O Problema do Caixeiro Viajante

Filipe Barreto Diniz[1] and Pedro Mendonça Maia[2]

***Abstract*— O Problema do Caixeiro Viajante é um tradicional problema da literatura bastante explorado em diferentes aplicações da computação. O problema consiste em se encontrar a menor rota possível para se percorrer um conjunto de cidades sem repetir nenhuma delas retornando a inicial. Para este trabalho porém, será desenvolvida uma abordagem biobjetivo, ou seja, além de se preocupar em percorrer a menor distância possível, também será necessário se preocupar com o tempo.**

I. INTRODUÇÃO

O Problema do Caixeiro Viajante (PCV) trata-se de um problema NP-difícil, ou seja, que não pode ser resolvido em tempo polinomial, e que por sua facilidade de entendimento ao se tratar de algo muito corriqueiro, é muito utilizado para o estudo de aplicações em otimização. O problema tem em sua principal concepção a de encontrar uma rota na qual se possa percorrer todo um conjunto de cidades, retornando a origem, sem repetição e com o menor custo possível, assim como é mostrado na figura 1. Nesse trabalho uma variante do problema será explorada, no caso uma alternativa biobjetivo no qual se pretende minimizar a distância (em km) e o tempo (em horas).

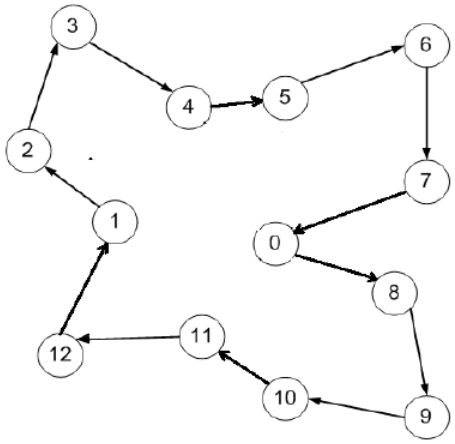

Fig. 1: Exemplo de rota do caixeiro viajante

A primeira vista esses objetivos podem não parecer muito contraditórios, pois ao se reduzir o tempo, a distância também tende a diminuir. Entretanto, no caso deste trabalho, rotas com tempos menores podem percorrer distâncias maiores.

II. MODELAGEM

Este capítulo visa apresentar não apenas a modelagem, mas todos os conceitos necessários e as decisões de implementação tomadas pelos autores para o desenvolvimento do trabalho.

A. Instâncias

Para se entender mais facilmente como é representado o problema do caixeiro viajante, é necessário pensar cada cidade sendo um nó de um grafo e suas arestas sendo a ligação entre elas, contendo para este problema dois pesos, a distância e o tempo. Um exemplo de um grafo é mostrado na figura 2, sendo os pontos azuis os nós e as linhas as arestas. Por ser muito intuitivo, praticamente todas modelagens desse problema encontrado na literatura utilizará o conceito dos grafos para analisar quais cidades estão ligadas e qual a distância e o tempo deve ser considerada entre cada rota possível. Com os dados disponibilizados, foi possível perceber que se tratava de um grafo praticamente completo, tendo poucas instâncias com valor de distância ou tempo igual a zero. Para estes casos foi considerado como uma rota inexistente.

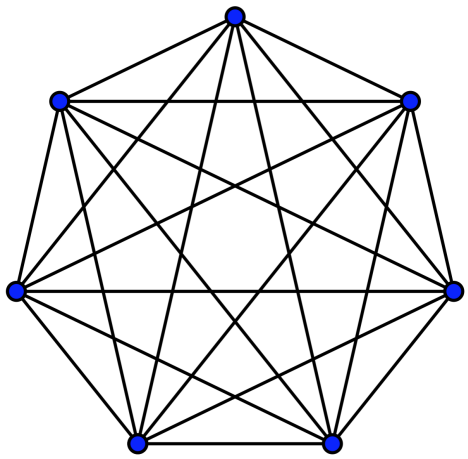

Fig. 2: Grafo Completo

Para desenvolvimento o do trabalho, foram disponibilizado dois arquivos .csv, sendo um com os valores da distância e o outro com o valor do tempo para cada aresta. Os dados consideram a relação entre 250 cidades.

B. Formulação

Dado o conjunto $1, .., n$ as cidades a serem visitadas, o PCV pode ser modelado como um problema de otimização inteiro com a seguinte variável de decisão

$$x_{i,j} = \begin{cases} 1, & \text{se a aresta da cidade } i \text{ para a } j \text{ é usada} \\ 0, & \text{caso contrário} \end{cases}$$

Então um modelo multiobjetivo pode ser construído da seguinte forma

$$\min \begin{cases} \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} d_{i,j} x_{i,j} \\ \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} t_{i,j} x_{i,j} \end{cases} \quad (1)$$

$$\sum_{i=1, i\neq j}^{n} x_{i,j} = 1 \quad j = 1, ..., n \quad (2)$$

$$\sum_{j=1, j\neq i}^{n} x_{i,j} = 1 \quad i = 1, ..., n \quad (3)$$

$$\sum_{i\in S} \sum_{j\in S} x_{i,j} \leq |S| - 1 \quad \forall S \subset Q, |Q| \subseteq \left\{2, ..., \frac{n}{2}\right\} \quad (4)$$

$$x_{i,j} \in [0, 1] \quad \forall i = 1, ..., n \quad \forall j = 1, ..., n \quad (5)$$

onde $d_{i,j}$ e $t_{i,j}$ representam a distância e o tempo entre as cidades i e j, respectivamente.

As funções objetivo da equação (1) visam minimizar a distância total percorrida e o tempo total de viagem. As restrições (2) e (3) garantem que o caixeiro viajante entrará e sairá apenas uma vez de cada cidade. A restrição (4) impede que sejam formadas sub-rotas. E (5) define que as variáveis de decisão são binárias.

C. Modelagem proposta

Como mencionado anteriormente, o Problema do Caixeiro Viajante é um caso conhecido de problema NP-difícil e ao ver que tínhamos 250 cidades para analisar, se tornou extremamente necessário se abrir mão da otimalidade para se obter resultados em um tempo viável.

Definiu-se então um solução como sendo uma rota. A rota é representada no modelo como um vetor de N inteiros distintos. A sequência desse vetor representa a ordem de cidades visitadas pelo caixeiro viajante. A figura 3 mostra um exemplo com 7 cidades.

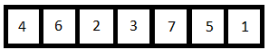

Fig. 3: Exemplo de rota seguida pelo caixeiro viajante

Nesse exemplo, ele inicia a rota pela cidade 4, depois passará pela 6, em seguida para a 2, e assim por diante até chegar à cidade 1. Após passar pela última cidade (no exemplo a 1) ele retorna à cidade inicial (no exemplo a 4), fechando um ciclo.

Cada rota armazena duas métricas, a distância total percorrida e o tempo total do percurso. Essas duas métricas são calculadas a partir do somatório da distância e do tempo, respectivamente, entre os nós adjacentes do vetor. Os pesos entre o nó final e o inicial também são somados para fechar a distância e o tempo totais do ciclo.

É importante ressaltar que, por se tratar de um ciclo, a cidade inicial não é relevante, pois apenas a ordem importa. A rota figura 3 é idêntica à rota 3-7-5-1-4-6-2, por exemplo.

Outro importante ponto de destaque é representação da rota conforme a figura 3, também facilita a implementação, pois ela já garante o cumprimento das restrições (2), (3) e (4) ao definir que todas cidades estarão presentes sem repetição.

III. Algoritmo de solução

Com a definição de que a solução obtida não seria ótima e de como o problema seria modelado, se buscou na literatura heurísticas que se propunham a resolver problemas multiobjetivos de maneira eficiente e com um custo aceitável. Analisando as alternativas e aproveitando de um conhecimento prévio, decidiu-se então utilizar um algoritmo evolucionário. A questão então ficou em escolher entre o *Strength Pareto Evolucionary Algorithm* (SPEA 2) ou o *Non-dominated Sorting Genetic Algorithm II* (NSGA II), que são os dois algorítmos evolutivos mais utilizados para problemas multiobtivos conhecidos na literatura. Através das experiências e conhecimentos obtidos no conteúdo lecionado, constatou-se que para problemas de até três objetivos o NSGA II se comporta de maneira eficiente, desta forma foi o escolhido para a resolução do trabalho.

O NSGA II é um algorítimo evolutivo que foi proposto por Deb et al.2000[3] e que implementa o conceito de Dominância, classificando as populações em fronts de acordo com a dominância dos indivíduos. A figura 4 mostra como essa divisão é feita, sendo os indivíduos da primeira classe os que não são dominados, os da segunda os que são dominados apenas pelos da primeira e assim por diante. O processo de formação dos fronts termina assim que todas tenham sido ranqueadas.

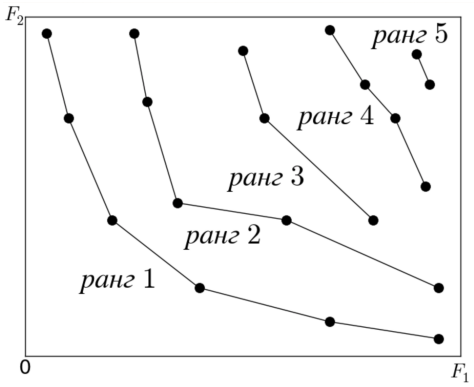

Fig. 4: Divisão de indivíduos em fronts de acordo com a dominância.

A principal diferença entre o NSGA original e o NSGA II está na forma como esses fronts são calculado. A introdução do *Fast Non-Dominated Sorting*, um algoritmo mais rápido e eficiente para o cálculo dos fronts, apresentou um ganho significativo de desempenho para o NSGA

II. Além desse importante atributo, o NSGA II possui outras importantes características, como o *Crowding Distance Assignment* que aumenta a diversidade das soluções encontradas sem precisar de ajuste externo e a garantia de elitismo. Um ponto importante a se observar nesse caso é que melhorar soluções em algoritmos genéticos sem precisar de parâmetros externos é na grande maioria das vezes, uma ótima estratégia, já que este é sempre, e nesse trabalho não foi diferente, o ponto mais complicado quando se opta por utilizar estratégias evolutivas. A figura 5 mostra o fluxograma de como é o processo de seleção do NSGA II.

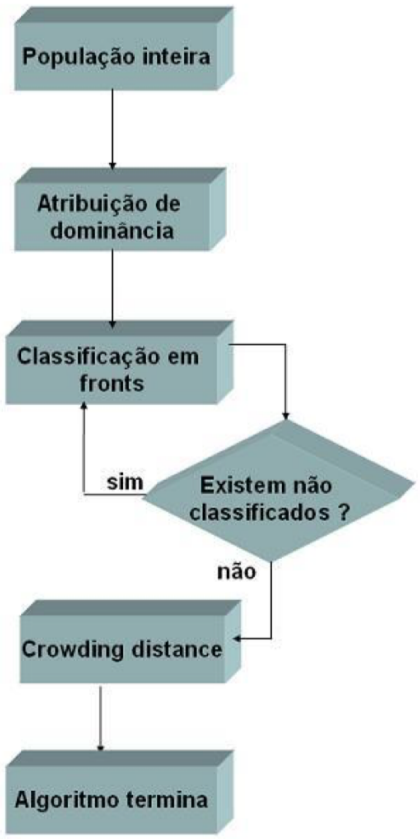

Fig. 5: Representação do Fluxo do processo de seleção do NSGA II. Obtido por Marinho[1]

A. Criação da população inicial

Em um algoritmo evolutivo é interessante que haja uma diversificação dos indivíduos para que tenha uma maior probabilidade de explorar todo o espaço de busca e fugir de mínimos locais. Entretanto, também é importante gerar boas soluções já de início para guiar o algoritmo para uma direção em que provavelmente se possa obter melhores resultados. Dessa forma, a população inicial do algoritmo foi gerada tanto a partir de rotas aleatórias quanto usando uma heurística construtiva.

A heurística construtiva adotada é a do vizinho mais próximo, explicada a seguir. Seleciona-se um nó aleatório inicialmente como ponto de partida da rota. A partir do último nó selecionado, encontra-se o nó mais próximo a ele dentre os nós ainda não escolhidos. Esse nó mais próximo é adicionado ao final da rota. O procedimento é repetido até que todos os nós estejam na rota. Caso em algum momento existam dois ou mais nós mais próximos, escolhe-se um aleatoriamente.

Por se tratar de uma abordagem multi-objetivo, é preciso considerar todos os objetivos nessa etapa. Para tanto, a distância e o tempo das arestas foram normalizados e um custo c foi criado ponderando esses valores, conforme a equação (6). Cada indivíduo é então gerado considerando um custo c com pesos w_t e w_d complementares diferentes. Isso é feito visando maximizar a diversidade da população inicial.

$$c_{i,j} = w_t \frac{t_{i,j}}{||t||} + w_d \frac{d_{i,j}}{||d||} \tag{6}$$

Nos testes computacionais, 50% da população é criada aleatoriamente e 50% a partir da heurística descrita acima.

Esses processos são repetidos até que todos os indivíduos sejam criados. Se em algum momento um indivíduo criado contenha alguma aresta inexistente, ele é descartado e um novo indivíduo é gerado novamente seguindo a mesma regra.

Um exemplo da população inicial gerada com essa abordagem é vista na figura 6.

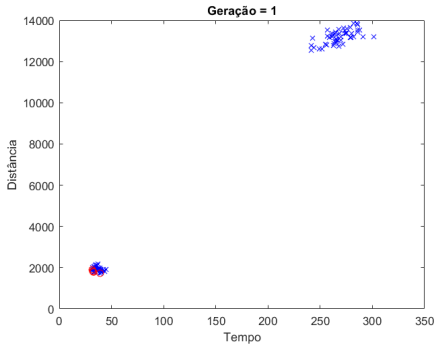

Fig. 6: Exemplo de população inicial gerada

B. Método de cruzamento

A modelagem de rota proposta requer que um indivíduo contenha N inteiros distintos, portanto o método de cruzamento entre dois indivíduos do algoritmo genético deve garantir que o indivíduo gerado mantenha essa propriedade.

No trabalho foi adotado o Partially-Mapped Crossover (PMX), exemplificado na figura 7.

A partir de dois indivíduos $p1$ e $p2$ previamente selecionados, define-se um ponto de corte c aleatório. O indivíduo $p1$ é então copiado e o indivíduo $p2$ é usado de referência. A ordem das primeiras c cidades visitadas por $p2$ são copiadas para o indivíduo $p1$, mas para garantir

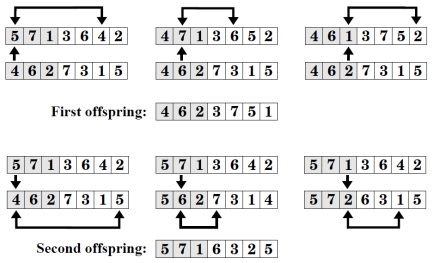

Fig. 7: Método de cruzamento

a integridade da solução realiza-se uma permutação de cidades para cada índice copiado.

No exemplo da figura 7, o ponto de corte é $c = 3$ e a primeira rota é gerada usando o indivíduo 4-6-2-7-3-1-5 de referência e o 5-7-1-3-6-4-2 de cópia. A primeira cidade visitada pela referência é a 4, portanto essa deve ser cidade inicial da rota do novo indivíduo. Como a cidade 4 é a sexta cidade a ser visitada pela cópia, as cidades nas posições 1 e 6 da cópia são permutadas, gerando uma solução temporária 4-7-1-3-6-5-2. Esse procedimento é repetido para as segunda e terceira cidades visitadas pelo indivíduo de referência até gerar a solução 4-6-2-3-7-5-1.

Cada processo de cruzamento gera dois indivíduos a partir do mesmo par de pais, um usando $p1$ de referência e outro usando $p2$. Caso algum dos indivíduos gerados contenha alguma aresta inexistente, ambos são descartados e o processo é repetido.

C. Método de Mutação

A mutação é feita removendo-se n arestas aleatórias da rota, criando $n + 1$ sub-rotas e reconectando-as em outra em outra ordem.

Esse método foi inspirado no *Cross-Exchange*, apresentado na figura 8, onde quatro arestas são removidas e reconectadas de forma cruzada.

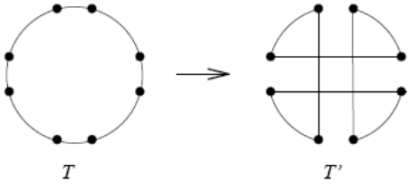

Fig. 8: Exemplo Cross-Exchange

No trabalho, uma ideia similar é aplicada, porém 20% das arestas são removidas.O religamento é então realizando unindo a primeira sub-rota com a última, a segunda com a penúltima e assim por diante. Para o trabalho foi utilizado uma taxa de mutação de 50%.

Caso algum indivíduo gerado contenha alguma aresta inexistente, ele é descartado e o processo é repetido para gerar um novo indivíduo.

IV. Resultados

O algoritmo proposto foi executado cinco vezes, cada uma com uma população de 100 indivíduos e por 100 gerações. As execuções demoraram aproximadamente 90 minutos cada. A tabela I mostra o resultado das convergências e as fronteiras Pareto obtidas ao fim das execuções.

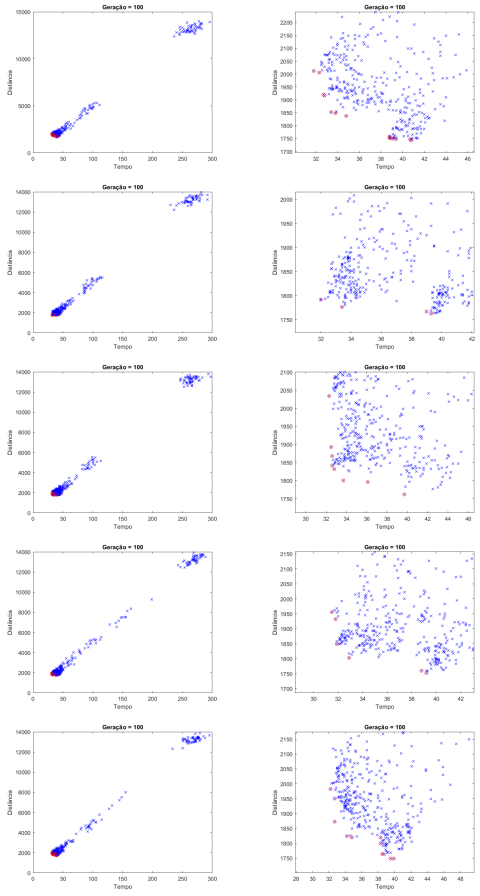

TABLE I: Resultados das execuções do algoritmo

A tabela I mostra que a convergência de todas as execuções foi bem similar. Em todos os casos as soluções aleatórias foram rapidamente abandonadas e novas soluções intermediárias e em torno da fronteira Pareto foram sendo encontradas.

As fronteira finais também foram bem similares, apresentando soluções parecidas. Entretanto, em todos os casos houveram espaços não explorados. Isso mostra que super-indivíduos acabaram guiando a busca nesses casos, fazendo que com o algoritmo buscasse soluções em torno deles e não explorando tanto a fronteira. É possível que outros métodos de mutação e cruzamento aumentassem a diversidade.

A. Medidas de qualidade

Como foi utilizada uma abordagem heurística, é importante medir a qualidade da solução desenvolvida. Um bom conjunto solução deve apresentar boa convergência e diversidade. A convergência pode ser medida a partir da proximidade da fronteira Pareto encontrada em relação à fronteira Pareto ótima. Já a diversidade é dada por quão bem espalhada a fronteira encontrada é.

Para realizar essas medidas, foram implementadas duas métricas de qualidade, a Δ-measure e o hipervolume. Elas precisam de pontos de referência para realizar os cálculos. Neste trabalho, como não se tem uma fronteira pareto ótima para ser referência, decidiu-se usar o ponto utópico e o anti-utópico, ou *nadir*. O ponto utópico é obtido a partir do melhor valor de função objetivo considerando-se cada objetivo isoladamente. Já o nadir é dado pelo valor da outra função objetivo na mesma solução.

Os pontos utópico e *nadir* foram obtidos a partir da otimização para cada objetivo isoladamente. Para tanto, foi usada uma modelagem do problema do caixeiro viajante semelhante à desse trabalho, mas a meta-heurística GRASP foi usada para obter as melhores rotas para cada objetivo separadamente. Foi considerado o melhor resultado após cinco otimizações mono-objetivo para cada objetivo. Os pontos utópico e nadir são mostrados na tabela II. No cálculo das métricas o valor *nadir* foi afastado em 10%.

Ponto	Tempo	Distância
Utópico	29.4	1572.6
nadir	36.9	2408.5

TABLE II: Ponto utópico e nadir usados de referência

1) Hipervolume: É uma das métrica mais utilizadas na literatura por apresentar bons resultados e ser de simples entendimento e implementação. Sua ideia básica é calcular a área total dominada pelas soluções encontradas usando como referência o ponto *nadir*, como mostra a figura 9. Esse cálculo é realizado a partir do somatório das áreas entre cada solução da fronteira, de modo que se duas soluções estão muito próximas a área entre elas fica muito pequena.

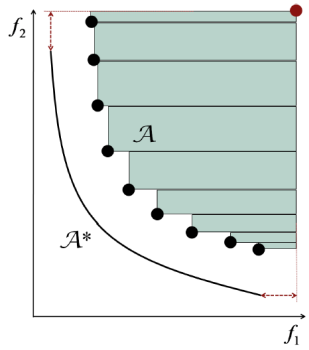

Fig. 9: Funcionamento do Hipervolume

Quanto maior o valor da métrica, melhores são a convergência e a diversidade. No caso da convergência, quanto mais afastada a fronteira Pareto está do ponto anti-utópico, maior o valor da área. A diversidade é medida pelo somatório das áreas entre cada solução. Se as soluções estão muito próximas, essas áreas são pequenas, e consequentemente a área total também fica pequena.

A tabela III mostra o avanço dessa métrica calculada em dez pontos ao longo de cada execução do algoritmo.

Iter	Exec. 1	Exec. 2	Exec. 3	Exec. 4	Exec. 5
10	0.5565	0.6198	0.5644	0.6144	0.5551
20	0.5588	0.6199	0.5648	0.6317	0.5551
30	0.5593	0.6202	0.5732	0.6329	0.5669
40	0.5593	0.6202	0.5739	0.6329	0.5669
50	0.5593	0.6202	0.5751	0.6331	0.5737
60	0.5593	0.6202	0.5806	0.6363	0.5755
70	0.5595	0.6202	0.5806	0.6363	0.5755
80	0.5860	0.6202	0.5806	0.6363	0.5756
90	0.5865	0.6223	0.5806	0.6364	0.5756
100	0.5865	0.6223	0.5806	0.6372	0.5756

TABLE III: Resultados Hipervolume

É possível perceber que os valores de todas as execuções estão bem próximos, o que evidencia a convergência similar em todos os casos. Além disso, a pequena variação dos valores com o avanço das iterações mostra como a utilização da heurística construtiva criou soluções iniciais muito boas, dificilmente batidas nas demais gerações do algoritmo. Entretanto, essa variação crescente mostra que a qualidade das soluções foi melhorando.

2) Δ-Measure: O Δ-Measure é outra métrica de qualidade muito usada na literatura. Ela é calculada conforme a equação (7). Os valores d_i^e dão a distância mínima entre a fronteira encontrada e cada objetivo. Esse valor indica quão distante a fronteira está do ponto ideal, como mostra a figura 10. Os demais valores quantificam a diversidade das soluções, sendo que $|A|$ é o número intervalos entre soluções na fronteira, d_i é a distância euclidiana entre soluções adjacentes e $\overline{d}$ é a média dessas distâncias. Quanto mais próximo de 0, maior o equilíbrio do espaçamento entre as soluções encontradas.

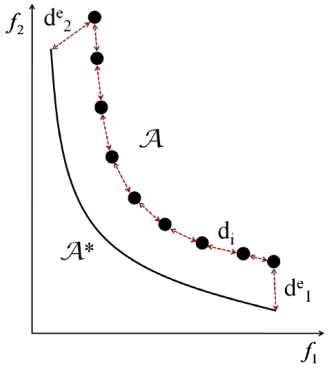

Fig. 10: Funcionamento do Δ-Measure

$$\Delta = \frac{\sum_{i=1}^{m} d_i^e + \sum_{i=1}^{|A|} |d_i - \overline{d}|}{\sum_{i=1}^{m} d_i^e + |A|\overline{d}} \tag{7}$$

A tabela IV mostra o avanço dessa métrica calculada em dez pontos ao longo de cada execução do algoritmo.

Iter	Exec. 1	Exec. 2	Exec. 3	Exec. 4	Exec. 5
10	0.7745	0.8473	0.9657	0.8274	0.7811
20	0.9069	0.8573	1.0743	0.7419	0.7811
30	1.0901	0.9173	1.0033	0.6534	0.8430
40	1.0901	0.9173	1.1087	0.8250	0.8430
50	1.0901	0.9173	0.9780	0.8841	0.8467
60	1.0901	0.9173	0.9362	0.6511	1.0380
70	1.0452	0.9173	0.9362	0.6511	1.0380
80	0.8485	0.9173	0.9218	0.6511	1.1138
90	0.8666	0.9195	0.9218	0.8486	1.1138
100	0.8710	0.9195	0.9218	0.7820	1.0153

TABLE IV: Resultados Δ

Os resultados dessa métrica ficaram sempre próximos de 1 para todas as execuções, principalmente ao final de cada execução. A evolução dos valores ao longo das execuções não apresentou nenhum padrão, o que mostra que a heurística construtiva teve um peso muito alto e dificilmente soluções melhores não dominadas eram encontradas. Por esse motivo, não foi possível realizar nenhuma outra análise a partir do valor do Δ.

B. Tomada de Decisão

A variante analisada do PCV por ser um problema biobjetivo, não é possível ter uma solução ótima. Como foi analisado com o andamento deste trabalho, a execução do NSGA II gera soluções que atendem os dois objetivos, porém ao se melhorar um, o outro pode piorar, logo não existe uma solução ótima, e sim um conjunto de soluções não dominadas que atendem ao problema de maneira diferente e priorizando certos objetivos. Sendo assim, ao final da execução é preciso uma análise para se escolher qual a solução atenderá melhor aos objetivos. Essa escolha pode ser feita por um decisor humano ou por algoritmos de tomada de decisão. A tabela V mostra todos os resultados obtidos pela execução do NSGA II, acrescidos do critério de velocidade que foi obtido pelo princípio físico $d = vt$.

Solução	Tempo	Distância	Velocidade
1	34,80	1837,60	52,80
2	33,80	1848,60	54,69
3	32,70	1920,50	58,73
4	32,30	2005,70	62,10
5	38,90	1751,70	45,03
6	33,40	1852,60	55,47
7	40,80	1743,10	42,72
8	38,80	1756,50	45,27
9	38,80	1752,00	45,15
10	32,80	1916,10	58,42
11	31,80	2011,70	63,26
12	40,70	1747,20	42,93
13	39,40	1747,80	44,36
14	39,10	1749,50	44,74
15	39,30	1762,60	44,85
16	32,00	1791,60	55,99
17	39,00	1767,00	45,31
18	33,40	1776,50	53,19
19	36,10	1795,90	49,75
20	33,70	1800,30	53,42
21	32,30	2034,00	62,97
22	32,60	1841,50	56,49
23	32,80	1831,80	55,85
24	32,50	1892,80	58,24
25	39,70	1761,80	44,38
26	32,60	1867,90	57,30
27	32,90	1802,80	54,80
28	31,90	1848,90	57,96
29	31,80	1932,20	60,76
30	31,50	1955,50	62,08
31	38,80	1759,80	45,36
32	39,20	1752,20	44,70
33	38,70	1762,60	45,55
34	34,20	1824,20	53,34
35	32,70	1872,70	57,27
36	34,80	1819,90	52,30
37	32,20	1982,30	61,56
38	38,30	1799,90	46,99
39	39,50	1748,70	44,27
40	38,30	1819,60	47,51
41	38,50	1764,60	45,83
42	39,90	1748,40	43,82
43	32,70	1950,40	59,65

TABLE V: Todas soluções obtidas após cinco execuções

Para este trabalho foi necessário utilizar dois métodos diferentes, o Analytic Hierarchy Process (AHP) e o Preference Ranking Organization METHod for Enrichment of Evaluations II (PROMÉTHÉE II) para se resolver o problema da tomada de decisão. O primeiro algoritmo foi utilizado estritamente para se definir os pesos que seriam usados para cada critério na utilização do segundo. Porém antes de se usar o algorítimo de decisão, foi preciso retirar todos os resultados dominados mostrados na figura V, já que por já serem dominados, não precisam ser analisados, já que existe outra solução melhor. Desta forma foram retiradas as vinte duas soluções a seguir: 1, 2, 6, 15, 17, 19, 20, 21, 23, 25, 26, 27, 32, 34, 35, 36, 37, 38, 39, 40, 42, 43. Restando assim as soluções mostradas na figura VI.

Solução	Tempo	Distância	Velocidade
1	32,70	1920,50	58,73
2	32,30	2005,70	62,10
3	38,90	1751,70	45,03
4	40,80	1743,10	42,72
5	38,80	1756,50	45,27
6	38,80	1752,00	45,15
7	32,80	1916,10	58,42
8	31,80	2011,70	63,26
9	40,70	1747,20	42,93
10	39,40	1747,80	44,36
11	39,10	1749,50	44,74
12	32,00	1791,60	55,99
13	33,40	1776,50	53,19
14	32,60	1841,50	56,49
15	32,50	1892,80	58,24
16	31,90	1848,90	57,96
17	31,80	1932,20	60,76
18	31,50	1955,50	62,08
19	38,80	1759,80	45,36
20	38,70	1762,60	45,55
21	38,50	1764,60	45,83

TABLE VI: Soluções não dominadas

Como dito anteriormente, o método AHP foi utilizado gerar os pesos que seriam utilizados no PROMETHEE II. Empregando a escala fornecida pelo AHP de se definir as importâncias entre os critérios, os pesos escolhidos foram mostrados na tabela VII. Analisando os valores definidos, é possível perceber que o tempo ficou sendo o critério mais importante, sendo a distância em segundo e a velocidade em terceiro. Vale ressaltar que todos esses pesos foram definidos pelos implementadores e que as preferências e importâncias podem variar dependendo da analise.

	Tempo	**Distância**	**Velocidade**
Tempo	1	3	5
Distância	1/3	1	3
Velocidade	1/5	1/3	1

TABLE VII: Pesos definidos pelo AHP

O PROMETHEE é um método de relações de subordinação que faz uma pré-ordem das soluções de acordo com as preferências parametrizadas pelo implementador. Pelo fato do PROMETHEE I gerar algumas alternativas incomparáveis, este criava uma pré-ordem parcial. Desta forma procurou-se atualizar o algoritmo para que uma pré-ordem completa fosse gerada, eliminando a possibilidade de alternativas incomparáveis, criando o PROMETHEE II.

Com os pesos obtidos pelo método AHP, pode se executar o PROMETHEE II sem maiores modificações. Para o grau de preferência do tomador de decisão em todos critérios, foi utilizada a matriz de preferência linear mostrada na figura 11 com o valor de $p = 150$. Esse valor foi obtido através de análise empírica dos resultados.

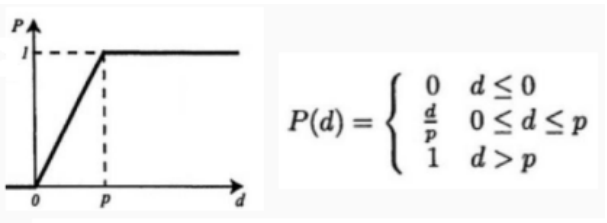

Fig. 11: Matriz de preferência linear

Com a execução do PROMETHEE II, foi obtida uma pré-ordem completa de todas soluções, e na tabela VIII, é possível ver as cinco melhores.

Solução	**Tempo**	**Distância**	**Velocidade**
2	32,30	2005,70	62,10
8	31,80	2011,70	63,26
17	31,80	1932,20	60,76
7	32,80	1916,10	58,42
1	32,70	1920,50	58,73

TABLE VIII: Pré-Ordem completa obtida

V. CONCLUSÃO

O trabalho desenvolvido cumpriu o esperado que foi exercitar os conceitos lecionados na disciplina sendo necessário realizar todos os passos para se resolver um problema multiobjetivo.

O PCV por mais que seja um problema de conceito muito simples, sua resolução se mostrou muito desafiadora, principalmente pelo fato de ser NP_Difícil, não existindo solução em tempo polinomial. Desta forma, ao se abdicar da otimalidade, na visão dos implementadores, a escolha por um algoritmo evolutivo se mostrou acertada por ser uma forma mais simples e que se mostrou eficiente. Entretanto, algoritmos evolutivos tem problemas próprios como a dificuldade em se ajustar os parâmetros corretamente e fugir de super-indivíduos quando se usam heurísticas construtivas, que por mais eficientes que sejam, é necessário ter cautela ao se utilizar. Desta forma é de extrema importância se utilizar métricas de qualidade para avaliar se os resultados obtidos foram bons.

Após a realização do trabalho foi analisado que mesmo tendo obtido resultados bons, estes poderiam ser melhorados se usando estratégias de busca local para tentar fugir de bacias de atração encontrada pelas soluções.

References

[1] MARINHO, Davi. Uma Aplicaçao do Algoritmo Genético Multiobjetivo NSGA II Para Seleçao de Imagens de Satélite de Trechos de Mata Atlântica. Recife, PE, 2009.

[2] HOLLAND, John Henry et al. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, 1992.

[3] DEB, Kalyanmoy et al. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: International conference on parallel problem solving from nature. Springer, Berlin, Heidelberg, 2000. p. 849-858.