Vegner Hizau dos Santos Utuni

Desenvolvimento e aplicação de um *software* cristalográfico com protocolo de acesso a banco de dados distribuído

Tese apresentada ao Instituto de Química, Universidade Estadual Paulista, como parte dos requisitos para obtenção do título de Doutor em Química

Orientador: Prof. Dr. Carlos de Oliveira Paiva Santos

Araraquara
2009

Dedico este trabalho a deusa luz, Lucina.

Agradecimentos

Agradeço aos meus pais e a minha família pelo exemplo, incentivo, amor e carinho.
Ao meu orientador pela coragem em me aceitar como orientado.
Aos meus amigos pela convivência e atenção nos momentos alegres e tristes
A CAPES pela bolsa concedida.

Resumo

Desde a revolução provocada pela segunda geração de computadores ocorrida por volta de 1960 e que permitiu a disseminação dos computadores para os diversos setores da sociedade, vem acontecendo uma evolução na capacidade de processamento dos chips e conseqüentemente no conceito de *software*. Em se tratando especificamente de *softwares* científicos, esse incremente no volume e velocidade de processamento de dados torna possível a aplicação de modelos físico-químicos cada vez mais complexos. Em 1969, o cristalógrafo Hugo Rietveld criou um método que utiliza este novo paradigma tecnológico e que hoje é conhecido como método de Rietveld. Desenvolvido especificamente para o refinamento de dados de difração raios X de amostra policristalinas, passou a ser utilizado em todas as áreas da pesquisa em novos materiais. Para uma boa estabilidade do processo de refinamento é necessário fornecer ao modelo uma aproximação inicial de cada fase que compõe a amostra. Esta exigência é necessária para permitir a estabilidade do processo iterativo que irá ajustar os dados experimentais à função teórica, característica que obriga a uma dependência de bancos de dados especializados. O processo de refinamento utilizando o método de Rietveld é complexo e não linear o que implica necessariamente no uso de um *software*. Esta característica aliada à dependência de bancos de dados cristalográficos justifica a utilização da nova tecnologia de bancos de dados distribuídos, qualidade desejável e de grande interesse para a comunidade científica. Um banco de dados distribuído permite que os vários *softwares* que empregam o método de Rietveld troquem entre si as informações necessárias para iniciar um refinamento. O gerenciamento deste banco de dados é feito de forma automática pelo próprio *software* sem interferência humana. A viabilidade da hipótese da utilização de uma rede P2P com arquivos CIF foi demonstrada através da implementação do *software* Hera. Algoritmos inéditos para automatizar a criação, a manutenção e a validação de bancos de dados distribuídos foram criados e implementados. Também foi apresentada uma nova forma para a nomenclatura de arquivos CIF, Crystallographic Information File, baseada unicamente na cela cristalográfica independente de data, instituição e técnica utilizada. O *software* Hera é um programa de Search-Match que utiliza apenas arquivos CIF. Estes arquivos são armazenados em um banco de dados que é atualizado automaticamente pela rede P2P.

Palavras-chave: CIF, Search-Match, *Software* Cristalografico.

Abstract

Since the revolution occurred for the second generation of computers in the 1960 and that it allowed the computers dissemination for the diverse sectors of the society, consequently it comes happening a development in the capacity of chips processing and in the software concept. Specifically treating to scientific software, the volume and processing speed of data increasing becomes possible the application of more complexes physical-chemistry models. In 1969, the crystallographer Hugo Rietveld created a method that uses this new technological paradigm and that today is known as the Rietveld method. Developed specifically to the refinement of polycrystalline X-ray diffraction data, it passed to be used in all areas of the research in new materials. For a good stability of the refinement processes it is necessary to supply to the model an initial approach of each phase that composes the sample. This condition is necessary to allow the stability of the iterative process that will go to adjust the experimental data to the theoretical function, characteristic that it compels to a dependence of specialized data bases. The refinement process using the Rietveld method is complex and not linear which implies necessarily in the use of a software. This characteristic in combination to the dependence of a crystallographic data bases strongest justified the use of the new technology of distributed data bases, desirable quality and of great interest for the scientific community. A distributed data base allows that the several software that using the Rietveld method change information between itself necessary to initiate the refinement process. The management of this data base is made automatically without human interference. The hypothesis viability of the use of a P2P network with CIF archives was demonstrated through the Hera software implementation. New algorithms to automatize the creation, the maintenance and the validation of distributed data bases had been created and implemented. Also a new form for nomenclature of CIF archives nomenclature, Crystallographic Information File, based uniquely in the crystallographic cell independently of the date, institution and technique used, was presented. Hera implements a P2P network to exchange files CIF, Crystallographic Information File. The Hera is a software program for using Search-Match only with CIF files. These files are stored in the database that is updated automatically by the P2P network.

Keywords: CIF, Search-Match, Crystallographic Software

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 Introdução

Em 1960 o engenheiro Frederick Brooks de uma empresa mediana chamada IBM iniciou o projeto 360. O objetivo do projeto era criar uma nova linha de computadores. Este projeto, revolucionou o conceito de computador. A partir dele as palavras *hardware* e *software* passaram a ter o significado utilizado nos dias atuais. Este projeto foi conseqüência de uma tendência iniciada na segunda guerra mundial. A partir dos anos 1930, os computadores passaram a ter uma importância estratégica para todos os governos do mundo. Assim, empresas interessadas neste lucrativo mercado começaram a investir em pesquisa e a desenvolver máquinas cada vez mais potentes.

Os primeiros computadores eram feitos sob encomenda. Uma única máquina para um único cliente, não existia duas máquinas iguais. O representante mais famoso desta geração foi o computador ENIAC. Utilizado inicialmente para fins militares era composto por 17468 válvulas, 70000 resistores, 10000 capacitores e 7200 diodos de cristal de quartzo. Ocupava 170 m^2 e consumia 170 kW/h de energia elétrica. Porém, a tecnologia do ENIAC tinha um problema. As instruções fornecidas para a máquina funcionar eram únicas. Não podiam ser utilizadas em outra máquina ENIAC.

Estas instruções, que hoje são designadas como *software*, levavam vários meses para serem construídas e se a configuração de uma única válvula fosse trocada elas teriam que ser totalmente refeitas. No início da utilização destas máquinas este problema era tolerável, pois os *softwares* eram simples, mas com o tempo, as instruções foram ficando cada vez mais complicadas tornando a operação e manutenção desta geração de computadores complexa e dispendiosa. Para resolver este problema uma foi iniciado o projeto IBM 360.

O projeto IBM 360, mostrado na Figura 2, custou US$200 bilhões. Como meio de comparação, o projeto Manhattan que gerou a tecnologia para construir a bomba atômica custou US$ 20 bilhões. Com a tecnologia gerada a IBM criou uma nova geração de computadores, influenciou toda a cadeia produtiva e alavancando a sua participação no mercado americano de 25% para mais de 70%. Os governos, as instituições militares e as grandes empresas passaram a utilizar em grande escala

esta nova geração de computadores. Em poucos anos, todas as grandes universidades americanas e européias passaram a utilizar a tecnologia IBM.

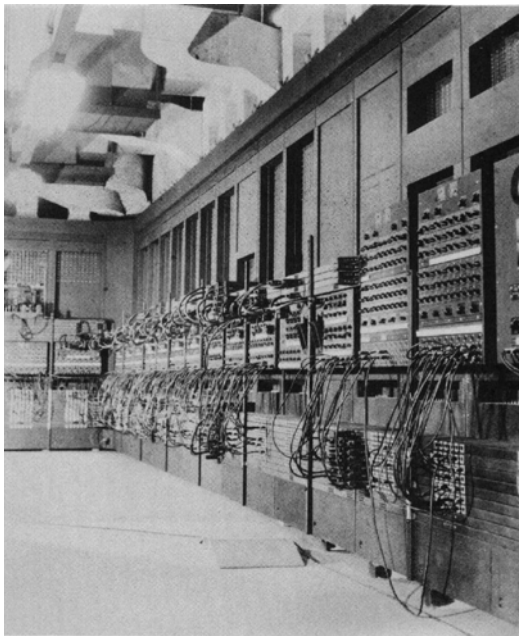

Figura 1 - Foto de partes do computador ENIAC.

Figura 2 - O computador IBM System 360/20. Museu Deutsches Munich Alemanha.

Os *softwares* agora podiam ser escritos para mais de uma máquina e podiam ser replicados. Um mesmo *software* podia rodar em mais de uma máquina sem nenhum problema. Além disso os *softwares* continuavam a funcionar mesmo quando o computador era atualizado. Estas características são banais nos dias atuais mas

nos anos 1960 causaram uma verdadeira revolução. Agora algoritmos que exigiam um *software* reutilizável, expansível e confiável poderiam ser utilizados em um computador, o impacto na Física e na Química foi enorme. A evolução dos conceitos originados no projeto 360 levou ao desenvolvimento do que hoje é identificado como *software* científico.

A revolução provocada pela segunda geração de computadores está ocorrendo novamente. Os primeiros *softwares* científicos utilizados na Física e na Química eram limitados pela capacidade de cálculo e memória disponíveis. Isto implicam na limitação da complexidade das funções que podiam ser implementadas.

Em 1995, a Universidade da Pennsylvania (SPIEGEL, 1995) reconstruiu um computador ENIAC utilizando as novas tecnologias disponíveis na época, que na época foi denominado projeto ENIAC-on-a-Chip. O resultado foi um micro chip de 0,5 cm^2 mostrado na Figura 3.

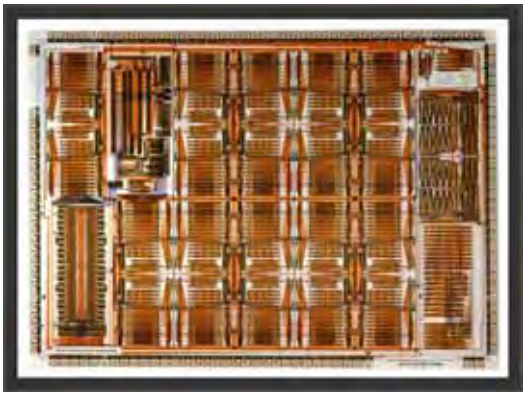

Figura 3- Chip ENIAC-on-a-Chip.

O projeto ENIAC-on-a-Chip demonstrou a evolução da tecnologia dos microchips em 50 anos. Um computador que custava US$10000000 em 1970 pode ser construído atualmente a um custo de US$10. A Figura 4 mostra a evolução dos microprocessadores nas últimas décadas. Todos os chips são do mesmo tamanho, a escala das imagens foi alterada conforme a capacidade de cálculos por segundo de cada um. A primeira imagem corresponde ao o chip ENIAC-on-a-Chip, a segunda ao Intel 8086, a terceira a um chip Intel 386, a quarta a um chip Intel Pentiun-I e a quinta um chip Intel CoreDuo-Quad. A figura ilustra o crescimento exponencial da

capacidade de cálculo dos computadores. A evolução foi de tal magnitude que hoje, no início do século XXI, está disponível para qualquer laboratório um supercomputador miniaturizado.

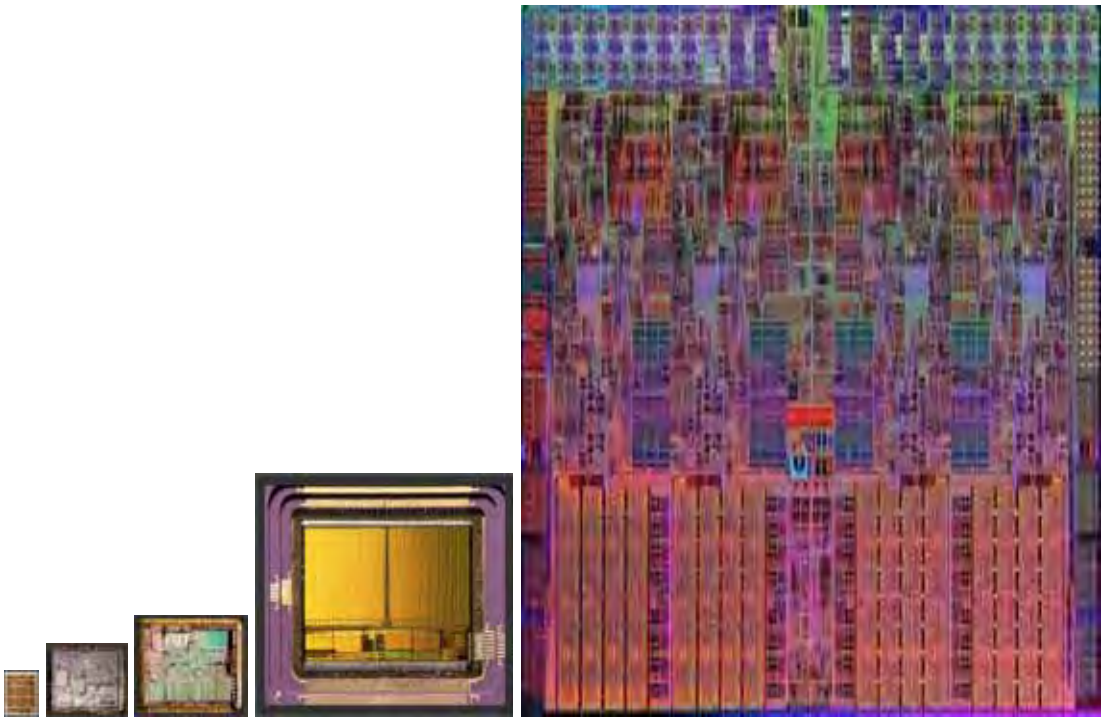

Figura 4 – Grafo ilustrando a evolução do microprocessadores.

Todos os modelos físico-químicos são baseados em três teorias fundamentais: a termodinâmica, a teoria da relatividade geral e a teoria da mecânica quântica. Quando um modelo físico-químico é implementado em um *software* científico, partes da teoria original são substituídas por equações mais simples. Isto é feito alterando-se a própria teoria do modelo de forma a utilizar equações mais lineares ou inserindo no *software* parâmetros previamente calculados por outra metodologia. Esta adaptação é necessária devido às limitações do hardware disponível.

A Figura 5 ilustra as mudanças relacionadas ao processo de evolução do desenvolvimento de aplicativos científicos que continuam ocorrendo nesta primeira década do século XXI. Nesta figura o circulo azul ilustra a segunda geração de *softwares* científicos e como este utilizam as teorias básicas. Estes utilizam máquinas com a tecnologia disponível até o início dos anos 1990. Os *softwares*

desenvolvidos nesta geração não podiam utilizar todo o potencial e nem utilizar todas as três teorias ao mesmo tempo.

O círculo em vermelho ilustra a nova geração. Como mostrado Figura 4, a partir dos anos 1990 uma nova geração de computadores ficou disponível para os desenvolvedores de *softwares* científicos. Esta nova geração de máquinas é formada pelos novos computadores pessoais.

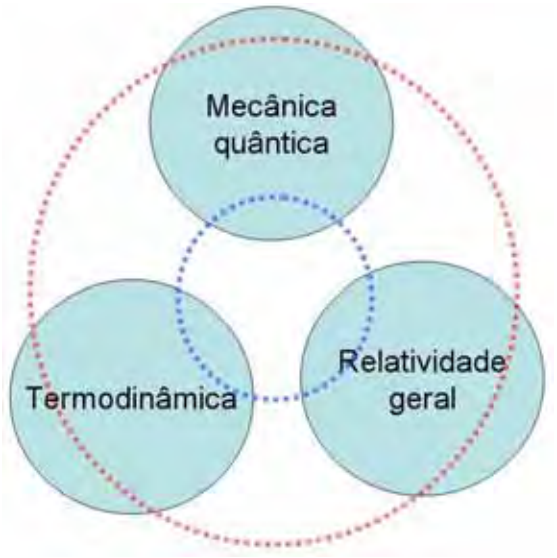

Figura 5 - Primeira e segunda geração de *software* científico.

Estes computadores agora permitem o desenvolvimento de *softwares* muito mais complexos e potentes. Esta nova geração de *software* é representada na Figura 5 pelo círculo em vermelho. A termodinâmica, a mecânica quântica e a teoria da relatividade geral agora são utilizadas de forma plena formando um conjunto sinérgico.

1.1 Hipótese e objetivo

Em 1969 o cristalógrafo Hugo Rietveld (RIETVELD, 1969) criou um método que utiliza este novo paradigma tecnológico. Hoje este método é conhecido como método de Rietveld. Desenvolvido especificamente para o refinamento de estruturas

cristalinas obtidas por difração raios-X e nêutrons em amostra policristalinas, passou a ser utilizado em todas as áreas da pesquisa em novos materiais.

O método de Rietveld ajusta uma função matemática de um modelo físico-químico aos dados experimentais utilizando o algoritmo apropriado, normalmente os mínimos quadrados (PRESS, 1992). A Equação 1 é a função que será minimizada. Nela **yo** representa um histograma experimental e **yc** o teórico. **Yc** é calculado utilizando a Equação 2 (YOUNG, 1992), um modelo de parâmetros físicos representado pelo vetor **v**. Os parâmetros são ajustados em ciclos iterativos até que a diferença entre **yc** e **yo** seja minimizada (YOUNG, SAKTHIVEL, MOSS & PAIVA-SANTOS, 1995).

$$f\left(\vec{v}\right) = \frac{\left(yo - yc\left(\vec{v}\right)\right)^2}{yo^2}$$

Equação 1 - Função a ser minimizado no método de Rietveld.

$$yc(\vec{v}) = S_r\left(\left(\sum_p S_p\left(\sum_k \left|F_{k,p}\right|^2 FP\left(2t_p - 2t_{k,p}\right) As_{k,p} L_{k,p} P_{k,p}\right)\right) + yb\right)$$

Equação 2 - Modelo físico para descrever o histograma teórico.

O algoritmo para o método de Rietveld realiza operações matemáticas sob todos os átomos de uma cela cristalográfica e em todos os planos hkl calculados. Em casos típicos são realizados cerca de 500.000 operações matemáticas em cada ciclo iterativo. Esta complexidade demonstra o quanto é importante dispor de um *software* reutilizável, expansível e confiável, conceitos possíveis a partir do projeto IBM 360.

O processo de refinamento é sensível à validade estatística dos dados originais, às orientações preferenciais dos planos de difração e à radiação de fundo da medida. O modelo físico-químico do método é sensível à escolha da função de perfil e à indexação das fases que compõem a amostra. O método de Rietveld não determina a estrutura cristalina do material, é um método de refinamento de uma estrutura conhecida. Para uma boa estabilidade do processo de refinamento é necessário fornecer ao modelo uma aproximação inicial de cada fase que compõe a

amostra. Esta exigência é necessária para permitir a estabilidade do processo iterativo que irá ajustar os dados experimentais à função teórica. Esta característica obriga a uma dependência de bancos de dados especializados. Estes bancos de dados fornecerão os valores iniciais adequados do grupo espacial, dos parâmetros da cela e das posições no espaço dos átomos assimétricos.

A característica de depender de bancos de dados especializados não é restrita ao método de Rietveld mas sim comum a vários métodos e técnicas científicas. Os bancos de dados científicos especializados são produzidos há várias décadas (NIST, 2004). Nos últimos 30 anos, com a disseminação da informática, a capacidade e o acesso a estes bancos cresceram exponencialmente (CHEMICAL A, 2004; CHEMICAL B, 2004).

O processo de refinamento utilizando o método de Rietveld é complexo e não linear, estas características implicam obrigatoriamente no uso de um *software.* Esta característica aliada à dependência de bancos de dados cristalográficos justifica a utilização da nova tecnologia de bancos de dados distribuídos, característica desejável e de grande interesse para a comunidade cristalográfica. Um banco de dados distribuído permite que os vários *softwares* utilizando o método de Rietveld troquem entre si as informações necessárias para iniciar um refinamento. O gerenciamento deste banco de dados é feito de forma automática pelo próprio *software* sem interferência humana.

A hipótese do projeto é que os *softwares* cristalográficos com ênfase em aplicativos que utilizam o refinamento do método de Rietveld sejam alterados ou reescritos para permitir o gerenciamento de um banco de dados distribuídos. Desta forma o usuário do método de Rietveld diminui a dependência de bancos de dados comerciais.

Considerando estas características como premissas iniciais o objetivo desta Tese foi desenvolver, implementar e aplicar a um *software* cristalográfico um algoritmo que gerencia a utilização de um banco de dados distribuído.

2 Metodologia

2.1 Fluxograma de trabalho

Como esta Tese envolvia o desenvolvimento de um *software* científico os trabalhos seguiram um fluxograma preestabelecido. Isso foi necessário devido à complexidade e ao grande número de detalhes necessários. Assim etapas diferentes do projeto podiam seguir sendo desenvolvidas separadamente e o foco no todo do projeto era mantido. O *software* desta Tese foi desenvolvido entre janeiro de 2005 e fevereiro de 2009. A cada seis meses os fluxogramas de trabalho eram reavaliados quanto a viabilidade e aos resultados obtidos.

A Figura 6 mostra a estrutura básica utilizada para a elaboração do software dividida em quatro etapas. A primeira etapa foi estudar o desenvolvimento de um novo *software* utilizando o método de Rietveld seguindo o objetivo da Tese. A segunda foi o estudo da viabilidade da adaptação dos *softwares* já existente e incluir neles a capacidade de trocar arquivos entre si. A terceira parte do desenvolvimento foi estudar quais arquivos seriam utilizados no novo protocolo desenvolvido na Tese.

A quarta etapa foi o desenvolvimento do protocolo da rede. Designada rede P2PCIF este protocolo é a lógica envolvida na troca dos arquivos entre várias cópias do *software*, um o algoritmo que garante o envio e o recebimento de um ou vários arquivos de forma segura. A topologia de rede é como os vários computadores utilizando o *software* se relacionam entre si.

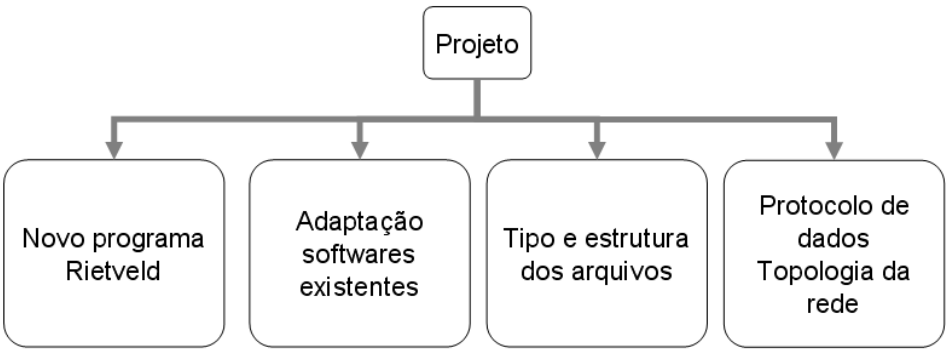

Figura 6 – Fluxograma de trabalho da tese.

2.2 Lógica dos diagramas

O protótipo do aplicativo apresentado nesta Tese foi implementado utilizando o ambiente Borland Delphi 6.0 (CODEGEAR, 2009; CANTÙ, 1996). A versão 0.38 que é apresentada nesta Tese é composta por mais de 64.000 linhas de código. Para apresentar todo o código fonte seriam necessárias mais de 1.800 páginas impressas. Como isto é inviável, neste texto todos os algoritmos são apresentados seguindo um diagrama formado por blocos simples. Este esquema é mais simples que os diagramas Nassi-Shneiderman e os diagramas de fluxo padrão (HEHL, 1986). A Figura 7 demonstra a lógica envolvida.

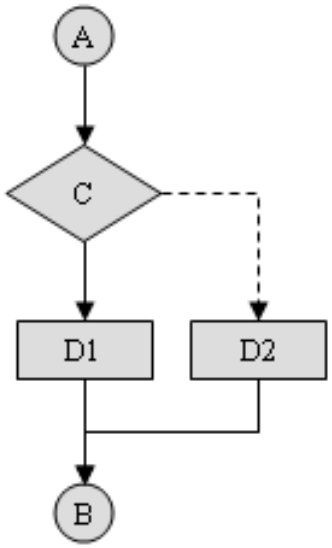

Figura 7 - Exemplo do diagrama dos códigos fontes.

Nestes diagramas os círculos representam o início e o fim do código. Os losângulos representam uma decisão a ser tomada, uma instrução condicional *IF*. Os retângulos representam as rotinas ou instruções que serão executas. As setas indicam o caminho que a execução do programa irá seguir, as linhas contínuas representam uma condição verdadeira, TRUE; e as setas tracejadas representam a condição falsa, FALSE. A tradução do algoritmo para alguma linguagem de programação é chamada de implementação, o texto resultante é designado por código fonte. O algoritmo da Figura 7 implementada em PASCAL e C ficariam:

```
{ Código fonte em pascal representando o algoritmo da figura 7 }
procedure NOME_PROCEDIMENTO;
begin  { Na imagem equivale ao circulo A }
        if C then D1 else D2;
end; { Na imagem equivale ao circulo B.}

// Código fonte em C representando o algoritmo da figura 7
void  NOME_PROCEDIMENTO()
{ // Na imagem equivale ao circulo A
        if (C)
        {
                D1;
        }
        else
        {
                D2;
        }
}   // Na imagem equivale ao circulo B.
```

Para os complexos códigos do aplicativo este esquema mostra de forma simples a lógica e os raciocínios envolvidos, sem a necessidade de apresentar milhares de linhas de código. Note que todo o código fonte também é disponibilizado no CD em anexo.

2.3 Novo programa utilizando o método de Rietveld

A Figura 8 mostra a proposta inicial para a criação de um novo *software* utilizando o método de Rietveld. O método exige três partes básicas. A primeira é responsável pelo cálculo da intensidade de cada plano hkl. Estas rotinas também devem verificar quais planos hkl são válidos. Para isso é necessário fornecer o grupo espacial, os átomos da unidade assimétrica e as suas coordenadas. Estas são as variáveis do banco de dados distribuído. A segunda parte corresponde às rotinas responsáveis pelas funções de perfil. Foi utilizada a função de Gauss e a função de

Voigt. A terceira parte do *software* é o ajuste dos vários parâmetros aos dados experimentais, na qual utilizou-se o método dos mínimos quadrados.

Todas as rotinas foram desenvolvidas de tal forma que funcionem independentemente umas das outras e que pudessem ser reutilizadas em outro *software*. Porém, foram projetadas seguindo os resultados da etapa responsável pelo desenvolvimento do protocolo da rede. No diagrama estas restrições são indicadas pelo quadro 3.3.

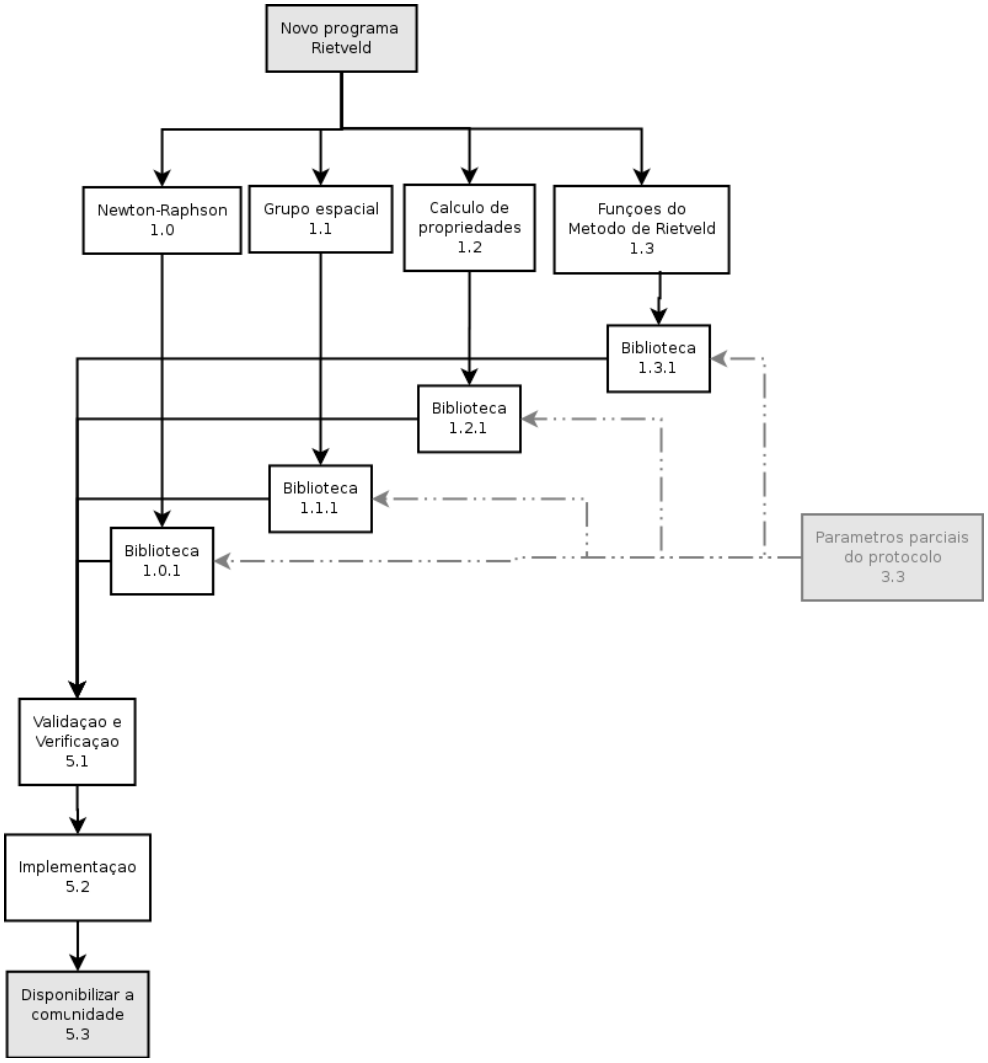

Figura 8 – Diagrama para um novo *software* com o método de Rietveld.

Esta etapa resultou o desenvolvimento de 3 aplicativos básicos. O primeiro foi um *software* utilizando o método de Rietveld clássico. O segundo é uma alteração do primeiro incluindo em algumas rotinas instruções oriundas da mecânica quântica. O

terceiro protótipo foi um programa de Search-Match. Devido à complexidade do segundo protótipo este foi pouco desenvolvido. A idéia era manter o código fonte do primeiro preparado para que partes deste fossem trocadas no futuro.

Os aplicativos 1 e 3 são semelhantes possuindo 90% do mesmo código. Desta forma foram desenvolvidos em conjunto, compartilhando quase todo o código fonte. Nesta Tese é apresentado o protótipo 3 designado por Hera. Hera é um programa de search-match. Neste tipo de aplicativo um histograma experimental é comparado aos planos hkl calculados ou lidos a partir de um banco de dados. Se o perfil do gráfico do banco de dados coincidir como padrão do perfil experimental admite-se que ambas as substâncias são idênticas. Este é um método eficiente para a identificação de amostras utilizando a difração de raios X, e também mais utilizado em laboratórios por ser mais simples.

3 Adaptação de *softwares* existentes

A outra abordagem foi adaptar *softwares* já existente e neles incluir rotinas que permitissem o acesso o banco de dados distribuído. Há dois *softwares* disponíveis com o código fonte. O programa DBWS (YOUNG, 1995) e o programa Rietan (IZUMI,2000; IZUMI, 2005). O mais utilizado nesta Tese foi o DBWS versão 98. O diagrama da Figura 9 mostra as etapas que foram seguidas. A adaptação é possível mas difícil. Como os *softwares* não foram originalmente projetados para isso, grande parte do código fonte original tive que ser alterado. Além disso foi trabalhosa a inclusão de rotinas gráficas a estes *softwares*. Devido a estes resultados indesejáveis esta etapa do desenvolvimento foi abortada priorizando-se o desenvolvimento de um *software* totalmente novo.

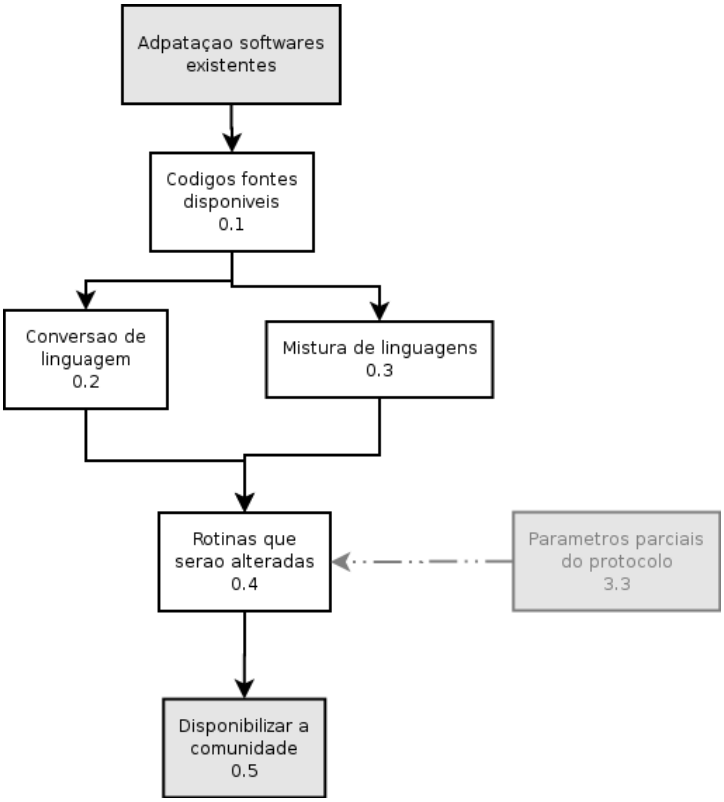

Figura 9 – Adaptação de *software* existente.

3.1 Escolha da topologia para a rede

O acesso à informação de confiança na atividade de pesquisa científica consiste em ponto importante no seu desenvolvimento e no processo da descoberta. A comunidade científica e a indústria utilizam esta informação na criação de modelos físico-químicos para explicar, visualizar e predizer o comportamento dos compostos químicos e dos materiais.

O método de Rietveld não determina a estrutura cristalina do material, é um método de refinamento de uma estrutura conhecida. Para uma boa estabilidade do processo de refinamento é necessário fornecer ao modelo uma aproximação inicial de cada fase que compõe a amostra. Esta exigência é necessária para permitir a estabilidade do processo iterativo que irá ajustar os dados experimentais à função teórica.

O processo de iteração, Equação 3, utilizado no método de Rietveld consiste em calcular incrementos, $\partial \vec{V}_n$, de tal forma que a diferença entre os dados observados e calculados seja mínima. Para que a minimização da Equação 1 seja

possível, os valores de $\partial\vec{v}_n$ devem ser pequenos em relação ao conjunto domínio das funções envolvidas.

$$\vec{v}_{n-1} = \vec{v}_n + \partial\vec{v}_n$$

Equação 3 - O processo iterativo.

O cálculo de $\partial\vec{V}_n$ é feito utilizando o inverso das derivadas parciais da função **yc** calculas sob cada ponto **yo**. Nesta metodologia, obrigatoriamente os valores inicias de $\vec{V}_n$ devem ser próximos dos valores finais. O algoritmo encontra um valor mais preciso mas não permite encontrar valores iniciais adequados. Caso os valores inicias não sejam adequados o processo iterativo fornecerá valores cada vez maiores e longe da realidade experimental.

Esta característica obriga a uma dependência de bancos de dados especializados. Estes bancos de dados fornecerão os valores iniciais adequados. Para o método de Rietveld é necessário conhecer previamente o grupo espacial, as dimensões da cela unitária e as posições atômicas dos átomos presentes na cela assimétrica. A característica de depender de bancos de dados especializados não é restrita ao método de Rietveld e sim comum a vários métodos e técnicas científicas. Os bancos de dados científicos especializados são produzidos há várias décadas (NIST, 2004). Nos últimos 30 anos com a disseminação da informática a capacidade e o acesso a estes bancos cresceram exponencialmente (CHEMICAL A, 2004; CHEMICAL B, 2004).

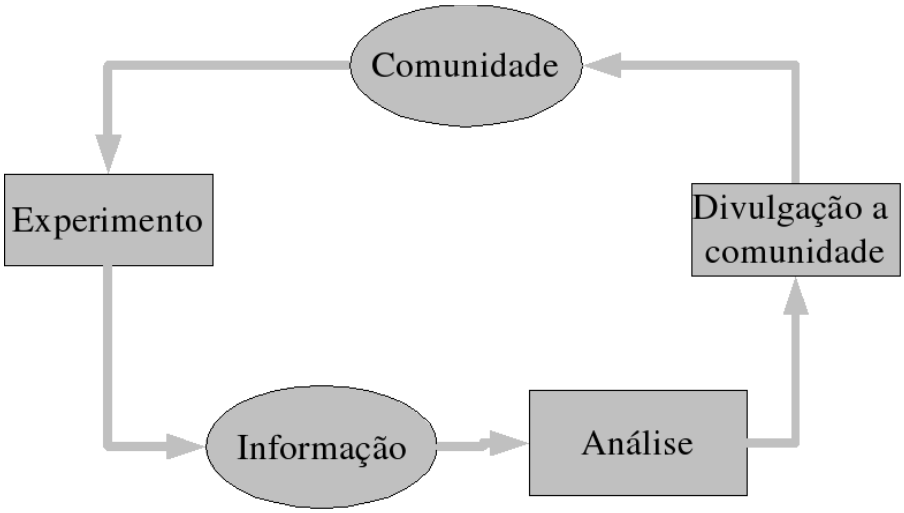

Figura 10 – Diagrama de produção e acesso à informação científica.

A Figura 14 apresenta um esquema possível para a produção da informação de caráter científico. Toda a informação é gerada em experimentos ou simulações controladas. Posteriormente, esta é analisada, filtrada, validada e comparada com o conhecimento já existente. Devido à complexidade e especificidade, o seu processo de produção e acesso obrigatoriamente é realizado por pessoas altamente especializadas nas respectivas áreas do conhecimento humano. Se os dados gerados forem pertinentes, a informação é disponibilizada à comunidade científica. Na Figura 10 a etapa *divulgação à comunidade* refere-se à publicação em periódicos, anais, resumos de congressos ou livros.

Este processo é cíclico, a informação disponível para a comunidade científica é reutilizada em novos experimentos gerando e aperfeiçoando a informação disponível. Com a evolução do próprio processo, criaram-se novas formas para a divulgação e acesso às informações. A nova forma consiste na utilização de meios eletrônicos como CD-ROMs e páginas WEB especializadas. Esta nova via de informação incentivou a criação de novos bancos de dados científicos. Há bancos de dados especializados em química analítica, química orgânica, engenharia química e bioquímica. As técnicas de espectroscopia também são utilizadas em conjunto com bancos de dados específicoa, como exemplos, podem ser citados a espectroscopia no infravermelho, de massa e de ressonância magnética nuclear.

Para aplicação em conjunto com o método de Rietveld destacam-se os seguintes bancos de dados:

- PDF-4, Powder Diffraction File (KABERKODU, 2002);
- ICDD, Inorganic Structure Database (FIZ & NIST, 2005);
- CRYSMET. Base de dados sobre ligas metálicas, compostos intermetálicos e minerais (WHITE, RODGERS & LE, 2002);
- IDITIS; Estrutura de proteínas (GARDNER & THORNTON, 1998);

Ao longo do tempo o tratamento de dados passou a ser feito quase que exclusivamente por *softwares* especializados. Isto foi provocado pela dependência dos bancos de dados científicos e pela necessidade de facilitar o processamento de uma quantidade de informações que cresce exponencialmente desde os anos 1990.

Os bancos de dados científicos atuais necessitam de uma equipe especializada para a sua criação e manutenção. Esta abordagem gera alguns problemas:

- Letargia na distribuição e inclusão de novas informações;
- Especialização dos bancos, impedindo inter-relações e cruzamentos de dados;
- Alto custo de manutenção e distribuição;
- Possibilidade de inserção de erros ou distorção das informações originais.

A busca da resolução destes problemas leva a uma solução alternativa. Ao invés de utilizar banco de dado centralizado utilizar bancos de dados distribuído (KANTERE, 2003; JUNGINGER, 2003; BERGER, 2003).

Figura 11 – A hipótese da Tese.

Figura 11 mostra a proposta da utilização de uma nova tecnologia no processo de criação e acesso à informação científica. A informação produzida, por cada pessoa, pode ser distribuída diretamente entre os usuários do novo *software* proposto nesta Tese. Na figura, ACESSO 1 representa as formas de organização e distribuição tradicionais e a conexão, ACESSO 2 ilustra a proposta desta Tese.

A hipótese da Tese é que os *softwares* utilizados no refinamento do método de Rietveld devem ser alterados ou reescritos para permitir a via de ACESSO 2. Nesta, a informação é organizada seguindo um protocolo rígido e distribuída diretamente entre os membros da comunidade. Nesta tecnologia a informação é organizada de tal forma que os dados possam ser redistribuídos e reorganizados em outros bancos de dados com formatos diferentes. Desta forma o usuário do método de Rietveld diminui a dependência de bancos de dados especializados, o próprio uso do *software* geraria o banco de dados.

Esta nova geração de *softwares* além de possibilitar o acesso aos bancos de dados distribuídos deve ser capaz de análises sofisticadas dos dados gerados nos experimentos para validação da informação, além de filtrar as informações vindas da comunidade pelo ACESSO 2. Estas características obrigam o *software* a conter uma enorme quantidade de conhecimento específico de cada área do conhecimento tornando-os extremamente especializados. O protocolo de troca das informações é geral e pode ser aplicado as variadas áreas do conhecimento humano, mas apesar de os *softwares* de acesso a estas informações são específicos de cada área.

Considerando estas características como premissas iniciais foram analisadas as tecnologias disponíveis que permitem a troca de informações entre dois *softwares* visando criar, desenvolver e implementar um protocolo para a utilização como banco de dados distribuídos na área de cristalografia. A
Figura 12 mostra o organograma inicialmente seguido.

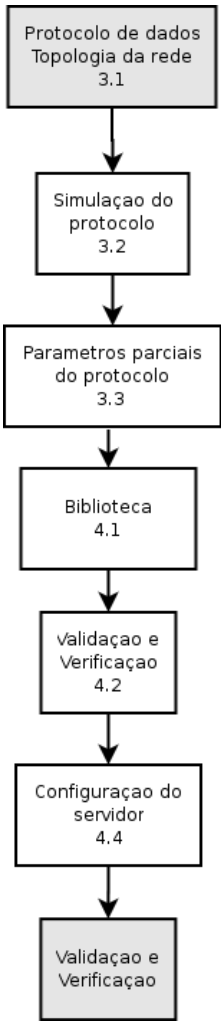

Figura 12 – Fluxograma referente ao desenvolvimento da rede.

Há duas topologias de rede básicas,: a rede Cliente/Servidor e a rede ponto a ponto. A Figura 13 ilustra a lógica da rede Cliente/Servidor. Na figura os quadros **cliente** representam os computadores dos usuários e o círculo **servidor** representa um computador central. Nesta rede todos os dados circulam passando pelo computador **servidor**. Para que uma pessoa envie ou receba dados de outra pessoa

obrigatoriamente ela deve enviá-los primeiro ao **servidor** e este envia para a outra pessoa. Se por algum motivo o computador servidor ficar inoperante todos os clientes não poderão mais trocar dados entre si. Este esquema de topologia é atualmente utilizado em páginas WEB.

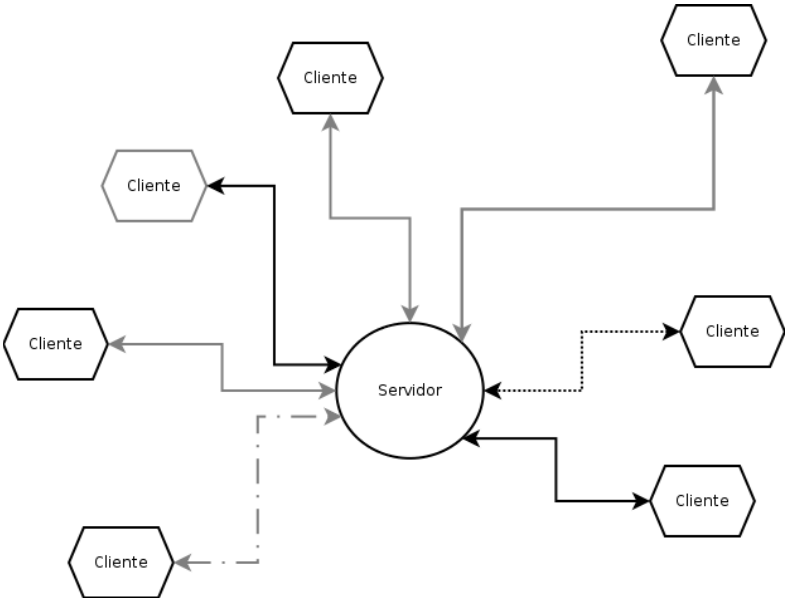

Figura 13 - Topologia cliente/servidor.

A topografia de rede ilustrada na Figura 14 representa uma rede designada como ponto a ponto ou *peer-to-peer*, P2P. Nesta topologia não existe um computador central. Todos os usuários podem enviar dados diretamente a outros usuários. A independência de um computador central permite que a rede continue em funcionamento mesmo se uma grande quantidade dos computadores estiver inoperante. Esta é a topologia utilizada na resolução de nomes WEB, DNS e em programas de troca de arquivos *peer-to-peer*.

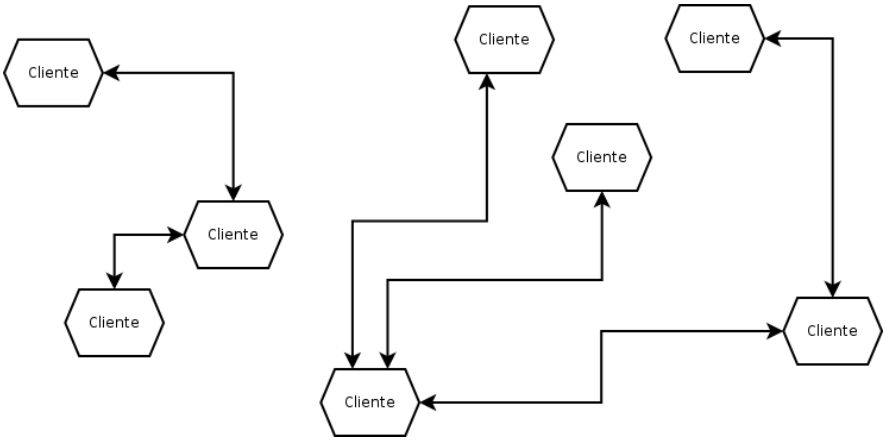

Figura 14 - Topografia rede P2P.

Nesta Tese avaliamos a utilização de ambas as redes, considerando as vantagens e desvantagens de ambas. Para isso o comportamento de ambas as redes foi simulado em condições próximas as utilizadas por *softwares* científicos. O algoritmo é mostrado na

Figura 17. Os gráficos da Figura 11 e Figura 16 e mostram os principais resultados. Os pontos dos gráficos foram obtidos realizando uma simulação de 50000 ciclos variando o limite de conexões da rede cliente/servidor. O limite de conexões dos servidores da rede P2P foi mantido fixo em 2. A taxa de novos arquivos foi obtida analisando a produção de novas estruturas cristalinas dos periódicos da União Internacional de Cristalografia, a probabilidade encontrada foi em torno de 0,0001. A probabilidade utilizada para criar um novo cliente na rede foi de 0,009. A probabilidade de um novo arquivo em cada ciclo foi de 0,1. Foi utilizado um fator 1000 vezes maior que o real para criar um ambiente de máxima tensão na rede. Todos os outros parâmetros foram mantidos iguais para ambas as redes incluindo a capacidade de cálculo e a largura de banda dos computadores cliente e servidor.

Nestas condições as velocidades de distribuição dos arquivos nas redes são diferentes, ambas as redes não conseguem distribuir os arquivos na mesma velocidade com que são produzidos. A Figura 15 mostra as conexões estabelecidas nas redes P2P e Cliente/Servidor utilizando servidores em duas condições. O servidor 1 foi considerado como um computador semelhante aos computadores clientes da simulação. Na configuração servidor 2 o computador utilizado possuía

uma CPU e uma conexão próxima uma situação real, mais potente mas com um custo 5 vezes maior. Abaixo de 1.000 conexões a eficiência da topologia cliente/servidor 2 é melhor e acima de 10.000 conexões a topologia P2P é melhor. Entre estes valores a eficiência de ambas as redes é semelhante.

A Figura 16 mostra a capacidade de distribuição de novos arquivos entre os usuários das redes. Na rede Cliente/Servidor o número de conexões é limitado pela capacidade do servidor, tanto de processamento, quanto da linha de transmissão de dados. A rede P2P cresce proporcionalmente ao número de clientes, por isso, está mantém um número quase constante de conexões. Devemos lembrar que os dados transmitidos são dados científicos, e que o tempo disponível no servidor é divido entre manter a rede e verificar a validade dos arquivos. No caso de arquivos científicos, os testes exigem muito processamento por parte dos computadores. Os resultados não serão os mesmos quando a rede utilizar arquivos mais simples como os formatos HTML, PDF e MP3.

Como não havia recursos disponíveis para um computador servidor a topologia de rede P2P foi escolhida para ser utilizada nos *softwares* desenvolvidos nesta Tese. A topologia *peer-to-peer* é menos exigente em *hardware* mas exige um grande número de usuários utilizando o mesmo *software*.

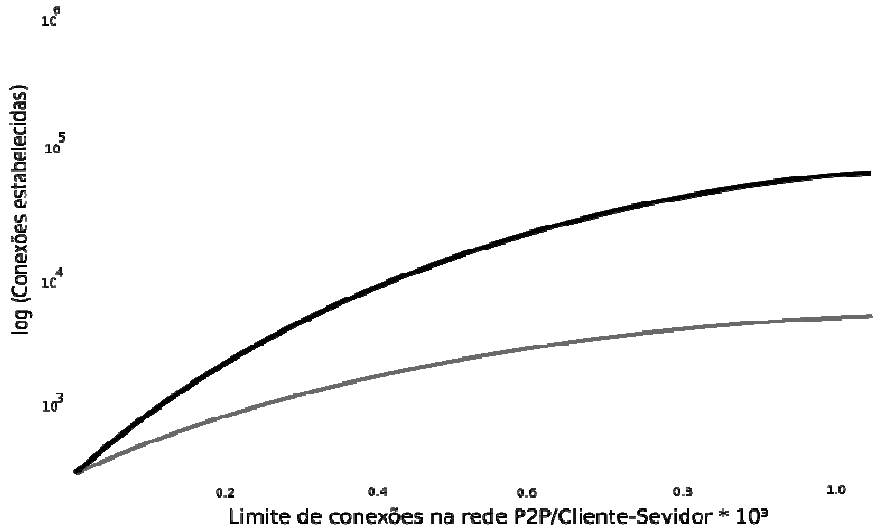

Figura 15 - Comparação entre redes, número de conexões.

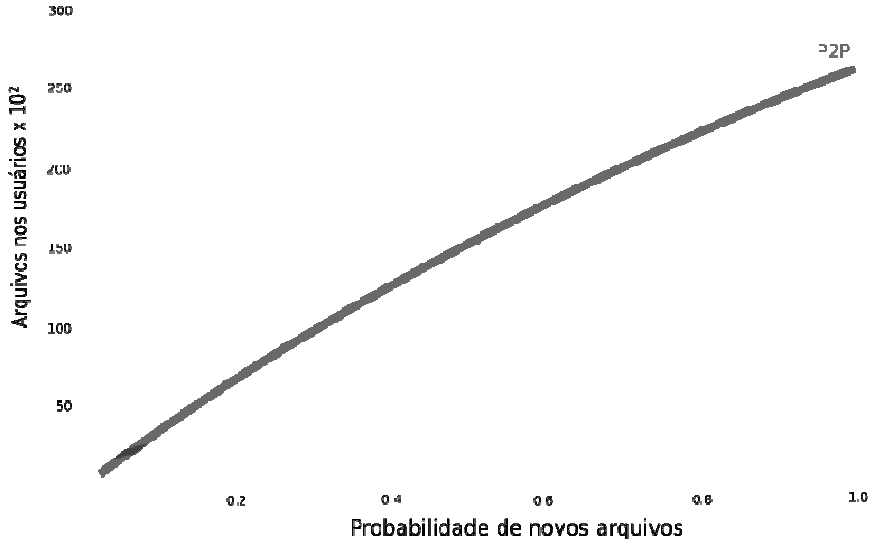

Figura 16 - Comparação entre redes, número de usuários.

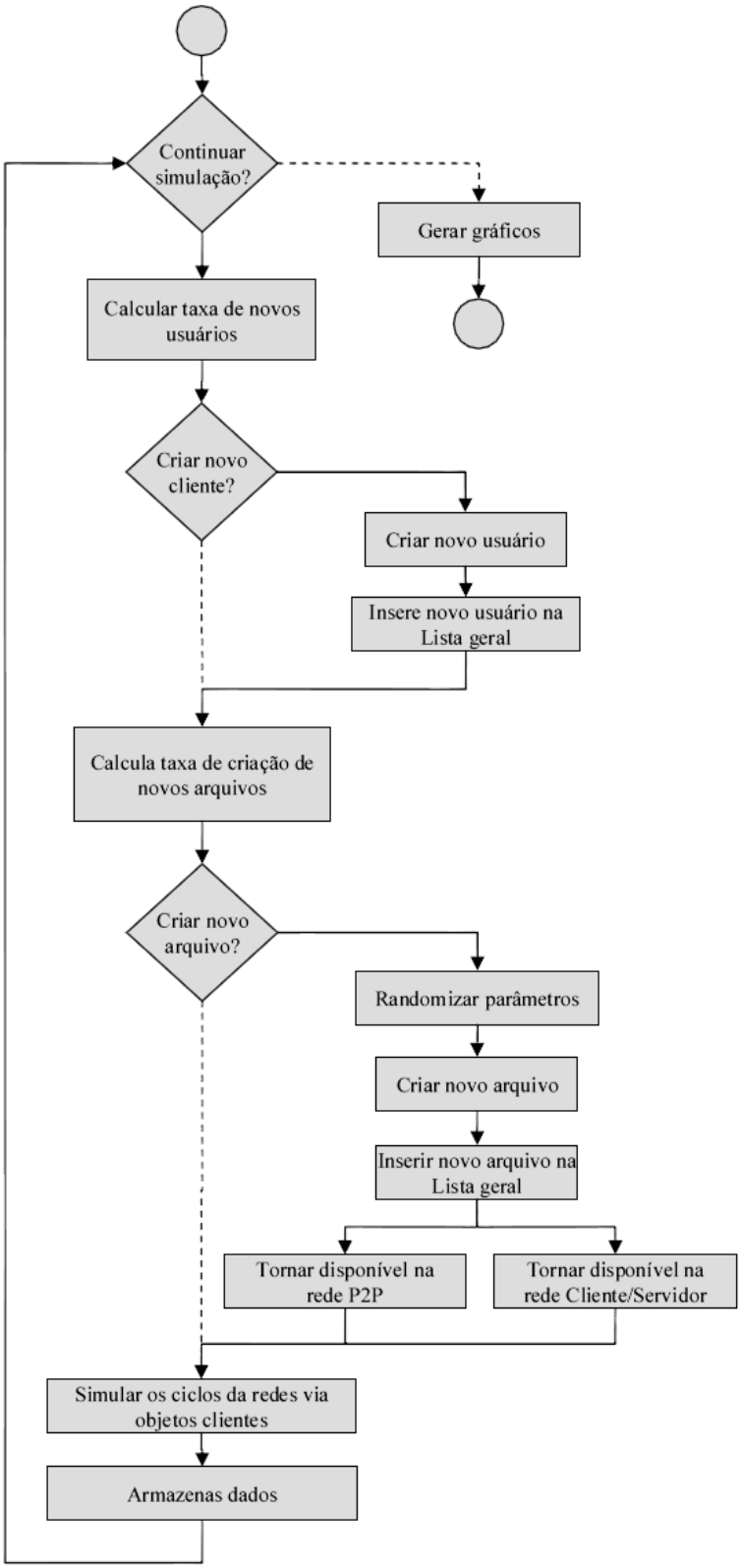

Figura 17 - Algoritmo para simulação de redes.

3.2 Tipo e estrutura dos arquivos

No início do desenvolvimento da Tese não estava claro qual o tipo de arquivo que seria utilizado. A proposta inicial era utilizar os arquivos aplicados em *softwares* utilizando o método de Rietveld disponíveis na Internet. Os principais arquivos encontrados e os arquivos a eles associados são mostrados na Tabela 1.

Para obtenção dos tipos de arquivos, mostrados na Tabela 1, foram utilizados os programas de conversão Convert (BOWWDEN, 2005), PowderV2 (DRAGOE, 2001), Powder4 (DRAGOE, 2004), DLConvert (GONTIER, 2005) e PowDLL (KOURKOUMELIS, 2004). Também foram analisados os seguintes programas de refinamento pelo método de Rietveld: DBWS (YOUNG et al, 1995; YOUNG, 2005) e RIETAN (IZUMI, 2005), GSAS(LARSON, 1997) e Fullprof (RODRÍGUEZ-CARVAJAL, 2001). Não foram utilizados programas comerciais que envolvessem o registro pago ou licenças.

Tabela 1 – Formatos *de arquivos utilizados com maior freqüência em difração de raios X*

Formato	***Software***
Philips VAX APD	Convert
Philips PC APD (RD format)	Convert, Powder4
GSAS (CW STD)	Convert, Powder4, DLConvert, GSAS
ASCII 2theta, I lists	Convert, Powder4
SCANPI	Convert
Sietronics CPI	Convert
Siemens DiffracPlus	Convert
FullProf	Convert, Powder4, FullProf
SRS Agular Dispersive Data	DLConvert
SRS Agular Dispersive Data 2.3 Xfit	DLConvert
SRS Agular Dispersive Data 9.1 XFit	DLConvert
SRS Agular Dispersive Data 2.3 CPI	DLConvert
SRS Agular Dispersive Data 9.1 CPI	DLConvert
ARGONNE Energy Dispersive Data – XFit	DLConvert

ARGONNE Energy Dispersive Data – CPI	DLConvert
ORTEC MCA Binary cnh Data – SRS 16.4 ASCII	DLConvert
RS 16.4 ASCII XY	DLConvert
DAT Rietan	RIETAN, Convert, DLConvert
DAT DBWS version 1	DBWS, DLConvert, Powder4
DAT DBWS version 2	DBWS, M4DBWS, DLConvert
PHILIPS RD	Powder4, PowDLL
Bruker/Siemens RAW version 1	Powder4, PowderV2, PowDLL
Bruker/Siemens RAW version 2	Powder4, PowderV2, PowDLL
Scintag ARD	PowDLL
Sietronics CPI	PowDLL
Riet7 DAT	PowDLL, Powder
Jade MDI	PowDLL
Rigaku RIG	Powder4
Philips UDF	Powder4, PowderV2
UXD	Powder4, PowDLL
XDA	Powder4, PowderV2
XDD	Powder4, PowDLL
CIF	Crystallographic Information File
XY	Powder4, Convert, DLConvert, PowderV2
XY – header	Powder4, Convert, DLConvert, PowderV2

O formato de um arquivo descreve como as informações estão organizadas dentro dele. Se há colunas, o número de colunas, se há variáveis e o número de variáveis que estão armazenas dentre de cada tipo de arquivo. Como um arquivo é apenas uma seqüência linear de bytes, estes podem ter infinitas combinações. Escolher qual formato um *software* utilizará é função do programador, e compreende uma das etapas inicias da criação de um *software*. Não era o objetivo desta Tese criar um novo formato por isso os vários formatos existente foram analisados para escolher qual era o mais adequado para o software desta Tese. Os resultados da comparação e análise dos vários formatos foram:

- Volatilidade dos arquivos.;
- Pouca diferença na estrutura dos arquivos;
- Os tipos de arquivo estão diretamente ligados ao tipo de difratômetro utilizado;
- Os formatos de arquivos são criados antes do fim do ciclo de vida útil de cada um.

Na Tabela 1 são mostradas a maioria dos formatos de arquivos encontrados e os mais freqüentemente utilizados. Esta diversidade gera uma dificuldade a mais para o pesquisador.

O tempo de vida curto é a principal característica destes arquivos. A associação com equipamentos explica parcialmente esta característica. A cada nova versão de um equipamento um novo formato é criado. Ou para justificar alteração mínima na interface destes equipamentos ou para não utilizar licenças ou patentes. A diferença entre os formatos é tão pequena que não justificaria a usa utilização. Esta diferença chega a ser de apenas duas a cinco letras em um arquivo com 1.000 linhas.

Uma causa desta diversidade é a grande amplitude de áreas utilizando métodos cristalográficos. Esta diversidade incluiu nos arquivos características de cada uma das áreas do conhecimento humano que se utilizavam da cristalografia. Originalmente, pretendia-se distribuir vários formatos de arquivos envolvidos com o método de Rietveld. Com estes resultados a proposta inicial foi alterada para a utilização um único formato de arquivo, o formato CIF ***C****rystallographic* ***I****nformation* ***F****ile* (HALL, 1991).

A diversidade de arquivos utilizados nas várias áreas da cristalografia chegou a um número tão grande que tornou-se inviável a manutenção dos vários formatos. No início da década de 1990 a União Internacional de Cristalografia passou a recomendar um único formato os chamados arquivos CIF. Estes arquivos possuem um formato padronizado com capacidade de representar desde dados simples até complexas estruturas de dados. O formato CIF permite incluir uma gigantesca diversidade de informações sem a necessidade de criar um novo formato para cada área do conhecimento humano.

3.3 Crystallographic Information File

O formato de arquivos CIF é o formato adotado como padrão pela União Internacional de Cristalografia (IUCR, 2009). A partir de 2001 o formato CIF passou a ser recomendado para todas as publicações da área de cristalografia e é obrigatório nos periódicos de maior impacto.

O CIF é um arquivo texto formatado em conformidade com rígidas e complexas especificações. Um arquivo CIF contém informações sobre um experimento cristalográfico, os seus resultados e as descrições destes dados. Ele é um arquivo somente de texto que poder ser visualizado ou editado por qualquer editor de texto independente da versão e do tipo de sistema operacional. Desta forma um CIF gerado em um ambiente Apple poder ser revisado em um ambiente UNIX e depois ser novamente editado em qualquer uma das versões do Windows.

A sintaxe completa de um arquivo CIF está apresentada na Tabela 2, onde todos os algoritmos e códigos fontes seguem o padrão desta tabela. No código fonte do programa a identificação de variáveis, constantes e classes associados ao arquivo CIF também seguem a Tabela 2.

Tabela 2 - A sintaxe completa do arquivo CIF (IUCR, 2009)

Unidade sintática	Sintaxe	Caixa
<CIF>	<Comments>? <WhiteSpace>? { <DataBlock> { <WhiteSpace> <DataBlock> }* { <WhiteSpace> }? }?	sim
<DataBlock>	<DataBlockHeading> {<WhiteSpace> { <DataItems> | <SaveFrame>} }*	sim
<DataBlockHeading>	<DATA_> { <NonBlankChar> }+	não
<SaveFrame>	<SaveFrameHeading> { <WhiteSpace> <DataItems> }+ <WhiteSpace> <SAVE_>	sim
<SaveFrameHeading>	<SAVE_> { <NonBlankChar> }+	não
<DataItems>	<Tag> <WhiteSpace> <Value> | <LoopHeader> <LoopBody>	sim

<LoopHeader>	<LOOP_> {<WhiteSpace> <Tag>}+	não
<LoopBody>	<Value> { <WhiteSpace> <Value> }*	sim
<DATA_>	{'D' | 'd'} {'A' | 'a'} {'T' | 't'} {'A' | 'a'} '_'	não
<LOOP_>	{'L' | 'l'} {'O' | 'o'} {'O' | 'o'} {'P' | 'p'} '_'	não
<GLOBAL_>	{'G' | 'g'} {'L' | 'l'} {'O' | 'o'} {'B' | 'b'} {'A' | 'a'} {'L' | 'l'} '_'	não
<SAVE_>	{'S' | 's'} {'A' | 'a'} {'V' | 'v'} {'E' | 'e'} '_'	não
<STOP_>	{'S' | 's'} {'T' | 't'} {'O' | 'o'} {'P' | 'p'}'_'	não
<Tag>	'_'{ <NonBlankChar>}+	não
<Value>	{ '.' | '?' | <Numeric> | <CharString> | <TextField> }	sim
<Numeric>	{ <Number> | <Number> '(' <UnsignedInteger> ')' }	não
<Number>	{<Integer> | <Float> }	não
<Integer>	{ '+' | '-' }? <UnsignedInteger>	não
<Float>	{ <Integer><Exponent> | { {'+'|'-'} ? { {<Digit>} * '.' <UnsignedInteger> } | { <Digit>} + '.' } } {<Exponent>} ? } }	não
<Exponent>	{ {'e' | 'E' } | {'e' | 'E' } { '+' | '- ' } } <UnsignedInteger>	não
<UnsignedInteger>	{ <Digit> }+	não
<Digit>	{ '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' }	não
<CharString>	<UnquotedString> | <SingleQuotedString> | <DoubleQuotedString>	sim

<eol><UnquotedString>	<eol><OrdinaryChar> {<NonBlankChar>}*	sim
<noteol><UnquotedString>	<noteol>{<OrdinaryChar>|';'} {<NonBlankChar>}*	sim
<SingleQuotedString> <WhiteSpace>	<single_quote>{<AnyPrintChar>}* <single_quote> <WhiteSpace>	sim
<DoubleQuotedString> <WhiteSpace>	<double_quote> {<AnyPrintChar>}* <double_quote> <WhiteSpace>	sim
<TextField>	{ <SemiColonTextField> }	sim
<eol><SemiColonTextField>	<eol>';' { {<AnyPrintChar>}* <eol> {{<TextLeadChar> {<AnyPrintChar>}*}? <eol>}* } ';'	sim
<WhiteSpace>	{ <SP> | <HT> | <eol> | <TokenizedComments>}+	sim
<Comments>	{ '#' {<AnyPrintChar>}* <eol>}+	sim
<TokenizedComments>	{ <SP> | <HT> | <eol> |}+ <Comments>	sim
<OrdinaryChar>	{ '!' | '%' | '&' | '(' | ')' | '*' | '+' | ',' | '-' | '.' | '/' | '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' | ':' | '<' | '=' | '>' | '?' | '@' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' | 'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z' | '\' | '^' | '`' | 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z' | '{' | '|' | '}' | '~' }	sim
<NonBlankChar>	<OrdinaryChar> | <double_quote> |	sim

	'#' | '$' | <single_quote> | '_' |';' | '[' | ']'	
<TextLeadChar>	<OrdinaryChar> | <double_quote> | '#' | '$' | <single_quote> | '_' | <SP> | <HT> |'[' | ']'	sim
<AnyPrintChar>	<OrdinaryChar> | <double_quote> | '#' | '$' | <single_quote> | '_' | <SP> | <HT> | ';' | '[' | ']'	sim

A Tabela 2 está organizada do campo mais complexo para o mais simples. A primeira coluna indica a identificação de cada componente ou parte de um arquivo CIF. A segunda coluna da tabela indica como o campo indicado na primeira coluna é formado. A terceira coluna apenas indica se as palavras da primeira e a segunda colunas irão fazer distinção entre letras maiúsculas e minúsculas.

As palavras entre os símbolos maior e menor, < >, identificam as partes que compõem o arquivo e que são chamadas de *tags*. Um arquivo CIF é a somatória de vários *tags* <> seguindo uma ordem lógica. O operador “|” indica um operador lógico ou, indica que um *tag* ou o outro *tag* pode ocorrer. O operador “&” indica o operador AND lógico, indica que ambos os tags obrigatoriamente devem ocorrer juntos. O operador “+” indica que várias tags devem ser somadas e utilizadas em conjunto para compor a informação desejada.

Tags entre chaves indicam a composição de campos mais complexos utilizando os operadores lógicos. O texto entre aspas indica a letra que pode ocorrer no arquivo. O tag <OrdinaryChar>, por exemplo, indica todas as letras que são válidas em um texto. Note que alguns caracteres associados às letras acentuadas são proibidos.

Com esta sintaxe um arquivo CIF permite criar muitos tipos de dados e estruturas de dados. O arquivo é formado pela seqüência de linhas de texto utilizando os caracteres padrões ASCII. As palavras data_, loop_, global_, save_ e stop_ são reservadas e não podem ser utilizadas em nomes e identificação de campos. Letras entre aspas simples e aspas duplas em uma mesma linha são identificadas como texto. Letras após um # fora de um texto são apenas comentários e não afetam a sintaxe de um arquivo CIF. Basicamente um arquivo CIF é formado

por blocos de dados que é a seqüência iniciada com a palavra chave data_. Dentro destes blocos tem-se variáveis e seus valores. As variáveis são palavras iniciadas com o caractere _. Como exemplo, segue um arquivo CIF simples.

```
# comentário
data_moleculateste
_symmetry_space_group_name_Hall      'P 61 2'
_cell_a                              8.53(1)
_cell_b                              8.53(1)
_cell_c                              20.37(1)
_symmetry_crystal_system             hexagonal
save_phenyl
        _object_class                    molecule_fragment
        loop_
            _atom_identy_node    _atom_identy_symbol
                                          1  C
                                          2   C
                                          3   C
save_
loop_        # mais um comentário
        _molecular_fragments  $ethyl  $phenyl  $methyl
loop_
        _atomic_name
        _atomic_energy
        _atomic_coefficient
                          Hidrogênio   1.3334897   1.0
                          Helio            2.0152458   1.2
# Isto também é um comentário. Fim do arquivo CIF
```

No exemplo, as palavras em preto são as palavras reservadas, em cinza são os comentários, em azul os nomes de variáveis e em vermelho os valores de variáveis.

Existe um tipo especial de arquivos CIF. Nestes arquivos estão definidos quais nomes de variáveis são permitidos em um arquivo e em qual contexto estes nomes podem aparecer. Estes CIF especiais são chamados de dicionários e são identificados pela extensão DIC, Dictionary Definition Languages. A União Internacional de Cristalografia recomenda o uso de 7 dicionários básicos, apresentadas na Tabela 3.

Tabela 3 - Dicionários CIF oficiais

Nome do dicionário	Descrição
coreCIF.dic	Núcleo básico
pdCIF.dic	Difração pelo método do pó
msCIF.dic	Estrutura de compósitos e de materiais modulados
rhoCIF.dic	Difração utilizando elétrons
mmCIF.dic	Cristalografia de macromoléculas
imgCIF.dic	Uso de imagens em cristalografia
symCIF.dic	Definição de simetria e grupos espaciais

O *software* Hera utiliza estes dicionários para verificar a integridade dos arquivos CIF que estão formando o banco de dados local. Há três versões destes dicionários, 1.0, 1.1 e 2.0. Na versão Hera 0.38 foi utilizada as especificações impostas pela versão 1.1 de dicionários. Esta opção foi escolhida para acumular a maior quantidade possível de arquivos CIF e permitir no futuro a implementação de uma rotina de conversão confiável. Caso fosse escolhida a versão mais nova, 2/3 das estruturas disponíveis seriam consideradas inválidas pelo algoritmo de testes.

4 Resultados e discussão

Neste item é descrito em detalhes o código fonte do *software* Hera versão 0.38 e os principais algoritmos desenvolvidos nesta Tese. Na Tabela 4 são apresentados todos os arquivos que formam o código fonte de Hera. A tabela está organizada na ordem em que cada arquivo é carregado na memória quando o arquivo Hera.exe é executado por um usuário.

Tabela 4 - Todos os arquivos do *software* Hera

Nome do arquivo	***Função***
Hera.dpr	O arquivo principal do programa Hera.
Elemento.pas	A definição dos elementos químicos. Do Hidrogênio ao Laurêncio.
UHeraWaveLength.pas	Os comprimentos de onda associadas a cada elemento, Kα1 e Kα2.
UPoint.pas	Define a classe TPonto. Utilizado para localizar um átomo no espaço.
Scater.pas	Tabelas com os fatores de espalhamentos do hidrogênio até o califórnio.
UAtom.pas	Define a classe TAtomo. A definição de átomo. A identificação, localização e o fator de espalhamento.
USpaceGroup.pas	Define classes para os 7 sistemas cristalinos e para os 230 grupos espaciais. Estas classes calculam os átomos simétricos a partir dos átomos assimétricos e dos sítios escolhidos.
UCela.pas	Define a classe para uma cela cristalografia. Aqui estão definidos a, b, c, alfa, beta e gama, a lista de hkls e o grupo espacial.
UHeraConstraintsTeta.pas	Uma janela gráfica para o usuário escolher as condições de contorno que serão utilizadas na busca. Na versão 0.38 esta rotina foi desligada.

UHeraConstraintsElement.pas	As restrições de busca relativas aos elementos químicos.
UHeraConstraintsCell.pas	As restrições relativas aos parâmetros da cela, a ,b , c, alfa, beta, gama e o grupo espacial.
UHeraDictionaryForm.pas	Uma janela que mostra quais dicionários estão ativos. Esta biblioteca é responsável pelos testes dos CIFs recebidos na rede P2PCIF.
XRDGraph.pas	A biblioteca responsável por mostrar um gráfico de Rietveld, um difratograma.
UHeraTokens.pas	Analisa os campos de um arquivo CIF ou dicionário.
UHeraNovoPeer.pas	Insere um novo nó, cliente, na lista de IPs válidos .
UHeraMensagem.pas	Mostra com maiores detalhes o quê está ocorrendo na rede e em outras partes do programa.
ServidorRedeP2PCIF.pas	Define a classe TServidorRedeP2PCIF. Este componente do software é o responsável por enviar arquivos CIF a outras cópias do programa. Junto com a classe TClienteRedeP2PCIF forma o banco de dados distribuído, a rede P2PCIF.
ClienteRedeP2PCIF.pas	Define a classe TClienteRedeP2PCIF. Esta é responsável por receber os arquivos CIF. Junto com a classe TServidorRedeP2PCIF forma o banco de dados distribuídos, a rede P2PCIF.
UEsolherPasta.pas	Uma ferramenta para o usuário incluir grande quantidades de arquivos CIF.
StarFile.pas	A base para leitura de arquivo CIF. Aqui estão definidas como as variáveis do arquivo ficarão na memória do computador e como serão acessadas pelas outras partes do programa.
md5.pas	Rotinas responsáveis por calcular as assinaturas digitais.
ConstRedeP2PCIF.pas	Constantes e a versão do *software* ficam aqui.

UHeraSobre.pas	A janela sobre (about).
Uhkl.pas	Uma classe que define cada um dos planos hkl e listas de hkls.
Hera.pas	A janela principal do *software* Hera. Esta janela é a primeira imagem que um usuário visualiza do *software* mas é a última parte a ser executada do programa.

No arquivo Elemento.pas são definidos os elementos químicos. No código fonte isso ocorre no tipo TELEMENTO. Aqui estão implementadas as rotinas básicas com a conversão do número atômico para o símbolo do elemento e vice versa. O arquivo UHeraWaveLength.pas completa as informações de TELEMENTO. Neste arquivo estão os comprimentos de onda Kα1 e Kα2 associados a cada elemento.

O arquivo UPoint.pas define uma classe importante do programa. A posição no espaço de um ponto, a classe TPONTO. Este tipo abstrato de dado será utilizado quando átomos forem definidos. Basicamente ele armazena a posição x,y,z de um átomo. Para economizar tempo de processamento e memória do computador os átomos gerados por operações de simetria a partir dos átomos assimétricos da cela cristalográfica são criados dentro desta classe. Eles ficam dentro de uma lista chamada CLONE. Esta lista apenas armazena as posições atômicas. As outras propriedades serão as mesmas do átomo da cela assimétrica que o gerou. Outra característica importante é a manutenção da descendência dos átomos.

Um átomo passa a existir no *software* utilizando a classe TATOMO do arquivo UAtom.pas. As propriedades disponíveis são a posição atômica, o número atômico, o símbolo do elemento químico e o sítio em que o átomo está dentro da cela cristalográfica, o fator de espalhamento do elemento e o número de átomos que foram gerados a partir deste por operações de simetria. A classe TATOMO só pode ser utilizada em átomos assimétricos.

Nos arquivos USpaceGroup.pas e UCela.pas estão implementadas as rotinas mais complexas do programa. Em USpaceGroup.pas estão definidos os sete sistemas cristalinos e os 230 grupos espaciais, incluindo as variações de cela, origem e eixo. A classe TBASESG implementa estas características.

Em TBASESG pode ser verificada se a indicação do grupo espacial é valida, se o átomo esta na unidade assimétrica e procurar a qual sítio um ponto no espaço

está associado. A partir do grupo espacial esta classe identifica se um átomo está em uma posição especial permitindo ou não que os valores de x,y ou z sejam alterados. Por exemplo, no grupo espacial **F m 3 m**, caso haja um átomo no sitio ***a*** ele deve obrigatoriamente ficar na posição 0,0,0. Nestax posições especiais os valores de x,y e z não podem ser alterados. A classe TBASESG realiza estas operações e impede que as outras partes do *software* alterem os valores de x,y ou z. Esta classe também é responsável por calcular as coordenadas dos átomos gerados pelas operações de simetria. No exemplo, para o átomo do sítio ***a*** esta classe gera as posições 1,0,0; 0,1,0; 0,0,1; 1,1,0; 0,1,1; 1,0,1; 1,1,1; 0,1/2,1/2; 1,1/2,1/2; 1/2,0,1/2; 1/2,1,1/2; 1/2,1/2,0 e 1/2,1/2,1. A última característica desta classe é verificar se um plano hkl é permitido ou não, isso é feito pela rotina ValidReflection. TBASESG é uma classe ancestral, todos os 230 grupos espaciais são descendentes desta classe. Para acessar estas classes descendentes o *software* deve utilizar a classe TCELA do arquivo UCela.pas.

TCELA implementa uma cela cristalográfica, o grupo espacial, as dimensões e os átomos da unidade assimétrica. Escolhido o grupo espacial, a, b, c, alfa, beta e gama a classe calcula o volume da cela e os planos hkl. O intervalo dos planos é controlado pelas constantes __hmin, __hmax, __kmin, __kmax, __lmin e __lmax. Na versão 0.38 é utilizado o intervalo de -10 até 10 para h,k e l. Ao escolher o grupo espacial TCELA, este se ajusta escolhendo a classe TBASESG apropriada. TBASESG elimina os hkls que são proibidos para a simetria escolhida. Os planos hkls são uma lista ordenada de classes THKL. A classe THKL foi implementada no arquivo Uhkl.pas. Esta classe possui mais duas propriedades d, a distância interplanar e I a intensidade difratada. TCELA calcula d utilizando a formula da Equação 3 (CULLITY, 1978) e a intensidade I a Equação 4 (TSANG, 2001; AUTHIER, 1996).

$$\frac{1}{d^2} = \frac{(S_{11})h^2 + (S_{22})k^2 + (S_{33})l^2 + 2(S_{12})hk + 2(S_{23})kl + 2(S_{13})hl}{\left(abc\sqrt{1-\cos^2\alpha - \cos^2\beta - \cos^2\delta + 2\cos\alpha\cos\beta\cos\delta}\right)^2}$$

$$S_{11} = b^2c^2 sen\alpha$$
$$S_{22} = a^2c^2 sen\beta$$
$$S_{33} = a^2b^2 sen\delta$$
$$S_{12} = abc^2(\cos\alpha\cos\beta - \cos\delta)$$
$$S_{23} = a^2bc(\cos\beta\cos\delta - \cos\alpha)$$
$$S_{13} = ab^2c(\cos\delta\cos\alpha - \cos\beta)$$

Equação 3 - Cálculo de d a partir dos planos hkl.

$$I_{hkl} = \left(\left(\sum_i m_i\left(\left(f_i^{0} + f_i^{'}\right)\cos\phi - f_i^{''} sen\phi\right)\right)^2 + \left(\sum_i m_i\left(\left(f_i^{0} + f_i^{'}\right)sen\phi + f_i^{''}\cos\phi\right)\right)^2\right)\left(\frac{1+\cos^2(2\theta_{hkl})}{sen^2\theta_{hkl}\cos\theta_{hkl}}\right)$$

$$\phi = 2\pi(hx_i + ky_i + lz_i)$$

Equação 4 - Cálculo da intensidade dos planos.

Na Equação 4 as somatórias são feitas nos átomos da cela e f são os fatores de espalhamento de cada átomo, a,b,c, alfa, beta e gama são os parâmetros da cela e h,k e l são os índices de Miller para cada plano. Os fatores de espalhamento estão implementados em Scater.pas. F' e f'' são constantes lidas da tabela _DispersionCorrectionsScattering e f^0 é calculado utilizando a Equação 5 com as constantes a e b lidas da tabela _CoefScatt, teta é o ângulo medido e lambda o comprimento de onda da luz utilizqada na medida.

$$f^0 = \sum_{i=1}^{4} a_i e^{-b_i \left(\frac{sen\theta}{\lambda} \right)^2}$$

Equação 5 - Fator de espalhamento.

UHeraConstraintsTeta.pas, UHeraConstraintsElement.pas e UHeraConstraintsCell.pas implementam as regras para que a busca seja feita. A busca no programa Hera é feitos lendo todos os arquivos CIF armazenados na pasta c:\hera\cif, calculando os planos hkls e as intensidades. Para tornar o processo mais rápido o usuário de escolher limites para esta busca. Basicamente este deve saber quais elementos químicos devem estar presentes nos arquivos CIFs e qual a simetria. O usuário pode escolher especificamente um grupo espacial ou um dos sete sistemas cristalinos. As interfaces gráficas destas rotinas são apresentadas na Figura 51 e Figura 52.

UHeraDictionaryForm.pas implementa classes associadas aos dicionários CIF, listados na Tabela 3. Todos os arquivos presentes na pasta c:\hera\dic\ são considerados dicionários e serão lidos quando o programa for iniciado. Caso o usuário deseje restringir os CIF que irão compor o banco de dados local, o usuário deve apagar o dicionário correspondente. Por exemplo, se um usuário não deseja manter CIFs relacionados com proteínas basta apagar o arquivo mmCIF.dic. Os campos relacionados com cristalografia de proteínas serão considerados inválidos nos teste e este arquivo CIF será ignorado em uma inclusão ou descartado na troca pela rede P2P. A interface gráfica é mostrada na Figura 49.

O arquivo XRDGraph.pas implementa a capacidade de apresentar histogramas. Estas são simples rotinas que mostram um gráfico X Y foram implementada pela classe TTACHART e a classe TCURVA. A classe TCURVA armazena os pontos x e y e a cor de cada ponto. A classe TTACHART foi construída para mostrar graficamente uma lista de curvas TCURVA. Estas classes são responsáveis pelos gráficos visualizados pelo usuário quando um histograma é lido. A Figura 53 mostra a imagem gerada.

A biblioteca md5.pas é responsável por calcular as assinaturas digitais e os nomes dos arquivos CIF armazenados no banco de dados local. MD5 é um algoritmo hash ou algoritmo de dispersão (RIVEST, 1991); que permite calcular uma

assinatura digital de uma sequência de bytes. A seqüência de bytes pode ser formada por algumas letras ou por um enorme arquivo com bilhões de bytes. A assinatura MD5 é uma função onde o conjunto domínio é toda a seqüência de bytes sob o qual a assinatura será gerada. A assinatura digital é gerada lendo seqüência de 7 em 7 bytes e aplicando a Equação 6. A somatória de todo o arquivo ou seqüência de bytes gera um número que é apresentado na forma hexadecimal com 32 letras. Estas 32 letras são a assinatura digital desejada. Por exemplo, a palavra labcacc gera a assinatura A01CE0D9499FDAABEF301B60416C2484. Mesmo alterando um único bit na seqüência a assinatura gerada será diferente, ou seja, Labcacc gera 28D90C70F9C9DABE42C9C5C811119B6F.

$$h = \sum b_1 + \left(b_4 xor\left(b_2 and\left(b_3 xor b_4\right)\right)\right) + \left(b_7 xor\left(b_5 and\left(b_6 xor b_7\right)\right)\right)$$

Equação 6 - Função hash MD5.

As assinaturas MD5 são utilizadas na transferência de arquivo na rede P2P para verificar se o arquivo enviado é idêntico ao arquivo recebido.

Um uso crítico deste algoritmo no *software* Hera é a nomeação dos arquivos CIF. Quando um arquivo CIF é incluído no banco de dados o seu nome original é abandonado. Um novo nome é gerado utilizando as assinaturas MD5. Esta abordagem é necessária para evitar a duplicação de arquivos no banco de dados local. Mas há um problema, Hera deve ser capaz de atribuir um nome único para CIFs que ainda não foram criados, ou seja, o algoritmo deve ser capaz de gerar um nome para estruturas que ainda nem foram determinadas e que só serão determinadas no futuro.

Para fazer isso o Hera monta um texto somando os campos _symmetry_space_group_name_H_M, _cell_length_a, _cell_length_b, _cell_length_c, _cell_angle_alpha, _cell_angle_beta, _cell_angle_gamma, _atom_site_type_symbol, _atom_site_fract_x, _atom_site_fract_y e _atom_site_fract_z. Para o NaCl, o texto resultante será “F m 3 m5.4535.4535.4539090902 Na Cl2 0.000 0.500 0.000 0.500 0.000 0.500” e a assinatura MD5 calculada será A0D47511AA13B462AD52BCA62E3AC5F3. Este será o nome do arquivo armazenado no banco de dados. O algoritmo ignora valores

após a terceira cada decimal, 0.500 e 0.5001 são iguais e serão utilizados como 0.500. Este recurso minimiza a duplicação de arquivos refinados utilizando o método de Rietveld. Caso isso não fosse utilizado uma mesma cela cristalografia seria duplicada dezenas de vezes na HD do usuário, a medida que cada cela fosse ligeiramente diferente uma da outra.

Em StarFile.pas estão definidas as classes e rotinas para ler os arquivos CIF e DIC, ambos são descritos no capítulo 4.1. A rede que utiliza o banco de dados distribuído é gerada e mantida pelas classes dos arquivos ClienteRedeP2PCIF.pas e ServidorRedeP2PCIF.pas. O funcionamento da rede P2P utilizada pelo *software* Hera é discutido no capítulo 4.2.

4.1 Arquivo CIF e DIC

O algoritmo criado para ler arquivos no formato CIF é apresentado na Figura 18. O mesmo algoritmo é utilizado para ler os dicionários, denominados arquivos DIC. O formato deste arquivo segue o padrão STAR. Os arquivos textos no formato STAR lidos sob as regras dos dicionários formam o padrão de arquivos cristalográficos CIF. Note que os arquivos DIC também estão submetidos às regras de outros dicionários.

O algoritmo verifica primeiro se o arquivo existe no disco rígido, HD, e se não está vazio. Estes são os primeiros erros possíveis. Caso o arquivo seja considerado válido são iniciadas as variáveis que armazenam as condições do processo. O algoritmo lê letra por letra do arquivo seguindo a sintaxe da Tabela 2, de tal forma que as instruções condicionais não são utilizadas na implementação. Quando são encontradas as letras predefinidas ou palavras reservadas o algoritmo envia estes dados para outros algoritmos menores e mais simples, os algoritmos da Figura 19 à Figura 31. Estes estão definidos como: <EOL>, <WhiteShape>, <sharp>, <dquote>, <squote>, <semicolon>, <underscore>, <leftbracket>, <rightbracket>, <shriek>, <dollar>, <ampersand> e <ordinarychar>. Na versão 1.1 alguns são muito semelhantes e poderiam ser utilizadas as mesmas rotinas para vários dos caracteres especiais, mas a partir da versão 2.0 estes campos foram ser tratados separadamente.

Basicamente, estas rotinas separam comentários, textos, nomes de variáveis e valores de variáveis. Os comentários são iniciados com a letra # mais espaço precedidos de <EOL> ou <WhiteShape>. O textos podem ter seqüências simples de letras entre aspas duplas ou simples e seqüências de linhas entre pontos e vírgulas. A variável comments marca se um comentário está sendo lido, a variável text se um texto está sendo lido e a variável *token* se um comando ou identificação estão montados. A variável LSU armazena o último *token* processado e PU a última letra.

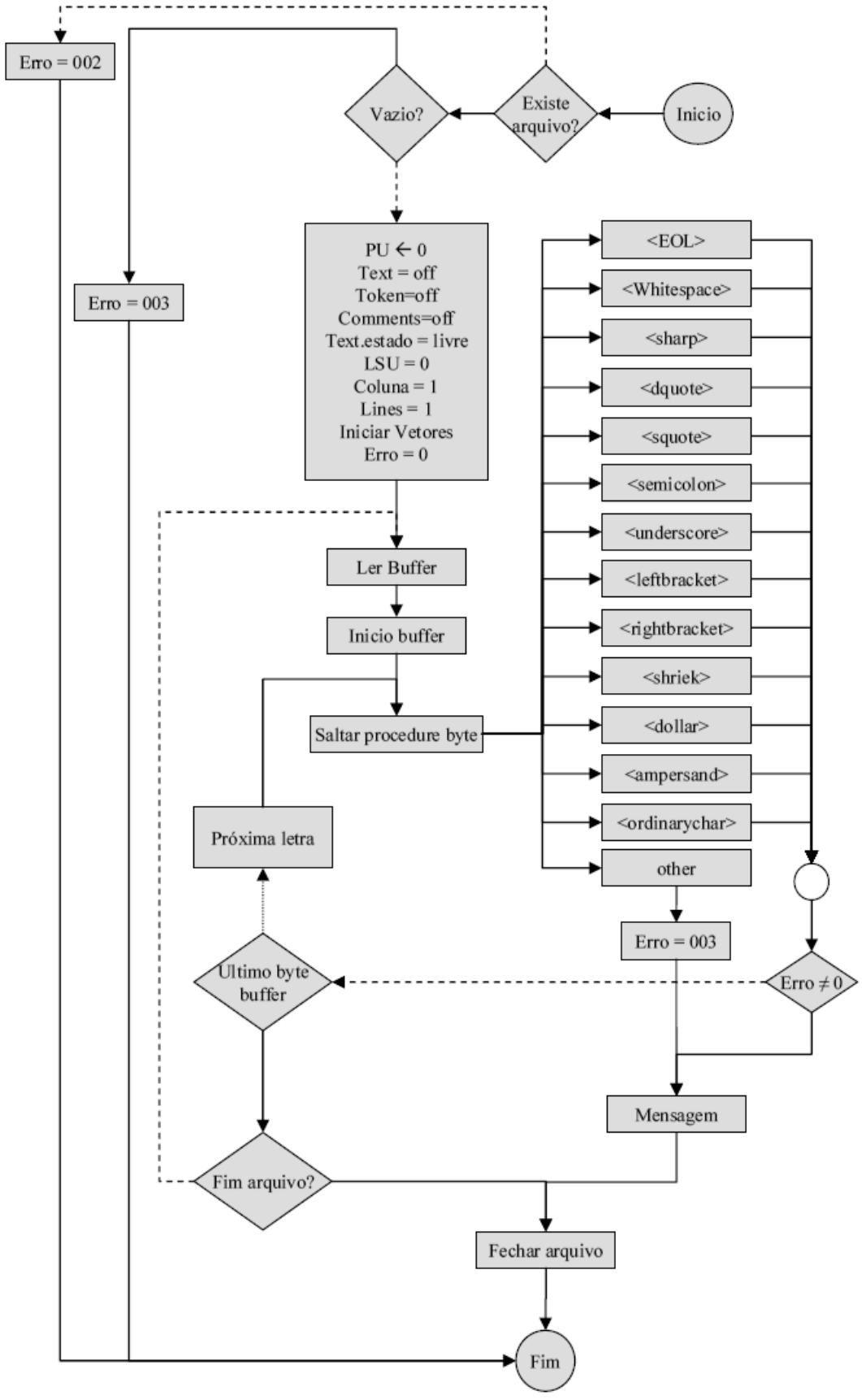

Figura 18 - Algoritmo para arquivos *STAR*.

A rotina <EOF> ocorre quando uma linha é finalizada. Os caracteres ASCII 10 e 13 marcam esta condição no sistema operacional Windows. Em outros sistemas operacionais são utilizados caracteres diferentes. O algoritmo é mostrado na Figura 19. Além de marcar o final de uma linha ele indica quando um comentário ou uma palavra terminam. Nesta condição, o algoritmo para processar *tokens*, Figura 33, é chamado.

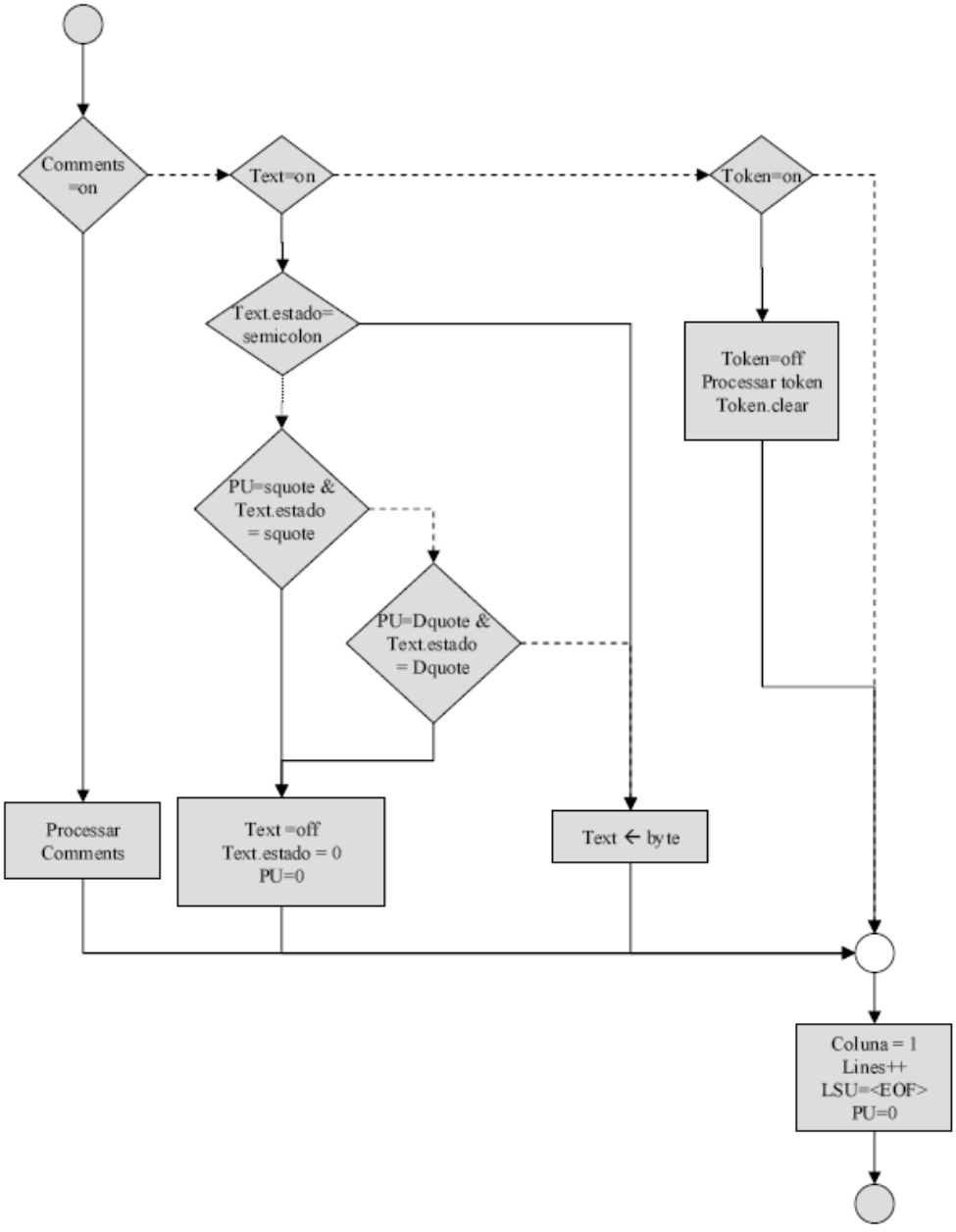

Figura 19 - Algoritmo para *STAR file* <EOF>.

Outro marcador importante é o espaço em branco, identificado como <whitespace>. O algoritmo é mostrado na Figura 20. Caso um comentário ou um texto esteja sendo lido, o espaço em branco é incluído no texto. Se um espaço em branco ocorrer fora destas condições, o algoritmo para processar *tokens* é chamado.

O algoritmo da Figura 21 processa o caracter #, <sharp>. Este é responsável por iniciar um comentário.

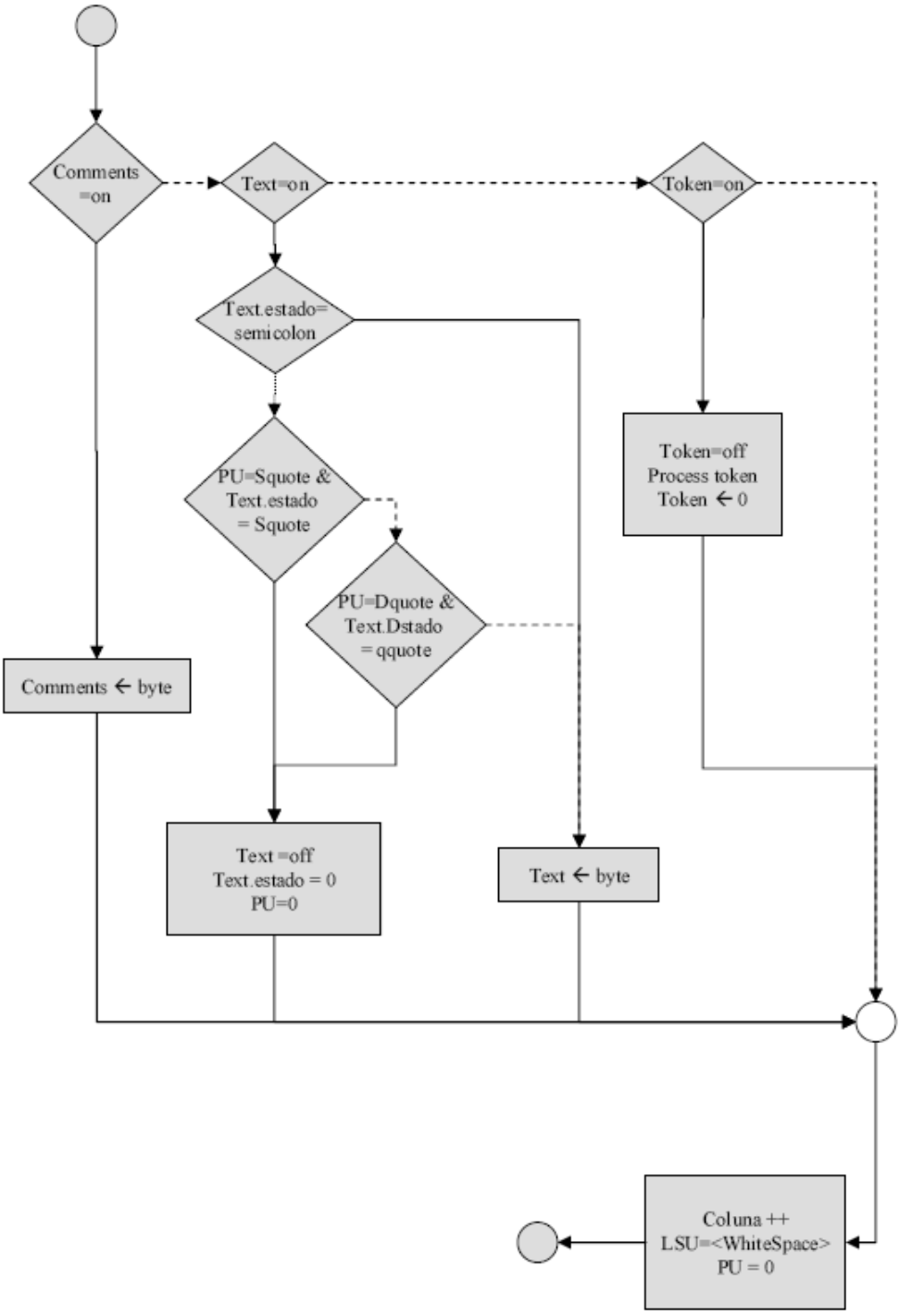

Figura 20 - Algoritmo para *STAR file* <whitespace>.

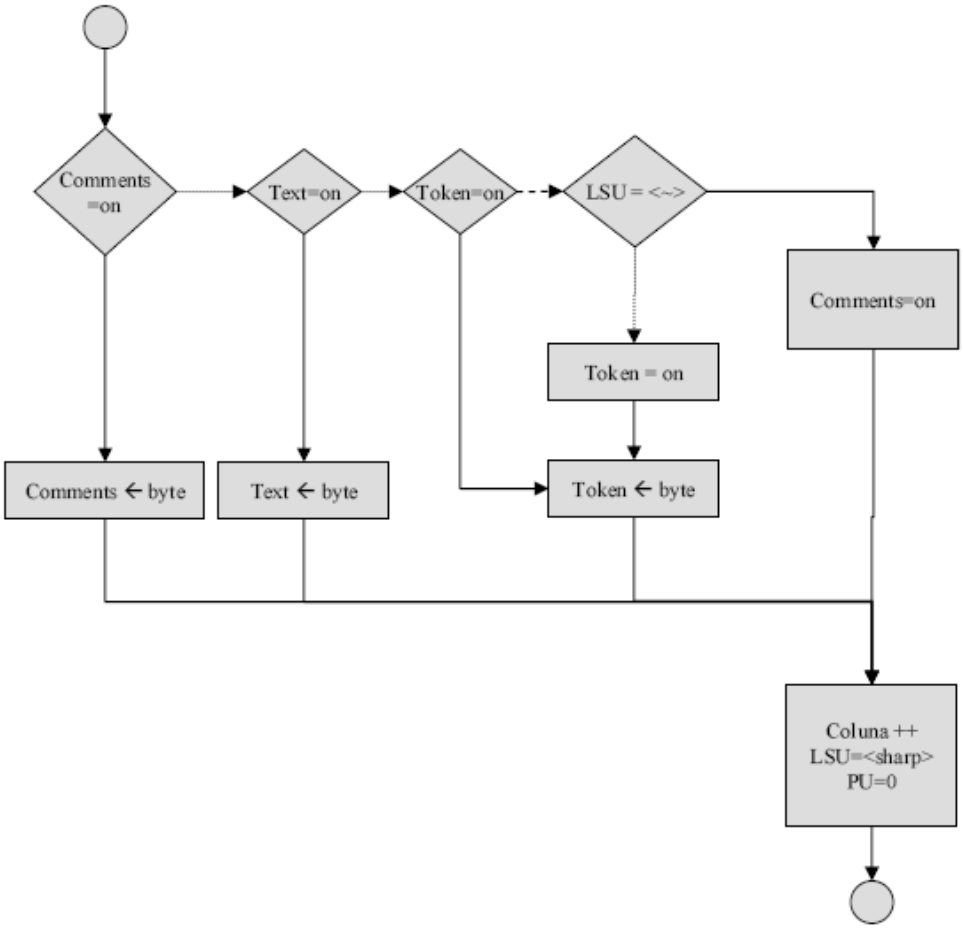

Figura 21 - Algoritmo para *STAR file* <sharp>.

Os algoritmos da Figura 22 e Figura 23 são quase idênticos, interpretam aspas e aspas duplas. As aspas fora de um comentário delimitam um texto. Para o algoritmo as seqüências “texto entre aspas duplas” e ‘texto entre aspas simples’ não são formadas por quatro palavras mas uma única palavra que foi iniciada e finalizada com aspas.

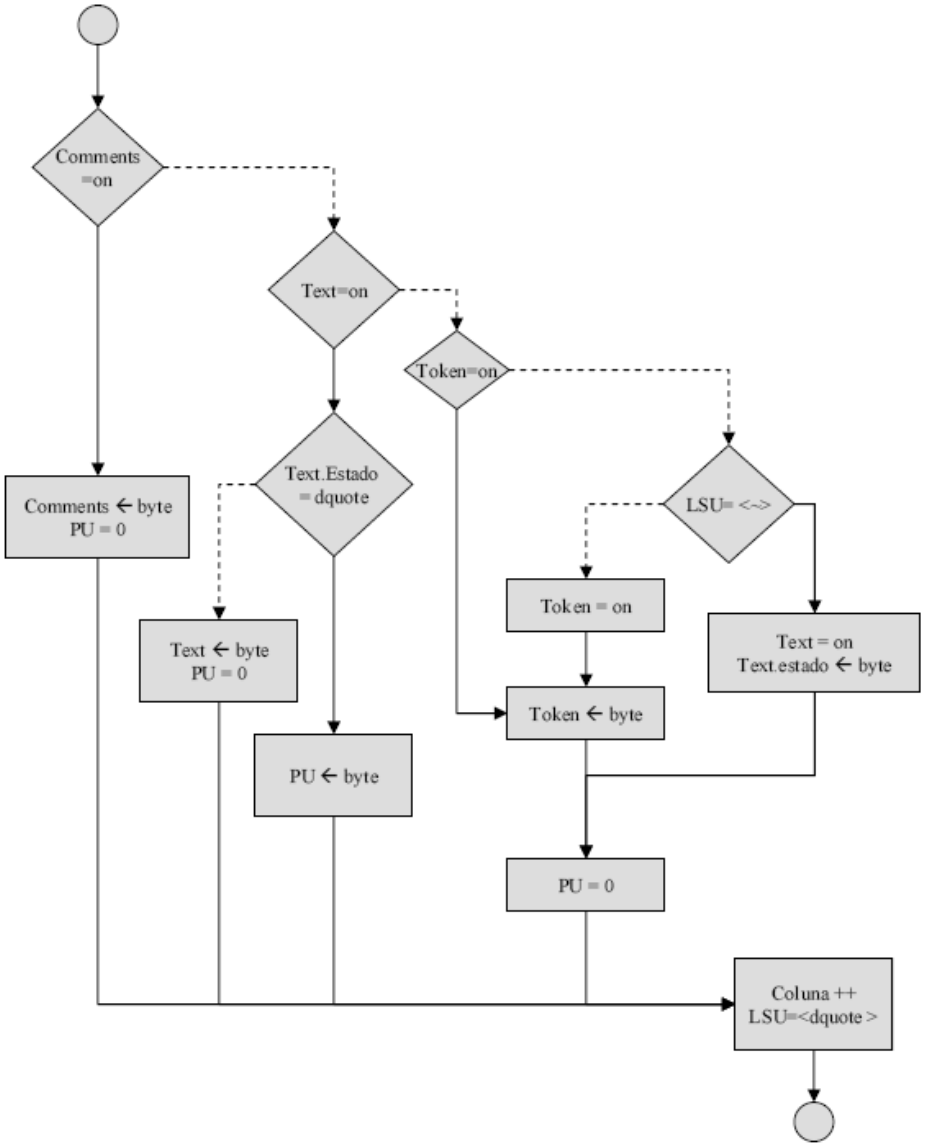

Figura 22 - Algoritmo para *STAR file* <dquote>.

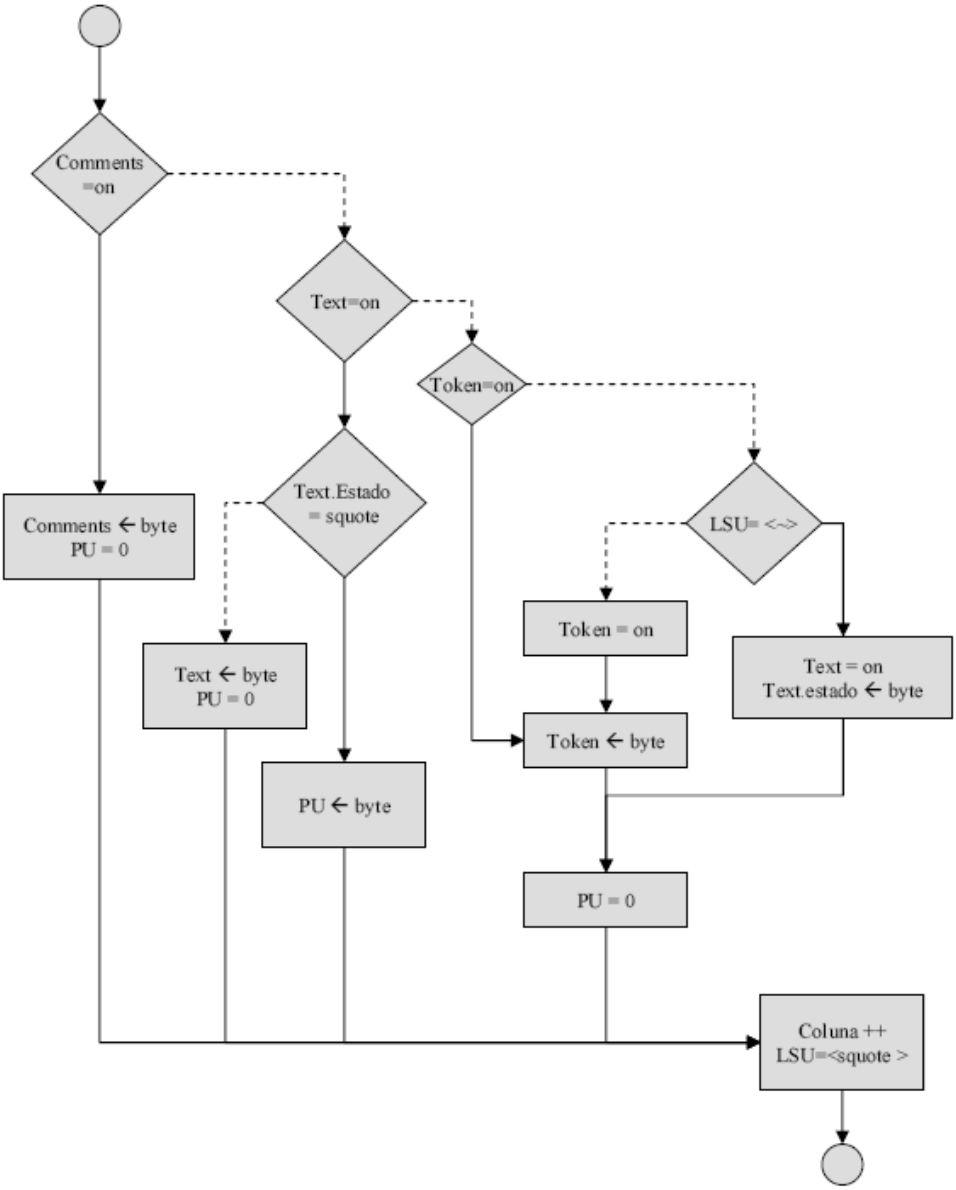

Figura 23 - Algoritmo para *STAR file* <squote>.

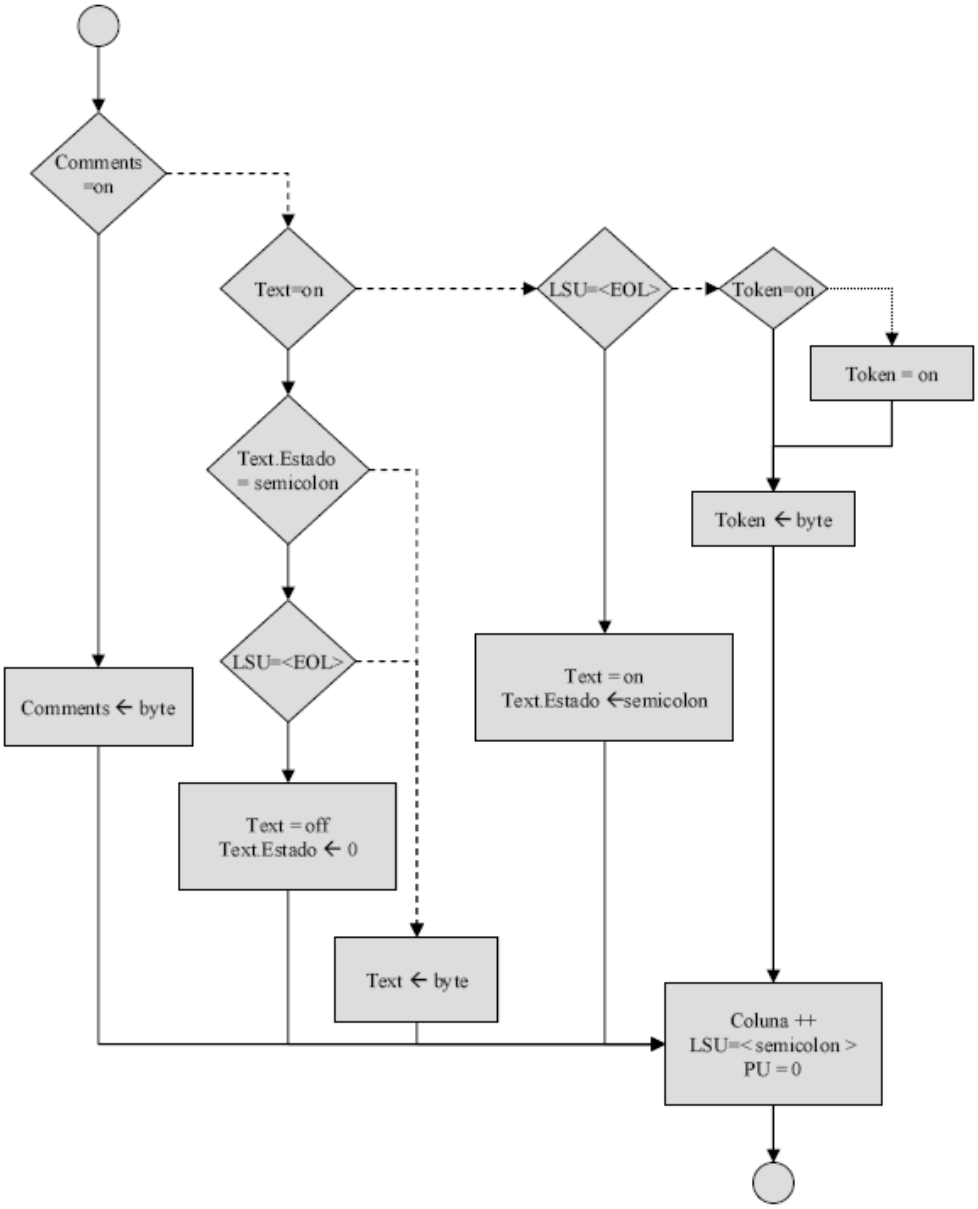

Figura 24 - Algoritmo para *STAR file* <semicolon>.

Textos com mais de uma linha são iniciados e finalizados com ponto e vírgula. O algoritmo é mostrado na Figura 24. O algoritmo identifica quando ; ocorre desviando o texto entre ; para uma classe especial. Os caracteres <underscore>, Figura 25, marcam o início do nome de uma variável em um arquivo CIF, por exemplo, _cell_length_a é o nome da variável com o parâmetro da cela cristalográfica.

Os algoritmos <leftbracket> mostrado na Figura 26, <rightbracket> na Figura 27, <shiriek> na Figura 28, <dollar> na Figura 29, <ampersand> na Figura 30 e <ordinarychar> na Figura 31 são idênticos. Apenas diferenciam o caractere atual e o

enviam a um *token*, texto ou comentário. Da tabela ASCII poucas letras são consideradas válidas, caso ocorra um caractere que não esteja em nenhuma destas 13 categorias, a leitura do CIF é encerrada e um aviso de erro é enviado ao usuário. Em 2.500 amostras de CIF aleatórias 4% foram recusadas por este motivo.

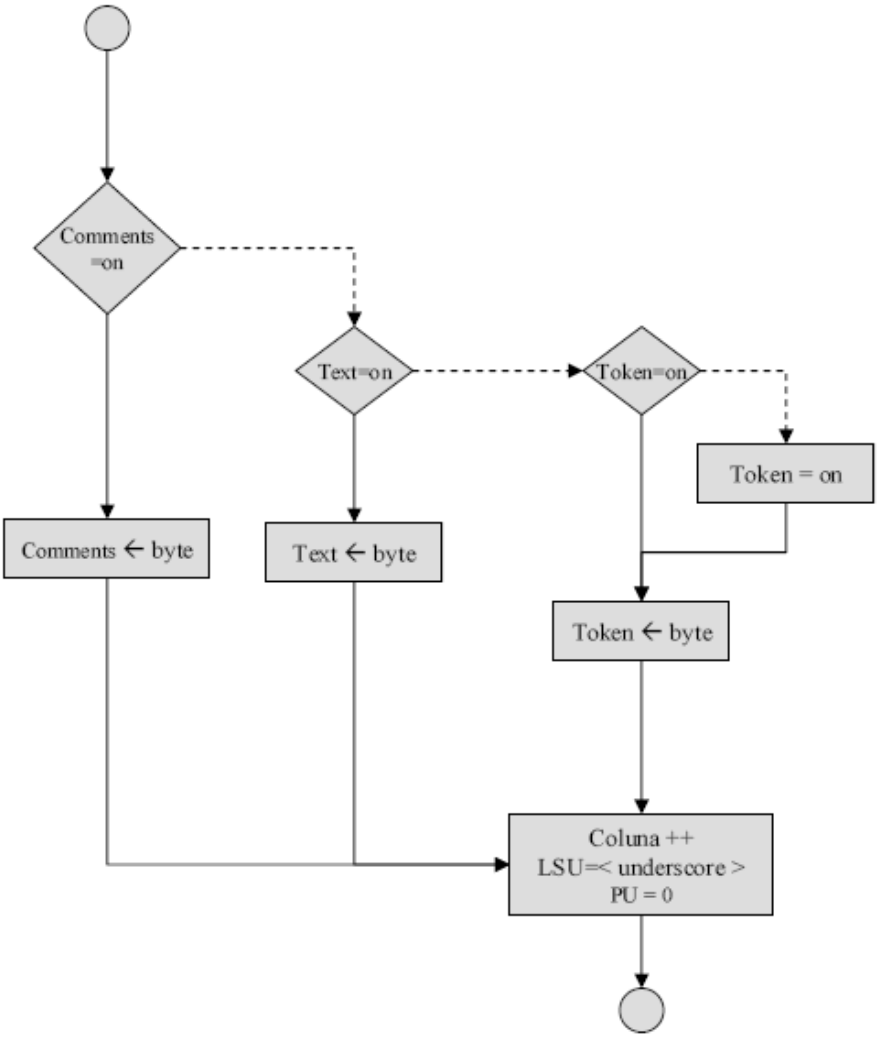

Figura 25 - Algoritmo para *STAR file* <underscore>.

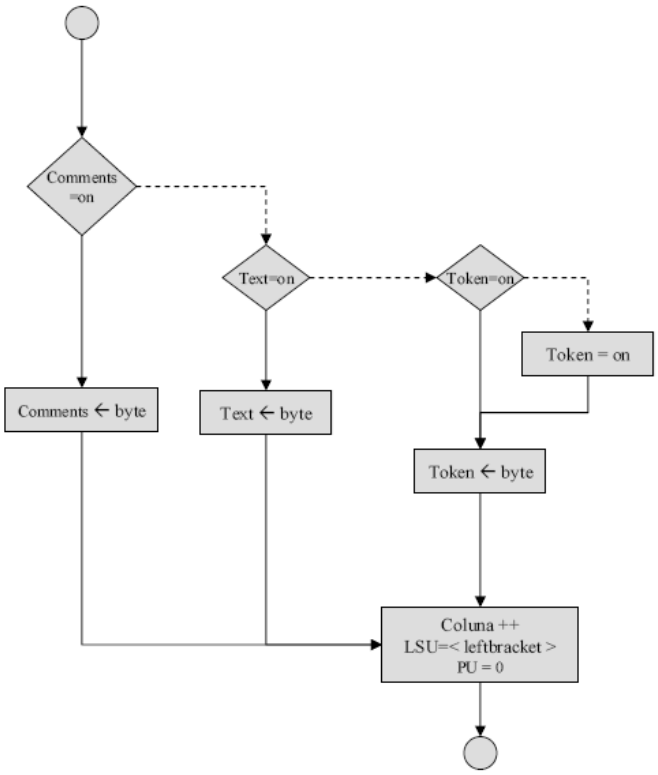

Figura 26 - Algoritmo para *STAR file* <leftbracket >.

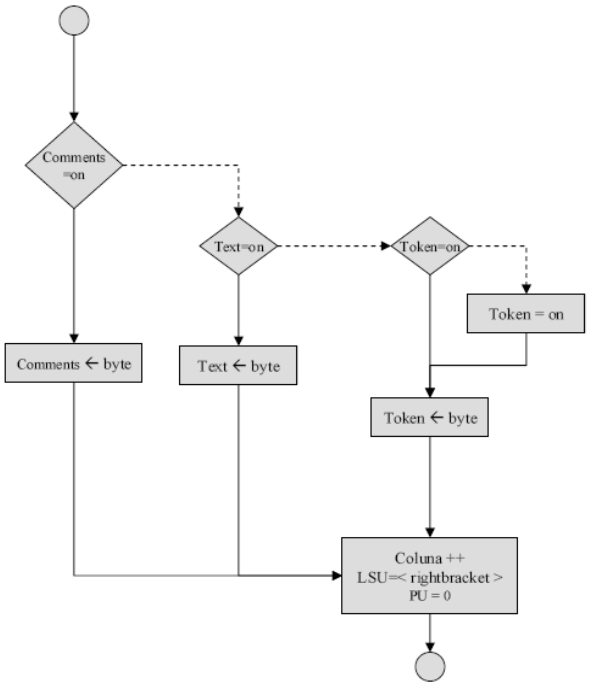

Figura 27 - Algoritmo para *STAR file* <rightbracket>.

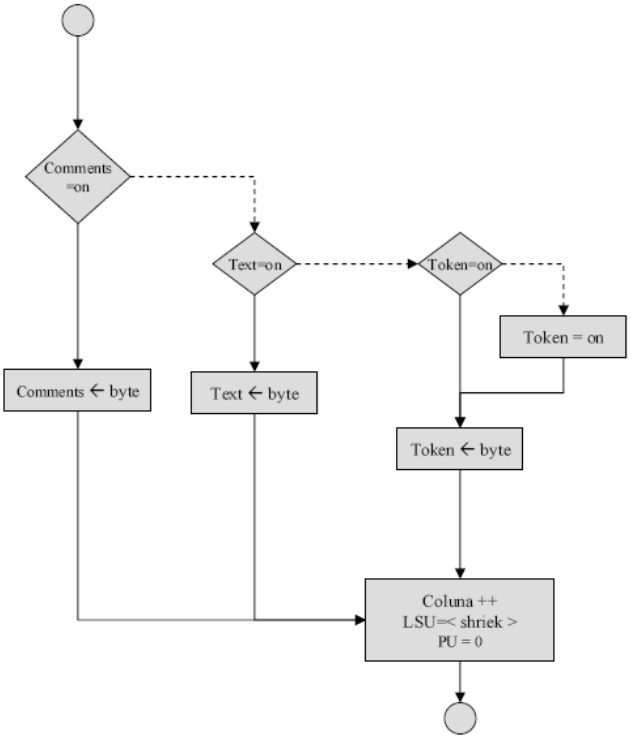

Figura 28 - Algoritmo para *STAR file* <shiek>.

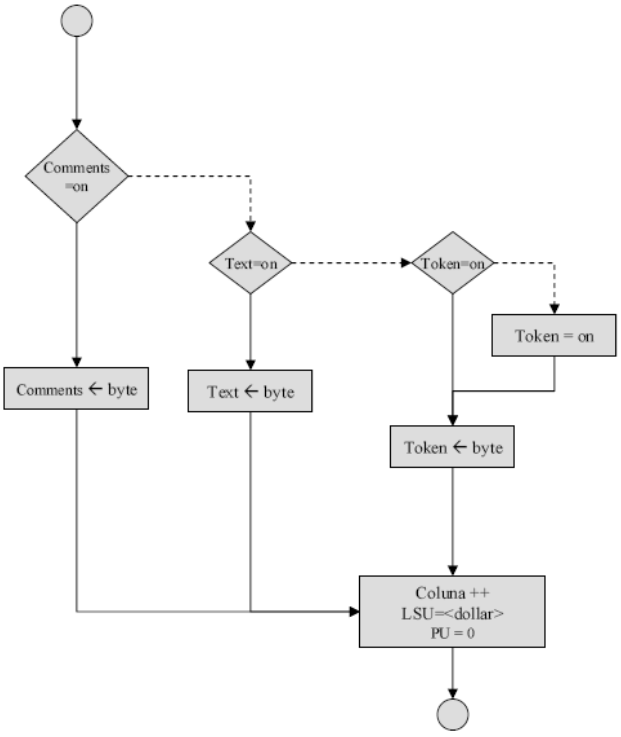

Figura 29 - Algoritmo para *STAR file* <dollar>.

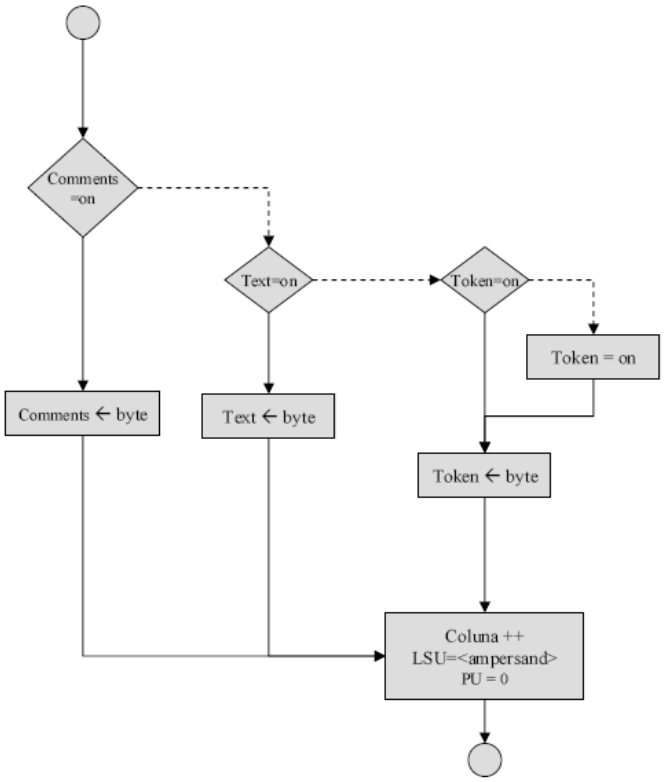

Figura 30 - Algoritmo para *STAR file* <ampersand>.

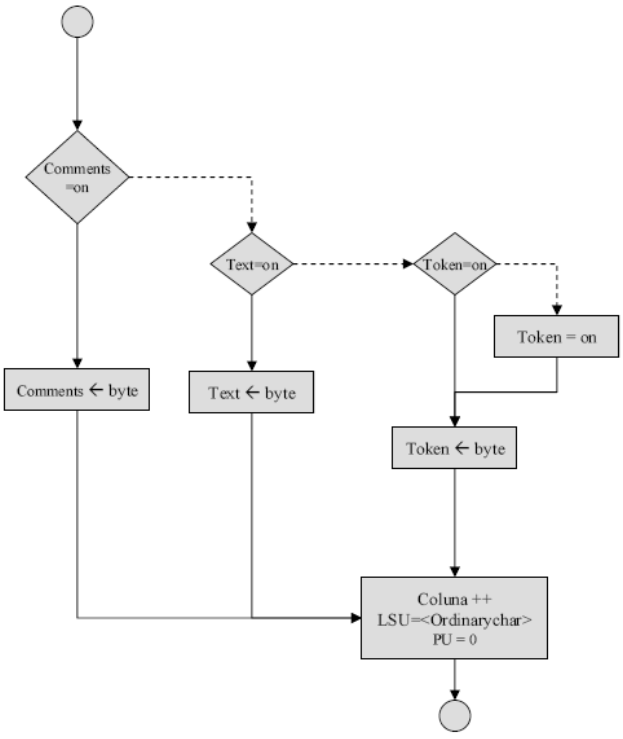

Figura 31 - Algoritmo para *STAR file* <ordinarychar>.

Depois que a letra do arquivo é processada, ela forma uma seqüência de caracteres designados *token* que são enviados ao algoritmo da Figura 33. Um *token* pode ser uma palavra, um nome, um texto entre aspas simples, um texto entre aspas duplas ou um número.

Na Figura 32 é apresentado o algoritmo para processar arquivos CIF com a regra de sintaxe 2.0. O aplicativo Hera possui a capacidade de ler ambos. Na versão 0.38 é dada ênfase ao padrão 1.1.

A função destes algoritmos é ler os nomes de variáveis e os valores associados a estas variáveis. O algoritmo identifica se um *token* enviado é do tipo <datablock>, <dataitems>, <saveframe>, <loopheader>, <data_>, <loop_>, <global_>, <save_> ou <stop_> e se a ocorrência relativa aos outros *tokens* está correta. Pela regras de sintaxe a seqüência em que estes *tokens* ocorrem é predefinida. As regras foram listadas na Tabela 2. Se o *tokens* não for associado a nenhuma das categorias anteriores é considerado um nome de variável ou valor de variável. A diferenciação é feita pelo caractere <underscore>. Neste algoritmo fica mais claro a função da palavra chave <loop_>. Ela permite que uma variável possua mais de um valor associado a mesma. Na sintaxe de um arquivo CIF uma variável é uma lista de valores. As variáveis e seus valores estão definidos no código fonte em classes TDATA definidas no arquivo StarFile.pas. A classe TDATA é um tipo abstrato de dados to tipo árvore alterado (WIRTH, 1989).

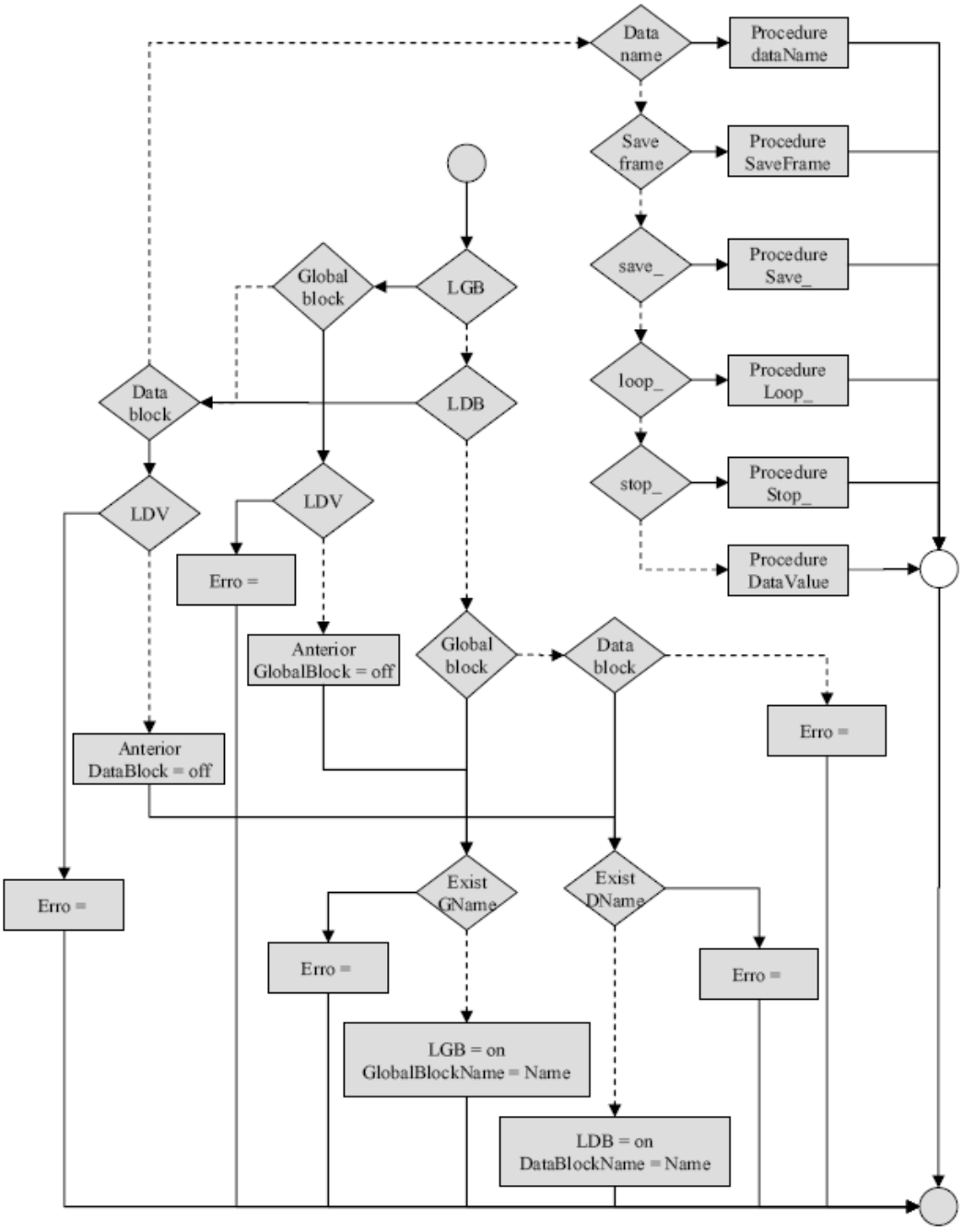

Figura 32 - Algoritmo para a leitura de arquivos CIF versão 2.0.

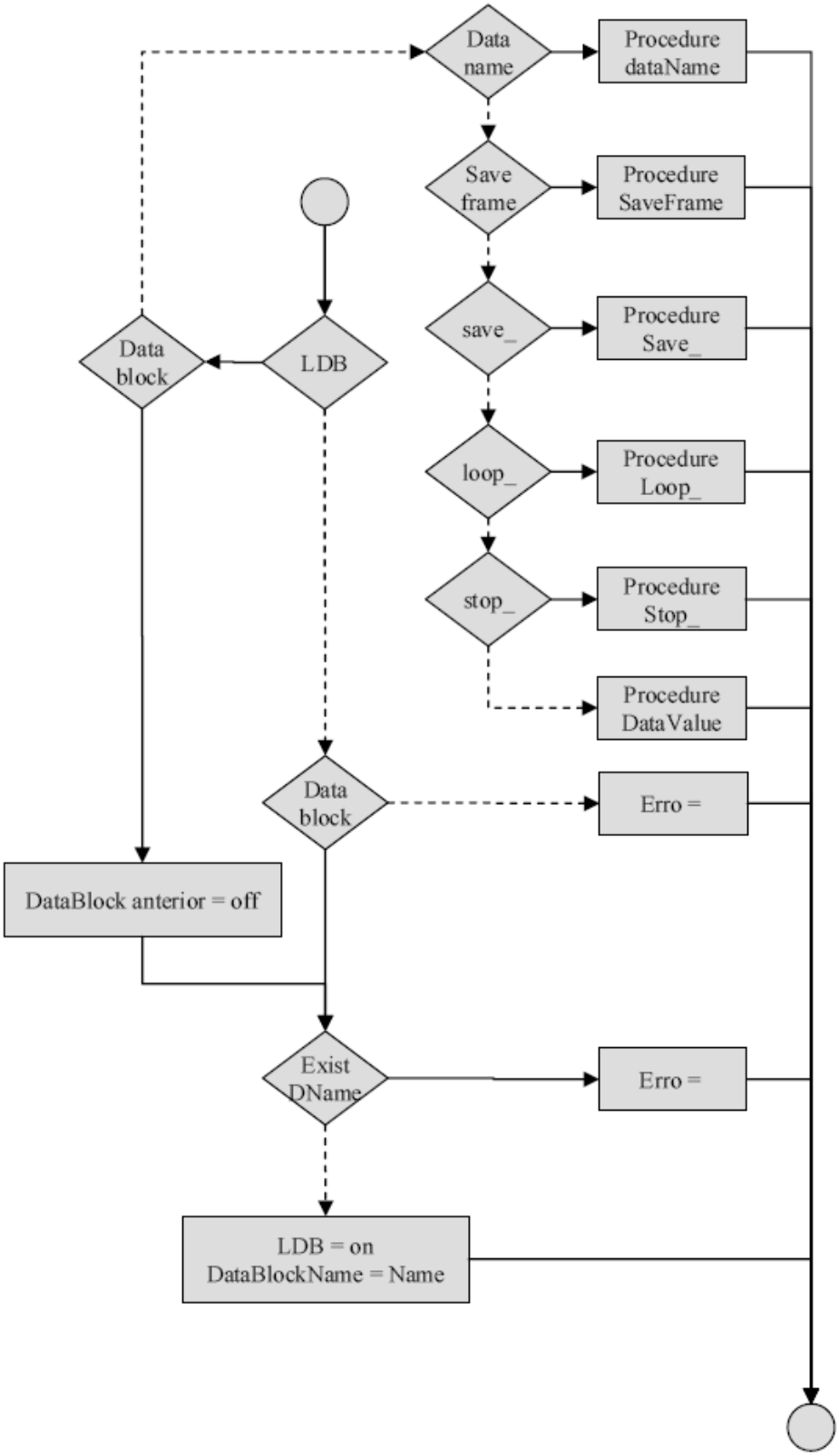

Figura 33 - Algoritmo para a leitura de arquivos CIF versão 1.0.

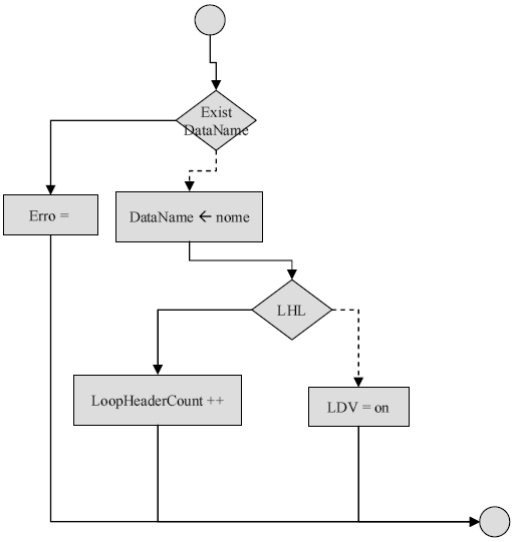

Figura 34 - Algoritmo para a *procedure* dataname.

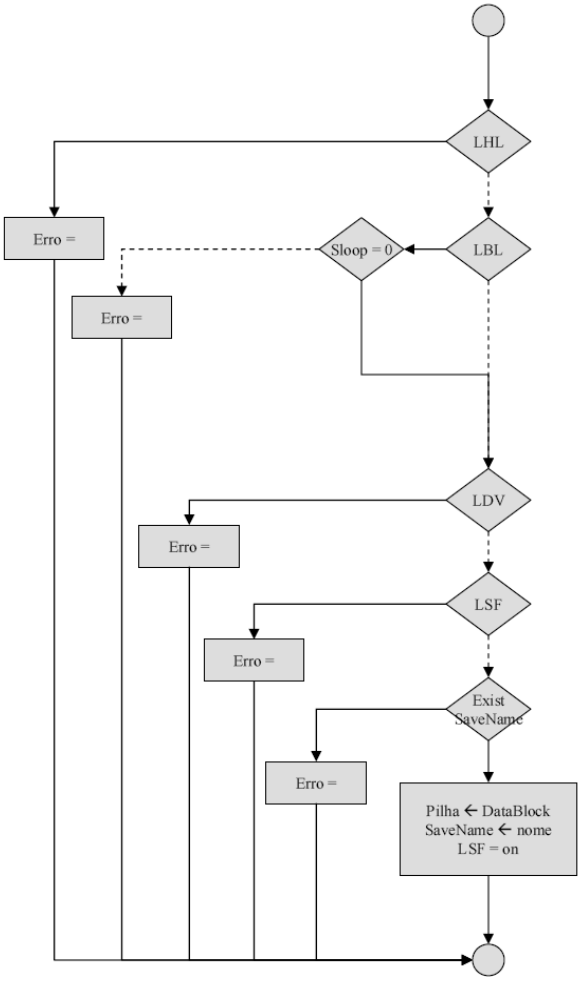

Figura 35 – Algoritmo para a *procedure* saveframe.

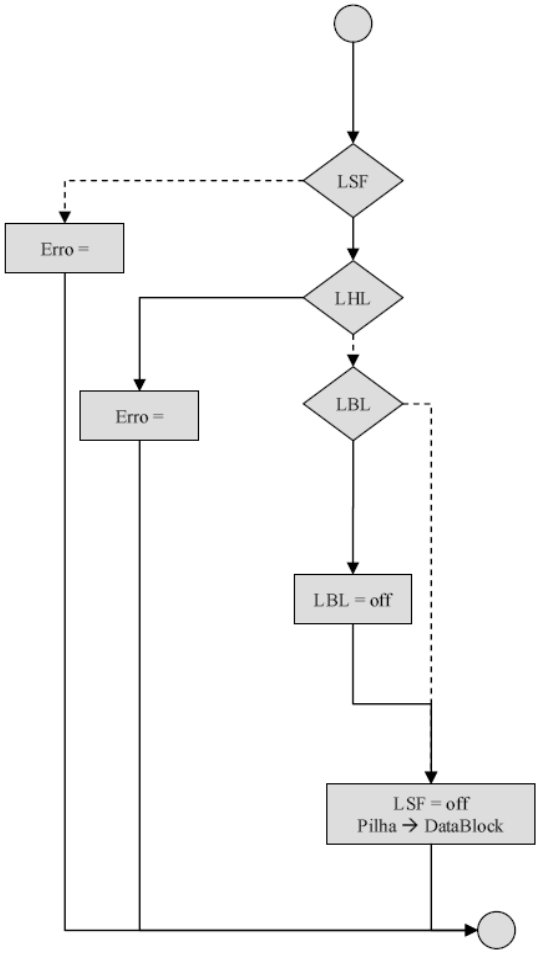

Figura 36 - Algoritmo para a *procedure* datablock.

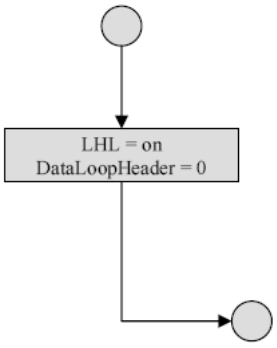

Figura 37 - Algoritmo para a *procedure* loop_.

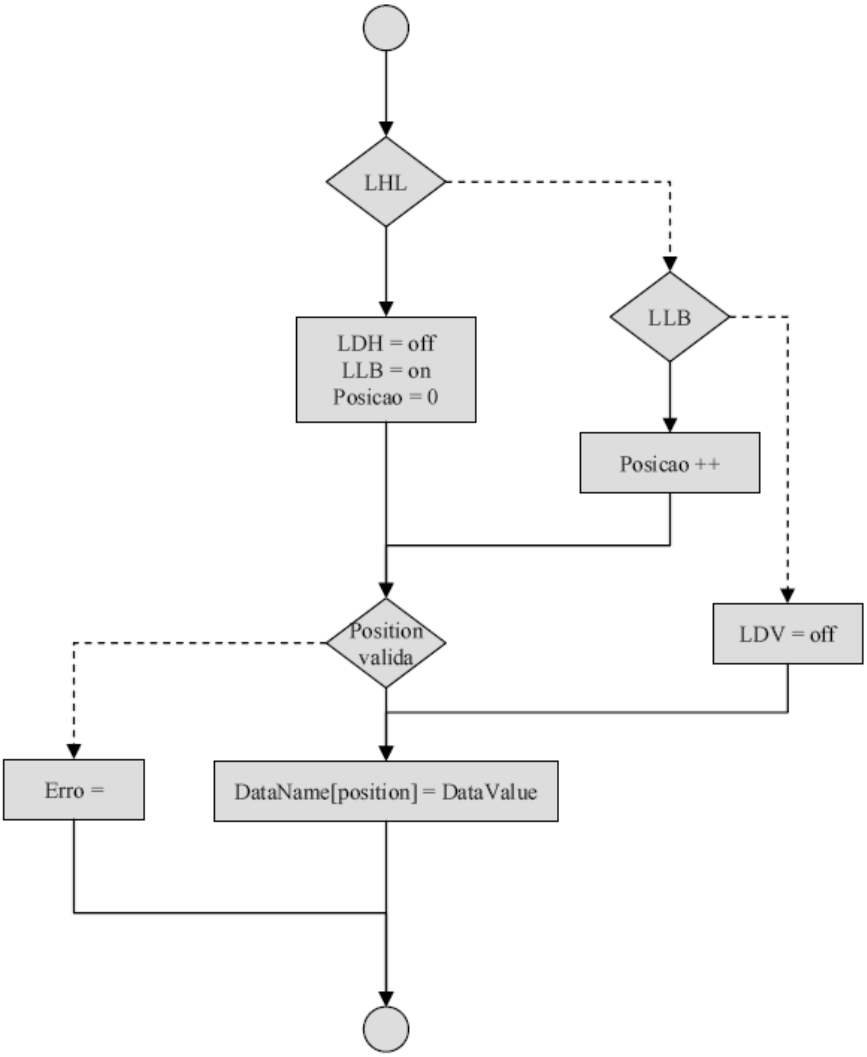

Figura 38 - Algoritmo para a *procedure* datavalue.

Os algoritmos das Figura 18 à Figura 38 estão implementados no arquivo StarFile.pas. Neste arquivo estão definidos os caracteres especiais _blank, _VerticalTab, _HorizontalTab, _NewLines, _FormFeeds, _CarriageReturn, _eol, _sharp, _Dquote, _Squote, _underscore, _SemiColon, _LeftBracket, _RightBracket, _shriek, _dollar e _ampersand. Há cinco classes TVALOR, TDATA, TSTARFILEFRAME, TBASESSTARFILE e TSTARFILE.

A classe TVALOR armazena e manipula os valores atribuídos às variáveis, tanto os valores numéricos, quanto os textos simples ou compostos. Esta classe foi concebida para ser polimórfica. Os diferentes tipos de dados definidos em um arquivo CIF são enviados a esta classe.

Para um CIF exemplo, com a estrutura do NaCl tem-se:

```
_atom_site_label
_atom_site_type_symbol
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_Uiso_or_equiv
Na Na  0.00000   0.00000   0.00000   0.01849
Cl Cl  0.50000   0.50000   0.50000   0.01494
```

Será criada uma variável associada ao campo _atom_site_fract_x. Nesta estrutura haverá dois objetos da classe TVALOR. Um para o valor x do Na, 0.00, e outro para o valor da coordenada x do Cl, ou seja o valor 0.500. Não há limites para a utilização da classe TVALOR. Há apenas os limites físicos do *hardware* do computador, que neste caso é a memória RAM. Caso seja necessário, o *software* pode criar milhares de objetos TVALOR.

A classe TDATA implementa os campos DATA e DATABLOCK de um arquivo CIF, que corresponde a uma variável. Esta classe possui uma lista de TVALOR e uma lista de objetos da TDATA. Esta capacidade recursiva simplifica a implementação de blocos de dados.

Para exemplificar, utiliza-se novamente a estrutura do NaCl.

```
_atom_site_label
_atom_site_type_symbol
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_Uiso_or_equiv
Na Na  0.00000   0.00000   0.00000   0.01849
Cl Cl  0.50000   0.50000   0.50000   0.01494
```

Para cada um dos campos _atom_site_label, _atom_site, _type_symbol, _atom_site_fract_x, _atom_site_fract_y, _atom_site_fract_z, e

_atom_site_Uiso_or_equiv será criado um objeto do tipo TDATA. Os valores serão enviados para estas classes. TDATA armazena os textos com os nomes originais para buscas futuras. A rotina de busca é utilizada nos testes utilizando os dicionários DIC. A classe TSTARFILEFRAME é uma estrutura para <SAVEFRAME> e junto com TBASESTARFILE formam classes de apoio a classe TSTARFILE.

A TSTARFILE é uma classe responsável por todo o processamento dos arquivos CIF, envolvendo a leitura, o processamento e os testes. Ela armazena o nome do arquivo lido, a ocorrência e quais foram os erros ocorridos e toda a informação lida em uma estrutura de dados do tipo árvore. Esta informação lida do arquivo CIF é acessível através da propriedade DataBlock que é um ponteiro a classe TDATA. A TSTARFILE também gerencia toda a memória necessária para que o *software* Hera utilize CIFs e DICs, incluindo pedidos de requisição e liberação ao sistema operacional.

A lista de dicionários fica armazenada na variável LISTADICIONARIOS. Esta é uma lista simples que armazena instâncias da classe TSTARFILE.

4.2 Banco de dados distribuído , a rede P2PCIF

A rede P2PCIF é a capacidade que várias cópias do *software* Hera troquem entre si arquivos CIF. A busca desta capacidade é baseada em um conceito simples. Supondo-se que duas pessoas, A e B, que não se conheçam necessitam utilizar um arquivo CIF e se a utilização ocorrer podem-se fazer três suposições com uma grande probabilidade de acerto. Primeiro que estas pessoas possuam uma formação especializa e com um grau de instrução elevado, incluindo conhecimentos sobre cristalografia. Segundo que obtiveram o arquivo CIF, ou copiaram de uma fonte qualquer ou elas mesmas determinaram a estrutura. A terceira suposição é que um *software* foi utilizado no processo. Neste contexto é possível construir algoritmos que mantenham um banco de dados armazenando arquivos CIF com um alto grau de confiabilidade. Estes algoritmos e suas implementações são identificados como rede P2PCIF.

Antes da rede P2PCIF, as pessoas depois de utilizarem os arquivos CIF os descartariam. Com a rede P2PCIF o usuário A envia o CIF que utilizou para o usuário B e vice versa. Assim, se um usuário C necessita de CIF utilizados por A e B

a rede P2PCIF copia os CIF armazenados nos computadores de A e B para o computador do usuário C.

Quando C utilizar um CIF que nunca tenha sido utilizado nem por A e nem por B, a rede P2PCIF envia este arquivo de C para A e de C para B. Se este processo for repetido por centenas de aplicativos centenas de vezes um enorme banco de dados será construído sem custo adicional e de forma automática.

A implementação da rede P2PCIF apresentada nesta Tese é a fusão de dois *softwares*. O primeiro *software* monta um servidor de internet simples. Este programa fica esperando seqüências de bytes predefinidas em uma das portas TCP/UDP do computador. Ao recebê-las, ela processa e reenvia a informação requisitada ao computador remoto. O segundo *software* faz o papel de cliente. Ele envia as seqüências de bytes a outros servidores, requisitando o novo arquivo CIF e testando as respostas dos servidores.

No *software* Hera a rede P2PCIF é possível devido aos algoritmos da Figura 42, Figura 43 e Figura 44. O algoritmo da Figura 42 apenas se comunica com o sistema operacional ativando a capacidade de utilizar a conexão com a Internet. Se o sistema operacional negar por algum motivo a conexão, a rotina apenas avisa o usuário. Este algoritmo está implementado em HERA.PAS. Na Figura 39 está a primeira versão do algoritmo, que foi abandona porque não funciou adequadamente nos ambiente multitarefa dos atuais sistemas operacionais.

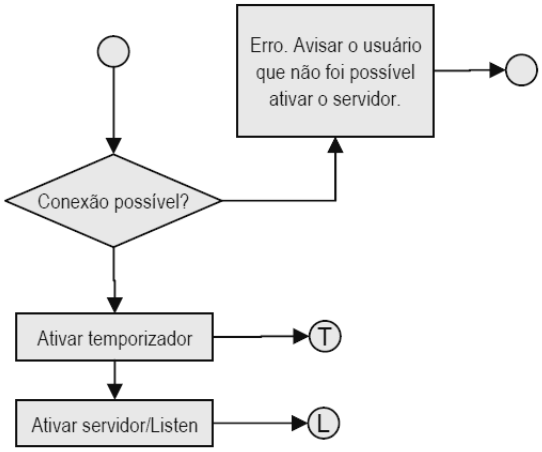

Figura 39 – Algoritmo para a rede P2PCIF versão 1.

O algoritmo da Figura 44 é o responsável por estabelecer as funções de servidor ao *software*. Um *software* que utilize o conceito P2P possui ao mesmo tempo a função de servidor e de cliente. Este algoritmo foi projetado para ser o mais simples possível. Isso foi necessário, pois este fica todo o tempo em execução junto com o *software* principal. A versão anterior, que é mostrada na Figura 41, foi substituída por este motivo.

Quando o programa é iniciado, esta rotina fica residente em uma das portas TCP/UDP da conexão com a internet. Por um padrão pré-estabelecido, foi utilizada a porta 12007 mas o usuário pode escolher qualquer uma das 64.000 portas disponíveis em um computador doméstico. A rotina responde a 7 eventos predefinidos. Estes eventos são indicados de EV1 à EV7. Para acionar um evento, o cliente deve enviar ao servidor uma seqüência predefinida de 40 letras. Isso é necessário para evitar uma confusão com outros serviços da Internet. A idéia é evitar que um cliente requisite um arquivo CIF e receba um arquivo desconhecido.

O primeiro evento, EV1, permite receber a lista de nós do servidor. Um nó é o endereço IP mais a porta TCP/UDP de um cliente. Quando o *software* passar a funcionar como cliente e for se conectar a outro servidor ele precisa conhecer qual o IP e qual a porta, deve utilizar. Estes endereços ficam armazenados em uma lista que é a lista acionada em EV1. Com este processo os computadores que estão participando da rede P2PCIF podem acessar um número cada vez maior de outros computadores.

O EV2 envia a assinatura MD5 do arquivo com a lista de nós do servidor. O computador cliente irá calcular a assinatura MD5 do arquivo recebido e fazer comparações. Se forem iguais pode-se afirmar que não ocorreu nenhum erro na transmissão do arquivo.

O EV3 envia a lista dos arquivos CIF que estão gravados no servidor. EV4 envia a assinatura MD5 deste arquivo. É realizado o mesmo processo de comparação entre as assinaturas recebidas e enviadas para verificar a integridade do arquivo.

Depois de receber a lista de arquivos CIF o cliente verifica quais CIF, ele não possui e os requisita ao servidor utilizando o evento EV5. O EV6 é idêntico a EV4 e EV2. A assinatura MD5 calculada no servidor é enviada ao cliente e comparada com a assinatura MD5 calculada no cliente. EV7 encerra a conexão entre o cliente e o servidor liberando a rotina do servidor para estabelecer uma conexão com outro

cliente. A implementação do algoritmo ECIF resultou em uma rotina que verifica se o arquivo CIF pedido pelo cliente existe. Se não existir, o servidor considera um erro grave e encerra a conexão com o cliente. O algoritmo da Figura 44 esta implementado na biblioteca ServidorRedeP2PCIF.pas.

No outro lado do processo, o cliente, é controlado pelo algoritmo da Figura 43. Para mostrar a evolução no desenvolvimento com relação às versões anteriores estes algoritmos são apresentados na Figura 40. O processo é o inverso do que ocorre no servidor. Os eventos EV1 à EV7 são enviados seguindo a mesma lógica. A rotina não envia os arquivos recebidos ao banco de dados. Eles são gravados em uma pasta temporária e testados depois que a conexão é encerrada. Isso foi necessário por que os testes são feitos utilizando os dicionários DIC. Estes dicionários possuem milhares de campos que estão interligados, o que torna o processo de teste lento. Assim, os testes foram separados do processo de envio e recebimento de arquivos CIF.

O início do algoritmo verifica se a pasta c:\hera\tmp\cif está ou não vazia. Se não estiver, significa que uma transmissão anterior foi encerrada no meio do processo. O algoritmo tentará recuperar estes arquivos. O algoritmo do cliente, TClienteRedeP2PCIF, está implementado na biblioteca ClienteRedeP2PCIF.pas.

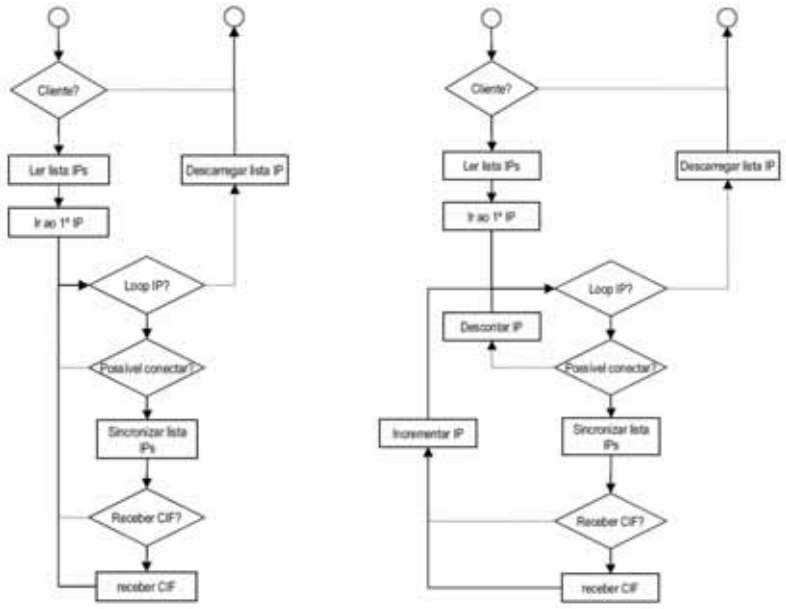

Figura 40 – Algoritmo para clientes da rede P2PCIF versão 1 e 2.

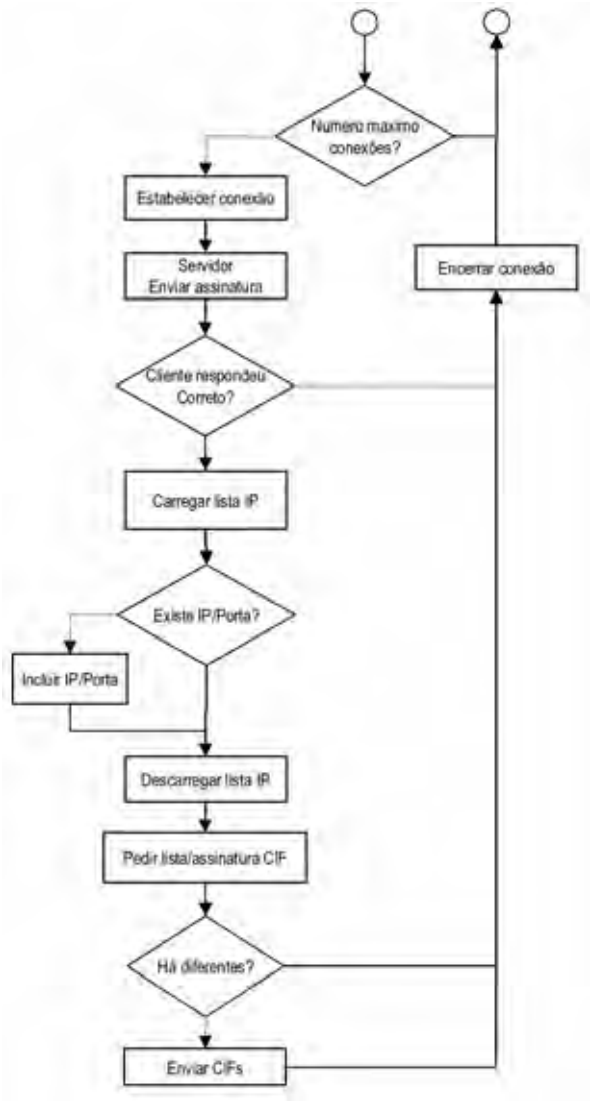

Figura 41 – Algoritmo para o servidor versão 1.

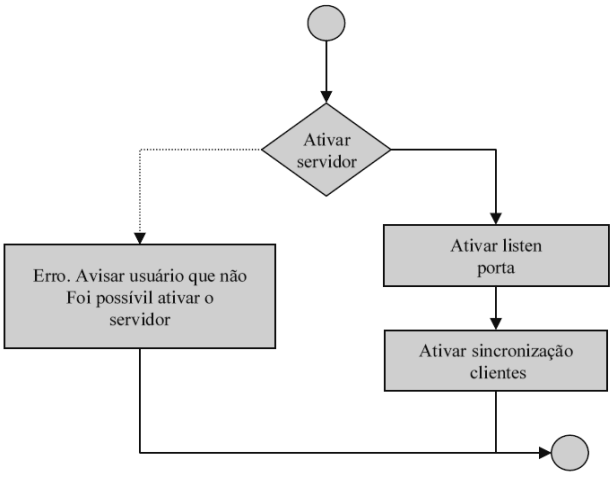

Figura 42 – Algoritmo para a rede P2PCIf versão 2.

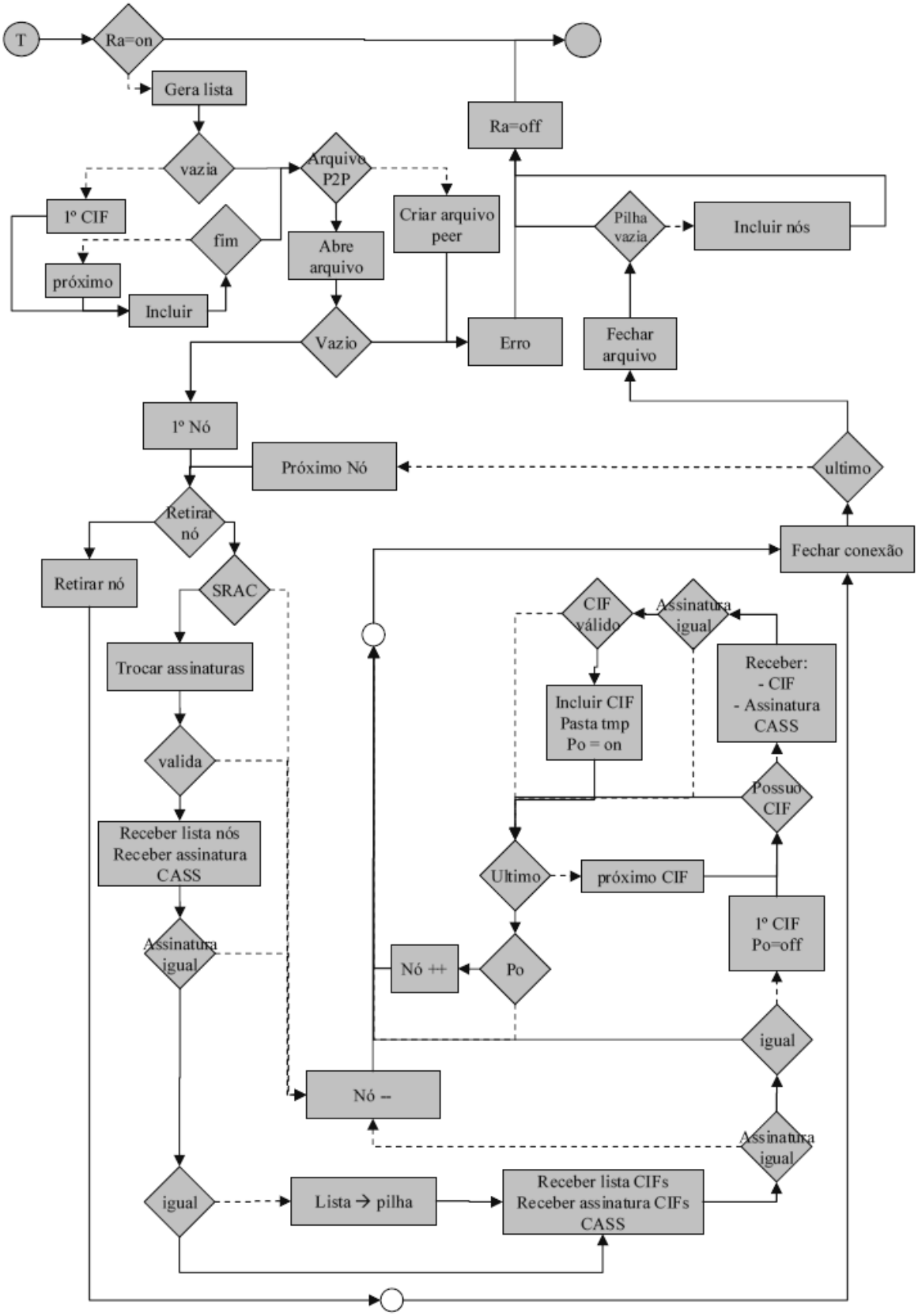

Figura 43 – Algoritmo para a rotina cliente da rede P2PCIF versão 3.

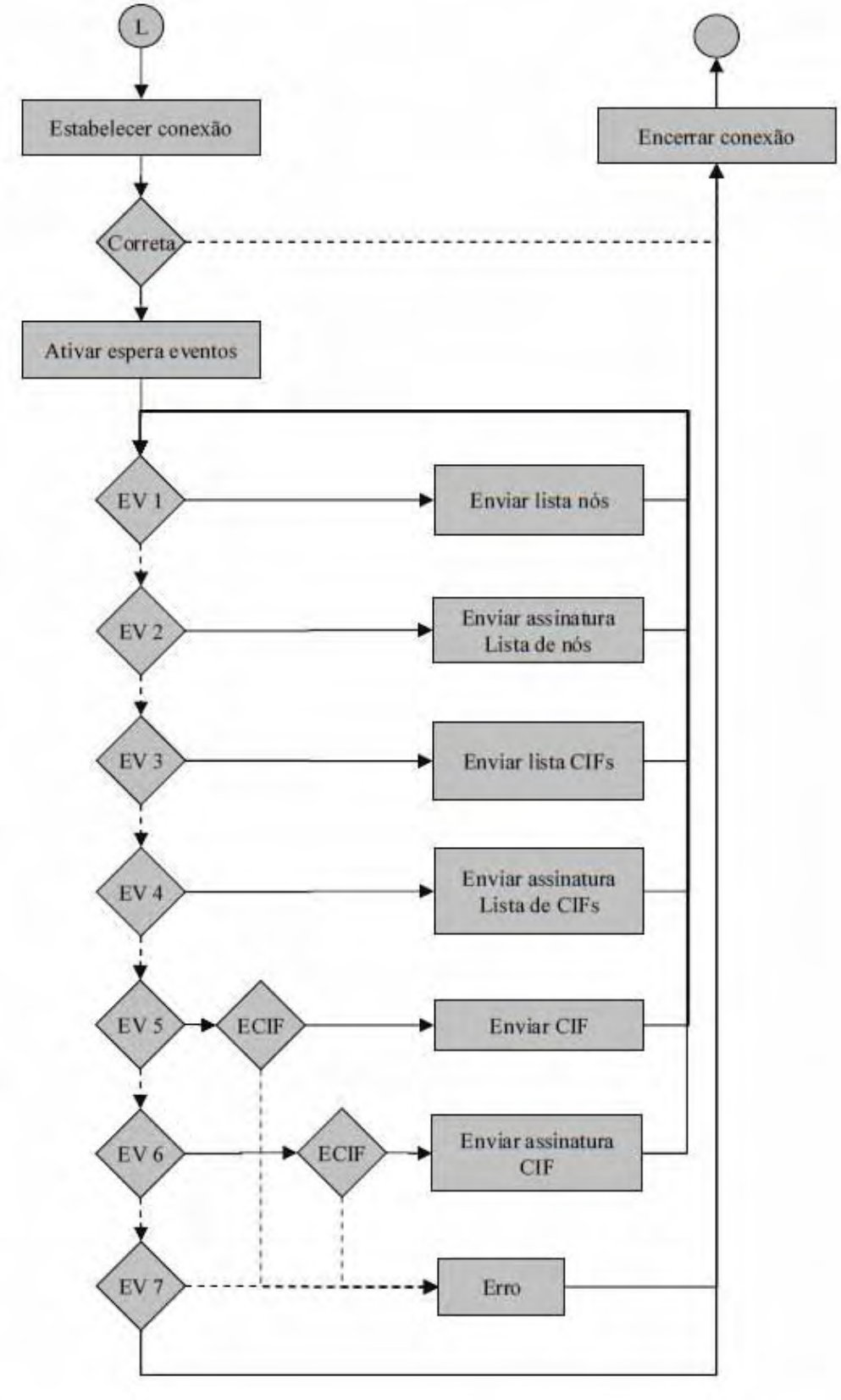

Figura 44 – Algoritmo para a rotina servidor da rede P2PCIF versão 3.

Na proposta inicial a rede permitia mais de uma conexão na porta do servidor . Isso foi alterado para permitir somente uma, pois os testes com os primeiros protótipos demonstraram que isso deixava o aplicativo lento e não contribuía para o crescimento da rede P2PCIF. Nas atuais versões Hera é permitida apenas uma única conexão entre o servidor e o cliente.

4.3 O desenho das interfaces gráficas

Todas as interfaces do *software* Hera seguem o padrão utilizado nas interfaces gráficas dos atuais sistemas operacionais . A lógica básica é seguir o caminho da leitura em idiomas ocidentais, ou seja, para baixo e da esquerda para a direita. Em todas as janelas os comandos que primeiro devem ser acionados ficam a direita e acima de todas as janelas, como os comandos para abrir os arquivos e os de configuração do programa. Os últimos ficam na parte de baixo a direita como os botões e menus para fechar uma janela. Isto gera 3 regiões básicas. Na Figura 45 é mostrada a divisão vertical destas regiões. As regiões A, B e C são indicadas pelos retângulos em cinza.

Na região A ficam os componentes que primeiro devem ser acionados pelo usuário. Na região B os componentes gráficos utilizados depois que a região A for utilizada. Na região C os componentes que serão utilizados por último. Na Figura 46 são mostradas as mesmas três regiões separadas horizontalmente.

As partes visíveis do programa seguem sempre este projeto. Três ações básicas associadas às três regiões A, B e C. O primeiro passo é associado a região A, o segundo passo à região B e o terceiro passo à região C. Na atual versão Hera 0.38 possui apenas os recursos básicos. Com a continuidade do desenvolvimento serão incluídos mais e mais recursos, ferramentas e novas janelas serão criadas. Isso tornará o *software* cada vez mais complexo e difícil de aprender a utilizar.

Com o atual projeto gráfico, a lógica de utilização continuará simples. Sempre três passos básicos associados às regiões A, B e C.

Figura 45 – Desenho vertical das regiões das interfaces gráficas do *software*.

Figura 46 - Desenho horizontal das regiões das interfaces gráficas do *software*.

4.4 Utilizando o software Hera 0.38

Hera além de tornar possível a rede P2PCIF é um aplicativo para identificação de fases em amostras reais. Quando o aplicativo Hera é acionado a janela principal é acionada. A imagem desta janela é mostra na Figura 47. Neste ponto todas as bibliotecas do programa foram lidas e processadas. Os dicionários DIC foram carregados para a memória do computador e a rede P2PCIF é acionada.

Se nenhum erro ocorreu o usuário notará a mensagem SERVER ON. Esta mensagem significa que o sistema operacional liberou a porta TCP requisitada e a busca por novos CIFS foi iniciada. Do lado direito da região B da mensagem “SERVER ON” aparecem dois números. O primeiro mostra quantos nós são conhecidos ou seja em quantos IPs o computador tentará se conectar. O segundo mostra a quantidade de arquivos CIF que estão armazenados no banco de dados local.

Figura 47 – Janela principal do *software* Hera.

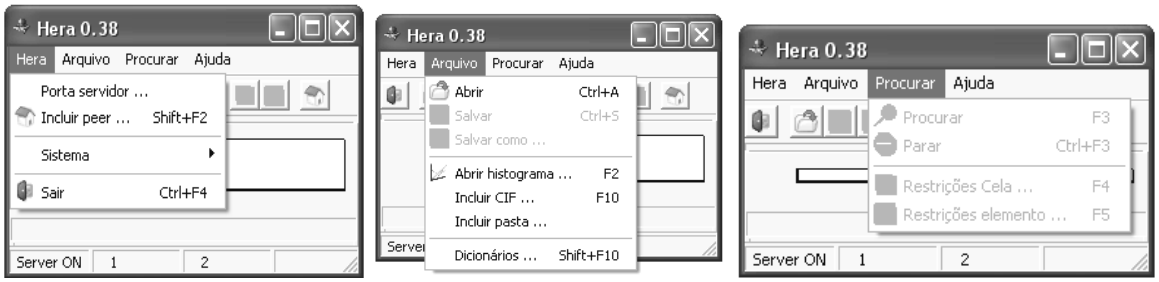

Figura 48 – Os menus do *software* Hera.

A Figura 48 mostra os menus disponíveis ao usuário. Os menus seguem a lógica apresentada no item 4.3. No primeiro menu o usuário poderá trocar a porta TCP do servidor, adicionar novos IPs ou novos nós, ligar e desligar as capacidades de servidor ou de cliente da rede P2PCIF. No segundo menu está disponível ferramentas para ler histogramas, ler arquivos de buscas anteriores, adicionarem novos CIF ao banco de dados local e visualizar os dicionários disponíveis. O

dicionário e os seus vários campos estão disponíveis na janela mostrada na Figura 49.

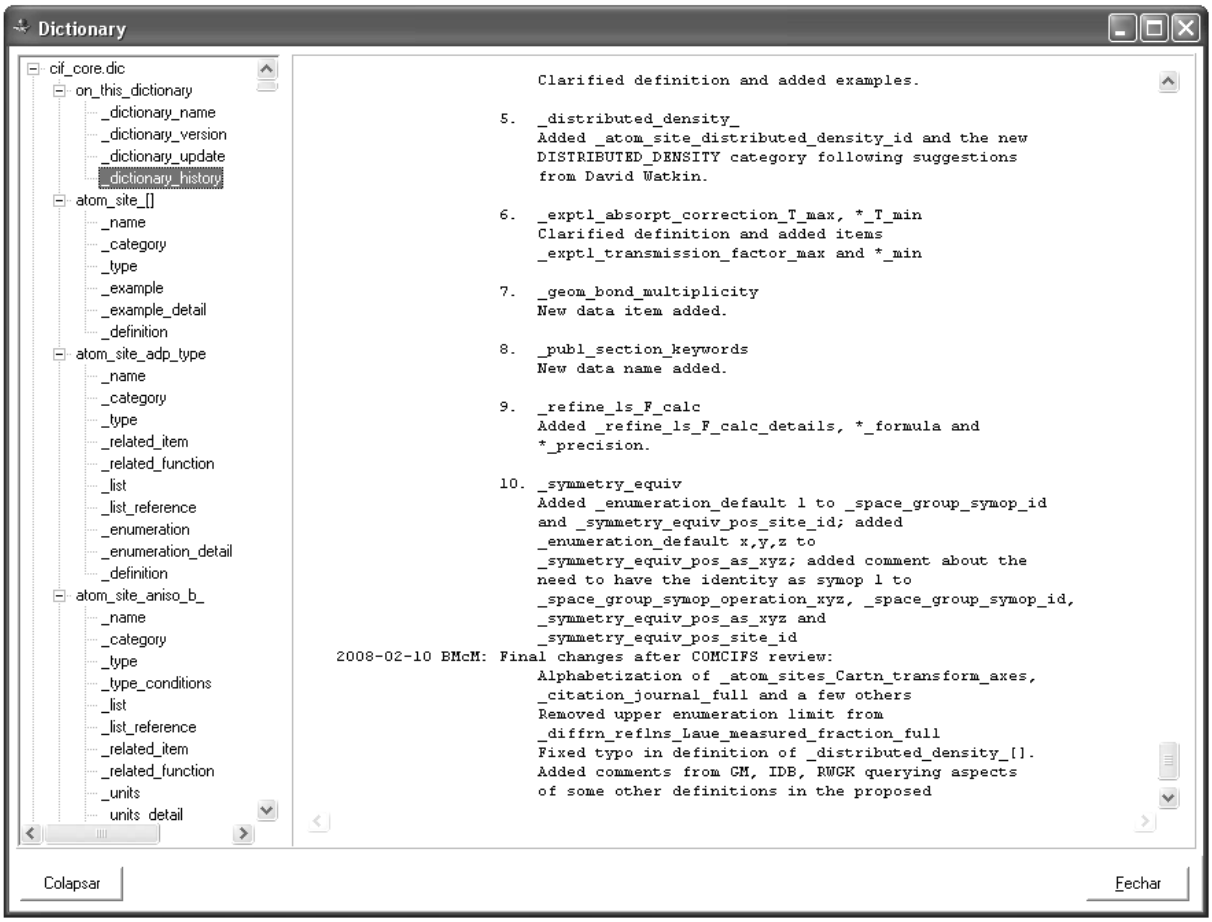

Figura 49 – Janela para mostrar os dicionários DIC disponíveis.

O terceiro menu realiza uma busca nos arquivo CIF do banco de dados local. A busca só será realizada se as condições forem escolhidas previamente. Não é feita busca na rede em outras cópias do *software*.

Ao abrir um histograma um gráfico semelhante ao apresentado na Figura 50 será mostrado, ou seja, um gráfico típico mostrando a contagem de fotons versus 2-teta. O algoritmo normaliza todos os gráficos para que o maior ponto no eixo Y

sempre fique com 100% de intensidade. É permitido o zoom em regiões selecionadas com o mouse.

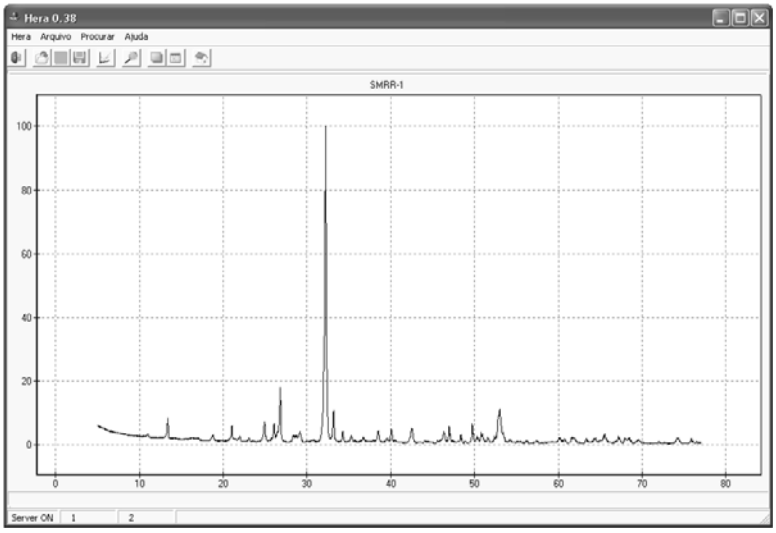

Figura 50 – Um histograma mostrado pelo *software* Hera 0.3.

Para que a busca seja realizada o usuário deve escolher condições que restrinjam os resultados. Há duas opções, escolher quais elementos químicos estão presentes nos arquivos CIF ou escolher qual o grupo espacial. Na janela da Figura 51 o usuário pode escolher entre os 230 grupos espaciais ou os 7 sistemas cristalinos. O programa irá mostrar nos resultados apenas os CIFs que coincidirem com a escolha.

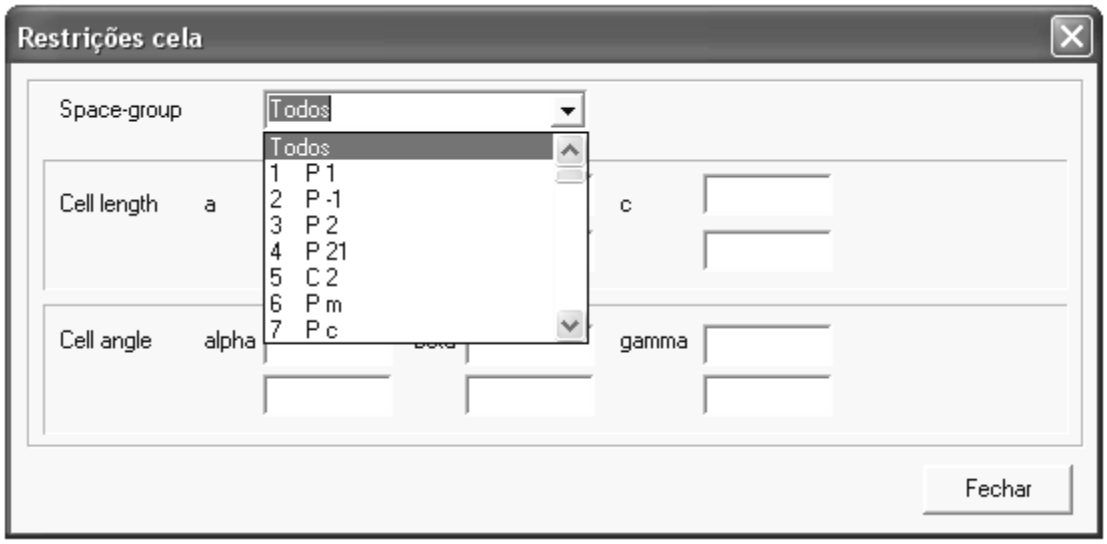

Figura 51 – Caixa de diálogo com as condições de busca utilizando informações cristalográficas.

A opção mais comum é restringir qual o elemento químico estará ou não presente no arquivo CIF. A janela da Figura 52 mostra a tabela periódica para o usuário selecionar os elementos químicos. Há dois operadores disponíveis,.com o operador AND somente os elementos selecionados devem estar presentes no arquivo CIF e com o operador OR a busca retorna os CIF que contenham os elementos selecionados combinados com os outros elementos químicos.

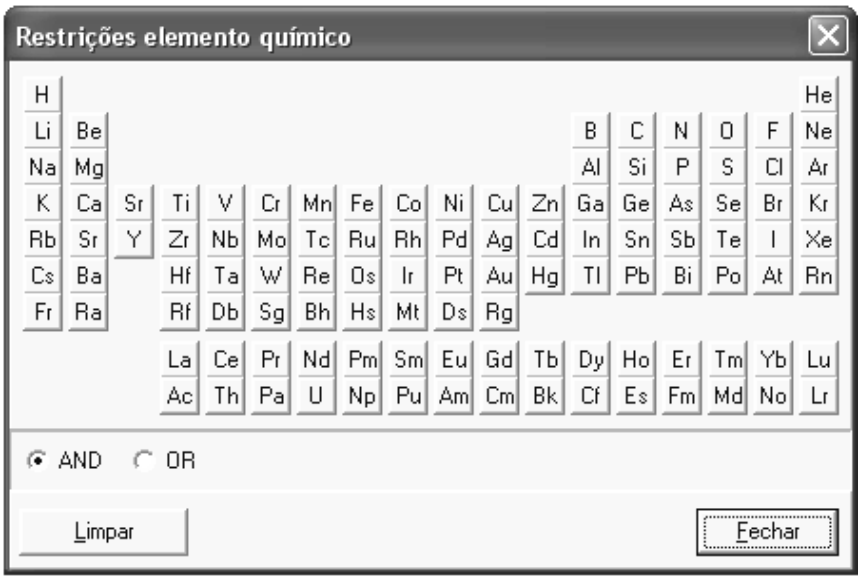

Figura 52 - Caixa de diálogo com as condições de busca para elementos.

Escolhida a condição, basta acionar a opção PROCURAR. O software irá ler todos os arquivos CIF no banco de dados local e verificar quais se enquadram nas regras escolhidas. Durante este processo uma barra de progresso é mostrada como na Figura 53.

O resultado final é mostrado na janela da Figura 54. Uma lista com os nomes das estruturas encontradas é mostrada na parte inferior à direita. Ao selecionar uma estrutura nesta lista, os planos hkls são mostrados junto ao histograma. A intensidade do maior hkl também é ajustada para 100%. O grupo espacial e as dimensões da cela cristalográfica escolhida também são mostrados.

Agora o usuário deve mudar a seleção com as setas do teclado e encontrar qual padrão coincide com a amostra experimental.

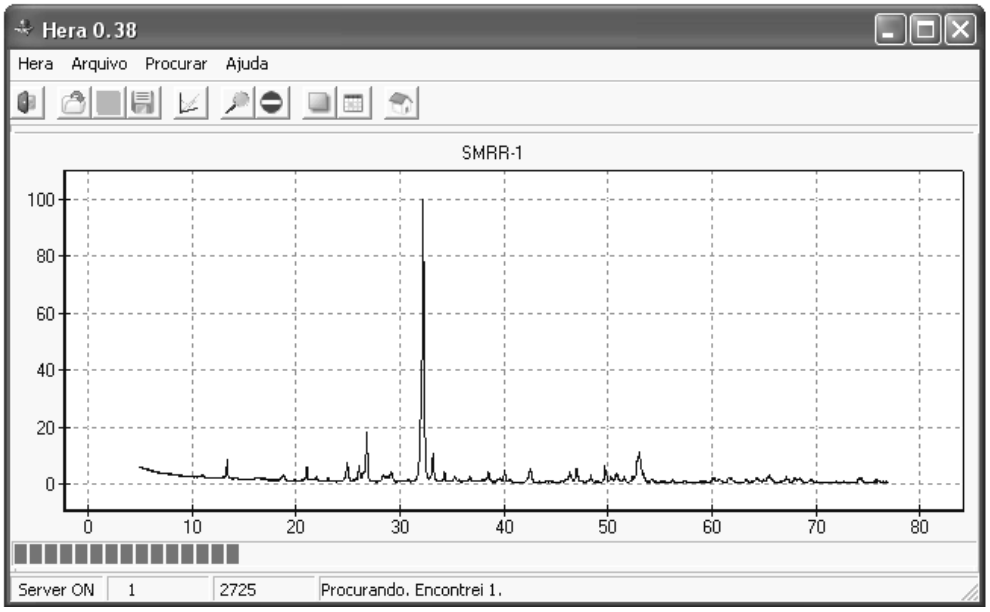

Figura 53 – A visualização do histograma escolhido.

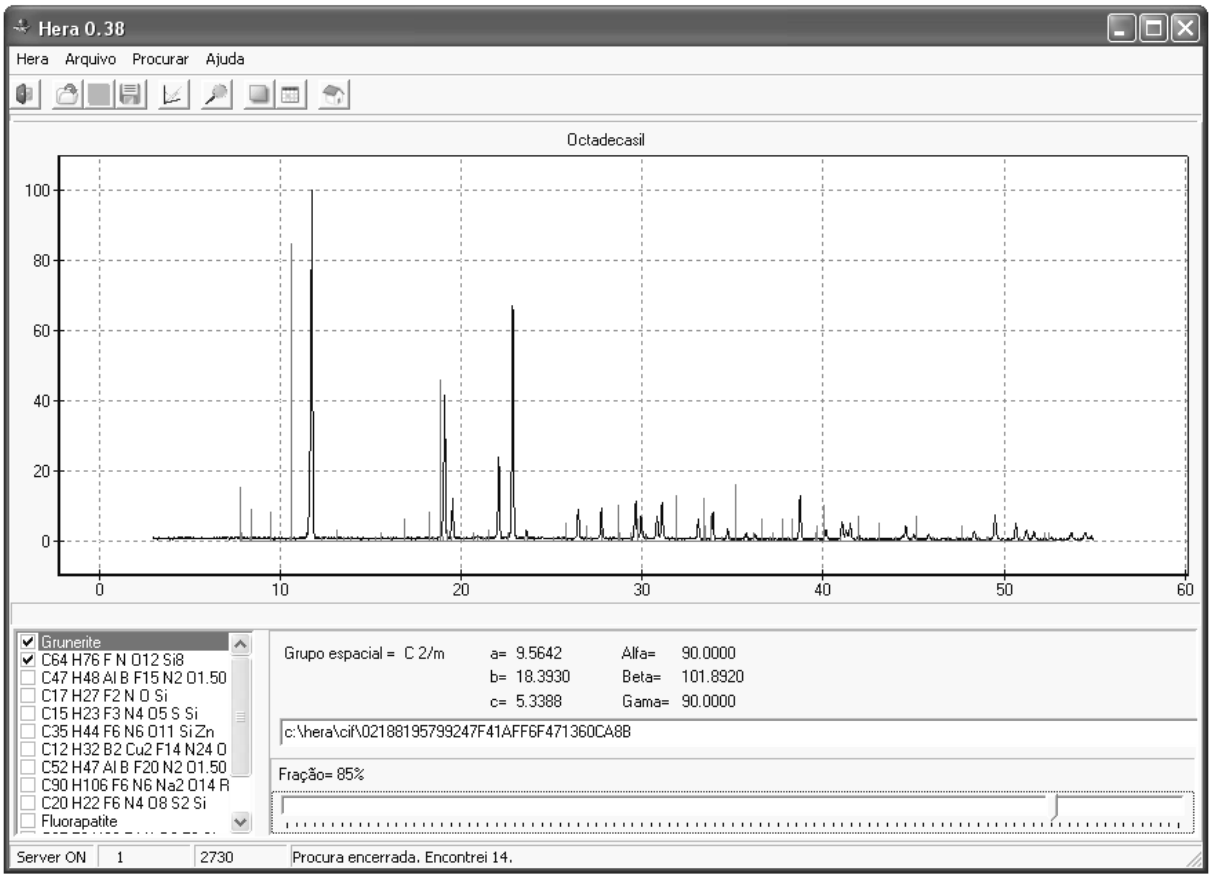

Figura 54 - O resultado da busca.

5 Conclusão

A viabilidade da hipótese da utilização de uma rede P2P com arquivos CIF foi demonstrada através da implementação de Hera. Algoritmos inéditos para automatizar a criação, a manutenção e a validação de bancos de dados distribuídos foram criados e implementados. Nesta Tese também foi apresentado uma nova forma para a nomenclatura de arquivos CIF, *Crystallographic Information File*, baseada unicamente na cela cristalográfica independente de data, instituição e técnica utilizada.

6 REFERÊNCIAS

AUTHIER, A.; LAGOMARSINO, S; TANNER, B. K. **X-Ray and Neutron Dynamical Diffraction**: theory and applications. New York: Plenum Press, 1996.

BOWDEN, M. **Programa Convert**. Disponível em: <http://www.ccp14.ac.uk/ccp/web-mirrors/convx/ >. Acesso em: 1 abr. 2009.

CANTÙ, M. **Dominando o Delphi**. Rio de Janeiro: Makron, 1996.

CARROLL, P. **A derrocada da IBM**: Big Blues. Rio de Janeiro: Ediouro, 1994.

CHEMICAL (A). **Databases Directory**. Disponível em: <http://directory.google.com/Top/Science/Chemistry/_chemical_Databases/>. Acesso em: 1 abr. 2009.

CHEMICAL (B). **Databases Directory**. Disponível em: <http://dir.yahoo.com/Science/Chemistry/Molecular_Databases/> Acesso em: 1 abr. 2009.

EMBARCADERO TECHNOLOGIES. **Delphi version 6.0.** Disponível em: <http://www.codegear.com/products/delphi/win32>. Acesso em: 1 abr. 2009.

CULLITY, B. D. **Elements of X-Ray Diffraction**. 2nd ed. London: Addison-Wesley, 1978.

DRAGOE, N. PowderV2: a suite of applications for powder X-ray diffraction calculations. **J. Appl. Cryst.**, London, v. 34, p. 535-535, July 2001.

DRAGOE, N. **Powder 4 2004**. Disponível em: <http://www.ccp14.ac.uk/ccp/web-mirrors/ndragoe/html/software.html#powder4>. Acesso em: 1 abr. 2009.

FIZ KARLSRUHE GERMANY ; NIST U.S.A. The National Institute of Standards and Technology. **ICSD-Inorganic Crystal Structure Database**. Disponível em: <http://icsdweb.fiz-karlsruhe.de/>. Acesso em: 1 abr. 2009.

GARDNER, S.; THORNTON, J. TDITIS. Protein Structure Database. **Acta Cryst.**, London, v. D, n. 54, p. 1071-1077, Nov. 1998.

GONTIER, S.; SMITH, M. **Programa DLConvert**: windows mass data converter. 2005. Disponível em: <http://www.ccp14.ac.uk/projects/dl-conv/>. Acesso em: 1 abr. 2009.

HAHN, T. **International Tables for Crystallography**. 3nd ed. Dordrecht: Reidel, 1993. v.A, p. 885.

HALL, R. S.; ALLEN, F. H.; BROWN, I. D. The Crystallography Information File (CIF): a new standard archive file for crystallography. **Acta Cryst.**, London, v. A, n. 47, p. 55-685, Jan. 1991.

HEHL, M. E. **Linguagem de programação estruturada**: FORTRAN 77. São Paulo: McGraw-Hill, 1986.

IZUMI, F. **Multi-Purpose Pattern-Fitting System RIETAN-2000**. Disponível em: <http://homepage.mac.com/fujioizumi/index.html>. Acesso em: 1 abr. 2009.

IZUMI, F.; IKEDA, T. A Rietveld analysis program RIETAN-98 and its applications to zeolites. **Mater. Sci. Forum.**, Zurich, v. 321-324, p. 198-205, Jan. 2000.

INTERNATIONAL UNION OF CRYSTALLOGRAPHY. **Crystallographic Information File**. Disponível em: < http://www.iucr.org/resources/cif/>. Acesso em: 1 abr. 2009.

KABERKODU, S. N.; FABER, J.; FAWCETT, T. New Powder Diffraction File (PDF-4) in relational database format: advantages and data-missing capabilities. **Acta Cryst.**, London, v. 2, n. 58, p. 333-337. May 2002.

KANTERE, T. H.; MYLOPOULOS, J.; KIRINGA, I. **A distributed rule mechaninsm for multidatabase systems**. Toronto: University of Toronto Press, 2003.

KOURKOUMELIS, N. **Programa PowDLL for Windows**: version 2004. Disponível em: <http://users.uoi.gr/nkourkou/powdll.htm>. Acesso em: 1 abr. 2009.

LARSON, A. C.; DREELE, R. B. V. **General structure analysis system (GSAS)**. Los Alamos: National Laboratory Report. 1994.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **Databases for chemistry**. Disponível em: <http://www.nist.gov/srd/chemistry.htm>. Acesso em: 1 abr. 2009.

PRESS, W. H. et al. **Numerical recipes in C. The art of scientific computing**. 2nd ed. Cambridge: Cambridge University Press, 1992.

RIETVELD, H. M. A profile refinement method for nuclear and magnetic structures. **J. Appl. Cryst.**, London, v. 2, n. 1, p. 65-71, Jan. 1969.

RIVEST, R. **The MD5 message digest algorithm**. MIT Laboratory for Computer Science. RFC 1321, Apr. 1992. Disponível em: <http://www.ietf.org/rfc/rfc1321.txt> Acesso em: 1 abr. 2009.

RODRÍGUEZ-CARVAJAL, J. Recent developments of the program FULLPROF. **Commission on Powder Diffraction (IUCr)**, London, v. 26, p. 12-19, Jan. 2001.

SPIEGEL, J. V. **ENIAC-on-a-Chip project**. University of Pennsylvania, 1995. Disponível em: <http://www.ese.upenn.edu/~jan/eniacproj.html>. Acesso em: 1 abr. 2009.

TSANG, L. et al. **Scattering of electromagnetic waves**: numerical simulations. New York: John Willey & Sons, 2001.

UTUNI, V. H. S. **Desenvolvimento de uma interface gráfica avançado para os pogramas DBWS-9807a e size2003 empregados no refinamento de estruturas obtidas por DRX**. 2004. 92 f. Dissertação (Mestrado em Química) – Instituto de Química, Universidade Estadual Paulista, Araraquara, 2004.

WHITE, P. S.; RODGERS, J. R.; LE, X. CRYSTMET: a database of the structures and powder patterns of metals and intermetallics. **Acta Cryst.**, London, v. 58, n. 1, p. 343-348, June 2002.

WILSON, A. J. C. **International tables for crystallography**: mathematical, physical and chemical tables. 2nd ed. Dordrecht: Klerwer Academic Publishers, 1995. v. C, cap. 8.6, p. 625-626.

WIRTH, N. **Algoritmos e estrutura de dados**. Rio de Janeiro: Prentice-Hall, 1989.

YOUNG, R. A.. **Program DBWS9807a Release22.02.00**. Disponível em: <http://www.physics.gatech.edu/downloads/young/download_dbws.html>. Acesso em: 1 abr. 2009.

YOUNG, R. A. et al. DBWS-9411 – an upgrade of the DBWS*.* programs for Rietveld refinement with PC and mainframe computers. **J. Appl. Cryst.**, London, v. 28, p.366-367, Jan. 1995.

7 CD Anexo

No CD em anexo está uma cópia da versão 0.38.2 do *software* Hera. Incluindo o código fonte. A instalação é feita copiando a pasta Hera do CD para o C:\.