Empacotando Caixas em *gblocos*

Lauro Didier Lins

16 de Junho de 2003

Dedicatória

A minha família pelo apoio e constante incentivo.

Agradecimentos

Tomo aqui emprestada a frase de Sir Isaac Newton "*If I have seen further than others, it is by standing upon the shoulders of giants*" para lembrar que as nossas conquistas não são pontuais, nem somos donos exclusivos delas; são resultado de um processo em que, certamente, muitos tiveram influência. Alguns passos neste processo são mais brilhantes que outros, como os que o próprio Newton deu, porém todos são importantes. Ao contrário de Newton, eu não vi mais longe do que os outros, mas, assim como ele, reconheço que o que vejo e o que fiz é devido em grande parte ao esforço de outras pessoas.

Correndo o risco de não citar (porém nunca de esquecer) alguém, eis algumas pessoas importantes para mim e para este trabalho:

Meu pai, Sóstenes, de quem eu herdei o interesse e um pouco da habilidade em computação e matemática.

Minha mãe, Bernardete, e minha irmã, Isis, que me dão todo o suporte para que eu siga meu caminho.

Minha família e meus amigos: Nadja, Lourdes, Luiz, Myriam, Lauro, Pedrinho, Fernandinha, João, Joana, Adriana, Adelaide, Heloiza, Niedja, Ivanzinho, Joabe.

Minha orientadora e professora, Kátia Guimarães.

Quero agradecer ao CNPq pelo suporte que me deu para a realização deste trabalho.

Resumo

Um dos problemas abertos mais básicos da área de corte e empacotamento é encontrar o maior número de (ℓ, w)-retângulos que podem ser empacotados ortogonalmente num retângulo maior (L, W). O termo ortogonalmente quer dizer, apenas, que cada lado de um (ℓ, w)-retângulo empacotado é paralelo ou perpendicular aos lados do retângulo maior (L, W). Motivados por este problema e suas variantes mais difíceis (ex. caso tridimensional), desenvolvemos, baseado no trabalho [2], uma abordagem heurística geral de decomposições de gblocos. Os gblocos são uma generalização dos blocos. Os blocos são simplesmente retângulos em dimensão 2 e paralelepípedos em dimensão 3 (e seus análogos em dimensões maiores).

Aplicando a abordagem de gblocos para o problema bidimensional aberto que mencionamos, mostramos se tratar, em termos de otimalidade, de um método superior à melhor heurística existente até o momento: a heurística de R. Morabito e S. Morales (1998). De fato ainda não é conhecido nenhum problema (ℓ, w, L, W) para o qual a nossa abordagem em gblocos não seja ótima. Esta observação empírica levanta a dúvida de estarmos diante de um método exato para o problema. Além do caso bidimensional, sugerimos também uma abordagem em gblocos para o caso tridimensional.

Melhores métodos de empacotamento têm importante implicação econômica. Hoje, caminhões, trens, navios e aviões transportam contêineres e paletes com uma carga menor do que poderiam. Esta Tese é um passo na busca de melhores métodos. Ela apresenta alguns resultados originais, formaliza uma linguagem adequada para o problema abstrato e, por fim, sugere um caminho promissor para o problema concreto no setor de transporte de carga.

Palavras-chave: *otimização combinatória*, *empacotamento 2D e 3D*, *problema de corte e empacotamento*, *problema de carregamento do palete do produtor*.

Conteúdo

1 Introdução **1**

1.1 A idéia chave deste trabalho 1

1.2 Organização da tese 3

2 Teoria dos gblocos e gtets **5**

2.1 Caixas, blocos e gblocos 5

2.2 Gtets e atribuições 7

2.3 Gtet pai e gtet filho 9

2.4 Similaridade entre gtets 9

2.5 Gtet representante 13

2.6 Gtet p-representante 14

2.7 Minors e dgtets degenerados 15

2.8 Esquemas, peças e decomposições 16

2.9 Complexidade de gtets e de esquemas 18

2.10 Realizando empacotamentos: a função o 19

2.11 Como implementar a função o? 21

2.12 A função o' 22

2.13 Limitante superior para a função o 27

3 Empacotando em dimensão 2 **29**

3.1 Problemas de empacotamento em dimensão 2 29

3.2 Breve histórico da evolução das abordagens para o MPL 30

3.3 A heurística R-em-5Rs 32

3.4 A heurística RL 33

3.5 Resultados Computacionais 34

3.6 Algumas questões teóricas 35

4 Empacotando em dimensão 3 **39**

4.1 A heurística B-em-9Bs 39

4.2 A heurística BALST 40

5 Perspectivas Futuras **43**

A Soluções RL para os 16 problemas MPL "difíceis" **45**

B Esquema RL **54**

B.1 Peças 54

B.2 Decomposições 55

C Esquema B-em-9Bs **57**

C.1 Peças 57

C.2 Decomposições 58

D Gtets do esquema BALST **60**

D.1 Peças 61

D.2 Decomposições 63

Lista de Figuras

1.1 Padrões de decomposição da heurística R-em-5Rs. 2
1.2 Solução exata para o problema $\ell = 7$, $w = 3$, $L = 43$ e $W = 26$ (53 caixas). 2
1.3 Solução exata de $MPL(7, 3, 43, 26)$ decomposta em peças L e peças R. . . . 3

2.1 Exemplos de região pconvexa e fconvexa em dimensão 2 6
2.2 Exemplos de blocos e gblocos 6
2.3 Exemplo de gtet 8
2.4 Exemplo de dgtet 9
2.5 GTets diferentes representando o mesmo gbloco K. 10
2.6 Empacotamentos refletidos. 13
2.7 O esquema da heurística RL 18
2.8 Para prova da Proposição 2.13: Fatias de K nas i-cotas crescentes $a - h, a, b, c, c + h$ 24

3.1 Decomposição do esquema $\mathcal{E}_{R5R}$. 32
3.2 Decomposição mais simples de R em R para $MPL(43, 26, 7, 3)$. 33
3.3 Prova de que a heurística RL domina a estratégia R-em-5Rs. 34
3.4 Peça L de dimensões $(43, 26, 22, 16)$ empacotadas com 33 caixas $(7, 3)$ e $(3, 7)$. 36
3.5 Casos para a prova da Proposição 3.2. 37

4.1 Decomposição base para a heurística B-em-9Bs. 39
4.2 Peças do esquema BALST. 40
4.3 As diferentes peças L, A e T. 40
4.4 As duas decomposições de um T em dois T's. 41
4.5 Prova de que a heurística *BALST* domina a estratégia *B-em-9Bs*. 42

A.1 $P_{53} = (43, 26, 7, 3)$ (53 caixas) 45
A.2 $P_{57} = (49, 28, 8, 3)$ (57 caixas) 46
A.3 $P'_{69} = (57, 34, 7, 4)$ (69 caixas) 46
A.4 $P''_{69} = (63, 44, 8, 5)$ (69 caixas) 47

A.5 $P_{71} = (61, 35, 10, 3)$ (71 caixas) 47
A.6 $P_{75} = (67, 37, 11, 3)$ (75 caixas) 48
A.7 $P'_{77} = (61, 38, 10, 3)$ (77 caixas) 48
A.8 $P''_{77} = (61, 38, 6, 5)$ (77 caixas) 49
A.9 $P_{81} = (67, 40, 11, 3)$ (81 caixas) 49
A.10 $P'_{82} = (74, 49, 11, 4)$ (82 caixas) 50
A.11 $P''_{82} = (93, 46, 13, 4)$ (82 caixas) 50
A.12 $P'_{96} = (106, 59, 13, 5)$ (96 caixas) 51
A.13 $P''_{96} = (141, 71, 13, 8)$ (96 caixas) 51
A.14 $P_{97} = (74, 46, 7, 5)$ (97 caixas) 52
A.15 $P_{99} = (86, 52, 9, 5)$ (99 caixas) 52
A.16 $P_{100} = (108, 65, 10, 7)$ (100 caixas) 53

Capítulo 1

Introdução

Neste breve capítulo descrevemos, primeiramente, qual é o problema de que tratamos e como surgiu a idéia usada para atacá-lo, bem como alguns resultados obtidos. Em seguida, damos uma visão geral do que abordamos em cada capítulo desta tese.

1.1 A idéia chave deste trabalho

Este trabalho é sobre empacotamento. Quando falamos de empacotamento, estamos nos referindo a arrumações ou posicionamentos de *caixas* em *contêineres* de tal forma que as *caixas* não entrem umas nas outras e, além disso, não ultrapassem as fronteiras dos *contêineres*. Por enquanto, deixamos em aberto quais são as *caixas* e os *contêineres* de que tratamos e vamos descrever o que originou esta tese.

Sejam (ℓ, w) as dimensões de um retângulo "pequeno". Como arrumar o maior número destes retângulos "pequenos" dentro de um retângulo "grande" de dimensões (L, W)? Este problema é conhecido como *Manufacturer's Pallet Loading* (MPL) ou Problema de Carregamento de Palete do Produtor. Apesar de ter um enunciado bastante simples, não é conhecido nenhum método eficiente para resolvê-lo. Alguns autores sugerem, embora não tenham provado, que se trata de um problema NP-completo (Nelissen [6]). Sem entrar no mérito desta questão, o fato é que muitos métodos aproximados[1] e exatos (computacionalmente ineficientes) já foram propostos para o MPL.

Dentre os métodos aproximados para o MPL, há um descrito por Morabito e Morales [4] obtido através de um refinamento da heurística de Bischoff e Dowsland [15] que é o que mais se aproxima de um método exato. A heurística de Morabito e Morales que denotamos por R-em-5Rs consiste em "aplicar" recursivamente, sem degeneração, os padrões de decomposição[2] da Figura 1.1 de todas as maneiras sobre o retângulo (L, W) e armazenar a melhor solução encontrada, isto é, a com o maior número de retângulos pequenos (ℓ, w).

Embora bastante efetiva, a heurística R-em-5Rs não resolve otimamente alguns problemas. Por exemplo, se $\ell = 7$, $w = 3$, $L = 43$ e $W = 26$ sua solução tem apenas 52 caixas (ou retângulos "pequenos"), enquanto que a solução exata tem 53 caixas. O que falta à

[1] O termo método aproximado ou algoritmo aproximado, neste trabalho, tem conotação de simplesmente não garantir o ótimo.

[2] Note que o terceiro padrão, se degenerado, pode gerar os outros dois padrões mais simples. Por exemplo, se no terceiro padrão o tamanho dos dois retângulos mais a direita e do quadrado central forem zero, obtemos o primeiro padrão.

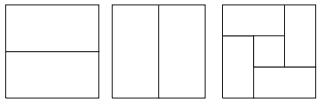


Figura 1.1: Padrões de decomposição da heurística R-em-5Rs.

heurística R-em-5Rs para que ela encontre a solução ótima quando $\ell = 7$, $w = 3$, $L = 43$ e $W = 26$? Analisando a solução exata deste problema (ver Figura 1.2) é fácil verificar que não existe nenhuma maneira de "aplicar" um dos três padrões da heurística R-em-5Rs sem que uma das linhas do padrão passe por dentro de uma caixa. Isto é, para capturarmos a solução exata deste problema, precisamos de pelo menos um padrão diferente dos três utilizados na heurística R-em-5Rs.

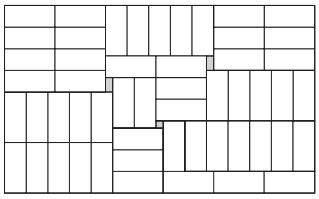


Figura 1.2: Solução exata para o problema $\ell = 7$, $w = 3$, $L = 43$ e $W = 26$ (53 caixas).

No conjunto de todos os padrões que decompõem retângulos em retângulos menores, como os da heurística R-em-5Rs, não há nenhum padrão que seja ao mesmo tempo simples e que capture a solução da Figura 1.2. O padrão de retângulos em retângulos menores, como os da Figura 1.1, mais simples para decompor a solução da Figura 1.2 divide o retângulo maior em 13 retângulos menores o que não pareceu uma extensão interessante para a heurística R-em-5Rs. Porém, se permitimos padrões de decomposição de retângulos em regiões no formato da letra L, vemos facilmente uma maneira de começar a decompor a arrumação da Figura 1.2 (veja o primeiro passo de decomposição na Figura 1.3). Após esta primeira decomposição, obtemos duas regiões no formato da letra L que precisam ser decompostas. A regra geral que estabelecemos foi então que os padrões de decomposição permitidos são todos aqueles que dividem uma região retangular ou uma região L em outras duas regiões menores, podendo cada uma destas regiões menores ser também um retângulo ou uma região L. A aplicação desta idéia esta ilustrada na Figura 1.3.

Batizamos esta abordagem original de decomposições de peças retangulares e em forma de L em duas outras peças menores também retangulares ou em forma de L como heurística RL. A heurística RL como provamos no capítulo 3 é uma aproximação para o problema MPL superior à heurística R-em-5Rs. De fato, até agora, não conhecemos nenhuma quádrupla (ℓ, w, L, W) para o problema MPL onde a heurística RL não tenha encontrado a solução ótima. Todos os casos conhecidos para os quais a heurística R-em-5Rs não obteve o ótimo foram solucionados pela heurística RL. Este fato empírico nos levanta a dúvida de estarmos diante de um algoritmo exato para o problema MPL. Como mostramos no capítulo 3, se tal dúvida se confirmasse o problema MPL teria solução polinomial em L (tamanho do maior lado do retângulo a ser empacotado).

A intuição obtida com a descoberta da heurística RL foi a de que permitir peças mais

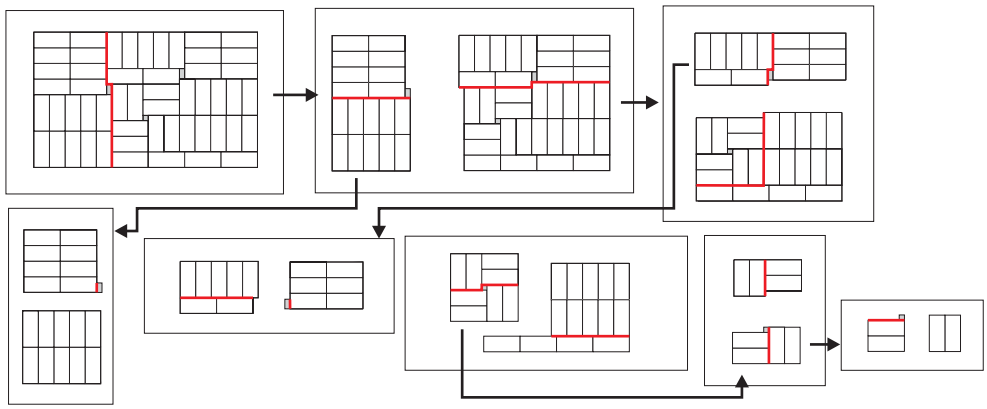


Figura 1.3: Solução exata de $MPL(7, 3, 43, 26)$ decomposta em peças L e peças R.

complicadas que um simples retângulo no caso bidimensional ou um paralelepípedo no tridimensional pode ser uma boa idéia para encontrar empacotamentos através de padrões de decomposição. A heurística RL, apesar de ter uma peça mais complicada (a peça L) que a heurística R-em-5Rs (só peças retangulares), tem padrões de decomposição mais simples. Os padrões da heurística RL dividem a peça maior em apenas duas peças menores enquanto que na heurística R-em-5Rs esta relação pode ser de até 1 pra 5.

Complementando o primeiro parágrafo desta seção, este trabalho é sobre uma abordagem específica para encontrar empacotamentos. A abordagem consiste em utilizar padrões de decomposição de peças (*contêineres*) para encontrar empacotamentos numa peça maior (o contêiner para o qual queremos um empacotamento de *caixas*). As caixas que queremos empacotar aqui são retângulos no caso bidimensional, paralelepípedos no caso tridimensional e seus análogos em dimensões maiores. Além disso, nos restringimos a uma família específica de empacotamentos onde as caixas não podem ser dispostas inclinadas: *empacotamentos ortogonais*. Nesta família, todos os lados de todas as caixas do empacotamento são ou paralelos ou perpendiculares aos lados do contêiner. Vale ressaltar que esta família, apesar de ser a mais comum nas aplicações, não "resolve" o problema de empacotamento geral.

1.2 Organização da tese

O capítulo 2 consiste na formalização das idéias mencionadas. Quais são as peças permitidas nos padrões de decomposição? Como é que, através dos padrões de decomposição, chegamos a um empacotamento? Como empacotar em dimensões maiores?

No capítulo 3, falamos sobre alguns problemas de empacotamento importantes em dimensão 2 e fazemos um breve histórico das abordagens desenvolvidas para estes problemas. Em seguida, descrevemos as heurística R-em-5Rs e RL utilizando a linguagem do capítulo 2. Mostramos por que a heurística RL é uma melhor aproximação para o problema MPL do que a heurística R-em-5Rs. Por fim, analisamos alguns aspectos teóricos específicos da heurística RL e propomos a conjectura de que a heurística RL é exata para o MPL.

No capítulo 4, abordamos problemas de empacotamento em dimensão 3. Descrevemos a heurística B-em-9Bs sugerida por Lins, Lins e Morabito [2] e apresentamos um método novo denotado por heurística BALST. Mostramos que a heurística BALST, em termos de

otimalidade, é superior à heurística B-em-9Bs.

O breve capítulo 5 indica os caminhos futuros que esta pesquisa pode tomar.

Capítulo 2

Teoria dos gblocos e gtets

Este capítulo introduz uma formalização para heurísticas de empacotamento, baseadas em esquemas de decomposição de peças em peças menores. Apesar de, no presente momento, só vermos aplicação para dimensões 2 e 3, esta formalização é adequada para quaisquer dimensões a partir de 2. As peças ou os *contêineres* em que desejamos empacotar *caixas* pertencem todos a uma classe de regiões chamada *gblocos*. Os *gblocos* têm a propriedade da *fconvexidade* que simplifica seus empacotamentos. Como vemos adiante, através desta propriedade é suficiente procurar decomposições que estejam numa *grade* adequada.

2.1 Caixas, blocos e gblocos

Esta seção caracteriza os elementos essenciais deste trabalho. Com estes elementos, é possível entender exatamente *o que* estamos buscando. Representações mais concretas destes elementos são formuladas nas seções posteriores.

caixa. Uma *caixa* ou, mais especificamente, uma *caixa de dimensão* n é um n-vetor com entradas inteiras positivas. Dizemos que duas caixas são iguais se e somente se seus n-vetores são iguais. Denotamos por $Caixa$ o conjunto de todas as possíveis caixas. Seja c uma caixa. Dizemos que o *volume da caixa* c ou $vol(c)$ é igual a $\prod_{i=1}^{n} c_i$.

Pela definição acima (ℓ, w) e (w, ℓ), com ℓ e w inteiros positivos, são caixas (em dimensão 2) diferentes. Isto pode parecer não intuitivo, pois caixas com lados iguais no mundo real podem ser alinhadas de maneira a que seus lados iguais fiquem todos orientados do mesmo jeito. Entretanto, estes alinhamentos não são permitidos aqui e, portanto, a caixa (ℓ, w) é diferente da caixa (w, ℓ).

bloco. Sejam $p = (p_1, p_2, \ldots, p_n)$ e $q = (q_1, q_2, \ldots, q_n)$ dois vetores de R^n com entradas reais, onde $p_i \leq q_i$ para $1 \leq i \leq n$. Dizemos que $\mathcal{B}(p, q)$, definido como a região em R^n formada pelos pontos $(x_1, x_2, \ldots, x_n)$ tais que $p_i \leq x_i \leq q_i$ para $1 \leq i \leq n$, é um *bloco* de dimensão n. Seja B o bloco $\mathcal{B}(p, q)$. O *tamanho de* B ou $tam(B)$ é o vetor $(q_1 - p_1, q_2 - p_2, \ldots, q_n - p_n)$. O *volume de* B ou $vol(B)$ é dado por $\prod_{i=1}^{n} (q_i - p_i)$. Denotamos por $Bloco$ o conjunto de todos os possíveis blocos.

Em dimensão 2, blocos são retângulos posicionados num plano onde cada um dos seus lados é paralelo a um dos eixos (x ou y). Em dimensão 3, blocos são paralelepípedos posicionados no espaço onde cada uma das suas faces é paralela a um dos eixos (x ou y ou z).

reflexões de eixos e translações. A *i-ésima reflexão de eixo* é a transformação linear que inverte o sinal da i-ésima coordenada. Uma *reflexão* é uma composição de i-reflexões para $i \in J$, com $J \subseteq \{1, 2, \ldots, n\}$. Uma reflexão é convenientemente descrita por um n-vetor com entradas em $\{0, 1\}$. Tal vetor é o vetor característico de $J \subseteq \{1, 2, \ldots, n\}$. Dado um ponto $x_0 \in R^n$, a x_0-*translação* é a transformação linear afim T_{x_0} definida por $T_{x_0}(x) = x + x_0$ para $x \in R^n$.

conjuntos pconvexos e fconvexos. Dado um ponto $x \in R^n$, denote por x^i o ponto obtido de x substituindo-se a sua i-ésima coordenada por zero. Ou seja, x^i é a projeção ortogonal de x no hiperplano coordenado perpendicular ao i-ésimo eixo. Seja A um subconjunto de R^n tal que para $x = (x_1, x_2, \ldots, x_n) \in A$ para todo $i \in \{1, 2, \ldots, n\}$ vale a seguinte propriedade: a interseção de A com a semi-reta que começa em x e estende-se na direção do i-ésimo eixo no sentido negativo é o segmento de reta entre x e x^i. Dizemos que A é um *conjunto projeção convexo* ou simplesmente um *conjunto pconvexo*. Um conjunto é *fracamente convexo* ou simplesmente *fconvexo* se por uma reflexão seguida de uma translação ele se torna pconvexo. Conjuntos fconvexos e seus posicionamentos pconvexos são centrais em nossa abordagem por causa da Proposição 2.12.

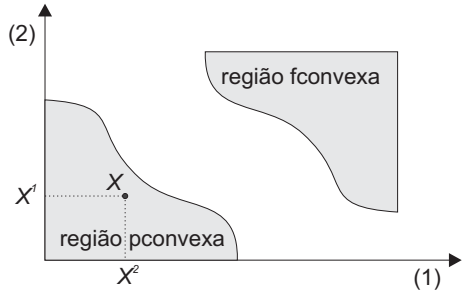


Figura 2.1: Exemplos de região pconvexa e fconvexa em dimensão 2

gbloco. Seja $B = \{B_1, B_2, \ldots, B_k\}$ um conjunto de blocos de mesma dimensão n. Se a união $\cup_{i=1}^{k} B_i$ é fconvexa, então ela é chamada um *gbloco* e é denotada por $U(B)$. Denotamos por $GBloco$ o conjunto de todos os possíveis gblocos.

Observe que, pela definição anterior, todo bloco é também um gbloco, mas nem todo gbloco é um bloco. A letra g em gbloco vem da palavra generalização. A figura a seguir ilustra exemplos de blocos e gblocos em dimensões 2 e 3.

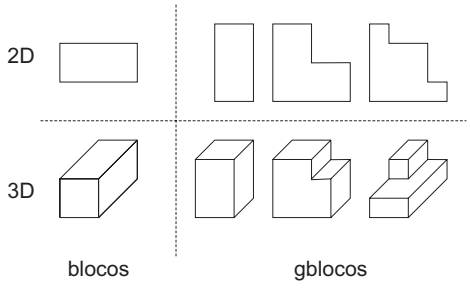


Figura 2.2: Exemplos de blocos e gblocos

empacotamento de caixas num gbloco. Dados um conjunto de caixas Y e um gbloco K, um *empacotamento de* Y *em* K é uma família de blocos $\mathcal{P}(Y, K)$ satisfazendo: (1) cada bloco membro está contido em K; (2) o tamanho de cada bloco é igual a uma caixa em Y (por isso os blocos de um empacotamento são chamados de *blocos-caixa*[1]); (3) o volume da interseção de dois blocos diferentes de $\mathcal{P}(Y, K)$ é zero. Note que cada lado de um bloco é paralelo a um dos eixos, caracterizando o que é conhecido como um *empacotamento ortogonal* [4]. A *ocupância* do empacotamento $\mathcal{P}(Y, K)$ é denotada por $ocup(\mathcal{P}(Y, K))$ e tem valor igual a soma dos volumes dos blocos que o constituem.

Encontrar bons empacotamentos de caixas em gblocos é o objetivo deste trabalho (entenda-se por bons empacotamentos aqueles que têm uma ocupância igual ou perto da maior ocupância possível). O método que usamos para buscar tais empacotamentos é, essencialmente, dividir gblocos maiores em gblocos menores, respeitando *padrões de decomposição* ou *decomposições* previamente estabelecidos, e avaliar quais divisões resultam nas melhores ocupâncias.

2.2 Gtets e atribuições

Podemos ver num gbloco dois elementos distintos: sua forma e seu tamanho/posição. A forma do gbloco é o elemento que diferencia por exemplo, em dimensão 2, um gbloco retangular de um gbloco em forma de L (ver Figura 2.2). Já o tamanho/posição do gbloco é o elemento que nos faz identificar que um gbloco é maior ou que está à esquerda de outro gbloco. Nesta seção, introduzimos o objeto chamado *gtet* que é uma maneira de representar a forma de um gbloco ou de mais de um gbloco. Se associarmos a um gtet uma *atribuição*, então passamos a poder também representar o tamanho/posição dos gblocos daquele gtet. O *gtet* é uma generalização do *tet* proposto em [2].

gtet. Seja $\Gamma = (G, f)$, tal que $G = (G_1, G_2, \ldots, G_n)$. $n \geq 1$. Cada G_i é um grafo sem laços com $v_i > 0$ vértices rotulados com os números $0, 1, \ldots, v_i - 1$. O grafo G_i também é dito um *perfil* do gtet Γ ou, mais especificamente, o i-ésimo *perfil* de Γ. O número de arestas em cada perfil de Γ é igual a b. As arestas de cada perfil são rotuladas com os números em $\{1, 2, \ldots, b\}$. Sejam a_j^i e b_j^i os rótulos dos dois vértices da aresta de rótulo j no perfil i. Denotamos por $m_i(j) = \min\{a_j^i, b_j^i\}$ e por $M_i(j) = \max\{a_j^i, b_j^i\}$. Cada rótulo de aresta j em $\{1, 2, \ldots, b\}$ define o bloco $B_j = \mathcal{B}((m_1(j), m_2(j), \ldots, m_n(j)), (M_1(j), M_2(j), \ldots, M_n(j)))$. Os perfis de Γ satisfazem a *propriedade de disjunção* que diz que para quaisquer dois rótulos de arestas diferentes α e β existe pelo menos um perfil G_i em que o intervalo $(\, m_i(\alpha), M_i(\alpha))\,)$ é disjunto do intervalo $(\, m_i(\beta), M_i(\beta)\,)$. A *função agrupamento* f é uma função cujo domínio é o conjunto de rótulos de arestas $\{1, 2, \ldots, b\}$ e sua imagem é o conjunto $\{1, 2, \ldots, k\}$, onde $k \leq b$. Seja A_h o subconjunto de rótulos de arestas cuja imagem sob f é h. Se para cada $h \in \{1, 2, \ldots, k\}$ temos que $\cup_{j \in A_h} B_j$ é uma região fconvexa e, além disso, a união $\cup_{j \in \{1,2,\ldots,b\}} B_j$ é também uma região fconvexa, dizemos que Γ é um *gtet de dimensão* n ou simplesmente um *gtet*. O conjunto de todos os gtets de qualquer dimensão é denotado por $GTet$.

[1] Intuitivamente, cada bloco do empacotamento é a imagem de uma caixa de Y por uma translação.

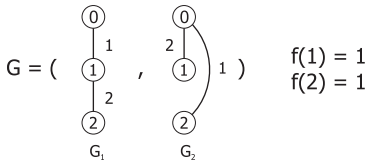


Figura 2.3: Exemplo de gtet

Analisando o exemplo de gtet acima temos que $n = 2$ (gtet de dimensão 2), $v_1 = 3$, $v_2 = 3$, $e = 2$. A *propriedade de disjunção* é verificada apesar de os intervalos para as arestas rotuladas com 1 e 2 terem interseção no segundo perfil G_2

$$(\, m_2(1), M_2(1)\,) \cap (\, m_2(2), M_2(2)\,) = (\, 0, 2\,) \cap (\, 0, 1\,) = (\, 0, 1\,) \neq \emptyset$$

no primeiro perfil G_1 os intervalos são disjuntos:

$$(\, m_1(1), M_1(1)\,) \cap (\, m_1(2), M_1(2)\,) = (\, 0, 1\,) \cap (\, 1, 2\,) = \emptyset$$

Neste exemplo, temos ainda que $b = 2$, $k = 1$ (a imagem de f é $\{1\}$) e $B_1 \cup B_2$ é uma região fconvexa ou um gbloco.

atribuição para um gtet e dgtet. Uma *atribuição de cotas δ para os vértices dos perfis de um gtet*, ou por simplicidade uma *atribuição* δ para um gtet de dimensão n é uma seqüência $\delta = (\delta_1, \delta_2, \ldots, \delta_n)$ onde cada δ_i é uma função dos vértices de G_i no conjunto R dos reais tal que $\delta_i(0) \leq \delta_i(1) \leq \ldots \leq \delta_i(v_i - 1)$. Uma atribuição δ é dita *não degenerada* se para $1 \leq i \leq n$ a condição $\delta_i(0) < \delta_i(1) < \ldots < \delta_i(v_i - 1)$ é verificada. Caso contrário, a atribuição δ é dita *degenerada*. Se todas as cotas de uma atribuição δ são inteiras e não negativas, dizemos que se trata de uma *atribuição inteira não negativa*. O conjunto de todas as atribuições possíveis é denotado por Δ. Doravante, quando falamos de um gtet Γ e de uma atribuição δ para este gtet, dizemos que se trata do *dgtet* (Γ, δ).

blocos induzidos por um dgtet. Denotamos por $bloco(\Gamma, \delta, i)$ o bloco $\mathcal{B}(p, q)$, onde $p = (\delta_1(m_1(i)), \delta_2(m_2(i)), \ldots, \delta_n(m_n(i))$ e $q = (\delta_1(M_1(i)), \delta_2(M_2(i)), \ldots, \delta_n(M_n(i))$. Também nos referimos a $bloco(\Gamma, \delta, i)$ como o *i-ésimo bloco induzido por* (Γ, δ). Denotamos por $blocos(\Gamma, \delta)$ o conjunto de blocos $\{B_1, B_2, \ldots, B_b\}$, onde $B_i = bloco(\Gamma, \delta, i)$ para $1 \leq i \leq b$. Também nos referimos a $blocos(\Gamma, \delta)$ como o *conjunto de blocos induzido por* (Γ, δ).

gblocos induzidos por um dgtet. Denotamos por $gbloco(\Gamma, \delta, i)$ o gbloco $U(\{bloco(\Gamma, \delta, j) \mid 1 \leq j \leq b, f(j) = i)\})$. Também nos referimos a $gbloco(\Gamma, \delta, i)$ como o *i-ésimo gbloco induzido por* (Γ, δ). Denotamos por $gblocos(\Gamma, \delta)$ o conjunto $\{K_1, K_2, \ldots, K_k\}$, onde $K_i = gbloco(\Gamma, \delta, i)$, $1 \leq i \leq k$. Também nos referimos a $gblocos(\Gamma, \delta)$ como o *conjunto de gblocos induzido por* (Γ, δ).

atribuição padrão para um gtet. Todo gtet tem uma *atribuição padrão*, $\delta^\star$, que associa ao vértice de rótulo r de um perfil a cota r. Doravante, quando falamos em blocos e gblocos de um gtet estamos nos referindo aos blocos e gblocos do gtet na sua atribuição padrão $\delta^\star$. Denotamos por $b(\Gamma)$ o número de blocos do gtet Γ ou $|blocos(\Gamma, \delta^\star)|$. Denotamos por $k(\Gamma)$ o número de gblocos do gtet Γ ou $|gblocos(\Gamma, \delta^\star)|$.

A Figura 2.4 exemplifica um dgtet de dimensão 2. Seus blocos são B_1 e B_2 e K_1 é seu único gbloco. A atribuição δ neste caso é uma atribuição *não degenerada*.

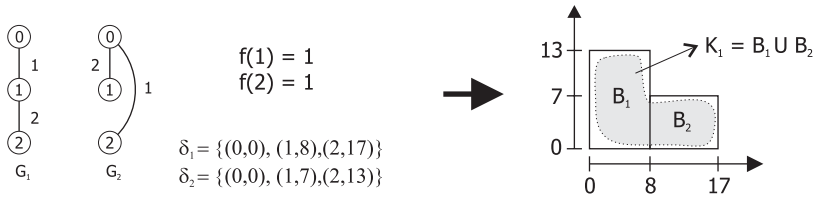


Figura 2.4: Exemplo de dgtet

2.3 Gtet pai e gtet filho

Apresentamos, nesta pequena seção, os conceitos de gtet pai e gtet filho. Estes são conceitos bastante simples que usamos nas próximas seções.

pai de um gtet. Seja $\Gamma = (G, f)$ com arestas rotuladas de 1 a b então $pai(\Gamma) = \Gamma'$ com $\Gamma' = (G, f')$, onde a função de agrupamento f' é a função constante $f'(j) = 1$ para $1 \leq j \leq b$. Assim o pai de um gtet induz um único gbloco que é igual a união dos seus gblocos.

Uma observação importante, decorrente da definição acima, é que se $k(\Gamma) = 1$ então $\Gamma = pai(\Gamma)$.

Proposição 2.1 *Qualquer atribuição para um gtet é também uma atribuição para seu pai e vice-versa.*

Prova: Como os vértices nos perfis de Γ e de $pai(\Gamma)$ são exatamente os mesmos, uma atribuição para Γ é também uma atribuição para $pai(\Gamma)$ e vice-versa. ■

filho de um gtet. Seja $\Gamma = ((G_1, G_2, \ldots, G_n), f)$ um gtet com arestas rotuladas de 1 a b então $filho(\Gamma, i) = \Gamma'$ com $\Gamma' = ((G'_1, G'_2, \ldots, G'_n), f')$, onde G'_j é o perfil G_j sem as arestas com rótulos em $\{a \mid f(a) \neq i\}$. As arestas que sobram, de rótulos $i_1 < i_2 < \ldots < i_{b'}$, são re-rotuladas $1, 2, \ldots, b'$ onde $b' = b - |\{a \mid f(a) \neq i\}|$. A função agrupamento f' é a função constante $f'(j) = 1$ para $1 \leq j \leq b'$. Desse modo, $filho(\Gamma, i)$ é o gtet correspondente ao i-ésimo gbloco de Γ. Os vértices nos perfis de $filho(\Gamma, i)$ são exatamente os mesmos nos perfis de Γ. Note que podem aparecer, em $filho(\Gamma, i)$, vértices isolados que não eram isolados em Γ.

Proposição 2.2 *Qualquer atribuição para um gtet é também uma atribuição para cada um de seus filhos e vice-versa.*

Prova: Como os vértices nos perfis de Γ e de $filho(\Gamma, i)$ são exatamente os mesmos, uma atribuição para Γ é também uma atribuição para $filho(\Gamma, i)$ e vice-versa. ■

2.4 Similaridade entre gtets

Neste ponto vale ressaltar que o objeto gtet é capaz de representar qualquer gbloco bem como qualquer decomposição de gbloco em gblocos, sendo esta a grande importância

deste objeto para nós. São os gblocos e suas decomposições a essência da nossa busca por empacotamentos. Entretanto, um problema que ainda existe é o fato de que um mesmo gbloco ou uma mesma decomposicão de um gbloco em gblocos é, no caso geral, representada por mais de um gtet diferente, como ilustra a Figura 2.5. A seguir, tratamos de resolver este problema introduzindo o conceito de gtet similar mínimo ou *simin*.

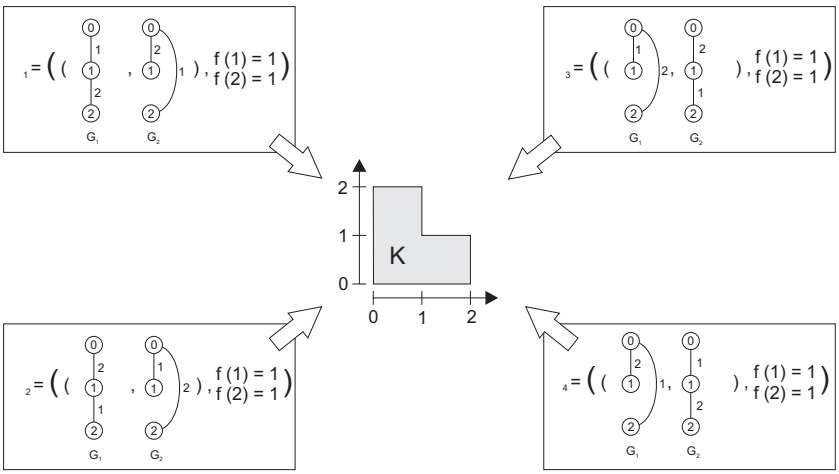


Figura 2.5: GTets diferentes representando o mesmo gbloco K.

gtets similares. Sejam Γ_1 e Γ_2 dois gtets. Dizemos que eles são similares, $\Gamma_1 \sim \Gamma_2$, se e somente se para toda atribuição não degenerada δ_i para Γ_i existe uma atribuição não degenerada δ_j para Γ_j onde (i, j) é uma permutação de $(1, 2)$ tal que $gblocos(\Gamma_1, \delta_1) = gblocos(\Gamma_2, \delta_2)$. Ou seja, dois gtets são ditos similares quando eles podem, a menos de atribuições não degeneradas, representar os mesmos gblocos. Note que a definição de *gblocos* independe da rotulação das arestas do gtet e da maneira específica como cada gbloco é particionado em blocos. A relação $\sim$ é, por definição, reflexiva e simétrica. Vê-se, facilmente, que ela é transitiva e, assim, é uma relação de equivalência particionando o conjunto $GTet$. Um algoritmo para decidir se gtets são similares é um problema, nesta generalidade, bastante difícil. Decidir similaridade é imprescindível. Portanto, nos casos restritos, com os quais vamos trabalhar, um algoritmo para esta tarefa é descrito.[2]

código de um gtet. A função $codigo : GTet \to (N \cup \{\text{“;”}, \text{“,”}, \text{“-”}\})^*$ mapeia um gtet numa palavra sobre o alfabeto contendo os número naturais, o ponto-e-vírgula “;”, a vírgula “,” e a barra “-”.

$$
\begin{aligned}
&codigo(\Gamma) = n; b;\\
&\quad v_1, v_2, \ldots, v_n;\\
&\quad m_1(1), m_2(1), \ldots, m_n(1) - M_1(1), M_2(1), \ldots, M_n(1);\\
&\quad m_1(2), m_2(2), \ldots, m_n(2) - M_1(2), M_2(2), \ldots, M_n(2);\\
&\quad \ldots\\
&\quad m_1(b), m_2(b), \ldots, m_n(b) - M_1(b), M_2(b), \ldots, M_n(b);\\
&\quad f(1), f(2), \ldots, f(b)
\end{aligned}
$$

O código para o gtet da Figura 2.3 é $2; 2; 3, 3; 0, 0 - 1, 2; 1, 0 - 2, 1; 1, 1$. Note que a função *codigo* é uma bijeção admitindo portanto a inversa $codigo^{-1} : (N \cup \{\text{“;”}, \text{“,”}, \text{“-”}\})^* \to GTet$.

[2] A necessidade de tratar atribuições com cotas reais decorre da noção de similar que seria incompleta caso estivéssemos restritos a atribuições com cotas inteiras.

ordem total entre códigos e gtets ($\preceq$). Sejam $c_1, c_2 \in (N \cup \{\text{“;”}, \text{“,”}, \text{“-”}\})^*$ dois códigos. Dizemos que $c_1 \preceq c_2$, c_1 é menor ou igual a c_2, quando a palavra c_1 for lexicograficamente menor ou igual à palavra c_2. Note que $\preceq$ é uma relação de ordem total, ou seja, para quaisquer dois códigos c_1, c_2 temos $c_1 \preceq c_2$ ou $c_2 \preceq c_1$. Fazemos esta ordem total ser induzida em *GTet* pela bijeção *codigo*: definimos $\Gamma_1 \preceq \Gamma_2$ se $codigo(\Gamma_1) \preceq codigo(\Gamma_2)$.

similar mínimo de um gtet. O *similar mínimo de um gtet* Γ ou, *simin de* Γ, é o gtet $\mathcal{S}(\Gamma)$ definido como o menor gtet similar a Γ. Um gtet Γ é dito um *simin* se e somente se $\Gamma = \mathcal{S}(\Gamma)$.

complexidade de um gbloco. A *complexidade de um gbloco* K é o menor número de blocos numa cobertura de K por blocos *quasi-disjuntos*, isto é, a interseção de quaisquer dois blocos distintos da cobertura tem volume zero.

algoritmo para encontrar o simin de um gtet. Depois da dimensão $n = n(\Gamma)$, a próxima entrada do $codigo(\Gamma)$ é $b = b(\Gamma)$. Assim, para um código ser de um simin, o valor de b é o menor possível. Para encontrar o simin de um *gtet* precisamos considerar todas as coberturas de cada um dos seus gblocos num número mínimo de blocos quasi-disjuntos cuja soma totaliza $b_\star$ blocos. Se os gblocos são $K_1, K_2, \ldots, K_k$, seja $\mathcal{C}ovBloc(K_i)$, $i = 1, \ldots, k$, o conjunto de todas as coberturas de K_i num menor número possível de blocos quasi-disjuntos. É fácil observar que cada elemento $\Pi \in \mathcal{C}ovBloc(\Gamma) = \mathcal{C}ovBloc(K_1) \times \mathcal{C}ovBloc(K_2) \times \ldots \times \mathcal{C}ovBloc(K_k)$ é uma cobertura quasi-disjunta dos gblocos de Γ num mínimo número de blocos possível $b_\star$. Além disto, os Π's estão em correspondência bijetiva com os gtets com $b = b_\star$ e que são similares a Γ. Denotamos por Γ_Π o gtet correspondente a Π por esta bijeção. Dado um gtet Γ denotamos por Γ^π o gtet obtido de Γ, removendo os vértices isolados dos seus perfis e re-rotulando os códigos das arestas (os blocos) pela permutação $\pi \in S_b$. Podemos então afirmar que

$$\mathcal{S}(\Gamma) = \min\{\Gamma^\pi_\Pi \mid \Pi \in \mathcal{C}ovBloc(\Gamma), \pi \in S_b\}.$$

Observe que $|\mathcal{C}ovBloc(\Gamma)|$ é finito. Precisamos apenas de um procedimento para encontrar $\mathcal{C}ovBloc(\Gamma)$ e outro para encontrar $codigo(\Gamma^\pi_\Pi)$. Encontrar $\mathcal{C}ovBloc(\Gamma)$ é, em geral, impraticável, porém nos casos relevantes para o nosso trabalho $b_\star$ é no máximo 6, $|\mathcal{C}ovBloc(\Gamma)|$ é no máximo 36 e $\mathcal{C}ovBloc(\Gamma)$ pode ser tabelada para os poucos Γ's relevantes. Quanto a encontrar π que minimize Γ^π_Π é uma tarefa bastante simples: a permutação de S_b que renomeia os rótulos de arestas (os blocos) está inteiramente amarrada e pode ser facilmente obtida quando exigimos que o código resultante seja mínimo. Dessa maneira, nos casos relevantes para o nosso trabalho, encontrar $\mathcal{S}(\Gamma)$ é uma tarefa computacionalmente barata. A importância do simin decorre da proposição seguinte.

Proposição 2.3 *Dois gtets são similares se e somente se seus simins são iguais:* $\Gamma_1 \sim \Gamma_2 \Leftrightarrow \mathcal{S}(\Gamma_1) = \mathcal{S}(\Gamma_2)$.

Prova: ($\Rightarrow$) Tome por hipótese que Γ_1 e Γ_2 são gtets similares. Como $\mathcal{S}(\Gamma_1)$ é similar a Γ_1, sabemos que para qualquer atribuição não degenerada δ'_1 para $\mathcal{S}(\Gamma_1)$ existe uma atribuição não degenerada δ_1 para Γ_1 que induz os mesmos gblocos. Da mesma forma, como Γ_2 é similar a Γ_1 então existe também uma atribuição não degenerada δ_2 para Γ_2 que induz os mesmos gblocos. Completando a cadeia, como $\mathcal{S}(\Gamma_2)$ é similar a Γ_2 então existe uma atribuição não degenerada δ'_2 para $\mathcal{S}(\Gamma_2)$ que induz os mesmos gblocos.

$$gblocos(\mathcal{S}(\Gamma_1), \delta'_1) = gblocos(\Gamma_1, \delta_1) = gblocos(\Gamma_2, \delta_2) = gblocos(\mathcal{S}(\Gamma_2), \delta'_2).$$

Isto é, dada uma atribuição não degenerada qualquer δ_1' para $\mathcal{S}(\Gamma_1)$, temos uma atribuição não degenerada δ_2' para $\mathcal{S}(\Gamma_2)$ que induz os mesmos gblocos. Como o raciocínio é simétrico, podemos também afirmar que dada uma atribuição não degenerada qualquer δ_2' para $\mathcal{S}(\Gamma_2)$ temos uma atribuição não degenerada δ_1' para $\mathcal{S}(\Gamma_1)$ que induz os mesmos gblocos. Segue então, pela definição de similar, que $\mathcal{S}(\Gamma_1)$ é similar a $\mathcal{S}(\Gamma_2)$. Pela minimalidade simultânea de $\mathcal{S}(\Gamma_1)$ e de $\mathcal{S}(\Gamma_2)$ eles têm de ser iguais

($\Leftarrow$) Tome por hipótese que $\mathcal{S}(\Gamma_1) = \mathcal{S}(\Gamma_2)$. Logo $\mathcal{S}(\Gamma_1)$ e $\mathcal{S}(\Gamma_2)$ são similares. Como Γ_1 é similar a $\mathcal{S}(\Gamma_1)$, sabemos que para qualquer atribuição não degenerada δ_1 para Γ_1 existe uma atribuição não degenerada δ_1' para $\mathcal{S}(\Gamma_1)$ que induz os mesmos gblocos. Da mesma forma, como $\mathcal{S}(\Gamma_2)$ é similar a $\mathcal{S}(\Gamma_1)$ então existe também uma atribuição não degenerada δ_2' para $\mathcal{S}(\Gamma_2)$ que induz os mesmos gblocos. Completando a cadeia, como Γ_2 é similar a $\mathcal{S}(\Gamma_2)$ então existe uma atribuição não degenerada δ_2 para Γ_2 que induz os mesmos gblocos.

$$gblocos(\Gamma_1, \delta_1) = gblocos(\mathcal{S}(\Gamma_1), \delta_1') = gblocos(\mathcal{S}(\Gamma_2), \delta_2') = gblocos(\Gamma_2, \delta_2).$$

Isto é, dada uma atribuição não degenerada qualquer δ_1 para Γ_1 temos uma atribuição não degenerada δ_2 para Γ_2 que induz os mesmos gblocos. Como o raciocínio é simétrico, podemos também afirmar que dada uma atribuição não degenerada qualquer δ_2 para Γ_2, temos uma atribuição não degenerada δ_1 para Γ_1 que induz os mesmos gblocos. Segue então, pela definição de similar, que Γ_1 é similar a Γ_2. ■

Proposição 2.4 *Sejam Γ_1 e Γ_2 dois simins, se existem atribuições não degeneradas δ_1 para Γ_1 e δ_2 para Γ_2 tais que $gblocos(\Gamma_1, \delta_1) = gblocos(\Gamma_2, \delta_2)$, então $\Gamma_1 = \Gamma_2$ e $\delta_1 = \delta_2$.*

Prova: O número de gblocos induzidos por Γ_1 e Γ_2 é igual, $k(\Gamma_1) = k(\Gamma_2)$, pois $gblocos(\Gamma_1, \delta_1) = gblocos(\Gamma_2, \delta_2)$ e δ_1 e δ_2 são não degeneradas. Sejam i_1 e i_2 tais que $gbloco(\Gamma_1, \delta_1, i_1) = gbloco(\Gamma_2, \delta_2, i_2)$, o filho i_1 de Γ_1 e i_2 de Γ_2 tem de ter o mesmo número de arestas (nos simins os gblocos induzidos têm o menor número de arestas possível). Logo, o número de arestas de Γ_1 tem de ser igual ao de Γ_2 (a soma das arestas dos filhos), ($b(\Gamma_1) = b(\Gamma_2)$. O número de vértices incidentes não isolados em cada filho tem de ser igual também, bem como suas cotas. Segue daí que $\delta_1 = \delta_2$. Logo toda atribuição para Γ_1 pode ser usada em Γ_2 que resultará nos mesmos gblocos induzidos e vice-versa. Concluímos então que Γ_1 é similar a Γ_2 e como os dois são simins então eles são iguais. ■

Proposição 2.5 *Sejam Γ_1 e Γ_2 dois gtets, se existem atribuições não degeneradas δ_1 para Γ_1 e δ_2 para Γ_2 tais que $gblocos(\Gamma_1, \delta_1) = gblocos(\Gamma_2, \delta_2)$, então Γ_1 é similar a Γ_2.*

Prova: Pela definição de similar, sabemos que existem atribuições δ_1' e δ_2' não degeneradas tais que:

$$gblocos(\mathcal{S}(\Gamma_1), \delta_1') = gblocos(\Gamma_1, \delta_1) = gblocos(\Gamma_2, \delta_2) = gblocos(\mathcal{S}(\Gamma_2), \delta_2').$$

Pela Proposição 2.4, temos então que $\mathcal{S}(\Gamma_1) = \mathcal{S}(\Gamma_2)$ o que, pelo Proposicao 2.3, implica em Γ_1 ser similar a Γ_2 ■

O gtet é a maneira concreta com que representamos os gblocos e as decomposicões de gbloco em gblocos. Tendo isto em mente, a importância dos resultados acima vem do

fato de que se não fôssemos capazes de representar um gbloco ou uma decomposição de gblocos por um único gtet (o gtet similar mínimo), correríamos o risco de estar procurando diversas vezes empacotamentos num mesmo gbloco ou usando diversas vezes uma mesma decomposição.

2.5 Gtet representante

Ainda com o mesmo espírito dos gtets simins (similares mínimos), existem outras maneiras de identificar gtets diferentes que são iguais do ponto de vista de empacotamento. Os simins, só para lembrar, são na verdade o passo mais básico e identificam gtets que representam exatamente a mesma região (gbloco) ou a mesma decomposição de uma região em regiões menores (gbloco em gblocos menores). Nesta seção e na próxima seção, mostramos outras duas maneiras de fazer tais identificações.

Intuitivamente, regiões que diferem umas das outras apenas por reflexões de eixo têm seus empacotamentos diferindo pelas mesmas reflexões como ilustra a Figura 2.6. Nesta figura, a região em forma de L tem três caixas empacotadas. Uma vez que um empacotamento destes quatro é descoberto, os outros são obtidos facilmente por reflexão de eixo. Da mesma forma que a reflexão, as translações das regiões não interferem na essência do empacotamento. A idéia então é identificar para os gtets uma reflexão e uma translação canônicas e trabalhar apenas com gtets neste estado, os chamados gtets *representantes*.

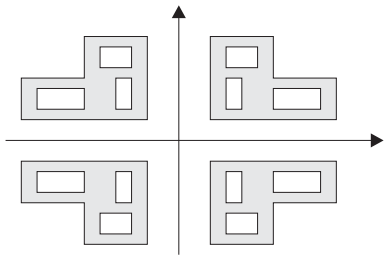


Figura 2.6: Empacotamentos refletidos.

reversão de um gtet (rev). Seja $\Gamma = ((G_1, G_2, \ldots, G_n), f)$ e $r = (r_1, r_2, \ldots, r_n)$ com $r \in \{0,1\}^n$ então $rev(\Gamma, r) = \Gamma'$ com $\Gamma' = ((G'_1, G'_2, \ldots, G'_n), f)$ onde $G'_i = G_i$ caso r_i seja igual a 0 e $G'_i = updown(G_i)$ caso r_i seja igual a 1 (*updown* simplesmente troca os rótulos dos v vértices de um grafo perfil usando a regra: vértice com rótulo j passa a ter rótulo $v_i - 1 - j$). Note que uma reversão de um gtet é realizada geometricamente por reflexões seguidas de uma translação apropriada.

reversão simétrica de um gtet. Seja r uma reversão e Γ um gtet. Se $\mathcal{S}(\Gamma) = \mathcal{S}(rev(\Gamma, r))$, dizemos que r é uma *reversão simétrica* para Γ. Note que a reversão nula é sempre uma reversão simétrica.

Como os exemplos de gtets desta tese são todos de dimensão menor ou igual a 3, podemos representar uma reversão por um único dígito decimal. Por exemplo, se as reversões simétricas de um determinado gtet de dimensão 3 são $(0,0,0)$ e $(0,0,1)$, fazendo a leitura da direita para a esquerda, temos os números em binário $(000)_2$ e $(001)_2$ que em decimal são representados por $(0)_{10}$ e $(4)_{10}$ ou, simplesmente, 0 e 4.

reversão de uma atribuição (rev'). Seja $\delta = (\delta_1, \delta_2, \ldots, \delta_n)$ uma atribuição e $r =$

$(r_1, r_2, \ldots, r_n)$ com $r \in \{0,1\}^n$ então $rev'(\delta, r) = \delta'$ com $\delta' = (\delta'_1, \delta'_2, \ldots, \delta'_n)$ onde $\delta'_i = \delta_i$ caso r_i seja igual a 0 e $\delta'_i = aupdown(\delta_i)$ caso r_i seja igual a 1 (*aupdown* faz o seguinte: $\delta'_i(j) = \delta_i(v_i - 1) - \delta_i(j)$).

gtet pconvexo. Um gtet Γ é pconvexo se o gbloco induzido pelo seu pai é pconvexo.

Proposição 2.6 *Se Γ é um gtet sem vértices isolados, então existe uma reversão r que o torna pconvexo.*

Prova: Pela definição de gtet a união de seus gblocos é fconvexa, portanto existe uma reflexão ref seguida de uma translação tr que torna esta região pconvexa. Uma reversão de um gtet, como sabemos, tem o efeito de uma reflexão seguida de uma translação específica em seus gblocos induzidos. Após uma reversão de um gtet (e antes também), o primeiro vértice de cada perfil tem, na atribuição padrão, cota igual a 0 (zero). Como, por hipótese, estamos falando de gtets sem vértices isolados, existem blocos induzidos pelo gtet revertido que contêm pontos com coordenadas igual a 0 (zero) em todas as dimensões. Isto é, a união dos gblocos induzidos por um gtet sem vértices isolados tangencia "por cima" todos os hiperplanos $H_i = \{x \in R^n \mid x_i = 0\}$ para $1 \leq i \leq n$. Esta propriedade, que denotamos por *condição de alinhamento*, é necessária para que uma região seja pconvexa. De fato, a translação tr , que é única, tem por objetivo fazer com que a região refletida por ref satisfaça a *condição de alinhamento*. Portanto, como o efeito da translação tr é obtido naturalmente por qualquer reversão do gtet sem vértices isolados, a reversão r que reverte os mesmos eixos refletidos pela reflexão ref torna pconvexa a união dos gblocos induzidos por aquele gtet. ∎

Embora a proposição anterior seja suficiente para o que precisamos, é possível mostrar que basta os vértices extremos (o de menor e maior rótulo) de cada perfil de Γ não serem isolados para que a reversão r exista.

representante de um gtet. Seja Γ um gtet. O *representante* de Γ é dado por $\mathcal{R}(\Gamma) = \min\{\mathcal{S}(rev(\Gamma, r)) \mid \mathcal{S}(rev(\Gamma, r))$ é pconvexo e $r \in \{0,1\}^n\}$. Denotamos por $\mathcal{R}_{rev}(\Gamma)$ a reversão lexicograficamente mínima em $\{r \mid \mathcal{R}(\Gamma) = \mathcal{S}(rev(\Gamma, r))$ e $r \in \{0,1\}^n\}$ (qualquer reversão deste conjunto serviria). Um gtet Γ é dito um *representante* se e somente se $\Gamma = \mathcal{R}(\Gamma)$.

Proposição 2.7 *Todo gtet representante é um simin.*

Prova: Esta proposição decorre diretamente da definição de representante, pois todos os elementos no conjunto $\{\mathcal{S}(rev(\Gamma, r)) \mid \mathcal{S}(rev(\Gamma, r))$ é pconvexo e $r \in \{0,1\}^n\}$, no qual o representante é o mínimo, são simins. ∎

2.6 Gtet p-representante

Sejam (ℓ, w) e (w, ℓ) duas caixas que queremos empacotar em maior número possível no retângulo (L, W). Seja $\mathcal{P}$ o empacotamento ótimo obtido para tal problema. Suponha agora que quiséssemos empacotar as mesmas duas caixas no retângulo (W, L). Como o conjunto de caixas possíveis não varia se trocarmos a dimensão x pela dimensão y, e o retângulo (W, L) é exatamente o retângulo (L, W) trocando a dimensão x pela dimensão

y, então a solução para este nosso novo problemas é simplesmente trocar as dimensões x e y no empacotamento $\mathcal{P}$. A noção de gtet p-representante é exatamente para capturar a idéia deste exemplo.

partição de eixos. Denotamos por P_n o conjunto de partições de $\{1, 2, \ldots, n\}$. Seja $p \in P_n$. Uma permutação π de S_n é p-fechada se, para todo $i = 1, 2, \ldots, n$, i e $\pi(i)$ estão na mesma parte de p. Ou seja, a imagem de cada parte de p sob π é ela mesma. Para p em P_n, o conjunto de todas as permutações p-fechadas forma um subgrupo de S_n denotado por S_n^p e é chamado de grupo de permutações induzidas por p. O papel da partição p é identificar quais eixos são equivalentes do ponto de vista de empacotamento. Pares de eixos na mesma parte de p são ditos *indistinguíveis*. Em dimensão 2, os papéis de eixos x_1 e x_2 são, em geral, indistinguíveis, e a partição associada tem uma única parte $\{\{1, 2\}\}$. Problemas em dimensão 3 nos quais a vertical dos blocos a serem empacotados (eixo x_3) precisa ser mantida, a partição conveniente é $\{\{1, 2\}, \{3\}\}$. Se os blocos podem tombar então a partição passa a ser $\{\{1, 2, 3\}\}$.

partição de eixos induzida por um conjunto de caixas. Seja Y o conjunto $\{Y_1, Y_2, \ldots, Y_m\}$ de m caixas de dimensão n. Seja $y_j = (y_{j1}, y_{j2}, \ldots, y_{jn})$ a j-ésima caixa em Y. Seja $swap(v, i, \ell)$ o vetor resultante da troca dos valores do vetor v nas posições i e ℓ. Seja $Y_{i\ell} = \{swap(y_j, i, \ell) \mid 1 \leq j \leq m\}$. Dizemos que os eixos i e ℓ são intercambiáveis se $Y = Y_{i\ell}$. Esta relação é uma relação de equivalência e particiona os eixos. Esta partição é denotada por $p(Y)$. Por exemplo, se $Y = \{(790, 513, 165), (513, 790, 165), (790, 165, 513), (165, 790, 513)\}$ então $p(Y) = \{\{1, 2\}, \{3\}\}$, pois $Y = Y_{12}$, $Y \neq Y_{13}$ e $Y \neq Y_{23}$.[3]

permutação de um gtet ($perm$). Seja $\Gamma = ((G_1, G_2, \ldots, G_n), f)$ e $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ uma permutação de $(1, 2, \ldots, n)$ então $perm(\Gamma, \alpha) = \Gamma'$ com $\Gamma' = ((G_{\alpha_1}, G_{\alpha_2}, \ldots, G_{\alpha_n}), f)$, ou seja, permutam-se os perfis de Γ através de α.

permutação de uma atribuição ($perm'$). Seja $\delta = (\delta_1, \delta_2, \ldots, \delta_n)$ uma atribuição e $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ uma permutação de $(1, 2, \ldots, n)$ então $perm'(\delta, \alpha) = \delta'$ com $\delta' = (\delta_{\alpha_1}, \delta_{\alpha_2}, \ldots, \delta_{\alpha_n})$.

p-representante de um gtet. Sejam p uma partição dos eixos e Γ um gtet. O *p-representante* de Γ é dado por $\mathcal{R}'(\Gamma, p) = \min\{\mathcal{R}(perm(\Gamma, \pi)) \mid \pi \in S_n^p\}$. Denotamos por $\mathcal{R}'_{perm}(\Gamma, p)$ a permutação lexicograficamente mínima em $\{\pi \mid \mathcal{R}'(\Gamma, p) = \mathcal{R}(perm(\Gamma, \pi))$ e $\pi \in S_n^p\}$ (qualquer permutação deste conjunto serviria). Um gtet Γ é dito um *p-representante* se e somente se $\Gamma = \mathcal{R}'(\Gamma, p)$.

Proposição 2.8 *Todo gtet p-representante é um representante.*

Prova: Esta proposição decorre diretamente da definição de p-representante, pois todos os elementos no conjunto $\{\mathcal{R}(perm(\Gamma, \pi)) \mid \pi \in S_n^p\}$, no qual o p-representante é o mínimo, são representantes. ∎

2.7 Minors e dgtets degenerados

Um dgtet com atribuição degenerada é dito um *dgtet degenerado*. A noção importante que pretendemos definir nesta seção e que está relacionada com atribuições degeneradas são os *minors*.

[3] O conjunto de caixas Y, neste exemplo foi obtido de um problema de empacotamento de fardos de papelão, onde existe a restrição de que a maior dimensão não pode estar na vertical.

limpando um dgtet degenerado dos blocos de volume zero (*limp*). Seja (Γ, δ) um dgtet com δ degenerado. Seja A o conjunto de rótulos de arestas cujas extremidades, em pelo menos um perfil, tenham cotas iguais. Seja Γ' o gtet obtido a partir de Γ da seguinte forma: (1) remover as arestas que tenham rótulos em A de todos os perfis de Γ; (2) identificar os vértices de um mesmo perfil que têm a mesma cota num único vértice; (3) re-rotular as arestas e adequar a função agrupamento para que Γ' seja um gtet. Este ajuste respeita a seguinte restrição: se as arestas de rótulos a' e b' em Γ' eram rotuladas como a e b em Γ, então $f'(a') = f'(b') \Leftrightarrow f(a) = f(b)$, onde f e f' são as funções agrupamento de Γ e Γ'. A atribuição δ' é obtida de δ eliminando as repetições nas seqüências de cotas de cada perfil, o que a torna uma atribuição não degenerada para Γ'. Seja δ' a atribuição obtida a partir de δ eliminando as repetições nas seqüências de cotas de cada perfil, tornando-a uma atribuição não degenerada para Γ'. Denotamos por $limp(\Gamma, \delta)$ qualquer par (Γ', δ') obtido da maneira definida acima. A construção é única a menos da re-rotulação dos blocos e re-rotulação dos filhos. Note que a limpeza de um dgtet não altera em nada os gblocos induzidos, uma vez que as arestas removidas (conjunto A) correspondem a blocos de volume zero.

minor de um gtet. Sejam Γ_m e Γ dois gtets. Dizemos que Γ_m é um *minor* de Γ se e somente se existe uma atribuição degenerada δ para Γ tal que $(\Gamma', \delta') = limp(\Gamma, \delta)$ e $\mathcal{S}(\Gamma') = \mathcal{S}(\Gamma_m)$.

2.8 Esquemas, peças e decomposições

Os esquemas são como uma lei para reger a maneira na qual podemos decompor uma *peça*. Seguindo este raciocínio, faz sentido falar em decomposições que obedecem a um dado esquema. A Figura 1.1 é uma boa maneira de visualizar um esquema. Nela estão os padrões ou *decomposições* permitidos para uma peça (neste caso, a única peça possível é o retângulo) segundo a heurística R-em-5Rs. Nesta seção, procuramos definir formalmente o que são os esquemas, as peças e as decomposições.

vértice móvel num perfil de um gtet. Seja δ uma atribuição não degenerada para o gtet Γ. Considere o vértice i pertencente ao j-ésimo perfil G_j de Γ. Seja δ' obtida a partir de δ modificando apenas a cota do vértice i de $\delta_j(i)$ para $\delta'_j(i) \in [\delta_j(i-1), \delta_j(i+1)]$. O vértice i é dito móvel se o gbloco induzido por $(pai(\Gamma), \delta')$ é exatamente igual ao gbloco induzido por $(pai(\Gamma), \delta)$. Note que δ e δ' também são atribuições para $pai(\Gamma)$ (Proposição 2.1). Ou seja, um vértice móvel do perfil j é aquele cuja cota pode mudar de valor (num intervalo adequado) sem que o gbloco induzido pelo pai daquele gtet mude. Um vértice que não é móvel é dito um vértice *fixo*.

Podemos, facilmente, provar que, para verificar se um vértice de um gtet Γ é móvel, é suficiente testar a condição de invariância do gbloco induzido para uma atribuição δ não degenerada particular, por exemplo, a atribuição padrão $\delta^\star$.

Proposição 2.9 *Se Γ é um gtet que induz um único gbloco, $k(\Gamma) = 1$, e tem vértice móvel, então existe Γ' similar a Γ tal que $\Gamma' < \Gamma$*

Prova: Seja i um vértice móvel no j-ésimo perfil de Γ. Seja δ uma atribuição não degenerada para Γ. Pela definição de vértice móvel podemos trocar a cota do vértice i no perfil j da atribuição δ por valores no intervalo $[\delta_j(i-1), \delta_j(i+1)]$ que o gbloco induzido não muda. Podemos simular esta mudança de cota da seguinte maneira: seja Γ' o gtet

obtido a partir de Γ, removendo o vértice i no perfil j, e fazendo com que todas as arestas, antes incidentes a este vértice, sejam incidentes agora ao vértice $i-1$ também do perfil j; seja δ' a atribuição não degenerada para Γ' obtida a partir de δ, alterando os vértices maiores do que $i-1$ no perfil j para as seguintes cotas $\delta'_j(i) = \delta_j(i+1), \delta'_j(i+1) = \delta_j(i+2) \ldots \delta'_j(v'_j - 1) = \delta'_j(v'_j)$, onde v'_j é o número de vértices no perfil j de Γ'. Pela construção de Γ' e δ', temos que o único gbloco induzido (Γ, δ) é igual ao único gbloco induzido por (Γ', δ') e isto implica, pela Proposição 2.5, que Γ' é similar a Γ. Como Γ' tem a mesma dimensão, o mesmo número de arestas, mas tem um vértice a menos que Γ, então $\Gamma' < \Gamma$ ∎

Proposição 2.10 *Um simin que induz um único gbloco não tem vértices móveis.*

Prova: Seja Γ um gtet qualquer que induz um único gbloco. Iterando a Proposição 2.9 a partir de Γ chegamos num gtet Γ' similar sem nenhum vértice móvel. Como $S(\Gamma)$ é similar a Γ, então, por transitividade, $\mathcal{S}(\Gamma)$ também é similar a Γ'. Logo $\mathcal{S}(\Gamma) \leq \Gamma'$. Como a introdução de vértices móveis em Γ' gera similares maiores que Γ', então $\mathcal{S}(\Gamma)$ não tem vértices móveis. Como argumentamos de maneira geral (um Γ qualquer), podemos concluir que todo simin que induz um único gbloco não tem vértices móveis. ∎

gtet $\mathbf{B^n}$. Denotamos por $\mathbf{B^n}$ o gtet de dimensão n cujos perfis têm dois vértices e uma única aresta de rótulo 1 ligando estes dois vértices. Este gtet induz um único gbloco que é também um bloco de dimensão n.

gtet $\mathbf{B^n_i}$. Denotamos por $\mathbf{B^n_i}$ o gtet de dimensão n cujos perfis, exceto pelo i-ésimo perfil, tem dois vértices e duas arestas de rótulos 1 e 2 ligando estes dois vértices. O i-ésimo perfil de $\mathbf{B^n_i}$, por sua vez, tem três vértices e duas arestas. A aresta de rótulo 1 liga o vértice 0 ao vértice 1 e a aresta de rótulo 2 liga o vértice 1 ao vértice 2. Além disto, $f(1) = 1$ e $f(2) = 2$. Este gtet induz dois gblocos que são também blocos de dimensão n. Intuitivamente, o gtet $\mathbf{B^n_i}$ representa um bloco cortado através de um hiperplano perpendicular ao eixo i resultando em dois blocos menores.

esquema, peças e decomposições. Um *esquema* $\mathcal{E}$ é definido por uma partição de eixos $p(\mathcal{E})$ e uma função que associa a cada elemento de seu domínio denotado por $\mathcal{P}ec(\mathcal{E}) \subseteq GTet$ um conjunto de gtets em 2^{GTet} (ver nota de rodapé [4]). Cada gtet em $\mathcal{P}ec(\mathcal{E})$ é chamado de *peça* do esquema $\mathcal{E}$. A imagem da função definida no esquema $\mathcal{E}$ para a peça Γ é denotada por $\mathcal{E}(\Gamma)$. Cada elemento de $\mathcal{E}(\Gamma)$ é chamado de *decomposição* de Γ. Um esquema satisfaz também:

(*i*) Se Γ é uma peça de $\mathcal{E}$, $\Gamma \in \mathcal{P}ec(\mathcal{E})$, então Γ é um $p(\mathcal{E})$-representante e $k(\Gamma) = 1$.

(*ii*) Se $\Gamma \in \mathcal{P}ec(\mathcal{E})$ e $\Gamma' \in \mathcal{E}(\Gamma)$, então Γ' tem pelo menos 2 filhos ($k(\Gamma') \geq 2$); $\Gamma' = \mathcal{S}(\Gamma')$; $\Gamma = \mathcal{S}(pai(\Gamma'))$; e, para $\ell = 1, \ldots, k(\Gamma')$, $\mathcal{R}'(filho(\Gamma', \ell), p(\mathcal{E}))$ está em $\mathcal{P}ec(\mathcal{E})$.

(*iii*) $\mathbf{B^n}$ pertence a $\mathcal{P}ec(\mathcal{E})$ e $\mathbf{B^n_1}, \ldots, \mathbf{B^n_n}$ estão todos em $\mathcal{E}(\mathbf{B^n})$.

(*iv*) Se $\Gamma \in \mathcal{P}ec(\mathcal{E})$ e $\Gamma \neq \mathbf{B^n}$, então existe pelo menos uma decomposição $\Gamma' \in \mathcal{E}(\Gamma)$ que não tem vértices móveis.

[4] Em geral, se A é um conjunto denotamos por 2^A o conjunto das partes de A, isto é, o conjunto formado por todos os subconjuntos de A.

Abaixo seguem algumas conseqüências decorrentes da definição de esquema.

- Se Γ é uma peça de $\mathcal{E}$, então ela não tem vértice móvel, pois Γ é um simin (lemas 2.8 e 2.7) e todo simin que induz um único gbloco não tem vértice móvel (Proposição 2.10).
- Toda atribuição para uma peça Γ diferente de um bloco $\mathbf{B^n}$ é também uma atribuição para a decomposição sem vértices móveis que definimos em (iv).
- Toda decomposição é um simin.

Estas condições sobre os esquemas servem para garantir que não importa quão complicadas sejam as peças de um esquema é sempre possível quebrá-las até o ponto em que só haja blocos. E, além disto, um bloco pode ser guilhotinado em qualquer um dos eixos em dois blocos menores.

Se uma decomposição tem k filhos, dizemos se tratar de uma *k-decomposição*. A seguir, apresentamos um esquema bidimensional. As peças deste esquema são duas: a peça R e a peça L. No total são 20 2-decomposições neste esquema sendo 5 para a peça R e 15 para a peça L. Observe que as peças R e L induzem gblocos que são invariantes pela permutação que troca os eixos x_1 e x_2.[5] Assim, para este esquema, tanto faz tomar $p(\mathcal{E}) = \{\{1\},\{2\}\}$ ou $p(\mathcal{E}) = \{\{1,2\}\}$. Este esquema é a base para a heurística RL que mencionamos no capítulo 1. Os detalhes deste esquema são descritos no capítulo 3 e no anexo A.

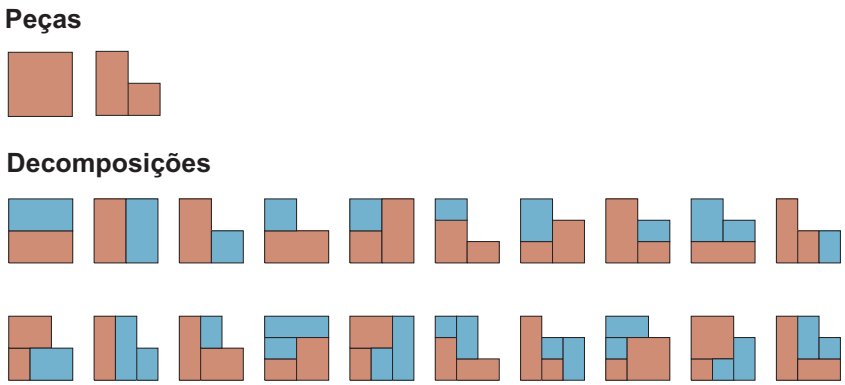


Figura 2.7: O esquema da heurística RL

2.9 Complexidade de gtets e de esquemas

Nesta breve seção, definimos uma noção de complexidade para gtets e para esquemas de decomposições. A complexidade de um gtet está relacionada com o número de possibilidades de atribuições para o similar mínimo daquele gtet da seguinte forma: suponha que existam $m+1$ cotas possíveis em cada uma das n dimensões e que a primeira cota seja sempre 0. Se um gtet $\mathcal{S}(\Gamma)$ tem $v_1, v_2, \ldots, v_n$ vértices em cada dimensão, então o número de atribuições diferentes para $\mathcal{S}(\Gamma)$ onde o primeiro vértice de cada perfil é zero é limitado superiormente por $m^{(v_1-1)+(v_2-1)+\ldots+(v_n-1)}$ (lembre-se de que, por definição, $v_i > 0$ para

[5] Geometricamente esta permutação é realizada por uma reflexão ao longo da reta $\{(x_1,x_2) \mid x_2 = x_1\}$

$1 \leq i \leq n$). O expoente $(v_1 - 1) + (v_2 - 1) + \ldots + (v_n - 1)$, que é totalmente definido a partir de um gtet, é a complexidade que usamos.

complexidade de um gtet. Seja Γ um gtet e $\mathcal{S}(\Gamma)$ seu simin. Se o número de vértices em cada um dos n perfis de $\mathcal{S}(\Gamma)$ é $v_1, v_2, \ldots, v_n$, então denotamos por $Complex(\Gamma) = (v_1 - 1) + (v_2 - 1) + \ldots + (v_n - 1)$ a *complexidade* de Γ.

complexidade de um esquema. Sejam $\Gamma_1, \Gamma_2, \ldots, \Gamma_m$ as decomposições de um esquema $\mathcal{E}$. Dizemos que $Complex(\mathcal{E}) = \max\{Complex(\Gamma_1), Complex(\Gamma_2), \ldots, Complex(\Gamma_n)\}$ é a *complexidade do esquema* $\mathcal{E}$.

complexidade de peça de um esquema. Sejam $\mathcal{E}$ um esquema e $\mathcal{P}ec(\mathcal{E})$ suas peças. Dizemos que $ComplexP(\mathcal{E}) = \max\{Complex(\Gamma) \mid \Gamma \in \mathcal{P}ec(\mathcal{E})\}$ é a *complexidade de peça do esquema* $\mathcal{E}$. Ou seja, $ComplexP(\mathcal{E})$ é a complexidade da peça mais complexa de $\mathcal{E}$.

2.10 Realizando empacotamentos: a função o

O problema empacotar o maior volume de blocos-caixa dentro de um gbloco não é simples. Entretanto, se restringirmos este problema de tal maneira que os blocos-caixa tenham tamanhos iguais (todos representando uma mesma caixa) e o gbloco onde queremos empacotar seja um bloco, então a questão se torna bem mais simples. De fato o número máximo de caixas que podemos empacotar nestas condições é dado por $\prod_{i=1}^{n} \lfloor \frac{b_i}{c_i} \rfloor$, onde $c = (c_1, \ldots, c_n)$ é a única caixa e $b = (b_1, \ldots, b_n)$ é o tamanho do bloco. Como vemos mais na frente, este problema restrito, e de solução trivial, é o caso base para encontrarmos os empacotamentos no caso geral.

melhor ocupação homogênea (moh). Sejam c uma caixa e b o tamanho de um bloco ambos n dimensionais. Denotamos por $moh(c, b)$ o volume dado pela expressão $vol(c) \cdot (\prod_{i=1}^{n} \lfloor \frac{b_i}{c_i} \rfloor)$.

A melhor ocupação homogênea é o volume ocupado pelo empacotamento de maior ocupância da caixa c dentro de um bloco de tamanho b. O nome "homogênea" vem do fato de que neste melhor empacotamento todos os blocos-caixas representam a mesma caixa.

empacotamento de caixas num dgtet. Sejam Y um conjunto de caixas e (Γ, δ) um dgtet que induz os gblocos $K_1, K_2, \ldots, K_k$. Um *empacotamento* $\mathcal{P}(Y, \Gamma, \delta)$ *de caixas de* Y *em* (Γ, δ) é uma união de empacotamentos de Y nos gblocos induzidos por (Γ, δ), ou seja, $\mathcal{P}(Y, \Gamma, \delta) = \mathcal{P}(Y, K_1) \cup \mathcal{P}(Y, K_2) \cup \ldots \cup \mathcal{P}(Y, K_k)$, onde, para $1 \leq \ell \leq k$, $\mathcal{P}(Y, K_\ell)$ é um empacotamento de Y em K_ℓ como definido anteriormente. Note que, por esta definição, cada bloco-caixa em $\mathcal{P}(Y, \Gamma, \delta)$ está completamente contido num único gbloco.

atribuições para compatibilizar um gtet com um dgtet. Sejam Γ_1 e Γ_2 dois gtets similares. Seja δ_2 uma atribuição para Γ_2 não degenerada. Denotamos por $compativeis(\Gamma_1, \Gamma_2, \delta_2)$ o conjunto $\{\delta_1 \mid gblocos(\Gamma_1, \delta_1) = gblocos(\Gamma_2\ \delta_2)$ e δ_1 é não degenerada$\}$. Se δ_2 é uma atribuição inteira não negativa, denotamos por $icompativeis(\Gamma_1, \Gamma_2, \delta_2)$ o conjunto $\{\delta_1 \mid gblocos(\Gamma_1, \delta_1) = gblocos(\Gamma_2\, delta_2)$ e δ_1 é uma atribuição inteira não-negativa e não-degenerada$\}$.

normalizando um dgtet. Denotamos por $norm(\Gamma, \delta, p)$ o par (Γ', δ') tal que $\Gamma' = \mathcal{R}'(\Gamma, p)$ e $\delta' = rev'(perm'(\delta, \pi), r)$, onde $\pi = \mathcal{R}'_{perm}(\Gamma, p)$ e $r = \mathcal{R}_{rev}(perm(\Gamma, \pi))$.

função o de ocupância máxima respeitando um esquema. Sejam $\mathcal{E}$ um esquema, Γ uma peça de $\mathcal{E}$ ($\Gamma \in \mathcal{P}ec(\mathcal{E})$), δ uma atribuição não degenerada para Γ com valores inteiros

não negatvos e Y um conjunto de caixas, onde $p(Y) = p(\mathcal{E})$. Seja $K = gbloco(\Gamma, \delta, 1)$. A função o é então definida como:

$$o(\mathcal{E}, \Gamma, \delta, Y) = \begin{cases} 0, \text{ se } K \text{ é um bloco e } vol(K) < \min\{vol(y) \mid y \in Y\}. \\ vol(K), \text{ se } K \text{ é um bloco e } vol(K) = \max_{y \in Y}\{moh(y, tam(K))\}. \\ \text{Caso contrário,} \\ \max\{ \sum_{\ell=1}^{k(\Gamma')} o(\mathcal{E}, \Gamma'_\ell, \delta'_\ell, Y) \mid \Gamma' \in \mathcal{E}(\Gamma), \\ \qquad\qquad \delta' \in icompativeis(pai(\Gamma'), \Gamma, \delta), \\ \qquad\qquad (\Gamma'_\ell, \delta'_\ell) = norm(filho(\Gamma', \ell), \delta', p(Y)) \ \}. \end{cases}$$

Na definição acima estão implícitos detalhes que vale a pena explicitar.

- Dado que Γ' é uma decomposição da peça Γ definida em $\mathcal{E}$ e que a atribuição δ para a peça Γ é inteira não negativa, sabemos que $pai(\Gamma')$ é similar a Γ (pela definição de decomposição de um esquema) e, portanto, $icompativeis(pai(\Gamma'), \Gamma, \delta)$ é bem definido.
- Pela Proposição 2.1 sabemos que δ' é uma atribuição tanto para $pai(\Gamma')$ como para Γ'. Do mesmo modo, pela Proposição 2.2, sabemos que δ' é também uma atribuição para os filhos de Γ'.
- Pela definição de *icompativeis* sabemos que δ' é inteira não negativa e é não degenerada. Se para uma determinada decomposição Γ' o conjunto $icompativeis(pai(\Gamma'), \Gamma, \delta)$ é vazio, a decomposição Γ' não é considerada.
- Pela definição de *norm* sabemos que, para $\ell = 1, 2, \ldots, k(\Gamma')$, Γ'_ℓ é uma peça em $\mathcal{E}$ e que δ'_ℓ é uma atribuição não degenerada para Γ'_ℓ.

Existem dois casos bases que na definição da função recorrente o. No primeiro caso base, estamos diante de uma peça (que é um bloco) cujo volume é menor do que o volume da menor caixa em Y. Neste caso, avaliamos a maior ocupação desta peça como sendo 0, pois nenhuma caixa pode ser empacotada nesta peça. No segundo caso base, a peça (que é um bloco) que estamos avaliando tem o tamanho exato de uma caixa em Y, portanto a maior ocupação desta peça é o próprio volume da peça. O caso remanescente é o caso recorrente, e nele avaliamos a maior ocupação da peça em questão como sendo a maior ocupação de um empacotamento de Y numa decomposição desta, contida no esquema $\mathcal{E}$. O fato de o é bem definida é mostrado na proposição que enunciamos a seguir.

Proposição 2.11 *A função o é bem definida.*

Prova: Se alguma condição de um dos dois casos base é satisfeita, então o está definida. Nos resta mostrar que, se nenhum dos casos bases pode ser aplicado, então o caso recorrente pode ser aplicado reduzindo o volume do gbloco corrente. Se a peça não for um bloco, pela definição de esquema, existe pelo menos uma decomposição Γ' para esta peça onde qualquer atribuição δ de Γ é também uma atribuição para a decomposição Γ' (não há vértices móveis na decomposição) e a fórmula recorrente é aplicável: δ é o único elemento de $icompativeis(pai(\Gamma'), \Gamma, \delta)$. Já se a peça for um bloco, então é possível, no mínimo, decompor o bloco em dois blocos menores usando pelo menos uma das decomposições

$\mathbf{B_1^n}, \mathbf{B_2^n}, \ldots, \mathbf{B_n^n}$. Como é sempre possível aplicar uma das regras, e como a cada decomposição o volume das peças diminui, todos os caminhos da recursão chegam a um dos casos base. ∎

Algumas características importantes de o são sua generalidade, pois está definida para qualquer dimensão maior que 0, e a sua flexibilidade, pois os esquemas, que são responsáveis pelo poder/eficiência da avaliação, são parâmetros de entrada para o. Ou seja, se queremos uma avaliação mais eficiente computacionalmente e menos poderosa, entramos com esquemas mais simples (menos decomposições e poucos vértices móveis), caso contrário, se estamos interessados em empacotamentos melhores, utilizamos esquemas mais elaborados. Os métodos utilizados em [4], [2], [1] podem todos ser obtidos através da função o, dado o esquema adequado.

2.11 Como implementar a função o?

A técnica para o desenvolvimento de algoritmos conhecida como *programação dinâmica* é a chave desta pergunta. O domínio da programação dinâmica são problemas que podem ser divididos em subproblemas menores e cuja solução pode ser encontrada combinando a solução destes subproblemas. Além disto, na divisão em subproblemas, um mesmo subproblema pode aparecer diversas vezes. Estes dois ingredientes básicos da programação dinâmica são chamados de *optimal substructure* (subestrutura ótima) e *overlapping subproblems* (sobreposição de subproblemas) [10]. A função o caracteriza perfeitamente um problema no domínio da programação dinâmica. A solução para um gbloco maior é encontrada combinando soluções de gblocos menores obtidos através das decomposições do esquema (*optimal substructure*). E um mesmo gbloco menor pode aparecer como subproblema de vários gblocos maiores (*overlapping subproblems*).

O primeiro passo para um algoritmo que obedece os princípios da programação dinâmica é definr uma tabela para armazenar soluções de todos os problemas "importantes". Normalmente, os problemas "importantes" são todos os subproblemas possíveis do problema principal que queremos resolver. Esta tabela deve ser eficientemente indexada pelo problema, de modo que seja fácil, dado um problema, verficar se já há uma solução para o mesmo. O segundo passo é preencher a tabela. A ordem de preencher a tabela deve ser dos problemas menores para os problemas maiores (de baixo para cima ou *bottom-up*). Esta ordem garante que todos os subproblemas do problema corrente já estão resolvidos. Ao final do preenchimento da tabela o problema principal, o último, estará resolvido.

A aplicação da programação dinâmica para calcular $o(\mathcal{E}, \Gamma_0, \delta_0, Y)$ é direta. Os problemas que a tabela deve indexar são todos os dgtets (Γ, δ), tais que Γ é uma peça de $\mathcal{E}$ e (Γ, δ) induz um gbloco que está contido no gbloco induzido por (Γ_0, δ_0). Os dgtets devem ser avaliados por ordem crescente de volume dos gblocos induzidos. Os dgtets diferentes que induzem gblocos de volumes iguais são avaliados em qualquer ordem, pois nenhum deles contém o outro. Desta forma, a regra de que todos os subproblemas do problema corrente já estão avaliados (ou tabelados) é obedecida.

Existe uma variação para a programação dinâmica que também pode ser aplicada para implementar a função o. Esta técnica é chamada de *memoization* (memorização). A mesma tabela da programação dinâmica é mantida. Porém, o controle que define a ordem do preenchimento da tabela é feito através de um algoritmo recursivo, iniciando no problema principal (o maior problema). À medida que um subproblema é necessário para

resolver um problema maior, a tabela é consultada. Caso ainda não haja solução para o subproblema, o algoritmo é recursivamente chamado para resolver aquele subproblema. Ao fim da avaliação de cada problema (ou subproblema) a solução é armazenada na tabela para que nunca mais aquela solução seja recalculada. Esta estratégia de preenchimento da tabela é chamada de *top-down* (de cima para baixo).

Além de uma técnica para projetar algoritmos, programação dinâmica também nos ajuda a analisar o custo computacional. Um algoritmo baseado em programação dinâmica tem custo computacional proporcional ao tamanho da tabela e ao tempo para dividir e combinar problemas. Mostramos, a seguir, um limitante superior para o custo computacional da implementação baseada em programação dinâmica para avaliar $o(\mathcal{E}, \Gamma_0, \delta_0, Y)$. Seja n a maior cota que aparece na atribuição δ_0 do gtet Γ_0. Seja $p = |\mathcal{P}ec(\mathcal{E})|$ o número de peças do esquema $\mathcal{E}$. Sejam $\Gamma_1, \Gamma_2, \ldots, \Gamma_p$ as peças de $\mathcal{E}$. Seja $x_i = Complex(\Gamma_i)$, para $1 \leq i \leq p$. Seja $d_i = |\mathcal{E}(\Gamma_i)|$, para $1 \leq i \leq p$, o número de decomposições de cada peça. E, por fim, seja m_i o maior número de vértices móveis para uma decomposição da peça Γ_i, para $1 \leq i \leq p$. A tabela tem no máximo $n^{x_1} + n^{x_2} + \ldots + n^{x_p}$ entradas. Um limite superior para o número de operações para divisão e composição de subproblemas na avaliação da peça Γ_i é dado por $d_i \cdot n^{m_i}$. Compondo tudo temos a seguinte expressão:

$$n^{x_1} \cdot d_1 \cdot n^{m_1} + n^{x_2} \cdot d_2 \cdot n^{m_2} + \ldots + n^{x_p} \cdot d_p \cdot n^{m_p} =$$

$$d_1 \cdot n^{x_1+m_1} + d_2 \cdot n^{x_2+m_2} + d_p \cdot n^{x_p+m_p}$$

Observe que $Complex(\mathcal{E}) = \max_{i=1}^{p}\{x_i + m_i\}$ e que $d_1, d_2, \ldots, d_p$ são constantes para um mesmo esquema. Usando a definição de $\mathcal{O}$, limitante superior assintótico [10], podemos finalmente concluir que

$$\mathcal{O}(n^{Complex(\mathcal{E})})$$

é um limitante superior para avaliar a função o numa peça do esquema $\mathcal{E}$ com uma atribuição de cota máxima igual a n. Em termos de espaço, é fácil ver que a tabela que usamos tem tamanho $\mathcal{O}(n^{ComplexP(\mathcal{E})})$.

Uma maneira de acelerar a etapa de divisão/composição de um problema é utilizar limitantes superiores para as soluções. Suponha que temos que resolver um problema e sabemos, ou podemos calcular rapidamente, que sua solução é no máximo α. Se em algum ponto na divisão/composição daquele problema encontrarmos uma solução de valor α não é mais preciso continuar naquele problema. Já encontramos uma solução ótima. No caso da função o, o primeiro limitante que temos é o próprio volume do gbloco corrente. Se no meio da etapa de divisão/composição daquele gbloco encontrarmos uma solução que ocupe todo seu volume, estamos diante de uma solução ótima e, portanto, o problema já está resolvido. O fato é que o limitante do volume não é muito eficaz, muitas vezes nenhuma solução pode ocupar todo o volume do gbloco. Propomos, na seção 2.13, um limitante superior baseado na própria função o.

A seguir, definimos uma nova função o' equivalente à função o em esquemas que satisfazem restrições adicionais. Porém, a função o' induz um algoritmo mais eficaz. As idéias de implementação e de limitante superior para o custo computacional elaboradas nesta seção continuam valendo para a função o'.

2.12 A função o'

Já temos em mãos a função o que é bem definida (proposição 2.11) e uma técnica adequada para implementar esta função (seção anterior), cujo algoritmo resultante tem limites

assintóticos polinomiais de tempo e espaço computacionais. Ou seja, neste ponto, conhecemos uma maneira correta de implementar nossa abordagem de empacotamentos e sabemos que seus limites não são explosivos computacionalmente (não-exponencial). Apesar disso, na prática, o que temos ainda pode ser muito caro e, portanto, qualquer melhoria é importante. Esta seção tem por objetivo apresentar a função o' cuja implementação é computacionalmente mais barata do que a implementação para a função o, embora atue sobre um domínio menor, os *esquemas de grade*. A idéia por trás desta melhoria não é original e vem dos conjuntos normais (*normal sets*) propostos por Beasley [13] e refinados por Scheithauer e Terno [5] para os conjuntos de pontos de varredura (*raster points*).

grade. Uma *grade de dimensão n* é um vetor X de seqüências de números naturais. Isto é, $X = (X_1, X_2, \ldots, X_n)$, onde X_i, para $1 \leq i \leq n$, é uma seqüência (infinita) de números não negativos crescentes. Um ponto $x = (x_1, x_2, \ldots, x_n)$ *pertence à grade* X, $x \in X$, se e somente se $x_i \in X_i$, para $1 \leq i \leq n$.

grade induzida por um conjunto de caixas. Seja Y um conjunto de m caixas de dimensão n. Seja $y_j = (y_{j1}, y_{j2}, \ldots, y_{jn})$ a j-ésima caixa de Y. Denotamos por $X_i(Y)$ a seqüência infinita das combinações lineares inteiras não negativas das i-ésimas coordenadas das caixas de Y, ordenadas de maneira crescente. A *grade* $X(Y)$ é $(X_1(Y), X_2(Y), \ldots, X_n(Y))$.

empacotamento alinhado a uma grade. Um empacotamento $\mathcal{P}(Y, K)$ é dito $X(Y)$-*alinhado* se cada um dos seus blocos tem todos os seus vértices na grade $X(Y)$. Do contrário, ele é dito *não* $X(Y)$-*alinhado*.

Proposição 2.12 *A partir de um empacotamento $\mathcal{P}(Y, K)$ de caixas de Y num gbloco pconvexo K não $X(Y)$-alinhado pode ser obtido outro empacotamento $\mathcal{P}'(Y, K)$ que é $X(Y)$-alinhado e com igual ocupância.*

Prova: Seja $\omega(\mathcal{P}(Y, K))$ o número total de coordenadas inferiores de blocos-caixa em $\mathcal{P}(Y, K)$ que estão fora de $X(Y)$. Assim, um bloco-caixa B em $\mathcal{P}(Y, K)$ contribui com um número inteiro entre 0 e um máximo de n para $\omega(\mathcal{P}(Y, K))$. Se $\omega(\mathcal{P}(Y, K)) = 0$, então $\mathcal{P}(Y, K)$ é $X(Y)$-alinhado. Do contrário, existe um bloco B e um eixo j tal que o j-ésimo limite inferior de B, x, não está em $X_j = (z_1 < z_2 < \ldots < z_u < \ldots)$. Tome um tal B tal que x seja mínimo e suponha que $z_h < x < z_{h+1}$. Afirmamos que não existe bloco $\mathcal{B}'$ de $\mathcal{P}(Y, K)$ cujo j-ésimo limite superior está em (z_h, x). Como a j-ésima dimensão d_j de $\mathcal{B}'$ é um dos geradores de X_j, se houvesse tal $\mathcal{B}'$, o seu j-ésimo limite inferior, $x - d_j$, não estaria em X_j e seria menor do que x, contradizendo a escolha de B. Desse modo, não existe bloco de $\mathcal{P}(Y, K)$ obstruindo o deslocamento de B definido pela translação paralela ao j-ésimo eixo até que seu j-ésimo limite inferior se torne z_h. Além disso, toda a região varrida por esta translação está contida em K, pois este é pconvexo. Seja $\mathcal{P}'(Y, K)$ o novo empacotamento com esta translação. Observe que $\omega(\mathcal{P}'(Y, K)) = \omega(\mathcal{P}(Y, K)) - 1$. Deixe $\mathcal{P}'(Y, K)$ assumir o papel de $\mathcal{P}(Y, K)$ e itere até que $\omega(\mathcal{P}(Y, K)) = 0$. Como as operações de translação não alteram a ocupância, a proposição está demonstrada. ∎

atribuição truncada módulo uma grade. Sejam δ uma atribuição e X uma grade. Denotamos por $trunc(\delta, X)$ a atribuição $\delta' = (\delta'_1, \delta'_2, \ldots, \delta'_n)$ obtida de $X = (X_1, X_2, \ldots, X_n)$ onde $\delta'_i(j)$ é definida como $\delta'_i(j) = \max\{z \in X_i \mid z \leq \delta_i(j)\}$. Observe que δ' pode ser degenerada mesmo que δ não o seja.

fatia de um gbloco em $x_i = x$. Seja K um gbloco. A *fatia de K na cota a do i-ésimo eixo*, denotada por K^i_a, é a interseção de K com o hiperplano $\{x_i = a\}$.

Proposição 2.13 *Seja K um gbloco posicionado de forma a ser pconvexo. Sejam $a < b$ cotas no i-ésimo eixo. Então, $K_b^i - (0, \ldots, b-a, \ldots, 0) \subseteq K_a^i$.*

Prova: Esta proposição é uma conseqüência direta do fato de que K é pconvexo. Intuitivamente, quando a cota cresce, as i-fatias se tornam menores no sentido de que transladadas paralelamente ao i-ésimo eixo de forma apropriada são subconjuntos de i-fatias com cotas menores. ■

A Figura abaixo exemplifica o resultado acima. As i-ésimas fatias de um gbloco 3-dimensional (engrossadas) são como escadas justapostas encaixantes.

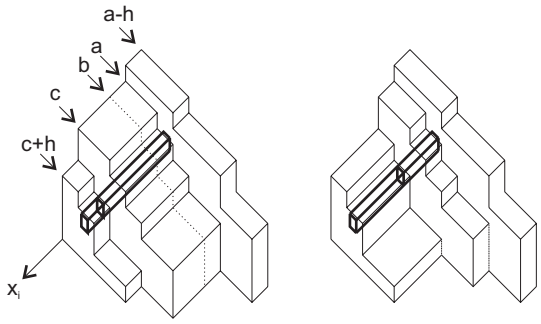


Figura 2.8: Para prova da Proposição 2.13: Fatias de K nas i-cotas crescentes $a-h, a, b, c, c+h$

Proposição 2.14 *Seja (Γ, δ) um dgtet induzindo um único gbloco K posicionado de forma a ser pconvexo. Seja i um perfil de Γ em que exista uma menor i-cota errada $c = \delta_i(j) \notin X_i(Y)$. Seja $b = \max\{z \in X_i \mid z < c\}$. Considere δ' definida por $\delta'_i(j) = b$ e $\delta'_h(\ell) = \delta_h(\ell)$ para $(h, \ell) \neq (i, j)$. Então, os mesmos blocos-caixa de um empacotamento $\mathcal{P}(Y, \Gamma, \delta)$ $X(Y)$-alinhado constituem um empacotamento $\mathcal{P}(Y, \Gamma, \delta')$ no único gbloco $K' \subseteq K$ que é induzido por (Γ, δ').*

Prova: Existe uma i-cota menor do que c que está em X_i, uma vez que o zero está na grade. Seja a a maior de tais cotas. Note que $a \leq b$. A figura acima ilustra a idéia da prova da proposição em dimensão 3. Note a relação entre K à esquerda e K' à direita: $K' \subseteq K$. Quando $a = b$, δ' é degenerada. Se as fatias em c e em $c+h$ são congruentes por uma i-translação, então $K' = K$. Para estabelecer a proposição é suficiente provar que qualquer bloco-caixa de $\mathcal{P}(Y, \Gamma, \delta)$ está contido em K'. Com efeito, precisamos apenas nos preocupar com os blocos-caixa $\mathcal{B}$ com interseção não vazia com K_c^i. Como a cota do i-ésimo limite inferior do bloco $\mathcal{B}$ está em X_i, a cota do seu i-ésimo limite superior é estritamente maior do que c. Assim existe $h > 0$ tal que $\mathcal{B} \cup K_{c+h}^i \neq \emptyset$. Note que $\mathcal{B}$ trespassa o hiperplano $\{x_i = c\}$ ortogonalmente, pois os lados de $\mathcal{B}$ são paralelos aos eixos coordenados. Pela Proposição 2.13, podemos inferir que $\mathcal{B} \subseteq K'$, provando a proposição. ■

Proposição 2.15 *Seja (Γ, δ) um dgtet induzindo um único gbloco K posicionado de forma a ser pconvexo. Seja $\mathcal{P}(Y, \Gamma, \delta)$ um empacotamento qualquer de caixas de Y. Existe um*

empacotamento $X(Y)$-alinhado, $\mathcal{P}(Y,\Gamma,\delta')$, onde $\delta' = trunc(\delta, X(Y))$ tal que $ocup(\mathcal{P}(Y,\Gamma,\delta')) = ocup(\mathcal{P}(Y,\Gamma,\delta))$.

Prova: Usando a Proposição 2.12 podemos supor que todos os blocos-caixa de $\mathcal{P}(Y,\Gamma,\delta)$ são $X(Y)$-alinhados. Usando a Proposição 2.14 obtemos o resultado. ∎

O resultado a seguir generaliza a proposição acima para gtets com dois filhos. Este resultado é a motivação para restringirmos nossos esquema a esquemas onde $|\mathcal{E}(\Gamma)| = 2$. Para três ou mais filhos, o que acontece quando trocamos (Γ,δ) por (Γ,δ') é incerto.

Proposição 2.16 *Seja (Γ,δ) um dgtet induzindo dois gblocos K_1 e K_2 de forma a que pelo menos um seja pconvexo, e que não existam cotas negativas. Além disto, suponha que cada vértice móvel de Γ seja incidente a pelo menos uma aresta em cada um dos dois filhos de Γ: $filho(\Gamma,1)$ e $filho(\Gamma,2)$. Seja $\mathcal{P}(Y,\Gamma,\delta)$ um empacotamento qualquer de caixas de Y. A atribuição δ' para Γ onde as cotas dos vértices móveis de Γ são $X(Y)$-truncadas é tal que existe um empacotamento $\mathcal{P}(Y,\Gamma,\delta')$ satisfazendo $ocup(\mathcal{P}(Y,\Gamma,\delta')) = ocup(\mathcal{P}(Y,\Gamma,\delta))$.*

Prova: Digamos que seja K_1 o gbloco induzido por (Γ,δ) que é pconvexo. Usando a Proposição 2.12, sem perda de generalidade, podemos assumir que cada bloco-caixa do empacotamento $\mathcal{P}(Y,K_1) \subseteq \mathcal{P}(Y,\Gamma,\delta)$ é $X(Y)$-alinhado. Iterando aplicações da Proposição 2.14 podemos afirmar que cada bloco-caixa de $\mathcal{P}(Y,K_1)$ está contido em K_1', o gbloco de (Γ,δ') que está contido em K_1. Denote por $\mathcal{P}(Y,K_1')$ o empacotamento que tem os mesmos blocos-caixa que $\mathcal{P}(Y,K_1)$. Como só alteramos cotas de vértices móveis, $K_2' = K_1 \cup K_2 - K_1' \supseteq K_2$. Logo, como K_2' contém K_2, o empacotamento $\mathcal{P}(Y,K_2) \subseteq \mathcal{P}(Y,\Gamma,\delta)$ é também um empacotamento para K_2'. Denote por $\mathcal{P}(Y,K_2')$ o empacotamento que tem os mesmos blocos-caixa que $\mathcal{P}(Y,K_2)$. Como $\mathcal{P}(Y,\Gamma,\delta') = \mathcal{P}(Y,K_1') \cup \mathcal{P}(Y,K_2')$, a proposição fica estabelecido. ∎

Uma versão mais geral da Proposição 2.16 seria importante para mostrar que é possível utilizar apenas atribuições na grade para as abordagens R-em-5Rs e B-em-9Bs que são apresentadas respectivamente nos capítulos 3 e 4. Com os resultados que temos aqui, não é possível concluir que apenas atribuições truncadas na grade $X(Y)$ são suficientes para estas duas abordagens.

reversão que torna um filho pconvexo. Seja Γ um gtet e r uma reversão tal que $\Gamma' = rev(\Gamma, r)$ e existe um $i \in \{1,2,\ldots,k(\Gamma)\}$ tal que $gbloco(\Gamma', \delta'^{\star}, i)$ é pconvexo. Denotamos por $\rho(\Gamma)$ a reversão lexicograficamente mínima que tenha a propriedade anterior (qualquer reversão com a propriedade anterior serviria). Ou seja, $\rho(\Gamma)$ é uma reversão para Γ que faz com que um dos gblocos induzidos seja pconvexo. Note que nem sempre tal reversão existe.

esquema de grade. Um *esquema de grade* $\mathcal{E}$ é um esquema que satisfaz as seguintes condições adicionais para cada uma das suas decomposições Γ': (i) $k(\Gamma') = 2$; (ii) os vértices móveis de Γ' são incidentes a arestas nos dois filhos de Γ'; (iii) a reversão $\rho(\Gamma')$ existe. Denotamos por $GEsquema$ o conjunto de todos os possíveis esquemas de grade.

A definição de esquemas de grade é motivada pela Proposicao 2.16. Para isto, restringimos suas decomposições a terem apenas 2 filhos e uma reversão que faça com que um dos gblocos induzidos seja pconvexo. A restrição a esquemas de grade nos leva a uma função mais "eficiente" para a função o. Este ganho se dá pelo fato de usarmos apenas

vértices móveis na grade $X(Y)$ (função *gcompativeis*). A grade $X(Y)$ é mais esparsa do que uma grade usando todos os inteiros como na função o. O ganho de eficiência é obtido em troca da perda de generalidade dos esquemas de grade em relação aos esquemas gerais. Entretanto, neste trabalho, esta perda não representa um problema, pois os esquemas bi e tri-dimensionais (RL e BALST) que introduzimos nos próximos capítulos são esquemas de grade.

atribuições com vértices móveis restritos a uma grade X. Denotamos por *gcompativeis*$(\Gamma_1, \Gamma_2, \delta_2, X)$ o conjunto de todas as atribuições $\delta_1 = (\delta_{11}, \delta_{12}, \ldots, \delta_{1n})$ para Γ_1 tal que a condição $gblocos(\Gamma_1, \delta_1) = gblocos(\Gamma_2, \delta_2)$ é satisfeita. Além disto, se m é um vértice móvel de Γ_1, então $\delta_{1j}(m) \in X_j$ para todo perfil j de Γ_1.

A grade $\mathcal{X}(X, r, \Gamma, \delta)$. Sejam X uma grade, $r = (r_1, r_2, \ldots, r_n)$ uma reversão, Γ um gtet com v_j vértices no seu j-ésimo perfil ($1 \leq j \leq n$) e δ uma atribuição para Γ. A *grade* $\mathcal{X}(X, r, \Gamma, \delta)$ é grade finita $X' = (X'_1, X'_2, \ldots, X'_n)$ definida como segue. Se $r_j = 0$, então $X'_j = \{z \mid z \in X_j \text{ e } 0 \leq z \leq \delta_j(v_j - 1)\}$. Se $r_j = 1$, então $X'_j = \{\delta_j(v_j - 1) - z \mid z \in X_j \text{ e } 0 \leq z \leq \delta_j(v_j - 1)\}$.

função o' de ocupância máxima respeitando um esquema de grade. Sejam $\mathcal{E}$ um esquema de grade, Γ uma peça de $\mathcal{E}$ ($\Gamma \in \mathcal{P}ec(\mathcal{E})$), δ uma atribuição não degenerada para Γ com cotas na grade $X(Y)$, onde Y é um conjunto de caixas e $p(Y) = p(\mathcal{E})$. Seja $K = gbloco(\Gamma, \delta, 1)$. A função o' é, então, definida como:

$$
o'(\mathcal{E}, \Gamma, \delta, Y) = \begin{cases}
0, \quad \text{se } K \text{ é um bloco e } vol(K) < \min\{vol(y) \mid y \in Y\}. \\
vol(K), \quad \text{se } K \text{ é um bloco e } vol(K) = \max_{y \in Y}\{moh(y, tam(K))\}. \\
\text{Caso contrário,} \\
\max\{ \;\; o'(\mathcal{E}, \Gamma'_1, \delta'_1, Y) + o'(\mathcal{E}, \Gamma'_2, \delta'_2, Y) \;\; \mid \\
\qquad\qquad \Gamma' \in \mathcal{E}(\Gamma), \\
\qquad\qquad \delta' \in gcompativeis(pai(\Gamma'), \Gamma, \delta, X'), \\
\qquad\qquad\quad \text{onde } X' = \mathcal{X}(X(Y), \rho(\Gamma'), \Gamma, \delta), \\
\qquad\qquad (\Gamma'_\ell, \delta'_\ell) = norm(\Gamma^L_\ell, \delta^L_\ell, p(Y)), \\
\qquad\qquad\quad \text{onde } (\Gamma^L_\ell, \delta^L_\ell) = limp(\Gamma^N_\ell, \delta^T_\ell), \\
\qquad\qquad\quad \delta^T_\ell = trunc(\delta^N_\ell, X(Y)), \\
\qquad\qquad\quad (\Gamma^N_\ell, \delta^N_\ell) = norm(filho(\Gamma', \ell), \delta', p(Y)), \\
\qquad\qquad\quad \text{para } \ell = 1, 2 \quad \}.
\end{cases}
$$

A diferença de o para o' está no caso recorrente. Os vértices móveis[6] das decomposições de um esquema de grade só precisam assumir cotas na grade específica X'. Os dgtets normalizados (Γ^N_1, δ^N_1) e (Γ^N_2, δ^N_2), filhos da decomposição (Γ', δ'), não têm necessariamente todas as cotas na grade $X(Y)$. O processo da definição das cotas de δ^N_1 e δ^N_2 parte de cotas na grade X', que não precisa estar contida na grade $X(Y)$, e, além disto, a normalização envolve operações de diferença que potencialmente geram pontos fora de uma grade original. Portanto, pela definição de o', as atribuições δ^N_1 e δ^N_2 precisam ser encaixadas na grade $X(Y)$ antes da avaliação recorrente. Este encaixe na grade $X(Y)$ ($\delta^T_1 = trunc(\delta^N_1, X(Y))$ e $\delta^T_2 = trunc(\delta^N_2, X(Y))$) pode resultar numa atribuição degenerada. É necessário então, antes da chamada recorrente, fazer a limpeza ($(\Gamma^L_1, \delta^L_1) = limp(\Gamma^N_1, \delta^T_1)$ e $(\Gamma^L_2, \delta^L_2) = limp(\Gamma^N_2, \delta^T_2)$) e posterior normalização ($(\Gamma'_1, \delta'_1) = norm(\Gamma^L_1, \delta^L_1, p(Y))$ e $(\Gamma'_2, \delta'_2) = norm(\Gamma^L_2, \delta^L_2, p(Y))$) antes da recorrência.

[6] Os vértices fixos por definição já têm suas cotas na grade $X(Y)$.

Antes de terminar esta seção, é importante ressaltar que os *esquemas de grade*, como definimos aqui, embora nos tenha sido útil, parecem artificiais. De fato o verdadeiro sentido do termo *esquema de grade* que tínhamos em mente, apesar da definição específica que demos, era a de qualquer esquema cuja avaliação utilizando as grades (função o') tem sempre o mesmo valor da avaliação pela função o que não utiliza as grades. Acreditamos, embora ainda não tenhamos provado, que qualquer esquema geral, como definimos na seção 2.8 é um esquema de grade neste sentido amplo.

2.13 Limitante superior para a função o

Seja $\mathcal{P}_3$ um empacotamento de blocos-caixa num paralelepípedo (dimensão 3). Seja $\mathcal{P}_2$ uma fatia em dimensão 2 de $\mathcal{P}_3$. Esta fatia é um retângulo com retângulos menores empacotados. O fato de $\mathcal{P}_2$ ser um empacotamento de retângulos pode parecer óbvio, mas induz um resultado geral interessante: através da ocupação máxima de fatias de um gbloco K é possível encontrar um limitante superior para a ocupação de K. A seguir, usando esta idéia, vamos formalizar o limitante superior $o^\star$ para a função o.

contração do j-ésimo perfil de um gtet ao nível i. Dado um gtet Γ de dimensão pelo menos 2, a *contração do j-ésimo perfil de Γ ao nível i* é o gtet de dimensão $n-1$, denotado por $\Gamma^i_{\overline{j}}$, obtido pela remoção do j-ésimo perfil do gtet $limp(\Gamma, \delta^{ji})$ onde $\delta^{ji}_\ell = \delta^\star_\ell$ para $\ell \neq j$, $\delta^{ji}_j(i') = 0$ para $i' < i$ e $\delta^{ji}_j(i') = 1$, para $i' \geq i$.

contração do j-ésimo perfil de um esquema. Dado um esquema n-dimensional ($n \geq 2$) $\mathcal{E}$, a *contração do j-ésimo perfil de $\mathcal{E}$* é o esquema $(n-1)$-dimensional, denotado por $\mathcal{E}_{\overline{j}}$, em que (1) decomposições de $\mathcal{E}_{\overline{j}}$ são as contrações do j-ésimo perfil de cada uma das decomposições de $\mathcal{E}$ a todos os possíveis níveis i; (2) a partição de eixos de $\mathcal{E}_{\overline{j}}$ é obtida da partição de $\mathcal{E}$ pela remoção do eixo j da parte em que ele aparece.

j-ésima contração de uma atribuição e de um conjunto de caixas. Dada uma atribuição n-dimensional ($n \geq 2$) δ, a *j-ésima contração de δ*, denotada por $\delta_{\overline{j}}$ é a atribuição $(n-1)$-dimensional obtida removendo-se a coordenada δ_j de δ. Analogamente, a *j-ésima contração de um conjunto de caixas n-dimensionais Y* é o conjunto de caixas $(n-1)$-dimensionais $Y_{\overline{j}}$ obtido removendo-se a j-ésima coordenada de cada uma das caixas em Y.

função limite superior $o^\star$. A função $o^\star : Esquema \times GTet_1 \times \Delta \times 2^{Caixa} \to N$ mapeia um esquema, um gtet com um único gbloco, uma atribuição com cotas inteiras para este gtet e um conjunto de blocos num número natural. Definimos

$$o^\star(\mathcal{E}, \Gamma, \delta, Y) = \min\{o_j(\mathcal{E}, \Gamma, \delta, Y) \mid 1 \leq j \leq n\},$$

onde $o_j(\mathcal{E}, \Gamma, \delta, Y) = \sum_{i=1}^{v_j - 1} o(\mathcal{E}_{\overline{j}}, \Gamma^i_{\overline{j}}, \delta_{\overline{j}}, Y_{\overline{j}})(\delta_j(i) - \delta_j(i-1))$.

A importância das contrações é que, permitindo a definição de $o^\star$, elas nos fornecem, como mostramos a seguir, um limitante superior para o.

Proposição 2.17 *Para quaisquer esquema $\mathcal{E}$, gtet Γ, atribuição δ e conjunto de caixas Y n-dimensionais ($n \geq 2$), a desigualdade $o(\mathcal{E}, \Gamma, \delta, Y) \leq o^\star(\mathcal{E}, \Gamma, \delta, Y)$ é satisfeita.*

Prova: Qualquer empacotamento de Y no gbloco induzido por (Γ, δ) respeitando o esquema $\mathcal{E}$ induz um empacotamento de $Y_{\overline{j}}$ no gbloco induzido por $(\Gamma^i_{\overline{j}}, \delta_{\overline{j}})$ respeitando o

esquema $\mathcal{E}_{\overline{j}}$. Pelo princípio de Cavalieri (ver página 327 de [18]), $o(\mathcal{E},\Gamma,\delta,Y) \leq \sum_{i=1}^{v_j-1} o(\mathcal{E}_{\overline{j}}, \Gamma^i_{\overline{j}}, \delta_{\overline{j}}, Y_{\overline{j}})(\delta_j(i)-\delta_j(i-1))$. Esta desigualdade se dá para todo j, implicando, pela definição de $o^\star$, $o(\mathcal{E},\Gamma,\delta,Y) \leq o^\star(\mathcal{E},\Gamma,\delta,Y)$. ■

Na seção 2.11, vimos que os limitante superiores são importantes no desempenho dos algoritmos para o.

Capítulo 3

Empacotando em dimensão 2

Um dos problemas abertos mais básicos da área de corte e empacotamento é encontrar o maior número de $\ell \times w$-retângulos que podem ser empacotados ortogonalmente num retângulo maior $L \times W$. Esta questão e sua importância prática foram as primeiras motivações para este trabalho. Neste capítulo, vamos, resumidamente, apresentar a evolução e em que ponto do conhecimento estamos nos problemas de empacotamento bidimensional. Em seguida, apresentamos alguns resultados originais que podem ampliar o entendimento destes problemas.

3.1 Problemas de empacotamento em dimensão 2

Os problemas de empacotamento bidimensional de grande importância prática (MPL e DPL), além do problema menos comum batizado aqui de L-MPL são definidos a seguir:

O problema MPL O problema denotado aqui por MPL, iniciais de *Manufacturer's Pallet Loading* ou, em português, Problema de Carregamento do Palete do Produtor, é também conhecido na literatura como *Manufacturer's PLP* (*Manufacturer's Pallet Loading Problem*). Este problema consiste em encontrar o maior número de $\ell \times w$-retângulos que podem ser empacotados ortogonalmente num retângulo maior $L \times W$. O termo ortogonalmente quer dizer simplesmente que os lados de todo $\ell \times w$-retângulo empacotado é paralelo a um dos lados do retângulo maior $L \times W$. Denotamos por $MPL(L, W, \ell, w)$ uma instância do problema MPL cujo retângulo maior tem dimensões $L \times W$ e o retângulo menor, que deve ser empacotado, tem dimensões $\ell \times w$.

O MPL aparece, como seu nome por extenso sugere, no caso em que um produtor fabrica um mesmo produto embalado em pacotes iguais na forma de paralelepípedos de lados (ℓ, w, h) que são posteriormente arrumados em paletes[1]. Os pacotes que são empilhados nos paletes vão normalmente arrumados em camadas de mesma altura. Por exemplo, se os pacotes (ℓ, w, h), por alguma restrição prática, não podem tombar, ou seja, a base dos pacotes tem de ser o retângulo $\ell \times w$, o problema passa a ser descobrir o maior número destes retângulos que pode ser empacotado no palete representado pelo retângulo $L \times W$. Depois disto, a solução do problema $MPL(L, W, \ell, w)$ pode ser empilhada em camadas de altura h até um certo limite prático H ($\lfloor H/h \rfloor$ camadas). Além da arrumação ou car-

[1] Os paletes são, normalmente, bases de madeira padronizadas, onde pacotes são empilhados. A utilidade dos paletes é basicamente facilitar a movimentação de pacotes através de empilhadeiras.

regamento do palete, o MPL também aparece no projeto de pacotes (*package design*) e no carregamento de caminhões entre outras aplicações.

Apesar de ter um enunciado simples, até hoje não se conhece uma solução eficiente para o MPL. Pelo contrário, alguns trabalhos sugerem que o MPL é NP-Completo [6]. Outro ponto importante de ser registrado é que cada problema $MPL(L, W, \ell, w)$ é na verdade um membro de uma classe infinita de problemas equivalentes ([14],[12] e [8]). Isto é, quando solucionamos um problema MPL estamos, de fato, resolvendo todos os problemas equivalentes a ele. Por exemplo, problemas cujas entradas diferem por uma mesma constante c são equivalentes: $MPL(L, W, \ell, w)$ é equivalente a $MPL(cL, cW, c\ell, cw)$.[2]

O problema DPL

O problema DPL, *Distributor's Pallet Loading* ou Problema de Carregamento do Palete do Distribuidor, é a generalização do problema MPL de um único $\ell \times w$-retângulo para muitos retângulos: $\ell_1 \times w_1$, $\ell_2 \times w_2, \ldots,$ $\ell_n \times w_n$. O problema consiste em encontrar um empacotamento ortogonal de ocupação máxima, utilizando quaisquer quantidades dos retângulos permitidos. O DPL também é conhecido como *Distributor's PLP* (*Distributor's Pallet Loading Problem*) ou *Rectangle Packing Problem* (RPP). O termo "distribuidor" faz referência ao fato de o distribuidor ter de arrumar, no palete, vários tipos de pacotes ao invés de apenas um tipo de pacote, como o produtor.

O problema L-MPL

Introduzimos nesta tese o problema que denotamos por L-MPL. Este problema nada mais é do que a extensão do problema MPL para "paletes" não mais retangulares, mas na forma da letra L. Esta noção de paletes em forma de L é apenas uma abstração e uma maneira de expor um problema importante para esta tese. O "palete" agora é definido através de uma quádrupla (L, W, L', W') com $L > L'$ e $W > W'$ que representa a região L obtida subtraindo do retângulo com diagonal que vai de $(0, 0)$ até (L, W) o retângulo com diagonal que vai de (L', W') até (L, W). Este problema consiste, assim como o MPL, em encontrar o maior número de $\ell \times w$-retângulos que podem ser empacotados ortogonalmente num "palete" (L, W, L', W') em forma de L. Denotamos por $LMPL(L, W, L', W', \ell, w)$ uma instância do problema L-MPL.

3.2 Breve histórico da evolução das abordagens para o MPL

Centenas de artigos já foram escritos sobre corte e empacotamento. A seguir faremos um breve histórico dos artigos que estão relacionados diretamente com o problema MPL.

Em 1985, Beasley [13] publicou uma modelagem em programação inteira 0-1 para o MPL. Esta abordagem é exata pois encontra uma melhor solução para o problema, porém o seu tempo computacional é explosivo, ou seja, para problemas de tamanho médio esta abordagem não funciona na prática. Outros autores também publicaram trabalhos com métodos exatos para resolver o MPL. Em 1987, Dowsland [11] apresentou uma abordagem interessante que consite em encontrar o maior conjunto estável de um grafo finito em particular. Nestes grafos, os nós representam retângulos posicionados no palete, e dois nós são adjacentes se os retângulos posicionados, representados por estes 2 nós, se sobrepuserem. Um conjunto estável de um grafo é um subconjunto dos vértices do grafo

[2] Nem todos os membros de uma mesma classe de equivalência diferem por uma constante, para maiores detalhes ver referências [14],[12] e [8].

com a propriedade de que não há dois vértices diferentes, neste subconjunto, que sejam adjacentes no grafo. O que Dowsland mostrou foi a existência de uma bijeção entre as soluções maximais e os conjuntos estáveis maximais do grafo correspondente ao problema MPL em questão. Esta abordagem é boa quando o número de retângulos empacotados é pequeno. Em 1993, Tsai et al sugeriram uma modelagem de programação inteira 0-1 diferente da de Beasley [9], mas que também sofre de um crescimento exponencial de suas restrições com o tamanho do problema.

Devido à complexidade dos métodos exatos para o MPL, algoritmos aproximados foram desenvolvidos. Dentre os métodos aproximados estão as heurísticas de bloco (não confundir com os blocos do capítulo 2). Blocos são retângulos onde todos os retângulos, empacotados dentro deste retângulo maior, têm a mesma orientação. Exemplos de heurísticas de blocos podem ser encontradas em Steudel [17] (1979) e em Smith e De Cani [16] (1980) onde o palete é dividido em dois blocos ou no máximo quatro blocos, e as caixas são, simplesmente, arrumadas com mesma orientação dentro de um mesmo bloco. Em 1982, Bischoff e Dowsland [15] refinaram o padrão utilizado por Smith e De Cani [16], permitindo uma divisão do palete em até cinco blocos. Em 1994 e 1995, Nelissen [8][6] propôs métodos de limitante superior e algoritmos genéticos para o MPL. Scheithauer e Terno [5], em 1996, propuseram uma heurística chamada de G4 que divide o palete recursivamente em no máximo quatro partes e armazena o melhor padrão G4. Em 1998, Morabito e Morales [4], utilizando o padrão de Bischoff e Dowsland [15] e a idéia da recursão utilizada por Scheithauer e Terno [5] definiram a heurística mais efetiva para o MPL até o momento. Essa heurística deixou de resolver otimamente apenas 16 classes de problema em um conjunto padrão de milhares de classes.

3.3 A heurística R-em-5Rs

O método aproximado mais efetivo para o MPL descrito, até o momento, foi desenvolvido por Morabito e Morales [4] como um refinamento da heurística de Bischoff e Dowsland [15]. Denotamos este método por heurística R-em-5Rs. Tal método consiste na aplicação recursiva das decomposições do esquema $\mathcal{E}_{R5R}$ (Figura 3.1). Seja $MPL(L, W, \ell, w)$ uma instância do problema MPL. A ocupação resultante da heurística R-em-5Rs, neste problema, é dada por:

$$o(\mathcal{E}_{R5R}, \mathbf{B^2}, ((0, L), (0, W)), \{(\ell, w), (w, \ell)\}),$$

onde $\mathcal{E}_{R5R}$ é o esquema que tem como peça apenas o gtet $\mathbf{B^2}$ (gtet que representa retângulos) e como decomposições os seguintes gtets:

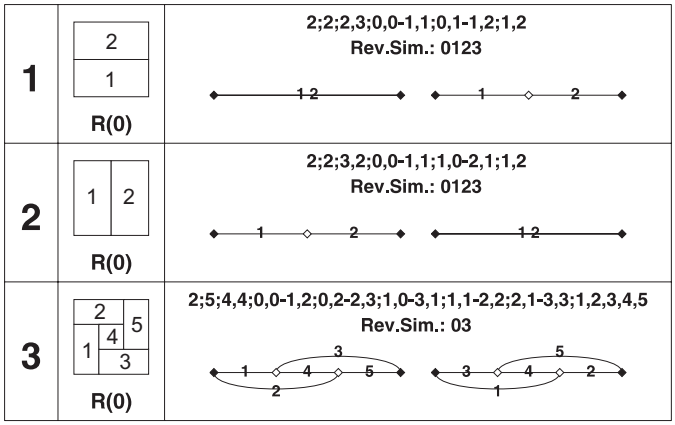


Figura 3.1: Decomposição do esquema $\mathcal{E}_{R5R}$.

Como podemos ver acima, o esquema da heurística R-em-5Rs tem apenas três decomposições: os dois padrões guilhotináveis mais simples e o padrão não-guilhotinável de primeira ordem mais simples que corta um retângulo em 5 retângulos menores.[3] A única peça neste esquema é o retângulo (peça R). Os vértices vazados (brancos) e os cheios (pretos), que aparecem nos perfis dos gtets acima, são respectivamente vértices móveis e vértices fixos. Aparecem também na figura, por motivo de padronização das apresentações dos esquemas, o código (código do gtet) e as reversões simétricas (reversões que não modificam o simin do gtet) das decomposições. A complexidade do esquema $\mathcal{E}_{R5R}$ é 6: a maior complexidade existente entre suas decomposições, no caso, a terceira decomposição.

A essência da heurística R-em-5Rs é uma aplicação bastante simples da teoria mais geral que desenvolvemos no capítulo 2. Esta heurística é muito efetiva para o problema MPL, tendo deixado abertas (sem solução ótima) apenas 16 classes em mais de 20000 classes padrão, na literatura (ver seção 3.5).

[3] Um padrão é guilhotinável se, em dimensão 2, existe um segmento que vai de uma extremidade da peça atá a outra [4].

3.4 A heurística RL

A heurística RL nasceu da busca de uma extensão para a heurística R-em-5Rs que capturasse as 16 classes do MPL sem solução ótima. Um suporte que tínhamos em tal busca eram exatamente as soluções ótimas de "representantes" destas classes. Tais soluções, por se tratarem de problemas pequenos, puderam ser computadas através da modelagem em programação linear inteira 0-1 para o problema MPL descrita por Beasley [13] usando o software de modelagem matemática MPL Modeling System/Cplex (o MPL do software é diferente do problema MPL e são as iniciais de *Mathematical Programming Language*).

Analisando os desenhos das 16 soluções "representantes" em questão foi possível verificar que uma extensão que utilizasse apenas peças retangulares (decomposição de retângulos em retângulos) era pouco promissora: havia, por exemplo, na solução para o $MPL(43, 26, 7, 3)$, a necessidade de uma decomposição de um retângulo em pelo menos 13 outros retângulos. A Figura 3.2 apresenta a solução para o $MPL(43{,}26{,}7, 3)$ e a menor decomposição de R em R necessária para capturar esta solução (o código, o gtet e a representação geométrica da decomposição). Isto obrigava a qualquer esquema de R em R que capturasse as 16 soluções a ter a decomposição da Figura 3.2 e, portanto, uma complexidade de pelo menos 14. Um salto de complexidade 6 do esquema $\mathcal{E}_{R5R}$ para 14 no esquema que estávamos tentando propor não parecia interessante.

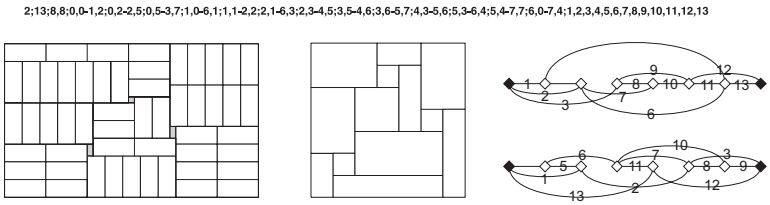


Figura 3.2: Decomposição mais simples de R em R para $MPL(43, 26, 7, 3)$.

Tendo desistido do princípio de usar apenas peças retangulares, a peça mais simples que estava "surgindo" naturalmente do reconhecimento de padrões que estávamos fazendo observando os desenhos era a peça L. Esta peça implicava num passo que aumentava a complexidade das peças consideradas (a peça L tem complexidade 4 enquanto que a peça retangular tem complexidade 2) portanto para que o passo valesse a pena deveria haver uma recompensa em algum lugar. Foi então que, no momento mais emocionante desta pesquisa, conseguimos, usando apenas decomposições de uma peça (L ou R) em duas outras peças (L ou R), decompor completamente todos os desenhos. Acabávamos de encontrar a recompensa: usando a peça L só precisávamos de decomposições em 2 partes. Lembre-se que na heurística R-em-5Rs existe uma decomposição de uma peça em 5 peças menores, portanto, neste sentido, o esquema que havíamos descoberto era mais simples. Batizamos tal esquema de *esquema* RL ou $\mathcal{E}_{RL}$. De modo surpreendente, a recompensa fez com que a complexidade de $\mathcal{E}_{RL}$ fosse igual a complexidade de $\mathcal{E}_{R5R}$.

O capítulo 2 desta tese é o resultado do trabalho de formalizar de maneira generalizada (dimensões maiores do que 1) a descoberta acima. A heurística RL nada mais é do que utilizar o esquema de grade $\mathcal{E}_{RL}$ cujos padrões de decomposição estão ilustrados na Figura 2.7 e os detalhes estão no apêndice B para os problemas MPL, L-MPL e DPL. Por exemplo, seja $MPL(L, W, \ell, w)$ uma instância do problema MPL, a heurística RL para esta instância é dada por:

$$o'(\mathcal{E}_{\mathsf{RL}}, \mathbf{B^2}, ((0, L), (0, W)), \{(\ell, w), (w, \ell)\}),$$

onde o' é a função de ocupância máxima para esquemas de grade (capítulo 2) como $\mathcal{E}_{RL}$ (apêndice B), $\mathbf{B^2}$ é o gtet que representa retângulos, o palete é o retângulo $L \times W$ e o retângulo menor é $\ell \times w$.

O seguinte resultado, de prova bastante simples, explica porque empacotamentos obtidos com a heurística RL são sempre melhores que os obtidos com R-em-5Rs.

Proposição 3.1 *Sejam E_1 e E_2 empacotamentos obtidos com* R-em-5Rs *e com* RL *respectivamente. Então a ocupância do empacotamento E_1 é no máximo igual a ocupância do empacotamento E_2.*

Prova: A prova advém do fato de que cada empacotamento considerado por E_1 é também um empacotamento considerado por E_2. De fato, como a figura abaixo mostra, qualquer subdivisão de um R em $5R's$ pode ser obtida em três níveis de subdivisões de $R's$ ou $L's$ em $R's$ ou $L's$. As outras duas decomposições do esquema R-em-5Rs também estão no esquema RL. Assim o resultado está provado. ∎

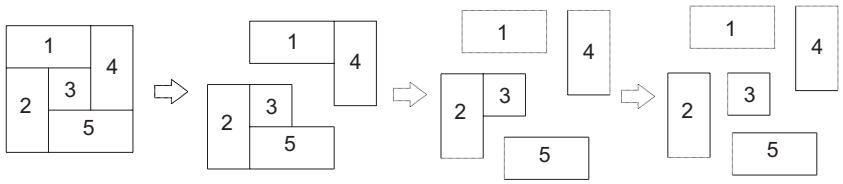


Figura 3.3: Prova de que a heurística RL domina a estratégia R-em-5Rs.

Apesar de a complexidade de $\mathcal{E}_{RL}$ e de $\mathcal{E}_{R5R}$ serem iguais, isto não é suficiente para afirmar que o tempo computacional dos dois esquemas é igual. Numa implementação para as funções o e o' utilizando programação dinâmica, por exemplo, a complexidade das peças é o que indica o número de subproblemas que podem vir a ser armazenados. Neste caso, o esquema $\mathcal{E}_{R5R}$ é mais simples que o esquema $\mathcal{E}_{RL}$ pois as peças mais complexas de cada um dos esquemas são respectivamente 2 e 4.

3.5 Resultados Computacionais

A heurística RL foi implementada em Java e rodou num microcomputador (Pentium III, 750Mhz e 256Mb de memória RAM). Para avaliar o desempenho, consideramos dois conjuntos de dados padrões na literatura de carregamento de paletes: o conjunto *Cover I* contendo 8274 instâncias moderadas satisfazendo $1 \leq LW/\ell w \leq 50$, e o conjunto *Cover II* contendo 41831 instâncias satisfazendo $1 \leq LW/\ell w \leq 100$ (onde (L, W) e $\ell, w)$ são respectivamente as dimensões do palete e da caixa). Como já mencionamos, o uso destas instâncias é comum na literatura (Nelissen, 1994, Scheithauer e Terno, 1996 Morabito e Morales, 1998, Letchford e Amaral, 2001, Morabito e Farago, 2001). Vale a pena ressaltar também que cada instância em *Covers I* e *II* é um representante (o menor membro) de

uma classe de equivalência (contendo infinitos elementos) de problemas (Dowsland, 1984, 1985, Nelissen, 1994).

Para testar a heurística R-em-5Rs, Morabito e Morales (1998) consideraram apenas exemplos com $L, W \leq 1000$, ou seja, 3179 e 16938 instâncias dentre as 8274 e 40831 instâncias dos conjuntos *Covers I* e *II*, respectivamente. A abordagem R-em-5Rs foi capaz de resolver otimamente todas as instâncias consideradas do conjunto *Covers I* e só não resolveu 16 instâncias das consideradas do conjunto *Covers II* (na verdade foi reportado no artigo original [4] que 18 instâncias não tinham sido resolvidas otimamente, porém, após revisão, concluiu-se que apenas 16 instâncias tinham sobrado [1]). Para obter a solução exata dos 16 problemas difíceis para a heurística R-em-5Rs, codificamos o modelo $(0,1)$-linear modelo de Beasley (1985) na linguagem de modelagem MPL e rodamos as 16 instâncias usando Cplex (versão 7).

Sabendo que a heurística RL domina a heurística R-em-5Rs (ver Proposição 3.1), restringimos os experimentos aos 16 problemas não resolvidos. A tabela abaixo apresenta os resultados obtidos pela heurística RL. Cada problema é denotado por $P_n = (L, W, \ell, w)$, onde n é o número de caixas da solução ótima. A coluna *Pontos da Grade* é o número de combinações lineares positivas das dimensões das caixa $(p \times q)$ menor ou igual a $L \times W$, *Subproblemas* é o total de peças $R's$ e $L's$ diferentes que podem ser definidas na grade, e *Tempo* é o tempo total em minutos que o programa rodou antes de obter a solução. Note que os problemas $P_{71} = (61, 35, 10, 3)$ and $P_{75} = (67, 37, 11, 3)$ são subproblemas de $P'_{77} = (61, 38, 10, 3)$ e $P_{81} = (67, 40, 11, 3)$, respectivamente.

Problema $P_n = (L, W, \ell, w)$	Pontos na Grade $p \times q$	Subproblemas $L's + R's$	Tempo $(min.)$
$P_{53} = (43, 26, 7, 3)$	37×20	$109,135$	2.83
$P_{57} = (49, 28, 8, 3)$	42×21	$159,537$	6.27
$P'_{69} = (57, 34, 7, 4)$	48×25	$294,450$	13.9
$P''_{69} = (63, 44, 8, 5)$	49×30	$418,200$	35.12
$P_{71} = (61, 35, 10, 3)$	52×26	$379,327$	31.65
$P_{75} = (67, 37, 11, 3)$	57×27	$499,959$	52.08
$P'_{77} = (61, 38, 10, 3)$	52×29	$457,243$	29.9
$P''_{77} = (61, 38, 6, 5)$	51×28	$411,747$	23.15
$P_{81} = (67, 40, 11, 3)$	57×30	$601,140$	46.08
$P'_{82} = (74, 49, 11, 4)$	59×34	$804,236$	131.52
$P''_{82} = (93, 46, 13, 4)$	75×28	$979,419$	134.78
$P'_{96} = (106, 59, 13, 5)$	82×35	$1,801,555$	326.07
$P''_{96} = (141, 71, 13, 8)$	99×31	$2,150,439$	359.85
$P_{97} = (74, 46, 7, 5)$	62×34	$905,318$	99.87
$P_{99} = (86, 52, 9, 5)$	70×36	$1,325,205$	179.3
$P_{100} = (108, 65, 10, 7)$	81×38	$2,033,342$	368.95

Table 1: Resultados computacionais dos 16 problemas difíceis

3.6 Algumas questões teóricas

Sob à luz dos resultados obtidos, é possível conjecturar que a heurística RL é ótima para o MPL ou problema do palete do produtor. Pelo menos até agora ela é empiricamente ótima, pois, nos experimentos realizados, usando tal heurística, obtivemos apenas soluções exatas.

Conjectura 3.1 *A heurística* RL *é ótima para o problema* MPL.

Em outras palavras: a heurística RL produz para o problema MPL definido pelos parâmetros (L, W, ℓ, w) um empacotamento com o maior número de blocos-caixa possível (empacotamento ótimo).

Se esta conjectura for verdadeira então temos em mãos um algoritmo exato de tempo $\mathcal{O}(L^6)$ para o MPL (ver seção 2.11).[4] Porém, se esta conjectura for falsa e tivermos um contra-exemplo (L, W, ℓ, w) ainda assim este resultado seria muito interessante para um melhor entendimento do problema, além do fato curioso de estarmos diante de um padrão nunca visto antes, ou pelo menos, nunca visto antes sob o aspecto que levantamos aqui. Que quatro números naturais teriam tal propriedade, se é que eles existem?

Descrevemos a seguir um resultado que, à primeira vista, pode indicar no sentido de que a conjectura 3.1 é falsa.

Teorema 3.1 *A heurística* RL *não é ótima para o problema* L-MPL.

Prova: A instância $(43, 26, 22, 16, 7, 3)$ do problema L-MPL não é resolvida exatamente pela heurística RL. A Figura 3.4 mostra uma solução exata específica para tal instância. Note que esta solução não pode ser decomposta de nenhuma maneira pelo esquema RL. Vale ressaltar que o fato de o empacotamento da Figura 3.4 não poder ser obtido pela heurística RL não implica que não existam outras soluções exatas para este mesmo problema que não possam ser capturadas pela heurística RL. Porém, aqui, este não é o caso, pois nenhuma outra solução de mesma ocupância é obtida pela heurística RL. ■

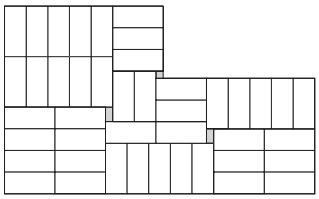


Figura 3.4: Peça L de dimensões $(43, 26, 22, 16)$ empacotadas com 33 caixas $(7, 3)$ e $(3, 7)$.

Note que, apesar de não ser obtido através da heurística RL, o empacotamento da Figura 3.4 está contido no empacotamento da Figura 1.2 (ele aparece de cabeça para baixo, retirando um retângulo com 10 caixas da esquina inferior direita) que foi obtido através da heurística RL para a instância $(43, 26, 7, 3)$ do problema MPL. Como interpretar esta observação? Deixamos em aberto.

Pela definição de gbloco, podemos mostrar que, em dimensão 2, os gblocos são "escadas" (ver Figura 2.2). A escada de um degrau é a peça R (um retângulo). A escada de dois degraus é a peça L. Até seja provado o contrário, existe a possibilidade de o problema L-MPL ser resolvido otimamente a partir de um esquema que permita escadas de até três degraus. O exemplo da Figura 3.4 é capturado a partir deste esquema.

[4] A expressão exata do tempo computacional da heurística RL não é o polinômio L^6, entretanto, como denota a notação $\mathcal{O}$, estamos nos referindo a um limitante superior para tal tempo.

Proposição 3.2 *Em dimensão 2, todo empacotamento em um gbloco K ou contém um único bloco-caixa de volume igual a $vol(K)$ ou pode ser bipartido em dois gblocos menores.*

Prova: Se o volume do gbloco K é igual ao volume da única caixa empacotada nele, então não é possível bipartir este gbloco em dois gblocos menores. Caso contrário, como vemos a seguir, sempre podemos bipartir o empacotamento no gbloco K em dois empacotamentos em gblocos menores (empacotamentos possivelmente vazios, mas que não deixam de ser empacotamentos). Os gblocos, em dimensão 2, são escadas. Os casos de 1 a 4, que nos referimos adiante, são relativos à Figura 3.5. Sempre que a esquina de um degrau da escada não está ocupado por nenhuma caixa, podemos aplicar o caso 1. Neste caso tiramos o maior retângulo possível da esquina vazia sem ultrapassar as linhas x_0 e y_0. Sobram então os casos em que há caixas empacotadas em todas as esquinas da escada que é o nosso gbloco K. Se existe uma caixa numa esquina que não ultrapassa os segmentos x_0 nem o segmento y_0 então é possível remover esta caixa como indica o caso 2 . Se uma caixa numa esquina de degrau ultrapassa um dos segmentos x_0 ou y_0, então esta caixa não pode ser removida, pois geraria um não-gbloco (caso 3). Então, a única maneira de não ser possível retirar nenhuma caixa de esquina de degrau é que toda esquina de degrau seja como no caso 3 (ou seu análogo cortando x_0). O problema é que isto não é possível em todos os degraus simultaneamente. Suponha que todos os degraus a partir do mais baixo até o penúltimo estejam como no caso 3. Isto é possível. Porém, o último degrau tem que ser do tipo mostrado no caso 4, isto é, um exemplo do caso 2 que pode ser bipartido. Os outros casos são todos análogos a este. Portanto, quando todos os degraus têm caixas na esquina, sempre haverá um degrau onde o caso 2 se aplica. Assim, a proposição está provada.

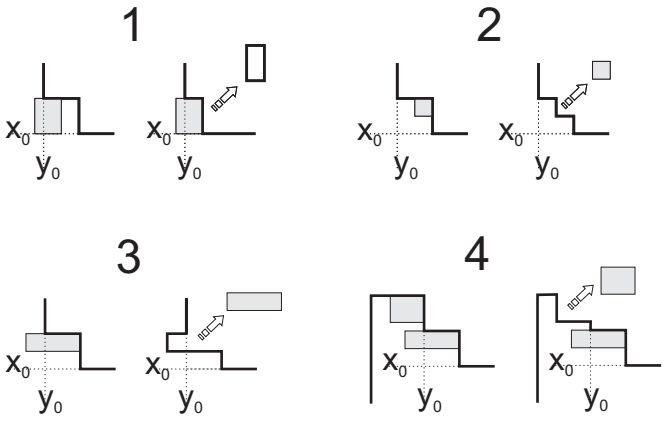


Figura 3.5: Casos para a prova da Proposição 3.2.

Suponha ser Ω_0 um problema MPL contra-exemplo para a conjectura 3.1 (não sabemos se Ω_0 realmente existe, é só um suposição). A partir da Proposição 3.2 é possível afirmar que, apesar do esquema RL não capturar Ω_0, existe um esquema $\mathcal{E}_0$ que captura Ω_0. Se existe um contra-exemplo Ω_1 para $\mathcal{E}_0$, então podemos novamente aplicar a Proposição 3.2 e dizer que existe um esquema $\mathcal{E}_1$ que resolve Ω_1. Disto segue que apenas uma das opções seguintes é verdadeira: ou este processo é infinito ($\mathcal{E}_1$,$\mathcal{E}_2$,...), ou existe um esquema $\mathcal{E}_{MPL}$ (que ainda pode ser o $\mathcal{E}_{RL}$) definitivo que resolve exatamente o MPL. Supondo que o processo é finito (segunda alternativa) e que α seja a maior complexidade de uma peça de

$\mathcal{E}_{MPL}$, existe um algoritmo de complexidade

$$\mathcal{O}(L^{2\cdot\alpha}),$$

pois $Complex(\mathcal{E}_{MPL}) \leq 2 \cdot \alpha$ (pela Proposição 3.2 $\mathcal{E}_{MPL}$ tem apenas 2-decomposições, portanto sua complexidade é limitada pelo dobro da complexidade da sua peça mais complexa). Ou seja, se o esquema $\mathcal{E}_{MPL}$ existe, então existe um algoritmo com custo polinomial em L (maior medida do retângulo maior $L \times W$) que resolve exatamente o problema MPL.

Capítulo 4

Empacotando em dimensão 3

Neste capítulo sugerimos um esquema (assim como o esquema RL em dimensão 2) para problemas de empacotamento em dimensão 3. Este esquema se encaixa na teoria descrita no capítulo 2 e foi obtido de maneira análoga ao que fizemos no caso bidimensional onde generalizamos a heurística R-em-5Rs na heurística RL. Em 2002, Lins, Lins e Morabito [2] sugerem o análogo tridimensional à heurística bidimensional R-em-5Rs. Esta abordagem, denotada aqui por B-em-9Bs, é baseada numa decomposição de um bloco (paralelepípedo) em 9 blocos (paralelepípedos) menores. O esquema que propomos neste capítulo generaliza a abordagem B-em-9Bs através de 2-decomposições de 5 tipos de peças: B, A, L, S e T.

4.1 A heurística B-em-9Bs

Motivados pelo excelente resultado prático obtido pela heurística bidimensional R-em-5Rs (apenas 16 problemas não ótimos num conjunto padrão de mais de 20000 problemas), Lins, Lins e Morabito propuseram sua generalização tridimensional [2]. Tal heurística é denotada aqui por B-em-9Bs devido a sua decomposição principal particionar um bloco (paralelepípedo) em 9 blocos (paralelepípedos) menores como mostra a figura a seguir.

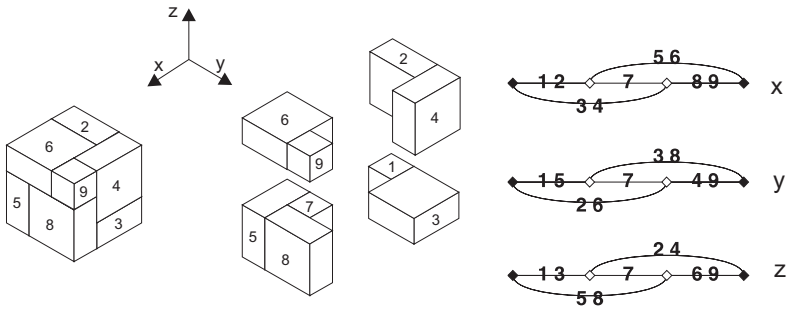


Figura 4.1: Decomposição base para a heurística B-em-9Bs.

O esquema $\mathcal{E}_{B9B}$ da heurística B-em-9Bs é apresentado completamente no apêndice C. Ele é constituído por todas as decomposições que são *minor* da decomposição da figura acima e que são *"primas"*, no sentido de que não podem ser obtidas partir de outras decomposições em etapas posteriores. A complexidade do esquema $\mathcal{E}_{B9B}$ é 9, como podemos verificar na Figura 4.1 que apresenta sua decomposição mais complexa. A peça mais

complexa deste esquema é o gtet $\mathbf{B^3}$ que representa paralelepípedos e cuja complexidade é 3. A idéia de *primos*, não considerada em [4] e [2] precisa ainda ser implementada e tem potencial para melhorar os tempos computacionais dos algoritmos destes artigos.

4.2 A heurística BALST

Com o conhecimento obtido no caso bidimensional, o objetivo agora era encontrar as peças tridimensionais que tornassem possível um esquema de 2-decomposições que generalizasse o esquema $\mathcal{E}_{B9B}$, isto é, as peças que seriam "análogas" à peça L bidimensional. A solução mais simples encontrada resultou nas seguintes peças:

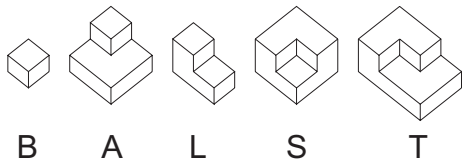


Figura 4.2: Peças do esquema BALST.

A intuição por trás dos nomes das peças é a seguinte: a peça B vem da palavra "bloco"; a peça A vem da palavra "aditivo", no sentido de um bloco adicionado de um bloco menor numa esquina; a peça L é devido à forma em "L"; a peça S vem da palavra subtrativo no sentido de um bloco subtraído de um bloco menor numa esquina; a peça T não vem de nenhuma idéia especial, foi uma escolha aleatória. A peça T, quando degenerada, pode gerar todas as outras quatro peças: B, A, L e S.

Note que nem todas as peças que apresentamos na Figura 4.2 são simétricas. A peça L e a peça A podem ser, cada uma, dispostas de três maneiras distintas que devem, a menos de equivalência entre eixos, ser consideradas peças diferentes. Já a peça T pode ser disposta de seis maneiras distintas.

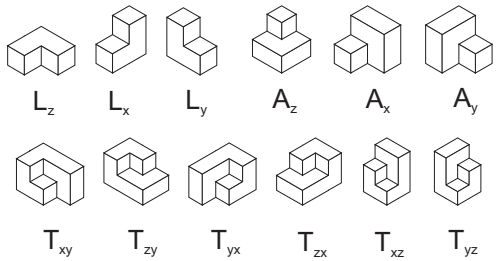


Figura 4.3: As diferentes peças L, A e T.

O esquema $\mathcal{E}_{BALST}$ com todos os detalhes está descrito no apêndice D. As 430 decomposições deste esquema são os *minors* de todas as decomposições de uma peça T em duas peças T. Para obtê-las, primeiro encontramos todas as maneiras de decompor uma peça T_{xy} em duas peças T quaisquer. Para isto, consideramos a maior peça T_{xy} que cabe em um cubo $3 \times 3 \times 3$. Esta peça é formada por 23 dos 27 cubinhos constituintes do cubo $3 \times 3 \times 3$. Em seguida, enumeramos as 2^{23} possibilidades de colorir cada cubinho com as cores 1 e 2. Filtramos as colorações cujos cubinhos de mesma cor formam um T. O resultado desta enumeração exaustiva são as duas decomposições mostradas abaixo. Note que a decomposição 419 é um T_{xy} formado por um T_{xy} e um T_{yz}. Já a decomposição 420 é também um T_{xy}, mas agora formado por dois T_{yz}'s. Assim estas decomposições são incongruentes por rotação e reflexão.

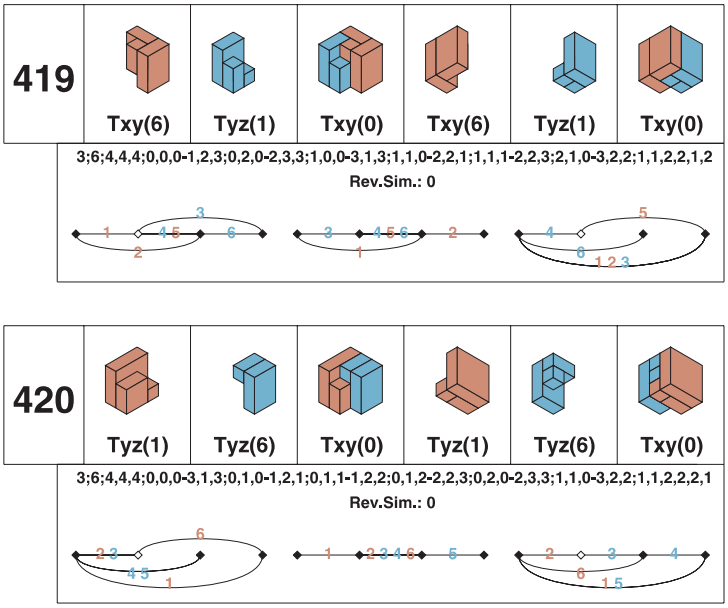


Figura 4.4: As duas decomposições de um T em dois T's.

Outras 10 decomposições de T em dois T's são obtidas permutando os eixos x, y e z das seis maneiras possíveis. As doze soluções são as decomposições 419 a 430 do Apêndice D. O esquema BALST é obtido tomando todos os minors destas 12 decomposições, originando as 430 decomposições exibidas no Apêndice. Os fatos relevantes sobre este esquema são os seguintes. Analogamente aos esquemas R-em-5Rs e RL no caso bidimensional, o esquema BALST é mais poderoso que o esquema B-em-9Bs como demonstra a Proposição 4.1 e, ainda assim, suas complexidades são iguais a 9. Entretanto, a peça mais complexa do esquema BALST tem complexidade 7, enquanto que a peça mais complexa do esquema B-em-9Bs tem complexidade 3.

Teorema 4.1 *Sejam E_1 e E_2 empacotamentos obtidos com* B-em-9Bs *e com a heurística* BALST *respectivamente. Então a ocupância de E_1 é no máximo igual a ocupância de E_2.*

Prova: A prova advém do fato de que cada empacotamento considerado por E_1 é também um empacotamento considerado por E_2. De fato, como a figura abaixo mostra, qualquer subdivisão de um B em $9B's$ pode ser obtida em quatro etapas de 2-decomposições

do esquema BALST. Apesar de não mostrarmos aqui, as outras 13 decomposições do esquema B-em-9Bs (ver apêndice C) são também obtidas a partir de decomposições do esquema BALST. Assim, o resultado está provado. ■

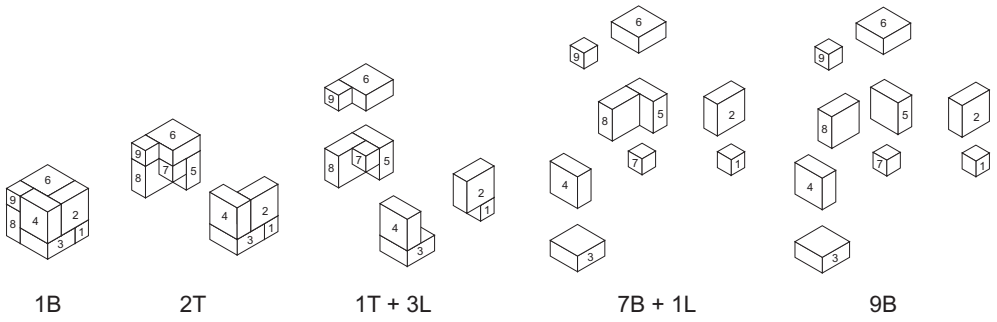


Figura 4.5: Prova de que a heurística *BALST* domina a estratégia *B-em-9Bs*.

Observe que a captura da decomposição de B em 9B do esquema $\mathcal{E}_{B9B}$ pelo esquema BALST apresentada na Figura 4.5 não utiliza as peças A nem S, apenas as peças B, L e T. Porém, as peças A e S são casos degenerados da peça T, elas aparecem nos *minors* das decomposições "base" de T em T. Seguimos o princípio, tanto no esquema RL quanto no esquema BALST de, dada a peça mais complexa do esquema (L e T, respectivamente), introduzir no esquema todas as peças degeneradas da mesma.

Capítulo 5

Perspectivas Futuras

Apesar de parecer teórica, esta tese tem uma forte motivação prática. É certo que, ainda hoje, caminhões, navios e aviões transportam contêineres e paletes que levam uma carga menor do que poderiam. A economia que a otimização dos carregamentos de paletes e contêineres poderia acarretar é significativa. Outra certeza é que, principalmente no caso tridimensional, não é conhecida nenhuma estratégia geral que seja boa na prática.

A linguagem construída neste trabalho: blocos, gblocos, gtets, esquemas, peças, decomposições etc, possibilitou a modelagem de heurísticas já existentes (R-em-5Rs e B-em-9Bs) e novas heurísticas (RL e BALST) de diferentes dimensões (2 em diante). Tais heurísticas passaram a ser vistas como parâmetro para um mesmo algoritmo (implementação da função o ou o'). Porém, a motivação maior para esta linguagem é chegar a estratégias de empacotamento (principalmente o tridimensional) que funcionem na prática.

A partir da definição formal de peças e decomposições através dos gtets, é possível vislumbrar uma estratégia que leve em consideração demanda. Suponha um cadastro de produtos onde, para cada produto, são armazenados vários empacotamentos em diversas peças (ex. B, A, L, S ou T) de tamanhos variados. Para cada empacotamento armazenado há também o registro do número de caixas do produto que vai empacotado naquela solução. O problema de empacotar uma certa demanda de produtos diferentes num determinado contêiner pode ser visto como montar o seguinte quebra-cabeça: as peças são os empacotamentos já armazenados no cadastro e as regras de encaixe das peças são descritas pelas decomposições, que neste caso exercem papel de composições, para formar uma peça maior: o bloco do tamanho do contêiner. Alguns passos já foram dados na direção de um cadastro de peças (gtets) como, por exemplo, a estrutura de dados descrita em [3].

A seguir listamos algumas atividades que dão continuidade à pesquisa realizada:

- Busca por implementações gerais e eficientes para as funções o e o' ainda são objetos de estudo. A partir da experiência que obtivemos em protótipos, temos confiança de que é possível chegar a uma implementação geral e eficiente.

- Confirmação ou negação da conjectura 3.1.

- Análise do comportamento da heurística BALST num conjunto padrão de problemas.

- Utilização dos esquemas como uma forma de compor novos empacotamentos considerando demanda.

- Em dimensão 3, é possível modificar a função o para considerar a estabilidade das caixas?
- Extensão das funções o e o' para considerar caixas como gblocos e não mais como blocos. Isto tem a ver com a montagem do quebra-cabeças que mencionamos no parágrafo acima.

Apêndice A

Soluções RL para os 16 problemas MPL "difíceis"

Os 16 "representantes" de classe MPL não resolvidos otimamente pela heurística R-em-5Rs e resolvidos otimamente pela heurística RL são apresentados a seguir. Ao lado de cada solução estão as decomposições em R e L utilizadas.

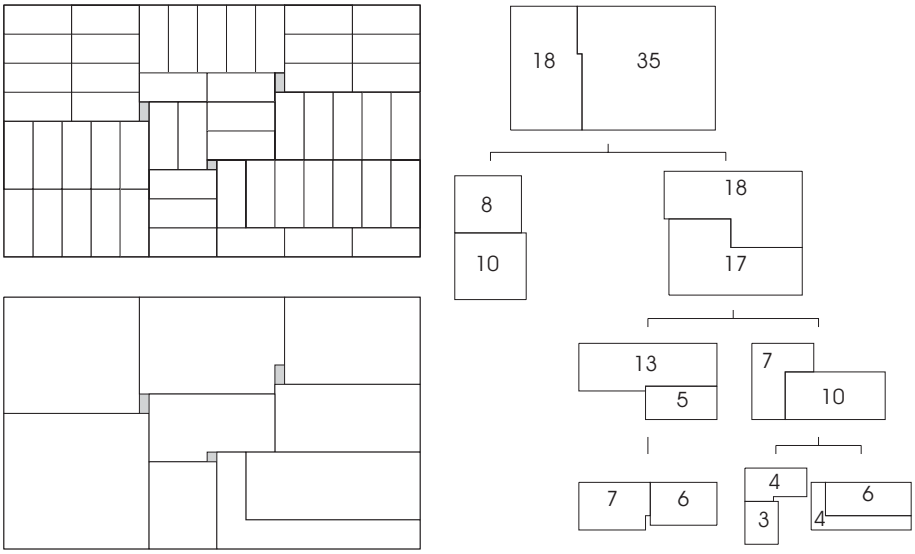


Figura A.1: $P_{53} = (43, 26, 7, 3)$ (53 caixas)

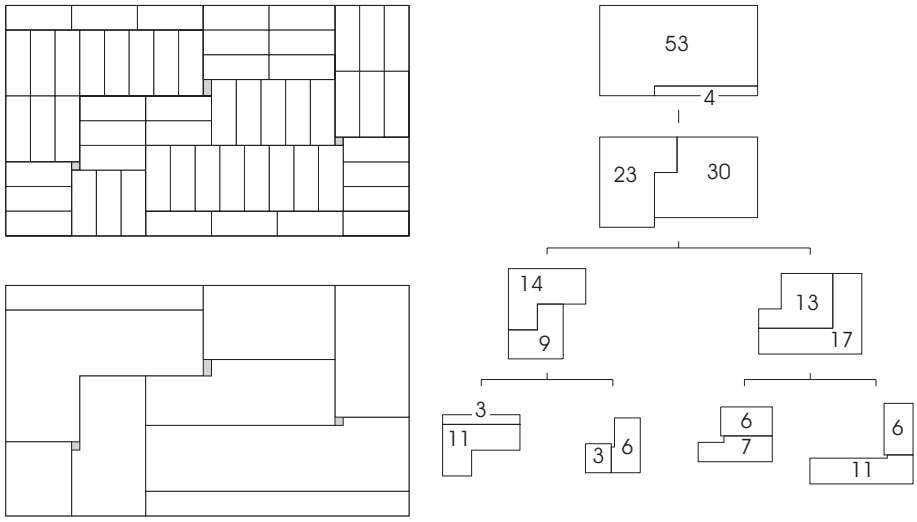


Figura A.2: $P_{57} = (49, 28, 8, 3)$ (57 caixas)

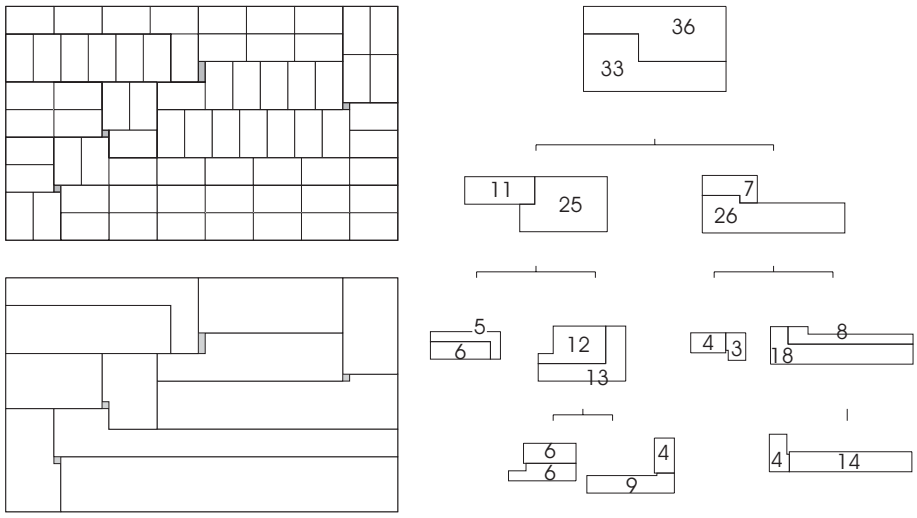


Figura A.3: $P'_{69} = (57, 34, 7, 4)$ (69 caixas)

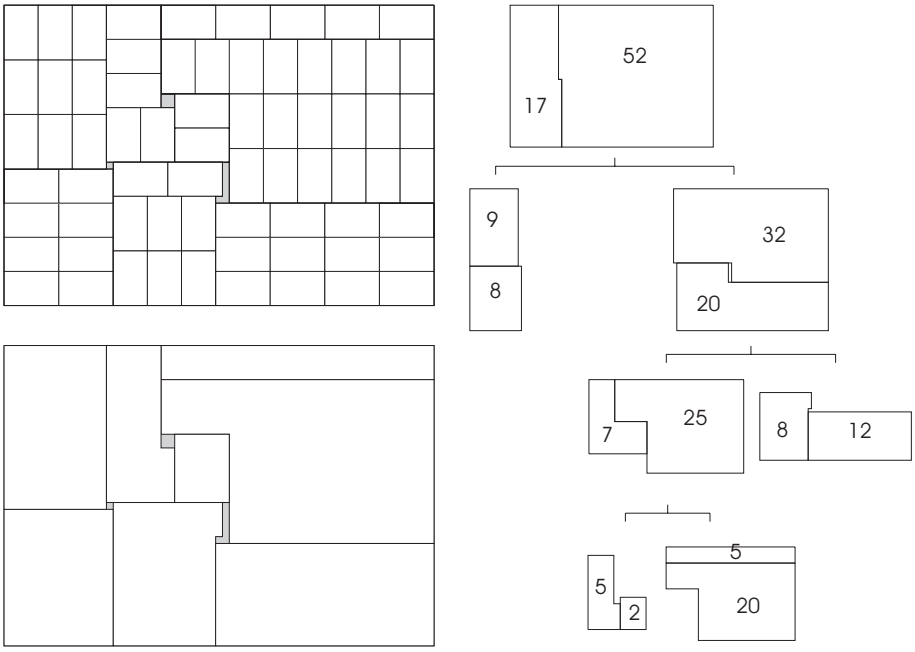


Figura A.4: $P''_{69} = (63, 44, 8, 5)$ (69 caixas)

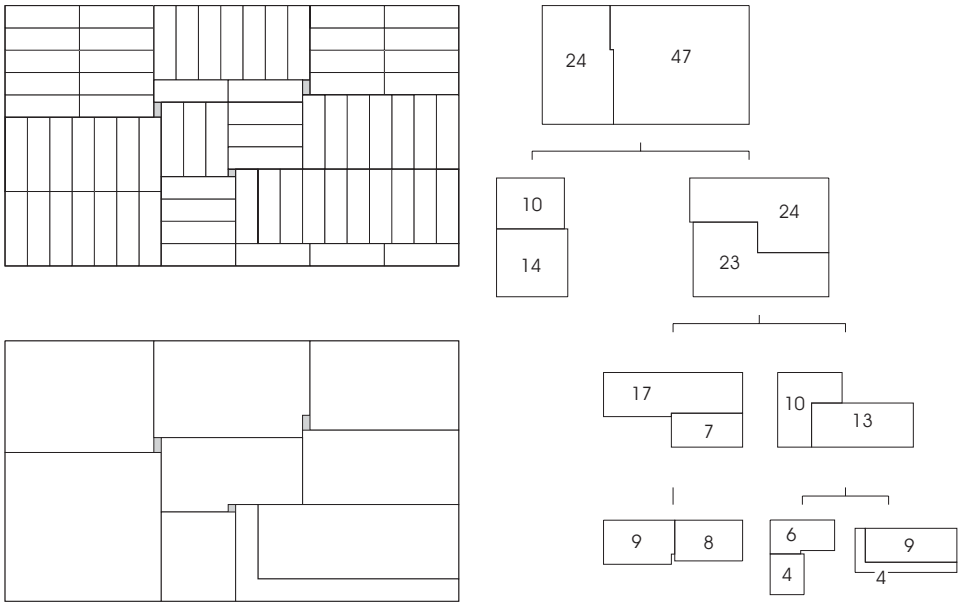


Figura A.5: $P_{71} = (61, 35, 10, 3)$ (71 caixas)

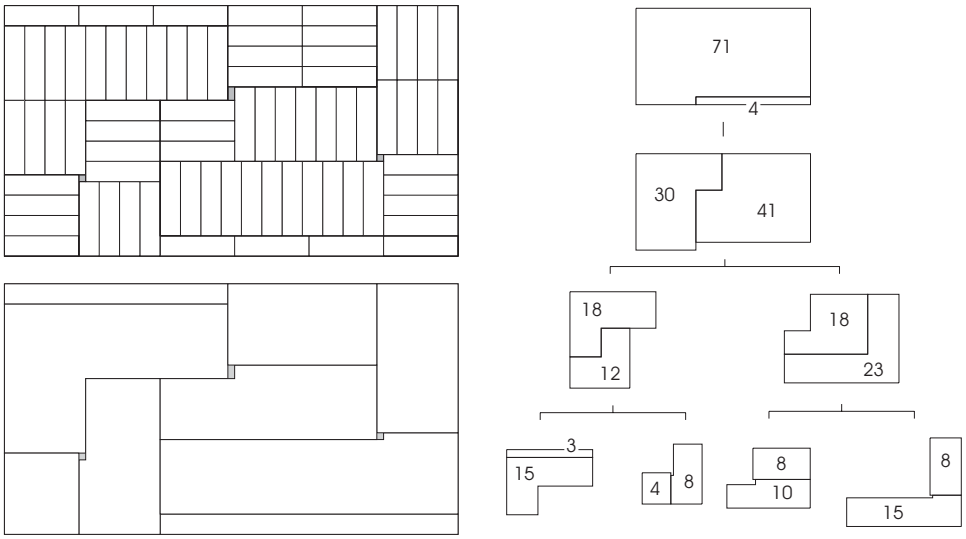


Figura A.6: $P_{75} = (67, 37, 11, 3)$ (75 caixas)

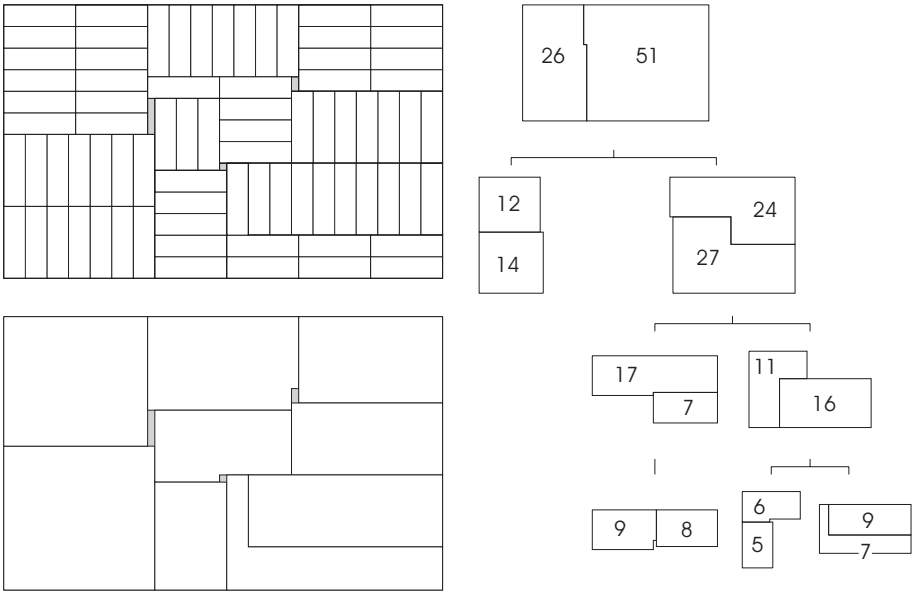


Figura A.7: $P'_{77} = (61, 38, 10, 3)$ (77 caixas)

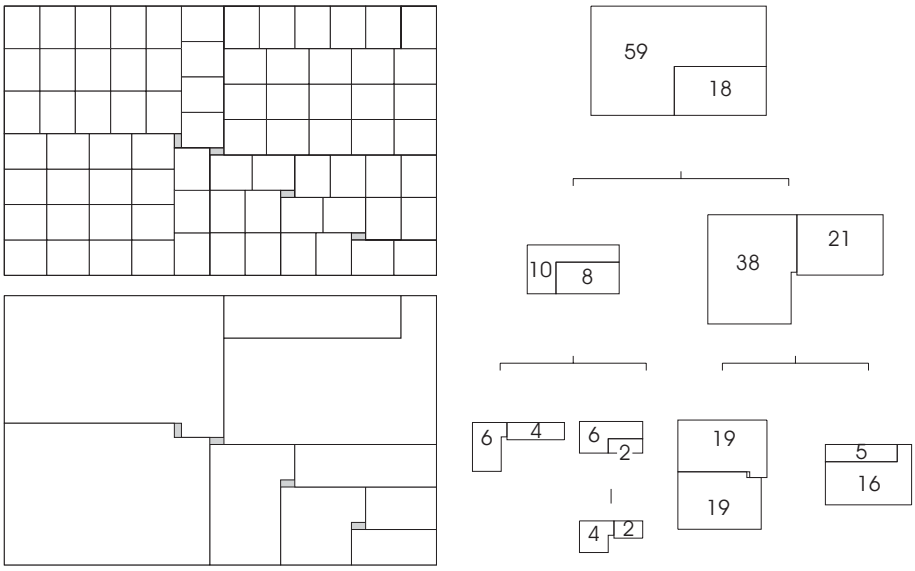


Figura A.8: $P''_{77} = (61, 38, 6, 5)$ (77 caixas)

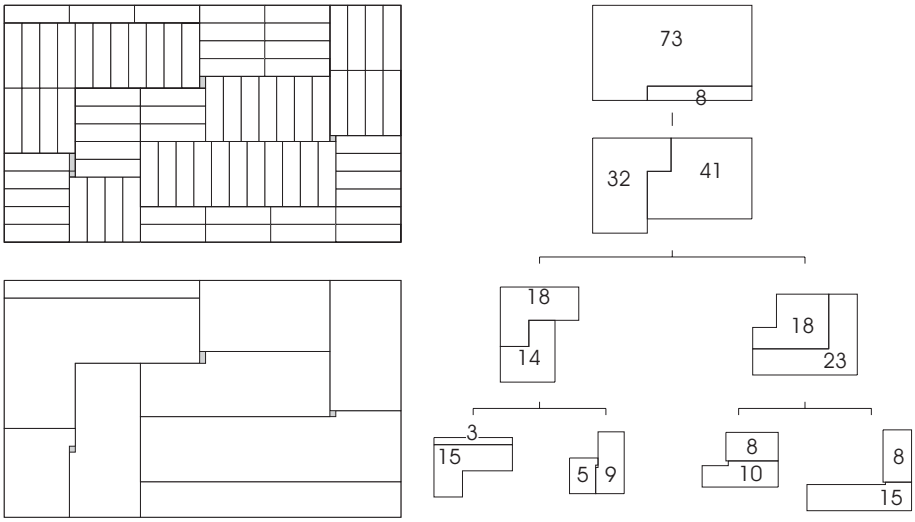


Figura A.9: $P_{81} = (67, 40, 11, 3)$ (81 caixas)

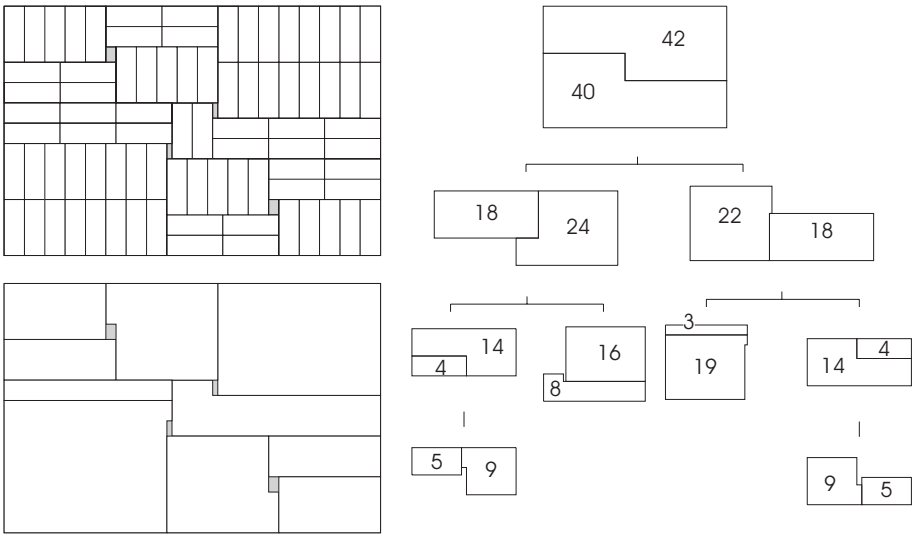


Figura A.10: $P'_{82} = (74, 49, 11, 4)$ (82 caixas)

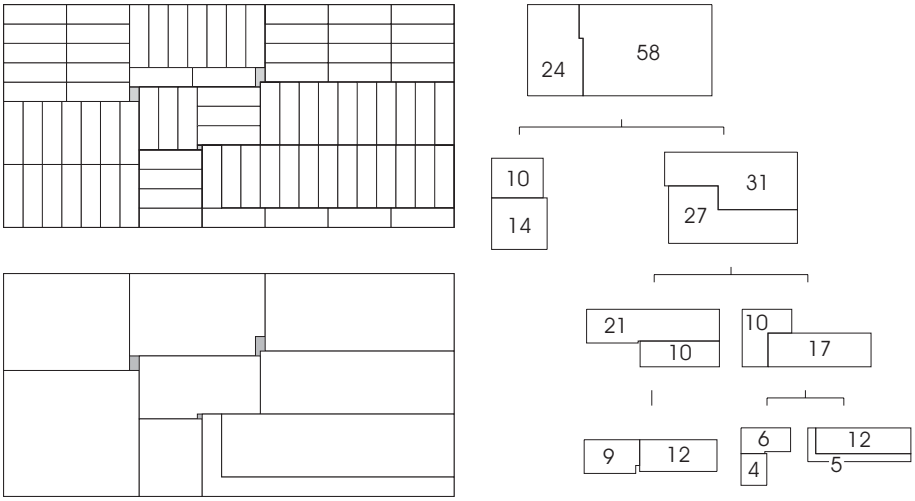


Figura A.11: $P''_{82} = (93, 46, 13, 4)$ (82 caixas)

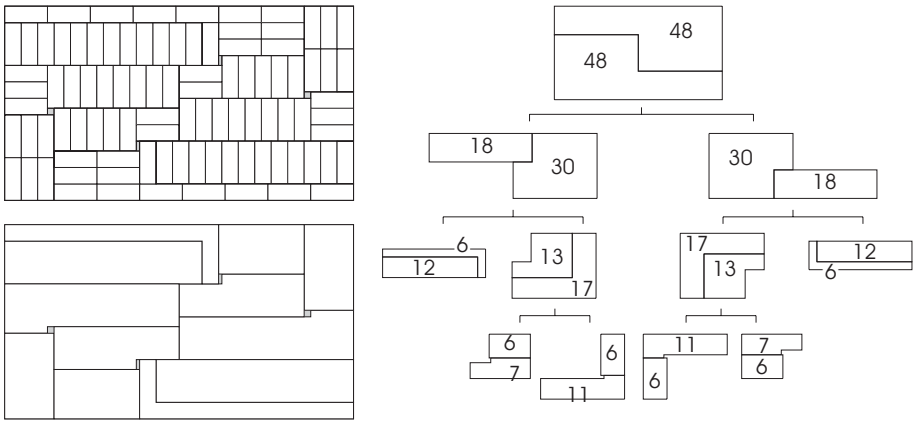


Figura A.12: $P'_{96} = (106, 59, 13, 5)$ (96 caixas)

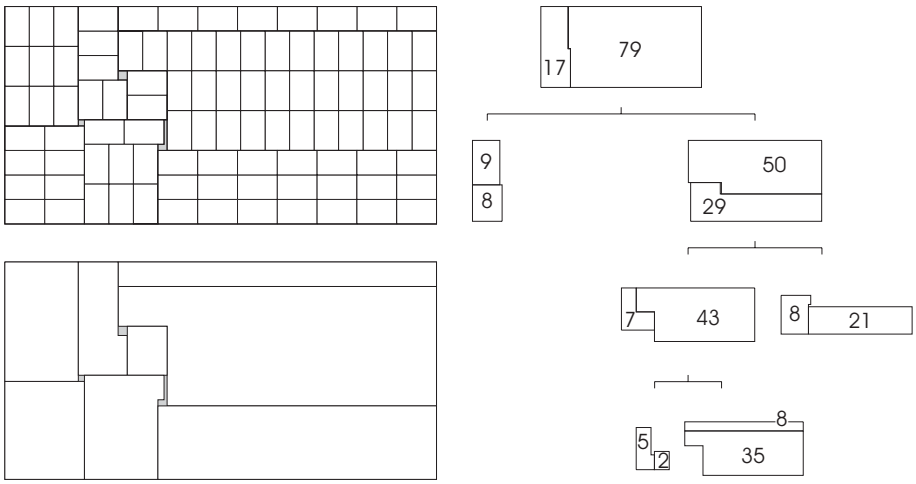


Figura A.13: $P''_{96} = (141, 71, 13, 8)$ (96 caixas)

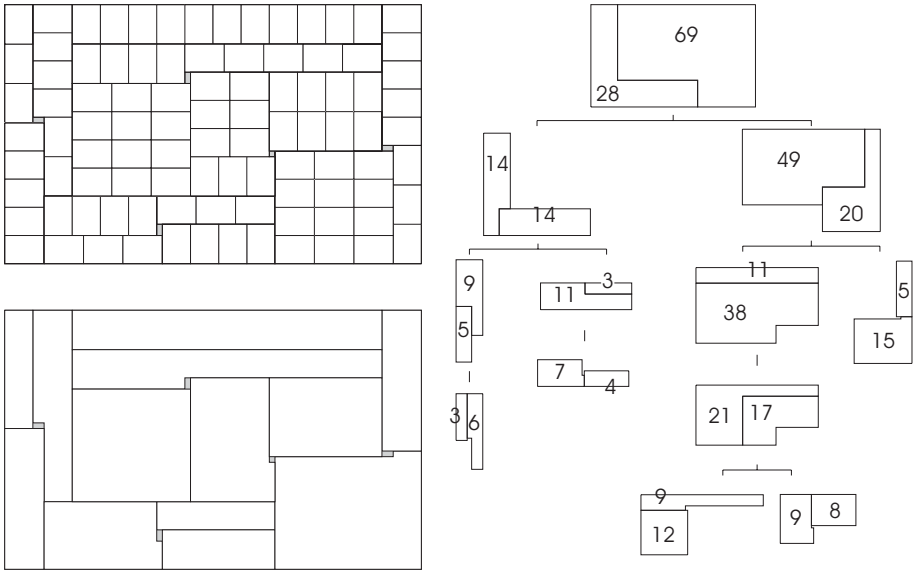


Figura A.14: $P_{97} = (74, 46, 7, 5)$ (97 caixas)

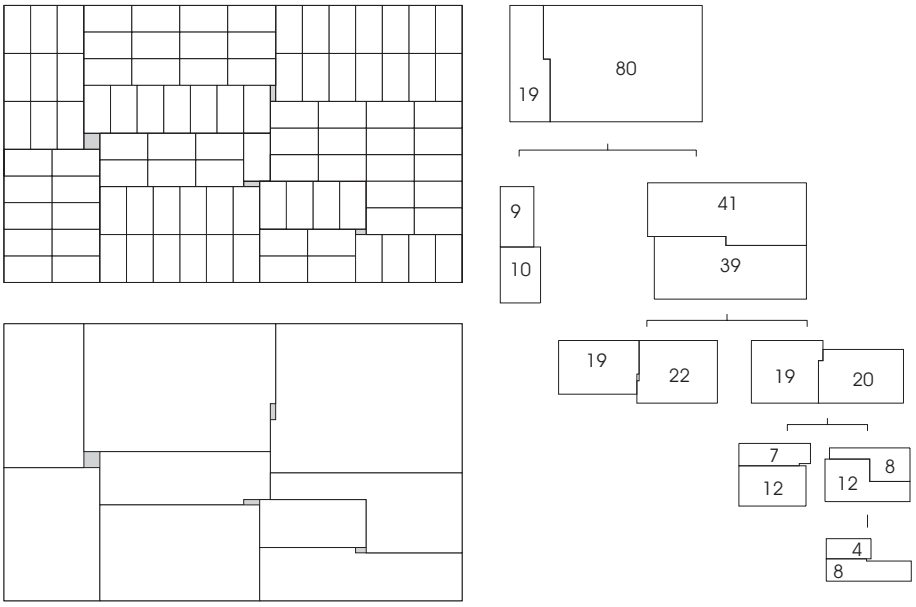


Figura A.15: $P_{99} = (86, 52, 9, 5)$ (99 caixas)

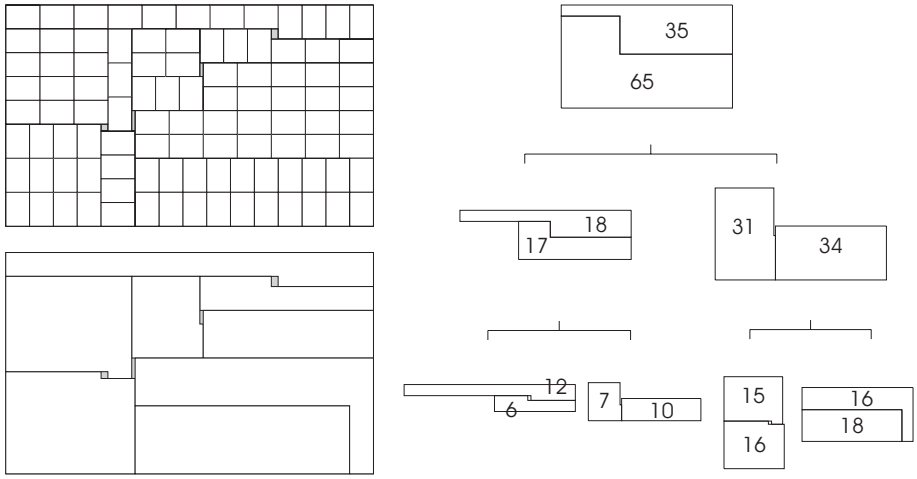


Figura A.16: $P_{100} = (108, 65, 10, 7)$ (100 caixas)

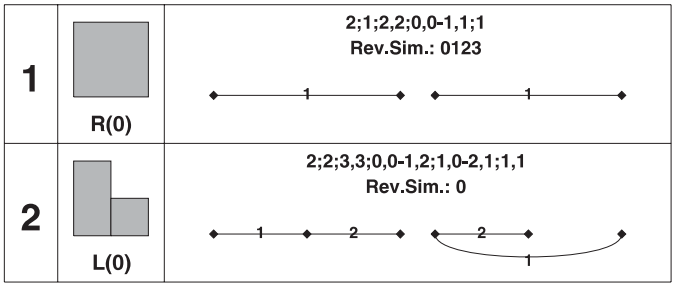
Apêndice B

Esquema RL

As peças e decomposições do esquema bidimensional RL ou $\mathcal{E}_{RL}$ são apresentadas a seguir. São, no total, 2 peças e 20 decomposições. A complexidade deste esquema é 6 (as decomposições de 16 a 20 têm complexidade máxima igual a 6). A peça L, a mais complexa do esquema, tem complexidade 4. A partição de eixos associada a este esquema pode ser tanto $\{\{1\},\{2\}\}$ como $\{\{1,2\}\}$.

Em cada peça/decomposição a seguir, apresentamos o código e as reversões simétricas daquela peça/decomposição. Além disso, abaixo da representação geométrica de cada peça/decomposição, indicamos o nome da peça (no caso de uma simples peça) ou o nome da peça "pai" (no caso de uma decomposição) e, entre parênteses, a reversão (em decimal) que se encontra aquela peça ou peça "pai". Os vértices cheios (pretos) são fixos e os vértices vazados (brancos) são móveis.

B.1 Peças



B.2 Decomposições

1	 R(0)	 R(0)	 R(0)	2;2;2,3;0,0-1,1;0,1-1,2;1,2 Rev.Sim.: 0123
2	 R(0)	 R(0)	 R(0)	2;2;3,2;0,0-1,1;1,0-2,1;1,2 Rev.Sim.: 0123
3	 R(0)	 R(0)	 L(0)	2;2;3,3;0,0-1,2;1,0-2,1;1,2 Rev.Sim.: 0 
4	 R(0)	 R(0)	 L(0)	2;2;3,3;0,0-2,1;0,1-1,2;1,2 Rev.Sim.: 0 
5	 L(1)	 R(0)	 R(0)	2;3;3,3;0,0-1,1;0,1-1,2;1,0-2,2;1,2,1 Rev.Sim.: 0 
6	 L(0)	 R(0)	 L(0)	2;3;3,4;0,0-1,2;0,2-1,3;1,0-2,1;1,2,1 Rev.Sim.: 0 
7	 L(1)	 R(0)	 L(0)	2;3;3,4;0,0-1,1;0,1-1,3;1,0-2,2;1,2,1 Rev.Sim.: 0 
8	 L(0)	 R(0)	 L(0)	2;3;3,4;0,0-1,3;1,0-2,1;1,1-2,2;1,1,2 Rev.Sim.: 0 
9	 R(0)	 L(0)	 L(0)	2;3;3,4;0,0-2,1;0,1-1,3;1,1-2,2;1,2,2 Rev.Sim.: 0 
10	 L(0)	 R(0)	 L(0)	2;3;4,3;0,0-1,2;1,0-2,1;2,0-3,1;1,1,2 Rev.Sim.: 0 

11	 L(2)	 R(0)	 L(0)	2;3;4,3;0,0-1,1;0,1-2,2;1,0-3,1;1,1,2 Rev.Sim.: 0 
12	 R(0)	 L(0)	 L(0)	2;3;4,3;0,0-1,2;1,0-2,2;2,0-3,1;1,2,2 Rev.Sim.: 0 
13	 L(0)	 R(0)	 L(0)	2;3;4,3;0,0-1,2;1,0-3,1;1,1-2,2;1,1,2 Rev.Sim.: 0 
14	 L(1)	 L(2)	 R(0)	2;4;3,4;0,0-1,1;0,1-1,2;0,2-2,3;1,0-2,2;1,2,2,1 Rev.Sim.: 03 
15	 L(2)	 L(1)	 R(0)	2;4;4,3;0,0-1,1;0,1-2,2;1,0-2,1;2,0-3,2;1,1,2,2 Rev.Sim.: 03 
16	 L(0)	 L(3)	 L(0)	2;4;4,4;0,0-1,2;0,2-1,3;1,0-3,1;1,1-2,3;1,2,1,2 Rev.Sim.: 0 
17	 L(0)	 L(3)	 L(0)	2;4;4,4;0,0-1,3;1,0-2,1;1,1-2,2;2,0-3,2;1,1,2,2 Rev.Sim.: 0 
18	 L(1)	 L(2)	 L(0)	2;4;4,4;0,0-1,1;0,1-1,2;0,2-2,3;1,0-3,2;1,2,2,1 Rev.Sim.: 0 
19	 L(2)	 L(1)	 L(0)	2;4;4,4;0,0-1,1;0,1-2,3;1,0-2,1;2,0-3,2;1,1,2,2 Rev.Sim.: 0 
20	 L(0)	 L(0)	 L(0)	2;4;4,4;0,0-1,3;1,0-3,1;1,1-2,3;2,1-3,2;1,1,2,2 Rev.Sim.: 0 

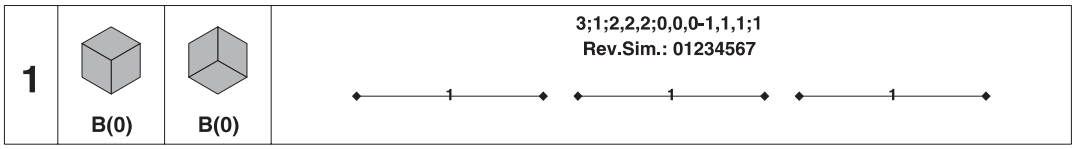
Apêndice C

Esquema B-em-9Bs

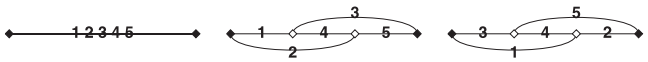
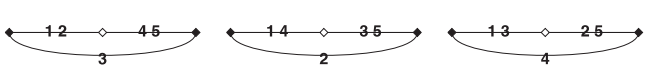
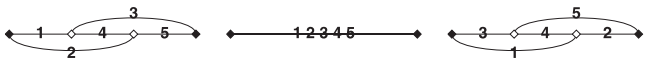
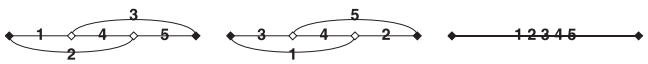
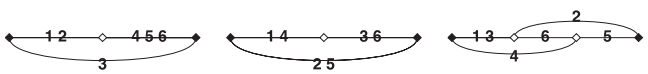
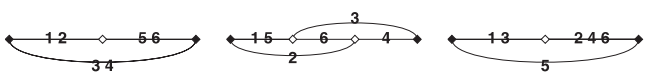
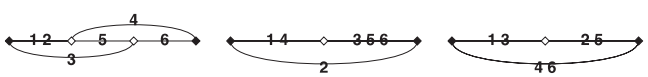
As peças e decomposições do esquema tridimensional B-em-9Bs ou $\mathcal{E}_{B9B}$ são apresentadas a seguir. São, no total, 1 peça e 14 decomposições. A complexidade deste esquema é 9 (a decomposição 14 tem complexidade máxima igual a 9). A única peça B tem complexidade 3. A partição de eixos associada a este esquema é qualquer uma.

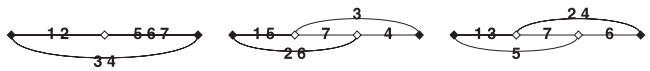
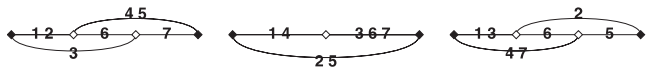
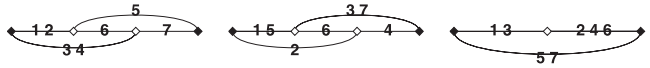
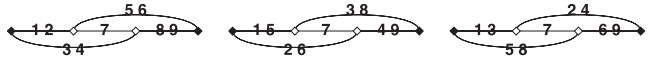
Em cada peça/decomposição a seguir, apresentamos o código e as reversões simétricas daquela peça/decomposição. Além disso, abaixo da representação geométrica de cada peça/decomposição, indicamos o nome da peça (no caso de uma simples peça) ou o nome da peça "pai" (no caso de uma decomposição) e, entre parênteses, a reversão (em decimal) que se encontra aquela peça ou peça "pai". Os vértices cheios (pretos) são fixos e os vértices vazados (brancos) são móveis.

C.1 Peças



C.2 Decomposições

1	 B(0)	 B(0)	3;2;2,2,3;0,0,0-1,1,1;0,0,1-1,1,2;1,2 Rev.Sim.: 01234567
2	 B(0)	 B(0)	3;2;2,3,2;0,0,0-1,1,1;0,1,0-1,2,1;1,2 Rev.Sim.: 01234567
3	 B(0)	 B(0)	3;2;3,2,2;0,0,0-1,1,1;1,0,0-2,1,1;1,2 Rev.Sim.: 01234567
4	 B(0)	 B(0)	3;5;2,4,4;0,0,0-1,1,2;0,0,2-1,2,3;0,1,0-1,3,1;0,1,1-1,2,2;0,2,1-1,3,3;1,2,3,4,5 Rev.Sim.: 0167 
5	 B(0)	 B(0)	3;5;3,3,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-2,2,1;1,0,0-2,1,2;1,1,1-2,2,2;1,2,3,4,5 Rev.Sim.: 0 
6	 B(0)	 B(0)	3;5;4,2,4;0,0,0-1,1,2;0,0,2-2,1,3;1,0,0-3,1,1;1,0,1-2,1,2;2,0,1-3,1,3;1,2,3,4,5 Rev.Sim.: 0257 
7	 B(0)	 B(0)	3;5;4,4,2;0,0,0-1,2,1;0,2,0-2,3,1;1,0,0-3,1,1;1,1,0-2,2,1;2,1,0-3,3,1;1,2,3,4,5 Rev.Sim.: 0347 
8	 B(0)	 B(0)	3;6;3,3,4;0,0,0-1,1,1;0,0,1-1,2,3;0,1,0-2,2,1;1,0,0-2,1,2;1,0,2-2,2,3;1,1,1-2,2,2;1,2,3,4,5,6 Rev.Sim.: 0 
9	 B(0)	 B(0)	3;6;3,4,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-2,3,1;0,2,1-2,3,2;1,0,0-2,1,2;1,1,1-2,2,2;1,2,3,4,5,6 Rev.Sim.: 0 
10	 B(0)	 B(0)	3;6;4,3,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-2,2,1;1,0,0-3,1,2;1,1,1-2,2,2;2,1,0-3,2,2;1,2,3,4,5,6 Rev.Sim.: 0 

11	 B(0)	 B(0)	3;7;3,4,4;0,0,0-1,1,1;0,0,1-1,2,3;0,1,0-2,3,1;0,2,1-2,3,3;1,0,0-2,1,2;1,0,2-2,2,3;1,1,1-2,2,2;1,2,3,4,5,6,7 Rev.Sim.: 0 
12	 B(0)	 B(0)	3;7;4,3,4;0,0,0-1,1,1;0,0,1-1,2,3;0,1,0-2,2,1;1,0,0-3,1,2;1,0,2-3,2,3;1,1,1-2,2,2;2,1,0-3,2,2;1,2,3,4,5,6,7 Rev.Sim.: 0 
13	 B(0)	 B(0)	3;7;4,4,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-2,3,1;0,2,1-2,3,2;1,0,0-3,1,2;1,1,1-2,2,2;2,1,0-3,3,2;1,2,3,4,5,6,7 Rev.Sim.: 0 
14	 B(0)	 B(0)	3;9;4,4,4;0,0,0-1,1,1;0,0,1-1,2,3;0,1,0-2,3,1;0,2,1-2,3,3;1,0,0-3,1,2;1,0,2-3,2,3;1,1,1-2,2,2;2,1,0-3,3,2;2,2,2-3,3,3;1,2,3,4,5,6,7,8,9 Rev.Sim.: 07 

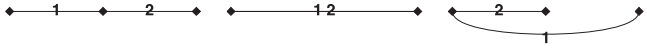
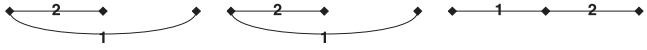
Apêndice D

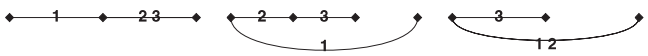
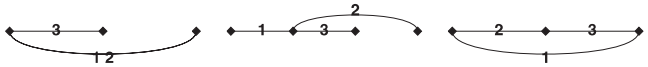
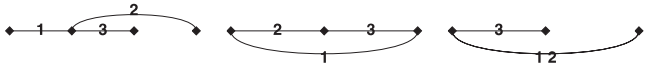
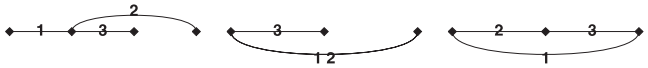
Gtets do esquema BALST

As peças e decomposições do esquema tridimensional BALST ou $\mathcal{E}_{BALST}$ são apresentadas a seguir. São, no total, 14 peças e 430 decomposições. A complexidade deste esquema é 9 (as decomposições de 419 a 430 têm complexidade máxima igual a 9). As peças T, as mais complexa do esquema, têm complexidade 7. A partição de eixos associada a este esquema é $\{\{1\},\{2\},\{3\}\}$, ou seja, nenhum eixo é equivalente. Se quisermos o esquema BALST com outra partição de eixos, este esquema resultante passa a ter como peças um subconjunto das peças (p-representantes das 14 peças bases, onde p é a partição de eixos desejada) do esquema aqui apresentado e as decomposições deste novo esquema são todas aquelas induzidas pelo subconjunto de peças obtidos. Por exemplo, se todos os eixos forem equivalentes, $p = \{\{1,2,3\}\}$, o número de peças passa de 14 para 5: as 3 peças L viram uma única peça L, as 3 peças A viram uma única peça A, as 6 peças T viram uma única peça T. Neste caso, o número de decomposições cai de 430 para 82.

Em cada peça/decomposição a seguir, apresentamos o código e as reversões simétricas daquela peça/decomposição. Além disso, abaixo da representação geométrica de cada peça/decomposição, indicamos o nome da peça (no caso de uma simples peça) ou o nome da peça "pai" (no caso de uma decomposição) e, entre parênteses, a reversão (em decimal) que se encontra aquela peça ou peça "pai". Os vértices cheios (pretos) são fixos e os vértices vazados (brancos) são móveis.

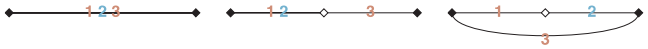
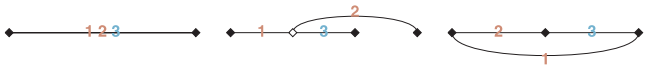
D.1 Peças

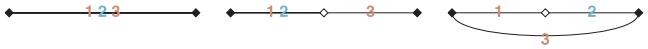
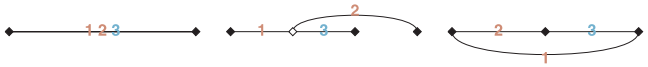
1	 **B(0)**	 **B(0)**	**3;1;2,2,2;0,0,0-1,1,1;1** **Rev.Sim.: 01234567**
2	 **Lx(0)**	 **Lx(0)**	**3;2;2,3,3;0,0,0-1,1,2;0,1,0-1,2,1;1,1** **Rev.Sim.: 01** 
3	 **Ly(0)**	 **Ly(0)**	**3;2;3,2,3;0,0,0-1,1,2;1,0,0-2,1,1;1,1** **Rev.Sim.: 02** 
4	 **Lz(0)**	 **Lz(0)**	**3;2;3,3,2;0,0,0-1,2,1;1,0,0-2,1,1;1,1** **Rev.Sim.: 04** 
5	 **Ax(0)**	 **Ax(0)**	**3;2;3,3,3;0,0,0-1,2,2;1,0,0-2,1,1;1,1** **Rev.Sim.: 0** 
6	 **Ay(0)**	 **Ay(0)**	**3;2;3,3,3;0,0,0-2,1,2;0,1,0-1,2,1;1,1** **Rev.Sim.: 0** 
7	 **Az(0)**	 **Az(0)**	**3;2;3,3,3;0,0,0-2,2,1;0,0,1-1,1,2;1,1** **Rev.Sim.: 0** 
8	 **S(0)**	 **S(0)**	**3;3;3,3,3;0,0,0-1,2,2;1,0,0-2,1,2;1,1,0-2,2,1;1,1,1** **Rev.Sim.: 0** 
9	 **Txz(0)**	 **Txz(0)**	**3;3;3,3,4;0,0,0-1,2,3;1,0,0-2,1,2;1,1,0-2,2,1;1,1,1** **Rev.Sim.: 0** 
10	 **Tyz(0)**	 **Tyz(0)**	**3;3;3,3,4;0,0,0-2,1,3;0,1,0-1,2,2;1,1,0-2,2,1;1,1,1** **Rev.Sim.: 0** 

11	 **Txy(0)**	 **Txy(0)**	**3;3;3,4,3;0,0,0-1,3,2;1,0,0-2,1,2;1,1,0-2,2,1;1,1,1** **Rev.Sim.: 0** 
12	 **Tzy(0)**	 **Tzy(0)**	**3;3;3,4,3;0,0,0-2,1,2;0,1,0-2,3,1;0,1,1-1,2,2;1,1,1** **Rev.Sim.: 0** 
13	 **Tyx(0)**	 **Tyx(0)**	**3;3;4,3,3;0,0,0-1,2,2;1,0,0-3,1,2;1,1,0-2,2,1;1,1,1** **Rev.Sim.: 0** 
14	 **Tzx(0)**	 **Tzx(0)**	**3;3;4,3,3;0,0,0-1,2,2;1,0,0-3,2,1;1,0,1-2,1,2;1,1,1** **Rev.Sim.: 0** 

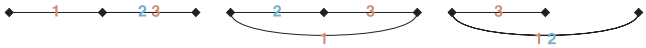
D.2 Decomposições

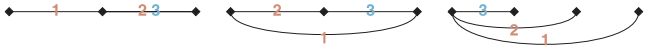
1	B(0)	B(0)	B(0)	B(0)	B(0)	B(0)	3;2;2,2,3;0,0,0-1,1,1;0,0,1-1,1,2;1,2 Rev.Sim.: 01234567
2	B(0)	B(0)	B(0)	B(0)	B(0)	B(0)	3;2;2,3,2;0,0,0-1,1,1;0,1,0-1,2,1;1,2 Rev.Sim.: 01234567
3	B(0)	B(0)	Lx(0)	B(0)	B(0)	Lx(0)	3;2;2,3,3;0,0,0-1,1,2;0,1,0-1,2,1;1,2 Rev.Sim.: 01
4	B(0)	B(0)	Lx(0)	B(0)	B(0)	Lx(0)	3;2;2,3,3;0,0,0-1,2,1;0,0,1-1,1,2;1,2 Rev.Sim.: 01
5	B(0)	B(0)	B(0)	B(0)	B(0)	B(0)	3;2;3,2,2;0,0,0-1,1,1;1,0,0-2,1,1;1,2 Rev.Sim.: 01234567
6	B(0)	B(0)	Ly(0)	B(0)	B(0)	Ly(0)	3;2;3,2,3;0,0,0-1,1,2;1,0,0-2,1,1;1,2 Rev.Sim.: 02
7	B(0)	B(0)	Ly(0)	B(0)	B(0)	Ly(0)	3;2;3,2,3;0,0,0-2,1,1;0,0,1-1,1,2;1,2 Rev.Sim.: 02
8	B(0)	B(0)	Lz(0)	B(0)	B(0)	Lz(0)	3;2;3,3,2;0,0,0-1,2,1;1,0,0-2,1,1;1,2 Rev.Sim.: 04
9	B(0)	B(0)	Lz(0)	B(0)	B(0)	Lz(0)	3;2;3,3,2;0,0,0-2,1,1;0,1,0-1,2,1;1,2 Rev.Sim.: 04
10	B(0)	B(0)	Ax(0)	B(0)	B(0)	Ax(0)	3;2;3,3,3;0,0,0-1,2,2;1,0,0-2,1,1;1,2 Rev.Sim.: 0
11	B(0)	B(0)	Ay(0)	B(0)	B(0)	Ay(0)	3;2;3,3,3;0,0,0-2,1,2;0,1,0-1,2,1;1,2 Rev.Sim.: 0
12	B(0)	B(0)	Az(0)	B(0)	B(0)	Az(0)	3;2;3,3,3;0,0,0-2,2,1;0,0,1-1,1,2;1,2 Rev.Sim.: 0

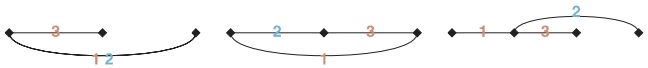
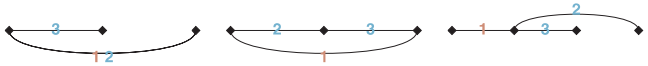
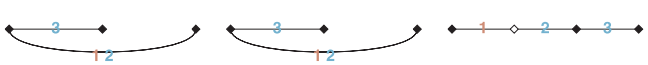
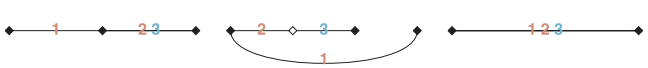
13	 Lx(3)	 B(0)	 B(0)	 Lx(3)	 B(0)	 B(0)	3;3;2,3,3;0,0,0-1,1,1;0,0,1-1,1,2;0,1,0-1,2,2;1,2,1 Rev.Sim.: 01 
14	 Lx(3)	 B(0)	 Lx(0)	 Lx(3)	 B(0)	 Lx(0)	3;3;2,3,4;0,0,0-1,1,1;0,0,1-1,1,3;0,1,0-1,2,2;1,2,1 Rev.Sim.: 01 
15	 Lx(0)	 B(0)	 Lx(0)	 Lx(0)	 B(0)	 Lx(0)	3;3;2,3,4;0,0,0-1,1,3;0,1,0-1,2,1;0,1,1-1,2,2;1,1,2 Rev.Sim.: 01 
16	 Lx(5)	 B(0)	 Lx(0)	 Lx(5)	 B(0)	 Lx(0)	3;3;2,4,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-1,3,1;1,1,2 Rev.Sim.: 01 
17	 Lx(0)	 B(0)	 Lx(0)	 Lx(0)	 B(0)	 Lx(0)	3;3;2,4,3;0,0,0-1,1,2;0,1,0-1,3,1;0,1,1-1,2,2;1,1,2 Rev.Sim.: 01 
18	 Ly(1)	 B(0)	 B(0)	 Ly(1)	 B(0)	B(0)	3;3;3,2,3;0,0,0-1,1,1;0,0,1-1,1,2;1,0,0-2,1,2;1,2,1 Rev.Sim.: 02
19	Ly(1)	B(0)	Ly(0)	Ly(1)	B(0)	Ly(0)	3;3;3,2,4;0,0,0-1,1,1;0,0,1-1,1,3;1,0,0-2,1,2;1,2,1 Rev.Sim.: 02
20	Ly(0)	B(0)	Ly(0)	Ly(0)	B(0)	Ly(0)	3;3;3,2,4;0,0,0-1,1,3;1,0,0-2,1,1;1,0,1-2,1,2;1,1,2 Rev.Sim.: 02
21	Lz(1)	B(0)	B(0)	Lz(1)	B(0)	B(0)	3;3;3,3,2;0,0,0-1,1,1;0,1,0-1,2,1;1,0,0-2,2,1;1,2,1 Rev.Sim.: 04
22	Ly(0)	B(0)	Ax(0)	Ly(0)	B(0)	Ax(0)	3;3;3,3,3;0,0,0-1,1,2;0,1,0-1,2,2;1,0,0-2,1,1;1,2,1 Rev.Sim.: 0
23	Lz(0)	B(0)	Ax(0)	Lz(0)	B(0)	Ax(0)	3;3;3,3,3;0,0,0-1,2,1;0,0,1-1,2,2;1,0,0-2,1,1;1,2,1 Rev.Sim.: 0
24	B(0)	Lx(6)	Ax(0)	B(0)	Lx(6)	Ax(0)	3;3;3,3,3;0,0,0-2,1,1;0,0,1-1,1,2;0,1,0-1,2,2;1,2,2 Rev.Sim.: 0

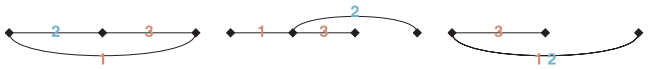
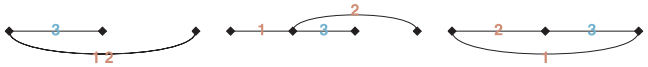
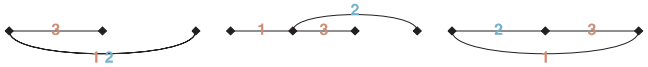
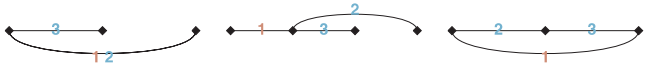
13	 Lx(3)	 B(0)	 B(0)	 Lx(3)	 B(0)	 B(0)	3;3;2,3,3;0,0,0-1,1,1;0,0,1-1,1,2;0,1,0-1,2,2;1,2,1 Rev.Sim.: 01 
14	 Lx(3)	 B(0)	 Lx(0)	 Lx(3)	 B(0)	 Lx(0)	3;3;2,3,4;0,0,0-1,1,1;0,0,1-1,1,3;0,1,0-1,2,2;1,2,1 Rev.Sim.: 01 
15	 Lx(0)	 B(0)	 Lx(0)	 Lx(0)	 B(0)	 Lx(0)	3;3;2,3,4;0,0,0-1,1,3;0,1,0-1,2,1;0,1,1-1,2,2;1,1,2 Rev.Sim.: 01 
16	 Lx(5)	 B(0)	 Lx(0)	 Lx(5)	 B(0)	 Lx(0)	3;3;2,4,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-1,3,1;1,1,2 Rev.Sim.: 01 
17	 Lx(0)	 B(0)	 Lx(0)	 Lx(0)	 B(0)	 Lx(0)	3;3;2,4,3;0,0,0-1,1,2;0,1,0-1,3,1;0,1,1-1,2,2;1,1,2 Rev.Sim.: 01 
18	 Ly(1)	 B(0)	 B(0)	 Ly(1)	 B(0)	B(0)	3;3;3,2,3;0,0,0-1,1,1;0,0,1-1,1,2;1,0,0-2,1,2;1,2,1 Rev.Sim.: 02
19	Ly(1)	B(0)	Ly(0)	Ly(1)	B(0)	Ly(0)	3;3;3,2,4;0,0,0-1,1,1;0,0,1-1,1,3;1,0,0-2,1,2;1,2,1 Rev.Sim.: 02
20	Ly(0)	B(0)	Ly(0)	Ly(0)	B(0)	Ly(0)	3;3;3,2,4;0,0,0-1,1,3;1,0,0-2,1,1;1,0,1-2,1,2;1,1,2 Rev.Sim.: 02
21	Lz(1)	B(0)	B(0)	Lz(1)	B(0)	B(0)	3;3;3,3,2;0,0,0-1,1,1;0,1,0-1,2,1;1,0,0-2,2,1;1,2,1 Rev.Sim.: 04
22	Ly(0)	B(0)	Ax(0)	Ly(0)	B(0)	Ax(0)	3;3;3,3,3;0,0,0-1,1,2;0,1,0-1,2,2;1,0,0-2,1,1;1,2,1 Rev.Sim.: 0
23	Lz(0)	B(0)	Ax(0)	Lz(0)	B(0)	Ax(0)	3;3;3,3,3;0,0,0-1,2,1;0,0,1-1,2,2;1,0,0-2,1,1;1,2,1 Rev.Sim.: 0
24	B(0)	Lx(6)	Ax(0)	B(0)	Lx(6)	Ax(0)	3;3;3,3,3;0,0,0-2,1,1;0,0,1-1,1,2;0,1,0-1,2,2;1,2,2 Rev.Sim.: 0

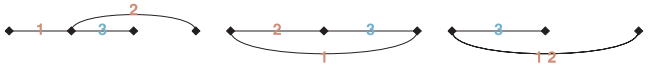
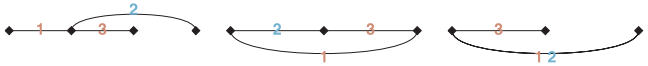
25	 Lx(0)	 B(0)	 Ay(0)	 Lx(0)	 B(0)	 Ay(0)	3;3;3,3,3;0,0,0-1,1,2;0,1,0-1,2,1;1,0,0-2,1,2;1,1,2 Rev.Sim.: 0 
26	 B(0)	 Ly(7)	 Ay(0)	 B(0)	 Ly(7)	 Ay(0)	3;3;3,3,3;0,0,0-1,2,1;0,0,1-1,1,2;1,0,0-2,1,2;1,2,2 Rev.Sim.: 0 
27	 Lz(0)	 B(0)	 Ay(0)	 Lz(0)	 B(0)	 Ay(0)	3;3;3,3,3;0,0,0-1,2,1;0,0,1-2,1,2;1,0,0-2,1,1;1,2,1 Rev.Sim.: 0 
28	 Lx(0)	 B(0)	 Az(0)	 Lx(0)	 B(0)	 Az(0)	3;3;3,3,3;0,0,0-1,1,2;0,1,0-1,2,1;1,0,0-2,2,1;1,1,2 Rev.Sim.: 0 
29	 B(0)	 Lz(3)	 Az(0)	 B(0)	 Lz(3)	 Az(0)	3;3;3,3,3;0,0,0-1,1,2;0,1,0-1,2,1;1,0,0-2,2,1;1,2,2 Rev.Sim.: 0 
30	 Ly(0)	 B(0)	 Az(0)	 Ly(0)	 B(0)	Az(0)	3;3;3,3,3;0,0,0-1,1,2;0,1,0-2,2,1;1,0,0-2,1,1;1,2,1 Rev.Sim.: 0
31	Ax(4)	B(0)	Lz(0)	Ax(4)	B(0)	Lz(0)	3;3;3,3,3;0,0,0-1,2,2;1,0,0-2,1,1;1,0,1-2,1,2;1,2,1 Rev.Sim.: 0
32	Ay(4)	B(0)	Lz(0)	Ay(4)	B(0)	Lz(0)	3;3;3,3,3;0,0,0-2,1,2;0,1,0-1,2,1;0,1,1-1,2,2;1,2,1 Rev.Sim.: 0
33	Ax(2)	B(0)	Ly(0)	Ax(2)	B(0)	Ly(0)	3;3;3,3,3;0,0,0-1,2,2;1,0,0-2,1,1;1,1,0-2,2,1;1,2,1 Rev.Sim.: 0
34	Az(0)	B(0)	Ly(0)	Az(0)	B(0)	Ly(0)	3;3;3,3,3;0,0,0-2,2,1;0,0,1-1,1,2;0,1,1-1,2,2;1,1,2 Rev.Sim.: 0
35	Ay(0)	B(0)	Lx(0)	Ay(0)	B(0)	Lx(0)	3;3;3,3,3;0,0,0-2,1,2;0,1,0-1,2,1;1,1,0-2,2,1;1,1,2 Rev.Sim.: 0
36	Az(0)	B(0)	Lx(0)	Az(0)	B(0)	Lx(0)	3;3;3,3,3;0,0,0-2,2,1;0,0,1-1,1,2;1,0,1-2,1,2;1,1,2 Rev.Sim.: 0

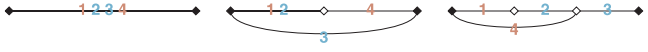
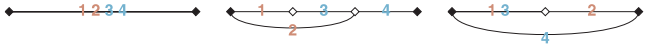
37	 B(0)	 Lx(0)	 S(0)	 B(0)	 Lx(0)	 S(0)	3;3;3,3,3;0,0,0-1,2,2;1,0,0-2,1,2;1,1,0-2,2,1;1,2,2 Rev.Sim.: 0 
38	 Ax(2)	 B(0)	 S(0)	 Ax(2)	 B(0)	 S(0)	3;3;3,3,3;0,0,0-1,2,2;1,0,0-2,1,2;1,1,0-2,2,1;1,2,1 Rev.Sim.: 0 
39	 Lz(0)	 B(0)	 S(0)	 Lz(0)	 B(0)	 S(0)	3;3;3,3,3;0,0,0-1,2,2;1,0,0-2,1,2;1,1,0-2,2,1;1,1,2 Rev.Sim.: 0 
40	 Ax(4)	 B(0)	 S(0)	 Ax(4)	 B(0)	 S(0)	3;3;3,3,3;0,0,0-1,2,2;1,0,0-2,2,1;1,0,1-2,1,2;1,2,1 Rev.Sim.: 0 
41	 Ly(0)	 B(0)	 S(0)	 Ly(0)	 B(0)	 S(0)	3;3;3,3,3;0,0,0-1,2,2;1,0,0-2,2,1;1,0,1-2,1,2;1,1,2 Rev.Sim.: 0 
42	 B(0)	 Ly(0)	 S(0)	 B(0)	 Ly(0)	S(0)	3;3;3,3,3;0,0,0-2,1,2;0,1,0-1,2,2;1,1,0-2,2,1;1,2,2 Rev.Sim.: 0
43	Ay(1)	B(0)	S(0)	Ay(1)	B(0)	S(0)	3;3;3,3,3;0,0,0-2,1,2;0,1,0-1,2,2;1,1,0-2,2,1;1,2,1 Rev.Sim.: 0
44	Ay(4)	B(0)	S(0)	Ay(4)	B(0)	S(0)	3;3;3,3,3;0,0,0-2,1,2;0,1,0-2,2,1;0,1,1-1,2,2;1,2,1 Rev.Sim.: 0
45	Lx(0)	B(0)	S(0)	Lx(0)	B(0)	S(0)	3;3;3,3,3;0,0,0-2,1,2;0,1,0-2,2,1;0,1,1-1,2,2;1,1,2 Rev.Sim.: 0
46	B(0)	Lz(0)	S(0)	B(0)	Lz(0)	S(0)	3;3;3,3,3;0,0,0-2,2,1;0,0,1-1,2,2;1,0,1-2,1,2;1,2,2 Rev.Sim.: 0
47	Az(1)	B(0)	S(0)	Az(1)	B(0)	S(0)	3;3;3,3,3;0,0,0-2,2,1;0,0,1-1,2,2;1,0,1-2,1,2;1,2,1 Rev.Sim.: 0
48	Az(2)	B(0)	S(0)	Az(2)	B(0)	S(0)	3;3;3,3,3;0,0,0-2,2,1;0,0,1-2,1,2;0,1,1-1,2,2;1,2,1 Rev.Sim.: 0

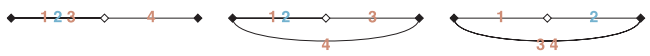
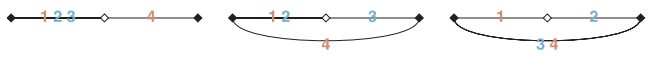
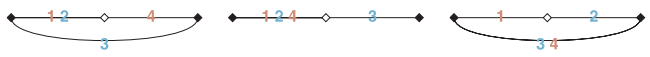
49	 Ax(0)	 B(0)	 Ax(0)	 Ax(0)	 B(0)	 Ax(0)	3;3;3,3,4;0,0,0-1,2,3;1,0,0-2,1,1;1,0,1-2,1,2;1,1,2 Rev.Sim.: 0 
50	 Ay(0)	 B(0)	 Ay(0)	 Ay(0)	 B(0)	 Ay(0)	3;3;3,3,4;0,0,0-2,1,3;0,1,0-1,2,1;0,1,1-1,2,2;1,1,2 Rev.Sim.: 0 
51	 B(0)	 Lx(0)	 Txz(0)	 B(0)	 Lx(0)	 Txz(0)	3;3;3,3,4;0,0,0-1,2,3;1,0,0-2,1,2;1,1,0-2,2,1;1,2,2 Rev.Sim.: 0 
52	 Ax(2)	 B(0)	 Txz(0)	 Ax(2)	 B(0)	 Txz(0)	3;3;3,3,4;0,0,0-1,2,3;1,0,0-2,1,2;1,1,0-2,2,1;1,2,1 Rev.Sim.: 0 
53	 Ax(0)	 B(0)	 Txz(0)	 Ax(0)	 B(0)	 Txz(0)	3;3;3,3,4;0,0,0-1,2,3;1,0,0-2,1,2;1,1,0-2,2,1;1,1,2 Rev.Sim.: 0 
54	 Ly(0)	 B(0)	 Txz(0)	 Ly(0)	 B(0)	Txz(0)	3;3;3,3,4;0,0,0-1,2,3;1,0,0-2,2,1;1,0,1-2,1,2;1,1,2 Rev.Sim.: 0
55	Az(1)	B(0)	Txz(0)	Az(1)	B(0)	Txz(0)	3;3;3,3,4;0,0,0-2,2,1;0,0,1-1,2,3;1,0,1-2,1,2;1,2,1 Rev.Sim.: 0
56	B(0)	Ax(0)	Txz(0)	B(0)	Ax(0)	Txz(0)	3;3;3,3,4;0,0,0-2,2,1;0,0,1-1,2,3;1,0,1-2,1,2;1,2,2 Rev.Sim.: 0
57	Ay(1)	B(0)	Tyz(0)	Ay(1)	B(0)	Tyz(0)	3;3;3,3,4;0,0,0-2,1,3;0,1,0-1,2,2;1,1,0-2,2,1;1,2,1 Rev.Sim.: 0
58	B(0)	Ly(0)	Tyz(0)	B(0)	Ly(0)	Tyz(0)	3;3;3,3,4;0,0,0-2,1,3;0,1,0-1,2,2;1,1,0-2,2,1;1,2,2 Rev.Sim.: 0
59	Ay(0)	B(0)	Tyz(0)	Ay(0)	B(0)	Tyz(0)	3;3;3,3,4;0,0,0-2,1,3;0,1,0-1,2,2;1,1,0-2,2,1;1,1,2 Rev.Sim.: 0
60	Lx(0)	B(0)	Tyz(0)	Lx(0)	B(0)	Tyz(0)	3;3;3,3,4;0,0,0-2,1,3;0,1,0-2,2,1;0,1,1-1,2,2;1,1,2 Rev.Sim.: 0

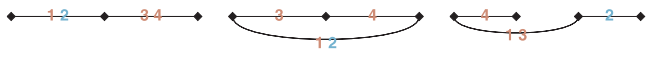
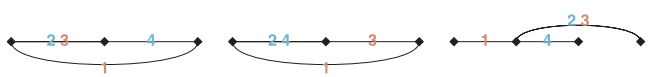
61	 Az(2)	 B(0)	 Tyz(0)	 Az(2)	 B(0)	 Tyz(0)	3;3;3,3,4;0,0,0-2,2,1;0,0,1-2,1,3;0,1,1-1,2,2;1,2,1 Rev.Sim.: 0 
62	 B(0)	 Ay(0)	 Tyz(0)	 B(0)	 Ay(0)	 Tyz(0)	3;3;3,3,4;0,0,0-2,2,1;0,0,1-2,1,3;0,1,1-1,2,2;1,2,2 Rev.Sim.: 0 
63	 B(0)	 Az(0)	 Az(0)	 B(0)	 Az(0)	 Az(0)	3;3;3,3,4;0,0,0-2,2,1;0,0,1-2,2,2;0,0,2-1,1,3;1,2,2 Rev.Sim.: 0 
64	 Lz(1)	 B(0)	 Lz(0)	 Lz(1)	 B(0)	 Lz(0)	3;3;3,4,2;0,0,0-1,1,1;0,1,0-1,3,1;1,0,0-2,2,1;1,2,1 Rev.Sim.: 04 
65	 Lz(0)	 B(0)	 Lz(0)	 Lz(0)	 B(0)	 Lz(0)	3;3;3,4,2;0,0,0-1,3,1;1,0,0-2,1,1;1,1,0-2,2,1;1,1,2 Rev.Sim.: 04 
66	 Ax(0)	 B(0)	 Ax(0)	 Ax(0)	 B(0)	Ax(0)	3;3;3,4,3;0,0,0-1,3,2;1,0,0-2,1,1;1,1,0-2,2,1;1,1,2 Rev.Sim.: 0
67	Az(0)	B(0)	Az(0)	Az(0)	B(0)	Az(0)	3;3;3,4,3;0,0,0-2,3,1;0,0,1-1,1,2;0,1,1-1,2,2;1,1,2 Rev.Sim.: 0
68	B(0)	Lx(0)	Txy(0)	B(0)	Lx(0)	Txy(0)	3;3;3,4,3;0,0,0-1,3,2;1,0,0-2,1,2;1,1,0-2,2,1;1,2,2 Rev.Sim.: 0
69	Lz(0)	B(0)	Txy(0)	Lz(0)	B(0)	Txy(0)	3;3;3,4,3;0,0,0-1,3,2;1,0,0-2,1,2;1,1,0-2,2,1;1,1,2 Rev.Sim.: 0
70	Ax(4)	B(0)	Txy(0)	Ax(4)	B(0)	Txy(0)	3;3;3,4,3;0,0,0-1,3,2;1,0,0-2,2,1;1,0,1-2,1,2;1,2,1 Rev.Sim.: 0
71	Ax(0)	B(0)	Txy(0)	Ax(0)	B(0)	Txy(0)	3;3;3,4,3;0,0,0-1,3,2;1,0,0-2,2,1;1,0,1-2,1,2;1,1,2 Rev.Sim.: 0
72	B(0)	Ax(0)	Txy(0)	B(0)	Ax(0)	Txy(0)	3;3;3,4,3;0,0,0-2,1,2;0,1,0-1,3,2;1,1,0-2,2,1;1,2,2 Rev.Sim.: 0

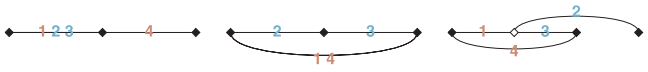
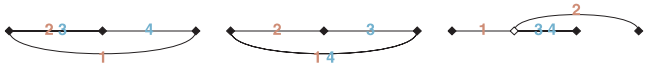
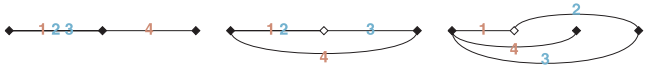
73	 Ay(1)	 B(0)	 Txy(0)	 Ay(1)	 B(0)	 Txy(0)	3;3;3,4,3;0,0,0-2,1,2;0,1,0-1,3,2;1,1,0-2,2,1;1,2,1 Rev.Sim.: 0 
74	 B(0)	 Ay(0)	 Ay(0)	 B(0)	 Ay(0)	 Ay(0)	3;3;3,4,3;0,0,0-2,1,2;0,1,0-2,2,2;0,2,0-1,3,1;1,2,2 Rev.Sim.: 0 
75	 Lx(0)	 B(0)	 Tzy(0)	 Lx(0)	 B(0)	 Tzy(0)	3;3;3,4,3;0,0,0-2,1,2;0,1,0-2,3,1;0,1,1-1,2,2;1,1,2 Rev.Sim.: 0 
76	 Ay(4)	 B(0)	 Tzy(0)	 Ay(4)	 B(0)	 Tzy(0)	3;3;3,4,3;0,0,0-2,1,2;0,1,0-2,3,1;0,1,1-1,2,2;1,2,1 Rev.Sim.: 0 
77	 B(0)	 Az(0)	 Tzy(0)	 B(0)	 Az(0)	 Tzy(0)	3;3;3,4,3;0,0,0-2,1,2;0,1,0-2,3,1;0,1,1-1,2,2;1,2,2 Rev.Sim.: 0 
78	 Az(1)	 B(0)	 Tzy(0)	 Az(1)	 B(0)	Tzy(0)	3;3;3,4,3;0,0,0-2,3,1;0,0,1-1,2,2;1,0,1-2,1,2;1,2,1 Rev.Sim.: 0
79	B(0)	Lz(0)	Tzy(0)	B(0)	Lz(0)	Tzy(0)	3;3;3,4,3;0,0,0-2,3,1;0,0,1-1,2,2;1,0,1-2,1,2;1,2,2 Rev.Sim.: 0
80	Az(0)	B(0)	Tzy(0)	Az(0)	B(0)	Tzy(0)	3;3;3,4,3;0,0,0-2,3,1;0,0,1-1,2,2;1,0,1-2,1,2;1,1,2 Rev.Sim.: 0
81	Ly(6)	B(0)	Ly(0)	Ly(6)	B(0)	Ly(0)	3;3;4,2,3;0,0,0-1,1,1;0,0,1-2,1,2;1,0,0-3,1,1;1,1,2 Rev.Sim.: 02
82	Ly(0)	B(0)	Ly(0)	Ly(0)	B(0)	Ly(0)	3;3;4,2,3;0,0,0-1,1,2;1,0,0-3,1,1;1,0,1-2,1,2;1,1,2 Rev.Sim.: 02
83	Lz(2)	B(0)	Lz(0)	Lz(2)	B(0)	Lz(0)	3;3;4,3,2;0,0,0-1,1,1;0,1,0-2,2,1;1,0,0-3,1,1;1,1,2 Rev.Sim.: 04
84	Lz(0)	B(0)	Lz(0)	Lz(0)	B(0)	Lz(0)	3;3;4,3,2;0,0,0-1,2,1;1,0,0-3,1,1;1,1,0-2,2,1;1,1,2 Rev.Sim.: 04

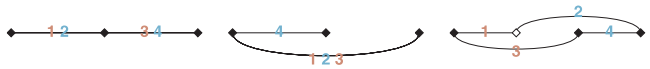
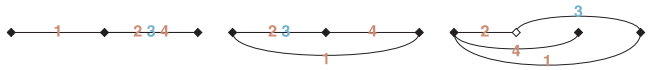
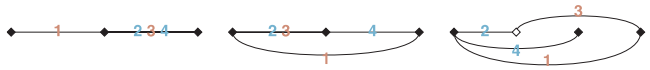
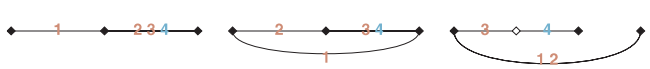
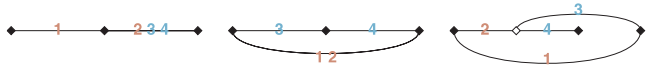
85	 B(0)	 Ax(0)	 Ax(0)	 B(0)	 Ax(0)	 Ax(0)	3;3;4,3,3;0,0,0-1,2,2;1,0,0-2,2,2;2,0,0-3,1,1;1,2,2 Rev.Sim.: 0 
86	 Ay(0)	 B(0)	 Ay(0)	 Ay(0)	 B(0)	 Ay(0)	3;3;4,3,3;0,0,0-3,1,2;0,1,0-1,2,1;1,1,0-2,2,1;1,1,2 Rev.Sim.: 0 
87	 Az(0)	 B(0)	 Az(0)	 Az(0)	 B(0)	 Az(0)	3;3;4,3,3;0,0,0-3,2,1;0,0,1-1,1,2;1,0,1-2,1,2;1,1,2 Rev.Sim.: 0 
88	 Lz(0)	 B(0)	 Tyx(0)	 Lz(0)	 B(0)	 Tyx(0)	3;3;4,3,3;0,0,0-1,2,2;1,0,0-3,1,2;1,1,0-2,2,1;1,1,2 Rev.Sim.: 0 
89	 Ax(2)	 B(0)	 Tyx(0)	 Ax(2)	 B(0)	 Tyx(0)	3;3;4,3,3;0,0,0-1,2,2;1,0,0-3,1,2;1,1,0-2,2,1;1,2,1 Rev.Sim.: 0 
90	 B(0)	 Ay(0)	 Tyx(0)	 B(0)	 Ay(0)	Tyx(0)	3;3;4,3,3;0,0,0-1,2,2;1,0,0-3,1,2;1,1,0-2,2,1;1,2,2 Rev.Sim.: 0
91	B(0)	Ly(0)	Tyx(0)	B(0)	Ly(0)	Tyx(0)	3;3;4,3,3;0,0,0-3,1,2;0,1,0-1,2,2;1,1,0-2,2,1;1,2,2 Rev.Sim.: 0
92	Ay(4)	B(0)	Tyx(0)	Ay(4)	B(0)	Tyx(0)	3;3;4,3,3;0,0,0-3,1,2;0,1,0-2,2,1;0,1,1-1,2,2;1,2,1 Rev.Sim.: 0
93	Ay(0)	B(0)	Tyx(0)	Ay(0)	B(0)	Tyx(0)	3;3;4,3,3;0,0,0-3,1,2;0,1,0-2,2,1;0,1,1-1,2,2;1,1,2 Rev.Sim.: 0
94	Ly(0)	B(0)	Tzx(0)	Ly(0)	B(0)	Tzx(0)	3;3;4,3,3;0,0,0-1,2,2;1,0,0-3,2,1;1,0,1-2,1,2;1,1,2 Rev.Sim.: 0
95	Ax(4)	B(0)	Tzx(0)	Ax(4)	B(0)	Tzx(0)	3;3;4,3,3;0,0,0-1,2,2;1,0,0-3,2,1;1,0,1-2,1,2;1,2,1 Rev.Sim.: 0
96	B(0)	Az(0)	Tzx(0)	B(0)	Az(0)	Tzx(0)	3;3;4,3,3;0,0,0-1,2,2;1,0,0-3,2,1;1,0,1-2,1,2;1,2,2 Rev.Sim.: 0

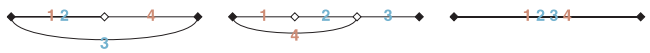
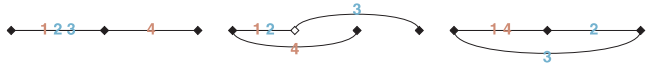
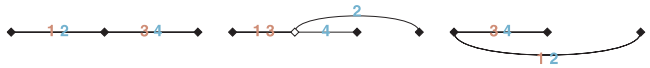
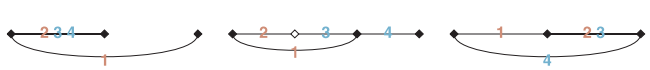
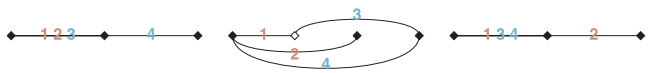
97	 B(0)	 Lz(0)	 Tzx(0)	 B(0)	 Lz(0)	 Tzx(0)	3;3;4,3,3;0,0,0-3,2,1;0,0,1-1,2,2;1,0,1-2,1,2;1,2,2 Rev.Sim.: 0 
98	 Az(2)	 B(0)	 Tzx(0)	 Az(2)	 B(0)	 Tzx(0)	3;3;4,3,3;0,0,0-3,2,1;0,0,1-2,1,2;0,1,1-1,2,2;1,2,1 Rev.Sim.: 0 
99	 Az(0)	 B(0)	 Tzx(0)	 Az(0)	 B(0)	 Tzx(0)	3;3;4,3,3;0,0,0-3,2,1;0,0,1-2,1,2;0,1,1-1,2,2;1,1,2 Rev.Sim.: 0 
100	 Lx(3)	 Lx(5)	 B(0)	 Lx(3)	 Lx(5)	 B(0)	3;4;2,3,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-1,2,3;0,1,0-1,2,2;1,2,2,1 Rev.Sim.: 0167 
101	 Lx(5)	 Lx(3)	 B(0)	 Lx(5)	 Lx(3)	 B(0)	3;4;2,4,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-1,2,1;0,2,0-1,3,2;1,1,2,2 Rev.Sim.: 0167 
102	 Lx(3)	 Lx(5)	 Lx(0)	 Lx(3)	 Lx(5)	Lx(0)	3;4;2,4,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-1,2,3;0,1,0-1,3,2;1,2,2,1 Rev.Sim.: 01
103	Lx(5)	Lx(3)	Lx(0)	Lx(5)	Lx(3)	Lx(0)	3;4;2,4,4;0,0,0-1,1,1;0,0,1-1,2,3;0,1,0-1,2,1;0,2,0-1,3,2;1,1,2,2 Rev.Sim.: 01
104	Ly(1)	Ly(6)	B(0)	Ly(1)	Ly(6)	B(0)	3;4;3,2,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-2,1,3;1,0,0-2,1,2;1,2,2,1 Rev.Sim.: 0257
105	Ly(1)	Lx(6)	Lz(0)	Ly(1)	Lx(6)	Lz(0)	3;4;3,3,3;0,0,0-1,1,1;0,0,1-1,1,2;0,1,0-1,2,2;1,0,0-2,1,2;1,2,2,1 Rev.Sim.: 0
106	Lz(0)	Lz(0)	Lz(0)	Lz(0)	Lz(0)	Lz(0)	3;4;3,3,3;0,0,0-1,2,1;0,0,1-1,2,2;1,0,0-2,1,1;1,0,1-2,1,2;1,2,1,2 Rev.Sim.: 04
107	Lz(1)	Lx(6)	Ly(0)	Lz(1)	Lx(6)	Ly(0)	3;4;3,3,3;0,0,0-1,1,1;0,0,1-1,1,2;0,1,0-1,2,2;1,0,0-2,2,1;1,2,2,1 Rev.Sim.: 0
108	Ly(0)	Ly(0)	Ly(0)	Ly(0)	Ly(0)	Ly(0)	3;4;3,3,3;0,0,0-1,1,2;0,1,0-1,2,2;1,0,0-2,1,1;1,1,0-2,2,1;1,2,1,2 Rev.Sim.: 02

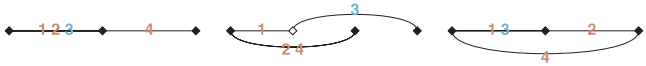
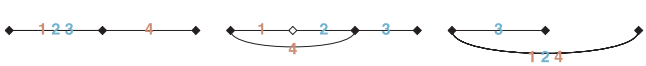
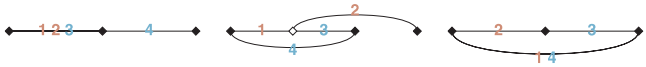
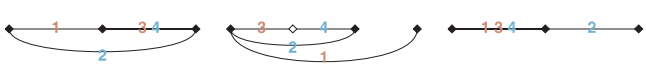
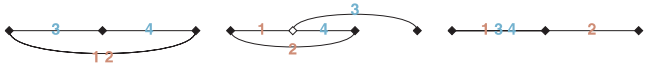
109	 Lz(2)	 Ly(7)	 Lx(0)	 Lz(2)	 Ly(7)	 Lx(0)	3;4;3,3,3;0,0,0-1,1,1;0,0,1-1,1,2;0,1,0-2,2,1;1,0,0-2,1,2;1,2,1,2 Rev.Sim.: 0 
110	 Lx(0)	 Lx(0)	 Lx(0)	 Lx(0)	 Lx(0)	 Lx(0)	3;4;3,3,3;0,0,0-1,1,2;0,1,0-1,2,1;1,0,0-2,1,2;1,1,0-2,2,1;1,1,2,2 Rev.Sim.: 01 
111	 S(3)	 B(0)	 B(0)	 S(3)	 B(0)	 B(0)	3;4;3,3,3;0,0,0-1,1,1;0,0,1-1,1,2;0,1,0-1,2,2;1,0,0-2,2,2;1,2,1,1 Rev.Sim.: 0 
112	 Ax(1)	 Lx(6)	 B(0)	 Ax(1)	 Lx(6)	 B(0)	3;4;3,3,3;0,0,0-1,1,1;0,0,1-1,1,2;0,1,0-1,2,2;1,0,0-2,2,2;1,2,2,1 Rev.Sim.: 0 
113	 Ly(1)	 Ay(6)	 B(0)	 Ly(1)	 Ay(6)	 B(0)	3;4;3,3,3;0,0,0-1,1,1;0,0,1-1,1,2;0,1,0-2,2,2;1,0,0-2,1,2;1,2,2,1 Rev.Sim.: 0 
114	 Lz(1)	 Az(6)	 B(0)	 Lz(1)	 Az(6)	B(0)	3;4;3,3,3;0,0,0-1,1,1;0,0,1-2,2,2;0,1,0-1,2,1;1,0,0-2,2,1;1,2,2,1 Rev.Sim.: 0
115	Ly(1)	Lx(6)	Ax(0)	Ly(1)	Lx(6)	Ax(0)	3;4;3,3,4;0,0,0-1,1,1;0,0,1-1,1,3;0,1,0-1,2,3;1,0,0-2,1,2;1,2,2,1 Rev.Sim.: 0
116	Lz(0)	Ax(0)	Ax(0)	Lz(0)	Ax(0)	Ax(0)	3;4;3,3,4;0,0,0-1,2,1;0,0,1-1,2,3;1,0,0-2,1,1;1,0,1-2,1,2;1,2,1,2 Rev.Sim.: 0
117	Ay(0)	Lx(6)	Ax(0)	Ay(0)	Lx(6)	Ax(0)	3;4;3,3,4;0,0,0-2,1,2;0,0,2-1,1,3;0,1,0-1,2,1;0,1,1-1,2,3;1,2,1,2 Rev.Sim.: 0
118	Lx(3)	Ly(7)	Ay(0)	Lx(3)	Ly(7)	Ay(0)	3;4;3,3,4;0,0,0-1,1,1;0,0,1-1,1,3;0,1,0-1,2,2;1,0,0-2,1,3;1,2,1,2 Rev.Sim.: 0
119	Lz(0)	Ay(0)	Ay(0)	Lz(0)	Ay(0)	Ay(0)	3;4;3,3,4;0,0,0-1,2,1;0,0,1-2,1,3;0,1,1-1,2,2;1,0,0-2,1,1;1,2,2,1 Rev.Sim.: 0
120	Ax(0)	Ly(7)	Ay(0)	Ax(0)	Ly(7)	Ay(0)	3;4;3,3,4;0,0,0-1,2,2;0,0,2-1,1,3;1,0,0-2,1,1;1,0,1-2,1,3;1,2,1,2 Rev.Sim.: 0

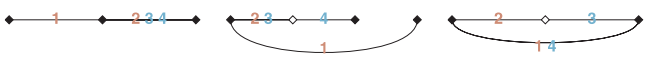
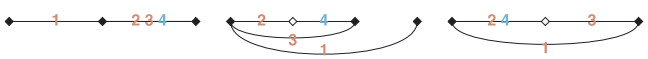
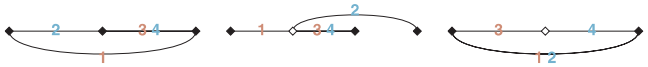
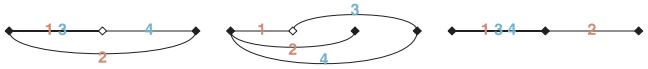
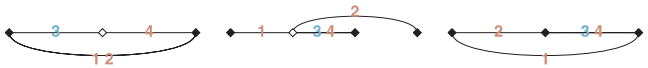
121	 Ly(0)	 Ly(0)	 Txz(0)	 Ly(0)	 Ly(0)	 Txz(0)	3;4;3,3,4;0,0,0-1,1,3;0,1,0-1,2,3;1,0,0-2,1,2;1,1,0-2,2,1;1,2,1,2 Rev.Sim.: 0 
122	 S(0)	 B(0)	 Txz(0)	 S(0)	 B(0)	 Txz(0)	3;4;3,3,4;0,0,0-1,2,2;0,0,2-1,2,3;1,0,0-2,1,2;1,1,0-2,2,1;1,2,1,1 Rev.Sim.: 0 
123	 Ay(1)	 Lx(6)	 Txz(0)	 Ay(1)	 Lx(6)	 Txz(0)	3;4;3,3,4;0,0,0-2,1,2;0,0,2-1,1,3;0,1,0-1,2,3;1,1,0-2,2,1;1,2,2,1 Rev.Sim.: 0 
124	 Lx(0)	 Lx(6)	 Txz(0)	 Lx(0)	 Lx(6)	 Txz(0)	3;4;3,3,4;0,0,0-2,1,2;0,0,2-1,1,3;0,1,0-2,2,1;0,1,1-1,2,3;1,2,1,2 Rev.Sim.: 0 
125	 Az(2)	 Ly(0)	 Txz(0)	 Az(2)	 Ly(0)	 Txz(0)	3;4;3,3,4;0,0,0-2,2,1;0,0,1-1,1,3;0,1,1-1,2,3;1,0,1-2,1,2;1,2,1,2 Rev.Sim.: 0 
126	 Lx(0)	 Lx(0)	 Tyz(0)	 Lx(0)	 Lx(0)	Tyz(0)	3;4;3,3,4;0,0,0-1,1,3;0,1,0-1,2,2;1,0,0-2,1,3;1,1,0-2,2,1;1,1,2,2 Rev.Sim.: 0
127	Ax(2)	Ly(7)	Tyz(0)	Ax(2)	Ly(7)	Tyz(0)	3;4;3,3,4;0,0,0-1,2,2;0,0,2-1,1,3;1,0,0-2,1,3;1,1,0-2,2,1;1,2,2,1 Rev.Sim.: 0
128	Ly(0)	Ly(7)	Tyz(0)	Ly(0)	Ly(7)	Tyz(0)	3;4;3,3,4;0,0,0-1,2,2;0,0,2-1,1,3;1,0,0-2,2,1;1,0,1-2,1,3;1,2,1,2 Rev.Sim.: 0
129	S(0)	B(0)	Tyz(0)	S(0)	B(0)	Tyz(0)	3;4;3,3,4;0,0,0-1,2,2;0,0,2-2,1,3;1,0,0-2,1,2;1,1,0-2,2,1;1,2,1,1 Rev.Sim.: 0
130	Az(1)	Lx(0)	Tyz(0)	Az(1)	Lx(0)	Tyz(0)	3;4;3,3,4;0,0,0-2,2,1;0,0,1-1,1,3;0,1,1-1,2,2;1,0,1-2,1,3;1,2,2,1 Rev.Sim.: 0
131	S(3)	B(0)	Az(0)	S(3)	B(0)	Az(0)	3;4;3,3,4;0,0,0-1,1,1;0,0,1-1,1,3;0,1,0-1,2,2;1,0,0-2,2,2;1,2,1,1 Rev.Sim.: 0
132	Lx(0)	Ax(7)	Az(0)	Lx(0)	Ax(7)	Az(0)	3;4;3,3,4;0,0,0-1,1,3;0,1,0-1,2,1;0,1,1-1,2,2;1,0,0-2,2,2;1,1,2,2 Rev.Sim.: 0

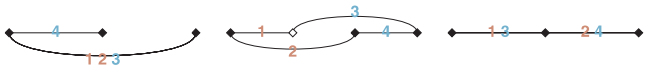
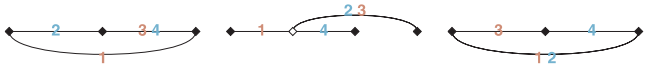
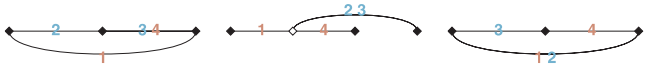
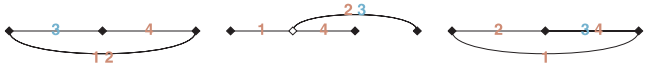
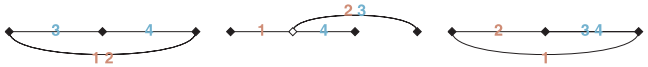
133	 Ly(0)	 Ay(7)	 Az(0)	 Ly(0)	 Ay(7)	 Az(0)	3;4;3,3,4;0,0,0-1,1,3;0,1,0-2,2,2;1,0,0-2,1,1;1,0,1-2,1,2;1,2,1,2 Rev.Sim.: 0 
134	 Ly(1)	 Lx(0)	 Az(0)	 Ly(1)	 Lx(0)	 Az(0)	3;4;3,3,4;0,0,0-1,2,1;0,0,1-1,1,3;0,1,1-1,2,2;1,0,0-2,2,2;1,2,2,1 Rev.Sim.: 0 
135	 Lx(3)	 Ly(0)	 Az(0)	 Lx(3)	 Ly(0)	 Az(0)	3;4;3,3,4;0,0,0-2,1,1;0,0,1-1,1,3;0,1,0-2,2,2;1,0,1-2,1,2;1,2,1,2 Rev.Sim.: 0 
136	 Az(0)	 Lz(3)	 Az(0)	 Az(0)	 Lz(3)	 Az(0)	3;4;3,3,4;0,0,0-2,2,1;0,0,1-1,1,3;0,1,1-1,2,2;1,0,1-2,2,2;1,1,2,2 Rev.Sim.: 0 
137	 Ax(1)	 Lx(6)	 Ly(0)	 Ax(1)	 Lx(6)	 Ly(0)	3;4;3,3,4;0,0,0-1,1,1;0,0,1-1,1,3;0,1,0-1,2,3;1,0,0-2,2,2;1,2,2,1 Rev.Sim.: 0 
138	 Txz(2)	 B(0)	 Ly(0)	 Txz(2)	 B(0)	Ly(0)	3;4;3,3,4;0,0,0-1,2,3;1,0,0-2,1,1;1,0,1-2,1,2;1,1,0-2,2,2;1,1,2,1 Rev.Sim.: 0
139	Ax(0)	Lx(6)	Ly(0)	Ax(0)	Lx(6)	Ly(0)	3;4;3,3,4;0,0,0-1,2,3;1,0,0-2,1,1;1,0,1-2,1,2;1,1,0-2,2,2;1,1,2,2 Rev.Sim.: 0
140	Az(3)	Ax(0)	Ly(0)	Az(3)	Ax(0)	Ly(0)	3;4;3,3,4;0,0,0-2,2,1;0,0,1-1,2,3;1,0,1-2,1,2;1,1,1-2,2,2;1,2,2,1 Rev.Sim.: 0
141	Ay(2)	Ly(7)	Lx(0)	Ay(2)	Ly(7)	Lx(0)	3;4;3,3,4;0,0,0-1,1,1;0,0,1-1,1,3;0,1,0-2,2,2;1,0,0-2,1,3;1,2,1,2 Rev.Sim.: 0
142	Tyz(1)	B(0)	Lx(0)	Tyz(1)	B(0)	Lx(0)	3;4;3,3,4;0,0,0-2,1,3;0,1,0-1,2,1;0,1,1-1,2,2;1,1,0-2,2,2;1,1,2,1 Rev.Sim.: 0
143	Ay(0)	Ly(7)	Lx(0)	Ay(0)	Ly(7)	Lx(0)	3;4;3,3,4;0,0,0-2,1,3;0,1,0-1,2,1;0,1,1-1,2,2;1,1,0-2,2,2;1,1,2,2 Rev.Sim.: 0
144	Az(2)	Ay(1)	Lx(0)	Az(2)	Ay(1)	Lx(0)	3;4;3,3,4;0,0,0-2,2,1;0,0,1-2,1,3;0,1,1-1,2,2;1,1,1-2,2,2;1,2,1,2 Rev.Sim.: 0

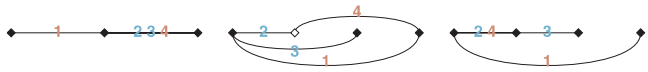
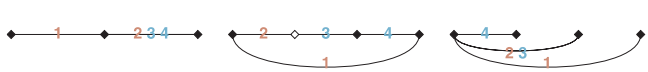
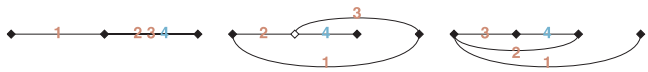
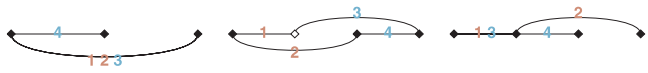
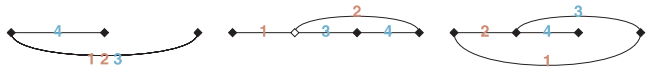
145	 Ly(1)	 Ax(4)	 S(0)	 Ly(1)	 Ax(4)	 S(0)	3;4;3,3,4;0,0,0-1,2,1;0,0,1-1,2,3;1,0,0-2,2,2;1,0,2-2,1,3;1,2,1,2 Rev.Sim.: 0 
146	 Txz(2)	 B(0)	 S(0)	 Txz(2)	 B(0)	 S(0)	3;4;3,3,4;0,0,0-1,2,3;1,0,0-2,1,1;1,0,1-2,1,3;1,1,0-2,2,2;1,1,2,1 Rev.Sim.: 0 
147	 Ax(4)	 Lx(3)	 S(0)	 Ax(4)	 Lx(3)	 S(0)	3;4;3,3,4;0,0,0-1,2,3;1,0,0-2,1,1;1,0,1-2,1,3;1,1,0-2,2,2;1,2,1,2 Rev.Sim.: 0 
148	 S(0)	 B(0)	 S(0)	 S(0)	 B(0)	 S(0)	3;4;3,3,4;0,0,0-1,2,3;1,0,0-2,1,3;1,1,0-2,2,1;1,1,1-2,2,2;1,1,1,2 Rev.Sim.: 0 
149	 Ly(0)	 Lx(0)	 S(0)	 Ly(0)	 Lx(0)	 S(0)	3;4;3,3,4;0,0,0-1,2,3;1,0,0-2,2,1;1,0,1-2,1,3;1,1,1-2,2,2;1,1,2,2 Rev.Sim.: 0 
150	 Lx(3)	 Ay(4)	 S(0)	 Lx(3)	 Ay(4)	S(0)	3;4;3,3,4;0,0,0-2,1,1;0,0,1-2,1,3;0,1,0-2,2,2;0,1,2-1,2,3;1,2,1,2 Rev.Sim.: 0
151	Tyz(1)	B(0)	S(0)	Tyz(1)	B(0)	S(0)	3;4;3,3,4;0,0,0-2,1,3;0,1,0-1,2,1;0,1,1-1,2,3;1,1,0-2,2,2;1,1,2,1 Rev.Sim.: 0
152	Ay(4)	Ly(1)	S(0)	Ay(4)	Ly(1)	S(0)	3;4;3,3,4;0,0,0-2,1,3;0,1,0-1,2,1;0,1,1-1,2,3;1,1,0-2,2,2;1,2,1,2 Rev.Sim.: 0
153	Lx(0)	Ly(0)	S(0)	Lx(0)	Ly(0)	S(0)	3;4;3,3,4;0,0,0-2,1,3;0,1,0-2,2,1;0,1,1-1,2,3;1,1,1-2,2,2;1,1,2,2 Rev.Sim.: 0
154	Az(1)	Ax(2)	S(0)	Az(1)	Ax(2)	S(0)	3;4;3,3,4;0,0,0-2,2,1;0,0,1-1,2,3;1,0,1-2,1,3;1,1,1-2,2,2;1,2,1,2 Rev.Sim.: 0
155	Az(3)	Lz(0)	S(0)	Az(3)	Lz(0)	S(0)	3;4;3,3,4;0,0,0-2,2,1;0,0,1-1,2,3;1,0,1-2,1,3;1,1,1-2,2,2;1,2,2,1 Rev.Sim.: 0
156	Az(2)	Ay(1)	S(0)	Az(2)	Ay(1)	S(0)	3;4;3,3,4;0,0,0-2,2,1;0,0,1-2,1,3;0,1,1-1,2,3;1,1,1-2,2,2;1,2,1,2 Rev.Sim.: 0

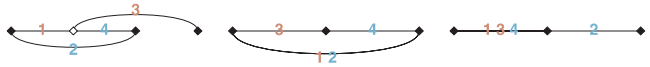
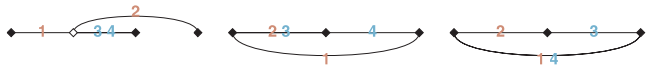
157	 Lz(1)	 Lz(2)	 B(0)	 Lz(1)	 Lz(2)	 B(0)	3;4;3,4,2;0,0,0-1,1,1;0,1,0-1,2,1;0,2,0-2,3,1;1,0,0-2,2,1;1,2,2,1 Rev.Sim.: 0347 
158	 Lz(1)	 Lx(6)	 Ax(0)	 Lz(1)	 Lx(6)	 Ax(0)	3;4;3,4,3;0,0,0-1,1,1;0,0,1-1,1,2;0,1,0-1,3,2;1,0,0-2,2,1;1,2,2,1 Rev.Sim.: 0 
159	 Ly(0)	 Ax(0)	 Ax(0)	 Ly(0)	 Ax(0)	 Ax(0)	3;4;3,4,3;0,0,0-1,1,2;0,1,0-1,3,2;1,0,0-2,1,1;1,1,0-2,2,1;1,2,1,2 Rev.Sim.: 0 
160	 Az(0)	 Lx(6)	 Ax(0)	 Az(0)	 Lx(6)	 Ax(0)	3;4;3,4,3;0,0,0-2,2,1;0,0,1-1,1,2;0,1,1-1,2,2;0,2,0-1,3,2;1,1,2,2 Rev.Sim.: 0 
161	 Lx(5)	 Lz(3)	 Az(0)	 Lx(5)	 Lz(3)	 Az(0)	3;4;3,4,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-1,3,1;1,0,0-2,3,1;1,1,2,2 Rev.Sim.: 0 
162	 Ly(0)	 Az(0)	 Az(0)	 Ly(0)	 Az(0)	Az(0)	3;4;3,4,3;0,0,0-1,1,2;0,1,0-2,3,1;0,1,1-1,2,2;1,0,0-2,1,1;1,2,2,1 Rev.Sim.: 0
163	Ax(0)	Lz(3)	Az(0)	Ax(0)	Lz(3)	Az(0)	3;4;3,4,3;0,0,0-1,2,2;0,2,0-1,3,1;1,0,0-2,1,1;1,1,0-2,3,1;1,2,1,2 Rev.Sim.: 0
164	S(0)	B(0)	Txy(0)	S(0)	B(0)	Txy(0)	3;4;3,4,3;0,0,0-1,2,2;0,2,0-1,3,2;1,0,0-2,1,2;1,1,0-2,2,1;1,2,1,1 Rev.Sim.: 0
165	Lz(0)	Lz(0)	Txy(0)	Lz(0)	Lz(0)	Txy(0)	3;4;3,4,3;0,0,0-1,3,1;0,0,1-1,3,2;1,0,0-2,2,1;1,0,1-2,1,2;1,2,1,2 Rev.Sim.: 0
166	Ay(4)	Lz(0)	Txy(0)	Ay(4)	Lz(0)	Txy(0)	3;4;3,4,3;0,0,0-2,1,2;0,1,0-1,3,1;0,1,1-1,3,2;1,1,0-2,2,1;1,2,1,2 Rev.Sim.: 0
167	Lx(0)	Lx(6)	Txy(0)	Lx(0)	Lx(6)	Txy(0)	3;4;3,4,3;0,0,0-2,1,2;0,1,0-2,2,1;0,1,1-1,2,2;0,2,0-1,3,2;1,1,2,2 Rev.Sim.: 0
168	Az(1)	Lx(6)	Txy(0)	Az(1)	Lx(6)	Txy(0)	3;4;3,4,3;0,0,0-2,2,1;0,0,1-1,2,2;0,2,0-1,3,2;1,0,1-2,1,2;1,2,2,1 Rev.Sim.: 0

169	 S(5)	 B(0)	 Ay(0)	 S(5)	 B(0)	 Ay(0)	3;4;3,4,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-1,3,1;1,0,0-2,2,2;1,1,2,1 Rev.Sim.: 0 
170	 Lz(1)	 Lx(0)	 Ay(0)	 Lz(1)	 Lx(0)	 Ay(0)	3;4;3,4,3;0,0,0-1,1,2;0,1,0-1,2,2;0,2,0-1,3,1;1,0,0-2,2,2;1,2,2,1 Rev.Sim.: 0 
171	 Lx(0)	 Ax(7)	 Ay(0)	 Lx(0)	 Ax(7)	 Ay(0)	3;4;3,4,3;0,0,0-1,1,2;0,1,0-1,3,1;0,1,1-1,2,2;1,0,0-2,2,2;1,1,2,2 Rev.Sim.: 0 
172	 Lz(0)	 Az(7)	 Ay(0)	 Lz(0)	 Az(7)	 Ay(0)	3;4;3,4,3;0,0,0-1,3,1;0,0,1-2,2,2;1,0,0-2,1,1;1,1,0-2,2,1;1,2,1,2 Rev.Sim.: 0 
173	 Lx(5)	 Lz(0)	 Ay(0)	 Lx(5)	 Lz(0)	 Ay(0)	3;4;3,4,3;0,0,0-2,1,1;0,0,1-2,2,2;0,1,0-1,3,1;1,1,0-2,2,1;1,1,2,2 Rev.Sim.: 0 
174	 Ay(0)	 Ly(7)	 Ay(0)	 Ay(0)	 Ly(7)	Ay(0)	3;4;3,4,3;0,0,0-2,1,2;0,1,0-1,3,1;0,1,1-1,2,2;1,1,0-2,2,2;1,1,2,2 Rev.Sim.: 0
175	Lz(0)	Lz(3)	Tzy(0)	Lz(0)	Lz(3)	Tzy(0)	3;4;3,4,3;0,0,0-1,2,2;0,2,0-1,3,1;1,0,0-2,1,2;1,1,0-2,3,1;1,2,1,2 Rev.Sim.: 0
176	Lx(0)	Lx(0)	Tzy(0)	Lx(0)	Lx(0)	Tzy(0)	3;4;3,4,3;0,0,0-1,2,2;0,2,0-1,3,1;1,0,0-2,1,2;1,1,0-2,3,1;1,1,2,2 Rev.Sim.: 0
177	Ax(4)	Lz(3)	Tzy(0)	Ax(4)	Lz(3)	Tzy(0)	3;4;3,4,3;0,0,0-1,2,2;0,2,0-1,3,1;1,0,0-2,3,1;1,0,1-2,1,2;1,2,2,1 Rev.Sim.: 0
178	S(0)	B(0)	Tzy(0)	S(0)	B(0)	Tzy(0)	3;4;3,4,3;0,0,0-1,2,2;0,2,0-2,3,1;1,0,0-2,1,2;1,1,0-2,2,1;1,2,1,1 Rev.Sim.: 0
179	Ay(1)	Lx(0)	Tzy(0)	Ay(1)	Lx(0)	Tzy(0)	3;4;3,4,3;0,0,0-2,1,2;0,1,0-1,2,2;0,2,0-1,3,1;1,1,0-2,3,1;1,2,2,1 Rev.Sim.: 0
180	Ax(1)	Lx(6)	Lz(0)	Ax(1)	Lx(6)	Lz(0)	3;4;3,4,3;0,0,0-1,1,1;0,0,1-1,1,2;0,1,0-1,3,2;1,0,0-2,2,2;1,2,2,1 Rev.Sim.: 0

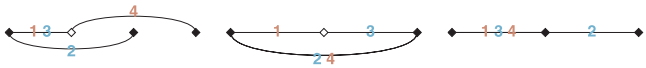
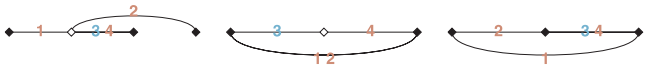
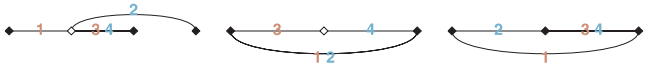
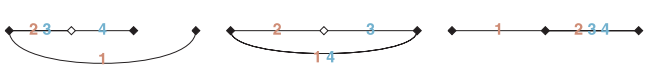
181	 Ax(0)	 Lx(6)	 Lz(0)	 Ax(0)	 Lx(6)	 Lz(0)	3;4;3,4,3;0,0,0-1,3,2;1,0,0-2,1,1;1,0,1-2,1,2;1,1,0-2,2,2;1,1,2,2 Rev.Sim.: 0 
182	 Txy(4)	 B(0)	 Lz(0)	 Txy(4)	 B(0)	 Lz(0)	3;4;3,4,3;0,0,0-1,3,2;1,0,0-2,1,1;1,0,1-2,2,2;1,1,0-2,2,1;1,1,1,2 Rev.Sim.: 0 
183	 Ay(1)	 Ax(4)	 Lz(0)	 Ay(1)	 Ax(4)	 Lz(0)	3;4;3,4,3;0,0,0-2,1,2;0,1,0-1,3,2;1,1,0-2,2,1;1,1,1-2,2,2;1,2,1,2 Rev.Sim.: 0 
184	 Az(4)	 Lz(3)	 Lx(0)	 Az(4)	 Lz(3)	 Lx(0)	3;4;3,4,3;0,0,0-1,1,1;0,0,1-2,2,2;0,1,0-1,3,1;1,0,0-2,3,1;1,1,2,2 Rev.Sim.: 0 
185	 Tzy(1)	 B(0)	 Lx(0)	 Tzy(1)	 B(0)	 Lx(0)	3;4;3,4,3;0,0,0-2,1,2;0,1,0-2,3,1;0,1,1-1,2,2;1,1,1-2,2,2;1,1,2,1 Rev.Sim.: 0 
186	 Ay(4)	 Az(1)	 Lx(0)	 Ay(4)	 Az(1)	Lx(0)	3;4;3,4,3;0,0,0-2,1,2;0,1,0-2,3,1;0,1,1-1,2,2;1,1,1-2,2,2;1,2,1,2 Rev.Sim.: 0
187	Az(0)	Lz(3)	Lx(0)	Az(0)	Lz(3)	Lx(0)	3;4;3,4,3;0,0,0-2,3,1;0,0,1-1,1,2;0,1,1-1,2,2;1,0,1-2,2,2;1,1,2,2 Rev.Sim.: 0
188	Lz(1)	Ax(2)	S(0)	Lz(1)	Ax(2)	S(0)	3;4;3,4,3;0,0,0-1,1,2;0,1,0-1,3,2;1,0,0-2,2,2;1,2,0-2,3,1;1,2,1,2 Rev.Sim.: 0
189	Txy(4)	B(0)	S(0)	Txy(4)	B(0)	S(0)	3;4;3,4,3;0,0,0-1,3,2;1,0,0-2,1,1;1,0,1-2,2,2;1,1,0-2,3,1;1,1,1,2 Rev.Sim.: 0
190	Ax(2)	Lx(5)	S(0)	Ax(2)	Lx(5)	S(0)	3;4;3,4,3;0,0,0-1,3,2;1,0,0-2,1,1;1,0,1-2,2,2;1,1,0-2,3,1;1,2,2,1 Rev.Sim.: 0
191	Lz(0)	Lx(0)	S(0)	Lz(0)	Lx(0)	S(0)	3;4;3,4,3;0,0,0-1,3,2;1,0,0-2,1,2;1,1,0-2,2,2;1,2,0-2,3,1;1,1,2,2 Rev.Sim.: 0
192	S(0)	B(0)	S(0)	S(0)	B(0)	S(0)	3;4;3,4,3;0,0,0-1,3,2;1,0,0-2,1,2;1,1,0-2,3,1;1,1,1-2,2,2;1,1,1,2 Rev.Sim.: 0

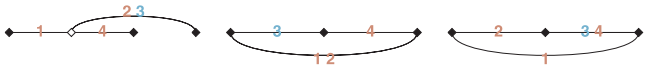
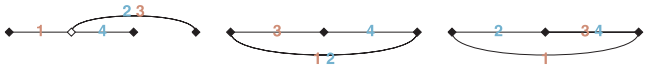
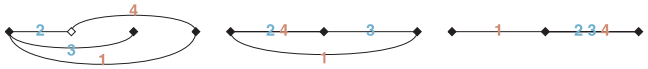
193	 Lx(5)	 Az(2)	 S(0)	 Lx(5)	 Az(2)	 S(0)	3;4;3,4,3;0,0,0-2,1,1;0,0,1-2,2,2;0,1,0-2,3,1;0,2,1-1,3,2;1,1,2,2 Rev.Sim.: 0 
194	 Ay(1)	 Ax(4)	 S(0)	 Ay(1)	 Ax(4)	 S(0)	3;4;3,4,3;0,0,0-2,1,2;0,1,0-1,3,2;1,1,0-2,3,1;1,1,1-2,2,2;1,2,1,2 Rev.Sim.: 0 
195	 Ay(5)	 Ly(0)	 S(0)	 Ay(5)	 Ly(0)	 S(0)	3;4;3,4,3;0,0,0-2,1,2;0,1,0-1,3,2;1,1,0-2,3,1;1,1,1-2,2,2;1,2,2,1 Rev.Sim.: 0 
196	 Tzy(1)	 B(0)	 S(0)	 Tzy(1)	 B(0)	 S(0)	3;4;3,4,3;0,0,0-2,1,2;0,1,0-2,3,1;0,1,1-1,3,2;1,1,1-2,2,2;1,1,2,1 Rev.Sim.: 0 
197	 Lx(0)	 Lz(0)	 S(0)	 Lx(0)	 Lz(0)	 S(0)	3;4;3,4,3;0,0,0-2,1,2;0,1,0-2,3,1;0,1,1-1,3,2;1,1,1-2,2,2;1,1,2,2 Rev.Sim.: 0 
198	 Ay(4)	 Az(1)	 S(0)	 Ay(4)	 Az(1)	S(0)	3;4;3,4,3;0,0,0-2,1,2;0,1,0-2,3,1;0,1,1-1,3,2;1,1,1-2,2,2;1,2,1,2 Rev.Sim.: 0
199	Az(2)	Lz(1)	S(0)	Az(2)	Lz(1)	S(0)	3;4;3,4,3;0,0,0-2,3,1;0,0,1-1,1,2;0,1,1-1,3,2;1,0,1-2,2,2;1,2,1,2 Rev.Sim.: 0
200	Ax(1)	Lx(6)	Ax(0)	Ax(1)	Lx(6)	Ax(0)	3;4;3,4,4;0,0,0-1,1,1;0,0,1-1,1,3;0,1,0-1,3,3;1,0,0-2,2,2;1,2,2,1 Rev.Sim.: 0
201	Ax(0)	Lx(6)	Ax(0)	Ax(0)	Lx(6)	Ax(0)	3;4;3,4,4;0,0,0-1,3,3;1,0,0-2,1,1;1,0,1-2,1,2;1,1,0-2,2,2;1,1,2,2 Rev.Sim.: 0
202	Ax(4)	Lx(3)	Txy(0)	Ax(4)	Lx(3)	Txy(0)	3;4;3,4,4;0,0,0-1,3,3;1,0,0-2,1,1;1,0,1-2,1,3;1,1,0-2,2,2;1,2,1,2 Rev.Sim.: 0
203	Txy(0)	B(0)	Txy(0)	Txy(0)	B(0)	Txy(0)	3;4;3,4,4;0,0,0-1,3,3;1,0,0-2,1,3;1,1,0-2,2,1;1,1,1-2,2,2;1,1,1,2 Rev.Sim.: 0
204	Ax(0)	Lx(0)	Txy(0)	Ax(0)	Lx(0)	Txy(0)	3;4;3,4,4;0,0,0-1,3,3;1,0,0-2,2,1;1,0,1-2,1,3;1,1,1-2,2,2;1,1,2,2 Rev.Sim.: 0

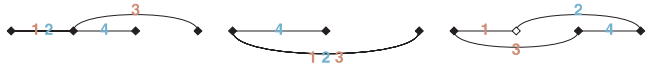
205	 Ax(2)	 Lx(5)	 Txz(0)	 Ax(2)	 Lx(5)	 Txz(0)	3;4;3,4,4;0,0,0-1,3,3;1,0,0-2,1,1;1,0,1-2,2,2;1,1,0-2,3,1;1,2,2,1 Rev.Sim.: 0 
206	 Ax(0)	 Lx(0)	 Txz(0)	 Ax(0)	 Lx(0)	 Txz(0)	3;4;3,4,4;0,0,0-1,3,3;1,0,0-2,1,2;1,1,0-2,2,2;1,2,0-2,3,1;1,1,2,2 Rev.Sim.: 0 
207	 Txz(0)	 B(0)	 Txz(0)	 Txz(0)	 B(0)	 Txz(0)	3;4;3,4,4;0,0,0-1,3,3;1,0,0-2,1,2;1,1,0-2,3,1;1,1,1-2,2,2;1,1,1,2 Rev.Sim.: 0 
208	 Lx(5)	 Az(2)	 Tyz(0)	 Lx(5)	 Az(2)	 Tyz(0)	3;4;3,4,4;0,0,0-2,1,1;0,0,1-2,2,3;0,1,0-2,3,1;0,2,1-1,3,2;1,1,2,2 Rev.Sim.: 0 
209	 Lx(0)	 Ay(0)	 Tyz(0)	 Lx(0)	 Ay(0)	 Tyz(0)	3;4;3,4,4;0,0,0-2,1,3;0,1,0-2,3,1;0,1,1-2,2,3;0,2,1-1,3,2;1,1,2,2 Rev.Sim.: 0 
210	 Lx(3)	 Ay(4)	 Tzy(0)	 Lx(3)	 Ay(4)	Tzy(0)	3;4;3,4,4;0,0,0-2,1,1;0,0,1-2,1,3;0,1,0-2,3,2;0,1,2-1,2,3;1,2,1,2 Rev.Sim.: 0
211	Lx(0)	Az(0)	Tzy(0)	Lx(0)	Az(0)	Tzy(0)	3;4;3,4,4;0,0,0-2,1,3;0,1,0-2,3,1;0,1,1-2,3,2;0,1,2-1,2,3;1,1,2,2 Rev.Sim.: 0
212	Ly(6)	Ly(1)	B(0)	Ly(6)	Ly(1)	B(0)	3;4;4,2,3;0,0,0-1,1,1;0,0,1-2,1,2;1,0,0-2,1,1;2,0,0-3,1,2;1,1,2,2 Rev.Sim.: 0257
213	Ly(1)	Ly(6)	Ly(0)	Ly(1)	Ly(6)	Ly(0)	3;4;4,2,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-2,1,3;1,0,0-3,1,2;1,2,2,1 Rev.Sim.: 02
214	Ly(6)	Ly(1)	Ly(0)	Ly(6)	Ly(1)	Ly(0)	3;4;4,2,4;0,0,0-1,1,1;0,0,1-2,1,3;1,0,0-2,1,1;2,0,0-3,1,2;1,1,2,2 Rev.Sim.: 02
215	Lz(2)	Lz(1)	B(0)	Lz(2)	Lz(1)	B(0)	3;4;4,3,2;0,0,0-1,1,1;0,1,0-2,2,1;1,0,0-2,1,1;2,0,0-3,2,1;1,1,2,2 Rev.Sim.: 0347
216	S(6)	B(0)	Ax(0)	S(6)	B(0)	Ax(0)	3;4;4,3,3;0,0,0-1,1,1;0,0,1-2,1,2;0,1,0-2,2,2;1,0,0-3,1,1;1,1,1,2 Rev.Sim.: 0

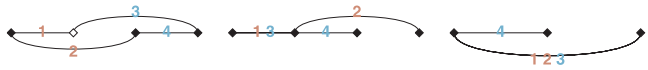
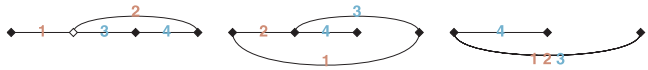
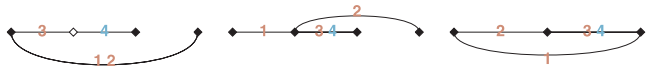
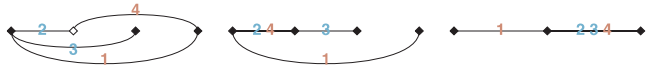
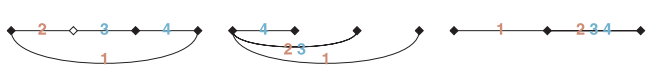
217	 Lz(2)	 Ly(0)	 Ax(0)	 Lz(2)	 Ly(0)	 Ax(0)	3;4;4,3,3;0,0,0-1,1,2;0,1,0-2,2,2;1,0,0-2,1,2;2,0,0-3,1,1;1,1,2,2 Rev.Sim.: 0 
218	 Ly(0)	 Ay(7)	 Ax(0)	 Ly(0)	 Ay(7)	 Ax(0)	3;4;4,3,3;0,0,0-1,1,2;0,1,0-2,2,2;1,0,0-3,1,1;1,0,1-2,1,2;1,2,1,2 Rev.Sim.: 0 
219	 Ly(6)	 Lz(0)	 Ax(0)	 Ly(6)	 Lz(0)	 Ax(0)	3;4;4,3,3;0,0,0-1,2,1;0,0,1-2,2,2;1,0,0-2,2,1;2,0,0-3,1,1;1,1,2,2 Rev.Sim.: 0 
220	 Lz(0)	 Az(7)	 Ax(0)	 Lz(0)	 Az(7)	 Ax(0)	3;4;4,3,3;0,0,0-1,2,1;0,0,1-2,2,2;1,0,0-3,1,1;1,1,0-2,2,1;1,2,1,2 Rev.Sim.: 0 
221	 Ax(0)	 Lx(6)	 Ax(0)	 Ax(0)	 Lx(6)	 Ax(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-3,1,1;1,0,1-2,1,2;1,1,0-2,2,2;1,1,2,2 Rev.Sim.: 0 
222	 Lz(2)	 Ly(7)	 Ay(0)	 Lz(2)	 Ly(7)	Ay(0)	3;4;4,3,3;0,0,0-1,1,1;0,0,1-1,1,2;0,1,0-2,2,1;1,0,0-3,1,2;1,2,1,2 Rev.Sim.: 0
223	Lx(0)	Ay(0)	Ay(0)	Lx(0)	Ay(0)	Ay(0)	3;4;4,3,3;0,0,0-1,1,2;0,1,0-1,2,1;1,0,0-3,1,2;1,1,0-2,2,1;1,1,2,2 Rev.Sim.: 0
224	Az(0)	Ly(7)	Ay(0)	Az(0)	Ly(7)	Ay(0)	3;4;4,3,3;0,0,0-2,2,1;0,0,1-1,1,2;1,0,1-2,1,2;2,0,0-3,1,2;1,1,2,2 Rev.Sim.: 0
225	Ly(6)	Lz(3)	Az(0)	Ly(6)	Lz(3)	Az(0)	3;4;4,3,3;0,0,0-1,1,1;0,0,1-2,1,2;0,1,0-1,2,1;1,0,0-3,2,1;1,1,2,2 Rev.Sim.: 0
226	Lx(0)	Az(0)	Az(0)	Lx(0)	Az(0)	Az(0)	3;4;4,3,3;0,0,0-1,1,2;0,1,0-1,2,1;1,0,0-3,2,1;1,0,1-2,1,2;1,1,2,2 Rev.Sim.: 0
227	Ay(0)	Lz(3)	Az(0)	Ay(0)	Lz(3)	Az(0)	3;4;4,3,3;0,0,0-2,1,2;0,1,0-1,2,1;1,1,0-2,2,1;2,0,0-3,2,1;1,1,2,2 Rev.Sim.: 0
228	S(0)	B(0)	Tyx(0)	S(0)	B(0)	Tyx(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-2,1,2;1,1,0-2,2,1;2,0,0-3,1,2;1,1,1,2 Rev.Sim.: 0

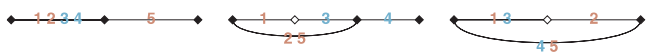
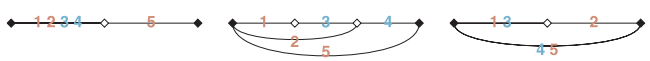
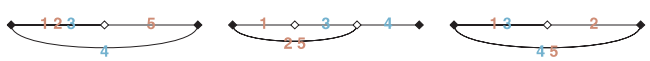
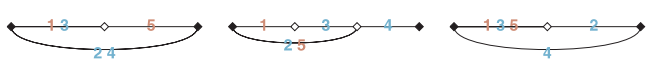
229	Ly(0)	Ly(7)	Tyx(0)	Ly(0)	Ly(7)	Tyx(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-2,2,1;1,0,1-2,1,2;2,0,0-3,1,2;1,1,2,2 Rev.Sim.: 0
230	Ax(4)	Lz(0)	Tyx(0)	Ax(4)	Lz(0)	Tyx(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-2,2,1;1,0,1-3,1,2;2,0,0-3,1,1;1,2,1,2 Rev.Sim.: 0
231	Lz(0)	Lz(0)	Tyx(0)	Lz(0)	Lz(0)	Tyx(0)	3;4;4,3,3;0,0,0-2,2,1;0,0,1-1,2,2;1,0,1-3,1,2;2,0,0-3,1,1;1,2,2,1 Rev.Sim.: 0
232	Az(2)	Ly(7)	Tyx(0)	Az(2)	Ly(7)	Tyx(0)	3;4;4,3,3;0,0,0-2,2,1;0,0,1-2,1,2;0,1,1-1,2,2;2,0,0-3,1,2;1,2,1,2 Rev.Sim.: 0
233	S(0)	B(0)	Tzx(0)	S(0)	B(0)	Tzx(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-2,1,2;1,1,0-2,2,1;2,0,0-3,2,1;1,1,1,2 Rev.Sim.: 0
234	Lz(0)	Lz(3)	Tzx(0)	Lz(0)	Lz(3)	Tzx(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-2,1,2;1,1,0-2,2,1;2,0,0-3,2,1;1,1,2,2 Rev.Sim.: 0
235	Ax(2)	Ly(0)	Tzx(0)	Ax(2)	Ly(0)	Tzx(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-2,1,2;1,1,0-3,2,1;2,0,0-3,1,1;1,2,1,2 Rev.Sim.: 0
236	Ly(0)	Ly(0)	Tzx(0)	Ly(0)	Ly(0)	Tzx(0)	3;4;4,3,3;0,0,0-2,1,2;0,1,0-1,2,2;1,1,0-3,2,1;2,0,0-3,1,1;1,2,2,1 Rev.Sim.: 0
237	Ay(4)	Lz(3)	Tzx(0)	Ay(4)	Lz(3)	Tzx(0)	3;4;4,3,3;0,0,0-2,1,2;0,1,0-2,2,1;0,1,1-1,2,2;2,0,0-3,2,1;1,2,1,2 Rev.Sim.: 0
238	Ly(1)	Ay(6)	Lz(0)	Ly(1)	Ay(6)	Lz(0)	3;4;4,3,3;0,0,0-1,1,1;0,0,1-1,1,2;0,1,0-2,2,2;1,0,0-3,1,2;1,2,2,1 Rev.Sim.: 0
239	Tyx(0)	B(0)	Lz(0)	Tyx(0)	B(0)	Lz(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-3,1,2;1,1,0-2,2,1;1,1,1-2,2,2;1,1,1,2 Rev.Sim.: 0
240	Ax(6)	Ay(0)	Lz(0)	Ax(6)	Ay(0)	Lz(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-3,1,2;1,1,0-2,2,1;1,1,1-2,2,2;1,2,2,1 Rev.Sim.: 0

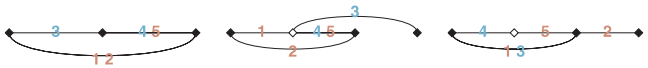
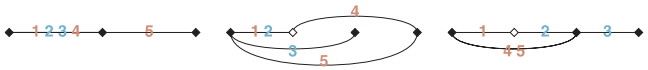
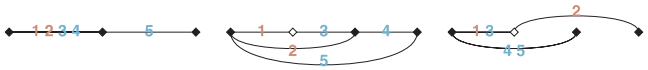
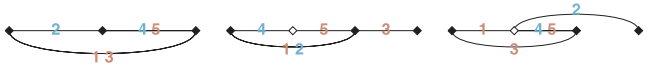
241	 Ay(4)	 Ly(1)	 Lz(0)	 Ay(4)	 Ly(1)	 Lz(0)	3;4;4,3,3;0,0,0-3,1,2;0,1,0-1,2,1;0,1,1-1,2,2;1,1,0-2,2,2;1,2,1,2 Rev.Sim.: 0 
242	 Lz(1)	 Az(6)	 Ly(0)	 Lz(1)	 Az(6)	 Ly(0)	3;4;4,3,3;0,0,0-1,1,1;0,0,1-2,2,2;0,1,0-1,2,1;1,0,0-3,2,1;1,2,2,1 Rev.Sim.: 0 
243	 Tzx(2)	 B(0)	 Ly(0)	 Tzx(2)	 B(0)	 Ly(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-3,2,1;1,0,1-2,1,2;1,1,1-2,2,2;1,1,2,1 Rev.Sim.: 0 
244	 Ax(4)	 Az(2)	 Ly(0)	 Ax(4)	 Az(2)	 Ly(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-3,2,1;1,0,1-2,1,2;1,1,1-2,2,2;1,2,1,2 Rev.Sim.: 0 
245	 Az(0)	 Lz(3)	 Ly(0)	 Az(0)	 Lz(3)	 Ly(0)	3;4;4,3,3;0,0,0-3,2,1;0,0,1-1,1,2;0,1,1-1,2,2;1,0,1-2,2,2;1,1,2,2 Rev.Sim.: 0 
246	 Lz(2)	 Ay(1)	 S(0)	 Lz(2)	 Ay(1)	S(0)	3;4;4,3,3;0,0,0-1,1,2;0,1,0-2,2,2;1,0,0-3,1,2;2,1,0-3,2,1;1,1,2,2 Rev.Sim.: 0
247	Ly(6)	Az(1)	S(0)	Ly(6)	Az(1)	S(0)	3;4;4,3,3;0,0,0-1,2,1;0,0,1-2,2,2;1,0,0-3,2,1;2,0,1-3,1,2;1,1,2,2 Rev.Sim.: 0
248	Lz(0)	Ly(0)	S(0)	Lz(0)	Ly(0)	S(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-3,1,2;1,1,0-2,2,2;2,1,0-3,2,1;1,1,2,2 Rev.Sim.: 0
249	Tyx(4)	B(0)	S(0)	Tyx(4)	B(0)	S(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-3,1,2;1,1,0-3,2,1;1,1,1,1-2,2,2;1,1,2,1 Rev.Sim.: 0
250	S(0)	B(0)	S(0)	S(0)	B(0)	S(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-3,1,2;1,1,0-3,2,1;1,1,1,1-2,2,2;1,1,1,2 Rev.Sim.: 0
251	Ax(6)	Lx(0)	S(0)	Ax(6)	Lx(0)	S(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-3,1,2;1,1,0-3,2,1;1,1,1,1-2,2,2;1,2,2,1 Rev.Sim.: 0
252	Ax(2)	Ay(4)	S(0)	Ax(2)	Ay(4)	S(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-3,1,2;1,1,0-3,2,1;1,1,1,1-2,2,2;1,2,1,2 Rev.Sim.: 0

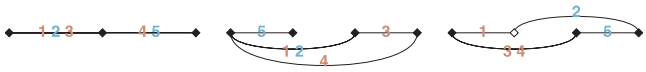
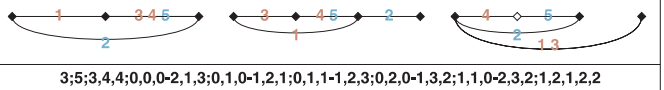
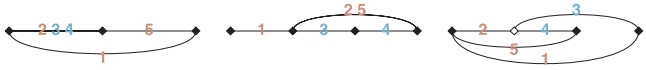
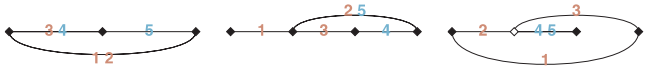
253	 Ly(0)	 Lz(0)	 S(0)	 Ly(0)	 Lz(0)	 S(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-3,2,1;1,0,1-2,2,2;2,0,1-3,1,2;1,1,2,2 Rev.Sim.: 0 
254	 Tzx(2)	 B(0)	 S(0)	 Tzx(2)	 B(0)	 S(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-3,2,1;1,0,1-3,1,2;1,1,1-2,2,2;1,1,2,1 Rev.Sim.: 0 
255	 Ax(4)	 Az(2)	 S(0)	 Ax(4)	 Az(2)	 S(0)	3;4;4,3,3;0,0,0-1,2,2;1,0,0-3,2,1;1,0,1-3,1,2;1,1,1-2,2,2;1,2,1,2 Rev.Sim.: 0 
256	 Ay(1)	 Ly(6)	 S(0)	 Ay(1)	 Ly(6)	 S(0)	3;4;4,3,3;0,0,0-3,1,2;0,1,0-1,2,1;0,1,1-2,2,2;1,1,0-3,2,1;1,2,2,1 Rev.Sim.: 0 
257	 Az(1)	 Lz(2)	 S(0)	 Az(1)	 Lz(2)	 S(0)	3;4;4,3,3;0,0,0-3,2,1;0,0,1-1,1,2;0,1,1-2,2,2;1,0,1-3,1,2;1,2,2,1 Rev.Sim.: 0 
258	 Ay(2)	 Ly(7)	 Ay(0)	 Ay(2)	 Ly(7)	Ay(0)	3;4;4,3,4;0,0,0-1,1,1;0,0,1-1,1,3;0,1,0-2,2,2;1,0,0-3,1,3;1,2,1,2 Rev.Sim.: 0
259	Ay(0)	Ly(7)	Ay(0)	Ay(0)	Ly(7)	Ay(0)	3;4;4,3,4;0,0,0-3,1,3;0,1,0-1,2,1;0,1,1-1,2,2;1,1,0-2,2,2;1,1,2,2 Rev.Sim.: 0
260	Tyx(0)	B(0)	Tyx(0)	Tyx(0)	B(0)	Tyx(0)	3;4;4,3,4;0,0,0-1,2,3;1,0,0-3,1,3;1,1,0-2,2,1;1,1,1-2,2,2;1,1,1,2 Rev.Sim.: 0
261	Ay(4)	Ly(1)	Tyx(0)	Ay(4)	Ly(1)	Tyx(0)	3;4;4,3,4;0,0,0-3,1,3;0,1,0-1,2,1;0,1,1-1,2,3;1,1,0-2,2,2;1,2,1,2 Rev.Sim.: 0
262	Ay(0)	Ly(0)	Tyx(0)	Ay(0)	Ly(0)	Tyx(0)	3;4;4,3,4;0,0,0-3,1,3;0,1,0-2,2,1;0,1,1-1,2,3;1,1,1-2,2,2;1,1,2,2 Rev.Sim.: 0
263	Ly(6)	Az(1)	Txz(0)	Ly(6)	Az(1)	Txz(0)	3;4;4,3,4;0,0,0-1,2,1;0,0,1-2,2,3;1,0,0-3,2,1;2,0,1-3,1,2;1,1,2,2 Rev.Sim.: 0
264	Ly(0)	Ax(0)	Txz(0)	Ly(0)	Ax(0)	Txz(0)	3;4;4,3,4;0,0,0-1,2,3;1,0,0-3,2,1;1,0,1-2,2,3;2,0,1-3,1,2;1,1,2,2 Rev.Sim.: 0

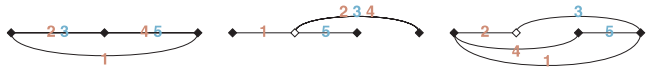
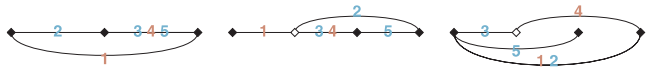
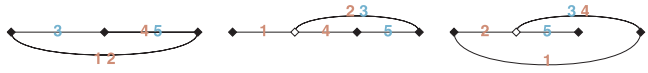
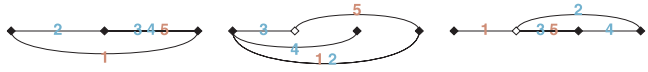
265	 Ay(1)	 Ly(6)	 Tyz(0)	 Ay(1)	 Ly(6)	 Tyz(0)	3;4;4,3,4;0,0,0-3,1,3;0,1,0-1,2,1;0,1,1-2,2,2;1,1,0-3,2,1;1,2,2,1 Rev.Sim.: 0 
266	 Ay(0)	 Ly(0)	 Tyz(0)	 Ay(0)	 Ly(0)	 Tyz(0)	3;4;4,3,4;0,0,0-3,1,3;0,1,0-1,2,2;1,1,0-2,2,2;2,1,0-3,2,1;1,1,2,2 Rev.Sim.: 0 
267	 Tyz(0)	 B(0)	 Tyz(0)	 Tyz(0)	 B(0)	 Tyz(0)	3;4;4,3,4;0,0,0-3,1,3;0,1,0-1,2,2;1,1,0-3,2,1;1,1,1-2,2,2;1,1,1,2 Rev.Sim.: 0 
268	 Ly(1)	 Ax(4)	 Tzx(0)	 Ly(1)	 Ax(4)	 Tzx(0)	3;4;4,3,4;0,0,0-1,2,1;0,0,1-1,2,3;1,0,0-3,2,2;1,0,2-2,1,3;1,2,1,2 Rev.Sim.: 0 
269	 Ly(0)	 Az(0)	 Tzx(0)	 Ly(0)	 Az(0)	 Tzx(0)	3;4;4,3,4;0,0,0-1,2,3;1,0,0-3,2,1;1,0,1-3,2,2;1,0,2-2,1,3;1,1,2,2 Rev.Sim.: 0 
270	 Lz(1)	 Lz(2)	 Lz(0)	 Lz(1)	 Lz(2)	Lz(0)	3;4;4,4,2;0,0,0-1,1,1;0,1,0-1,2,1;0,2,0-2,3,1;1,0,0-3,2,1;1,2,2,1 Rev.Sim.: 04
271	Lz(2)	Lz(1)	Lz(0)	Lz(2)	Lz(1)	Lz(0)	3;4;4,4,2;0,0,0-1,1,1;0,1,0-2,3,1;1,0,0-2,1,1;2,0,0-3,2,1;1,1,2,2 Rev.Sim.: 04
272	Az(4)	Lz(3)	Az(0)	Az(4)	Lz(3)	Az(0)	3;4;4,4,3;0,0,0-1,1,1;0,0,1-2,2,2;0,1,0-1,3,1;1,0,0-3,3,1;1,1,2,2 Rev.Sim.: 0
273	Az(0)	Lz(3)	Az(0)	Az(0)	Lz(3)	Az(0)	3;4;4,4,3;0,0,0-3,3,1;0,0,1-1,1,2;0,1,1-1,2,2;1,0,1-2,2,2;1,1,2,2 Rev.Sim.: 0
274	Tzx(0)	B(0)	Tzx(0)	Tzx(0)	B(0)	Tzx(0)	3;4;4,4,3;0,0,0-1,3,2;1,0,0-3,3,1;1,0,1-2,1,2;1,1,1-2,2,2;1,1,1,2 Rev.Sim.: 0
275	Az(2)	Lz(1)	Tzx(0)	Az(2)	Lz(1)	Tzx(0)	3;4;4,4,3;0,0,0-3,3,1;0,0,1-1,1,2;0,1,1-1,3,2;1,0,1-2,2,2;1,2,1,2 Rev.Sim.: 0
276	Az(0)	Lz(0)	Tzx(0)	Az(0)	Lz(0)	Tzx(0)	3;4;4,4,3;0,0,0-3,3,1;0,0,1-2,1,2;0,1,1-1,3,2;1,1,1-2,2,2;1,1,2,2 Rev.Sim.: 0

277	 Lz(2)	 Ay(1)	 Txy(0)	 Lz(2)	 Ay(1)	 Txy(0)	3;4;4,4,3;0,0,0-1,1,2;0,1,0-2,3,2;1,0,0-3,1,2;2,1,0-3,2,1;1,1,2,2 Rev.Sim.: 0 
278	 Lz(0)	 Ax(0)	 Txy(0)	 Lz(0)	 Ax(0)	 Txy(0)	3;4;4,4,3;0,0,0-1,3,2;1,0,0-3,1,2;1,1,0-2,3,2;2,1,0-3,2,1;1,1,2,2 Rev.Sim.: 0 
279	 Tzy(0)	 B(0)	 Tzy(0)	 Tzy(0)	 B(0)	 Tzy(0)	3;4;4,4,3;0,0,0-3,1,2;0,1,0-3,3,1;0,1,1-1,2,2;1,1,1-2,2,2;1,1,1,2 Rev.Sim.: 0 
280	 Az(1)	 Lz(2)	 Tzy(0)	 Az(1)	 Lz(2)	 Tzy(0)	3;4;4,4,3;0,0,0-3,3,1;0,0,1-1,1,2;0,1,1-2,2,2;1,0,1-3,1,2;1,2,2,1 Rev.Sim.: 0 
281	 Az(0)	 Lz(0)	 Tzy(0)	 Az(0)	 Lz(0)	 Tzy(0)	3;4;4,4,3;0,0,0-3,3,1;0,0,1-1,2,2;1,0,1-2,2,2;2,0,1-3,1,2;1,1,2,2 Rev.Sim.: 0 
282	 Lz(1)	 Ax(2)	 Tyx(0)	 Lz(1)	 Ax(2)	Tyx(0)	3;4;4,4,3;0,0,0-1,1,2;0,1,0-1,3,2;1,0,0-3,2,2;1,2,0-2,3,1;1,2,1,2 Rev.Sim.: 0
283	Lz(0)	Ay(0)	Tyx(0)	Lz(0)	Ay(0)	Tyx(0)	3;4;4,4,3;0,0,0-1,3,2;1,0,0-3,1,2;1,1,0-3,2,2;1,2,0-2,3,1;1,1,2,2 Rev.Sim.: 0
284	S(3)	Lx(5)	Ly(0)	S(3)	Lx(5)	Ly(0)	3;5;3,3,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-1,2,3;0,1,0-1,2,2;1,0,0-2,2,2;1,2,2,1,1 Rev.Sim.: 0
285	S(3)	Ly(6)	Lx(0)	S(3)	Ly(6)	Lx(0)	3;5;3,3,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-2,1,3;0,1,0-1,2,2;1,0,0-2,2,2;1,2,2,1,1 Rev.Sim.: 0
286	Txz(3)	Lx(5)	B(0)	Txz(3)	Lx(5)	B(0)	3;5;3,3,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-1,2,3;0,1,0-1,2,2;1,0,0-2,2,3;1,2,2,1,1 Rev.Sim.: 0
287	Tyz(3)	Ly(6)	B(0)	Tyz(3)	Ly(6)	B(0)	3;5;3,3,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-2,1,3;0,1,0-2,2,3;1,0,0-2,1,2;1,2,2,1,1 Rev.Sim.: 0
288	S(3)	Az(4)	B(0)	S(3)	Az(4)	B(0)	3;5;3,3,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-2,2,3;0,1,0-1,2,2;1,0,0-2,2,2;1,2,2,1,1 Rev.Sim.: 0

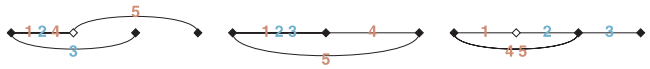
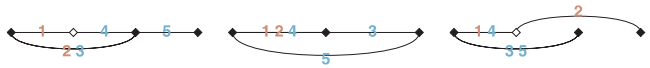
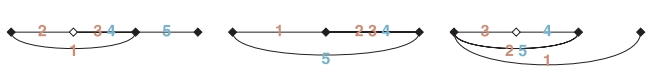
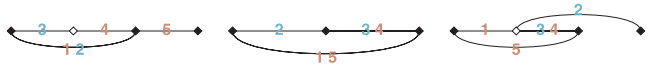
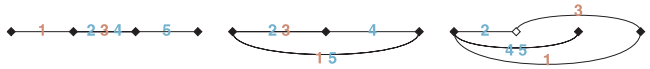
289	 S(5)	 Lx(3)	 Lz(0)	 S(5)	 Lx(3)	 Lz(0)	3;5;3,4,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-1,2,1;0,2,0-1,3,2;1,0,0-2,2,2;1,1,2,2,1 Rev.Sim.: 0 
290	 S(5)	 Lz(2)	 Lx(0)	 S(5)	 Lz(2)	 Lx(0)	3;5;3,4,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-1,2,1;0,2,0-2,3,1;1,0,0-2,2,2;1,1,2,2,1 Rev.Sim.: 0 
291	 Txy(5)	 Lx(3)	 B(0)	 Txy(5)	 Lx(3)	 B(0)	3;5;3,4,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-1,2,1;0,2,0-1,3,2;1,0,0-2,3,2;1,1,2,2,1 Rev.Sim.: 0 
292	 S(5)	 Ay(2)	 B(0)	 S(5)	 Ay(2)	 B(0)	3;5;3,4,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-1,2,1;0,2,0-2,3,2;1,0,0-2,2,2;1,1,2,2,1 Rev.Sim.: 0 
293	 Lz(1)	 Tzy(6)	 B(0)	 Lz(1)	 Tzy(6)	 B(0)	3;5;3,4,3;0,0,0-1,1,1;0,0,1-2,2,2;0,1,0-1,2,1;0,2,0-2,3,2;1,0,0-2,2,1;1,2,2,2,1 Rev.Sim.: 0 
294	 Lz(0)	 Txy(0)	 Txy(0)	 Lz(0)	 Txy(0)	Txy(0)	3;5;3,4,4;0,0,0-1,3,1;0,0,1-1,3,3;1,0,0-2,2,1;1,0,1-2,1,3;1,1,1-2,2,2;1,2,1,2,2 Rev.Sim.: 0
295	Tyz(1)	Lx(6)	Txy(0)	Tyz(1)	Lx(6)	Txy(0)	3;5;3,4,4;0,0,0-2,1,3;0,1,0-1,2,1;0,1,1-1,2,3;0,2,0-1,3,3;1,1,0-2,2,2;1,1,2,2,1 Rev.Sim.: 0
296	Ly(0)	Txz(0)	Txz(0)	Ly(0)	Txz(0)	Txz(0)	3;5;3,4,4;0,0,0-1,1,3;0,1,0-1,3,3;1,0,0-2,1,2;1,1,0-2,2,2;1,2,0-2,3,1;1,2,1,2,2 Rev.Sim.: 0
297	Tzy(1)	Lx(6)	Txz(0)	Tzy(1)	Lx(6)	Txz(0)	3;5;3,4,4;0,0,0-2,1,2;0,0,2-1,1,3;0,1,0-2,3,1;0,1,1-1,3,3;1,1,1-2,2,2;1,2,1,2,1 Rev.Sim.: 0
298	Lx(3)	Txz(5)	Ay(0)	Lx(3)	Txz(5)	Ay(0)	3;5;3,4,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-1,2,3;0,1,0-1,3,2;1,0,0-2,2,3;1,2,2,1,2 Rev.Sim.: 0
299	S(5)	Lx(3)	Ay(0)	S(5)	Lx(3)	Ay(0)	3;5;3,4,4;0,0,0-1,1,1;0,0,1-1,2,3;0,1,0-1,2,1;0,2,0-1,3,2;1,0,0-2,2,3;1,1,2,2,1 Rev.Sim.: 0
300	Txy(0)	Az(7)	Ay(0)	Txy(0)	Az(7)	Ay(0)	3;5;3,4,4;0,0,0-1,3,2;0,0,2-2,2,3;1,0,0-2,1,2;1,1,0-2,2,1;1,1,1-2,2,2;1,2,1,1,2 Rev.Sim.: 0

301	 Tyz(5)	 Ax(0)	 Ay(0)	 Tyz(5)	 Ax(0)	 Ay(0)	3;5;3,4,4;0,0,0-2,1,2;0,0,2-2,2,3;0,1,0-1,3,2;1,1,0-2,2,1;1,1,1-2,2,2;1,1,2,2,1 Rev.Sim.: 0 
302	 S(3)	 Lx(5)	 Az(0)	 S(3)	 Lx(5)	 Az(0)	3;5;3,4,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-1,2,3;0,1,0-1,3,2;1,0,0-2,3,2;1,2,2,1,1 Rev.Sim.: 0 
303	 Lx(5)	 Txy(3)	 Az(0)	 Lx(5)	 Txy(3)	 Az(0)	3;5;3,4,4;0,0,0-1,1,1;0,0,1-1,2,3;0,1,0-1,2,1;0,2,0-1,3,2;1,0,0-2,3,2;1,1,2,2,2 Rev.Sim.: 0 
304	 Txz(0)	 Ay(7)	 Az(0)	 Txz(0)	 Ay(7)	 Az(0)	3;5;3,4,4;0,0,0-1,2,3;0,2,0-2,3,2;1,0,0-2,1,2;1,1,0-2,2,1;1,1,1-2,2,2;1,2,1,1,2 Rev.Sim.: 0 
305	 Tzy(3)	 Ax(0)	 Az(0)	 Tzy(3)	 Ax(0)	 Az(0)	3;5;3,4,4;0,0,0-2,2,1;0,0,1-1,2,3;0,2,0-2,3,2;1,0,1-2,1,2;1,1,1-2,2,2;1,2,1,2,1 Rev.Sim.: 0 
306	 S(5)	 Ax(2)	 Tyz(0)	 S(5)	 Ax(2)	Tyz(0)	3;5;3,4,4;0,0,0-1,1,2;0,0,2-1,2,3;0,1,0-1,3,2;1,0,0-2,2,3;1,2,0-2,3,1;1,1,2,1,2 Rev.Sim.: 0
307	S(0)	Az(7)	Tyz(0)	S(0)	Az(7)	Tyz(0)	3;5;3,4,4;0,0,0-1,3,2;0,0,2-2,2,3;1,0,0-2,1,2;1,1,0-2,3,1;1,1,1-2,2,2;1,2,1,1,2 Rev.Sim.: 0
308	Lx(5)	S(0)	Tyz(0)	Lx(5)	S(0)	Tyz(0)	3;5;3,4,4;0,0,0-2,1,2;0,0,2-2,2,3;0,1,0-1,3,2;1,1,0-2,2,2;1,2,0-2,3,1;1,1,2,2,2 Rev.Sim.: 0
309	Tyz(5)	Ly(0)	Tyz(0)	Tyz(5)	Ly(0)	Tyz(0)	3;5;3,4,4;0,0,0-2,1,2;0,0,2-2,2,3;0,1,0-1,3,2;1,1,0-2,3,1;1,1,1-2,2,2;1,1,2,2,1 Rev.Sim.: 0
310	Tyz(0)	Ly(7)	Tyz(0)	Tyz(0)	Ly(7)	Tyz(0)	3;5;3,4,4;0,0,0-2,1,3;0,1,0-1,3,2;0,1,2-1,2,3;1,1,0-2,3,1;1,1,1-2,2,3;1,1,2,1,2 Rev.Sim.: 0
311	Tzy(1)	Lx(0)	Tyz(0)	Tzy(1)	Lx(0)	Tyz(0)	3;5;3,4,4;0,0,0-2,1,3;0,1,0-2,3,1;0,1,1-1,2,3;0,2,1-1,3,2;1,1,1-2,2,3;1,1,2,2,1 Rev.Sim.: 0
312	Az(2)	S(5)	Tyz(0)	Az(2)	S(5)	Tyz(0)	3;5;3,4,4;0,0,0-2,3,1;0,0,1-1,1,2;0,0,2-1,2,3;0,1,1-1,3,2;1,0,1-2,2,3;1,2,2,1,2 Rev.Sim.: 0

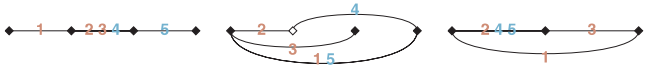
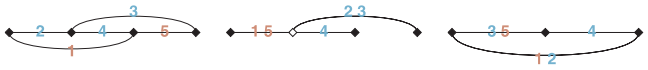
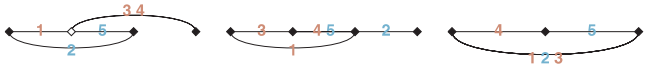
313	 S(3)	 Ax(4)	 Tzy(0)	 S(3)	 Ax(4)	 Tzy(0)	3;5;3,4,4;0,0,0-1,2,1;0,0,1-1,2,3;0,2,0-1,3,2;1,0,0-2,3,2;1,0,2-2,1,3;1,2,1,1,2 Rev.Sim.: 0 
314	 S(0)	 Ay(7)	 Tzy(0)	 S(0)	 Ay(7)	 Tzy(0)	3;5;3,4,4;0,0,0-1,2,3;0,2,0-2,3,2;1,0,0-2,1,3;1,1,0-2,2,1;1,1,1-2,2,2;1,2,1,1,2 Rev.Sim.: 0 
315	 Ay(4)	 S(3)	 Tzy(0)	 Ay(4)	 S(3)	 Tzy(0)	3;5;3,4,4;0,0,0-2,1,3;0,1,0-1,2,1;0,1,1-1,2,3;0,2,0-1,3,2;1,1,0-2,3,2;1,2,1,2,2 Rev.Sim.: 0 
316	 Tyz(1)	 Lx(0)	 Tzy(0)	 Tyz(1)	 Lx(0)	 Tzy(0)	3;5;3,4,4;0,0,0-2,1,3;0,1,0-1,3,1;0,1,1-1,2,3;0,2,1-1,3,2;1,1,0-2,3,2;1,1,2,2,1 Rev.Sim.: 0 
317	 Tzy(0)	 Lz(3)	 Tzy(0)	 Tzy(0)	 Lz(3)	 Tzy(0)	3;5;3,4,4;0,0,0-2,1,3;0,1,0-2,3,1;0,1,1-1,2,3;0,2,1-1,3,2;1,1,1-2,3,2;1,1,1,2,2 Rev.Sim.: 0 
318	 Lx(3)	 S(0)	 Tzy(0)	 Lx(3)	 S(0)	Tzy(0)	3;5;3,4,4;0,0,0-2,2,1;0,0,1-1,2,3;0,2,0-2,3,2;1,0,1-2,1,3;1,1,1-2,2,2;1,2,1,2,2 Rev.Sim.: 0
319	Tzy(3)	Lz(0)	Tzy(0)	Tzy(3)	Lz(0)	Tzy(0)	3;5;3,4,4;0,0,0-2,2,1;0,0,1-1,2,3;0,2,0-2,3,2;1,0,1-2,1,3;1,1,1-2,2,2;1,2,1,2,1 Rev.Sim.: 0
320	S(3)	Az(4)	Lx(0)	S(3)	Az(4)	Lx(0)	3;5;3,4,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-2,2,3;0,1,0-1,3,2;1,0,0-2,3,2;1,2,2,1,1 Rev.Sim.: 0
321	S(5)	Ay(2)	Lx(0)	S(5)	Ay(2)	Lx(0)	3;5;3,4,4;0,0,0-1,1,1;0,0,1-1,2,3;0,1,0-1,2,1;0,2,0-2,3,2;1,0,0-2,2,3;1,1,2,2,1 Rev.Sim.: 0
322	Txz(2)	Lx(5)	S(0)	Txz(2)	Lx(5)	S(0)	3;5;3,4,4;0,0,0-1,3,3;1,0,0-2,1,1;1,0,1-2,1,2;1,0,2-2,2,3;1,1,0-2,3,2;1,1,2,2,1 Rev.Sim.: 0
323	Txy(4)	Lx(3)	S(0)	Txy(4)	Lx(3)	S(0)	3;5;3,4,4;0,0,0-1,3,3;1,0,0-2,1,1;1,0,1-2,2,3;1,1,0-2,2,1;1,2,0-2,3,2;1,1,1,2,2 Rev.Sim.: 0
324	S(0)	Lx(0)	S(0)	S(0)	Lx(0)	S(0)	3;5;3,4,4;0,0,0-1,3,3;1,0,0-2,1,3;1,1,0-2,3,1;1,1,1-2,2,3;1,2,1-2,3,2;1,1,1,2,2 Rev.Sim.: 0

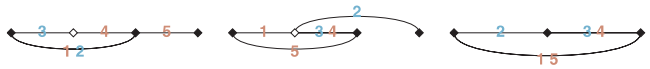
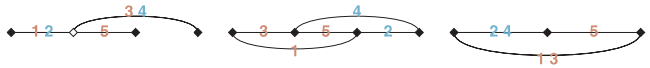
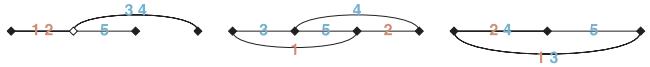
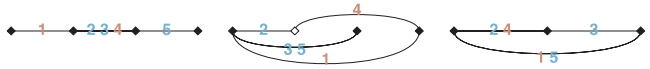
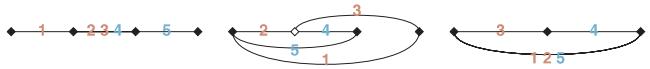
325	 Tyz(1)	 Ax(4)	 S(0)	 Tyz(1)	 Ax(4)	 S(0)	3;5;3,4,4;0,0,0-2,1,3;0,1,0-1,3,1;0,1,1-1,3,3;1,1,0-2,3,2;1,1,2-2,2,3;1,1,2,1,2 Rev.Sim.: 0 
326	 Ay(5)	 Txz(2)	 S(0)	 Ay(5)	 Txz(2)	 S(0)	3;5;3,4,4;0,0,0-2,1,3;0,1,0-1,3,3;1,1,0-2,2,1;1,1,1-2,2,3;1,2,0-2,3,2;1,2,2,1,2 Rev.Sim.: 0 
327	 Tzy(1)	 Ax(2)	 S(0)	 Tzy(1)	 Ax(2)	 S(0)	3;5;3,4,4;0,0,0-2,1,3;0,1,0-2,3,1;0,1,1-1,3,3;1,1,1-2,2,3;1,2,1-2,3,2;1,1,2,1,2 Rev.Sim.: 0 
328	 Az(3)	 Txy(4)	 S(0)	 Az(3)	 Txy(4)	 S(0)	3;5;3,4,4;0,0,0-2,3,1;0,0,1-1,3,3;1,0,1-2,1,2;1,0,2-2,2,3;1,1,1-2,3,2;1,2,2,2,1 Rev.Sim.: 0 
329	 S(6)	 Ly(1)	 Lz(0)	 S(6)	 Ly(1)	 Lz(0)	3;5;4,3,3;0,0,0-1,1,1;0,0,1-2,1,2;0,1,0-2,2,2;1,0,0-2,1,1;2,0,0-3,1,2;1,1,1,2,2 Rev.Sim.: 0 
330	 S(6)	 Lz(1)	 Ly(0)	 S(6)	 Lz(1)	Ly(0)	3;5;4,3,3;0,0,0-1,1,1;0,0,1-2,1,2;0,1,0-2,2,2;1,0,0-2,1,1;2,0,0-3,2,1;1,1,1,2,2 Rev.Sim.: 0
331	S(6)	Ax(1)	B(0)	S(6)	Ax(1)	B(0)	3;5;4,3,3;0,0,0-1,1,1;0,0,1-2,1,2;0,1,0-2,2,2;1,0,0-2,1,1;2,0,0-3,2,2;1,1,1,2,2 Rev.Sim.: 0
332	Ly(6)	Tyx(3)	B(0)	Ly(6)	Tyx(3)	B(0)	3;5;4,3,3;0,0,0-1,1,1;0,0,1-2,1,2;0,1,0-2,2,2;1,0,0-2,1,1;2,0,0-3,2,2;1,1,2,2,2 Rev.Sim.: 0
333	Lz(2)	Tzx(5)	B(0)	Lz(2)	Tzx(5)	B(0)	3;5;4,3,3;0,0,0-1,1,1;0,0,1-2,2,2;0,1,0-2,2,1;1,0,0-2,1,1;2,0,0-3,2,2;1,2,1,2,2 Rev.Sim.: 0
334	Ly(1)	Tyz(6)	Ax(0)	Ly(1)	Tyz(6)	Ax(0)	3;5;4,3,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-2,1,3;0,1,0-2,2,3;1,0,0-3,1,2;1,2,2,2,1 Rev.Sim.: 0
335	S(6)	Ly(1)	Ax(0)	S(6)	Ly(1)	Ax(0)	3;5;4,3,4;0,0,0-1,1,1;0,0,1-2,1,3;0,1,0-2,2,3;1,0,0-2,1,1;2,0,0-3,1,2;1,1,1,2,2 Rev.Sim.: 0
336	Tyx(0)	Az(7)	Ax(0)	Tyx(0)	Az(7)	Ax(0)	3;5;4,3,4;0,0,0-1,2,2;0,0,2-2,2,3;1,0,0-3,1,2;1,1,0-2,2,1;1,1,1-2,2,2;1,2,1,1,2 Rev.Sim.: 0

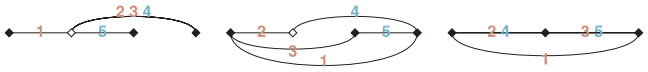
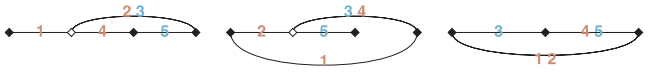
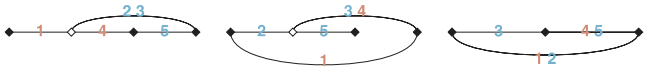
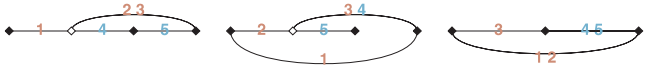
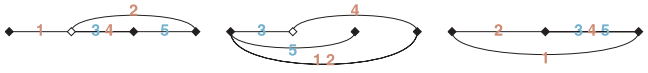
337	Txz(6)	Ay(0)	Ax(0)	Txz(6)	Ay(0)	Ax(0)	3;5;4,3,4;0,0,0-1,2,2;0,0,2-2,2,3;1,0,0-3,1,2;1,1,0-2,2,1;1,1,1-2,2,2;1,1,2,2,1 Rev.Sim.: 0
338	Txz(2)	Ly(7)	Tyx(0)	Txz(2)	Ly(7)	Tyx(0)	3;5;4,3,4;0,0,0-1,2,3;1,0,0-2,1,1;1,0,1-2,1,3;1,1,0-2,2,2;2,0,0-3,1,3;1,1,2,1,2
339	Lz(0)	Tyx(0)	Tyx(0)	Lz(0)	Tyx(0)	Tyx(0)	3;5;4,3,4;0,0,0-2,2,1;0,0,1-1,2,3;1,0,1-3,1,3;1,1,1-2,2,2;2,0,0-3,1,1;1,2,2,2,1
340	S(6)	Ay(1)	Txz(0)	S(6)	Ay(1)	Txz(0)	3;5;4,3,4;0,0,0-1,1,2;0,0,2-1,2,3;0,1,0-2,2,3;1,0,0-3,1,2;2,1,0-3,2,1;1,1,1,2,2
341	Ly(6)	S(0)	Txz(0)	Ly(6)	S(0)	Txz(0)	3;5;4,3,4;0,0,0-1,2,2;0,0,2-2,2,3;1,0,0-2,2,2;2,0,0-3,1,2;2,1,0-3,2,1;1,1,2,2,2
342	S(0)	Az(7)	Txz(0)	S(0)	Az(7)	Txz(0)	3;5;4,3,4;0,0,0-1,2,2;0,0,2-2,2,3;1,0,0-3,1,2;1,1,0-3,2,1;1,1,1-2,2,2;1,2,1,1,2
343	Txz(6)	Lx(0)	Txz(0)	Txz(6)	Lx(0)	Txz(0)	3;5;4,3,4;0,0,0-1,2,2;0,0,2-2,2,3;1,0,0-3,1,2;1,1,0-3,2,1;1,1,1-2,2,2;1,1,2,2,1
344	Txz(0)	Lx(6)	Txz(0)	Txz(0)	Lx(6)	Txz(0)	3;5;4,3,4;0,0,0-1,2,3;1,0,0-3,1,2;1,0,2-2,1,3;1,1,0-3,2,1;1,1,1-2,2,3;1,1,2,1,2
345	Tzx(2)	Ly(0)	Txz(0)	Tzx(2)	Ly(0)	Txz(0)	3;5;4,3,4;0,0,0-1,2,3;1,0,0-3,2,1;1,0,1-2,1,3;1,1,1-2,2,3;2,0,1-3,1,2;1,1,2,1,2
346	Az(1)	S(6)	Txz(0)	Az(1)	S(6)	Txz(0)	3;5;4,3,4;0,0,0-3,2,1;0,0,1-1,1,2;0,0,2-2,1,3;0,1,1-2,2,3;1,0,1-3,1,2;1,2,2,2,1
347	Lx(0)	Tyz(0)	Tyz(0)	Lx(0)	Tyz(0)	Tyz(0)	3;5;4,3,4;0,0,0-1,1,3;0,1,0-1,2,2;1,0,0-3,1,3;1,1,0-2,2,2;2,1,0-3,2,1;1,1,2,2,2
348	Tzx(2)	Ly(7)	Tyz(0)	Tzx(2)	Ly(7)	Tyz(0)	3;5;4,3,4;0,0,0-1,2,2;0,0,2-1,1,3;1,0,0-3,2,1;1,0,1-3,1,3;1,1,1-2,2,2;1,2,1,2,1

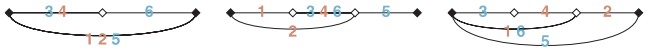
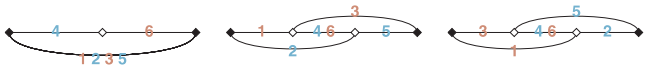
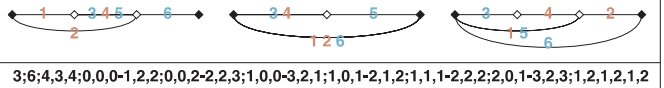
349	 S(3)	 Ly(6)	 Az(0)	 S(3)	 Ly(6)	 Az(0)	3;5;4,3,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-2,1,3;0,1,0-1,2,2;1,0,0-3,2,2;1,2,2,1,1 Rev.Sim.: 0 
350	 Ly(6)	 Tyx(3)	 Az(0)	 Ly(6)	 Tyx(3)	 Az(0)	3;5;4,3,4;0,0,0-1,1,1;0,0,1-2,1,3;0,1,0-2,2,2;1,0,0-2,1,1;2,0,0-3,2,2;1,1,2,2,2 Rev.Sim.: 0 
351	 Tyz(0)	 Ax(7)	 Az(0)	 Tyz(0)	 Ax(7)	 Az(0)	3;5;4,3,4;0,0,0-2,1,3;0,1,0-1,2,2;1,1,0-2,2,1;1,1,1-2,2,2;2,0,0-3,2,2;1,1,1,2,2 Rev.Sim.: 0 
352	 Tzx(3)	 Ay(0)	 Az(0)	 Tzx(3)	 Ay(0)	 Az(0)	3;5;4,3,4;0,0,0-2,2,1;0,0,1-2,1,3;0,1,1-1,2,2;1,1,1-2,2,2;2,0,0-3,2,2;1,2,2,1,1 Rev.Sim.: 0 
353	 Ax(4)	 S(3)	 Tzx(0)	 Ax(4)	 S(3)	 Tzx(0)	3;5;4,3,4;0,0,0-1,2,3;1,0,0-2,1,1;1,0,1-2,1,3;1,1,0-2,2,2;2,0,0-3,2,2;1,2,1,2,2 Rev.Sim.: 0 
354	 S(0)	 Ax(7)	 Tzx(0)	 S(0)	 Ax(7)	Tzx(0)	3;5;4,3,4;0,0,0-1,2,3;1,0,0-2,1,3;1,1,0-2,2,1;1,1,1-2,2,2;2,0,0-3,2,2;1,1,1,2,2 Rev.Sim.: 0
355	Txz(2)	Ly(0)	Tzx(0)	Txz(2)	Ly(0)	Tzx(0)	3;5;4,3,4;0,0,0-1,2,3;1,0,0-3,1,1;1,0,1-2,1,3;1,1,0-3,2,2;2,0,1-3,1,2;1,1,2,1,2 Rev.Sim.: 0
356	Tzx(0)	Lz(3)	Tzx(0)	Tzx(0)	Lz(3)	Tzx(0)	3;5;4,3,4;0,0,0-1,2,3;1,0,0-3,2,1;1,0,1-2,1,3;1,1,1-2,2,2;2,0,1-3,2,2;1,1,1,2,2 Rev.Sim.: 0
357	S(3)	Ay(4)	Tzx(0)	S(3)	Ay(4)	Tzx(0)	3;5;4,3,4;0,0,0-2,1,1;0,0,1-2,1,3;0,1,0-2,2,2;0,1,2-1,2,3;2,0,0-3,2,2;1,2,1,2,1 Rev.Sim.: 0
358	Ly(1)	S(0)	Tzx(0)	Ly(1)	S(0)	Tzx(0)	3;5;4,3,4;0,0,0-2,2,1;0,0,1-1,2,3;1,0,1-2,1,3;1,1,1-2,2,2;2,0,0-3,2,2;1,2,2,2,1 Rev.Sim.: 0
359	Tzx(3)	Lz(0)	Tzx(0)	Tzx(3)	Lz(0)	Tzx(0)	3;5;4,3,4;0,0,0-2,2,1;0,0,1-1,2,3;1,0,1-2,1,3;1,1,1-2,2,2;2,0,0-3,2,2;1,2,2,1,1 Rev.Sim.: 0
360	S(3)	Az(4)	Ly(0)	S(3)	Az(4)	Ly(0)	3;5;4,3,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-2,2,3;0,1,0-1,2,2;1,0,0-3,2,2;1,2,2,1,1 Rev.Sim.: 0

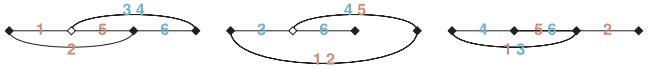
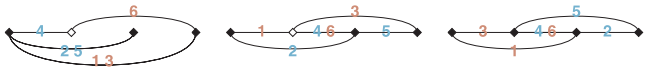
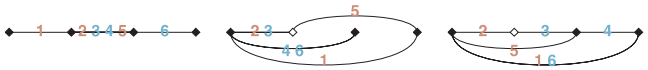
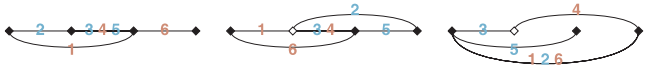
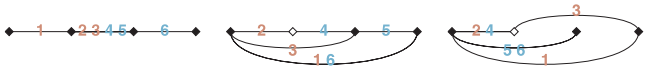
361	S(6)	Ax(1)	Ly(0)	S(6)	Ax(1)	Ly(0)	3;5;4,3,4;0,0,0-1,1,1;0,0,1-2,1,3;0,1,0-2,2,3;1,0,0-2,1,1;2,0,0-3,2,2;1,1,1,2,2 Rev.Sim.: 0
362	Txz(2)	Ay(4)	S(0)	Txz(2)	Ay(4)	S(0)	3;5;4,3,4;0,0,0-1,2,3;1,0,0-3,1,1;1,0,1-3,1,3;1,1,0-3,2,2;1,1,2-2,2,3;1,1,2,1,2 Rev.Sim.: 0
363	Tyx(4)	Ly(1)	S(0)	Tyx(4)	Ly(1)	S(0)	3;5;4,3,4;0,0,0-1,2,3;1,0,0-3,1,3;1,1,0-2,2,1;1,1,1-2,2,3;2,1,0-3,2,2;1,1,2,1,2 Rev.Sim.: 0
364	Ax(6)	Tyz(1)	S(0)	Ax(6)	Tyz(1)	S(0)	3;5;4,3,4;0,0,0-1,2,3;1,0,0-3,1,3;1,1,0-2,2,1;1,1,1-2,2,3;2,1,0-3,2,2;1,2,2,1,2 Rev.Sim.: 0
365	S(0)	Ly(0)	S(0)	S(0)	Ly(0)	S(0)	3;5;4,3,4;0,0,0-1,2,3;1,0,0-3,1,3;1,1,0-3,2,1;1,1,1-2,2,3;2,1,1-3,2,2;1,1,1,2,2 Rev.Sim.: 0
366	Tzx(2)	Ay(1)	S(0)	Tzx(2)	Ay(1)	S(0)	3;5;4,3,4;0,0,0-1,2,3;1,0,0-3,2,1;1,0,1-3,1,3;1,1,1-2,2,3;2,1,1-3,2,2;1,1,2,1,2 Rev.Sim.: 0
367	Tyz(1)	Ly(6)	S(0)	Tyz(1)	Ly(6)	S(0)	3;5;4,3,4;0,0,0-3,1,3;0,1,0-1,2,1;0,1,1-1,2,2;0,1,2-2,2,3;1,1,0-3,2,2;1,1,2,2,1 Rev.Sim.: 0
368	Az(3)	Tyx(4)	S(0)	Az(3)	Tyx(4)	S(0)	3;5;4,3,4;0,0,0-3,2,1;0,0,1-1,2,3;1,0,1-3,1,3;1,1,1-3,2,2;1,1,2-2,2,3;1,2,2,1,2 Rev.Sim.: 0
369	S(6)	Lz(1)	Ax(0)	S(6)	Lz(1)	Ax(0)	3;5;4,4,3;0,0,0-1,1,1;0,0,1-2,1,2;0,1,0-2,3,2;1,0,0-2,1,1;2,0,0-3,2,1;1,1,1,2,2 Rev.Sim.: 0
370	Lz(1)	Tzy(6)	Ax(0)	Lz(1)	Tzy(6)	Ax(0)	3;5;4,4,3;0,0,0-1,1,1;0,0,1-2,2,2;0,1,0-1,2,1;0,2,0-2,3,2;1,0,0-3,2,1;1,2,2,2,1 Rev.Sim.: 0
371	Tzx(0)	Ay(7)	Ax(0)	Tzx(0)	Ay(7)	Ax(0)	3;5;4,4,3;0,0,0-1,2,2;0,2,0-2,3,2;1,0,0-3,2,1;1,0,1-2,1,2;1,1,1-2,2,2;1,2,1,1,2 Rev.Sim.: 0
372	Txy(6)	Az(0)	Ax(0)	Txy(6)	Az(0)	Ax(0)	3;5;4,4,3;0,0,0-1,2,2;0,2,0-2,3,2;1,0,0-3,2,1;1,0,1-2,1,2;1,1,1-2,2,2;1,1,2,2,1 Rev.Sim.: 0

373	 Txy(4)	 Lz(3)	 Tzx(0)	 Txy(4)	 Lz(3)	 Tzx(0)	3;5;4,4,3;0,0,0-1,3,2;1,0,0-2,1,1;1,0,1-2,2,2;1,1,0-2,3,1;2,0,0-3,3,1;1,1,1,2,2 Rev.Sim.: 0 
374	 Ly(0)	 Tzx(0)	 Tzx(0)	 Ly(0)	 Tzx(0)	 Tzx(0)	3;5;4,4,3;0,0,0-2,1,2;0,1,0-1,3,2;1,1,0-3,3,1;1,1,1-2,2,2;2,0,0-3,1,1;1,2,2,2,1 Rev.Sim.: 0 
375	 S(6)	 Az(1)	 Txy(0)	 S(6)	 Az(1)	 Txy(0)	3;5;4,4,3;0,0,0-1,2,1;0,0,1-2,2,2;0,2,0-2,3,2;1,0,0-3,2,1;2,0,1-3,1,2;1,1,1,2,2 Rev.Sim.: 0 
376	 Lz(2)	 S(0)	 Txy(0)	 Lz(2)	 S(0)	 Txy(0)	3;5;4,4,3;0,0,0-1,2,2;0,2,0-2,3,2;1,0,0-2,2,2;2,0,0-3,1,2;2,1,0-3,2,1;1,1,2,2,2 Rev.Sim.: 0 
377	 S(0)	 Ay(7)	 Txy(0)	 S(0)	 Ay(7)	 Txy(0)	3;5;4,4,3;0,0,0-1,2,2;0,2,0-2,3,2;1,0,0-3,1,2;1,1,0-3,2,1;1,1,1-2,2,2;1,2,1,1,2 Rev.Sim.: 0 
378	 Txy(6)	 Lx(0)	 Txy(0)	 Txy(6)	 Lx(0)	Txy(0)	3;5;4,4,3;0,0,0-1,2,2;0,2,0-2,3,2;1,0,0-3,1,2;1,1,0-3,2,1;1,1,1-2,2,2;1,1,2,2,1 Rev.Sim.: 0
379	Tyx(4)	Lz(0)	Txy(0)	Tyx(4)	Lz(0)	Txy(0)	3;5;4,4,3;0,0,0-1,3,2;1,0,0-3,1,2;1,1,0-2,3,1;1,1,1-2,3,2;2,1,0-3,2,1;1,1,2,1,2 Rev.Sim.: 0
380	Txy(0)	Lx(6)	Txy(0)	Txy(0)	Lx(6)	Txy(0)	3;5;4,4,3;0,0,0-1,3,2;1,0,0-3,1,2;1,1,0-3,2,1;1,1,1-2,2,2;1,2,0-2,3,2;1,1,1,2,2 Rev.Sim.: 0
381	Ay(1)	S(6)	Txy(0)	Ay(1)	S(6)	Txy(0)	3;5;4,4,3;0,0,0-3,1,2;0,1,0-1,2,1;0,1,1-2,2,2;0,2,0-2,3,2;1,1,0-3,2,1;1,2,2,2,1 Rev.Sim.: 0
382	S(5)	Lz(2)	Ay(0)	S(5)	Lz(2)	Ay(0)	3;5;4,4,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-1,2,1;0,2,0-2,3,1;1,0,0-3,2,2;1,1,2,2,1 Rev.Sim.: 0
383	Lz(2)	Tzx(5)	Ay(0)	Lz(2)	Tzx(5)	Ay(0)	3;5;4,4,3;0,0,0-1,1,1;0,0,1-2,2,2;0,1,0-2,3,1;1,0,0-2,1,1;2,0,0-3,2,2;1,2,1,2,2 Rev.Sim.: 0
384	Tzy(0)	Ax(7)	Ay(0)	Tzy(0)	Ax(7)	Ay(0)	3;5;4,4,3;0,0,0-2,1,2;0,1,0-2,3,1;0,1,1-1,2,2;1,1,1-2,2,2;2,0,0-3,2,2;1,1,1,2,2 Rev.Sim.: 0

385	 Tyx(5)	 Az(0)	 Ay(0)	 Tyx(5)	 Az(0)	 Ay(0)	3;5;4,4,3;0,0,0-2,1,2;0,1,0-2,3,1;0,1,1-1,2,2;1,1,1-2,2,2;2,0,0-3,2,2;1,2,2,1,1 Rev.Sim.: 0 
386	 Tyx(4)	 Lz(3)	 Tzy(0)	 Tyx(4)	 Lz(3)	 Tzy(0)	3;5;4,4,3;0,0,0-1,2,2;0,2,0-1,3,1;1,0,0-3,1,2;1,1,0-3,3,1;1,1,1-2,2,2;1,2,1,2,1 Rev.Sim.: 0 
387	 Lx(0)	 Tzy(0)	 Tzy(0)	 Lx(0)	 Tzy(0)	 Tzy(0)	3;5;4,4,3;0,0,0-1,2,2;0,2,0-1,3,1;1,0,0-3,1,2;1,1,0-3,3,1;1,1,1-2,2,2;1,1,2,2,2 Rev.Sim.: 0 
388	 Ax(2)	 S(5)	 Tyx(0)	 Ax(2)	 S(5)	 Tyx(0)	3;5;4,4,3;0,0,0-1,3,2;1,0,0-2,1,1;1,0,1-2,2,2;1,1,0-2,3,1;2,0,0-3,2,2;1,2,2,1,2 Rev.Sim.: 0 
389	 S(0)	 Ax(7)	 Tyx(0)	 S(0)	 Ax(7)	 Tyx(0)	3;5;4,4,3;0,0,0-1,3,2;1,0,0-2,1,2;1,1,0-2,3,1;1,1,1-2,2,2;2,0,0-3,2,2;1,1,1,2,2 Rev.Sim.: 0 
390	 Txy(4)	 Lz(0)	 Tyx(0)	 Txy(4)	 Lz(0)	Tyx(0)	3;5;4,4,3;0,0,0-1,3,2;1,0,0-3,1,1;1,0,1-3,2,2;1,1,0-2,3,1;2,1,0-3,2,1;1,1,1,2,2 Rev.Sim.: 0
391	Tyx(0)	Ly(7)	Tyx(0)	Tyx(0)	Ly(7)	Tyx(0)	3;5;4,4,3;0,0,0-1,3,2;1,0,0-3,1,2;1,1,0-2,3,1;1,1,1-2,2,2;2,1,0-3,2,2;1,1,1,2,2 Rev.Sim.: 0
392	S(5)	Az(2)	Tyx(0)	S(5)	Az(2)	Tyx(0)	3;5;4,4,3;0,0,0-2,1,1;0,0,1-2,2,2;0,1,0-2,3,1;0,2,1-1,3,2;2,0,0-3,2,2;1,1,2,2,1 Rev.Sim.: 0
393	Lz(1)	S(0)	Tyx(0)	Lz(1)	S(0)	Tyx(0)	3;5;4,4,3;0,0,0-2,1,2;0,1,0-1,3,2;1,1,0-2,2,2;1,2,0-2,3,1;2,0,0-3,2,2;1,2,2,2,1 Rev.Sim.: 0
394	Tyx(5)	Ly(0)	Tyx(0)	Tyx(5)	Ly(0)	Tyx(0)	3;5;4,4,3;0,0,0-2,1,2;0,1,0-1,3,2;1,1,0-2,3,1;1,1,1-2,2,2;2,0,0-3,2,2;1,2,2,1,1 Rev.Sim.: 0
395	S(5)	Ay(2)	Lz(0)	S(5)	Ay(2)	Lz(0)	3;5;4,4,3;0,0,0-1,1,1;0,0,1-1,2,2;0,1,0-1,2,1;0,2,0-2,3,2;1,0,0-3,2,2;1,1,2,2,1 Rev.Sim.: 0
396	S(6)	Ax(1)	Lz(0)	S(6)	Ax(1)	Lz(0)	3;5;4,4,3;0,0,0-1,1,1;0,0,1-2,1,2;0,1,0-2,3,2;1,0,0-2,1,1;2,0,0-3,2,2;1,1,1,2,2 Rev.Sim.: 0

397	 Txy(4)	 Az(2)	 S(0)	 Txy(4)	 Az(2)	 S(0)	3;5;4,4,3;0,0,0-1,3,2;1,0,0-3,1,1;1,0,1-3,2,2;1,1,0-3,3,1;1,2,1-2,3,2;1,1,1,2,2 Rev.Sim.: 0 
398	 Tyx(4)	 Az(1)	 S(0)	 Tyx(4)	 Az(1)	 S(0)	3;5;4,4,3;0,0,0-1,3,2;1,0,0-3,1,2;1,1,0-3,3,1;1,1,1-2,3,2;2,1,1-3,2,2;1,1,2,1,2 Rev.Sim.: 0 
399	 Ax(6)	 Tzy(1)	 S(0)	 Ax(6)	 Tzy(1)	 S(0)	3;5;4,4,3;0,0,0-1,3,2;1,0,0-3,1,2;1,1,0-3,3,1;1,1,1-2,3,2;2,1,1-3,2,2;1,2,2,1,2 Rev.Sim.: 0 
400	 S(0)	 Lz(0)	 S(0)	 S(0)	 Lz(0)	 S(0)	3;5;4,4,3;0,0,0-1,3,2;1,0,0-3,1,2;1,1,0-3,3,1;1,1,1-2,3,2;2,1,1-3,2,2;1,1,1,2,2 Rev.Sim.: 0 
401	 Tzx(2)	 Lz(1)	 S(0)	 Tzx(2)	 Lz(1)	 S(0)	3;5;4,4,3;0,0,0-1,3,2;1,0,0-3,3,1;1,0,1-2,1,2;1,1,1-2,3,2;2,0,1-3,2,2;1,1,2,1,2 Rev.Sim.: 0 
402	 Ay(5)	 Tzx(2)	 S(0)	 Ay(5)	 Tzx(2)	S(0)	3;5;4,4,3;0,0,0-3,1,2;0,1,0-1,3,2;1,1,0-3,3,1;1,1,1-3,2,2;1,2,1-2,3,2;1,2,2,1,2 Rev.Sim.: 0
403	Tzy(1)	Lz(2)	S(0)	Tzy(1)	Lz(2)	S(0)	3;5;4,4,3;0,0,0-3,1,2;0,1,0-3,3,1;0,1,1-1,2,2;0,2,1-2,3,2;1,1,1-3,2,2;1,1,2,2,1 Rev.Sim.: 0
404	S(6)	Ax(1)	Ax(0)	S(6)	Ax(1)	Ax(0)	3;5;4,4,4;0,0,0-1,1,1;0,0,1-2,1,3;0,1,0-2,3,3;1,0,0-2,1,1;2,0,0-3,2,2;1,1,1,2,2 Rev.Sim.: 0
405	S(5)	Ay(2)	Ay(0)	S(5)	Ay(2)	Ay(0)	3;5;4,4,4;0,0,0-1,1,1;0,0,1-1,2,3;0,1,0-1,2,1;0,2,0-2,3,2;1,0,0-3,2,3;1,1,2,2,1 Rev.Sim.: 0
406	S(3)	Az(4)	Az(0)	S(3)	Az(4)	Az(0)	3;5;4,4,4;0,0,0-1,1,1;0,0,1-1,1,2;0,0,2-2,2,3;0,1,0-1,3,2;1,0,0-3,3,2;1,2,2,1,1 Rev.Sim.: 0
407	Tyz(4)	S(3)	Lx(0)	Tyz(4)	S(3)	Lx(0)	3;6;3,4,4;0,0,0-2,1,2;0,0,2-2,2,3;0,1,0-1,2,1;0,1,1-1,2,2;0,2,0-1,3,2;1,1,0-2,3,2;1,1,2,1,2,2 Rev.Sim.: 0
408	Tzy(2)	S(5)	Lx(0)	Tzy(2)	S(5)	Lx(0)	3;6;3,4,4;0,0,0-2,2,1;0,0,1-1,1,2;0,0,2-1,2,3;0,1,1-1,2,2;0,2,0-2,3,2;1,0,1-2,2,3;1,2,2,1,1,2 Rev.Sim.: 0

409	 Tyz(4)	 Tyz(3)	 B(0)	 Tyz(4)	 Tyz(3)	 B(0)	3;6;3,4,4;0,0,0-2,1,2;0,0,2-2,2,3;0,1,0-1,2,1;0,1,1-1,2,2;0,2,0-2,3,3;1,1,0-2,2,2;1,1,2,1,2,2 Rev.Sim.: 07 
410	 Tzy(1)	 Tzy(6)	 B(0)	 Tzy(1)	 Tzy(6)	 B(0)	3;6;3,4,4;0,0,0-2,1,2;0,0,2-2,2,3;0,1,0-2,3,1;0,1,1-1,2,2;0,2,1-2,3,3;1,1,1-2,2,2;1,2,1,2,2,1 Rev.Sim.: 07 
411	 Txz(4)	 S(3)	 Ly(0)	 Txz(4)	 S(3)	 Ly(0)	3;6;4,3,4;0,0,0-1,2,2;0,0,2-2,2,3;1,0,0-2,1,1;1,0,1-2,1,2;1,1,0-2,2,2;2,0,0-3,2,2;1,1,2,1,2,2 Rev.Sim.: 0 
412	 Tzx(1)	 S(6)	 Ly(0)	 Tzx(1)	 S(6)	 Ly(0)	3;6;4,3,4;0,0,0-2,2,1;0,0,1-1,1,2;0,0,2-2,1,3;0,1,1-2,2,3;1,0,1-2,1,2;2,0,0-3,2,2;1,2,2,2,1,1 Rev.Sim.: 0 
413	 Txz(4)	 Txz(3)	 B(0)	 Txz(4)	 Txz(3)	 B(0)	3;6;4,3,4;0,0,0-1,2,2;0,0,2-2,2,3;1,0,0-2,1,1;1,0,1-2,1,2;1,1,0-2,2,2;2,0,0-3,2,3;1,1,2,1,2,2 Rev.Sim.: 07 
414	 Tzx(2)	 Tzx(5)	 B(0)	 Tzx(2)	 Tzx(5)	B(0)	3;6;4,3,4;0,0,0-1,2,2;0,0,2-2,2,3;1,0,0-3,2,1;1,0,1-2,1,2;1,1,1-2,2,2;2,0,1-3,2,3;1,2,1,2,1,2 Rev.Sim.: 07
415	Txy(2)	S(5)	Lz(0)	Txy(2)	S(5)	Lz(0)	3;6;4,4,3;0,0,0-1,2,2;0,2,0-2,3,2;1,0,0-2,1,1;1,0,1-2,2,2;1,1,0-2,2,1;2,0,0-3,2,2;1,1,2,2,1,2 Rev.Sim.: 0
416	Tyx(1)	S(6)	Lz(0)	Tyx(1)	S(6)	Lz(0)	3;6;4,4,3;0,0,0-2,1,2;0,1,0-1,2,1;0,1,1-2,2,2;0,2,0-2,3,2;1,1,0-2,2,1;2,0,0-3,2,2;1,2,2,2,1,1 Rev.Sim.: 0
417	Txy(2)	Txy(5)	B(0)	Txy(2)	Txy(5)	B(0)	3;6;4,4,3;0,0,0-1,2,2;0,2,0-2,3,2;1,0,0-2,1,1;1,0,1-2,2,2;1,1,0-2,2,1;2,0,0-3,3,2;1,1,2,2,1,2 Rev.Sim.: 07
418	Tyx(4)	Tyx(3)	B(0)	Tyx(4)	Tyx(3)	B(0)	3;6;4,4,3;0,0,0-1,2,2;0,2,0-2,3,2;1,0,0-3,1,2;1,1,0-2,2,1;1,1,1-2,2,2;2,1,0-3,3,2;1,2,1,2,1,2 Rev.Sim.: 07
419	Txy(6)	Tyz(1)	Txy(0)	Txy(6)	Tyz(1)	Txy(0)	3;6;4,4,4;0,0,0-1,2,3;0,2,0-2,3,3;1,0,0-3,1,3;1,1,0-2,2,1;1,1,1-2,2,3;2,1,0-3,2,2;1,1,2,2,1,2 Rev.Sim.: 0
420	Tyz(1)	Tyz(6)	Txy(0)	Tyz(1)	Tyz(6)	Txy(0)	3;6;4,4,4;0,0,0-3,1,3;0,1,0-1,2,1;0,1,1-1,2,2;0,1,2-2,2,3;0,2,0-2,3,3;1,1,0-3,2,2;1,1,2,2,2,1 Rev.Sim.: 0

421	 Txz(6)	 Tzy(1)	 Txz(0)	 Txz(6)	 Tzy(1)	 Txz(0)	3;6;4,4,4;0,0,0-1,3,2;0,0,2-2,3,3;1,0,0-3,1,2;1,1,0-3,3,1;1,1,1-2,3,2;2,1,1-3,2,2;1,1,2,2,1,2 Rev.Sim.: 0 
422	 Tzy(1)	 Tzy(6)	 Txz(0)	 Tzy(1)	 Tzy(6)	 Txz(0)	3;6;4,4,4;0,0,0-3,1,2;0,0,2-2,2,3;0,1,0-3,3,1;0,1,1-1,2,2;0,2,1-2,3,3;1,1,1-3,2,2;1,2,1,2,2,1 Rev.Sim.: 0 
423	 Txz(2)	 Txz(5)	 Tyx(0)	 Txz(2)	 Txz(5)	 Tyx(0)	3;6;4,4,4;0,0,0-1,3,3;1,0,0-2,1,1;1,0,1-2,1,2;1,0,2-2,2,3;1,1,0-2,3,2;2,0,0-3,2,3;1,1,2,2,1,2 Rev.Sim.: 0 
424	 Tyx(5)	 Txz(2)	 Tyx(0)	 Tyx(5)	 Txz(2)	 Tyx(0)	3;6;4,4,4;0,0,0-2,1,3;0,1,0-1,3,3;1,1,0-2,2,1;1,1,1-2,2,3;1,2,0-2,3,2;2,0,0-3,2,3;1,2,2,1,2,1 Rev.Sim.: 0 
425	 Txy(4)	 Txy(3)	 Tzx(0)	 Txy(4)	 Txy(3)	 Tzx(0)	3;6;4,4,4;0,0,0-1,3,3;1,0,0-2,1,1;1,0,1-2,2,3;1,1,0-2,2,1;1,2,0-2,3,2;2,0,0-3,3,2;1,1,1,2,2,2 Rev.Sim.: 0 
426	 Tzx(3)	 Txy(4)	 Tzx(0)	 Tzx(3)	 Txy(4)	Tzx(0)	3;6;4,4,4;0,0,0-2,3,1;0,0,1-1,3,3;1,0,1-2,1,2;1,0,2-2,2,3;1,1,1-2,3,2;2,0,0-3,3,2;1,2,2,2,1,1 Rev.Sim.: 0
427	Tzx(2)	Tzx(5)	Tyz(0)	Tzx(2)	Tzx(5)	Tyz(0)	3;6;4,4,4;0,0,0-1,3,2;0,0,2-2,2,3;1,0,0-3,3,1;1,0,1-2,1,2;1,1,1-2,3,2;2,0,1-3,2,3;1,2,1,2,1,2 Rev.Sim.: 0
428	Tyz(5)	Tzx(2)	Tyz(0)	Tyz(5)	Tzx(2)	Tyz(0)	3;6;4,4,4;0,0,0-3,1,2;0,0,2-3,2,3;0,1,0-1,3,2;1,1,0-3,3,1;1,1,1-3,2,2;1,2,1-2,3,2;1,1,2,2,1,2 Rev.Sim.: 0
429	Tyx(4)	Tyx(3)	Tzy(0)	Tyx(4)	Tyx(3)	Tzy(0)	3;6;4,4,4;0,0,0-1,2,3;0,2,0-2,3,2;1,0,0-3,1,3;1,1,0-2,2,1;1,1,1-2,2,3;2,1,0-3,3,2;1,2,1,2,1,2 Rev.Sim.: 0
430	Tzy(3)	Tyx(4)	Tzy(0)	Tzy(3)	Tyx(4)	Tzy(0)	3;6;4,4,4;0,0,0-3,2,1;0,0,1-1,2,3;0,2,0-3,3,2;1,0,1-3,1,3;1,1,1-3,2,2;1,1,2-2,2,3;1,2,1,2,1,2 Rev.Sim.: 0

Bibliografia

[1] L. Lins, S. Lins and R. Morabito (2002). An L-Approach for Packing (ℓ, w)-Rectangles into Rectangular and L-Shaped Pieces. Aceito (janeiro de 2003) para publicação no *Journal of the Operational Research Society*.

[2] L. Lins, S. Lins and R. Morabito (2002). An n-tet graph approach for non-guillotine packings of n-dimensional boxes into an n-container. *European Journal of Operational Research* 141(2), 421-439.

[3] L. Lins, S. Lins and S. Melo (2002). PHORMA: Symmetry robust memory management of multidimensional arrays. Aceito (fevereiro de 2003) para publicação no *Discrete Applied Mathematics*.

[4] R. Morabito and S. Morales (1998). A simple and effective recursive procedure for the manufacturer's pallet loading problem. *Journal of the Operational Research Society*, 49, 819-828.

[5] G. Scheithauer and J. Terno (1996). The G4-heuristic for the pallet loading problem. *Journal of the Operational Research Society* 47, 511-522.

[6] J. Nelissen (1995). How to use structural constraints to compute an upper bound for the pallet loading problem. *European Journal of Operational Research* 84, 662-680.

[7] E. Bischoff and M. Ratcliff (1995). Loading multiple pallets. *Journal of the Operational Research Society* 46, 1322-1336.

[8] J. Nelissen (1994). Solving the pallet loading problem more efficiently. Working Paper, Graduiertenkolleg Informatik und Technik, Aachen.

[9] R. Tsai, E. Malstrom and W. Kuo (1993). Three dimensional palletization of mixed box sizes. *IEE Transactions* 25(4), 64-75.

[10] T. Cormen, C. Leiserson and R. Rivest (1990). *Introduction to algorithms*. The MIT Press, McGraw-Hill Book Company.

[11] K. Dowsland (1987). An exact algorithm for the pallet loading problem. *European Journal of Operational Research* 31, 78-84.

[12] K. Dowsland (1985). Determining an upper bound for a class of rectangular packing problems. *Computers and Operations Research* 12, 201-205.

[13] J. Beasley (1985). An exact two-dimensional non-guillotine tree search procedure. *Operations Research* 33, 49-64.

[14] K. Dowsland (1984). The three-dimensional pallet chart: an analysis of the factors affecting the set of feasible layouts for a class of two-dimensional packing problems. *Journal of the Operational Research Society* 35, 895-905.

[15] E. Bischoff and W. Dowsland (1982). An application of the micro to product design and distribution. *Journal of the Operational Research* 33, 271-280.

[16] A. Smith and P. De Cani (1980). An algorithm to optimize the layout of boxes in pallets. *Journal of the Operational Research Society* 31, 573-578.

[17] H. Steudel (1979). Generating pallet loading patterns: A special case of the two-dimensional cutting stock problem. *Management Science* 10, 997-1004.

[18] P. Sweeney and E. Paternoster (1969). Calculus of the Elementary Function *Hold, Rinehart and Winston.*