Universidade Federal do Rio Grande do Sul

Instituto de Informática

Busca Tabu aplicada ao problema de Particionamento de Conjuntos

Arthur Selle Jacobs
asjacobs@inf.ufrgs.br

29 de Agosto de 2016

Conteúdo

1 Introdução

Particionamento de Conjuntos é um dos problemas fundamentais na área de análise combinatório, tendo aplicações nas mais diversas áreas, principalmente em problemas de escalonamento [1]. Sua aplicação mais conhecida é no problema de escalonamento de tripulação, onde um conjunto de n tripulações, com uma lista de possíveis voos e um custo associado, deve ser designada a um conjunto de m voos de uma linha área, de forma que todos os voos possuam uma tripulação [2].

Esse é um exemplo concreto, e economicamente relevante, do problema de Particionamento de Conjuntos. Devido ao alto custo envolvido com viagens aéreas, diversos estudos relacionados a essa questão podem ser encontrados na literatura. Assim, até mesmo melhorias pequenas podem trazer grandes benefícios econômicos a uma empresa [1]. Problemas como esse eram, antigamente, resolvidos à mão. Contudo, devido ao grande crescimento na escala de problemas de escalonamento de tripulação, o uso de computadores para auxiliar sua resolução se tornou necessário. Por esse ser um problema NP-completo, ou seja, não pode ser resolvido em tempo polinomial, diversos algoritmos foram desenvolvidos para tentar encontrar uma solução ótima [1].

Esses algoritmos frequentemente se valem da aplicação de heurísticas de busca e de construção para encontrar resultados satisfatórios. Uma das heurísticas mais conhecidas e utilizadas é a Busca Tabu, proposta oficialmente por Fred Glover, em 1986 [5]. Este relatório apresenta a metodologia e resultados da aplicação da meta-heurística Busca Tabu ao problema de Particionamento de Conjuntos, comparando os resultado obtidos com a resolução do problema utilizando um *solver* genérico.

O restante desse relatório está organizado da seguinte maneira: na seção 2 o problema de Particinamento de Conjuntos é definido; a heurística Busca Tabu é discutida na seção 3; a seção 4 mostra como foi realizada a abordagem do problema utilizando a heurística; os resultados são apresentados na seção 4; o relatório é concluído na seção 5.

2 Particionamento de Conjuntos

Uma instância do problema de Particionamento de Conjuntos consiste em um universo U de n subconjuntos e m elementos, de forma que cada subconjunto j possui de 1 a m elementos e um custo c_j associado. A solução desse problema reside em encontrar um conjunto S de subconjuntos que cubra todos os m elementos, de forma que cada elementos seja coberto por apenas 1 subconjunto, minimizando a soma dos pesos dos subconjuntos selecionados.

Esse problema pode ser formulado com programação inteira da seguinte maneira:

$$c_j \in \mathbb{Z}, \quad \text{custo associado ao subconjunto } j \tag{1}$$

$$x_j = \begin{cases} 1 & \text{se o subconjunto } j \text{ faz parte da solução } S \\ 0 & \text{caso contrário} \end{cases} \tag{2}$$

$$a_{i,j} = \begin{cases} 1 & \text{se o elemento } i \text{ está contido no subconjunto } j \\ 0 & \text{caso contrário} \end{cases} \tag{3}$$

$$\begin{aligned} \text{Minimizar} \quad & \sum_{j=1}^{n} x_j c_j \\ \text{Sujeito a} \quad & \sum_{j=1}^{n} x_j a_{i,j} = 1 \quad \forall i \in 1..m \end{aligned} \tag{4}$$

3 Busca Tabu

A meta-heurística Busca Tabu tem como objetivo guiar outros procedimentos para impedir que eles fiquem limitados a uma mesma solução ótima local. Este método pode ser utilizado para guiar qualquer processo que empregue pequenas modificações a uma solução, e que forneça uma função de avaliação da solução, para que seja possível encontrar a melhor dessas modificações [3], como uma Busca Local. Assim como outras meta-heurísticas baseadas no processo de busca local, como Simulated Annealing (SA), a Busca Tabu também permite movimentos de piora, apesar de escolher sempre a melhor solução vizinha, visto que a melhor solução vizinha pode ter um valor menor que a solução atual [4].

A Busca Tabu tradicionalmente realiza apenas movimentos determinísticos, diferentemente de outras meta-heurísitcas com movimentos randômicos, como SA. Isto é, em geral, a Busca Tabu se vale de uma Busca Local com *Best Improvement*, de forma que a solução recebe sempre o melhor vizinho possível. Entretanto, existem casos onde encontrar o melhor vizinho é extremamente custoso. Para esses casos, é possível adicionar aspectos randômicos ou até mesmo cálculos probabilísticos [4]. Essa meta-heurística se baseia no princípio de manter em memória uma lista de movimentos que provavelmente não levariam a uma solução ótima (e.g. as últimas modificações realizadas). Essa lista de movimentos "proibidos"é chamada de Lista Tabu. O tamanho dessa lista é um parâmetro importante para o algoritmo, de forma que se a lista possui n posições, um movimento ficara "proibido"por n iterações. Existem diversos critérios de parada possíveis para o algoritmo (e.g. quando todos movimentos fazem parte da Lista Tabu, ou se x movimentos foram feitos sem melhora) [5]. O algoritmo de Busca Tabu, em geral, pode ser representado da seguinte maneira.

Considere,

- S_0 a solução inicial.
- S a solução atual.
- S^* a melhor solução conhecida.
- f^* o valor da melhor solução conhecida.
- $N(S)$ a vizinhança da solução S.
- $\tilde{N}(S)$ vizinhança "permitida"de S, ou seja, apenas vizinhos não tabu.
- T a Lista Tabu.

```
1  BuscaTabu()
2  {
3      S_0 := EncontraSolucaoInicial();
4      S := S_0; S* := S_0; f* := f(S_0); T := ∅;
5      enquanto (!<criterio_de_parada>)
6      {
7          Ñ(S) := ∅;
8          para cada (s em N(S))
9          {
10             se (!EhTabu(s, T))
11             {
12                 Ñ(S) := Ñ(S) + s;
13             }
14         }
15         S := EncontraMelhorVizinho(Ñ(S));
16         se (f(S) < f*)
17         {
18             S* := S; f* := f(S);
19         }
20         T := T + S;
21     }
22     retorna S*;
23 }
```

4 Aplicação da Meta-heurística ao Problema

Como descrito na seção anterior, para aplicar a Busta Tabu a um problema, é necessário definir como cada elemento da busca (e.g. solução inicial, vizinhança, etc.) será determinado. Assim, a seguir são definidas tais características para o problema de Particinamento de Conjuntos.

4.1 Representação do Problema

Para representar uma instância do problema de Particionamento de Conjuntos foi utilizado uma lista de vetores de inteiros, respeitando as seguintes regras:

- A lista de vetores representa o conjunto de subconjuntos.
- Cada vetor de inteiros representa um subconjunto, sendo o índice do subconjunto na lista de vetores seu identificador.
- Cada subconjunto tem $m + 1$ posições, sendo m o número elementos em U.
- A posição com índice 0 de cada subconjunto contém seu custo, e o restante das posições contém 1 se o elemento correspondente ao índice faz parte do subconjunto, e 0 caso contrário.

Uma solução, por sua vez, foi representada como dois vetores de inteiros, um contendo os subconjuntos que formam a solução, e outro contendo os elementos cobertos pela solução e a soma dos pesos dos subconjuntos. A representação da solução respeita as seguintes regras:

- O vetor contendo os subconjuntos que fazem parte da solução possui n posições, sendo n o número de subconjuntos existentes na instancia.

- Cada posição do vetor de subconjuntos recebe 1 caso o subconjunto faça parte da solução, e 0 caso contrário.
- O vetor contendo os elementos cobertos pela solução tem $m + 1$ posições, sendo m o número elementos em U.
- A posição com índice 0 do vetor contendo os elementos cobertos contém a soma dos custos dos subconjuntos que formam a solução.
- O restante das posições do vetor de elementos contém 1 se o elemento correspondente ao índice está coberto pela solução, e 0 caso contrário.

4.2 Solução Inicial

Como encontrar a solução inicial deste problema, por si só, já é um problema NP-Completo, não foi possível realizar um busca exaustiva para encontrar uma solução qualquer. Logo a abordagem utilizada foi a seguinte:

1. Para cada elemento i não coberto pela solução, faz uma lista dos subconjuntos que cobrem tal elemento.
2. Seleciona o subconjunto, que cobre o elemento i, com menor custo, e o adiciona à solução inicial.
3. Repita este processo até que todos os m elementos estejam cobertos.

Note que este procedimento não considera o caso onde dois subconjuntos cobrem o mesmo elemento, apenas levando em conta os custos de cada subconjuntos. Logo, esta solução inicial será muito provavelmente inviável. Para tentar amenizar este problema, após a utilização deste procedimento, é aplicado um operador heurístico de viabilidade, proposto por P.C. Chu, J.E. Beasley na seção 3.3 de [2]. A utilização deste operador não garante uma solução viável; contudo, aumenta significativamente a viabilidade da solução, se não a torna completamente viável. Esta heurística segue os seguintes passos:

1. Seleciona aleatoriamente um subconjunto que faz parte da solução.
2. Se o subconjunto contém algum elemento que é coberto mais de uma vez, o subconjunto é removido da solução.
3. Repete os passos 1 e 2 até que todos os subconjuntos da solução tenham sido analisados.
4. Seleciona aleatoriamente um elemento que não está coberto pela solução.
5. Dentre todos os subconjuntos que possuem o elemento selecionado, e não possuem nenhum outro elemento já coberto pela solução, seleciona o que tiver o menor custo e o adiciona à solução. Caso não exista nenhum subconjunto que satisfaça esses critérios, não faz nada.
6. Repete os passos 4 e 5 até que todos os elementos não cobertos tenham sido analisados.

4.3 Vizinhança

Como mencionado na seção anterior, normalmente a Busca Tabu utiliza uma vizinhança determinística, analisando todos os possíveis vizinhos para encontrar o melhor possível. Esta abordagem de *Best Improvement* apresenta resultados satisfatórios para instancias pequenas (i.e. instâncias com até 10 mil subconjuntos); entretanto, para instâncias maiores (i.e.instâncias com 100 mil ou mais subconjuntos) torna o algoritmo extremamente lento. Ainda assim, os resultados obtidos por outros métodos, como a adição de aspectos aleatórios, obtiverem resultados muito piores que a abordagem *Best Improvement*. Assim, a vizinhança de soluções foi determinada da seguinte maneira:

1. Seleciona um subconjunto pertence à solução atual.
2. Para cada subconjunto que não faz parte da solução, verifica que se o vizinho está na Lista Tabu: se sim, pula para próximo vizinho; se não, troca o subconjunto da solução com o que não faz parte da solução e adiciona vizinho à vizinhança.
3. Repete os itens 2 e 3 até que todas as combinações sejam testadas.

Com a vizinhança formada, foi preciso então avaliar cada solução pertencente à vizinhança para determinar qual a melhor encontrada. Além disso, como a implementação permite soluções inviáveis, é preciso punir de alguma forma soluções "mais inviáveis". Para avaliar cada solução, foi utilizada a seguinte equação:

$$f_k = \sum_{j=1}^{n} c_j \times (1 + \sum_{i=1}^{m} \bar{b}_{i,k}) \quad \forall k \in \tilde{N}(S) \tag{5}$$

Onde,

- f_k é o valor da solução vizinha k.
- c_j é o custo do subconjunto j.
- $\bar{b}_{i,k}$ recebe 1 se o elemento i não é coberto pela solução k, ou se é multiplamente coberto.
- $\tilde{N}(S)$ vizinhança "permitida"de S, ou seja, apenas vizinhos não tabu.

Assim, ao procurar a melhor solução vizinha, é selecionado o vizinho que apresente o menor valor. Desta forma, soluções viáveis apresentaram como valor apenas a soma dos pesos dos subconjuntos que as formam, enquanto soluções inviáveis serão penalizadas com um fator correspondente ao número de elementos não cobertos ou cobertos por mais de um subconjunto. Acatamos, então, o objetivo de tentar minimizar a soma dos pesos dos subconjuntos da solução.

5 Testes

5.1 Calibragem de Parâmetros

A meta-heurísta implementada recebe dois parâmetros que influenciam diretamente no resultado obtido: tamanho da Lista Tabu e número máximo de

iterações sem melhora. Buscando um melhor desempenho, foi necessário calibrar tais parâmetros, de forma a encontrar uma combinação ideal de valores. A calibragem foi realizada utilizando três instâncias de tamanho relativamente pequenos (i.e. menos de mil subconjuntos), por demandarem um tempo menor de execução, apesar de, eventualmente, gerar valores não ideais para instâncias maiores. Os resultados obtidos durantes os testes podem ser verificados, para cada instancia na tabelas 1, 2 e 3.

Considere:

- T.L.T o tamanho da Lista Tabu.
- It. Max. o máximo de iterações sem melhora.
- S.L. o valor da solução inicial encontrada.
- S.F. o valor da solução final encontrada.
- T. Exec. o tempo de execução total, em segundos.
- Num. It. o número total de iterações.
- (V) e (I), ao lado do valor da solução correspondem, respectivamente, a uma solução viável ou inviável.

Existem ainda mais dois possíveis parâmetros que influenciam o resultado da Busca Tabu: *i* tempo máximo de execução da meta-heurística (em segundos), e *ii* a semente aleatória para o operador de viabilidade utilizado para encontrar a solução inicial. Ambos parâmetros mantiveram valores constantes durante a calibragem (tempo máximo de 900 segundos, e semente aleatória 515151), com o intuito de analisar isoladamente os parâmetros mais importante para a meta-heurística.

Ao analisar os resultados obtidos por esta calibragem, podemos perceber que soluções com um limite de iterações muito baixo tendem a não avaliar o espaço de solução devidamente, encerrando sem melhorar a solução inicial. Contudo, soluções com um limite muito alto tendem a ter um tempo de execução mais alto. Além disso, é importante notar que soluções onde a Lista Tabu assumiu tamanhos 20 ou 30, obtiveram sempre os menores resultados encontrados, logo seria ideal que ela assumisse um desses valores.

Desta maneira, a combinação de valores para estes parâmetros escolhida foi:

- Tamanho da Lista Tabu := 30.
- Número máximo de iterações sem melhora := 200.

5.2 Resultados

Com os principais parâmetros da meta-heurística calibrados, foi possível então testar a eficiência da implementação com diversas instâncias. Cada instância foi executada 5 vezes, modificando, em cada execução, a semente aleatória utilizada para gerar a solução inicial, mantendo sempre um tempo máximo de 900 segundos de execução. O tamanho da Lista Tabu e o número máximo de iterações sem melhora mantiveram sempre seus valores calibrados.

Instãncia: sppnw41						
	T.L.T.	It. Max.	S.I.	S.F.	T. Exec. (seg)	Num. It.
Config. 1	10	50	19449 (V)	18576 (V)	4,17	64
	20	50	19449 (V)	17592 (V)	8,44	114
	30	50	19449 (V)	17835 (V)	4,11	87
	40	50	19449 (V)	17781 (V)	12,44	109
	50	50	19449 (V)	17835 (V)	13,47	81
Config. 2	10	100	19449 (V)	17409 (V)	10,56	231
	20	100	19449 (V)	17592 (V)	9,62	164
	30	100	19449 (V)	17544 (V)	9,02	213
	40	100	19449 (V)	17781 (V)	14,80	159
	50	100	19449 (V)	17835 (V)	15,22	131
Config. 3	10	200	19449 (V)	17409 (V)	17,51	331
	20	200	19449 (V)	17592 (V)	16,54	264
	30	200	19449 (V)	17544 (V)	15,05	313
	40	200	19449 (V)	17781 (V)	43.45	416
	50	200	19449 (V)	17835 (V)	30,37	231
Config. 4	10	500	19449 (V)	17409 (V)	59,99	631
	20	500	19449 (V)	17592 (V)	70,78	564
	30	500	19449 (V)	17544 (V)	45,25	613
	40	500	19449 (V)	17718 (V)	139,04	716
	50	500	19449 (V)	17835 (V)	107,33	531
Config. 5	10	1000	19449 (V)	17409 (V)	58,63	1131
	20	1000	19449 (V)	17592 (V)	68,87	1064
	30	1000	19449 (V)	17544 (V)	50,81	1113
	40	1000	19449 (V)	17718 (V)	14,80	1216
	50	1000	19449 (V)	17409 (V)	293,31	2004

Tabela 1: Calibragem para instancia sppnw41.

Instância: sppnw34						
	T.L.T.	It. Max.	S.I.	S.F.	T. Exec. (seg)	Num. It.
Config. 1	10	50	20838 (V)	20838 (V)	74,754	50
	20	50	20838 (V)	20838 (V)	42,516	50
	30	50	20838 (V)	20838 (V)	20,587	50
	40	50	20838 (V)	20838 (V)	54,558	50
	50	50	20838 (V)	20838 (V)	137,671	50
Config. 2	10	100	20838 (V)	20838 (V)	113,992	100
	20	100	20838 (V)	19860 (V)	145,846	187
	30	100	20838 (V)	20838 (V)	39,324	100
	40	100	20838 (V)	20838 (V)	144,064	100
	50	100	20838 (V)	20838 (V)	290,291	100
Config. 3	10	200	20838 (V)	20838 (V)	235,712	200
	20	200	20838 (V)	19860 (V)	222,122	287
	30	200	20838 (V)	19482 (V)	162,012	472
	40	200	20838 (V)	20838 (V)	460,053	200
	50	200	20838 (V)	20838 (V)	313,174	200
Config. 4	10	500	20838 (V)	20838 (V)	791,837	500
	20	500	20838 (V)	19860 (V)	627,814	587
	30	500	20838 (V)	19482 (V)	375,567	772
	40	500	20838 (V)	19482 (V)	900,41	273
	50	500	20838 (V)	20838 (V)	900,526	212
Config. 5	10	1000	20838 (V)	20838 (V)	900,239	600
	20	1000	20838 (V)	19860 (V)	708,68	1087
	30	1000	20838 (V)	19482 (V)	479,208	1272
	40	1000	20838 (V)	19482 (V)	901,628	333
	50	1000	20838 (V)	19482 (V)	901,53	588

Tabela 2: Calibragem para instância sppnw34.

Instância: sppnw32						
	T.L.T.	It. Max.	S.I.	S.F.	T. Exec. (seg)	Num. It.
Config. 1	10	50	19314 (V)	18816 (V)	7,129	56
	20	50	19314 (V)	18816 (V)	46,674	56
	30	50	19314 (V)	18816 (V)	17,758	56
	40	50	19314 (V)	18816 (V)	14,922	56
	50	50	19314 (V)	18816 (V)	9,402	56
Config. 2	10	100	19314 (V)	18816 (V)	5,785	106
	20	100	19314 (V)	18816 (V)	76,827	106
	30	100	19314 (V)	18816 (V)	24,296	106
	40	100	19314 (V)	18816 (V)	23,68	106
	50	100	19314 (V)	18816 (V)	15,607	106
Config. 3	10	200	19314 (V)	18816 (V)	12,036	206
	20	200	19314 (V)	18816 (V)	187,181	206
	30	200	19314 (V)	18816 (V)	46,351	206
	40	200	19314 (V)	18816 (V)	47,906	206
	50	200	19314 (V)	18816 (V)	37,288	206
Config. 4	10	500	19314 (V)	18816 (V)	42,266	506
	20	500	19314 (V)	18816 (V)	742,318	506
	30	500	19314 (V)	18816 (V)	171,505	506
	40	500	19314 (V)	18816 (V)	150,423	506
	50	500	19314 (V)	18816 (V)	149,486	506
Config. 5	10	1000	19314 (V)	18816 (V)	58,507	1006
	20	1000	19314 (V)	18816 (V)	899,45	1006
	30	1000	19314 (V)	18816 (V)	203,441	1006
	40	1000	19314 (V)	18816 (V)	193,715	1006
	50	1000	19314 (V)	18816 (V)	208,651	1006

Tabela 3: Calibragem para instância sppnw32.

Os resultados obtidos estão exibidos na tabela 4, realizando uma comparação com resultados obtidos por um *solver* genérico GLPK.

Considere:

- S.L. o valor da solução inicial encontrada.
- S.F. o valor da solução final encontrada.
- (V) e (I), ao lado do valor da solução correspondem, respectivamente, a uma solução viável ou inviável.
- Solver corresponde ao valor encontrados pelo solver genérico GLPK.
- M.S.C a melhor solução conhecida para a instância.
- Desvio (M.S.C.) o desvio percentual entre a melhor solução conhecida e a solução encontrada.
- T. Exec. o tempo de execução total, em segundos.
- Num. It. o número total de iterações.
- Semente a semente aleatória utilizada para encontrar a solução inicial.
- $\emptyset$ representando a insolubilidade por falta de memória.

6 Conclusão

O problema de Particionamento de Conjuntos, um problema NP-completo, normalmente não pode ser resolvido em tempo polinomial. Entretanto, métodos heurísticos e meta-heurísticos, como a Busca Tabu, conseguem, através de aproximações e buscas locais, encontrar soluções muito próximas da otimalidade.

Analisando os resultados obtidos da aplicação da Busca Tabu ao problema de Particionamento de Conjunto, é possível notar que, de uma forma geral, a solução encontrada pela meta-heurísrica não se aproxima muito da melhor solução conhecida. Provavelmente, isso deve-se ao fato de a calibração dos parâmetros ter sido feita apenas com instâncias pequenas. Desta forma, para instâncias maiores e complexas como *meteor* e *heart*, não foi possível nem encontrar uma solução viável.

Todavia, em casos como *sppnw05* e *sppnw16*, nos quais o solver genérico GLPK, utilizando métodos de programação inteira, não conseguiu resolver a instância, a meta-heurística, apesar de lenta, conseguiu encontrar resultados bem próximos da melhor solução ótima conhecida. Isso mostra a importância e a capacidade destes métodos heurísticos em resolver problemas NP-Completo em um tempo relativamente pequeno.

Resultados obtidos								
Instância	S.I.	S.F.	Solver	M.S.C.	Desvio (M.S.C.)	T. Exec. (seg).	Num. It.	Semente
sppnw41	19449 (V)	17544 (V)	11307	11307	65%	11.26	313	515151
	16788 (V)	15690 (V)	11307	11307	72%	8.46	257	159753
	14511 (V)	14031 (V)	11307	11307	80%	6.65	209	333333
	16425 (V)	15750 (V)	11307	11307	71%	7.94	215	299001
	16719 (V)	16422 (V)	11307	11307	67%	8.77	247	100992
sppnw32	19314 (V)	18816 (V)	14877	14877	79%	16.63	206	515151
	19314 (V)	18816 (V)	14877	14877	79%	16.61	206	159753
	19626 (V)	18816 (V)	14877	14877	79%	19.02	220	333333
	17400 (V)	17097 (V)	14877	14877	87%	18.70	209	299001
	19626 (V)	18816 (V)	14877	14877	79%	21.36	220	100992
sppnw34	20838 (V)	19482 (V)	10488	10488	54%	209.77	472	515151
	12786 (I)	14439 (V)	10488	10488	73%	79.18	241	159753
	19590 (V)	18561 (V)	10488	10488	56%	116.05	233	333333
	16461 (I)	19527 (V)	10488	10488	53%	341.19	756	299001
	15942 (I)	17916 (V)	10488	10488	58%	207.90	423	100992
heart	135 (I)	135 (I)	180	180	—	202.89	200	515151
	144 (I)	144 (I)	180	180	—	219.25	200	159753
	135 (I)	135 (I)	180	180	—	224.32	200	333333
	126 (I)	126 (I)	180	180	—	208.07	200	299001
	135 (I)	135 (I)	180	180	—	219.93	200	100992
delta	90 (I)	90 (I)	126	126	—	83.51	200	515151
	90 (I)	90 (I)	126	126	—	82.95	200	159753
	81 (I)	81 (I)	126	126	—	64.01	200	333333
	81 (I)	81 (I)	126	126	—	67.01	200	299001
	90 (I)	90 (I)	126	126	—	82.38	200	100992
sppnw36	5764 (I)	9454 (V)	7314	7314	77%	83.47	414	515151
	7106 (I)	8862 (V)	7314	7314	82%	65	299	159753
	8558 (I)	8328 (V)	7314	7314	87%	44.80	247	333333
	4126 (I)	8246 (V)	7314	7314	86%	62.61	350	299001
	6332 (I)	8952 (V)	7314	7314	81%	140.44	674	100992
meteor	42 (I)	42 (I)	60	60	—	86.53	200	515151
	42 (I)	42 (I)	60	60	—	104.67	200	159753
	42 (I)	42 (I)	60	60	—	98.02	200	333333
	42 (I)	42 (I)	60	60	—	97.46	200	299001
	36 (I)	36 (I)	60	60	—	82.08	200	100992
sppaa06	18014 (I)	18858 (I)	27040	27040	—	900	11	515151
	20409 (I)	20495 (I)	27040	27040	—	900	11	159753
	18400 (I)	18559 (I)	27040	27040	—	900	11	333333
	18666 (I)	15829 (I)	27040	27040	—	900	12	299001
	19259 (I)	20559 (I)	27040	27040	—	900	11	100992
sppnw05	211960 (V)	211960 (V)	∅	132878	62%	900	1	515151
	180098 (V)	180098 (V)	∅	132878	73%	900	1	159753
	267448 (V)	267448 (V)	∅	132878	50%	900	1	333333
	219854 (V)	219854 (V)	∅	132878	60%	900	1	299001
	220386 (V)	220386 (V)	∅	132878	60%	900	1	100992
sppnw16	1241460 (V)	1241460 (V)	∅	1181590	95%	900	1	515151
	1241342 (V)	1241342 (V)	∅	1181590	95%	900	1	159753
	1269486 (V)	1269486 (V)	∅	1181590	93%	900	1	333333
	1260070 (V)	1260070 (V)	∅	1181590	93%	900	1	299001
	1231948 (V)	1231948 (V)	∅	1181590	95%	900	1	100992
sppus01	9065 (I)	9065 (I)	∅	10036	—	900	1	515151
	12690 (I)	13121 (I)	∅	10036	—	900	1	159753
	10695 (I)	10695 (I)	∅	10036	—	900	1	333333
	9630 (I)	9746 (I)	∅	10036	—	900	1	299001
	103272 (I)	10601 (I)	∅	10036	—	900	1	100992

Tabela 4: Resultados dos teste para cada instância.

Referências

[1] David Levine *A Parallel Genetic Algorithm for the Set Partitioning Problem*. Argonne National Laboratory, Maio de 1994.

[2] P.C. Chu, J.E. Beasley *A Genetic Algorithm for the Set Partitioning Problem*. Imperial College, Abril de 1995.

[3] Fred Glover *Tabu Search: A Tutorial*. Junho-Agosto de 1990.

[4] Michel Gendreau, Jean-Yves Potvin *Search Methodologies, Chapter 6 - Tabu Search*. Editado por Edmund K. Burke e Graham Kendall, Springer. Junho de 2005.

[5] Marcus Ritt, Luciana Buriol. *INF05010 — Otimização combinatória. Notas de aula*. 2013/08/07.