4
Heurísticas Propostas

Este capítulo descreve as abordagens propostas, apresentando os experimentos computacionais realizados e expondo os resultados numéricos obtidos.

4.1
Metodologia

O ambiente computacional utilizado na implementação e nos testes realizados foi:

i) *Hardware*: Intel Core2Duo CPU @ 2.33Ghz com 2048MB DDR2 de memória RAM;

ii) Sistema Operacional: Windows XP Service Pack 3;

iii) **ILOG CPLEX Academic Research Edition 12.2** [30];

iv) Linguagem de Programação: C++; e

v) Compilador: Microsoft Visual C++ Express 2008.

Já as instâncias usadas nos testes são aquelas pertencentes à comunidade de algoritmos exatos e consagradas na literatura. Mais especificamente, correspondem aos conjuntos 'A', 'B', 'E', 'F', 'M' e 'P' propostas originalmente em [3], [10], [11] e [20]. Essas se encontram disponíveis em [44]. Como o número de instâncias totais é grande, adotou-se para exposição neste trabalho o conjunto de testes exposto na Tabela 4.1, contendo instâncias de tamanho 13 a 101:

Tabela 4.1: Conjunto de testes usado nos experimentos.

Nome	No. Vértices	No. Veículos	Ótimo
E-n13-k4	13	4	247
P-n16-k8	16	8	450
P-n20-k2	20	2	216
A-n32-k5	32	5	784
A-n34-k5	34	5	778
B-n50-k7	50	7	741
E-n76-k10	76	10	830
E-n101-k14	101	14	1067

Todos os cortes definidos e utilizados nos experimentos estão sobre a formulação de arestas (ver subseção 2.3.1) e são inseridos na estrutura do BCP sobre a formulação *Dantzig-Wolfe Master*, conforme descrito na subseção 2.3.3. Ressalta-se também que o BCP é o responsável pela interação com a ferramenta **CPLEX**, utilizando-a para resolver problemas lineares através dos métodos primal simplex e dual simplex.

Os parâmetros utilizados no BCP foram usados nos seus valores padrão, incluindo a eliminação de s–ciclos, com $s = 3$. Adotou-se, ainda, como tempo limite para a execução do BCP, para uma dada chamada em uma iteração, o correspondente a $5000s$.

Finalmente, as soluções iniciais usadas nas buscas foram obtidas seguindo o paradigma *Cluster-First Route-Second*, em que primeiramente foram alocados os vértices em cada rota e, após isso, determinou-se a posição desses na permutação. Simples heurísticas construtivas foram implementadas com o propósito de obter uma solução viável: *First Fit*, *Best Fit*, *Worst Fit* e uma heurística gulosa, ordenando os vértices pelo critério de menor demanda. Para realizar a otimização de cada rota individualmente, foi utilizada uma heurística 2–opt delineada para o TSP.

4.2 Busca Local

Uma busca local baseada na vizinhança de bola sobre a estrutura do BCP foi implementada, aplicando-se o corte de *local branching* (seção 3.1). Dada uma solução viável $\bar{x}$ para o problema, obtém-se uma vizinhança t–OPT através do corte abaixo:

$$\Delta(x, \bar{x}) = \sum_{e \mid \bar{x}_e = 1} (1 - x_e) + \sum_{e \mid \bar{x}_e = 0} x_e \leq 2t \qquad (4\text{-}1)$$

A restrição (4-1) usa $O(n^2)$ variáveis, já que cada variável correspondente a uma aresta do grafo deve estar presente. Entretanto, aproveitando-se do fato de que, em qualquer solução viável, o número de arestas é constante e corresponde a $n + K - 1$, pode-se reescrevê-la da seguinte forma reduzida:

$$\sum_{e \mid \overline{x}_e = 1} (1 - x_e) \leq t \Rightarrow$$

$$\sum_{e \mid \overline{x}_e = 1} x_e \geq (n + K - 1) - t \tag{4-2}$$

envolvendo apenas $O(n)$ variáveis x_e.

Dessa forma, o algoritmo 1 descreve de forma geral a estrutura da busca local implementada. Nesse, ressalta-se que comentários são adicionados entre chaves:

Algorithm 1 Busca Local

Entrada: Solução inicial x_{ini}, Tamanho da vizinhança t
Saída: Melhor solução obtida x_{cur}
{Realiza uma busca local usando a vizinhança t–OPT}
$x_{cur} \leftarrow x_{ini}$ {Solução corrente}
$melhoria \leftarrow verdadeiro$
while $melhoria$ **do**
  $melhoria \leftarrow falso$
  adicionar restrição $\Delta(x, x_{curr}) \leq t$
  $bestUB \leftarrow custo(x_{curr})$ {Define o *upper bound* inicial como sendo o custo de x_{curr}}
  $x \leftarrow ResolverMIP$ {Aplica o BCP}
  if $custo(x) < custo(x_{curr})$ **then**
    $x_{curr} \leftarrow x$
    $melhoria \leftarrow verdadeiro$
  else
    $melhoria \leftarrow falso$
  end if
end while
return x_{curr}

O algoritmo 1 possui como entradas a solução viável inicial x_{ini} e o tamanho da vizinhança t–OPT. Já o retorno, por sua vez, corresponde à melhor solução encontrada na busca.

Como pode ser observado, o *upper bound* ($bestUB$) da estrutura do BCP é definido como o valor da solução corrente, objetivando acelerar a poda na árvore de *Branch-and-Bound*. Na sequência, é feita uma chamada ao resolvedor MIP (BCP), que realiza a otimização sobre a vizinhança.

Neste momento, uma decisão importante deve ser destacada: a estratégia da busca local, isto é, qual o melhor vizinho a ser retornado pelo BCP. Isso pode implicar em analisar ou não a vizinhança toda.

De fato, a estratégia que obteve o melhor desempenho em termos de tempo de processamento foi a de primeira melhoria, isto é, adotando-se a primeira solução viável encontrada na árvore como a solução retornada pelo resolvedor. Na medida em que o *upper bound* é definido como o valor da solução corrente, isso garante que a solução retornada será melhor do que a atual, ou nenhuma, caso exceda o tempo limite ou a árvore seja extinguida sem encontrar alguma solução.

A escolha por essa estratégia também teve como causa um problema encontrado, que é a dificuldade de se obterem soluções viáveis na vizinhança, a ser discutido mais adiante.

As Tabelas 4.2, 4.3, 4.4, 4.5 e 4.6 exibem o desempenho da busca local sobre o conjunto de instâncias selecionadas, para alguns tamanhos de vizinhança. Nessa, valores de solução destacados em negrito correspondem àqueles que atingiram o ótimo.

Tabela 4.2: Resultados para busca local com $t = 2$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	414	376	34,31	0,41	3
P-n16-k8	450	495	495	9,09	0,18	1
P-n20-k2	216	485	239	9,62	30,70	12
A-n32-k5	784	1898	1773	55,78	4,81	3
A-n34-k5	778	2095	1158	32,82	101,83	14
B-n50-k7	741	2892	2731	72,87	112,67	4
E-n76-k10	830	2375	1259	34,07	5936,78	40
E-n101-k14	1067	2664	2630	59,43	847,13	3

Tabela 4.3: Resultados para busca local com $t = 4$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	414	251	1,59	0,47	5
P-n16-k8	450	495	482	6,64	0,36	2
P-n20-k2	216	485	222	2,70	31,33	8
A-n32-k5	784	1898	959	18,25	128,69	16
A-n34-k5	778	2095	790	1,52	105,05	18
B-n50-k7	741	2892	746	0,67	1039,50	31
E-n76-k10	830	2375	857	3,15	6134,77	51
E-n101-k14	1067	2664	1906	44,02	74207,20	10

Tabela 4.4: Resultados para busca local com $t = 8$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	414	**247**	0,00	0,44	4
P-n16-k8	450	495	473	4,86	0,22	2
P-n20-k2	216	485	**216**	0,00	16,63	5
A-n32-k5	784	1898	**784**	0,00	42,63	9
A-n34-k5	778	2095	**778**	0,00	56,77	10
B-n50-k7	741	2892	**741**	0,00	291,27	14
E-n76-k10	830	2375	837	0,84	2088,48	24
E-n101-k14	1067	2664	1083	1,48	12203,40	28

Tabela 4.5: Resultados para busca local com $t = 10$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	414	**247**	0,00	0,19	3
P-n16-k8	450	495	**450**	0,00	0,38	3
P-n20-k2	216	485	**216**	0,00	9,73	5
A-n32-k5	784	1898	**784**	0,00	26,03	6
A-n34-k5	778	2095	786	1,02	58,22	8
B-n50-k7	741	2892	**741**	0,00	143,77	10
E-n76-k10	830	2375	839	1,07	2328,30	19
E-n101-k14	1067	2664	1079	1,11	10586,80	27

Tabela 4.6: Resultados para busca local com alguns valores de t.

Instância	**Ótimo**	**t**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	7	414	**247**	0,00	0,13	4
P-n16-k8	450	7	495	453	0,66	0,53	4
P-n20-k2	216	15	485	**216**	0,00	9,58	3
A-n32-k5	784	20	1898	**784**	0,00	7,66	3
A-n34-k5	778	20	2095	**778**	0,00	29,63	5
B-n50-k7	741	20	2892	**741**	0,00	92,42	5
E-n76-k10	830	20	2375	840	1,19	1479,02	9
E-n101-k14	1067	20	2664	1079	1,11	8439,52	16

Para vizinhanças pequenas, os valores obtidos são, conforme esperado, ruins, visto que apenas uma pequena parcela do espaço de busca é percorrida, atigindo-se ótimos locais de baixa qualidade. Além disso, um maior número de iterações é realizado. Para tamanhos de vizinhança sucessivamente maiores, um percentual maior do espaço é analisado, gerando melhores resultados e em menor número de iterações.

A realização de buscas locais com grandes vizinhanças (10, 15 e 20) em razoáveis tempos de execução mostram a vantagem dessa abordagem, ao se utilizar o poder do resolvedor MIP. Esses tamanhos de vizinhança vão além do que buscas locais simples baseadas em t–OPT tradicionais podem fazer ou, até mesmo, abordagens como a *Very Large Neighborhood Search* apresentada por Ahuja *et al.* [2]. Ademais, a tabela 4.7 corrobora esse fato, exibindo os resultados do uso de uma vizinhança de tamanho 30 para instâncias com número de vértices a partir de 50:

Tabela 4.7: Resultados para busca local com $t = 30$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
B-n50-k7	741	2892	**741**	0,00	51,19	4
A-n60-k9	1354	3741	**1354**	0,00	4997,30	8
A-n64-k9	1401	3527	**1401**	0,00	2718,17	9
E-n76-k14	1021	2456	**1021**	0,00	2980,83	7
E-n76-k10	830	2375	**830**	0,00	3687,50	9
B-n78-k10	1221	4369	**1221**	0,00	7678,59	7
A-n80-k10	1763	5441	**1763**	0,00	1823,52	7
E-n101-k8	815	2432	821	0,73	10684,30	7
E-n101-k14	1067	2664	**1067**	0,00	7131,72	10
M-n101-k10	820	1384	**820**	0,00	127,81	5

Percebeu-se um comportamento do BCP após a aplicação do corte de *local branching*. Para tamanhos de vizinhança menores, o tempo de processamento tende a ser grande, ao passo que, com valores de t cada vez mais altos, esse tempo de computação diminui. Tal fato representou o contrário do que intuitivamente era esperado, ou seja, buscas locais com vizinhanças menores deveriam ser mais rápidas do que com vizinhanças maiores, já que um menor percentual do espaço de soluções é analisado (o que ocorre em buscas locais tradicionais, por exemplo).

Isso se deve à dificuldade que o BCP encontrou para reconstruir rotas na geração de colunas após a inserção da restrição. Assim, com valores de t mais restritivos, há maior dificuldade de se obterem rotas nessa etapa do algoritmo, tomando-se mais tempo. Com valores de t maiores, permitindo um maior número de trocas de arestas, tem-se mais liberdade na reconstrução de rotas e a geração de colunas executa mais rapidamente.

Esse fato está diretamente relacionado à dificuldade que se tem em obter soluções viáveis na vizinhança, sendo a razão por ter sido escolhida a estratégia de primeira melhoria como a solução retornada pelo BCP, já que a análise de toda a vizinhança toma uma grande quantidade de tempo. Resultou ainda que, para instâncias contendo 135 ou mais vértices (consideradas as mais difíceis e com grande parte ainda em aberto), não foi possível encontrar soluções viáveis em tempo exequível na vizinhança.

Entretanto, a experiência adquirida nos experimentos com buscas locais usando a restrição de *local branching* forneceu um direcionamento de que a vizinhança elipsoidal poderia ajudar a reconstruir rotas mais facilmente.

4.3 Busca Elipsoidal

Buscas elipsoidais, baseadas na vizinhança elipsoidal, foram implementadas para duas e três soluções de referência. A forma geral está descrita no algoritmo 2.

Algorithm 2 Busca Elipsoidal

Entrada: Conjunto de soluções iniciais S, Tamanho da vizinhança t, Número máximo de falhas $max\_failures$
Saída: Melhor solução obtida x_{best}
{Realiza uma busca elipsoidal com tamanho de vizinhança t}
$x_{best} \leftarrow x \mid \min\ custo(x)\ \forall x \in S$ {Melhor solução em S}
$num\_failures \leftarrow 0$
while $num\_failures < max\_failures$ **do**
  adicionar restrição elipsoidal com base nas soluções em S
  $bestUB \leftarrow media(x \in S)$
  $x \leftarrow ResolverMIP$ {Aplica o BCP}
  atualizar conjunto S
  if $custo(x) < custo(x_{best})$ **then**
    $x_{best} \leftarrow x$
    $num\_failures \leftarrow 0$
  else
    $num\_failures \leftarrow num\_failures + 1$
  end if
end while
return x_{best}

No algoritmo 2, tem-se que um conjunto de soluções iniciais S é dado como entrada, além de t e do número máximo de falhas $max\_failures$. O conjunto S corresponde às soluções-base para a busca, a serem utilizadas na aplicação do corte. Para os resultados expostos no trabalho, correspondendo àqueles que obtiveram o melhor desempenho, o número máximo de falhas foi definido como 1.

O valor do *upper bound* (variável $bestUB$) corresponde à média dos valores das soluções em S. Isso tem como objetivo introduzir um componente de diversificação na estrutura da busca, para que soluções piores do que a melhor obtida, mas que podem levar a uma região promissora do espaço, não sejam descartadas.

Já a atualização do conjunto S, após a obtenção de x na chamada ao BCP, pode ser feita de algumas formas, entre as quais: aleatória, remoção da pior

solução ou remoção da melhor. A estratégia que obteve melhor desempenho nos testes, foi a de retirar a pior solução e introduzir a nova solução x em S.

Finalmente, os cortes aplicados para os dois tipos de busca serão detalhadamente descritos nas próximas seções.

4.3.1 Busca Elipsoidal de 2 Soluções

A restrição que define a vizinhança elipsoidal para um dado t assume a forma abaixo quando aplicada ao problema, dadas duas soluções-base $\overline{x^1}$ e $\overline{x^2}$:

$$\sum_{e \mid \overline{x_e^1}=1,\, \overline{x_e^2}=0} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=1} x_e + \sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=1} 2x_e \geq |\overline{x_e^1} \cap \overline{x_e^2}| + (n+K-1) - t$$

em que $|\overline{x^1} \cap \overline{x^2}|$ fornece a cardinalidade da interseção entre $\overline{x^1}$ e $\overline{x^2}$, ou seja, o número de arestas que ambas as soluções compartilham. Analogamente ao explicado no corte de *local branching*, a expressão $(n+K-1)$ corresponde ao total de arestas presentes em uma solução do problema.

Na restrição acima, nota-se claramente que arestas presentes em ambas as soluções são priorizadas, recebedo uma ponderação igual a 2, enquanto arestas presentes em apenas uma solução possuem coeficiente 1.

Contudo, sabendo-se que a seguinte igualdade é válida

$$\sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=1} x_e = |\overline{x^1} \cap \overline{x^2}|$$

pode-se reescrever o corte da seguinte forma reduzida:

$$\sum_{e \mid \overline{x_e^1}=1,\, \overline{x_e^2}=0} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=1} x_e + \sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=1} x_e \geq (n+K-1) - t \quad (4\text{-}3)$$

A expressão (4-3) corresponde à utilizada na implementação do algoritmo 2.

As Tabelas 4.8, 4.9, 4.10 e 4.11 expõem o resultado da execução da busca elipsoidal usando a restrição (4-3) para o conjunto de instâncias. Nessas, a coluna **Inicial** exibe o valor da média das soluções no conjunto S, e soluções obtidas que chegaram ao ótimo estão destacadas em negrito:

Tabela 4.8: Resultados para busca elipsoidal de 2 com $t = 2$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	428	276	10,51	0,53	4
P-n16-k8	450	491	486	7,41	0,03	1
P-n20-k2	216	512	251	13,94	43,16	11
A-n32-k5	784	1995	929	15,61	148,42	19
A-n34-k5	778	2066	863	9,85	148,30	24
B-n50-k7	741	2817	2283	67,54	178,52	2
E-n76-k10	830	2412	1477	43,81	2762,59	21
E-n101-k14	1067	3172	1970	45,84	3787,91	11

Tabela 4.9: Resultados para busca elipsoidal de 2 com $t = 4$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	428	**247**	0,00	0,80	4
P-n16-k8	450	491	461	2,39	0,13	2
P-n20-k2	216	512	233	7,30	15,56	5
A-n32-k5	784	1995	838	6,44	91,20	12
A-n34-k5	778	2066	817	4,77	83,36	11
B-n50-k7	741	2817	804	7,84	294,30	13
E-n76-k10	830	2412	1125	26,22	1609,50	16
E-n101-k14	1067	3172	1698	37,16	6002,09	11

Tabela 4.10: Resultados para busca elipsoidal de 2 com $t = 10$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	428	**247**	0,00	0,06	2
P-n16-k8	450	491	**450**	0,00	0,55	3
P-n20-k2	216	512	**216**	0,00	7,92	4
A-n32-k5	784	1995	**784**	0,00	11,25	4
A-n34-k5	778	2066	784	0,77	30,36	5
B-n50-k7	741	2817	**741**	0,00	151,13	10
E-n76-k10	830	2412	894	7,16	978,39	10
E-n101-k14	1067	3172	1146	6,89	3864,83	15

Tabela 4.11: Resultados para busca busca elipsoidal de 2 com alguns valores de t.

Instância	**Ótimo**	**t**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	7	428	**247**	0,00	0,08	3
P-n16-k8	450	12	491	**450**	0,00	0,59	3
P-n20-k2	216	12	512	**216**	0,00	11,41	4
A-n32-k5	784	30	1995	**784**	0,00	1,67	2
A-n34-k5	778	30	2066	**778**	0,00	12,30	3
B-n50-k7	741	30	2817	**741**	0,00	18,34	3
E-n76-k10	830	30	2412	837	0,84	864,02	5
E-n101-k14	1067	30	3172	1082	1,39	1753,61	6

A vizinhança elipsoidal definida dessa forma não surtiu exatamente o efeito que se esperava, apesar de ter melhorado os tempos de execução para algumas instâncias em relação à busca local, e ter se saído melhor para $t = 2$. No entanto, constantou-se que, conforme esperado, menos tempo era gasto na construção de rotas.

Assim, com a experiência adquirida nos experimentos para busca local, buscou-se adicionar alguns componentes na restrição (4-3) que auxiliassem ainda mais na reconstrução de rotas, mas que ainda assim não se aproximassem da resolução completa do problema.

A primeira estratégia adotada foi adicionar na restrição as arestas que ligam o vértice v_0, referente ao depósito, a cada um dos outros vértices do grafo, conforme ilustrado na Figura 4.1. Nessa, tem-se uma possível solução-base e as arestas extras adicionadas, que seguem destacadas em vermelho e tracejadas:

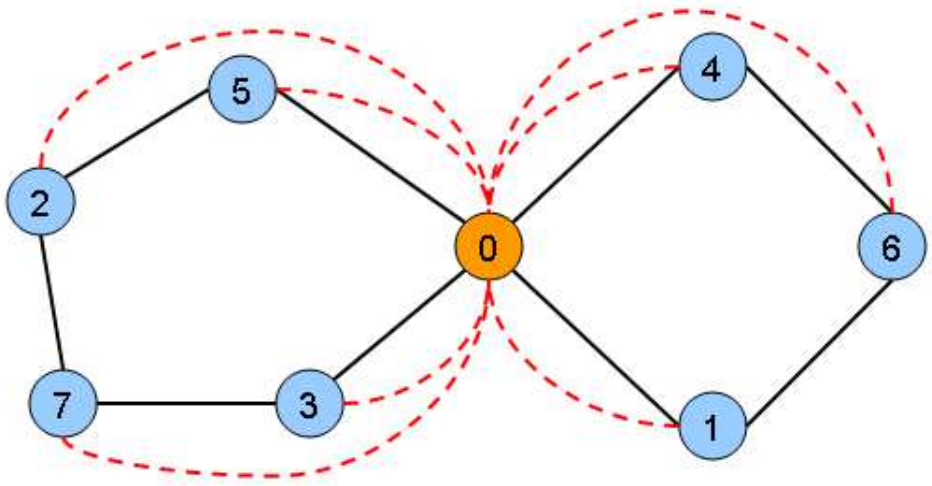

Figura 4.1: Arestas do depósito.

A restrição, então, assume a seguinte configuração:

$$\sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=0} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=1} x_e + \sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=1} x_e + \sum_{v \in V_+} x_{(v_0,v)} \geq (n+K-1)-t \tag{4-4}$$

em que as arestas adicionais acrescentadas aparecem do lado esquerdo.

As Tabelas 4.12, 4.13 e 4.14 resumem alguns resultados alcançados pela busca elipsoidal usando-se esse novo corte. Nessas, as colunas possuem o mesmo significado como anteriormente descrito.

Tabela 4.12: Resultados para busca elipsoidal de 2 usando arestas do depósito com $t = 2$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	428	**247**	0,00	0,09	2
P-n16-k8	450	491	**450**	0,00	0,67	3
P-n20-k2	216	512	**216**	0,00	20,25	6
A-n32-k5	784	1995	**784**	0,00	18,34	5
A-n34-k5	778	2066	869	10,47	26,47	3
B-n50-k7	741	2817	**741**	0,00	75,66	6
E-n76-k10	830	2412	841	1,31	723,16	6
E-n101-k14	1067	3172	1084	1,57	1744,83	6

Tabela 4.13: Resultados para busca elipsoidal de 2 usando arestas do depósito com $t = 10$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	428	**247**	0,00	0,06	2
P-n16-k8	450	491	**450**	0,00	0,55	3
P-n20-k2	216	512	218	0,92	2,69	2
A-n32-k5	784	1995	**784**	0,00	2,78	3
A-n34-k5	778	2066	**778**	0,00	21,67	4
B-n50-k7	741	2817	**741**	0,00	42,00	5
E-n76-k10	830	2412	856	3,04	782,00	4
E-n101-k14	1067	3172	**1067**	0,00	1416,95	6

Tabela 4.14: Resultados para busca elipsoidal de 2 usando arestas do depósito com alguns valores de t.

Instância	**Ótimo**	**t**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	7	428	**247**	0,00	0,06	2
P-n16-k8	450	12	491	**450**	0,00	0,56	3
P-n20-k2	216	12	512	**216**	0,00	10,41	4
A-n32-k5	784	30	1995	**784**	0,00	1,69	2
A-n34-k5	778	30	2066	**778**	0,00	14,08	3
B-n50-k7	741	30	2817	**741**	0,00	12,66	3
E-n76-k10	830	30	2412	831	0,12	303,70	3
E-n101-k14	1067	30	3172	1070	0,28	1207,97	4

Como se pode ver, os resultados com essa nova restrição foram muito bons, diminuindo consideravelmente tanto o tempo quanto o valor das soluções, em comparação com a busca local e a elipsoidal de 2 pura. Aproveitando-se, portanto, do fato de que a adição de componentes na restrição elipsoidal foi bem-sucedida, testou-se também o uso do conceito de rota gigante (*giant tour*).

Essa corresponde a uma única rota, gigante, que passa por todos os vértices do grafo através da concatenação das rotas de uma solução-base. O único vértice que se repete é o depósito. A título de exemplo, para a solução da Figura 4.2, uma possível *giant tour* corresponde a $0{-}5{-}2{-}7{-}3{-}0{-}4{-}6{-}1{-}0$.

Consideram-se, assim, as "cordas", que são arestas com extremidades nos vértices distando 2 na sequência. Portanto, essas seriam: $\{(0,2),(5,7),(2,3),(7,0),(3,4),(0,6),(4,1),(1,5)\}$, as quais estão destacadas em vermelho e tracejadas na Figura 4.2:

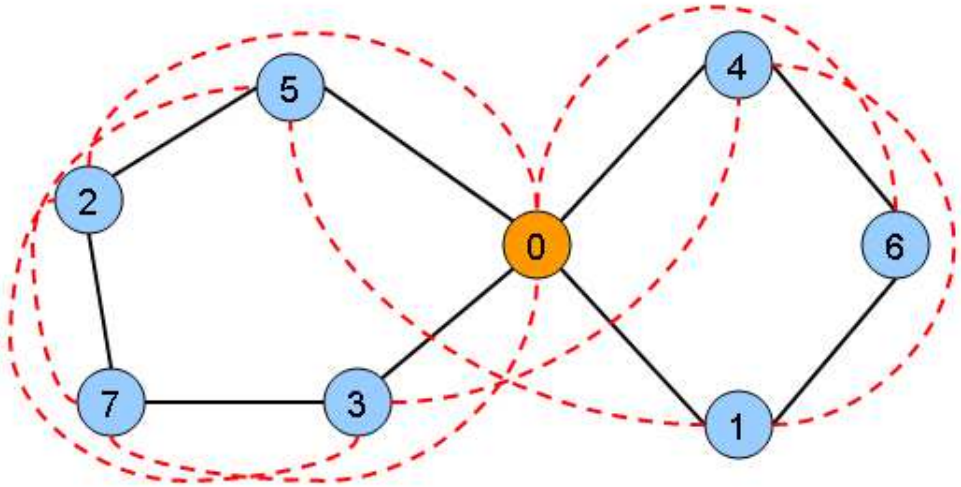

Figura 4.2: Arestas obtidas a partir da *giant tour*.

Feita a adição dessas arestas, a estrutura do corte obtido é da seguinte forma:

$$\sum_{e \mid \overline{x_e^1}=1,\, \overline{x_e^2}=0} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=1} x_e + \sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=1} x_e + \sum_{e \,\in\, G} x_e \geq (n + K - 1) - t$$

em que G corresponde ao conjunto das arestas obtidas a partir de uma *giant tour* de ambas as soluções $\overline{x^1}$ e $\overline{x^2}$.

As Tabelas 4.15, 4.16 e 4.17 exibem alguns resultados fornecidos pela busca elipsoidal de 2 com base no corte supracitado.

Tabela 4.15: Resultados para busca elipsoidal de 2 usando arestas de uma *giant tour* com $t = 2$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	428	**247**	0,00	0,09	2
P-n16-k8	450	491	**450**	0,00	0,56	3
P-n20-k2	216	512	**216**	0,00	6,72	4
A-n32-k5	784	1995	**784**	0,00	27,02	4
A-n34-k5	778	2066	**778**	0,00	20,19	4
B-n50-k7	741	2817	**741**	0,00	72,09	4
E-n76-k10	830	2412	846	1,89	1122,42	6
E-n101-k14	1067	3172	**1067**	0,00	1737,33	7

Tabela 4.16: Resultados para busca elipsoidal de 2 usando arestas da *giant tour* com $t = 10$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	428	**247**	0,00	0,08	2
P-n16-k8	450	491	**450**	0,00	0,59	3
P-n20-k2	216	512	218	0,92	3,34	2
A-n32-k5	784	1995	**784**	0,00	2,59	3
A-n34-k5	778	2066	**778**	0,00	10,64	3
B-n50-k7	741	2817	**741**	0,00	24,78	4
E-n76-k10	830	2412	834	0,48	883,77	5
E-n101-k14	1067	3172	1071	0,37	1432,80	5

Tabela 4.17: Resultados para busca elipsoidal de 2 usando arestas da *giant tour* com alguns valores de t.

Instância	**Ótimo**	**t**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	7	428	**247**	0,00	0,08	2
P-n16-k8	450	12	491	**450**	0,00	0,58	3
P-n20-k2	216	12	512	218	0,92	3,34	2
A-n32-k5	784	30	1995	**784**	0,00	1,69	2
A-n34-k5	778	30	2066	**778**	0,00	7,36	2
B-n50-k7	741	30	2817	**741**	0,00	29,91	3
E-n76-k10	830	30	2412	832	0,24	947,94	5
E-n101-k14	1067	30	3172	1069	0,19	1554,81	5

Os resultados obtidos pela adição das arestas oriundas da *giant tour* foram excelentes, atingindo o ótimo na maioria das instâncias e deixando o *gap* pequeno, até mesmo para $t = 2$. Evidencia-se, assim, que a estratégia de inserir componentes no corte, a fim de ajudar na obtenção de rotas, de fato, funciona.

Finalmente, para ilustrar o conceito da vizinhança elipsoidal, sejam as Figuras 4.3 e 4.4 a representação gráfica de duas soluções $\overline{x^1}$ e $\overline{x^2}$ para a instância A-n32-k5. Ao se aplicar a vizinhança elipsodal com $t = 0$, pode-se ter uma ideia da "junção" entre tais soluções, obtendo-se a solução exposta na Figura 4.5, que corresponde à ótima:

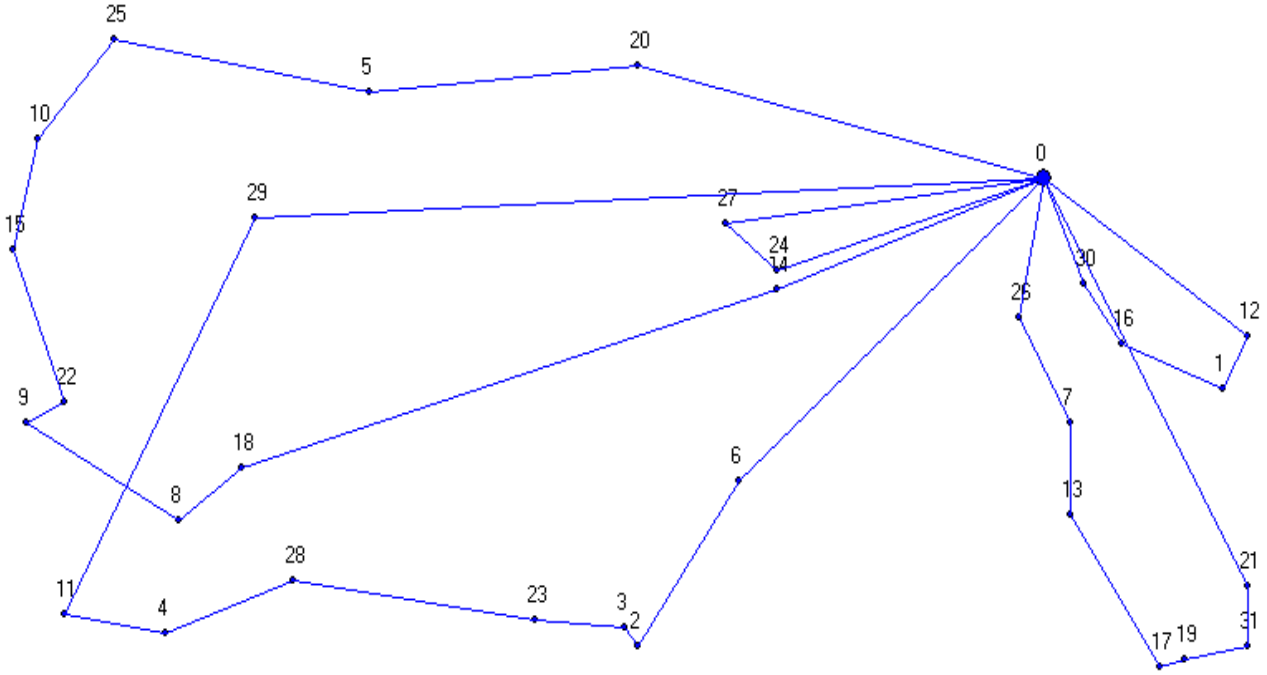

Figura 4.3: Representação gráfica de uma solução-base $\overline{x^1}$.

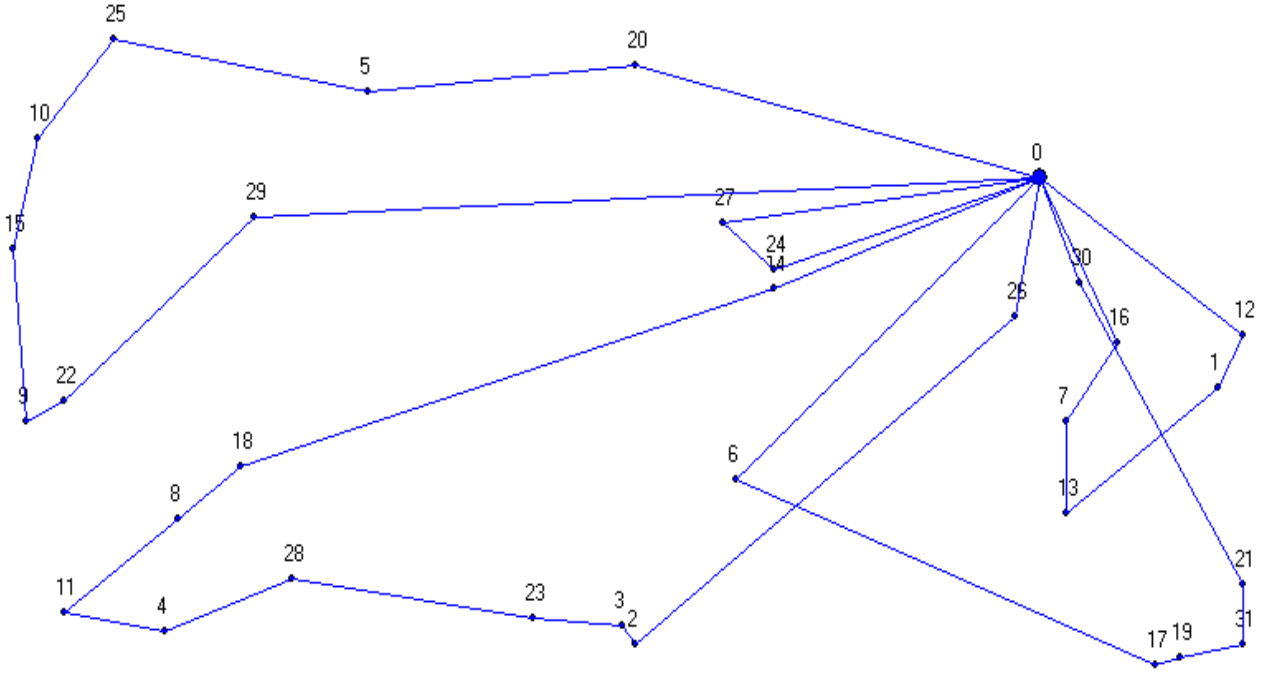

Figura 4.4: Representação gráfica de uma solução-base $\overline{x^2}$.

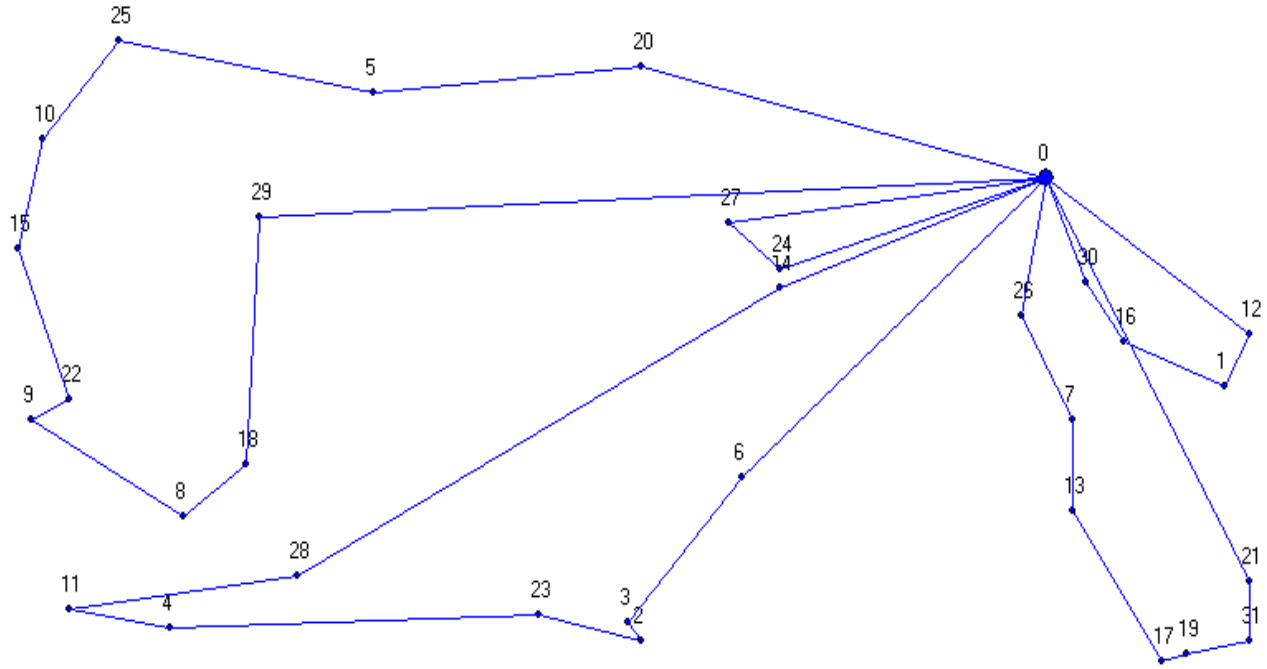

Figura 4.5: Solução obtida na aplicação da vizinhança elipsoidal a partir de $\overline{x^1}$ e $\overline{x^2}$.

As Figuras 4.6 e 4.7 apresentam o grafo de duas soluções $\overline{x^3}$ e $\overline{x^4}$ para a instância A-n39-k6. O resultado de uma iteração da busca elipsoidal com $t = 10$ segue exposto na Figura 4.8, cujo ótimo é mostrado na Figura 4.9:

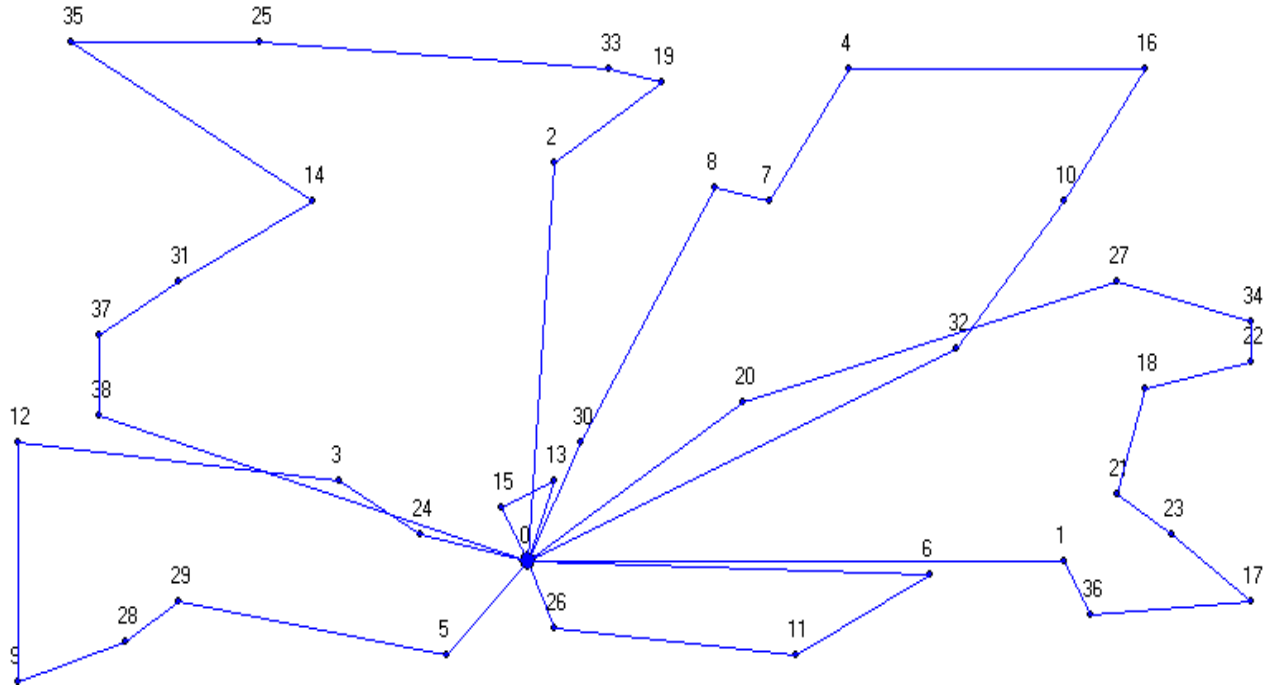

Figura 4.6: Representação gráfica de uma solução-base $\overline{x^3}$.

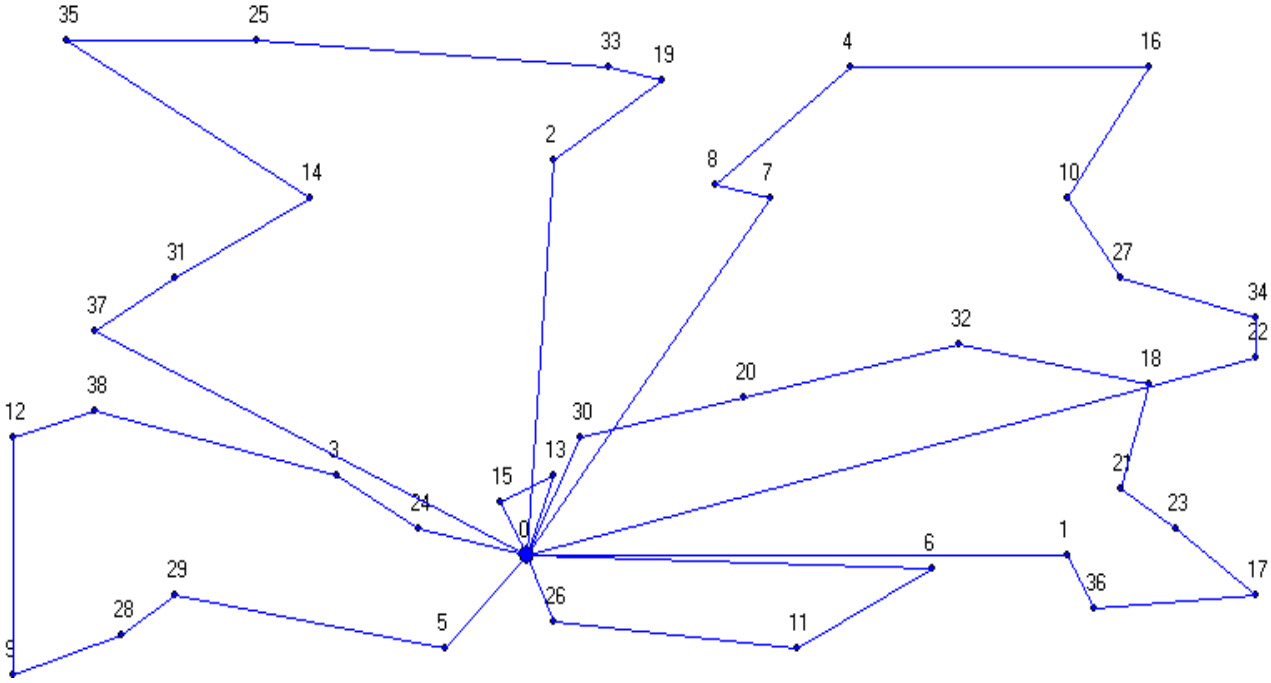

Figura 4.7: Representação gráfica de uma solução-base $\overline{x^4}$.

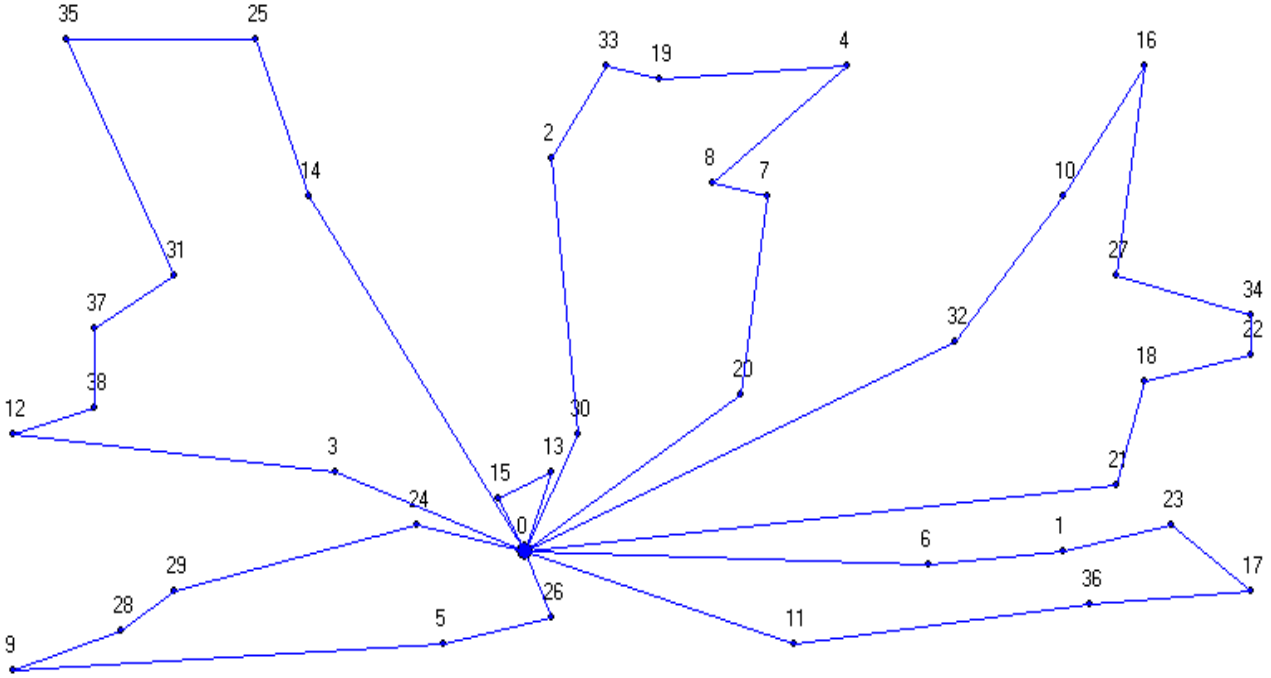

Figura 4.8: Solução obtida na aplicação da vizinhança elipsoidal a partir de $\overline{x^3}$ e $\overline{x^4}$.

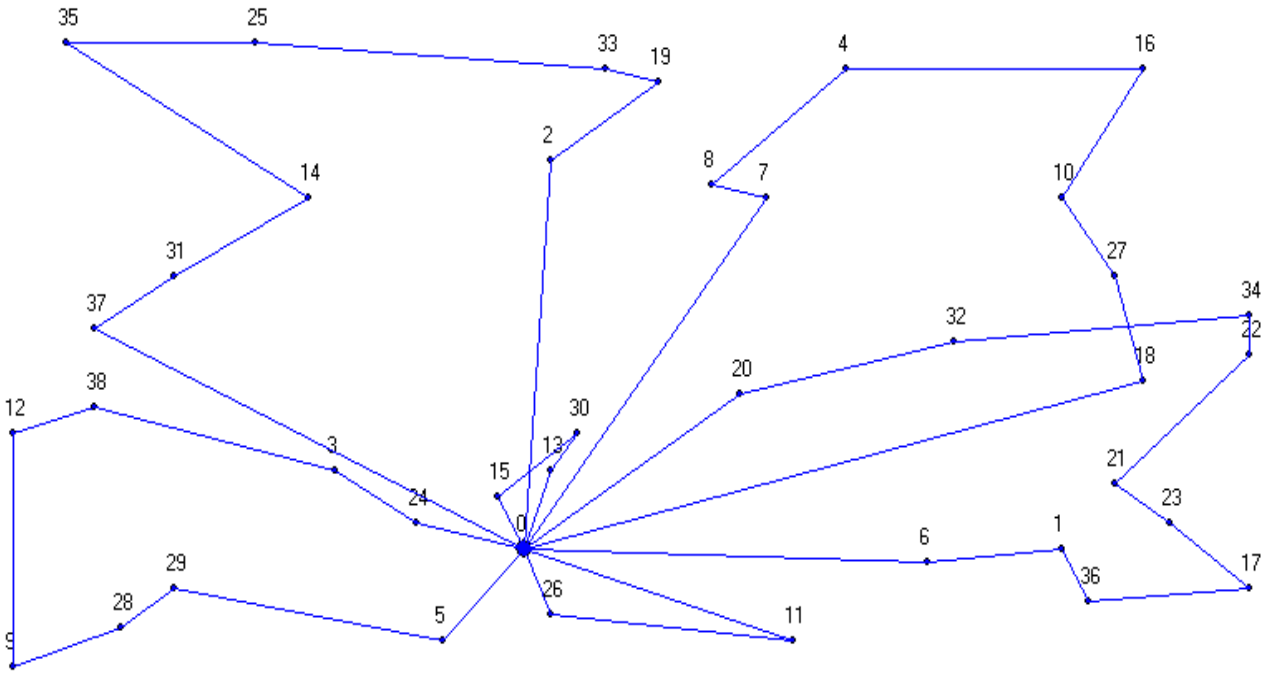

Figura 4.9: Solução ótima para a instância A-n39-k6.

4.3.2
Busca Elipsoidal de 3 Soluções

Foi feita uma extensão do conceito de vizinhança elipsoidal para abranger o caso de 3 soluções, com base no trabalho de Amaral [1]. A estrutura da busca é aquela descrita no algoritmo 2, em que o conjunto S passa a conter 3 soluções. Logo, para três soluções-base $\overline{x^1}$, $\overline{x^2}$ e $\overline{x^3}$, tem-se a seguinte definição para a restrição:

$$\sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=0, \overline{x_e^3}=0} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=1, \overline{x_e^3}=0} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=0, \overline{x_e^3}=1} x_e +$$

$$\sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=1, \overline{x_e^3}=0} 2x_e + \sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=0, \overline{x_e^3}=1} 2x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=1, \overline{x_e^3}=1} 2x_e +$$

$$\sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=1, \overline{x_e^3}=1} 3x_e \geq (n+K-1)+|\overline{x^1} \cap \overline{x^2}|+|\overline{x^1} \cap \overline{x^3}|+|\overline{x^2} \cap \overline{x^3}|+2|\overline{x^1} \cap \overline{x^2} \cap \overline{x^3}|-t$$

em que arestas presentes em duas soluções possuem coeficiente 2 e analogamente para arestas que apareçam em três soluções. Assim, a cardinalidade das respectivas interseções aparecem do lado direito, junto com a "folga" t e o número de arestas em uma solução $(n + K - 1)$.

Entretanto, aplicando-se o mesmo raciocínio usado para duas soluções, é possível reduzir a restrição à seguinte forma:

$$\sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=0, \overline{x_e^3}=0} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=1, \overline{x_e^3}=0} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=0, \overline{x_e^3}=1} x_e + \quad (4\text{-}5)$$

$$\sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=1, \overline{x_e^3}=0} x_e + \sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=0, \overline{x_e^3}=1} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=1, \overline{x_e^3}=1} x_e +$$

$$\sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=1, \overline{x_e^3}=1} x_e \geq (n + K - 1) - t$$

que corresponde à estrutura do corte utilizada na implementação. Assim sendo, as Tabelas 4.18, 4.19, 4.20 e 4.21 mostram o resultado da execução da busca para as instâncias do conjunto. Nessas, a coluna **Inicial** exibe o valor da média das soluções iniciais, e valores de solução que correspondem ao ótimo estão destacados em negrito:

Tabela 4.18: Resultados para busca elipsoidal de 3 com $t = 2$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	428	259	4,63	0,42	4
P-n16-k8	450	507	486	7,41	0,03	1
P-n20-k2	216	498	224	3,57	17,72	9
A-n32-k5	784	2001	1125	30,31	25,38	6
A-n34-k5	778	2056	1264	38,45	39,58	6
B-n50-k7	741	2768	1205	38,51	329,75	11
E-n76-k10	830	2385	1822	54,45	437,06	3
E-n101-k14	1067	2976	1566	31,86	4994	17

Tabela 4.19: Resultados para busca elipsoidal de 3 com $t = 4$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	428	**247**	0,00	0,55	5
P-n16-k8	450	507	**450**	0,00	0,20	4
P-n20-k2	216	498	258	16,28	8,22	3
A-n32-k5	784	2001	**784**	0,00	24,52	9
A-n34-k5	778	2056	979	20,53	26,83	5
B-n50-k7	741	2768	829	10,62	240,77	12
E-n76-k10	830	2385	985	15,74	2791,45	26
E-n101-k14	1067	2976	1381	22,74	4490,42	16

Tabela 4.20: Resultados para busca elipsoidal de 3 com $t = 10$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	428	**247**	0,00	0,11	2
P-n16-k8	450	507	**450**	0,00	0,56	3
P-n20-k2	216	498	218	0,92	10,67	5
A-n32-k5	784	2001	**784**	0,00	11,36	5
A-n34-k5	778	2056	793	1,89	23,31	4
B-n50-k7	741	2768	**741**	0,00	70,86	8
E-n76-k10	830	2385	849	2,24	889,89	11
E-n101-k14	1067	2976	1114	4,22	2772,55	15

Tabela 4.21: Resultados para busca elipsoidal de 3 com alguns valores de t.

Instância	**Ótimo**	**t**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	7	428	**247**	0,00	0,09	3
P-n16-k8	450	12	507	**450**	0,00	0,56	3
P-n20-k2	216	12	498	**216**	0,00	9,41	4
A-n32-k5	784	30	2001	**784**	0,00	1,73	2
A-n34-k5	778	30	2056	**778**	0,00	12,09	3
B-n50-k7	741	30	2768	**741**	0,00	18,89	3
E-n76-k10	830	30	2385	831	0,12	521,44	5
E-n101-k14	1067	30	2976	**1067**	0,00	1399,44	6

Pela análise das tabelas, percebe-se que os resultados obtidos pela busca elipsoidal de 3 soluções foram relativamente bons, principalmente para os valores de t a partir de 10. Para os casos de t igual a 2 e 4, o desempenho não foi tão significativo, mas ainda assim foi melhor do que o realizado pela elipsoidal de 2 pura. Evidentemente, isso era o que se esperava, já que neste caso se dispõe de mais uma solução-base, ampliando a ação da vizinhança sobre o espaço de busca.

Neste ponto, adota-se também a ideia de adicionar à restrição as arestas ligando o vértice v_0 aos demais. Dessa maneira, o corte obtido possui a seguinte estrutura:

$$\sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=0, \overline{x_e^3}=0} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=1, \overline{x_e^3}=0} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=0, \overline{x_e^3}=1} x_e + \quad (4\text{-}6)$$

$$\sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=1, \overline{x_e^3}=0} x_e + \sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=0, \overline{x_e^3}=1} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=1, \overline{x_e^3}=1} x_e +$$

$$\sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=1, \overline{x_e^3}=1} x_e + \sum_{v \in V_+} x_{(v_0,v)} \geq (n + K - 1) - t$$

e seus resultados seguem expostos nas Tabelas 4.22, 4.23 e 4.24:

Tabela 4.22: Resultados para busca elipsoidal de 3 usando arestas do depósito com $t = 2$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	428	**247**	0,00	0,08	2
P-n16-k8	450	507	**450**	0,00	0,58	3
P-n20-k2	216	498	226	4,42	14,00	4
A-n32-k5	784	2001	**784**	0,00	17,16	5
A-n34-k5	778	2056	**778**	0,00	45,00	7
B-n50-k7	741	2768	**741**	0,00	63,14	6
E-n76-k10	830	2385	1822	54,45	437,06	3
E-n101-k14	1067	2976	1078	1,02	1867,36	6

Tabela 4.23: Resultados para busca elipsoidal de 3 usando arestas do depósito com $t = 10$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	428	**247**	0,00	0,08	2
P-n16-k8	450	507	**450**	0,00	0,55	3
P-n20-k2	216	498	218	0,92	3,88	2
A-n32-k5	784	2001	**784**	0,00	2,44	3
A-n34-k5	778	2056	**778**	0,00	21,25	4
B-n50-k7	741	2768	**741**	0,00	59,38	5
E-n76-k10	830	2385	**830**	0,00	780,08	6
E-n101-k14	1067	2976	1071	0,37	1370,11	5

Tabela 4.24: Resultados para busca elipsoidal de 3 usando arestas do depósito com alguns valores de t.

Instância	**Ótimo**	**t**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	7	428	**247**	0,00	0,08	2
P-n16-k8	450	12	507	**450**	0,00	0,55	3
P-n20-k2	216	12	498	218	0,92	6,08	2
A-n32-k5	784	30	2001	**784**	0,00	1,67	2
A-n34-k5	778	30	2056	**778**	0,00	7,64	2
B-n50-k7	741	30	2768	**741**	0,00	15,34	3
E-n76-k10	830	30	2385	831	0,12	449,58	4
E-n101-k14	1067	30	2976	**1067**	0,00	1374,84	4

Tais experimentos apontam que a adição das arestas do depósito também foi bem-sucedida para este caso com 3 soluções, auxiliando no processo de reconstrução de rotas. Analogamente, a adição de arestas baseadas na *giant tour* também ajuda. Define-se, pois, a sua restrição relacionada:

$$\sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=0, \overline{x_e^3}=0} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=1, \overline{x_e^3}=0} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=0, \overline{x_e^3}=1} x_e + \quad (4\text{-}7)$$

$$\sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=1, \overline{x_e^3}=0} x_e + \sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=0, \overline{x_e^3}=1} x_e + \sum_{e \mid \overline{x_e^1}=0, \overline{x_e^2}=1, \overline{x_e^3}=1} x_e +$$

$$\sum_{e \mid \overline{x_e^1}=1, \overline{x_e^2}=1, \overline{x_e^3}=1} x_e + \sum_{e \in G} x_e \geq (n + K - 1) - t$$

em que G corresponde ao conjunto de arestas obtidas a partir da *giant tour* para $\overline{x^1}$, $\overline{x^2}$ e $\overline{x^3}$, conforme explicado na seção 4.3.1.

Os resultados da execução da busca com essa restrição sobre o conjunto de testes segue nas Tabelas 4.25, 4.26 e 4.27:

Tabela 4.25: Resultados para busca elipsoidal de 3 usando arestas da *giant tour* com $t = 2$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	428	**247**	0,00	0,08	2
P-n16-k8	450	507	**450**	0,00	0,56	3
P-n20-k2	216	498	**216**	0,00	9,44	4
A-n32-k5	784	2001	**784**	0,00	10,59	3
A-n34-k5	778	2056	**778**	0,00	13,03	3
B-n50-k7	741	2768	**741**	0,00	28,11	4
E-n76-k10	830	2385	835	0,60	489,39	4
E-n101-k14	1067	2976	1070	0,28	1349,56	4

Tabela 4.26: Resultados para busca elipsoidal de 3 usando arestas da *giant tour* com $t = 10$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	428	**247**	0,00	0,08	2
P-n16-k8	450	507	**450**	0,00	0,56	3
P-n20-k2	216	498	**216**	0,00	3,30	2
A-n32-k5	784	2001	**784**	0,00	1,80	2
A-n34-k5	778	2056	**778**	0,00	15,84	3
B-n50-k7	741	2768	**741**	0,00	32,00	3
E-n76-k10	830	2385	845	1,78	5321,69	3
E-n101-k14	1067	2976	1070	0,28	1984,42	6

Tabela 4.27: Resultados para busca elipsoidal de 3 usando arestas da *giant tour* com alguns valores de t.

Instância	**Ótimo**	**t**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
E-n13-k4	247	7	428	**247**	0,00	0,08	2
P-n16-k8	450	12	507	**450**	0,00	0,58	3
P-n20-k2	216	12	498	**216**	0,00	3,33	2
A-n32-k5	784	30	2001	**784**	0,00	1,67	2
A-n34-k5	778	30	2056	**778**	0,00	7,81	2
B-n50-k7	741	30	2768	**741**	0,00	16,25	3
E-n76-k10	830	30	2385	839	1,07	5311,80	3
E-n101-k14	1067	30	2976	**1067**	0,00	840,844	3

Analogamente ao uso de *giant tour* pela elipsoidal de 2 soluções, tem-se que excelentes resultados foram obtidos, alcançando o ótimo de quase todas as instâncias do conjunto, em um baixo tempo de computação. A Tabela 4.28 ratifica esse fato, exibindo o resultado da execução sobre diversas instâncias de grande porte usando $t = 30$:

Tabela 4.28: Resultados para busca elipsoidal de 3 usando arestas da *giant tour* com $t = 30$.

Instância	**Ótimo**	**Inicial**	**Obtido**	*Gap (%)*	**Tempo (s)**	**Iterações**
A-n60-k9	1354	3751	1360	0,44	243,36	5
A-n64-k9	1401	3469	1406	0,36	233,63	3
A-n69-k9	1159	4077	1171	1,02	254,25	3
E-n76-k7	682	2322	685	0,44	834,94	4
E-n76-k8	735	2315	736	0,14	387,89	3
E-n76-k14	1021	2450	1024	0,29	120,14	2
B-n78-k10	1221	4338	**1221**	0,00	292,78	3
A-n80-k10	1763	5154	1767	0,23	429,42	3
E-n101-k8	815	2829	824	1,09	3547,14	3
M-n101-k10	820	2033	**820**	0,00	52,33	2
M-n121-k7	1034	3409	**1034**	0,00	10996,90	4

A próxima seção apresenta alguns gráficos comparativos entre o desempenho da busca local e variantes da busca elipsoidal.

4.4 Gráficos de Desempenho

Nesta seção, faz-se uma análise do desempenho de algumas das abordagens implementadas e descritas na seção anterior. Para isso, são usados gráficos baseados nos valores das soluções encontradas e no tempo em que isso ocorreu. Assim, a Figura 4.10 estabelece uma comparação do desempenho para a instância A-n34-k5, sobre uma vizinhança de tamanho $t = 4$:

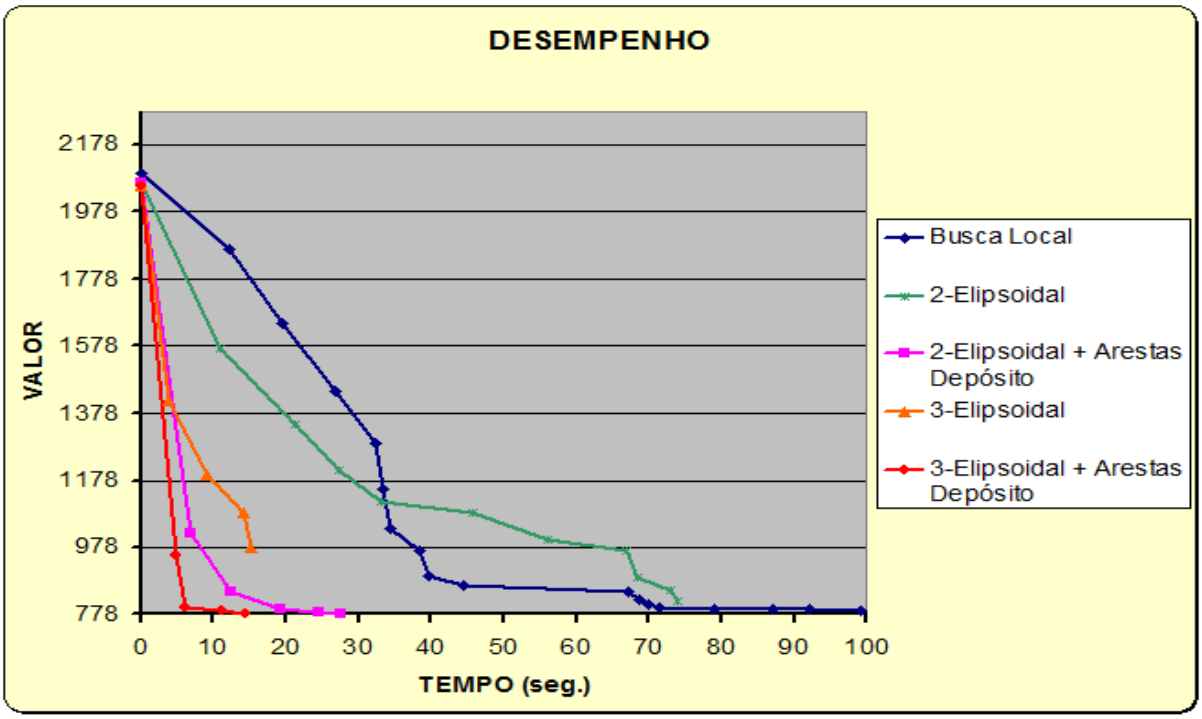

Figura 4.10: Desempenho comparativo para instância An-34-k5 com $t = 4$.

Como se pode ver, a busca local toma um tempo muito superior às buscas elipsoidais, sem sequer chegar ao ótimo. Isso se justifica pelo fato de ser uma vizinhança pequena ($t = 4$), em que a dificuldade de criar rotas na geração de colunas se faz presente e afeta em demasia o processo.

Entre as buscas elipsoidais, aquela baseada em 3 soluções obteve o pior desempenho, alcançando uma solução longe do ótimo. Entretanto, a adição de arestas do depósito muda drasticamente o panorama, fazendo-a atingir o ótimo rapidamente. Ademais, ambas as buscas elipsoidais para 2 soluções obtiveram um bom desempenho, com destaque ao fato de que as arestas do depósito levaram a um melhor tempo de computação.

A Figura 4.11 apresenta o desempenho da busca local e de variantes das buscas elipsoidais de 2 e de 3 soluções, para a instância B-n50-k7 sobre uma vizinhança de tamanho $t = 6$:

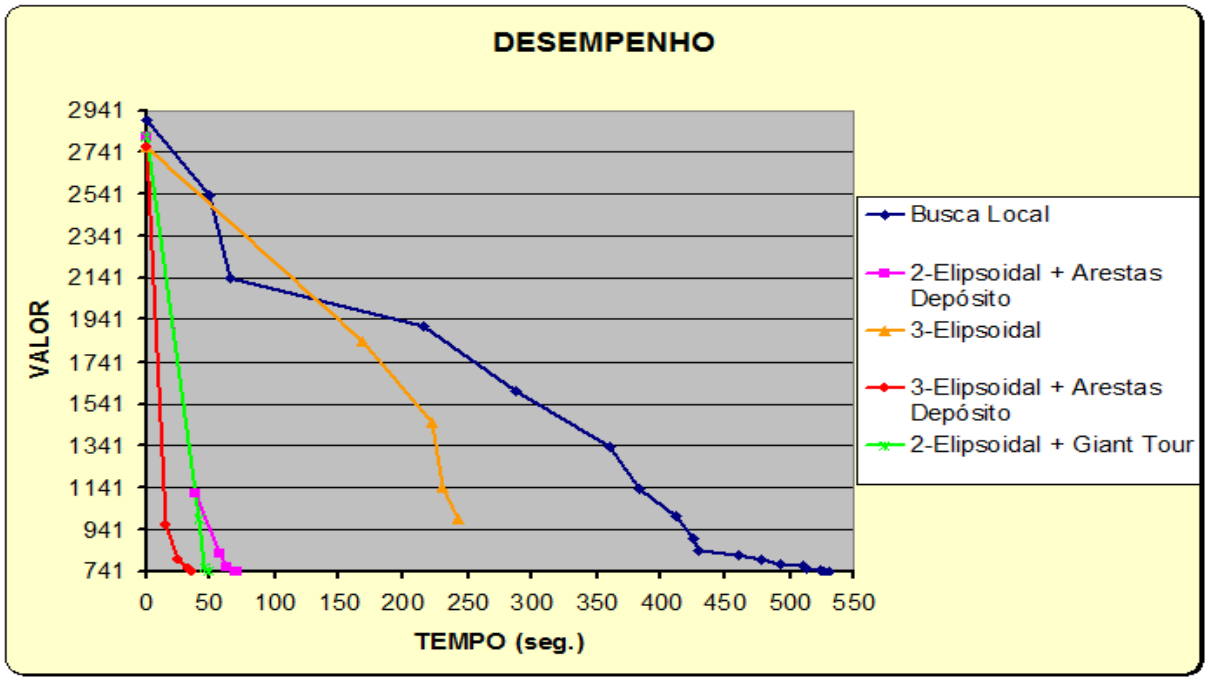

Figura 4.11: Desempenho comparativo para instância B-n50-k7 com $t = 6$.

Novamente, a busca local apresenta um desempenho muito ruim, com um tempo de computação elevado em relação às outras abordagens, por conta do tamanho pequeno de t. A elipsoidal de 3 também mostrou uma má performance, parando em uma solução bem acima do ótimo, porém consistindo na abordagem. Conforme visto, a adição das arestas do depósito aumenta a eficência dessa busca. Ademais, ambas as buscas elipsoidais de 2 soluções tiveram desempenho bem similares.

A Figura 4.12 exibe o gráfico para a instância E-n76-k10, com tamanho de vizinhança t igual a 10, em que novamente a elipsoidal de 3 com arestas do depósito conseguiu o melhor desempenho.

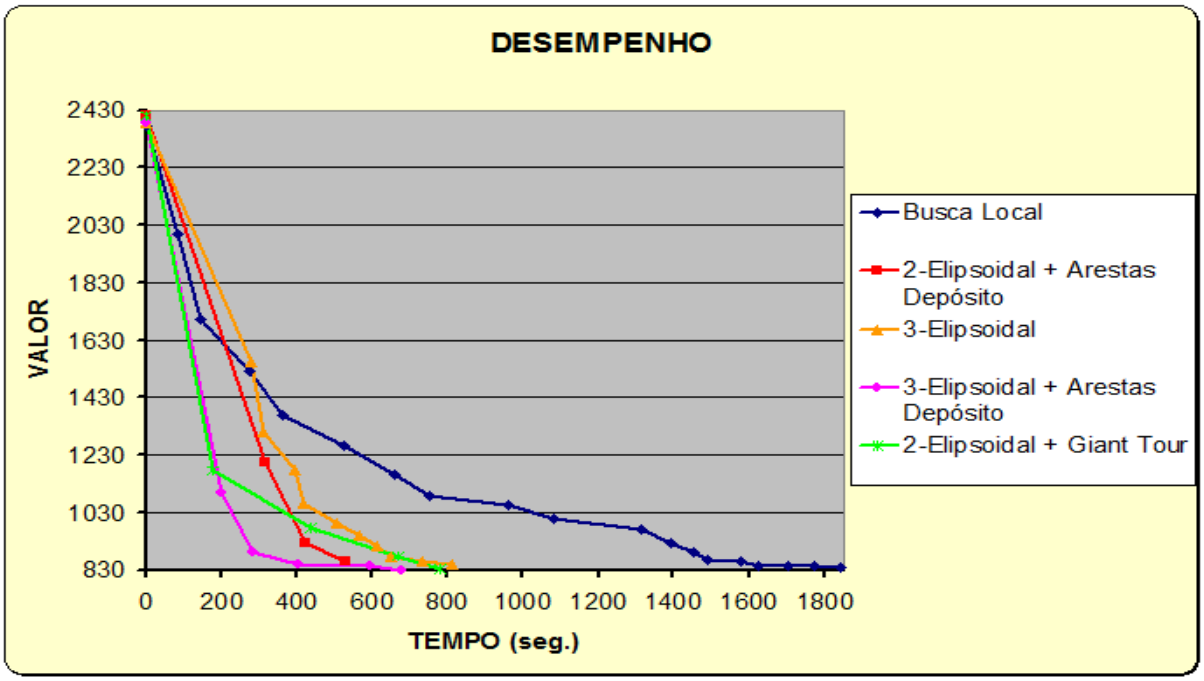

Figura 4.12: Desempenho comparativo para instância E-n76-k10 com $t = 10$.

A Figura 4.13 ilustra a mesma instância E-n76-k10, agora com t igual a 30. O desempenho de todas as abordagens, excetuando-se a busca local, foi relativamente similar, com leve vantagem para a elipsoidal de 2 com arestas do depósito. À medida que o tamanho da vizinhança t aumenta, tem-se que a diferença entre as abordagens tende a diminuir, como pode ser visto na próxima figura.

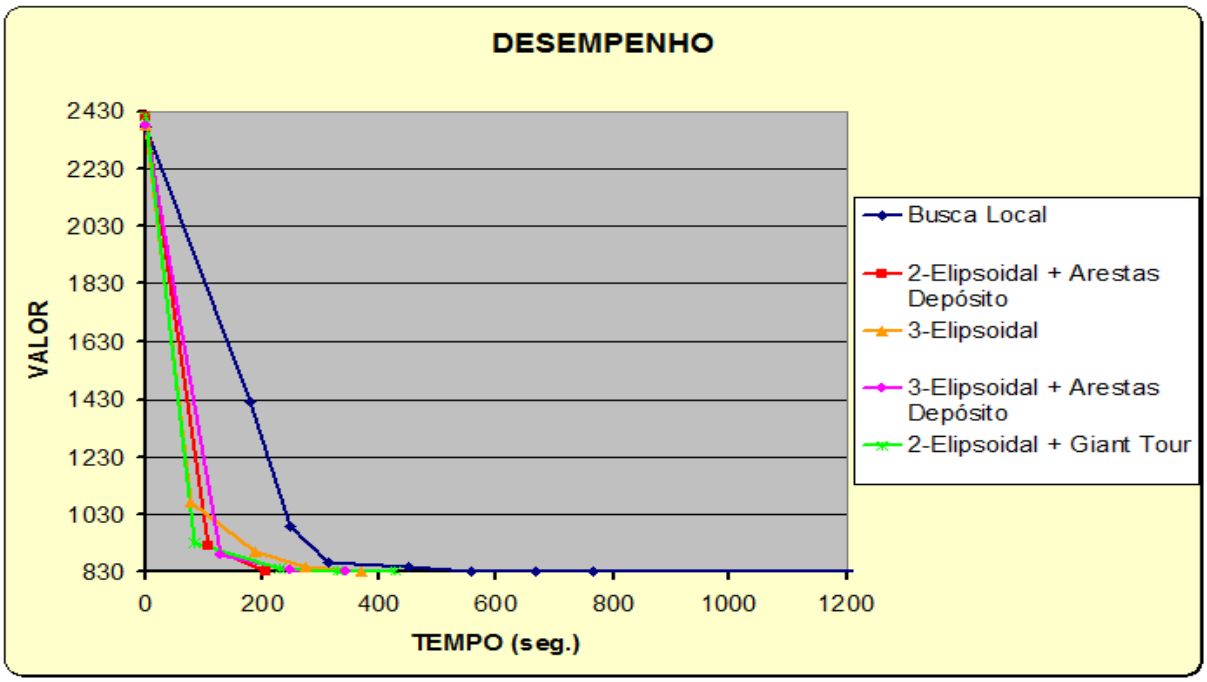

Figura 4.13: Desempenho comparativo para instância E-n76-k10 com $t = 30$.

Finalmente, pela análise dos gráficos nesta seção, percebe-se que as estratégias de inserção de componentes na estrutura dos cortes elipsoidais foram bem-sucedidas, na medida em que obtiveram os melhores tempos de execução e valores de solução encontrados. No entanto, ainda assim não foram suficientes para facilitar a obtenção de soluções viáveis na vizinhança, quando se está lidando com instâncias de maior porte. Acredita-se, pois, que o uso de estratégias de *Randomized Rounding* a partir da relaxação linear da vizinhança possa a vir a contornar esse problema.