Ricardo Antônio Naberegny de Lara

Estudo de alocação de recursos em redes de acesso utilizando ferramentas da Teoria dos Jogos e Algoritmos Genéticos

Monografia do projeto de conclusão do curso de Ciência da Computação apresentada ao Departamento de Ciência d a Computação da Universidade Federal de Lavras como parte das exigências para obtenção do título de Bacharel em Ciência da Computação

Orientador

Professora Renata Couto Moreira

Lavras
Minas Gerais – Brasil
2002

Ricardo Antônio Naberegny de Lara

Estudo de alocação de recursos em redes de acesso utilizando ferramentas da Teoria dos Jogos e Algoritmos Genéticos

Monografia do projeto de conclusão do curso de Ciência da Computação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 16 de Dezembro de 2002

Professora e orientadora Renata Couto Moreira
Departamento de Ciência da Computação – UFLA

_____

Professor José Monserrat
Departamento de Ciência da Computação – UFLA

_____

Lavras
Minas Gerais - Brasil

O que é a vida senão uma espiral sem fim, em que nos perdemos irremediavelmente e, ao olhar para os lados, não há pontos de referência em que possamos nos basear? Caindo sempre, embora a sensação da falta de peso sirva para uma compreensão falsa de liberdade; escolha uma.

Fique a salvo da leviandade incrédula de mentes vazias ou busque a vanguarda do que você próprio, embora céptico, possa criar. A espiral pode se tornar uma infinidade de caminhos concorrentes, assim que você parar seu falso giro. De que adiantaria, afinal, todos os caminhos seriam infinitamente os mesmos, sempre.... Continue caindo, melhor para alguns, a falsa percepção de um mundo em colapso sobre si mesmo, como aquela espiral.

Ricardo Antônio Naberegny de Lara
naberegny@hotmail.com

Dedico este trabalho às pessoas queridas que deixaram marcas positivas em minha personalidade

Agradecimentos

Agradeço à minha família e aos amigos verdadeiros, que têm tornado possível a perspectiva positiva que ainda tenho do mundo, motivo maior de minha caminhada até aqui. Agradeço à minha orientadora, que me mostrou a existência do problema e pelas opiniões construtivas, quando tinha tempo para conversar. E também àqueles professores da graduação que não consideravam dar aula como sendo uma obrigação, nem subestimavam o fato de que alunos também podem compartilhar idéias, pois foram indispensáveis no processo de edificação pessoal, como profissional e ser humano.

Estudo de alocação de recursos em redes de acesso utilizando ferramentas da Teoria dos Jogos e Algoritmos Genéticos

Resumo

Foi estudado o problema de alocação de custos entre clientes de uma rede de acesso. Seus fundamentos, limitações, implicações e recursos matemáticos necessários à sua resolução.

Foram apresentadas considerações e deduções do problema, que foi modelado como sendo um jogo na forma cooperativa. Um importante resultado obtido foi a análise por diversos ângulos do problema, bem como sua interpretação geométrica.

Partindo-se do entendimento do problema, da interpretação geométrica, e do estudo de um algoritmo exato para a obtenção do *nucleolus* cuja complexidade de tempo e espaço é exponencial, chegou-se a uma proposta de heurística que trabalha com um número linear de restrições, com base no uso de Algoritmos Genéticos associados ao Simplex.

Palavras-chave: redes de acesso, Teoria dos Jogos, Algoritmos Genéticos, interpretação geométrica, *nucleolus*, restrições, Simplex.

An study on resource allocation in access network using concepts of Games Theory and Genetic Algorithms

Abstract

We studied the problem of cost allocation among users of an access network. Its foundations, limitations, implications, and mathematics resources necessaries to the problem resolution.

We presented considerations and deductions about the problem, which was modeled as a cooperative game. An important result accomplished was an analisys by different perspectives as well as its geometric interpretation.

Starting with the problem understanding and the study of an exact algorithm to obtain the *nucleolus* whose complexity of time and space is exponential, and also the geometric interpretation of the problem, we obtained a proposal of an heuristic which works with a linear number of constraints, based on the use of Genetic Algorythm associated to Simplex.

Keywords: access network, Game Theory, Genetic Algorythm, geometric interpretation, *nucleolus*, constraints, Simplex.

Sumário

Capítulo 1 – Introdução 1

Capítulo 2 – Referencial Teórico 4

2.1 Problemas de Otimização em Redes de Acesso **4**
2.2.1Definindo o jogo de Árvore geradora mínima **7**
2.2.2 O núcleo do Jogo de Árvore Geradora Mínima **8**
2.2 A Teoria dos Jogos Cooperativos **6**
2.3 O nucleolus **10**
2.4 Algumas características de funções **12**
2.5 A Têmpera Simulada **13**
2.6 O Algoritmo Branch & Bound **15**
2.7 Algoritmos Genéticos **15**
2.7.1 Introdução **15**
2.7.2 Aplicando a Técnica de Algoritmos Genéticos **17**
2.7.3 Operadores Genéticos **18**
2.8 O Método Simplex **20**
2.8.1 Uma visão geral do método **20**
2.8.2 Fundamentos teóricos do simplex **21**
2.8.3 O Algoritmo Primal Simplex **22**
2.9 A Dualidade em programação matemática **23**
2.10 Um algoritmo para computar o Nucleolus em um jogo de árvore geradora mínima **24**
2.10.1 O nucleolus **25**
2.10.2 Os poliedros ômega **31**
2.10.3 Modificação da geração de restrições para computar o nucleolus com (n-1) iterações **32**

Capítulo 3 – Resultados e discussão 34

3.1 O Problema **34**
3.2 Modelagem do problema **34**
3.3 Porquê n restrições? (possibilidades) **37**
3.3.1 Possibilidade 1 ..... **37**
3.3.2 Possibilidade 2 ..... **37**

3.3.3 Possibilidade 3 46
3.3.4 Possibilidade 4 48
3.4 Então, porquê n restrições afinal (conclusão)? .. 49
3.4.1 – Interpretação geométrica do *nucleolus* ... 57
3.5 Escolhendo as n restrições que interessam 59

Capítulo 4 - Usando Algoritmos Genéticos associados ao Simplex para se computar soluções próximas do nucleolus 61

4.1 Uma visão geral 61
4.2 Uma proposta de heurística 62
4.2.1 Os indivíduos 62
4.2.2 A geração 1 62
4.2.3 Critério de aptidão ou adaptabilidade 63
4.2.4 Reprodução dos indivíduos 65
4.2.5 Quantas gerações serão computadas em cada AG 66
4.2.6 Quando parar o algoritmo 67
4.2.7 Como serão usados os indivíduos 68
4.2.8 Uma representação da idéia do algoritmo .. 69
4.2.9 Parâmetros a serem testados e modificados .. 69

Capítulo 5 – Conclusão e Propostas 71

Capítulo 6 – Bibliografia 73

Lista de figuras

Figura 2.1. Usuários e o fornecedor do serviço **4**

Figura 2.2. Um grafo com os usuários e o fornecedor **5**

Figura 2.3. Máximos locais. 1e2- Máximos locais. 3- Platô. 5- Subida de encosta. 4-Máximo Global **13**

Figura 4.1. Regiões delimitadas por uma restrição **38**

Figura 4.2. Região retangular delimitada por duas restrições **38**

Figura 4.3. Região delimitada pela restrição $x \leq 3$ **39**

Figura 4.4. Região delimitada por uma restrição que corta xy a 45 graus **40**

Figura 4.5. Triângulo equilátero e escaleno **42**

Figura 4.6. Pirâmide regular **43**

Figura 4.7. Cubo deslocado à direita da origem, no eixo x **45**

Figura 4.8. Só uma dimensão basta para representar um jogador **51**

Figura 4.9. Redução de uma dimesão (reta) a um único ponto **51**

Figura 4.10. Espaço bidimensional com restrições envolvendo dois jogadores **52**

Figura 4.11. Redução de uma área (bidimensional) a uma reta **52**

Figura 4.12. O poliedro hiperdimensional , delimitando o núcleo (concepção pessoal) **54**

Figura 4.13. O *nucleolus*. Os números representam os pontos que tocam em cada um dos *n* eixos que representam os *n* jogadores. Determinam os valores que cada um

deles irá pagar **56**
Figura 4.14. O nucleolus seria único? **58**
Figura 5.1 A heurística proposta **69**

Capítulo 1 – Introdução

Vários problemas do dia-a-dia, e até mesmo o maior desafio para alguns de nós, a nossa vida, podem ser encarados como um jogo. Há, claro, os jogadores, em que nós próprios nos incluímos, regras a serem seguidas, objetivos, punições, cooperação e competição entre os participantes.

Se conseguirmos definir de maneira satisfatória esses componentes para atingirmos um objetivo ou superarmos um problema, podemos fazê-lo utilizando ferramentas matemáticas e analíticas. Assim surgiu a Teoria dos Jogos, que é usada nos mais diversos campos, desde Ecologia, Economia, Pedagogia e até na Inteligência Militar.

Um problema na computação, assim como em outros setores, é o de alocação de recursos entre vários usuários de uma rede, como água, luz, ou de uma rede de computadores.

A solução deste problema é de suma importância, com o aumento da privatização no fornecimento dos serviços, pois um custo mínimo precisa ser calculado, ao mesmo tempo em que este seja repassado de forma justa a todos que compartilham da mesma rede. Nenhum subgrupo pode arcar com um custo superior ao obtido pela solução encontrada para toda a rede.

Infelizmente o problema não é tão simples assim. A rede pode ser representada por um grafo, com o nó raiz representando o fornecedor do serviço, e vários nós representado os clientes deste serviço. Mas há restrições a serem observadas, e estas fazem a complexidade do problema crescer exponencialmente com o tamanho da rede.

A partir do estudo de um algoritmo exato, que usa Têmpera Simulada e Branch & Bound para obtenção de uma solução otimizada mas não mostrou como esse uso foi feito,

propõe-se uma outra heurística para se tentar resolver o problema.

Além da Teoria dos Jogos, será utilizada outra ferramenta extremamente poderosa de otimização de problemas, o método Simplex, capaz de lidar com programação linear.

Outra idéia interessante a ser utilizada no trabalho é a dos algoritmos genéticos, que seguem leis evolutivas, pois fazem alusão à própria cadeia cromossômica, à maneira de como a informação mais bem sucedida (darwinismo) é passada adiante, inspirando este tipo promissor de programação.

Este trabalho tem por objetivo unir os conceitos e aplicações da Teoria dos Jogos Cooperativos, Algoritmos Genéticos, Simplex e estudo de uma heurística exata cuja complexidade de tempo e espaço é exponencial, para propor uma heurística que venha a obter uma solução aproximada para o problema de alocação de custos em redes de acesso, cuja complexidade seja linear. O estudo foi focado no problema em se tratar redes muito grandes (com mais de 25 pontos).

Isto é muito útil, já que de nada adianta a melhor solução dentre todas ser possível de ser calculada, se ela levar tempo demais para ser obtida. Entenda por tempo demais, em computação, algo que varia de meses a dezenas de anos ou mais para ser calculado. Definitivamente, o que não desejamos esperar.

Segundo estudos, realmente não é necessário todo o espaço de restrições, que é da ordem de 2^n, onde n é o número de usuários da rede. A heurística busca justamente selecionar somente as restrições que interessam dentre todas as possíveis, diminuindo assim a complexidade de tempo e espaço de exponencial para linear, obtendo uma solução próxima da ótima, em tempo e com recursos aceitáveis.

Finalmente, além de ser um trabalho exploratório e propor uma heurística, é também de computação científica, apresentando a interpretação geométrica do problema, que facilita a visualização e entendimento de uma série de conceitos formais e se revelou de enorme importância. Isto possibilitou também formular uma explicação própria de porquê é possível a redução da complexidade de tempo e

espaço de exponencial para linear, sustentando a aplicabilidade da heurística apresentada.

Ou seja, o trabalho não se limitou a usar idéias e conceitos já existentes, seguindo por vezes uma linha de raciocínio própria, fundamentada com base no que foi estudado e usando conceitos matemáticos aceitos e bem definidos, numa área extremamente interessante, pelas implicações matemáticas por trás do problema.

Capítulo 2 - Referencial Teórico

2.1 Problemas de Otimização em Redes de Acesso

Como podemos representar uma rede, seja ela de água, luz, telefone, de computadores, etc? (Fig. 2.1).

Uma estrutura adequada, em matemática, chama-se grafo. Um grafo é constituído de nós, e de arestas, ligando estes nós.

Formalmente, chamamos $G=(N,E)$ uma rede de acesso representada por um grafo com um conjunto N de nós, representando os pontos a serem conectados, no caso, os usuários de um serviço na rede, e um conjunto E de arcos, representando as possibilidades de ligações entre estes pontos, ambos finitos (Fig. 2.2).

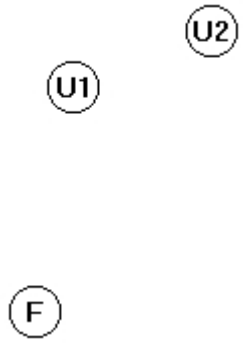

Figura 2.1. Usuários e o fornecedor do serviço

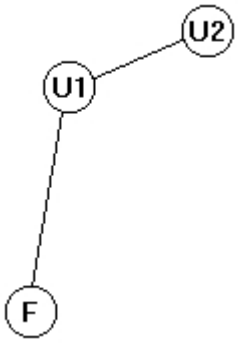

Figura 2.2. Um grafo com os usuários e o fornecedor

Para cada nó $i \in N$ pode ser associada uma demanda di, que no caso mais geral pode ser maior que zero (nó preto ou nó de cliente), ou seja, os que estão ligados ao nó fornecedor, menor que zero (nó fornecedor) ou igual a zero (nó bifurcação ou transbordo) [MOREIRA, 2002]. A presença de nós de transbordo, ou nós brancos, é encarada como situações em que a decisão de menor custo pode seguir inicialmente por uma trajetória comum e só a partir de determinado ponto, bifurcar o caminho para ligar os clientes, aumentando a possibilidade de ligações do grafo. Estes nós, porém, não precisam necessariamente participar da solução ótima [MOREIRA, 2002].

Considerar os nós de transbordo na solução, implica em trabalharmos com um problema que é NP-Difícil [MOREIRA, 2002].

Os nós do grafo também podem ter capacidades e custos associados, útil para situações em que um posto de fornecimento de um serviço pode ser instalado em um nó com um certo custo de construção e uma certa capacidade de atendimento, encontradas nos problemas de localização de facilidades, por exemplo [MOREIRA, 2002]. Isto torna o uso dos grafos mais flexível, podendo-se trabalhar com outros tipos de objetivo.

Capacidades de atendimento devem ser levadas em conta principalmente nos problemas em que as ligações têm um limite físico de transporte, como energia elétrica, redes hidráulicas, etc.

De forma semelhante, cada ligação possui um custo c_{ij} embutido caso o arco $(i,j) \in E$ seja escolhido na solução ótima. O arco pode possuir também limites superiores e inferiores para o fluxo do produto a ser trafegado pela rede, dependendo da situação.

Com a configuração destes parâmetros, o custo mínimo para a construção da rede em questão pode ser obtido através de extensões de problemas de caminho mínimo, ou de árvore geradora mínima, ou de árvore de Steiner, ou do problema de localização de concentradores entre outros conhecidamente eficientes para estas situações [MOREIRA, 2002].

Como ligar os nós de usuários ao posto de fornecimento de forma barata? Como o fornecedor é comum, a maneira mais indicada é usando-se uma árvore [MOREIRA, 2002]. A árvore pode em certos casos ser obtida por algoritmos de obtenção de árvores geradoras mínimas, aquelas que abrangem todos os nós, e que não possuem ciclos, isto é, mais de uma ligação chegando a um determinado nó.

Ciclos podem ser importantes em redes em que certos usuários do serviço não podem ficar de maneira alguma sem fornecimento. Se uma ligação física até eles for interrompida, estes usuários precisam de outra via para receber o serviço.

Mas apesar da árvore ser a forma mais barata para fazer a conexão entre clientes e fornecedor, em telecomunicações, as redes de acesso podem possuir outras estruturas, como a de anel, varal, e combinação de várias topologias, por exemplo. Isso deve-se ao fato de outras características serem também levadas em consideração, como hierarquia na rede, disposição dos equipamentos, etc.

“Obtida a solução ótima para a instância considerada, naturalmente surge o problema do rateio deste custo mínimo entre os consumidores de uma maneira justa, buscando conciliar duas metas, a de formação da rede ótima e a satisfação de todos. Em outras palavras, procura-se repartir o custo da rede de uma forma que nenhum subconjunto de usuários prefira montar sua própria sub-rede em vez de participar da proposta pela solução de custo mínimo total. Estas considerações sugerem o uso da teoria dos jogos

cooperativos para a solução do problema.”[MOREIRA, 2002].

A Teoria dos Jogos tem se mostrado uma ferramenta eficiente nas mais diversas áreas, e é empregada com sucesso também na resolução de problemas envolvendo grafos e árvores, como é o caso do rateio de custo entre usuários em uma rede.

2.1.1 Definindo o jogo de Árvore geradora mínima

É usada a Árvore Geradora Mínima, chamada comumente por AGM, pois dentre as várias possibilidades de se ligar usuários a um fornecedor em uma rede, esta é a maneira mais simples que existe, porque possui o menor custo associado aos nós clientes, pois não possui ligações desnecessárias, ou seja, que precisem existir para ligar nós que já não tenham sido ligados [MOREIRA, 2002].

Esta árvore possui como raiz o nó que representa o fornecedor comum do serviço considerado, ramificando-se em nós clientes que têm uma demanda maior que zero deste serviço. Estes nós clientes podem ser as folhas da árvore ou estarem entre as folhas e a raiz.

Existem diversos algoritmos já conhecidamente eficientes para obtermos uma AGM em um grafo, não sendo assunto de discussão neste texto.

A obtenção da AGM é um problema já solucionado de forma bastante eficiente e conhecida. Mas após isso surge o problema principal. Como dividir o custo entre os clientes de forma mínima? Claro, para que todos queiram participar da solução, é preciso que ninguém se sinta prejudicado, pagando um valor mais alto ao que pagaria se ligasse de outra forma ao fornecedor.

É neste contexto que entra a Teoria dos Jogos, que trata do problema transformando-o em um jogo de AGM, e cooperativo, por haver colaboração entre os jogadores, ou usuários do serviço.

2.1.2 O núcleo do Jogo de Árvore Geradora Mínima

O núcleo do jogo de AGM, é definido como o conjunto das combinações de alocações de custos viáveis, determinado pelas restrições que representam o custo máximo que cada combinação de jogadores pode pagar [MOREIRA, 2002].

Qualquer combinação de distribuição de custos entre os jogadores em que nenhum deles irá pagar mais do que pagaria ao se ligar de outra forma ao fornecedor faz parte do núcleo do jogo.

É comprovado em estudos que o núcleo do jogo nunca será vazio, ou seja, sempre vai existir ao menos uma solução justa para os jogadores [MOREIRA, 2002].

Temos também que as soluções que pertencem ao núcleo podem ser lidas diretamente do grafo considerado na solução [MOREIRA, 2002].

2.2 A Teoria dos Jogos Cooperativos

Para este problema, é mais recomendado, para o bem de todos os jogadores, que estes colaborem conjuntamente para a solução geral. Não há, então, competição entre eles, e sim colaboração. Por esse motivo, usamos a vertente cooperativa da Teoria dos Jogos, em vez da Teoria dos Jogos Competitivos.

Neste tipo de abordagem, a Teoria dos Jogos tem definida como função característica um jogo cooperativo, com os nós de usuários formando o conjunto de jogadores N e a função característica c definida no conjunto P, formado por todos os subconjuntos de N, (combinações possíveis de jogadores) que chamaremos de coalizões [MOREIRA, 2002].

O valor da função característica para uma dada coalizão $S \subseteq N$ deve ser o custo da sub-rede ótima provendo serviço àquele subconjunto de clientes do serviço.

Definimos formalmente o modelo como [MOREIRA, 2002]:

Seja $N=\{1,2,\ldots,n\}$ o conjunto finito de jogadores, $c:P->R$, $c(\varnothing)=0$ será a função característica $\forall S \subseteq N$ sendo $c(N)$ o custo que deve ser repartido entre os jogadores, então o par $(N;c)$ é chamado jogo cooperativo.

É necessário a definição das restrições para podermos garantir a formação e a estabilidade da solução de menor custo. São elas que limitam o que cada coalizão pode arcar na solução. Elas definem o *núcleo* do jogo e são do tipo [MOREIRA, 2002]:

$$x \in R^n \mid f(x,S) \leq c(S) \ \forall S \subseteq N;$$
$$\text{sendo } f(x,S) \equiv \sum_{j \in S} x_j$$
$$f(x,N)=c(N) \ \forall x$$

Aqui, $f(x,S)$ deve ser interpretado como a parte do custo total pago pela coalizão S quando a alocação x for considerada. O núcleo consiste de todas as alocações de custo x onde nenhuma coalizão tenha incentivos de montar sua própria sub-rede, pois para qualquer subconjunto de jogadores, o custo da sua sub-rede ótima $c(S)$ é sempre maior ou igual ao valor com que este subconjunto estará carregado $f(x,S)$. Existem situações em que o núcleo pode ser vazio (redes que usam nós de transbordo), que não é o tipo de rede estudado neste trabalho. Usar nós de transbordo (auxiliares, não são nem usuários nem o fornecedor) ou nós de Steiner, torna o problema NP-Difícil [MOREIRA,2001].

O número exponencial de restrições do núcleo, uma restrição para cada subconjunto possível de N, dificultam muito o cálculo do mesmo.

Da maneira como formulamos o problema pela Teoria dos Jogos Cooperativos, é necessário que este jogo tenha então um objetivo bem definido. Apesar de o núcleo conter em muitos casos várias soluções possíveis, algumas são preferíveis a outras.

Em toda a solução, nenhuma coalizão terá incentivos de montar sua própria sub-rede, pois irá ter que pagar mais ou o mesmo valor que pagaria se participasse da solução do problema.

Chamamos cada solução de vetor de alocação *x*, e a escolha dos melhores dentre todos os que compõe o núcleo pode ser obtida de várias maneiras já propostas. Neste vetor, cada elemento é o quanto cada coalizão vai pagar participando da solução geral obtida. Melhores respostas são aquelas cujos vetores de alocações de custos definam um valor menor possível a ser pago por todas as coalizões.

Dentre as formas de se obter a melhor solução (ou as melhores soluções) dentre todas as que compõe o núcleo, temos o n*ucleolus*.

2.3 O *nucleolus*

A diferença essencial deste conceito para outros na escolha do melhor vetor de alocação é a idéia de excesso, definido formalmente como [MOREIRA, 2002]:

Para um jogo (N;c) e para um vetor de alocação de custos x, seja $e(x,S) = c(S) - f(x,S)$ *o excesso de S relativo a x* $\forall S \neq \varnothing$, $S \subset N$, *e seja* $e(x)$ *um vetor em* $R^{(2n-n)}$ *cujos elementos são* $e(x,S)$, $\forall S \neq \varnothing$, $S \subset N$, *em ordem não decrescente.*

Temos que o *núcleo* do jogo será o conjunto de todos os vetores de alocação de custos *x* para os quais todas as coalizões possuem um excesso maior ou igual a zero.

Excesso zero para uma coalizão significa que esta irá pagar o mesmo valor participando da solução geral ao que pagaria ao se ligar de forma autônoma ao fornecedor.

O *nucleolus* é o conjunto de vetores *x* que maximizam *e(x)* lexicograficamente (que obtêm seguidamente um vetor de alocação cada vez mais justo para todas as coalizões). Definimos ordem lexicográfica, e sua relação de ordem ($>_{lo}$) da seguinte forma [MOREIRA, 2002]:

Quando consideramos dois vetores $a=(a_1,a_2,a_3,\ldots,a_n)$ e $b=(b_1,b_2,b_3,\ldots,b_n)$, dizemos que a é estritamente maior lexicograficamente que b, ou seja $a >_{lo} b$, se existir um inteiro q, $l \leq q \leq n$, tal que:

$a_i = b_i$ para $l \leq i \leq n$
$a_q >_{lo} b_q$

A relação de estritamente maior lexicograficamente introduz a relação de maior ou igual lexicograficamente ($>_{lo}$) que pode ser definida como:

$a \geq_{lo} b$ se $a >_{lo} b$ ou $a = b$

Para facilitar, lembre que o dicionário também está em ordem lexicográfica.

Assim, o *nucleolus*, sobre o conjunto de alocações de entrada $X=\{x| \sum_{i \in N} x_i = c(N)\}$, é o conjunto definido por:

$\{x| x \in N \text{ se } m \in X, \text{ então } e(x) \geq_{lo} e(m)\}$

"O *nucleolus* é o conjunto de alocações de entrada que primeiro maximiza o menor excesso dentre todas coalizões possíveis. Nele está embutida a idéia de fazer o conjunto de membros mais 'insatisfeito' ficar o mais feliz possível. Feito isso, passa-se a tentar deixar o mais feliz possível o segundo conjunto de membros mais 'insatisfeito', e assim por diante." [MOREIRA, 2002]. Coalizões mais insatisfeitas são aquelas com o menor excesso, ou seja, aquelas que vão pagar tanto quanto ou quase o que pagariam ao se ligar de forma autônoma ao fornecedor do serviço.

Concluímos então que o melhor vetor de alocação, aquele que é o maior lexicograficamente pertencente ao *nucleolus*, em que as coalizões mais insatisfeitas terão seu excesso maximizado.

Conseguir o *nucleolus*, contudo, não é uma tarefa fácil. O número de restrições cresce exponencialmente com o tamanho do problema. Uma maneira de se tentar diminuir o custo computacional de se obter o *nucleolus* seria, por exemplo, diminuir o número de restrições que precisamos levar em conta para resolver o problema.

Foi proposta uma heurística [GEFFARD, 1997] que usa Têmpera Simulada e Branch & Bound para trabalhar com as restrições que interessam.

Este conceito de excessos das coalizões foi imprescindível na formulação da heurística proposta neste trabalho, por ser usado como avaliação de adaptabilidade no Algoritmo Genético.

2.4 Algumas características em funções

Numa curva que representa a função de solução de um problema podemos ter não apenas um máximo, ou ponto ótimo, mas vários pontos máximos, ou mesmo um ponto máximo e vários pontos de solução que representam um atendimento à função em menor escala, ou ótimos locais.

Vários elementos podem ser encontrados no gráfico de uma função, como o máximo global, máximos locais, platôs [FERNEDA, 2002].

Como os algoritmos geralmente caminham sempre no sentido de se melhorar cada vez mais ou obter de forma direta uma solução, pode ser que o resultado obtido não seja o melhor dentre todos, pois pode ter atingido um platô (região da função em que todos os pontos têm o mesmo valor) ou um máximo local (Fig. 2.3).

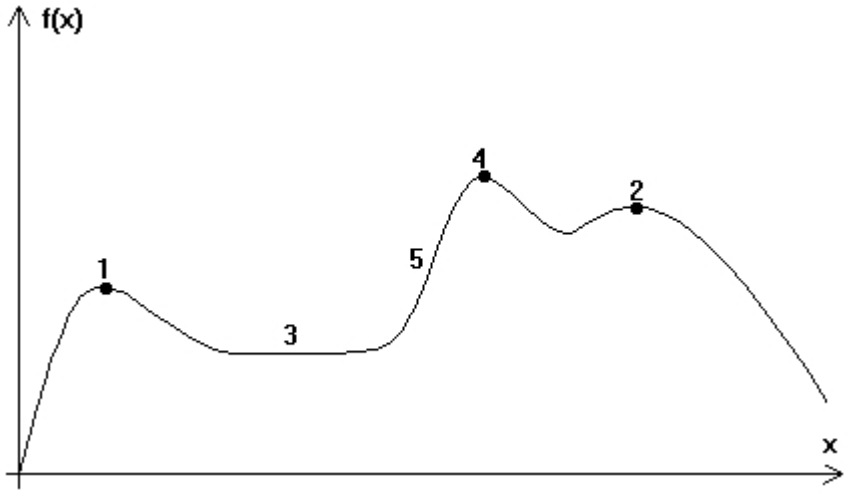

Figura 2.3. Máximos locais. 1e2- Máximos locais. 3- Platô. 5- Subida de encosta. 4-Máximo Global

Os máximos locais são soluções para o problema, porém menos otimizadas. São picos mais baixos do que aquela com do ponto ótimo [FERNEDA,2002].

Para se evitar este tipo de problema, o de se obter um máximo local, foram propostas várias idéias, como algoritmos de subida e descida de encosta, Têmpera Simulada e os Algoritmos Genéticos.

2.5 A Têmpera Simulada

A Têmpera Simulada é uma heurística que visa tratar os problemas explorando todas soluções possíveis, modificando-as para escapar de máximos locais. A idéia é a mesma da têmpera da indústria, que usa um cronograma de aquecimento e resfriamento, que visa otimizar a solução. Ela usa um mapeamento de resfriamento em instantes de tempo com diferentes temperaturas [UFPE, 2002].

O que este método vem a propor é fazer com que o processo de obtenção da solução possa retroceder de vez em quando, para dar a chance de se tentar outros caminhos para

conseguirmos uma solução melhor, porém sempre progredindo de maneira geral.

Isto é implementado atribuindo-se uma dada quantia de energia inicial ao processo de obtenção da solução [ROISENBERG, 2002].

Enquanto a energia for suficiente, o processo continua, podendo gastá-la aos poucos, subindo encostas na função, descer encostas, percorrer platôs. Com a energia diminuindo, a solução então se estabiliza em um máximo que tenha uma boa chance de ser o ponto ótimo da solução do problema [ROISENBERG, 2002].

Em muitos casos é necessário retroceder o processamento do algoritmo para que possamos encontrar a solução ótima. Por exemplo, o *Quebra Cabeça de 8*, um jogo em que temos oito peças deslizáveis em que o objetivo é colocá-las em ordem crescente. Podemos ter situações em que devemos tirar uma peça de sua posição definitiva, para dar espaço a outra peça para passar.

Para uma visualização mais fácil, é útil pensar no processo de fabricação do vidro temperado. O processo de têmpera prevê um resfriamento mais lento para formar o vidro a partir de sua matéria prima. Assim, dando às moléculas mais tempo para se arranjarem numa estrutura mais compacta, o mesmo fica mais resistente a trincas, que desta maneira não se propagarão facilmente. No caso de um algoritmo, pode ser interpretado como um tempo maior e exploração de todo o espaço de soluções, para que o ponto ótimo seja alcançado lentamente.

2.6 O Algoritmo Branch & Bound

A idéia de Branch & Bound é a de dividir e conquistar, ou seja, subdividir um problema grande em outros menores, a fim de facilitar e reduzir o custo de se computar grandes quantidades de informação.

Por ser aplicável a uma gama muito grande de problemas, o Branch & Bound pode se apresentar de várias formas, porém, basicamente, sua idéia é a mesma.

Aplica-se o algoritmo para a resolução do problema não diretamente a todo o universo de possibilidades, ou

dados de entrada do problema, mas sim em conjuntos menores de informação ou quantias menores de processamento. Um ponto importante neste tipo de heurística é que ela se torna uma forte candidata ao auxílio da computação em paralelo [TAVARES, 2002].

Neste processamento em paralelo, os processos recebem parcelas de trabalho de mesma magnitude, ou distribuição de carga [TAVARES, 2002].

Os problemas do tipo NP-Difíceis são geralmente os candidatos mais indicados a serem tratados por esta heurística [TAVARES,2002].

2.7 Algoritmos Genéticos

2.7.1 Introdução

Dentre os vários tipos de Sistemas existentes na computação, temos, por exemplo, aqueles inspirados na natureza. Seu funcionamento simula um processo ou evento natural, aplicando esta idéia a um problema real. Como exemplo, podem ser citados as Redes Neurais e os Algoritmos Genéticos [REZENDE, 2002].

Os Algoritmos Genéticos simulam os processos de evolução das espécies, em que elas convergem para uma anatomia eficiente ao seu modo de vida, adequadas ao ambiente e à sua sobrevivência.

Assim, ao aplicarmos esta idéia de evolução e seleção natural a uma possível solução ou possíveis soluções de um problema, estas irão evoluir e aperfeiçoar-se continuamente até convergirem a uma solução ótima ou bem próxima dela.

Esta convergência das possíveis soluções a uma solução otimizada conta com conceitos como a aptidão dos indivíduos e operadores genéticos[REZENDE, 2002].

Esse sistema baseado na natureza não trata a solução que queremos obter do problema como sendo o fenótipo dos indivíduos, ou seja, a forma com que eles se apresentam fisicamente, mas do ponto de vista genético, seu genótipo, como eles apresentam sua codificação genética. Esta

representação consiste na codificação de um cromossomo, composto por uma cadeia de caracteres, com a solução do problema [REZENDE, 2002].

Com base na codificação genética dos indivíduos, cada um representando uma possível solução para o problema que o Algoritmo Genético se propôs a resolver, os mesmos evoluem com base em um critério de seleção, e os menos aptos tendem a desaparecer durante a evolução. Este processo de seleção natural foi descrito por Charles Darwin em seu livro "Origem das Espécies" [REZENDE, 2002].

Como observado na natureza, os indivíduos menos aptos na competição entre si por recursos como água e comida tendem a deixar um número menor de descendentes, e consequentemente, não obtém muito êxito na transmissão de seus genes [REZENDE, 2002].

Por outro lado, indivíduos mais bem adaptados irão sobreviver o suficiente para transmitir seus genes às próximas gerações, recombinando-os assim com os genes de indivíduos de mesma espécie e que também sejam bem adaptados. Como resultado deste cruzamento, novos indivíduos com características combinadas de seus pais têm chance se serem igualmente e até mais bem adaptados do que aqueles.

Outro processo presente na evolução observada na natureza e imitado pelos Algoritmos Genéticos são as alterações casuais em genes do cromossomo: as mutações.

Estas têm importância controversa [REZENDE, 2002], mas se mostram importantes para se varrer todas as possibilidades e devem ser testadas com diferentes taxas de incidência, mas de maneira aleatória [REZENDE, 2002].

Os Algoritmos Genéticos imitam esse processo, fazendo uma analogia direta com o mesmo. Cada indivíduo representa uma possível resposta ou solução para o problema. Dada uma geração de indivíduos, a cada um é atribuído uma pontuação de sua adaptação, dependendo de quão boa é a resposta deste indivíduo ao problema em questão.

Aos mais bem adaptados é dada uma chance maior de se reproduzir mediante reprodução sexuada, com outros indivíduos da população daquela geração, produzindo filhos com características em comum aos dois, ou de forma

assexuada, simplesmente passando de forma inalterada seus genes.

Claro, parâmetros de avaliação, mutações, codificação dos indivíduos, etc, devem ser desenvolvidos corretamente, para que o programa possa então convergir para uma resposta otimizada. O importante é que todo o espaço de possibilidades tenha a chance de surgir na população para ser avaliado.

Os Algoritmos Genéticos (também chamados de AG's) diferem dos métodos tradicionais de busca e otimização, principalmente em quatro aspectos [REZENDE, 2002]:

1- AGs trabalham com uma codificação do conjunto de parâmetros e não com os próprios parâmetros;

2- AGs trabalham com uma população de soluções e não com um único ponto, ou solução;

3- AGs utilizam informações de custo ou recompensa e não derivadas ou outro conhecimento auxiliar;

4- AGs utilizam regras de transição probabilísticas e não determinísticas.

Para uma grande variedade de problemas, os AG's são eficientes para a busca de uma solução ótima, pois não possuem as limitações dos métodos de busca tradicionais, tendo, por exemplo, a chance de se fugir de um máximo local [REZENDE, 2002].

Chamamos os AG's de Algoritmos Genéticos, no plural mesmo, e não um Algoritmo Genético, pois sua idéia básica de evolução deve ser mantida, mas os mesmos podem apresentar as mais diversas variações, em vários pontos do processo.

2.7.2 Aplicando a Técnica de Algoritmos Genéticos

Primeiramente é gerada uma população de indivíduos, a primeira geração, de forma totalmente aleatória, onde cada um deles pode ser visto como uma possível solução para o problema.

Estas possíveis soluções são codificadas em uma sequência de bits ou caracteres, formando um cromossomo

com seus genes, e cada cromossomo é o genótipo de um indivíduo.

O processo evolutivo é feito da seguinte forma. Esta população é avaliada, onde cada indivíduo recebe uma nota de acordo com o quanto sua solução atende ao problema. Indivíduos cujas soluções sejam melhores, receberão notas melhores, ou seja, são mais bem adaptados.

Os mais bem adaptados são mantidos na evolução, passarão seus genes para a frente, enquanto os menos aptos serão descartados (evolução natural, ou darwinismo).

Os indivíduos bem adaptados então se cruzarão entre si, trocando material genético (crossover) dando origem a indivíduos com características em comum com os pais, poderão se reproduzir assexuadamente, neste caso serão simplesmente copiados na próxima geração, e também sofrendo ocasionalmente alguma mutação.

Estes novos indivíduos que são descendentes de pais bem adaptados da geração anterior irão compor a próxima geração.

Este processo continua, e, no decorrer das gerações, é esperada uma população de indivíduos bastante aptos, com soluções otimizadas.

Apesar de parecer simplistas do ponto de vista biológico, estes algoritmos são suficientemente complexos para fornecer mecanismos de busca adaptativos poderosos [REZENDE, 2002].

2.7.3 Operadores Genéticos

Estes operadores têm a função de tornar possíveis as transformações no decorrer das gerações, até que uma solução satisfatória para o problema tenha sido encontrada. Sem eles, não haveria variabilidade nos genes dos indivíduos em diferentes gerações, atualizando as características adquiridas na adaptação.

O operador de seleção tem como objetivo selecionar os pais mais aptos, após terem sido selecionados, para se reproduzirem e assim aumentar a probabilidade de termos indivíduos na próxima geração que possuam boas soluções para o problema.

Um modo bastante utilizado para esta seleção é na forma de uma roleta [REZENDE, 2002]. Nela, indivíduos mais bem adaptados, e, por conseguinte, com uma nota maior obtida neste critério, terão mais números desta roleta (fatia de pizza maior). Assim, terão mais chances, no sorteio, de serem mantidos e se reproduzirem entre si para gerarem os indivíduos da próxima geração.

Outros mecanismos de seleção podem permitir um tratamento matemático mais rigoroso [REZENDE, 2002].

O operador de mutação deve ter uma taxa pequena de incidência sobre a geração de novos indivíduos, mas deve ser aplicado para ver se a solução obtida ao final é mais otimizada.

Outro operador, o de cruzamento, é o responsável pelo rearranjo do material genético entre os indivíduos selecionados, fazendo com que os novos indivíduos possuam características de seus pais. Esta troca de segmentos de material genético (crossover) é realizada em taxa muito superior à da mutação. Pode ser das seguintes formas [REZENDE, 2002]:

- Em um ponto: O ponto de cruzamento é escolhido, e a partir deste ponto as informações genéticas dos pais serão trocadas. As informações anteriores a este ponto nos pais serão mantidas.

Exemplo, suponhamos que tenhamos dois pais :

Pai 1: 111001100
Pai 2: 000101000

Se aleatoriamente o ponto "3" fosse selecionado então teríamos os filhos com as seguintes configurações:

Filho 1: 111101000
Filho 2: 000001100

Assim, até o terceiro caracter a sequência é mantida, do quarto em diante, o material genético foi trocado.

- Multi-ponto: é similar ao do único ponto, mas nesse caso são escolhidos vários pontos, com muitas trocas de segmentos do material genético.

- Uniforme: não se utiliza de pontos de cruzamento, mas determina, através de um parâmetro global, qual a probabilidade de cada variável ser trocada entre os pais.

2.8 O Método Simplex

2.8.1 Uma visão geral do método

O modelo de programação linear reduz um sistema real, a um conjunto de equações ou inequações onde pretendemos otimizar uma função objetivo. O conjunto de equações ou deverá ser, em princípio, um conjunto indeterminado, de forma que o número das soluções ditas "viáveis" é infinito [GOLDBARG, 2000]. Mesmo sabendo que a solução ótima será encontrada em um ponto extremo do conjunto das soluções viáveis, nosso trabalho não será pequeno se desejarmos determiná-lo.

Infelizmente, esse número de pontos extremos ou vértices pode ser muito grande, em número exponencial, em relação às variáveis. Podemos resumir em dois blocos as dificuldades que deverão ser vencidas por quem deseja encontrar a melhor solução possível para um sistema indeterminado de equações lineares, segundo um critério qualquer [GOLDBARG, 2000]:

- Como obter soluções viáveis básicas do sistema de equações.
- Como evitar o teste de todas as soluções viáveis básicas possíveis para garantir a otimização do sistema.

É nesse contexto que o algoritmo simplex destaca-se como uma das grandes contribuições à Programação Matemática desse século [GOLDBARG, 2000]. Trata-se de um algoritmo geral extremamente eficiente para a solução de sistemas lineares, adaptável ao cálculo computacional (apesar de algumas dificuldades clássicas), e cuja compreensão funcional embasará vários outros métodos [GOLDBARG, 2000]. O estudo desse algoritmo é

indispensável para o profissional que deseja dominar as técnicas quantitativas de análise e solução de problemas em um contexto razoavelmente avançado.

2.8.2 Fundamentos teóricos do simplex

O simplex é um algoritmo. Genericamente, podemos entender por algoritmo qualquer estratégia para solucionar problemas.

Ele se utiliza de um ferramental baseado na Álgebra Linear para determinar, por um método iterativo, a solução ótima de um Problema de Programação Linear (PPL) [GOLDBARG, 2000]. Sua concepção básica é simples, e por isso mesmo, eficiente. Em linhas bastante gerais, o algoritmo parte de uma solução viável do sistema de equações que constituem as restrições do PPL, solução essa normalmente extrema (vértice). A partir dessa solução inicial vai identificando novas soluções viáveis de valor igual ou melhor que a corrente.

O algoritmo, possui um critério de escolha que permite encontrar sempre novos e melhores vértices do envoltório convexo do problema, e um outro critério que consegue determinar se o vértice escolhido é ou não um vértice ótimo [GOLDBARG, 2000].

O Simplex segue então um procedimento sistemático para resolver um problema, movendo de um ponto extremo, a um ponto melhor da função objetivo [MOKHTAR, 1979].

Os componentes básicos do modelo de programação linear, são [JESUS, 2001]:

- Função objetivo;
- Restrições;
- Variáveis de decisão;

2.8.3 O Algoritmo Primal Simplex

O algoritmo Simplex descreve uma sequência de passos para a solução de sistemas de equações lineares sujeitos a uma função objetivo. Basicamente, ele dispõe sobre três situações [GOLDBARG, 2000]:

- O método de inversão da matriz básica *m* x *n* deduzida a partir de *A*, uma matriz de restrições *m* x *n*.
- As condições de troca de variáveis dentro da matriz básica, para que exista garantia de uma contínua melhoria da solução ao longo do desenvolvimento dos passos do algoritmo.
- As regras de parada do algoritmo e a interpretação dessa situação final.

Eventualmente, o algoritmo pode ser adaptado para promover a escolha da base viável inicial, ou solução viável de partida. Contudo, a essência do método desconhece o problema da escolha de uma base viável inicial [GOLDBARG, 2000].

A discussão do primeiro aspecto envolvido no algoritmo, ou seja, o método de inversão da matriz básica, é bastante evidente na apresentação dos "quadros" do simplex e de suas operações de "pivoteamento". O método normalmente sugerido na literatura é o das *operações elementares* [GOLDBARG, 2000].

Essa técnica permite que a cada passo do algoritmo o esforço de inversão já despendido em iterações anteriores seja completamente aproveitado. É importante que se entenda que não existe, de fato, um compromisso do algoritmo com um método de inversão específico para a matriz básica [GOLDBARG, 2000]. Em última análise, o método simplex não obriga que a matriz tenha que ser invertida por um método de pivoteamento, apesar de que todo o seu raciocínio seja aplicado tradicionalmente junto com essa técnica [GOLDBARG, 2000].

O segundo ponto é abordado por um critério bastante simples que envolve o cálculo da possível contribuição para o acréscimo ou decréscimo da função objetivo (conforme o

caso: maximização ou minimização) com a possível entrada na base de uma variável não básica. O critério aponta a escolha da variável de maior contribuição imediata [GOLDBARG, 2000]. A eficiência do método Simplex e seu extraordinário poder de funcionar na prática está associado ao critério adotado nesse cálculo [GOLDBARG, 2000].

O terceiro tema diz respeito ao teste de parada que inclui a identificação das condições em que não existe mais a possibilidade de que uma troca de variáveis na base possa melhorar o critério de otimização, ou ainda situações em que é identificado um comportamento patológico de crescimento indefinido ou de inviabilidade do problema [GOLDBARG, 2000].

2.9 A Dualidade em programação matemática

Dualidade é um conceito amplo que engloba a possibilidade do tratamento de duas naturezas distintas de uma mesma entidade. Inúmeros fenômenos físicos e químicos podem ser representados por modelos cujas estruturas e comportamentos são iguais; contudo, podem ser interpretados de modos completamente diferentes [GOLDBARG, 2000]. Na economia, engenharia e física esses casos são até comuns.

No caso dos modelos matemáticos, a definição de dualidade tem um cunho próprio e associado ao processo de solução e aplicação prática dos modelos. Em ambas as vertentes, a dualidade é hoje um instrumento indispensável [GOLDBARG, 2000].

É definido como *duais* um par de modelos de programação matemática denominados *primal* e *dual*, que preservam as seguintes condições [GOLDBARG, 2000]:

- Possuem funções objetivo simétricas, ou seja, se o primal for de minimização o dual será de maximização e vice-versa. Alguns autores fixam o primal como um problema de minimização ou de maximização, contudo isso não é necessário para caracterizar a dualidade.

- Possuem simetria na descrição das restrições, ou seja, se na forma canônica o primal possui restrições ≤ então o dual possuirá restrições ≥.

- Os termos independentes no primal surgem como os coeficientes da função objetivo no dual e vice-versa.

- O número de restrições do primal é igual ao número de variáveis do dual e vice-versa.

- A matriz de restrição do primal é a transposta da matriz de restrição do dual e vice-versa.

2.10 Um algoritmo para computar o *Nucleolus* em um jogo de árvore geradora mínima

Como já foi dito, o *nucleolus* consiste de uma alocação de custos otimizada através de iterações sucessivas de programação linear que maximizam lexicograficamente as soluções possíveis do núcleo.

Este algoritmo de obtenção do *nucleolus* [GEFFARD, 1997] produz justamente uma sequência de conjuntos-solução, o último deles o *nucleolus*, ao se resolver a última iteração da Programação Linear. No entanto, ele difere do Simplex por não apenas manipular variáveis que entram ou saem da base, mas ele também gera novas restrições.

Existe o problema de sempre, o de que o número exponencial de restrições torna o cálculo impraticável. Mas segundo [GEFFARD, 1997] podemos obter o *nucleolus* com *(n-1)* passos relevantes de Programação Linear, onde *n* é o número de jogadores, ou usuários da rede.

Na descrição do algoritmo, consideremos [GEFFARD, 1997]:

- $\boldsymbol{N} = \{1,2,3,\ldots n\}$ um conjunto de usuários;
- Uma coalizão S como um subconjunto de N;
- $c(S)$ o custo de árvore geradora de custo mínimo (AGCM) conectando a coalizão S ao provedor de serviço.

- $x = (x_1, x_2, \ldots, x_n)$ como um vetor de n.

Dadas as convenções para a descrição do algoritmo, temos que um vetor de alocação de custos é definido da seguinte forma [GEFFARD, 1997]:

$$\sum_{i \in N} x_i = C(N)$$
e
$$x_i \leq C(\{i\}) \text{ para todo } i \in N$$

Isto significa que a soma dos custos que cada jogador vai arcar participando da solução geral é igual ao custo gasto com a rede toda, e o custo que cada coalizão vai arcar deve ser igual ou menor ao que ela pagaria ao se ligar de forma autônoma ao fornecedor do serviço.

O núcleo do jogo é o conjunto de n-vetores $x = (x_1,\ldots,x_n)$ que é determinado por [GEFFARD, 1997]:

$$\sum_{i \in N} x_i = C(N)$$
e
$$\sum_{i \in S} x_i \leq C(S) \quad \forall S \subset N$$

Ou seja, o núcleo é formado por todos os vetores de alocação de custos em que, para cada um destes vetores, cada usuário vai pagar igual ou menos que pagaria ao se ligar de outra forma, considerando qualquer coalizão possível. O núcleo, então, é o conjunto de todas as soluções viáveis.

2.10.1 O *nucleolus*

De maneira formal, o *nucleolus* pode ser descrito assim [GEFFARD, 1997]:

Seja $e(x,S)$ o excesso de qualquer coalizão $S \subset N$, em que:

$$e(x,S) = C(S) - \sum_{i \in S} x_i$$

O *nucleolus* é o conjunto:

$$\{x \mid x \in X, \text{ se } m \in X, \text{ então } E(x) \geq_{lo} E(m)\}$$

Nucleolus é um vetor de alocação se pertence ao núcleo, e, considerando um outro vetor *m* qualquer, o *nucleolus* tem seu excesso maior lexicograficamente do que *m*.

Em suma, como já foi dito, o *nucleolus* é o vetor de alocação de custos em que, para qualquer coalizão dele considerada, o custo que ela vai arcar participando da solução geral é igual ou menor ao de qualquer outro obtido em quaisquer outros vetores de alocação de custos do núcleo. O n*ucleolus* é alcançado após uma sucessão de vetores tratados por programação linear, onde um após o outro tem o excesso da coalizão mais insatisfeita maximizado.

A sequência de $(LP(k))_{1 \leq k \leq n-1}$ (onde *LP* é a Programação Linear e *n* é o número de jogadores) iterações na Programação Linear são resolvidas [GEFFARD, 1997]. Seus conjuntos solução otimizados geram uma sequência, $(\Omega(i))_{1 \leq k \leq n-1}$ em que o conjunto solução $\Omega(i+1)$ está contido no conjunto $\Omega(i)$ (aquele é mais limitante do que este, garantindo um excesso maior ou igual para uma ou mais coalizões), convergindo para o *nucleolus* [GEFFARD, 1997].

Essa idéia de convergência usa o conceito de ordenar lexicograficamente os conjuntos, maximizando sucessivamente os excessos para todas as coalizões. Ou seja, o *nucleolus* é o vetor pertencente ao núcleo que possui o maior excesso em todas as restrições se comparado a qualquer outro vetor de alocação de custos do núcleo. Mas a obtenção deste segue a ordem de maximização da coalizão mais insatisfeita (menor excesso) à menos insatisfeita.

Muitas *LP's* podem ser desconsideradas no cálculo do *nucleolus* [GEFFARD, 1997].

Essas melhorias no algoritmo são extremamente úteis, e efetivas contra instâncias de problemas NP-Difíceis (como se considerarmos nós de transbordo, ou nós de Steiner) [GEFFARD, 1997].

A primeira *LP* da sequência, *LP(1)* é a seguinte [GEFFARD, 1997]:

Max ε_1:

$$
\begin{array}{lll}
 & | \quad \sum_{i \in 1..n} xi = C(N) & \\
 & | & \\
LP(1) & | \sum_{i \in S} xi + \varepsilon_1 \leq C(S) & | \ \forall S \subset \{1..N\} \\
 & | & | \ S \neq \varnothing;\ S \neq \{1..n\} \\
 & | \ \forall i \ xi \geq 0; \quad \varepsilon_1 \geq 0 &
\end{array}
$$

Após um determinado número de *LP's*, obtemos um conjunto solução para o problema [GEFFARD, 1997].

Vamos chamar de *P(k)* um conjunto de soluções possíveis de obtermos de *LP(1)*, e *O(k)* é o conjunto de solução ótimas.

Consideremos também um conjunto $\Omega(k)$ que é definido da seguinte forma [GEFFARD, 1997]:

$$\Omega(k) = \{(x_1,\ldots,x_n) / \exists\ \varepsilon_k st.(x_1,\ldots,x_n,\ \varepsilon_k) \in O(k)\}$$

O conjunto $\Omega(k)$ contém os vetores de alocação otimizados, em que, na atual iteração, todos eles terão todas suas restrições se aproximando da igualdade com o excesso conseguido em *LP(k)*, pertencentes ao conjunto *O(k)*.

Ao se resolver *LP(1)*, será produzido tanto o excesso ε_1 , ótimo, quanto x_1 , um vetor de custos otimizado para este excesso [GEFFARD, 1997]. O x_1 visa maximizar primeiramente o excesso para a coalizão mais insatisfeita. Assim, qualquer coalizão se beneficiará ao menos com o excesso ε_1 em x_1.

Desta primeira solução ótima $(x_1|\varepsilon_1)$ podemos definir *O(1)*, como o conjunto de todas as soluções ótimas de *LP(1)*. *O(1)* é definido transformando as inequações relevantes de *P(1)* em equações, que limitam esta solução ótima.

Inequações atingem a igualdade, ou melhor, se tornam equações, quando as mesmas têm seu excesso maximizado.

Então, de *O(1)* nós podemos obter *Ω(1)*, ou o conjunto de todos os vetores para os quais qualquer coalizão se beneficiará ao menos com excesso ε_1.

No próximo passo o que nos interessa são os vetores de *Ω(1)* que maximizam o ganho das coalizões menos satisfeitas dentre as seguintes [GEFFARD, 1997]: em todas aquelas inequações dos vetores de *Ω(1)* que não se tornaram equações podemos esperar um excesso maior que ε_1. Consequentemente, nós computamos *LP(2)* para obtermos *Ω(2)* de *O(2)*.

Assim, à medida que as restrições que têm seus excessos maximizados são transformadas em equações e podem ser desconsideradas para a próxima *LP*.

Chamemos de *OPT(k)* o conjunto de coalizões relacionadas às inequações de *P(k)* que se tornaram equações em *O(k)*. Assim, *LP(2)* vai ser [GEFFARD, 1997]:

Max ε_2:

$$
\begin{array}{lll}
 & \sum_{i \in 1..n} x_i = C(N) & \\
 & \sum_{i \in S} x_i = C(S) - \varepsilon_1 \quad \forall S \in OPT(1) & \\
 & & \\
LP(2) & \sum_{i \in S} x_i + \varepsilon_2 \leq C(S) & \forall S \subset \{1..N\}; \\
 & & S \neq OPT(1) \\
 & & S \neq \varnothing;\ S \neq \{1..n\} \\
 & \forall i\ x_i \geq 0;\ \varepsilon_2 \geq 0 &
\end{array}
$$

O resultado de *LP(2)* nos fornece *O(2)* e, a propósito, também *Ω(2)*, que é o conjunto de vetores que maximizam o excesso das coalizões mais insatisfeitas em segundo lugar.

O processo continua com (*P(k)*, *LP(k)*, *O(k)*, *Ω(k)*) para $k \geq 3$.

Para obtermos o *nucleolus*, que é o ponto principal, o processo vai continuar até *LP(k)* que produz *O(K)* atingido em uma única solução. Portanto, ***Ω(k)* = *Nucleolus***!

Max εk:

$$
\begin{array}{lll}
 & | \ \sum_{i \in 1..n} x_i = C(N) & \\
 & | \ \sum_{i \in S} x_i = C(S) - \varepsilon_L & | \ \forall S \in OPT(L) \\
 & | & | \ \forall L;\ 1 \leq L < k \\
 & | & \\
LP(k) & | \ \sum_{i \in S} x_i + \varepsilon_k \leq C(S) & | \ \forall S \subset \{1..N\}; \\
 & | & | \ S \neq OPT(L); \\
 & | & | \ 1 \leq L < k; \\
 & | & | \ S \neq \varnothing;\ S \neq \{1..n\} \\
 & | \ \forall i \ x_i \geq 0; \quad \varepsilon_k \geq 0 &
\end{array}
$$

Esta é a descrição geral do algoritmo, mas, no entanto, este não é praticável, pois trabalha com todas as restrições.

Embora esta seja uma dificuldade maior na obtenção do *nucleolus*, há como ser simplificada. Isto porque, na realidade, muitas das iterações descritas podem ser eliminadas, ou seja, muitas *LP's* não são indispensáveis para obtermos o *nucleolus*[GEFFARD, 1997].

Logicamente, se nem todas as restrições são necessárias, nem todas iterações são necessárias. Na construção no poliedro *O(.)* apenas uma subsequência de *LP's* são necessárias.

Porém, não é fácil determinar quais das *LP's* são relevantes para a resolução do problema.

Reduzindo-se o número de *LP's* para se resolver o problema, temos um novo programa para achar o *nucleolus*.

Este levará em conta ainda a restrição de que o custo total da rede deve ser coberto, e todas as restrições em forma de inequação, que terão seus excessos maximizados, tornando-se equações[GEFFARD, 1997].

Há no máximo *(n-1)* equações em qualquer das *LP(.)'s* que serão consideradas [GEFFARD, 1997]. A geração de restrições apenas dirá respeito às inequações [GEFFARD, 1997]. Assim, as iterações não abrangerão todas as inequações, diminuindo de um número exponencial para

linear de inequações, para um determinado número de jogadores.

Assim, esta *LP* não envolve todas as inequações. Chamando esta *LP* modificada de *RP*, temos que o excesso ε_k obtido em uma iteração *RP(k)* é otimizado, superior ao excesso ε_k obtido em uma *LP* tradicional, na iteração *LP(k)*[GEFFARD, 1997].

Consequentemente, o algoritmo de geração de restrições procura coalizões cujas inequações que as representem não sejam ainda uma equação, e nem satisfeitas. Formalmente [GEFFARD, 1997]:

$$\left|\sum_{i \in S} x_i + \varepsilon_k > C(S)\right.$$
$$\left| S \notin OPT(L); \quad \forall L;\ 1 \leq L < k\right.$$
$$\left| S \neq \varnothing; \quad S \neq \{1..n\}\right.$$

No que tange à complexidade, isto é equivalente a fazer uma busca a uma coalizão que [GEFFARD, 1997]:

$$S = Min_{S \subset \{1,\ldots,n\},\ S \notin Q} \{MCST(S) - \sum_{i \in S} x_i\}$$

em que: $|Q$ *é um subconjunto de N*
$|x = \{x_1,\ldots,x_n\}$ *um vetor de custos*

Entretanto, esse processo de geração de restrições não é simples, dada a complexidade polinomial do algoritmo de árvore geradora mínima, tornando este problema NP-Difícil, similar ao problema da árvore de Steiner [GEFFARD, 1997]. Há duas heurísticas para a busca por restrições insatisfeitas. Uma é usando-se Têmpera Simulada[GEFFARD, 1997].

No entanto, há um ponto em que a heurística não pode mais achar novas restrições insatisfeitas [GEFFARD, 1997], então usa-se um algoritmo Branch & Bound [GEFFARD, 1997]. O fato de que há a necessidade de usar-se Branch & Bound é um fator limitante ao processo, mas garante a exatidão do processo de geração de restrições [GEFFARD, 1997].

2.10.2 Os poliedros ômega

O método Simplex estima um *LP* ótimo para uma solução em particular, que é um vértice do poliedro de soluções, não sendo obtidos diretamente dos conjuntos *O(.)* (conjunto de soluções ótimas).

Os vários vértices do poliedro hiperdimensional são as possíveis soluções pertencentes ao núcleo. Será descrito como definir o poliedro de soluções otimizadas com os vértices fornecidos pelo algoritmo Simplex [GEFFARD, 1997].

Chamemos de *(Γs)* o espaço delimitado que consiste de todos os vetores solução $(x_1,\ldots,x_n, \varepsilon)$ limitados pela seguinte inequação:

$$\sum_{i \in S} x_i + \varepsilon \leq C(S)$$

Ou seja, o espaço onde temos todas as soluções possíveis, em que todas as coalizões pagarão, no máximo, o custo que elas pagariam ao se ligar por conta própria ao fornecedor do serviço. O conjunto solução otimizado de *LP(k)*, *O(k)*, é a interseção do poliedro solução, *P(k)*, e um espaço limitante, *(Γs)*. A questão é saber qual *(Γs)* contém todas as soluções ótimas, delimitadas por ele [GEFFARD, 1997].

Mas o problema é que algumas soluções ótimas podem também fazer parte de alguns *Γs* em que outras soluções não fazem parte. Como o caso do vértice ótimo obtido no Simplex quando há mais de uma solução ótima.

Pode ser que uma dada restrição abranja todas soluções ótimas, e outra não [GEFFARD, 1997]. Quer dizer, uma restrição delimite uma região de todas prováveis soluções que outra não delimita, ou delimita de maneira incompleta [GEFFARD, 1997]. Estas inequações são chamadas de *equações fortes* (***TI***, do inglês tight inequality) [GEFFARD, 1997].

Dessa forma, o *Γs* que contém uma solução ótima, mostra-se “amarrado” ou delimitado por uma *TI* para esta solução. Definimos os *O(.)* poliedros como sendo aqueles formados por relevantes *TI’s*, que limitam fortemente as

soluções ótimas. Estas *TI's* que são limites para todas as soluções ótimas são chamados de **inequações altamente limitantes,** ou ***HTI*** (Hard Tight Inequality).

TI's em que seu coeficiente *dual* relacionado é não nulo, são *HTI's*, como foi provado[GEFFARD, 1997].

Só as inequações relevantes já são suficientes para obtermos a solução ótima.

Quando estas *HTI's* tornam-se equações no processo que cria o poliedro P(.), esse novo sistema define nada menos que o poliedro O(.).

Existe ao menos uma *HTI* cuja variável dual relacionada é não-nulo[GEFFARD, 1997]. Além disso, uma *HTI* com uma variável dual nula é redundante na definição do conjunto ótimo [GEFFARD, 1997].

2.10.3 Modificação da geração de restrições para computar o *nucleolus* com *(n-1)* iterações

Esta proposta de algoritmo produz uma sequência de conjuntos *x* que geram o *nucleolus*. Seria útil transformar em equações um tipo de *HTI* [GEFFARD, 1997].

Estas *HTI's* cortam algumas soluções dos poliedros *P(.)* (são mais limitantes), mas deixam inalterados os conjuntos *Ω(.)*. Ou seja, elas não aparam todas as soluções quando se transformam em equações.

Há outro modo de se identificar se uma *TI* é uma *HTI* (o primeiro é verificar se seu coeficiente relacionado *dual* é não nulo), que é saber quais equações ou inequações são dependentes [GEFFARD, 1997]..

Duas equações (*E*) ou inequações são *dependentes* quando suas coalizões (*S*) relacionadas têm vetores de alocação de custos *linearmente dependentes* [GEFFARD, 1997], como por exemplo:

$$E1(S1) = E1(\{1,2\}) = \{1,2,3,4,5,6,7\}$$
$$E2(S2) = E2(\{4,5\}) = \{2,3,4,5,6,7,8\}$$
$$E3(S3) = E3(\{1,2,4,5\}) = \{3,5,7,9,11,13,15\}$$

$$N=\{1,\ldots,7\}$$

A equação *E1*, representando a coalizão *S1*, formada pelos jogadores *{1,2}*, juntamente com a equação *E2*, formam uma relação de dependência com *E3*, sendo que esta última é dependente das duas primeiras, pois seu vetor de custos tem cada um de seus elementos como função da soma dos elementos a ele correspondentes em *E1* e *E2*.

$$S3 = S1 + S2$$

É inútil transformarmos em equações as *HTI's* que são dependentes de outras equações [GEFFARD, 1997]. Porque uma *HTI* dependente não nos dá nenhuma informação relevante para as variáveis x_i. Assim, para este tipo de *HTI*, o $\sum_{i \in S3} xi$ é determinado pelo sistema de equações [GEFFARD, 1997].

Se transformarmos em equações o *HTI*, esta nova equação é dependente linearmente de equações que já estão formadas [GEFFARD, 1997]. E é bem conhecido que se adicionarmos equações linearmente dependentes em um sistema, deixamos esse novo sistema igual ao outro, ou seja, inalterado. Se ignorarmos estas restrições dependentes, produziríamos conjuntos $\Omega(.)$ diferentes [GEFFARD, 1997].

Assim, o algoritmo de geração das restrições é alterado para ignorar *HTI's* dependentes [GEFFARD, 1997], e o poliedro $\Omega(k)$ perde ao menos uma dimensão a cada iteração, pois, para cada dimensão, há ao menos uma *HTI*. Assim, computamos o *nucleolus* em apenas *(n-1) LP(s)* que são relevantes [GEFFARD, 1997].

Capítulo 3 - Resultados e discussão

3.1 O Problema

Por se tratar de um problema não trivial, foram envolvidos neste estudo tópicos como a Teoria dos Jogos, Teoria dos Grafos, Programação Linear (Simplex), Algoritmos Genéticos, Álgebra, Geometria Plana, Analítica e Espacial, teorema da dualidade matemática e redes de acesso.

Nas próximas seções, serão apresentadas análises e considerações por diversos ângulos do problema, e idéias que, mesmo reconhecidamente superadas por outras mais elaboradas, foram mantidas no texto como marcas deixadas no percurso de interpretação dos fundamentos teóricos e matemáticos do problema. Becos sem saída e busca de respostas usando-se vários recursos foram necessários para uma visão geral do problema, e necessários no processo de construção da idéia da heurística.

Nesta discussão de resultados será evitado grande parte dos formalismos matemáticos, que visa abstrair ao máximo idéias, conclusões e interpretações pessoais.

3.2 Modelagem do problema

A rede é modelada como um grafo, que é a estrutura matemática mais indicada para esta tarefa. Nós representam clientes e fornecedor, e arestas representam as ligações entre eles, com a vantagem de poderem ter custos e capacidades associadas.

Conciliar uma solução em que o custo da rede seja coberto, ou seja, a soma do que todos os usuário irão pagar deve ser igual ao custo total da rede, e repartir o ganho obtido em ser ligar os usuários entre si, entre os próprios usuários de forma mais justa possível, é o problema principal encontrado.

A Teoria dos Jogos foi utilizada porque este problema, bem como uma vasta gama de problemas em áreas diversas, pode ser encarado como um jogo. Um jogo em que os usuários são jogadores cooperando entre si com o objetivo de pagarem o menos possível. Esta é a razão de ter sido utilizada a vertente *cooperativa* da Teoria dos Jogos.

Temos que representar então o problema como se fosse um jogo, e definirmos seu objetivo. O jogo cooperativo em forma de função característica é definido da seguinte forma:

Seja $N=\{1,2,...,n\}$ o conjunto finito de jogadores, $c:P->R$, $c(\varnothing)=0$ será a função característica $\forall S \subseteq N$ sendo $c(N)$ o custo que deve ser repartido entre os jogadores, então o par $(N;c)$ é chamado jogo cooperativo.

Se cada usuário, ou jogador, ao se ligar sozinho ao fornecedor, ele teria que arcar com o custo desta ligação. Mas, participando da solução que leva em conta uma rede ligando todos os jogadores ao fornecedor, ele deverá pagar, no máximo, este valor.

Isto pode ser entendido da seguinte forma. Com dois jogadores, próximos um do outro, e distantes do fornecedor, teriam que pagar, individualmente, o custo da ligação individual que os liga àquele. Mas se estiverem ligados entre si, apenas uma ligação é necessária para conectá-los ao fornecedor, ligação esta que vai partir do jogador mais próximo a ele. Assim, economiza-se evitando-se a ligação do jogador mais longe do fornecedor, estando esta reduzida a uma pequena ligação ao jogador próximo a este.

Esta sub-rede ligando os dois jogadores ao fornecedor pode ser obtida com um algoritmo de **"AGM"** (Árvore Geradora Mínima), que é a forma mais barata de se ligar um fornecedor aos usuários. O fornecedor é a raiz desta árvore

e os usuários são os nós folha ou estão no caminho entre a raiz e as folhas.

O algoritmo de AGM nos fornece, mediante a informação de posicionamento e distâncias entre a raiz e os nós dos usuários, o novo custo que estes irão arcar se ligarem desta forma ao fornecedor. A forma mais barata de se fazer esta ligação realmente é com uma árvore.

Modelando esta sub-rede em forma de função característica de Teoria dos Jogos, obtemos um custo máximo que os dois jogadores irão pagar se participarem da solução geral. De novo, este custo deverá ser de no máximo o valor que eles pagariam ao se ligar por conta própria (mas um ajudando o outro, formando uma coalizão, que é um subgrafo) ao fornecedor.

Pode ser que este custo seja igual ao custo que eles teriam se ligassem cada um por conta própria ao fornecedor. Isto pode ser possível se o fornecedor e os dois usuários estiverem dispostos em forma de triângulo equilátero, por exemplo. Assim, tanto faz ligar cada um dos dois usuários independentemente ao fornecedor, quanto ligar um usuário ao outro e um dos dois ao fornecedor.

Esta nova função que leva em conta caso só houvessem estes dois jogadores ligados ao fornecedor chama-se **restrição.** Ela se apresenta na forma de uma inequação, por exemplo:

$$x1+x2 \leq 300$$

Aqui, *x1* representa o custo que um usuário deve arcar, *x2* é o custo que o outro usuário deve arcar, se ligarem juntos ao fornecedor. Este custo de 300 é obtido por um algoritmo de AGM, e é o quanto eles dois juntos devem pagar no máximo.

Para cada coalizão (ou combinação) possível de jogadores representa-se uma restrição. Estas são necessárias para “amarrarem” a solução ótima em que todos participem. Se todas participarem, pagarão menos, e terão um excesso maior.

Mas, segundo estudos, é necessário apenas *n* restrições para que achemos o *nucleolus*, com *n* jogadores. Um dos objetivos do trabalho foi o de analisar essa

possibilidade e formular uma explicação própria a respeito de porque esta redução é possível.

Há 2^n-1 combinações possíveis entre *n* jogadores, pois estamos excluindo o conjunto vazio.

3.3 Porquê *n* restrições? (possibilidades)

Como este trabalho é exploratório, foi importante levantar algumas dúvidas e idéias de porque somente n restrições seriam relevantes para se chegar à solução.

3.3.1 Possibilidade 1

Dentre as várias ligações possíveis em um grafo completo, seriam suficientes apenas *n* delas para formar o grafo, ou uma árvore geradora mínima otimizada? Obviamente haveria uma relação com o número de jogadores, pois *n* excluiria uma desnecessária ligação do provedor do serviço com ele mesmo. Mas, dado um grafo, o algoritmo de AGM nos fornece apenas a árvore otimizada. Então, a irrelevância da maioria das restrições poderia dizer respeito a combinações entre jogadores desnecessárias.

Mas o problema não é combinação entre jogadores, mas sim a divisão de custos entre eles. Hipótese descartada.

3.3.2 Possibilidade 2

Outra possibilidade seria a respeito do número de dimensões do poliedro hiperdimensional. Se for considerando cada eixo cartesiano representando uma restrição, delimitando assim no espaço hiperdimensional uma região onde todas as possíveis soluções para o problema podem estar, temos para *n* eixos, um espaço *n*-dimensional.

Por exemplo, para dois jogadores, um plano cartesiano representaria, com dois pontos, um em cada eixo, uma reta, dividindo o espaço bidimensional em duas regiões? (Fig. 4.1). Ou delimitaria uma área retangular em que dois de seus lados seriam limites desta área, cada um dos dois paralelos, um ao eixo *x*, outro ao eixo *y*? (Fig. 4.2).

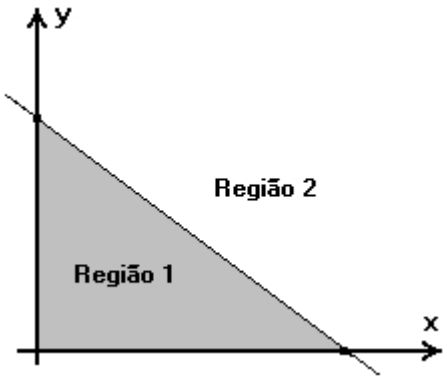

Figura 4.1. Regiões delimitadas por uma restrição

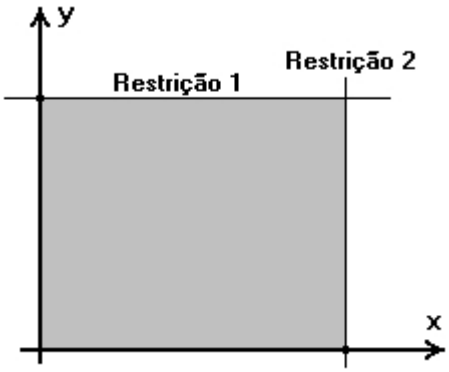

Figura 4.2. Região retangular delimitada por duas restrições

No caso de uma inequação, digamos, $x \leq 3$, o eixo *x* delimitaria, no ponto $x=3$, em um espaço tridimensional, um plano paralelo ao plano *yz*, cortando ortogonalmente no ponto 3. Qualquer valor no espaço, seja ela bi ou

tridimensional, abaixo deste valor de *x* pelo qual passa um plano imaginário, seria uma solução válida para a inequação (Fig. 4.3). Assim, seria natural entender os dois pontos em *x* e *y* delimitando uma região retangular no espaço.

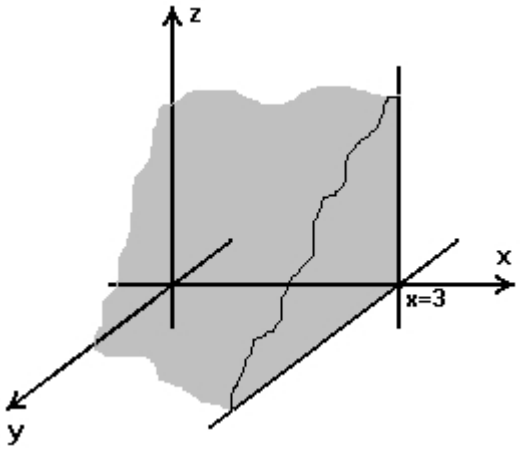

Figura 4.3. Região delimitada pela restrição *x≤3*

Para três dimensões, teríamos um sólido, regular, caso a interpretação correta fosse essa, com cada restrição sendo uma região delimitada por uma reta ortogonal ao eixo dimensional.

Mas e para um espaço hiperdimensional? Teríamos um poliedro hiperdimensional.

Isso se considerarmos cada dimensão representando unicamente uma restrição. Nesse caso, cada face do poliedro seria um recorte de um plano ortogonal àquele eixo dimensional.

Por outro lado, voltando ao exemplo do início da Subseção 4.3.1, duas restrições, uma no eixo *x*, outra no eixo *y*, formem realmente uma reta que é resultado das duas restrições, então não podemos mais considerar cada face do poliedro como sendo uma região do plano ortogonal àquele eixo. Isso pode ser facilmente visualizado em duas dimensões.

Uma reta que corta o plano *xy* a 45 graus, no ponto, digamos *(x,y) = (3,3)*, temos uma região delimitada pelas restrições que não é mais retangular (Fig. 4.4). Para três

dimensões, o que vamos ter é um plano, que corta dois eixos, caso este seja paralelo a qualquer um dos três planos formados pelo eixo cartesiano tridimensional (*xy, xz ou yz*), ou cortando os três eixos, em caso contrário.

Assim, cada eixo não cortaria o plano a ele ortogonal, formando uma face do poliedro hiperdimensional, mas sim, esses teriam pontos que formariam os vértices do poliedro.

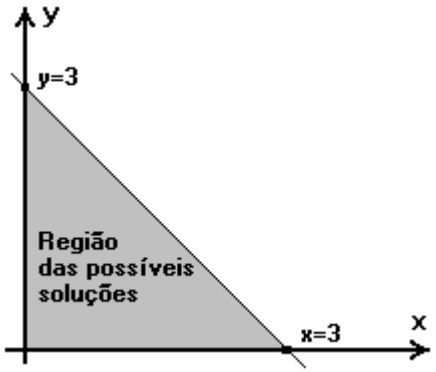

Figura 4.4. Região delimitada por uma restrição que corta xy a 45 graus

Há entretanto uma grande complicação para este modelo. E se for considerado de que há mais restrições a serem representadas do que dimensões no hiperespaço? (o termo aqui é usado para indicar um espaço com mais de três dimensões).

Isto seria até mais natural de se considerar, pois se temos *n* jogadores, há $2^n - 1$ combinações de coalizões possíveis entre eles, e consequentemente, 2^n-1 restrições.

Temos porém muito mais restrições do que jogadores, obviamente, numa relação de crescimento exponencial, daí advém toda a dificuldade de obtenção por meios convencionais do *nucleolus*.

Mas cada eixo dimensional vai representar, afinal, um jogador ou uma restrição? Se tivermos um poliedro *n*-dimensional, com n igual ao número de jogadores, e quisermos representar TODAS as restrições, então obviamente temos mais restrições que têm que ser

representadas neste espaço do que dimensões desse mesmo espaço.

Exemplo:

Com 5 jogadores, temos 31 restrições. Se temos uma hiperdimensão de 5 dimensões temos que representar 31 restrições!

Se usarmos cada dimensão para representar uma restrição, não há problema. Uma restrição para cada dimensão. Certo? Um provável desperdício de matemática, computação e imaginação.

Esse é o ponto chave da questão. Seriam todas as restrições imprescindíveis à construção do poliedro hiperdimensional? Não seriam várias delas (precisamente, $\mathbf{2^n - 1 - (n-1) = 2^n - n}$) restrições totalmente irrelevantes para a construção desse modelo?

E como seria isso?

Várias podem ser as visualizações. Primeiro. Não existiria apenas um, mas vários poliedros. Se um estivesse totalmente contido ou fosse totalmente coincidente (todos seus vértices e consequentemente faces seriam os mesmos de outro poliedro), então esse seria o único dos dois poliedros que precisaríamos considerar, pois ele delimita um espaço mais restrito do que o outro, o que significaria um custo menor para todas os jogadores pagarem.

Se vários poliedros que fundissem e se interpenetrarem, teríamos um pequeno poliedro, central, mais restrito, e o único considerado, juntamente com vários pedaços, poliedros menores, com o mesmo número de dimensões, ou até menos (se um poliedro hiperdimensional for partido, é possível que cada pedaço tenha menos dimensões. Por exemplo, um cubo cortado ao meio ainda teria duas metades tridimensionais, mas se for cortado em infinitas fatias, cada uma delas teria, na dimensão perpendicular aos cortes, um tamanho infinitesimal, considerado zero, formando fatias de duas dimensões).

Estes pedaços seriam entendidos como poliedros menores aderidos ao lado de fora do poliedro menor. Poderiam ser todos descartados, pois teriam um limite maior do que o poliedro interno. Novamente, apenas um poliedro seria suficiente para restringir o espaço de possíveis regiões.

Outra possibilidade para afirmar a irrelevância da maioria de restrições seria a seguinte. Dada uma face do poliedro, poderíamos ter, em seu mesmo plano, outra reta (representando uma restrição) que não contribui para a formação desta face. Ou mesmo um vértice que não contribua para essa face.

A questão, nesse caso, é: haveriam faces em que um número menor de restrições que a formam podem ser considerados, e somente elas? Ainda teríamos que descobrir se há de fato, e quais delas são irrelevantes.

Imagine um triângulo equilátero e um triângulo escaleno. A base dos dois triângulos são coincidentes e de tamanho igual, ou seja, é a mesma. Só que o triângulo escaleno tem o vértice superior deslocado à direita do ponto direito de sua base (Fig. 4.5). Imagine também um sólido regular tridimensional, cujas faces sejam triângulos equiláteros (uma pirâmide regular triangular) (Fig. 4.6).

Na face em que temos as duas composições, a do triângulo equilátero e triângulo escaleno, o triângulo escaleno pode ser desconsiderado, pois não acrescenta em nada (não é mais delimitante) ao poliedro tridimensional piramidal. Ele está definindo uma região menos delimitante ao poliedro piramidal, levando-se em conta as outras faces que o compõe.

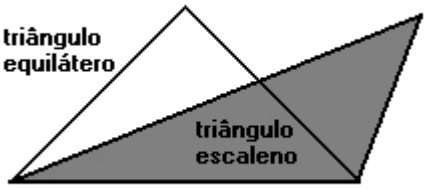

Figura 4.5. Triângulo equilátero e escaleno

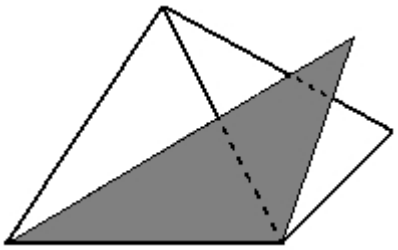

Figura 4.6. Pirâmide regular

Assim, pode ser que só *n* restrições sejam suficientes para construir um poliedro que restrinja e seja ao mesmo tempo o limite superior para todo o núcleo do problema.

Cada solução possível para o problema estaria contida nessa região hiperdimensional formada no interior do poliedro hiperdimensional.

Vetores solução definidos pelos vértices ou sejam coincidentes a faces desse poliedro podem ser entendidos como soluções pouco otimizadas, já que, ao atingirem o limite máximo, atingiram também a igualdade em alguma ou mais de uma restrição. Isso implica que alguma coalizão vai ter que arcar com o mesmo custo que ela já arcaria se ligasse sozinha ao fornecedor do serviço, ou seja, teria um excesso zero.

Soluções inteiramente contidas no interior do poliedro e sem contato com as faces do mesmo teriam todas as coalizões com excesso diferente de zero. Porque, como o poliedro delimita uma região de possíveis soluções, ele é o limite máximo, ou seja, delimita o núcleo. Pontos em seu interior representam vetores de alocação de custos em que nenhuma coalizão está no limite, ou seja, não possuem excesso zero. Por essa linha de raciocínio, facilmente podemos imaginar a melhor solução como sendo aquela que esteja mais longe possível de todas as faces do poliedro, ou seja, aquela que ocupa seu centro absoluto. Esse seria o *nucleolus* do problema.

Um poliedro hiperdimensional com faces regulares, ou seja, um poliedro regular, possui um núcleo (ou um *nucleolus*) bem definido. Mas e um poliedro irregular?

Por não ter um núcleo exato, ou que seja exato mas com coordenadas não inteiras, podemos esperar também um *nucleolus* com valores aproximados.

E no caso de excessos negativos? Aqueles em que fica mais fácil subsidiar o excesso a ter aquela coalizão desistente da solução geral? (Nesse caso, poderíamos pensar em um poliedro não convexo, ou seja, existiria um ponto fora dele, que também seja solução. É pouco provável, já que as soluções estão ou contidas nele, ou em suas faces, e estas são construídas por inequações de primeiro grau e de mesmo peso entre os coeficientes).

Ainda sim a coalizão não ganharia para participar, apenas teria seu excesso subsidiado. Esse subsídio poderia ser de no máximo o valor que a coalizão pagaria. Nesse caso particular, a coalizão passaria a não pagar nada, pois o subsídio teria o mesmo valor do custo a que ela deve arcar na solução geral.

Um vetor solução com coordenadas com valor zero, é aquele que coincide com a origem dos eixos do espaço hiperdimensional. Mas se esse vetor faz parte da solução, então o poliedro deve conter a origem dos eixos, caso contrário esse vetor não pertenceria ao núcleo. E um poliedro que não contenha a origem dos eixos não pode ter cada face cortada ortogonalmente por uma dimensão!

Se temos um poliedro hiperdimensional deslocado da origem dos eixos, então temos faces em que seria impossível serem cortadas ortogonalmente por um eixo dimensional.

Por exemplo. Um cubo num espaço dimensional que não contenha a origem dos eixos, por exemplo, tem um lado de sua base que vai do ponto (x,y,z), $(1,0,0)$ ao ponto $(10,0,0)$, ou seja, esteja totalmente contido no lado positivo do eixo x, e à direita da origem dos eixos não pode ter seu lado que é paralelo ao plano xy cortado pelo eixo z (Fig. 4.7). Logo, não temos as faces que formam o cubo cortadas, todas elas, pelos eixos. Na verdade, nenhuma das faces, nesse exemplo, as são. O máximo que temos é um lado coincidente ao eixo x.

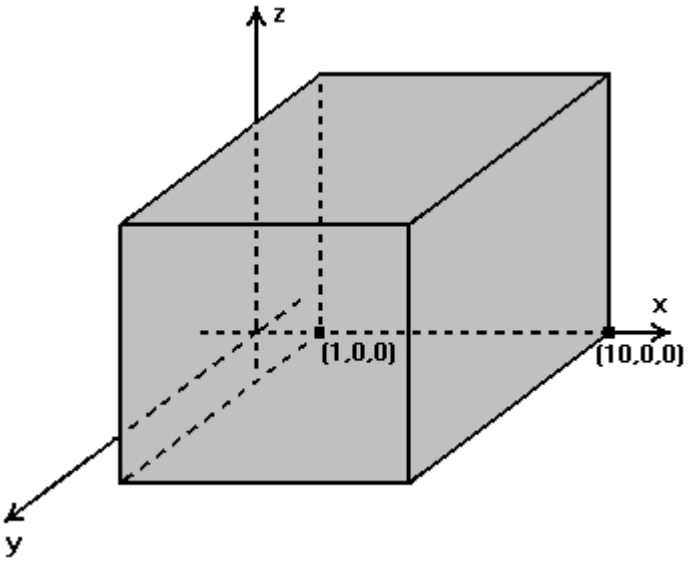

Figura 4.7. Cubo deslocado à direita da origem, no eixo x

Assim, se um poliedro hiperdimensional não possui em seu interior (e não coincidente a uma de suas faces) a origem dos eixos, suas faces não podem ser suficientemente compostas pela idéia de que cada eixo dimensional corta ortogonalmente cada uma de suas faces.

Certamente o poliedro hiperdimensional tem suas faces compostas por mais restrições do que são necessárias, se usarmos todo o espaço de restrições. Cada face é composta por três ou mais arestas (duas arestas não são suficientes). As arestas compostas por retas definidas pelas restrições, e os vértices são pontos de interseção entre elas.

Haveria, então um modo de se achar retas que estejam no mesmo plano a uma face. Basta verificar se a mesma é linearmente dependente de quaisquer outra. Ou seja, dada uma reta num plano, com uma operação, ou de soma, ou de deslocamento, ou de inclinação, ou ainda com a combinação entre elas, obtemos a outra reta. Mas o problema ainda persiste. Dadas n retas no plano, como saber quais delas são suficientes, e somente elas, para formarem uma face do poliedro hiperdimensional?

Dado uma coalizão, por exemplo, entre dois jogadores, temos somente uma restrição que diz respeito aos

dois em particular, representada por uma reta no espaço bidimensional que representa as soluções possíveis para eles arcarem com os custos. Mas há mais retas neste plano, se considerarmos a projeção de cada poliedro delimitado por quaisquer inequações que possuam estes dois jogadores como coeficientes.

3.3.3 Possibilidade 3

Tentar analisar pelo ponto de vista do funcionamento do Simplex.

O Simplex atinge uma solução que se aproxima ao máximo do *nucleolus* da seguinte forma. Na primeira iteração, em todo o espaço de restrições, são achadas aquelas que possuem o mesmo e menor excesso dentre todas as restrições. Assim, elas atingem a igualdade, são fixadas e não serão consideradas na próxima iteração.

Na próxima iteração, ocorre a mesma coisa, as restrições com o menor excesso atingem a igualdade e serão fixadas a partir da próxima iteração, e assim sucessivamente, até que todas as restrições atinjam a igualdade.

O que o método faz então pode ser visto com atenção. Na primeira iteração, as coalizões com menor excesso, ou seja, as menos beneficiadas, terão seus excessos maximizados. Na segunda iteração, não consideraremos mais o menor excesso, mas o segundo menor excesso, já que as restrições que atingiram a igualdade com aquele menor excesso já não farão parte do espaço de busca desta iteração.

Simplificando, o que o método faz é achar as coalizões mais insatisfeitas (por um método de programação linear, onde a função objetivo e de se maximizar o excesso, por exemplo) e aumentar seus excessos, depois as coalizões mais insatisfeitas em segundo lugar (que estão insatisfeitas, mas nem tanto quanto as primeiras) e assim por diante.

Isso traz uma dedução importante. Se mais de uma restrição atingir a igualdade ao mesmo tempo, com o mesmo excesso, basta calcular esse excesso para uma delas

somente. Mas como saber quais das restrições estão para atingir a igualdade ao mesmo tempo?

Para termos ganho computacional, essa pesquisa tem que ter custo menor do que simplesmente ignorar esse fato e rodar o Simplex em todas as restrições possíveis.

Esse é o grande problema. Se o custo computacional de se procurar restrições superar o de considerar todas as restrições, o balanço que teremos vai ser negativo.

Quantas iterações do Simplex são necessárias para obtemos o *nucleolus*? Será que depende de cada instância do problema, cada arranjo possível de jogadores, cada arquitetura possível da rede? Ou seria esta uma questão intrínseca do método de obtenção do *nucleolus* com o uso do jogo de árvore geradora mínima?

Pelo algoritmo exato, em que todas as restrições têm que ser levadas em conta, teríamos, no pior caso, (2^n-1) iterações, uma para cada maximização de excesso para as restrições.

Se a questão for realmente levar em conta todas as restrições em cada iteração, sem se preocupar com quais seriam irrelevantes, poderíamos atingir o *nucleolus* em n iterações? (de agora em diante será dito somente n, e não mais em $n-1$, entenda n como já excluindo o fornecedor do serviço, e, ao maximizar lexicograficamente até a penúltima coalizão mais insatisfeita, a última já fica automaticamente ordenada em ordem lexicográfica). Para problemas com muitas restrições, pode ser impraticável a verificação por métodos convencionais para redes com mais de 20 ou 25 pontos, mas se provarmos por indução que há uma relação previsível entre o número de iterações e o número de restrições, então isto pode ser verificado.

Parece óbvio que um ganho em se diminuir o número de iterações só pode ser conseguido com a diminuição do número de restrições na Programação Linear, realmente descartando restrições irrelevantes.

Assim, é necessário apenas n iterações para se obter o *nucleolus*, cada iteração maximizando o excesso de uma das n restrições relevantes ao problema.

Como há mais de n restrições, e ao final TODAS terão seu excesso maximizado, logicamente então mais de uma restrição, em média, vai atingir a igualdade em cada iteração, se processarmos n iterações.

3.3.4 Possibilidade 4

Do ponto de vista geométrico, o problema pode ser visto da seguinte forma.

Como o *nucleolus* seria aquele ponto contido o mais próximo possível do centro absoluto do poliedro hiperdimensional, que define o núcleo, este assumiu essa forma condensada após sucessivas modificações, com base em iterações do Simplex.

Se a cada iteração uma ou mais restrições estão para atingir a igualdade com um dado excesso, igual para todas, então geometricamente uma parte ou mais do vetor solução estão igualmente próximas dos seus respectivos limites, ou faces (ou vértices) do poliedro.

Quando fixamos as restrições e atendemos sua insatisfação quanto a ter um menor excesso naquela iteração, é como se afastássemos aqueles ponto que define o vetor de alocação dos limites do poliedro, em direção ao centro.

Na iteração seguinte, afastaremos o vetor solução das faces do poliedro que estiverem mais próximas, e assim por diante até atingirmos valores próximos do *nucleolus*, ou seja, modificar a posição do vetor de alocações (igual a um ponto, pois define os valores que cada coalizão vai arcar. Um vetor de alocações) até que este esteja no centro ou próximo ao centro do poliedro.

Mas, de novo, como saber quais restrições atingirão a igualdade na atual iteração?

Com base na geometria, haveria, duas formas. Uma, achar os valores do vetor de alocação que estão a igual distância dos limites do poliedro, ou seja, quais coalizões estão com o mesmo excesso a ser maximizado. Outra, caso as restrições tenham seus excessos maximizados e alterem as coordenadas do vetor de alocação de modo que este vá em direção ao o centro do poliedro, precisaríamos saber essa distância ao centro, **comum a todas as restrições que atingirão a igualdade naquela iteração**.

Não é fácil saber como isso pode ser feito. Toda a nossa geometria espacial está formulada para um espaço dimensional de no máximo três dimensões. Tentar achar valores, como pontos e distâncias em uma hiperdimensão

implicaria numa revisão, adaptação e criação de regras e conceitos novos para diferentes hiperdimensões.

3.4 Então, porquê *n* restrições afinal? (conclusão)

Após várias possibilidades para a questão, a conclusão é a seguinte: cada jogador vai ter que arcar com um custo máximo definido por uma ou mais restrições que representam coalizões em que ele faz parte. Estas restrições, na forma de inequações, definem uma região no espaço de prováveis soluções para os jogadores nela representados.

Toda solução em que cada jogador pague no máximo o que pagaria se ligasse de qualquer outra forma, ou mesmo sozinho, ao fornecedor, juntamente com o fato de que a soma do que todos irão pagar deve ser igual ao custo total da rede, pertence ao núcleo do problema, isto é, são soluções válidas. O conjunto de jogadores (ou seja, um sub-grafo) chama-se coalizão.

A Teoria dos Jogos Cooperativos foi usada porque tem-se um ganho no custo da rede que liga uma coalizão ao fornecedor, se os jogadores cooperarem entre si. Na figura 2.1, se cada jogador se ligasse sozinho ao fornecedor teríamos um custo total da rede que é maior que a rede da figura 2.2. Assim, este ganho com a economia de ligações pode ser repartido entre os jogadores.

Quando maximizamos o excesso de uma coalizão, ou seja, diminuímos o quanto esta coalizão irá pagar se participar da solução geral, diminuímos a região de prováveis soluções para esta coalizão. Cada coalizão separadamente pode ser encarada como uma instância do problema total, como um fractal. Um fractal é uma figura em que uma parte dela própria, seja de que tamanho for, é igual à figura original.

A junção de todas as soluções menores é a solução total, mas leva-se em conta a interação entre elas, por ser um jogo cooperativo.

Para uma coalizão de um jogador, temos, por exemplo, que ele deve pagar no máximo 100, que é o preço que ele

pagaria se ligasse sozinho ao fornecedor. Se um algoritmo de Árvore Geradora Mínima for rodado para este caso, obteria:

$$X1 \leq 100$$

Onde x1 representa este jogador. Ao se ligar a outros, espera-se que este custo para ele diminua, por ser um jogo cooperativo.

Apenas uma dimensão é suficiente para representar esta inequação. Um eixo, *x*, em seu espaço unidimensional, teria um ponto 100, e tudo abaixo deste valor, incluindo ele próprio, seria um espaço de prováveis soluções.

Quando um método de programação linear iterativo como o Simplex é rodado com todas as restrições, com o objetivo de maximizar o excesso, o que ele faz é afastar o máximo possível de 100 o valor que x1 deve pagar, levando em conta todas as outras restrições.

Ao maximizar o excesso de x1, transformamos uma inequação em equação. Por exemplo:

$$X1 = 95$$

O que era uma região anteriormente agora tornou-se um ponto. Assim, o eixo *x* representa, na solução geral, o que x1 deve pagar.

Para representarmos o que os *n* jogadores devem pagar, precisamos de um espaço *n*-dimensional. onde cada eixo determina os valores possíveis de serem pagos por cada jogador.

As restrições que envolvam determinados jogadores, delimitarão uma região possível de soluções, representada por um poliedro construído naqueles eixos, por aquelas restrições. Restrições são as retas que compõe as faces, suas arestas, e o encontro delas, ou seja, seus vértices, são pontos que representam uma solução possível. Essas soluções são os vetores de alocação, pois, dado um ponto no espaço *n*-dimensional, este fica definido com *n* coordenadas.

Por exemplo, para um jogador, temos uma região unidimensional, uma reta, de soluções possíveis (Fig. 4.8). Quando maximizamos o excesso para aquela coalizão de um

jogador, reduzimos o que seria uma reta para um ponto (Fig. 4.9).

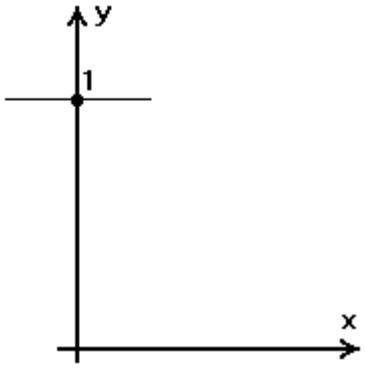

Figura 4.8. Só uma dimensão basta para representar um jogador

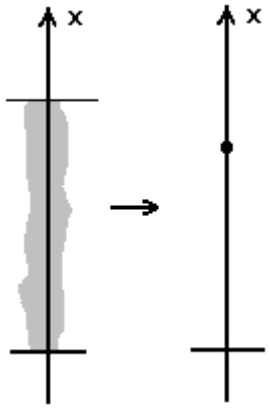

Figura 4.9. Redução de uma dimesão (reta) a um único ponto

Restrições que envolvam dois jogadores, são representadas em forma de retas no espaço bidimensional (Fig. 4.10). Quando maximizamos o excesso para esta coalizão de dois jogadores, transformamos o que seria uma área em uma reta, ou seja, a inequação passou a ser uma equação linear (Fig. 4.11).

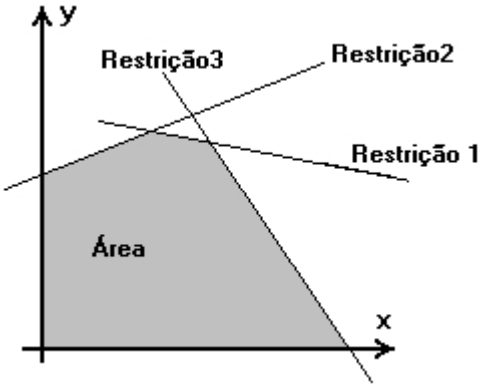

Figura 4.10. Espaço bidimensional com restrições envolvendo dois jogadores

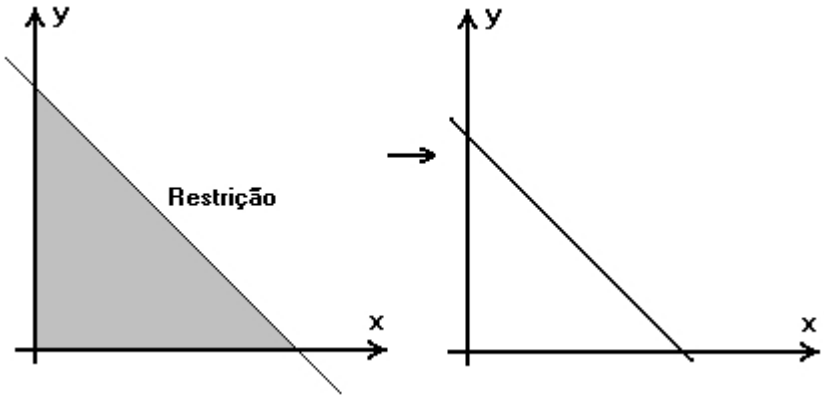

Figura 4.11. Redução de uma área (bidimensional) a uma reta

Estendendo o raciocínio para três ou mais dimensões, o que está acontecendo, na verdade, quando maximizamos o excesso das coalizões, estamos afastando a solução contida no espaço possível de soluções delimitado por aquela restrição do ponto máximo possível, o de excesso zero.

A delimitação espacial do núcleo (o poliedro hiperdimensional) é o limite máximo das soluções, ou seja, qualquer ponto contido em seus limites, ou vértices (e não totalmente contido em seu interior) está no limite de ao

menos uma restrição, logo, ao menos uma coalizão terá excesso zero.

Afastando-se deste limite máximo e fixando uma faixa de valores, diminuindo o número de dimensões, implica que, à medida que vamos maximizando os excessos, vamos diminuindo a região de soluções possíveis (núcleo) delimitada pelo poliedro hiperdimensional. Este vai diminuindo de tal maneira que vai se resumir a uma pequena região de soluções possíveis, ou uma solução possível (retirando suas faces estamos também retirando suas dimensões, o que é possível por serem várias delas compostas por restrições irrelevantes, como que lapidando-o, diminuindo-o). Este poliedro restrito ao máximo delimita o *nucleolus*.

Esta diminuição do poliedro, no sentido de se limitar a região delimitada por ele para as possíveis soluções, é limitada pela restrição que define que o custo total da rede deve ser coberto. Não podemos diminuir o custo de que cada coalizão vai arcar de tal maneira que esta não pague nada. A maximização do excesso pelo método iterativo de programação linear (Simplex) leva isto em conta.

Para este problema em particular, temos que todas as restrições são inequações de primeiro grau, e seus coeficientes são todos unitários. Não temos, por exemplo, um coeficiente na restrição do tipo

$$x_1^2$$

tampouco temos coeficientes de pesos diferentes, como por exemplo:

$$x_1 + 2x_2 \leq 100$$

Assim, as restrições se apresentam na forma de retas, e, como os coeficientes têm o mesmo peso, como por exemplo

$$x_1 + x_3 + x_{41} + x_{105} \leq 100$$

podemos supor que, no caso de duas dimensões temos uma reta que corta o plano, e, como as duas componentes

têm o mesmo peso, ela deve ter inclinação de 45 graus (Fig. 4.4).

Desta forma, todas as projeções das restrições naquele eixo podem ter esta mesma inclinação (linearmente dependente, pois é paralela). Porém não é tão simples.

Devido ao fato de que cada jogador se relaciona a cada um dos demais, e também pelo fato de termos a restrição de que o custo total da rede deve ser coberto, o poliedro hiperdimensional não deve ser necessariamente uma figura regular.

Recapitulando, o poliedro hiperdimensional pode ser visualizado da seguinte forma. Suas faces são construídas tendo como arestas os limites das restrições, e os vértices são os encontros destas restrições. Apesar de não ser possível desenhar mais de três dimensões, o leitor não pode ser deixado sem ao menos uma concepção parecida com a obtida com o trabalho, que é como foi imaginado (Fig. 4.12).

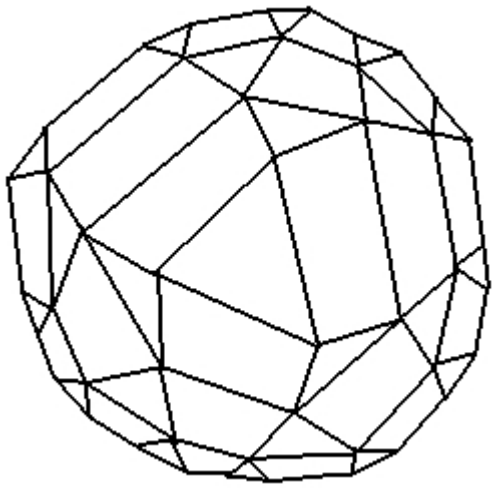

Figura 4.12. O poliedro hiperdimensional , delimitando o núcleo (concepção pessoal)

Como o poliedro em si já é uma região delimitante no espaço *n*-dimensional para as soluções possíveis, ou seja, o

núcleo, podemos esperar soluções melhores aqueles pontos afastados ao máximo de quaisquer faces, em direção ao seu centro.

Um vértice é um ponto no espaço *n*-dimensional, ou seja, para defini-lo, teremos um valor para cada componente deste espaço, ou um custo para cada jogador. Mas não é uma boa solução, afinal, ao menos para uma coalizão, esta terá que pagar o mesmo que pagaria ao se ligar sozinha ao provedor do serviço. O mesmo ocorreria se fosse um ponto qualquer de uma face, pois está no limite máximo do poliedro. Pontos que delimitam o poliedro são pontos onde ao menos uma restrição atingiu a igualdade. Mas a programação busca respostas otimizadas nos vértices dos poliedros, ou numa face, caso seja possível múltiplas respostas.

Por exemplo, para *x1* = 50, em uma dimensão, 50 é o ponto limite, de excesso 0.

Quando maximizamos o excesso para uma coalizão, estamos afastando-a ao máximo de uma face do poliedro, em direção ao centro.

Outro fruto deste trabalho foi a dedução (com base na interpretação geométrica, maximizacão de excessos, diminuição dimensional ao transformarmos inequações em equações, e todos os outros conceitos), de porque precisamos de somente n restrições para atingir o *nucleolus.*

O que um processo iterativo faz, com base nas restrições, é maximizar o excesso das coalizões, mas até um ponto em que o equilíbrio entre o que seja justo para todos os jogadores não seja quebrado, e que a condição de que o custo total da rede seja coberto.

Ao atingir esse equilíbrio, atingimos o *nucleolus*, que é a solução ótima. Ao maximizar o excesso das coalizões mais insatisfeitas, limitamos a área que as representam a um ponto, e as transformamos em equações (Fig. 4.13).

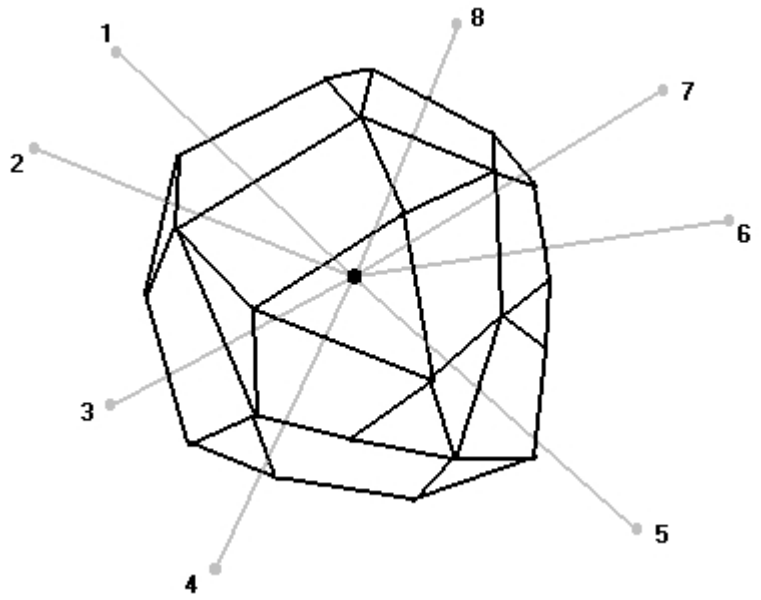

Figura 4.13. O *nucleolus*. Os números representam os pontos que tocam em cada um dos *n* eixos que representam os *n* jogadores. Determinam os valores que cada um deles irá pagar

Ao final do processo, não precisaríamos ter transformado todas as restrições em equações.

Porquê?

Porque precisamos saber o quanto cada um vai pagar. Se cada jogador é uma variável, e temos *n* jogadores, há *n* variáveis, ou incógnitas. Se transformarmos todas as restrições em equações, teremos $2^n - 1$ equações. Ou seja, mais equações do que incógnitas.

Só precisamos de *n* equações linearmente independentes para determinarmos *n* incógnitas, ou, em outras palavras, para termos um **Sistema Possível Determinado**. Isso se soubéssemos previamente o excesso de cada uma das restrições.

Assim, há $2^n - 1 - (n-1) = (2^n - n)$ restrições desnecessárias à obtenção do *nucleolus*.

3.4.1 – Interpretação geométrica do *nucleolus*

Se determinamos o *nucleolus* com *n* equações, então este deve ser mesmo um *ponto* (vértices também são pontos, mas de soluções pouco otimizadas, já que estão nos limites do poliedro) no espaço n-dimensional, pois, dado que seja um ponto, este define só um valor em cada uma das *n* dimensões necessárias para determiná-lo. Geometricamente, é bem intuitivo pensar no *nucleolus* como somente um ponto (Fig. 4.13).

O motivo para que a maioria das equações sejam irrelevantes é que elas não estão definindo coordenadas do *nucleolus* em eixos que já não tenham sido definidas, ou que definam pontos cujo excesso seja menor para estes jogadores (componham regiões menos restritas, ou que não acrescentem nenhuma informação que já não tenha sido definida na composição de uma face).

Uma outra questão pertinente, seria a seguinte. Se alterarmos em algum ponto, a ordem em que estamos maximizando lexicograficamente o excesso das coalizões, poderíamos obter outro ponto, outro *nucleolus*?

Para ser o maior vetor em ordem lexicográfica, podemos ter, por exemplo, somente a coalizão menos satisfeita com seu excesso maximizado. Ele já seria o maior, lexicograficamente. O nome ‘Ricardo’, por exemplo, vem primeiro, em ordem lexicográfica, que o ‘Samira’, mesmo que a segunda letra desta palavra venha antes no alfabeto do que a segunda letra daquele nome.

Se maximizarmos o excesso, de coalizão a coalizão, da mais insatisfeita à menos insatisfeita, teremos um vetor que é o melhor dentre todos, por ser o mais justo. Acontece que, se mais de uma coalizão possui o mesmo excesso a ser maximizado, pode fazer diferença qual delas maximizar primeiro, pois são coalizões diferentes, em número e/ou coeficientes (jogadores). Mas também pode não fazer diferença, ou seja, esperar para maximizar o excesso tanto de uma quanto de outra coalizão que atinja o excesso ao mesmo tempo, não garantindo um ganho maior para ela mais adiante. Isto vem do fato de que todos os jogadores estão interligados, e cada caso é um caso, temos infinitas possibilidades, e por ser este um problema modelado de

acordo com a Teoria dos Jogos Cooperativos. Esta bifurcação possível na ordem de maximização pode gerar mais de um vetor de alocação que seja justo.

A questão, interpretada aqui, não é se podemos ter um ou mais *nucleolus*, mas se este é um ponto ou uma região hiperdimensional, muito mais limitada que o núcleo, contida no interior deste, em que seus vértices sejam soluções, todas justas (Fig. 4.14).

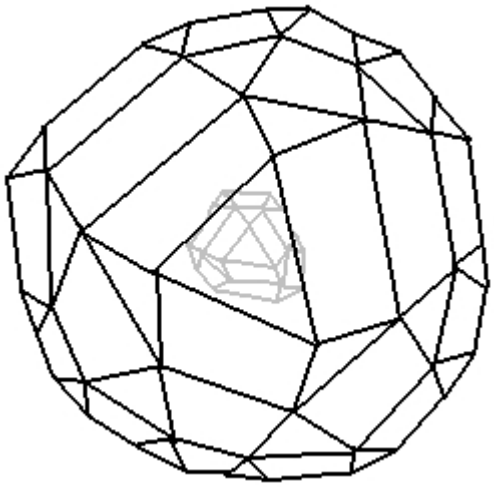

Figura 4.14. O nucleolus seria único?

Cada rede é um caso, e seria difícil supor que não exista mais de uma solução ótima. Para que o n*ucleolus* seja sempre um ponto único, teríamos que ter, sempre, dentre os possíveis vetores de alocação justos, somente um, ordenado lexicograficamente, que tenha os excessos maximizados em cada coalizão, da mais insatisfeita à menos insatisfeita. Isso originando o vetor de custos mais justo possível.

3.5 Escolhendo as n restrições que interessam

Dizer que só *n* restrições interessam não quer dizer que quaisquer *n* restrições, se resolvidas, definem o

nucleolus. Assim, qualquer conjunto de *n* restrições que formem um Sistema Possível Determinado determinariam o *nucleolus*.

Há uma questão fundamental a ser claramente entendida. Primeiro, deve-se montar o poliedro *n*-dimensional apenas com as *n* restrições que interessam, ou que sejam em número muito menor às $2^n - 1$, mas que construam um poliedro otimizado, restrito ao máximo (que é o principal propósito de todo o trabalho), e após montado, a programação linear acha o *nucleolus*.

Com todas as restrições geradas, poderíamos inventar um método para saber quais definiriam pontos em que os excessos fossem menores aos alcançados por outras restrições que também dizem respeito a estas mesmas dimensões (só que com algumas a mais). Mas seria impraticável, pois teríamos que gerar todas as restrições. Qual o problema?

Novamente expondo a dificuldade em se trabalhar com espaço e tempo exponenciais. Para uma rede de digamos, 3 jogadores, temos 7 restrições. Talvez nem precisássemos fazer contas, só de olhar no grafo poderíamos supor sua Árvore Geradora Mínima, e na mão faríamos as contas.

Agora, se estamos falando de uma rede de acesso realista, seria normal termos, por exemplo, 150 jogadores. O número de restrições que temos que gerar, sem esquecer que para cada coalizão temos que calcular seu custo máximo através de um algoritmo de Árvore Geradora Mínima, é de aproximadamente...

1.427.247.693.000.000.000.000.000.000.000.000.000.000.000.000

... restrições....

Para atingir o *nucleolus*, precisamos de 150 restrições destas. Se conseguíssemos escolher as certas, diminuiríamos uma complexidade que era exponencial para linear. A chance de gerar só as restrições que interessam e tratá-las com programação linear para obtenção do *nucleolus* diminui também exponencialmente com o tamanho do problema. Isto porque, quanto mais jogadores, mas difícil determinar só as

restrições relevantes, pois estas crescem linearmente com o problema, e as restrições totais crescem exponencialmente.

Existem heurísticas que usam recursos maciços, como a Têmpera Simulada juntamente com Branch & Bound. Ainda sim não são muito eficientes. Segundo estudos, o problema de se gerar restrições é NP-Difícil.

O que é proposto no trabalho é uma heurística usando-se Algoritmos Genéticos para ajudar na obtenção de uma solução que se aproxime ao máximo do *nucleolus* para redes muito grandes.

Capítulo 4 - Usando Algoritmos Genéticos associados ao Simples para se computar soluções próximas do nucleolus

4.1 Uma visão geral

Se o *nucleolus* for mesmo único, então só utilizando-se as *n* restrições certas para chegar a ele. Isto é impraticável, dada a dificuldade de se encontrar as restrições certas dentre todas as possíveis em redes muito grandes.

O problema não é trivial. Como reduzir o poliedro hiperdimensional que representa o núcleo da maneira mais otimizada possível, lapidá-lo, usando-se algoritmos genéticos?

Os principais pontos (no caso, soluções) para este problema ao se levar em conta para que algoritmos genéticos sejam usados para achar uma solução são os seguintes:

- Determinar quem serão os indivíduos, e como representarão a solução do problema;
- Como e quantos indivíduos serão criados para formarem a primeira geração;
- Como avaliar o grau de adaptabilidade dos indivíduos;
- Como combinar os indivíduos entre si para gerarem filhos de forma sexuada ou quando preferir a reprodução assexuada;
- Com que frequência e como as mutações podem se apresentar;
- Pontos preferenciais para crossover;

- Critério de parada nas gerações e parâmetros a serem modificados para testar o algoritmo;

- Possibilidade de testar-se todo o universo de possibilidades, e fugir de convergências a máximos locais;

- Dar ao algoritmo genético a chance de encontrar uma solução bem próxima da solução ótima, se esta existir, desconsiderando a questão de tempo e recursos computacionais por ajustes nas variáveis do programa.

4.2 Uma proposta de heurística

Foi considerada uma heurística que se utiliza dos princípios dos AG's, e não um Algoritmo Genético em sua forma tradicional, o que pode limitar seu uso e eficiência.

Primeiramente, cada indivíduo não é uma resposta ao problema em si, mas parte dela. Segundo, não haverá muita diversificação de indivíduos em uma mesma geração, já que a idéia proposta no trabalho é que sejam rodados vários AG's, cada um convergindo para parte da resposta.

Para simplificar o entendimento, considere que, o que for definido para um algoritmo genético, será seguido em todos os que forem executados (em paralelo ou em sequência).

Detalhando melhor:

4.2.1 Os indivíduos

Cada indivíduo representa uma restrição. Assim, a solução para o problema será a um sistema de restrições montado pelos diversos AG's, e não cada indivíduo em si.

4.2.2 A geração 1

Inicialmente podem ser gerados, por exemplo, *n* indivíduos, onde *n* é o número de jogadores.

Estes indivíduos são gerados de forma aleatória, tanto nos coeficientes, quanto no número de coeficientes. O único cuidado é não gerar indivíduos cujas restrições contenham um jogador mais de uma vez, e não gerar indivíduos iguais.

Após geradas as coalizões (combinações dos jogadores), roda-se um algoritmo de AGM com base nos dados reais de localização de usuários e fornecedor, e criam-se as restrições (inequações com um valor máximo a ser pago pela coalizão), ou melhor, os indivíduos.

4.2.3 Critério de aptidão ou adaptabilidade

Se trabalharmos com todas as restrições possíveis, as relevantes são aquelas que, juntas, determinam um poliedro hiperdimensional tão restrito quanto possível, definindo o núcleo. Assim, garante-se que o núcleo seja construído com apenas as *n* restrições relevantes, para a partir dele, através de programação linear, obtermos o *nucleolus*.

Se as restrições são relevantes, são mais restritas que outras, definido uma região ótima (menor) para aqueles jogadores, ou seus coeficientes. Estas restrições já têm um excesso maximizado implícito, pois definem custos menores aos jogadores por elas representados. Usando-se essa idéia é que o algoritmo genético foi proposto. Além disso, foi verificado, em um algoritmo exato, que o Simplex acha que o menor excesso é justamente daquelas *n* restrições relevantes (de *n* coalizões), logo na primeira iteração, para algumas redes de acesso verificas.

Para verificar a adaptabilidade, roda-se o Simplex em toda a geração e separamos a(s) restrição(ões) que atingiu(ram) a igualdade na primeira iteração. Isto significa que ela(s) tem(êm) o menor excesso, sendo portanto mais restritivas na solução geral.

Assim, somente estes indivíduos terão nota máxima de aptidão e serão mantidos na próxima geração. Os outros indivíduos serão avaliados da seguinte forma. Todos os que tiverem alguma interseção com os indivíduos selecionados, isto é, contiverem um ou mais jogadores em comum com os indivíduos de máxima aptidão, terão igual chance de

transmitirem seus genes, mas não será de forma assexuada. Assim, terão uma aptidão menor.

O processo vai gerar, propositadamente, restrições linearmente dependentes, com combinações diferentes dos mesmos indivíduos. A intenção é selecionar justamente aquela ou aquelas mais restritiva somente (a última geração vai selecionar, dentre todos os indivíduos, somente aquele ou aqueles cujas restrições representadas delimitem a menor região espacial, ou seja, tenham o menor excesso), dentre as muitas linearmente dependentes que vão ser comparadas. Lembrando, o número de restrições relevantes cresce linearmente com o problema, enquanto o número total cresce exponencialmente.

Ao fim de um determinado número de gerações, teremos uma ou algumas poucas restrições, advindas de cada AG's, que sozinhas não são suficientes para montarmos um sistema que vá encontrar uma solução próxima ao *nucleolus*. Porém, como essa heurística utiliza várias vezes um algoritmo genético, em sequência, ou vários em paralelo, selecionaremos ao fim destes vários processamentos, várias restrições que terão boa chance de serem linearmente independentes. Isto porque cada algoritmo genético independentemente vai selecionar não indivíduos com a resposta total, mas para parte dela (cada indivíduo é uma restrição).

Como contamos com a imprevisibilidade, não sabemos ao certo se vai haver apenas um ou mais de um indivíduo com aptidão máxima, ou seja, que vão atingir a igualdade pela avaliação do Simplex, naquela geração. Se estes indivíduos representarem as restrições relevantes, dificilmente seriam escolhidos primeiramente, com este conjunto pequeno de indivíduos e se consideramos uma rede grande. Ou melhor, para um número muito grande de restrições, é difícil que, contando com a aleatoriedade na escolha dos indivíduos na primeira geração, estes sejam justamente aquelas ou algumas das *n* restrições relevantes para a obtenção do *nucleolus*.

Mas como é uma heurística que usa os algoritmos genéticos, estamos buscando uma boa solução usando-se recursos e tempo toleráveis. A intenção é montar um sistema de restrições que definam um núcleo tão restritivo quanto possível.

4.2.4 Reprodução dos indivíduos

Os indivíduos com aptidão máxima se reproduzirão de forma assexuada, então serão simplesmente repetidos na próxima geração. Os outros de menor aptidão serão recombinados entre si para formar novos indivíduos. Quando uma inequação é partida, recombinada ou alterada, pode-se fazer a decomposição desta em novas inequações. Matematicamente esta divisão respeitaria os atuais pesos de cada coeficiente na região *n*-dimensional restritiva, ou deveria ser rodado novamente o algoritmo de AGM nos coeficientes desejados, obtendo um novo valor de custo total para aquela coalizão menor, com base na Teoria dos Jogos.

A diferença entre os dois é que, com mais genes de jogadores em um indivíduo, é entendido que o custo total desta coalizão seja menor do que a soma de custos das coalizões menores que formem juntas esta coalizão. Porque é baseada na Teoria dos Jogos Cooperativos. Decompor as inequações e calcular os novos valores de custo para as partes criadas fazendo-se projeções (ignorando as componentes nos eixos dos outros coeficientes e recalculando a nova região que o poliedro hiperdimensional vai delimitar) dividimos o custo menor obtido com o jogo cooperativo entre as novas coalizões. Se rodarmos um algoritmo de AGM, este custo para as mesmas coalizões criadas na divisão da coalizão original vai possivelmente aumentar.

Como o jogo é baseado em cooperação o aconselhável é uma nova execução do algoritmo de AGM nas novas restrições. Mas pode ser interessante testar o algoritmo só com a decomposição das inequações.

Voltando, combinamos de forma aleatória (crossover em qualquer ponto, juntar dois indivíduos ou mais, por exemplo) todos os indivíduos selecionados, mas com menor aptidão, gerando novas restrições, os seus filhos.

Todos os indivíduos (tanto os pais, quanto os filhos) podem e devem sofrer mutações, numa taxa pequena, em que algum segmento do material genético (um ou mais coeficientes) podem ser retirados ou acrescentados.

Na próxima geração, teremos indivíduos com seus respetivos materiais genéticos contendo muitos genes em comum, salvo o caso de mutações e outros genes sem interseção com os genes dos indivíduos que obtiveram máxima nota de aptidão.

Uma característica importante dos algoritmos genéticos, a diversidade da população, não seria observada dentro de uma geração, deste modo, mas haveria diversidade entre indivíduos de processos rodando o algoritmo genético em paralelo, ou em sequência. Isto é proposital, visto que queremos obter a restrição mais otimizada dentre aquelas linearmente dependentes (ao menos, com projeções linearmente dependentes) com ela.

4.2.5 Quantas gerações serão computadas em cada AG

Vale ressaltar, novamente, que a proposta **usa** os algoritmos genéticos, e não é um algoritmo genético em sua forma tradicional.

A proposta é executar uma sequência de algoritmos genéticos, ou em paralelo, cada um otimizando uma parte da solução, como um Branch & Bound. Como temos que tentar eliminar restrições linearmente dependentes, cada um dos processos rodando um algoritmo genético tenta selecionar algumas poucas restrições, ou somente uma ao final, esta linearmente independente, ao menos representando uma coalizão bem diferente, àquela obtida pelos outros algoritmos genéticos.

A proposta seria para os casos de redes grandes, 100, 150, 200 pontos ou mais. A aleatoriedade na escolha da geração 1 garantiria em teoria a convergência a diferentes pedaços da solução. É uma heurística nova, fruto deste trabalho, numa área em que ainda não temos soluções exatas para redes grandes.

Após um número determinado de gerações em cada AG, o algoritmo rodaria o Simplex mais uma vez e separaria apenas o(s) indivíduo(s) de menor excesso.

Apenas estes poucos indivíduos seriam então armazenados. Seria feita uma verificação. Estes indivíduos obtidos pelos AG's são em número no mínimo igual ao número de jogadores, e contêm todos os jogadores, compondo um sistema possível determinado?

Em caso positivo, para-se aí e serão estes indivíduos, e mais alguns (combinações entre estes, para usar a vantagem da cooperação e assim formar mais alguns indivíduos, podendo-se usar também mutações) os selecionados para compor o núcleo. Estes formariam o sistema de restrições usadas no Simplex, que é rodado desta vez até o final, não mais usado como ferramenta para o critério de aptidão dos indivíduos, mas como ferramenta para solucionar este problema de programação linear, obtendo uma solução próxima ao *nucleolus*.

Em caso negativo, obtém-se mais alguns indivíduos com o uso dos Algoritmos Genéticos.

4.2.6 Quando parar o algoritmo

Toda vez que cada algoritmo genético roda, ao fim de um número de gerações, produz alguma restrição ou restrições com um excesso bastante pequeno.

Rodamos então o algoritmo até que os indivíduos, armazenados sucessivamente, contenham todos os jogadores (cada jogador é um coeficiente na restrição) e tenhamos restrições que sejam no mínimo em número igual ao número de jogadores.

Caso houver demora demais para juntar indivíduos selecionados suficientes para se formar um sistema possível determinado, ou seja, algum jogador ainda não apareceu (não há gene de um certo jogador, x_2, por exemplo, presente em quaisquer dos indivíduos) podemos limitar o número de vezes em que cada AG é rodado, ou seja, o número de gerações de cada um deles (cada vez que é rodado, nos fornece um indivíduo ou indivíduos bem aptos ao fim de determinado número de gerações).

4.2.7 Como serão usados os indivíduos

Como os indivíduos não representam a solução do problema em si, mas partes dele, estes serão usados em conjunto, reunidos dos vários algoritmos genéticos separados, para comporem um sistema que será usado para obtermos um vetor de alocação de custos próximo ao *nucleolus*.

A total aleatoriedade na escolha dos indivíduos na primeira geração cada vez que rodamos o algoritmo genético nos garante uma boa heurística que varre todo o espaço possível. Mesmo assim, podemos modificá-lo, por exemplo, se for tendencioso na seleção dos indivíduos, ao final das gerações.

Indivíduos selecionados que contenham de forma geral os mesmos jogadores em seus genes, podem indicar um máximo local. Assim, estes jogadores podem e devem ter menos chance de serem sorteados na próxima elaboração da primeira geração de indivíduos de outro Algoritmo Genético.

Devem ser feitas também combinações entre indivíduos selecionados, e incluir seus filhos no conjunto definitivo de restrições que será rodado no Simplex. Isto é importante para testarmos os ganhos em fazer interseções ente coalizões. Esta é uma forma de se levar em conta de que é um algoritmo que usa Teoria dos Jogos Cooperativos, aplicada numa idéia de Dividir e Conquistar (Branch & Bound). Isto pois, além de dividir o problema em pedaços, ao fim unimos os mesmos em novas partes, recombinando as restrições para testar novas coalizões.

Achar soluções ótimas separadamente e uni-las para uma solução geral não deve ser ineficiente, mesmo se consideramos que um jogador coopera com outro. Dividindo o problema, dividimos a rede que é uma árvore em outras árvores, estrutura esta que é recursiva por natureza, como um fractal.

Essas recombinações entre os indivíduos já selecionados podem ser de união, interseção ou crossover, junto com eventuais mutações de genes.

4.2.8 Uma representação da proposta do algoritmo

Para resumir a idéia da heurística, abstraindo o processo, segue abaixo sua representação:

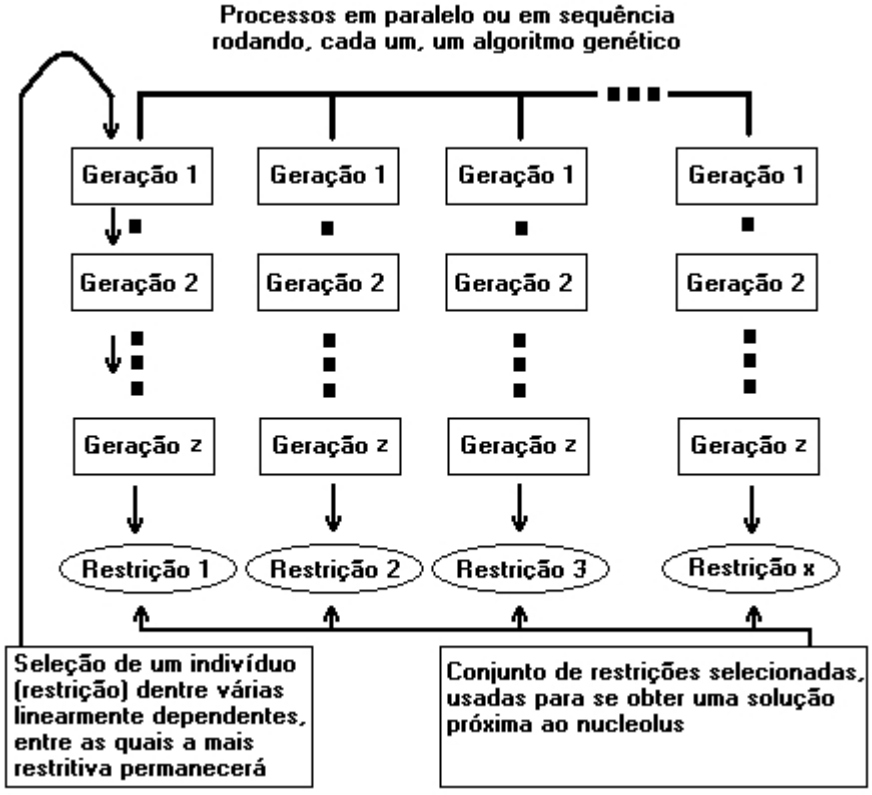

Figura 5.1 A heurística proposta

4.2.9 Parâmetros a serem testados e modificados

Podem ser modificados, para testar a eficiência do algoritmo, o tamanho da população inicial, o número de

combinações entre os pais da primeira geração, tipos e a frequência com que aparecem as mutações, número de gerações, número de AG's, dentre outros.

Foi levando em conta a possibilidade de se obter as *n* restrições relevantes com bastante confiabilidade. Se a geração 1 de um dos AG's tiver $2^N - 1$ indivíduos, então teremos representadas todas as restrições. Todas as relevantes atingirão a igualdade (por serem mais restritivas), e serão selecionadas até o fim. Assim, após a primeira geração, os indivíduos já poderão ser usados para acharmos o *nucleolus*. Se o Simplex for rodado como ferramenta de avaliação de adaptabilidade em todos os indivíduos selecionados por todos os AG's, estas serão as restrições com menor excesso, ou seja, que representam coalizões mais insatisfeitas.

Obviamente, o problema seria o mesmo: gerar todas as restrições. Seria melhor então usar um algoritmo exato de uma vez.

.

Capítulo 5 – Conclusão e Propostas

Conclui-se, com o trabalho, que o problema de alocação de custos em redes de acesso é de solução não-trivial, com implicações impossíveis de terem sido previstas ao defrontar com o mesmo inicialmente. O problema revelou-se incrivelmente interessante e desafiador. É necessário o conhecimento de muitas áreas diferentes e específicas de conhecimento, suas interligações possíveis, além de dedicação e criatividade para visualizar-se o problema por vários ângulos diferentes.

Foi conseguida uma interpretação geométrica do problema, de como os jogadores, restrições, núcleo e *nucleolus* podem ser visualizados, algo não encontrado na literatura. Foi um fruto importantíssimo, por ser concebido em um espaço dimensional não convencional e de visualização complexa. Isto foi imprescindível na dedução própria de porquê não precisamos de todo o espaço de restrições para construirmos o poliedro delimitador do núcleo.

A partir de uma revisão bibliográfica de algoritmos de otimização, Simplex, Algoritmos Genéticos, Teoria dos Jogos e da própria interpretação geométrica, foi concebida uma heurística para ajudar a calcular uma solução próxima ao *nucleolus*. A idéia inicial era se estudar uma heurística existente que usa Têmpera Simulada, e criar uma que fizesse o mesmo, só que com Algoritmos Genéticos, mas esta não foi publicada no artigo estudado. O que foi feito foi o estudo do algoritmo exato, mas não teria sido muito difícil deduzir algo semelhante. Mas este não é o problema, o principal é se pensar em uma maneira de não aplicar este algoritmo exato em todo o espaço de restrições, já que sua complexidade é exponencial.

Foi conseguido também uma explicação própria da viabilidade da redução de espaço e tempo de exponencial para linear, sustentando a idéia da heurística obtida. A heurística não é um Algoritmo Genético em sua forma tradicional, mas se utiliza dos AG's para se obter, cada um, parte da solução (Branch & Bound), construindo um poliedro delimitador do núcleo que seja o mais restrito possível e formado sem a maioria das redundâncias.

Uma proposta para trabalhos futuros seria o de explorar variações do problema, como por exemplo topologias diferentes, ou de se levar em conta mais de um fornecedor, ou mesmo explorar o mesmo assunto, ainda com pontos a serem explicados. Além disso, devem ser analisadas vantagens e desvantagens da proposta apresentada, para se chegar ao algoritmo efetivamente implementado.

Capítulo 6 – Bibliografia

FERNEDA, E. *Busca Heurística* Avaliable from http://www.dsc.ufpb.br/~edilson/IA_Pagina/material-IA/IA-05.ppt. 2002

GEFFARD, J., LUTTON, J., PREMAOR, F. An algorithm to compute the Nucleolus of the minimum cost Spanning Tree Games. p602-611. 1997

GOLDBARG, M.C., LUNA, P.L. Otimização Combinatória e Programação Linear. In: p81-89, 129-131. 2000.

JESUS, J.C.S. Notas de aula de pesquisa operacional: UFLA, p1-17. 2001

MOKHTAR, B. Nonlinear Programming, p64-68.1979

MOREIRA, R.C. *Estudo dos Problemas de Alocação de Custos em Redes de Acesso*, 2001, Tese (Mestrado) – Universidade Federal de Minas Gerais

REZENDE, C.A., MAIA, C.P., ALEXANDRA, ROBERTO, M., WEBER, A., RAVAZZI, L.B., JÚNIOR, C.S. *Algoritmos Genéticos* Avaliable from http://www.din.uem.br/ia/geneticos/ , 2002

ROISENBERG, M. *Representação de Conhecimento, Métodos de Busca* Avaliable from http://www.inf.ufsc.br/~mauro/ine6102/slide/aula3-buscaheur/tsld019.htm , 1999

TAVARES, A.S. *Balanceamento de Carga em Algoritmos Branch-and-Bound* Avaliable from http://www.lch.dcc.ufmg.br/node4.html , 2002