Comparação de Performance Aplicando Busca Local em Algoritmos Genéticos na Resolução do Problema P-Medianas

Arthur Rodrigues Batista[1] **Douglas Ferreira Delefrati**[2]

[1]Departamento de Informática – Universidade Estadual de Maringá (UEM)
Maringá – PR – Brasil

ra105422@uem.br[1]

ra103654@uem.br[2]

Abstract. *This work proposes an approach to the capacited p-medians problem using a genetic algorithm. We compare the results of the classic GA implementation with the addition of the local search method. The metrics used for the comparison were convergence of solutions for the optimal result at each iteration, in addition to the execution time of each generation of possible solutions. According to the results, the local search strategy showed superiority in convergence, however the computational effort to generate new individuals was well above the classic implementation.*

Resumo. *Este trabalho propõe uma abordagem do problema de p-medianas capacitado utilizando algoritmo genético. Contrapomos os resultados da implementação clássica do AG com a adição do método de busca local. As métricas utilizadas para a comparação fora convergência das soluções para o resultado ótimo à cada iteração, além do tempo de execução de cada geração de possíveis soluções. De acordo com os resultados, a estratégia de busca local apresentou superioridade na convergência, contudo o esforço computacional para geração de novos indivíduos foi bastante acima da implementação clássica.*

1. Introdução

A busca de um número de vértices em um espaço Euclidiano é um problema clássico na computação. Este problema consiste em localizar um número p (inteiro e positivo) em um determinado espaço que satisfaça n pontos de demanda de maneira que a soma total das distâncias entre cada ponto e sua mediana mais próxima é minimizada [Lorena and Senne 2003]. Existem diversas variações deste problema, a escolhida para abordamos no trabalho foi o P-Medianas capacitado, no qual cada mediana candidata tem uma capacidade fixa, ou seja, um valor máximo de vértices que podem estar conectados a ela. O problema de p-medianas é classificado como NP-Difícil [Kariv and Hakimi 1979], consequentemente até os algoritmos heurísticos especializados em resolver esse problema requer um esforço computacional considerável [Correa et al. 2004]. Isto posto, a proposta deste trabalho é implementar e comparar duas estratégias para a resolução do problema P-medianas, sendo estas: Algoritmos Genéticos com e sem a utilização de busca local.

O Algoritmo Genético é uma heurística de pesquisa inspirada na teoria da evolução natural. Esse algoritmo reflete o processo de seleção natural, em que os indivíduos mais aptos são selecionados para reprodução, a fim de produzir descendentes da

próxima geração. De forma semelhante, os algoritmos de busca local passam de solução em solução no espaço das soluções candidatas (o espaço de pesquisa) aplicando alterações locais, até que uma solução considerada ótima seja encontrada ou que um tempo limite seja decorrido. A intenção de utilizar essas técnicas se justifica pelo fato de que, como citado, o problema P-Medianas pertence à classe NP-Difícil, ou seja, não se conhece algoritmo eficiente para resolvê-lo, utilizando essas técnicas, não é necessário executar todas as possibilidades para obter uma solução plausível (e às vezes até ótima).

Neste contexto, faremos uma analise com base na execução do Algoritmo Genético, com e sem a aplicação da busca local, a fim de averiguar se esta técnica seja viável para complementar aquela. Para tal, avaliaremos o quanto as estratégias se aproximaram da solução ótima com base em 6 casos testes. Ao final, concluiremos que a utilização da busca local juntamente ao Algoritmo Genético foi uma estratégia interessante e que demonstra potencial para se aproximar da solução ótima dos experimentos.

2. Metodologia

Esta seção será subdividida em quatro subseções. Na primeira abordaremos o funcionamento do Algoritmo Genético. Em seguida, comentar-se-á acerca dos parâmetros ajustados para o algoritmo citado em relação ao problema P-Medianas. Ademais, discutiremos a respeito da busca local. Por fim, serão especificadas as configurações da máquina na qual o algoritmo será executado. Ao final, pretende-se executar uma gama de testes a fim de validar se a utilização da busca local melhorou ou não o conjunto solução a cada nova geração.

2.1. Algoritmo Genético

O Algoritmo Genético (AG) é uma meta-heurística inspirada no processo de seleção natural, isto é, subconjunto que pertence à classe de algoritmos evolutivos (AE). Os Algoritmos Genéticos são comumente usados para gerar soluções de alta qualidade para problemas de otimização e busca, contando com operações bio-inspiradas, como mutação, cruzamento e seleção [Mitchell 1996].

2.1.1. Comportamento Biológico e Representação Computacional

Em um AG, uma população de soluções candidatas (chamadas de indivíduos ou fenótipos) são escolhidas, tipicamente aleatoriamente, e passam por um processo evolutivo, de modo que, a cada nova geração, são produzidos indivíduos mais aptos a sobreviverem se comparado às gerações anteriores. O conceito biológico que está ligado a esse processo é a cruzamento ou *crossing over*, cuja ideia consiste em cruzar o material genético (genótipo) e produzir indivíduos que contém características semelhantes aos seus criadores. Computacionalmente, os indivíduos podem ser representados por meio de cadeias de números binários, mas outras codificações também são implementadas [Whitley 1994]. A principal propriedade que torna essa representação genética conveniente é que suas partes são facilmente alinhadas devido ao seu tamanho fixo, o que facilita operações de cruzamento simples, apenas alterando os bits.

2.1.2. Visão Geral do Algoritmo

A priori é esboçado um espaço de busca (população) como entrada para o algoritmo, o tamanho da população depende da natureza do problema, mas normalmente contém centenas ou milhares de possíveis soluções. Ainda de acordo com o problema, as soluções podem ser escolhidas em áreas que soluções ótimas podem ser encontradas. Em cada geração, uma parcela da população existente é selecionada para gerar sucessores. As soluções individuais são selecionadas por meio de um processo baseado em condicionamento numérico, em que as soluções mais adaptativas (avaliadas por uma função objetiva) têm uma probabilidade maior de serem selecionadas. Certos métodos de seleção classificam a adequação de cada solução e selecionam as melhores. Outros métodos classificam apenas uma amostra aleatória da população, já que o processo de seleção requer um tempo computacional significativo em alguns casos.

A função objetivo é definida em conformidade à representação genética e mede a qualidade da solução representada. A função é sempre dependente do problema. Por exemplo, no problema da mochila, queremos maximizar o valor total de objetos que podem ser colocados em uma mochila que possui uma capacidade fixa. Uma representação da solução pode ser expresso por meio de cadeia de bits, em que cada bit representa um objeto diferente e o valor do bit (0 ou 1) representa se o objeto está ou não na mochila. Nem todas as cadeias são válidas, pois o peso dos objetos pode exceder a capacidade da mochila. A função objetiva, neste caso, poderia ser expressa como:

$$opt(array, n) = \begin{cases} \sum_{i=1}^{n} value(array[i]) & \text{Se não exceder à capacidade} \\ 0 & \text{Caso contrário.} \end{cases} \quad (1)$$

em que *opt* corresponde à função objetivo, *array* à cadeia de bits e *n* ao seu tamanho.

Após a seleção dos indivíduos, a próxima etapa do algoritmo é a combinação de dois operadores genéticos: *crossover* (também chamado de recombinação) e mutação. Para cada nova solução a ser produzida, um par de soluções (pais) é selecionado para reprodução. Ao produzir uma solução ”filho”usando os operadores mencionados, cria-se uma nova solução que normalmente compartilha muitas das características de seus ”pais”. Novos pais são selecionados para cada novo filho, e o processo continua até que uma nova população de soluções de tamanho desejável seja gerada. Embora os métodos de reprodução baseados no uso de dois pais sejam mais inspirados na biologia, alguns autores [Eiben et al. 1994] sugerem que a relação de mais que dois indivíduos geram cromossomos de maior qualidade.

Em muitos casos a aptidão média terá aumentado por este procedimento, uma vez que apenas os melhores organismos da primeira geração são selecionados para reprodução, juntamente com uma pequena proporção de soluções menos adequadas. Essas soluções menos adequadas garantem a diversidade genética dentro do patrimônio genético dos pais e, portanto, garantem a diversidade genética da geração subsequente de filhos. Além disso, certas soluções são escolhidas aleatoriamente para sofrerem mutação genética, isto é, alterar a codificação dos genes do indivíduo, garantido, também, maior variabilidade na população.

Embora o cruzamento e a mutação sejam conhecidos como os principais operadores genéticos, é possível usar outros operadores, como o reagrupamento, a extinção de

colonização ou a migração em algoritmos genéticos [Reza and Ziarati 2011].

Esse processo é repetido até que uma condição de finalização seja atingida. Condições comuns de terminação são:

- Uma solução é encontrada que satisfaça critérios mínimos;
- Número fixo de gerações atingidas;
- Orçamento de alocação (tempo de computação / dinheiro) atingido;
- Sucessores não produzem resultados melhores.

2.2. Parâmetros

Como foi possível observar, há várias maneiras diferentes de construir as partes do algoritmo genético, sendo fortemente dependente do problema que visa-se resolver. Desta forma, comentaremos a respeito de cada parâmetros ajustado no algoritmo para o problema P-Medianas.

As escolhas serão divididas em 7 categorias, sendo resumido nas etapas do processo evolutivo já comentado:

1. Indivíduos e População Inicial
2. Função objetivo (*fitness*)
3. Seleção
4. Cruzamento
5. Mutação
6. Atualização da População
7. Condição de Parada

2.2.1. Indivíduos e População Inicial

A princípio foi necessário pensar em uma estrutura que representasse uma possível solução para o problema P-medianas. Para tal, utilizamos um conjunto aleatório de vértices para serem candidatos à mediana (solução). O tamanho do conjunto seria proporcional à quantidade de medidas da instância. Por exemplo, se essa quantidade fosse 12 e a quantidade de vértices, 100 (enumerados de 1 a 100), uma possível solução seria o conjunto $(1, 20, 32, 4, 9, 10, 99, 4, 55, 87)$.

O número de indivíduos da população inicial foi fixado em 100, de tal forma que durante toda execução do algoritmo esse número não variasse. A motivação de uma escolha moderada, neste caso, deve-se ao fato de que populações pequenas geralmente não garantem muita variabilidade e, por outro lado, as grandes tendem a consumir bastante recursos computacionais ao calcular o *fitness*.

2.2.2. Função objetivo

Uma vez definida à população de soluções candidatas, a função objetivo foi formulada em duas etapas:

- Definição dos Vizinhos: para cada vértice não mediana, calcula-se a distância deste com as medidas e, caso essa distância seja mínima, faz-se uma ligação se o

limite de capacidade for respeitado, caso contrário, o processo se renínica até que todos os vértices não medidas sejam ligadas com uma mediana. Caso a solução seja inviável, ou seja, algum vértice ficou sobrando, atribuímos à solução um valor alto de fitness.

- Somar as Distância: uma vez definida as ligações dos vértices não medidas às medidas, o fitness foi calculado somando à distância de todas as arestas.

2.2.3. Seleção

Para a seleção das soluções que participarão na criação da próxima geração, foi utilizada a técnica Seleção por Torneio, cuja ideia consiste em definir k indivíduos, neste caso 4, aleatoriamente e escolher os dois mais adeptos (menor fitness) para cruzarem.

2.2.4. Cruzamento

Ao definir os pais cujos genes serão misturados a fim de gerar uma solução filho, a etapa de cruzamento foi feita utilizando os seguintes passos:

1. Inclua os genes (medidas) que pai e mãe têm em comum no do filho
2. Enquanto o filho não tiver uma quantidade de genes iguais ao pais
3. Sorteie um número k, 1 ou 0
4. Se k for igual a 1 escolha uma medida aleatória do pai
5. Se k for igual a 0 escolha uma medida aleatória do mãe
6. Caso a mediana não faça parte do material genético do filho, inclua a mediana
7. Repita

2.2.5. Mutação

O processo de mutação se assemelha ao de cruzamento, sorteia-se um número 0 ou 1, caso o número seja 1 o filho recém gerado sofrerá mutação. Essa etapa se resume em sortear uma mediana para ser substituída com outro vértice não mediana.

2.2.6. Atualização da População

A manutenção da população seguiu um critério Steady Stated, ou seja, a cada interação a população é atualizada. Para que o filho gerado seja incluso nesta população, primeiro calcula-se sua taxa fitness e, caso seja melhor do que o pior indivíduo, realiza-se uma substituição deste por aquele.

2.2.7. Condição de Parada

O critério de parada foi simplesmente fixado em um número de interações/gerações, sendo: 500

2.3. Busca Local

A busca local é um método heurístico para resolver problemas de otimização que exigem muito poder computacional. A busca local pode ser usada em problemas que são formulados para encontrar uma solução que maximize um critério entre várias soluções candidatas. Os algoritmos de busca local atuam no espaço das soluções candidatas (o espaço de pesquisa) aplicando alterações locais, até que uma solução considerada ótima seja encontrada ou que um tempo limite seja atingido [Arya et al. 2004]. A estratégia adotada neste trabalho denomina-se first adjustment, no qual a primeira solução vizinha que possui um fitness menor do que atual, transforma-se no vértice atual, assim a busca continua com os novos vizinhos do vértice recém substituído, até que possua um vértice em que o fitness seja menor que todos os seus vizinhos. Para construir as soluções vizinhas, uma mediana era retirada do conjunto solução e um vértice aleatório era adicionado em seu lugar. Esse processo era repetido p vezes, gerando p soluções vizinhas no final do processo.

2.4. Especificações do Ambiente de Execução

O experimento foi executado na plataforma Google Cloud, com uma máquina que possuía um processador de uma vCPU e 120 GB de memória RAM. Para o sistema operacional, utilizou-se o Ubuntu 18.04 LTS x64, com kernel 4.15.0-1028-gcp e gcc 7.3.0-27ubuntu1. Por fim, a linguagem de programação adotada foi Python.

3. Resultados e Discussões

Nesta seção realizaremos uma análise referente à execução dos casos testes, de modo que seja possível inferir se a utilização da busca local seja uma boa opção juntamente ao Algoritmo Genético.

3.1. Fitness e Desvio Relativo

A princípio, na tabela abaixo, temos as informações dos casos de executados, incluindo o nome do caso, a melhor solução conhecida na literatura (MS), a melhor solução sem (ALG) e com (ALG-BL) a utilização de busca local e seus respectivos desvio com a solução ótima (GAP2 e GAP1, respectivamente).

Comparação de *Fitness*					
Caso	MS	ALG-BL	GAP1	ALG	GAP2
SJC1	17288,99	19302	11,64	20845	20,56
SJC2	33270,94	36678	10,24	41382	24,37
SJC3a	45335,16	45998	1,46	52453	15,70
SJC3b	40635,90	45822	12.76	55212	35,87
SJC4a	61925,51	70009	13.05	85288	37,7
SJC4b	52458,02	57603	9.08	69856	33.16

De imediato verificamos que a estratégia utilizando busca local se sobressaiu em todos os casos testes, tendo uma resultado surpreendente, em especial, comparada à solução ótima do caso SJC3a. O motivo para tal se deve ao fato de que, uma vez gerado os sucessores da próxima geração, para cada sucessor, busca-se melhorar sua solução para que depois seja inserida na população. Desta forma, a população ficará mais ”forte”em um quantidade menor gerações, pois a etapa de aprimoramento antecede a inclusão, e, por conseguinte, poderá reproduzir indivíduos mais adeptos também em um período menor.

3.2. Convergência

Com base naquilo mencionado acima, acreditamos que a solução utilizando busca local foi melhor em detrimento do aprimoramento ante incluir a solução na população e, de fato, se observarmos, por exemplo, a Figura 1, isto é, gráfico de convergência do caso SJC1 (análogos aos demais casos), verificamos que a solução convergiu mais rapidamente à ótima do que sem o uso da busca local. Ainda, a maior discrepância das estratégias são identificada logo de início, em que o uso da busca local dá um salto para aprimorar a população enquanto a outra estratégia demora mais para que isso ocorra.

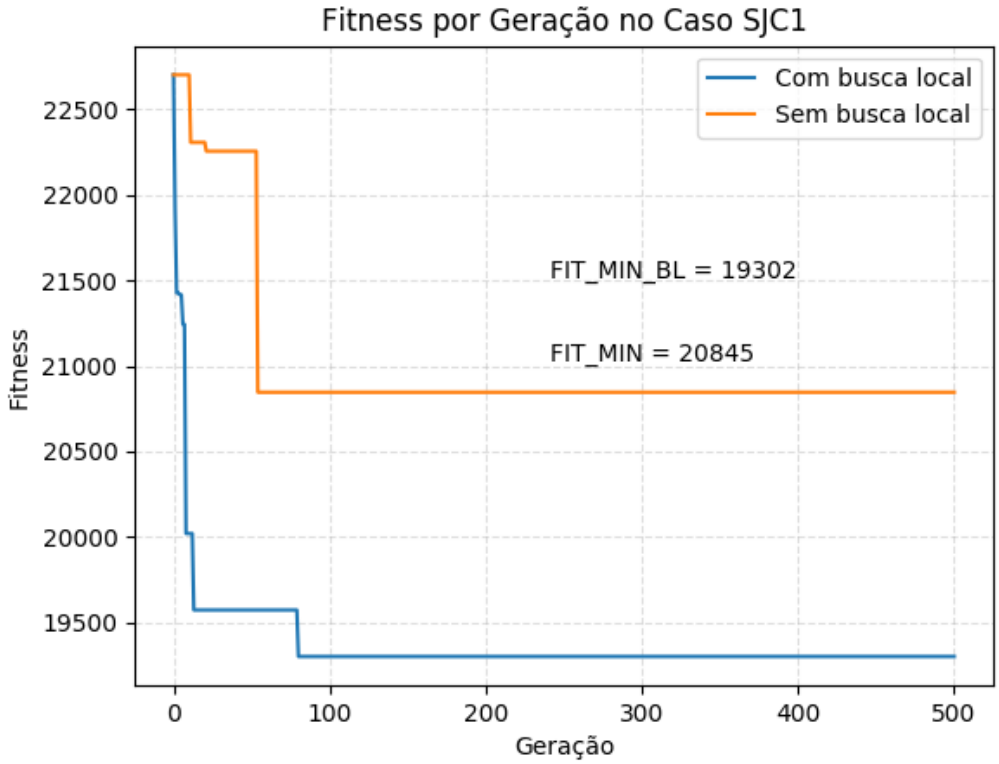

Figura 1.

3.3. Tempo de execução

Embora o uso da busca local tenha sido interessante para produzir indivíduos mais adeptos em um período menor de interações, ao analisarmos o gráfico de tempo de execução a cada geração (Figura 2), identifica-se que o tempo gasto para realizar essa elite é significativamente maior do que sem o uso da técnica.

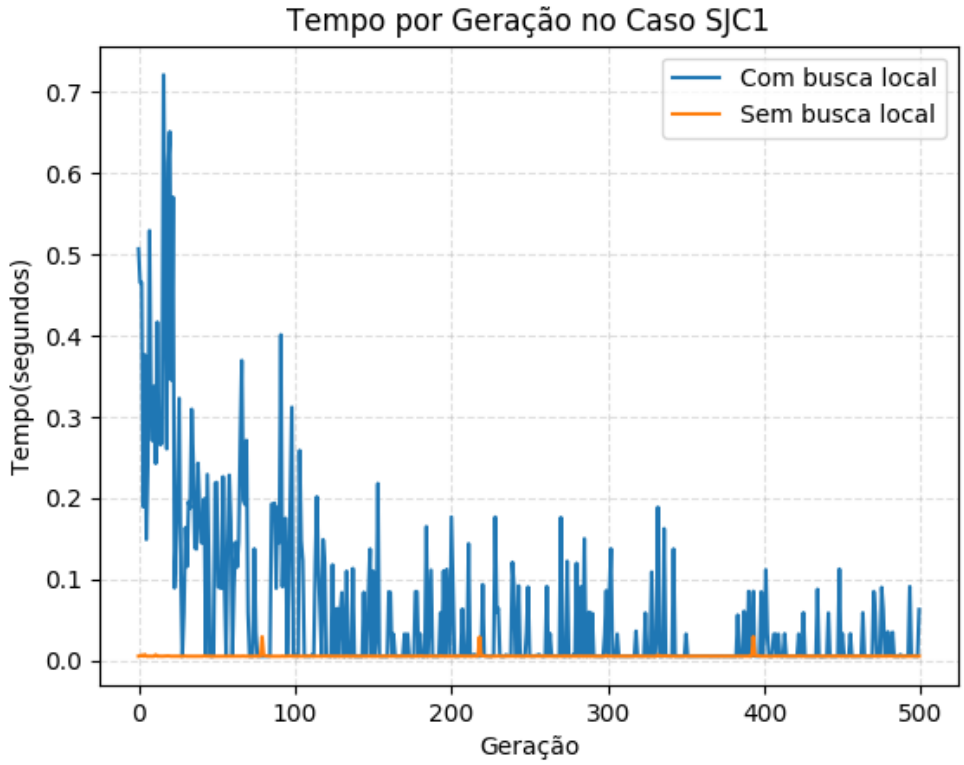

Figura 2.

4. Conclusão

Neste trabalho, fora proposto abordar o problema p-medianas capacitado utilizando um algoritmo genético. Também realizamos testes comparando o AG com e sem a estratégia de busca local, com o intuito de averiguar o impacto que essa técnica teria nos resultados. Foi avaliado duas métricas, a convergência para o a solução ótima e o esforço computacional, no qual foi possível observar que o comportamento do algoritmo com a busca local destacou-se na convergência, porém, com um tempo de execução bastante elevado quando comparado a abordagem sem essa estratégia. No entanto, observando a (figura 1) podemos presumir que o valor do fitness de ambas abordagens se encontrariam se a quantidade de gerações fosse aumentada. Sendo assim, uma interessante direção de pesquisa para futuros trabalhos seria realizar outros experimentos aumentado o número de iterações do AG, com a finalidade de analisar se compensa utilizar o algoritmo sem a busca local.

Referências

Arya, V., Garg, N., Khandekar, R., Meyerson, A., Munagala, K., and Pandit, V. (2004). Local search heuristics for k-median and facility location problems. *SIAM Journal on computing*, 33(3):544–562.

Correa, E. S., Steiner, M. T. A., Freitas, A. A., and Carnieri, C. (2004). A genetic algorithm for solving a capacitated p-median problem. *Numerical Algorithms*, 35(2-4):373–388.

Eiben, A. E., Raué, P. E., and Ruttkay, Z. (1994). Genetic algorithms with multi-parent recombination. In Davidor, Y., Schwefel, H.-P., and Männer, R., editors, *Parallel Problem Solving from Nature — PPSN III*, pages 78–87, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kariv, O. and Hakimi, S. L. (1979). An algorithmic approach to network location problems. i: The p-centers. *SIAM Journal on Applied Mathematics*, 37(3):513–538.

Lorena, L. A. N. and Senne, E. L. F. (2003). Local search heuristics for capacitated p-median problems. *Networks and Spatial Economics*, 3(4):407–419.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, USA.

Reza, A. and Ziarati, K. (2011). A multilevel evolutionary algorithm for optimizing numerical functions. *International Journal of Industrial Engineering Computations*, 2.

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85.