Felipe Soares, Iury Alcantara, Kaiky Gomes, Mateus Ramos

Projeto e análise de algoritmos

Procedimento de Busca Aleatória Gulosa Adaptativa

Instituto Federal Fluminense

Campus Campos Centro

Campos dos Goytacazes, Brasil

09/02/2024

Sumário

1. Procedimento de Busca Aleatória Gulosa Adaptativa

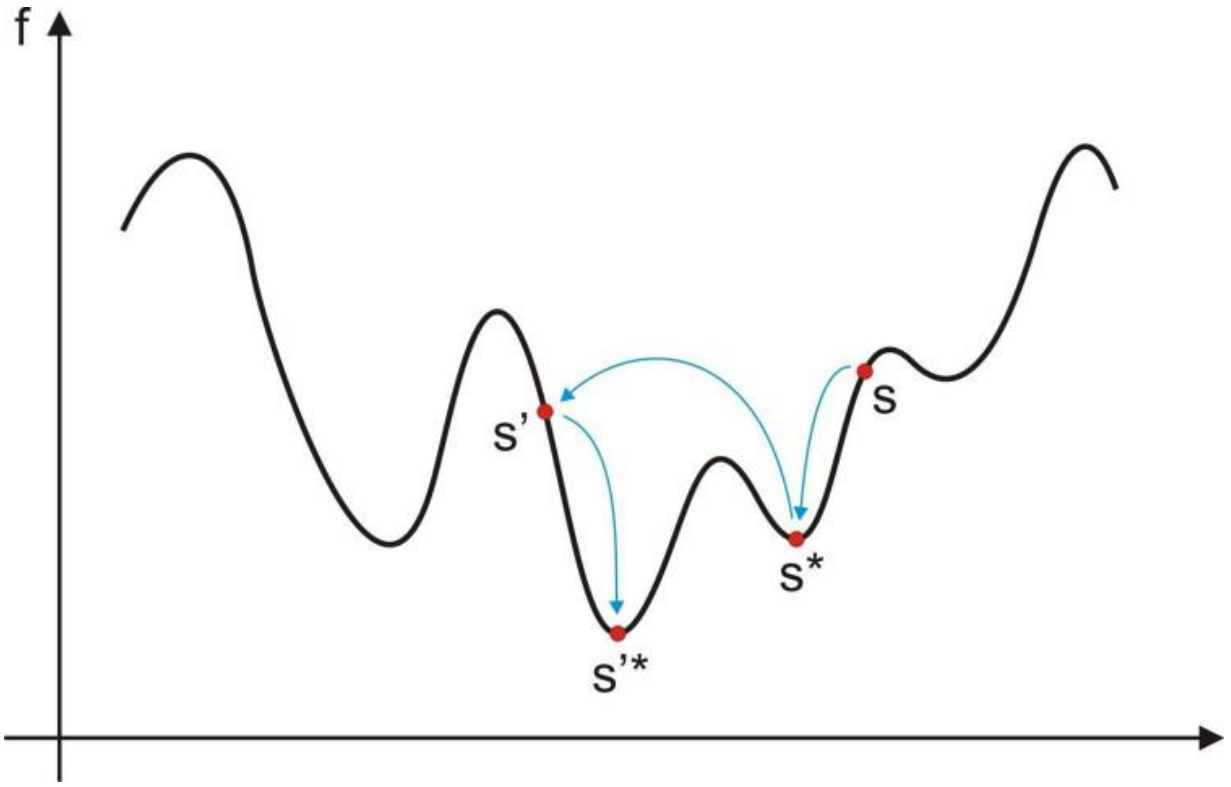
O Greedy Randomized Adaptive Search Procedure (GRASP) é uma poderosa abordagem de otimização que combina estratégias gulosas e aleatórias para enfrentar problemas complexos de otimização combinatória. Projetado para lidar com situações onde encontrar a solução ótima é desafiador devido à natureza computacionalmente difícil do problema, o GRASP se destaca ao equilibrar a exploração do espaço de busca. Sua flexibilidade e adaptabilidade tornam-no uma ferramenta valiosa em diversos domínios, proporcionando uma solução eficaz para problemas práticos que variam desde alocação de recursos até design de redes.

1.1 Características

O GRASP é uma técnica de otimização heurística que combina uma abordagem gananciosa com aleatoriedade para construir soluções iterativamente. Utiliza heurísticas de construção para formar soluções parciais, introduzindo aleatoriedade para explorar amplamente o espaço de solução.

Após a construção inicial, emprega busca local para aprimorar a solução. Sua adaptabilidade e aplicação abrangente em problemas complexos o tornam eficaz em situações onde soluções ótimas são desafiadoras de serem encontradas.

Figura 1 - A Busca Local feita com a solução inical [S] gerada pelo algoritmo guloso, buscando uma solução a partir de um ótimo sub-ótimo mínimo local.



BUSCA LOCAL ITERADA - Francisco A. M. Gomes

1.2 Componentes do GRASP

• **Algoritmo Guloso** - Produz uma solução inicial, com base em um algoritmo guloso usado para a geração de uma solução inicial sub-ótima;

• **Busca Local** - Retorna uma solução melhorada em relação a inicial com base na sua vizinhança;

• **Perturbação** - Modifica a solução corrente guiando a uma solução intermediária, evitando que o algoritmo prenda em ótimos sub-ótimos locais;

• **Critério de Aceitação** - Decide de qual solução a próxima perturbação será aplicada;

1.2.1 Solução Inicial

O algoritmo GRASP busca resolver problemas de otimização combinatória por meio de uma abordagem iterativa. Inicia-se com uma solução vazia e emprega um algoritmo guloso na etapa de construção inicial. Essa fase gulosa utiliza uma heurística para selecionar, de maneira sequencial, elementos a serem adicionados à solução parcial. A escolha é orientada por critérios como menor custo, maior benefício ou outra métrica relevante ao problema. Após a construção inicial, o algoritmo aplica uma busca local para aprimorar a qualidade da solução, repetindo esse processo ao longo de várias iterações. A aleatoriedade introduzida na escolha dos elementos contribui para evitar soluções subótimas, proporcionando uma abordagem adaptativa e eficaz para uma variedade de problemas de otimização combinatória.

Na geração da solução inicial, o algoritmo guloso parte de uma solução vazia, selecionando elementos com base em uma heurística específica. Essa heurística orienta a adição de elementos à solução parcial, utilizando critérios como custo mínimo ou benefício máximo. A adição gulosa é repetida de acordo com um critério de parada predefinido, podendo ser o número de elementos adicionados ou outra condição relevante ao problema. O processo de construção inicial é iterativo, permitindo a geração de diferentes soluções iniciais em cada iteração, proporcionando ao algoritmo GRASP flexibilidade para explorar diversas soluções potenciais. Esse método, combinando elementos gananciosos com busca local, visa encontrar soluções de alta qualidade em problemas complexos de otimização combinatória.

1.2.2 Busca Local

- O GRASP é ocasionalmente tratado como uma caixa preta, permitindo a utilização de qualquer heurística existente para o problema em questão.
- A única exigência é que o método escolhido seja capaz de melhorar uma solução dada.
- Metaheurísticas como a busca tabu e o recozimento simulado também podem ser aplicadas na busca local do GRASP.
- Geralmente, um algoritmo de busca mais eficiente resulta em melhores desempenhos.
- No entanto, em alguns casos, um algoritmo simples e de baixo custo pode ser preferível, dependendo da estratégia de perturbação adotada.

1.2.3 Perturbação

Como mencionado anteriormente, a perturbação na busca local do GRASP não pode ser excessivamente grande a ponto de transformá-la em uma busca local com recomeços, mas deve ser suficientemente forte para evitar que a busca local a reverta. Portanto, é crucial que a busca local e a perturbação sejam definidas de forma coordenada.

Isso significa que a intensidade da perturbação deve ser cuidadosamente ajustada para equilibrar a exploração de diferentes soluções com a intensificação em torno de ótimos locais. Uma perturbação muito fraca pode não permitir que a busca escape de mínimos locais, enquanto uma perturbação muito forte pode levar a um comportamento semelhante ao de uma busca local com recomeços, reduzindo a eficácia do GRASP.

Dessa forma, a escolha adequada da perturbação é essencial para garantir que o GRASP alcance soluções de alta qualidade de maneira eficiente,

combinando exploração e intensificação de forma coordenada.

1.2.4 Critério de Aceitação

Usualmente, é exigido que a solução melhore. Nesse caso, a solução s' só é aceita se $f(s') < f(s)$, onde f representa a função de avaliação.

No entanto, também é possível aceitar uma solução s' que seja pior do que s com uma certa probabilidade baixa de sucesso. Outra abordagem é utilizar uma regra de aceitação similar à do recozimento simulado, definindo a probabilidade de aceitação em função da temperatura.

Em muitos casos, essas estratégias de diversificação são aplicadas apenas quando a regra usual de aceitação não obtém sucesso após muitas iterações. Isso permite que a busca local integrada explore soluções potencialmente piores em busca de regiões mais promissoras do espaço de soluções, aumentando a diversificação e evitando ficar preso em mínimos locais subótimos.

2 Aplicação do GRASP no Problema do Caixeiro Viajante com coleta de prêmios

2.1 Introdução

O Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP) é uma extensão do problema clássico de otimização combinatória que acrescenta a dimensão da coleta de prêmios durante as visitas às cidades. Nesse problema, o viajante busca a rota mais curta para visitar um conjunto de cidades, passando por cada uma delas exatamente uma vez, coletando prêmios associados a cada

cidade e retornando à cidade de origem. O objetivo é minimizar o custo total da rota, considerando as distâncias percorridas e os prêmios coletados.

Assim como o PCV, o PCVCP é um problema NP-difícil, o que significa que encontrar a solução ótima para instâncias grandes é uma tarefa computacionalmente desafiadora. O número de possíveis rotas cresce exponencialmente com o número de cidades, aumentando a complexidade da busca pela solução ideal.

Dada a natureza complexa do PCVCP, uma variedade de algoritmos de otimização e heurísticas foram desenvolvidos para encontrar soluções aproximadas. A coleta de prêmios adiciona uma dimensão adicional ao problema, tornando-o mais desafiador e exigindo abordagens especiais para considerar tanto o custo das rotas quanto os benefícios dos prêmios coletados.

Assim como o PCV, o PCVCP também encontra aplicações em diversas áreas, incluindo logística, transporte, genética, biologia computacional e outras. A resolução desse problema tem potencial para otimizar processos em que a coleta de prêmios adiciona um elemento importante na tomada de decisões, agregando valor à busca por rotas eficientes e econômicas.

2.2 Função Objetivo

O Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP) pode ser visualizado como um cenário onde um caixeiro viajante precisa visitar cada cidade para coletar um prêmio associado, mas também deve pagar uma penalidade por cada cidade não visitada. O objetivo é minimizar a soma dos custos de viagem entre as cidades, considerando o custo de deslocamento entre elas, bem como as penalidades, e ao mesmo tempo garantir que o caixeiro visite um número suficiente de cidades que lhe permita coletar um prêmio mínimo

pré-estabelecido. Portanto, a solução ideal busca otimizar tanto os custos de viagem quanto as penalidades, garantindo a inclusão das cidades necessárias para atingir o valor mínimo de prêmios estabelecido.

2.3 Solução

O algoritmo GRASP (Greedy Randomized Adaptive Search Procedure) busca encontrar um ciclo hamiltoniano em um grafo, onde cada cidade é um nó com um prêmio associado e as arestas representam os custos das viagens. O objetivo é minimizar a soma dos custos de viagem e aplicar penalidades com base nos prêmios mínimos estabelecidos. O GRASP combina a geração de uma solução inicial gulosa, com busca local, fazendo pequenas modificações na solução atual, e com reinicializações aleatórias para explorar diferentes regiões do espaço de busca, buscando uma solução próxima ao ótimo global para o problema.

2.4 A vizinhança utilizada

No contexto do problema do Caixeiro Viajante (TSP), a vizinhança é um conjunto de soluções que são obtidas a partir de uma solução atual, realizando pequenas modificações ou perturbações que afetam a ordem das cidades visitadas. A ideia é explorar soluções próximas à solução atual para tentar melhorar a rota e minimizar o custo total da viagem.

Existem várias formas de definir a vizinhança para o TSP, algumas delas incluem:

- Troca de duas cidades adjacentes na rota: Nessa vizinhança, escolhemos duas cidades adjacentes na rota atual e trocamos suas posições. Isso pode levar a uma nova rota que potencialmente reduz o custo total da viagem.

- Inversão de um subconjunto de cidades: Nessa vizinhança, selecionamos um subconjunto de cidades consecutivas na rota atual e invertemos a ordem dessas cidades. Isso pode gerar uma nova rota que pode ser mais eficiente em termos de custo.
- Troca de duas cidades não adjacentes na rota: Nessa vizinhança, selecionamos duas cidades não adjacentes na rota atual e trocamos suas posições. Isso também pode resultar em uma rota melhorada.
- 2-opt: Essa é uma heurística comumente usada para definir a vizinhança no TSP. Ela envolve remover duas arestas da rota atual e reconectar as cidades de forma a obter uma nova rota que evite cruzamentos entre as arestas.

A definição da vizinhança é um aspecto fundamental na busca local iterada aplicada ao TSP, e a escolha da vizinhança pode afetar significativamente o desempenho e a qualidade das soluções encontradas. O objetivo é explorar eficientemente o espaço de soluções em torno da rota atual, em busca de soluções mais próximas do ótimo global do problema do Caixeiro Viajante.

2.5 Resultados obtidos pela Metaheurística na resolução do Problema.

Foi realizado um estudo em um artigo sobre Metaheurísticas híbridas para resolver o Problema do Caixeiro Viajante com Coleta de Prêmios (PCVCP). Os problemas-teste foram gerados com distâncias entre cidades (vértices do grafo) obtidas de uma distribuição uniforme no intervalo [50, 1000]. Os prêmios e penalidades associados a cada cidade também foram

gerados de forma uniforme, considerando os intervalos [1, 100] e [1, 750], respectivamente. O prêmio mínimo a ser coletado foi definido como 75% do somatório de todos os prêmios associados às cidades, seguindo intervalos utilizados em trabalhos anteriores na literatura.

Os resultados computacionais foram obtidos a partir de 30 execuções realizadas para cada problema-teste gerado. Os algoritmos foram implementados em C++, e os testes foram executados no sistema operacional Windows, utilizando um microcomputador com processador Athlon XP de 1,53 GHz e 256 MB de memória RAM.

A Tabela 1 apresenta os resultados dos experimentos computacionais. A primeira coluna contém os problemas-teste do PCVCP, e a segunda coluna ($|V|$) representa a cardinalidade do conjunto de vértices que compõem cada problema. A terceira e quarta colunas mostram o desempenho do algoritmo exato, incluindo o tempo de execução (em segundos) e o valor do ótimo global encontrado.

As colunas 5, 6 e 7 referem-se ao método heurístico proposto. A quinta coluna mostra o tempo médio de execução (em segundos), e a sexta coluna apresenta os melhores valores da função de avaliação (FOMelhor) obtidos com o modelo heurístico. Na sétima coluna, temos o desvio dos valores médios (FOMedia) em relação à melhor solução obtida em cada um dos problemas-teste, calculado conforme a equação .

$$Desvio = \frac{FOMedia - FOMelhor}{FOMelhor} \tag{2.1}$$

Na comparação dos resultados obtidos pelos métodos exato e heurístico, observou-se que o algoritmo heurístico proposto sempre alcançou o ótimo

global para os problemas teste em que esse é conhecido. O Procedimento de Busca Aleatória Gulosa Adaptativa também demonstrou robustez, pois a partir de diferentes soluções iniciais, alcançou soluções finais que apresentam um pequeno desvio em relação à melhor solução encontrada.

Portanto, os resultados obtidos validam a utilização do GRASP como um método eficaz para resolver o Problema do Caixeiro Viajante com Coleta de Prêmios. A capacidade do algoritmo heurístico de encontrar soluções próximas do ótimo global, juntamente com sua robustez em relação às soluções iniciais, reforçam sua aplicabilidade na solução desse desafiador problema de otimização combinatória.

Tabela 1: Resultados dos experimentos computacionais.

		MÉTODO EXATO		GRASP+VNS		
PROBLEMA-TESTE	| V |	ÓTIMO GLOBAL	TEMPO (S)	FOMelhor	TEMPO (S)	DESVIO (%)
v10	11	1765	1	1765	0,10	0,00
v20	21	2302	65	2302	1,04	0,00
v30a	31	3582	86	3582	5,43	0,00
v30b	31	2515	100	2515	3,83	0,00
v30c	31	3236	1786	3236	7,83	0,05
v50a	51	-	10800	4328	132,45	0,42
v50b	51	-	10800	3872	43,76	0,31
v100a	101	-	-	6892	692,09	0,52
v100b	101	-	-	6814	446,81	0,12
v250a	251	-	-	15310	918,33	0,88
v250b	251	-	-	14678	996,72	0,76
v500a	501	-	-	28563	2145,79	0,67
v500b	501	-	-	28524	2410,21	0,82

3 Referências

- https://www.researchgate.net/figure/Flow-chart-for-greedy-and-GRASP-approaches_fig1_308249881, Acesso em 23/01/2024;
- https://slideplayer.com/slide/4841357/, Acesso em 23/01/2024;
- https://www2.imm.dtu.dk/courses/02719/grasp/4grasp.pdf, Acesso em 23/01/2024;
- https://www.inf.ufpr.br/aurora/disciplinas/topicosia2/downloads/trabalhos/GraspTSP.pdf, Acesso em 23/01/2024;
- https://www.ime.unicamp.br/~mac/db/2015-1S-120022.pdf, Acesso em 23/01/2024;
- https://www.youtube.com/watch?v=9R-_BL_5Gno, Acesso em 23/01/2024;
- https://www.youtube.com/watch?v=ju_A9hHhjDo, Acesso em 23/01/2024;
- https://www.youtube.com/watch?v=xyLcxUoJjMM, Acesso em 23/01/2024;