# Fundamentação Teórica e Validação de Meta-heurísticas para o Problema de Particionamento de Grafos

## I. Princípios Fundamentais do Problema de Particionamento de Grafos

Esta seção estabelece a base teórica sobre a qual toda a análise subsequente é construída. Ela define formalmente o problema, contextualiza sua dificuldade computacional e introduz o paradigma dominante utilizado nos solucionadores heurísticos de ponta.

### 1.1. Definição Formal, Complexidade e Objetivos Centrais

O Problema de Particionamento de Grafos (GPP, do inglês *Graph Partitioning Problem*) é uma tarefa onipresente em diversas áreas da ciência da computação e engenharia.[1] Formalmente, dado um grafo não direcionado

$G=(V,E)$, onde $V$ é o conjunto de vértices (nós) e $E$ é o conjunto de arestas, potencialmente com pesos associados aos vértices $c:V\rightarrow R\geq 0$ e às arestas $\omega:E\rightarrow R>0$, o problema de particionamento k-way consiste em dividir o conjunto de vértices $V$ em $k$ subconjuntos disjuntos, também chamados de partições ou blocos, $V1,V2,...,Vk$.[3]

Os objetivos primários do particionamento de grafos são tipicamente duplos e concorrentes:

1. **Minimização do Corte (Cut Size):** O objetivo mais proeminente é minimizar o "tamanho do corte", que é a soma dos pesos das arestas que conectam vértices em partições diferentes. O corte entre duas partições $Vr$ e $Vs$ é definido como $Cut(Vr,Vs)=\Sigma u\in Vr,v\in Vs\omega(u,v)$. O objetivo é minimizar o corte total, $\Sigma i<jCut(Vi,Vj)$.[2] Minimizar o corte equivale a maximizar a conectividade interna das partições, um

princípio fundamental em aplicações como a detecção de comunidades em redes sociais e a minimização da comunicação em computação paralela.[1]

2. **Manutenção do Equilíbrio (Balance):** Para ser útil na prática, a partição deve ser equilibrada. Isso significa que a soma dos pesos dos vértices em cada partição deve ser aproximadamente igual. Formalmente, o peso de cada partição Vi não deve exceder um tamanho máximo permitido, geralmente definido como $(1+\varepsilon)\times kc(V)$, onde c(V) é o peso total de todos os vértices e ε é um fator de desequilíbrio máximo permitido.[3] Esta restrição é crucial para aplicações como balanceamento de carga em simulações científicas, onde cada partição corresponde à carga de trabalho de um processador.[6]

A principal razão para a extensa pesquisa em métodos heurísticos para o GPP reside em sua complexidade computacional. O problema de particionamento de grafos balanceado é **NP-completo** para $k \geq 2$.[3] Isso implica que não existe um algoritmo conhecido que possa encontrar a solução ótima para todas as instâncias do problema em tempo polinomial. Consequentemente, para grafos de grande escala encontrados em aplicações do mundo real, a busca por soluções ótimas exatas é computacionalmente inviável, tornando o uso de heurísticas e meta-heurísticas não apenas uma opção, mas uma necessidade.[3]

A importância do GPP é sublinhada por sua vasta gama de aplicações. Em computação de alto desempenho, é usado para mapear simulações científicas em processadores paralelos para minimizar a comunicação entre eles.[3] No projeto de circuitos VLSI, o particionamento é usado para agrupar componentes em sub-circuitos, otimizando o layout e reduzindo o comprimento dos fios.[10] Em análise de redes sociais, o particionamento é análogo à detecção de comunidades, identificando grupos de indivíduos densamente conectados.[1] Outras aplicações incluem segmentação de imagens, onde pixels são agrupados para identificar objetos, e bioinformática.[2]

## 1.2. O Paradigma Algorítmico Dominante: O Método Multinível

Diante da complexidade do GPP, a comunidade de pesquisa desenvolveu várias abordagens heurísticas. Entre elas, o **método multinível** (*multilevel method*) emergiu como a estrutura algorítmica mais bem-sucedida e amplamente adotada para particionar grafos de grande escala de forma eficiente e com alta qualidade.[2] A abordagem multinível não é um único algoritmo, mas sim um paradigma que orquestra várias heurísticas mais simples em um processo de três fases.

As três fases principais do processo multinível são:

1. **Agregação (Coarsening):** Nesta fase, o grafo original é recursivamente contraído para criar uma série de grafos menores e mais grosseiros, cada um preservando a estrutura

essencial do nível anterior. A contração é tipicamente alcançada através de algoritmos de casamento de arestas (*edge matching*), onde pares de vértices conectados são fundidos em um único "super-vértice" no próximo nível do grafo. O objetivo é reduzir drasticamente o tamanho do grafo, mantendo sua topologia global.[3]

2. **Particionamento Inicial (Initial Partitioning):** Uma vez que o grafo tenha sido agregado até um tamanho gerenciável (geralmente algumas centenas de vértices), um algoritmo de particionamento é aplicado a este grafo menor e mais grosseiro. Como o grafo é pequeno, métodos mais caros computacionalmente, que seriam inviáveis no grafo original, podem ser usados para encontrar uma partição inicial de alta qualidade.[3]
3. **Desagregação e Refinamento (Uncoarsening and Refinement):** A partição obtida no grafo mais grosseiro é então projetada de volta através da hierarquia de grafos, um nível de cada vez. Em cada etapa de desagregação, a partição é refinada usando uma heurística de melhoria local. Algoritmos de refinamento, como as variantes de Kernighan-Lin (KL) ou Fiduccia-Mattheyses (FM), movem iterativamente os vértices entre as partições para reduzir o tamanho do corte, aproveitando o fato de que as soluções em níveis mais grosseiros fornecem um excelente ponto de partida para o refinamento em níveis mais finos.[3]

A análise dos solucionadores de GPP de ponta revela uma mudança de paradigma significativa. O foco se deslocou da seleção de um único algoritmo monolítico para a orquestração de um processo multifásico. A abordagem multinível exemplifica essa mudança, pois não é um algoritmo único, mas uma estrutura que combina de forma inteligente múltiplas heurísticas (agregação, particionamento inicial, refinamento local) em um todo coeso e poderoso. Isso tem uma implicação profunda na validação das meta-heurísticas em questão (Guloso, SA, GA). Elas não são tipicamente aplicadas diretamente a um grafo massivo. Em vez disso, seu papel mais eficaz é como componentes *dentro* da estrutura multinível. Por exemplo, uma heurística gulosa rápida pode ser ideal para o particionamento inicial do grafo mais grosseiro, enquanto um algoritmo genético ou simulated annealing pode servir como um motor de refinamento poderoso durante a fase de desagregação. Portanto, a validação dessas meta-heurísticas requer não apenas uma análise isolada, mas também uma compreensão de seus papéis potenciais dentro deste paradigma dominante. A questão muda de "qual algoritmo é o melhor?" para "como cada algoritmo pode contribuir da melhor forma para uma estrutura de particionamento de ponta?".

# II. Validação do Portfólio de Meta-heurísticas Centrais

Esta seção aborda diretamente a validação dos algoritmos selecionados (Guloso, Simulated Annealing e Algoritmo Genético). A análise é comparativa, destacando seus mecanismos

distintos, pontos fortes e fracos, especificamente no contexto do GPP.

## 2.1. O Papel das Heurísticas Gulosas como Linha de Base Fundamental

As heurísticas gulosas representam uma das abordagens mais simples e diretas para problemas de otimização combinatória. No contexto do GPP, um exemplo proeminente é o algoritmo de crescimento de partição de grafo (GGP, *graph growing partitioning*). Nesse método, uma partição é iniciada com um vértice semente aleatório e é expandida iterativamente adicionando o vértice adjacente que oferece o maior "ganho" (por exemplo, maximizando a conectividade interna) até que a restrição de equilíbrio da partição seja satisfeita.[4]

O principal atrativo dos algoritmos gulosos é sua velocidade e simplicidade de implementação. Eles podem produzir rapidamente soluções razoáveis, embora frequentemente subótimas. Sua natureza determinística (dado um ponto de partida fixo) pode ser vantajosa para a reprodutibilidade. No entanto, sua principal desvantagem é a alta suscetibilidade a ficarem presos em ótimos locais; uma vez que uma decisão "gulosa" é tomada, ela nunca é revertida, o que pode levar a soluções de baixa qualidade no geral.[14]

Nos solucionadores modernos de GPP, o papel principal das heurísticas gulosas é duplo: servir como uma linha de base robusta para comparação e atuar como um componente dentro de esquemas mais complexos. Devido à sua velocidade, são frequentemente usadas para gerar a partição inicial no grafo mais grosseiro do método multinível.[4] Além disso, a importância de baselines simples e fortes não pode ser subestimada. Análises de benchmark recentes para problemas relacionados, como o Maximum Cut, demonstraram que heurísticas de busca local simples, incluindo abordagens gulosas, podem superar em desempenho abordagens de aprendizado de máquina altamente citadas em termos de qualidade da solução, tempo de inferência e escalabilidade.[15] Isso valida a inclusão de um algoritmo guloso não apenas como um ponto de partida, mas como um benchmark de desempenho sério. Superar uma heurística gulosa rápida e bem implementada é o primeiro obstáculo não trivial para qualquer meta-heurística mais complexa.

## 2.2. Simulated Annealing (SA): Uma Abordagem Probabilística para Escapar de Ótimos Locais

O Simulated Annealing (SA) é uma meta-heurística baseada em trajetória que se inspira no

processo de recozimento em metalurgia. Ele explora o espaço de soluções começando com uma solução inicial e movendo-se iterativamente para soluções vizinhas. A vizinhança para o GPP é tipicamente definida por movimentos simples, como mover um único vértice para outra partição ou trocar dois vértices entre partições diferentes. A característica distintiva do SA é seu critério de aceitação. Ele sempre aceita movimentos que melhoram a qualidade da solução (reduzem o corte). No entanto, ele também aceita movimentos que pioram a solução com uma certa probabilidade, que é função de um parâmetro de controle chamado "temperatura" (T) e da magnitude da piora (ΔE). A probabilidade de aceitar um movimento de piora é geralmente dada por $e^{-\Delta E/T}$.[16]

Essa capacidade de aceitar movimentos de piora permite que o SA escape de ótimos locais e explore o espaço de busca de forma mais ampla. À medida que o algoritmo progride, a temperatura é gradualmente reduzida de acordo com um "cronograma de recozimento", diminuindo a probabilidade de aceitar movimentos ruins e fazendo com que o algoritmo eventualmente convirja para uma solução de alta qualidade.[17]

O SA tem sido aplicado com sucesso ao GPP.[6] Seu desempenho, no entanto, é altamente sensível à escolha do cronograma de recozimento. Estudos comparativos históricos indicaram que a eficácia relativa do SA em comparação com outras heurísticas, como o algoritmo de Kernighan-Lin, pode depender da estrutura do grafo. O SA tende a ter uma vantagem em grafos esparsos e relativamente uniformes, enquanto heurísticas baseadas em trocas, como o KL, podem ser superiores para grafos com uma estrutura mais definida.[19]

## 2.3. Algoritmos Genéticos (GA): Busca Baseada em População e Hibridização

Os Algoritmos Genéticos (GAs) são meta-heurísticas baseadas em população que mantêm e evoluem um conjunto de soluções candidatas (uma "população" de partições) ao longo de várias "gerações". O processo evolutivo é impulsionado por operadores inspirados na evolução natural:

- **Seleção:** Soluções de maior qualidade (com menor corte e melhor equilíbrio) têm maior probabilidade de serem selecionadas para reprodução.
- **Crossover (Recombinação):** Duas soluções "parentais" são combinadas para criar uma ou mais soluções "filhas", com o objetivo de mesclar as boas características de ambos os pais.
- **Mutação:** Pequenas alterações aleatórias são introduzidas nas soluções filhas para manter a diversidade na população e evitar a convergência prematura.[16]

A verdadeira força dos GAs em solucionadores modernos de GPP não reside em sua forma

pura, mas em sua capacidade de hibridização. A tendência clara na pesquisa de ponta é a transição de meta-heurísticas puras para algoritmos híbridos e meméticos.

- **Algoritmos Meméticos:** Um GA é frequentemente combinado com uma heurística de busca local (como KL/FM), que é aplicada a cada solução filha gerada pelo crossover. Essa abordagem, conhecida como algoritmo memético, combina a exploração global do GA com a poderosa exploração local da heurística de refinamento, resultando em uma convergência muito mais rápida para soluções de alta qualidade.[2]
- **Híbridos Multinível:** Os solucionadores mais avançados, como o KaFFPaE (*KaFFPa-Evolutionary*), utilizam um GA como uma estratégia de busca de alto nível que orienta um particionador multinível completo (KaFFPa). Nesse sistema, os operadores de crossover e mutação são altamente sofisticados; em vez de operar em representações de baixo nível, eles combinam características de partições de alta qualidade encontradas por execuções completas do motor multinível subjacente.[3]

Estudos comparativos em domínios de particionamento, como o layout de circuitos VLSI, frequentemente incluem GAs e demonstram seu forte desempenho, embora abordagens mais recentes possam oferecer maior velocidade.[11] A validação do portfólio de algoritmos, portanto, não se trata de uma simples classificação. Trata-se de caracterizar os papéis estratégicos: o algoritmo guloso serve como uma linha de base essencial e um componente rápido; o SA é um explorador robusto de solução única; e o GA é uma estrutura poderosa para busca global, mas que atinge seu potencial máximo apenas quando hibridizado com busca local dentro de um paradigma multinível.

| Algoritmo | Estratégia de Busca | Principais Pontos Fortes | Principais Pontos Fracos | Papel Típico em Solucionadores de GPP | Referências Relevantes |
|---|---|---|---|---|---|
| **Algoritmo Guloso** | Local, determinístico, construtivo | Muito rápido; simples de implementar; bom para gerar soluções iniciais razoáveis. | Altamente suscetível a ótimos locais; qualidade da solução geralmente subótima. | Linha de base para comparação; particionador inicial para o grafo mais grosseiro em métodos | [4] |

| | | | | | |
|---|---|---|---|---|---|
| | | | | multinível. | |
| **Simulated Annealing (SA)** | Baseado em trajetória, probabilístico | Capacidade de escapar de ótimos locais; conceitualmente elegante; robusto para problemas pouco conhecidos. | Desempenho altamente dependente do cronograma de recozimento; pode ser lento para convergir. | Refinador de solução única; explorador de vizinhança para problemas com estruturas de grafo específicas (esparsas, uniformes). | [6] |
| **Algoritmo Genético (GA)** | Baseado em população, estocástico | Exploração global robusta; inerentemente paralelo; excelente para hibridização com busca local (algoritmos meméticos). | Pode ser lento para convergir em sua forma pura; o design de operadores de crossover eficazes é crucial e não trivial. | Estrutura de busca de alto nível que guia heurísticas locais ou solucionadores multinível completos (por exemplo, KaFFPaE). | [2] |

# III. Operadores de Crossover Avançados para Algoritmos Genéticos em Particionamento de Grafos

A eficácia de um Algoritmo Genético depende criticamente do design de seus operadores, especialmente o operador de crossover. Para problemas com uma estrutura inerente forte, como o GPP, a escolha do crossover é fundamental. Esta seção aprofunda um desafio técnico

específico: o desenvolvimento de operadores de GA que respeitem a estrutura do problema, o que é crucial para uma busca eficiente.

## 3.1. O Desafio: A Ruptura dos Blocos de Construção

Operadores de crossover padrão e "cegos", como o crossover de 1 ponto, k-pontos ou uniforme, tratam o cromossomo (a representação da partição) como uma simples cadeia de bits ou inteiros. Ao aplicar um ponto de corte e trocar segmentos, esses operadores não têm conhecimento da topologia do grafo que o cromossomo codifica. Como resultado, eles são frequentemente altamente disruptivos para o GPP.[22]

Eles podem destruir "blocos de construção" — grupos de vértices que estão fortemente conectados no grafo e que idealmente deveriam permanecer na mesma partição. Quando um crossover divide esses blocos, ele gera soluções filhas de baixa qualidade que são fundamentalmente inválidas ou distantes de qualquer ótimo local. Essas soluções geralmente requerem mecanismos de reparo extensivos, o que efetivamente transforma o crossover em uma forma de mutação em larga escala, minando seu propósito de combinar de forma inteligente as boas características dos pais.

## 3.2. Análise de Operadores Conscientes da Estrutura

Para superar as limitações dos operadores padrão, a pesquisa se concentrou no desenvolvimento de operadores de crossover que preservam a estrutura. A eficácia de um operador de crossover não é uma propriedade intrínseca, mas está profundamente acoplada à representação do cromossomo escolhida. A abordagem mais eficaz é co-projetá-los. Uma representação inteligente pode tornar um operador simples poderoso e consciente da estrutura.

- **Representação de Adjacência Baseada em Locus:** Uma abordagem elegante envolve uma representação onde o valor do gene na posição i é um nó j, significando uma conexão (i,j) no grafo da solução.[24] Nesse esquema, cada indivíduo na população do GA representa um subgrafo do grafo original. As partições (comunidades) são então determinadas pelos componentes conectados desse subgrafo. A principal vantagem é que um operador simples como o
  **Crossover Uniforme** se torna inerentemente preservador da estrutura. Ao criar um filho, para cada gene i, ele seleciona o alelo (o nó j) de um dos dois pais. Como o filho em cada posição i contém um valor j vindo de um dos pais, a aresta (i,j) existia em pelo menos um

dos pais. Isso garante que o grafo do filho seja composto inteiramente por arestas parentais, preservando os blocos de construção estruturais.[24]

- **Operadores Baseados em Permutação (por exemplo, Edge Recombination Crossover - ERX):** Embora projetado principalmente para problemas de permutação como o Problema do Caixeiro Viajante (TSP), a lógica do ERX é instrutiva. O ERX funciona construindo um filho que preserva o máximo possível das adjacências (vizinhos) de ambos os pais.[25] Para o GPP, isso pode ser adaptado para uma representação baseada em agrupamento. A ideia seria construir uma partição filha onde a probabilidade de dois vértices
  u e v estarem na mesma partição é maior se eles estavam na mesma partição em um ou ambos os pais.
- **Crossovers Geométricos:** Esta é uma estrutura mais abstrata, poderosa e independente da representação para projetar operadores de crossover.[28] O conceito central é que os filhos são definidos como soluções que se encontram em um "caminho mais curto" entre os pais no espaço de solução, de acordo com uma métrica de distância específica. A aplicação deste conceito força o projetista a primeiro definir uma "distância" significativa entre duas partições, o que por si só requer um pensamento estrutural sobre o problema.
  - Para problemas de agrupamento como o GPP, um desafio chave é a redundância da codificação (por exemplo, trocar os rótulos das partições {1,2} para {2,1} resulta na mesma partição). Um **crossover independente de rotulagem** pode ser projetado para superar isso.
  - Além disso, um **crossover de ciclo** para permutações com repetições pode ser usado para garantir que, se os pais têm k partições de tamanhos específicos, o filho também terá k partições dos mesmos tamanhos, preservando a viabilidade sem a necessidade de reparo. A combinação dessas duas ideias resulta em um crossover geométrico altamente eficaz para o GPP, que supera os operadores anteriores.[28]

| Operador | Princípio Subjacente | Representação do Cromossomo | Lógica de Preservação | Vantagens | Desvantagens | Referências Relevantes |
|---|---|---|---|---|---|---|
| **Uniforme sobre Base de Locus** | Combinação de arestas parentais | Adjacência baseada em locus (gene i = nó j) | O filho é construído apenas com arestas (i,j) que existiam | Simples, elegante, preserva inerentemente os componentes | A representação pode ser menos intuitiva do que a | [24] |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | nos pais. | conectados dos pais. O número de partições é determinado emergentemente. | representação de agrupamento direto. | |
| **Edge Recombination (ERX)** | Preservação de adjacências | Permutação (indireta para GPP) | Constrói um filho priorizando as adjacências (vizinhos) presentes nos pais. | Focado na preservação da vizinhança local, que é um forte indicador de estrutura. | Projetado para TSP; a adaptação para GPP não é trivial e pode ser menos direta do que outras abordagens. | [25] |
| **Ciclo Geométrico / Independente de Rotulagem** | Distância no espaço de solução | Agrupamento (permutação com repetições) | O filho está no "caminho mais curto" entre os pais. O crossover de ciclo preserva o número e os tamanhos das | Formalmente robusto; preserva a viabilidade (tamanhos das partições) sem necessidade de reparo; | Conceitualmente mais complexo; a implementação pode ser mais desafiadora do que a de operadores mais | [28] |

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | partições. O componente independente de rotulagem ignora a numeração arbitrária das partições. | lida com a redundância da codificação. | simples. | |

# IV. Adaptação de Meta-heurísticas Contínuas para Otimização Discreta

Esta seção fornece o arcabouço teórico necessário para aplicar algoritmos originalmente projetados para espaços de busca contínuos (como PSO, TLBO e GWO) ao problema discreto de particionamento de grafos, um requisito central da consulta.

## 4.1. A Dicotomia Contínuo-Discreta

O desafio fundamental reside no fato de que meta-heurísticas como Particle Swarm Optimization (PSO), Teaching-Learning-Based Optimization (TLBO) e Grey Wolf Optimizer (GWO) foram concebidas para operar em espaços de busca contínuos e de valor real. Suas mecânicas centrais — como a atualização de posição e velocidade através de aritmética vetorial (adição, subtração, multiplicação por escalar) — não são diretamente aplicáveis a problemas de otimização combinatória como o GPP. Em problemas discretos, uma solução não é um ponto em um espaço euclidiano, mas sim uma atribuição combinatória de elementos a conjuntos, como a atribuição de vértices a partições.[16] A tentativa de aplicar diretamente esses operadores contínuos a um problema discreto resultaria em soluções

inválidas. Portanto, são necessárias técnicas de adaptação sistemáticas.

## 4.2. Melhores Práticas em Discretização: O Arcabouço de Binarização

A abordagem mais comum e eficaz para adaptar essas meta-heurísticas contínuas para problemas de otimização binária (uma forma de otimização discreta) é o "esquema de binarização em duas etapas".[30] Este método funciona como uma ponte conceitual, permitindo que a lógica de busca do algoritmo contínuo guie a tomada de decisões no domínio discreto.

1. **Etapa 1: Função de Transferência:** A primeira etapa envolve a aplicação de uma função de transferência. O algoritmo contínuo executa sua operação de atualização de posição ou velocidade normalmente, resultando em um vetor de valores reais. Cada componente deste vetor (correspondendo a uma dimensão da solução, por exemplo, um vértice do grafo) é então passado por uma função de transferência. O papel desta função é mapear o valor real, que pode estar em um intervalo ilimitado, para o intervalo $$. O valor resultante é interpretado não como uma posição, mas como a *probabilidade* de a correspondente dimensão da solução discreta assumir o valor '1'.[30] As funções de transferência mais comuns são em forma de S (sigmoide) ou em forma de V.
2. **Etapa 2: Regra de Binarização:** A probabilidade gerada pela função de transferência é então usada por uma regra de binarização para tomar uma decisão discreta final (0 ou 1). A regra mais simples é estocástica: um número aleatório r é gerado no intervalo $$, e se r for menor que a probabilidade calculada, a posição do bit é definida como 1; caso contrário, é 0. Existem também regras mais complexas, como estratégias elitistas, que podem levar em consideração o estado anterior da solução.[30]

Este processo não é uma simples operação de arredondamento ou truncamento, que perderia a maior parte da informação direcional contida no vetor de velocidade. Em vez disso, ele funciona como uma "ponte probabilística". A lógica de busca do algoritmo contínuo calcula um vetor de movimento que possui magnitude e direção, indicando uma direção promissora no espaço de busca. A função de transferência converte essa informação vetorial em um vetor de probabilidades. Uma grande velocidade positiva em uma dimensão se traduz em uma alta probabilidade de essa dimensão se tornar '1', enquanto uma grande velocidade negativa se traduz em uma baixa probabilidade. A regra de binarização então amostra a partir dessa distribuição de probabilidade para construir a nova solução discreta. Este processo permite que o algoritmo "direcione" a busca no domínio discreto usando a mesma lógica com a qual foi projetado para o domínio contínuo, preservando a inteligência da heurística original.

## 4.3. Estratégias Alternativas e Híbridas

Embora o arcabouço de binarização em duas etapas seja o mais prevalente, existem outras abordagens. Pesquisas mais recentes exploraram o uso de técnicas de aprendizado de máquina para guiar o processo de binarização. Por exemplo, o algoritmo k-means pode ser usado para agrupar as soluções contínuas e informar a discretização, modificando a operação normal da meta-heurística para melhorar sua capacidade de encontrar soluções ótimas.[30] Outras abordagens incluem métodos binários inspirados na computação quântica, que utilizam princípios como Q-bits para representar soluções.[30] Essas técnicas representam direções de pesquisa mais avançadas e podem oferecer melhorias de desempenho em domínios de problemas específicos.

# V. Análise de Algoritmos de Enxame e Inspirados na Natureza para Particionamento de Grafos

Esta seção aplica o arcabouço de adaptação da Seção IV aos algoritmos específicos solicitados (PSO, TLBO, GWO), fornecendo detalhes concretos de implementação e citando aplicações documentadas. Embora as metáforas inspiradoras sejam muito diferentes (enxames de partículas, uma sala de aula, uma matilha de lobos), as soluções de engenharia subjacentes para adaptá-las a problemas discretos são notavelmente semelhantes. Todas são meta-heurísticas baseadas em população, onde as posições dos indivíduos são atualizadas com base em informações de outros indivíduos de melhor desempenho. O arcabouço de binarização funciona como um "adaptador" de propósito geral que pode ser conectado a qualquer um desses algoritmos. A lógica central do PSO (usando pbest, gbest) ou do GWO (usando $\alpha,\beta,\delta$) permanece, mas a etapa final de atualização da posição é roteada através deste adaptador comum. Isso demonstra um poderoso princípio de engenharia de abstração e modularidade, onde a lógica da meta-heurística pode ser separada da lógica de discretização.

## 5.1. Particle Swarm Optimization (PSO)

O Particle Swarm Optimization (PSO) é uma meta-heurística inspirada no comportamento social de bandos de pássaros ou cardumes de peixes. Cada "partícula" no enxame representa uma solução candidata e "voa" através do espaço de busca. O movimento de cada partícula é influenciado por sua própria melhor posição encontrada até agora (pbest) e pela melhor

posição encontrada por todo o enxame (gbest).[32]

- **Representação da Solução:** Para o particionamento de grafos de 2 vias, uma partícula pode ser representada como um vetor binário de comprimento n, onde n é o número de vértices. Um valor de '0' na posição i significa que o vértice i pertence à partição A, e um valor de '1' significa que pertence à partição B.[32] Para o particionamento k-way, a representação pode ser estendida para usar vetores de inteiros ou múltiplas dimensões por vértice.
- **Mecanismo do PSO Discreto (DPSO):** A adaptação do PSO para o GPP, conhecida como PSO Discreto (DPSO), segue o arcabouço de binarização. As equações centrais de atualização de velocidade do PSO permanecem as mesmas, operando em vetores de velocidade de valor real. A adaptação crucial ocorre na atualização da posição da partícula (o vetor binário). A velocidade contínua atualizada Vi(t+1) para cada dimensão é passada por uma função de transferência sigmoide, Sig(Vi(t+1)), para obter uma probabilidade no intervalo $$. Essa probabilidade é então usada para decidir se o novo bit de posição Xi(t+1) será 0 ou 1, geralmente comparando-a com um número aleatório.[32]
- **Aplicações Documentadas:** O DPSO tem sido amplamente aplicado a problemas relacionados ao GPP. A literatura documenta seu uso bem-sucedido em particionamento de circuitos VLSI [10], no problema de partição máxima [32] e como uma fase de refinamento eficaz dentro de uma estrutura de particionamento multinível, onde é usado para melhorar uma partição inicial.[4]

## 5.2. Teaching-Learning-Based Optimization (TLBO)

O Teaching-Learning-Based Optimization (TLBO) é uma meta-heurística inspirada no ambiente de uma sala de aula. A população de soluções ("alunos") melhora em duas fases: a "Fase do Professor", onde os alunos aprendem com a melhor solução da população (o "professor"), e a "Fase do Aluno", onde os alunos interagem e aprendem uns com os outros.[33] Uma vantagem notável do TLBO é seu conceito "sem parâmetros específicos do algoritmo", o que significa que ele não requer o ajuste de parâmetros de controle como taxa de crossover ou mutação, ao contrário do GA ou PSO.[33]

- **Aplicação por Proxy: Problema de Atribuição Quadrática (QAP):** Embora a pesquisa não tenha revelado aplicações diretas do TLBO ao GPP, foi encontrada uma aplicação robusta e bem-sucedida ao Problema de Atribuição Quadrática (QAP), que é explicitamente observado como estando relacionado ao GPP.[33] Esta aplicação serve como um excelente modelo para a adaptação.
- **Adaptação Discreta:** A adaptação do TLBO para o QAP (um problema de permutação) não utilizou o arcabouço de binarização. Em vez disso, redefiniu as fases de "professor" e "aluno" usando operadores discretos. O "aprendizado" foi implementado através de

operadores de recombinação poderosos e sem parâmetros (como o Order Based Crossover) e hibridizado com um motor de Busca Tabu Robusta (RTS) para refinamento local.[33]

- **Inferência para o GPP:** Para o GPP, uma adaptação do TLBO poderia seguir duas vias. A primeira seria usar o arcabouço de binarização, com uma representação binária da partição. Uma abordagem potencialmente mais poderosa, inspirada na solução para o QAP, seria uma abordagem híbrida onde a solução do "professor" (uma partição de alta qualidade) é usada para guiar uma heurística dedicada ao GPP (como KL/FM) para melhorar as soluções dos "alunos".

## 5.3. Grey Wolf Optimizer (GWO)

O Grey Wolf Optimizer (GWO) é uma meta-heurística inspirada na hierarquia social e no comportamento de caça dos lobos cinzentos. A população de busca é organizada em quatro níveis: alfa (α), beta (β), delta (δ) e ômega (ω). Os lobos α, β e δ representam as três melhores soluções encontradas até o momento e guiam os outros lobos (ω) na busca pela presa (a solução ótima).[34]

- **Adaptação e Aplicação:** A aplicação direta do GWO ao GPP não é proeminente na pesquisa disponível. No entanto, seu uso bem-sucedido em problemas de particionamento e distritamento relacionados [35] e a existência de variantes de "GWO Discreto" (DGWO) são documentados, indicando sua aplicabilidade.[36]
- **Mecanismo do GWO Discreto:** A adaptação do GWO a um problema discreto como o GPP seguiria naturalmente o arcabouço de binarização da Seção IV. As posições dos lobos alfa, beta e delta seriam vetores binários representando as melhores partições. As equações contínuas de atualização de posição do GWO, que calculam a nova posição de um lobo ômega com base nas posições dos três líderes, seriam calculadas normalmente. Os vetores de posição resultantes, de valor real, seriam então passados por funções de transferência e regras de binarização para gerar as novas posições discretas (binárias) dos lobos ômega.[34]

| Algoritmo | Representação da Solução | Lógica de Atualização Central (Análogo Discreto) | Aplicações Documentadas Relacionadas ao GPP | Referências Relevantes |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| **Particle Swarm Optimization (PSO)** | Vetor binário de comprimento n (para 2-vias). | A velocidade é atualizada continuamente. A posição é atualizada usando uma função de transferência (sigmoide) na velocidade para obter uma probabilidade, seguida por uma regra de binarização. | Particionamento de circuitos; Problema de partição máxima; Refinamento em métodos multinível. | [4] |
| **Teaching-Learning-Based Optimization (TLBO)** | Vetor binário ou de inteiros. | *Abordagem 1 (Binarização):* Semelhante ao PSO, usando as soluções do professor e do colega para guiar as atualizações. *Abordagem 2 (Híbrida):* Usa operadores de recombinação discretos (crossover) para implementar as fases de professor e aluno. | Problema de Atribuição Quadrática (QAP), que está relacionado ao GPP. | [33] |
| **Grey Wolf Optimizer (GWO)** | Vetor binário de comprimento | As posições contínuas são calculadas | Problemas de distritamento/ particionament | [34] |

| | | | | |
|---|---|---|---|---|
| | n. | com base nos três líderes (α,β,δ). As novas posições discretas são geradas aplicando o arcabouço de binarização (função de transferência + regra de binarização) aos resultados. | o; variantes de GWO Discreto (DGWO) existem para otimização combinatória. | |

# VI. Estratégia de Validação Experimental e Benchmarking

A implementação correta de meta-heurísticas é apenas o primeiro passo; sua validação empírica rigorosa é o que confere credibilidade aos resultados. Esta seção final fornece diretrizes sobre como projetar e executar experimentos para validar os algoritmos escolhidos, uma etapa crítica em qualquer projeto de pesquisa sério. A confluência de algoritmos estocásticos, numerosos parâmetros ajustáveis e a falta de práticas de relatório padronizadas cria um risco significativo de resultados irreprodutíveis e alegações de superioridade enganosas. Esforços para criar benchmarks padronizados, como os desafios DIMACS e o LDBC Graphalytics, existem precisamente para combater esse problema.[1] Portanto, o sucesso de um projeto depende não apenas da implementação correta dos algoritmos, mas de sua avaliação rigorosa, mantendo um ceticismo saudável em relação a resultados publicados que não usam linhas de base fortes ou não fornecem análise estatística sobre múltiplas execuções.

## 6.1. Métricas de Qualidade para Particionamento

A escolha de uma métrica de qualidade apropriada é fundamental para orientar a otimização

e avaliar os resultados.

- **Foco na Modularidade:** Para o particionamento centrado na comunidade, a **modularidade** é uma métrica padrão e crucial. Ela quantifica a qualidade de uma partição medindo a fração de arestas que caem dentro das partições, subtraída da fração esperada se as arestas fossem distribuídas aleatoriamente, mantendo a mesma sequência de graus dos vértices.[24] Um valor de modularidade mais alto indica uma estrutura comunitária mais forte.[41]
- **Vantagens e Limitações da Modularidade:** A modularidade é amplamente reconhecida e fornece uma única pontuação que pode ser usada diretamente como função objetivo para a otimização.[41] No entanto, ela possui limitações conhecidas. A mais notável é o "limite de resolução", onde a modularidade pode falhar em identificar comunidades pequenas, mas bem definidas, em redes grandes. Além disso, a formulação padrão da modularidade normalmente ignora os nós desconectados dentro de uma mesma comunidade, uma limitação que é abordada por extensões como a Modularidade Max-Min, que penaliza a presença de nós não relacionados na mesma partição.[41]

## 6.2. A Importância de Benchmarks Padronizados

Uma avaliação robusta e comparável requer o teste dos algoritmos em conjuntos de dados padronizados.

- **Grafos do Mundo Real vs. Sintéticos:** É essencial testar em ambos os tipos de grafos. Grafos do mundo real (por exemplo, redes sociais, redes de colaboração) capturam a estrutura complexa, ruidosa e muitas vezes irregular de problemas práticos. Grafos sintéticos, por outro lado, permitem experimentos controlados onde parâmetros como tamanho, densidade e força da estrutura comunitária podem ser sistematicamente variados para testar os limites dos algoritmos.[43]
- **Coleções de Benchmark:** Para garantir a comparabilidade com a pesquisa existente, é fundamental utilizar coleções de benchmark estabelecidas. Exemplos importantes incluem os conjuntos de dados do **10º Desafio de Implementação DIMACS** sobre Particionamento e Agrupamento de Grafos, que fornecem instâncias de aplicações reais.[1] Para tarefas de análise de grafos mais amplas, o benchmark **LDBC Graphalytics**, de nível industrial, oferece um conjunto rigoroso de algoritmos, conjuntos de dados do mundo real e geradores de dados sintéticos para uma comparação objetiva de plataformas.[40]

## 6.3. Melhores Práticas para o Desenho Experimental

A natureza estocástica das meta-heurísticas exige um desenho experimental cuidadoso para produzir conclusões válidas.

- **Rigor Estatístico:** Uma única execução de uma meta-heurística não é suficiente para uma avaliação significativa. Devido à sua natureza aleatória, diferentes execuções produzirão resultados diferentes. É essencial realizar múltiplas execuções (tipicamente 30 ou mais) com sementes aleatórias diferentes e relatar medidas estatísticas agregadas, como a média, o desvio padrão, e os melhores e piores resultados obtidos.[46] Isso fornece uma imagem muito mais completa e honesta do desempenho do algoritmo.
- **Comparação Justa:** Ao comparar diferentes algoritmos, é crucial garantir uma base de comparação justa. Comparar com base apenas no número de gerações ou iterações pode ser enganoso, pois o custo computacional por geração pode variar drasticamente entre os algoritmos. Uma abordagem mais justa é alocar um orçamento computacional igual para cada algoritmo, seja em termos de tempo total de execução ou do número total de avaliações da função de aptidão.[46]
- **Linhas de Base Fortes:** As comparações são mais significativas quando feitas contra linhas de base fortes e bem estabelecidas, e não apenas contra outros métodos novos ou exóticos. Isso deve incluir heurísticas simples e rápidas (como algoritmos gulosos) e particionadores multinível padrão da indústria, como o METIS.[6] Como demonstrado em benchmarks recentes, superar uma heurística de busca local bem implementada é um desafio significativo e um requisito mínimo para reivindicar a superioridade de uma abordagem mais complexa.[15]

# VII. Síntese e Recomendações Estratégicas

## 7.1. Avaliação Consolidada do Portfólio Algorítmico

A análise aprofundada do problema de particionamento de grafos e das meta-heurísticas associadas revela que não existe uma solução única. O desempenho é dependente do contexto, e a abordagem mais eficaz é quase invariavelmente uma que combina os pontos fortes de múltiplas técnicas dentro de uma estrutura híbrida e multinível.

- **Algoritmo Guloso:** Validado como um componente essencial. Sua principal função é servir como uma linha de base de desempenho rápida e robusta e como um particionador inicial eficiente dentro do paradigma multinível. Sua simplicidade e

velocidade o tornam indispensável.

- **Simulated Annealing (SA):** Validado como um método de refinamento de solução única robusto e bem fundamentado. Sua capacidade de escapar de ótimos locais o torna valioso, especialmente para grafos com topologias menos estruturadas. No entanto, seu desempenho sensível aos parâmetros e sua natureza sequencial podem limitar sua escalabilidade em comparação com abordagens baseadas em população.
- **Algoritmo Genético (GA):** Validado como a estrutura mais poderosa para a exploração global do espaço de soluções. Seu verdadeiro potencial é desbloqueado através da hibridização. Um GA puro é muitas vezes ineficiente; um **algoritmo memético**, que integra uma busca local após cada operação de crossover, é significativamente mais eficaz. O uso de operadores de crossover que preservam a estrutura, co-projetados com a representação do cromossomo, é fundamental para o sucesso.
- **Algoritmos Contínuos (PSO, TLBO, GWO):** A aplicação desses algoritmos ao GPP é viável e teoricamente sólida através de mecanismos de adaptação, principalmente o arcabouço de binarização em duas etapas. O PSO Discreto (DPSO) é o mais documentado e maduro para problemas de particionamento. O TLBO e o GWO, embora menos explorados para o GPP especificamente, podem ser adaptados usando os mesmos princípios, oferecendo novas vias para a exploração de algoritmos sem parâmetros (TLBO) ou com diferentes dinâmicas de busca (GWO).

## 7.2. Diretrizes para Implementação e Trabalho Futuro

Com base nesta análise, um caminho estratégico para um projeto de pesquisa ou desenvolvimento sobre este tema pode ser delineado em fases:

1. **Fase 1: Estabelecimento da Linha de Base:** A primeira etapa deve ser a implementação de uma heurística de linha de base forte. Isso inclui um **algoritmo guloso** rápido (como o GGP) e, se possível, a integração de um particionador multinível padrão como o **METIS**. Esses algoritmos fornecerão o benchmark contra o qual todas as outras abordagens mais complexas devem ser medidas.
2. **Fase 2: Implementação das Meta-heurísticas Avançadas:**
    - Implementar o **Simulated Annealing** como um método de refinamento de solução única.
    - Desenvolver um **Algoritmo Genético**, mas projetá-lo desde o início como um **algoritmo memético**. Isso significa incorporar um passo de refinamento de busca local (por exemplo, uma heurística baseada em trocas como a KL/FM) que é aplicado a cada novo indivíduo. A pesquisa e implementação de um **operador de crossover que preserva a estrutura** (como o crossover uniforme em uma representação baseada em locus ou um crossover geométrico) deve ser uma prioridade.
    - Implementar uma versão discreta de um dos algoritmos contínuos, sendo o **DPSO** a

escolha mais segura devido à sua maior documentação na literatura para problemas de particionamento. Isso envolverá a implementação do arcabouço de binarização (função de transferência e regra de binarização).

3. **Fase 3: Benchmarking Rigoroso:** Conduzir uma campanha de benchmarking abrangente, seguindo as melhores práticas descritas na Seção VI. Utilizar conjuntos de dados de benchmarks padronizados (como os do desafio DIMACS). Para cada algoritmo, realizar múltiplas execuções com sementes aleatórias diferentes e coletar estatísticas de desempenho (melhor, pior, média, desvio padrão do tamanho do corte) sob um orçamento computacional justo (tempo de execução ou número de avaliações).

Em conclusão, o campo do particionamento de grafos é dinâmico e maduro. A contribuição mais significativa não virá da simples aplicação de uma meta-heurística conhecida, mas de uma aplicação cuidadosa, bem fundamentada e empiricamente validada de abordagens híbridas e estruturalmente conscientes, comparadas rigorosamente com as linhas de base do estado da arte.

**Referências citadas**


1. Graph Partitioning and Graph Clustering - David A. Bader, acessado em agosto 29, 2025, https://davidbader.net/publication/2013-bmsw/2013-bmsw.pdf
2. Recent Advances in Graph Partitioning, acessado em agosto 29, 2025, https://www.eecis.udel.edu/~isafro/papers/advpart.pdf
3. High Quality Graph Partitioning ?, acessado em agosto 29, 2025, https://sites.cc.gatech.edu/dimacs10/papers/%5B01%5D-high_quality_graph_partitioning_final.pdf
4. NOVEL MULTILEVEL PARTICLE SWARM OPTIMIZATION ALGORITHM FOR GRAPH PARTITIONING - SCIK Publishing Corporation, acessado em agosto 29, 2025, https://scik.org/index.php/jmcs/article/download/6917/3342
5. A fast and high quality multilevel scheme for partitioning irregular graphs - UT Computer Science, acessado em agosto 29, 2025, https://www.cs.utexas.edu/~pingali/CS395T/2009fa/papers/metis.pdf
6. JA-BE-JA: A Distributed Algorithm for Balanced Graph Partitioning - DiVA, acessado em agosto 29, 2025, http://kth.diva-portal.org/smash/get/diva2:668109/FULLTEXT01
7. More Recent Advances in (Hyper)Graph Partitioning - arXiv, acessado em agosto 29, 2025, https://arxiv.org/pdf/2205.13202
8. Shifting niches for community structure detection - SciSpace, acessado em agosto 29, 2025, https://scispace.com/pdf/shifting-niches-for-community-structure-detection-1xytbqzmuk.pdf
9. Figures of Graph Partitioning by Counting, Sequence and Layer Matrices - Semantic Scholar, acessado em agosto 29, 2025, https://pdfs.semanticscholar.org/8e81/7822d7c07936a26aebaaf67d320d0e57a4ec.pdf
10. A balanced computational approach for multilevel VLSI circuit partitioning

strategy with bat algorithm - AIMS Press, acessado em agosto 29, 2025, https://www.aimspress.com/aimspress-data/electreng/2025/3/PDF/electreng-09-03-018.pdf
11. IEEE Paper Template in A4 (V1) - Journal of computing technologies, acessado em agosto 29, 2025, http://jctjournals.com/august2012/v6.pdf
12. Evolutionary clustering algorithm for community detection using graph-based information - CMAP, acessado em agosto 29, 2025, http://www.cmap.polytechnique.fr/~nikolaus.hansen/proceedings/2014/WCCI/CEC-2014/PROGRAM/E-14680.pdf
13. Image Segmentation Using Swarm Intelligence ... - IOSR Journal, acessado em agosto 29, 2025, https://www.iosrjournals.org/iosr-jece/papers/Conf.17009-2017/Volume-2/4.%2017-22.pdf
14. A Greedy Strategy for Graph Cut - arXiv, acessado em agosto 29, 2025, https://arxiv.org/pdf/2412.20035
15. MaxCutBench: Revisiting and Benchmarking Graph Neural Networks for Maximum Cut - OpenReview, acessado em agosto 29, 2025, https://openreview.net/pdf?id=322PpCGAX8
16. Metaheuristics for Hard Optimization - National Academic Digital Library of Ethiopia, acessado em agosto 29, 2025, http://ndl.ethernet.edu.et/bitstream/123456789/48958/1/6..pdf
17. arXiv:cond-mat/0006374v1 [cond-mat.dis-nn] 23 Jun 2000, acessado em agosto 29, 2025, https://arxiv.org/pdf/cond-mat/0006374
18. Comparative Analysis of Genetic and Simulated Annealing Algorithms for Road Traffic Congestion Management, acessado em agosto 29, 2025, https://www.ijcst.org/Volume7/Issue5/p6_7_5.pdf
19. A seed-growth heuristic for graph bisection - Harvard DASH, acessado em agosto 29, 2025, https://dash.harvard.edu/bitstreams/7312037c-49f0-6bd4-e053-0100007fdf3b/download
20. Finding Optimal Cayley Map Embeddings Using Genetic Algorithms - Rollins Scholarship Onlin, acessado em agosto 29, 2025, https://scholarship.rollins.edu/cgi/viewcontent.cgi?article=1175&context=honors
21. Comparative Study of Various Evolutionary Approaches for Digital Circuit Layout based on Graph Partitioning Technique - IJCAIT, acessado em agosto 29, 2025, https://ijcait.com/32/321.pdf
22. GA Design Concepts - VTechWorks, acessado em agosto 29, 2025, https://vtechworks.lib.vt.edu/bitstream/handle/10919/27347/appB.pdf
23. Hybrid Genetic Algorithms for Bin-packing and Related Problems - CiteSeerX, acessado em agosto 29, 2025, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ddcc693febf882140dc77c4611413a91cca1b825
24. Boosting the Detection of Modular Community Structure with ... - CNR, acessado em agosto 29, 2025, https://staff.icar.cnr.it/pizzuti/pubblicazioni/SAC2012.pdf
25. TOWARDS A GEOMETRIC UNIFICATION OF EVOLUTIONARY ..., acessado em

agosto 29, 2025, https://eden.dei.uc.pt/~moraglio/Thesis_final.pdf
26. On metaheuristic algorithms for combinatorial ... - SciSpace, acessado em agosto 29, 2025, https://scispace.com/pdf/on-metaheuristic-algorithms-for-combinatorial-optimization-1cgjfzekeq.pdf
27. Handbook of Nature-inspired and Innovative Computing : Integrating Classical Models With Emerging Technologies - WordPress.com, acessado em agosto 29, 2025, https://kamenpenkov.files.wordpress.com/2016/01/zomaya-2006.pdf
28. Geometric Crossover for Permutations with Repetitions: Application ..., acessado em agosto 29, 2025, https://eden.dei.uc.pt/~moraglio/ppsn-workshop.pdf
29. Metaheuristics for (Variable-Size) Mixed Optimization Problems: A Unified Taxonomy and Survey - arXiv, acessado em agosto 29, 2025, https://arxiv.org/pdf/2401.03880
30. Binarization of Metaheuristics: Is the Transfer Function Really ..., acessado em agosto 29, 2025, https://zeus.inf.ucv.cl/~bcrawford/MII-902-OptimizacionEstocastica/Binarizacion/Binarization%20of%20Metaheuristics%20Is%20the%20Transfer%20Function%20Really%20Important-biomimetics-08-00400.pdf
31. Enhancing a machine learning binarization framework by perturbation operators: analysis on the multidimensional knapsack problem, acessado em agosto 29, 2025, https://ris.utwente.nl/ws/portalfiles/portal/178150068/Garcia_2020_Enhancing_a_machine_learning_binari.pdf
32. HYBRIDIZATION OF PARTICLE SWARM OPTIMIZATION AND ..., acessado em agosto 29, 2025, https://scik.org/index.php/jmcs/article/download/5513/2709
33. Hybrid teaching-learning-based optimization algorithms for the Quadratic Assignment Problem - METU/CEng, acessado em agosto 29, 2025, https://user.ceng.metu.edu.tr/~e1451970/tlbo-qap.pdf
34. An Improved Grey Wolf Optimization with Multi ... - Semantic Scholar, acessado em agosto 29, 2025, https://pdfs.semanticscholar.org/439b/06c8e53cb4b966a843f6fc6c2bc3424c950d.pdf
35. A novel optimization model for designing compact, balanced, and contiguous healthcare districts - Madjid Tavana, acessado em agosto 29, 2025, http://tavana.us/publications/JORS-HCD.pdf
36. 2020 IEEE International Conference on Systems, Man, and, acessado em agosto 29, 2025, https://www.proceedings.com/content/056/056961webtoc.pdf
37. THE 13TH CONFERENCE OF PHD STUDENTS IN COMPUTER SCIENCE, acessado em agosto 29, 2025, http://www.inf.u-szeged.hu/~cscs/cscs2022/pdf/cscs2022.pdf
38. Soft Computing - MDPI, acessado em agosto 29, 2025, https://mdpi-res.com/bookfiles/topic/9951/Soft_Computing.pdf?v=1745975227
39. CIE51 HANDBOOK & PROGRAM SCHEDULE - OPUS at UTS, acessado em agosto 29, 2025, https://opus.lib.uts.edu.au/bitstream/10453/185541/3/CIE51-%20Samira%20Alvan

di-%20%20Session%20SS6%20From%20Data%20to%20Action%20Leveraging%20Artificial%20Intelligence.pdf
40. LDBC Graphalytics: A Benchmark for Large-Scale Graph Analysis ..., acessado em agosto 29, 2025, https://www.vldb.org/pvldb/vol9/p1317-iosup.pdf
41. Toward an Optimal Solution to the Network Partitioning Problem, acessado em agosto 29, 2025, https://annals-csis.org/Volume_35/drp/pdf/2832.pdf
42. A Convex Formulation of Modularity Maximization for ... - IJCAI, acessado em agosto 29, 2025, https://www.ijcai.org/Proceedings/11/Papers/370.pdf
43. On the Efficiency and Programmability of Large Graph Processing in the Cloud - Microsoft, acessado em agosto 29, 2025, https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/Surfer_tr.pdf
44. An Investigation into Using Unsupervised Metrics to Optimise GNNs for Node Clustering - arXiv, acessado em agosto 29, 2025, https://www.arxiv.org/pdf/2402.07845v1
45. Random Graph Modeling: A survey of the concepts, acessado em agosto 29, 2025, https://www.ispras.ru/upload/iblock/838/838e4b7f25738401e63fd9ed20b05148.pdf
46. A comparison of the performance of different metaheuristics on the timetabling problem, acessado em agosto 29, 2025, https://www.metaheuristics.org/downloads/tt_camera_ready.pdf
47. A two stage converging genetic algorithm for graph clustering - Gigvvy Science, acessado em agosto 29, 2025, https://gigvvy.com/journals/ijase/articles/ijase-202009-17-3-299.pdf
48. On the Estimation of the Expected Performance of a Metaheuristic ..., acessado em agosto 29, 2025, https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=800a5f55ce2bb62892c00834779d4e83fd2fa759