**Aplicações de computação paralela
em otimização contínua**

Ricardo Luiz de Andrade Abrantes

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA
OBTENÇÃO DO TÍTULO
DE
MESTRE EM CIÊNCIAS

Área de Concentração: Ciência da Computação
Orientador: Ernesto G. Birgin

São Paulo, 01 de maio de 2008

Aplicações de computação paralela em otimização contínua

Este exemplar corresponde à redação
final da dissertação devidamente corrigida
e defendida por Ricardo Luiz de Andrade Abrantes
e aprovada pela Comissão Julgadora.

Banca Examinadora:

- Prof. Dr. Ernesto Julián Goldberg Birgin (orientador) - IME-USP.
- Prof. Dr. Marco Dimas Gubitoso - IME-USP.
- Prof. Dr. Ricardo Caetano Azevedo Biloti - IMECC-UNICAMP.

*Dedico este trabalho ao meu irmão,
que me acompanhou por quase 22 anos
e me ensinou muito sobre a vida,
e a minha mãe,
que sempre esteve ao meu lado.*

Agradecimentos

Muitas pessoas cruzaram minha vida durante estes três anos de mestrado e contribuíram de diversas formas para este trabalho.

Inicialmente gostaria de agradecer à minha mãe Sonia e avó Marja por me apoiarem e me amarem incondicionalmente, se não fosse por elas, jamais teria trilhado este caminho.

Agradeço especialmente ao meu irmão Rodrigo, por ter tido paciência e compreensão em relação à minha ausência nos eventos familiares e pela minha excessiva sinceridade. Seu carinho jamais será esquecido, não importa onde estiver.

Agradeço também à minha namorada Livia por toda compreensão e apoio, mesmo quando eu passava finais de semana e feriados estudando ou trabalhando.

Alguns professores me apoiaram e incentivaram durante toda a graduação e depois. Sou muito grato ao grande e querido mestre João de Oliveira Dantas e ao eterno amigo Fernando Giorno. Com muito carinho agradeço também o apoio de José Arruda Alves, Maurício Pontuschka, Carlos Paes, Demi Getschko, Aritanan Gruber e Cassio Polpo de Campos.

Agradeço aos excelentes professores e amigos do grupo de otimização contínua do IME-USP com quem tive o prazer de trabalhar e estudar. Sou grato ao professor Carlos Humes Jr. por ter sido meu orientador de programa e por ter me ajudado com os primeiros passos neste instituto com conselhos valiosos. Em especial o meu orientador Ernesto Birgin por toda sua dedicação, paciência, conselhos e amizade. Sem ele certamente este trabalho também não seria concretizado. Agradeço ao Marcelo Gomes de Queiroz, Paulo José da Silva e Silva, Walter Figueiredo Mascarenhas e Alexandre Roma pelas aulas e conselhos.

Agradeço aos professores e colegas do projeto InteGrade por toda ajuda e companheirismo. Em especial gostaria de ressaltar aqui os professores Fabio Kon e Alfredo Goldman por todas as conversas, conselhos e incentivos.

São tantos os amigos que estiveram ao meu lado durante o mestrado. Peço desculpas por eventualmente não citar alguém. Alguns nomes muito queridos e importantes são: Sérgio Ribeiro Savastano de Souza, Maria Aparecida dos Santos Mikó de Souza e Rodrigo de Carvalho Martínez por serem como meus irmãos e estarem ao meu lado sempre que precisei; Daniel Cordeiro, Alexandra Cesar Tavares Vidal, José de Ribamar Braga Pinheiro Júnior, Marco Netto, Raphael Yokoingawa de Camargo, por todo o carinho, amizade e ajuda em diversos momentos que precisei e nas inúmeras aventuras no LCPD e InteGrade; Arlindo Flávio da Conceição, Giuliano Mega, Celina Maki Takemura, Andresa Fabiana, Eduardo Guerra, Vladimir Moreira Rocha, Cláudia de Oliveira Melo, Ulisses Kendi Hayashida, Rafael Peres Correia, Vinicius Pinheiro, Fernando Castor Filho e Helves por serem meus eternos companheiros e por temos nos divertido tanto juntos em momentos muitas vezes difíceis de se divertir; Marina Andretta, Francisco Sobral, Rafael Durbano Lobato, Marcelo Hashimoto, Ellen Hidemi Fukuda, Antonio Carlos dos Santos, Domingos Soares Neto e Fernando Taietti por serem grandes amigos e me ajudarem em diversos momentos ao longo desse trabalho ou nas disciplinas que cursamos juntos; Paulo Salem, Carlos Cardonha, Marcel Kenji de Carli Silva, Cristiane Maria Sato, Gordana Manic e toda a turma da “salinha” por estudarmos juntos e por compartilharmos tantos bons momentos em “pizzadas”, reuniões diversas e bate papos.

Todos aqui citados e também aqueles que não mencionei, mas que de alguma forma me acompanharam nestes últimos três anos, influenciaram de alguma forma minha formação. Sou muito grato a todos.

Abstract

In this work we studied some concepts of parallel programming, some ways of using parallel computing in continuous optimization methods and two optimization methods.

The first method we present is called PUMA (Pointwise Unconstrained Minimization Approach), and it retrieves optical constants and thicknesses of thin films from transmittance data. The problem of retrieve thickness and optical constants is modeled as an inverse problem and solved with aid of an optimization method. Through the paralelization of PUMA we managed to retrieve optical constants and thicknesses of thin films in structures with one and two superposed films. We describe some results and discuss the performance of the parallel PUMA and the quality of the retrievals.

The second studied method is used to build an initial configuration of molecules for molecular dynamics simulations and it is called PACKMOL. The problem of create an initial configuration of molecules is modeled as a packing problem and solved with aid of an optimization method. We developed a parallel version of PACKMOL and we show the obtained performance gains.

Keywords: continuous optimization methods, optimization applications, parallel algorithms.

Resumo

No presente trabalho, estudamos alguns conceitos relacionados ao desenvolvimento de programas paralelos, algumas formas de aplicar computação paralela em métodos de otimização contínua e dois métodos que envolvem o uso de otimização.

O primeiro método que apresentamos, chamado PUMA (*Pointwise Unconstrained Minimization Approach*), recupera constantes óticas e espessuras de filmes finos a partir de valores de transmitância. O problema de recuperação é modelado como um problema inverso e resolvido com auxílio de um método de otimização. Através da paralelização do PUMA viabilizamos a recuperação empírica de constantes e espessuras de sistemas compostos por até dois filmes sobrepostos. Relatamos aqui os resultados obtidos e discutimos o desempenho da versão paralela e a qualidade dos resultados obtidos.

O segundo método estudado tem o objetivo de obter configurações iniciais de moléculas para simulações de dinâmica molecular e é chamado PACKMOL. O problema de obter uma configuração inicial de moléculas é modelado como um problema de empacotamento e resolvido com o auxílio de um método de otimização. Construimos uma versão paralela do PACKMOL e mostramos os ganhos de desempenho obtidos com a paralelização.

Palavras-chaves: métodos de otimização contínua, aplicações de otimização, algoritmos paralelos.

Sumário

1	Introdução	1
2	Computação Paralela	3
2.1	Arquiteturas de computadores paralelos	3
2.1.1	Taxonomia de Flynn	3
2.1.2	Organização da memória	4
2.2	Modelos de programação paralela	5
2.2.1	Troca de mensagens	5
2.2.2	Memória compartilhada	6
2.2.3	<i>Bulk Synchronous Parallel</i>	6
2.2.4	Paralelismo de dados	7
2.2.5	<i>Threads</i>	7
2.3	Desempenho de um algoritmo paralelo	7
2.4	Paralelismo em otimização contínua	9
2.4.1	Possibilidades de paralelização	10
3	PUMA	11
3.1	Modelo matemático do problema	12
3.2	Técnica de otimização	17
3.3	Estratégia de paralelização	18
3.4	Resultados numéricos	20
3.4.1	Filmes gerados por computador	20
3.4.2	Filmes reais	35
3.4.3	Desempenho paralelo	36
3.5	Análise dos resultados	43
4	Packmol	45
4.1	Modelo matemático do problema	45
4.2	O Método e sua Implementação	48
4.3	Paralelização do PACKMOL	53
4.4	Considerações adicionais sobre a paralelização	58

SUMÁRIO

4.5 Resultados Numéricos	59
5 Conclusões	65
Referências Bibliográficas	67
A PUMA	71
B PACKMOL	73

Capítulo 1

Introdução

Desde a criação dos primeiros computadores busca-se a melhoria de seu desempenho através do desenvolvimento de processadores mais velozes. Em 1965 Gordon E. Moore, um dos fundadores da Intel, afirmou que possivelmente nos 10 anos seguintes a quantidade de transistores em um *chip*, de forma a se obter um custo mínimo por componente, dobraria a cada ano. Essa afirmação aos poucos se tornou uma espécie de lei para a indústria de microprocessadores, que passou a buscar sempre dobrar a capacidade de processamento de seus processadores a cada ano, ou a cada dois anos.

Porém, recentemente, esse otimismo vem mudando e acredita-se que estamos muito próximos de limites físicos que impedirão o crescimento da capacidade de processamento de forma indefinida. Uma maneira de contornar o problema e continuar aumentando a capacidade de processamento dos computadores é através de computadores paralelos. Grandes membros da indústria de processadores já estão trabalhando em *chips* com múltiplos núcleos, capazes de realizar operações em paralelo. Recentemente a Intel lançou um microprocessador com quatro núcleos. Essa mesma empresa tem investido na pesquisa e desenvolvimento de um processador com 80 núcleos e com poder computacional de um teraflop que potencialmente estará disponível comercialmente em cinco anos. Outra forma de se conseguir maior poder computacional é através da conexão de diversos computadores, com a finalidade de trabalharem em conjunto para realizar uma tarefa. Muitos dos chamados supercomputadores atuais são construídos dessa forma. Atualmente o supercomputador mais poderoso que se tem notícia é conhecido por BlueGene da IBM, localizado no laboratório nacional de Lawrence Livermore, em Livermore nos Estados Unidos. Ele possui 131072 processadores, atingindo uma taxa de processamento de 360 TeraFlops.

Essa busca por poder computacional tem o objetivo de permitir a resolução de problemas computacionais cada vez maiores e, conseqüentemente, mais difíceis. Alguns exemplos de áreas que têm se beneficiado com o uso de técnicas de paralelismo são: álgebra linear, simulação de sistemas físicos, imageamento sísmico, bioinformática, entre outros.

Nossa intenção nesse trabalho é estudar como a computação paralela pode nos ajudar a resolver alguns problemas de otimização contínua. Estudamos uma aplicação de otimização contínua, o PACKMOL e um método de otimização, o PUMA e como paralelizá-los.

O PUMA (*Pointwise Unconstrained Minimization Approach*) é um método para recuperar cons-

tantes óticas e espessuras de filmes finos a partir de valores de transmitância. O problema de recuperação é modelado como um problema inverso e resolvido com auxílio de um método de otimização. Através da paralelização do PUMA viabilizamos a recuperação empírica de constantes e espessuras de sistemas compostos por até dois filmes sobrepostos. Relatamos aqui os resultados obtidos e discutimos o desempenho da versão paralela e a qualidade dos resultados obtidos.

O PACKMOL tem o objetivo de obter configurações iniciais de moléculas para simulações de dinâmica molecular. O problema de obter uma configuração inicial de moléculas é modelado como um problema de empacotamento e resolvido com o auxílio de um método de otimização. Construímos uma versão paralela desse método e mostramos os ganhos de desempenho obtidos com a paralelização.

O segundo capítulo deste trabalho apresenta alguns conceitos básicos relacionados ao desenvolvimento de aplicações paralelas e de que forma podemos desenvolver algoritmos de otimização contínua paralelos. O terceiro capítulo detalha nosso trabalho com o PUMA. O quarto capítulo expõe nosso trabalho com o PACKMOL. O quinto capítulo resume nossas conclusões gerais e acresce linhas para trabalhos futuros.

Capítulo 2

Computação Paralela

A computação paralela é a execução simultânea de processos computacionais em múltiplos processadores a fim de resolver um mesmo problema computacional de forma mais rápida. Nesse capítulo, fortemente baseado nos trabalhos [Fos95], [BT89] e [Cen96], daremos uma breve introdução às diferentes arquiteturas de computadores paralelos e distribuídos, aos modelos de computação paralela e a algumas técnicas para a avaliação de desempenho de um algoritmo paralelo.

2.1 Arquiteturas de computadores paralelos

Há cerca de 60 anos a grande maioria dos computadores possui virtualmente a mesma arquitetura, chamada de arquitetura de von Neumann. Nela os computadores são divididos basicamente em unidade de processamento e memória. Os programas e os dados são localizados na memória. A unidade de processamento lê instruções de um programa e dados contidos na memória, decodifica as instruções e sequencialmente as executa. Nesse caso, temos uma única unidade de processamento que acessa a memória buscando sempre um conjunto de dados, não havendo nenhum tipo de paralelismo. Quando tratamos de computadores paralelos é necessário entender os modelos de organização dessas máquinas para podermos modelar e implementar algoritmos ou aplicações paralelas.

Segundo Ian Foster em [Fos95], um computador paralelo é um conjunto de processadores interconectados, capazes de trabalhar de forma simultânea e cooperativa a fim de resolver um problema computacional. Essa definição é suficientemente ampla para incluir supercomputadores paralelos que possuem centenas ou milhares de computadores, redes locais de computadores, estações de trabalho com múltiplos processadores, entre outros.

2.1.1 Taxonomia de Flynn

Uma ampla classificação das arquiteturas de máquinas paralelas foi oferecida por Flynn em [Fly72] e é até hoje considerada importante para a compreensão do assunto. Segundo Flynn os computadores podem ser classificadas da seguinte maneira:

- **SISD** (*Single Instruction stream and Single Data stream*): Arquitetura onde uma unidade de processamento pode executar uma única instrução em um único conjunto de dados, não há paralelismo. Corresponde à arquitetura de von Neumann. Há aproximadamente uma década essa arquitetura tem deixado de ser o padrão na construção de computadores, sendo substituída por máquinas do tipo SIMD e MIMD.
- **SIMD** (*Single Instruction stream and Multiple Data stream*): Arquitetura onde uma unidade de processamento pode executar uma mesma instrução e operar em múltiplos conjuntos de dados. Aplicações que precisam executar a mesma operação em grandes vetores ou matrizes podem tirar vantagem desse tipo de arquitetura. Um exemplo popular e recente de máquinas que seguem essa arquitetura é a tecnologia MMX da Intel ou a 3dNow! da AMD. Apesar dos processadores com essa tecnologia serem capazes de se comportar de acordo com a arquitetura SISD, um subconjunto de instruções operam de acordo com a arquitetura SIMD.
- **MISD** (*Multiple Instruction stream and Single Data stream*): Arquitetura onde múltiplas unidades de processamento executam diferentes instruções simultaneamente em um mesmo conjunto de dados. Não se conhece muitos exemplos de máquinas desse tipo e ele é citado apenas por uma questão de completude.
- **MIMD** (*Multiple Instruction stream and Multiple Data stream*): Arquitetura onde múltiplas unidades de processamento executam diferentes instruções simultaneamente em diversos conjuntos de dados diferentes.

2.1.2 Organização da memória

A categoria MIMD citada na seção anterior abrange diversos subtipos de arquiteturas. A classificação de Flynn é habitualmente considerada incompleta por não apresentá-los. Outros pesquisadores estenderam essa classificação completando-a. Vamos citar aqui três arquiteturas do tipo MIMD que diferem com relação à organização da memória.

- **Memória compartilhada:** Computadores com memória compartilhada possuem mais de um processador e uma área de memória comum a todos. Os processadores executam de forma independente uns dos outros, mas têm acesso aos mesmos endereços de memória, de forma que mudanças na memória realizadas por um sejam imediatamente visíveis a todos os demais processadores. Os computadores com memória compartilhada podem ser divididos em duas grandes classes: *Unified Memory Access* (UMA) e *Non-Unified Memory Access* (NUMA). UMA diz respeito a computadores com memória compartilhada onde os processadores são idênticos e possuem acesso e tempo de acesso à memória iguais. NUMA refere-se à computadores com processadores que possuem diferentes tempos de acesso a memória.
- **Memória distribuída:** Sistemas com memória distribuída geralmente são compostos por diversas unidades de processamento conectadas por uma rede de dados, onde cada processador

possui uma área de memória local apenas. Os processadores podem operar independentemente uns dos outros e se precisarem acessar endereços de memória não-local geralmente esses dados devem ser transferidos pela rede de uma área de memória para outra.

- **Sistemas híbridos:** São sistemas onde ambas formas de organização de memória estão presentes. São comuns em sistemas distribuídos, onde cada computador presente no sistema possui duas ou mais unidades de processamento que compartilham endereços de memória entre si. A grande maioria dos supercomputadores possuem arquitetura híbrida de memória.

2.2 Modelos de programação paralela

Desenvolver algoritmos paralelos é uma tarefa mais complexa que desenvolver algoritmos sequenciais. Além de todas as abstrações usadas no modelo sequencial de programação, os modelos paralelos usam outros conceitos como: comunicação entre tarefas, concorrência, escalabilidade e localidade.

A comunicação entre tarefas diz respeito à necessidade de troca de dados entre duas tarefas. Dependendo do modelo de programação considerado, as tarefas não possuem acesso direto a todos os dados relacionados ao problema e é necessário que dados sejam trocados entre elas. A localidade diz respeito a quais dados são locais (*i.e.*, diretamente acessíveis) a cada tarefa ou não. A escalabilidade é a capacidade da aplicação ser eficiente a medida que ocorrem acréscimos na quantidade de processadores em que ela é executada. A concorrência é quando há o acesso simultâneo por duas ou mais tarefas a um recurso do sistema, como um endereço de memória por exemplo.

A seguir daremos uma introdução aos modelos de programação paralela mais usados.

2.2.1 Troca de mensagens

O modelo de troca de mensagens é um dos mais populares modelos para programação paralela. Nele uma aplicação paralela é formada por diversas tarefas independentes que se comunicam umas com as outras através de troca de mensagens a fim de coordenar suas ações, ou compartilhar dados. Uma forma de visualizar esse modelo é através de um grafo direcionado, onde cada vértice representa uma tarefa e cada aresta direcionada representa uma comunicação entre duas tarefas.

Cada tarefa pode realizar operações de leitura ou escrita em sua memória local, operações aritméticas, chamadas de funções locais e enviar ou receber mensagens. Normalmente o envio de uma mensagem é instantâneo e ocorre de forma assíncrona, enquanto o recebimento ocorre de forma síncrona e bloqueia o fluxo de execução da tarefa até que a mensagem seja recebida.

Esse modelo considera que um grupo de tarefas é criado no início da aplicação e cada uma é atribuída a um processador. As tarefas representam diferentes fluxos de execuções e isso permite que o modelo de troca de mensagens seja visto como um modelo *multiple program multiple data* (MPMD). O modelo MPMD claramente se relaciona com a arquitetura do tipo MIMD, já que nessa arquitetura cada processo pode ser executado por uma unidade de processamento diferente, permi-

tindo uma execução mais rápida do programa paralelo. Seguindo esse raciocínio, é interessante que programas paralelos com a estrutura MPMD sejam executados em máquinas do tipo MIMD.

A MPI ou *Message Passing Interface* é a especificação de uma interface para programação paralela com troca de mensagens, independentemente de linguagem ou protocolo de comunicação. Existem implementações dessa interface para as principais linguagens de programação como C, C++ ou Fortran. A MPI foi criada com o objetivo de permitir alto desempenho, escalabilidade e portabilidade. Apesar de ser bem sucedida nesses objetivos, é bastante criticada por ser considerada uma interface de baixo nível que expressa o paralelismo explicitamente.

2.2.2 Memória compartilhada

No modelo de memória compartilhada as tarefas compartilham um espaço de endereçamento de memória onde podem ler e escrever de forma assíncrona. Nesses sistemas a programação pode ser simplificada pela ausência da necessidade de troca explícita de dados entre as tarefas, porém fica mais complicado controlar a localidade das ações em memória. Frequentemente é necessário o uso de semáforos ou travas (*locks*) para controlar o acesso de escrita na memória. As tarefas podem, eventualmente, ser independentes umas das outras e trabalhar com diversos conjuntos de dados diferentes, apesar de compartilharem um mesmo espaço de endereçamento. Por isso, esse modelo de programação também é considerado um modelo MPMD.

Uma API para programação em ambientes de memória compartilhada é o OpenMP (*Open Multi-Processing*) [Ope]. Considerado um padrão pela indústria de computação, o OpenMP foi desenvolvido por um grupo de pesquisadores que formam o *OpenMP Architecture Review Board* (ARB) e foi lançado em outubro de 1997. Consiste na especificação de algumas diretivas de compilação e funções para permitirem a programação com memória compartilhada. Sua especificação contempla as linguagens C, C++ e Fortran (77, 90 e 95). Diversos compiladores já possuem suporte a esta API. Dentre eles, podemos citar compiladores da Intel, Sun Studio, Visual C++ e GCC.

2.2.3 Bulk Synchronous Parallel

O *Bulk Synchronous Parallel* ou BSP [Val90] é um modelo para programação paralela proposto por Valiant em 1990. Esse modelo é caracterizado por unidades de processamento com memória local e uma rede de comunicação que permite o acesso de diferentes processadores à memória dos demais. Essa rede de comunicação pode ser um barramento de alta velocidade, ou uma rede de velocidade reduzida, como uma rede *ethernet* por exemplo, dependendo da arquitetura de hardware considerada na implementação do modelo.

Uma aplicação BSP é definida por conjuntos de super passos (*superstep*). Cada super passo é uma coleção de ações, envolvendo computação local e/ou comunicação entre tarefas, seguidas de uma barreira de sincronização. Na barreira de sincronização, as tarefas que já terminaram suas ações de computação ou comunicação, aguardam até que todas tenham terminado o programado para o super passo em questão. Quando todas as tarefas terminam suas ações, prosseguem para o próximo super passo.

Existe uma especificação de uma API, para a linguagem C, que define funções para a programação paralela usando o modelo BSP chamada BSPLib [HMS⁺98]. O sítio [BSP] contém informações sobre implementações dessa especificação.

2.2.4 Paralelismo de dados

Neste modelo, o trabalho a ser realizado em paralelo consiste principalmente em operações de leitura e escrita na memória. Os dados são geralmente organizados em uma estrutura como vetor ou matriz. Todas as tarefas operam sobre esta estrutura, acessando conjuntos disjuntos de dados evitando concorrência pelo acesso à mesma área da memória. As tarefas possuem a restrição de realizarem sempre as mesmas ações, só que em conjuntos distintos de dados.

Aplicações onde possa-se realizar a decomposição de domínio de dados, ou seja, o particionamento dos dados em grupos para que sejam realizadas operações sobre eles, encaixam-se bem nesse modelo. Um exemplo é a multiplicação de matrizes. As matrizes são divididas de modo uniforme e cada tarefa opera sobre uma porção dos dados de forma paralela.

2.2.5 *Threads*

No modelo de *threads*, as tarefas têm uma área de memória local e podem executar de forma independente umas das outras por isso esse modelo é visto como um modelo MPMD. Em alguns casos duas ou mais tarefas podem compartilhar uma área de endereçamento de memória de forma similar ao modelo de memória compartilhada. Tarefas podem ser criadas ou destruídas dinamicamente a medida que o programa é executado.

Aplicações onde se possa realizar a decomposição funcional, ou seja, o particionamento da aplicação em grupos de tarefas distintas, encaixam-se bem nesse modelo. Dentre as bibliotecas de funções mais populares para programação baseada em *threads*, temos o OpenMP, citado na Seção 2.2.2 e ainda *POSIX Threads* (PThreads) [IG04]. PThreads é um padrão que define uma API para criação e manipulação de *threads* desenvolvido pelo *The Open Group* e pelo IEEE.

2.3 Desempenho de um algoritmo paralelo

Para avaliar o desempenho empírico de um algoritmo paralelo podemos usar as medidas de aceleração (ou *speedup*) e eficiência. *Speedup* refere-se a quão mais rápido um algoritmo paralelo é em relação à sua versão sequencial. A medida de *speedup* é dada por:

$$S_{\varphi} = \frac{\tau_1}{\tau_{\varphi}}$$

onde:

- φ é a quantidade de processadores;

- τ_1 é o tempo de execução da versão seqüencial do algoritmo para uma instância. Pode ser o tempo de execução de uma implementação seqüencial, ou o tempo de uma implementação paralela sendo executada com um único processo;
- τ_p é o tempo de execução da versão paralela do algoritmo para a mesma instância usada para medir τ_1 .

É considerado um *speedup* ideal, também chamado de linear, quando $S_p = p$. Em alguns casos o valor do *speedup* pode ser superior à quantidade de processadores utilizados. Nesses casos anômalos, o *speedup* chamado de super linear. Existem algumas causas para esse fenômeno, uma bastante comum é a agregação de memória cache dos vários processadores presentes no sistema. Esse efeito pode reduzir significativamente o tempo de acesso à memória e tornar a execução muito mais rápida.

A eficiência de um algoritmo paralelo é dado pela fórmula:

$$E_p = \frac{S_p}{p}$$

onde:

- p é a quantidade de processadores;
- S_p é o *speedup*.

Essa medida é sempre um valor entre 0 e 1 (considerando teoricamente que $S_p \leq p$), que estima quão bem utilizados estão sendo os processadores em relação aos custos com comunicação e sincronização.

Outro conceito importante relacionado ao desempenho de algoritmos paralelos é a lei de Amdahl. Essa lei fornece o *speedup* máximo esperado de um algoritmo quando apenas uma porção dele é paralelizada. O valor do *speedup* máximo esperado é dado por:

$$\frac{1}{F + \frac{(1 - F)}{p}}$$

onde:

- p é a quantidade de processadores;
- F é a porcentagem de código que é executada seqüencialmente.

Se considerarmos que $p \rightarrow \infty$ o *speedup* máximo que poderemos obter é $\frac{1}{F}$. Essa afirmação pode ser considerada uma limitação à aplicação da computação paralela. Porém, segundo Ian Foster em seu livro [Fos95]: “Nos primórdios da computação paralela acreditava-se que este efeito¹ limitaria a utilidade da computação paralela a um pequeno número de aplicações especializadas. Contudo,

¹O fato da lei de Amdahl oferecer um limite superior para o *speedup*

a experiência prática demonstra que esta maneira inerentemente sequencial de pensar é de pequena relevância em problemas reais.”

Fatores como quantidade de operações de entrada e saída (E/S), necessidade de sincronização entre processos e efeitos de cache, influenciam o desempenho de um algoritmo paralelo.

A quantidade de operações de E/S e sincronizações entre processos tem efeito negativo sobre o *speedup*. As operações de E/S mais comuns em programas paralelos são acesso ao disco e comunicação através de uma interface de rede. Tais operações são bastante custosas e se realizadas em grande quantidade podem prejudicar o desempenho do programa. A sincronização entre processos, em geral, faz com que alguns processos fiquem parados enquanto outros processam, antes que uma determinada ação possa continuar. Dessa forma, implica em desperdício de recursos computacionais. Outro fator negativo da sincronização é que ela pode envolver também a realização de operações de E/S para conseguir um estado consistente dos processos. Por isso, sincronizações devem ser utilizadas com cautela para não prejudicar muito o *speedup*.

A quantidade de níveis de memória cache e o tamanho de cada nível podem influenciar positivamente o *speedup* da execução de um algoritmo paralelo. Em caches grandes, potencialmente, todos os dados são carregados na memória cache. Em programas paralelos, cada processo deve processar um conjunto menor de dados, em relação ao programa sequencial. Considerando isso e o fato de que os dados podem ser lidos mais rapidamente da memória cache que da memória principal, a computação pode ser acelerada, propiciando maiores valores de *speedup*.

2.4 Paralelismo em otimização contínua

Diversos profissionais lidam diariamente com problemas de otimização. Seja reduzir o custo de uma linha de produção, projetar estruturas mecânicas, ou identificar estruturas protéicas, o que interessa é obter uma solução que minimize ou maximize determinados fatores.

Um problema de otimização contínua pode ser genericamente definido da seguinte forma:

$$\begin{array}{ll} \text{minimizar} & f(x) \\ \text{s.a} & \\ & g_i(x) \leq 0 \quad \text{para } i = 1, \dots, m \\ & h_i(x) = 0 \quad \text{para } i = 1, \dots, q \\ & \ell \leq x \leq u \end{array}$$

onde $f, g_1, \dots, g_m, h_1, \dots, h_q$ são funções definidas em R^n , x é um vetor de n componentes e $\ell, u \in R^n$ são respectivamente os limitantes superiores e inferiores para x .

O objetivo do problema acima é encontrar variáveis $x_1, \dots, x_n$ que satisfaçam $g_i(x) \leq 0$, $h_i(x) = 0$ e $\ell \leq x \leq u$ e que minimizem a função f , conhecida por função objetivo, ou função de mérito. As inequações $g_i(x)$ e as equações $h_i(x)$ são chamadas de restrições de desigualdade e igualdade respectivamente. As restrições $\ell \leq x \leq u$ são chamadas restrições de caixa. Os pontos x que as satisfazem são chamados de soluções viáveis. O problema de otimização acima pode ser

compreendido como um problema de encontrar um ponto viável x^* tal que $f(x^*) \leq f(x)$ para todo ponto viável x .

2.4.1 Possibilidades de paralelização

A motivação por trás do interesse em computação paralela aplicada em otimização contínua está em resolver, de forma eficiente, problemas que são computacionalmente caros. Robert Schnabel, em seu artigo [Sch95], explica que em geral a origem do grande consumo de processamento dos algoritmos de otimização pode estar:

- No custo individual de avaliação da função objetivo, restrições e/ou suas derivadas;
- Na quantidade de restrições ou variáveis que, caso sejam muito grandes, podem aumentar consideravelmente o custo de cada iteração;
- Na necessidade de realizar muitas avaliações das funções ou suas derivadas;
- Na necessidade de realizar muitas iterações do método.

Segundo [Sch95], essas fontes de consumo de processamento nos levam à três maneiras de paralelizar esse tipo de algoritmo:

- Paralelizam-se as operações de álgebra linear utilizadas pelo algoritmo: Já existe bastante trabalho envolvendo paralelismo e álgebra linear, inclusive existem pacotes gratuitos que implementam algoritmos paralelos para diversas operações (PBLAS [PBL], LAPACK [PLA], ScaLAPACK [Sca], entre outros).
- Paraleliza-se a avaliação da função objetivo, restrições e/ou suas derivadas: Essa opção é dependente do problema que se quer resolver. Necessita que a função objetivo e/ou restrições possuam uma estrutura capaz de ser paralelizada.
- Paraleliza-se um método de otimização: Obter um método para resolver o problema de otimização em sua forma genérica, ou em alguma outra forma mais específica, é interessante por permitir a resolução de todas as instâncias de problemas formulados de tal maneira. Nesse caso pode-se tentar partir de um método sequencial, buscando oportunidades de ganho de eficiência através de paralelismo, ou tentar desenvolver um método completamente novo e intrinsecamente paralelo.

Nesta dissertação propomos investigar duas das três formas que Schnabel sugere para paralelizar algoritmos de otimização contínua. A primeira consiste na paralelização da avaliação da função objetivo de um problema de empacotamento de moléculas resolvido através de um método de otimização. A segunda consiste na paralelização da estrutura de um método de otimização aplicado a resolução de um problema inverso para estimar constantes óticas de filmes finos.

Capítulo 3

PUMA

O PUMA (*Pointwise Unconstrained Minimization Approach*) [BCM99],[Poi] é um método que recupera os valores de espessuras e constantes óticas de filmes finos. a partir de medições da transmitância de um sistema composto por filmes e um substrato transparente.

Um filme fino é uma fina camada de material de espessura normalmente inferior a um micron. Um sistema de filmes finos é dado por camadas de filmes finos sobrepostos sob e/ou sobre um substrato. A medida de transmitância é a relação entre a quantidade de luz que atravessa o sistema e a quantidade de luz incidente. A transmitância pode ser obtida com o uso de um espectrofotômetro. Veja a Figura 3.1.

Figura 3.1: Em (a) temos a ilustração de um sistema de filmes finos com dois filmes na configuração filme-substrato-filme. Em (b) temos um exemplo de como é feita a medição da transmitância.

As constantes óticas que são recuperadas pelo método são o índice de refração e os coeficientes de absorção de cada filme no sistema. Esses valores são de extrema importância em diversas aplicações modernas de filmes finos, como a produção de semicondutores, revestimentos de diversos materiais, biotecnologia e a geração e conservação de energia (painéis solares) [OM02].

A grande inovação do PUMA é permitir, a partir de um modelo de programação não-linear, a recuperação dessas constantes independentemente da existência de qualquer padrão de interferência, o que não ocorria com as soluções propostas anteriormente [MGF76], [Swa83], [Swa84]. A maior

desvantagem do PUMA é o custo computacional exigido para resolver o problema de programação não-linear. Nosso objetivo aqui foi o de permitir, através da paralelização do método, a identificação de constantes de sistemas com vários filmes. Tal recuperação não era possível antes, dado o tempo de processamento consumido por algumas instâncias que chegava a cerca de um mês. Isso tornava impraticável sua avaliação e uso.

Os tópicos abordados neste capítulo são: o modelo matemático do problema, a técnica de otimização utilizada, a estratégia de paralelização e, por fim, diversos experimentos numéricos ilustrando os resultados obtidos.

3.1 Modelo matemático do problema

Fisicamente estamos lidando com uma pilha de filmes finos depositados em ambos os lados de um substrato transparente. Definimos como “topo” da pilha o lado em que incide a luz. Claramente, a “base” é o lado oposto. Medidas da transmitância são realizadas utilizando-se um espectrofotômetro. Essas medidas estão disponíveis para um intervalo de comprimento de onda que varia de $\lambda_{\min}$ até $\lambda_{\max}$ nanômetros.

O índice de refração s do substrato é conhecido. Sua espessura também, porém, ela não influencia os cálculos uma vez que o substrato é transparente e sua espessura é muito maior que a dos filmes [CM01].

Para cada filme temos como valores desconhecidos: a espessura, o índice de refração e o coeficiente de atenuação. Esses dois últimos são funções do comprimento de onda.

Considere que $m_t \geq 0$ filmes estão depositados acima do substrato e $m_b \geq 0$ abaixo dele. Para cada $i = 1, \dots, m_t$, $\lambda \in [\lambda_{\min}, \lambda_{\max}]$ denotamos:

$$\begin{aligned} d_i^t &= \text{espessura do filme } i \text{ acima do substrato,} \\ n_i^t(\lambda) &= \text{índice de refração do filme } i \text{ acima do substrato,} \\ \kappa_i^t(\lambda) &= \text{coeficiente de atenuação do filme } i \text{ acima do substrato.} \end{aligned}$$

Analogamente, para cada $i = 1, \dots, m_b$, $\lambda \in [\lambda_{\min}, \lambda_{\max}]$ denotamos:

$$\begin{aligned} d_i^b &= \text{espessura do filme } i \text{ abaixo do substrato,} \\ n_i^b(\lambda) &= \text{índice de refração do filme } i \text{ abaixo do substrato,} \\ \kappa_i^b(\lambda) &= \text{coeficiente de atenuação do filme } i \text{ abaixo do substrato.} \end{aligned}$$

Para cada comprimento de onda λ temos que, a transmitância teórica é função da espessura, índices de refração e coeficientes de atenuação. Assim:

$$T_{\text{teórica}}(\lambda) = T(s(\lambda), \{d_i^t\}_{i=1}^{m_t}, \{n_i^t(\lambda)\}_{i=1}^{m_t}, \{\kappa_i^t(\lambda)\}_{i=1}^{m_t}, \{d_i^b\}_{i=1}^{m_b}, \{n_i^b(\lambda)\}_{i=1}^{m_b}, \{\kappa_i^b(\lambda)\}_{i=1}^{m_b}), \quad (3.1)$$

onde T representa a transmitância.

A fim de simplificar a notação façamos:

$$\begin{aligned} d_{\text{topo}} &= \{d_i^t\}_{i=1}^{m_t}, & d_{\text{base}} &= \{d_i^b\}_{i=1}^{m_b} \\ n_{\text{topo}} &= \{n_i^t\}_{i=1}^{m_t}, & n_{\text{base}} &= \{n_i^b\}_{i=1}^{m_b} \\ \kappa_{\text{topo}} &= \{\kappa_i^t\}_{i=1}^{m_t}, & \kappa_{\text{base}} &= \{\kappa_i^b\}_{i=1}^{m_b} \end{aligned}$$

e:

$$d_{\text{filmes}} = \{d_{\text{topo}}, d_{\text{base}}\}, \quad n_{\text{filmes}} = \{n_{\text{topo}}, n_{\text{base}}\}, \quad \kappa_{\text{filmes}} = \{\kappa_{\text{topo}}, \kappa_{\text{base}}\}.$$

Assim, podemos reescrever (3.1) como:

$$T_{\text{teórica}}(\lambda) = T(s(\lambda), d_{\text{filmes}}, n_{\text{filmes}}(\lambda), \kappa_{\text{filmes}}(\lambda)).$$

Se $m_t = 1$ e $m_b = 0$, a fórmula teórica da transmitância é a fornecida e usada em [BCM99] para recuperar parâmetros de um único filme depositado sobre um substrato transparente. Neste trabalho usamos a fórmula introduzida em [Ven05]. A transmitância de um par de filmes diferentes depositados sobre um substrato transparente pode ser expressada de forma compacta. Se dois filmes forem depositados na mesma face do substrato, a fórmula compacta da transmitância é dada por:

$$T_{\text{FFS}} = \frac{64s(n_1^2 + \kappa_1^2)(n_2^2 + \kappa_2^2)x_1x_2}{x_1x_2(Ax_1x_2 + Bx_1 + Cx_2 + D + E) + x_1(Fx_1 + G) + x_2(Hx_2 + I) + J},$$

onde

$$\begin{aligned} A &= -[(n_1 - 1)^2 + \kappa_1^2][(n_2 + n_1)^2 + (\kappa_2 + \kappa_1)^2][(s^2 - n_2)(n_2 - 1) - \kappa_2^2], \\ B &= 2[(n_1 - 1)^2 + \kappa_1^2][B_s \sin(\varphi_2) - B_c \cos(\varphi_2)], \\ B_s &= \kappa_2(s^2 + 1)(n_2^2 - n_1^2 + \kappa_2^2 - \kappa_1^2) - 2(n_2^2 + \kappa_2^2 - s^2)(\kappa_2 n_1 - \kappa_1 n_2), \\ B_c &= (\kappa_2^2 + n_2^2 - s^2)(n_2^2 - n_1^2 + \kappa_2^2 - \kappa_1^2) + 2\kappa_2(s^2 + 1)(\kappa_2 n_1 - \kappa_1 n_2), \\ C &= 2[(s^2 - n_2)(n_2 - 1) - \kappa_2^2][2C_s \sin(\varphi_1) - C_c \cos(\varphi_1)], \\ C_s &= \kappa_1[(n_1^2 + \kappa_1^2 + n_2)(n_2 - 1) + \kappa_2^2] - \kappa_2 n_1(n_1^2 + \kappa_1^2 - 1), \\ C_c &= (n_1^2 + \kappa_1^2 - 1)(n_2^2 - n_1^2 + \kappa_2^2 - \kappa_1^2) + 4\kappa_1(\kappa_2 n_1 - \kappa_1 n_2), \\ D &= 2[(n_2 + n_1)^2 + (\kappa_2 + \kappa_1)^2][D_s \sin(\varphi_1 + \varphi_2) + D_c \cos(\varphi_1 + \varphi_2)], \\ D_s &= 2\kappa_1(n_2^2 + \kappa_2^2 - s^2) + \kappa_2(s^2 + 1)(n_1^2 + \kappa_1^2 - 1), \\ D_c &= 2\kappa_1\kappa_2(s^2 + 1) - (n_2^2 + \kappa_2^2 - s^2)(n_1^2 + \kappa_1^2 - 1), \\ E &= 2[(n_2 - n_1)^2 + (\kappa_2 - \kappa_1)^2][E_s \sin(\varphi_2 - \varphi_1) - E_c \cos(\varphi_2 - \varphi_1)], \\ E_s &= \kappa_2(s^2 + 1)(n_1^2 + \kappa_1^2 - 1) - 2\kappa_1(n_2^2 + \kappa_2^2 - s^2), \\ E_c &= 2\kappa_1\kappa_2(s^2 + 1) + (n_2^2 + \kappa_2^2 - s^2)(n_1^2 + \kappa_1^2 - 1), \\ F &= [(n_1 - 1)^2 + \kappa_1^2][(n_2 - n_1)^2 + (\kappa_2 - \kappa_1)^2][(s^2 + n_2)(n_2 + 1) + \kappa_2^2], \\ G &= -2[(s^2 + n_2)(n_2 + 1) + \kappa_2^2][2G_s \sin(\varphi_1) - G_c \cos(\varphi_1)], \\ G_s &= \kappa_1(n_2^2 - n_1^2 + \kappa_2^2 - \kappa_1^2) + (\kappa_2 n_1 - \kappa_1 n_2)(n_1^2 + \kappa_1^2 - 1), \\ G_c &= -4\kappa_1(\kappa_2 n_1 - \kappa_1 n_2) + (n_2^2 - n_1^2 + \kappa_2^2 - \kappa_1^2)(n_1^2 + \kappa_1^2 - 1), \\ H &= -[(n_1 + 1)^2 + \kappa_1^2][(n_2 - n_1)^2 + (\kappa_2 - \kappa_1)^2][(s^2 - n_2)(n_2 - 1) - \kappa_2^2], \\ I &= 2[(n_1 + 1)^2 + \kappa_1^2][I_s \sin(\varphi_2) + I_c \cos(\varphi_2)], \\ I_s &= 2(n_2^2 + \kappa_2^2 - s^2)(\kappa_2 n_1 - \kappa_1 n_2) + (s^2 + 1)\kappa_2(n_2^2 - n_1^2 + \kappa_2^2 - \kappa_1^2), \\ I_c &= 2\kappa_2(s^2 + 1)(\kappa_2 n_1 - \kappa_1 n_2) - (n_2^2 + \kappa_2^2 - s^2)(n_2^2 - n_1^2 + \kappa_2^2 - \kappa_1^2), \\ J &= [(n_1 + 1)^2 + \kappa_1^2][(n_2 + n_1)^2 + (\kappa_2 + \kappa_1)^2][(s^2 + n_2)(n_2 + 1) + \kappa_2^2], \end{aligned}$$

e

$$\beta_1 = \frac{4\pi d_1}{\lambda}, \quad \beta_2 = \frac{4\pi d_2}{\lambda}, \quad \varphi_1 = \beta_1 n_1, \quad \varphi_2 = \beta_2 n_2, \quad x_1 = \exp(-\beta_1 \kappa_1), \quad x_2 = \exp(-\beta_2 \kappa_2).$$

Se dois filmes são depositados em lados diferentes do substrato, a fórmula compacta para a transmitância é dada por:

$$T_{\text{FSF}} = \frac{64s(n_1^2 + \kappa_1^2)(n_2^2 + \kappa_2^2)x_1x_2}{x_1x_2(Ax_1x_2 + Bx_1 + Cx_2 + D + E) + x_1(Fx_1 + G) + x_2(Hx_2 + I) + J},$$

onde

$$\begin{aligned} A &= -[(n_1 - 1)^2 + \kappa_1^2][(n_2 - 1)^2 + \kappa_2^2][n_2(n_1^2 + \kappa_1^2 + s^2) + n_1(n_2^2 + \kappa_2^2 + s^2)], \\ B &= 2[(n_1 - 1)^2 + \kappa_1^2][B_s \sin(\varphi_2) + B_c \cos(\varphi_2)], \\ B_s &= \kappa_2[2n_1(s^2 - n_2^2 - \kappa_2^2) + (n_1^2 + \kappa_1^2 + s^2)(n_2^2 + \kappa_2^2 - 1)], \\ B_c &= n_1(n_2^2 + \kappa_2^2 - 1)(n_2^2 + \kappa_2^2 - s^2) + 2\kappa_2^2(n_1^2 + \kappa_1^2 + s^2), \\ C &= 2[(n_2 - 1)^2 + \kappa_2^2][C_s \sin(\varphi_1) + C_c \cos(\varphi_1)], \\ C_s &= \kappa_1[(n_2^2 + \kappa_2^2 + s^2)(n_1^2 + \kappa_1^2 - 1) - 2n_2(n_1^2 + \kappa_1^2 - s^2)], \\ C_c &= 2\kappa_1^2(n_2^2 + \kappa_2^2 + s^2) + n_2(n_1^2 + \kappa_1^2 - s^2)(n_1^2 + \kappa_1^2 - 1), \\ D &= 2[\kappa_2(n_1^2 + \kappa_1^2 - s^2) + \kappa_1(n_2^2 + \kappa_2^2 - s^2)][D_s \sin(\varphi_1 + \varphi_2) - 2D_c \cos(\varphi_2 + \varphi_1)], \\ D_s &= 4\kappa_1\kappa_2 - (n_1^2 + \kappa_1^2 - 1)(n_2^2 + \kappa_2^2 - 1), \\ D_c &= \kappa_2(n_1^2 - 1) + \kappa_1(n_2^2 - 1) + \kappa_1\kappa_2(\kappa_1 + \kappa_2), \\ E &= 2[\kappa_1(n_2^2 + \kappa_2^2 - s^2) - \kappa_2(n_1^2 + \kappa_1^2 - s^2)][E_s \sin(\varphi_2 - \varphi_1) - 2E_c \cos(\varphi_2 - \varphi_1)], \\ E_s &= 4\kappa_1\kappa_2 + (n_2^2 + \kappa_2^2 - 1)(n_1^2 + \kappa_1^2 - 1), \\ E_c &= \kappa_1(n_2^2 - 1) - \kappa_2(n_1^2 - 1) + \kappa_1\kappa_2(\kappa_2 - \kappa_1), \\ F &= [(n_1 - 1)^2 + \kappa_1^2][(n_2 + 1)^2 + \kappa_2^2][n_2(n_1^2 + \kappa_1^2 + s^2) - n_1(n_2^2 + \kappa_2^2 + s^2)], \\ G &= -2[(n_2 + 1)^2 + \kappa_2^2][G_s \sin(\varphi_1) + G_c \cos(\varphi_1)], \\ G_s &= -\kappa_1[2n_2(n_1^2 + \kappa_1^2 - s^2) + (n_2^2 + \kappa_2^2 + s^2)(n_1^2 + \kappa_1^2 - 1)], \\ G_c &= n_2(n_1^2 + \kappa_1^2 - s^2)(n_1^2 + \kappa_1^2 - 1) - 2\kappa_1^2(n_2^2 + \kappa_2^2 + s^2), \\ H &= -[(n_1 + 1)^2 + \kappa_1^2][(n_2 - 1)^2 + \kappa_2^2][n_2(n_1^2 + \kappa_1^2 + s^2) - n_1(n_2^2 + \kappa_2^2 + s^2)], \\ I &= -2[(n_1 + 1)^2 + \kappa_1^2][I_s \sin(\varphi_2) + I_c \cos(\varphi_2)], \\ I_s &= -\kappa_2[2n_1(n_2^2 + \kappa_2^2 - s^2) + (n_1^2 + \kappa_1^2 + s^2)(n_2^2 + \kappa_2^2 - 1)], \\ I_c &= n_1(n_2^2 + \kappa_2^2 - s^2)(n_2^2 + \kappa_2^2 - 1) - 2\kappa_2^2(n_1^2 + \kappa_1^2 + s^2), \\ J &= [(n_1 + 1)^2 + \kappa_1^2][(n_2 + 1)^2 + \kappa_2^2][n_2(n_1^2 + \kappa_1^2 + s^2) + n_1(n_2^2 + \kappa_2^2 + s^2)], \end{aligned}$$

e

$$\beta_1 = \frac{4\pi d_1}{\lambda}, \quad \beta_2 = \frac{4\pi d_2}{\lambda}, \quad \varphi_1 = \beta_1 n_1, \quad \varphi_2 = \beta_2 n_2, \quad x_1 = \exp(-\beta_1 \kappa_1), \quad x_2 = \exp(-\beta_2 \kappa_2).$$

Ambas as formulações equivalem à formulação de Swanepoel [Swa83] quando zeramos a espessura de um dos filmes. Usando o espectrofotômetro obtemos as medidas $T_{\text{medida}}(\lambda_i)$ para comprimentos de onda λ_i , $i = 1, \dots, N$, onde:

$$\lambda_{\min} \leq \lambda_1 < \dots < \lambda_N \leq \lambda_{\max}.$$

Idealmente, para todo $i = 1, \dots, N$, os parâmetros reais devem satisfazer a equação:

$$T_{\text{medida}}(\lambda_i) = T_{\text{teórica}}(s(\lambda_i), d_{\text{filmes}}, n_{\text{filmes}}(\lambda_i), \kappa_{\text{filmes}}(\lambda_i)). \quad (3.2)$$

Entretanto, este é um sistema com $(m_t + m_b)(2N + 1)$ variáveis e apenas N equações, o que o torna indeterminado. Mais ainda, os valores obtidos empiricamente estão sujeitos a erros de medida e o modelo não é completamente adequado à realidade, já que suposições como homogeneidade, transparência total do substrato e paralelismo de interfaces entre os filmes são difíceis de serem satisfeitas exatamente. Os muitos graus de liberdade que são inerentes a esse problema nos levam a introduzir restrições físicas e fenomenológicas que devem ser satisfeitas pelos parâmetros dos filmes em consideração. Veja [BCM99] para a aplicação desta metodologia ao problema de estimação com um único filme. Ao invés de considerarmos o sistema não-linear em (3.2), definimos o seguinte problema de otimização:

$$\begin{aligned} \text{Minimizar} \quad & \sum_{i=1}^M [T_{\text{teórica}}(s(\lambda_i), d_{\text{filmes}}, n_{\text{filmes}}(\lambda_i), \kappa_{\text{filmes}}(\lambda_i)) - T_{\text{medida}}(\lambda_i)]^2 \\ \text{s.a} \quad & \text{Restrições Físicas.} \end{aligned} \quad (3.3)$$

No caso de um único filme, essa estratégia foi usada com sucesso diversas vezes no projeto PUMA [Poi].

Aqui usamos as mesmas restrições do caso com um único filme [BCM99], que são adequadas a filmes finos a-semicondutores amorfos na vizinhança do pico de absorção fundamental dos a-semicondutores. Essas restrições são descritas a seguir. Consideramos que $n(\lambda)$ é o índice de refração e $\kappa(\lambda)$ é o coeficiente de atenuação de um filme genérico na pilha.

RF1: $n(\lambda) \geq 1$ e $\kappa(\lambda) \geq 0$ para todo $\lambda \in [\lambda_{\min}, \lambda_{\max}]$;

RF2: $n(\lambda)$ e $\kappa(\lambda)$ são funções decrescentes em λ ;

RF3: $n(\lambda)$ é convexo;

RF4: Existe $\lambda_{\text{infl}} \in [\lambda_{\min}, \lambda_{\max}]$ tal que $\kappa(\lambda)$ é convexo se $\lambda \geq \lambda_{\text{infl}}$ e côncavo se $\lambda < \lambda_{\text{infl}}$.

Em [BCM99] é mostrado que **RF1–RF4** são satisfeitas se, e somente se,

$$n(\lambda_{\max}) \geq 1, \quad \kappa(\lambda_{\max}) \geq 0, \quad (3.4)$$

$$n'(\lambda_{\max}) \leq 0, \quad \kappa'(\lambda_{\max}) \leq 0, \quad (3.5)$$

$$n''(\lambda) \geq 0 \text{ para todo } \lambda \in [\lambda_{\min}, \lambda_{\max}], \quad (3.6)$$

$$\kappa''(\lambda) \geq 0 \text{ para todo } \lambda \in [\lambda_{\text{infl}}, \lambda_{\max}], \quad (3.7)$$

$$\kappa''(\lambda) \leq 0 \text{ para todo } \lambda \in [\lambda_{\min}, \lambda_{\text{infl}}], \text{ e} \quad (3.8)$$

$$\kappa'(\lambda_{\min}) \leq 0. \quad (3.9)$$

Como em [BCM99], eliminamos as restrições do problema realizando uma troca de variáveis:

$$n(\lambda_{\max}) = 1 + u^2, \quad \kappa(\lambda_{\max}) = v^2, \quad (3.10)$$

$$n'(\lambda_{\max}) = -u_1^2, \quad \kappa'(\lambda_{\max}) = -v_1^2, \quad (3.11)$$

$$n''(\lambda) = w(\lambda)^2 \text{ para todo } \lambda \in [\lambda_{\min}, \lambda_{\max}], \quad (3.12)$$

$$\kappa''(\lambda) = z(\lambda)^2 \text{ para todo } \lambda \in [\lambda_{\text{infl}}, \lambda_{\max}], \text{ e} \quad (3.13)$$

$$\kappa''(\lambda) = -z(\lambda)^2 \text{ para todo } \lambda \in [\lambda_{\min}, \lambda_{\text{infl}}]. \quad (3.14)$$

Considerando um cenário real, onde os dados são fornecidos por um conjunto de N pontos igualmente espaçados no intervalo $[\lambda_{\min}, \lambda_{\max}]$, definimos:

$$h = (\lambda_{\max} - \lambda_{\min})/(N - 1), \text{ e}$$

$$\lambda_i = \lambda_{\min} + (i - 1)h \text{ para } i = 1, \dots, N.$$

Usaremos a notação n_i , κ_i , w_i e z_i para as estimativas de $n(\lambda_i)$, $\kappa(\lambda_i)$, $w(\lambda_i)$ e $z(\lambda_i)$, para todo $i = 1, \dots, N$. Discretizando (3.10–3.14) obtemos:

$$n_N = 1 + u^2, \quad v_N = v^2, \quad (3.15)$$

$$n_{N-1} = n_N + u_1^2 h, \quad \kappa_{N-1} = \kappa_N + v_1^2 h, \quad (3.16)$$

$$n_i = w_i^2 h^2 + 2n_{i+1} - n_{i+2} \text{ para } i = 1, \dots, N - 2, \quad (3.17)$$

$$\kappa_i = z_i^2 h^2 + 2\kappa_{i+1} - \kappa_{i+2}, \quad \text{se } \lambda_{i+1} \geq \lambda_{\text{infl}}, \text{ e} \quad (3.18)$$

$$\kappa_i = -z_i^2 h^2 + 2\kappa_{i+1} - \kappa_{i+2}, \quad \text{se } \lambda_{i+1} < \lambda_{\text{infl}}. \quad (3.19)$$

É importante frisar que os índices de refração e os coeficientes de absorção de todos os filmes devem satisfazer essas restrições.

3.2 Técnica de otimização

O problema (3.3) com as restrições definidas por (3.15–3.19) é um problema de otimização com $(m_t + m_b)(2N + 1)$ variáveis.

As restrições são representadas pela não-negatividade das espessuras e o fato de que os pontos de inflexão λ_{infl} devem estar no intervalo $[\lambda_{\text{min}}, \lambda_{\text{max}}]$. As demais variáveis são irrestritas e adimensionais. Essas são as razões principais para tratá-las de forma diferente das espessuras e pontos de inflexão. Nossa estratégia é definir, para cada conjunto de pontos de inflexão e cada conjunto de espessuras, um problema de otimização contínua irrestrito, onde as variáveis são definidas pelos índices de refração e coeficientes de atenuação. Como em [BCM99], o problema será resolvido usando o método do gradiente espectral [Ray97]. Uma abstração de alto-nível da estratégia de solução é dada a seguir:

Algoritmo 3.1

Passo 1: Definir uma grade de granularidade grossa com respeito às variáveis *espessura* e *ponto de inflexão*.

Passo 2: Resolver o problema (3.3),(3.15)-(3.19) para cada ponto da grade onde as variáveis *espessura* e *ponto de inflexão* estão fixadas. Obter o conjunto de espessuras e o melhor conjunto de pontos de inflexão da grade de forma que, depois de resolver os problemas de otimização, tenha-se a menor soma quadrática.

Passo 3: Definir uma nova grade, de granularidade menor, na vizinhança das melhores espessuras e pontos de inflexão obtidos no **Passo 2**.

Passo 4: Resolver os problemas de otimização definidos por cada ponto da nova grade. Adotar como solução aqueles valores que fornecerem a menor soma quadrática.

Em sistemas com dois filmes, a grade inicial para a espessura é dada pelos pontos na caixa $[d_{\text{min}}^1, d_{\text{max}}^1] \times [d_{\text{min}}^2, d_{\text{max}}^2]$, onde d_{min}^i e d_{max}^i são (para o i -ésimo filme) os limitantes inferior e superior para a espessura do filme.

Com relação ao ponto de inflexão, existem duas opções. A primeira consiste em fixar o ponto de inflexão no valor de λ_{min} e deve ser usada caso o coeficiente de absorção seja convexo no intervalo considerado. A segunda, consiste em agir de forma análoga ao caso da espessura. Constrói-se uma grade de alta granularidade dada pelos pontos na caixa $[Q_{\text{min}}^1, Q_{\text{max}}^1] \times [Q_{\text{min}}^2, Q_{\text{max}}^2]$, onde Q_{min}^i e Q_{max}^i são (para o i -ésimo filme) os limitantes inferior e superior para os pontos de inflexão, respectivamente.

Vale mencionar que de acordo com a estratégia adotada pelo PUMA [Poi], o problema de minimização (3.3) deve ser resolvido para cada conjunto $(d_{\text{teste}}^1, d_{\text{teste}}^2, Q_{\text{teste}}^1, Q_{\text{teste}}^2)$. Mais ainda, para cada conjunto de espessuras e pontos de inflexão, o método de otimização necessita de estimativas iniciais para $n(\lambda)$ e $\kappa(\lambda)$. De fato, o método resolve o problema de otimização partindo

de diversas aproximações iniciais para $n(\lambda)$ e $\kappa(\lambda)$. A geração de cada estimativa inicial é baseada na aproximação de $n(\lambda)$ e $\kappa(\lambda)$ por funções lineares por partes e segue um processo similar ao explicado em [BCM99]. A maior diferença fica na geração da aproximação do $\kappa(\lambda)$. Enquanto no trabalho [BCM99], a aproximação era feita por dois segmentos de reta, aqui usamos três segmentos, conseguindo uma melhor aproximação da função real (veja a Figura 3.2).

Figura 3.2: Ilustração da aproximação do κ do silício por funções lineares por partes (a) usando 2 segmentos e (b) usando 3 segmentos

3.3 Estratégia de paralelização

A metodologia de solução adotada demanda um esforço computacional muito grande, que aumenta exponencialmente de acordo com a quantidade de filmes presentes no sistema. Examinaremos a quantidade de vezes que o problema (3.3),(3.15–3.19) é resolvido no caso de um sistema com dois filmes, com espessuras 100nm e 600nm, respectivamente. Ao variarmos as espessuras entre 10nm e 200nm e entre 300nm e 900nm, respectivamente (com passo de 10nm em ambos os casos), resultando em 420 pares de espessuras. Considerando o intervalo espectral [1000,2000]nm teremos 11 pontos de inflexão (obtidos com um tamanho de passo de 100nm) para cada filme. Isso resulta em 50.820 instâncias do problema (3.3),(3.15–3.19). Como cada instância é resolvida para 6 pares de estimativas iniciais [BCM99] diferentes de $n(\lambda)$ e $\kappa(\lambda)$ para cada filme, teremos 1.829.520 processos de otimização. Assim, a paralelização do método é essencial para que sua implementação tenha uma utilidade prática.

A estratégia de paralelização é bem simples e consiste em distribuir a resolução dos diversos problemas, obtidos variando-se os parâmetros de (3.3),(3.15)–(3.19), entre os processadores disponíveis. Por tratar-se de problemas independentes, praticamente inexistente comunicação entre processos, exceto pela necessidade de informar o resultado de cada problema a um processo coordenador, que escolherá o resultado adequado como solução do método. Tal estrutura é conhecida como *bag-of-tasks* ou sacola de tarefas.

Figura 3.3: Funcionamento do PUMA paralelo entre o processo mestre (1) e um escravo (3).

O modelo de paralelismo utilizado foi o de troca de mensagens. Esse modelo é bastante apropriado para aplicações *bag-of-tasks*, já que existe um paralelismo de alta granularidade e pouca comunicação entre tarefas, maximizando a eficiência da paralelização.

Considere que temos $\mathcal{P}$ processos, numerados de 1 até $\mathcal{P}$. O processo 1 é responsável por gerar e distribuir os problemas, bem como armazenar o melhor resultado. Os demais processos têm a tarefa de resolver os problemas e fornecer seus resultados ao processo 1. Mais detalhadamente, o processo 1 lê os parâmetros de entrada do algoritmo (para maiores detalhes sobre os parâmetros de entrada, veja [Poi]) e gera as diferentes combinações de espessura, pontos de inflexão, $n(\lambda)$ e $\kappa(\lambda)$ que caracterizam cada problema. Cada processo $\rho = 2, \dots, \mathcal{P}$ requisita uma instância do problema (3.3),(3.15–3.19) ao processo 1, resolve e envia o valor do erro quadrático e os demais valores que caracterizam o ponto ótimo ao processo 1. Ao receber um resultado de um processo ρ , o processo 1 compara o valor do erro quadrático recebido com o menor valor já obtido até então. Se o resultado recebido for menor, é mantido (substitui-se a antiga melhor solução pela recebida), caso contrário é descartado. A Figura 3.3 ilustra o funcionamento da versão paralela do PUMA.

Pode-se argumentar que o processador onde o processo 1 será executado permanecerá a maior parte de seu tempo de execução ocioso, desperdiçando recursos computacionais. Para contornar esse problema, pode-se iniciar a aplicação com $\mathcal{P} = N_{proc} + 1$ processos, onde N_{proc} é a quantidade de processadores disponíveis e iniciar o processo adicional no mesmo processador em que o processo 1 foi iniciado.

3.4 Resultados numéricos

Através da versão paralela do PUMA foi possível recuperar as contantes óticas e espessuras de sistemas compostos por dois filmes finos, feito impraticável de ser realizado antes deste trabalho. Os experimentos apresentados a seguir foram realizados em três ambientes diferentes. Utilizamos a grade do LCPD (Laboratório de Computação Paralela e Distribuída), o aglomerado Revoada do grupo InteGrade e o aglomerado do grupo de otimização contínua.

A grade do LCPD é composta por sete máquinas. Uma máquina Intel(R) Xeon(TM) CPU 3.00GHz de 32 bits com Hyper-threading e 2GB de RAM. Seis máquinas AMD Athlon(tm) XP 2800+ 2.25GHz de 32 bits com 1GB de RAM.

O aglomerado Revoada possui oito máquinas com processador Intel(R) Pentium(R) 4 3.00GHz de 32 bits com 1GB de RAM. O aglomerado do grupo de otimização contínua possui quatro máquinas com processador Intel(R) Core 2 Quad 2.40GHz de 64 bits e 4 GB de RAM.

Alguns testes anteriores, com a intenção de calibrar o método, foram realizados na grade do projeto GRID 5000 ([GRI]). Utilizamos também, para a realização de testes preliminares, o cluster Alcatéia do LCCA (Laboratório de Computação Científica Avançada), que conta com 32 nós Intel Xeon 2.4 GHz biprocessadas e com 2.5GB de memória RAM por nó.

O PUMA paralelo foi implementado em linguagem C, usando funções da *Message Passing Interface*. Os binários utilizados em nossos experimentos foram obtidos através do compilador gcc (versão 4.2.1, parte integrante da projeto GNU *Compiler Collection*) e ligados com a biblioteca LAM MPI (versão 7.1.2). Nosso código também foi testado com a biblioteca MPICH2 (versão 1.4 e versão 1.5 modificada para o *middleware* InteGrade [Int]).

3.4.1 Filmes gerados por computador

Em nossos experimentos, consideramos que $\lambda_{\text{infl}} = \lambda_{\text{min}}$. Denominaremos de A, B, C e D os filmes gerados computacionalmente e introduzidos em [BCM99]. Suas constantes óticas podem ser encontradas no Anexo A. Em particular, temos:

Filme A: Este filme simula um filme fino de silício (a-Si:H) com espessura real de 100nm.

Filme B: Este filme é idêntico ao filme A, com espessura real de 600nm.

Filme C: Este filme simula um filme fino de germânio (a-Ge:H) com espessura real de 100nm.

Filme D: Este filme é idêntico ao filme C, com espessura real de 600nm.

Trabalhamos com sistemas de dois filmes e um substrato de vidro. Referiremos ao substrato de vidro usando a letra v , assim, $A \vee B$ será um sistema composto por: Filme A, substrato de vidro e Filme B. Relataremos alguns resultados obtidos com sistemas de filmes idênticos (filme 1 = filme 2) e com sistemas de filmes diferentes (filme 1 $\neq$ filme 2).

Sistemas com filmes *gedanken* de mesmo material e espessura

Para os experimentos com filmes idênticos gerados computacionalmente, usamos o filme A e consideramos as seguintes configurações: AAv, AvA, vAA. O intervalo espectral utilizado no processo de recuperação foi $570 - 1560nm$. O valor de λ_{min} corresponde ao menor comprimento de onda tal que a transmitância medida do sistema é maior ou igual a 10^{-4} . O valor de λ_{max} corresponde a $\lambda_{min} + 990nm$, de forma que possamos considerar 100 medidas da transmitância com $10nm$ de distância entre cada uma delas.

Nossas três configurações (AAv, AvA e vAA) podem ser resumidas em dois casos: Filmes depositados do mesmo lado do substrato (AAv e vAA) e filmes depositados em lados diferentes do substrato (AvA). Quando os filmes estão depositados do mesmo lado do substrato, o PUMA encontra diversos pares de espessuras d_1 e d_2 com $d_1 + d_2 \approx 200nm$ correspondentes a um sistema com a transmitância próxima à medida. Veja a Figura 3.4 referente à configuração AAv. No caso do sistema com um filme de cada lado do substrato, a indeterminação, embora existente, não está relacionada com a soma das espessuras dos dois filmes de silício, como pode ser observado na Figura 3.5. Dependendo da espessura recuperada, n e κ se ajustam para definir precisamente a transmitância no intervalo de comprimento de onda considerado. Por outro lado, caso seja informada a espessura real de um dos filmes, ou informado que ambos os filmes possuem a mesma espessura o PUMA é capaz de recuperar os valores esperados em ambos os casos.

A Figura 3.6 mostra que se fosse utilizada como informação adicional uma das seguintes: $d_1 = 100nm$, $d_2 = 100nm$ ou $d_1 = d_2$, a indeterminação seria removida. A Figura 3.8 mostra os valores recuperados usando $d_1 = d_2$. É interessante notar que o método recupera uma aproximação razoável do índice de refração de ambos os filmes. Fizemos uma média dos valores do índice de refração recuperados para ambos os filmes e notamos que os valores obtidos representam uma recuperação mais precisa da função verdadeira.

Usando a informação adicional de que $d_1 = d_2$ com as recuperações envolvendo os sistemas AvA e vAA obtivemos recuperações similares ao caso AAv. A Tabela 3.1 mostra as espessuras recuperadas e os erros quadráticos correspondentes para os experimentos com os três sistemas.

O motivo da recuperação das espessuras e constantes óticas ser ruim para o caso AvA é explicado na Figura 3.4.1. Nela podemos observar a variação da função de transmitância em relação às espessuras dos filmes. Note a pequena variação dos valores de transmitância com a alteração nas espessuras para o caso de filmes de mesma natureza.

A razão da recuperação não funcionar bem para os casos AAv e vAA deve-se também a um fenômeno relacionado com as transmitâncias desses sistemas. Para casos em que dois ou mais filmes são sobrepostos no mesmo lado de um substrato, os valores de transmitância obtidos são idênticos aos obtidos no caso de termos um único filme, na mesma posição, com espessura igual à soma das espessuras dos filmes sobrepostos. Dessa forma, a transmitância do sistema AAv, por exemplo, é idêntica à transmitância do sistema com um único filme de silício com $200nm$ posicionado acima do substrato. Mais ainda, é igual à transmitância de qualquer sistema com dois filmes de silício sobrepostos acima do substrato cuja soma de espessuras igual à $200nm$.

Figura 3.4: Erro quadrático do processo de minimização em função das espessuras testadas para o primeiro e segundo filme do sistema AAv. O método não foi capaz de recuperar a espessura verdadeira já que valores pequenos e bem próximos aparecem para qualquer par de espessuras que somem aproximadamente $200nm$.

Figura 3.5: Erro quadrático do processo de minimização em função das espessuras testadas para o primeiro e segundo filme do sistema AvA. O método não foi capaz de recuperar a espessura verdadeira já que valores pequenos e bem próximos aparecem para diversos pares de espessuras. Porém, diferentemente do sistema AAv, elas não somam aproximadamente $200nm$. Independentemente disso, adicionando a informação de que a espessura de um dos dois filmes é conhecida, ou que ambos os filmes possuem a mesma espessura, os valores verdadeiros podem ser recuperados.

Figura 3.6: Uma visão do erro quadrático mostrado na Figura (3.4). A esquerda a espessura é variada com passo 10nm, enquanto que a direita esse valor é refinado para 1nm. Os gráficos (a,b) representam o caso em que fixamos a espessura do primeiro filme em 100nm. Os gráficos (c,d) representam o caso em que fixamos a espessura do segundo filme em 100nm. Já os gráficos (e,f) representam o caso em que a informação adicional utilizada é o fato das espessuras serem iguais. Em todos os casos vemos que a informação adicional foi suficiente para a recuperação correta das espessuras de ambos os filmes.

Figura 3.7: Todas as figuras correspondem à sistemas com dois filmes depositados em diferentes lados de um substrato de vidro com espessuras $d_1 + d_2 = 1000\text{nm}$. A primeira “coluna” (gráficos (a), (d) e (g)) corresponde aos filmes com mesmo material (a-Si:H) e mesma espessura ($\bar{d}_1 = \bar{d}_2 = 500\text{nm}$). A segunda coluna corresponde aos filmes com mesmo material (a-Si:H) espessuras diferentes ($\bar{d}_1 = 200\text{nm}$ e $\bar{d}_2 = 800\text{nm}$). A terceira coluna corresponde aos filmes com material diferente (a-Si:H e a-Ge:H) e mesma espessuras ($\bar{d}_1 = \bar{d}_2 = 500\text{nm}$). Os gráficos da primeira “linha” correspondem às transmitâncias geradas variando $d_1 \in \{\bar{d}_1 - 10\text{nm}, \bar{d}_1, \bar{d}_1 + 10\text{nm}\}$ (lembre-se que $d_1 + d_2 = 1000\text{nm}$). As duas “linhas” seguintes correspondem à variação de d_1 em um intervalo maior. Observe que nos casos de filmes com mesmo material e espessuras próximas, a variação na transmitância é menor que nos demais casos. Uma possível explicação para isso é a simetria nos gráficos da transmitância, que não ocorre nos outros casos.

(a)

(b)

(c)

Figura 3.8: Valores verdadeiros e recuperados para (a) a transmitância, (b) o índice de refração e (c) o coeficiente de absorção do sistema de filmes finos AAv. O gráfico (b) mostra que embora a recuperação do índice de refração dos dois filmes seja apenas satisfatória, os valores obtidos fazendo a média dos índices recuperados para ambos os filmes é muito boa.

(a)

(b)

(c)

Figura 3.9: Valores verdadeiros e recuperados para (a) a transmitância, (b) o índice de refração e (c) o coeficiente de absorção do sistema de filmes finos AvA. O gráfico (b) mostra que embora a média dos valores dos índices de refração dos dois filmes seja melhor que as recuperações em separado, os valores obtidos individualmente foram melhores que no caso AA $\bar{v}$.

Sistema	1º filme		2º filme		Erro quadrático
	$d_{\text{verdadeira}}$	$d_{\text{recuperada}}$	$d_{\text{verdadeira}}$	$d_{\text{recuperada}}$	
AAv	100	100	100	100	3.472713e-05
vAA	100	100	100	100	5.314831e-05
AvA	100	100	100	100	5.044862e-05

Tabela 3.1: Recuperações para sistemas de filmes idênticos no intervalo 540-1530nm, usando a informação adicional de que ambos os filmes são idênticos, a fim de resolver o problema da indeterminação.

Sistemas com filmes *gedanken* de espessura e/ou material distintos

Nos testes com filmes diferentes gerados artificialmente, usamos sistemas de dois filmes combinando filmes de silício e germânio depositados em lados opostos de um substrato de vidro. Os sistemas considerados são: AvB, BvD, CvB e CvD. Os intervalos espectrais utilizados no processo de recuperação das espessuras e constantes óticas dos filmes de cada sistema foram 631 – 1621nm, 934 – 1924nm, 817 – 1807nm e 945 – 1935nm, respectivamente.

Nesses sistemas, não houve casos de indeterminação das espessuras como nos sistemas com filmes idênticos. As espessuras são inequivocadamente determinadas e o PUMA é capaz de recuperá-las corretamente. A Figura 3.10 mostra os gráficos das curvas de nível do erro quadrático do processo de minimização em função das espessuras dos filmes. Poder-se-ia argumentar que ainda existe alguma indeterminação nos exemplos mostrados, causada provavelmente por erros de natureza numérica. Apesar disso, a região que possui menor erro quadrático assemelha-se mais a uma “pequena bola” ao redor da espessura verdadeira, bem diferente dos gráficos de erro quadrático obtido para os sistemas com filmes do mesmo tipo mostrados anteriormente (ver Figuras 3.4 e 3.5). As Figuras 3.11, 3.12, 3.13 e 3.14 mostram os valores recuperados para os sistemas AvB, BvD, CvB e CvD, respectivamente. A Tabela 3.2 resume as espessuras recuperadas bem como os valores de erro quadrático correspondentes.

Nos sistemas com filmes de mesmo material e espessuras diferentes (AvB e CvD), o coeficiente de absorção foi bem recuperado para valores altos o suficiente de α para produzir uma variação na transmitância. Os índices de refração de ambos os filmes foram bem recuperados no intervalo considerado.

No sistema de filme de materiais diferentes e mesmas espessuras (BvD) o coeficiente de absorção só só foi recuperados corretamente para o filme de germânio, mais opaco e que possui maior influência na transmitância. Os índices de refração de ambos os filmes foram bem recuperados no intervalo considerado.

No sistema de filme de materiais e espessuras diferentes (CvB) os coeficientes de absorção de ambos os filmes não puderam ser recuperado corretamente. Uma possível explicação para isso é o fato do filme de material mais opaco ser mais fino e dessa forma produzir uma influência pequena e similar à influência causado pelo filme de material mais transparente na transmitância. Os índices de refração de ambos os filmes foram bem recuperados no intervalo considerado.

Sistema	1º filme		2º filme		Erro quadrático
	$d_{\text{verdadeira}}$	$d_{\text{recuperada}}$	$d_{\text{verdadeira}}$	$d_{\text{recuperada}}$	
AvB	100	100	600	600	4.597773e-05
BvD	600	604	600	599	1.237465e-04
CvB	100	118	600	588	1.419527e-06
CvD	100	100	600	599	3.344105e-05

Tabela 3.2: Espessuras e erros quadráticos obtidos para os sistemas AvB, BvD, CvB e CvD.

Figura 3.10: Gráficos do erro quadrático em função das espessuras dos filmes dos sistemas (a) AvB, (b) BvD, (c) CvB e (d) CvD.

(a)

(b)

(c)

Figura 3.11: Valores verdadeiros e recuperados (a) da transmitância, (b) do índice de refração e (c) do coeficiente de absorção do sistema AvB.

(a)

(b)

(c)

Figura 3.12: Valores verdadeiros e recuperados (a) da transmitância, (b) do índice de refração e (c) do coeficiente de absorção do sistema BvD.

(a)

(b)

(c)

Figura 3.13: Valores verdadeiros e recuperados (a) da transmitância, (b) do índice de refração e (c) do coeficiente de absorção do sistema CvB.

(a)

(b)

(c)

Figura 3.14: Valores verdadeiros e recuperados (a) da transmitância, (b) do índice de refração e (c) do coeficiente de absorção do sistema CVD.

3.4.2 Filmes reais

Chamaremos de E e F aos filmes reais cujos dados de transmitância foram obtidos através de um espectrofotômetro:

Filme E: Um filme fino a-Si:H com espessura real de aproximadamente 1150nm

Filme F: Um filme fino a-Ge:H com espessura real de aproximadamente 800nm

O substrato de silício cristalino é chamado de s. Foram criados sistemas com um e dois filmes. Nos sistemas com um filme, consideramos filmes de silício e germânio depositado sobre o substrato de silício cristalino. Nos sistemas com dois filmes, consideramos filmes de silício e germânio depositados um sobre o outro, no mesmo lado de um substrato de silício cristalino. Resumidamente, os sistemas considerados são: Es, Fs, e EFs.

A preparação dos filmes reais consistiu no uso de um sistema *rf-sputtering* usando argônio (Ar) como gás ionizador. A taxa de depósito dos filmes sobre o substrato de silício foi de 0.1nm/seg e o substrato foi mantido a cerca de 200°C durante todo o processo. Os tempos de depósito dos filmes foram de 110 minutos para o filme de germânio e 180 minutos para o filme de silício. As espessuras foram estimadas mecanicamente através do uso de um *Talystep* (instrumento utilizado para realizar medições de espessuras de materiais). Uma justificativa para não se usar sempre o *Talystep* para realizar este tipo de medição é que esta ferramenta pode danificar a estrutura do filme fino medido. Os valores 1150nm para o filme E e 800nm para o filme F foram obtidos através de uma média de medições realizadas em diferentes pontos da superfície dos sistemas. É importante notar que as medições estão sujeitas a erros mecânicos e variações estruturais dos sistemas (dependendo da ordem em que os filmes são depositados, a estrutura final dos filmes pode ser alterada). Devido a esses erros e variações, a espessura de ambos os filmes sobrepostos possui uma pequena diferença em relação à soma das espessuras obtidas ao medir os sistemas com um único filme. Antes de falarmos a respeito da recuperação das constantes óticas, observaremos os resultados relativos à recuperação das espessuras na Tabela 3.3.

Os dados reais medidos da transmitância de um substrato de silício cristalino e de cada um dos três sistemas Es, Fs, e EFs possuem um pouco de ruído e, a fim de obter uma boa recuperação com o PUMA, optamos por suavizar os dados das transmitâncias. A Figura 3.17 mostra as curvas originais e suavizadas do índice de refração do substrato e da transmitância dos três sistemas de filmes. O índice de refração foi obtido partir das medições da transmitância do substrato (usando uma versão simplificada da equação (3.1) onde o n e o κ dos filmes foram substituídos pelos do ar).

A Figura 3.15, mostra (a) a transmitância do substrato de silício (c-Si) e do sistema EFs, bem como o intervalo em que foi realizada a recuperação, e (b) a transmitância dos sistemas Es e Fs. Para a recuperação das constantes óticas, consideramos apenas o intervalo de energia do fóton determinado pelas linhas verticais pontilhadas na Figura 3.15 (a), ou seja, o intervalo espectral 1200 – 2190nm. Na Figura 3.15 também é possível notar que a transmitância do sistema com dois filmes sobrepostos é estruturalmente mais parecida com a transmitância do filme de germânio. Isso ocorre pois o filme de germânio é mais denso e menos transparente que o de silício.

Filme	Espessura recuperada (em nanometros)		
	Mecanicamente	Usando o PUMA para os sistemas com um filme Es e Fs	Usando o PUMA para o sistema com dois filmes EFs
E	1150	1135	1136
F	800	712	724
EF	1790	–	1860

Tabela 3.3: Recuperação da espessura dos filmes E e F, nos sistemas com um filme e com dois filmes sobrepostos, pelo PUMA e pelo método mecânico

As Figuras 3.18, 3.19 e 3.20 mostram os gráficos do erro quadrático em função da espessura de cada filme, obtidos durante a recuperação de espessuras e constantes óticas dos filmes E e F nos sistemas Es, Fs e EFs, respectivamente. Durante a recuperação, após o **Passo 2** do Algoritmo 3.1, obtivemos pontos com erro quadrático muito próximos. As regiões demarcadas nas Figuras 3.18, 3.19 e 3.20 indicam as espessuras candidatas à solução obtidas. Após uma varredura fina em cada uma das regiões foi escolhida como solução a espessura associada ao menor erro quadrático. Para os sistemas Es, Fs e EFs, os menores erros quadráticos obtidos foram respectivamente 8.237549×10^{-05} , 1.187394×10^{-06} e 1.881044×10^{-06} .

A Figura 3.16 compara as constantes óticas recuperadas dos filmes de silício e germânio isolados com aquelas recuperadas a partir do sistema EFs. Sobre os experimentos com filmes reais, podemos notar que: (i) Temos claramente a recuperação de dois filmes distintos, com diferentes constantes óticas; (ii) A recuperação das espessuras foi muito boa; (iii) A Figura 3.16 também indica que a recuperação das constantes é melhor para o filme de germânio, em comparação com o de silício (já que é ele que possui maior influência na transmitância do sistema);

3.4.3 Desempenho paralelo

A versão paralela do PUMA atingiu nossas expectativas, mostrando-se eficiente e escalável.

A Figura 3.21 mostra dois gráficos de speedup do PUMA para o uso de até 7 e 16 processadores em paralelo. O *speedup* foi obtido executando-se o método para uma mesma instância do problema (3.3),(3.15–3.19). A cada execução era variada apenas a quantidade de processos em paralelo. Cada execução foi repetida três vezes e então feita uma média nos tempos obtidos. Essa metodologia fornece uma medida mais precisa do tempo de execução do método, atenuando possíveis flutuações na medida dos tempos de execução. Esperávamos obter um *speedup* praticamente linear, uma vez que existe pouca comunicação entre os processos. Na Figura 3.21 (b) podemos ver que mesmo usando 16 processadores, o *speedup* ainda se mantém próximo do esperado.

(a)

(b)

Figura 3.15: Transmitâncias do substrato c-Si e do sistema de dois filmes EFs (acima) e dos sistemas de um único filme Es e Fs (abaixo).

Figura 3.16: Constantes óticas dos filmes E e F recuperados através do PUMA, a partir do sistema de dois filmes e dos sistemas de um único filme

Figura 3.17: Em (a) temos a curva do índice de refração do substrato original e suavizado. Em (b), (c) e (d) temos as transmitâncias do sistema com filme de silício, germânio e ambos respectivamente.

Figura 3.18: Gráfico do erro quadrático para o sistema Es e a tabela com os valores encontrados em cada região demarcada.

Figura 3.19: Gráfico do erro quadrático para o sistema Fs e a tabela com os valores encontrados em cada região demarcada.

Figura 3.20: Gráfico do erro quadrático para o sistema EFs e a tabela com os valores encontrados em cada região demarcada.

Figura 3.21: Curva de *speedup* do método usando até (a) 7 processadores e (b) até 16 processadores. Em (a) os experimentos foram realizados na grade do LCPD e em (b) no aglomerado do grupo de otimização contínua.

3.5 Análise dos resultados

Sobre as recuperações, podemos concluir que:

- A recuperação das propriedades óticas torna-se muito difícil, quando o sistema é composto por filmes de mesma natureza e espessura.
- Para sistemas com filmes do mesmo material, porém com espessuras diferentes o PUMA recupera adequadamente as espessuras e constantes óticas.
- A recuperação das espessuras nos casos de filmes de diferentes materiais também é muito boa. Conforme pode ser observado nos gráficos de erro quadrático (Figura 3.10), a indeterminação foi reduzida drasticamente. Isso mostra que as restrições físicas reduzem o espaço de busca de forma eficiente para sistemas com filmes de natureza distintas.
- De uma forma geral, no caso de filmes diferentes, a recuperação do coeficiente de absorção é melhor nos filmes mais densos, que possuem maior influência na transmitância. Já os índices de refração são bem recuperados em todos os casos, especialmente nas regiões de baixa absorção no espectro considerado.

É bastante surpreendente que um problema inverso com tão alto grau de indeterminação possa ser corretamente resolvido com um algoritmo de minimização.

A paralelização do método demonstrou um *speedup* linear e possibilitou a recuperação das constantes óticas e espessuras de filmes finos em sistemas com dois filmes, feito que não era empiricamente possível com a versão sequencial do método.

Capítulo 4

Packmol

PACKMOL [MM03] é um método criado para resolver o problema de posicionar conjuntos de diferentes moléculas em uma região fechada no espaço tridimensional, respeitando uma distância mínima e algumas restrições de posicionamento entre as moléculas. Este método tem a finalidade de obter configurações iniciais de moléculas para simulações de dinâmica molecular. Neste capítulo temos o objetivo de descrever como foi construída a versão paralela do PACKMOL e os resultados obtidos com a paralelização. Os tópicos abordados a seguir são: o modelo matemático do problema, a implementação do método em sua versão sequencial, a paralelização do método e a implementação da versão paralela, incluindo os experimentos numéricos.

4.1 Modelo matemático do problema

Nesta seção consideramos o problema de posicionar moléculas em uma região limitada do espaço, de forma que seja respeitada uma distância mínima entre seus átomos sujeitos ou não à restrições de posicionamento.

Uma molécula é uma estrutura rígida, representada pelas coordenadas cartesianas de seus átomos. Por estrutura rígida, entende-se que átomos de uma mesma molécula devem manter uma mesma posição relativa entre si, não podendo ser posicionados independentemente uns dos outros.

Cada molécula e cada átomo possuem um índice associado para permitir sua identificação. Neste capítulo usaremos i e j para denotar respectivamente os índices de moléculas e átomos. A quantidade total de moléculas é expressada por M e a quantidade de átomos de uma molécula de índice i é denotada por A_i .

O centro geométrico de uma molécula é o ponto caracterizado pela média aritmética das coordenadas de seus átomos. Vamos considerar que todas as moléculas são posicionadas com seus centros geométricos na origem do eixo cartesiano. As coordenadas cartesianas iniciais de cada um dos átomos são denotadas por:

$$c^{i,j} = \begin{pmatrix} c_x^{i,j} \\ c_y^{i,j} \\ c_z^{i,j} \end{pmatrix}, i = 1, \dots, M, j = 1, \dots, A_i,$$

Figura 4.1: Aplicação de translação e rotação a uma molécula.

Cada átomo pode sofrer uma rotação e uma translação a partir do ponto de origem. É importante lembrar que moléculas são estruturas rígidas, e assim, todos os átomos de uma mesma molécula sofrem as mesmas transformações. Veja a Figura 4.1 Essas transformações reposicionam os átomos e são denotadas por:

$$p^{i,j} = b^i + R^i c^{i,j},$$

onde

$$b^i = \begin{pmatrix} b_x^i \\ b_y^i \\ b_z^i \end{pmatrix} \quad (4.1)$$

e

$$R^i = \begin{pmatrix} \cos \theta_x^i \cos \theta_y^i \cos \theta_z^i - \sin \theta_x^i \sin \theta_z^i & \sin \theta_x^i \cos \theta_y^i \cos \theta_z^i + \cos \theta_x^i \sin \theta_z^i & -\sin \theta_y^i \cos \theta_z^i \\ -\cos \theta_x^i \cos \theta_y^i \sin \theta_z^i - \sin \theta_x^i \cos \theta_z^i & -\sin \theta_x^i \cos \theta_y^i \sin \theta_z^i + \cos \theta_x^i \cos \theta_z^i & -\sin \theta_y^i \sin \theta_z^i \\ \cos \theta_x^i \sin \theta_y^i & \sin \theta_x^i \sin \theta_y^i & \cos \theta_y^i \end{pmatrix} \quad (4.2)$$

representam respectivamente uma translação e uma rotação a serem aplicadas à cada um dos A_i átomos de cada uma das M moléculas.

Desejamos encontrar ângulos de rotação $\theta_x^i, \theta_y^i, \theta_z^i$ e coordenadas de translação (b_x^i, b_y^i, b_z^i) de forma que:

$$\forall i, i' = 1, \dots, M, \quad i \neq i', \quad \forall j = 1, \dots, A_i \text{ e } \forall j' = 1, \dots, A_{i'}, \quad \|p^{i,j} - p^{i',j'}\| \geq d, \quad (4.3)$$

com

$$\begin{aligned} \ell_x &\leq p_x^{i,j} \leq v_x, \\ \ell_y &\leq p_y^{i,j} \leq v_y, \\ \ell_z &\leq p_z^{i,j} \leq v_z, \end{aligned} \quad (4.4)$$

onde $\bar{d} > 0$ é a distância mínima entre os átomos e a norma usada é a euclidiana. Os valores ℓ_x, ℓ_y, ℓ_z e v_x, v_y, v_z representam, respectivamente, os limitantes inferiores e superiores que definem a caixa onde as moléculas devem ser posicionadas. As inequações denotadas em (4.3) impõem uma distância mínima $\bar{d}$ entre os átomos de diferentes moléculas. Também é possível atribuir a cada átomo uma ou mais restrições de posicionamento (**RP**). Tais restrições permitem limitar o posicionamento de conjuntos de átomos a determinadas regiões do espaço, como por exemplo, impor que os átomos de todas as moléculas de um determinado tipo só possam ser posicionadas dentro de uma esfera ou cubo. A Figura 4.2 ilustra um exemplo desse tipo de restrição. Uma explicação detalhada de cada um dos 12 tipos de restrições do posicionamento pode ser encontradas no Anexo B.

Figura 4.2: Exemplo de restrição no posicionamento de moléculas. Um grupo de lipídios é limitado a formar uma esfera no centro da região de empacotamento. Um grupo de moléculas de água é limitado ao interior da esfera e outro ao exterior.

Uma forma de resolver o problema de encontrar ângulos de rotação $\theta_x^i, \theta_y^i, \theta_z^i$ e coordenadas de translação (b_x^i, b_y^i, b_z^i) que respeitem (4.3), (4.4) e as **RP** é escreve-lo como um problema de minimização.

Vamos escrever a função de mérito da seguinte forma:

$$f(b, \theta) = f_{\text{dist}}(b, \theta) + f_{\text{reg}}(b, \theta) + f_{\text{pos}}(b, \theta), \quad (4.5)$$

onde:

$$\begin{aligned}
f_{\text{dist}}(b, \theta) &= \sum_{i=1}^M \sum_{j=1}^{A_i} \left(\sum_{i'=i+1}^M \sum_{j'=1}^{A_{i'}} \max\{0, d^2 - \|p^{i,j} - p^{i',j'}\|^2\}^2 \right), \\
f_{\text{reg}}(b, \theta) &= \sum_{i=1}^M \sum_{j=1}^{A_i} \left(\sum_{\text{coord} \in \{x,y,z\}} \max\{0, p_{\text{coord}}^{i,j} - v_{\text{coord}}, \ell_{\text{coord}} - p_{\text{coord}}^{i,j}\}^2 \right), \\
f_{\text{pos}}(b, \theta) &= \sum_{i=1}^M \sum_{j=1}^{A_i} \left(\sum_{t=1}^{12} \psi_t(p^{i,j})^2 \right),
\end{aligned} \tag{4.6}$$

$b = (b_x^1, b_y^1, b_z^1, \dots, b_x^M, b_y^M, b_z^M)$ e $\theta = (\theta_x^1, \theta_y^1, \theta_z^1, \dots, \theta_x^M, \theta_y^M, \theta_z^M)$. A função $\psi_t(p^{i,j})$, determina a penalização referente ao átomo na posição $p^{i,j}$ por desrespeitar a **RP** do tipo t . Veja o Anexo B para mais detalhes a respeito da função ψ .

A formulação do problema passa a ser:

$$\text{minimizar } f(b, \theta). \tag{4.7}$$

A função objetivo f é contínua e diferenciável. O número de variáveis do Problema (4.7) é $6 \times M$ (três ângulos e uma translação no espaço tridimensional por molécula).

O foco de nosso trabalho é a construção de uma versão paralela do PACKMOL. Antes de explicar os detalhes envolvidos na construção da versão paralela, é importante explicar alguns aspectos da implementação da versão sequencial do método, que foi utilizada como base para nosso trabalho. A seguir discutiremos a respeito da implementação sequencial do PACKMOL, incluindo algumas estratégias utilizadas para avaliar a função de mérito e seu gradiente.

4.2 O Método e sua Implementação

O cálculo de f_{dist} é o que consome mais tempo de processamento na avaliação da função de mérito (4.6). O termo

$$\sum_{i=1}^M \sum_{j=1}^{A_i} \left(\sum_{i'=i+1}^M \sum_{j'=1}^{A_{i'}} \max\{0, d^2 - \|p^{i,j} - p^{i',j'}\|^2\}^2 \right)$$

de f_{dist} , calcula as distâncias entre todos os pares de átomos de moléculas distintas existentes em uma dada instância do problema. Calcular a distância de cada átomo com todos os demais é custoso e desnecessário. Note que nem todos os pares de átomos contribuem para o somatório da função. Se dois átomos estão posicionados a mais de $\bar{d}$ unidades de distância um do outro,

$\max \left\{ d^2 - \|p^{i,j} - p^{i',j'}\|^2, 0 \right\}^2 = 0$. O ideal para evitar computações desnecessárias, seria considerar apenas os pares de átomos para os quais $\max \left\{ d^2 - \|p^{i,j} - p^{i',j'}\|^2, 0 \right\}^2 \neq 0$, porém, identificá-los não é uma tarefa simples.

Por outro lado, facilmente conseguimos identificar alguns pares de átomos tais que $\max \{ d^2 - \|p^{i,j} - p^{i',j'}\|^2, 0 \}^2 = 0$. Uma forma de fazer isso, é descrita no trabalho [BS08]. Tal estratégia foi utilizada na implementação da avaliação da função objetivo do PACKMOL sequencial, para torná-la mais eficiente. Ela consiste em subdividir a caixa definida em (4.4), que representa todo o espaço onde as moléculas devem ser posicionadas, em $\bar{C} = \bar{C}_1 \times \bar{C}_2 \times \bar{C}_3$ subcaixas menores, com lado de comprimento $\bar{d}$. Se dois átomos não estão em caixas adjacentes, certamente, distam mais de $\bar{d}$ unidades um do outro e sua distância não precisa ser calculada. Assim, os únicos átomos que devem ser levados em consideração no cálculo da soma são os que estão na mesma caixa ou em caixas adjacentes. A Figura 4.3 exemplifica o que foi dito acima usando apenas duas dimensões, a fim de facilitar a compreensão.

Computacionalmente, cada caixa é representada por uma lista c_k de átomos, conforme ilustrado pela Figura 4.4. Cada caixa é identificada por uma tripla $k = (k_1, k_2, k_3)$. A vizinhança de c_k , conjunto de caixas adjacentes à c_k , é identificado por um conjunto de triplas $k' = (k'_1, k'_2, k'_3)$ e definido como:

$$\begin{aligned} V(k) = \{ (k'_1, k'_2, k'_3) : \\ k'_1 \in \{k_1, k_1 + 1, k_1 - 1\}, k'_2 \in \{k_2, k_2 + 1, k_2 - 1\}, \\ k'_3 \in \{k_3, k_3 + 1, k_3 - 1\}, 0 \leq k_1 < \bar{C}_1, 0 \leq k_2 < \bar{C}_2, 0 \leq k_3 < \bar{C}_3 \} \setminus \{ (k_1, k_2, k_3) \} \end{aligned} \quad (4.8)$$

Para construir as listas c_k , é necessário saber onde cada átomo deve ser colocado. Para isso, dados os índices i de uma molécula e j de um átomo, podemos computar o índice k da seguinte forma:

$$\begin{aligned} k = (k_1, k_2, k_3) = \left(\max\{0, \min\{\frac{p_x^{i,j}}{\bar{d}}, \bar{C}_1 - 1\}, \right. \\ \max\{0, \min\{\frac{p_y^{i,j}}{\bar{d}}, \bar{C}_2 - 1\}, \\ \left. \max\{0, \min\{\frac{p_z^{i,j}}{\bar{d}}, \bar{C}_3 - 1\} \right) \end{aligned} \quad (4.9)$$

Em (4.9), observe que as caixas das bordas podem conter átomos que estão fora da região de empacotamento, nesse sentido são consideradas como caixas abertas.

O algoritmo abaixo mostra como calcular a função f_{dist} (4.6) usando as subcaixas para evitar o cálculo desnecessário de algumas distâncias:

Passo 1: Para $i = 1, \dots, M$ faça:

Passo 1.1: Para $j = 1, \dots, A_i$ faça:

Passo 1.1.1 Calcule o índice k da caixa à qual o átomo de índice j da molécula de índice i pertence, usando a expressão 4.9

Passo 1.1.2 Adicione (i, j) à lista c_k

Figura 4.3: Neste exemplo, em duas dimensões, é ilustrado como a avaliação da função (4.6) é simplificada. Em (a) é ilustrada a divisão da caixa em subcaixas menores com lado de tamanho $\bar{d}$ dentro da região, aqui denominada Ω , definida em (4.4). Em (b) é ilustrada uma das etapas do cálculo das distâncias. Os átomos estão desenhados como esferas cinzas. Note que apenas são consideradas as distâncias (linhas cinzas) do átomo a , no interior da caixa cinza escuro, com os demais átomos de sua caixa e das caixas adjacentes à sua.

Figura 4.4: Listas usadas para codificar a estrutura de caixas. Uma molécula, possui três átomos na caixa $(0, 0, 0)$. Outra molécula possui um átomo na caixa $(0, 1, 1)$ e dois na $(1, 1, 1)$. Para cada uma das oito caixas temos uma lista correspondente. Como apenas três caixas possuem átomos, somente três listas possuem elementos e são indicadas na lista de caixas não vazias. Em cada posição das listas não nulas temos o par (i, j) onde i é o índice de uma molécula e j o índice de um átomo.

Passo 2 $f \leftarrow 0$

Passo 3 Para $k = (0, 0, 0), (0, 0, 1), \dots, (\bar{C}_1 - 1, \bar{C}_2 - 1, \bar{C}_3) - 1$ faça:

Passo 3.1 Para cada (i, j) na lista c_k e cada (i', j') em posição posterior à (i, j) na lista c_k faça

Passo 3.1.1 $f \leftarrow f + \max\{d^2 - \|p^{i,j} - p^{i',j'}\|^2, 0\}^2$

Passo 3.2 Para $k' \in V(k)$ faça

Passo 3.2.1 Para cada (i, j) na lista c_k e cada (i', j') na lista $|c_{k'}|$ faça

Passo 3.2.1.1 $f_{\text{dist}} \leftarrow f + \max\{\bar{d}^2 - \|p^{i,j} - p^{i',j'}\|^2, 0\}^2$

Inicialmente definimos a caixa de cada átomo. Em seguida, para cada caixa c_k , computamos $\max\{\bar{d}^2 - \|p^{i,j} - p^{i',j'}\|^2, 0\}^2$ considerando todos pares de átomos de moléculas distintas no interior de c_k e considerando todos os pares de átomos de moléculas distintas com um átomo em c_k e outro em sua vizinhança. No **Passo 3.2**, apesar de não estar explícito, percorremos as caixas vizinhas sempre em uma mesma ordem.

Note que o termo $\|p^{i,j} - p^{i',j'}\|^2$ é computado duas vezes para pares de átomos em caixas distintas (veja Figura 4.5).

Figura 4.5: Em ambas as figuras, a região demarcada com a cor cinza escuro representa a caixa que está sendo analisada. A região cinza clara representa uma de suas adjacências. Na figura da esquerda, uma das distâncias que o método deve computar é entre os átomos (1,1) e (2,1). Na figura da direita, que representa algumas iterações adiante, uma das distâncias que o método vai computar é novamente entre os átomos (2,1) e (1,1). Dessa forma, usando a estratégia das caixas adjacentes, o método computa duas vezes uma mesma distância.

Para evitar isso, reduzimos a quantidade de caixas adjacentes que são consideradas no cálculo das distâncias. Para cada caixa c_k consideramos como sua vizinhança apenas as caixas com índices

dado por:

$$\begin{aligned}
 V'(k) = & \{(k'_1, k'_2, k'_3) : k'_1 \in \{k_1 + 1\}, k'_2 \in \{k_2, k_2 + 1, k_2 - 1\}, k'_3 \in \{k_3, k_3 + 1, k_3 - 1\}\} \\
 & \cup \{(k'_1, k'_2, k'_3) : k'_1 \in \{k_1\}, k'_2 \in \{k_2, k_2 + 1\}, k'_3 \in \{k_3, k_3 + 1\}\} \\
 & \cup \{(k_1, k_2 + 1, k_3 - 1)\} \\
 & \quad \setminus \{(k_1, k_2, k_3)\} \\
 & \setminus \{(k'_1, k'_2, k'_3) : k_1 < 0 \vee k_1 \geq \bar{C}_1, k_2 < 0 \vee k_2 \geq \bar{C}_2, k_3 < 0 \vee k_3 \geq \bar{C}_3\}
 \end{aligned} \tag{4.10}$$

A Figura 4.6 mostra quais caixas adjacentes devem ser consideradas de forma a impedir o cálculo duplicado de distâncias para átomos que estão em caixas diferentes.

Figura 4.6: Ilustração da estratégia de redução da vizinhança. Em (a) temos a ilustração em três dimensões e em (b) temos uma projeção em duas dimensões para facilitar a compreensão.

A avaliação do gradiente da função de mérito conta com alguns artifícios adicionais para torná-la mais eficiente.

As derivadas na forma $\frac{\partial f_{\text{dist}}(b, \theta)}{\partial p_{\text{coord}}^{i,j}}$, $\forall i = 1, \dots, M$, $\forall j = 1, \dots, A_j$, $\forall \text{coord} \in \{x, y, z\}$, podem ser escritas como:

$$\frac{\partial f_{\text{dist}}}{\partial p_{\text{coord}}^{i,j}} = \sum_{i'=1}^i \sum_{j'=1}^{A_{i'}} \frac{\partial \max\{\bar{d}^2 - \|p^{i,j} - p^{i',j'}\|^2, 0\}^2}{\partial p_{\text{coord}}^{i,j}} \tag{4.11}$$

onde,

$$\frac{\partial \max\{\bar{d}^2 - \|p^{i,j} - p^{i',j'}\|^2, 0\}^2}{\partial p_{\text{coord}}^{i,j}} = 4 \max_{i' \neq i} \{\bar{d}^2 - \|p^{i,j} - p^{i',j'}\|^2, 0\} (p_{\text{coord}}^{i,j} - p_{\text{coord}}^{i',j'}) \tag{4.12}$$

Observe que:

$$\frac{\partial \max\{\bar{d}^2 - \|p^{i,j} - p^{i',j'}\|^2, 0\}^2}{\partial p_{coord}^{i,j}} = - \frac{\partial \max\{\bar{d}^2 - \|p^{i,j} - p^{i',j'}\|^2, 0\}^2}{\partial p_{coord}^{i',j'}} \quad (4.13)$$

Apenas o cálculo de um dos termos é necessário, bastando inverter o sinal de um para obter o outro. Em (4.11), para obter $\frac{\partial f_{dist}}{\partial p_{coord}^{i,j}}$, calcula-se (4.12) para todos os pares de átomos que contém o átomo de índice j da molécula de índice i . Por outro lado, a implementação deste cálculo conta com a estratégia de considerar apenas os átomos nas caixas vizinhas.

Outras derivadas parciais possuem resultados conhecidos que não precisam ser calculados: $\frac{\partial p_{coord}^{i,j}}{\partial b_{coord}^{i,j}} = 1$ e $\frac{\partial p_{coord}^{i,j}}{\partial b_{coord'}^{i,j}} = 0, \forall i = 1, \dots, M, \forall j = 1, \dots, A_j, \forall coord, coord' \in \{x, y, z\}$ e $coord \neq coord'$.

Na implementação do PACKMOL, o método de otimização utilizado para resolver o problema de minimização (4.7) é o GENCAN [BM02a]. Trata-se de um método baseado em restrições ativas para resolver problemas de otimização contínua com restrições de caixas. Para uma introdução às técnicas básicas de otimização e aos métodos de restrições ativas, sugerimos [JS83] e [Lue84]. GENCAN adota como critério para abandono de face o critério descrito em [BM01], que utiliza os gradientes espectrais projetados definidos em [BMR00] e [BMR01]. Para a minimização interna às faces, GENCAN utiliza um algoritmo de busca linear que combina *backtracking* e extrapolação. GENCAN utiliza, também, um método de Newton Truncado para escolher uma direção de busca a cada passo no interior das faces. Uma descrição detalhada do método pode ser encontrada em [BM02b]. O GENCAN pode ser obtido na página *web* do projeto TANGO [TAN] do qual ele faz parte.

A implementação sequencial do PACKMOL foi construída usando as estratégias descritas nessa seção a fim de obter-se uma maior eficiência. A seguir falaremos sobre a versão paralela do PACKMOL, cuja implementação foi baseada na versão sequencial do método e incorpora algumas das estratégias citadas.

4.3 Paralelização do PACKMOL

Um dos objetivos do nosso trabalho é construir uma versão paralela do PACKMOL. Para isso, optamos por paralelizar a avaliação da função de mérito e seu gradiente. Baseamos tal decisão no fato de tais funções consumirem cerca de 85% do tempo de execução do PACKMOL. Nesta seção, falaremos a respeito da paralelização da função de mérito e de seu gradiente.

A estratégia usada para paralelizar a função de mérito consiste em atribuir a tarefa de calcular as distâncias entre um átomo e aqueles posicionados na mesma caixa e nas caixas vizinhas, pertencentes à diferentes moléculas, a processos distintos. Ao término desses cálculos, os resultados individuais de cada processo são combinados, resultando na soma desejada.

O modelo de paralelismo adotado envolve o uso de *threads* (processos leves) e memória compartilhada. Uma justificativa para tal decisão é a necessidade de minimizar os custos com operações de E/S e sincronização entre processos. O fato de todos os processos terem acesso a uma mesma região

de memória contendo os dados sobre as moléculas e suas posições nos permite construir uma versão paralela eficiente do PACKMOL.

Nossa primeira versão paralela da implementação da função (4.6) seguiu exatamente esta estratégia. Construímos-na a partir da implementação sequencial e procuramos realizar o menor número possível de modificações na implementação original. Veja a seguir um algoritmo ilustrando seu funcionamento:

Passo 1 Para cada processo $\rho = 1, \dots, \mathcal{P}$ faça:

Passo 1.1 $f_\rho \leftarrow 0$

Passo 1.2 Requisita uma lista de índices de caixas $\bar{k}^1, \dots, \bar{k}^K$

Passo 1.3 Para $k = \bar{k}^1, \dots, \bar{k}^K$

Passo 1.3.1 Para cada (i, j) na lista c_k e cada (i', j') em posição posterior à (i, j) na lista c_k faça

Passo 1.3.1.1 $f_\rho \leftarrow f_\rho + \max\{\bar{d}^2 - \|p^{i,j} - p^{i',j'}\|^2, 0\}^2$

Passo 1.3.2 Para $k' \in V'(k)$ faça

Passo 1.3.2.1 Para cada (i, j) na lista c_k e cada (i', j') na lista $|c_{k'}|$ faça

Passo 1.3.2.1.1 $f_\rho \leftarrow f_\rho + \max\{\bar{d}^2 - \|p^{i,j} - p^{i',j'}\|^2, 0\}^2$

Passo 1.4 Se ainda existem caixas para serem distribuídas, vá para **Passo 1.2**.

Passo 2 Cada processo $\rho \neq 1$ informa o valor de f_ρ para o processo 1.

Passo 3 O processo 1 computa $f_{\text{dist}} \leftarrow \sum_{\rho=1}^{\mathcal{P}} f_\rho$.

Nossa primeira versão possui um inconveniente. Ao variarmos a quantidade de processos, obtemos resultados diferentes para a resolução de uma mesma instância do problema. Isso acontece porque não é imposta uma ordem na realização da soma dos termos da função de mérito. Executar o método usando diferentes números de processos significa realizar a soma em ordens diferentes. Isso resulta em avaliações de função com valores distintos (é importante lembrar que, computacionalmente, uma alteração na ordem dos elementos a serem somados pode alterar o resultado final de uma soma), levando o método a convergir para pontos diferentes. O fato do método poder encontrar soluções diferentes para um mesmo problema, além de ser extremamente indesejável para o usuário, dificulta a medição do impacto da paralelização da função na execução do método.

Esse inconveniente nos levou a considerar a necessidade de impor uma ordem na soma das distâncias de forma que o resultado final fosse sempre o mesmo, independentemente da quantidade de processos ou da forma de distribuir os átomos entre eles.

Nossa solução para tal problema consiste em computar e armazenar resultados parciais da avaliação da função (4.6). Esses resultados parciais são computados sempre da mesma forma, independentemente da quantidade de processos. Depois, são somados sequencialmente, sempre seguindo

uma mesma ordem, a fim de obtermos um valor que corresponda à avaliação da função em um dado ponto. Esses resultados parciais são chamados de somas parciais. Abaixo temos um algoritmo ilustrando a avaliação da função (4.6) usando as somas parciais:

Passo 1 Para $q = 1, \dots, \bar{C}$ faça:

Passo 1.1 $w_q \leftarrow 0$

Passo 2 Para cada processo com índice $\rho = 1, \dots, \mathcal{P}$ faça:

Passo 2.1 Requisita uma lista de índices de caixas $\bar{k}^1, \dots, \bar{k}^K$

Passo 2.2 Para $k = \bar{k}^1, \dots, \bar{k}^{|\bar{K}|}$

Passo 2.2.1 Para cada (i, j) na lista c_k e cada (i', j') em posição posterior à (i, j) na lista c_k faça

Passo 2.2.1.1 $q \leftarrow \bar{C}_1 \bar{C}_1 k_1 + \bar{C}_2 k_2 + (k_3 + 1)$

Passo 2.2.1.2 $w_q \leftarrow +w_q \max\{\bar{d}^2 - \|p^{i,j} - p^{i',j'}\|^2, 0\}^2$

Passo 2.2.2 Para $k' \in V'(k)$ faça

Passo 2.2.2.1 Para cada (i, j) na lista c_k e cada (i', j') na lista $|c_{k'}|$ faça

Passo 2.2.2.1.1 $q \leftarrow \bar{C}_1 \bar{C}_1 k_1 + \bar{C}_2 k_2 + (k_3 + 1)$

Passo 2.2.2.1.2 $w_q \leftarrow +w_q \max\{\bar{d}^2 - \|p^{i,j} - p^{i',j'}\|^2, 0\}^2$

Passo 2.3 Se ainda existem caixas para serem distribuídas, vá para **Passo 1.2**.

Passo 3 O processo 1 realiza $\sum_{q=1}^C w_q$ obtendo o valor da função.

Cada soma parcial é armazenada em uma posição do vetor w , cujo tamanho corresponde exatamente à $\bar{C}$. Por fim realiza-se a soma de todas as somas parciais, obtendo-se o valor da função. É importante ressaltar que cada soma parcial é computada inteiramente por um mesmo processo. Veja a Figura 4.7 para um exemplo ilustrado de todo esse procedimento.

Avaliamos também a possibilidade de realizar a soma dos resultados parciais em paralelo. A necessidade de impor uma ordem nas operações, para que a soma resulte sempre em um mesmo valor, independentemente da quantidade de processos envolvidos, foi o maior obstáculo a ser superado.

Identificamos um algoritmo recursivo para realizar tal operação. Ele consiste em duas fases. Na primeira, o vetor de somas parciais é dividido pela metade em dois outros vetores, que são distribuídos entre dois processos. Essa divisão e distribuição é feita recursivamente até que o vetor distribuído possua apenas dois elementos. Na segunda fase, cada processo deve então somar esses dois valores, resultando em um vetor apenas com a soma. Os resultados devem ser distribuídos de forma que metade dos processos anteriores possuam vetores com dois elementos. A soma e concatenação são repetidos até que todos os números sejam somados em relação aos elementos originais. É importante mencionar que nesse processo de soma e concatenação, é necessário manter as

Figura 4.7: As figuras acima ilustram o processo de ordenação e paralelização de operações da função objetivo em um cenário com dois processos. Em (a), a cada processo é atribuído um conjunto de caixas. Em (b) e (c), são ilustradas as caixas envolvidas na computação das somas parciais das caixas 1 e 4 respectivamente. Em (d), temos o cálculo do valor da função objetivo a partir das somas parciais pelo processo 1.

posições relativas dos elementos somados. Por exemplo, um vetor com os elementos v_1, v_2, v_3 e v_4 , nessa ordem, deve ser somado da seguinte forma $((v_1 + v_2) + (v_3 + v_4))$. Segue uma versão iterativa do algoritmo de soma dos elementos de um vetor em paralelo:

Passo 1 $q \leftarrow 2$

Passo 2 Enquanto $q \leq |w|$ faça:

Passo 2.1 Para $l = 1, \dots, |w|$ com passo q faça:

Passo 2.1.1 Se $w_{l+q/2}$ existe então:

Passo 2.1.2 $w_l \leftarrow w_l + w_{l+q/2}$

Passo 2.2 $q \leftarrow q * 2$

Passo 3 $w_1 \leftarrow w_1 + w_{1+q/2}$

No final desse algoritmo, w_1 conterá o resultado desejado. Independentemente da quantidade de processos disponíveis, a soma final será sempre a mesma, já que as somas são realizadas sempre na mesma ordem.

Esse algoritmo pode atingir um desempenho de $O(\log(C))$. Para isso, é necessário que a quantidade de processadores seja pelo menos $\frac{C}{2}$. Dessa forma, todas as somas são sempre realizadas em paralelo. Usando apenas 4 processadores, o que temos disponível atualmente, o desempenho da soma paralela é muito inferior ao desempenho obtido por uma soma sequencial para qualquer problema real resolvido pelo PACKMOL. Por esse motivo, acabamos descartando a idéia.

Resumidamente, a avaliação paralela da função (4.6) funciona da seguinte forma:

Passo 1 Cada processo recebe um conjunto de índices de caixas.

Passo 2 Para cada caixa c_k cada processo computa a soma parcial e armazena o resultado em uma posição correspondente em w .

Passo 3 Se houver caixas que ainda não foram distribuídas, vá para o **Passo 1**.

Passo 4 O processo 1 percorre w e computam a soma de todos seus elementos, em paralelo, obtendo o valor da função.

Em relação ao gradiente da função (4.6), na nossa primeira versão paralela, onde dividimos os cálculos sem nos preocupar com a ordem das somas dos resultados obtidos por cada processo, tivemos o mesmo inconveniente de obter resultados diferentes para avaliações do gradiente com diferentes quantidades de processos.

Para resolver isso, novamente precisamos impor uma ordenação no cálculo de somas. Dessa vez trata-se das somas para obter as derivadas parciais descritas pela equação (4.11). Nossa primeira abordagem foi tentar isolar conjuntos de operações através de somas parciais de forma similar àquela feita na avaliação da função de mérito. Para isso foi necessário o uso intensivo de listas e memória. Combinar os resultados também representou um problema, já que para isso era necessário quase o mesmo esforço que o consumido para efetuar os cálculos de forma sequencial. Depois de alguns testes e tentativas mal sucedidas de melhorar o desempenho, descartamos essa estratégia.

Nossa segunda abordagem foi modificar a forma como o gradiente era avaliado. Precisávamos criar uma estrutura que fosse mais fácil de se paralelizar, considerando a restrição de que as operações matemáticas precisavam ser feitas sempre em uma mesma ordem. Notamos que a maior dificuldade em isolar os cálculos se devia ao já mencionado fato da implementação do gradiente aproveitar o cálculo de algumas parcelas da soma para obter outras. De acordo com a equação (4.13) é

possível economizar esforço computacional no cálculo de, por exemplo, $\frac{\partial \max\{\bar{d}^2 - \|p^{i,j} - p^{i',j'}\|^2, 0\}^2}{\partial p_x^{i,j}}$ e $\frac{\partial \max\{\bar{d}^2 - \|p^{i,j} - p^{i',j'}\|^2, 0\}^2}{\partial p_x^{i',j'}}$, pois são iguais, a menos de um sinal, bastando o cálculo de um dos termos.

Ao calcular derivadas parciais da função objetivo em relação às coordenadas de átomos no interior de uma caixa, essa igualdade é aproveitada para somar termos de derivadas parciais em outras posições

do vetor gradiente também. No caso da divisão da tarefa de computar o gradiente entre vários processos, os acessos concorrentes a diversas posições do vetor gradiente nos impedem de impor uma ordem na realização dos cálculos sem que o desempenho paralelo seja severamente prejudicado.

Remover tal artifício computacional teve o impacto de aumentar o custo de avaliação do gradiente. Por outro lado, permitiu facilmente isolar e dividir algumas operações de forma a permitir-nos construir uma versão paralela da avaliação do gradiente que de fato fosse eficiente e que obtivesse resultados idênticos independentemente da quantidade de processos.

A versão paralela da avaliação do gradiente funciona da seguinte forma:

Passo 1 Para cada processo $\rho = 1, \dots, \mathcal{P}$ faça:

Passo 1.1 Requisita uma lista de índices de caixas $\bar{k}^1, \dots, \bar{k}^K$

Passo 1.2 Para $k = \bar{k}^1, \dots, \bar{k}^{|\mathcal{K}|}$

Passo 1.2.1 Para cada (i, j) na lista c_k

Passo 1.2.1.1 Compute $\frac{\partial f}{\partial p_x^{i,j}}, \frac{\partial f}{\partial p_y^{i,j}}, \frac{\partial f}{\partial p_z^{i,j}}$ através de (4.11)

Passo 1.2.1.2 Compute as derivadas parciais de $p_x^{i,j}, p_y^{i,j}$ e $p_z^{i,j}$ em relação a cada um dos ângulos $\theta_x^i, \theta_y^i, \theta_z^i$ e a cada um dos termos b_x^i, b_y^i, b_z^i .

Passo 1.2.1.3 Aplique a regra da cadeia considerando os valores calculados no **Passo 1.2.1.1** e **Passo 1.2.1.2**, armazenando os resultados diretamente no vetor gradiente.

Dessa forma conseguimos que a avaliação paralela do gradiente da função (4.6) em um determinado ponto forneça sempre o mesmo resultado, independentemente da quantidade de processos envolvidos na computação.

4.4 Considerações adicionais sobre a paralelização

Além de paralelizar os cálculos que mais consumiam tempo na avaliação da função objetivo e seu gradiente, optamos por paralelizar alguns laços que inicializam vetores e a avaliação das restrições de posicionamento.

Conforme foi dito na Seção 4.2, o PACKMOL fornece a possibilidade de restringir o posicionamento de grupos de moléculas a regiões específicas. A paralelização da avaliação dessas restrições resumiu-se em dividir as moléculas entre os processos disponíveis e computar tais penalizações em paralelo. Para evitar o problema de somar as penalizações em ordem arbitrária, usamos a estratégia de calcular as somas parciais, conforme explicado na Seção 4.3 e na Figura 4.7.

Até agora nada foi dito sobre o modelo de paralelismo utilizado. Inicialmente consideramos a possibilidade de usar o modelo de troca de mensagens para a implementação da versão paralela do PACKMOL. Após alguns experimentos, percebemos que o custo das operações de E/S necessárias para transmitir os dados (referentes às posições dos átomos) entre os processos é muito alto e prejudica drasticamente o desempenho da versão paralela.

Pensando em eliminar o custo de transmitir os dados, optamos pelo modelo de *threads* com memória compartilhada. Dessa forma, todos os cálculos são feitos em uma região de memória compartilhada por todos os processos e há menor sobrecarga com sincronização dos estados dos mesmos. Com a popularização de computadores com múltiplos núcleos ou com diversos processadores compartilhando uma mesma memória principal, o uso desse modelo vem tornando-se interessante do ponto de vista prático.

4.5 Resultados Numéricos

Nossos experimentos numéricos tem o objetivo de avaliar a eficiência da paralelização da avaliação da função objetivo e de seu gradiente e o impacto dessa paralelização na execução do método.

Na implementação usamos a API OpenMP [Ope] e o código foi todo escrito em Fortran 77. Os binários foram obtidos através do compilador gfortran (parte integrante da projeto GNU *Compiler Collection*) versão 4.2 para Linux. A única opção de compilação usada foi `'-O3'`. A máquina onde os testes foram rodados possui um processador Intel(R) Core 2 Quad de 2.40GHz, 4 MB de cache e 4 GB de RAM.

Nos experimentos aqui realizados consideramos cinco problemas diferentes:

- Problema A: Uma mistura de água e uréia com 6200 átomos.
- Problema B: Hormônio em uma interface de água e clorofórmio com 4087 átomos.
- Problema C: Camada de lípidios envolvido por uma camada moléculas de água acima e outra abaixo. Contem 7200 átomos.
- Problema D: Uma esfera de lipídios com água em seu interior e exterior. Contém 69912 átomos.
- Problema E: Uma esfera de moléculas de água e íons contendo uma molécula de proteína. Contém 53796 átomos.

Inicialmente comparamos o desempenho da versão sequencial com o desempenho da versão paralela executada com um único processo, concluindo que esta última é cerca de 1,5 vezes mais lenta que a primeira.

Em seguida, fizemos uma análise da aceleração da versão paralela. Nesse caso, escolhemos como τ_1 o tempo de execução da versão paralela executada com um único processo. Os tempos de execução relatados aqui correspondem sempre à média dos tempos de três execuções.

Para medir o ganho com a paralelização na avaliação da função objetivo e seu gradiente, executamos o método fixando uma instância do problema de empacotamento e variando a quantidade de processos em paralelo (de 1 até 4). Nessas execuções, contabilizamos o tempo gasto apenas pelos trechos paralelizados. O resultado, que pode ser observado nas Figuras 4.8 e 4.9, foi próximo do esperado apenas para os problemas D e E. Com os problemas A, B e C, não obtivemos um *speedup* linear, conforme esperado. Uma possível explicação para esse fenômeno é a pequena quantidade

de átomos associada ao custo computacional de inicialização e gerenciamento do ambiente paralelo (rotinas para criação e destruição de *threads*, divisão dinâmica de dados, entre outras) a cada avaliação da função objetivo e seu gradiente. A pequena quantidade de átomos faz com que o custo das ações a serem realizadas em paralelo seja próximo ao custo computacional exigido pelas rotinas envolvidas na gerência do ambiente paralelo, o que prejudica o desempenho do método. Outros fatores que podem prejudicar o desempenho são a quantidade de restrições de posicionamento e a quantidade de átomos das moléculas às quais as restrições são aplicadas. Estes fatores explicam o melhor desempenho obtido com o problema A em relação ao C. Apesar de possuírem quantidades de átomos parecidas, têm uma diferença relevante: o problema A não impõe qualquer restrição de posicionamento, enquanto que o C restringe o posicionamento de grupos de moléculas de água e lipídios. De forma similar, o problema D, apesar de possuir mais átomos que o E, possui também um conjunto menos uniforme de moléculas. Uniformidade de moléculas significa moléculas diferentes mas com aproximadamente a mesma quantidade de átomos entre si. Por esse motivo, o melhor desempenho é obtido com E.

Para avaliar o impacto da paralelização da função objetivo e seu gradiente na execução do método, medimos tempo total de execução do método com até 4 processos em paralelo. A Tabela 4.1 mostra os tempos de execução do método para cada problema.

Quantidade	Tempo de CPU				
	Problema A	Problema B	Problema C	Problema D	Problema E
1	15.606	3.740	25.536	482.0931	200.79
2	10.573	2.836	18.223	284.6802	117.13
3	8.726	2.440	14.283	215.3903	89.203
4	7.780	2.356	13.243	182.7004	75.816

Tabela 4.1: Tempos consumidos pelo método ao ser executado com os problemas A, B, C, D e E, usando de 1 até 4 processos em paralelo.

Para avaliar os ganhos obtidos com a paralelização do método, decidimos comparar o *speedup* teórico, calculado a partir de dados obtidos através de um *profiling* do PACKMOL original, com o *speedup* obtido a partir dos dados mostrados na Tabela 4.1.

O *profiling* consistiu em medir o consumo de processamento de cada função do método ao resolver diferentes instâncias do problema de empacotamento. Para realizar o *profiling* usamos o programa gprof e contabilizamos dados da execução do método com 5 instâncias diferentes. Dessa forma, descobrimos que a parte paralelizada equivale a aproximadamente 85% do tempo de execução do método. Considerando esse dado, podemos obter o *speedup* teórico para uma execução com 4 processos:

$$s_{\text{teórico}} = \frac{1}{0.15 + \frac{0.85}{4}} = 2.759$$

Os gráficos do *speedup* medido comparado com o teórico calculado podem ser vistos na Figura 4.10. Para os casos A, B e C, a versão paralela tem um desempenho bem inferior ao teórico. Tal

desempenho já era previsto, uma vez que o *speedup* obtido para esses mesmos problemas, ao contabilizar apenas o tempo dos trechos paralelizados da função de mérito e seu gradiente (ver Figuras 4.8 e 4.9), também é inferior ao teoricamente suposto. Nos casos D e E, a versão paralela comporta-se aproximadamente conforme esperado.

Figura 4.8: Gráficos do *speedup* da avaliação da função objetivo.

Figura 4.9: Gráficos do *speedup* da avaliação do gradiente da função objetivo.

Figura 4.10: Gráfico do *speedup* real do método comparado com o speedup teórico para cada um dos experimentos.

Capítulo 5

Conclusões

Neste trabalho, apresentamos a motivação no uso de computação paralela em otimização contínua, alguns conceitos básicos da área de computação paralela e da área de otimização contínua. Falamos também a respeito de uma aplicação de otimização contínua para a obtenção de configurações iniciais para simulação de dinâmica molecular (PACKMOL) e um método de otimização específico para estimar espessuras e constantes óticas de filmes finos (PUMA), em suas versões sequenciais e paralelas, incluindo a avaliação dos ganhos com a paralelização.

A versão paralela do PACKMOL foi construída usando-se um modelo misto de memória compartilhada e *threads*. Obtivemos, na prática, um *speedup* próximo ao previsto teoricamente, apenas para instâncias com um grande número de moléculas.

A versão paralela do PUMA foi construída usando-se o modelo de troca de mensagens. Por se tratar de uma aplicação com estrutura *bag-of-tasks*, conseguimos excelente desempenho e escalabilidade. A versão paralela permitiu usar o método para recuperar as espessuras e constantes óticas de sistemas com dois filmes finos e um substrato. Teoricamente isso já era possível, porém era impraticável usar o PUMA com esse tipo de sistema, devido ao longo tempo de espera necessário para cada etapa do método. Com a paralelização, conseguimos recuperar os dados de sistemas com dois filmes com sucesso e desenvolvemos uma metodologia de uso similar ao que já era usado para os sistemas com um único filme, para obter os dados nos sistemas duplos. Sobre a qualidade das recuperações, identificamos os casos em que as espessuras ou as constantes óticas são naturalmente indeterminadas. No caso das espessuras, apresentamos alternativas para resolver as indeterminações.

Também identificamos a necessidade de alterar a forma do método obter a aproximação inicial do κ de cada filme. Trocamos de um aproximação linear por partes usando dois segmentos, para uma aproximação melhor usando três segmentos.

O conhecimento prático e teórico adquirido ao longo da elaboração desse trabalho, nos permite concluir que o uso de computação paralela em algoritmos de otimização contínua garante um melhor aproveitamento de processadores com múltiplos núcleos e computadores multiprocessados, ou grades e aglomerados de computadores, que estão a cada dia mais acessíveis. Conseqüentemente, também permite a resolução mais rápida de problemas, ou até mesmo a solução problemas que antes não podiam ser resolvidos.

Referências Bibliográficas

- [ABC⁺06] Krste Asanovic, Ras Bodik, Bryan Christopher Catanzaro, Joseph James Gebis, Parry Husbands, Kurt Keutzer, David A. Patterson, William Lester Plishker, John Shalf, Samuel Webb Williams, and Katherine A. Yelick. The landscape of parallel computing research: A view from berkeley. Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley, December 18 2006.
- [Bar] Blaise Barney. Introduction to parallel computing. https://computing.llnl.gov/tutorials/parallel_comp/. Lawrence Livermore National Laboratory Online Tutorial.
- [BCM99] E. G. Birgin, I. Chambouleyron, and J. M. Martínez. Estimtion of the optical constants and the thickness of thin films using unconstrained optimization. *Journal Of Computational Physics*, 151:862–880, 1999.
- [BM01] E.G. Birgin and J.M. Martínez. A box constrained optimization algorithm with negative curvature irections and spectral projected gradients. *Computing [Supple]*, 15:49–60, 2001.
- [BM02a] E. G. Birgin and J. M. Martínez. Large-scale active-set box-constrained optimization method with spectral projected gradients. *Computational Optimization and Applications*, 23:101125, 2002.
- [BM02b] E.G. Birgin and J.M. Martínez. Large-scale active-set box-constrained optimization method with spectral projected gradients. *Computational Optimization and Aplications*, 23:101–125, 2002.
- [BMR00] E.G. Birgin, J.M. Martínez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10:1196–1211, 2000.
- [BMR01] E.G. Birgin, J.M. Martínez, and M. Raydan. Algorithm 813 spg: Software for convex-constrained optimization. *ACM Transactions on Mathematical Software*, 27:340–349, 2001.
- [BS08] E.G. Birgin and F.N.C. Sobral. Minimizing the object dimensions in circle and sphere packing problems. *Computers & Operations Research*, 35:2357–2375, 2008.

- [BSP] <http://www.bsp-worldwide.org/>.
- [BT89] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice-Hall International Editions, 1989.
- [Cen96] Y. Censor. *Introduction to methods of parallel optimization*. IMPA, 1996.
- [CM01] I. Chambouleyron and J. M. Martínez. Optical properties of dielectric and semiconductor thin films. *Handbook of Thin Films Materials*, 3:593–622, 2001.
- [Fly72] M. J. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, 21(9):948–960, 1972.
- [Fos95] I. Foster. *Designing and Building Parallel Programs*. Addison-Wesley, 1995.
- [GRI] <http://www.grid5000.fr>.
- [HMS⁺98] Jonathan M. D. Hill, Bill McColl, Dan C. Stefanescu, Mark W. Goudreau, Kevin Lang, Satish Rao, Torsten Suel, Thanasis Tsantilas, and Rob H. Bisseling. Bsplib: The bsp programming library. *Parallel Computing*, 24(14):1947–1980, 1998.
- [IG04] IEEE and The Open Group. Ieee std 1003.1. *The Open Group Base Specifications*, Issue 6, 2004.
- [Int] <http://www.integrate.org.br>.
- [JS83] J.E. Dennis Jr. and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.
- [Lue84] D.G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 1984.
- [MD79] N. F. Mott and E. A. Davis. *Electronic properties in non crystalline materials*. Clarendon Press, 1979.
- [MGF76] J. C. Manifacier, J. Gasiot, and J. P. Fillard. A simple method for the determination of the optical constants n, k and the thickness of a weakly absorbin film. *Journal of Physics E: Scientific Instruments*, 9:1002, 1976.
- [MM03] J. M. Martínez and L. Martínez. Packing optimization for automated generation of complex system’s initial configurations for molecular dynamics and docking. *Journal of Computational Chemistry*, 24:819–825, 2003.
- [Moo65] G. E. Moore. Cramming more components onto integrated circuits. *Electronics Magazine*, April 19 1965.
- [OM02] A. Ohring and A. Milton. *Materials Science of Thin Films*. Academic Press, 2002.

- [Ope] <http://www.openmp.org>.
- [PBL] http://www.netlib.org/scalapack/pblas_qref.html.
- [PLA] <http://www.cs.utexas.edu/~plapack/>.
- [Poi] <http://www.ime.usp.br/~egbirgin/puma/>.
- [Ray97] M. Raydan. The Barzilai and Borwein Gradient Method for the Large Scale Unconstrained Minimization Problem. *SIAM Journal on Optimization*, 7(1):26–33, 1997.
- [Sca] <http://www.netlib.org/scalapack/>.
- [Sch95] R. B. Schnabel. A view of the limitations, opportunities, and challenges in parallel nonlinear optimization. *Parallel Computing*, 21:875–905, 1995.
- [Swa83] R. Swanepoel. Determination of the thickness and optical constants of amorphous silicon. *Journal of Physics E: Scientific Instruments*, 16:1214–1222, 1983.
- [Swa84] R. Swanepoel. Determination of surface roughness and optical constants of inhomogeneous amorphous silicon films. *Journal of Physics E: Scientific Instruments*, 17:896–903, 1984.
- [TAN] <http://www.ime.usp.br/~egbirgin/tango/>.
- [Val90] Leslie G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, 1990.
- [Ven05] Sérgio Drumond Ventura. *Técnicas de otimização para estimação de parâmetros de filmes finos*. PhD thesis, UNICAMP, 2005.

Anexo A

PUMA

Aqui, colocamos as expressões analíticas usadas para computar o índice de refração do substrato, bem como as constantes óticas de cada filme.

$$s_{vidro}(\lambda) = \sqrt{1 + (0.7568 - 7930/\lambda^2)^{-1}}. \quad (A.1)$$

a-Si:H

Índice de refração:

$$n_{\text{verdadeiro}}(\lambda) = \sqrt{1 + (0.09195 - 12600/\lambda^2)^{-1}}. \quad (A.2)$$

Coefficiente de absorção:

$$\ln(\alpha_{\text{verdadeiro}}(E)) = \begin{cases} 6.5944 \cdot 10^{-6} \exp(9.0846E) - 16.102, & 0.60 < E < 1.40; \\ 20E - 41.9, & 1.40 < E < 1.75; \\ \sqrt{59.56E - 102.1} - 8.391, & 1.75 < E < 2.20. \end{cases} \quad (A.3)$$

a-Ge:H

Índice de refração:

$$n_{\text{verdadeiro}}(\lambda) = \sqrt{1 + (0.065 - (15000/\lambda^2)^{-1}}. \quad (A.4)$$

Coefficiente de absorção:

$$\ln(\alpha_{\text{verdadeiro}}(E)) = \begin{cases} 6.5944 \cdot 10^{-6} \exp(13.629E) - 16.102, & 0.50 < E < 0.93; \\ 30E - 41.9, & 0.93 < E < 1.17; \\ \sqrt{89.34E - 102.1} - 8.391, & 1.17 < E < 1.50. \end{cases} \quad (A.5)$$

Nas expressões acima, o comprimento de onda λ é dado em nm , o valor da energia do fóton $E = 1240/\lambda$ em eV , e o coeficiente de absorção α em nm^{-1} .

Anexo B

PACKMOL

A implementação do PACKMOL contempla restrições que podem ser aplicadas ao posicionamento de átomos. Tais restrições permitem limitar o posicionamento de conjuntos de átomos a determinadas regiões do espaço. O método trata tais restrições como penalizações, aplicadas à função de mérito (4.6).

Descrevemos, a seguir, cada uma das 12 restrições e a forma de calcular a penalização $\psi_t, t = 1, \dots, 12$ correspondente. Para simplificar a notação, a posição de um átomo será denotada por $\chi = (x, y, z)$.

1. *Dentro de um cubo:* Considere $x_{\min}, y_{\min}, z_{\min}$ e d quatro números reais. Dessa forma as coordenadas (x, y, z) devem satisfazer:

$$\begin{aligned}x_{\min} &\leq x \leq x_{\min} + d, \\y_{\min} &\leq y \leq y_{\min} + d, \\z_{\min} &\leq z \leq z_{\min} + d.\end{aligned}$$

A penalização é calculada da seguinte forma:

$$\begin{aligned}\psi_1(\chi) &= \min(x - x_{\min}, 0)^2 + \min(y - y_{\min}, 0)^2 + \min(z - z_{\min}, 0)^2 \\&\quad + \max(x - (x_{\min} + d), 0)^2 + \max(y - (y_{\min} + d), 0)^2 \\&\quad + \max(z - (z_{\min} + d), 0)^2.\end{aligned}$$

2. *Dentro de uma caixa:* Considere $x_{\min}, y_{\min}, z_{\min}, x_{\max}, y_{\max}, z_{\max}$ seis números reais. Dessa forma as coordenadas (x, y, z) devem satisfazer:

$$\begin{aligned}x_{\min} &\leq x \leq x_{\max}, \\y_{\min} &\leq y \leq y_{\max}, \\z_{\min} &\leq z \leq z_{\max}.\end{aligned}$$

A penalização é calculada da seguinte forma:

$$\begin{aligned}\psi_2(\chi) &= \min(x - x_{\min}, 0)^2 + \min(y - y_{\min}, 0)^2 + \min(z - z_{\min}, 0)^2 \\&\quad + \max(x - x_{\max}, 0)^2 + \max(y - y_{\max}, 0)^2 \\&\quad + \max(z - z_{\max}, 0)^2.\end{aligned}$$

3. *Dentro de uma esfera:* Considere a, b, c e d quatro números reais. Dessa forma as coordenadas (x, y, z) devem satisfazer:

$$(x - a)^2 + (y - b)^2 + (z - c)^2 - d^2 \leq 0.$$

A penalização é calculada da seguinte forma:

$$\psi_3(\chi) = \max \left((x - a)^2 + (y - b)^2 + (z - c)^2 - d^2, 0 \right)^2.$$

4. *Dentro de uma elipsóide:* Considere $a_1, b_1, c_1, a_2, b_2, c_2, d$ sete números reais. Dessa forma as coordenadas (x, y, z) devem satisfazer:

$$\frac{(x - a_1)^2}{a_2^2} + \frac{(y - b_1)^2}{b_2^2} + \frac{(z - c_1)^2}{c_2^2} - d^2 \leq 0.$$

A penalização é calculada da seguinte forma:

$$\psi_4(\chi) = \max \left(\frac{(x - a_1)^2}{a_2^2} + \frac{(y - b_1)^2}{b_2^2} + \frac{(z - c_1)^2}{c_2^2} - d^2, 0 \right)^2.$$

5. *Fora de um cubo:* Considere $x_{\min}, y_{\min}, z_{\min}$ e d quatro números reais. Dessa forma as coordenadas (x, y, z) devem satisfazer:

$$\begin{aligned} (x &\leq x_{\min} \quad \text{ou} \quad x \geq x_{\min} + d) \text{ ou} \\ (y &\leq y_{\min} \quad \text{ou} \quad y \geq y_{\min} + d) \text{ ou} \\ (z &\leq z_{\min} \quad \text{ou} \quad z \geq z_{\min} + d). \end{aligned}$$

A penalização é calculada da seguinte forma:

$$\begin{aligned} \psi_1(\chi) = & \max(x - x_{\min}, 0)^2 + \max(y - y_{\min}, 0)^2 + \max(z - z_{\min}, 0)^2 \\ & + \min(x - (x_{\min} + d), 0)^2 + \min(y - (y_{\min} + d), 0)^2 \\ & + \min(z - (z_{\min} + d), 0)^2. \end{aligned}$$

6. *Fora de uma caixa:* Considere $x_{\min}, y_{\min}, z_{\min}, x_{\max}, y_{\max}, z_{\max}$ seis números reais. Dessa forma as coordenadas (x, y, z) devem satisfazer:

$$\begin{aligned} (x &\leq x_{\min} \quad , \quad \text{ou} \quad x \geq x_{\max}) \text{ ou} \\ (y &\leq y_{\min} \quad , \quad \text{ou} \quad y \geq y_{\max}) \text{ ou} \\ (z &\leq z_{\min} \quad , \quad \text{ou} \quad z \geq z_{\max}). \end{aligned}$$

A penalização é calculada da seguinte forma:

$$\begin{aligned} \psi_2(\chi) = & \max(x - x_{\min}, 0)^2 + \max(y - y_{\min}, 0)^2 + \max(z - z_{\min}, 0)^2 \\ & + \min(x - x_{\max}, 0)^2 + \min(y - y_{\max}, 0)^2 \\ & + \min(z - z_{\max}, 0)^2. \end{aligned}$$

-
7. *Fora de uma esfera:* Considere a, b, c e d quatro números reais. Dessa forma as coordenadas (x, y, z) devem satisfazer:

$$(x - a)^2 + (y - b)^2 + (z - c)^2 - d^2 \geq 0$$

A penalização é calculada da seguinte forma:

$$\psi_7(\chi) = \min \left((x - a)^2 + (y - b)^2 + (z - c)^2 - d^2, 0 \right)^2.$$

8. *Fora de uma elipsóide:* Considere $a_1, b_1, c_1, a_2, b_2, c_2, d$ sete números reais. Dessa forma as coordenadas (x, y, z) devem satisfazer:

$$\frac{(x - a_1)^2}{a_2^2} + \frac{(y - b_1)^2}{b_2^2} + \frac{(z - c_1)^2}{c_2^2} - d^2 \geq 0.$$

A penalização é calculada da seguinte forma:

$$\psi_8(\chi) = \min \left(\frac{(x - a_1)^2}{a_2^2} + \frac{(y - b_1)^2}{b_2^2} + \frac{(z - c_1)^2}{c_2^2} - d^2, 0 \right)^2.$$

9. *Sob um plano:* Considere a, b, c, d quatro números reais. As coordenadas (x, y, z) devem satisfazer:

$$ax + by + cz - d \leq 0.$$

A penalização é calculada da seguinte forma:

$$\psi_9(\chi) = \min (ax + by + cz - d, 0)^2.$$

10. *Sobre um plano:* Considere a, b, c, d quatro números reais. As coordenadas (x, y, z) devem satisfazer:

$$ax + by + cz - d \geq 0.$$

A penalização é calculada da seguinte forma:

$$\psi_{10}(\chi) = \max (ax + by + cz - d, 0)^2.$$

11. *Dentro de um cilindro:* Considere um ponto c representando o centro da base de um cilindro, um vetor unitário $\hat{v}$ definindo a direção desse, bem como dois valores reais l e r correspondendo ao comprimento e raio do mesmo. Um átomo com coordenadas χ deve respeitar o seguinte:

$$0 \leq \langle \chi - c, \hat{v} \rangle \leq l \text{ e } \|(\chi - c) - \omega \hat{v}\| \leq r,$$

onde $\omega = \langle \chi - c, \hat{v} \rangle$. A penalização é calculada da seguinte forma:

$$\begin{aligned} \psi_{11}(\chi) = & \max \left(\|(\chi - c) - \omega \hat{v}\|^2 - r^2, 0 \right)^2 \\ & + \max \left(\langle \chi - c, \hat{v} \rangle - l, 0 \right)^2 \\ & + \min \left(\langle \chi - c, \hat{v} \rangle, 0 \right)^2 \end{aligned}$$

12. *Fora de um cilindro:* Considere um ponto c representando o centro da base de um cilindro, um vetor unitário $\hat{v}$ definindo a direção desse, bem como dois valores reais l e r correspondendo ao comprimento e raio do mesmo. Um átomo com coordenadas χ deve respeitar o seguinte:

$$\langle \chi - c, \hat{v} \rangle \leq 0 \text{ ou } \langle \chi - c, \hat{v} \rangle \geq l \text{ e } \|(\chi - c) - \omega \hat{v}\| \geq r,$$

onde $\omega = \langle \chi - c, \hat{v} \rangle$. A penalização é calculada da seguinte forma:

$$\begin{aligned} \psi_{12}(\chi) = & \max \left(\|(\chi - c) - \omega \hat{v}\|^2 - r^2, 0 \right)^2 \\ & + \begin{cases} (\langle \chi - c, \hat{v} \rangle - l)^2, & \text{se } \langle \chi - c, \hat{v} \rangle < l \\ (\langle \chi - c, \hat{v} \rangle)^2, & \text{se } \langle \chi - c, \hat{v} \rangle > 0 \end{cases} \end{aligned}$$