Modelos e Algoritmos para o Problema da Diversidade Máxima

Henrique Barros Lopes
Departamento de Computação – Centro Federal de Educação Tecnológica de Minas Gerais
Belo Horizonte – MG – Brazil
henriquelouko@gmail.com

João Fernando Machry Sarubbi
Departamento de Computação – Centro Federal de Educação Tecnológica de Minas Gerais
Belo Horizonte – MG – Brazil
joaosarubbi@gmail.com

RESUMO

O Problema da Diversidade Máxima é um problema NP-Completo com um grande número de aplicações práticas. Ele consiste em escolher, em uma população, um subconjunto com a maior diversidade possível entre os elementos. Neste trabalho nós propomos um novo modelo matemático e uma heurística GRASP para resolvê-lo. Os resultados preliminares mostraram que a heurística se mostrou competitiva encontrando o valor ótimo na maioria das instâncias testadas em um tempo computacional baixo.

PALAVRAS CHAVE. Problema da Diversidade Máxima, Modelo de Programação Linear, Heurística GRASP.

ABSTRACT

The Maximum Diversity Problem is NP-Complete with a large number of practical applications. It consists of choosing, in a population, a subset with the greatest possible diversity among the elements. In this paper we propose a new mathematical model and a GRASP heuristic to solve it. Preliminary results showed that the heuristic proved competitive finding great value in most instances tested at a low computational time.

KEYWORDS. Problema da Diversidade Máxima, GRASP, Modelo de Programação Linear Inteira Mista.

1. Introdução

O Problema da Diversidade Máxima (PDM) (Glover et al., 1977) (Kuo et al., 1993) é um problema de natureza matemática, mais precisamente do ramo da análise combinatória. Podemos definí-lo formalmente como: dado um conjunto N com n elementos distintos ($N = \{1, 2, 3, ..., n\ \}$), é necessário selecionar um subconjunto M desse conjunto N com m elementos ($M = \{2, 3, ..., m\ \}$) de forma que a soma das diferenças entre cada um desses elementos seja a maior possível. Diferença é uma medida comparativa entre dois elementos, portanto o PDM só faz sendito se existir 2 ou mais elementos no subconjunto M tomando $m < n$.

Para entender melhor o conceito de diferença no PDM consideremos três pessoas: João, José e Maria. João tem 27 anos de idade, José tem 30 e Maria 38. Caso quiséssemos utilizar a definição do PDM para selecionar um subconjunto com duas pessoas desse conjunto de forma que a diferença da idade seja a maior possível primeiro precisamos comparar elemento a elemento a diferença de suas idades. Dessa forma temos que a diferença entre João e José são 3 anos, de José e Maria são 8 anos e de Maria e João são 11 anos. Podemos representar essas diferenças por meio de um grafo como na figura (1).

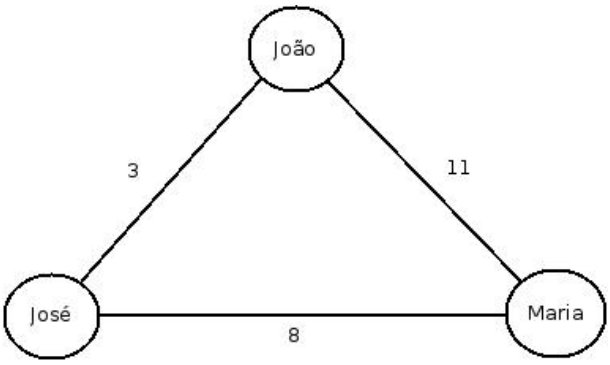

Figura 1. Grafo exemplo

Em geral a diferença é chamada de distância entre dois elementos. A forma de representar a distância entre dois elementos não influencia na resolução do problema. A utilização de uma métrica fica a critério da contextualização do problema. Existem diversos tipo de métricas utilizadas para contextualizar o PDM, Ghosh (1996) o fez utilizando a norma-p dada pela equação (1).

$$d_{ij} = \sqrt[p]{\sum_{m \in N} (|a_{im} - a_{jm}|)^p} \tag{1}$$

A variável d_{ij} representa a diferença existente entre o elemento i e o elemento j do conjunto analisado, onde $i \in N$ e $j \in N$. O valor dessa variável é definido pelo somatório da diferença entre cada uma das respectivas m características de cada elemento.

Definida uma métrica para calcular a diferença entre dois elementos do conjunto, podemos definir sua diversidade como o somatório de todas as diferenças existentes. A equação (2) mostra o cálculo da diversidade de um subconjunto, onde a variável d_{ij} representa a diferença calculada pela métrica escolhida.

$$\zeta(M) = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} d_{ij} \tag{2}$$

Se representarmos o PDM por meio de um grafo podemos considerar também que sua diversidade é o somatório dos valores de todas as suas arestas. Sabendo como calcular a diversidade de um conjunto, para resolver o PDM, basta selecionar o subconjunto que expresse a maior diversidade possível dentre os subconjuntos de M.

2. Trabalhos Relacionados

Glover et al. (1977) foram os primeiros a definirem o PDM. Eles trataram o PDM como um problema de otimização em um contexto de recursos genéticos.

Aringhieri et al. (2008) mostram um exemplo de aplicação onde existe a necessidade de formar equipes de trabalho, jurados para um julgamento e grupos de estudantes para trabalhar em projetos. Em todas essas situações é mostrada a necessidade de selecionar indivíduos com características distintas afim de a equipe formada corresponder melhor à situação. No caso da seleção de jurados, deve-se explorar o maior número de opiniões para que o júri não tome uma decisão equivocada a respeito do réu. É citado também a vantagem de se aplicar o PDM em problemas de tomada de decisão nas organizações, nesses problemas a diversidade valoriza a criatividade e qualidade de gerenciamento da organização (Fernandez, 1991), (Cox, 1993).

O PDM também possui grande aplicação no ramo da genética animal e vegetal, quando temos o objetivo de obter novas variedades através da reprodução de indivíduos distintos de forma controlada (Porter et al., 1975), estímulo de diversidade étnica entre imigrantes (McConnell, 1988), preservação de diversidade biológica (Glover et al., 1995), equilíbrio ambiental, desenho de produto, gerenciamento de força de trabalho e engenharia genética (Glover et al., 1998).

Diversos autores propuseram heurísticas para o PDM. Podemos citar Silva et al. (2007) que desenvolveram uma heurística GRASP para o problema, Duarte and Marti (2007) que desenvolveram um algoritmo de Busca Tabu, Aringhieri et al. (2008) que realizaram uma comparação entre os algoritmos GRASP e Busca Tabu e, finalmente, Marti et al. (2010) que fizeram uma extensa comparação entre os métodos heurísticos desenvolvidos para o PDM.

3. O Modelo Matemático Proposto

Kuo et al. (1993) apresentam três modelos matemáticos para o problema. O primeiro modelo denominado F1, equações (3)-(5), apresenta o PDM como um programa inteiro quadrático 0-1.

$$max \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j \tag{3}$$

$$s.a. \sum_{i=1}^{n} x_i = m \tag{4}$$

$$x_i \in \{0,1\}, \forall i \in N \tag{5}$$

A função objetivo (3) desse modelo mostra um somatório duplo com o produto $d_{ij}x_ix_j$, onde d_{ij} é a distância entre o elemento i e o elemento j. A variável x indica se o elemento do conjunto N pertence ou não ao subconjunto M, x_i assume o valor 1 se o elemento i faz parte do subconjunto M e 0 se não faz parte. Na função objetivo caso uma das variáveis x_i ou x_j seja 0, significa que um dos elementos não faz parte do conjunto M e, portanto, não podemos contabilizar sua diferença na diversidade do subconjunto.

A restrição (4) indica que o somatório das variáveis x_i deve ser igual a m, ou seja, ao número de elementos no subconjunto. Esse número m é sempre constante em uma instância do PDM. Cada x_i que é igual a 1 equivale a um elemento no subconjunto M, então essa restrição garante que selecionaremos apenas m elementos em nosso subconjunto, mais elementos torna o problema infactível. As restrições (5) indicam que as variáveis x são binárias, ou seja, podem assumir apenas os valores 0 e 1.

O maior problema do modelo F1 é a presença de uma função objetivo não-linear. Esse tipo de função pode fazer com que o algoritmo utilizado para resolver o problema demore para convergir. Para contornar esse detalhe (Kuo et al., 1993) utilizaram uma abordagem apresentada em (Glover and Wolsey, 1974) para transformá-lo em um modelo linear, substituindo a parte não linear da função objetivo em uma nova variável y_{ij} elaborando o modelo F2, equações (6)-(12).

$$max \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} y_{ij} \tag{6}$$

$$s.a. \sum_{i=1}^{n} x_i = m \tag{7}$$

$$x_i + x_j - y_{ij} \leqslant 1, \forall i, j \in N : i < j \tag{8}$$

$$-x_i + y_{ij} \leqslant 0, \forall i, j \in N : i < j \tag{9}$$

$$-x_j + y_{ij} \leqslant 0, \forall i, j \in N : i < j \tag{10}$$

$$y_{ij} \geq 0, \forall i, j \in N : i < j \tag{11}$$

$$x_i \in \{0, 1\}, \forall i \in N \tag{12}$$

No modelo F2 notamos a presença do produto $d_{ij}y_{ij}$ na função objetivo (6). A variável y_{ij} representa a existência dos elementos i e j no subconjunto M. O parâmetro d_{ij} apresenta as diferenças calculadas assim como no modelo F1. A restrição (7) é a mesma do modelo F1 e indica que apenas m elementos podem fazer parte do subconjunto M. O conjunto de restrições (8) serve para conectar as variáveis x com as variáveis y fazendo com que sempre que x_i e x_j forem iguais a 1, y_{ij} deverá ser 1 também. As restrições (9) e (10) obrigam que sempre que as variáveis x_i ou x_j forem zero as variáveis correspondentes y_{ij} também serão zero. As restrições (11) definem que as variáveis y devem ser positivas e as restrições (12) dizem que as variáveis x devem ser binárias assim como no modelo F1.

Observando as características do problema acrescentamos 3 novos conjuntos de restrições ao modelo F2 com o intuito de gerar cortes válidos que auxiliem na resolução do modelo via software CPLEX. Esse novo modelo foi chamado de F2 Melhorado ou, simplesmente, F2M.

$$\sum_{i,j \in N} y_{ij} = \frac{m(m-1)}{2} \quad (13)$$

$$\sum_{j \in N} y_{ij} = m-1, \forall i \in N \quad (14)$$

$$\sum_{i \in N} y_{ij} = m-1, \forall j \in N \quad (15)$$

A nova restrição (13) diz que o somatório de todas as variáveis y_{ij} deve ser iguais ao número total de arestas no subgrafo que representa o subconjunto M. O conjunto de restrições (14) diz que cada vértice que pertence ao subconjunto M possui $m-1$ arestas conectadas a outros vértices do subconjunto M que contabilizarão a diversidade de M. O conjunto de restrições (15) diz que cada vértice tem $m-1$ arestas provenientes de outros vértices conectadas à ele.

4. A Metaheurística GRASP

O método GRASP, do inglês, *Greedy Randomized Adaptive Search Procedure*, em português (procedimento de busca guloso, adaptativo e aleatório) (Feo and Resende, 1995) (Resende and Ribeiro, 2003), (Silva et al., 2007), (Aringhieri et al., 2008) foi o escolhido para encontrar soluções viáveis para o PDM.

Essa meta-heurística é dividida em duas fases: a construção de uma solução viável e um subseqüente procedimento de Busca Local. Essas duas fases são repetidas a cada iteração. Na Fase de Construção, uma função gulosa e aleatória é usada para construir uma solução inicial. Essa solução é então utilizada como solução inicial para a Fase de Busca Local. Essa fase é usada para tentar melhorar a solução dada pela Fase de Construção. O resultado final é simplesmente a melhor solução encontrada após todas as iterações.

Na Fase de Construção, uma técnica gulosa e aleatória gera soluções viáveis. Cada solução viável é iterativamente construída, um elemento por vez. No algoritmo proposto começa-se de uma solução vazia e a cada passo é calculado o custo da solução acrescentando cada nodo que está fora da solução e suas respectivas arestas incidentes. A equação (16) apresenta o índice $\iota(i)$ proposto para esse cálculo. O índice $\iota(i)$ elaborado para inserção dos elementos é semelhante ao proposto por (Ghosh, 1996), a diferença é que não é calculado um limite superior ou inferior para os vértices candidatos a entrar no conjunto solução. Pela equação, nota-se que o índice $\iota(i)$ trata-se apenas do somatório das arestas que conectam o i-ésimo elemento fora da solução M e todos os elementos que já estão no conjunto M.

Entretanto, ao invés de se escolhermos sempre o melhor elemento, isto é, o nodo com melhor $\iota(i)$, criamos uma lista restrita de candidatos (LRC), isto é, uma lista dos melhores elementos. Aleatoriamente, um desses elementos é escolhido a cada passo e não necessariamente o melhor elemento. Um parâmetro da LRC, α, determina o quanto o algoritmo é guloso ou aleatório. Quando $\alpha = 0$, tem-se uma solução totalmente gulosa, isto é, o tamanho da LCR é igual a 1. Por outro lado, quando $\alpha = 1$, tem-se uma solução totalmente aleatória. Nesse caso, o tamanho da LCR é igual ao número de vértices que ainda não foram acrescentados a solução. Normalmente não é utilizado um $\alpha = 0$ pois,

caso isso ocorra, em todas as iterações a solução oriunda da fase de construção será a mesma. Da mesma forma, normalmente não é interessante utilizar um $\alpha = 1$ pois assim a solução calculada em cada iteração da fase de construção será totalmente aleatória. O que torna o método GRASP diferenciado é que em cada iteração a busca local gera soluções diferentes, mas não totalmente aleatórias.

$$\iota(i) = \sum_{j \in M} d(i,j), i \in N - M \tag{16}$$

A busca local implementada consiste em verificar a troca de cada elemento fora da solução por cada elemento dentro da solução. Após realizar todas as trocas possíveis (dois a dois), a nova solução será aquela em que a diversidade da nova solução for a maior encontrada. Isto é, é realizada uma busca local exaustiva verificando todas as possibilidades de troca de elementos dois a dois. Esse procedimento é repetido até que não seja mais encontrada uma melhora na solução.

5. Resultados

Nesta seção serão apresentadas e comparadas as soluções dos modelos e algoritmos apresentados anteriormente. Os testes foram realizados usando um computador Core i5 2.9 GHz da Intel, com o sistema Operacional Mac OS e 8 GB de memória RAM.

Para resolver os modelos matemáticos citados via programação linear inteira mista, utilizamos o *software* CPLEX da IBM em sua versão 12.2 com tempo limite de quatro horas. O método para o cálculo da relaxação linear foi o Barreira. O algoritmo GRASP implementado teve como parâmetros α=0.2 e 2000 iterações. O tempo limite dos algoritmos foi de 4 horas. Todos os tempos presentes neste trabalho foram medidos em segundos e todos os GAPs são um percentual. Para os resultados do GRASP apresentamos o tempo que o melhor valor foi encontrado e não o tempo que até a última iteração.

Para os testes dos modelos e do algoritmo foram utilizados os conjuntos de instâncias GKD-a (Glover et al., 1998), GKD-b (Marti et al., 2010), GKD-c (Duarte and Marti, 2007) e SOM-b (Silva et al., 2007).

As instâncias dos conjunto GKD são instâncias que foram geradas calculando a distância euclidiana entre pontos gerados aleatoriamente com coordenadas de valores entre zero e dez. As instâncias do conjunto SOM foram construídas a partir da geração aleatória de números entre zero e nove de uma distribuição uniforme inteira.

Cada tabela estará relacionada a um conjunto de instâncias e mostrará os resultados referentes ao modelo F2M e ao algoritmo GRASP implementado.

Os testes feitos com os modelos F1 e F2 mostraram que o CPLEX resolve o modelo F2 em menos tempo que o F1. Esses testes foram omitidos por não terem grande relevância para este trabalho.

Na tabela 1 são apresentados os resultados referentes as instâncias GKD-a. Para esse conjunto de instâncias será comparado o tempo e o *gap* de relaxação linear dos modelos F2 e F2M. Também será apresentado o resultado do algoritmo GRASP para essas instâncias. Cada linha da tabela 1 apresenta o resultado das médias de tempos e *gaps* das 5 instâncias de cada conjunto.

Os campos n e m indicam o tamanho dos conjuntos N e M de cada conjunto de instâncias respectivamente. No campo *GAP RL (%)* é exibido, para os dois modelos testados, a média dos *gaps* de relaxação linear. O *gap* de relaxação linear informa a quantidade, em percentual, que a relaxação linear dista da solução ótima. Esse *gap* é calculado através da equação (17), onde *LS* é a relaxação linear e O^* é o valor da solução ótima da instância. Como o PDM é um problema de maximização, a relaxação linear nos dá um valor acima do valor ótimo. A coluna *Tempo RL (s)* indica a média de tempo em segundos gasto por cada modelo para encontrar a relaxação linear das cinco instâncias deste tipo e a coluna *Tempo O** indica o tempo gasto, em média, para provar a otimalidade dessas cinco instâncias. Os campos *Gap O* (%)* e *Tempo (s)* fazem referência aos resultados do algoritmo GRASP implementado e apresentam a média de *gaps* entre a solução ótima de cada instância e o tempo gasto pelo algoritmo GRASP para encontrar a melhor solução.

GKD-a		F2			F2M			GRASP	
n	m	GAP RL (%)	Tempo RL (s)	Tempo O* (s)	GAP RL (%)	Tempo RL (s)	Tempo O* (s)	Gap O* (%)	Tempo (s)
10	2	31,29	0,02	0,26	31,29	0,00	0,01	0,00	0,00
10	3	10,43	0,02	0,22	10,43	0,01	0,02	0,00	0,00
10	4	5,16	0,01	0,26	5,16	0,01	0,01	0,00	0,00
10	6	1,71	0,01	0,27	1,71	0,01	0,01	0,00	0,00
10	8	0,51	0,02	0,11	0,51	0,01	0,01	0,00	0,00
15	3	25,51	0,02	0,51	25,51	0,02	0,03	0,00	0,01
15	4	12,16	0,01	0,58	12,16	0,02	0,04	0,00	0,01
15	6	5,18	0,02	1,07	5,18	0,02	0,04	0,00	0,01
15	9	1,64	0,03	0,75	1,64	0,02	0,03	0,00	0,01
15	12	0,51	0,02	0,30	0,51	0,02	0,02	0,00	0,01
30	6	21,97	0,04	77,71	21,97	0,25	0,45	0,00	0,01
30	9	9,44	0,04	788,07	9,44	0,21	0,37	0,00	0,01
30	12	4,74	0,07	9023,26	4,74	0,18	0,33	0,00	0,01
30	18	1,63	0,08	19710,0	1,63	0,18	0,32	0,00	0,01
30	24	0,48	0,02	3,88	0,48	0,21	0,24	0,00	0,01

Tabela 1. GKD-a - Comparação F2 e F2M - GRASP

$$GAP\ RL = \left|1 - \frac{LS}{O^*}\right| \tag{17}$$

Os resultados da tabela 1 indicam que as inserção das restrições adicionais no modelo F2M não melhoraram o *gap* de relaxação linear do problema, mas fizeram com que a solução ótima fosse encontrada em um tempo, em alguns casos, ordens de grandeza menor. Além disso, o algoritmo GRASP encontrou o valor ótimo para todas as 75 instâncias do conjunto GKDd-a. Apesar do algoritmo GRASP demorar, em alguns casos, mais de 3 segundos para terminar as 2000 iterações, em todos os casos a solução ótima foi encontrada nas primeiras iterações com menos de 0.2 segundos.

$$GAP\ RL = \left|1 - \frac{MVL}{MVE}\right| \tag{18}$$

GKD-b		F2M		DBR		GRASP	
n	m	Tempo (s)	Gap (%)	Tempo (s)	GAP (%)	Tempo (s)	Gap O* (%)
25	2	0,11	0,00	0,12	0,00	0,0	0,00
25	7	0,17	0,00	0,40	0,00	0,0	0,00
50	5	2,61	0,00	1,56	0,00	0,0	0,00
50	15	26,19	0,00	94,89	0,00	0,0	0,00
100	10	179,36	0,00	25,53	0,00	0,1	0,00
100	30	4648,31	0,52	975,95	0,16	0,2	0,00
125	12	9143,57	3,33	286,22	0,00	0,3	0,00
125	37	14400	1,02	3183,7	0,24	0,5	0,00

Tabela 2. GKD-b - Comparação F2M DBR - GRASP

A Tabela 2 é referente as instâncias GKD-b. Para essas instâncias não comparamos o modelo F2 pois já verificamos que o modelo F2M tem um resultado superior. Dessa forma, resolvemos comparar o modelo F2M com um algoritmo especializado baseado na Decomposição de Benders (Benders, 1962) proposto por Takane (2011). Nessa mesma tabela apresentamos os resultados do algoritmo GRASP implementado.

Para essa tabela também apresentamos os resultados das médias dos tempos e *gaps* das 5 instâncias de cada conjunto. Os campos da tabela 2 são os mesmos da tabela 1, mas ao invés de apresentarmos o *gap* de relaxação linear, o campo *Gap (%)* apresenta a média dos *gaps* entre a solução ótima e a solução encontrada pelo modelo F2M e entre a solução ótima e a solução encontrada pelo algoritmo DBR de Takane.

É possível observar que o algoritmo DBR encontra resultados com tempos e *gaps* melhores para todas as instâncias. Podemos observar que, como método exato, o algoritmo DBR é superior ao modelo F2M elaborado. Porém, devido ao fato do modelo F2M ter se mostrado superior ao modelo F2, em termos de tempo total de execução, podemos aplicar a Decomposição de Benders no modelo F2M e obter um novo algoritmo exato que possa encontrar bons resultados na resolução do PDM.

A Tabela 3 apresenta os resultados do conjunto de instâncias GKD-c utilizando o modelo F2M e o algoritmo GRASP. Cada linha da tabela apresenta os resultados de uma instância específica. O campo *MVL* apresenta o melhor valor encontrado na literatura para a respectiva instância. O campo *Tempo RL (s)* apresenta o tempo, em segundos, para se encontrar a solução relaxada e o campo *Sol* apresenta o melhor solução inteira após 4 horas de execução do modelo no software CPLEX. O campo *GAP (%) MVL* apresenta a diferença, em percentual, entre os valores de MVL e a melhor solução inteira encontrada pelo CPLEX. Esse campo é calculado pela equação (18). O campo *Sol GRASP* apresenta a solução encontrada pela heurística GRASP após 2 horas. Os campos *Tempo (s)* e *Iter Sol* mostram em quanto tempo e em qual iteração do algoritmo o valor da melhor solução do GRASP foi encontrada.

As instâncias com um * indicam que o resolvedor CPLEX gastou mais de 4 horas para encontrar a solução relaxada não entrando no *Branch-and-Bound*.

Apesar de, em duas das vinte instâncias contidas no conjunto GKD-c, o algoritmo GRASP ter encontrado uma solução melhor do que a encontrada em (Gallego et al., 2009),

GKD-c								
Instância		CPLEX			GRASP			
	MVL	Tempo RL (s)	Sol	GAP (%) MVL	Tempo (s)	Sol GRASP	GAP (%) MVL	Iter Sol
GKD-c-1	19485,18	322,19	15815,3	23,20	98	19458,17	0,00	1
GKD-c-2*	19701,53	224026	15957,5	23,46	61	19701,53	0,00	3
GKD-c-3*	19547,20	14713,74	15984,61	22,28	31	19547,20	0,00	1
GKD-c-4	19596,46	4023,16	16375,4	19,67	36	19596,46	0,00	1
GKD-c-5	19602,62	1071,72	18427,0	6,379	17	19602,58	0,00	1
GKD-c-6	19421,55	365,44	18599,7	4,418	26	19421,54	0,00	1
GKD-c-7	19534,30	664,84	15708,40	24,35	42	19534,30	0,00	1
GKD-c-8	19487,32	494,44	15984,60	21,91	119	19487,33	0,00	1
GKD-c-9	19221,63	279,95	18202,90	5,59	16	19221,63	0,00	1
GKD-c-10	19703,55	1051,02	15918,30	23,77	33	19703,33	0,00	1
GKD-c-11	19587,12	418,10	16567,90	18,22	43	19587,09	0,00	1
GKD-c-12	19360,23	1396,29	15681,50	23,45	15	19360,21	0,00	1
GKD-c-13	19366,69	19446,51*	15940,75	21,49	82	19366,69	0,00	1
GKD-c-14	19458,56	19445,64*	16164,60	20,37	43	19458,56	0,00	4
GKD-c-15	19422,15	19446,09*	15591,28	24,57	117	19422,15	0,00	1
GKD-c-16	19680,20	19445,15*	16457,25	19,58	19	19680,22	0,00	1
GKD-c-17	19331,38	19447,29*	15843,22	22,01	18	19331,37	0,00	1
GKD-c-18	19461,39	19447,55*	15893,67	22,44	15	19461,39	0,00	1
GKD-c-19	19411,32	19448,05*	16091,82	21,03	29	19477,31	0,00	1
GKD-c-20	19604,84	19446,16*	15326,42	27,91	15	19604,84	0,00	1

Tabela 3. Resultados GKD-c

não podemos afirmar que essa solução foi melhorada, porque a diferença entre uma solução e outra é de no máximo $0,02$. É possível que estes valores sejam fruto de erros de arredondamento cumulativos. O mesmo vale para o caso das instâncias que o algoritmo GRASP não alcançou o melhor valor da literatura. A diferença foi tão pequena que é possível que tenha havido erros de arredondamento e a solução seja a mesma. Outra observação importante é que como as instâncias são muito grandes, a melhor solução encontrada pelo algoritmo foi, em dezoito dos vinte casos, na primeira iteração do GRASP. Isso acontece porque o GRASP não realiza iterações suficientes em duas horas para melhorar esse valor.

Assim como a Tabela 3, a Tabela 4 também apresenta os resultados do conjunto de instâncias, nesse caso o conjunto SOM-b, utilizando o modelo F2M e o algoritmo GRASP. Os campos também são os mesmos da Tabela 3. Podemos notar que em alguns casos o CPLEX demorou mais de 4 horas somente para achar a solução relaxada e que essa relaxação chegou, em alguns casos, a distar 30% da solução ótima. Por isso, em alguns instâncias o tempo do CPLEX terminou em mais de 4 horas. O solver CPLEX somente termina sua execução após a solução de programação linear. Em relação aos resultados heurísticos o algoritmo implementado teve um desempenho satisfatório ficando, encontrando o ótimo em metade das instâncias do conjunto e, quando não o encontrou, a solução encontrada ficou, no pior caso, a 0.3% do ótimo.

SOM-b										
Instância				CPLEX			GRASP			
Instância	n	m	MVL	Tempo (s)	Sol	GAP (%) MVL	Tempo (s)	Sol	GAP (%) MVL	Iter Sol
Som-b-1	100	10	333	4,49	320	4,06	0	333	0,00	15
Som-b-2	100	20	1195	3,77	856	39,60	0	1195	0,00	13
Som-b-3	100	30	2457	3,20	2393	2,67	0	2457	0,00	5
Som-b-4	100	40	4142	3,66	4142	0,00	12	4142	0,00	36
Som-b-5	200	20	1247	18,43	1103	13,05	3	1247	0,00	19
Som-b-6	200	40	4450	1376	3503	27,03	569	4448	0,04	602
Som-b-7	200	60	9437	18,86	8462	11,52	17	9412	0,00	7
Som-b-8	200	80	16225	15,00	14592	11,19	246	16225	0,00	65
Som-b-9*	300	30	2694	52895	2030	32,70	1210	2694	0,00	954
Som-b-10	300	60	9698	77,16	8046	20,42	8840	9683	0,15	1337
Som-b-11*	300	90	20743	15471	17993	15,28	9696	20731	0,05	856
Som-b-12	300	120	35881	47,19	32648	9,01	2020	35881	0,00	454
Som-b-13*	400	40	4658	129671	3619	28,70	2664	4658	0,00	517
Som-b-14	400	80	16956	82,81	14405	17,70	14410	16924	0,18	487
Som-b-15*	400	120	36317	14276	32121	13,06	8022	36233	0,23	110
Som-b-16*	400	160	62487	19448	57792	8,12	7200	62400	0,13	6
Som-b-17*	500	50	7141	19441	5599	27,54	7365	7118	0,30	324
Som-b-18*	500	100	26258	19442	22404	17,20	3243	26199	0,22	19
Som-b-19*	500	150	56572	20726	50204	12,68	7200	56544	0,04	4
Som-b-20*	500	200	97344	19441	89362	8,93	950	97189	0,15	3

Tabela 4. Resultados SOM-b

$$GAP\ G/C = \left|1 - \frac{G}{C}\right| \tag{19}$$

6. Conclusão e Trabalhos Futuros

Neste trabalho apresentamos um novo modelo de programação linear inteira mista para o Problema da Diversidade Máxima (PDM) baseado no modelo apresentado por (Kuo et al., 1993). Esse novo modelo, denominado F2M, apesar de possuir o mesmo *gap* de relaxação linear, mostrou-se mais rápido que o modelo original, quando resolvido pelo CPLEX. Comparando o modelo F2M com o algoritmo de Takane (Takane, 2011), baseado na Decomposição de Benders, verificamos que o modelo F2M, apesar de não conseguir os mesmos resultados que o algoritmo específico implementado para o PDM, mostrou-se competitivo.

Também desenvolvemos uma heurística baseada na meta-heurística GRASP para o PDM. Essa heurística mostrou-se competitiva encontrando o valor ótimo em todas as instâncias dos conjunto GKD-a e GKD-c. Para as instâncias GKD-b o algoritmo encontrou o valor ótimo para todas as instâncias que conhecemos o valor ótimo. No conjunto Som-b o algoritmo encontrou o ótimo em 10 das 20 instâncias. Nas instâncias que não encontrou o ótimo o resultado calculado ficou a no máximo 0.3% do valor ótimo.

Como trabalhos futuros é possível sugerir uma Decomposição de Benders, baseada no trabalho de Takane (Takane, 2011), utilizando o modelo F2M ao invés do modelo F2 utilizado. Outra possibilidade de trabalho seria utilizar o valor encontrado pelo algoritmo GRASP como solução inicial para o modelo F2M, adiantando o trabalho que o CPLEX teria para encontrar limites inferiores.

Referências

Aringhieri, R., Cordone, R., and Melzani, Y. (2008). Tabu search vs. grasp for the maximum diversity problem. *A Quarterly Journal of Operations Research*, v. 6:45–60.

Benders, J. F. (1962). Partitioning algorithm for solving mixed integer variables programming problems. *Numerische Mathematik*, 4:238–252.

Cox, T. (1993). *Cultural diversity in organizations: theory, research and practice*. Berrett-Koehler Publishers, Inc, San Francisco, CA.

Duarte, A. and Marti, R. (2007). Tabu search for the maximum diversity problem. *Optsicom*, v. 22:180–182.

Feo, T. and Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–135.

Fernandez, J. (1991). *Managing a diverse work force*. Lexington Books, Lexington, MA.

Gallego, M., Duarte, A., Laguna, M., and Martí, M. (2009). The scatter search methodology.

Ghosh, J. (1996). Computational aspects of the maximum diversity problem. *Operations Research Letters*, v. 19:175–191.

Glover, F., Hersh, G., and McMillian, C. (1977). Selecting subset of maximum diversity.

Glover, F., Kuo, C., and Dhir, K. (1995). A discrete optimization model for preserving biological diversity. *Appl. Math. Modelling*, 19(11):696–701.

Glover, F., Kuo, C., and Dhir, K. (1998). Heuristic algorithms for the maximum diversity problem. *Journal of Information and Optimization Sciences*, 19(1):109–132.

Glover, F. and Wolsey, R. (1974). Converting the 0-1 polynomial programming problem to a 0-1 linear programming. *Operations Research*, v. 22:180–182.

Kuo, C., Glover, F., and Dhir, K. S. (1993). Analyzing and modeling the maximum diversity problem by zero-one programming. *Decision Sciences*, v. 24:1171–1185.

Marti, R., Gallego, M., Duarte, A., and Pargo, E. (2010). Heuristics and metaheuristics for the maximum diversity problem. *Optsicom*, v. 22:180–182.

McConnell, S. (1988). The new battle over immigration. *Fortune*, 117:89–102.

Porter, W. M., Rawal, K. M., Rachie, K. O., Wien, H. C., and Williams, R. C. (1975). *Cowpea germplasm catalog no 1*. International Institute of Tropical Agriculture.

Resende, M. and Ribeiro, C. (2003). A GRASP with path-relinking for private virtual circuit routing. *Networks*, 41(1):104–114.

Silva, G., Andrade, M., Ochi, L., Martins, S., and Plastino, A. (2007). New heuristics for the maximum diversity problem. *Journal of Heuristics*, v. 13:315 –336.

Takane, B. (2011). Um algoritmo exato para o problema da diversidade máxima. Master's thesis, Universidade Federal de Minas Gerais.