Wagner de Melo Reck

Algoritmos de agrupamento capacitado aplicado ao Problema de Despacho de Ordens de Serviço

Alegrete - RS

2010

Wagner de Melo Reck

Algoritmos de agrupamento capacitado aplicado ao Problema de Despacho de Ordens de Serviço

Monografia apresentada para obtenção do Grau de Bacharel em Ciência da Computação pela Universidade Federal do Pampa.

Orientador: Vinícius Jacques Garcia

Alegrete - RS

2010

Wagner de Melo Reck

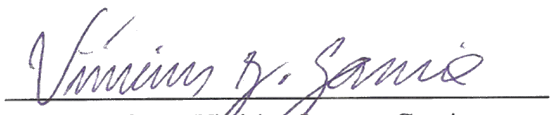
Algoritmos de agrupamento capacitado aplicado ao Problema de Despacho de Ordens de Serviço

Monografia apresentada ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Área de concentração: Ciências Exatas e da Terra

Dissertação defendida e aprovada em: 12 de Julho de 2010.

Banca examinadora:



Prof. Dr. Vinícius Jacques Garcia
Orientador
UNIPAMPA - Campus Alegrete

Prof. Msc.. Marcelo Cezar Pinto
UNIPAMPA - Campus Alegrete

Prof. Dr. Cleo Zanella Billa
UNIPAMPA - Campus Alegrete

Dedico esse trabalho à minha amada esposa Joseane,
pela paciência, carinho e apoio incondicional em todos
meus projetos acadêmicos e pessoais.

AGRADECIMENTO

Meu primeiro agradecimento vai para minha família, sem a qual eu não estaria aqui vivo. Agradeço pela paciência e pela compreensão no momentos que mais precisei (principalmente nos vários meses sem visitá-los), pelas palavras de apoio e conforto (*"logo logo termina!"*) e claro pela sua índole exemplar e conhecimento único que pude desfrutar durante minha vida. Muito obrigado por tudo, pai, mãe e mano, os aplausos são para vocês também!

Acho que eu não estaria aqui, escrevendo esse trabalho, se não fosse por meu querido mestre, tutor e amigo, meu professor Vinícius Jacques Garcia, ao qual eu deve grande parte de meu conhecimento, não só acadêmico mas conduta. Obrigado pelas conversas amigas e paciência com minha dúvidas. Espero trilhar novos caminhos ao teu lado.

Também deve dizer que sem uma outra pessoa eu não estaria aqui, quer dizer, eu até estaria mas possivelmente tomando algum tarja preta ou em uma consulta num psiquiatra, o que eu quero dizer é que se eu mantenho ainda um pouco de lucidez mental é graças a minha amada esposa Joseane (mas chamem-na de Jô). Muito obrigado pelos ótimos momentos ao seu lado, por ser minha companheira nas minhas idéias geniais (leia-se loucas), pela força nas aulas de cálculo e por rir das minhas piadas (pelo menos das boas).

Agradeço ao meu gato, que tem o nome de POG mas ninguém chama ele por esse nome, por me fazer rir quando tentava pegar o cursor do mouse :).

Tenho um agradecimento especial aos demais professores que me acompanharam e incentivaram durante a graduação, compartilhando seu tempo, paciência e conhecimento, contribuindo na minha formação. Obrigado Amanda, Vanessa, Diego, MC(Marcelo), Alessandro, Divane, Fabiane, Fernando, Daniel, Rogério, Antônio, Vinícius(Montagner), Deise, Eduardo, Cleo.

Além do conhecimento na graduação levo também a amizade e companheirismo de vários colegas do nosso curso e de outros também. Obrigado pelos momentos divertidos durante os trabalhos e momentos de estudo. Obrigado Parizi, Rodrigo, Tati, Ângela, Igor, Daniel, Peuchibo e outros tantos que encheriam muitas outras páginas.

#finalDoTCC

Você acha que o seu problema é sério? — exclamou Marvin, como se estivesse se dirigindo ao novo morador de uma sepultura — E eu? O que eu faço se eu sou um robô maníaco-depressivo? Não, nem tente me responder; eu sou 50 mil vezes mais inteligente que você e nem eu sei a resposta. Só de tentar me colocar no seu nível intelectual, fico com dor de cabeça.

Douglas Adams — O Guia do Mochileiro das Galaxias

RESUMO

Com uma concorrência cada vez maior, as empresas devem tornar suas operações cada vez mais enxutas para evitar gastos desnecessários. Para empresas que atendem clientes distribuídos por uma cidade, esses atendimentos devem ser feitos de maneira otimizada, evitando desperdícios, tanto financeiros quanto de tempo com deslocamentos desnecessários. Para evitar tais deslocamentos, o despacho dos atendimentos para as equipes deve levar em consideração a localização dos atendimentos e o tempo necessário para executá-lo. De modo a promover tais benefícios, podemos olhar esse problema como um problema de agrupamento capacitado, onde desejamos associar os atendimentos com as equipes de modo que todos atendimentos associados a uma equipe estejam próximos uns dos outros e que seja possível de executar todos eles dentro de sua jornada de trabalho.

Este trabalho faz um estudo de alguns métodos heurísticos clássicos na literatura para o problema de agrupamento capacitado e de busca local aplicados ao problema de despacho de ordens de serviço (PDOS), tratando também sobre algumas características das instâncias como a capacidade total excedente e a dispersão dos atendimentos no espaço. Além disso, nós desenvolvemos um pequena alteração em um dos métodos já existentes, de modo que o método resultante conseguiu produzir soluções melhores que que obtidas com o método original para a grande maioria das instâncias.

Palavras-chave: PO; Heurísticas; PAC, Agrupamento.

ABSTRACT

With competition increasing, companies must make their operations more lean to avoid unnecessary expenses. For companies that serve customers spread over a city, these services must be made optimally, avoiding waste, both financial and time with unnecessary travels. To avoid such displacement, dispatching of calls for the teams must take into account the location of call in space and time required to perform such call. To promote these benefits, we can look at this problem as a capacitated clustering problem, where we want to associate the calls with the teams so that all calls associated with a team are close to each other, reducing the distance between the calls, and that it is possible they all run within team's workday.

This work is a study of some classical heuristics methods in the literature for the clustering problem and local search apllyed to the service order dispatch problem (SODP), dealing also with some characteristics of the instances as the total capacity surplus and dispersion of care in space. In addition, we developed a small change in one of the existing methods, so that the resulting method could produce better solutions than that obtained with the original method to the vast majority of instances.

Key-words: OR; Heuristic; CCP, Clustering;

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

OS	Ordem de serviço,	p. 11
PDOS	Problema de Despacho de Ordens de Serviço,	p. 12
CCP	Problema de Agrupamento Capacitado - Capacitated Clustering Problem,	p. 12
CPMP	Problema das P-Medianas Capacitado - Capacitated P-Medians Problem,	p. 12
CTSP	Problema do Caixeiro Viajante Agrupado - Clustered Travel Salesman Problem,	p. 13
CCCP	Problema de Agrupamento Centrado Capacitado - Capacitated Centred Clustering Problem,	p. 18
GCC	Coleção de Compiladores GNU - GNU Compiler Collection,	p. 31
GNU	GNU Não é Unix - GNU is Not Unix,	p. 31

SUMÁRIO

1 Introdução

Em alguns setores de empresas, para efetuar o atendimento a um cliente, elas devem deslocar um funcionário (ou uma equipe deles) da sua base de operações até a casa do cliente para executar o serviço. Isso ocorre principalmente em empresas que prestam algum tipo de serviço de atendimento domiciliar aos clientes, como por exemplo empresas de entrega de correspondência, provedores de internet no que diz respeito a suporte aos clientes e companhias de distribuição de energia elétrica também referente aos suporte a clientes.

Esses tipos de atendimentos aos clientes são chamados de ordens de serviço (OS), podendo ter outros nomes conforme a área de atuação da empresa, mas sempre com o intuito de registrar que a empresa deve fazer um atendimento a um determinado cliente. Cada OS contem alguns dados básicos como a localização do atendimento e tempo previsto de atendimento. Trataremos aqui a localização como um ponto no espaço $\Re^2$ e o tempo como a demanda de tempo que a OS levará para ser atendida.

Como medida de deslocamento apenas consideraremos a distância euclidiana entre os pontos, pois algumas instâncias reais se fossem representadas com a distância real que a equipe percorreria, iria demandar um mapa completo da localidade e mãos das ruas, o que nem sempre é disponível.

Nesse cenário de estudo tomamos algumas premissas como válidas:

- Uma OS deve ser atendida por um funcionário/equipe;
- Não levamos em consideração agendamentos de horários, apenas que todas as OS devem ser executas;
- Uma OS possui um tempo que a equipe levará para executa-la;
- Uma equipe consegue executar todas as OS atribuídas a ela no tempo estimado e predefinido de cada uma delas;

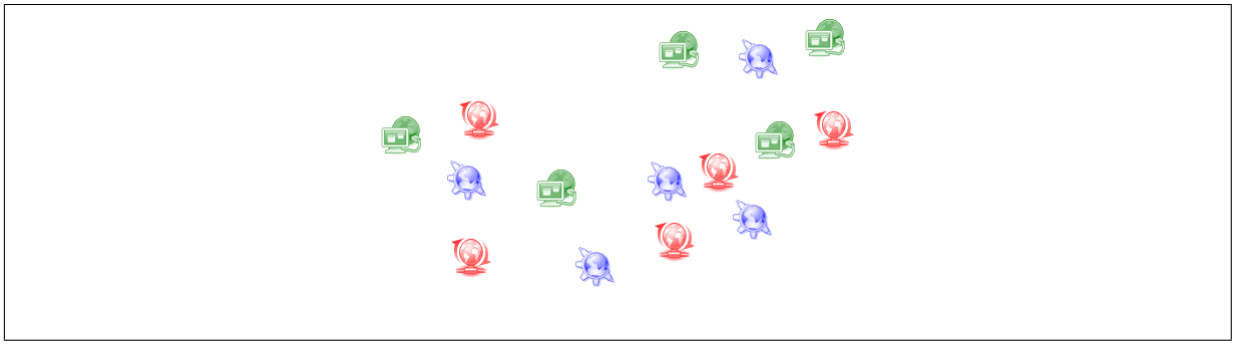


FIGURA 1.1: Alocação de OS apenas pelas Demandas e Capacidades

- O tempo de atendimento de todas as ordens atribuídas a uma equipe não deve ultrapassarem sua jornada de trabalho que é préviamente estipulada;
- As ordens são as de apenas um dia. As que entrarem durante o atendimento ficam agendadas o próximo dia.
- Não será considerado o tempo de deslocamento entre os atendimentos das OS.
- Será tentado inserir ao menos um atendimento para cada equipe, mesmo que eles execute apenas tal atendimento.

Essas premissas definem o PDOS , onde devemos associar todas OS as equipes de modo que uma OS seja atendida por uma, e não mais que uma, equipe e é desejado minimizar o deslocamento entre os atendimentos, uma vez que isso diminui os custos e o tempo de deslocamento entre os atendimentos devido a menor distância percorrida, o que é algo desejável para as empresas.

Ao designar um conjunto de OS a uma equipe, deve-se levar em consideração a carga horária de cada uma delas e as demandas de tempo que estão relacionadas com o conjunto de OS atribuído a equipe. Isso por si só pode ser visto como um Problema da Mochila, visto que temos vários atendimentos, cada um com seu tempo indivídual, e cada equipe possuí uma capacidade.

Um exemplo hipotético de atribuição de OS a equipes é ilustrado na FIGURA 1.1, constando 3 equipes para realizar 15 atendimentos. É possível perceber que os atendimentos de cada equipe encontram-se dispersos, requisitando um maior deslocamento das equipes de um atendimento a outro quando se vislumbra um agrupamento em que os pontos de atendimento estivessem mais próximos.

Um problema que engloba essas características apresentadas é o CCP (OSMAN; CHRISTOFIDES, 1994) , clássico na literatura e também conhecido como (CPMP) . Na figura 1.2

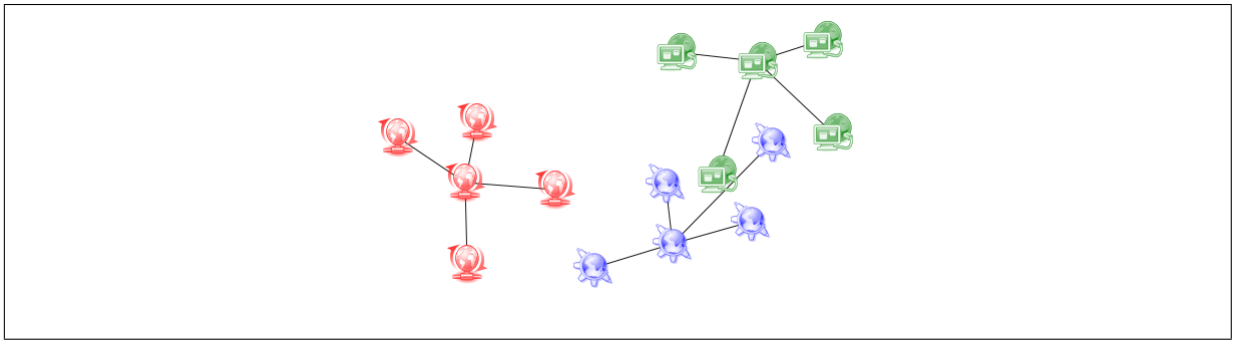


FIGURA 1.2: Possível atribuição das Mesmas OS por uma agrupamento capacitado

temos um exemplo de como as ordens de serviço poderiam estar associadas às equipes se associadas utilizando de métodos de agrupamento capacitado.

Mesmo com tal cuidado ao executar o despacho das ordens de serviço para as equipes, é possível notar que ainda assim não foi possível atribuir as OS de modo que todas OS de de uma equipe fique próximas de uma OS escolhida como centro. Isso é devido as restrições de capacidade das equipes e demandas das OS.

As instâncias inéditas desse problema foram retiradas de agendamentos reais de ordens de atendimento de uma empresa e tiveram que ter a capacidade de atendimento normalizadas para tornar o problema factível. Essa normalização foi necessária uma vez que a demanda dos atendimentos é maior que a capacidade de todas equipes. Esse fato ocorre por existirem demandas adicionais de dias anteriores e possíveis eventos especiais (por exemplo um temporal em uma cidade que tenha causado avarias na rede elétrica). A frente neste trabalho será apresentado um estudo sobre essas folgas e ajustes executados nas mesmas.

Não será tratado a geração de rotas entre os atendimentos, mas esse problema já foi tratado na literatura como o CTSP , onde desejamos entrar os menor ciclo Hamiltoniano nos pontos, sendo que os pontos dos agrupamentos devem ser visitados em sequência (POTVIN; GUERTIN, 1996).

Esse trabalho foi dividido em 5 partes: na primeira é apresentado uma breve visão do problema de agurpamentos, citando sua complexidade e variações, e no final uma abordagem ao problema de agrupamento e de como o PDOS pode ser visto como um problema de agrupamento capacitado. Na segunda parte são apresentadas cada uma das heurísticas estudadas, bem como um as técnicas de busca local utilizadas. Na terceira parte apresentamos as ferramentas computacionais desenvolvidas. Na quarta parte é mostrado um estudo sobre as características de agrupamento e demanda das instâncias. Na quinta parte constam os resultados obtidos pelos algoritmos implementados e as comparações com resultados apresentados na literatura.

2 *Problema de Agrupamento*

Agrupamento não apenas expressa o ato de separar dados em grupos, mas separá-los de modo que os elementos dos grupos resultantes tenham algumas características comuns entre si. Segundo Lorena e Furtado (LORENA; FURTADO, 2001), “problemas de agrupamento geralmente aparecem em classificação de dados para algum propósito como armazenar e recuperar ou analisar dados. Todo algoritmo de agrupamento tentará determinar algum agrupamento inerente ou natural nos dados, usando medidas de distância ou similaridade entre os dados”.

Um exemplo bem simples de agrupamento seria o ato de separar poucas frutas com boa aparência boa dasquelas que apresentam algum sinal de degradação aparente. Para esse exemplo, um ser humano conseguiria executar a tarefa de forma competitiva com um algoritmo executado por um computador, mas quando incluímos um grande de dados de entrada e desejamos que sejam levadas em consideração outras medidas ou outras restrições (ex. dividir as mesmas frutas em 10 grupos, sendo 5 de frutas de melhor aparência e 5 de piores, alem de que os objetos de cada grupo devem ser de cores e tamanhos semelhantes), os algoritmos de agrupamento podem chegar a uma solução muito melhor em menor tempo.

Dentre as aplicações práticas podemos destacar o uso de agrupamentos para data mining, processamento de imagens, procura por padrões, redução de cores de imagens, localização de facilidades, reconhecimento de padrões, alocação de tarefas a máquinas, definição de áreas de coleta de lixo entre muitas outras aplicações.

Os problemas de particionamento podem ser organizados segundo a taxonomia mostrada por (JAIN; MURTY; FLYNN, 1999) reproduzida na figura 2.1.

O problema de agrupamento está relacionado a classe de problemas de clustering, tendo o método *k-means* como um dos mais conhecidos exemplos de resolução.

Os problemas de particionamento, que serão estudados nesse trabalho, são problemas onde é desejado descobrir partições nos conjuntos de dados de modo que cada partição contenha apenas elementos similares entre si.

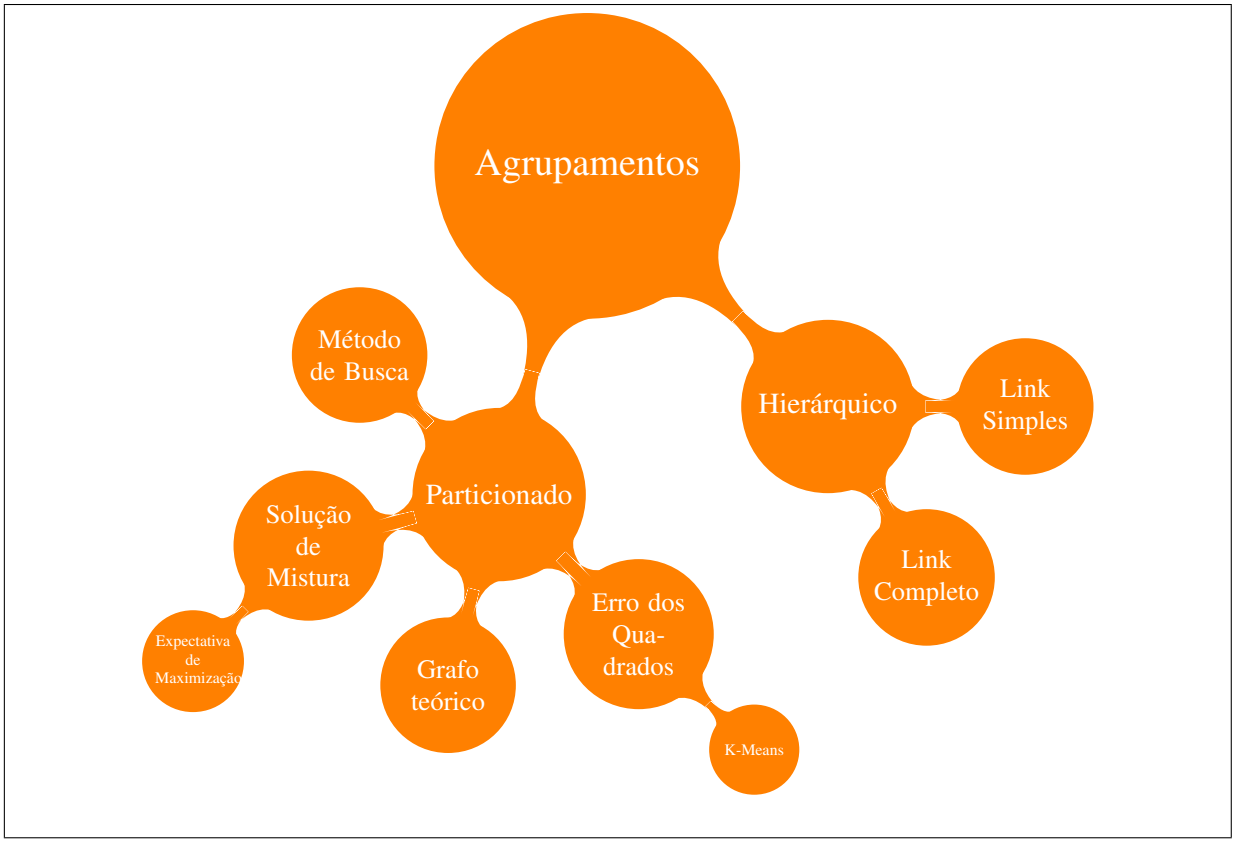


FIGURA 2.1: Taxonomia para os problemas de agrupamento

A escala de dificuldade para resolver o problema de particionamento pode ser vista pela expressão de p partições distintas existentes em n pontos,$N(n,p)$, definida por (LIU, 1968):

$$N(p,n) = \frac{1}{p!}\sum_{i=0}^{p}(-1)^{p-1}\binom{p}{i}i^n \tag{2.1}$$

Ahmadi e Osman apresentam em (AHMADI; OSMAN, 2004) um exemplo do resultado da computação dessa expressão para uma instância com 50 pontos, o resultado pode ser visto na imagem 2.2.

Pode ser observado que mesmo para instâncias não tão grandes, o espaço de busca é gigante. Quando consideramos instâncias maiores, como as resolvidas por Taillard (TAILLARD, 2003), onde são tratados 2863 pontos, o espaço de busca cresce ainda mais. O uso de métodos exatos é proibitivo pelo tempo computacional que seria necessário para essas instâncias maiores, e sendo esses problema da classe NP-Completo (GAREY; JOHNSON, 1979) não existe um algoritmo que resolva de forma exata em tempo polinomial, assim o que resta é o uso de métodos heurísticos e híbridos. Outro fato sobre esses dados é que a ocorrência de mais agrupamentos possíveis ocorra com $p = 16$, o que vem a mostrar que o problema é assimétrico em relação a p.

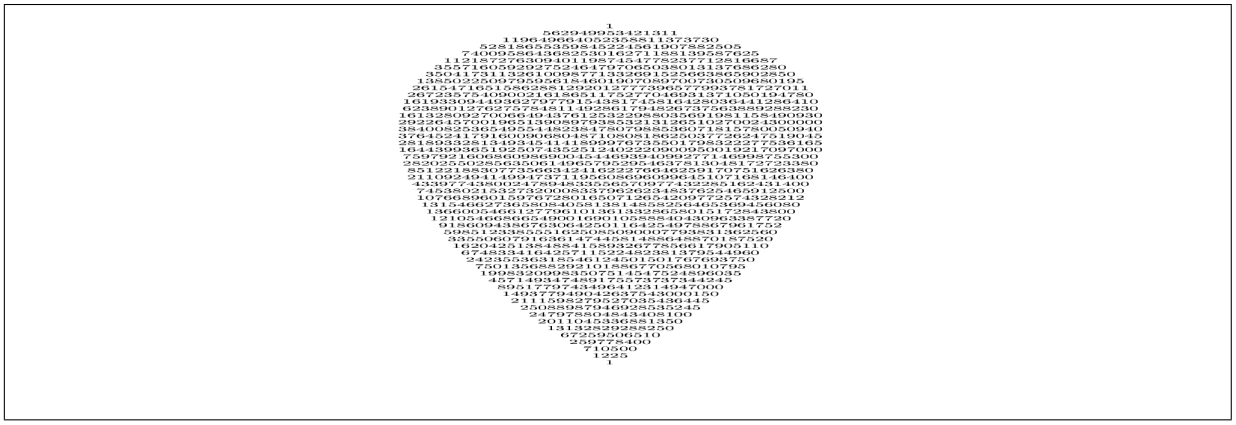


FIGURA 2.2: Valores para N(n, p). No topo p = 1 e na base p = 50.

Os problemas de agrupamentos por particionamento estão, do ponto de vista de complexidade, entre os problemas combinatoriais mais difíceis de serem resolvidos (AHMADI; OSMAN, 2004)

Quando temos um conjunto de pontos no espaço ($\Re^2$ por exemplo) e desejamos agrupar os pontos pela sua localização, podemos utilizar como medida de dispersão dos grupos a distância euclidiana dos pontos em relação ao centróide calculado pelos pontos do agrupamento. Esse é também conhecido como Problema das K-Médias (K-Means) apresentado por (MACQUEEN et al., 1966)

O problema das P-Medianas é uma variação do problema das K-Médias, com a diferença de utilizar uma mediana (um dos indivíduos) como referência para medição das similaridades dos pontos de cada um dos agrupamentos.

Acrescentando a restrição de capacidade aos agrupamentos, temos o problemas das P-Medianas Capacitado ou também tratado na literatura por CCP.

2.1 CCP - Problema de Agrupamento Capacitado

Como já apresentado anteriormente, o CCP tem por objectivo agrupar indivíduos de modo a maximizar as similaridades, nesse caso a proximidade, entre entre todos indivíduos de um agrupamento e a mediana do próprio agrupamento. Para o caso dos indivíduos serem representados por pontos no $\Re^2$, temos como similaridade o posicionamento dos mesmos no plano, a qual é medida pela distância entre eles. Na Fig. 2.3 é apresentado um exemplo de uma possível solução do CCP dado um conjunto de pontos.

O problema de agrupamento capacitado pode ser visto como: dado um conjunto I de n indivíduos com suas respectivas demandas d_i $i \in I$, é desejado agrupar os n pontos em p agru-

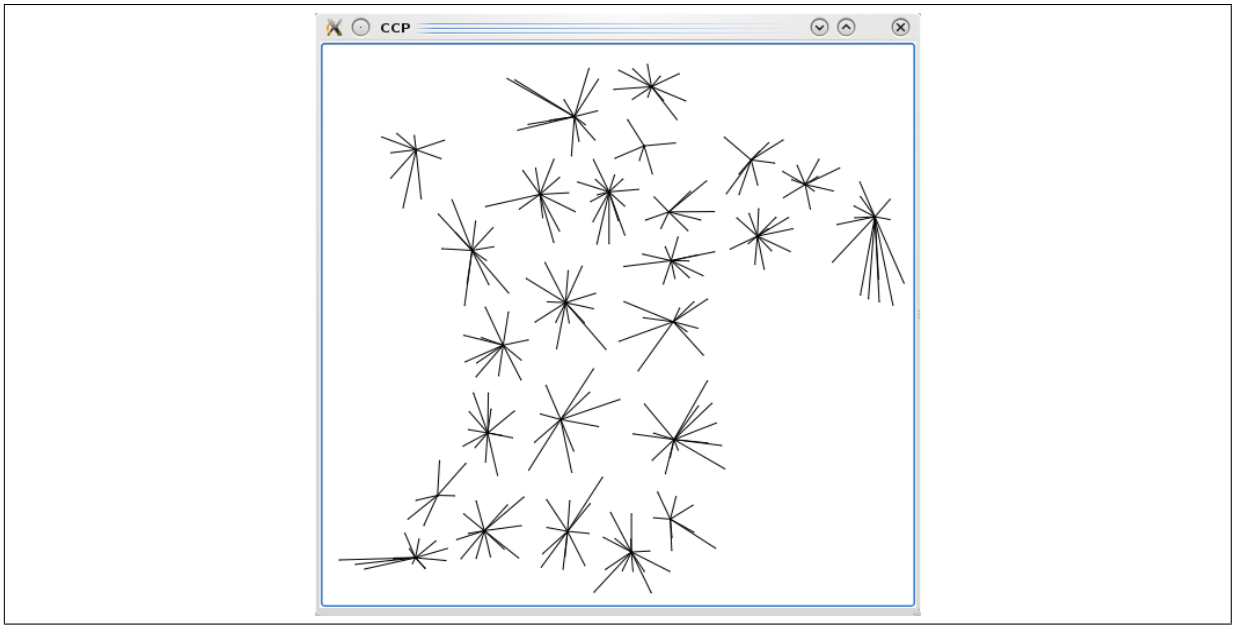


FIGURA 2.3: Exemplo de uma solução do CCP

pamentos, onde a soma das demandas em cada agrupamento deve ser menor que a capacidade total do agrupamento, $Q_j, \forall j \in P$, sendo P o conjunto de agrupamentos, e é desejado minimizar as dissimilaridades entre uma mediana do agrupamento, para isso possuímos uma matriz c_{IxI} com as distâncias entre todos os pontos. O problema pode ser representado pela seguinte formulação matemática (AHMADI; OSMAN, 2004):

$$Min \sum_{i \in P} \sum_{j \in I} c_{ij} x_{ij} \tag{2.2}$$

Sujeito à:

$$\sum_{j \in p} x_{ij} = 1, \ \forall i \in I \tag{2.3}$$

$$\sum_{j \in P} y_j = p \tag{2.4}$$

$$x_{ij} \leq y_{j,} \forall i \in I, \forall j \in P \tag{2.5}$$

$$\sum_{i \in I} q_i x_{ij} \leq Q_j, \ \forall j \in P \tag{2.6}$$

$$x_{ij}, y_j \in \{0,1\}, \ \forall i \in I, \ \forall j \in P \tag{2.7}$$

Em (2.2), temos a função objetivo que minimiza a distância entre os componentes de cada agrupamento com a mediana do agrupamento. A medida c_{ij}, do ponto i até o ponto j (mediana

do agrupamento), somente é considerada se o ponto i está associado ao agrupamento j pela variável binária x_{ij}, sendo $x_{ij} = 1$ caso o ponto i esteja associado ao agrupamento j e 0 caso contrário. A restrição (2.3) indica que cada ponto deve estar associado a um agrupamento. Já (2.4) indica que devem existir p agrupamentos. Os indivíduos somente devem ser associados a agrupamentos existentes (2.5), sendo y_j o conjunto de pontos escolhidos como medianas, sendo y_j igual a 1 se j for uma mediana e 0 caso contrário. Para garantir que a capacidade Q_j de um agrupamento j não será ultrapassado, é colocada a restrição (2.6). A última restrição, (2.7), especifica as variáveis inteiras de decisão.

Quando temos o valor de capacidade dos agrupamentos homogêneo, ou seja, o valor de $Q_j = Q_i, \forall j \, e \, \forall i \in P$, podemos dizer que esse é um problema das $p-medianas$ capacitado.

Um problema semelhante ao CCP foi proposto por Negreiros e Palhano em (NEGREIROS; PALHANO, 2006). Ele é o CCCP que consiste em agrupar os pontos levando em consideração a distância não para um dos pontos (mediana) mas para o centróide calculados com os pontos de cada agrupamento como apresentado na figura 2.4. São propostas 2 variações deste problema: uma onde o número de agrupamentos (p)é previamente conhecido ($p-CCCP$) e outra onde desejamos minimizar o número total de agrupamentos ($g-CCCP$). Após analisarmos esse problemas, decidimos permanecer trabalhando apenas com o problema CCP por melhor representar o PDOS.

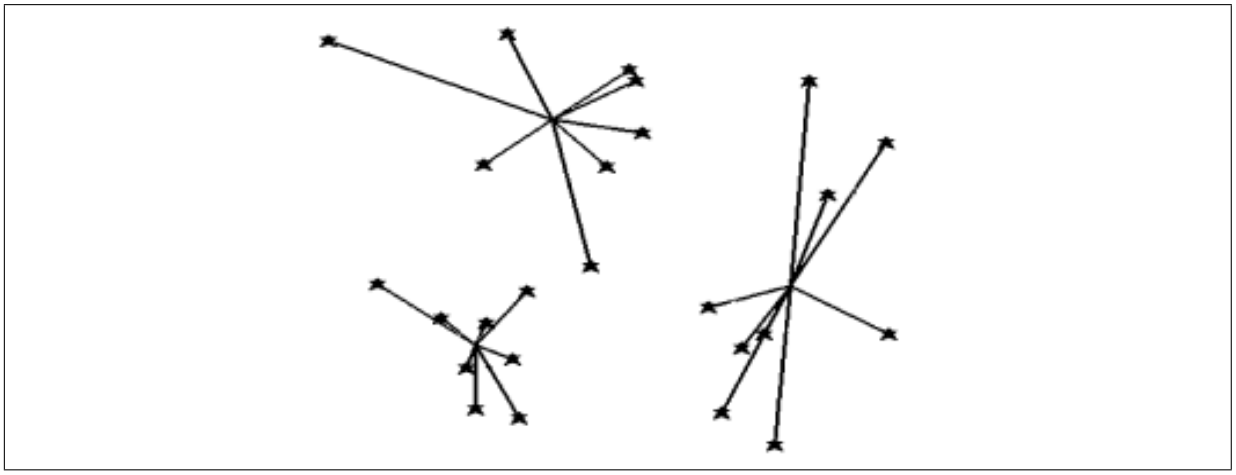


FIGURA 2.4: Uma solução para CCCP (NEGREIROS; PALHANO, 2006)

2.2 Métodos de resolução

A utilização de métodos exatos ou mesmo de *branch and bound* não são eficientes para instâncias com maior número de pontos (indivíduos) (OSMAN; CHRISTOFIDES, 1994), pelo grande tempo de processamento exigido.

Para obter soluções em tempos computacionais viáveis, é necessário a utilização de heurísticas. Métodos heurísticos podem não gerar uma solução ótima, ou mesmo próxima dela, mas conseguem gerar soluções muito boas em um tempo computacional polinomial.

Podemos distinguir 2 tipos de heurísticas(OSMAN; CHRISTOFIDES, 1994):

- Construtivas: Algoritmos desenvolvidos especificamente para uma determinada classe de problemas. Normalmente as soluções estão em um ótimo local ou muito próximo de um;
- Melhoramento: A partir de uma solução inicial válida, e busca na vizinhança iterativamente por outras soluções melhores. A busca na vizinha normalmente conduz a um vale/pico, chegando a um ótimo local ao final das iterações.

No próximo capitulo apresentamos as heurísticas escolhidas para essetrabalho (uma construtiva e 4 iterativas) e sobre a busca local utilizada para melhorar as soluções geradas por as heurísticas.

3 *Métodos Heurísticos para o CCP*

O CCP, é descrito na literatura como um problema $\mathscr{NP} - completo$(GAREY; JOHNSON, 1979), não existindo algoritmos que resolvam o problema de forma ótima em tempo polinomial. Para os métodos exatos, que apresentam melhores soluções, eles somente podem ser utilizados para instância pequenas (cerca de 100 indivíduos ou menos), uma vez que é um problema de programação inteira e o número de variáveis de decisão é da ordem de $n^2 + n$ (MULVEY; BECK, 1984), onde n é o número pontos de demanda do problema. Para instâncias muito grandes, o uso de heurísticas traz soluções muito boas, algumas vezes muito próximas aos métodos exatos, por isso a sua escolha nesse trabalho.

Serão apresentadas algumas heurísticas construtivas para o CCP:

Farthest: Proposta por Osman e Christofides em (OSMAN; CHRISTOFIDES, 1994), tem como base escolher os pontos mais afastados para serem os primeiros centros

Density: Proposta por Ahmadi e Osman em (AHMADI; OSMAN, 2004), visa construir uma solução inicial usando a densidade de pontos para construir a solução inicial.

H-Means: O método de Forgy (também conhecido, e tratado aqui, como H-Means) foi inicialmente proposto por (FORGY, 1965) como melhoria ao método K-Means e tem como principio a procura iterativa dos melhores centros para os agrupamentos realocando os pontos a cada troca.

J-Means: Proposta por Hansen e Mladenović(HANSEN; MLADENOVIĆ, 2002), usa a noção de pontos ocupados e não ocupados (a uma certa distância do agrupamento) e a inserção de tais pontos não ocupados como novos agrupamentos, retirando algum agrupamento antigo.

Randon Density: Uma alteração na heurística de Density que faz uso da seleção de indivíduos aleatórios para escapar de mínimos locais.

Em todos os métodos apresentados, é necessário recalcular o centro dos agrupamentos, ou

seja, procurar dentre os indivíduos de cada agrupamento qual é o que minimiza a soma das distâncias de todos indivíduos do seu agrupamento até si mesmo. Testar todos os indivíduos de um agrupamento com $\frac{n}{p}$ objetos, seria necessário medir a distância de cada um dos $\frac{n}{p}$ indivíduos até os demais $\frac{n}{p} - 1$. Como esse recálculo dos centros é muito referenciado nos métodos, foi desenvolvido o algoritmo 1 que calcula o centroide do agrupamento e mede a distância de todos os pontos com o centroide calculado. Após isso, são testados os k pontos mais próximos do centroide como prováveis centros. O valor de k foi de 30% dos pontos existentes no agrupamento. Esta escolha foi baseada num compromisso entre diversidade de pontos e complexidade do procedimento associado.

ALGORITMO 1 FindBestCenterOfCluster(Cluster)

1: $np \leftarrow Cluster.numObj$
2: $k \leftarrow np * 0.3$
3: $x \leftarrow 0$
4: $y \leftarrow 0$
5: $foundBetter \leftarrow False$
6: **for all** $o \in CLuster.Objects$ **do**
7: $x \leftarrow x + o.x$
8: $y \leftarrow y + o.y$
9: **end for**
10: $c \leftarrow Centroid(\frac{x}{np}, \frac{y}{np})$ {Constrói o centroide.}
11: **for all** $o \in CLuster.Objects$ **do**
12: Map.Insere(D(o,c), o) {D(o,c) é a distância de o até c}
13: **end for**
14: **for all** $j \in Map.first(k)$ **do**
15: **if** D(j, Cluster.Objects()-j) ¡ MinDist **then**
16: MinDist = d(j, Cluster.Objects()-j)
17: Cluster.center = j
18: $foundBetter \leftarrow True$
19: **end if**
20: **end for**
21: **return** foundBetter

A estrutura Map utilizada no algoritmo tem o funcionamento semelhante a uma tabela hash com o diferencial de manter as chaves ordenadas (crescentente) e permitir inserir chaves duplicadas. Essa estrutura facilita a obtenção dos indivíduos mais próximos ao centroide calculado.

3.1 CCP

Para o problema de agrupamento capacitado (CCP), apresentamos 5 heurísticas:

3.1.1 Farthest

Osman e Christofides em (OSMAN; CHRISTOFIDES, 1994) apresentaram essa heurística construtiva com base no afastamento inicial dos primeiros centros (mais afastados entre si) e após um recálculo dos centros de cada agrupamento. Essa é uma heurística muito rápida mas que, segundo os autores, pode não encontrar soluções factíveis para o caso de a soma de todas demandas seja muito próxima das capacidades de todos agrupamentos. A heurística pode ser vista no algoritm 2

ALGORITMO 2 Farthest

```
1:  dist ← 0
2:  for all i ∈ I do
3:      for all j ∈ I do
4:          if d(i, j) > dist then
5:              c1 ← i
6:              c2 ← j
7:              dist ← d(i, j)
8:          end if
9:      end for
10: end for
11: C ← i, j
12: while | C |< p do
13:     dist ← 0
14:     for all i ∈ {I − C} do
15:         tmpDist ← 1
16:         for all j ∈ C do
17:             tmpDist ← tmpDist · d(i, j)
18:         end for
19:         if tmpDist > dist then
20:             dist ← tmpDist
21:             nC ← i
22:         end if
23:     end for
24: end while
25: for all i ∈ {I − C} do
26:     tmpDist ← ∞
27:     for all j ∈ C do
28:         if d(i, j) < tmpDist and j.CapacidadeRestante ≥ i.Demanda then
29:             c ← j
30:             tmpDist ← d(i, j)
31:         end if
32:     end for
33:     c.adiciona(i) {Associa i ao agrupamento mais próximo}
34: end for
```

Essa heurística tem como objetivo a construção rápida de uma solução, sendo baseada em

escolhas gulosas e não iterativas. Como as soluções geradas são de baixa qualidade, para se obter melhores soluções deve-se empregar heurísticas de busca na vizinhança da solução obtida com essa heurística construtiva. Em (OSMAN; CHRISTOFIDES, 1994) é apresentado um método híbrido que combina Busca Tabu e Simulated Annealing.

3.1.2 Density

Para resolver um problema de agrupamento, uma abordagem possível pode ser aquela que relaciona a densidade dos pontos para a escolha dos centros, assim a distância entre esse centro candidato e os possíveis pontos componentes será minimizada. Essa abordagem tem caráter guloso e é míope por escolher os pontos candidatos a centros analisando apenas sua densidade sem prever o rumo que essa solução está tomando. Para corrigir isso, é apresentado por (AHMADI; OSMAN, 2004) um método baseado na ideia de densidade dos pontos, mas que utiliza de aspectos de computação adaptativa com uma construção-desconstrução periódica. Essa meta-heurística construtiva apresenta soluções muito boas.

Para a etapa construtiva da solução, são utilizados os seguintes métodos, tendo o início pelo algoritmo ProcedimentoPrincipal 5

ALGORITMO 3 EncontraVizinhos(k)

```
1:  maxV ← n/p
2:  V ← ∅
3:  demAcum ← 0
4:  while demAcum < k.Cap and | V |≤ maxV  do
5:     for all i ∈ I do
6:        if ¬i.estaAssociado() and i ∉ V then
7:           if i.Dem + demAcum ≤ k.Cap  then
8:              V ← V ∪ i
9:              demAcum ← demAcum + i.Dem
10:          end if
11:       end if
12:    end for
13: end while
14: return  V
```

EncontraVizinhos(k) (algoritmo 3): Encontra os $m_k$vizinhos mais próximos de ponto k, sendo que $m_k \leq \frac{n}{p}$ e todos os pontos Y_k (conjunto de pontos vizinhos do centro k) tem a soma de suas demandas inferior ou igual a capacidade do agrupamento k.

CalculaDensidade(k, Y): Com o conjunto de pontos vizinhos do centro k, conseguimos calcular a densidade do centro através de

$$D_k = \frac{m_k}{T(a_k, Y)}$$

sendo $T(a_i, Y)$ a função que devolve a soma das distâncias entre o ponto a_i até todos os pontos de Y.

CalculaArrependimento(k, C): Após definir os pontos candidatos a centros (C), durante a fase iterativa é necessário definir os pontos que serão atribuídos a cada um dos agrupamentos. Para priorizar a inserção dos indivíduos é utilizada essa função para calcular o arrependimento de associar um nó com um centro que não seja o mais próximo. Para um ponto i, tendo os pontos j_1 *e* j_2 como o centro mais próximo a i e segundo mais próximo, respectivamente com capacidade disponível para atender a demanda de i, calculamos o arrependimento de associar i com um centro que não seja j_1 como sendo:

$$R_i = d_{ij_2} - d_{ij_1}$$

Ao final do método, o centro mais próximo a i é retornado.

EncontreOsMelhoresAgrupamentos (C, A): tendo A como um conjunto de pontos e C o conjunto de centros, essa função tenta associar os pontos em A com os agrupamentos em C e está descrita no algoritmo 4. Esse método faz uso também do algoritmo 1 para recalcular as medianas.

Com as funções definidas, um procedimento principal definido no algoritmo 5 é chamado e apresenta ao final o conjunto dos agrupamentos C. De modo iterativo, esse método vai construindo um agrupamento por vez e recalculando a densidade (D) do conjunto.

3.1.3 H-Means

A ideia básica desse algoritmo é partir de um conjunto de medianas, escolhidas de maneira aleatória e alocar cada um dos pontos ao agrupamento de mediana mais próxima a ele e, ao final dessa etapa, para cada agrupamento $j = 1, \cdots, p$ encontrar um ponto k para ser a nova mediana do agrupamento j de modo que

$$\sum_{i \in P_j} c_{i,k} = \text{MIN}_{i \in P_j} \sum_{i \in P_j} c_{i,k}$$

Move pontos para agrupamentos, depois escolhe os melhores centros. Desassocia os pontos dos agrupamento e associa para os centros mais próximos. O método pode ser visto no

ALGORITMO 4 EncontreOsMelhoresAgrupamentos(C,A)

```
t = 1;
Minor ← MAX_Iter
changed ← True
while changed = True and t < Minor do
  t ← t + 1
  chaged ← False
  while A ≠ ∅ do
    maxA ← 0
    for all i ∈ A do
      nC ← CalculaArrependimento(i, C)
      if i.Arrependimento ¿ maxA then
        maxA ← i.Arrependimento
        greatA ← i
        nearCenter ← nC
      end if
    end for
    nearCenter.associa(greatA)
  end while
  for all c ∈ C do
    if FindBestCenterOfCluster(c) then
      changed ← True
    end if
  end for
end while
```

ALGORITMO 5 ProcedimentoPrincipal

```
X ← I
Y ← ∅ {conjunto de pontos já atribuídos}
k = 0;
while k < p do
  k ← k + 1
  for all i ∈ X do
    V ← EncontraVizinhos(i)
    CalculaDensidade(i, V)
  end for
  newC ← FindMaxDensity(X) {Encontra o ponto com maior densidade.}
  C_k ← newC
  Y ← Y ∪ EncontraVizinhos(newC)
  X ← X \ Y
  if k ≥ 2 then
    EncontreOsMelhoresAgrupamentos(C, Y)
  end if
end while
EncontreOsMelhoresAgrupamentos(C, I)
```

algoritmo 6

ALGORITMO 6 H-Means

```
C ← RandonPick(p) {Seleciona p pontos aleatóriamente}
changed ← True
while changed do
  changed ← False
  for all i ∈ I do
    allocToNearst(i,C) {Associa i à mediana (C) mais próxima.}
  end for
  for all c ∈ C do
    if FindBestCenterOfCluster(c) then
      chaged ← True
    end if
  end for
end while
```

3.1.4 J-Means

Essa heurística proposta por (HANSEN; MLADENOVIĆ, 2002) utiliza a ideia de indivíduos não ocupados (indivíduos que se encontram afastadas da mediana do agrupamento por uma certa tolerância), para criar um novo agrupamento em alguma desses indivíduos que substitua algum dos agrupamentos existentes, de modo a diminuir o valor da função objetivo.

A escolha das partições inicias é feita de forma aleatória, selecionando as medianas e para todos indivíduos, atribua-os ao agrupamento mais próximo com capacidade disponível para atender a demanda do indivíduo.

Uma comparação entre as soluções geradas pelas heurísticas pode ser visto na figura 3.1. A Instância é a mesma aplicada em ambos casos. Como podemos ver, a heurística de densidade apresenta um solução melhor, apesar do tempo de processamento ser muito alto pela complexidade do algoritmo.

3.1.5 Random Density

A heurística proposta por (AHMADI; OSMAN, 2004) produz soluções muito boas, mas na grande maioria das vezes cai em um mínimo local, tanto que o refinamento da solução obtida apresentado no mesmo artigo é através da alteração dos dados de entrada, pela perturbação de alguns indivíduos e recalculando os novos agrupamentos. Com esses novos agrupamentos é recalculado a função objetivo levando em consideração o posicionamento original dos indivíduos. A noção de densidade utilizada por Ahmadi dá uma boa ideia da provável localização

ALGORITMO 7 J-Means

```
1:  C ← RandonPick(p)
2:  for all i ∈ I do
3:     allocToNearst(i,C)
4:  end for
5:  repeat
6:     optC ← C
7:     unocu ← ∅
8:     for all i ∈ I do
9:        if FarFromMedian(i, ε) then
10:          unocu ← unocu ∪ i
11:       end if
12:    end for
13:    rmC ← 0
14:    newC ← 0
15:    reduction ← 0;
16:    for all u ∈ unocu do
17:       for all c ∈ C do
18:          C ← C \ c
19:          C ← C ∪ u
20:          for all i ∈ I do
21:             allocToNearst(i,C)
22:          end for
23:          if (f(optC) - f(C)) ¿ reduction  then
24:             reduction ← f(optC) − f(C))
25:             rmC ← c
26:             newC ← u
27:          end if
28:       end for
29:    end for
30:    if newC ≠ 0 then
31:       C ← C \ rmC
32:       C ← C ∪ newC
33:       for all i ∈ I do
34:          allocToNearst(i,C)
35:       end for
36:    end if
37: until f(C) ¡ f(optC)
38: return  optC
```

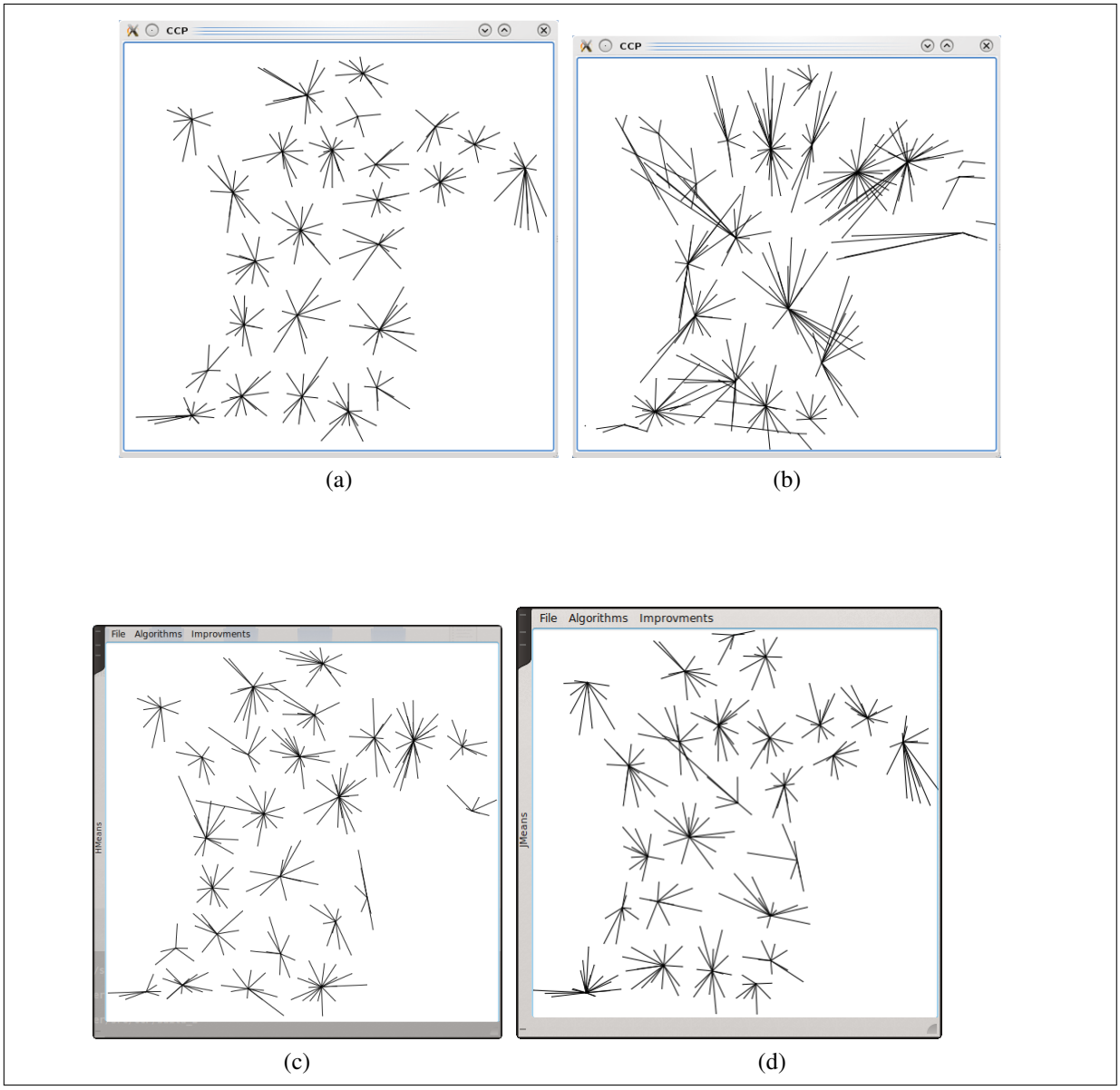


FIGURA 3.1: A solução gerada pelas heurística Density (a), Farthest (b), HMeans(c) e JMeans(d). Para todas soluções foi calculado com o mesmo valor de p (número de agrupamentos).

dos indivíduos candidatos a centros de agrupamentos. Com essa concepção e a técnica construtiva/desconstrutiva também representado no mesmo trabalho, criamos o algoritmo 8:

Note que agora na linha 10 do algoritmo 8 não é mais selecionado o indivíduo com maior densidade, mas sim um dos pontos com maior densidade, retirando o movimento guloso do algoritmo original que acabava por conduzir sempre a uma mesma solução. Com essa pequena alteração conseguímos soluções muito boas (na média melhores que os demais métodos na maioria das instâncias) e também que permitem uma aplicação de busca local para melhorar o resultado obtido. Em contrapartida, o tempo de processamento computacional aumentou. Isso se deve pela necessidade de maior número de iterações no procedimento $EncontreOsMelhoresAgrupamentos(C,Y)$ (4), agora que as escolhas de indivíduos como cen-

ALGORITMO 8 RandomDensity

1: $X \leftarrow I$
2: $Y \leftarrow \emptyset$ {conjunto de pontos já atribuídos}
3: $k = 0$;
4: **while** $k < p$ **do**
5: $k \leftarrow k + 1$
6: **for all** $i \in X$ **do**
7: $V \leftarrow EncontraVizinhos(i)$
8: CalculaDensidade(i, V)
9: **end for**
10: $newC \leftarrow GetRandonDensity(X)$ {Seleciona um dos p pontos com maior densidade.}
11: $C_k \leftarrow newC$
12: $Y \leftarrow Y \cup EncontraVizinhos(newC)$
13: $X \leftarrow X \setminus Y$
14: **if** $k \geq 2$ **then**
15: $EncontreOsMelhoresAgrupamentos(C, Y)$
16: **end if**
17: **end while**
18: $EncontreOsMelhoresAgrupamentos(C, I)$

tros pode gerar uma maior necessidade de movimentação dos indivíduos entre os agrupamentos na fase de construção/desconstrução.

Uma forma de obter soluções boas é após gera-las com alguma heurística, aplicar algum método de busca local, procurando na solução obtida uma solução melhor na vizinhança da solução atual.

3.2 Busca Local

Usando os movimentos descritos por (OSMAN; CHRISTOFIDES, 1994) de *interchange* (intercâmbio) e *shift* (mudança) temos os mecanismos de geração de soluções vizinhas à solução atual. Ambos movimentos respeitam a capacidade de cada agrupamento, não inserindo um indivíduo em um agrupamento que ultrapasse sua capacidade.

O movimento *shift* é uma troca simples onde apenas é removido um indivíduo i de um agrupamento C e o mesmo indivíduo é inserido em um outro agrupamento C', sendo $C \neq C'$ e C' possui capacidade suficiente para atender a demanda de i. Esse movimento não permite grande exploração da vizinhança, uma vez que para executá-lo é necessário que exista uma capacidade ociosa nos agrupamentos para receber novos pontos. Na figura 3.2 é apresentado um exemplo de movimento *shift* executado em uma solução válida do problema.

O movimento de *interchange*, visa trocar um indivíduo i' de um agrupamento C' por um

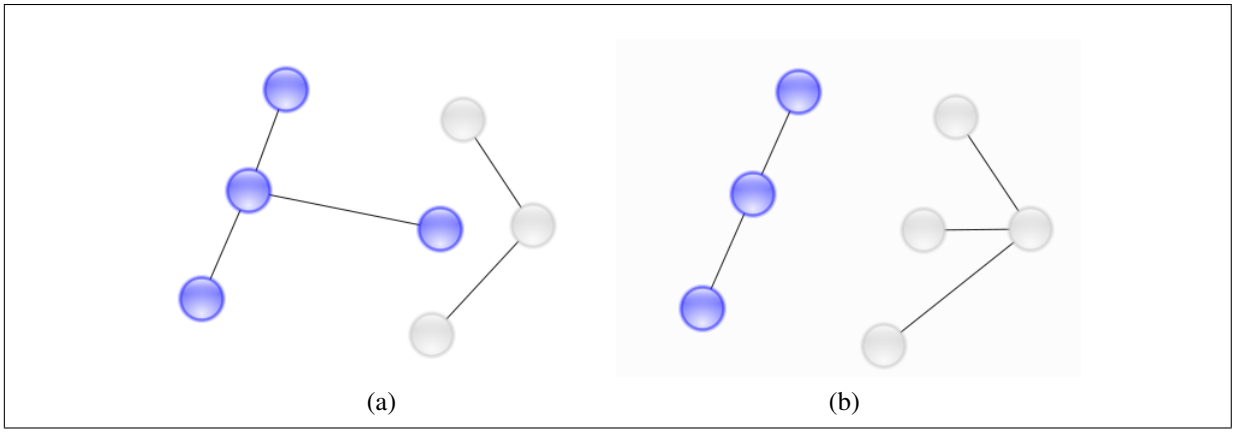


FIGURA 3.2: Em 3.1a temos uma solução com 2 agrupamentos e em 3.1b uma solução vizinha após movimento de *shift*.

outro indivíduo i de um outro agrupamento C, sendo $i \neq i'$ e $C \neq C'$. Esse movimento permite que sejam encontradas novas soluções vizinhas a atual mesmo quando os agrupamentos se encontram com demanda alta (pouca capacidade disponível), pois a troca é feita em um único momento sem que necessariamente um agrupamento deva ter capacidade suficiente para receber um ponto extra. Nas figuras em 3.3, pode ser visto o movimento *interchange*, sendo em (a) os agrupamentos existentes e que permitem a troca dos pontos entre si pelo movimento *interchange* e em (b) a solução obtida pelo movimento.

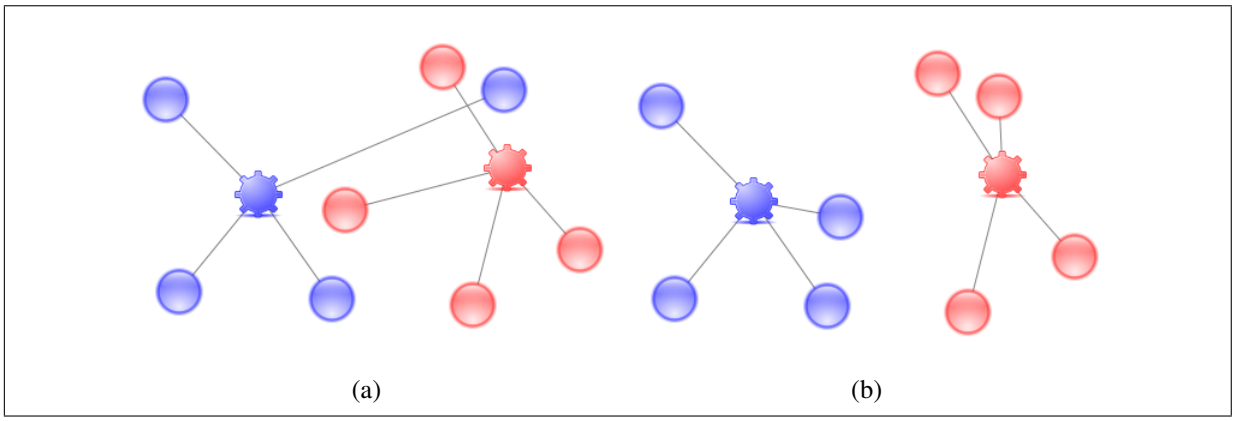


FIGURA 3.3: Em 3.1a temos uma solução com 2 agrupamentos e em 3.1b uma solução vizinha após movimento de *interchange*.

Ao final de cada movimento, as medianas são recalculadas de modo a minimizar a função objetivo do problema.

Para busca local, foram implementados 2 algoritmos, ambos usando uma busca gulosa na vizinhança por soluções com valores da função objetivo menores e cada um deles usando um dos movimentos acima descritos.

Para o algoritmo com o movimento *shift*, chamado internamente na ferramenta computa-

cional ShiftHillClimb, ele faz uma busca completa de todas possibilidades de mudança do indivíduo do agrupamento atual para os demais, efetuando a troca do indivíduo que dê um maior ganho para a função objetivo. Ele segue o algoritmo apresentado em 9. Esse algoritmo é executados iterativamente até que não existam soluções vizinha com uma função objetivo menor (um mínimo local) e tem complexidade de $O(n \cdot p)$ para cada iteração.

ALGORITMO 9 ShiftHillClimb

1: $MelhorSol \leftarrow AtualSol$
2: **repeat**
3: $NovaSol \leftarrow MelhorSol$
4: **for all** $i \in I$ **do**
5: Procura a melhor troca de i para um dos demais p-1 agrupamentos
6: **end for**
7: $NovaSol \leftarrow melhormovimentodeShift$
8: **until** NovaSol $<$ MelhorSol

O algoritmo de busca local implementado usando o movimento de *interchange*, originalmente tentava executar a troca de um ponto com todos os pontos dos demais agrupamentos que não o seu. Isso o deixava com uma complexidade $O(n \cdot (p-1) \cdot \frac{n}{p})$, algo muito próximo a $O(n^2)$ para cada iteração.Para reduzir o espaço de busca do algoritmo, uma vez que muitos dos intercâmbios não geram soluções melhores, implementamos o algoritmo 10.

ALGORITMO 10 InterchangeHillClimb

1: $MelhorSol \leftarrow AtualSol$
2: $Q \leftarrow \lceil p * 0.3 \rceil$
3: $K \leftarrow \lfloor \frac{n}{p} \cdot 0.5 \rfloor$
4: **repeat**
5: $NovaSol \leftarrow MelhorSol$
6: **for all** $i \in I$ **do**
7: **for all** $j \in K$ agrupamentos mais próximos de i **do**
8: $Procuradeicom Q$indivíduos $\in jmais$próximos
9: **end for**
10: **end for**
11: $NovaSol \leftarrow melhormovimentodeinterchange$
12: **until** NovaSol $<$ MelhorSol

Com essa metodologia, conseguimos reduzir o espaço de busca pela eliminação das soluções que não apresentariam reduções no valor da função objetivo. Uma rápida analise na complexidade $O(n \cdot Q \cdot K)$, onde Q = 30% dos agrupamentos ($p \cdot 0.3$) e K é 50% da média de pontos por agrupamento ficando uma complexidade muito próxima de $O(n)$ para cada iteração.

O número de iterações para cada um desses algoritmos é inversamente proporcional a qualidade da solução, uma vez que uma solução boa tem poucas, ou até mesmo nenhuma, soluções

com valor da função objetivo melhor que a sua, o que demanda menos passos no algoritmo para chegar em mínimo loca.

4 *Ferramenta Computacional*

Para melhor executar os estudos do problema, foi desenvolvida uma ferramenta computacional que provê a visualização dos resultados, permite programar execuções de algoritmos para arquivos abertos, oferecendo a opção de aplicar ou não os algoritmos de busca local na solução final, além de permitir rodar arquivos em lote. Os resultados são apresentado em uma saída de texto e salvos em um arquivo o que permite manter um histórico das execuções para cada arquivo.

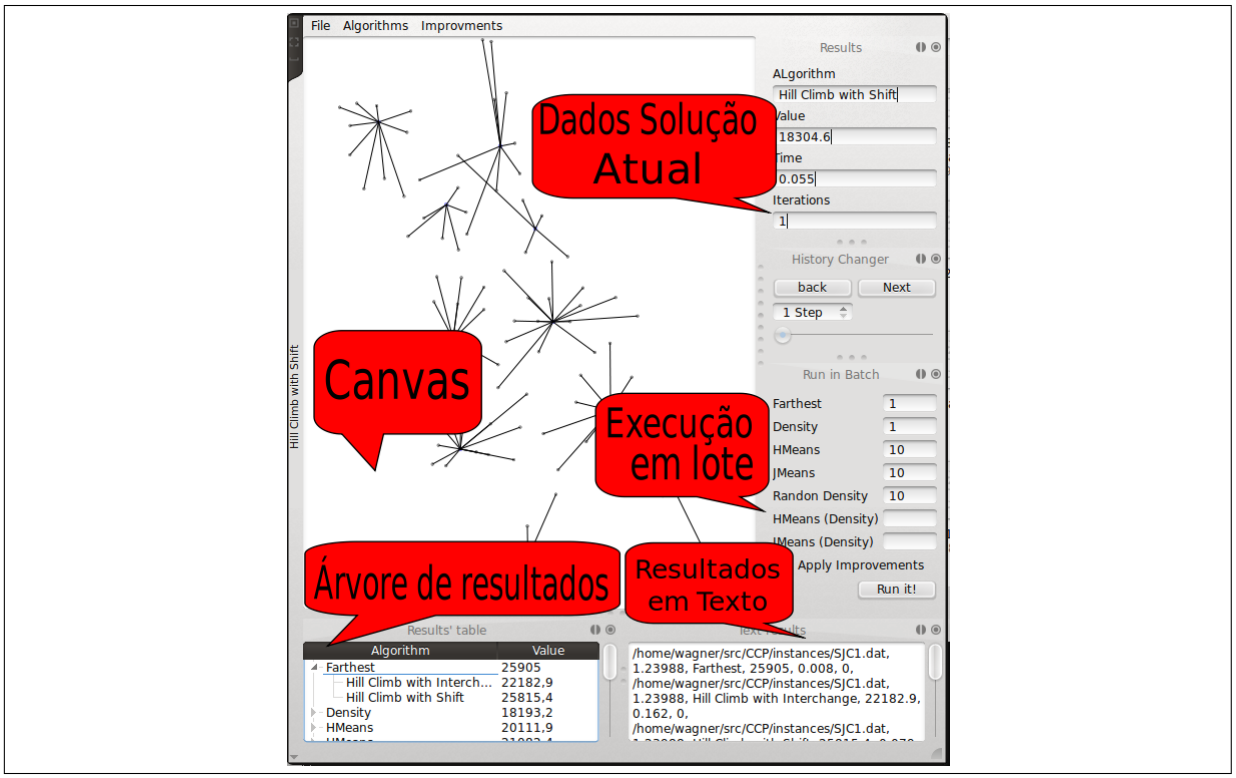


FIGURA 4.1: Tela principal da ferramenta desenvolvida

Quando o utilizador da aplicação selecional um método para ser executado (alguma das heurísticas, busca local ou ainda uma execução em lote) sobre um instância carregada, tais ações são armazenados em um fila e tem o processamento executado em uma thread (processo leve) separada, permitindo que a interface do utilizador não fique bloqueada enquanto o dados são processados, possibilitando a análise dos dados já processados.

Como os algoritmos de busca local podem ser aplicados nas soluções geradas por eles, foi implementada uma organização dos resultados derivados em formato de árvore. Essa árvore permite a navegação nela, mostrando na janela de visualização de resultados (canvas) os resultados nela selecionados.

Como alguns algoritmos tem a escolha das medianas iniciais de forma aleatória, é possível indicar quantas vezes os algoritmos com essa propriedade sejam executados.

Para o desenvolvimento foi escolhida a linguagem de programação C++ fazendo uso do paradigma orientado a objetos e utilizado como compilador o GCC da GNU. A interface foi construída utilizando o framework Qt da Nokia, o qual serviu não apenas de base da interface, mas também para codificação dos casos de teste, comunicação entre objetos, processamento de arquivos e concorrência (threads). Para o gerenciamento de versões do projeto foi utilizado o git pelas características descentralizadas e facilidade de uso dessa ferramenta , tendo os códigos da nossa ferramenta disponibilizados no repositório http://github.com/wiglot/CCP.

Para efetuar a geração dos arquivos makefiles, utilizados para executar a compilação da ferramenta, foi utilizado o gerador de makefiles CMake pela facilidade de uso, poder no controle de decisões e por possibilitar a execução dos casos de teste de forma automatizada pelo pacote CTest, o qual faz parte do próprio CMake.

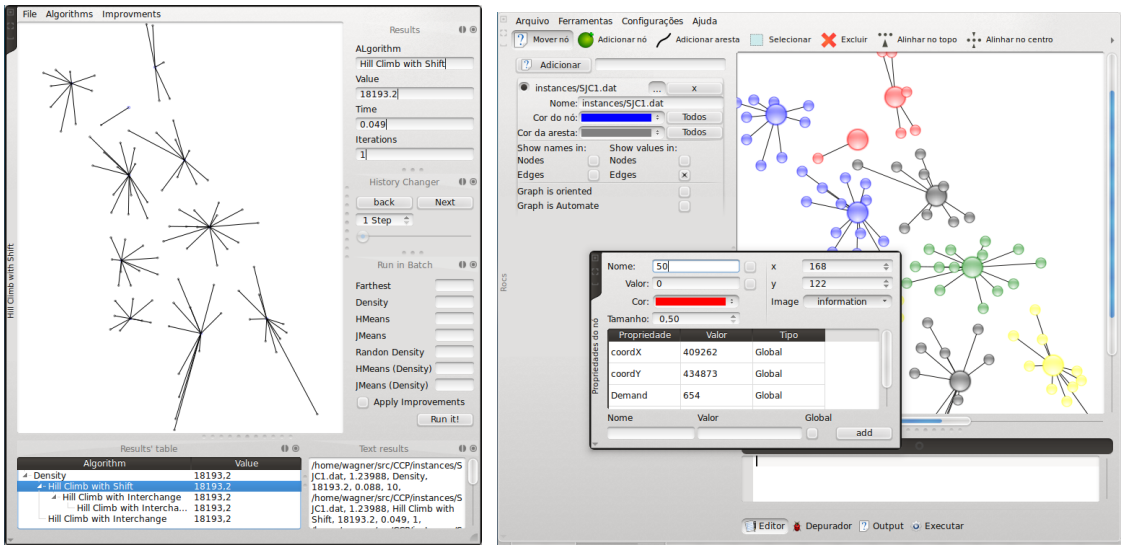
Como contribuição, também foi desenvolvido uma primeira versão de um plugin (biblioteca dinâmica) para o Rocs (IDE de estudo/pesquisa de algoritmos em grafos)(CANABRAVA; RECK, 2010) do KDE-EDU (conjunto de softwares educacionais do KDE), permitindo uma possível interação com a soluções geradas pelos algoritmos aqui tratados através de comandos na linguagem java script e também a criação de algoritmos para análise de tais soluções. Essa primeira versão do plugin permite a abertura de arquivos do problema CCP e a execução de algum dos algoritmos desenvolvidos aqui. Na figura 4.2 temos um exemplo das duas ferramentas com a mesma instância e o mesmo algoritmo rodado (density).

Tanto o plugin para o Rocs quanto a ferramenta desenvolvida como um todo, estão licenciados pela GPL, o que permite a futuros estudantes e pesquisadores utilizarem e modificarem a ferramenta conforme suas necessidades, não necessitando recriar todo um sistema do zero.

A ferramenta está dividida nas seguintes bibliotecas:

CCPAlgorithms: Algoritmos construtivos. Não inclui busca local nem o controle da execução dos algoritmos (concorrência)

CCPClusterView: Define as janelas e itens gráficos.



(a) (b)

Duas ferramentas, a desenvolvida nesse trabalho (a) e o Rocs (b) ambos com os mesmos algoritmos rodados sobre a mesma instância

FIGURA 4.2: Rocs e a ferramenta desenvolvida

CCPIOlib: Tratam da entrada de dados dos arquivos.

CCPModel: Representa o problema com todas estruturas básicas para representar uma instância a solução. Cuida também do enfileiramento dos pedidos de execução dos algoritmos e buscas locais.

A imagem 4.3 apresenta os modulos (bibliotecas) do sistema e a ligação entre eles e as imagens 4.4 e 4.5

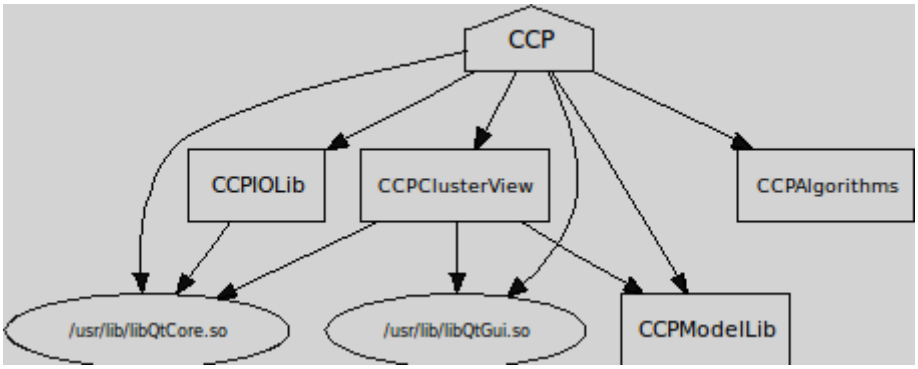


FIGURA 4.3: Bibliotecas que fazem parte do sistema.

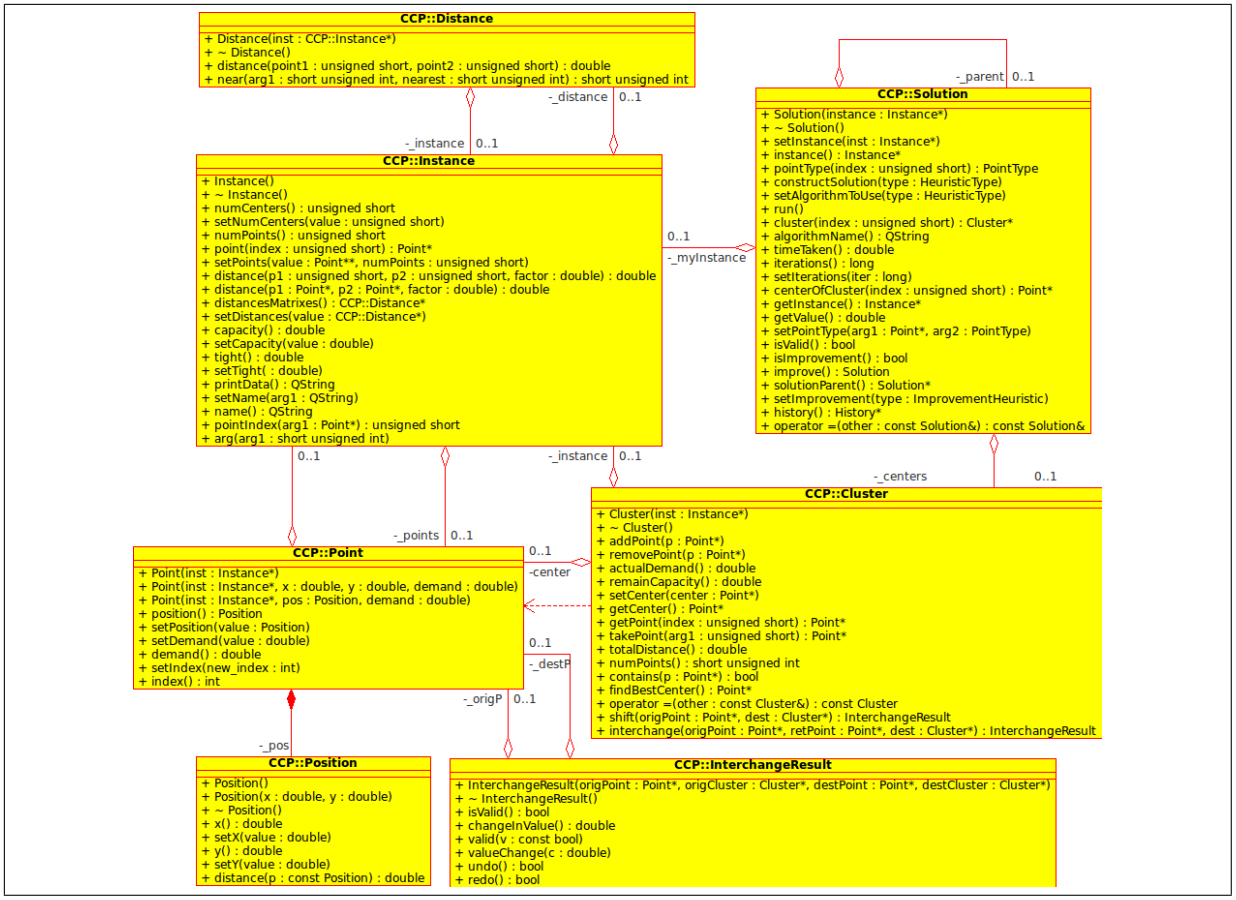


FIGURA 4.4: Classes que representam uma Instância e as soluções

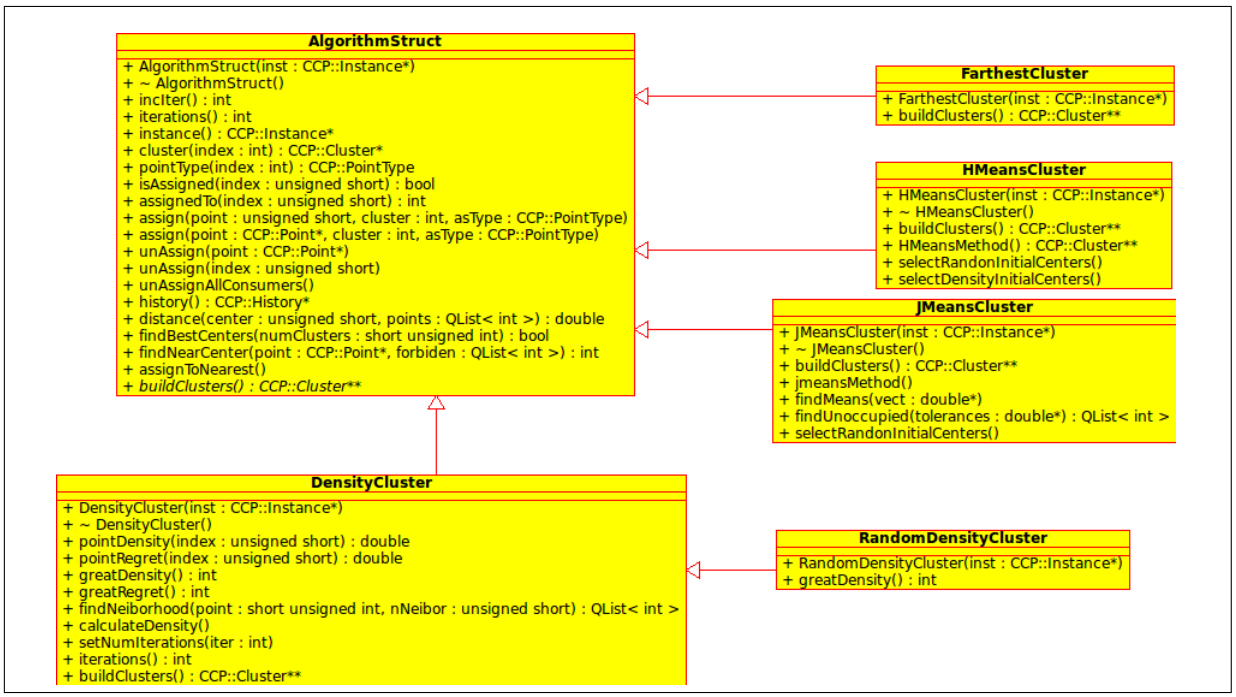


FIGURA 4.5: Classes dos algoritmos das heurísticas de construção

5 *Estudo das Características das Instâncias*

Chamamos de razão da soma das demanda por a capacidade total oferecida. Para comparar entre as diferentes instâncias, utilizamos a formula 5.1:

$$F = \frac{\sum_{p \in P} p_{cap}}{\sum_{i \in I} i_{dem}} \tag{5.1}$$

Onde i_{dem} é a demanda do indivíduo i e p_{cap} a capacidade do agrupamento p. F é maior que 1 se a demanda total é menor que a capacidade total. Para valores de F muito próximos a 1, mas maiores que 1, as heurísticas tendem a não conseguir uma solução factível, principalmente heurísticas que utilizam de procedimentos gulosos para alocação de indivíduos, como por exemplo a escolha do agrupamento com mediana mais próxima do indivíduo a ser alocado. Essa dificulade se dá uma vez que pode não ser possível associar todas demanda (OS) aos agrupamentos (equipes) uma vez que as demandas não podem ser fracionadas entre várias equipes. Para valores menores que 1 não existe uma solução válida para a instância, uma vez que existe maior demanda que oferta (capacidade).

Para o caso de $\sum_{i \in I} i_{dem} \geq p_{cap}, \forall p \in P$ podemos dizer que o problema é equivalente ao problemas das P-medianas não capacitado, uma vez que a capacidade não chega a ser uma restrição no problema e todos os indivíduos podem ser associados a apenas uma agrupamento.

Essa ultima colocação mostra o quanto o valor de folga pode interferir na geração de soluções das instâncias. Nas instâncias inéditas apresentadas aqui, modificamos a folga de modo a ser possível analisar e registrar o comportamento dos algoritmos e os valores da função objetivo das soluções obtidas. A tabela 5.1 mostra as demandas médias, bem com os valores mínimos e máximos, e a folga de cada uma das instâncias.

TABELA 5.1:
variação das demandas dos indivíduos das instâncias.

Instância	Folga	D_{Med}[1]	D_{Min}[2]	D_{Max}[3]
SJC1	1,23988	58,07	1	654
SJC2	1,35937	46,345	1	654
SJC3a	1,64371	37,5167	1	654
SJC3b	1,97246	37,5167	1	654
SJC4a	1,57658	39,7612	1	654
SJC4b	2,1021	39,7612	1	654
SJC5	1,05105	39,7612	1	654
AEAL_10	1,1	22,7321	3	52
AEAL_20	1,2	22,7321	3	52
AEAL_30	1,3	22,7321	3	52
AECP_10	1,1	22,2222	3	52
AECP_20	1,2	22,2222	3	52
AECP_30	1,3	22,2222	3	52
AELJ_10	1,1	22,6192	3	122
AELJ_20	1,2	22,6192	3	122
AELJ_30	1,3	22,6192	3	122
AENH_10	1,1	23,2514	3	122
AENH_20	1,2	23,2514	3	122
AENH_30	1,3	23,2514	3	122
AESM_10	1,1	25,0175	3	52
AESM_20	1,2	25,0175	3	52
AESM_30	1,3	25,0175	3	52

[1] Demanda média dos indivíduos

[2] Demanda mínima dos indivíduos

[3] Demanda máxima dos indivíduos

Para medir a dispersão dos indivíduos no espaço, Desenvolvemos um método que mede a distância dos pontos de centros de massa calculados usando como peso para cada ponto sua própria demanda. O algoritmo 11 produz como resultado a média das distâncias dos pontos aos centroides e as distâncias mínima e máxima. Para o primeiro centro de massa, todos os pontos são considerados. Já para os demais centros de massa, os pontos mais próximos ao

último centro de massa calculado que possuam a soma de suas demandas inferior ou igual a capacidade Cap, são removidos do cálculo do próximo centro de massa.

ALGORITMO 11 CalculaDispersão(I, Cap)

$S \leftarrow I$
$MinD \leftarrow \infty$
$MaxD \leftarrow 0$
$MedD \leftarrow 0$
$NumC \leftarrow o$
while $S \neq \emptyset$ **do**
  $C \leftarrow CentroDeMassa(S)$
  $V \leftarrow EncontraVizinhos(C, S, Cap)$
  for all $v \in V$ **do**
    $MinD \leftarrow Min(MinD, d(v,C))$
    $MaxD \leftarrow Max(MaxD, d(v,C))$
    $MedD \leftarrow MedD + d(v,C)$
  end for
  $NumC \leftarrow NumC + 1$
  $S \leftarrow S - V$
end while
$MedD \leftarrow MedD/NumC$

O método EncontraVizinhos, apresentado no algoritmo 12, procura os pontos mais próximos do centroide C dentre os pontos no conjunto S. Para isso é respeitada a capacidade Cap passada.

ALGORITMO 12 EncontraVizinhos(C,S, Cap)

$acum \leftarrow 0$
$k \leftarrow 0$
$list \leftarrow$
for all $i \in S$ **do**
  Map.insert(d(i,C), i)
end for
while $| Map | \neq 0$ **and** $Map[k].demanda + acum \leq Cap$ **do**
  $list \leftarrow Map[k] \cup list$
  $acum \leftarrow acum + Map[k].demanda$
  $k \leftarrow k + 1$
end while
return list

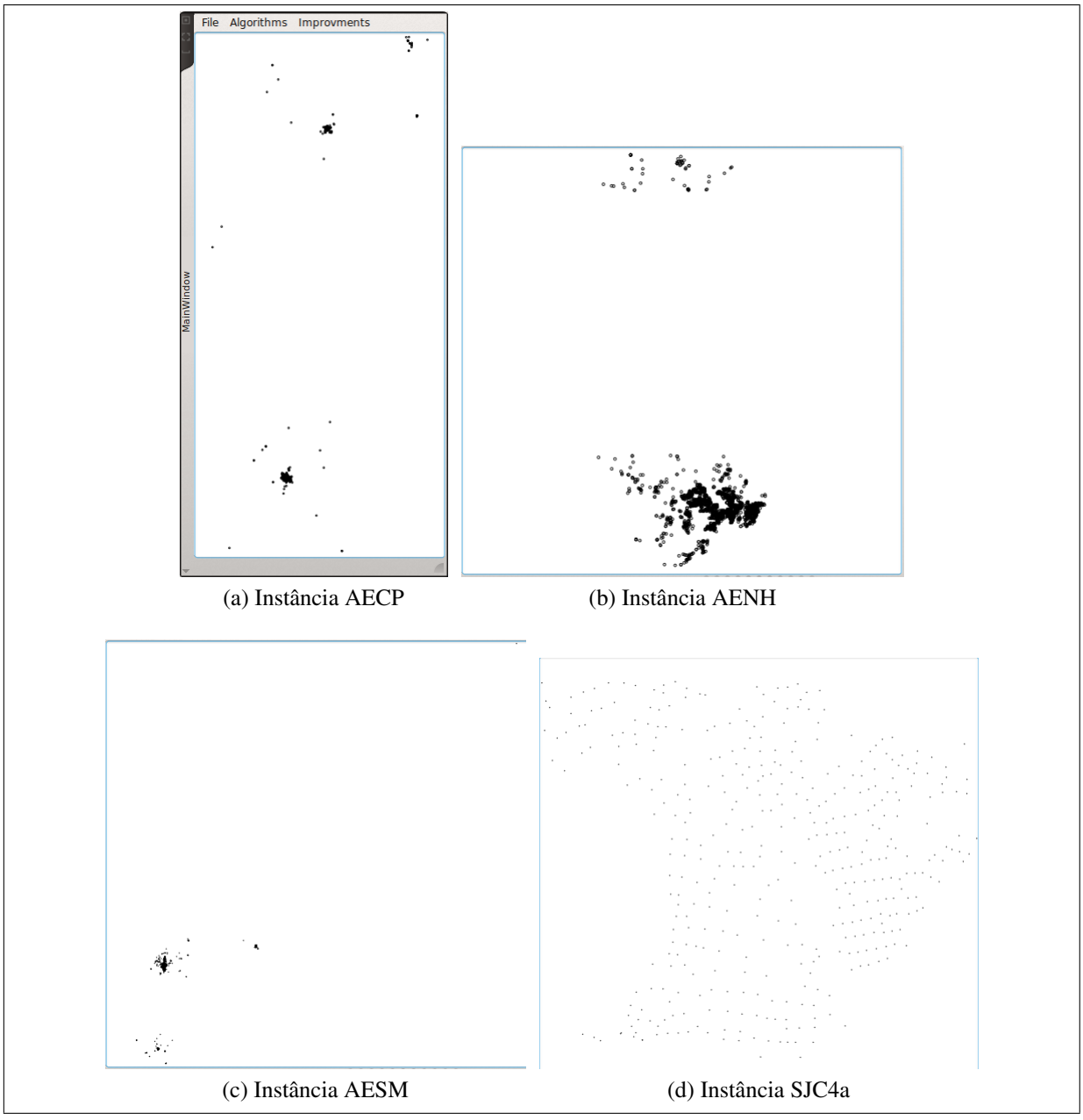
Para encontrar o centro de massa, as coordenadas (x e y) dos pontos pertencentes a S são somadas e multiplicadas pela sua demanda. O valor final é dividido pelo número de pontos em S e é criado um ponto com a localização resultante. O algoritmo que executa tal processamento é apresentado em 13.

ALGORITMO 13 CentroDeMassa(S)

$X \leftarrow 0$
$Y \leftarrow 0$
$dem \leftarrow 0$
for all $i \in S$ **do**
  $X \leftarrow X + i.x \cdot i.demanda$
  $Y \leftarrow Y + i.y \cdot i.demanda$
  $dem \leftarrow dem + i.demanda$
end for
return Ponto($\frac{X}{dem}, \frac{Y}{dem}$)

Antes de apresentar os resultados, separamos na figura 5.1 algumas instâncias onde é possível ver a localização dos pontos. Adiantando um pouco os resultados obtidos, temos a instância SJC4a como uma das que apresentou maior média de distância dos centros de massa para os indivíduos e como pode ser visto na figura 5.1d os indivíduos são distribuídos quase que uniformemente no espaço. Já a instância AECP, apesar de apresentar focos de concentração de pontos, existem vários desses focos o que leva os centros de massa serem calculados entre esses focos afastados das concentrações, e assim ter uma maior distância entre os pontos e os centros de massa. As duas instâncias com menores médias de distâncias foram AENH(5.1b) e AESM(5.1c) possuem apenas um grande foco de concentração dos pontos, sendo nesse caso onde se concentrará a maioria dos centros de massa calculados.

Aplicando o algoritmo sobre as instâncias, obtivemos os dados apresentados na tabela 5.2. Colocamos também o gráfico 5.2 dos dados para melhor visualização. Para comparar as dispersão entre as diferentes instâncias, os valores foram normalizados por $Max(d(i,k)),\ i,k \in I$, sendo 1 a maior distâncias entre 2 pontos de uma instância. O que nós tomamos como medida de dispersão é a medida média de distâncias, apesar da medida mínima e máxima indicar algumas peculiaridades das instâncias, como por exemplo a distância máxima muito alta e a distância média e distância mínima muito baixa pode indicar que os pontos se encontram agrupados bem próximos, mas que exitem pontos isolados (afastados) dessas concentrações.



(a) Instância AECP (b) Instância AENH

(c) Instância AESM (d) Instância SJC4a

FIGURA 5.1: Localização dos pontos de algumas instâncias

TABELA 5.2:
Dispersão dos indivíduos nas instâncias.

Instância	D_{min}[1]	D_{max}[2]	D_{med}[3]	CM[4]
SJC1	0,01348	0,5895	0,1937	10
SJC2	0,02145	0,7027	0,2725	12
SJC3a	0,00402	0,7217	0,2576	17
SJC3b	0,00402	0,7217	0,2576	17
SJC4a	0,02442	0,6849	0,2844	21
SJC4b	0,02442	0,6849	0,2844	21
SJC5	0,02442	0,6849	0,2844	21
AEAL_10	0,00627	0,5366	0,2048	14
AEAL_20	0,00620	0,5337	0,2004	13
AEAL_30	0,00621	0,6475	0,1945	12
AECP_10	0,04394	0,4484	0,2864	10
AECP_20	0,01667	0,5008	0,2689	9
AECP_30	0,14215	0,7061	0,2848	8
AELJ_10	0,01586	0,5326	0,1293	13
AELJ_20	0,01586	0,5513	0,1317	12
AELJ_30	0,01586	0,5227	0,1315	11
AENH_10	0,00690	0,5209	0,0873	16
AENH_20	0,00690	0,6348	0,0766	15
AENH_30	0,00690	0,6889	0,0821	14
AESM_10	0,00409	0,8907	0,0334	20
AESM_20	0,00409	0,9446	0,0333	18
AESM_30	0,00409	0,8907	0,0352	17

[1] Distância mínima dos indivíduos até seu o centro de massa

[2] Distância máxima dos indivíduos até seu o centro de massa

[3] Distância média dos indivíduos até seu o centro de massa

[4] Número de centros de massa calculados

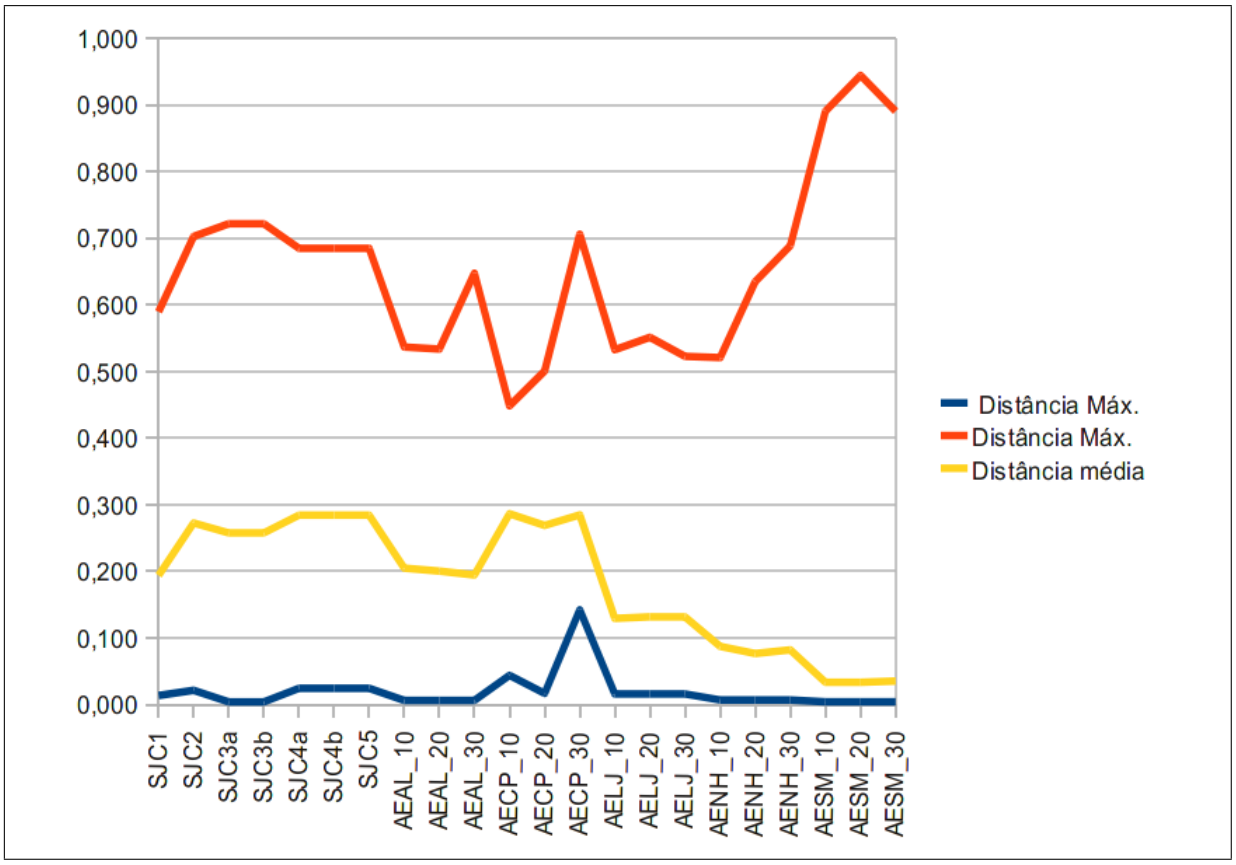


FIGURA 5.2: Dispersão dos indivíduos nas instâncias

6 *Resultados Computacionais*

Todos os testes foram executados em uma máquina AMD X4 de 3GHz com 2 Gigabytes de memória principal rodando o sistema operacional Linux 2.6. Para a execução dos testes foi definido que seriam executados 10 vezes os algoritmos que fazem uso de aleatoriedade para a escolha dos pontos iniciais, sendo utilizada a solução com menor valor de função objetivo para comparação com as demais heurísticas. Para mostrar a varição dos valores da solução, também é apresentado a média das soluções bem como o desvio padrão das execuções.

Foram escolhidos 2 conjuntos de instâncias para executar os testes. O primeiro é das instâncias disponibilizadas em http://www.lac.inpe.br/∼lorena/instancias.html criadas através de dados coletados da cidade de São José dos Campos. As instâncias desse primeiro conjunto possuem o prefixo SJC e são utilizadas para comparação dos valores obtidos por nós com valores apresentados na literatura.

O segundo conjunto de 5 instâncias, são instâncias reais inéditas obtidos de coleta de dados de atendimentos agendados para um certo dia em uma concessionária, sendo cada instância em uma determinada cidade aqui do estado. Desses dados possuíamos apenas o tempo necessário para cada atendimento e a localização de cada um deles, bem como o número de equipes que podem atende-los. Apesar de equipes de trabalho possuirem uma jornada fixa de trabalho por dia, foi preciso relaxar essa restrição pois a demanda dos atendimentos era maior que a capacidade das equipes. Isso ocorre devido a atendimentos de dias anteriores que são re-agendados para um novo dia por não terem sidos atendidos.

A definição das capacidades das equipes foi definida por a fórmula

$$Cap = \frac{F \cdot \sum_{i \in I} i_{dem}}{|P|}$$

onde F é a folga desejada e i_{dem} a demanda do indivíduo i. Cap será a capacidade de cada um dos agrupamentos. Temos com isso o mesmo valor de capacidade para todas equipes. Para mostrar o que influencia uma maior folga da capacidade total, esse último conjunto de instâncias teve as capacidades totais alteradas para 10%, 20% e 30% além da demanda total deles.

A tabela 6.1 apresenta os valores da função objetivo das soluções obtidas com os 5 algoritmos implementados. Foi colocado também os valores para n (número de indivíduos/OS) e p (número de agrupamentos/equipes). Como a heurística Farthest foi a que apresentou as soluções com maiores valores para todos os casos, deixamos apenas o valor dela e os valores percentuais dos demais métodos em relação a ela. Os valores apresentados são os valores mínimos das funções objetivo para as heurísticas que utilizam aleatoriedade. Como comparação, os gráficos 6.1 e 6.2 apresentam os valores percentuais que as soluções se encontram acima das melhores soluções encontradas para a instância. Para as instâncias AE, o gráfico foi cortado em 100%, uma vez que as soluções da heurística Farthest apresentaram valores que ficaram até 734% acima da melhor solução encontrada. Logo após, na tabela 6.2, apresentamos os tempos tomados para executar o processamento das soluções apresentadas como mínimas.

TABELA 6.1:
Valores da obtidos das instâncias pelos algoritmos construtivos.

Instância	n	p	Farthest	Density	H-Means	J-Means	Random Density
SJC1	100	10	25905,0	70,23%	74,00%	78,16%	68,30%
SJC2	200	15	52121,3	66,23%	67,50%	66,37%	65,51%
SJC3a	300	25	76345,3	62,84%	64,49%	63,06%	60,13%
SJC3b	300	30	69956,2	60,75%	60,91%	60,51%	60,25%
SJC4a	402	30	103171,0	63,64%	64,71%	65,52%	61,65%
SJC4b	402	40	92948,9	58,18%	61,42%	59,62%	58,57%
SJC5	402	20	134798,0	72,47%	73,82%	74,27%	64,04%
AESM_10	1042	21	96738,2	21,21%	16,00%	16,96%	15,77%
AESM_20			97798,8	16,84%	16,84%	14,06%	11,93%
AESM_30			69541,8	23,85%	19,42%	18,49%	16,72%
AECP_10	324	10	27877,4	23,13%	23,25%	24,56%	23,17%
AECP_20			9889,6	62,43%	59,66%	64,02%	59,00%
AECP_30			8374,6	73,50%	67,42%	66,97%	69,25%
AENH_10	2327	17	86174,4	28,70%	30,45%	30,24%	27,75%
AENH_20			73593,2	33,36%	33,78%	32,91%	31,02%
AENH_30			74153,2	34,77%	33,60%	37,06%	32,34%
AELJ_10	646	14	32398,0	28,32%	30,81%	29,95%	31,41%
AELJ_20			32248,7	27,28%	25,15%	25,69%	26,27%
AELJ_30			27915,8	31,29%	26,99%	27,39%	28,97%
AEAL_10	448	15	29700,4	36,53%	36,81%	37,76%	36,49%
AEAL_20			27219,8	39,54%	33,67%	34,30%	33,16%
AEAL_30			26814,7	40,10%	32,96%	33,66%	32,89%

Os tempos computacionais tomados por cada algoritmo para gerar as soluções apresentadas anteriormente, é apresentado na tabela 6.2.

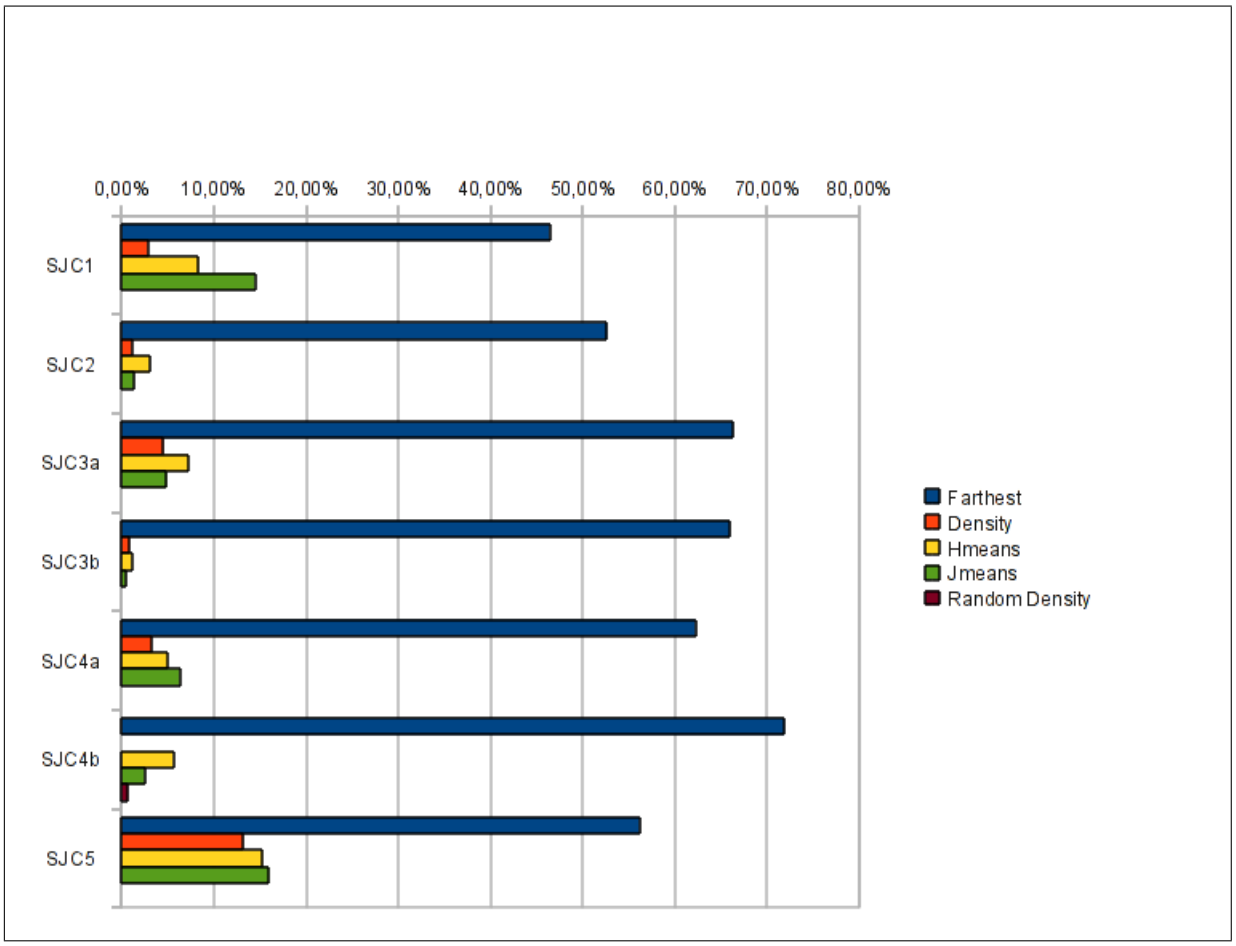


FIGURA 6.1: Percentual do valor acima da melhor solução encontrada para as instâncias SJC

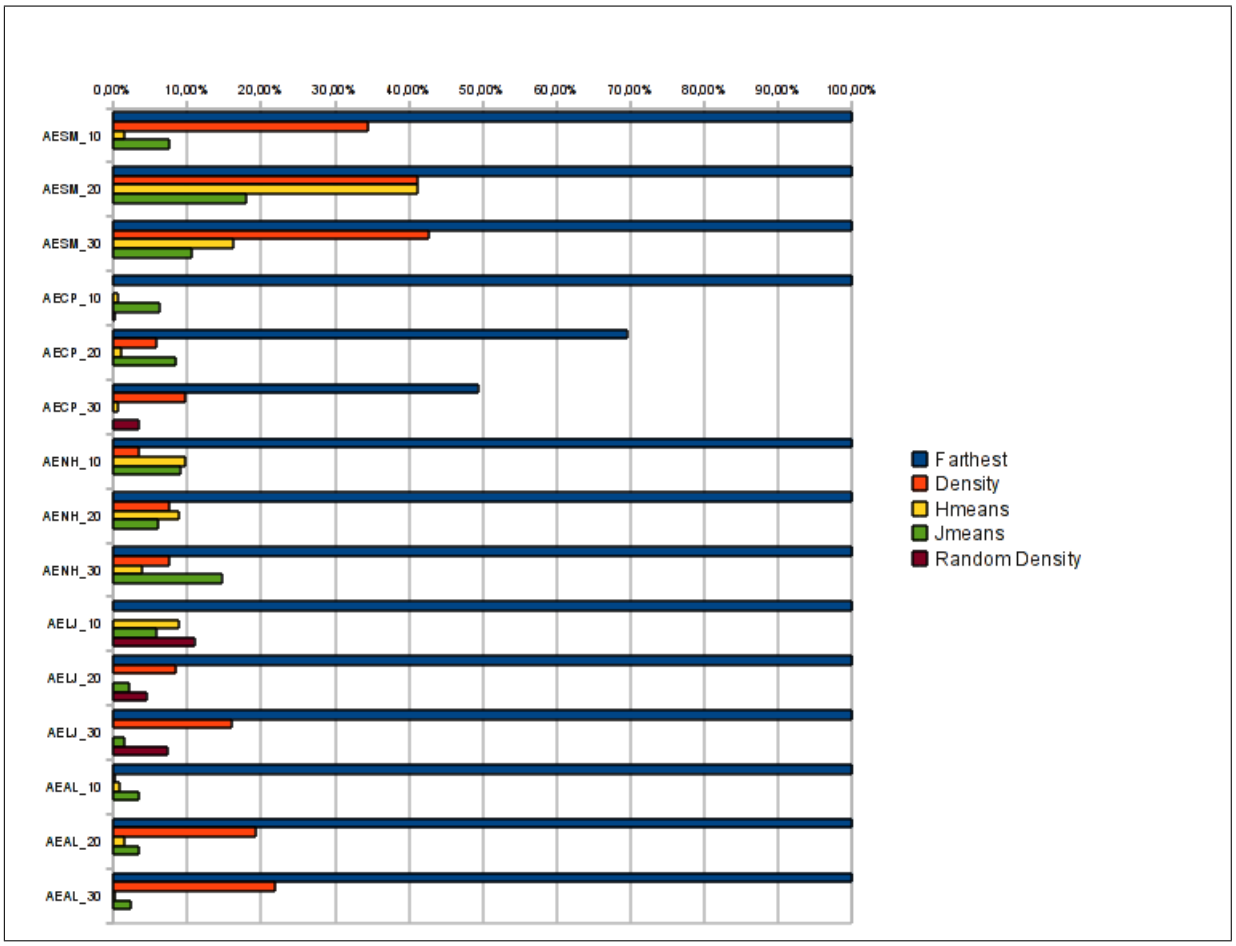


FIGURA 6.2: Percentual do valor acima da melhor solução encontrada para as instâncias AE

TABELA 6.2:
Tempo (em segundos) para cálculo das soluções.

Instância	Farthest	Density	H-Means	J-Means	Random Density
SJC1	0,009	0,085	0,002	0,177	0,092
SJC2	0,030	0,393	0,003	3,002	0,726
SJC3a	0,036	1,678	0,013	28,983	3,568
SJC3b	0,046	1,839	0,010	50,773	3,484
SJC4a	0,056	6,396	0,021	74,550	20,727
SJC4b	0,054	6,254	2,981	199,509	8,679
SJC5	0,050	5,110	0,026	49,322	6,070
AESM_10	0,87	108,90	0,19	482,74	358,33
AESM_20	0,08	53,41	0,16	530,03	266,70
AESM_30	0,08	57,73	0,15	375,48	141,06
AECP_10	0,07	2,06	0,02	3,04	2,36
AECP_20	0,01	1,79	0,01	2,53	3,08
AECP_30	0,01	1,39	0,01	0,86	2,24
AENH_10	4,14	458,44	1,21	4.857,36	1.407,63
AENH_20	0,26	461,44	1,34	3.838,05	1.572,15
AENH_30	0,26	361,01	0,76	3.281,53	961,22
AELJ_10	0,04	7,96	5,81	54,66	55,24
AELJ_20	0,03	10,57	6,18	82,39	25,52
AELJ_30	0,04	1,39	0,04	53,57	20,55
AEAL_10	0,18	13,04	0,03	10,59	13,66
AEAL_20	0,02	6,21	0,03	24,00	9,03
AEAL_30	0,03	4,40	0,03	19,47	15,70

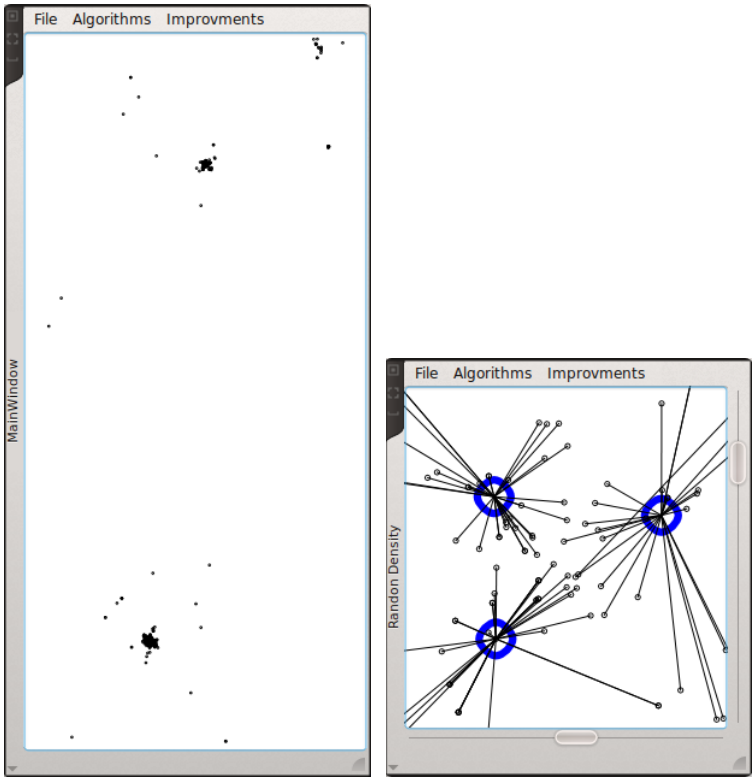
A heurística Random Density foi a que obteve os menores valores na maioria das instâncias, seguida da Heurística Density, seguida das heurísticas HMeans e JMeans que apresentaram soluções semelhantes. a Heurística farthest apresentou resultados nada bons em comparação com as demais, mas os tempos de processamento são bastante baixos, sendo uma heurística 'barata' caso se deseje apenas gerar uma solução inicialmente factível e trabalhar com a busca local, como Simulated Anneling e/ou Busca Tabu ou mesmo algoritmos genéticos. Esses valores altos se devem em parte pela escolha gulosa levando em consideração apenas a distância dos pontos para a seleção dos pontos iniciais e por executar apenas uma vez a alocação dos pontos e escolha dos melhores centros.

As heurísticas HMeans e JMeans apesar de escolherem aleatoriamente os indivíduos para serem os centros iniciais, o que pode gerar centros iniciais com localização pior que os escolhidos por outras heurísticas, e iterativamente vão melhorando a solução, escolhendo melhores centros e reassociando os pontos nos melhores centros até não encontrar uma solução melhor. Isso torna os métodos rápidos, no caso o HMeans, e com boas soluções sendo ainda possível uma pequena melhora no valor da função objetivo com uma busca local.

A Heurística Density, bem como sua derivada Random Density, utilizam do conceito de densidade, tentado localizar os melhores centros inicias e utilizam de métodos construtivos/desconstrutivos para a escolha das melhores associações dos indivíduos e dos indivíduos escolhidos como medianas. Apesar das melhores soluções serem dessas 2 heurísticas, os seus tempos de computação são na média maiores que os apresentados pelas heurísticas HMeans e Farthest. JMeans é um método que normalmente foi o meais lento, o que se deve pelo fato de testar todos os indivíduos que se encontram afastados das medianas como possíveis novos centros.

As soluções com aleatoriedade tem os desvios padrão apresentados na tabela 6.3. A Heurística Random density apresentou um menor desvio padrão para as instâncias AE, onde os pontos possuem uma distribuição agrupadada, ou seja ou pontos estão localizados em grupos e estes se encontram afastados uns dos outros.

A figura 6.3 mostra a distribuição dos pontos da instância (6.3a) e uma parte de uma solução do Random Density 6.3b, que mostra que as concentrações de pontos existentes nem sempre representam que ali estará apenas um agrupamento. Para as demais instâncias (SJC), a heurística Random Density apresentou devios padrões abaixo dos outros métodos que fazem uso de aleatoriedade, na maioria das instâncias. Na média os valores obtidos com essa mesma heurística, Random Density, apresentou os menores valores para todos os casos se comparados com os demais métodos que usam de aleatoriedade.



(a) (b)

FIGURA 6.3: Distribuição dos pontos na instância AECP 6.3a e uma parte de uma solução gerada pelo Random Density6.3b

TABELA 6.3:
Desvio padrão dos valores obtidos pelos algoritmos com aleatoriedade.

Instância	H-Means		J-Means		Random Density	
	Média	DP[1]	Média	DP	Média	DP
SJC1	20.066,6	619,3	21.214,0	966,1	18.430,9	653,4
SJC2	36.767,7	1.205,0	36.109,6	1.269,5	34.700,3	355,3
SJC3a	50.651,7	1.019,7	50.041,5	1.631,5	47.394,5	954,9
SJC3b	45.381,0	1.433,6	44.164,0	1.257,6	42.872,4	333,9
SJC4a	73.892,6	7.058,7	69.357,4	884,2	65.340,5	1.219,2
SJC4b	62.457,6	5.679,8	56.585,4	640,5	55.867,4	715,6
SJC5	102.325,8	1.894,5	109.244,7	9.669,6	91.377,1	2.616,6
AESM_10	22.434,58	5.723,65	21.166,50	3.880,00	19.874,29	2.410,11
AESM_20	19.000,31	2.825,51	16.322,42	1.568,66	14.684,51	1.445,29
AESM_30	16.080,65	1.764,01	15.418,38	1.564,28	13.392,51	928,93
AECP_10	6.594,54	61,89	8.130,36	3.288,99	6.560,09	112,85
AECP_20	6.318,05	349,87	6.702,14	564,02	6.107,97	144,67
AECP_30	10.319,98	4.241,99	7.239,48	2.448,47	6.482,20	1.455,36
AENH_10	28.441,94	2.038,19	29.939,45	7.388,23	24.126,65	503,47
AENH_20	26.918,98	1.808,29	26.577,87	1.844,80	23.861,87	742,50
AENH_30	25.554,47	693,58	28.506,26	7.758,78	23.412,79	561,11
AELJ_10	14.735,16	4.248,58	11.158,81	1.699,12	10.782,29	372,57
AELJ_20	14.386,39	5.332,40	8.883,50	528,24	8.543,30	244,10
AELJ_30	14.186,13	5.082,51	8.491,72	601,71	8.138,73	70,00
AEAL_10	12.527,41	4.291,75	11.779,82	330,40	10.901,17	68,26
AEAL_20	12.438,82	4.874,71	10.504,24	792,56	9.812,93	732,16
AEAL_30	11.998,70	3.951,95	10.980,83	2.566,85	9.297,77	648,16

[1] Desvio Padrão

Tomando como base os resultados apresentados por Lorena e por negreiros para as instâncias SJC, apresentamos a tabela 6.4 onde temos os melhores resultado obtidos pelos trabalhos e por os métodos por nós implementados. Como pode ser visto, Lorena possui os melhores resultados, por causa disso, seus dados servirão de base para calcular a porcentagem que as demais soluções se encontram acima de suas soluções. É apresentado o gráfico 6.4 de quanto

as nossas soluções por estão acima das apresentadas na literatura.

TABELA 6.4:
Comparação dos resultados com a literatura.

	N[1]		L[2]		TCC[3]		N(%)		TCC(%)	
	V[a]	T[b]	V	T	V	T	V(%)	T(%)	V(%)	T(%)
SJC1	17288,9	0,02	17252,1	68,62	17693,9	0,09	0,21	0,029	2,56	0,13
SJC2	33370,2	0,02	33223,6	2083,92	34147,2	0,72	0,44	0,001	2,78	0,03
SJC3a	45335,1	0,08	45313,4	2604,92	45907,9	3,56	0,05	0,003	1,31	0,13
SJC3b	0	0	40634,9	867,68	42151,3	3,48	—	—	3,73	0,40
SJC4a	62026,9	0,08	61842,4	27717,11	63602,9	20,72	0,30	0,000	2,85	0,07
SJC4b	0	0	52396,5	4649,47	54077,1	6,25	—	—	3,21	0,13

[1] Negreiros (NEGREIROS; PALHANO, 2006)

[2] Lorena (LORENA; SENNE, 2003)

[3] Algoritmos implementados

[a] Valor da função objetivo, [b] Tempo gasto pelo algoritmo em segundos.

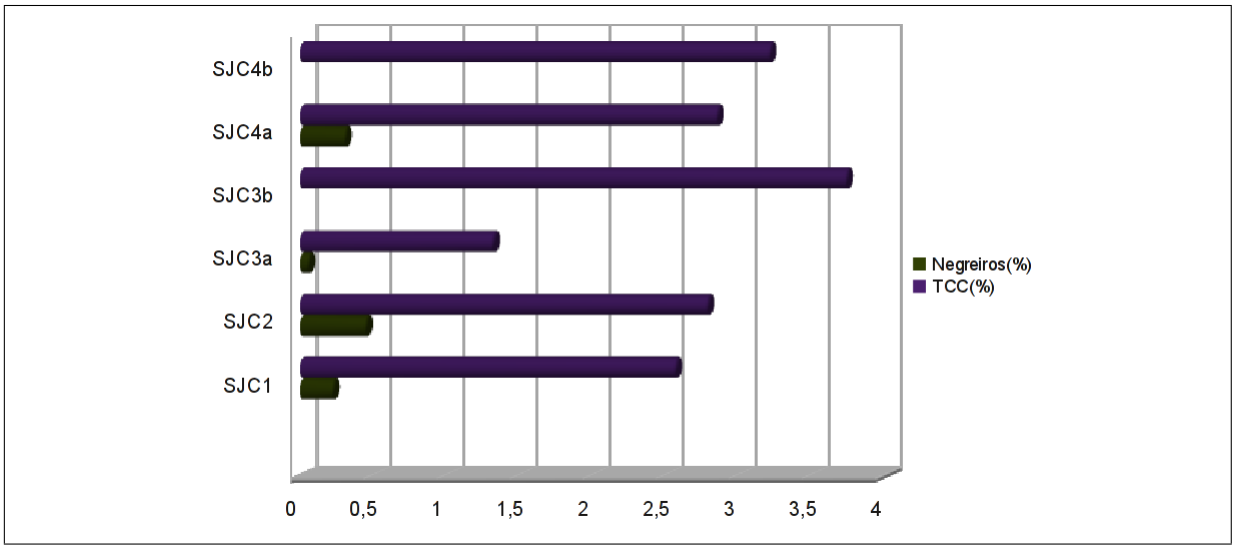


FIGURA 6.4: Diferença dos valores obtidos em relação a literatura

As buscas locais, apesar de utilizarem de métodos gulosos, ainda assim conseguiram melhorar várias soluções obtidas com os algoritmos construtivos implementados.

A tabela 6.5 apresenta os percentuais médios de redução nos valores das soluções, bem como os tempos médios também para execuçãos das buscas utilizando o movimento interchange

e na tabela 6.6 é apresentado os valores utilizando o movimento shift.

TABELA 6.5:
Redução na função objetivo com a busca local usando o movimento Interchange.

Instância	Farthest		Density		Hmeans		Jmeans		R. Density	
	RD[1]	T(s)[2]	RD[1]	T(s)[2]	RD[1]	T(s)[2]	RD[1]	T(s)[2]	RD[1]	T(s)[2]
SJC1	-13,99	0,06	0,00	0,01	-2,53	0,03	-2,70	0,04	-0,46	0,01
SJC2	-11,03	1,09	0,00	0,03	-2,25	0,22	-0,89	0,18	-0,01	0,04
SJC3a	-16,40	5,87	0,00	0,20	-1,60	0,66	-0,15	0,32	0,00	0,09
SJC3b	-10,11	3,38	0,00	0,20	-0,76	0,39	-0,03	0,17	0,00	0,11
SJC4a	-14,14	12,82	-0,37	0,83	-2,52	2,09	-0,73	1,28	-0,50	0,51
SJC4b	-13,95	11,80	0,00	0,35	-0,58	0,78	-0,25	0,54	-0,09	0,29
SJC5	-15,66	12,53	-4,42	3,02	-6,65	6,09	-16,90	11,21	-2,22	2,59
AECP_10	-21,65	16,76	-1,04	0,42	-2,63	3,64	-6,55	12,44	-3,00	1,69
AECP_20	-29,06	19,59	-0,03	0,43	-2,79	4,52	-3,31	7,89	-0,19	1,08
AECP_30	-14,46	18,82	-0,11	1,99	-4,27	5,16	-3,96	8,45	-0,31	1,02
AESM_10	-38,72	262,04	-3,00	269,41	-7,07	308,11	-8,92	306,89	-2,16	180,14
AESM_20	-40,67	320,24	0,00	7,69	-3,94	299,11	-2,65	274,58	-0,98	113,16
AESM_30	-36,67	347,72	-0,01	16,43	-6,07	255,42	-3,28	193,66	-0,44	62,82
AELJ_10	-16,75	80,42	-0,56	12,04	-8,08	71,73	-10,01	62,08	-8,63	62,53
AELJ_20	-21,01	90,19	0,00	1,14	-4,82	66,51	-2,98	39,94	-1,27	25,13
AELJ_30	-23,79	107,99	0,00	1,15	-4,96	126,78	-5,21	113,24	-2,12	55,95
AEAL_10	-39,14	35,03	-0,02	1,38	-4,96	126,78	-5,21	113,24	-2,12	55,95
AEAL_20	-9,27	26,70	0,00	0,29	-4,96	126,78	-5,21	113,24	-2,12	55,95
AEAL_30	-22,84	27,53	0,00	0,35	-5,19	139,09	-5,07	123,32	-2,03	61,37

[1] Redução no valor da função objetivo

[2] Tempo tomado pela busca local

O movimento shift mostra algum resultado sgnificante normalmente quando aplicado a instâncias com maior folga. Para as instâncias com menor folga, não existe muita capacidade ociosa nos agrupamentos para receber novos indivíduos, por isso sua redução é sempre menor

nesses casos.

TABELA 6.6:
Redução na função objetivo com a busca local usando o movimento Shift.

Instância	Farthest		Density		Hmeans		Jmeans		R. Density	
	RD[1]	T(s)[2]	RD[1]	T(s)[2]	RD[1]	T(s)[2]	RD[1]	T(s)[2]	RD[1]	T(s)[2]
SJC1	-0,35	0,09	0,00	0,05	-0,07	0,05	0,99	0,11	0,00	0,04
SJC2	0,00	0,25	0,00	0,24	0,00	0,30	0,00	0,29	0,00	0,20
SJC3a	0,00	0,70	-0,01	1,48	-1,60	0,66	0,10	1,06	-0,01	0,60
SJC3b	0,00	0,67	0,00	0,69	0,00	0,81	0,00	0,73	0,00	0,51
SJC4a	0,00	1,73	0,00	1,39	0,00	1,40	0,08	1,56	0,00	0,95
SJC4b	0,00	1,58	0,00	1,23	0,00	1,36	0,00	1,23	0,00	0,89
SJC5	0,00	0,70	0,00	0,88	-0,22	1,48	2,01	1,90	-0,01	0,79
AECP_10	0,00	0,20	-0,03	1,99	-0,05	0,56	-0,76	0,71	-0,04	0,65
AECP_20	0,00	0,25	0,00	0,79	-0,03	0,72	-1,35	0,85	-0,01	0,79
AECP_30	0,00	0,36	0,00	0,76	0,00	0,67	-0,03	0,65	-0,02	0,84
AESM_10	-4,14	5,01	-0,02	10,47	-0,04	7,54	-3,00	9,08	0,00	4,45
AESM_20	0,00	4,06	-0,56	36,45	-0,07	11,20	-0,04	11,41	0,00	9,70
AESM_30	0,00	5,85	0,00	15,48	-0,03	17,10	-0,26	15,10	0,00	13,91
AELJ_10	0,00	1,11	0,00	2,10	0,00	1,90	-0,05	2,65	0,00	1,90
AELJ_20	0,00	1,66	-0,01	11,49	0,00	2,60	-0,63	5,88	-0,02	4,68
AELJ_30	0,00	1,44	0,00	4,84	-0,03	5,29	-0,76	5,79	-0,01	4,61
AEAL_10	0,00	0,38	-0,06	3,40	-0,03	5,29	-0,76	5,79	-0,01	4,61
AEAL_20	0,00	0,49	-0,07	2,85	-0,03	5,29	-0,76	5,79	-0,01	4,61
AEAL_30	0,00	0,60	-0,08	7,86	-0,03	5,76	-0,76	6,30	-0,01	5,01
AENH_10	0,00	61,48	0,00	117,76	0,00	106,79	-0,74	188,35	0,00	100,62
AENH_20	0,00	93,98	0,00	190,92	-0,03	16,88	-0,78	25,61	-0,01	15,41
AENH_30	0,00	112,56	0,00	365,72	0,00	244,28	0,00	271,66	0,00	297,94

[1] Redução no valor da função objetivo

[2] Tempo tomado pela busca local

Considerações Finais

Neste trabalho foi apresento um CCP oriundo do estudo do PDOS, fazendo uso de métodos já existentes em um problema relativamente novo. Dos métodos tratados, procuramos abordar os métodos heurísticos mais tratados da literatura e compara-los quando aplicados aos dados do PDOS.

Dentre os métodos trabalhados, os que apresentaram melhores soluções foram o Random density e o Density, sendo o primeiro uma variação do segundo, com a diferença de inserir elementos de aleatoriedade para escolha dos centros, não desconsiderando o cálculo de densidade o qual nos dá um bom indicativo de onde devem ser inseridos os centros. Mas, apesar de os valores obtidos serem menores que os demais métodos, o seu tempo de computação é cerca de 3 vezes mais alto que o levado pelo método density, que para instâncias com grande número de indivíduos e de centros (por exemplo algumas instâncias com mais de 3.000 pontos e 600 agrupamentos), tem um tempo de computação significativo. Esse tempo elevado não desqualifica o uso do método para essa classe de problemas, visto que os atendimentos são processados por cidades como apresentado nas instâncias AE e possuem grande número de atendimentos e poucas equipes.

Dos demais métodos, Farthest é uma heurística construtiva que gera soluções de baixa qualidade mas rapidamente. HMeans gera soluções de qualidade boas de modo até mais rápida que o último. JMeans gera soluções de qualidade boa também, mas com a desvantagem de ser extremamente lento na fase de busca de novos centros e reassociação dos indivíduos. Essa etapa dessa heurística pode sofrer melhorias para acelerar como um todo o algoritmo, visto que essa é a etapa mais repetida durante as iterações. Como exemplo de melhoria seria a diminuição do número de centros removidos para a inserção de um novo centro para apenas os centros mais próximos a esse novo centro, evitando que se insira um centro onde já exista um grande número de centros.

As ferramentas computacionais também ficam como contribuição para futuros trabalhos ou de base para implementação de novos algoritmos fornecendo toda representação do problema bem como a visualização do mesmo para análise dos dados.

Para trabalhos futuros consideramos a utilização de métodos híbridos, onde métodos exatos

seriam utilizados apenas para resolver subproblemas de complexidade menor e utilização de buscas locais com espaço de busca ampliado pela relaxação de algumas restrições, gerando soluções infactíveis mas com valor da função objetivo melhor que serviria de base para uma nova busca por soluções factíveis em uma vizinhança que anteriormente não era alcançável ou que exigia grande refinamento na etapa de busca local. Outra consideração a ser levada para os trabalhos futuros é que, para aplicações em cenários reais, deve-se levar em consideração o deslocamento entre ordens de serviço, o que iria contribuir para o aumento do tempo (valor da demanda) para a equipe que está atendendo as ordens de serviço e a carga horária das equipes poder ser variável de uma para outra. Nesse caso teríamos um problema heterogêneo quanto a capacidade e também teríamos o problema de roteirização das OS, sendo que ao calcular uma rota pode acontecer de aumentar tempo, devido a um maior deslocamento entre os novos atendimentos, para um valor além da capacidade da equipe, gerando uma solução infactível.

A parte construtiva das heurísticas também permitem trabalhos futuros no que diz respeito à escolha de indivíduos como centros iniciais. A escolha com o cálculo por densidade se mostrou muito boa, mas se forem utilizados dados estatísticos é possível de se definir os centros com melhores caracteríscas. Para isso seria necessário a aplicação de algum método como o GRASP (MAURICIO; THOMAS, 1989) onde uma solução fornece dados para a geração de uma nova solução.

Referências Bibliográficas

AHMADI, S.; OSMAN, I. Density based problem space search for the capacitated clustering p-median problem. *Annals of Operations Research*, Springer, v. 131, n. 1, p. 21–43, 2004.

CANABRAVA, T.; RECK, W. *Rocs - Rocs Graph Theory*. Berlin, Germany, 2010. Disponível em: <http://edu.kde.org/rocs>.

FORGY, E. Cluster analysis of multivariate data: efficiency versus interpretability models. *Biometrics*, v. 61, n. 3, p. 768–769, 1965.

GAREY, M.; JOHNSON, D. *Computers and intractability*. [S.l.]: Freeman San Francisco, 1979.

HANSEN, P.; MLADENOVIĆ, N. *J-Means: A New Local Search Heuristic for Minimum Sum-of-Squares Clustering*. 2002.

JAIN, A.; MURTY, M.; FLYNN, P. Data clustering: a review. *ACM computing surveys (CSUR)*, ACM, v. 31, n. 3, p. 264–323, 1999.

LIU, G. *CL Liu,Introduction to combinatorial mathematics*. [S.l.]: New York, McGraw Hill, 1968.

LORENA, L.; FURTADO, J. Constructive genetic algorithm for clustering problems. *Evolutionary Computation*, MIT Press, v. 9, n. 3, p. 309–327, 2001.

LORENA, L.; SENNE, E. Local search heuristics for capacitated p-median problems. *Networks and Spatial Economics*, Springer, v. 3, n. 4, p. 407–419, 2003.

MACQUEEN, J. et al. *Some methods for classification and analysis of multivariate observations*. [S.l.]: WESTERN MANAGEMENT SCIENCE INST UNIV OF CALIFORNIA LOS ANGELES, 1966.

MAURICIO, G. F.; THOMAS, A. A probabilistic heuristic for a computationally difficult set covering problem* 1. *Operations research letters*, Elsevier, v. 8, n. 2, p. 67–71, 1989.

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

NEGREIROS, M.; PALHANO, A. The capacitated centred clustering problem. *Computers & operations research*, Elsevier Science, v. 33, n. 6, p. 1639–1663, 2006.

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

POTVIN, J.; GUERTIN, F. *The clustered travel salesman problem: A Genetic approach*. [S.l.: s.n.], 1996. 619–631 p.

TAILLARD, É. Heuristic methods for large centroid clustering problems. *Journal of Heuristics*, Springer, v. 9, n. 1, p. 51–73, 2003.