Aplicação de Heurísticas de Busca Local ao Projeto de Bancos de Dados Distribuídos

Mutaleci de Goes Miranda (m_g_miranda@uol.com.br)
Paulo Roberto de Lira Gondim (pgondim@ime.eb.br)
Asterio Kiyoshi Tanaka (tanaka@uniriotec.br)[1]
Instituto Militar de Engenharia
Departamento de Engenharia de Sistemas
Praça General Tibúrcio 80 – Praia Vermelha
CEP 22290-000 – Rio de Janeiro – RJ - Brasil

ABSTRACT

A maior parte das abordagens de projeto de bancos de dados distribuídos têm tratado isoladadamente as fases de alocação e fragmentação de dados. Além disso, algoritmos de otimização específicos têm sido largamente empregados. O projeto de bancos de dados distribuídos como um todo é um problema de otimização NP-completo e, portanto, pode ser tratado por heurísticas de busca local. Neste trabalho, é proposta uma estrutura de vizinhança para o problema, que constitui um elemento fundamental para a obtenção de bons resultados quando este tipo de heurística é empregado.

Palavras-chave: projeto de bancos de dados, otimização, sistemas distribuídos.

1. INTRODUÇÃO

O projeto da distribuição de um banco de dados tem como objetivo gerar, a partir de um esquema conceitual global obtido em um projeto de banco de dados tradicional, esquemas conceituais locais através da distribuição de entidades pelos nós de um sistema distribuído[10]. O projeto deve ser realizado de maneira que o sistema de gerência de bancos de dados distribuídos (SGBDD) apresente um desempenho ótimo ou próximo do ótimo ao realizar consultas sobre o banco de dados distribuído (BDD) resultante. Medidas de desempenho normalmente utilizadas são o custo de comunicação ou o tempo de resposta do sistema. Além do esquema global, outros dados utilizados como entrada para o projeto são os padrões de acesso das aplicações aos dados e a topologia do sistema distribuído.

O projeto da distribuição é normalmente dividido em duas fases: fragmentação e alocação. Na fase de fragmentação, são determinadas as unidades de distribuição dos dados, ou fragmentos. Neste trabalho, serão tratados apenas bancos de dados relacionais e, neste caso, os fragmentos são normalmente subconjuntos de relações. Na fase de alocação são determinados os nós da rede que vão abrigar as unidades de distribuição.

A fragmentação de uma relação pode ser horizontal, vertical ou híbrida. Na fragmentação horizontal, cada relação global é dividida em subconjuntos de tuplas, enquanto a fragmentação vertical e

[1] Professor do Departamento de Informática Aplicada da UNI-RIO (Universidade do Rio de Janeiro). Parcialmente patrocinado pelo projeto CNPq-INRIA processo nº 68.0139/98.2.

caracterizada por projeções das relações globais. A fragmentação híbrida é a combinação das duas abordagens [8, 10]. As fragmentações horizontais podem ainda ser classificadas em primárias e derivadas. Em uma fragmentação primária, os subconjuntos de tuplas são obtidos através da aplicação de seleções, ou seja, cada fragmento é formado pelas tuplas que satisfaçam a uma determinada condição de seleção definida sobre atributos da relação. Um algoritmo formal para determinação destas condições é apresentado por Özsu e Valduriez [10]. Em geral, são utilizadas conjunções de predicados simples e complementos destes predicados para representar as condições. Um predicado simples é uma igualdade ou desigualdade entre uma única coluna de uma relação e algum valor constante. Se uma relação que sofreu fragmentação primária possuir relações filhas, isto é, relações ligadas à relação mãe por chave estrangeira, as relações filhas podem sofrer fragmentação derivada da relação mãe. Neste caso, para cada fragmento da relação mãe é definido um fragmento correspondente da relação filha. Cada fragmento derivado é formado por tuplas que recebem valores da chave estrangeira que liga as relações mãe e filha apenas de seu fragmento primário correspondente. Grande parte das abordagens de fragmentação vertical recebe como entrada a matriz de utilização de atributos, que fornece a freqüência com que cada coluna de uma relação é utilizada pelas aplicações, para derivar a matriz de afinidade de atributos, que fornece a freqüência com que cada par de colunas é utilizada conjuntamente pelas aplicações. A partir das informações da matriz de afinidade, as colunas são agrupadas para formar os fragmentos verticais através de técnicas tais como o "bond energy algorithm" [7] e a utilização de grafos de afinidades [9]. Um outro enfoque possível para a fragmentação vertical é a definição de uma função de custo explícita [3] para posterior minimização por algum método de otimização genérico.

Para que seja mantida a consistência dos dados, qualquer fragmentação realizada em um banco de dados deve obedecer à propriedade de completude, ou seja, todo item de dado presente nas relações globais deve estar presente em algum dos fragmentos obtidos após a fragmentação. Além disso, devem existir operações bem conhecidas que, ao serem aplicadas aos fragmentos, reconstruam as relações globais. As reconstruções de fragmentos horizontais e verticais são realizadas através de uniões e junções, respectivamente. Finalmente, os fragmentos devem ser disjuntos, ou seja, um item de dado não pode estar presente em mais de um fragmento. As exceções a esta última regra são os atributos chaves das relações, que devem estar presentes em todos os fragmentos verticais das mesmas a fim de permitir a reconstrução das relações globais.

A alocação em um banco de dados distribuído pode ser não-replicada, totalmente replicada ou parcialmente replicada. A alocação é dita não-replicada se cada fragmento é alocado em apenas um nó da rede. Se as relações não são fragmentadas e todo o banco de dados é replicado em todos os nós, tem-se a alocação totalmente replicada, que é a solução trivial de distribuição. Por último, quando réplicas de fragmentos podem estar presentes em um número arbitrário de nós, tem-se uma solução parcialmente replicada. A obtenção de uma alocação ótima corresponde à minimização de uma função de custo relacionada ao consumo de recursos de comunicações e processamento necessários para a execução de aplicações sobre o banco de dados distribuído. Para tratar esse problema, tanto algoritmos exatos, que fornecem soluções ótimas, quanto algoritmos heurísticos têm sido utilizados. Estes últimos fornecem soluções aproximadas, porém, em contrapartida, permitem que o problema seja modelado com simplificações menos restritivas, além de serem mais eficientes em termos de tempo de execução. Exemplos de métodos heurísticos podem ser encontrados em [1, 2, 5, 11], e alguns métodos exatos são apresentados em [1, 5, 6]. Os métodos descritos em [6, 11] podem fornecer soluções replicadas, enquanto os demais métodos citados fornecem soluções não-replicadas, a partir das quais uma solução replicada pode ser obtida através de métodos tais como o apresentado em [12].

O projeto de bancos de dados distribuído pode ser considerado um único problema de otimização. Embora o espaço de soluções torne-se bastante complexo sob esta abordagem, vamos considerar que o mesmo pode ser adequadamente explorado através de heurísticas de busca local tais como a busca tabu [4]. Qualquer subconjunto das soluções de um determinado espaço, que podem ser atingidas através da aplicação de operações simples sobre uma dada solução, é denominado vizinhança da solução. Uma busca local se caracteriza por ser um método iterativo que, partindo de uma solução inicial fornecida pelo usuário ou gerada dentro do algoritmo, seleciona a cada passo uma solução pertencente à vizinhança da solução corrente. As operações que podem ser aplicadas a uma solução para gerar sua vizinhança são denominadas movimentos e definem a estrutura da vizinhança. Neste trabalho, é proposta uma estrutura de vizinhança que permite que uma única heurística de busca local atinja soluções correspondentes a um projeto com fases de fragmentação horizontal primária, fragmentação vertical e alocação com replicação.

O restante do trabalho está organizado como se segue. Na seção 2 são definidos os movimentos que compõe a estrutura de vizinhança proposta. Na seção 3 são descritos os experimentos que validaram a viabilidade do emprego da estrutura, através da heurística de busca tabu. Finalmente, na seção 4, são apresentadas nossas conclusões.

2. A ESTRUTURA DE VIZINHANÇA

Uma solução de um problema de distribuição de banco de dados é o resultado do agrupamento das colunas e tuplas das relações de um esquema global em fragmentos e da alocação de uma ou mais cópias desses fragmentos aos nós da rede. Dados um esquema global formado pelas relações $R_1, \ldots, R_r$ e uma rede de computadores composta pelos nós $n_1, \ldots, n_s$, um fragmento será definido como uma tripla $F_{ji} = (C_{ji}, P_{ji}, N_{ji})$, $1 \leq i \leq r$, $1 \leq j \leq nf_i$, onde C_{ji} é um subconjunto das colunas não chave da relação R_i, P_{ji} é um subconjunto dos predicados simples definidos sobre colunas da relação R_i, N_{ji} é um subconjunto dos nós da rede, r é o número de relações globais, s é o número de nós na rede e nf_i é o número de fragmentos da relação R_i. Em outras palavras, definimos um fragmentos em termos das colunas contidas no fragmento, dos predicados simples satisfeitos pelas tuplas do fragmento e dos nós da rede nos quais existe uma cópia do fragmento. Dessa forma, um conjunto correto de fragmentos, ou seja, que satisfaça as propriedades de completude, disjunção e reconstrução, representará uma solução de fragmentação híbrida e de alocação simultaneamente.

Os movimentos da vizinhança proposta foram definidos como operação sobre fragmentos. Os tipos de movimentos concebidos foram o Particionamento Vertical, a Junção, o Particionamento Horizontal, a União, a Replicação, a Exclusão e a Realocação. Este conjunto de tipos de movimentos foi elaborado de modo a atingir os seguintes objetivos:

- Manter a correção do conjunto de fragmentos, ou seja, garantir a completude, a disjunção e a reconstrução dos mesmos;
- Obter uma quantidade de movimentos possíveis a cada passo da busca de ordem polinomial, a fim de não comprometer a eficiência do algoritmo de busca;
- Permitir que qualquer solução de fragmentação híbrida gerada por fragmentações horizontais primárias e fragmentações verticais alternadas possa ser eventualmente atingida;
- Determinar vizinhanças simétricas para as soluções. Tal objetivo é atingido pela definição de um tipo de movimento reverso para cada tipo de movimento. A existência de movimentos reversos assegura ainda a reconstrução dos fragmentos;

- Incorporar ações de fragmentação vertical e horizontal e de alocação com replicação que possam ser escolhidas indistintamente a cada passo da busca.

Nas seções 2.1 a 2.7 , serão definidas as características de cada tipo de movimento.

2.1 - Particionamento Vertical

Um movimento de Particionamento Vertical substitui um fragmento $F_{ji} = (C_{ji} , P_{ji} , N_{ji})$, pelos fragmentos $(C_{ji} - \{c\}, P_{ji} , N_{ji})$ e $(\{c\}, P_{ji} , N_{ji})$, onde $c \in C_{ji}$ e $\#C_{ji} > 1$, ou seja , realiza uma fragmentação vertical no fragmento de partida gerando um fragmento de chegada com apenas uma das colunas do fragmento de partida e um outro fragmento de chegada com as colunas restantes. Os predicados simples dos fragmentos de chegada são herdados do fragmento de partida garantindo que cada tupla do fragmento de partida estará presente nos fragmentos de chegada. Isto implica, juntamente com o fato de que a coluna ausente em um dos fragmentos de chegada compõe o segundo fragmento, a completude dos fragmentos. As localizações dos fragmentos de chegada também são herdadas do fragmento de partida, o que significa que todas as réplicas são particionadas no movimento, assegurando a disjunção dos fragmentos.

A definição do movimento limita em apenas $\#C_{ji}$ o número de Particionamentos Verticais possíveis a cada passo da busca para o fragmento F_{ji} , em contraste com as $O(n^n)$ possibilidades existentes, onde $n = \#C_{ji}$, se for admitida uma fragmentação vertical com número livre de fragmentos e de colunas dentro dos fragmentos [10]. Entretanto, combinações de sucessivos movimentos de Particionamento Vertical e movimentos de Junção podem gerar qualquer uma dessas possibilidades.

2.2 – Junção

Um movimento de Junção é o reverso de um movimento de Particionamento Vertical. O movimento substitui um par de fragmentos $F_{ki} = (C_{ki} , P_{ki} , N_{ki})$, $F_{li} = (C_{li} , P_{li} , N_{li})$ pelo fragmento $(C_{ki} \cup C_{li}, P_{ki}, N_{ki})$, onde $P_{ki} = P_{li}$, $N_{ki} = N_{li}$, $\#C_{li} = 1$ e $C_{li} \not\subset C_{ki}$. Portanto, o fragmento de chegada contém as colunas de ambos os fragmentos de partida, sendo um destes formado obrigatoriamente por apenas uma coluna. Além disso, a definição determina que o par de fragmentos de partida deve possuir conjuntos de predicados simples e de nós idênticos, condições que serão denominadas compatibilidade para junção.

2.3 - Particionamento Horizontal

Um movimento de Particionamento Horizontal substitui um fragmento $F_{ji} = (C_{ji} , P_{ji}, N_{ji})$ pelos fragmentos $(C_{ji}, P_{ji} \cup \{p\}, N_{ji})$ e $(C_{ji}, P_{ji} \cup \{\sim p\}, N_{ji})$, onde p é um predicado simples da relação R_i, $p \notin P_{ji}$, $\sim p \notin P_{ji}$ e nem $P_{ji} \cup \{p\}$, nem $P_{ji} \cup \{\sim p\}$determinam fragmentos vazios, ou seja , realiza uma fragmentação horizontal no fragmento de partida gerando um fragmento de chegada com as tuplas do fragmento de partida que satisfazem a p e um outro fragmento de chegada com as tuplas restantes. As colunas e a localização dos fragmentos de chegada são herdadas do fragmento de partida. Então, de

forma análoga ao movimento de particionamento vertical, ficam garantidas a completude e a disjunção dos fragmentos de saída.

Esta definição limita em $\#(P_i - P_{ji})$ o número de movimentos possíveis a cada passo da busca no pior caso para o fragmento F_{ji}, onde P_i é o conjunto dos predicados simples de Ri . Entretanto, ela permite que através de Particionamentos Horizontais sucessivos sejam produzidos quaisquer dos $O(2^n)$ fragmentos possíveis em uma fragmentação horizontal baseada em um conjunto de predicados simples com n elementos [10].

2.4 - União

Um movimento de União é o reverso de um movimento de Particionamento Horizontal. O movimento substitui um par de fragmentos $F_{ki} = (C_{ki}, P_{ki}, N_{ki})$, $F_{li} = (C_{li}, P_{li}, N_{li})$ pelo fragmento $(C_{ki}, P_{ki} \cap P_{li}, N_{ki})$, onde $C_{ki} = C_{li}$, $N_{ki} = N_{li}$, $P_{ki} - P_{li} = p$, $P_{li} - P_{ki} = \sim p$. Portanto, o fragmento de chegada contém as tuplas de ambos os fragmentos de partida. Além disso, a definição determina que o par de fragmentos de partida deve possuir conjuntos de colunas e de nós idênticos e conjuntos de predicados diferentes em apenas um predicado simples. Enquanto o referido predicado deve pertencer a um dos fragmentos, seu complemento deverá estar presente no outro fragmento do par. Estas condições serão denominadas compatibilidade para união.

2.5 – Replicação

Um movimento de Replicação substitui um fragmento $F_{ji} = (C_{ji}, P_{ji}, N_{ji})$ pelo fragmento $(C_{ji}, P_{ji}, N_{ji} \cup \{n\})$, onde $n \in N$ e $n \notin N_{ji}$, ou seja , realiza uma replicação do fragmento de partida gerando um fragmento de chegada com as mesmas colunas e predicados porém, presente em um nó adicional, o que corresponde fisicamente à criação de uma nova cópia do fragmento.

A cada passo da busca existem $\#(N - N_{ji})$ movimentos de Replicação possíveis para o fragmento F_{ji}, onde N é o conjunto dos nós da rede de computadores considerada.

2.6 – Exclusão

Um movimento de Exclusão é o reverso de um movimento de Replicação, ou seja, substitui um fragmento $F_{ji} = (C_{ji}, P_{ji}, N_{ji})$ pelo fragmento $(C_{ji}, P_{ji}, N_{ji} - \{n\})$ onde $n \in N_{ki}$ e $\#N_{ki} > 1$. Portanto, o fragmento de chegada possui as mesmas colunas e predicados, porém não está presente em um dos nós do fragmento de partida , o que corresponde fisicamente à remoção de uma cópia do fragmento. Além disso, a definição assegura que um fragmento não replicado não pode ser excluído para evitar perda de dados e conseqüente incorreção da solução.

A cada passo da busca existem $\#N_{ji}$ movimentos de Exclusão possíveis para o fragmento F_{ji}

2.7 – Realocação

Um movimento de Realocação substitui um fragmento $F_{ji} = (C_{ji}, P_{ji}, N_{ji})$ pelo fragmento $(C_{ji}, P_{ji}, N_{ji} - \{n_o\} \cup \{n_d\})$, onde $n_d \in N$, $n_o \in N_{ji}$ e $n_d \notin N_{ji}$, ou seja , realiza uma realocação de uma cópia do fragmento de partida, gerando um fragmento de chegada com as mesmas colunas e predicados porém,

com uma cópia alocada em um nó diferente, o que corresponde fisicamente à transferência da cópia do fragmento entre o nó de origem n_o e o nó de destino n_d.

Um movimento de Realocação corresponde a um movimento de Replicação seguido de um movimento de Exclusão porém, foi definido de forma autônoma por acrescentar diversidade à vizinhança de uma solução sem adicionar grande complexidade à busca. A cada passo da busca existem $\# N_{ji} * \#(N - N_{ji})$ movimentos de Realocação possíveis para o fragmento F_{ji}.

3. EXPERIMENTOS

A fim de validar a estrutura apresentada, implementamos um algoritmo de busca tabu. A busca é iniciada em alguma solução viável e a cada iteração o processo procura realizar um movimento de descida, ou seja, encontrar uma solução com custo menor que o da anterior, embora movimentos de subida possam ser aceitos para tentar escapar de mínimos locais. Quando um movimento é realizado, os fragmentos de entrada do mesmo recebem uma restrição tabu, ou seja, qualquer movimento que forneça como saída os fragmentos que receberam a restrição não pode ser executado antes que se passe um certo número de iterações, após o qual a restrição é relaxada. Este número de iterações é denominado duração tabu. Movimentos podem ser executados mesmo sob restrição tabu quando satisfazem um critério de aspiração. No presente algoritmo, estamos considerando a aspiração por "default" e a aspiração por objetivo. A aspiração por "default" é aplicada quando todos os movimentos disponíveis são tabu e não satisfazem nenhum outro critério de aspiração. Neste caso, o movimento com restrição tabu mais próxima de ser relaxada é executado. A aspiração por objetivo é satisfeita quando um movimento tabu leva a uma solução cujo custo seja menor que o da melhor solução já visitada.

Após receber a solução inicial, o algoritmo constrói uma lista de movimentos possíveis para a mesma. Esta lista é ordenada de forma crescente na variação de custo provocada pelo movimento. A cada iteração, o primeiro movimento da lista ordenada é selecionado. Se esse movimento satisfaz a aspiração por objetivo, ele é aceito. Caso contrário, uma busca pelo primeiro movimento sem restrições tabu da lista é iniciada. Se nenhum movimento sem restrição tabu é encontrado para ser realizado, a aspiração por "default" é aplicada. Após a realização de um movimento, as listas de movimentos possíveis e de fragmentos tabu são atualizadas. Então, uma condição de parada é avaliada para determinar se o processo termina ou se uma nova iteração é executada. A condição de parada empregada é a realização de um número máximo de movimentos não proveitosos (MaxMNP), ou seja, movimentos que não levam a uma solução com menor custo que a melhor já visitada. Ao fim do processo, a melhor solução visitada é fornecida como saída. O valor de MaxMNP e da duração tabu para fragmentos que recebem restrição em uma dada iteração são calculados, respectivamente, como o número de movimentos possíveis para a solução corrente e a raiz quadrada deste número.

A função de custo utilizada foi definida como o somatório do custo individual de cada fragmento:

$$CUSTO = \sum_{i=1}^{r} \sum_{j=1}^{nfi} CF_{ji},$$

onde r é o número de relações globais e nf_i é o número de fragmentos da relação R_i. O custo CF_{ji} de um fragmento será definido, por sua vez, como:

$$CF_{ji} = CA_{ji} + CT_{ji} + CS_{ji} + CD_{ji},$$

onde CA_{ji}, CT_{ji}, CS_{ji} e CD_{ji} são respectivamente os custos parciais referentes ao ajustamento entre o fragmento e as aplicações que o utilizam, à transmissão de dados gerada pela execução das aplicações, ao armazenamento do fragmento nos nós e à disponibilidade do fragmento. Exceto no caso do custo de armazenamento, cada custo parcial de um fragmento foi definido como a média dos custos parciais em relação às aplicações que o utilizam, ponderada pela freqüência de execução das aplicações:

$$CA_{ji} = \sum_{Qk \in Q(Fji)} A_{jik} * f(Q_k),\ CT_{ji} = \sum_{Qk \in Q(Fji)} T_{jik} * f(Q_k),\ CD_{ji} = \sum_{Qk \in Q(Fji)} D_{jik} * f(Q_k),$$

onde $Q(F_{ji})$ é o conjunto da aplicações que utilizam o fragmento F_{ji} e $f(Q_k)$ é a freqüência de execução da aplicação Q_k. O custo parcial referente ao armazenamento é dado por:

$$CS_{ji} = \left(\sum_{n \in Nji} s(n) \right) * S_{ji},$$

onde s(n) é o custo de armazenamento por unidade de informação do nó n.

O custo parcial de ajustamento do fragmento F_{ji} em relação à aplicação Q_k é dado por:

$$A_{jik} = S_{ji} + QS_{ki} - 2*S_{jik},$$

onde S_{ji}, QS_{ki} e S_{jik} são os números de bytes necessários para representar respectivamente o fragmento F_{ji}, os dados que a aplicação Q_k utiliza da relação R_i e os dados que a aplicação Q_k utiliza do fragmento F_{ji}.

Por sua vez, o custo parcial de transmissão do fragmento F_{ji} em relação à aplicação Q_k é dado por:

$$T_{jik} = \left(\mathrm{MIN}_{n \in Nji} t(n, no(Q_k)) \right) * S_{jik}, \text{ se } Q_k \text{ é uma consulta ou}$$

$$T_{jik} = \left(\sum_{n \in Nji} t(n, no(Q_k)) \right) * S_{jik}, \text{ se } Q_k \text{ é uma atualização,}$$

onde $t(n, no(Q_k))$ é o custo de transmissão por unidade de informação transmitida entre o nó n e o nó que invoca Q_k. A expressão toma esse formato por estarmos considerando que a estratégia de controle de replicação "read-one-write-all" (ROWA) é utilizada e que os dados de todos os fragmentos envolvidos em uma aplicação são enviados para o nó que a executa para posterior processamento. Se existe uma ligação direta l entre dois nós, o custo de transmissão t recebe o valor do custo total da ligação (ctl(l)). Caso contrário, t recebe o custo total do caminho mínimo entre os nós. O valor de ctl(l) é dado pela expressão:

$$ctl(l) = (1 + ct(l)) * (1 + cd(l)) - 1,$$

onde ct(l) é o custo da ligação l e cd(l) é o custo referente à disponibilidade da mesma. O valor de cd(l) é calculado por:

$cd(l) = (1 - d(l))/ d(l)$,

onde d(l) é a disponibilidade da ligação l.

Finalmente, o custo parcial referente à disponibilidade do fragmento F_{ji} em relação à aplicação Q_k é dado por:

$D_{jik} = i_{jik} /(1 - i_{jik})$

onde a indisponibilidade i_{jik} do fragmento F_{ji} em relação à aplicação Q_k,é dada por:

$i_{jik} = \prod_{n \in Nji} (1 - d(n))$, se Q_k é uma consulta e por

$i_{jik} = 1 - \prod_{n \in Nji} d(n)$, se Q_k é uma atualização,

onde d(n) é a disponibilidade do nó n. Mais uma vez, a forma da expressão deve-se ao uso da estratégia ROWA.

Com o algoritmo descrito, foram realizados ensaios em uma série de pequenas instâncias de projeto de bancos de dados distribuídos. Como esquema global, foi utilizada uma relação fixa com oito colunas não-chave de tamanhos variados, sendo que sobre uma das colunas são definidos três predicados simples, cada um com seletividade 0.33, e sobre uma segunda coluna são definidos outros dois predicados simples, cada um com seletividade 0.5. Todos os predicados definidos são mutuamente excludentes em relação aos outros predicados da mesma coluna. Foram consideradas redes de 4, 8 e 12 nós com topologias em anel (A), em estrela (E) e totalmente conectada (T), perfazendo um total de nove configurações diferentes. Em cada nó foi definida uma única consulta. A solução inicial escolhida foi a relação não fragmentada e alocada em apenas um dos nós. Para cada configuração foram realizados dez ensaios com as seguintes variações:

- disponibilidades e custos de cada nó bem como cada ligação e freqüência de execução de cada consulta selecionados aleatoriamente com distribuição uniforme no intervalo [0.5, 1] ;
- tipo de cada consulta selecionado aleatoriamente com iguais probabilidades para leitura e atualização;
- número de colunas utilizadas em cada consulta selecionado aleatoriamente com igual probabilidade entre {1,...,4} e colunas utilizadas por cada consulta escolhidas também aleatoriamente com igual probabilidade entre as colunas da relação;
- número de predicados simples utilizados em cada consulta selecionado aleatoriamente com igual probabilidade entre {0,...,2} e predicados utilizados por cada consulta escolhidos também aleatoriamente com igual probabilidade, não podendo ocorrer, entretanto, a seleção de mais de um predicado da mesma coluna;
- nó que abriga a relação global na solução inicial selecionado aleatoriamente com igual probabilidade entre os nós da rede

Nas figuras 1, 2 e 3 são apresentados os gráficos comparativos entre a busca tabu e uma heurística baseada no método de Apers estático, que também foi implementada, para redes de topologia A (Anel), E (Estrela) e T (Totalmente conectada), respectivamente. A heurística baseada no método de Apers utiliza uma solução inicial com fragmentos mínimos alocados a nós virtuais [1] mas a função de

custo e os movimentos empregados são os do presente trabalho. Os gráficos exibem os percentuais máximo, médio e mínimo de redução de custo obtidos com a busca tabu em relação ao método de Apers entre os dez ensaios de cada configuração. O gráfico da figura 4 é a superposição dos três anteriores e indica que o desempenho da heurística é similar em todas as topologias avaliadas.

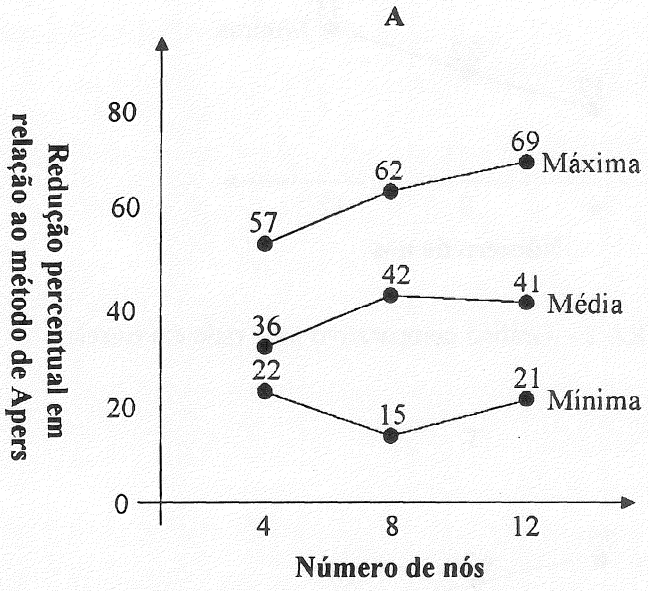

FIGURA 1 – Gráfico comparativo para rede em anel

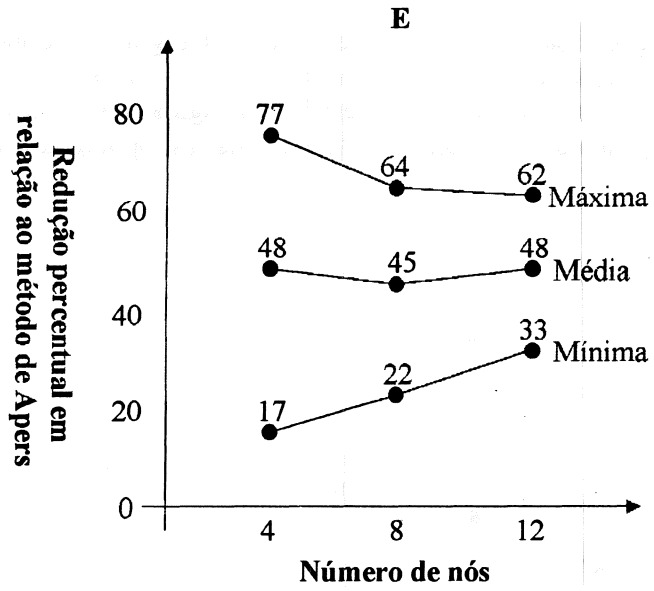

FIGURA 2 – Gráfico comparativo para rede em estrela

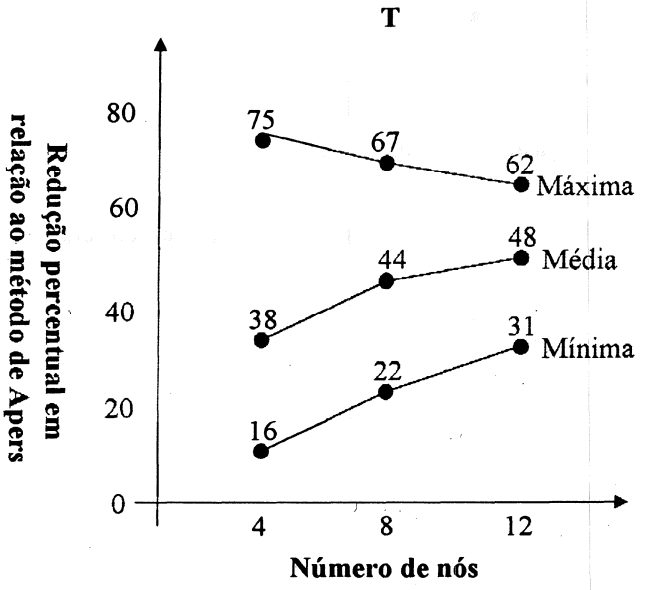

FIGURA 3 – Gráfico comparativo para rede totalmente conectada

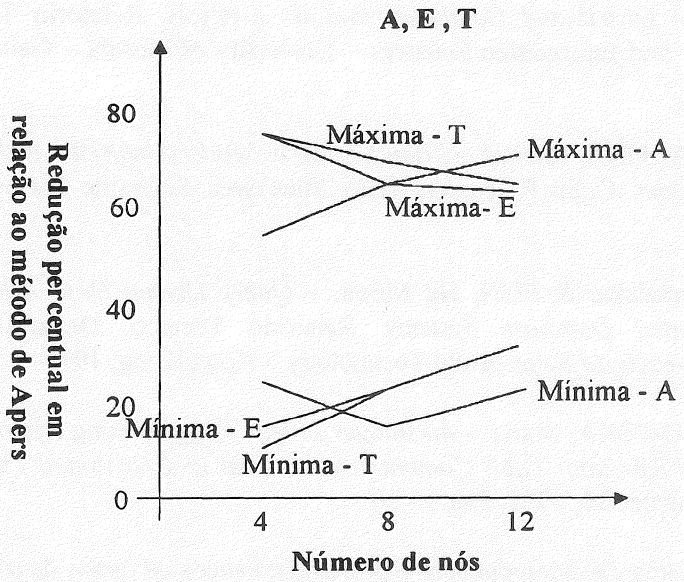

FIGURA 4 – Gráfico comparativo para as três topologias

4. CONCLUSÃO

Neste trabalho foi descrita e avaliada uma estrutura de vizinhança para o projeto de bancos de dados distribuídos, que consiste em um passo fundamental para o tratamento deste problema por heurísticas de busca local. A presente abordagem apresentou não só resultados quantitativos promissores, mas também a capacidade de aliviar o usuário de decisões relativas à escolha do tipo e da ordem das fragmentações a realizar no banco de dados.

Em [7] , foi especificada e prototipada uma ferramenta para o projeto de BDD utilizando a presente abordagem, que se mostrou adequada para as necessidades do Sistema de Pessoal do Exército Brasileiro (SISPEX), para o qual o trabalho foi originalmente desenvolvido.

Como trabalhos futuros, sugerimos a extensão da estrutura de vizinhança para admitir a fragmentação horizontal derivada e a avaliação da abordagem sob outras funções de custo, e outras heurísticas de busca ou até mesmo formas mais sofisticadas de busca tabu, uma vez que apenas os princípios básicos foram aqui empregados.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] APERS, Peter M. - *Data Allocation in Distributed Database Systems*. ACM Transactions on Database Systems, Vol. 13, N°. 3, Setembro de 1988, Pag. 263 - 304.

[2] BLANKINSHIP, Rex; HEVNER, Alan R.; YAO, S. Bing - *An iterative method for distributed database optimization*. Data & Knowledge Engineering 21 (1997), Pag. 1-30.

[3] CHAKRAVARTHY, Sharma et alii - *An Objective Function For Vertically Partitioning Relations in Distributed Databases and its Analysis*. Relatório Técnico. Department of Computer and Information Sciences – University of Florida – Gainesville, 1992.

[4] GLOVER, Fred & LAGUNA, Manuel. - *Tabu Search* in *Modern heuristic techniques for combinatorial problems*. Colin R, Reeves (Ed). Blackwell Scientific Publications, 1993.

[5] KARLAPALEM, Kamalakar & PUN, Ng Moon. – *Query Driven Data Allocation Algorithms for Distributed Database Systems*. Relatório Técnico. Department of Computer Science – University of Science and Technology – Hong Kong, 1995.

[6] LIN, Xuemin & ORLOWSKA, Maria. - *An Integer Linear Programming Approach to Data Allocation with the Minimun Total Communication Cost in Distributed Database* Systems. Information Sciences 85, 1995, Pag 1 - 10.

[7] MIRANDA, M.G. – Uma metodologia para o projeto de bancos de dados distribuídos. Tese de mestrado. Instituto Militar de Engenharia, Rio de Janeiro, Brasil, 1999.

[7] NAVATHE, S.;CERI, S.; WIEDERHOLD, G.; DOU, J. *Vertical - Partitioning Algorithms for Database Design*. ACM Transactions on Database Systems, Vol 9 No 4, Dec. 1984.

[8] NAVATHE, Shamkant B.; KARLAPALEM, Kamalakar ; RA, Minyoung. - *A Mixed Fragmentation Methodology for Initial Distributed Database Design*. Relatório Técnico. Database Systems R&D Center - University of Florida – Gainesville, 1994.

[9] NAVATHE, S. & RA, M. – *Vertical Partitioning for Database Design: A Graphical Algorithm*. ACM SIGMOD, Portland, June 1989.

[10] ÖZSU, M. Tamer & VALDURIEZ, Patrick. – *Principles of Distributed Database Systems*. Prentice Hall, Inc., 2nd. Edition, 1998.

[11] PAVLIDES, George S. – *Efficient manipulation of a set of fragments. Information and Software Technology*, 1995, 37(4), Pag 233 - 243.

[12] WOLFSON, Ouri & MILO, Amir. - *The Multicast Policy and Its Relationship to Replicated Data Placement*. ACM Transactions on Database Systems, Vol 16 No 1, March 1991, Pag 181 - 205.