Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Uma abordagem híbrida de Meta-heurísticas para solução do Problema da Clusterização Capacitada

Vinícius Carlos de Oliveira

JUIZ DE FORA
JANEIRO, 2026

Uma abordagem híbrida de Meta-heurísticas para solução do Problema da Clusterização Capacitada

Vinícius Carlos de Oliveira

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Luciana Brugiolo Gonçalves

Coorientador: Stênio Sã Rosário Furtado Soares

JUIZ DE FORA

JANEIRO, 2026

Uma abordagem híbrida de Meta-heurísticas para solução do Problema da Clusterização Capacitada

Vinícius Carlos de Oliveira

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Luciana Brugiolo Gonçalves
D.Sc. em Ciência da Computação

Stênio Sã Rosário Furtado Soares
D.Sc. em Ciência da Computação

Lorenza Leão Oliveira Moreno
D.Sc. em Informática

Luciana Conceição Dias Campos
D.Sc. em Engenharia Elétrica

JUIZ DE FORA

20 DE JANEIRO, 2026

A Deus, por tudo que fez
para que eu chegasse até aqui.

Resumo

O Problema da Clusterização Capacitada é amplamente estudado na literatura, tendo diversas aplicações relevantes, como a entrega de encomendas e roteamento de veículos. Diversas técnicas de otimização são usadas para tratar o problema, entre elas destaca-se o Simulated Annealing (SA), uma meta-heurística que auxilia na busca do espaço de solução, fazendo com que, por meio de perturbação e simulação de temperatura, aceite soluções piores que a corrente, auxiliando a sair de ótimos locais. Para aprimorar as soluções obtidas através do SA, a estratégia é combinada com o Randomized Variable Neighborhood Descent (RVND), que realiza buscas locais sistemáticas em diferentes vizinhanças, contribuindo para o refinamento das soluções obtidas. Além disso, incorpora-se o Reactive Greedy Randomized Adaptive Search Procedure (RGRASP) junto com VND, com o objetivo de gerar soluções iniciais mais promissoras e diversas. Essa integração entre diversas meta-heurísticas visa acelerar a convergência para gerar soluções mais eficientes e atrativas para problemas combinatórios complexos de otimização, como o PCC.

Palavras-chave: Inteligência Computacional, Otimização, Problema da Clusterização Capacitada, Buscal Local, Metaheurísticas, Simulated Annealing, Reactive Greedy Randomized Adaptive Search Procedure, Randomized Variable Neighborhood Descent, SA-RGRASP-RVND.

Abstract

The Capacitated Clustering Problem (CCP) has been widely studied in the literature and presents several relevant applications, such as parcel delivery and vehicle routing. Various optimization techniques have been employed to address this problem, among which Simulated Annealing (SA) stands out. SA is a metaheuristic that assists in exploring the solution space by allowing, through perturbations and temperature simulation, the acceptance of solutions worse than the current one. This mechanism helps generate diverse candidate solutions and escape local optima, enabling the search to move toward new regions of the solution space. To further improve the solutions obtained by SA, this strategy is combined with the Randomized Variable Neighborhood Descent (RVND), which performs systematic local searches across different neighborhoods, contributing to the refinement of the obtained solutions. Additionally, the Reactive Greedy Randomized Adaptive Search Procedure (RGRASP) is incorporated together with VND to generate more promising and diverse initial solutions, favoring local search in multiple regions of the search space rather than relying solely on greedy strategies. This integration of multiple metaheuristics aims to accelerate convergence and produce more efficient and competitive solutions for complex combinatorial optimization problems such as the CCP.

Keywords: Computational Intelligence, Optimization, Capacitated Clustering Problem, Local Search, Metaheuristics, Simulated Annealing, Reactive Greedy Randomized Adaptive Search Procedure, Randomized Variable Neighborhood Descent, SA-RGRASP-RVND.

Agradecimentos

Agradeço primeiramente a Deus por me impulsionar cada vez mais a novos e incríveis desafios e por ter me mostrado um caminho tão especial para que eu pudesse trilhar até aqui. Com Ele presente em todos os momentos da minha vida, tudo isso faz sentido e me motiva a avançar cada vez mais, com vontade de fazer acontecer e de entregar o meu melhor em cada lugar por onde eu passar.

Aos meus orientadores Luciana, Stênio e Lorenza, pelos quais tenho um carinho especial, pela amizade, compreensão e paciência ao longo de todos esses anos, especialmente pelo tempo de espera até que fosse possível a conclusão deste trabalho. Estendo também meus agradecimentos aos orientadores e coordenadores de projetos que passei, dentre eles Marcelo Caniato, Alessandreia, Heder, Alex, Fabrício, Cristina Dusi, por doarem o máximo de si aos projetos e liderarem os alunos com grande competência e dedicação.

Da mesma forma, nada disso seria possível sem minha companheira de todos os dias, Flávia, minha esposa, da qual com tanto carinho me ajuda a estar sempre firme e disposto para enfrentar todas as batalhas da vida, sempre com sorriso no rosto e ânimo renovado.

Aos meus pais, Vicente e Fátima, que com o *sim* deles à minha vida neste mundo, me proporcionaram estar aqui hoje, colhendo os frutos do plantio que realizaram ao longo de tantos anos. Tenho plena certeza que fizeram de tudo para me conduzir pelo melhor caminho possível. Minha gratidão eterna por serem quem são, pelos ensinamentos, conselhos e por todo carinho de sempre.

Ao meu irmão Vitor, meu fiel escudeiro, que desde bem novo me acompanhava em tudo o que fazia e seguia meus passos. Destaco o orgulho que sinto por ele e o quanto aprendo em nossas conversas e agradeço por estarmos sempre juntos. Hoje, é um grande profissional e, felizmente, escolheu trilhar alguns caminhos profissionais diferentes dos

meus, de modo que não precisamos ter dois Vinícius na família.

Aproveito para deixar um agradecimento extremamente especial a todos os meus familiares que, ao longo de todos esses anos de graduação, me ensinaram por meio de cada atitude e palavra, sempre me encorajando a seguir em frente na busca pelos meus sonhos. Em especial, agradeço aos meus primos Erivelton e Thiago, que desde sempre estiveram ao meu lado como verdadeiros irmãos.

Aos meus amigos Leonardo e Tales, registro meu agradecimento por toda a trajetória que construímos juntos ao longo dos anos, compartilhando diferentes fases da vida, desafios e conquistas. A convivência constante, marcada pelo apoio mútuo, pela presença sincera e pelo incentivo recíproco, foi fundamental em diversos momentos dessa caminhada. Estendo também meus agradecimentos a todos os demais amigos que, de forma direta ou indireta, participaram da minha formação pessoal e acadêmica. Mesmo que o contato atualmente seja menor, todos ocupam um lugar especial na minha história.

Aos meus sócios na Real Grana, Gabriel e Pedro, com quem aceitei o desafio de construir uma *startup* do zero, agradeço pela força, parceria e companhia ao longo dessa jornada, que muitas vezes é solitária. Hoje compreendo ainda mais a importância da paciência que precisamos cultivar juntos para construir uma empresa que, de fato, gere valor para a sociedade. Ressalto que essa trajetória só se tornou possível através de uma oportunidade ímpar de conhece-los através da LMF UFJF, e que a partir das experiências adquiridas nessa admirável universidade me permitem, atualmente, como CTO, aplicar de forma prática todos os aprendizados construídos ao longo dessa formação.

Preciso também registrar importância dos diversos projetos e pessoas que marcaram minha vida acadêmica desde o início da graduação, sem citar nomes, pois são muitos e sabem quem são. Agradeço a todos com quem tive contato ao longo dessa caminhada, em especial às turmas do curso e membros dos projetos em que trabalhei. Agradeço ainda desde meus primeiros passos como monitor de Algoritmos, experiência que despertou em mim o entusiasmo pelo ensino, ao GETComp e aos amigos GETianos, por um período marcante regado a café, ensino, pesquisa e extensão dentro da universidade. Agradeço também ao projeto Buddy e à DRI, pela oportunidade de contato com a cultura global; à Code Empresa Júnior, por me ensinar os caminhos do empreendedorismo e da liderança;

à Liga de Mercado Financeiro (LMF) — hoje Finance UFJF — por ter me acolhido em uma de minhas grandes paixões; e à Liga de Empreendedorismo, Inovação e Startups (LEIS), da qual fui idealizador e fundador e que hoje gera grandes frutos como projeto de extensão da Faculdade de Administração da UFJF.

Por fim, deixo meu agradecimento especial a todos os professores do DCC e do ICE, dos quais obtive ensinamentos valiosos e com quem levarei memórias marcantes de pessoas excepcionais. Estendo também minha gratidão, de forma geral, ao quadro de funcionários do curso e da UFJF, que ao longo desses anos contribuíram, de alguma forma, para o meu enriquecimento pessoal e profissional.

"Você precisa sonhar grande, tem que mirar na lua, porque se você errar pelo menos você acerta nas estrelas". Inclusive, "Don't Panic!"

Lair Ribeiro; Douglas Adams

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computação
UFJF	Universidade Federal de Juiz de Fora
PCC	Problema da Clusterização Capacitada
CCP	*Capacitated Clustering Problem*
SA	*Simulated Annealing*
TS	*Tabu Search*
GRASP	*Greedy Randomized Adaptive Search Procedure*
RGRASP	*Reactive Greedy Randomized Adaptive Search Procedure*
VNS	*Variable Neighborhood Search*
VND	*Variable Neighborhood Descent*
RVND	*Randomized Variable Neighborhood Descent*
IRACE	*Iterated Racing for Automatic Algorithm Configuration*
ILS	*Iterated Local Search*
IG	Iterated Greedy
IC	Inteligência Computacional

1 Introdução

A versão clássica do problema de clusterização é amplamente estudada na literatura de Otimização Combinatória, sendo fundamental em diversas áreas como análise de dados, logística, bioinformática e redes complexas. Em sua forma clássica, o problema consiste em agrupar elementos similares em um mesmo grupo denominado *cluster*, e elementos não similares em *clusters* distintos (LUXBURG; WILLIAMSON; GUYON, 2012), oferecendo uma maneira de extrair padrões relevantes em grandes conjuntos de dados.

Em muitos cenários, a versão clássica da clusterização não é suficiente para modelar adequadamente as restrições de problemas reais, exigindo a incorporação de restrições adicionais, como de capacidade. Entre essas variantes, destaca-se o Problema da Clusterização Capacitada (PCC), introduzido por (MULVEY; BECK, 1984).

O Problema de Clusterização Capacitado (PCC) é um problema de Otimização Combinatória (NEGREIROS et al., 2022) que consiste em particionar um conjunto de elementos em *clusters* disjuntos. Cada elemento possui um peso associado e, para cada par de elementos, define-se um valor de benefício que é contabilizado caso ambos sejam alocados no mesmo *cluster*. O objetivo do PCC é formar os *clusters* de modo que a soma dos pesos dos elementos em cada um deles respeite limites mínimo e máximo de capacidade, enquanto se maximiza a função objetivo, definida como a soma dos benefícios associados aos pares de elementos agrupados em um mesmo *cluster* (LAI et al., 2021).

Esse problema surge naturalmente em contextos nos quais há restrições operacionais ou físicas sobre os *clusters* formados, ao mesmo tempo em que se deseja maximizar algum critério de afinidade ou ganho interno aos *cluster*. De forma geral, o PCC possui diversas aplicações relevantes que surgem em vários contextos diferentes (MURITIBA et al., 2022), sendo adequado a problemas em que se deseja particionar um conjunto de elementos em *clusters* de modo que os elementos de cada *cluster* apresentem alta afinidade interna, respeitando simultaneamente limites mínimos e máximos de capacidade. Esses limites representam restrições físicas, operacionais ou econômicas, enquanto os benefícios associados aos pares de elementos expressam similaridade, proximidade ou redução de

custos, conforme discutido em (LAI et al., 2021).

Uma aplicação clássica do PCC ocorre no planejamento de instalações e na definição de zonas operacionais em sistemas logísticos (DENG; BARD, 2011; MARTINEZ-GAVARA et al., 2017). Nesses cenários, os nós representam regiões ou unidades de atendimento, os pesos dos nós correspondem à demanda, os limites de capacidade asseguram a viabilidade operacional de cada *cluster* e os benefícios refletem ganhos logísticos ao agrupar determinadas regiões. Essa modelagem é amplamente utilizada em redes de processamento e distribuição, como serviços postais e empresas de transporte, onde o PCC auxilia o desenho de áreas de atendimento e a integração de redes, possibilitando ganhos operacionais e redução de recursos, como frota e tempo de deslocamento (BARD; JARRAH, 2013; MURITIBA et al., 2022).

O PCC também é aplicado em redes de telecomunicações, como no problema de minimização de *handover* em redes celulares (MORÁN-MIRABAL et al., 2013; MARTÍNEZ-GAVARA et al., 2015). O *handover* ocorre quando um dispositivo móvel, como um *smartphone*, precisa trocar a estação rádio-base à qual está conectado durante uma comunicação ativa, devido à mobilidade do usuário ou à variação da qualidade do sinal. Trocas frequentes podem aumentar a sinalização da rede e degradar a qualidade do serviço. Nessa modelagem, os nós representam as estações rádio-base, responsáveis por fornecer acesso à rede, enquanto os *clusters* correspondem aos controladores de rede (RNCs) que gerenciam conjuntos dessas estações. Os benefícios refletem a redução de *handovers* ao agrupar estações sob o mesmo controlador; os pesos representam a carga de tráfego das estações, e os limites de capacidade correspondem às restrições técnicas dos RNCs.

Segundo (ZHOU et al., 2019), o PCC pertence a classe de problemas NP-Difíceis, caracterizado por sua natureza combinatória e pela inexistência de algoritmos de tempo polinomial conhecidos para a obtenção da solução ótima (RUAN; DA, 2010). Dessa forma, torna-se necessário buscar uma heurística eficiente e o uso de estratégias de Otimização e Inteligência Computacional, com o objetivo de obter soluções aproximadas de boa qualidade em tempo computacional viável (NEGREIROS et al., 2022; LAI et al., 2021).

Por fim, o PCC tem sido amplamente investigado na literatura por meio de diferentes abordagens baseadas em algoritmos de otimização, e esses modelos têm como

objetivo resolver problemas complexos, para os quais métodos exatos se tornam computacionalmente inviáveis. Nesse contexto, o presente trabalho visa contribuir para o avanço do estado da arte por meio da obtenção de soluções aproximadas para o PCC. Para isso, propõe-se o desenvolvimento de um algoritmo híbrido que combina distintas meta-heurísticas, buscando soluções sub-ótimas de forma eficiente e robusta, conforme discutido em (HUSSAIN et al., 2019).

Nos próximos capítulos, o trabalho está organizado da seguinte forma. No Capítulo 2, o problema abordado é descrito. O Capítulo 3 apresenta a revisão da literatura, enfatizando os principais conceitos e as contribuições dos trabalhos relacionados ao problema. O Capítulo 4 é dedicado à apresentação da fundamentação teórica e dos conceitos fundamentais que norteiam este trabalho. No Capítulo 5, são detalhadas as abordagens propostas neste trabalho. No Capítulo 6 são apresentados os experimentos computacionais realizados e a análise dos resultados obtidos. Por fim, o Capítulo 7 apresenta as conclusões, destacando os principais aprendizados alcançados e indica possíveis trabalhos futuros.

2 O Problema de Clusterização Capacitada

O Problema de Clusterização Capacitada (PCC) pode ser formalmente definido a partir de um grafo não direcionado $G = (V, E)$, em que $V = \{1, \ldots, n\}$ representa o conjunto de elementos (ou vértices) a serem agrupados e E o conjunto de arestas que conectam pares de vértices. Cada vértice $i \in V$ possui um peso w_i, enquanto a cada aresta $(i, j) \in E$ está associado um benefício b_{ij}, que representa o ganho obtido caso os vértices i e j sejam alocados no mesmo *cluster*.

Como podemos ver na figura 2.1, que ilustra visualmente o objetivo fundamental do problema abordado. À esquerda apresenta o conjunto de dados de entrada, representando os vértices dispersos no espaço. Na direita exemplifica a partição ideal almejada no Problema de Clusterização Capacitado (PCC): a divisão desses vértices em subconjuntos disjuntos, que são diferenciados pelas cores, de modo que a homogeneidade interna de cada grupo seja maximizada — ou seja, os membros de um mesmo *cluster* estejam geograficamente próximos. Assim podemos ver de maneira visual que o benefício de um vértice estar associado ao outro dentro de um *cluster* é o benefício que eles geram de estarem próximos. Em outros problemas essa proximidade pode ser relativa a outros fatores, como quantidade de entregas por exemplo.

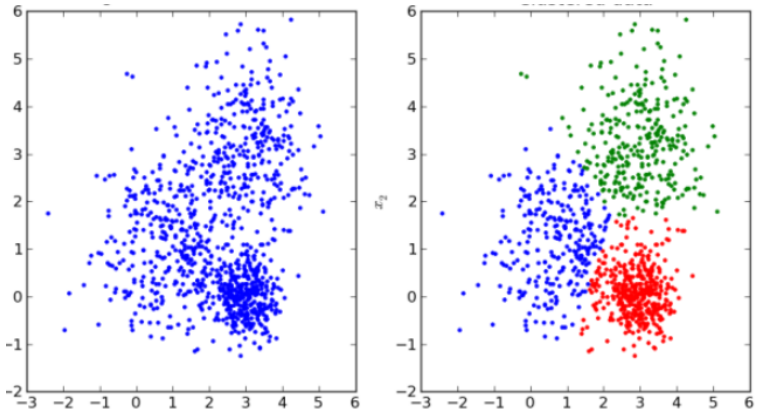

Figura 2.1: Exemplo de vértices de um grafo clusterizado em 3 *clusters* distintos.

O objetivo do PCC é particionar o conjunto de vértices V em p *clusters* disjuntos. O peso de cada *cluster* é dado pela soma dos pesos dos vértices que o compõem, devendo

respeitar limites mínimo e máximo de capacidade previamente definidos. A função objetivo consiste em maximizar a soma dos benefícios das arestas cujos vértices pertencem ao mesmo *cluster*, modelando situações práticas em que é necessário formar *clusters* equilibrados sob restrições de capacidade, ao mesmo tempo em que se busca maximizar a afinidade ou o ganho interno entre seus elementos do mesmo *cluster*.

Pode-se descrever o PCC matematicamente como um problema de Programação Não-Linear Inteira, considerando os seguintes parâmetros e variáveis:

- V é o conjunto de vértices;
- E é o conjunto de arestas;
- C é o conjunto de clusters;
- n é o número de vértices;
- p é o número fixo de clusters;
- w_i é o peso associado ao vértice $i \in V$;
- b_{ij} é o benefício da aresta $e = (i, j)$;
- c_k é um *cluster* do conjunto C;
- L é o limite inferior para o somatório de pesos dos vértices de um *cluster*;
- U é o limite superior para o somatório de pesos dos vértices de um *cluster*;
- x_{ic} é uma variável binária que assume valor 1 caso o vértice $i \in V$ seja alocado ao *cluster* $c \in C$, e 0 caso contrário.

$$\text{Maximizar } z = \sum_{c=1}^{p} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} b_{ij} x_{ic} x_{jc} \tag{1a}$$

$$\text{sujeito a} \quad \sum_{c=1}^{p} x_{ic} = 1 \qquad \forall i \in V \tag{1b}$$

$$L \leq \sum_{i=1}^{n} w_i x_{ic} \leq U \qquad \forall c \in \{1, 2, ..., p\} \tag{1c}$$

$$x_{ic} \in \{0, 1\} \qquad \forall i \in V, \ \ \forall c \in \{1, 2, ..., p\} \tag{1d}$$

As restrições (1b) garantem que cada vértice i seja atribuído a exatamente um *cluster*. As Restrições de peso (1c) asseguram que os requisitos de capacidade mínima (L) e a capacidade máxima (U) para cada *cluster* sejam satisfeitas. As Restrições (1d) definem o domínio das variáveis. Por fim, a função objetivo (1a) consiste em maximizar o somatório do benefício interno de cada *cluster*, dado pelo somatório dos benefícios associados às arestas incidentes à vértices do mesmo *cluster*.

A Figura 2.2 ilustra um exemplo de instância para o PCC, em que dado um grafo $G = (V, E)$ ponderado nos vértices e nas arestas com $|V| = 14$ e $w_i = 1$ para todos os vértices $v_i \in V$, com quantidade de *clusters* $p = 4$ e limites inferior e superior $[L, U] = [3, 4]$ para todos os *clusters*. Assim, o objetivo é particionar o conjunto V de vértices em 4 *clusters* c_k onde $k \in \{1, \ldots, 4\}$, de forma que a soma dos benefícios das arestas internas aos *clusters* (destacadas em vermelho) seja maximizada.

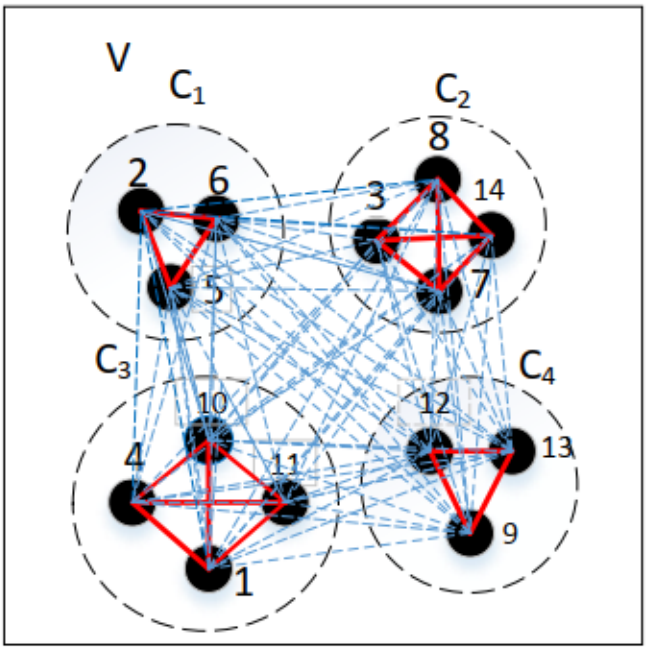

Figura 2.2: Exemplo de instância do PCC com 14 nós e 4 *clusters*.

Neste capítulo foi apresentado o Problema da Clusterização Capacitada (PCC), um desafio clássico da Otimização Combinatória. Destaca-se que o PCC é um problema NP-Difícil, o que implica que não há algoritmo conhecido capaz de encontrar uma solução ótima em tempo polinomial. Portanto, é necessário recorrer a estratégias de Otimização e Inteligência Computacional para obter boas soluções para o problema.

3 Trabalhos Relacionados

Neste capítulo é apresentado o estado da arte relacionado ao Problema da Clusterização Capacitada (PCC), são discutidos os principais conceitos, modelos e abordagens propostos na literatura e destacam-se as contribuições das pesquisas que influenciam diretamente o desenvolvimento deste estudo.

O PCC foi introduzido formalmente na literatura por (MULVEY; BECK, 1984), estabelecendo as bases fundamentais para a modelagem deste problema combinatório. Após ser definido, ele tem sido objeto de estudo contínuo, evoluindo com o surgimento de novas técnicas de otimização ao longo dos anos. Desta forma, os parágrafos seguintes dedicam-se a detalhar os avanços alcançados na literatura recente, apresentando e discutindo cronologicamente os artigos mais relevantes dos últimos anos e suas respectivas abordagens para a solução do problema.

No trabalho (MARTÍNEZ-GAVARA et al., 2015) foram realizados testes com várias estratégias de busca, incluindo as que se movem para fora da região viável e uso de estruturas de vizinhança dentro da estrutura GRASP. Foi explorado um TS com uma vizinhança de troca 2-1 que se mostrou eficaz, realizando novos testes em instâncias que até então não existiam na literatura.

No trabalho de (LAI; HAO, 2016) é apresentado o método IVNS que combina um método Variable Neighborhood Descent (VND) estendido (EVND) para intensificação das soluções e um procedimento de perturbação que remodela a solução corrente de forma aleatória para explorar efetivamente o espaço de busca e ter diversificação das soluções. O algoritmo proposto melhorou os resultados de 28 das 83 instâncias do tipo de minimização *handover* e encontrou o melhor resultado conhecido para outras 55 instâncias de maximização , melhorando os resultados alcançados anteriormente.

No trabalho de (MARTINEZ-GAVARA et al., 2017) foi investigado a metaheurística (Greedy Randomized Adaptive Search Procedure) GRASP e a (Iterated Greedy) IG, propondo uma nova vizinhança de troca 2-1 no GRASP, um algoritmo baseado no IG e um algoritmo híbrido IG-GRASP. Utiliza-se uma construção de soluções indepen-

dentes do GRASP sem memória das soluções anteriores e usa o IG para obter soluções baseadas em outras soluções na memória a fim de construir soluções a partir da reconstrução parcial das anteriores do IG. Em seus resultados foi visto que a hibridização do IG-GRASP foi capaz de encontrar bons resultados comparando aos métodos anteriores a 2017. Portanto, este trabalho demonstrou que a construção baseada em memória é um mecanismo eficaz dentro das técnicas de busca heurística e que algoritmos híbridos combinando métodos gulosos com randômicos têm um grande potencial.

Em (BRIMBERG et al., 2019), são propostas duas abordagens baseadas na metaheurística (Variable Neighborhood Search) VNS. O primeiro segue uma abordagem VNS padrão chamado de VNS Geral (GVNS - *General* VNS) e a segunda um VNS enviesado (SVNS - *Skewed* VNS) que permite movimentos para soluções piores. Tanto o SVNS quando o GVNS superaram o estado da arte até aquele momento, sendo o SVNS melhor no geral. Isso sugere que o uso de critérios de aceitação antes de permitir outros movimentos para novas soluções do SVNS é preferível à abordagem aleatória que é usada no GVNS para mover para novas regiões do espaço de solução.

No trabalho (ZHOU et al., 2019) são apresentados dois algoritmos efetivos para solucionar o PCC, um (Tabu Search) TS chamado de FITS que alterna entre exploração em regiões de espaços de busca viáveis e inviáveis e um Algoritmo Memético (MA) que combina o FITS com um *crossover* baseado em *cluster* dedicado. Os experimentos computacionais foram executados para 183 instâncias e indicaram que FITS e MA geram boas soluções.

Em (MAXIMO; NASCIMENTO, 2021) os autores apresentam um algoritmo híbrido baseado em (Iterated Local Search) ILS, que combina a busca local com perturbações e estratégias de intensificação e diversificação para melhorar os resultados obtidos. Os experimentos mostram que o algoritmo proposto é eficaz na obtenção de boas soluções.

Em (LAI et al., 2021) foi proposto um algoritmo de pesquisa de vizinhança variável usando decomposição de vizinhança (NDVNS). Essa estratégia acelera o processo de busca e usa uma forma de perturbação probabilística para controlar a troca entre intensificação de busca e diversificação, assim isola soluções candidatas promissoras a serem consideradas em cada iteração de busca, acelera a busca na vizinhança e permite buscas

mais focadas. Os resultados foram executados para 110 instâncias comumente usadas na literatura, encontrando a melhor média em todas as instâncias e melhor resultado em 70% em relação a trabalhos anteriores.

Outro trabalho a tratar o PCC foi a abordagem de (LIU; GUO; ZENG, 2022b) que propõe um Algoritmo Evolucionário de Membrana. Esse algoritmo foi proposto em (OROZCO-ROSAS; MONTIEL; SEPúLVEDA, 2019) e usa operações evolutivas de membrana, como divisão, fusão, seleção, citólise e outros. Os resultados deste trabalho mostram que o número de instâncias em que o MEACCP pode encontrar a solução ótima é até 9 vezes maior do que nos algoritmos da literatura e a estabilidade é melhor do que todos os algoritmos comparados no passado.

Em (LIU; GUO; ZENG, 2022a) foi proposto um algoritmo híbrido chamado HA-PCC, que possui um método de construção de solução viável mais guloso utilizando restrições mais restritas que as originais da instância, com destruição e reconstrução da solução para aumentar a diversidade da população e melhorar a velocidade de convergência. Das 90 instâncias utilizadas nos testes, em 58 dessas o HA-PCC possui a melhor solução média da literatura, sendo a proposta que possui a melhor estabilidade.

Em (FALKNER; SCHMIDT-THIEME, 2023) foi proposto um método chamado *Neural Capacited Clustering* que é a primeira abordagem baseada em *Deep Learning*. Os autores utilizam K-Means para refinar a solução e uma rede neural para aprender e prever as probabilidades de escolher um vértice para um *cluster* a partir de um conjunto de dados de soluções subótimas passadas de outras instâncias do problema. Apesar de superar várias soluções heurísticas da literatura, este trabalho utiliza instâncias de outro problema, dificultando a comparação com os trabalhos que utilizam as instâncias próprias PCC.

O trabalho mais recente da literatura (ALTUWAIM, 2023) propõe uma metaheurística baseada no algoritmo Artificial Bee Colony (ABC). A abordagem utiliza uma estratégia construtiva inicial seguida da geração de soluções vizinhas e da aplicação de busca local com probabilidade controlada, visando melhorar a qualidade das soluções ao longo das iterações. Os resultados obtidos alcançam os melhores resultados da literatura, embora sua eficiência dependa de uma escolha adequada de parâmetros para cada tipo

de instância e tenha sido utilizado apenas um conjunto de 10 instâncias com 82 vértices, dificultando comparação com as demais instâncias usadas na literatura.

Por fim, foi visto que diversos trabalhos da literatura abordam o PCC por meio de diferentes estratégias. Em conjunto, esses trabalhos reforçam a relevância do uso de heurísticas e meta-heurísticas como ferramentas eficazes para tratar o PCC, especialmente em cenários onde métodos exatos se tornam computacionalmente inviáveis.

A partir da revisão da literatura, observa-se que as principais abordagens propostas para o PCC se concentram no uso de estratégias baseadas em busca local, com destaque para métodos como VNS, GRASP, Busca Tabu e estratégias híbridas que combinam mecanismos de intensificação e diversificação. Trabalhos mais recentes também exploram abordagens híbridas mais sofisticadas, incorporando aprendizado de máquina ou estruturas bioinspiradas, evidenciando a maturidade do campo e o esforço contínuo em obter soluções de melhor qualidade e maior estabilidade computacional.

Apesar desses avanços, nota-se que a maior parte dos trabalhos concentra-se em estratégias nas quais a diversificação ocorre predominantemente por meio de perturbações ou de mecanismos clássicos de mudança de vizinhança. Em contrapartida, há uma lacuna na exploração sistemática de meta-heurísticas baseadas em critérios probabilísticos de aceitação de soluções de não melhora, como o *Simulated Annealing* (SA).

De forma geral, os resultados reportados indicam que abordagens híbridas tendem a apresentar desempenho superior quando comparadas a heurísticas isoladas. Dessa forma, identifica-se uma oportunidade de pesquisa na proposição de uma abordagem híbrida que explore explicitamente a capacidade do SA em escapar de ótimos locais por meio da aceitação controlada de soluções piores, aliada à geração de soluções iniciais de alta qualidade utilizando GRASP, bem como ao refinamento sistemático via VND por meio de estratégias de busca local baseadas em múltiplas vizinhanças. Essa lacuna motiva o desenvolvimento do método proposto neste trabalho, que combina SA, GRASP reativo e VND Randômico para alcançar soluções competitivas para o PCC.

4 Fundamentação Teórica

Este capítulo é destinado a introduzir a fundamentação teórica, necessária para o entendimento do trabalho, apresentando definições, modelos, algoritmos e indicando os conceitos fundamentais para o desenvolvimento deste trabalho.

4.1 Inteligência Computacional

A Inteligência Computacional (IC) compreende um conjunto de técnicas e algoritmos inspirados em processos naturais e cognitivos, como algoritmos genéticos e métodos evolutivos, com o objetivo de resolver problemas complexos. Nesse contexto, a IC possibilita a obtenção de soluções aproximadas de boa qualidade em tempo reduzido, o que justifica o crescente interesse da comunidade científica no desenvolvimento de métodos eficientes para tratar problemas dessa natureza (SOUZA, 2008). Para atender a esse desafio, foram desenvolvidas diversas heurísticas e técnicas voltadas à exploração eficiente do espaço de busca. Entre elas, destacam-se as meta-heurísticas, que consistem em estruturas gerais de busca capazes de orientar a construção e o aprimoramento de soluções por meio de diferentes estratégias. Algumas dessas heurísticas e meta-heurísticas são utilizadas neste trabalho e serão descritas nas seções subsequentes.

4.2 Heurísticas Construtivas

As heurísticas construtivas têm como objetivo gerar soluções iniciais para problemas de otimização combinatória. Essas heurísticas podem empregar diferentes estratégias, como abordagens gulosas, randomizadas ou híbridas, além de mecanismos baseados em seleção probabilística e critérios de inserção. A combinação dessas estratégias é frequentemente utilizada com o intuito de produzir soluções iniciais diversificadas. O principal objetivo dos métodos construtivos é obter soluções viáveis de forma rápida, com baixo custo computacional, servindo como ponto de partida para etapas posteriores do processo de

otimização (BURKE et al., 2013).

4.3 Métodos de Busca Local

Um dos tipos de heurísticas mais utilizados para a obtenção de soluções de boa qualidade em problemas de otimização combinatória são os métodos de Busca Local. Essas abordagens são particularmente relevantes em problemas da classe NP-Difícil, nos quais se busca otimizar uma função objetivo em um espaço de soluções que respeite um conjunto de restrições. No entanto, devido à complexidade do espaço de busca, os algoritmos podem convergir para soluções correspondentes a ótimos locais, sem conseguir identificar soluções de melhor qualidade situadas em outras regiões do espaço de soluções.

O conceito fundamental que rege o funcionamento desses algoritmos é a definição de uma estrutura de vizinhança. Formalmente, para uma solução candidata s pertencente ao espaço de busca S, define-se a vizinhança $N(s)$ como o subconjunto de soluções $s' \in S$ que podem ser alcançadas diretamente a partir de s através de uma operação de movimento ou transformação simples entre vértices e estruturas criadas para problema. Esses movimentos consistem em pequenas modificações na estrutura da solução corrente, como a troca de elementos entre *clusters* ou a realocação de vértices. Dessa forma, a busca local navega pelo espaço de soluções movendo-se iterativamente de uma solução atual para uma de suas vizinhas, avaliando a qualidade destas para determinar se houve melhoria e se será aceita como nova solução.

A Figura 4.1 ilustra a presença de ótimos locais em um problema de maximização. Observa-se que algoritmos de busca local tendem a ficar presos nesses pontos de máximo local e planícies sem reconhecer que pode haver soluções melhores. Assim as buscas locais alteram a solução com objetivo de alcançar novos resultados e quando chegam em um ponto como esses ótimos locais não conseguem sair deles, pois melhorias imediatas podem não estar disponíveis na vizinhança.

Nesse contexto, os métodos de Busca Local atuam promovendo modificações controladas na solução corrente, com o objetivo de explorar novas regiões do espaço de busca ainda não visitadas. Essas modificações podem permitir a identificação de soluções associadas a ótimos locais de melhor qualidade ou, em casos ideais, a aproximação do ótimo

global, que corresponde à solução ótima do problema.

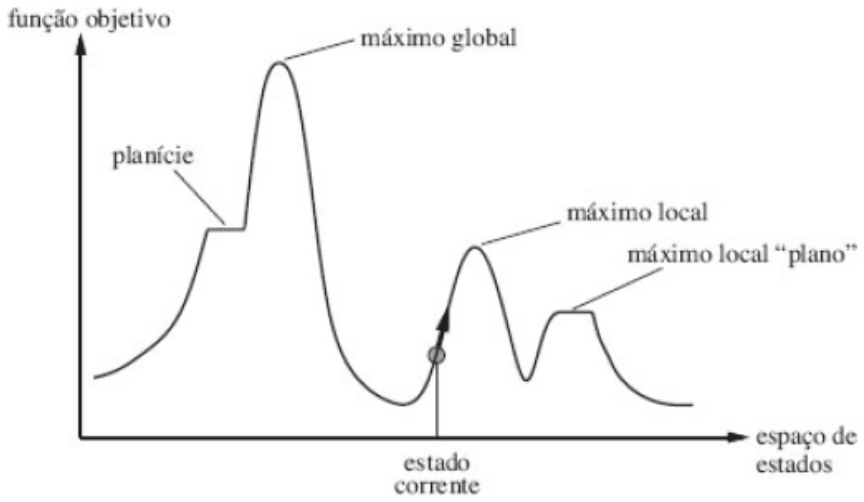

Figura 4.1: Ótimos locais e global em um espaço de solução explorado por uma heurística de busca local em um problema de maximização.

De forma geral, a Busca Local realiza um refinamento da solução inicial por meio da exploração de soluções vizinhas (VOSS et al., 2012). Essas soluções podem ser geradas a partir de operações simples, como, por exemplo, no caso do PCC, operações de trocas de vértices entre *clusters* distintos, bem como inserções ou remoções de vértices em um *cluster*.

O processo de Busca Local é conduzido de forma iterativa, avaliando a cada etapa as soluções vizinhas geradas a partir da solução corrente. Esse procedimento é repetido até que nenhum vizinho proporcione melhoria, interrompendo a execução no primeiro ótimo local encontrado.

4.4 Meta-heurísticas

Na literatura de Otimização e Inteligência Computacional, as meta-heurísticas são amplamente empregadas na resolução de problemas da classe NP-Difícil, para os quais não se conhece algoritmo de tempo polinomial capaz de garantir a solução ótima. Essas técnicas atuam orientando o processo de busca no espaço de soluções, equilibrando mecanismos de intensificação e diversificação, reduzindo a probabilidade de estagnação em ótimos locais (HUSSAIN et al., 2019).

Diferentemente de heurísticas puramente construtivas ou gulosas, as meta-heurísticas, via diferentes estratégias, buscam escapar de ótimos locais e explorar soluções potencialmente superiores em outras regiões do espaço de busca. Exemplos clássicos de meta-heurísticas incluem *Simulated Annealing* (SA), *Greedy Randomized Adaptive Search Procedures* (GRASP) e *Variable Neighborhood Search* (VNS).

Neste trabalho, algumas dessas meta-heurísticas são exploradas com o objetivo de resolver o Problema da Clusterização Capacitada (PCC). As técnicas adotadas e sua integração no método proposto são detalhadas nas seções subsequentes.

4.5 *Simulated Annealing*

Em (OSMAN; CHRISTOFIDES, 1994) foi proposta uma abordagem baseada em *Simulated Annealing* (SA), capaz de guiar um método de busca local iterativo, permitindo que o processo de busca continue mesmo após a detecção de um ótimo local. Isso é realizado por meio de critérios determinísticos ou probabilísticos que controlam a aceitação ou rejeição de soluções recém-geradas, inclusive soluções piores do que a corrente.

O SA é uma meta-heurística estocástica inspirada no processo físico de recozimento de metais, no qual um material é aquecido a altas temperaturas e posteriormente resfriado de maneira controlada, permitindo que a estrutura cristalina alcance estados de menor energia. De maneira análoga, o SA busca soluções de alta qualidade explorando o espaço de busca com maior liberdade no início da execução, aceitando soluções piores com maior probabilidade e reduzindo gradualmente essa probabilidade de aceitação à medida que a temperatura diminui. Dessa forma, o desempenho do algoritmo depende fortemente da escolha adequada de seus parâmetros, especialmente da estratégia de resfriamento da temperatura, tornando essencial um estudo cuidadoso de parametrização. Com a redução gradual da temperatura, o algoritmo passa a se comportar de maneira semelhante a uma busca local clássica, concentrando-se na intensificação da busca em regiões promissoras.

A Figura 4.2 exemplifica o processo realizado pelo SA, que busca escapar da armadilha de cair em ótimos locais por existir uma barreira de piores soluções entre a atual e uma possível melhor solução.

Formalmente, o *Simulated Annealing* admite a aceitação de uma solução vizinha

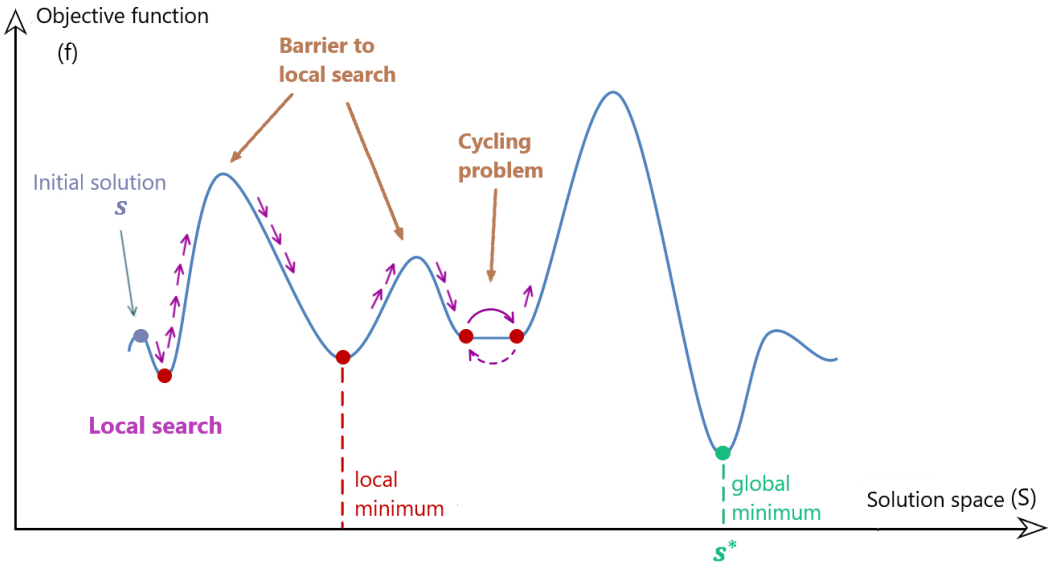

Figura 4.2: Barreiras para a busca local encontrar novos ótimos locais.

s' com valor da função objetivo pior que o da solução corrente s segundo um critério probabilístico, usualmente definido por

$$P(\Delta f) = \exp\left(-\frac{\Delta f}{T}\right),$$

em que $\Delta f = f(s') - f(s)$ representa a variação da função objetivo e T denota a temperatura do sistema. Esse mecanismo permite que o algoritmo atravesse barreiras do espaço de busca associadas a ótimos locais, possibilitando a exploração de regiões que não seriam acessíveis por métodos de busca local estritamente gulosos. À medida que a temperatura é gradualmente reduzida conforme um esquema de resfriamento, a probabilidade de aceitar soluções piores diminui, conduzindo o algoritmo a um comportamento progressivamente mais intensificativo, favorecendo a convergência para soluções de melhor qualidade.

4.6 Variable Neighborhood Descent

O Variable Neighborhood Descent (VND) é uma meta-heurística determinística amplamente empregada na resolução de problemas de otimização combinatória. Trata-se de um método iterativo que explora sistematicamente diferentes estruturas de vizinhança com o

objetivo de aprimorar uma solução inicial. A ideia central do VND consiste em alternar entre distintas vizinhanças, permitindo uma exploração mais ampla do espaço de soluções e reduzindo a probabilidade de estagnação em ótimos locais.

Este método, visto em (BRIMBERG et al., 2016) inicia a partir de uma solução viável e a cada iteração, aplica um procedimento de busca local sobre uma solução específica. Caso seja encontrada uma solução de melhor qualidade, essa solução passa a ser a nova solução corrente e o algoritmo retorna à primeira vizinhança. Caso contrário, o método avança para a próxima vizinhança definida. O processo é encerrado quando nenhuma melhoria é obtida após a exploração de todas as vizinhanças, caracterizando um ótimo local em relação ao conjunto de vizinhanças consideradas.

O método de VND pode ser utilizado com uma modificação em sua estrutura clássica. Em vez de explorar as vizinhanças em uma ordem fixa, adota-se uma estratégia de randomização da ordem de exploração das vizinhanças. Essa abordagem caracteriza o método como um *Randomized Variable Neighborhood Descent*(RVND), cujo objetivo é aumentar a capacidade de diversificação da busca e reduzir a dependência da ordem pré-definida das estruturas de vizinhança.

O RVND explora múltiplas estruturas de vizinhança; assim, o RVND promove um refinamento mais robusto das soluções candidatas, especialmente quando integrado a meta-heurísticas baseadas em perturbação, como o *Simulated Annealing*.

4.7 *Greedy Randomized Adaptive Search Procedure*

O *Greedy Randomized Adaptive Search Procedure* (GRASP) (RESENDE; RIBEIRO, 2016) é uma meta-heurística iterativa amplamente utilizada na resolução de problemas de otimização combinatória. O método combina uma fase de construção gulosa e randomizada com uma fase de busca local, visando explorar diferentes regiões do espaço de soluções e reduzir a dependência de uma única solução inicial.

Na fase de construção, uma solução viável é gerada de forma incremental por meio de um critério guloso. Após a fase construtiva, aplica-se um procedimento de busca local com o objetivo de aprimorar a solução obtida, explorando sua vizinhança até que um ótimo local seja alcançado. Neste trabalho, a etapa de refinamento do GRASP é realizada

por meio do *Variable Neighborhood Descent* (VND), que permite a exploração sistemática de diferentes estruturas de vizinhança. Esse ciclo de construção e refinamento é repetido por um número predefinido de iterações, sendo a melhor solução encontrada ao longo de todas as execuções armazenada como resultado final do algoritmo.

Pode ser utilizado uma variante reativa dessa meta-heurística, intitulada *Reactive Greedy Randomized Adaptive Search Procedure* (RGRASP). Diferentemente do GRASP clássico, o RGRASP utiliza um algoritmo guloso randomizado reativo na fase construtiva em que ajusta de forma reativa a escolha do parâmetro α com base no desempenho histórico das soluções geradas, favorecendo valores que produzem soluções de melhor qualidade.

5 Abordagens Propostas

Neste capítulo são detalhadas as abordagens propostas para a solução do PCC, descrevendo como os algoritmos discutidos nos capítulos anteriores foram adaptados, combinados e implementados, bem como as principais decisões de projeto adotadas ao longo do desenvolvimento deste trabalho. As abordagens propostas exploram diferentes estratégias heurísticas e meta-heurísticas, com o objetivo de equilibrar qualidade de solução e custo computacional, considerando explicitamente as restrições de capacidade inerentes ao problema.

Para elucidar melhor alguns pontos importantes das abordagens com foco no PCC, será tratado aqui algumas características que influenciam a criação dos métodos. Podemos inicialmente trazer a importância do peso dos vértices e do benefício da associação entre dois vértices. Iniciando pelo peso, este é associado a cada vértice a partir da instância, ou seja, essa informação é constante para cada vértice e foi determinado pelos autores na criação da instância, sendo que neste capítulo será visto que a soma dos pesos de vértices no mesmo *cluster* é o valor de peso daquele *cluster*, que seria algo como a capacidade alocada. No caso do benefício, ele também é determinado a partir da instância sendo um valor constante para cada aresta do grafo completo. Ao criar uma solução para o PCC, gera-se um sub-grafo completo em que todas arestas entre vértices de um mesmo *cluster* possuem um benefício associado, com isso a soma destas arestas é o benefício associado a ao *cluster* e a soma do benefício de cada *cluster* é o benefício da solução.

Este trabalho utiliza uma abordagem híbrida de algoritmos heurísticos e meta-heurísticos que visam gerar soluções distintas por meio de diferentes estratégias combinadas. A abordagem proposta consiste em um algoritmo *Simulated Annealing* (SA) que recebe como solução inicial o resultado de um algoritmo GRASP reativo. O objetivo de se utilizar o algoritmo *Reactive Greedy Randomized Adaptive Search Procedure* (RGRASP) na fase inicial do SA é gerar uma solução inicial de boa qualidade para facilitar a convergência. Na fase de busca local, o SA utiliza o RVND com três diferentes buscas locais,

que fazem trocas de vértices entre *clusters*. Além disso, a combinação com modificação da solução atual por meio de uma perturbação de uma porcentagem de vértices e *clusters* gera vantagem ao permitir escapar de bacias de atração e evitar convergência precoce do SA.

Em síntese, este trabalho usa dois algoritmos como base: o RGRASP, que provou ser mais efetivo na geração de boas soluções iniciais, e o SA para explorar o espaço de busca a partir da solução criado pelo RGRASP. A combinação proposta entre SA e RGRASP incorpora, ao longo de suas execuções, o método *Variable Neighborhood Descent* (RVND) como mecanismo de refinamento, visando aprimorar as diferentes soluções geradas durante o processo de busca.

O *Simulated Annealing* (SA) é adotado como a meta-heurística principal deste trabalho e caracteriza-se como um método de busca local estocástica, no qual soluções vizinhas são exploradas a partir de perturbações controladas, sendo aceitas conforme um critério probabilístico dependente da temperatura. Esse mecanismo permite ao algoritmo escapar de ótimos locais ao admitir, de forma controlada, soluções de pior qualidade. No entanto, observou-se que as soluções obtidas imediatamente após tais perturbações nem sempre apresentam qualidade suficiente para contribuir efetivamente com a intensificação da busca. Nesse contexto, a aplicação do *Random Variable Neighborhood Descent* (RVND) após as perturbações possibilita o refinamento sistemático das soluções geradas, conduzindo o algoritmo a regiões mais promissoras do espaço de busca e favorecendo a identificação de ótimos locais distintos daqueles previamente encontrados.

Além disso, constatou-se que a qualidade da solução inicial exerce influência significativa sobre o desempenho final do SA. Diante disso, o RGRASP foi empregado como método para a geração da solução inicial. O RGRASP opera de forma iterativa, combinando uma fase construtiva gulosa e randomizada com a aplicação do RVND para o refinamento de cada solução gerada. Esse processo é executado por um número pré-determinado de iterações, apresentando baixo custo computacional. Apesar de sua rápida execução, o uso do RGRASP para a construção da solução inicial resulta em ganhos expressivos de qualidade, impactando positivamente o desempenho global da abordagem híbrida proposta.

5.1 Solução inicial do RGRASP com Algoritmo Construtivo

O Algoritmo 1 descreve a heurística construtiva gulosa e randomizada proposta neste trabalho para a geração de uma solução inicial executada dentro do RGRASP. A heurística tem como objetivo construir uma solução viável respeitando as restrições de capacidade e custo, ao mesmo tempo em que busca uma boa qualidade inicial para posterior refinamento pela meta-heurística base SA.

Algoritmo 1: Heurística Construtiva Gulosa e Randomizada

```
   Entrada: grafoPonderado G, α, indMaxInserir
   Saída: solucaoInicial
 1 início
 2 |  listaCandidatos ← geraCandidatos(G)
 3 |  listaCandidatos ← ordenarPorCusto(listaCandidatos)
 4 |  para cada Cluster j do vetor clusters faça
 5 |  |  vertMaiorPeso ← escolheCandMaiorPeso(listaCandidatos)
 6 |  |  adicionaVertCluster(vertMaiorPeso, cluster[j])
 7 |  fim para cada
 8 |  para cada Cluster j do vetor clusters faça
 9 |  |  enquanto listaCandidatos ≠ Vazio faça
10 |  |  |  listaCandidatos ← calculaBeneficioEmCadaCluster()
11 |  |  |  listaCandidatos ← ordenarPorBeneficio(listaCandidatos)
12 |  |  |  indMaxInserir ← listaCandidatos.size * α
13 |  |  |  indMelhorVertCluster ← randNumEntre(0, indMaxInserir)
14 |  |  |  melhorCand ← listaCandidatos[indMelhorVertCluster]
15 |  |  |  melhorVertice ← melhorCand.vertice
16 |  |  |  melhorCluster ← melhorCand.cluster
17 |  |  |  addBestVertCluster(bestVert, bestCluster)
18 |  |  |  removeListaCandidatos(indiceVertInserir)
19 |  |  fim enqto
20 |  fim para cada
21 |  retorna solucao;
22 fim
```

Inicialmente, nas linhas 2 e 3 é gerado um conjunto de vértices candidatos, os quais são ordenados de acordo com um critério de custo associado ao problema , que são os vértices de maior peso, ou seja, são ordenados do vértice de maior peso até o vértice de menor peso. Em seguida nas linhas 4 a 7, para cada *cluster* a ser formado, seleciona-se um vértice de maior peso dentre os candidatos disponíveis, garantindo que cada *cluster* seja inicializado com ao menos um vértice representativo. Essa etapa visa assegurar uma

distribuição inicial equilibrada e viável dos vértices entre os *clusters*.

Após a fase de inicialização, a partir da linha 8 até a linha 20 o algoritmo entra em um processo iterativo de inserção dos vértices remanescentes. Enquanto houver candidatos não alocados, inicia-se a iteração dos *clusters* na linha 8 e na linha 9 outra iteração que continua enquanto a lista de candidatos houver elementos. Dentro do processo iterativo é avaliado o benefício da inserção de cada vértice em cada *cluster* na linha 10, considerando as restrições do problema sendo que para cada vértice da lista de candidatos consideramos o benefício dele se associar a algum *cluster*, a partir da soma do benefício associado a ele e todos os vértices de cada *cluster*. Sendo assim, a cada iteração onde um vértice é escolhido e adicionado a um *cluster*, o benefício de todos os vértices da lista de candidatos tem que ser atualizada para considerar o novo benefício de ser inserido naquel *cluster* que foi atualizado com a entrada de mais um vértice.

Na linha 11 os candidatos são então ordenados de acordo com esse benefício e uma Lista Restrita de Candidatos (LRC) é definida a partir do parâmetro de aleatoriedade $\alpha \in [0, 1]$, o qual controla o equilíbrio entre comportamento guloso e diversificação. O parâmetro α irá configurar qual a porcentagem dos melhores vértices da LRC será utilizada para escolher de forma randômica o próximo vértice a ser adicionado na solução.

Assim, na linha 12 é calculado a quantidade máxima de melhores candidatos podem ser selecionados para a solução com base no valor de $\alpha \in [0, 1]$ escolhido. Após essa escolha, entre as linhas 13 e 16 um vértice e um *cluster* são selecionado de forma semi aleatória a partir da LRC, entre o primeiro vértice e o índice de vértice máximo calculado com base no α, e assim inserido no *cluster* correspondente e removido da LRC nas linhas 17 e 18, desde que a inserção preserve a viabilidade da solução.

Ao final do processo, obtém-se uma solução inicial viável para o PCC, construída de forma gulosa e randomizada, usada como primeira solução do RGRASP para ser comparada com as próximas soluções geradas internamente.

5.2 Algoritmo Construtivo Randomizado Reativo

Nesta seção é apresentado o Algoritmo Construtivo Randomizado Reativo, utilizado como componente construtivo do RGRASP para o Problema de Clusterização Capaci-

tada (PCC). O objetivo desse algoritmo é gerar soluções iniciais diversificadas e de boa qualidade, equilibrando critérios gulosos e aleatórios por meio do uso de uma lista de candidatos controlada pelo parâmetro α. Além disso, a abordagem reativa permite ajustar dinamicamente o parâmetro α ao longo das iterações, favorecendo automaticamente aqueles valores que conduzem a soluções de melhor qualidade e aumentando a eficiência do processo construtivo.

Para elucidar o conceito do parâmetro α e ser melhor compreendido ao ser lido o pseudocódigo, podemos dizer que o valor dele é o parâmetro que

O valor de α é atualizado entre as linhas 20 e 24 onde é selecionado de forma reativa a partir de um conjunto pré-definido, variando de 0,05 a 0,50, com probabilidades inicialmente uniformes que são atualizadas após um número fixo de iterações, favorecendo os valores de α associados às melhores soluções encontradas.

Para elucidar o conceito do parâmetro α e facilitar a compreensão do pseudocódigo, define-se tal valor como o coeficiente que controla o equilíbrio entre a intensificação e a diversificação durante a fase de construção, assim controlando o quanto o algoritmo será guloso. Na prática, o α determina o limiar de qualidade para a inclusão de elementos da Lista Restrita de Candidatos (LRC): valores baixos de α tornam o algoritmo mais guloso, selecionando apenas as melhores opções locais, enquanto valores mais altos aumentam a variância das soluções geradas. Em termos práticos, o α delimita o tamanho da Lista Restrita de Candidatos (LRC): um valor de $\alpha = 0{,}2$, por exemplo, instrui o algoritmo a restringir o sorteio aleatório exclusivamente aos 20% melhores vértices candidatos disponíveis naquela iteração. Periodicamente, as probabilidades de seleção de cada α são recalculadas, de modo que os valores que historicamente geraram as melhores soluções tenham suas chances de escolha aumentadas nas iterações seguintes.

Para elucidar o conceito do parâmetro α e facilitar a compreensão do pseudocódigo, define-se tal valor como o coeficiente que controla o equilíbrio entre a voracidade e a aleatoriedade na construção da solução. O valor de α é atualizado entre as linhas 20 e 24 onde é selecionado de forma reativa a partir de um conjunto pré-definido, variando de 0,05 a 0,50, com probabilidades inicialmente uniformes que são atualizadas após um número fixo de iterações, favorecendo os valores de α associados às melhores soluções encontradas.

Algoritmo 2: Algoritmo construtivo randomizado reativo

```
   Entrada: grafoG, α, iterPorAlpha, iterMax
   Saída: Solução S*
 1 início
 2     contadorIter ← 0;
 3     S ← ∅;
 4     Cria vetor de probabilidades com distribuição uniforme P;
 5     enquanto Não atingir iterMax iterações faça
 6         Cria lista de candidatos ordenada LC;
 7         Adiciona um candidato aleatório dentre os α melhores da LC à cada
           cluster de S;
 8         Atualiza e reordena a LC;
 9         enquanto LC não estiver vazia faça
10             Candidato ← Escolhe aleatoriamente um Candidato dentre os α
               melhores;
11             MelhorCluster ← Encontra o melhor cluster para o Candidato;
12             Insere o Candidato no MelhorCluster de S;
13             Atualiza e reordena a LC;
14         fim enqto
15         se Solução for inviável então
16             viabilize (S);
17         fim se
18         S* ← Escolha a melhor solução entre S e S*;
19         Incremente ContadorIter;
20         se contadorIter atingir iterPorAlpha então
21             Atualize vetor de probabilidades P;
22             Escolha um novo α;
23             ContadorIter ← 0;
24         fim se
25     fim enqto
26     retorna S*;
27 fim
```

O Algoritmo 2 descreve o procedimento construtivo empregado a cada iteração do RGRASP. Inicialmente nas linhas 2 a 4 é inicializadas as variáveis e criado o vetor de probabilidades que faz o algoritmo ser reativo as melhores soluções, assim inicia-se a iteração entre as linhas 5 e 25, na linha 6 todos os vértices do grafo são inseridos em uma Lista de Candidatos (LC), a qual é atualizada e ordenada na linha 8 de acordo com o somatório dos benefícios das arestas associadas a cada vértice. A construção da solução inicia-se pela inserção de vértices escolhidos aleatoriamente dentre os α melhores candidatos da LC na iteração interna entre as linhas 9 e 14, garantindo diversidade desde as primeiras decisões. Em seguida, enquanto houver candidatos não alocados, um

vértice é selecionado aleatoriamente a partir da LC e inserido no *cluster* que proporciona o maior benefício incremental. O valor de α é atualizado entre as linhas 20 e 24 onde é selecionado de forma reativa a partir de um conjunto pré-definido, variando de 0,05 a 0,50, com probabilidades inicialmente uniformes que são atualizadas após um número fixo de iterações, favorecendo os valores de α associados às melhores soluções encontradas.

5.3 Randomized Variable Neighborhood Descent

O RVND é o núcleo do processo de intensificação deste trabalho, ele explora diferentes estruturas de vizinhança de forma randômica a partir da aplicação das buscas locais para refinara solução atual. Esse processo continua até que todas as vizinhanças sejam exploradas sem melhora ou até que o limite de iterações seja atingido. O funcionamento geral segue descrito abaixo:

1. Definição das vizinhanças e embaralhamento da ordem inicial.
2. Execução da busca local correspondente à vizinhança atual N_l.
3. Se houver melhoria, aceitar a solução e retornar à primeira vizinhança.
4. Caso contrário, avançar para a próxima vizinhança.
5. Encerrar quando todas as vizinhanças forem exploradas sem melhora.

Além da randomização da sequência de vizinhanças, o RVND implementado incorpora um limite máximo de iterações como critério de parada adicional, evitando ciclos excessivos de busca local e controlando o custo computacional do método. O critério de aceitação das soluções segue a melhoria do valor da função objetivo, de modo que apenas soluções que apresentem ganho de benefício em relação à solução corrente são aceitas. Essas modificações tornam o procedimento de busca local mais flexível e adequado à integração com as meta-heurísticas empregadas neste trabalho.

O critério de parada padrão é o fim das estruturas de vizinhança, o que equivale a não se encontrar uma solução melhor em nenhuma vizinhança, ou seja, as buscas locais implementadas já não surtem efeitos positivos na solução. Adicionalmente, foi incorporado

um limite máximo de iterações como critério de parada suplementar, controlando assim o custo computacional do algoritmo, visando restringir o número de buscas locais aplicadas às soluções intermediárias geradas pelo SA.

No contexto do procedimento de busca local utilizando RVND, as estruturas de vizinhança definidas abaixo não são exploradas em uma sequência determinística. A cada iteração do RVND, a lista de vizinhanças é embaralhada, e o algoritmo explora o espaço de busca na ordem definida por essa permutação aleatória. O funcionamento de cada movimento pode ser descrito da seguinte forma:

- **Realocação (*Shift*):** examina a transferência unilateral de um vértice de seu *cluster* atual para um *cluster* de destino. A operação é validada apenas se a inserção respeitar a capacidade do grupo receptor, sendo fundamental para ajustes finos no balanceamento de carga.

- **Troca (*Swap*):** consiste na permuta simples entre dois vértices pertencentes a grupos distintos. Diferente da realocação, este movimento tende a manter o nível de preenchimento dos *clusters* relativamente estável, pois a entrada de um elemento é compensada pela saída de outro.

- **Troca Assimétrica (*Swap 2-1*):** realiza a substituição de dois vértices de um mesmo grupo por um único vértice proveniente de outro grupo. Trata-se de um movimento mais agressivo, capaz de alterar significativamente a cardinalidade e a configuração espacial dos agrupamentos envolvidos.

A combinação dessas três vizinhanças dentro da estratégia do RVND confere robustez ao algoritmo. Enquanto os movimentos de Realocação e Troca atuam refinando a solução, o movimento assimétrico (2-1) além de refinar, também facilita a exploração de novas regiões do espaço de busca, permitindo que o método escape de ótimos locais onde trocas simples seriam insuficientes para promover melhorias.

O Algoritmo 3 descreve formalmente o procedimento padrão do RVND que inicia a partir de uma solução viável s dada como entrada e percorre iterativamente o conjunto de vizinhanças $N_1, N_2, \ldots, N_k$. O algoritmo inicia randomizando a lista de vizinhanças na linha 2, assim a ordem de uso das buscas locais é aleatorizada. A cada iteração entre

as linhas 5 e 17, em que o critério de parada é finalizar a busca em todas as vizinhanças sem melhoria naquele ponto, alcançando então um ótimo local.

Dentro do processo iterativo, na linha 6 uma nova solução candidata s' é gerada por meio da aplicação da vizinhança corrente, fazendo então a modificação da solução atual por meio de uma das buscas locais. Entre as linhas 7 e 10 caso seja observada uma melhoria no valor da função objetivo, isto é, $f(s') > f(s)$, a solução é atualizada na linha 8 e o processo retorna à primeira vizinhança na linha 9, reiniciando o ciclo de exploração. Caso contrário, a busca avança para a próxima vizinhança na linha 12. O procedimento é encerrado quando todas as vizinhanças são exploradas sem que qualquer melhoria seja encontrada, que é o critério de parada da iteração que pode ser visto na linha 5, assim caracterizando um ótimo local em relação ao conjunto de vizinhanças consideradas. Na linha 15 o algoritmo verifica se a quantidade máxima de iterações do RVND foi alcançada, o que faz a iteração finalizar logo em seguida caso tenha alcançado. No contexto deste trabalho, o RVND é utilizado como mecanismo de intensificação, sendo acoplado a meta-heurísticas de mais alto nível, contribuindo significativamente para o refinamento das soluções geradas e para o aumento da qualidade final dos resultados obtidos.

Algoritmo 3: Randomized Variable Neighborhood Descent(RVND)

Entrada: Grafo G, Solução inicial $bestSolution$, número máximo de iterações $maxIter$
Saída: Solução refinada $bestSolution$

```
1  início
2      Randomiza lista com k_max vizinhanças;
3      ContadorIter ← 0;
4      k ← 0;
5      enquanto k < k_max faça
6          s' ← gerarVizinhancaN^(k)(s);
7          se f(s') > f(s) então
8              s ← s';
9              k ← 0;
10         fim se
11         senão
12             Incrementar(k);
13         fim se
14         Incrementar(ContadorIter);
15         se ContadorIter = maxIter então
16             break;
17         fim se
18     fim enqto
19     retorna s;
20 fim
```

Essa abordagem permite um balanceamento eficaz entre intensificação e diversificação, sendo amplamente utilizada em problemas combinatórios. Além disso, a aleatorização inicial na ordem das vizinhanças reduz o risco de padrões determinísticos indesejados e reforça a capacidade exploratória do método. Abaixo as buscas locais utilizadas no trabalho são detalhadas para um melhor entendimento.

5.3.1 Busca Local 1 - Realocação (*Shift*) 1-0

A Busca Local *1-to-0*, tem como objetivo melhorar a solução movendo um único vértice de um *cluster* para outro, desde que a operação gere uma melhoria no benefício e mantenha a viabilidade dos *clusters* envolvidos.

Algoritmo 4: Busca Local 1–0

```
Entrada: Solucao s
Saída: Solução aprimorada s'
1  início
2      melhora ← 0;
3      embaralhar ordem dos clusters;
4      para cada cluster g1 faça
5          para cada vértice v ∈ g1 faça
6              para cada cluster g2 ≠ g1 faça
7                  verificar viabilidade de mover v para g2;
8                  calcular variação de benefício Δ;
9                  se Δ > 0 então
10                     realizar movimento;
11                     melhora ← melhora + Δ;
12                 fim se
13             fim para cada
14         fim para cada
15     fim para cada
16     retorna s';
17 fim
```

Este procedimento considera todos os *clusters* em ordem aleatória e, para cada vértice de cada *cluster*, avalia o impacto de removê-lo de seu *cluster* atual e inseri-lo em outro *cluster*. A operação é aplicada apenas quando:

1. os limites inferior e superior dos *clusters* permanecem respeitados;

2. o ganho de benefício é estritamente positivo;

3. a alteração não gera violação estrutural no problema.

A busca local do tipo 1–0 consiste em tentar mover um único vértice de um *cluster* para outro, avaliando o impacto dessa movimentação na função objetivo. Para cada vértice v pertencente ao *cluster* g_1, o método verifica a viabilidade de inseri-lo em outro *cluster* g_2, considerando tanto a restrição de limites de peso quanto a variação no benefício.

A melhoria é calculada como a diferença entre o benefício perdido em g_1 devido à remoção do vértice e o benefício ganho pela inserção em g_2. Se essa variação for positiva, o movimento é realizado. Esse processo é repetido sobre todos os vértices e *clusters*, conduzindo a solução a um ótimo local associado aos movimentos 1–0. Apesar de ser computacionalmente custoso, buscamos fazer a atualização apenas do que foi alterado.

A introdução de embaralhamento na ordem dos *clusters* reduz o viés determinístico e aumenta a capacidade de explorar múltiplas composições de vizinhança.

Ao longo do processo, o método armazena a melhor troca encontrada e, ao final, aplica efetivamente a modificação com a maior melhoria. Caso nenhuma troca resulte em benefício positivo, a busca local encerra-se sem modificações.

5.3.2 Busca Local 2 - Troca (*Swap*) 1-1

A Busca Local *1-to-1*, realiza trocas entre pares de vértices pertencentes a *clusters* distintos. O objetivo é encontrar melhorias no valor da função objetivo substituindo um vértice de um *cluster* por outro vértice de outro *cluster* e fazendo a troca entre eles.

A cada iteração, o algoritmo:

- seleciona dois *clusters* distintos;
- escolhe um vértice de cada *cluster*;
- avalia o impacto da troca simultânea entre os *clusters*.

Algoritmo 5: Busca Local 1–1

Entrada: Solução s
Saída: Solução aprimorada s

```
melhora ← 0;
embaralhar ordem dos clusters;
para cada cluster g1 faça
    para cada vértice v1 ∈ g1 faça
        para cada cluster g2 ≠ g1 faça
            para cada vértice v2 ∈ g2 faça
                verificar viabilidade da troca (v1, v2) entre g1 e g2
                calcular variação de benefício:
                Δ = Δ_rem(v1, g1) + Δ_rem(v2, g2) + Δ_add(v1, g2) + Δ_add(v2, g1)
                se Δ > 0 então
                    realizar troca (v1, v2);
                    melhora ← melhora + Δ
                fim se
            fim para cada
        fim para cada
    fim para cada
fim para cada
return s
```

A busca local 1–1 explora vizinhanças formadas pela troca simultânea de um vértice de cada par de *clusters* distintos. Dado um vértice v_1 pertencente ao *cluster* g_1 e

um vértice v_2 pertencente ao *cluster* g_2, o método avalia a troca $(v_1 \leftrightarrow v_2)$.

A troca somente é considerada se respeitar as restrições de viabilidade dos *clusters*, especialmente os limites inferior e superior de peso. A variação de benefício é computada como a soma entre as reduções de benefício decorrentes das remoções e os acréscimos provenientes das inserções. Se a troca apresentar melhoria positiva na função objetivo, ela é executada.

Esse tipo de movimento é mais expressivo que a busca local 1–0, pois permite modificar simultaneamente a configuração de ambos os *clusters* envolvidos, podendo escapar de ótimos locais associados apenas à remoção e inserção simples. A troca 1–1 é particularmente eficiente em problemas onde a estrutura da solução é sensível ao equilíbrio entre os *clusters*.

O custo e o benefício resultantes são comparados com o estado atual da solução. A operação é aplicada apenas quando:

1. ambos os *clusters* permanecem viáveis;
2. há ganho de benefício maior que zero.

O processo termina quando nenhuma troca adicional melhora a solução.

5.3.3 Busca Local 3 - Troca Assimétrica (*Swap*) 2-1

A Busca Local *2-to-1* consiste na remoção de dois vértices de um *cluster* e inserção de um vértice em seu lugar, ou em movimentos equivalentes que envolvem múltiplos vértices. Esse tipo de vizinhança é mais complexa, pois altera simultaneamente a composição e o peso total dos *clusters*.

Algoritmo 6: Busca Local 2–1

Entrada: Solução s
Saída: Solução aprimorada s

melhora $\leftarrow 0$;
embaralhar ordem dos *clusters*;
para cada *cluster* g_1 **faça**
  para cada *cluster* $g_2 \neq g_1$ **faça**
    para cada *par de vértices* $(v_a, v_b) \subset g_1$ **faça**
      para cada *vértice* $v_c \in g_2$ **faça**
        verificar viabilidade da troca $\{v_a, v_b\} \leftrightarrow \{v_c\}$
        calcular variação de benefício:
        $\Delta_{rem} \leftarrow \Delta_{rem}(v_a, g_1) + \Delta_{rem}(v_b, g_1) + \Delta_{rem}(v_c, g_2)$
        $\Delta_{rem} \leftarrow \Delta_{add}(v_a, g_2) + \Delta_{add}(v_b, g_2) + \Delta_{add}(v_c, g_1)$
        $\Delta \leftarrow \Delta_{rem} + \Delta_{add}$
        se $\Delta > 0$ **então**
          remover v_a e v_b de g_1 e inserir em g_2 remover v_c de g_2 e inserir em g_1 melhora $\leftarrow$ melhora + Δ
        fim se
      fim para cada
    fim para cada
  fim para cada
fim para cada
return s

A busca local 2–1 expande a vizinhança explorada ao permitir trocas assimétricas entre *clusters*, nas quais dois vértices de um *cluster* são trocados por um único vértice do *cluster* vizinho. Esse movimento é especialmente útil em problemas onde os *clusters* possuem faixas de viabilidade amplas e onde combinações de vértices podem gerar benefícios substancialmente superiores aos obtidos por modificações unitárias.

Para cada par de vértices (v_a, v_b) pertencentes ao *cluster* g_1 e cada vértice v_c pertencente ao *cluster* g_2, é avaliada a troca $\{v_a, v_b\} \leftrightarrow \{v_c\}$. O movimento é considerado apenas se ambos os *clusters* permanecerem dentro dos limites de peso estabelecidos após a troca.

A variação de benefício é calculada pela combinação das remoções e inserções realizadas em ambos os *clusters*. Caso Δ seja positivo, a troca é executada. Esse tipo de vizinhança permite mudanças mais profundas na estrutura da solução, proporcionando saltos maiores no espaço de busca e aumentando a probabilidade de escapar de ótimos locais que não podem ser superados por vizinhanças 1–0 ou 1–1. Seu uso é particularmente relevante em cenários onde a estrutura dos *clusters* impede melhorias por vizinhanças mais simples.

5.4 Perturbação - Desconstrução e Reconstrução

O mecanismo de perturbação é feito de forma estruturada e é utilizado em dois momentos diferentes, no RGRASP e no SA, sendo essencial para sair de ótimos locais e saltar para novos espaços de busca para que o refinamento aconteça. Ele desempenha um papel fundamental na estratégia deste trabalho, sendo executado a cada iteração na solução corrente. Logo após a execução dele, é realizado um refinamento com as estruturas de busca local do RVND, operando de uma forma que a pertubação consiga sair de um ótimo local quando é alcançado e encontrar novos espaços de busca. Esse algoritmo de perturbação opera em duas fases principais, a Desconstrução e Reconstrução e podem ser vistas abaixo em detalhes.

Na etapa de desconstrução, remove-se parcialmente a estrutura da solução atual. No algoritmo implementado, remove-se uma fração dos vértices de alguns *clusters*, escolhidos probabilisticamente com base nos parâmetros:

- *pElementos*: probabilidade de um vértice de um *cluster* ser removido;
- *pClusters*: probabilidade de um *cluster* ser selecionado para remoção.

Os vértices removidos são armazenados em uma lista de candidatos.

Após a desconstrução, é feito o processo de reconstrução, em que cada vértice removido é reinserido na solução utilizando uma estratégia gulosa. O algoritmo:

1. primeiro preenche *clusters* abaixo do limite inferior;
2. depois insere os vértices restantes, escolhendo, para cada um, o *cluster* que maximiza o benefício incremental.

Como explicado acima e também pode ser visto no algoritmo 7, ele primeiro faz a desconstrução de parte da solução e depois reconstrói de forma gulosa. Esse processo permite escapar de ótimos locais e reconstruir soluções mais robustas, preservando parcialmente a estrutura original.

Algoritmo 7: Perturbação da Solução

Entrada: Solução s, parâmetros $probabil_{elem}$ e $probabil_{grp}$
Saída: Solução perturbada s'

início
  $s' \leftarrow$ cópia de s; criar lista de candidatos L;
  para cada *cluster* g *em* s' **faça**
    $q = \lfloor |g| \cdot probabil_{elem} \rfloor$;
    gerar probabilidade $u \in [0, 1]$;
    se $u \leq probabil_{grp}$ **então**
      x **para** $i \leftarrow 1$ **to** q **faça**
        gerar probabilidade $r \in [0, 1]$;
        se $r \leq probabil_{elem}$ **então**
          selecionar vértice aleatório $v \in g$;
          remover v de g e adicionar v a L;
        fim se
      fim para
    fim se
  fim para cada
  enquanto *existe cluster* g *com* $custo(g) < lower(g)$ **faça**
    para cada candidato $c \in L$ calcular benefício de inserção;
    inserir o candidato com maior benefício no cluster correspondente;
  fim enqto
  para cada *candidato disponível* $c \in L$ **faça**
    encontrar cluster g de maior benefício viável;
    inserir c em g;
  fim para cada
  retorna s';
fim

A etapa de perturbação tem como objetivo diversificar a busca, deslocando a solução atual para uma região distinta do espaço de busca. A estratégia adotada remove probabilisticamente um subconjunto de vértices de cada *cluster*, controlado pelos parâmetros $probabil_{elem}$ (probabilidade de remoção de elementos) e $probabil_{grp}$ (probabilidade de um *cluster* ser perturbado). Os vértices removidos são armazenados em uma lista de candidatos e posteriormente reinseridos de maneira gulosa.

A reinserção ocorre em duas fases. Primeiro, preenchem-se os *clusters* abaixo do limite inferior, garantindo a viabilidade estrutural da solução. Em seguida, os demais candidatos são inseridos com base no maior benefício marginal, definido como a soma das distâncias entre o candidato e os elementos do *cluster* receptor. Esse procedimento conduz à solução para uma nova região promissora, preservando parte da boa estrutura previamente construída e permitindo a saída de ótimos locais.

5.5 Reactive Greedy Randomized Adaptive Search Procedure

Esta seção descreve a utilização do RGRASP, apresentado no Algoritmo 8, com foco em explicar a meta-heurística adotada, o conjunto de valores α considerados, o fluxo de execução, o mecanismo reativo empregado para sua seleção ao longo das iterações e o critério de atualização da melhor solução.

Algoritmo 8: RGRASP com RVND

Entrada: Grafo G, número máximo de iterações $maxIter$
Saída: Melhor solução encontrada

início
  $bestSolution \leftarrow$ GreedyAlgorithm(G);
  $valBest \leftarrow$ valor($bestSolution$);
  Definir conjunto de alfas $\alpha = \{0.10, 0.15, \ldots, 0.70\}$;
  Inicializar probabilidades iguais $p(\alpha_i)$;
  Inicializar somaResultados[i] $\leftarrow 0$;
  Inicializar qtdEscolhido[i] $\leftarrow 0$;
  Criar distribuição discreta baseada em $p(\alpha_i)$;
  para $iter = 1$ *é* $maxIter$ **faça**
    $i \leftarrow$ índice sorteado segundo distribuição discreta;
    $\alpha_{atual} \leftarrow \alpha[i]$;
    $currentSolution \leftarrow$ RandomizedGreedyAlgorithm(G, α_{atual});
    RVND_CCP($G, currentSolution, 200$);
    somaResultados[i] $\leftarrow$ somaResultados[i] + valor($currentSolution$);
    qtdEscolhido[i] $\leftarrow$ qtdEscolhido[i] + 1;
    se *currentSolution é viável* **e** *valor(currentSolution)* $>$ *valBest* **então**
      $bestSolution \leftarrow currentSolution$;
      $valBest \leftarrow$ valor($currentSolution$);
    fim se
    se $iter \mod 30 = 0$ **então**
      para cada j *em* $[1, \ldots, |\alpha|]$ **faça**
        se *qtdEscolhido[j]* $>$ *0* **então**
          média $\leftarrow$ somaResultados[j] / qtdEscolhido[j];
          $q[j] \leftarrow \left(\frac{\text{média}}{valBest}\right)^{\delta}$, com $\delta = 20$;
        fim se
      fim para cada
      Normalizar vetor q para obter novas probabilidades;
      Atualizar distribuição discreta $p(\alpha)$;
    fim se
  fim para
  retorna $bestSolution$;
fim

O RGRASP é utilizado para a construção das soluções iniciais posteriormente

empregadas no SA, no qual o processo construtivo guloso é combinado com mecanismos de aleatoriedade controlada e adaptação dinâmica do parâmetro de aleatoriedade. Além disso, cada solução construída é refinada por meio do RVND, após executar uma perturbação dos vértices e *clusters*, resultando em soluções iniciais mais robustas e próximas de ótimos locais de alta qualidade, o que exerce influência direta na eficiência e na qualidade da busca conduzida pelo *Simulated Annealing*, pois, quando o SA é executado, as soluções inicialmente criadas são promissoras.

O Algoritmo 8 inicia-se com a obtenção de uma solução inicial por meio de um algoritmo guloso determinístico na linha 2, utilizado como referência para a avaliação das soluções subsequentes. A cada iteração do RGRASP, um valor de α é selecionado de forma probabilística a partir de um conjunto pré-definido na linha 4 e na distribuição discreta de probabilidades na linha 8. Ao iniciar a iteração da linha 9 a 30 usando como critério de parada o parâmetro de quantidade de iterações máximas, nas linhas 10 e 11 o algoritmo fixa o parâmetro α atual a ser empregado na fase construtiva do algoritmo guloso randomizado reativo na linha 12, responsável por gerar uma solução inicial diversificada. Em seguida na linha 13, essa solução é submetida ao procedimento RVND específico para o PCC, que explora sistematicamente diferentes estruturas de vizinhança visando a obtenção de um ótimo local.

Ao longo das iterações, estatísticas associadas a cada valor de α são atualizadas entre as linhas 20 e 26, permitindo o ajuste reativo das probabilidades de seleção com base na qualidade média das soluções obtidas. Periodicamente, como pode ser visto na linha 20, a cada 30 iterações, essas probabilidades são recalculadas e na linha 27 e 28 é feita a normalização e atualização da distribuição discreta, responsável por ser escolhido o parâmetro α na próxima iteração, favorecendo valores de α com melhor desempenho relativo, enquanto a melhor solução viável encontrada durante todo o processo é armazenada como solução final do algoritmo.

5.6 Abordagem Híbrida: *Simulated Annealing* com RGRASP e RVND

Nesta seção é apresentada a abordagem híbrida que integra as meta-heurísticas *Simulated Annealing* (SA), RGRASP e RVND em um único *framework* de otimização. A estratégia adotada explora a complementaridade entre essas técnicas, utilizando o RGRASP como mecanismo de geração de soluções iniciais diversificadas, o RVND como procedimento de intensificação local e o SA como método base e de aceitação probabilística capaz de escapar de ótimos locais. Essa combinação visa equilibrar de forma eficiente os processos de diversificação e intensificação da busca, permitindo uma exploração mais robusta do espaço de soluções. No fluxograma visto na imagem 5.1 e nos parágrafos a seguir, são apresentados como os principais componentes da abordagem proposta se interagem entre eles no decorrer do processo de otimização.

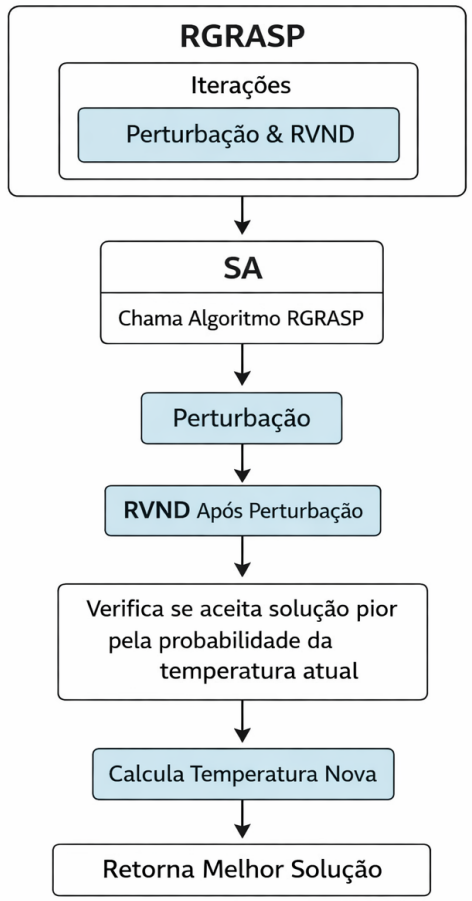

Figura 5.1: Fluxograma completo da abordagem proposta

Algoritmo 9: Meta-heurística *Simulated Annealing* (SA) aplicada ao PCC

Entrada: $grafo$, α, $txAceitacao$, $iterMaxSA$, $iterMaxVND$, $porctgPerturbElementos$, $porctgPerturbClusters$
Saída: s: solução subótima do PCC

```
1  início
2      tempAtual← temperaturaInicial(tempAtual, 1.25, 0.95, iterMax);
3      best ← RGRASP_RVND(grafo);
4      s* ← s;
5      contEstagnacao ← 0;
6      enquanto tempAtual > tempFinal faça
7          contEstagnacao ← contEstagnacao + 1;
8          enquanto iterTempAtual < iterMaxSA faça
9              se contEstagnacao > iterMax * 0.3 então
10                 break;
11             fim se
12             iterTempAtual ← iterTempAtual + 1 s' ←
               Perturbacao(s*, porctgPerturbElementos, porctgPerturbClusters)
               s' ← RVND(s*, iterMaxVND)
13             Δ ← s'.valor() − s.valor()
14             se Δ > 0 então
15                 se s' é viável e s'.valor() > best.valor() então
16                     best ← Copia(s') contEstagnacao ← 0
17                 fim se
18                 s ← Copia(s')
19             fim se
20             senão
21                 u ← Rand(0, 1) se u < e^(Δ/tempAtual) então
22                     s ← Copia(s')
23                 fim se
24             fim se
25         fim enqto
26         tempAtual ← α · tempAtual;
27         iterTempAtual ← 0
28     fim enqto
29     retorna best;
30 fim
```

O Algoritmo 9 descreve a meta-heurística híbrida proposta, na qual podem ser observadas com mais clareza a estruturação da meta-heurística SA e as etapas em que foram combinados o RGRASP e o RVND. Inicialmente, é calculada a temperatura inicial por meio de um procedimento específico de calibração, conforme descrito na subseção 5.6.1. Em seguida, uma solução inicial de alta qualidade é gerada utilizando o algoritmo RGRASP_RVND, o qual combina uma construção gulosa randomizada reativa com refinamento por RVND. Essa solução é utilizada como ponto de partida para o processo de busca do SA, sendo também armazenada como a melhor solução encontrada até o

momento.

O laço principal do algoritmo corresponde ao processo de resfriamento do SA, no qual a temperatura é progressivamente reduzida até atingir um valor final predefinido. Para cada temperatura corrente, é executado um laço interno que realiza um número máximo de iterações controlado por *iterMaxSA*. Em cada iteração, a solução corrente é submetida a um operador de perturbação, no qual uma porcentagem dos vértices e dos *clusters* da solução é modificada, conforme os parâmetros *porctgPerturbElementos* e *porctgPerturbClusters*. Essa perturbação visa promover diversificação e permitir a exploração de novas regiões do espaço de busca.

Após a perturbação, a solução candidata é refinada por meio do método RVND, limitado a um número máximo de iterações (*iterMaxVND*), atuando como mecanismo de intensificação local. A aceitação da nova solução segue o critério clássico do *Simulated Annealing*: soluções que apresentam melhora no valor da função objetivo são sempre aceitas, enquanto soluções piores podem ser aceitas com uma probabilidade dependente da variação de qualidade Δ e da temperatura atual, conforme definido na subseção 5.6.2.

Adicionalmente, o algoritmo incorpora um critério de estagnação, monitorado por meio de um contador de iterações consecutivas sem melhoria. Caso esse contador ultrapasse 30% do número máximo de iterações permitidas para a temperatura corrente, o laço interno é interrompido antecipadamente, evitando gasto computacional excessivo em regiões pouco promissoras do espaço de busca. Ao final de cada ciclo interno, a temperatura é atualizada segundo a taxa de resfriamento definida na Seção 5.6.3.

Por fim, ao término do processo de resfriamento, a melhor solução viável encontrada ao longo de toda a execução é retornada como solução subótima do PCC. A combinação entre diversificação controlada (SA), intensificação local (RVND) e construção inicial robusta (RGRASP) resulta em um método híbrido capaz de explorar eficientemente o espaço de soluções e obter soluções de alta qualidade para instâncias complexas do problema.

5.6.1 Cálculo da temperatura inicial

Para este trabalho, utilizou-se um método extra para realizar o cálculo da temperatura inicial. Este método recebe uma taxa de aceitação mínima β e itera sobre várias soluções, com incrementos graduais de temperatura enquanto a taxa de aceitação não atingir o mínimo esperado. Portanto, para o melhor entendimento de como é feito o cálculo da temperatura inicial, que é uma das coisas mais importantes nessa meta-heurística.

O algoritmo 10 ilustra o método utilizado para se encontrar uma boa temperatura inicial. Este método recebe uma taxa de aceitação mínima e itera sobre várias soluções, com incrementos graduais de temperatura ditados por β enquanto a aceitação não atingir o mínimo esperado.

Entrada: Grafo $G(V, E)$, β, $TaxaDeAceitacao$, $IterPorTemperatura$
Saída: Melhor solução encontrada, Temperatura inicial

```
1  início
2      s* ← ConstrutivoRandomizadoReativo();
3      s ← s*;
4      Temperatura ← 100;
5      ContadorIter ← 0;
6      aceitos ← 0;
7      enquanto Taxa de aceitação não for atingida faça
8          ContadorIter ← 0;
9          enquanto ContadorIter < IterPorTemperatura faça
10             Incrementar(ContadorIter);
11             s ← Perturbacao(s, 0.3);
12             Δ ← f(s) − f(s*);
13             se Δ > 0 então
14                 s* ← s;
15                 Incrementar(aceitos);
16             fim se
17             senão
18                 se Condição de aceitação for satisfeita então
19                     Incrementar(aceitos);
20                 fim se
21             fim se
22         fim enqto
23         Temperatura = Temperatura × β;
24     fim enqto
25     retorna s*, Temperatura;
26 fim
```

5.6.2 Taxa de aceitação de soluções

O critério de aceitação segue o modelo clássico do SA. Se a solução vizinha apresenta melhoria em relação à solução corrente, ela é imediatamente aceita. Caso contrário, a solução pior pode ser aceita com uma probabilidade dada por:

$$P(\text{aceitação}) = e^{\Delta/T},$$

onde Δ representa a variação de qualidade (benefício) entre as soluções, e T é a temperatura atual. Essa estratégia controlada de aceitação de soluções piores permite ao algoritmo escapar de ótimos locais e explorar regiões mais amplas, sendo especialmente relevante em problemas combinatórios altamente restritivos como o PCC.

Ao fim de cada ciclo de iteração, a temperatura é atualizada de acordo com um fator de resfriamento α, que aplica um decaimento multiplicativo. O processo prossegue até que a temperatura final seja atingida. Após o término do resfriamento, o algoritmo realiza uma busca local final utilizando RVND para garantir que a melhor solução encontrada esteja devidamente otimizada.

A taxa de aceitação A formulada na equação 1 considera a diferença de qualidade Δ dividida pela temperatura atual T.

$$A = \exp^{\Delta/T} \tag{1}$$

Sempre que uma solução é aceita, seja pela equação 1 ou por ela ser a melhor conhecida, é realizado um número fixo de iterações de refinamento pelo RVND.

5.6.3 Taxa de resfriamento

Para a taxa de resfriamento T_n formulada na equação 2, os decrementos ocorrem em função de um percentual α e um valor c. Este valor constante foi necessário para acelerar o resfriamento em baixas temperaturas, devido à alta taxa de estagnação das soluções. Assim, a nova temperatura T_n é dada por:

$$T_n = (T_{n-1} \times \alpha) - (c \times NumeroDeEstagnacoes) \tag{2}$$

6 Experimentos Computacionais

Neste capítulo apresentam-se a metodologia de teste, o ambiente computacional, parâmetros, os resultados dos testes da estratégia proposta, a análise e comparação com os resultados das estratégias de referência na literatura para solução do PCC.

6.1 Ambiente de testes

Todos os algoritmos foram implementados em *C++ 17*, compilados utilizando o *GCC 6.3.0* com a diretiva *-O3* e executados em um computador com a seguinte configuração: processador Intel(R) Core(TM) i7-8700 CPU @ 3.20 GHz, com 32 GB de memória RAM DDR4, no sistema operacional Linux Deepin.

Buscando garantir uma comparação justa com a literatura, já que são executados em processadores diferentes, buscamos um multiplicador baseado no benchmark PassMark CPU Mark[1]. Verificou-se que o processador (Intel Xeon E5-2670) utilizado no trabalho de referência da literatura em (LIU; GUO; ZENG, 2022a) tem 6% mais poder computacional que o processador (Intel Core i7-8700) usado para executar os resultados deste trabalho, com pontuação de 12.748, contra 13.473. Portanto, os tempos de execução foram multiplicados por um fator aproximado de 0,94 para manter a paridade entre o hardware dos dois trabalhos.

6.2 Instâncias de teste

Para a avaliação computacional dos métodos propostos, foram utilizadas instâncias pertencentes ao conjunto de *benchmark* denominado CCPLIB[2], amplamente empregado na literatura para experimentos envolvendo o PCC. Ao todo, foram consideradas 50 instâncias, divididas em diferentes subconjuntos, conforme descrito a seguir.

Essas instâncias utilizadas na literatura possuem algumas características: consti-

[1]Disponível em: <https://www.cpubenchmark.net/compare/3099vs2337/>
[2]Disponível em: <https://grafo.etsii.urjc.es/optsicom/ccp.html>

tuem grafos completos, as instâncias são agrupadas conforme o número de vértices, todos os vértices possuem peso e todos os *clusters* têm o mesmo limite de capacidade. O primeiro conjunto é baseado nas instâncias propostas por Deng e Bard (DENG; BARD, 2011) no contexto do problema de entrega de correspondências, consistindo em 10 instâncias de $n = 82$ vértices e $p = 8$ *clusters*, com limites de capacidade definidos como $L = 25$ e $U = 75$. O segundo conjunto, denominado RanReal, foi originalmente proposto por (GALLEGO et al., 2013) no contexto do *Maximally Diverse Grouping Problem* (MDGP). Esse conjunto é dividido em dois subconjuntos: 20 instâncias com $n = 240$, $p = 12$, limite inferior $L = 75$ e limite superior $U = 125$, e outras 20 instâncias com $n = 480$, $p = 20$, $L = 100$ e $U = 150$.

De forma resumida, as seguintes instâncias da CCPLIB foram utilizadas nos experimentos:

- 10 instâncias Sparse82.
- 20 instâncias RanReal240.
- 20 instâncias RanReal480.

6.3 Parametrização de variáveis

Com o código desenvolvido, foi necessária a realização de testes, buscando uma melhor parametrização com o intuito de estabelecer os melhores parâmetros para serem utilizados nas meta-heurísticas desenvolvidas (JÚNIOR et al., 2004). Para ajustar os parâmetros do método proposto, utilizou-se o pacote *Iterated Racing for Automatic Algorithm Configuration* (IRACE) (LÓPEZ-IBÁÑEZ et al., 2016), que permite identificar, de forma progressiva e estatisticamente guiada, as combinações de parâmetros que levam ao melhor desempenho do algoritmo em um conjunto representativo de instâncias. O IRACE utiliza o mecanismo de *racing*, no qual diferentes configurações competem entre si e aquelas com desempenho estatisticamente inferior são descartadas progressivamente, permitindo uma exploração eficiente do espaço de parâmetros.

A etapa de parametrização tem papel fundamental na obtenção de soluções de alta qualidade e a utilização do IRACE ajuda a identificar a parametrização de variáveis

e definir a melhor combinação que alinhe o melhor custo-benefício entre esforço computacional e tempo de execução. Foram executados testes extensivos com uma grande diversidade de intervalos dos parâmetros distintos e algumas instâncias representativas do conjunto. Esses parâmetros, juntamente com as instâncias representativas do problema, compõem o conjunto de entrada do IRACE. Dados os testes realizados para ajustar os melhores parâmetros a serem utilizados, o IRACE sugeriu como parâmetros ideais para executar os experimentos computacionais:

- Quantidade máxima de iterações do RVND: 400
- Decaimento de temperatura do SA (α): 0.4
- Quantidade de iterações em uma mesma temperatura do SA: 600
- Temperatura final do SA: 10
- Quantidade de iterações do GRASP: 10
- Porcentagem de perturbação dos elementos: 0.4
- Porcentagem de perturbação dos *clusters*: 0.4
- Algoritmo Construtivo Guloso Randomizado: 10 vezes para cada parâmetro $\alpha = 0{,}1$, $\alpha = 0{,}2$, $\alpha = 0{,}3$, sendo 100 iterações da Heurística Construtiva;
- Algoritmo Construtivo Guloso Randomizado Reativo: 10 vezes para cada instância, sendo 1000 iterações da Heurística Construtiva e a cada 50 iterações, as probabilidades são recalculadas.

Ao final da execução do IRACE foram identificadas as variáveis com maior impacto no comportamento do método SA-RGRASP-RVND, retornando o conjunto de parâmetros mais promissor, que então foi utilizado na fase final dos experimentos. Esse processo sistemático de parametrização permite melhorar o desempenho do algoritmo, assegurar reprodutibilidade e fundamentação na escolha dos parâmetros.

6.4 Análise dos resultados obtidos

Para cada instância de teste, o algoritmo SA-RGRASP-RVND foi executado 10 vezes para realizar a seleção dos melhores resultados e calcular as médias dos e tempo de execução das soluções. Nas tabelas de comparações deste capítulo, os valores destacados em negrito indicam os melhores resultados quanto ao custo da solução. Optou-se por apresentar a comparação da média e do tempo dos resultados para 5 instâncias de cada conjunto que são representativas ao comparar os resultados das outras instâncias de cada conjunto.

A Tabela 6.1 apresenta as melhores comparações entre as diferentes combinações das abordagens propostas para tratar o problema. Comparando os algoritmos construídos nesse trabalho, a combinação do SA, RGRASP e RVND é significativamente superior a todas as outras em todas as instâncias testadas e podemos identificar a mehoria que cada método meta-heurística traz na construção do resultado final da abordagem híbrida.

Tabela 6.1: Comparação das diferentes abordagens

Instância	SA	RGRASP-RVND	SA-RVND	**SA-RGRASP-RVND**
Sparse82_01	1164.06	1280.38	1239.80	**1343.08**
Sparse82_02	1145.77	1209.39	1257.30	**1306.64**
Sparse82_03	1157.73	1262.39	1243.98	**1353.94**
Sparse82_04	1127.13	1196.87	1201.51	**1291.22**
Sparse82_05	1216.10	1174.40	1304.02	**1352.35**
RanReal240_01	168789.48	218270.09	214159.67	**224968.01**
RanReal240_02	156239.54	193872.50	199701.43	**204624.36**
RanReal240_03	152533.29	170139.34	186315.37	**199059.56**
RanReal240_04	165881.31	209216.79	195997.35	**225627.16**
RanReal240_05	152252.34	190255.18	190271.18	**195564.48**
RanReal480_01	378306.50	514388.34	543450.75	**555489.92**
RanReal480_02	377185.34	480280.37	486389.75	**511280.50**
RanReal480_03	369109.68	473195.12	463515.68	**497725.86**
RanReal480_04	380258.15	504997.56	505911.87	**522572.81**
RanReal480_05	373964.62	454005.96	467338.46	**484084.66**

Nas figuras 6.1, 6.2 e 6.3 é possível ver com mais clareza a dimensão dos resultados em comparação. Como dito, o SA-RGRASP-RVND ganha de todos, porém na construção desses algoritmos pode-se perceber em qual área cada um se destaca com base no que foi estudado na literaturae comprovado nas prática deste trabalho.

Nas figuras citadas percebe-se que na maioria das instâncias o SA-RVND é o segundo vencedor na comparação dos resultados. Isso pode ser explicado dado a natureza

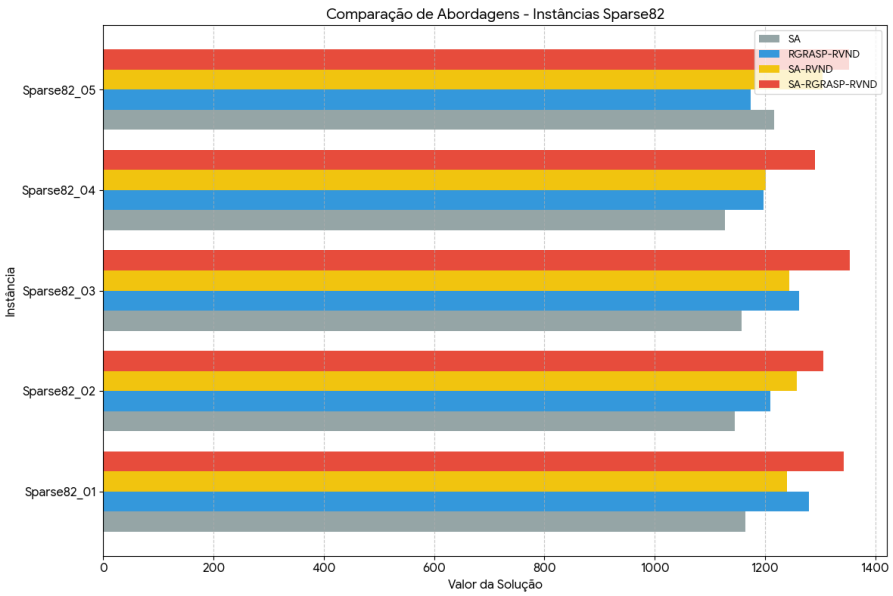

Figura 6.1: Comparação da média de resultados dos algoritmos propostos em 5 instâncias do conjunto de 82 vértices

do SA de continuar realizando sucessivas pertubações e em seguida a execução do RVND, sempre em uma mesma solução. Sendo que enquanto ele itera até chegar na temperatura mínima permitida, ele probabilisticamente aceita novas soluções que podem ser piores, fazendo com que os espaço de busca seja movido muitas vezes para um outro espaço com um ótimo local que seja melhor. Portanto, a partir do momento que ele aceitou piorar a solução, podendo estar em outro local de busca diferente - porém de certa forma já bastante aprimorado - ele pode através do RVND executar as buscas locais sucessivas e assim encontrar uma solução em uma melhor solução em outro ótimo local.

Já se tratando do RGRASP-RVND, seu benefício é focado na construção de diversas soluções aprimoradas, pois a característica principal do RGRASP é criar uma nova solução com construtivo randomizado e reativo a cada iteração, dessa forma ele sempre busca criar as soluções reativamente mais promissoras com os melhores valores de α. Somado a isso, a cada iteração onde é gerada uma nova solução, esta passa por um processo de pertubação quando chega um número determinado de iterações sem melhora e pela execução do RVND que faz a busca local até que não consiga melhorar a solução corrente. Sendo então um algoritmo promissor que sai de ótimos locais e é refinado, percebe-se que ele entrega boas soluções, próximas da solução do SA-RVND, com algumas instâncias até ultrapassando a solução do SA-RVND, porém sem a característica do SA

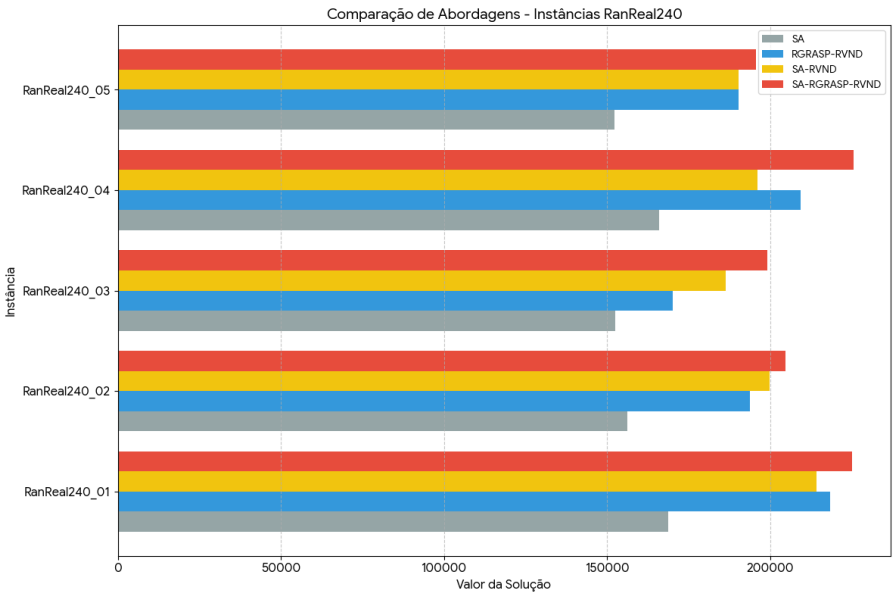

Figura 6.2: Comparação da média de resultados dos algoritmos propostos em 5 instâncias do conjunto de 240 vértices

de aceitar soluções piores ele não consegue sair de alguns ótimos locais e fica atrás dele.

Por fim, é possível perceber que o SA só funciona bem juntamente com outros métodos, já que sozinho ele não realiza buscas locais, sendo este o benefício do RVND, e não consegue trazer diferentes soluções do espaço de busca ao não usar o RGRASP. Portanto, utilizando apenas um algoritmo guloso randomizado e a própria perturbação, ele não tem força para conseguir refinar a solução e testar diversas soluções iniciais que podem ser promissoras.

Assim, apesar da combinação do RGRASP utilizando apenas RVND apresentar resultados competitivos, a meta-heurística *Simulated Annealing* quando combinada apenas com RVND conseguiu ter melhores resultados para a maioria das instâncias, indicando que o SA ajuda a diversificar as soluções com a aceitação de soluções piores. Portanto, ao combinar essas 3 meta-heurísticas, é possível ao mesmo tempo iniciar com uma boa solução usando RGRASP, diversificar bem as soluções pelo uso do SA e fazer um bom trabalho de busca local com RVND.

A Tabela 6.2 mostra a comparação com a literatura, considerando a melhor solução obtida para cada instância dentre as 10 execuções realizadas. Pode-se notar que o método proposto consegue ter um resultado melhor que o da literatura recente em duas instâncias do tipo Sparse, com 82 vértices e nas demais desse conjunto é possível alcan-

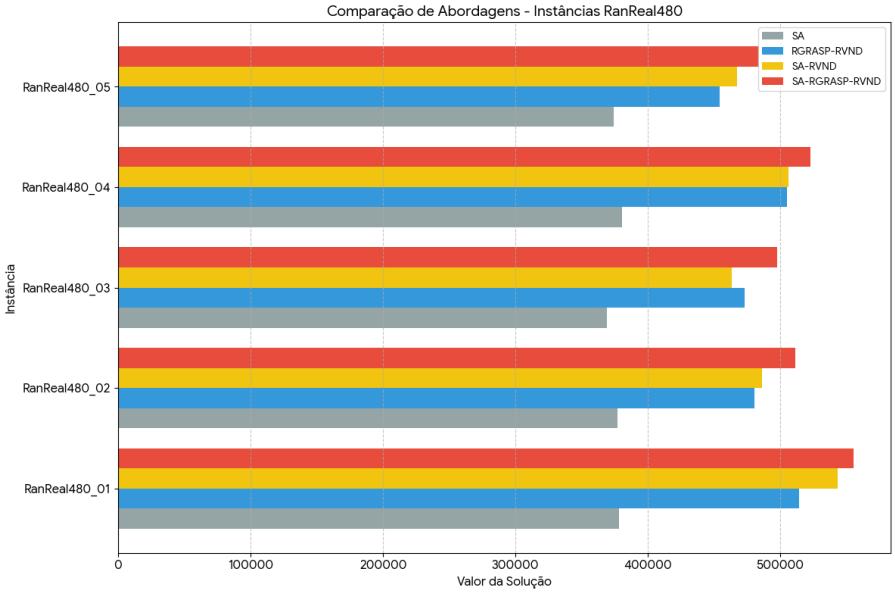

Figura 6.3: Comparação da média de resultados dos algoritmos propostos em 5 instâncias do conjunto de 480 vértices

çar a melhor solução. Para os demais conjuntos de instâncias, RanReal de 240 e 480 vértices, a abordagem da literatura apresenta resultados melhores. Vale destacar que o SA-RGRASP-RVND apresenta resultados muito próximos aos da melhor solução da literatura, em algumas instâncias com diferença apenas na segunda casa decimal, mostrando o potencial do método proposto.

Como forma de analisar a robustez da abordagem proposta em relação à literatura, a tabela 6.3 apresenta a comparação dos resultados quanto à média da qualidade das soluções obtidas, considerando todas as 10 execuções. Além disso, a tabela mostra o tempo de processamento, em segundos, demandado por cada abordagem.

Podemos ver que o resultado e o tempo médio de execução do SA-RGRASP-RVND são competitivos em todas as instâncias. Ao comparar o tempo médio de execução, observa-se nas instâncias de 82 e 240 vértices tempos bem próximos aos da literatura recente. Pode-se conferir que há um aumento no tempo de execução à medida que a quantidade de vértices cresce. Essa discrepância no tempo de execução, principalmente para instâncias com maior quantidade de vértices, ocorreu a partir do resultado mais recente da literatura atualizado como referência neste trabalho. Ao acompanhar a literatura é possível ver que para resultados anteriores ao atual, a abordagem proposta também ganha no tempo, possuindo uma execução pelo menos 4 vezes mais rápida.

Tabela 6.2: Comparação com a melhor solução da literatura

Instancia	Best Literatura	Best SA-RGRASP-RVND	*Diferença (%)*
Sparse82_01	1342.17	**1343.08**	0.07
Sparse82_02	**1306.64**	**1306.64**	0.00
Sparse82_03	**1353.94**	**1353.94**	0.00
Sparse82_04	**1291.22**	**1291.22**	0.00
Sparse82_05	**1352.35**	**1352.35**	0.00
Sparse82_06	1354.61	**1354.62**	0.01
Sparse82_07	**1266.94**	**1266.94**	0.00
Sparse82_08	**1393.02**	**1393.02**	0.00
Sparse82_09	**1294.12**	**1294.12**	0.00
Sparse82_10	**1356.98**	**1356.98**	0.00
RanReal240_01	**225003.70**	224768.07	-0.11
RanReal240_02	**204624.36**	204456.67	-0.08
RanReal240_03	**199079.37**	198971.21	-0.05
RanReal240_04	**225683.17**	224727.21	-0.42
RanReal240_05	**195564.48**	195440.94	-0.06
RanReal240_06	**216747.32**	216721.07	-0.01
RanReal240_07	**209305.70**	209288.34	-0.01
RanReal240_08	**205246.82**	205140.91	-0.05
RanReal240_09	**209186.90**	209015.07	-0.08
RanReal240_10	**193062.60**	192992.24	-0.04
RanReal240_11	**204722.75**	204625.61	-0.05
RanReal240_12	**201117.11**	200888.69	-0.11
RanReal240_13	**202345.48**	202035.87	-0.15
RanReal240_14	**228971.03**	228703.47	-0.12
RanReal240_15	**191263.28**	190835.77	-0.22
RanReal240_16	**204081.46**	203879.95	-0.10
RanReal240_17	**195561.36**	194978.68	-0.30
RanReal240_18	**195167.14**	194874.72	-0.15
RanReal240_19	**199307.33**	199060.84	-0.12
RanReal240_20	**212323.22**	211998.91	-0.15
RanReal480_01	**556639.68**	554743.59	-0.34
RanReal480_02	**511666.95**	509753.95	-0.37
RanReal480_03	**497846.57**	496546.98	-0.26
RanReal480_04	**523588.42**	521849.75	-0.33
RanReal480_05	**485122.31**	483128.12	-0.41
RanReal480_06	**534982.12**	533885.20	-0.21
RanReal480_07	**546892.45**	545801.83	-0.20
RanReal480_08	**533411.78**	532362.44	-0.20
RanReal480_09	**556112.36**	555996.69	-0.02
RanReal480_10	**520104.92**	519327.55	-0.15
RanReal480_11	**523450.04**	523228.93	-0.04
RanReal480_12	**501596.63**	501074.90	-0.10
RanReal480_13	**534638.19**	533759.92	-0.16
RanReal480_14	**513777.84**	513171.43	-0.12
RanReal480_15	**516941.11**	516133.40	-0.16
RanReal480_16	**549371.23**	549235.94	-0.03
RanReal480_17	**537483.76**	537289.90	-0.04
RanReal480_18	**525813.39**	524926.00	-0.17
RanReal480_19	**522158.86**	521548.87	-0.12
RanReal480_20	**518288.03**	518080.79	-0.04

Tabela 6.3: Média das soluções em comparação com a literatura

	Média da qualidade e do tempo de processamento				
Instancia	Aver_Lit	T_Lit (s)	Aver_SA-RGRASP-RVND	T_SA-VND (s)	*Dif. (%)*
Sparse82_01	**1342.17**	**13**	**1342.17**	41	0.00
Sparse82_02	**1306.64**	**20**	**1306.64**	36	0.00
Sparse82_03	**1353.94**	**4**	**1353.94**	46	0.00
Sparse82_04	**1291.22**	**27**	**1291.22**	72	0.00
Sparse82_05	**1352.35**	**4**	**1352.35**	42	0.00
RanReal240_01	**224897.01**	**127**	224086.06	149	-0.36
RanReal240_02	**204515.06**	**111**	204041.70	165	-0.23
RanReal240_03	**198915.84**	**135**	198263.04	139	-0.33
RanReal240_04	**225346.54**	**124**	224673.38	147	-0.30
RanReal240_05	**195469.00**	**135**	195009.36	142	-0.24
RanReal480_01	**555338.06**	**340**	554171.29	1840.00	-0.21
RanReal480_02	**510924.27**	**355**	509113.23	2174.00	-0.35
RanReal480_03	**497109.59**	**352**	496301.99	1879.00	-0.16
RanReal480_04	**521999.31**	**399**	521397.65	2461.00	-0.12
RanReal480_05	**484092.59**	**335**	483007.46	1608.00	-0.22

7 Conclusão

Este trabalho propõe abordagens para a solução do Problema da Clusterização Capacitada (PCC). Foi utilizada uma abordagem híbrida contendo SA, RGRASP e RVND, que em conjunto trouxeram bons resultados. Foi usado um método de perturbação que faz uma desconstrução parcial da solução e posteriormente a reconstrução e o uso de 3 métodos de vizinhança no RVND para realizar buscas locais a fim de gerar campos de busca diferentes, otimizando sempre o valor objetivo do problema e obedecendo as restrições do PCC.

A partir dos testes foi visto que os resultados são consistentes, ou seja, possuem uma média próxima da literatura e não variam muito. Isso se deve ao refinamento com RVND, onde as soluções convergem para um ótimo local e em conjunto o SA ajuda a não estacionar em apenas um ótimo local, que através da perturbação e do critério de aceitação de piores soluções faz com que uma nova solução gerada mude de espaço de busca. Adicionalmente, ao usar o RGRASP em conjunto, entrega-se uma vantagem para o SA que inicia com a primeira solução já em um ótimo local promissor e assim inicia o processo do SA já minimamente refinado.

Algumas decisões importantes tomadas foram feitas ao observar que a solução convergia e ficava presa em um ótimo local em poucas repetições. Tal comportamento acontecia mais frequentemente dado à utilização do VND em sua versão não randomizada, associado ao fato da estratégia de busca ser utilizada após a perturbação, onde os parâmetros até então não estavam sendo suficientes para sair da zona de atração de um ótimo local. Com o objetivo de obter melhorias mais significativas nas soluções e mitigar as limitações observadas, adotou-se uma abordagem randomizada com RVND, além de ajustar os parâmetros para intensificar o nível de perturbação sobre os vértices e *clusters*.

Através das comparações entre os algoritmos propostos, notou-se que o SA puro apenas com solução inicial e perturbações não consegue obter soluções boas. É necessário ter outra meta-heurística que dê suporte ao SA fazendo sucessivas buscas locais com o RVND, sendo possível gerar melhores soluções pelo refinamento do espaço de busca. Porém, mesmo que o RVND consiga soluções muito boas, ele também não consegue me-

lhorar mais o resultado sozinho, pois ao encontrar ótimos locais ele não consegue sair por si só. Portanto, o SA se mostra um grande aliado do RVND ao sair dos ótimos locais e diversificar as próximas soluções.

Com os resultados experimentais evidenciou que o fator responsável pela melhoria da qualidade das soluções dentro da meta-heurística *Simulated Annealing* foi o mecanismo de perturbação e posterior refinamento com RVND. Observou-se que ao aplicar o RVND de forma criteriosa após perturbações significativas, desempenha papel fundamental na condução da busca a novos ótimos locais. Já a aplicação excessiva do RVND logo após a fase construtiva do RGRASP mostrou-se pouco eficaz, que apesar de gerar várias soluções e melhorar a solução gerada pelo RGRASP, muitas iterações não trazem maiores ganhos. Isso indica que investir demasiadamente na melhoria da solução inicial pode levar à estagnação em ótimos locais, reduzindo a capacidade exploratória do método.

Verificou-se que mesmo após mil iterações do RGRASP, a melhor solução obtida não superou a encontrada pelo SA-GRASP, reforçando a importância de um equilíbrio entre qualidade inicial e potencial de diversificação. Em contrapartida, o uso do RGRASP para geração das soluções iniciais do SA ajudou muito o processo de buscar uma solução de melhor qualidade, tendo resultados melhores no método híbrido em comparação com apenas usar um método guloso, semi-guloso ou randomizado como solução inicial.

Além disso, os experimentos demonstraram que estratégias de perturbação mais agressivas no SA, como a remoção de até 50% dos elementos da solução e perturbar pelo menos 30% a 60% dos *clusters*, foram decisivas para alcançar soluções de qualidade equivalente às melhores da literatura. A redução da porcentagem de *clusters* perturbados de 80% para 30% mostrou-se eficaz para manter o benefício máximo com menor tempo de execução, evidenciando a sensibilidade do método à parametrização das variáveis. Foi visto que apesar de aumentar o tempo computacional, uma maior porcentagem de perturbação gera soluções que são bem próximas à literatura. Então com isso, a partir da parametrização pelo *IRACE*, optou-se por utilizar a combinação de parâmetros com melhor custo beneficio, assim utilizamos os parâmetros que resultavam nas melhores soluções, porém apenas aqueles com o menor tempo de execução do método.

Por fim, foi constatado que a combinação entre perturbação e RVND revelou-se

particularmente eficiente, caracterizando um comportamento próximo ao de uma busca local iterada (ILS) embutida no SA, na qual a diversificação é promovida pela perturbação e a intensificação pela busca local randomizada. Essa configuração permitiu que 90% das execuções de testes nas instâncias de 82 vértices atingisse a melhor solução conhecida em menos de 1 minuto, com algumas soluções geradas em até 3 segundos. Assim, os resultados reforçam a adequação do uso do RGRASP para geração da solução inicial, do RVND como mecanismo de refinamento e do SA como abordagem de perturbação e aceitação de soluções piores para sair de ótimos locais é particularmente interessante, destacando-se portanto, a importância na decisão do uso híbrido das meta-heurísticas e na parametrização de variáveis para se atingir soluções melhores em tempo mais rápido.

Durante o desenvolvimento do método, foram identificadas limitações importantes, principalmente relacionadas ao aumento do tempo computacional quando parâmetros mais agressivos eram adotados, efeito que se intensifica conforme o tamanho das instâncias cresce. Observou-se elevada sensibilidade no processo de ajuste dos parâmetros, uma vez que pequenas variações impactam significativamente tanto a qualidade das soluções quanto o tempo de execução, dificultando a definição de configurações estáveis, principalmente pela dificuldade em encontrar parâmetros que funcionem de forma satisfatória em todos os conjuntos de instâncias. Em particular, o controle da taxa de aceitação de soluções piores no SA mostrou-se desafiador. Além disso, a definição de uma combinação híbrida eficiente entre SA, RGRASP e RVND evidenciou a complexidade inerente ao equilíbrio entre diversificação e intensificação da busca.

De forma mais específica, constatou-se dificuldade em estabelecer uma parametrização adequada, sugerindo que configurações fixas tendem a apresentar desempenho diferentes quando usadas em vários conjuntos de instâncias com características distintas. Nesse contexto, mostra-se promissora a definição de parâmetros baseada em características estruturais das instâncias, como o número de vértices, o número de *clusters* e a soma dos benefícios associados ao grafo completo. Observou-se também que, para instâncias de grande porte, especialmente aquelas com 480 vértices, o custo combinatório cresce de forma acentuada, impactando diretamente a escalabilidade do método. Embora o desempenho seja competitivo para instâncias pequenas e médias, esse comportamento

evidencia a necessidade de refatoração do código com foco em otimização computacional, bem como de estratégias que permitam aceitar mais soluções piores nas fases iniciais do SA, por meio de funções de resfriamento da temperatura mais suaves, principalmente no início do método.

Por fim, constatou-se que o método é extremamente sensível à escolha dos parâmetros, sendo que pequenas variações podem impactar significativamente os valores finais, a média das execuções e, principalmente, o tempo de processamento. Como evidência, após a calibração automática dos parâmetros utilizando o IRACE, foi possível manter — e em alguns casos melhorar — a qualidade das soluções, ao mesmo tempo em que o tempo médio de execução para instâncias com 240 vértices foi reduzido de aproximadamente 2100 segundos para 149 segundos, tornando o método competitivo em termos de tempo em comparação com as melhores abordagens da literatura. Dessa forma, estudos futuros devem continuar explorando estratégias de ajuste de parâmetros e otimizações estruturais, visando melhorar os resultados sem comprometer excessivamente o tempo de execução, ou, quando viável, permitir um aumento controlado do tempo computacional para alcançar soluções ainda mais próximas do estado da arte.

Como trabalhos futuros, pretende-se investigar estratégias mais robustas de parametrização automática, capazes de adaptar dinamicamente os parâmetros do algoritmo às características das instâncias, como o número de vértices, de *clusters* e soma de benefícios do grafo completo, incluindo também o uso de técnicas de aprendizado de máquina para apoiar a seleção dinâmica de parâmetros e decisões de busca. Além disso, planeja-se aprimorar a etapa de perturbação principalmente quando há estagnação, tornando-a semi-gulosa, bem como reduzir o tempo de execução por meio da otimização de recálculos redundantes e da incorporação de estruturas de memória para armazenamento de custos previamente computados. Outras direções incluem a avaliação de funções de aceitação e resfriamento alternativas no Simulated Annealing, o uso do algoritmo construtivo em cenários de estagnação, técnicas de paralelização e poda, além da aplicação da abordagem proposta em instâncias de maior escala e a outros problemas de otimização combinatória, visando ampliar a validade e a generalização dos resultados obtidos.

Bibliografia

ALTUWAIM, B. A metaheuristic to solve the capacitated clustering problem. *Journal of Al-Azhar University Engineering Sector*, Al-Azhar University; Faculty of Engineering, v. 18, n. 69, p. 884–899, 2023. ISSN 1687-8418. Disponível em: <https://jaes.journals.ekb.eg/article_324294.html>.

BARD, J. F.; JARRAH, A. I. Integrating commercial and residential pickup and delivery networks: A case study. *Omega*, Elsevier, v. 41, n. 4, p. 706–720, 2013.

BRIMBERG, J.; MLADENOVIĆ, N.; TODOSIJEVIĆ, R.; UROŠEVIĆ, D. Variable neighborhood descent for the capacitated clustering problem. In: SPRINGER. *Discrete Optimization and Operations Research: 9th International Conference, DOOR 2016, Vladivostok, Russia, September 19-23, 2016, Proceedings 9*. [S.l.], 2016. p. 336–349.

BRIMBERG, J.; MLADENOVIĆ, N.; TODOSIJEVIĆ, R.; UROŠEVIĆ, D. Solving the capacitated clustering problem with variable neighborhood search. *Annals of Operations Research*, Springer, v. 272, p. 289–321, 2019.

BURKE, E. K.; GENDREAU, M.; HYDE, M.; KENDALL, G.; OCHOA, G.; ÖZCAN, E.; QU, R. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, Springer, v. 64, p. 1695–1724, 2013.

DENG, Y.; BARD, J. F. A reactive grasp with path relinking for capacitated clustering. *Journal of Heuristics*, v. 17, n. 2, 2011.

FALKNER, J. K.; SCHMIDT-THIEME, L. Neural capacitated clustering. *arXiv preprint arXiv:2302.05134*, 2023.

GALLEGO, M.; LAGUNA, M.; MARTI, R.; DUARTE, A. Tabu search with strategic oscillation for the maximally diverse grouping problem. *The Journal of the Operational Research Society*, v. 64, 05 2013.

HUSSAIN, K.; SALLEH, M. N. M.; CHENG, S.; SHI, Y. Metaheuristic research: a comprehensive survey. *Artificial intelligence review*, Springer, v. 52, p. 2191–2233, 2019.

JÚNIOR, A. D.; SILVA, R. S.; MUNDIM, K. C.; DARDENNE, L. E. Performance and parameterization of the algorithm simplified generalized simulated annealing. *Genetics and Molecular Biology*, SciELO Brasil, v. 27, p. 616–622, 2004.

LAI, X.; HAO, J.-K. Iterated variable neighborhood search for the capacitated clustering problem. *Engineering Applications of Artificial Intelligence*, v. 56, p. 102–120, 2016. ISSN 0952-1976. Disponível em: <https://www.sciencedirect.com/science/article/pii/S095219761630135X>.

LAI, X.; HAO, J.-K.; FU, Z.-H.; YUE, D. Neighborhood decomposition-driven variable neighborhood search for capacitated clustering. *Computers & Operations Research*, Elsevier, v. 134, p. 105362, 2021.

LIU, Y.; GUO, P.; ZENG, Y. Ha-ccp: A hybrid algorithm for solving capacitated clustering problem. *Computational Intelligence and Neuroscience*, Hindawi, v. 2022, p. 1–24, 01 2022.

LIU, Y.; GUO, P.; ZENG, Y. Meaccp: A membrane evolutionary algorithm for capacitated clustering problem. *Information Sciences*, Elsevier, v. 591, p. 319–343, 2022. ISSN 0020-0255.

LÓPEZ-IBÁÑEZ, M.; DUBOIS-LACOSTE, J.; CÁCERES, L. P.; BIRATTARI, M.; STÜTZLE, T. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, Elsevier, v. 3, p. 43–58, 2016.

LUXBURG, U. V.; WILLIAMSON, R. C.; GUYON, I. Clustering: Science or art? In: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. *Proceedings of ICML workshop on unsupervised and transfer learning*. [S.l.], 2012. p. 65–79.

MARTÍNEZ-GAVARA, A.; CAMPOS, V.; GALLEGO, M.; LAGUNA, M.; MARTÍ, R. Tabu search and grasp for the capacitated clustering problem. *Computational Optimization and Applications*, Springer, v. 62, p. 589–607, 2015.

MARTINEZ-GAVARA, A.; LANDA-SILVA, D.; CAMPOS, V.; MARTI, R. Randomized heuristics for the capacitated clustering problem. *Information Sciences*, Elsevier, v. 417, p. 154–168, 2017.

MAXIMO, V. R.; NASCIMENTO, M. C. A hybrid adaptive iterated local search with diversification control to the capacitated vehicle routing problem. *European Journal of Operational Research*, v. 294, n. 3, p. 1108–1119, 2021. ISSN 0377-2217. Disponível em: <https://www.sciencedirect.com/science/article/pii/S037722172100117X>.

MORÁN-MIRABAL, L.; GONZÁLEZ-VELARDE, J.; RESENDE, M. G.; SILVA, R. M. Randomized heuristics for handover minimization in mobility networks. *Journal of Heuristics*, Springer, v. 19, p. 845–880, 2013.

MULVEY, J. M.; BECK, M. P. Solving capacitated clustering problems. *European Journal of Operational Research*, Elsevier, v. 18, n. 3, p. 339–348, 1984.

MURITIBA, A. E. F.; GOMES, M. J. N.; SOUZA, M. F. de; ORIA, H. L. G. Path-relinking with tabu search for the capacitated centered clustering problem. *Expert Systems with Applications*, Elsevier, v. 198, p. 116766, 2022.

NEGREIROS, M. J.; MACULAN, N.; PALHANO, A. W.; MURITIBA, A. E.; BATISTA, P. L. Capacitated clustering models to real life applications. *artificial intelligence (AI)*, v. 3, p. 8, 2022.

OROZCO-ROSAS, U.; MONTIEL, O.; SEPúLVEDA, R. Mobile robot path planning using membrane evolutionary artificial potential field. *Applied Soft Computing*, v. 77, p. 236–251, 2019. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494619300420>.

OSMAN, I. H.; CHRISTOFIDES, N. Capacitated clustering problems by hybrid simulated annealing and tabu search. *International Transactions in Operational Research*, Elsevier, v. 1, n. 3, p. 317–336, 1994.

RESENDE, M. G.; RIBEIRO, C. C. *Optimization by GRASP*. [S.l.]: Springer, 2016.

RUAN, D.; DA, R. *Computational Intelligence in Complex Decision Making Systems*. [S.l.]: Springer, 2010.

SOUZA, M. J. F. Inteligência computacional para otimização. *Notas de aula, Departamento de Computaçao, Universidade Federal de Ouro Preto, disponıvel em http://www. decom. ufop. br/prof/marcone/InteligenciaComputacional/InteligenciaComputacional. pdf*, v. 6, 2008.

VOSS, S.; MARTELLO, S.; OSMAN, I. H.; ROUCAIROL, C. Meta-heuristics: Advances and trends in local search paradigms for optimization. Springer Science & Business Media, 2012.

ZHOU, Q.; BENLIC, U.; WU, Q.; HAO, J.-K. Heuristic search to the capacitated clustering problem. *European Journal of Operational Research*, Elsevier, v. 273, n. 2, p. 464–487, 2019.