# 4
# O Problema de Escalonamento em Processadores Paralelos Idênticos ($P||C_{\max}$)

Um dos componentes da heurística híbrida de melhoria `HI_BP`, apresentada no capítulo anterior para o problema de *bin packing*, é o uso de abordagem dual que remete ao problema de escalonamento em processadores paralelos idênticos ($P||C_{\max}$). Neste capítulo investiga-se a aplicabilidade da heurística `HI_BP` a instâncias deste problema.



## 4.1
## Definição do Problema

Considera-se o problema de escalonamento de tarefas em processadores paralelos idênticos ($P||C_{\max}$) definido pelo conjunto de $n$ tarefas $T = \{t_1, \ldots, t_n\}$ com tempos de processamento $w_i$ $(i = 1, \ldots, n)$ e pelo conjunto de $m$ processadores $P = \{P_1, \ldots, P_m\}$, $m \geq 2$, cada um deles podendo processar uma única tarefa de cada vez. O problema consiste em alocar cada uma das $n$ tarefas a exatamente um dos $m$ processadores de forma a minimizar o tempo de escalonamento, dado pelo maior tempo de processamento entre todos os processadores (*makespan*). Utiliza-se $P_j$ para representar tanto o $j$-ésimo processador, assim como também o conjunto das tarefas nele alocadas. A função $w(P_j) = \sum_{i \in P_j} w_i$ fornece o tempo de processamento necessário para executar todas as tarefas no processador $P_j$. Uma solução $S = (P_1, \ldots, P_m)$ é representada pela lista das tarefas alocadas aos processadores, com *makespan* $C_{\max}(S) = \max_{j=1}^{m} w(Pj)$. Seja $z_{P||C_{\max}}(T, m)$ o valor da solução ótima do problema caracterizado pelo conjunto de tarefas $T$ e pelo número $m$ de processadores. Quando não houver ambiguidade, $z_{P||C_{\max}}$ será usado para denotar $z_{P||C_{\max}}(T, m)$. Este problema é NP-difícil [10, 32].

Assume-se que as tarefas são ordenadas de tal modo que

$$w_1 \geq w_2 \geq \cdots \geq w_n. \tag{4-1}$$

Além disso, assume-se, sem perda de generalidade, que os tempos de processamento $w_i$ são inteiros positivos.

No contexto dos problemas de escalonamento, o problema de *bin packing* pode ser visto como aquele de encontrar o menor número de processadores necessários para processar todas as tarefas dentro de um certo limite de tempo.

## 4.2 Revisão Bibliográfica

O problema de escalonamento em processadores paralelos idênticos ($P||C_{\max}$) é um dos mais estudados dentro da teoria de seqüenciamento e escalonamento determinísticos, ver Graham et al. (1979) [41], Cheng e Sin (1990) [11] e Lawler et al. (1993) [60]. Nesta seção faz-se um breve resumo dos principais algoritmos de aproximação para este problema, para os quais a razão de execução de pior caso é conhecida. Além disso, apresentam-se alguns exemplos de métodos exatos e heurísticos. Em particular, destacam-se as constribuições que serviram de base para a heurística proposta no próximo capítulo.

Considerando a possibilidade de preempção (pode-se interromper uma tarefa), McNaughton (1959) [67] introduz o seguinte limite inferior:

$$L_1 = \max\left\{\left\lceil \frac{1}{m}\sum_{i=1}^{n} w_i \right\rceil, \max_{i=1,\ldots,n}\{w_i\}\right\}.$$

Este limite diz que o escalonamento vai ser no mínimo tão longo quanto o máximo entre a carga média dos processadores (limite $L_0$) e o maior tempo de processamento entre todas as tarefas (impede-se que uma tarefa seja executada em paralelo em diversos processadores).

A classe de algoritmos baseados em "processamento de listas" (*list scheduling algorithms*) considera uma lista de prioridades de tarefas. A próxima tarefa disponível é escalonada para o primeiro processador disponível. Graham (1966) [38] prova que qualquer algoritmo baseado em "processamento de listas" para $P||C_{\max}$ possui razão absoluta de pior caso $r(\mathrm{LS}) = 2-1/m$. Posteriormente, Graham (1969) [39] mostra que se a tarefa selecionada for a mais longa (*longest processing time* LPT), a razão absoluta de execução de pior caso é $r(\mathrm{LPT}) = 4/3 - 1/3m$. Sua complexidade $O(n\log n + n\log m)$ é dominada pela ordenação inicial.

O algoritmo de aproximação MULTIFIT (MF) proposto por Coffman et al. (1978) [13] explora a relação entre BP e $P||C_{\max}$ procurando encontrar,

através de pesquisa binária, a menor capacidade $C$ (*makespan*) tal que a solução obtida pela heurística FFD para empacotar as $n$ tarefas (objetos) use no máximo $m$ processadores (caixas). Na busca binária, utilizam o limite inferior igual a $L_1$ e o superior igual ao máximo entre $2\,L_0$ e $L_1$. Os autores mostram que se a heurística FFD for aplicada $k$ vezes, MF($k$) (a versão do algoritmo MF que especifica o número de $k$ iterações) possui complexidade $O(n \log n + kn \log m)$ e provam que sua razão de pior caso é $r(\mathrm{MF}(k)) = 1.22 + 2^{-k}$. Comparam o desempenho de MF($k$) com o de LPT. Quando $k > m$, LPT é superior em relação ao tempo computacional. Entretanto, os autores mostram que um bom valor de $k$ é 7. Assim, para valores grandes de $n$, os dois algoritmos são dominados pela ordenação inicial. Posteriormente, Friesen [29] (1984) melhorou este limite para $r(\mathrm{MF}(k)) = 1.20 + 2^{-k}$ e Yue [90] (1990) para 13/11. Friesen e Langston (1986) [30] refinaram MF($k$) para obter o algoritmo melhorado MFI($k$) com mesma complexidade $O(n \log n + kn \log m)$ e razão de pior caso $r(\mathrm{MFI}(k)) = 72/61 + 2^{-k}$.

Finn e Horowitz (1979) [24] apresentam o algoritmo 0/1-INTERCHANGE. Inicialmente, as tarefas são atribuídas aos processadores de forma aleatória. A seguir, considera-se os processadores ordenados em ordem não crescente de carga e a diferença entre os tempos de processamento dos processadores mais carregado e menos carregado. Se existe uma tarefa no processador mais carregado cujo tempo de processamento é menor do que esta diferença, então esta tarefa é transferida para o processador menos carregado. Este procedimento é repetido até que não haja mais tarefas nesta situação. Uma característica deste método é que não há necessidade de ordenação inicial das tarefas. Sua complexidade é $O(n \log m)$ e sua razão de execução de pior caso é 2. Os autores mostram que, considerando-se o número de tarefas no processador mais carregado e fazendo o valor deste parâmetro maior do que 6, a razão de pior caso é menor do que 1.22 para qualquer valor de $m$. Comparam a heurística 0/1-INTERCHANGE com LPT para um conjunto de problemas testes em que os tempos de processamento foram gerados segundo diversas distribuições. Pouca diferença existe entre a qualidade das soluções dos dois algoritmos.

Blackstone e Phillips (1981) [7] introduzem uma heurística simples para melhorar a solução obtida por LPT, quando esta não é comprovadamente ótima, isto é, o valor obtido é diferente do limite $L_0$. A heurística tenta identificar pares de tarefas que, se trocadas entre os processadores, reduzem seu *makespan*. Para o caso especial em que $m = 2$, mostram as condições necessárias para que um *makespan* ótimo possa ser obtido através destas

trocas. Experimentos computacionais foram realizados com 700 instâncias. Os autores observam que a heurística LPT não produz soluções de qualidade quando o intervalo de tempos de processamento possíveis é pequeno. A heurística proposta resolve de forma exata a maioria das instâncias nestas circunstâncias.

Langston (1982) [59] observa que a alocação aleatória das $m$ tarefas iniciais em 0/1-INTERCHANGE pode gerar escalonamentos muito ruins e propõe duas alterações. O algoritmo *interchange with improvement* (ICI) fixa as $m$ tarefas mais pesadas, uma em cada um dos $m$ processadores, e não permite que elas sejam trocadas nos passos seguintes determinados pela regra de 0/1-INTERCHANGE. A razão de pior caso é $r(\text{ICI}) = 3/2 - 1/(2m)$. O segundo algoritmo ICII coloca as $2m$ tarefas mais pesadas usando a mesma regra que a utilizada pela heurística LPT e em seguida usa a regra de 0/1-INTERCHANGE. A razão de pior caso é $r(\text{ICII}) = 4/3 - 1/3m$. Estas modificações melhoram a razão de pior caso da heurística 0/1-INTERCHANGE significativamente, sem comprometer a complexidade de $O(n \log m)$.

O método de diferenciação de Karmarkar e Karp (1982) [55], descrito na Seção 3.2 para o caso especial em que $m = 2$, é extensivo para um número arbitrário de processadores e produz escalonamentos em que a diferença dos tempos de processamento dos processadores mais e menos carregado é $O(n^{-c \log n})$, para um valor positivo $c$.

Błazewicz (1987) [8] propõe um algoritmo exato baseado em uma abordagem que usa programação dinâmica. Sua complexidade é $O(nC^m)$, onde $C$ é um limite superior do *makespan*. Assim, para valores fixos de $m$, o algoritmo proposto é pseudopolinomial e, para valores pequenos de $m$ e $C$, é computacionalmente viável.

Hochbaum e Shmoys (1987) [48] propõem uma nova abordagem para a construção de algoritmos de aproximação dual (que será explicada detalhadamente na Seção 4.2.1), em que o objetivo é encontrar soluções inviáveis porém superótimas. Sugerem que esta nova abordagem seja considerada para outros problemas de otimização combinatória. Os autores mostram que encontrar uma $\epsilon$-aproximação para o problema $P||C_{\max}$ pode ser reduzido ao problema de encontrar uma $\epsilon$-aproximação dual para BP. Em particular, apresentam uma $\epsilon$-aproximação dual para BP com $\epsilon = 1/5$. Este algoritmo gera uma solução com garantia de que o número de caixas não é maior do que o valor ótimo e a capacidade da caixa é ultrapassada no máximo em 1/5 de seu valor. Aplicando a mesma abordagem de MF e substituindo-se o algoritmo FFD por este de aproximação dual, obtém-se uma $1/5 + 2^{-k}$-

aproximação para o problema $P||C_{\max}$ (razão equivalente à do MF) com complexidade $O(n(k + \log n))$, onde $k$ é o número de iterações da busca. Anteriormente, Sahni [73] (1976) e Graham [39] (1969) apresentaram esquemas de $\epsilon$-aproximação para valores fixos de $m$, porém com complexidade exponencial em $m$.

Lee e Massey (1988) [62] apresentam uma extensão do algoritmo MULTIFIT [13] chamada EXTENDED. Substituem a heurística FFD pelo algoritmo *Repetitive Modified Greedy* (RMG) da seguinte forma. Inicialmente, consideram-se as tarefas em ordem não-crescente de tempos de processamento, um limite dado para o *makespan* e um processador. Sucessivamente, colocam-se as tarefas no processador enquanto seu tempo total de processamento não atingir o limite dado. Este processo é equivalente à heurística gulosa padrão para o problema de soma de conjuntos (*Standard Greedy* - SG). Aplicando-se iterativamente SG a uma instância de $P||C_{\max}$ produz-se uma solução semelhante à gerada pelo procedimento FFD para a mesma instância. A versão modificada RMG, em cada iteração, compara os resultados obtidos por duas versões de SG. A primeira versão considera o conjunto completo de tarefas e a outra versão elimina do conjunto de tarefas aquela mais longa. A versão de SG que gerou o maior *makespan* determina o conjunto de tarefas a serem escalonadas para o processador corrente. No final do procedimento RMG tem-se, assim como quando usando-se FFD, o número de processadores necessários para escalonar todas as tarefas. Em seguida, EXTENDED aplica RMG da mesma forma como MULTFIT aplica FFD. A complexidade é igual à do MULTIFIT. Os autores provam que, para o problema restrito a dois processadores, a razão de pior caso é $r(\text{EXTENDED}(k)) = 1.1 + 2^{-k}$ onde $k$ é o número de iterações da busca binária.

Posteriormente, Lee e Massey (1988) [63] combinam os algoritmos LPT e MF para compor uma nova heurística denominada COMBINE. A motivação é ter um algoritmo cuja complexidade nunca seja pior do que a do LPT e o desempenho de pior caso nunca seja pior do que o do MF. A partir de uma solução inicial gerada pela heurística LPT, COMBINE aplica MF. Os limites inferior e superior utilizados na busca binária são mais fortes do que os utilizados por MF. Na pior hipótese, o número de iterações é o mesmo do que o número de iterações usado por MF, embora na prática este número costuma ser menor. A qualidade da solução é melhor ou igual à gerada por LPT. Os autores provam que a razão de pior caso nunca será pior do que a do MF. Sua complexidade é igual à do MF, isto é, $O(n \log n + kn \log m)$.

Müller (1993) [70] e França et al. (1994) [26] apresentam o algoritmo

3-PHASE composto das seguintes fases: na primeira, as tarefas são classificadas de acordo com seus tempos de processamento e alocadas a processadores de forma a obter cargas razoavelmente balanceadas. Assim como em 0/1-INTERCHANGE, não há necessidade de ordenação nesta fase. Na próxima fase, tarefas do processador mais carregado são transferidas para o processador menos carregado, com o objetivo de melhorar o balanceamento da carga. Por fim, na terceira e última fase, um balanceamento de carga mais sofisticado é realizado envolvendo a troca de duas tarefas entre dois processadores. Um experimento computacional é realizado envolvendo instâncias com tempos de processamento selecionados de três intervalos distintos considerando-se distribuições uniforme, normal e exponencial. Em [26] apresentam somente os resultados da distribuição uniforme, visto que, segundo observação dos autores, as demais distribuições mostram basicamente o mesmo comportamento. O algoritmo é comparado com as heurísticas LPT, 0/1-INTERCHANGE e a versão melhorada de Langston.

Hübscher e Glover (1994) [49] propõem um algoritmo de busca tabu que usa 2-*exchanges* e utiliza *influential diversification* (ver Seção 2.2.3).

Ho e Wong (1995) [45] apresentam o algoritmo *Two Machine Optimal Scheduling* (TMO) para a solução exata do problema, baseado em busca exaustiva, considerando o caso particular em que $m = 2$. Uma extensão deste algoritmo (X-TMO) é proposta para o caso com $m \geq 3$ processadores. Dada uma solução inicial (obtida por qualquer heurística), considera-se dois processadores alvo, os de maior e menor carga no problema inicial, e aplica-se o algoritmo TMO para a solução exata deste problema particular. Caso o resultado melhore o *makespan* corrente, a solução do problema original é atualizada. Este processo é repetido para os processadores de maior carga e de menor carga ainda não considerados, enquanto suas cargas forem diferentes e tal par existir. Extenso estudo computacional é realizado para os dois casos comparando as heurísticas LPT, MF e *Repetitive Modified Greedy* (RMG) de Lee e Massey [62]. Para o primeiro caso, em que se conhece a solução ótima obtida por TMO, observam que o desempenho das três heurísticas é melhor quando o número de tarefas cresce. Enquanto RMG possui um desempenho melhor do que o de LPT quando $n$ é pequeno, o contrário acontece quando $n$ é grande. Para o caso geral, X-TMO mostra ser capaz de melhorar consideravelmente as soluções iniciais geradas por LPT, MF e RMG.

Dell'Amico e Martello (1995) [16] desenvolvem um algoritmo de *branch-and-bound*, daqui por diante referenciado por B&B, para a solução exata para $P||C_{\max}$. A abordagem MF apresentada por Coffman e posteri-

ormente utilizada por Hochbaum, substituindo o FFD por um algoritmo de aproximação dual, também foi por eles utilizada usando, na busca binária, a versão aproximada do algoritmo MTP de Martello e Toth [65]. MTP, como citado no capítulo anterior, é um algoritmo de *branch-and-bound* para a solução exata de BP. Sua versão aproximada é obtida especificando-se um limite para o número de *backtrackings* permitidos. Nos casos em que MTP retorna a solução exata (quando o número de *backtrackings* necessários é menor do que o permitido) e o valor retornado é maior do que o número $m$ de processadores, então é possível melhorar o limite inferior do problema $P||C_{\max}$. Segundo os autores, apesar desta abordagem ser capaz de resolver otimamente diversas instâncias, o algoritmo exato proposto para $P||C_{\max}$ é superior. Outra heurística proposta, chamada MULTI-SUBSET (MS), tem duas fases. Na primeira, considera-se um processador de cada vez e nele coloca-se o subconjunto das tarefas ainda não escalonadas de valor mais próximo a um determinado limite inferior. Se todas as tarefas foram escalonadas, tem-se uma solução ótima. Para determinar o subconjunto de tarefas, usam o algoritmo $G^2$ proposto por Martello e Toth [65] para o problema de *Subset Sum*. A segunda fase considera as tarefas restantes em ordem decrescente de tempo de processamento e os processadores em ordem crescente de carga, colocando cada tarefa em um processador. As heurísticas LPT, MS e a abordagem de MF com MTP são utilizadas para gerar uma solução inicial para o *branch-and-bound*. Outra contribuição dos autores é a introdução de novos limites inferiores, que serão descritos com detalhes na Seção 4.2.2.

Scholl e Voß (1996) [78] consideram duas versões do problema clássico de balanceamento em linhas de produção (*simple assembly line balancing* - SALBP). SALBP-1 consiste em alocar tarefas a estações de trabalho de forma que, para uma dada taxa de produção (medida em função do tempo de execução de um ciclo de produção), o número necessário de estações é minimizado. SALBP-2 consiste em maximizar a taxa de produção ou, de forma equivalente, minimizar a soma de tempos ociosos para um dado número de estações. Nos dois casos deve-se considerar restrições de precedência. SALBP-1 se reduz ao problema de *bin packing*, enquanto que SALBP-2, se as restrições de precedência são omitidas, reduz-se ao problema de escalonamento em processadores paralelos idênticos. Os autores descrevem heurísticas construtivas para o problema SALBP-1, com regras de prioridade dinâmicas. Estas heurísticas são aplicadas, de forma iterativa e usando diferentes métodos de busca, com o objetivo de resolver o problema SALBP-2. Os métodos de busca considerados são: método de limite inferior, método de limite superior, busca binária, busca de Fibonacci e variações.

Consideram a abordagem de busca tabu. Discutem a dificuldade de se definir um movimento de busca tabu para o problema SALBP-1. O movimento natural baseado em troca de tarefas entre processadores dificilmente produz efeito no valor da função objetivo. Considera-se, por exemplo, uma solução viável cujo valor da função objetivo é $m$ estações. Para diminuir este valor seria necessário remover todas as tarefas de uma dada estação. Os possíveis movimentos podem ser categorizados em três tipos: $m$ é reduzido de uma unidade, $m$ permanece igual ou $m$ é incrementado de uma unidade. A quantidade de movimentos possíveis diminui a medida que $m$ se aproxima do valor ótimo. Considerando-se ainda as restrições de precedência, uma situação possível é a existência somente de movimentos do segundo ou terceiro tipos, sem distinção de como avaliar qual é o melhor. Para resolver este problema, usam a relação entre SALBP-1 e SALBP-2 (que chamaram de relação dual) para propor a seguinte estrutura de um método genérico de busca. A abordagem de busca tabu [37][pág. 296] para o problema SALBP-2 tenta encontrar uma solução viável com $m$ estações e tempo do ciclo de produção não maior do que um dado valor. Se tal solução for encontrada, $m$ é um limite superior no número de estações, do contrário a busca continua com valores maiores para $m$. O procedimento BISON [77] para a solução exata do BP, apresentado na Seção 2.2.1, usa esta abordagem. A heurística híbrida `HI_BP` proposta nesta tese, também. Antes, porém, no contexto de algoritmos para o $P||C_{\max}$, Coffman et al. [13], Hochbaum e Shmoys [48] e Dell'Amico e Martello [16], apresentaram abordagem semelhante. Hübscher e Glover [49] também exploraram a relação entre $P||C_{\max}$ e BP.

Fatemi-Ghomi e Jolai-Ghazvini (1998) [21] propõem uma busca local baseada na troca de um par de tarefas de diferentes processadores, visando minimizar a soma das diferenças de tempos de término entre todos os pares de processadores. Foi realizada uma análise do comportamento do algoritmo em relação à variação de três fatores: solução inicial, distribuição dos tempos de processamento e tamanho do problema. O maior erro percentual foi observado para a distribuição uniforme e problemas pequenos. Foram propostas quatro diferentes estratégias para a construção da solução inicial. Considerou-se uma versão do algoritmo para cada uma destas quatro estratégias e uma quinta versão composta por uma iteração de cada uma delas. Considerou-se 80 instâncias do pior caso (distribuição uniforme e problemas pequenos) e comparou-se o desempenho das cinco versões do algoritmo com os resultados obtidos pelas heurísticas LPT e MF. Os melhores resultados foram obetidos pela versão composta.

Algoritmos de melhoria baseados em fluxo em redes são métodos de

busca local que executam sequências de movimentos dependentes (chamados *multi-exchanges*) e que utilizam técnicas de fluxo em redes para identificar movimentos promissores. Baseados nessa abordagem, Frangioni et al. (2000) [27, 28] propõem diversos algoritmos de melhoria para o problema de escalonamento em processadores paralelos, que executam diversas trocas de tarefas entre processadores. Apresentam um trabalho metodológico de estudo de diferentes técnicas de construção de estruturas de vizinhanças para este problema, explorando propriedades peculiares dos grafos e investigando em que condições as novas estruturas podem produzir soluções de qualidade. O problema com processadores idênticos ($P||C_{\max}$) é selecionado para um estudo de caso. Os autores disponibilizam instâncias de uma nova classe "altamente não-uniforme" de problemas testes.

Recentemente, Costa et al. (2002) [15] apresentam heurística de busca local inspirada em sistemas imunológicos artificiais. Experimentos computacionais são realizados com dois grupos de problemas testes: a classe de problemas uniformes criadas por França et. al [26] e 25 instâncias com tempos de processamento muito grandes e apenas dois processadores. A abordagem proposta mostra melhores resultados em especial para esta última classe de problemas.



### 4.2.1 Relação de Dualidade

Nesta seção apresentam-se noções de algoritmos de aproximação dual, segundo Hochbaum e Shmoys [47, 48], e uma $\epsilon$-aproximação para o problema $P||C_{\max}$.

Tradicionalmente, algoritmos de aproximação primais tentam encontrar soluções viáveis sub-ótimas, onde a medida de qualidade é dada pelo grau de afastamento em relação ao valor ótimo. Alternativamente, algoritmos de aproximação dual buscam soluções superótimas, porém não viáveis, onde a medida de qualidade é medida pelo grau de inviabilidade. Para o problema BP, o primeiro tipo de algoritmo produz soluções viáveis em que o número de caixas utilizadas pode ser superior ao valor ótimo. O segundo tipo encontra empacotamentos aproximados onde no máximo o número ótimo de caixas é utilizado, porém permite-se ultrapassar a capacidade das caixas. Existem aplicações práticas que se enquadram neste modelo: seja, por exemplo, um conjunto de tarefas que devem ser alocadas a um conjunto de recursos idênticos (trabalhadores, por exemplo) que estão disponíveis por um custo fixo por um determinado tempo (o tempo regular) e que podem

ser usados por um custo adicional, proporcional ao tempo (o tempo extra), se necessário [17].

Seja $z_{BP}(N, C)$ o valor da solução ótima do problema BP caracterizado pelo conjunto de objetos $N$ e pela capacidade da caixa $C$. Analogamente $z_{P||C_{\max}}(N, m)$ é o valor da solução ótima do problema $P||C_{\max}$ caracterizado pelo mesmo conjunto $N$ de tarefas e por $m$ processadores. Observa-se que $z_{BP}(N, C) \leq m$ se e somente se $z_{P||C_{\max}}(N, m) \leq C$. Logo, se existe um procedimento para resolver otimamente o problema BP, este pode ser utilizado em conjunto com pesquisa binária para obter um solução ótima para o problema $P||C_{\max}$. Uma analogia natural seria fazer a mesma consideração para algoritmos de aproximação. Esta é a abordagem utilizada em MULTIFIT (MF) [13]. No entanto, segundo Hochbaum e Shmoys, utilizar um algoritmo para o problema BP como uma "caixa preta" não é garantia de sucesso, a ferramenta ideal seria usar o dual do BP.

Considera-se o problema de decisão onde existem dois parâmetros críticos, por exemplo o número de caixas (processadores) e a capacidade da caixa (*makespan*): "Existe empacotamento viável do conjunto $N$ de objetos em $m$ caixas com capacidade $C$?". Dois problemas de otimização podem ser gerados deste problema de decisão fixando-se um dos dois parâmetros e então otimizando-se o outro sujeito às restrições. Então, as seguintes considerações podem ser feitas:

1. Problema $P||C_{\max}$: dados $N$ e o número de processadores $m$, minimizar o *makespan* $C$, sendo $z_{P||C_{\max}}(N, m)$ seu valor ótimo. Uma $\epsilon$-aproximação primal obtém solução com *makespan* no máximo $(1 + \epsilon)\, z_{P||C_{\max}}(N, m)$ e número de processadores $m$.

2. Problema BP: dados $N$ e a capacidade da caixa $C$, minimizar o número de caixas $m$, sendo $z_{BP}(N, C)$ o valor da solução ótima. Uma $\epsilon$-aproximação dual obtém solução com no máximo $z_{BP}(N, C)$ caixas e capacidade até $(1 + \epsilon)\, C$.

Encontrar uma $\epsilon$-aproximação primal para o problema $P||C_{\max}$ equivale a encontrar uma $\epsilon$-aproximação dual para o problema BP.

No Apêndice B apresenta-se o pseudo-código do algoritmo `1/5-dual-BP` para o problema BP. Os autores provam que o número de caixas utilizadas nunca ultrapassa $z_{BP}(N, C)$ e que nenhuma caixa pesará mais do que $6/5$. Para finalizar esta seção, como o objetivo primeiro do trabalho citado é apresentar uma $\epsilon$-aproximação para o problema $P||C_{\max}$, Hochbaum e Shmoys utilizam a mesma abordagem de MF [13] substituindo

```
Procedimento AHS
Entrada:    N = {1,...,n}, w_i (i = 1,...,n), L, U, m
Saída:      S = {P_1,...,P_m}, L_HS.
1  esquerda ← L;
2  direita ← U;
3  enquanto esquerda < direita − 1 faça
4      C ← ⌊(esquerda + direita)/2⌋;
5      S', m' ← 1/5_dual_BP(N, w_i/C);
6      se m' > m então esquerda ← C;
7      senão direita ← C; S ← S';
8  fim-enquanto
9  S', m' ← 1/5_dual_BP(N, w_i/esquerda);
10 se m' > m então faça
11     S, m' ← 1/5_dual_BP(N, w_i/direita);
12     L_HS ← direita;
13 senão
14     L_HS ← esquerda; S ← S';
15 fim-se
16 retorne
fim AHS
```

Figura 4.1: Pseudo-código do algoritmo AHS

a heurística FFD por esta aproximação dual na busca binária. O algoritmo resultante é uma $\epsilon$-aproximação para o problema $P\|C_{\max}$ com $\epsilon = 1/5 + 2^{-k}$ e complexidade $O(n(k + \log n))$, onde $k$ é o número de iterações da busca. O pseudo-código da Figura 4.1 descreve este algoritmo, daqui por diante referenciado por AHS (Algoritmo de Hochbaum e Shmoys). Como entrada, considera-se o conjunto de tarefas/objetos $N$ e seus tempos/pesos $w_i$; os limites inferior $L$ e superior $U$ para o *makespan*; e o número de processadores $m$. No final da busca binária, além da solução aproximada $S$ para o problema $P\|C_{\max}$, tem-se também o limite inferior $L_{HS}$ para o *makespan*.

### 4.2.2 Limites e Estrutura da Solução

Dell'Amico e Martello [16] introduzem novos limites inferiores, analisam a estrutura da solução ótima e determinam condições suficientes para se estabelecer limites inferiores e superiores no número de tarefas por processador. Descreve-se nesta seção os limites inferiores apresentados neste trabalho, que serão utilizados pelo algoritmo proposto no próximo capítulo.

O limite $L_0$ (em qualquer escalonamento algum processador deve atingir a média da carga dos processadores), apresentado na seção anterior, possui desempenho de pior caso $R(L_0) = 0$, que é muito ruim. Isto é possível de se verificar considerando-se a série de instâncias com $n = m+1$, $w_1 = m$,

$w_2 = w_3 = \cdots = w_n = 1$, onde $L_0 = 2$ e $z_{P\|C_{\max}} = m$. Logo a razão $L_0/z_{P\|C_{\max}}$ é arbitrariamente próxima de 0 para $m$ muito grande. O limite melhorado $L_1$ (qualquer escalonamento deve processar cada tarefa) possui performance de pior caso $R(L_1) = 1/2$ (ver prova em [16]). A complexidade para calular $L_0$ e $L_1$ é $O(n)$ , uma vez que não é necessário ordenar as tarefas segundo (4-1).

Considerando-se a relaxação de $P\|C_{\max}$ obtida retirando-se as $(n - m - 1)$ menores tarefas $t_{m+2}, \ldots, t_n$, tem-se que o valor da solução ótima do problema não pode ser menor do que $w_m + w_{m+1}$, resultando no limite inferior melhorado

$$L_2 = \max\left\{L_1,\ w_m + w_{m+1}\right\},$$

o que melhora a razão do desempenho de pior caso para $R(L_2) = 2/3$. A complexidade para calular $L_2$ é $O(n)$ , uma vez que não é necessário ordenar as tarefas segundo (4-1).

O limite inferior descrito a seguir usa o mesmo princípio utilizado para o cálculo do limite $L_{HS}$, detalhado na Seção 4.2.1. Para qualquer algoritmo para BP, caracterizado pelo conjunto de objetos $T$ de tamanho $w_i$ $(i = 1, \ldots, n)$ e capacidade da caixa $L$, se o valor da solução ótima excede $m$, então $L + 1$ é um limite inferior válido para $P\|C_{\max}$.

**Teorema 4.1 [16]**. *Dados uma instância do $P\|C_{\max}$, e dois valores $L$ e $\overline{w} \leq L/2$, sejam*

$$\begin{aligned} I_1 &= \{i \in T : w_i > L - \overline{w}\}; \\ I_2 &= \{i \in T : L - \overline{w} \geq w_i > L/2\}; \\ I_3 &= \{i \in T : L/2 \geq w_i \geq \overline{w}\}; \end{aligned}$$

$$B_\alpha(L, \overline{w}) = |I_1| + |I_2| + \max\left\{0, \left\lceil \frac{\sum_{i \in I_3} w_i - (|I_2| \cdot L - \sum_{i \in I_2} w_i)}{L} \right\rceil\right\} \ e$$

$$B_\beta(L, \overline{w}) = |I_1| + |I_2| + \max\left\{0, \left\lceil \frac{|I_3| - \sum_{i \in I_2} \lfloor \frac{L - w_i}{\overline{w}} \rfloor}{\lfloor \frac{L}{\overline{w}} \rfloor} \right\rceil\right\}.$$

*Se $B_\alpha(L, \overline{w}) > m$ ou $B_\beta(L, \overline{w}) > m$, então $L + 1$ é um limite inferior válido para esta instância de $P\|C_{\max}$.*

Considera-se a instância de BP definida pelos objetos de peso $w_i$ $(i = 1, \ldots, n)$ e capacidade da caixa $L$. Os objetos em $I_1$ e $I_2$ não podem ser combinados com nenhum outro objeto nestes conjuntos. Desta forma, o número de objetos nos dois conjuntos é um limite inferior no número de caixas. Objetos em $I_3$ só podem ser combinados com objetos em $I_2$. Para

$B_\alpha(L, \overline{w})$, caso a soma dos pesos dos objetos em $I_3$ seja maior do que a soma da capacidade residual das caixas ocupadas pelos objetos de $I_2$, o limite pode ser aumentado pelo termo mais à direita, que aplica a mesma regra de $L_0$ (ver Seção 2.2.4.2). Para $B_\beta(L, \overline{w})$ relaxa-se a instância assumindo-se que todos os objetos de $I_3$ possuem tamanho igual a $\overline{w}$, assim $\lfloor \frac{L-w_i}{\overline{w}} \rfloor$ é a maior quantidade de objetos de $I_3$ que pode ser colocada na caixa com o objeto de tamanho $w_i (i \in I_2)$, logo $|I_3| - \sum_{i \in I_2} \lfloor \frac{L-w_i}{\overline{w}} \rfloor$ é o menor número de objetos de $I_3$ que devem ser colocados em caixas extra. Uma vez que cada nova caixa comporta no máximo $\lfloor L/\overline{w} \rfloor$ objetos de $I_3$, $B_\alpha(L, \overline{w})$ é um limite inferior no número de caixas válido para a instância de BP.

**Corolário 4.1 [16].** *Um limite inferior válido para* $P||C_{\max}$ *é*

$$L_3 = \max\{\, L+1 : \exists\ \overline{w} \leq \frac{L}{2} \ \textit{tal que}\ B_\alpha(L, \overline{w}) > m \ \textit{ou}\ B_\beta(L, \overline{o}) > m\}. \quad (4\text{-}2)$$

Para ilustrar o cálculo dos limites inferiores descritos acima, considera-se um pequeno exemplo com $n = 10$ tarefas, $m = 4$ processadores e tempos de processamento $w = (99, 76, 76, 75, 25, 13, 13, 13, 1, 1)$, onde $L_0 = 98, L_1 = 99, L_2 = 100$. Aplicando-se o Teorema 4.1 com $L = 100$ e $\overline{w} = 13$, tem-se que $I_1 = \{1\}, I_2 = \{2, 3, 4\}, I_3 = \{5, 6, 7, 8\}$. Logo

$$B_\alpha(100, 13) = \quad 1 + 3 + \max\left\{0, \left\lceil \frac{64-(300-227)}{100} \right\rceil \right\} = 4,$$
$$B_\beta(100L, 13) = \quad 1 + 3 + \max\left\{0, \left\lceil \frac{4-\left(\lfloor \frac{24}{13} \rfloor + \lfloor \frac{24}{13} \rfloor + \lfloor \frac{25}{13} \rfloor\right)}{\lfloor \frac{100}{13} \rfloor} \right\rceil \right\} = 5 > m.$$

Logo, 101 é um limite válido para esta instância.

Tanto $B_\alpha(L, \overline{w})$ como $B_\beta(L, \overline{w})$ são calculados para instâncias relaxadas obtidas eliminando-se as tarefas com tempo de processamento $w_i < \overline{w}$. Dada uma instância relaxada obtida eliminando-se as tarefas $t_{m+2}, \ldots, t_n$, então o valor de $L_2$ é o valor ótimo. Esta observação prova a proposição a seguir, útil para o cálculo eficiente de $L_3$.

**Proposição 4.2 [16].** *Uma vez que* $L_2$ *tenha sido calculado, valores melhores para o limite inferior podem ser obtidos através do Teorema 4.1 somente para valores de* $L$ *e* $\overline{w}$ *tais que* $L \geq L_2$ *e* $\overline{w} \leq w_{m+2}$.

Como provado pelos autores, o limite inferior $L_3$ pode ser calculado em tempo $O(n^2 \log U)$, onde $U$ é um limite superior para $C_{\max}$.

A seguir, descreve-se como calcular os limites inferior e superior no número de tarefas por processador considerando um valor para uma solução

viável, e de que forma estes limites podem estabelecer um novo limite inferior para $C_{\max}$.

A partir do valor $U$ de uma solução viável para uma instância de $P||C_{\max}$, é possível definir o seguinte limite superior no número de tarefas por processador:

$$\Theta = \max\left\{q\colon \sum_{i=n-q+1}^{n} w_i < U\right\}.$$

Este limite implica em que nenhuma solução com valor inferior a $U$ pode ter mais do que $\Theta$ tarefas por processador. Além disso, o caso especial onde $\Theta = 2$ ($n \leq 2m$) implica na solução ótima com a seguinte configuração: associar a tarefa $t_i$ ao processador $w_i$ para $i = 1, \ldots, m$ e a tarefa $t_{m+k}$ ao processador $P_{m-k+1}$ para $k = 1, \ldots, n-m$. Seja o número médio de tarefas por processador $\mu = \lfloor n/m \rfloor$. Então um limite inferior $\vartheta$ no número de tarefas por processador tem que ser menor ou igual a $\mu$. A seguir duas condições suficientes para determinar valores de $\sigma$ para os quais $\vartheta \geq \sigma$.

**Proposição 4.3 [16]**. *Dada uma instância de $P||C_{\max}$ e um inteiro $\sigma \leq \mu$, seja $L(\sigma)$ um limite inferior no valor da solução da instância relaxada com $m-1$ processadores e tarefas $t_\sigma, \ldots, t_n$. Se $L(\sigma) \geq U$, então qualquer solução com valor menor do que $U$ deve possuir pelo menos $\sigma$ tarefas em cada processador.*

*Prova.* Para cada valor possível de $\sigma$ considera-se uma solução onde um processador, por exemplo, $P_1$, possui $\sigma-1$ ou menos tarefas. Sobram para os demais processadores $P_2, \ldots, P_m$ pelo menos $n-(\sigma-1)$ tarefas. O *makespan* correspondente é pelo menos igual a $L(\sigma)$, onde $L$ é um limite inferior para a instância com $m-1$ processadores e as tarefas $t_\sigma, \ldots, t_n$. A solução final não pode ter valor menor do que $U$. □

**Proposição 4.4 [16]**. *Dados uma instância de $P||C_{\max}$, um limite $L$ no valor da sua solução e um inteiro $\sigma \leq \mu$, se $\sum_{i=1}^{\sigma} w_i \leq L$ então existe uma solução ótima com pelo menos $\sigma$ tarefas por processador.*

As Proposições 4.3 e 4.4 levam ao seguinte teorema:

**Teorema 4.5 [16]**. *Para qualquer instância de $P||C_{\max}$, dados uma solução viável de valor $U$ e um limite inferior $L$, então o limite inferior $\vartheta$ no número de tarefas por processador é dado por*

$$\vartheta = \max\left\{\sigma\colon L(\sigma) \geq U \quad ou \quad \sum_{i=1}^{\sigma} w_i \leq L\right\}, \tag{4-3}$$

*onde $L(\sigma)$ é definido segundo a Proposição 4.3.*

Os tempos para calcular $\vartheta$ e $\Theta$ dependem da forma como $U$, $L$ e $L(\sigma)$ são calculados. Se as tarefas estão ordenadas de acordo com (4-1); $U$ é determinado utilizando o algoritmo LPT (ver Seção 4.2); $L$ e $L(\sigma)$ são calculados usando $L_2$ (dado $L(\sigma)$, $L(\sigma+1)$ pode ser calculado em tempo constante), então $\vartheta$ e $\Theta$ podem ser determinados em tempo $O(n \log n)$.

O caso especial $\Theta = \vartheta + 1$ permite determinar o número $m_\vartheta$ de processadores com exatamente $\vartheta$ tarefas e o número $m_\Theta$ de processadores com exatamente $\Theta$ tarefas, resolvendo-se o sistema $m = m_\vartheta + m_\Theta$ e $n = \vartheta m_\vartheta + \Theta m_\Theta$:

$$m_\vartheta = (\vartheta + 1)m - n \tag{4-4}$$

$$m_\Theta = n - \vartheta m \tag{4-5}$$

Sabe-se então que a solução ótima é formada pela união das soluções de dois problemas menores, ainda que também difíceis. Entretanto, estas considerações permitem introduzir um novo limite para $P||C_{\max}$.

Dado o número médio de tarefas por processador $n/m$, pelo menos um processador deverá possuir $v = \lceil n/m \rceil$ ou mais tarefas. Assim, o somatório $L_\nu = \sum_{i=n-\nu+1}^{n} w_i$ das $v$ menores tarefas é um limite válido para a instância caracterizada pelo conjunto $N$ de tarefas e $m$ processadores.

Quando o caso especial $\Theta = \vartheta + 1$ acontece, dependendo dos valores dos limites inferior e superior correntes (uma vez que $\Theta$ é função de $U$ e $\vartheta$ é função de $L$ e $U$), tem-se

$$L_\vartheta(L,U) = \max\left\{\left\lceil \sum_{i=n-\vartheta m_\vartheta+1}^{n} w_i/m_\vartheta \right\rceil, \left\lceil \sum_{i=n-\Theta m_\Theta+1}^{n} w_i/m_\Theta \right\rceil\right\}. \tag{4-6}$$

Quando $\Theta = 2$, a solução para $L_\vartheta(L,U)$ é a solução ótima descrita anteriormente. Quando $\Theta > \vartheta + 1$, deve-se considerar instâncias relaxadas obtidas eliminando-se as últimas $\overline{n}$ tarefas ($\overline{n} = 1, 2, \ldots$), determinando-se $\overline{n}$ de tal forma que o caso especial ocorra e calculando-se então $L_\vartheta(L,U)$ para a instância relaxada. Seja $\overline{L}$ o valor de $L_\vartheta(L,U)$ para a instância (original ou relaxada) em que o caso especial $\Theta = \vartheta + 1$ acontece. Então

$$L_\vartheta = \max\{\overline{L}, L_\nu\} \tag{4-7}$$

é um limite válido para $C_{\max}$. O tempo para calcular $L_\vartheta$ é $O(n + n \log n)$, considerando-se a ordenação inicial.

Para ilustrar o cálculo do limite $L_\vartheta$, considera-se a instância definida por $n = 50, m = 5$ e tempos de processamento $w_i (i = 1, \ldots, n)$. A Tabela 4.1 mostra os seguintes valores: na primeira coluna tem-se a identificação da tarefa $i = 1, \ldots, n$; a seguir o tempo de processamento correspondente $w_i$, em ordem não-crescente; a terceira coluna mostra os tempos de processamento acumulados das $i$ tarefas $\bar{w}_i = w_1 + w_2 + \ldots + w_i$ para $i = 1, \ldots, n$; na última coluna tem-se o limite superior $U_i$ obtido aplicando-se a heurística LPT à instância relaxada definida pelas $i$ tarefas, $i = 1, \ldots, n$.

| $i$ | $w_i$ | $\bar{w}_i$ | $U_i$ | $i$ | $p_i$ | $\bar{w}_i$ | $U_i$ | $i$ | $p_i$ | $\bar{w}_i$ | $U_i$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 100 | 100 | 100 | 18 | 96 | 1766 | 392 | 35 | 93 | 3385 | 678 |
| 2 | 100 | 200 | 100 | 19 | 96 | 1862 | 392 | 36 | 93 | 3478 | 769 |
| 3 | 99 | 299 | 100 | 20 | 96 | 1958 | 392 | 37 | 93 | 3571 | 770 |
| 4 | 99 | 398 | 100 | 21 | 96 | 2054 | 487 | 38 | 92 | 3663 | 770 |
| 5 | 99 | 497 | 100 | 22 | 96 | 2150 | 487 | 39 | 92 | 3755 | 770 |
| 6 | 99 | 596 | 198 | 23 | 96 | 2246 | 488 | 40 | 92 | 3847 | 770 |
| 7 | 99 | 695 | 198 | 24 | 96 | 2342 | 488 | 41 | 92 | 3939 | 861 |
| 8 | 98 | 793 | 198 | 25 | 96 | 2438 | 488 | 42 | 92 | 4031 | 861 |
| 9 | 98 | 891 | 198 | 26 | 96 | 2534 | 583 | 43 | 92 | 4123 | 861 |
| 10 | 98 | 989 | 198 | 27 | 95 | 2629 | 583 | 44 | 91 | 4214 | 861 |
| 11 | 98 | 1087 | 295 | 28 | 95 | 2724 | 583 | 45 | 91 | 4305 | 861 |
| 12 | 98 | 1185 | 296 | 29 | 95 | 2819 | 583 | 46 | 91 | 4396 | 952 |
| 13 | 98 | 1283 | 296 | 30 | 95 | 2914 | 583 | 47 | 91 | 4487 | 952 |
| 14 | 97 | 1380 | 296 | 31 | 95 | 3009 | 677 | 48 | 91 | 4578 | 952 |
| 15 | 97 | 1477 | 296 | 32 | 95 | 3104 | 678 | 49 | 90 | 4668 | 952 |
| 16 | 97 | 1574 | 392 | 33 | 94 | 3198 | 678 | 50 | 11 | 4679 | 952 |
| 17 | 96 | 1670 | 392 | 34 | 94 | 3292 | 678 | | | | |

Tabela 4.1: Dados da instância usada como exemplo para o cálculo do limite $L_\vartheta$

Calculando-se o limite trivial tem-se $L_0 = \lceil \bar{w}_n/m \rceil = \lceil 4679/5 \rceil = 936$. Os limite $L_1$ e $L_2$ não melhoram este resultado. A aplicação do algoritmo LPT à instância resulta no limite superior $U = 952$. O limite superior no número máximo de tarefas por processador $\Theta$ é função de $U$ e é definido pelo maior número de tarefas que podem ser alocadas em um processador sem atingir o limite superior $U$, sendo igual a 11. O número médio de tarefas por processador é $\mu = \lfloor n/m \rfloor = 10$. A seguir calcula-se o limite inferior $\vartheta \leq \mu$ no número de tarefas por processador em função de $L$ e $U$. A propriedade (4.3) é satisfeita para valores de $\sigma \leq 9$, como pode ser verificado, por exemplo, para $\sigma = 9$. Calculando-se o limite $L_0$ para a instância com $m - 1$ processadores e tarefas $t_9, t_{10}, \ldots, t_n$, tem-se $L_0 = \lceil \frac{\sum_{i=\sigma}^{n} w_i}{m-1} \rceil = \lceil \frac{\bar{w}_{50} - \bar{w}_8}{4} \rceil = \lceil \frac{4679-793}{4} \rceil = 972 \geq U$. A propriedade (4.4) é satisfeita para valores de $\sigma \leq 9$ como pode ser verificado, por exemplo, para $\sigma = 9$, na coluna $\bar{w}_9$ da tabela.

Então, $\vartheta = 9$ satisfaz o Teorema (4-3) e uma vez que $\Theta > \vartheta + 1$, relaxa-se a instância $I$ eliminando-se a última tarefa $t_{50}$ e obtendo-se a instância relaxada $I_{49}$. A seguir, repete-se os cálculos para $I_{49}$. O limite inferior $L_0$ é dado por $\lceil \frac{\bar{w}_{49}}{5} \rceil = 934$. O limite superior $U$ é dado por $U_{49} = 952$ e, em função de $U$, calcula-se $\Theta = 10$. O número médio de tarefas por processador é $\mu = \lfloor n/m \rfloor = \lfloor 49/5 \rfloor = 9$. Fazendo-se $\sigma = 9$ e calculando-se o limite $L_0(\sigma)$ para a instância com $m-1$ processadores e tarefas $t_9, t_{10}, \ldots, t_{49}$, tem-se $L_0(\sigma) = \lceil \frac{\sum_{i=\sigma}^{49} w_i}{m-1} \rceil = \lceil \frac{\bar{w}_{49} - \bar{w}_8}{4} \rceil = \lceil \frac{4668-793}{4} \rceil = 967 \geq U$. Este é o maior valor possível para $\sigma \leq \mu$ que satisfaz a Proposição (4.3). Como $\Theta = \vartheta + 1$, pode-se então calcular o número de processadores $m_\Theta$ com exatamente $\Theta$ tarefas e o número de processadores $m_\vartheta$ com exatamente $\vartheta$ tarefas resolvendo-se o sistema (4-4), resultando em $m_\Theta = 4$ e $m_\vartheta = 1$. A seguir, substituindo-se os valores na expressão (4-6) tem-se

$$\bar{L}_\vartheta(934, 952) = \max\left\{ \left\lceil \sum_{i=49-(9\cdot 1)+1=41}^{49} \frac{w_i}{1} \right\rceil, \left\lceil \sum_{i=49-10\cdot 4+1=10}^{49} \frac{w_i}{4} \right\rceil \right\} =$$

$$= \max\left\{ \left\lceil \frac{\bar{w}_{49} - \bar{w}_{40}}{1} \right\rceil, \left\lceil \frac{\bar{w}_{49} - \bar{w}_9}{4} \right\rceil \right\} = \max\{\lceil 821 \rceil, \lceil 3777/4 \rceil\} = 945$$

Por fim, a expressão (4-7) é calculada e $L_\vartheta = \max\{945, 832\} = 945$.

A heurística proposta para $P||C_{\max}$, apresentada no próximo capítulo, usa os limites $L_2$, $L_3$, $L_{HS}$ e $L_\vartheta$ e os algoritmos de aproximação LPT e AHS apresentados nesse capítulo.