são apresentados para operações de limiarização com diferentes níveis. O CCQPSO+VND obteve 13 vitórias sobre o PSO e CCQPSO, sendo concentradas principalmente em instâncias de maior dificuldade (com mais limiares). Na maioria das outras instâncias houve empate entre o CCQPSO e CCQPSO+VND. O PSO demonstrou desempenho inferior aos outros algoritmos, conseguindo apenas empates para instâncias consideradas mais fáceis.

De forma semelhante, a Tabela 5.9 apresenta o desvio padrão obtido bem como o tempo computacional médio necessário para atingir a solução resultante. Com respeito ao desvio padrão, podemos concluir que o CCQPSO+VND demonstrou ser o mais robusto dos algoritmos, principalmente quando comparado ao PSO. Pode-se notar facilmente que o PSO, apesar do desempenho inferior, possui tempo computacional médio para atingir a solução mais baixo que o CCQPSO e CCQPSO+VND. Mesmo assim, deve-se destacar a influência da busca local VND na convergência do CCQPSO, uma vez que o tempo computacional médio baixou nitidamente após a sua inserção (com exceção das imagens Borboleta e Mapa), além da melhora no desempenho já discutida.

As Figuras 5.5, 5.6 e 5.7 ilustram as imagens segmentadas com dois e três limiares obtidos pelo algoritmo CCQPSO+VND.

Tabela 5.8: Resultados obtidos através de 50 execuções do PSO, CCQPSO, CCQPSO+VND sobre o primeiro conjunto de teste.

		Melhores limiares encontrados			Variância entre classes (média)		
Imagem	k	PSO	CCQPSO	CCQPSO+VND	PSO	CCQPSO	CCQPSO+VND
Lena	2	70,144	70,144	70,144	3609,5601	3609,5601	3609,5601
	3	57,116,154	57,116,154	57,116,154	3683,2842	3683,3247	**3683,3513**
	4	41,94,139,169	42,95,139,169	41,94,139,169	3739,1869	3739,2316	**3739,2401**
	5	36,83,122,149,173	36,84,122,149,173	36,83,122,149,173	3760,1533	3769,9879	**3770,0495**
Barbara	2	51,116	51,116	51,116	3064,2112	3064,2112	3064,2112
	3	36,86,135	36,86,135	36,86,135	3213,402	3213,446	3213,446
	4	30,72,111,146	30,72,111,146	30,72,111,146	3268,896	3269,4969	**3269,4991**
	5	23,54,89,122,152	22,53,88,122,152	22,53,88,122,152	3308,0597	3308,1302	**3308,1309**
Cameraman	2	70,144	70,144	70,144	3609,5601	3609,5601	3609,5601
	3	57,116,154	57,116,154	57,116,154	3683,2842	3683,3247	**3683,3513**
	4	41,94,139,169	42,95,139,169	41,94,139,169	3739,1869	3739,2316	**3739,2401**
	5	36,83,122,149,173	36,84,122,149,173	36,83,122,149,173	3760,1533	3769,9879	**3770,0495**
Caçador	2	51,116	51,116	51,116	3064,2112	3064,2112	3064,2112
	3	36,86,135	36,86,135	36,86,135	3213,402	3213,446	3213,446
	4	30,72,111,146	30,72,111,146	30,72,111,146	3268,896	3269,4969	**3269,4991**
	5	23,54,89,122,152	22,53,88,122,152	22,53,88,122,152	3308,0597	3308,1302	**3308,1309**
Avião	2	116,174	116,174	116,174	1837,7974	1837,7974	1837,7974
	3	95,146,191	95,146,191	95,146,191	1911,666	1911,7236	1911,7236
	4	88,131,173,203	88,131,173,203	88,132,174,203	1954,9537	1955,0062	**1955,024**
	5	71,108,143,179,204	71,109,144,180,204	71,108,143,179,204	1979,2831	1979,852	**1979,9232**
Casa	2	55,129	56,129	56,127	3421,2828	3421,2828	3421,2828
	3	42,98,161	42,98,161	42,98,161	3623,486	3623,486	3623,486
	4	26,66,114,169	26,67,114,169	31,74,123,179	3725,4203	3725,6335	3725,6335
	5	24,55,91,130,176	24,55,92,130,178	24,55,91,130,176	3785,8353	3785,985	3785,985
Babuíno	2	97,149	97,149	97,149	1548,1408	1548,1408	1548,1408
	3	85,124,160	85,125,161	85,125,161	1638,2773	1638,3205	1638,3205
	4	72,106,137,168	72,106,137,168	72,106,137,168	1692,0821	1692,1491	1692,1491
	5	67,99,125,149,174	67,99,125,149,174	67,99,125,149,174	1717,6371	1717,8205	**1717,8577**
Borboleta	2	98,151	98,151	99,151	1552,8825	1553,0734	1553,0734
	3	81,119,160	81,119,160	82,119,160	1669,2783	1669,2783	1669,2783
	4	71,99,126,162	71,98,126,161	71,99,126,162	1702,8821	1711,2193	1711,2193
	5	71,99,125,154,179	71,99,125,153,180	71,99,125,154,179	1731,3185	1736,6572	1736,6572
Mapa	2	115,183	108,179	120,185	2340,395	2340,395	2340,395
	3	105,144,194	104,145,201	101,139,202	2529,9384	2529,9384	2529,9384
	4	90,122,165,219	81,132,170,227	93,129,171,226	2618,8342	2621,1476	2621,1476
	5	74,114,145,182,225	69,120,140,182,225	75,115,135,183,217	2651,3785	2670,064	2670,064

Tabela 5.9: Desvio padrão e tempo computacional médio obtido pelo PSO, CCQPSO e CCQPSO+VND.

		Desvio padrão			Tempo computacional médio (s)		
Imagem	k	PSO	CCQPSO	CCQPSO+VND	PSO	CCQPSO	CCQPSO+VND
Lena	2	$< 10^{-11}$	$< 10^{-11}$	$< 10^{-11}$	0,021172	0,058617	0,043434
	3	0,011195	0,003767	$< 10^{-11}$	0,037028	0,25467	0,12379
	4	0,02084	0,014711	0,0094231	0,039696	0,57803	0,2261
	5	1,3988	0,018661	0,0039482	0,054781	0,86114	0,46028
Barbara	2	$< 10^{-12}$	$< 10^{-12}$	$< 10^{-12}$	0,094541	0,11935	0,11109
	3	0,0096829	$< 10^{-11}$	$< 10^{-11}$	0,094725	0,2583	0,15125
	4	0,08864	0,0070603	0,0042729	0,10134	0,52295	0,29042
	5	0,017914	0,0036996	0,0024935	0,10808	0,6916	0,24013
Cameraman	2	$< 10^{-11}$	$< 10^{-11}$	$< 10^{-11}$	0,021172	0,058617	0,043434
	3	0,011195	0,003767	$< 10^{-11}$	0,037028	0,25467	0,12379
	4	0,02084	0,014711	0,0094231	0,039696	0,57803	0,2261
	5	1,3988	0,018661	0,0039482	0,054781	0,86114	0,46028
Caçador	2	$< 10^{-12}$	$< 10^{-12}$	$< 10^{-12}$	0,094541	0,11935	0,11109
	3	0,0096829	$< 10^{-11}$	$< 10^{-11}$	0,094725	0,2583	0,15125
	4	0,08864	0,0070603	0,0042729	0,10134	0,52295	0,29042
	5	0,017914	0,0036996	0,0024935	0,10808	0,6916	0,24013
Avião	2	$< 10^{-11}$	$< 10^{-11}$	$< 10^{-11}$	0,084732	0,11442	0,10207
	3	0,0087483	$< 10^{-11}$	$< 10^{-11}$	0,090064	0,29581	0,13922
	4	0,018085	0,012836	0,0035076	0,097985	0,63304	0,29557
	5	0,096464	0,025791	0,013326	0,10264	0,84872	0,32512
Casa	2	$< 10^{-11}$	$< 10^{-11}$	$< 10^{-11}$	0,083126	0,09668	0,10524
	3	$< 10^{-12}$	$< 10^{-12}$	$< 10^{-12}$	0,0894	0,20021	0,13105
	4	0,2413	0,12918	0,18309	0,096997	0,31815	0,22244
	5	0,021278	$< 10^{-11}$	$< 10^{-11}$	0,10265	0,46418	0,1701
Babuíno	2	$< 10^{-11}$	$< 10^{-11}$	$< 10^{-11}$	0,084907	0,10535	0,10097
	3	0,0067594	$< 10^{-11}$	$< 10^{-11}$	0,090585	0,22318	0,17396
	4	0,016705	$< 10^{-12}$	$< 10^{-12}$	0,098499	0,40575	0,21068
	5	0,063298	0,01492	0,011779	0,10977	0,71606	0,55104
Borboleta	2	0,037783	$< 10^{-11}$	$< 10^{-11}$	0,084099	0,097395	0,10649
	3	$< 10^{-11}$	$< 10^{-11}$	$< 10^{-11}$	0,086259	0,14421	0,11211
	4	1,1791	$< 10^{-11}$	$< 10^{-11}$	0,093259	0,24823	0,12961
	5	0,8983	$< 10^{-12}$	$< 10^{-12}$	0,10329	0,4324	0,18402
Mapa	2	0	0	0	0,083883	0,084786	0,099777
	3	$< 10^{-11}$	$< 10^{-11}$	$< 10^{-11}$	0,08504	0,087278	0,11796
	4	1,0251	0	0	0,085522	0,10852	0,17672
	5	2,6425	$< 10^{-11}$	$< 10^{-11}$	0,096398	0,11771	0,17846

Figura 5.5: Segmentação resultante do CCQPSO+VND para dois e três limiares nas imagens Lena, Barbara e Cameraman.

Figura 5.6: Segmentação resultante do CCQPSO+VND para dois e três limiares nas imagens Caçador, Avião e Casa.

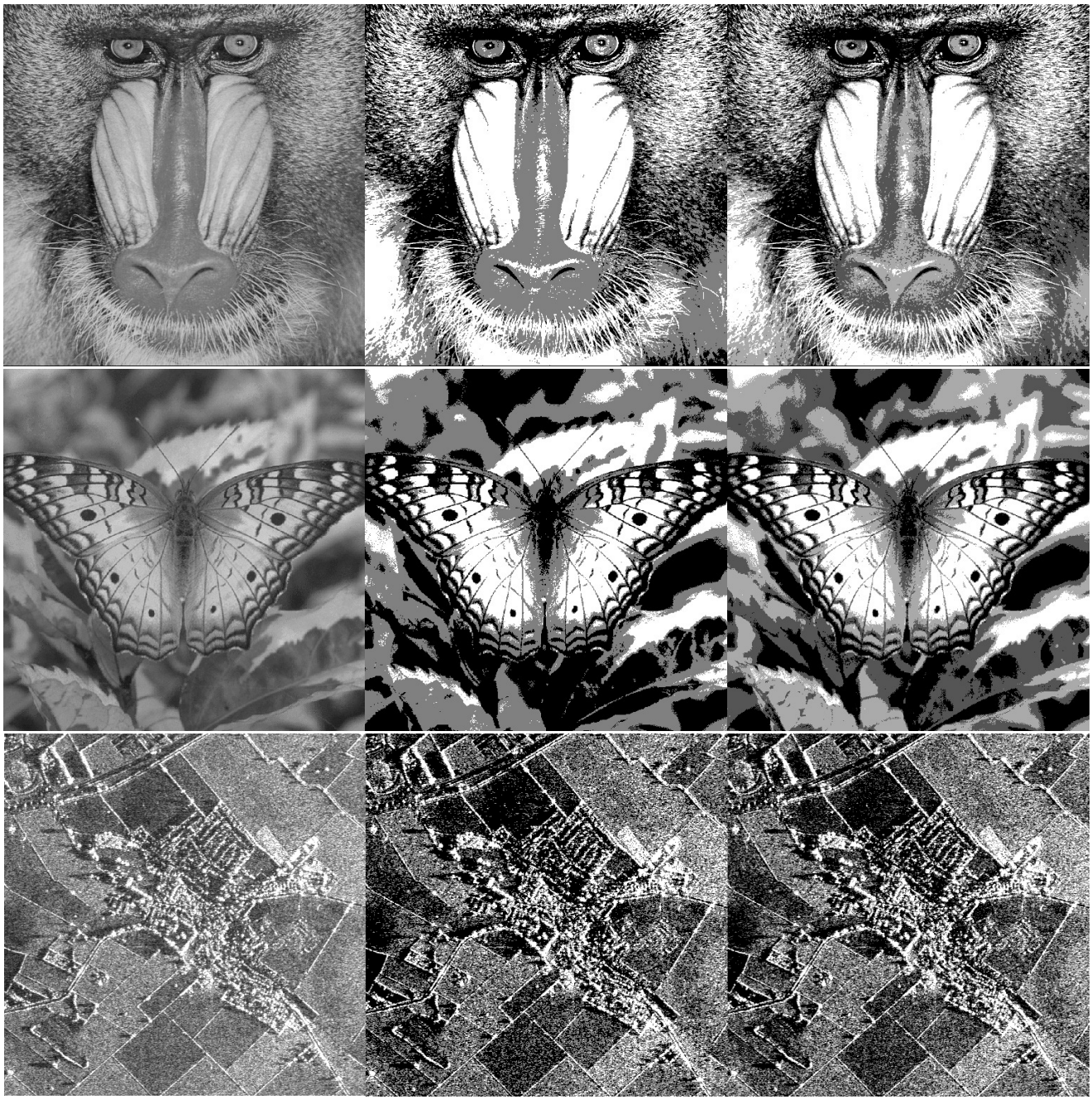


Figura 5.7: Segmentação resultante do CCQPSO+VND para dois e três limiares nas imagens Babuíno, Borboleta e Mapa.

Os resultados obtidos para o segundo conjunto de imagens são apresentados na Tabela 5.10. Pode-se notar uma leve vantagem da abordagem CCQPSO+VND que obteve os melhores resultados para 10 testes contra 4 do CCQPSO e nenhuma vitória do PSO. Com relação ao tempo computacional médio para atingir a solução, fica mais evidente que o PSO consegue convergir mais rapidamente, principalmente para tarefas com 10, 15 e 20 limiares. Isso pode ser justificado pelo custo computacional adicionado pela etapa de cooperação de partículas realizada no CCQPSO. Mesmo assim, pode-se verificar novamente a contribuição da busca local VND, uma vez que o tempo médio do CCQPSO+VND é geralmente menor do que tempo obtido pelo CCQPSO, principalmente para tarefas com 10, 15 e 20 limiares.

		Variância entre classes (média)			Tempo médio (segundos)		
Imagem	k	PSO	CCQPSO	CCQPSO+VND	PSO	CCQPSO	CCQPSO+VND
BEI01	3	599.952341	599.988057	599.988057	0.090475	0.109291	0.115232
	4	642.991025	643.755267	643.755267	0.111607	0.232656	0.195685
	5	665.379285	668.573036	**668.573997**	0.092735	0.383605	0.213719
	10	701.569938	710.853685	**710.866143**	0.119421	2.042868	1.672238
	15	711.660951	720.406666	**720.436059**	0.093856	3.591087	3.299511
	20	717.491729	724.056692	**724.095623**	0.113044	5.156501	4.497750
BEI02	3	777.117691	777.137035	777.137035	0.089228	0.103250	0.105573
	4	819.583849	821.514716	821.514716	0.093665	0.147158	0.171828
	5	842.954820	847.018680	847.018680	0.090445	0.178225	0.201434
	10	880.592083	**889.812747**	889.803200	0.093451	1.404010	0.724974
	15	890.884168	**899.265549**	899.252143	0.090627	2.673245	1.653127
	20	896.126987	902.522515	**902.600770**	0.132780	5.281497	3.762042
BEI03	3	1076.140292	1076.164119	1076.164119	0.088605	0.111764	0.104546
	4	1172.383523	1173.080263	1173.080263	0.091807	0.184781	0.127002
	5	1216.930237	1220.150692	**1220.155543**	0.092306	0.324694	0.247778
	10	1287.062181	**1298.470814**	1298.460055	0.093085	1.691532	0.766330
	15	1303.812564	1314.102358	**1314.166207**	0.112017	2.838264	1.448285
	20	1312.401482	1319.640759	**1319.681470**	0.092138	5.351329	3.019044
BEI04	3	477.412131	477.427264	477.427264	0.108727	0.114167	0.102892
	4	543.139200	543.595321	543.595321	0.094831	0.149893	0.113851
	5	566.587782	570.835182	570.835182	0.103603	0.317727	0.230345
	10	600.216732	608.071249	**608.112352**	0.112065	1.770649	0.733474
	15	610.028589	617.071974	**617.139278**	0.111356	2.973637	1.145458
	20	614.635724	**620.134070**	620.105899	0.091524	4.505474	2.492589

Tabela 5.10: Resultados da performance obtida pelo PSO, CCQPSO e CCQPSO+VND para o segundo conjunto de teste.

5.2.3 Análise microestrutural em sistemas cimentícios por segmentação de imagens BEI

Como já discutido na seção 2.5, Imagens por Elétrons Retro Espalhados (BEI, do inglês *Backscattered Electron Image*), são imagens em escala de cinza com intensidade determinada pelo número atômico médio do local na amostra. Uma segmentação eficiente dessas imagens pode fornecer uma quantificação automática das fases em amostras de pasta de cimento hidratada, tais como: conteúdo de cimento hidratado, conteúdo de cimento desidratado, agregados, trincas e poros. Como pode-se observar na segmentação resultante do algoritmo `CCQPSO+VND` para a imagem BEI01 com 2 e 3 limiares (Figuras 5.8 (a) e (c)) o segmento de trincas não foi bem separado, demonstrando uma fragilidade do critério de limiarização utilizado para essa aplicação. Com uma simples ponderação da função objetivo, podemos priorizar certos segmentos, como descrito na equação 5.1. Para dois limiares, bons resultados foram obtidos com os pesos $\lambda = [0,93, 1, 1]$, como mostra a Figura 5.8 (b), onde as trincas, a pasta hidratada (junto com o agregado) e a pasta desidratada foram bem segmentadas. Para três limiares, os pesos $\lambda = [0,95, 0,96, 1, 1]$ geraram bons resultados, conseguindo separar as partículas de agregados presentes na amostra (veja a Figura 5.8 (d)). Utilizando os mesmos pesos para as quatro imagens de teste, podemos ver que a segmentação resultante é bastante satisfatória (Figura 5.9).

$$f(t) = \lambda_1 w_1(\mu_1 - \mu)^2 + \lambda_2 w_2(\mu_2 - \mu)^2 + \lambda_{k+1} w_{k+1}(\mu_{k+1} - \mu)^2 \tag{5.1}$$

Após a tarefa de segmentação, a imagem binária das trincas é uma importante fonte de atributos de interesse. Para extração desses atributos, um tratamento por modelagem em grafos, semelhante ao realizado nos trabalhos de Zou et al. (2012) e Luo et al. (2015), é utilizado.

Dado uma imagem binária do segmento que possui trincas e poros (em níveis baixos de cinza da imagem original), podemos criar um grafo $G = (E, V)$ com conjunto de vértices (ou nós) V definido por cada pixel da trinca (pixels pretos) e o conjunto de arestas E definido pela adjacência ortogonal e diagonal (pixels vizinhos) de pixels (nó) da trinca, como ilustrado na Figura 5.10(a-b). Usando algoritmos de busca em profundidade (TARJAN, 1972), podemos facilmente separar as trincas por explorar a conectividade e eliminar componentes (subgrafos) de baixa cardinalidade, por considerá-las elementos ruidosos da imagem (Figura 5.10(b)).

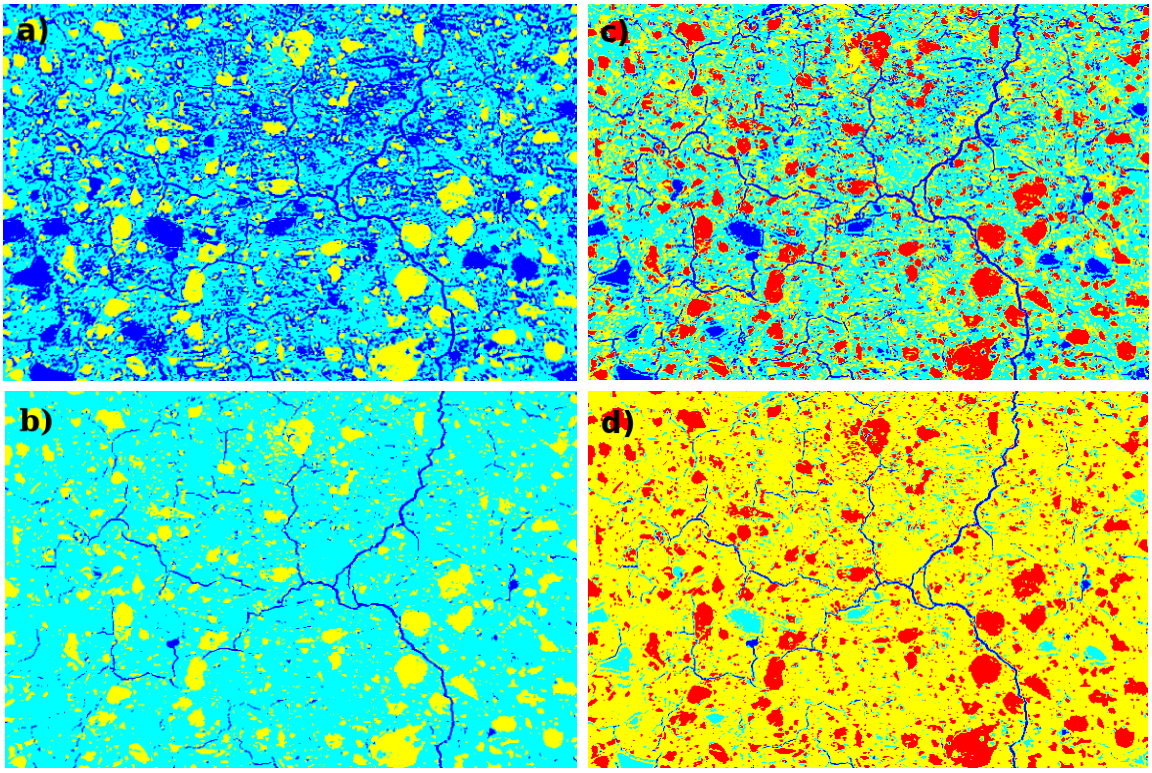


Figura 5.8: Resultados de segmentação para o critério Otsu original e uma adaptação realizada.

Um filtro de poros pode ser projetado para eliminar componentes conectadas pela *relação cardinalidade-diâmetro* (RCD), que remove objetos com forma arredondada através da condição 5.2, onde para uma componente conectada G_c, temos que $d(G_c)$ retorna o diâmetro do grafo (WEST et al., 2001). Quando mais próximo de 0 for t_p, mais rigoroso se torna o filtro, podendo descartar trincas verdadeiras, que possuem RCD extremamente baixas. Obter a Árvore Geradora de Custo Mínimo (MST, do inglês *Minimum Spanning Tree*) (WEST et al., 2001) de G pode reduzir a dimensionalidade para procedimentos de extração posteriores, como ilustrado na Figura 5.10(c)). Em seguida, um procedimento de poda para remover ramos internos à largura da trinca pode ser aplicado, produzindo um grafo esqueleto que fornece o comprimento das trincas por aplicar novamente o cálculo do diâmetros em cada componente conectada, como ilustrado na Figura 5.10(d)). O procedimento completo com segmentação CCQPSO+VND com Otsu ponderado e a detecção de trincas por algoritmos em grafos é ilustrado para outras duas amostras de imagens BEI na Figura 5.11.

$$\frac{|G_c|}{d(G_c)} > t_p \tag{5.2}$$

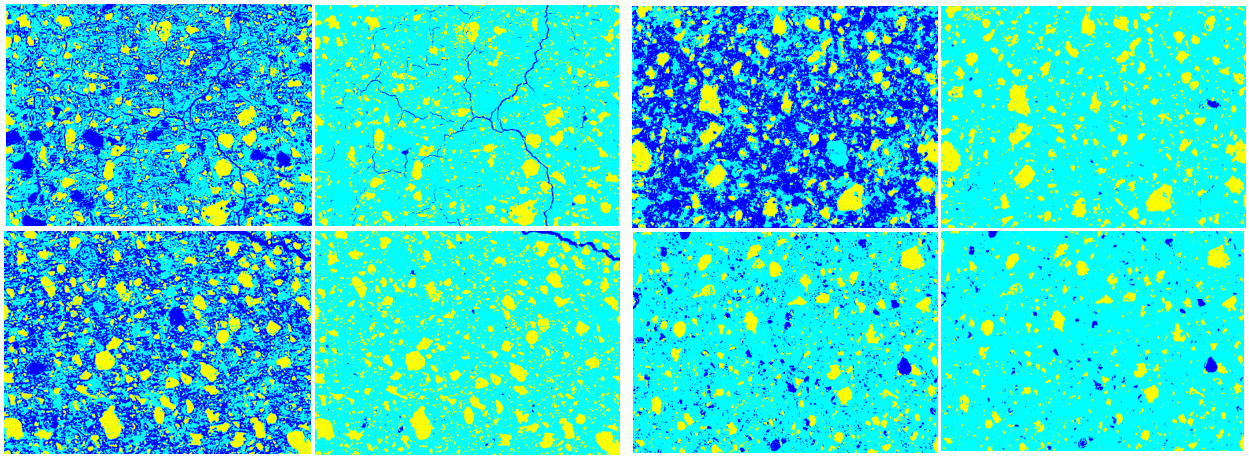


Figura 5.9: Resultados obtidos por limiarização binível com critério de Otsu original e ponderado com $\lambda = [0, 93, 1, 1]$..

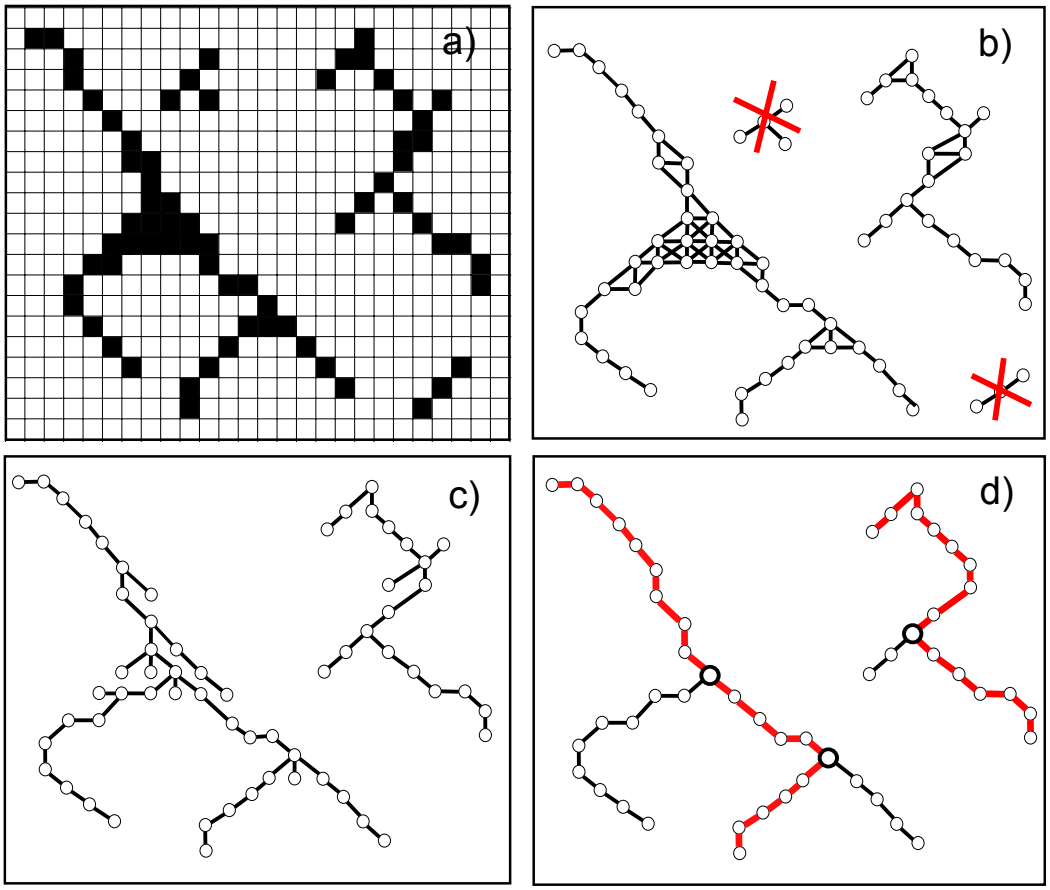


Figura 5.10: Procedimento de geração do grafo esqueleto a partir da imagem binária de trincas.

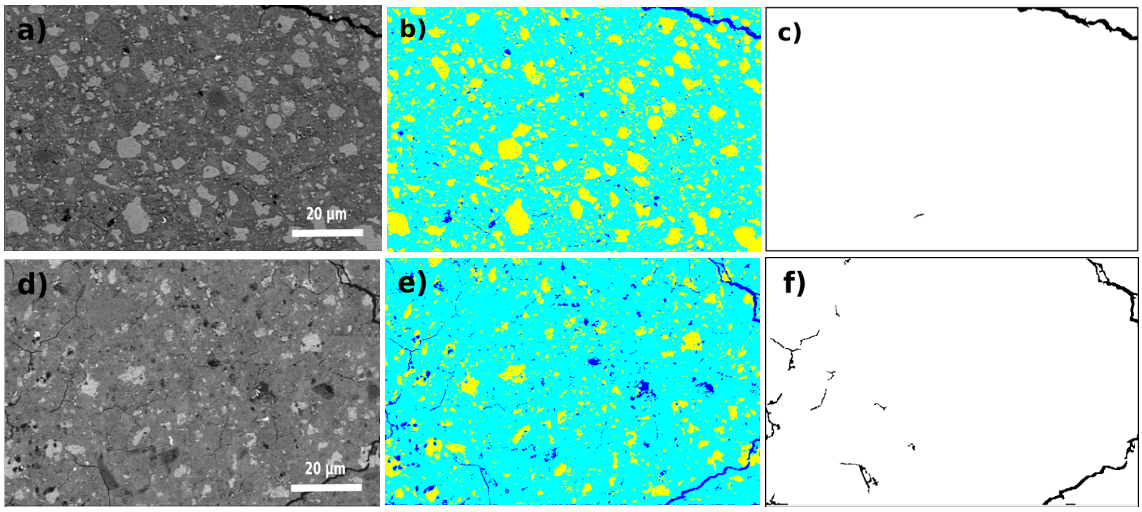


Figura 5.11: Segmentação e detecção de trincas por limiarização CCQPSO+VND e algoritmos em grafos para filtros de poros.

Capítulo 6

CONSIDERAÇÕES FINAIS

Neste trabalho foram estudas técnicas de resolução baseadas em meta-heurísticas para os problemas de clusterização de dados e segmentação de imagens.

O problema de clusterização particional com mínima distância intracluster foi tratado através de uma nova abordagem baseada na meta-heurística *Continuous* GRASP. Extensivos experimentos computacionais em bases de dados conhecidas mostraram que o método proposto foi capaz de sistematicamente encontrar soluções de alta qualidade e boa robustez quando comparado a técnicas conhecidas e heurísticas do estado da arte da literatura. Como trabalhos futuros, pode-se investigar estratégias alternativas para gerar direções de descida promissoras durante a etapa de busca local, bem como hibridizações com outros procedimentos, tal como o k-means. Além disso, o algoritmo proposto pode ser adaptado para resolver outras classes de problemas de clusterização com diferentes objetivos, restrições e medidas de similaridade.

O problema de limiarização multinível com critério de Otsu foi tratado por uma abordagem híbrida que combina uma versão cooperativa do *Quantum-behaved* PSO com uma nova busca local VND. O método foi aplicado a imagens conhecidas da literatura e um conjunto de imagens de materiais cimentícios obtidas por microscopia eletrônica. Bons resultados foram obtidos em termos de qualidade de solução quando comparado a versão sem busca local e o PSO puro. Os resultados visuais indicam limitações no critério utilizado para aplicação em análise de materiais, requerendo ajustes para o passo de aquisição de imagens e operações adicionais. Como trabalhos futuros, o estudo de novas estratégias de busca local e mecanismos de cooperação mais sofisticados podem ser realizados, além da exploração de novos critérios adequados para a aplicação desejada.

Referências

AGGARWAL, C. C.; ZHAI, C. A survey of text clustering algorithms. In: *Mining Text Data*. [S.l.]: Springer, 2012. p. 77–128.

AKAY, B. A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding. *Applied Soft Computing*, Elsevier, v. 13, n. 6, p. 3066–3091, 2013.

AL-SULTAN, K. S. A tabu search approach to the clustering problem. *Pattern Recognition*, Elsevier, v. 28, n. 9, p. 1443–1451, 1995.

ARBELAEZ, P. et al. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 33, n. 5, p. 898–916, 2011.

BRITO, D. M. et al. A novel method to predict genomic islands based on mean shift clustering algorithm. *PloS one*, Public Library of Science, v. 11, n. 1, 2016.

BUBECK, S.; MEILĂ, M.; LUXBURG, U. von. How the initialization affects the stability of the k-means algorithm. *ESAIM: Probability and Statistics*, Cambridge Univ Press, v. 16, p. 436–452, 2012.

CAI, X.; LI, W. A spectral analysis approach to document summarization: clustering and ranking sentences simultaneously. *Information Sciences*, Elsevier, v. 181, n. 18, p. 3816–3827, 2011.

CALVO, B.; SANTAFE, G. scmamp: Statistical comparison of multiple algorithms in multiple problems. *The R Journal*, Accepted for publication, 2015. Accessed: 2016-08-03. Disponível em: <https://cran.r-project.org/web/packages/scmamp/>.

CELENK, M. A color clustering technique for image segmentation. *Computer Vision, Graphics, and Image Processing*, Elsevier, v. 52, n. 2, p. 145–170, 1990.

CHEN, C.-Y.; YE, F. Particle swarm optimization algorithm and its application to clustering analysis. In: IEEE. *Networking, Sensing and Control, 2004 IEEE International Conference on*. [S.l.], 2004. v. 2, p. 789–794.

CHUANG, L.-Y.; HSIAO, C.-J.; YANG, C.-H. Chaotic particle swarm optimization for data clustering. *Expert systems with Applications*, Elsevier, v. 38, n. 12, p. 14555–14563, 2011.

COELHO, L. dos S. Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. *Expert Systems with Applications*, Elsevier, v. 37, n. 2, p. 1676–1683, 2010.

CURA, T. A particle swarm optimization approach to clustering. *Expert Systems with Applications*, Elsevier, v. 39, n. 1, p. 1582–1588, 2012.

DANTZIG, G. B. *Linear programming and extensions*. [S.l.]: Princeton university press, 1998.

DAS, S.; SIL, S. Kernel-induced fuzzy clustering of image pixels with an improved differential evolution algorithm. *Information Sciences*, Elsevier, v. 180, n. 8, p. 1237–1256, 2010.

DEDAVID, B. A.; GOMES, C. I.; MACHADO, G. *Microscopia eletrônica de varredura: aplicações e preparação de amostras: materiais poliméricos, metálicos e semicondutores*. [S.l.]: EdiPUCRS, 2007.

DERRAC, J. et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, Elsevier, v. 1, n. 1, p. 3–18, 2011.

DESCHNER, F. et al. Quantification of fly ash in hydrated, blended portland cement pastes by backscattered electron imaging. *Journal of microscopy*, Wiley Online Library, v. 251, n. 2, p. 188–204, 2013.

DEY, S. et al. Multi-level thresholding using quantum inspired meta-heuristics. *Knowledge-Based Systems*, Elsevier, v. 67, p. 373–400, 2014.

DJEROU, L. et al. Automatic multilevel thresholding using binary particle swarm optimization for image segmentation. In: IEEE. *Soft Computing and Pattern Recognition, 2009. SOCPAR'09. International Conference of*. [S.l.], 2009. p. 66–71.

DURAISAMY, S. P.; KAYALVIZHI, R. et al. A new multilevel thresholding method using swarm intelligence algorithm for image segmentation. *Journal of Intelligent Learning Systems and Applications*, Scientific Research Publishing, v. 2, n. 03, p. 126, 2010.

EBERHART, R. C.; KENNEDY, J. A new optimizer using particle swarm theory. In: NEW YORK, NY. *Proceedings of the sixth international symposium on micro machine and human science*. [S.l.], 1995. v. 1, p. 39–43.

FEO, T.; RESENDE, M. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, Kluwer Academic Publishers, v. 6, n. 2, p. 109–133, 1995. ISSN 0925-5001. Disponível em: <http://dx.doi.org/10.1007/BF01096763>.

GERRILD, P. M.; LANTZ, R. J. *Chemical analysis of 75 crude oil samples from Pliocene sand units, Elk Hills oil field, California*. [S.l.], 1969.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, Elsevier, v. 13, n. 5, p. 533–549, 1986.

GOLDBERG, D. E.; DEB, K. A comparative analysis of selection schemes used in genetic algorithms. *Foundations of genetic algorithms*, v. 1, p. 69–93, 1991.

GONZALEZ, T. F. On the computational complexity of clustering and related problems. In: *System modeling and optimization*. [S.l.]: Springer, 1982. p. 174–182.

HAMMOUCHE, K.; DIAF, M.; SIARRY, P. A comparative study of various meta-heuristic techniques applied to the multilevel thresholding problem. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 23, n. 5, p. 676–688, 2010.

HANSEN, P. The steepest ascent mildest descent heuristic for combinatorial programming. In: *Congress on numerical methods in combinatorial optimization, Capri, Italy*. [S.l.: s.n.], 1986. p. 70–145.

HANSEN, P.; JAUMARD, B. Cluster analysis and mathematical programming. *Mathematical programming*, Springer, v. 79, n. 1-3, p. 191–215, 1997.

HATAMLOU, A. Black hole: A new heuristic optimization approach for data clustering. *Information sciences*, Elsevier, v. 222, p. 175–184, 2013.

HATAMLOU, A.; ABDULLAH, S.; NEZAMABADI-POUR, H. A combined approach for clustering based on k-means and gravitational search algorithms. *Swarm and Evolutionary Computation*, Elsevier, v. 6, p. 47–52, 2012.

HEAD, M. K.; BUENFELD, N. R. Measurement of aggregate interfacial porosity in complex, multi-phase aggregate concrete: Binary mask production using backscattered electron, and energy dispersive x-ray images. *Cement and concrete research*, Elsevier, v. 36, n. 2, p. 337–345, 2006.

HIRSCH, M. J. et al. Global optimization by continuous grasp. *Optimization Letters*, Springer, v. 1, n. 2, p. 201–212, 2007.

HIRSCH, M. J.; PARDALOS, P. M.; RESENDE, M. G. Speeding up continuous grasp. *European Journal of Operational Research*, Elsevier, v. 205, n. 3, p. 507–521, 2010.

HRUSCHKA, E. R.; CAMPELLO, R. J.; CASTRO, L. N. D. Evolving clusters in gene-expression data. *Information Sciences*, Elsevier, v. 176, n. 13, p. 1898–1927, 2006.

IGARASHI, S.-i.; KAWAMURA, M.; WATANABE, A. Analysis of cement pastes and mortars by a combination of backscatter-based sem image analysis and calculations based on the powers model. *Cement and Concrete Composites*, Elsevier, v. 26, n. 8, p. 977–985, 2004.

KAO, Y.-T.; ZAHARA, E.; KAO, I.-W. A hybridized approach to data clustering. *Expert Systems with Applications*, Elsevier, v. 34, n. 3, p. 1754–1762, 2008.

KRISHNASAMY, G.; KULKARNI, A. J.; PARAMESRAN, R. A hybrid approach for data clustering based on modified cohort intelligence and k-means. *Expert Systems with Applications*, Elsevier, v. 41, n. 13, p. 6009–6016, 2014.

LI, W.; JAROSZEWSKI, L.; GODZIK, A. Clustering of highly homologous sequences to reduce the size of large protein databases. *Bioinformatics*, Oxford Univ Press, v. 17, n. 3, p. 282–283, 2001.

LI, Y. et al. Dynamic-context cooperative quantum-behaved particle swarm optimization based on multilevel thresholding applied to medical image segmentation. *Information Sciences*, Elsevier, v. 294, p. 408–422, 2015.

LIN, Z.; WANG, Z.; ZHANG, Y. Image thresholding using particle swarm optimization. In: IEEE. *2008 International Conference on MultiMedia and Information Technology*. [S.l.], 2008. p. 245–248.

LIU, Y. et al. Modified particle swarm optimization-based multilevel thresholding for image segmentation. *Soft Computing*, Springer, v. 19, n. 5, p. 1311–1327, 2015.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. *Iterated local search*. [S.l.]: Springer, 2003.

LUO, Z. et al. Extraction of microcracks in rock images based on heuristic graph searching and application. *Computers & Geosciences*, Elsevier, v. 85, p. 22–35, 2015.

MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. 1967. v. 1, n. 14, p. 281–297. Disponível em: <http://projecteuclid.org/euclid.bsmsp/1200512992>.

MAULIK, U.; BANDYOPADHYAY, S. Genetic algorithm-based clustering technique. *Pattern recognition*, Elsevier, v. 33, n. 9, p. 1455–1465, 2000.

MELANIE, M. An introduction to genetic algorithms. 1996.

MERZ, C.; BLAKE, C. *UCI Repository of Machine Learning Databases*. 2016. Accessed: 2016-08-03. Disponível em: <http://archive.ics.uci.edu/ml/>.

MOURET, M.; RINGOT, E.; BASCOUL, A. Image analysis: a tool for the characterisation of hydration of cement in concrete–metrological aspects of magnification on measurement. *Cement and Concrete Composites*, Elsevier, v. 23, n. 2, p. 201–206, 2001.

NIKNAM, T.; AMIRI, B. An efficient hybrid approach based on pso, aco and k-means for cluster analysis. *Applied Soft Computing*, Elsevier, v. 10, n. 1, p. 183–197, 2010.

NOCEDAL, J.; WRIGHT, S. J. Numerical optimization, second edition. *Numerical optimization*, Springer New York, p. 497–528, 2006.

OTSU, N. A threshold selection method from gray-level histograms. *Automatica*, v. 11, n. 285-296, p. 23–27, 1975.

PAKRASHI, A.; CHAUDHURI, B. B. A kalman filtering induced heuristic optimization based partitional data clustering. *Information Sciences*, Elsevier, v. 369, p. 704–717, 2016.

PAL, N. R.; PAL, S. K. A review on image segmentation techniques. *Pattern recognition*, Elsevier, v. 26, n. 9, p. 1277–1294, 1993.

PAL, S. K.; MAJUMDER, D. D. Fuzzy sets and decision making approaches in vowel and speaker recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, v. 7, n. 8, p. 625–629, 1977.

RASMUSSEN, E. Information retrieval. In: FRAKES, W. B.; BAEZA-YATES, R. (Ed.). Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992. cap. Clustering Algorithms, p. 419–442. ISBN 0-13-463837-9. Disponível em: <http://dl.acm.org-/citation.cfm?id=129687.129703>.

SCRIVENER, K. L. The microstructure of anhydrous cement and its effect on hydration. In: CAMBRIDGE UNIV PRESS. *MRS Proceedings*. [S.l.], 1986. v. 85, p. 39.

SCRIVENER, K. L. Backscattered electron imaging of cementitious microstructures: understanding and quantification. *Cement and Concrete Composites*, Elsevier, v. 26, n. 8, p. 935–945, 2004.

SELIM, S. Z.; ALSULTAN, K. A simulated annealing algorithm for the clustering problem. *Pattern recognition*, Elsevier, v. 24, n. 10, p. 1003–1008, 1991.

SHABANZADEH, P.; YUSOF, R. An efficient optimization method for solving unsupervised data classification problems. *Computational and mathematical methods in medicine*, Hindawi Publishing Corporation, v. 2015, 2015.

SHELOKAR, P.; JAYARAMAN, V.; KULKARNI, B. An ant colony approach for clustering. *Analytica Chimica Acta*, v. 509, n. 2, p. 187 – 195, 2004. ISSN 0003-2670. Disponível em: <http://www.sciencedirect.com/science/article/pii-/S0003267003016374>.

SHI, Y.; EBERHART, R. C. Empirical study of particle swarm optimization. In: IEEE. *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. [S.l.], 1999. v. 3.

SOUZA, M. J. F. Inteligência computacional para otimização. *Notas de aula, Departamento de Computação, Universidade Federal de Ouro Preto, disponível em http://www. decom. ufop. br/prof/marcone/InteligenciaComputacional/InteligenciaComputacional. pdf*, 2008.

SUN, J.; FENG, B.; XU, W. Particle swarm optimization with particles having quantum behavior. In: *Congress on Evolutionary Computation*. [S.l.: s.n.], 2004.

SUN, J.; XU, W.; FENG, B. Adaptive parameter control for quantum-behaved particle swarm optimization on individual level. In: IEEE. *Systems, Man and Cybernetics, 2005 IEEE International Conference on*. [S.l.], 2005. v. 4, p. 3049–3054.

TARJAN, R. Depth-first search and linear graph algorithms. *SIAM journal on computing*, SIAM, v. 1, n. 2, p. 146–160, 1972.

TSAI, C.-Y.; KAO, I.-W. Particle swarm optimization with selective particle regeneration for data clustering. *Expert Systems with Applications*, Elsevier, v. 38, n. 6, p. 6565–6576, 2011.

VIEIRA, A. d. A. P. et al. Microdureza aplicada ao estudo do dano em revestimentos compósitos para superfícies metálicas. Universidade Federal da Paraíba, 2010.

WANG, R. et al. Flower pollination algorithm with bee pollinator for cluster analysis. *Information Processing Letters*, Elsevier, v. 116, n. 1, p. 1–14, 2016.

WEI, C.; KANGLING, F. Multilevel thresholding algorithm based on particle swarm optimization for image segmentation. In: IEEE. *Control Conference, 2008. CCC 2008. 27th Chinese*. [S.l.], 2008. p. 348–351.

WEST, D. B. et al. *Introduction to graph theory*. [S.l.]: Prentice hall Upper Saddle River, 2001.

WHITLEY, D. A genetic algorithm tutorial. *Statistics and computing*, Springer, v. 4, n. 2, p. 65–85, 1994.

WONG, H.; HEAD, M.; BUENFELD, N. Pore segmentation of cement-based materials from backscattered electron images. *Cement and Concrete Research*, Elsevier, v. 36, n. 6, p. 1083–1090, 2006.

YANG, R.; BUENFELD, N. Binary segmentation of aggregate in sem image analysis of concrete. *Cement and Concrete Research*, Elsevier, v. 31, n. 3, p. 437–441, 2001.

YE, Y. *Interior point algorithms: theory and analysis*. [S.l.]: John Wiley & Sons, 2011.

YEH, W.-C.; LAI, C.-M. Accelerated simplified swarm optimization with exploitation search scheme for data clustering. *PloS one*, Public Library of Science, v. 10, n. 9, p. e0137246, 2015.

ZHANG, C.; OUYANG, D.; NING, J. An artificial bee colony approach for clustering. *Expert Systems with Applications*, Elsevier, v. 37, n. 7, p. 4761–4767, 2010.

ZOU, Q. et al. Cracktree: Automatic crack detection from pavement images. *Pattern Recognition Letters*, Elsevier, v. 33, n. 3, p. 227–238, 2012.