

# Capítulo IX

## Resultados computacionais

### IX.1 - Introdução:

Apresentamos neste capítulo alguns dos resultados computacionais obtidos em nossa implementação. Fazemos uma análise de algumas das instâncias disponíveis na literatura para o problema de roteamento de veículos com restrições de capacidade (vide Reinelt [91]).

Na seção IX.2, apresentamos 3 versões distintas na determinação dos limites inferiores. Na primeira versão (representada por S), utilizamos apenas as desigualdades de eliminação de sub-rotas. Na segunda versão (representada por SC), utilizamos simultaneamente as desigualdades de eliminação de sub-rotas e as desigualdades *comb*. Finalmente, na última versão (representadas por SCM), utilizamos as desigualdades de eliminação de sub-rotas, *combs* e *multistars* respectivamente. Estas 3 versões foram implementadas com a utilização de uma máquina RISK 6000 190.

Fazemos uma comparação entre alguns dos limites inferiores obtidos em nossa abordagem com as abordagens desenvolvidas por Fisher[94.b] (capítulo IV) e Miller[95] (capítulo I). Em seguida, fazemos uma avaliação de nossa heurística lagrangeana (C&W lagrangeano) e do procedimento que fixa variáveis.

Na seção IX.3, apresentamos alguns resultados computacionais obtidos por nossa implementação do algoritmo *branch-and-bound*. Para as instâncias aqui resolvidas, apresentamos o número total de nós gerados na árvore de busca bem como o tempo total de processamento utilizado.

## IX.2 - Resultados computacionais (limites inferiores):

Nesta seção, são apresentados alguns dos resultados obtidos em nossa implementação na determinação de limites inferiores para o problema de roteamento.

A tabela da figura IX.1, apresenta limites inferiores para alguns dos problemas testes da biblioteca TSPLIB (Reinelt[91]). Fazemos uma “comparação” entre os limites inferiores obtidos por Fisher[94.b] e as 3 versões distintas de nosso algoritmo, envolvendo, sub-rotas (S); sub-rotas e *combs* (SC); sub-rotas, *combs* e *multistars* (SCM) respectivamente. Os valores numéricos presentes nos nomes de cada instância (coluna (1)), indicam o número total de clientes a serem atendidos por cada frota de veículos. A quantidade de veículos em cada frota é representada na coluna (2). Os melhores limites superiores obtidos na literatura e associados a cada instância estão representados na coluna (7). Algumas soluções ótimas estão assinaladas com (\*). Consideramos neste caso, assim como em Fisher[94.b], apenas distâncias euclidianas entre cada um dos clientes e o depósito.

(1)	(2)	(3)	(4)	(5)	(6)	(7)
Prob.	num. de veículos	Fisher	(S)	(SC)	(SCM)	Limite superior
c50.dat	5	507.09	511.95	513.66	515.17	524.61 (*)
c75.dat	10	755.50	765.48	765.56	765.06	835.26
c100.dat	8	785.86	793.65	795.36	791.52	826.14
c150.dat	12	932.68	950.86	954.61	950.90	1028.42
c199.dat	16	1096.72	1149.70	1148.50	1146.63	1294.25
c100b.dat	10	817.77	817.54	817.66	817.85	819.56 (*)
f44.dat	4	720.76	721.20	722.55	721.68	723.54 (*)
f71.dat	4	237.76	238.51	238.44	238.65	241.97 (*)
f134.dat	7	1133.73	1154.22	1152.21	1150.70	1162.96

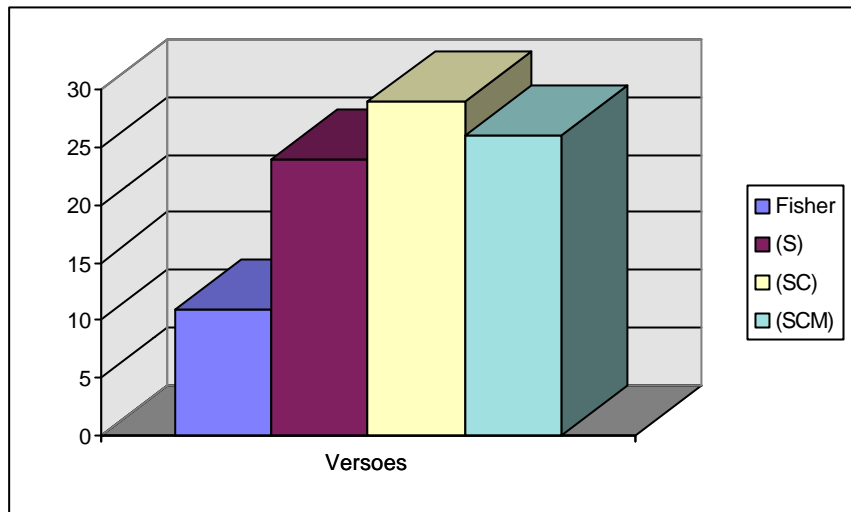
**Figura IX.1: Limites inferiores (distância euclidianas)**

Como discutido no capítulo VIII (seção VIII.2), devemos, no método subgradiente, atribuir diferentes valores a seus parâmetros. No método subgradiente implementado por Fisher, ele utiliza uma redução de 75% no tamanho do passo a cada 30 iterações do subgradiente sem atualização do limite inferior. Em nosso caso, consideramos uma redução de 85% no tamanho do passo a cada 50 iterações sem atualização do limite inferior. Fica difícil, neste caso, estabelecer quais os parâmetros ideais (no subgradiente) para o conjunto de problemas testes analisados. Consideramos taxas de redução (no tamanho do passo) que oscilavam entre 60% a 85%, e um número total de iterações (sem atualização dos limites inferiores) variando entre 40 e 60 unidades. Em todas as situações analisadas, obtivemos bons limites inferiores quando comparados àqueles obtidos por Fisher[94.b].

Utilizamos o mesmo número de iterações no subgradiente que nos testes realizados por Fisher[94.b]. As instâncias iniciadas pela letra *c* foram executadas em 3000 iterações do subgradiente, enquanto que as instâncias iniciadas por *f* foram utilizadas 2000 iterações.

Observando a tabela da figura IX.1, podemos constatar uma melhora dos limites inferiores, por nós obtidos, em quase todas as instâncias analisadas. Na coluna (3), temos os limites inferiores gerados por Fisher[94.b] com a utilização das restrições de eliminação de sub-rotas. Note que os resultados apresentados na coluna (4), mostram uma superioridade de nossa versão que utiliza também as desigualdades de eliminação de sub-rotas. Como discutido anteriormente, isto se deve ao fato de trabalharmos com um número maior de restrições candidatas à dualização. Na verdade, utilizamos uma formulação mais forte para o problema de roteamento que aquela apresentada por Fisher[94.b] (vide capítulo VI/ seção VI.2.1).

No gráfico da figura IX.2, buscamos fazer uma “avaliação” do desempenho das 4 versões implementadas para os problemas testes da figura IX.1. Colocamos em ordem crescente os limites inferiores obtidos e atribuímos (para cada uma das versões) pesos crescentes que variam de 1 a 4. O gráfico da figura IX.2, mostra o somatório destes pesos para cada uma das versões implementadas. Quanto maior o somatório dos pesos, melhor a qualidade dos limites inferiores obtidos para o conjunto das instâncias avaliada.



**Figura IX.2: Somatório dos “pesos” de cada versão (limites inferiores)**

Como discutido no capítulo III, ao menos teoricamente, a introdução de novas classes de restrições sempre proporciona novos limites inferiores maiores ou iguais aos anteriores. Em nosso caso entretanto, a introdução de novas classes de restrições (*combs* e *multistars*) não garante sempre um incremento estrito do limite inferior. Em função da lenta “convergência” do subgradiente para uma solução do problema dual lagrangeano, devemos estipular um número máximo de iterações para sua execução. Isto pode levar a situações onde a utilização de um número maior de classes de restrições não garanta uma melhora dos limites inferiores ao final do método subgradiente.

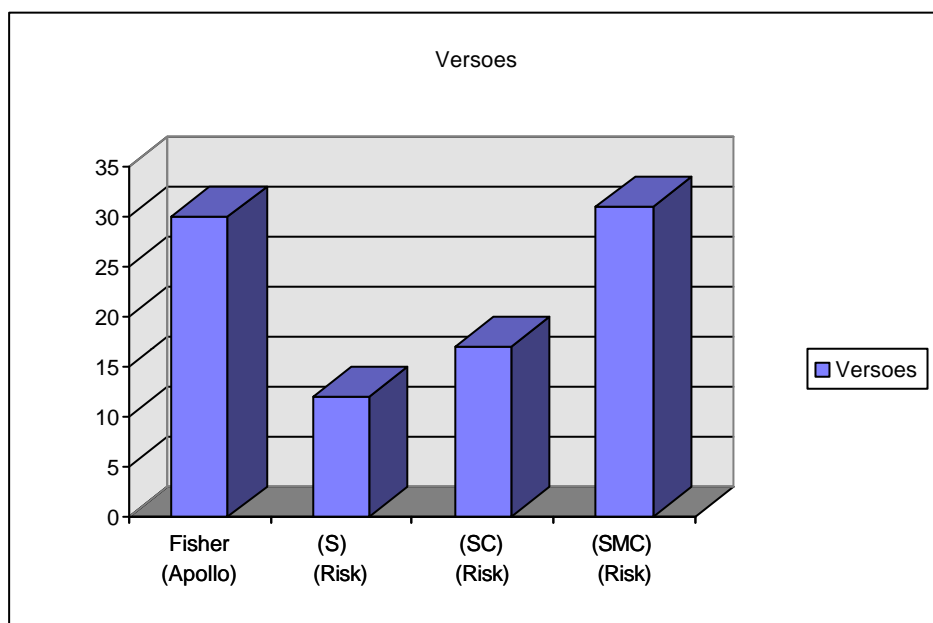
Na tabela da figura IX.3, apresentamos os tempos de processamento, calculados em minutos, para cada uma das instâncias da tabela IX.1. Vale ressaltar que as instâncias tratadas por Fisher (coluna 2) foram implementadas em uma máquina Apollo Domain 3000, o que dificulta a comparação entre os tempos de processamento.

(1)	(2)	(3)	(4)	(5)
Prob.	Fisher	(S)	(SC)	(SCM)
c50.dat	95.75	8.92	9.82	13.98
c75.dat	183.97	18.31	20.28	49.80
c100.dat	307.95	106.37	76.62	183.13

c150.dat	682.41	238.24	296.27	810.10
c199.dat	1186.00	504.24	568.97	2131.43
c100b.dat	259.64	55.13	66.55	136.00
f44.dat	49.74	20.47	13.97	51.67
f71.dat	105.33	102.05	102.77	169.47
f134.dat	253.84	635.68	711.00	1018.92

**Figura IX.3: Tempo de processamento em minutos**

Analogamente ao gráfico da figura IX.2, fazemos uma “avaliação” dos tempos de processamento para as 4 versões apresentadas. Colocamos os tempos de processamento em ordem crescente e atribuímos pesos que variam de 1 a 4 nesta ordem. O gráfico da figura IX.4 representa o somatório dos “pesos” para cada umas das versões analisadas. Quanto menor o somatório dos pesos, menor o tempo de processamento utilizado por cada versão.



**Figura IX.4: Somatório dos “pesos” de cada versão (tempo em minutos)**

Note que a versão (S), levando em consideração a diferença entre as máquinas utilizadas, tem o menor tempo de processamento entre as 4 abordagens apresentadas.

Na tabela da figura IX.5, apresentamos os limites inferiores obtidos por nossa implementação, considerando agora, distâncias inteiras entre os clientes. Sempre que a distância euclidiana entre dois vértices for um valor fracionário  $x$ , ela será aproximada para o inteiro mais próximo (para  $\lfloor x \rfloor$  ou  $\lceil x \rceil$  respectivamente). Em todas as instâncias assinaladas com (\*), foram utilizadas 3000 iterações do subgradiente. Para as demais instâncias utilizamos 2000 iterações.

Problemas	Subt. (S)	S+C (SC)	S+C+M (SCM)	Limites Superiores
c50.dat (*)	510	511	512	521
c75.dat (*)	759	761	762	830
c100.dat (*)	784	781	782	815
c150.dat (*)	938	939	943	1015
c199.dat (*)	1138	1121	1120	1280
c100b.dat (*)	819	820	818	820
f44.dat	723	723	722	724
f71.dat	233	233	233	237
f134.dat	1152	1153	1126	1162

**Figura IX.5: Limites inferiores para instâncias “normalizadas” (dist. inteiras)**

Todos os problemas apresentados acima, foram executados utilizando-se uma redução de 75% no tamanho do passo a cada 50 iterações do subgradiente sem atualização do limite inferior.

Na tabela da figura IX.6 a seguir, fazemos uma “comparação” entre nossa abordagem, que utiliza K-árvores mínimas, com a abordagem apresentada por Miller[95], que trabalha com o problema do *b-matching* na geração dos limites inferiores (vide capítulo I). Analogamente à tabela da figura IX.5, consideramos apenas distâncias inteiras entre os clientes e o depósito. Utilizamos novamente, uma redução de 75% no tamanho do passo, a cada 50 iterações do subgradiente sem atualização do limite inferior. Miller[95] por sua vez, utiliza um tamanho de passo que varia em função do número de atualizações dos limites inferiores.

Os valores numéricos presentes no nome das instâncias abaixo representam o número total de vértices associado ao problema (incluindo clientes e depósito).

(1)	(2)	(3)	(4)	(5)	(6)
Problemas	Miller	(S)	(SC)	(SCM)	Sol. ótima
eil7.vrp	114	114	114	114	114
eil13.vrp	268	277	277	277	277
eil22.vrp	374	375	375	375	375
eil23.vrp	569	569	569	569	569
eil30.vrp	509	509	509	509	534
eil33.vrp	826	830	830	831	835
eil51.vrp	505	510	511	512	521

**Figura IX.6: Limites Inferiores (distâncias “normalizadas”)**

Apesar dos resultados numéricos satisfatórios obtidos por nossa abordagem, (apresentados acima), não podemos concluir diretamente que a relaxação apresentada por Miller[95] na geração dos limites inferiores (*b-matching*) seja inferior à relaxação que trabalha com a K-árvores mínimas. Em seu trabalho, Miller[95] utiliza uma variação do algoritmo subgradiente na geração dos limites inferiores. Desta maneira, vários fatores podem influir na qualidade da relaxação utilizada.

A tabela seguinte, da figura IX.7, representa a porcentagem de variáveis fixadas em *zero* e *um* respectivamente por nosso procedimento que fixa variáveis (capítulo VII). Os limites superiores utilizados foram tomados de problemas testes da literatura (biblioteca TSPLIB). Trabalhamos neste caso, apenas com distâncias euclidianas entre cada par de vértices associado ao problema. Executamos o subgradiente empregando somente as restrições de eliminação de sub-rotas (versão (S)) e adotando uma taxa de redução de 75% no tamanho do passo a cada 50 iterações sem incremento do limite inferior.

As porcentagens de 0's e 1's foram calculadas, respectivamente, sobre o número máximo de variáveis *nulas* e iguais a *um* no grafo completo associado. Note que, se  $K$  é o total de veículos da frota, então  $n_z = |E| - (|V| + K)$  é o número máximo de variáveis que podem ser fixadas em *zero* e  $n_u = |V| + K$ , o número máximo de variáveis que podem ser fixadas em *um*.

Problemas	Limites inferiores (S)	Limites superiores	Porcent. de 0's	Porcent. de 1's
eil22.dat	375.28	375.28	91.19%	100%
eil23.dat	568.56	568.56	94.34%	100%
c50.dat	513.32	524.61	70.25%	0%
c75.dat	768.81	835.26	0%	0%
c100.dat	792.43	826.14	23.75%	0%
c100b.dat	818.07	819.56	93.89%	1.11%
c120.dat	1008.01	1042.11	8.88%	0%
f44.dat	723.24	723.54	93.49%	32.50%
f71.dat	238.63	241.97	76.84%	0%
tai75c.dat	1238.48	1291.01	4.65%	0%
tai75d.dat	1346.56	1365.46	56.11%	0%
tai100a.dat	1934.45	2047.90	0%	0%
tai100b.dat	1851.35	1940.61	0%	0%
tai100c.dat	1372.27	1406.20	9.71%	0%
tai100d.dat	1476.06	1581.25	0%	0%

**Figura IX.7: Porcentagem de variáveis fixadas em zero e um**

Os valores numéricos presentes no nome de cada instância, à exceção dos 2 primeiros problemas, representam o número de clientes a serem atendidos. As instâncias eil22.dat e eil23.dat, possuem 21 e 22 clientes respectivamente.

Observe que ocorre uma grande irregularidade com relação a quantidade de variáveis fixadas em *zero* e *um* respectivamente. No problema c100b.dat por exemplo (com 100 clientes), fixamos 93.89% de suas variáveis em *zero*, enquanto que no problema c75.dat (com 75 clientes) não fixamos nenhuma variável!

Finalizamos esta seção apresentado a tabela da figura IX.8. Fazemos uma comparação entre nossa heurística lagrangeana (C&W lagrangeano) e uma das heurísticas lagrangeanas apresentadas por Fisher[94.b] (seção IV.5 / figura IV.5). Consideramos apenas distâncias euclidianas entre os vértices associados ao problema.

(1)	(2)	(3)	(4)	(5)	(6)
Prob.	Heur. Fisher	(S)	(SC)	(SCM)	Limite superior
c50.dat	533.60	553.55	548.04	554.60	524.61
c75.dat	901.09	860.87	857.02	863.57	835.26
c100.dat	856.13	854.22	851.85	855.43	826.14
c150.dat	1150.20	1103.36	1097.93	1079.10	1028.42
c199.dat	1373.20	1374.88	1382.53	1386.83	1294.25
c100b.dat	924.39	819.56*	823.86	821.11	819.56
f44.dat	727.12	723.54*	723.54*	723.54*	723.54
f71.dat	257.11	248.25	253.58	248.54	241.97
f134.dat	1201.31	1192.24	1197.30	1197.30	1162.96

**Figura IX.8: Avaliação do C&W lagrangeano**

Observamos, durante a execução de nosso algoritmo, que os limites superiores eram atualizados, quase sempre, nas primeiras iterações do subgradiente. A frequência de atualizações é menor a medida que os multiplicadores de Lagrange vão se “aproximando” dos multiplicadores ótimos duais. Desta forma, a variação dos custos lagrangeanos diminui e a geração de novas soluções heurísticas (de menor custo) se torna mais difícil. Para evitar esse esforço computacional desnecessário, executamos a heurística lagrangeana até a metade das iterações do subgradiente.

### **IX.3 - Resultados computacionais (*Branch-and-bound*):**

Apresentamos nesta seção alguns resultados computacionais obtidos por nossa implementação na determinação de soluções exatas para o problema de roteamento. Como discutido anteriormente, utilizamos um *branching* binário e fazemos uma busca em árvore selecionando sempre a folha associada ao menor limite inferior da lista de abertos (*best-first*). Depois de selecionada a folha a ser pesquisada, executamos o subgradiente calculando,

paralelamente, os limites superiores (C&W lagrangeano) e o procedimento que fixa variáveis, buscando sempre uma diminuição no tempo total de processamento.

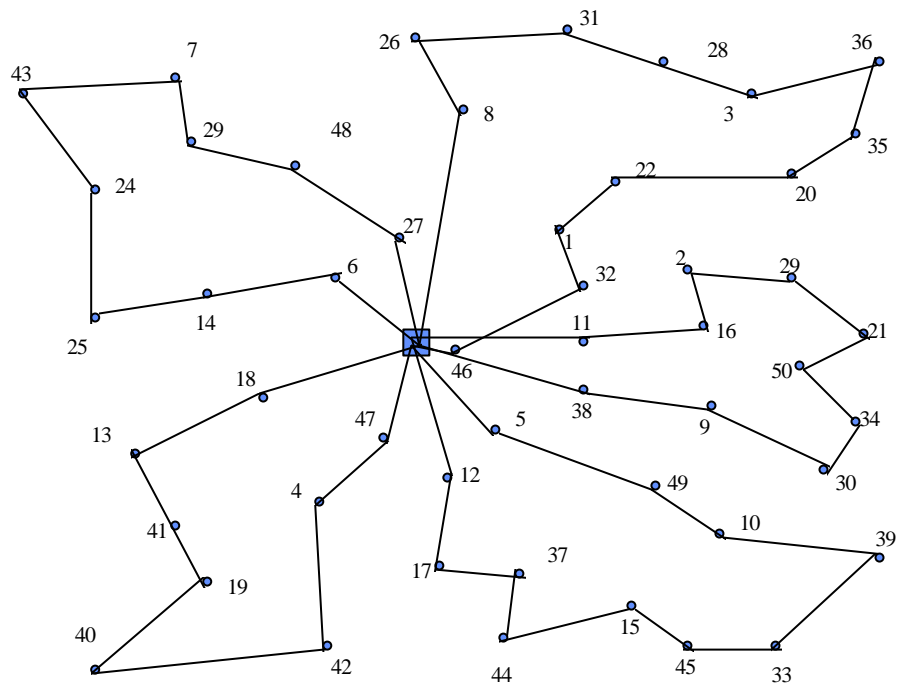
Na tabela da figura IX.9, apresentamos alguns resultados computacionais avaliando os limites inferiores obtidos (nó raiz), o número de nós e o tempo total de processamento gasto na busca em árvore. As instâncias avaliadas por Miller[95], foram implementadas em uma máquina *Sun Sparc 2*. Novamente, para os problemas apresentadas abaixo, consideramos distâncias “normalizadas” entre os clientes e o depósito.

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
Prob.	Lim. Inf. (raíz) (SCM)	Núm. de Nós (B&B)	tempo (cpu)	Lim. Inf. (raíz) (Miller)	Núm. de Nós (B&B)	tempo (cpu)	Sol. ótima
eil7.vrp	114	1	0,05 seg	114	1	0,22 seg	114
eil13.vrp	277	1	1,02 seg	268	60	14,83 seg	277
eil22.vrp	375	1	6,85 seg	374	46	33,94 seg	375
eil23.vrp	569	1	59,54 seg	569	1	4,07 seg	569
eil33.vrp	830	74	2,03 hs	826	379	9,16 min	835
eil51.vrp	512	1865	120,49 hs	505	7988	4,15 hs	521
c100a.dat	818	333	113,12 hs	-----	-----	-----	820
f44.dat	718	21	2,33 hs	-----	-----	-----	724

**Figura IX.9: Avaliação da Busca em Árvore**

Analisando-se o conjunto de problemas testes apresentados acima, observamos a melhor qualidade dos limites inferiores obtidos por nossa implementação. Isto provavelmente, contribuiu para a geração de um menor número de nós na árvore de busca na determinação de uma solução ótima para o problema original. A diferença no tempo de processamento entretanto, considerando-se, obviamente, a diferença entre as máquinas utilizadas, nos leva a concluir que o esforço computacional exigido por nossa abordagem foi maior que aquele despendido por Miller[95].

Ilustramos na figura IX.10, uma solução ótima para o problema *eil51.vrp* (biblioteca TSPLIB) com custo 521.



**Figura IX.10: Solução exata p/ o problema *eil51.dat***

Finalmente, na tabela da figura IX.11, apresentamos resultados computacionais para algumas instâncias geradas por Christofides et al.[95]. Novamente, neste caso, fixamos um total de 2000 iterações para o método subgradiente. A heurística C&W lagrangeano é executada nas primeiras 1000 iterações do subgradiente. O total de clientes em cada instância são representados na coluna (1):

(1)	(2)	(3)	(4)	(5)	(6)
Prob.	Num. de Veículos	Lim. Inf. (raíz)	Núm. de Nós (B&B)	tempo cpu	Sol. ótima
c21.dat	4	374.3	1	8,30 seg	374.3
c21a.dat	6	478.6	424	84,65 min	494.7
c15a.dat	5	322.9	137	9,72 min	334.1
c15b.dat	5	267.4	134	10,93 min	277.9
c20a.dat	6	426.3	16	1,75 min	429.9
c20b.dat	4	346.4	100	14,98 min	357.6

**Figura IX.11: Avaliação da busca em árvore.**

Neste caso, a distância entre os vértices é calculada, multiplicando-se por 10 a distância euclidiana, eliminando-se em seguida, sua parte fracionária. O número inteiro resultante é então dividido por 10.

Como ilustrado na tabela seguinte, um menor tempo de processamento pode ser conseguido no procedimento *branch-and-bound* se reduzimos o número total de iterações no subgradiente, de 2000, para aproximadamente, 900 iterações. A heurística lagrangeana (C&W lagrangeano) pode também ser executada, apenas nas primeiras iterações do subgradiente (em nosso caso 150 iterações). Obviamente outros parâmetros podem ser utilizados buscando-se ainda uma maior redução no tempo total de processamento.

(1)	(2)	(3)	(4)	(5)	(6)
Prob.	Num. de Veículos	Lim. Inf. (raíz)	Núm. de Nós (B&B)	tempo cpu	Sol. ótima
c21.dat	4	374.3	1	5,33 seg	374.3
c21a.dat	6	476.4	926	51,07 min	494.7
c15a.dat	5	321.0	127	2,65 min	334.1
c15b.dat	5	266.5	125	2,62 min	277.9
c20a.dat	6	424.7	27	1,38 min	429.9
c20b.dat	4	345.3	113	4,27 min	357.6

**Figura IX.12: Avaliação da busca em árvore.**

Observe que tivemos um ganho significativo no tempo de processamento embora o número de folhas de nossa árvore tenha aumentado em algumas das instâncias analisadas. A perda de qualidade nos limites inferiores (nó raíz) é compensada pela economia no tempo total de processamento na busca em árvore.