5
Heurística Híbrida de Melhoria Modificada para $P||C_{\max}$

Considera-se o procedimento `C+R+M_BP` definido na Seção 3.3. Este fornece uma solução S para a instância definida pelo conjunto de n objetos $N = \{1, \ldots, n\}$, o vetor de pesos associado w_i $(i = 1, \ldots, n)$ e a capacidade da caixa C com um número determinado m de caixas. Não necessariamente S é viável para BP, isto é, permitem-se soluções que violem a restrição de capacidade das caixas. Considerando-se o problema de escalonamento, dados o conjunto de n tarefas $T = \{t_1, \ldots, t_n\}$, o vetor de tempos associado w_i $(i = 1, \ldots, n)$, um valor $C_{\max}$ para o maior tempo de processamento entre todos os processadores e um número m de processadores, então a solução encontrada por `C+R+M_BP` é viável para $P||C_{\max}$. Porém, não há garantia de que o *makespan* encontrado pela solução S seja igual a $C_{\max}$, caso em que a solução não é viável para BP. Assim, se a solução S não for viável para BP, o valor de $C_{\max}$ poderá ser incrementado de uma unidade e o algoritmo `C+R+M_BP` aplicado novamente. Este procedimento é aplicado sucessivamente, até que seja encontrado um valor de $C_{\max}$ para o qual a solução obtida por `C+R+M_BP` seja viável para BP. Diversos métodos de busca podem ser considerados para reduzir o número de chamadas ao procedimento `C+R+M_BP`. A seguir, descreve-se as três fases do procedimento híbrido de melhoria modificado (`HI_PCmax`) proposto para o problema $P||C_{\max}$.

5.1
As Três Fases: `Construção`, `Redistribuição` e `Melhoria`

Na fase de `Construção` de uma solução inicial do procedimento `HI_BP` (Seção 3.1), considerou-se a criação de soluções viáveis para o problema dual do *bin packing* (DBP) com exatamente m caixas (processadores), com m igual ao melhor limite inferior conhecido para BP. Foram criadas quatro heurísticas construtivas (`DBFD`, `DWFD`, `DB3F`, `DWSFD`) para o problema de $P||C_{\max}$, com inspiração em heurísticas conhecidas para BP. Por exemplo,

as duas primeiras são baseadas em regras semelhantes às utilizadas pelas heurísticas correspondentes `BFD` e `WFD` para BP. Considerou-se, inicialmente, todas estas heurísticas como opção de heurística construtiva. Experimentos preliminares verificaram que `DBFD` e `DB3FD` foram as que menos contribuíram para melhorar a qualidade da solução e, portanto, decidiu-se não utilizá-las nessa fase. Por outro lado, o algoritmo `AHS` (detalhado na Seção 4.2.1) mostrou ser uma boa opção de heurística construtiva para essa fase. A seguir, descreve-se cada opção do conjunto final H de heurísticas construtivas selecionadas.

- **Longest Processing Time** (`LPT`): como já descrito na Seção 4.2, considerando-se as tarefas em ordem não-crescente de tempo de processamento, cada tarefa é alocada ao primeiro processador livre (aquele com menor carga). Esta é, provavelmente, a heurística construtiva mais popular para este problema e produz soluções equivalentes àquelas produzidas pela heurística `DWFD` na Seção 3.1.
- **Dual Worst Sum-Fit Decreasing** (`DWSFD`): esta heurística foi usada para instâncias de BP e é detalhada na Seção 3.1. Escalona-se a tarefa mais longa para um determinado processador e considera-se como folga o tempo de processamento restante em relação a um dado limite de tempo. Em seguida, tenta-se encontrar um subconjunto de tarefas cuja soma dos tempos de processamento mais se aproxime do valor da folga, colocando-as então no processador. Para encontrar este subconjunto, considerou-se o procedimento `MTSS(3)` [65] para o problema de soma de conjuntos. A relação com o problema de soma de conjuntos também foi explorada, no contexto de $P||C_{\max}$, por Dell'Amico e Martello [16] e por Lee e Massey [62].
- `AHS`: este algoritmo constrói escalonamento com *makespan* não maior do que $6/5+2^{-k}$ vezes a solução ótima e número de processadores não maior do que m, onde k é o número de iterações da busca. O número de processadores utilizados pode ser menor do que m, gerando uma estrutura de solução bem diversa se comparada às duas heurísticas anteriores.

Os algoritmos desta fase de `Construção` também serão utilizados na etapa de pré-processamento, anterior à chamada ao procedimento `C+R+M_BP`. O objetivo é determinar o melhor valor para uma solução viável inicial (limite superior) e eventualmente encontrar uma solução ótima.

As duas outras fases, `Redistribuição` e `Melhoria`, são idênticas às descritas no terceiro capítulo. Os parâmetros da heurística (especificados

na Seção 3.4.1) sofreram um pequeno ajuste. Experimentos computacionais preliminares estabeleceram um número inferior de iterações na busca tabu e o conjunto de heurísticas construtivas incorporou uma heurística nova e eliminou duas. A seguir a relação dos valores usados pelos parâmetros da implementação da versão modificada `HI_PCmax`:

- `MaxTentativas` $= 3$
- `MaxRedistribuição` $= 100$
- `TabuTenure` $\in [0.8\sqrt{n}, 1.2\sqrt{n}]$
- `MaxDelta` $= 0.2 \min_{i=1,\ldots,n} \{w_i\}$
- `MaxTabu` $= 1000$
- `Semente` $= 1$
- $H = \{$`DWSFD`, `AHS`, `LPT`$\}$

A seguir, descreve-se o procedimento híbrido de melhoria modificado (`HI_PCmax`) proposto para o problema $P||C_{\max}$, ilustrado na Figura 5.1. A etapa inicial (ou pré-processamento) compreende as linhas 1-14. Esta etapa calcula os limites inferiores ($L_2, L_3, L_\vartheta, L_{HS}$) descritos na Seção 4.2.2, aplica as heurísticas construtivas (`LPT`, `DWSFD`, `AHS`) descritas anteriormente, determina a melhor solução inicial S e os melhores limites inferior L e superior U e, eventualmente, encontra uma solução comprovadamente ótima com *makespan* igual a L, finalizando-se o procedimento. Quando este não é o caso, deseja-se encontrar o menor valor de $C_{\max}$ no intervalo $[L, U]$ para o qual o procedimento `C+R+M_BP` retorna um empacotamento viável para BP. Como mencionado anteriormente, este processo pode envolver um número elevado de iterações. Inicialmente, considerou-se dois métodos elementares de busca. O método de limite inferior inicializa a variável $C_{\max}$ com o valor de L e em seguida incrementa $C_{\max}$ em uma unidade até o procedimento `C+R+M_BP` encontrar solução viável para BP. O método de busca binária usa as variáveis *esquerda* e *direita* para denotar, respectivamente, o menor valor para o qual o procedimento `C+R+M_BP` pode encontrar uma solução viável e o menor valor de uma solução conhecida. A variável *esquerda* é inicializada com o valor do melhor limite inferior disponível L e a *esquerda* com o valor do melhor limite superior disponível U. Em cada iteração, divide-se o intervalo $[esquerda, direita]$ ao meio fazendo a variável $C_{\max}$ assumir o valor $\lfloor (esquerda + direita)/2 \rfloor$. De acordo com o resultado do procedimento `C+R+M_BP` atualiza-se o valor da variável *esquerda* ou *direita*, diminuindo-se o intervalo de valores possíveis, até quando não houver

valor algum para ser testado. Como sugerido em [78], foram consideradas algumas variantes e combinações destes métodos, discutidas na Seção 5.2.3. Experimentos computacionais mostraram que a melhor estratégia combina a busca binária e o método de limite inferior da seguinte forma. Na primeira chamada ao procedimento `C+R+M_BP` (linha 15 do pseudo-código) utiliza-se o melhor limite conhecido para o valor da variável $C_{\max}$. Quando este passo não encontra o escalonamento ideal, as próximas chamadas ao procedimento `C+R+M_BP` servem para diminuir a diferença entre os limites inferior e superior e consequentemente o erro relativo da solução que não será comprovadamente ótima. Nas linhas 17 e 18 atualizam-se as variáveis *esquerda* e *direita* para iniciar a busca binária compreendida entre as linhas 19–24. No início de cada repetição do laço, tem-se uma solução viável S_{CRM} ou S com *makespan* igual ao valor da variável *direita* e sabe-se que para o valor de $C_{\max}$ igual ao valor da variável *esquerda* o procedimento `C+R+M_BP` não produziu uma solução viável para BP. Quando ($esquerda = direita - 1$) esta relação permanece e a variável *direita* guarda o valor do melhor *makespan* encontrado, que é atualizado na linha 25. O comando da linha 26 verifica se a melhor solução S foi encontrada na etapa de pré-processamento ou se foi gerada durante a busca binária, atualizando a variável S com a melhor solução. A melhor solução é retornada no último passo do algoritmo na linha 27.

5.2 Experimentos Computacionais

Três estudos experimentais foram realizados. O objetivo do primeiro estudo é avaliar o impacto de diferentes métodos de busca. O próximo estudo analisa a importância de cada uma das três fases do algoritmo `HI_PCmax`: fase de `Construção` (`C`), fase de `Redistribuição` (`R`) e fase de `Melhoria` (`M`). O objetivo do último experimento é avaliar a eficiência do algoritmo proposto em relação à qualidade da solução e tempo de processamento. Comparou-se o algoritmo `HI_PCmax` com quatro métodos da literatura: a conhecida heurística construtiva `LPT`, a heurística de melhoria `3-PHASE` proposta por França et al. [26], o método exato de *branch and bound* proposto por Dell'Amico e Martello [16] (identificado como `B&B`) e a melhor solução encontrada pelos algoritmos *multi-exchange* recentemente propostos por Frangioni et al. [27] (identificados como `ME`). Na literatura, a comparação com `LPT` costuma ser referência para a maioria dos experimentos computacionais. Estabelecendo-se um limite de *backtracks* para o método exato `B&B`,

Procedimento HI_PCmax
Entrada: $T = \{t_1, \ldots, t_n\}$, m,
w_i $(i = 1, \ldots, n)$ onde $w_h \geq w_{h+1}, h = 1, \ldots, n-1$, H,
MaxTentativas, MaxRedistribuição, MaxTabu,
MaxDelta, Semente.
Saída: $S = \{P_1, \ldots, P_m\}$.
1 Gere S_{LPT} utilizando heurística LPT e inicialize $U \leftarrow C_{\max}(S_{LPT})$;
2 $L \leftarrow L_2$;
3 **se** $L = U$ **então** $S \leftarrow S_{LPT}$ **e retorne**
4 Calcule L_3 por busca binária entre $[L, U]$ e faça $L \leftarrow L_3$;
5 **se** $L = U$ **então** $S \leftarrow S_{LPT}$ **e retorne**
6 Calcule L_ϑ em função de L e U e atualize $L \leftarrow \max\{L, L_\vartheta\}$;
7 **se** $L = U$ **então** $S \leftarrow S_{LPT}$ **e retorne**
8 $S_{AHS}, L_{HS} \leftarrow$ AHS(T, w, L, U, m);
9 $L \leftarrow \max\{L, L_{HS}\}$;
10 **se** $L = U$ **então** $S \leftarrow S_{LPT}$ **e retorne**
11 Gere solução S_{DWSFD} utilizando a heurística DWSFD;
12 **se** $C_{\max}(S_{DWSFD}) < U$ **então** $U \leftarrow C_{\max}(S_{DWSFD})$; $S \leftarrow S_{DWSFD}$;
13 **se** $C_{\max}(S_{AHS}) < U$ **então** $U \leftarrow C_{\max}(S_{AHS})$; $S \leftarrow S_{AHS}$;
14 **se** $L = U$ **então retorne**
15 $S_{CRM} \leftarrow$ C+R+M_BP$(T, L, m, w, H,$ MaxTentativas,
MaxRedistribuição, MaxTabu, MaxDelta, Semente$)$;
16 **se** Viável$_{BP}(S_{CRM})$ **então** $S \leftarrow S_{CRM}$ **e retorne**
17 *esquerda* $\leftarrow L$;
18 *direita* $\leftarrow U$;
19 **enquanto** *esquerda* $<$ *direita* $- 1$ **faça**
20 $C_{\max} \leftarrow \lfloor (esquerda + direita)/2 \rfloor$;
21 $S_{CRM} \leftarrow$ C+R+M_BP$(T, C_{\max}, m, w, H,$ MaxTentativas,
MaxRedistribuição, MaxTabu, MaxDelta, Semente$)$;
22 **se** Viável$_{BP}(S_{CRM})$ **então** *direita* $\leftarrow C_{\max}$;
23 **senão** *esquerda* $\leftarrow C_{\max}$;
24 **fim-enquanto**
25 $C_{\max} \leftarrow$ *direita*;
26 **se** $C_{\max} \neq U$ **então** $S \leftarrow S_{CRM}$;
27 **retorne**
fim HI_PCmax

Figura 5.1: Pseudo-código da heurística híbrida modificada HI_PCmax

este é capaz de resolver muitas instâncias otimamente em poucos segundos e assim proporcionar uma medida de qualidade, além da possibilidade de acesso a seu código. A disponibilidade de instâncias de *benchmark* propostas pelos autores da heurítica 3-PHASE e pelos autores dos algoritmos que usam o método *multi-exchange* também permite comparações com estes dois trabalhos.

5.2.1 Problemas Testes

Para os experimentos computacionais, considerou-se dois grupos de problemas testes. Cada grupo é composto de 10 problemas testes de 39 classes, totalizando 390 instâncias em cada grupo. As classes são caracterizadas pela combinação dos seguintes parâmetros: $m \in \{5, 10, 25\}$, $n \in \{50, 100, 500, 1000\}$ ($m = 5$ também foi combinado com $n = 10$), e tempos de processamento pertencentes aos intervalos $[1, 100], [1, 1000]$ e $[1, 10000]$.

O primeiro grupo, chamado `uniforme`, foi introduzido por França et al. [26, 70] e é disponibilizado em [42]. Os tempos de processamento foram selecionados de uma distribuição uniforme no intervalo dado.

O gerador do segundo grupo de problemas foi desenvolvido por Frangioni et al. [28] e coletado de [71], juntamente com *scripts* para produzir as instâncias. Segundo os autores, instâncias com tempos de processamento altamente não uniforme são difíceis de serem resolvidas pela heurística `LPT`. Para gerar instâncias com estas características o gerador proposto seleciona 98% dos tempos de processamento de uma distribuição uniforme no intervalo $[(b-a)\,0.9, b]$ e o restante dos tempos de processamento do intervalo $[a, (b-a)\,0.02]$, onde $[a, b]$ é um intervalo de tempos de processamento possíveis. Este grupo é chamado `não-uniforme`.

5.2.2 Resultados Computacionais

Os algoritmos `HI_PCmax` e `LPT` foram codificados na linguagem `C` e compilados com a versão 2.95.2 do compilador `gcc`, utilizando a opção de otimização -O3. Os experimentos foram executados em um Pentium IV com relógio de 1.7 GHz e 256 MB de memória RAM. O código do algoritmo `B&B` foi cedido pelos autores, compilado e executado neste mesmo ambiente, usando a mesma opção de otimização. Este retorna o melhor limite inferior, a melhor solução e o tempo de processamento em segundos. O número de *backtracks* é fixado em 4000 por três motivos: este é o valor sugerido pelos autores; testes preliminares com o valor 8000 para o número de *backtracks* permitiram verificar que o número de soluções ótimas não aumentou e a melhoria no erro relativo foi ínfima, e, além disso, Frangioni et al. [27] fizeram experimentos com 40000 *backtracks* e obtiveram resultados semelhantes, porém com tempos computacionais elevados.

5.2.3 Estudo do Impacto de Diferentes Métodos de Busca para a Heurística `HI_PCmax`

Investigou-se o método de limite inferior (LI), o método de busca binária (BB), o método que combina busca de Fibonacci com busca binária (FBB) e o método que combina limite inferior com busca binária (LIB) [78]. Estes métodos iniciam com um intervalo $[L, U]$ de possíveis valores para $C_{\max}$, onde L é o valor do melhor limite inferior disponível e U o valor do melhor limite superior disponível. Os métodos básicos LI e BB foram explicados anteriormente. Assim como em BB, o método de busca de Fibonacci usa as variáveis *esquerda* e *direita* para guardar os menores valores de uma possível solução ainda não testada e de uma solução já conhecida. A busca inicia fazendo $esquerda = L$, $direita = U$ e $C_{\max} = esquerda$. Em cada iteração da busca, a seqüencia de números de Fibonacci $1, 2, 3, 5, \ldots$ é usada para incrementar $C_{\max}$. Desta forma, $C_{\max}$ assume os valores $L, L+1, L+3, L+6, L+11, \ldots$ Enquanto $C_{\max}$ for menor do que *direita* e não se encontrou solução viável com o valor de $C_{\max}$ a busca continua, atualizando-se sempre o valor da variável *esquerda* com o maior valor de $C_{\max}$ já testado. Quando a busca pára porque encontrou-se solução viável com o valor de $C_{\max}$, atualiza-se também o valor da variável $direita = C_{\max}$. Neste ponto tem-se um intervalo reduzido $[esquerda, direita]$ de valores possíveis para $C_{\max}$ e pode-se usar os métodos de limite inferior ou busca binária para concluir. O método denotado por FBB consiste então da busca de Fibonacci até que uma solução viável seja encontrada, ou se $C_{\max} > direita$, quando então é utilizada a busca binária para o novo intervalo $[esquerda, direita]$. O método LIB faz a primeira chamada ao procedimento `C+R+M_BP` usando o melhor limite inferior para $C_{\max}$ e, em seguida, se necessário, aplica a busca binária.

Este estudo mede a qualidade da solução em relação ao tempo de processamento. Utilizou-se os dois grupos de problemas descritos anteriormente. A Tabela 5.1 mostra os resultados sumarizados. A primeira linha de cada coluna identifica o método de busca. Para cada grupo, para cada intervalo e para cada método, relata-se o tempo total em segundos para executar todas as instâncias, e os números máximo e médio de chamadas ao procedimento `C+R+M_BP` realizadas entre todas as instâncias. O número de soluções ótimas não é mostrado na tabela, pois todos os métodos encontram os mesmos valores. As células em destaque mostram os melhores resultados. Verifica-se que o método LIB é o melhor de todos, pois, considerando-se a medida de tempo, consegue os melhores resultados para quatro de seis classes. Nas ou-

método de busca		LI			BB			LIB			FBB		
grupo	w_i	tempo	máx.	méd.	tempo	máx.	méd.	tempo	máx.	méd.	tempo	máx.	méd.
`uniforme`	$[1, 100]$	0.19	1	1.00	0.18	1	1.00	0.21	1	1.00	0.19	1	1.00
	$[1, 1000]$	4.63	14	2.88	3.19	6	3.38	3.12	6	2.38	3.62	9	2.25
	$[1, 10000]$	56.30	190	1.71	19.22	10	6.50	19.37	10	6.23	24.74	18	5.10
`não-`	$[1, 100]$	21.84	8	1.73	27.40	5	3.56	18.05	4	1.47	19.93	6	1.61
`uniforme`	$[1, 1000]$	57.06	26	1.77	68.96	9	6.06	26.21	6	1.15	32.62	11	1.31
	$[1, 10000]$	582.51	254	9.18	524.02	12	8.98	94.14	10	2.02	104.89	19	1.68

Tabela 5.1: Estudo dos diferentes métodos de busca para `HI_PCmax`

tras duas classes, a diferença em relação ao melhor resultado é desprezível. Este resultado é coerente com o fato de que para a maioria das instâncias testadas o melhor limite conhecido coincide com o valor do *makespan* encontrado, pois os limites são fortes e a heurística é bem sucedida na maioria das vezes. Sendo assim, a primeira chamada ao procedimento `C+R+M_BP` deve considerar este limite inferior como o valor de $C_{\max}$. Isto é feito por todos os métodos, exceto o método básico de busca binária (BB). No entanto, para os casos em que isso não acontece, a distância entre os limites inferior e superior iniciais pode ser grande, principalmente para instâncias em que o intervalo de tempos de processamento possíveis é grande, como pode ser observado na tabela pelos resultados do método básico de limite inferior (LI). O método indicado para mais rapidamente reduzir o intervalo da busca é o método de busca binária. LIB é o método que combina estes dois métodos e foi utilizado na versão final de `HI_PCmax`.

5.2.4 Estudo das Diferentes Fases da Heurística `HI_PCmax`

Para todas as instâncias executou-se três versões do algoritmo, a primeira somente com a fase de `Construção` (`C`), a segunda com a fase de `Construção` e `Redistribuição` (`C+R`) e a última corresponde à versão completa do algoritmo `HI_PCmax` com as três fases (`C+R+M`). A Tabela 5.2 mostra os resultados obtidos. Cada linha da tabela mostra estatísticas para 130 instâncias de um dado grupo de problemas (primeira coluna) e de um dado intervalo de tempos de processamento possíveis (segunda coluna). A etapa inicial (`I`) agrupa as próximas três colunas: as colunas `LPT`, `DWSFD` e `AHS` mostram, respectivamente, o número de soluções identificadas como ótimas antes da aplicação do procedimento `C+R+M_BP`, por cada uma dessas heurísticas construtivas. As próximas seis colunas estão organizadas em três blocos, um para cada uma das versões: `C`, `C+R` e `C+R+M`. Em cada bloco, mostra-se quantas soluções ótimas adicionais foram obtidas em relação à versão anterior com menos uma fase (+) e o tempo em segundos de execução da respectiva versão do algoritmo. A última coluna mostra o número total de soluções ótimas obtidas. Para cada grupo a última linha totaliza os valores de cada coluna.

Considerando-se o grupo `uniforme`, 366 (93.9%) das 390 instâncias foram resolvidas de forma exata, 353 (96.4%) destas na etapa inicial. Para o intervalo $[1, 100]$, as três versões não melhoram o tempo obtido pela etapa inicial. Isto pois, apesar de todas as soluções encontradas na etapa

						versões						
		I				C		C+R		C+R+M		
grupo	w_i	LPT	DWSFD	AHS	tempo	+	tempo	+	tempo	+	tempo	ótimos
uniforme	$[1,100]$	72	57	1	0.04	-	0.04	-	0.05	-	0.21	130
	$[1,1000]$	23	99	-	1.90	1	1.90	2	2.02	1	3.12	126
	$[1,10000]$	15	86	-	3.36	2	3.58	7	5.69	-	19.37	110
totais		110	242	1		3		9		1		366
não	$[1,100]$	21	50	-	0.83	-	1.50	14	24.95	35	18.05	120
uniforme	$[1,1000]$	20	45	-	1.13	-	2.34	6	51.37	57	26.21	128
	$[1,10000]$	20	48	-	1.92	-	5.06	2	109.15	51	94.14	121
totais		61	143	0		0		22		143		369

Tabela 5.2: Estudo da importância das três fases de HI_PCmax

inicial serem comprovadamente ótimas, em um único caso a melhor solução encontrada é diferente do melhor limite calculado pela heurística e então se perdeu tempo nas fases seguintes tentando encontrar uma solução com um valor impossível. Considerando-se as instâncias cujos pesos pertencem ao intervalo $[1, 1000]$, a versão `C` encontrou uma solução ótima a mais em relação à etapa inicial, a versão `C+R` encontrou duas soluções ótimas a mais em relação à versão anterior e a versão `C+R+M` encontrou mais uma solução ótima. Para o intervalo $[1, 10000]$, a versão `C` conseguiu melhorar apenas duas instâncias; a versão `C+R` contribuiu para melhorar sete instâncias, com um pequeno acréscimo de tempo. A execução também da fase de `Melhoria` (versão `C+R+M`) não alterou o número de soluções ótimas, com o custo de elevar o tempo de processamento.

No caso do grupo `não-uniforme`, a etapa incial resolveu 204 (52.3%) das 390 instâncias e sobraram 186 instâncias para as três fases da heurística. A versão `C` não resolveu nenhuma instância a mais em relação à etapa inicial. A versão `C+R` contribuiu para melhorar a qualidade da solução de mais 22 instâncias, com grande aumento no tempo computacional. A execução da última fase, pela versão `C+R+M`, foi fundamental para resolver a grande maioria das instâncias formada por 143 (76.9%) das 186 instâncias, e não por acaso o tempo computacional é menor do que o obtido pela versão `C+R`.

Para finalizar a análise deste estudo, pode-se dizer que a heurística construtiva `DWSFD` é uma ótima heurística para estas classes de problemas. O número total de instâncias resolvidas é 735. `LPT`, a primeira heurística construtiva da etapa inicial, resolveu 171 delas. Das 564 restantes, `DWSFD` encontrou a solução ótima de 385 (68.3%). A importância da heurística `AHS` não está na etapa inicial, mas sim como solução inicial gerada para as três fases. Das 366 instâncias resolvidas do grupo `uniforme`, todas, com exceção de uma, foram resolvidas na etapa inicial ou em uma das duas primeiras fases (`Construção` e `Redistribuição`). Neste caso, a execução da última fase (`Melhoria`) não contribuiu com melhora na qualidade das soluções e aumentou muito o tempo computacional. Em contrapartida, quando a fase de `Melhoria` é essencial para resolver as instâncias, sua execução ajuda a melhorar o tempo computacional, pois o problema é resolvido, na maioria das vezes, na primeira tentativa (isto é, com a primeira heurística construtiva). Isto pode ser verificado com as instâncias do grupo `não-uniforme`. Pode-se concluir que para instâncias "fáceis" a versão `C+R`, que executa as fases de `Construção` e `Redistribuição`, é por si só uma boa heurística. De forma geral, a heurística `HI_PCmax` é eficiente e robusta. Esta mesma tendência foi observada para estudo similar realizado com instâncias

do BP no terceiro capítulo.

A Tabela 5.3 resume estatísticas de uma execução da heurística `HI_PCmax`, equivalente à versão `C+R+M` da tabela anterior. Para cada grupo e para cada intervalo de pesos, mostra o número de ótimos, o número de execuções em que a solução ótima foi encontrada na etapa inicial (`I`), na fase de `Construção` (P1), na fase de `Redistribuição` (P2), ou na fase de `Melhoria` (P3); o número de execuções em que a solução ótima originou de uma solução inicial construída pelas heurísticas `DWSFD` (H1), `AHS` (H2) ou `LPT` (H3); e o valor máximo encontrado para o número de iterações realizadas pelo procedimento `Melhoria` (Figura 3.11), pelo procedimento `Redistribuição` (Figura 3.9), pelo procedimento `Equilíbrio` (Figura 3.6) e pelo procedimento `Desequilíbrio` (Figura 3.7). A importância de cada fase, foi discutida no parágrafo anterior. Observa-se que, o número de vezes que a solução construída por H1 leva à melhor solução final (125 vezes) é maior do que o número de vezes obtido por H2 (51 vezes) e este é maior do que o obtido por H3 (2 vezes). Como observado anteriormente para BP (Tabela 3.5), isto é devido à ordem da aplicação das heurísticas. Também constata-se a necessidade e a importância do uso de diferentes estratégias na construção de soluções iniciais. O valor máximo do número de iterações realizadas pelo procedimento `Melhoria` é grande para todas as classes e isso é coerente com o fato desta fase ser a mais demorada. Considerando-se todas as instâncias testadas, o procedimento `Redistribuição` realiza no máximo 10 iterações, e o número de iterações necessárias para avaliar todas as possibilidades de redistribuição executadas pelos procedimentos `Equilíbrio` e `Desequilíbrio`, converge com no máximo 594 iterações.

5.2.5 Comparação com Outros Métodos da Literatura

Este estudo foi dividido em duas partes. A primeira trata das instâncias do grupo `uniforme` e considera os métodos `LPT`, `B&B` e `3-PHASE`. A segunda trata das instâncias do grupo `não-uniforme` e considera os métodos `LPT`, `B&B` e `ME`. As Tabelas 5.4, 5.5 e 5.6, relativas à primeira parte, mostram resultados para os intervalos de tempos de processamento $[1, 100]$, $[1, 1000]$ e $[1, 10000]$. As Tabelas 5.7, 5.8 e 5.9 mostram os resultados para os mesmos intervalos para o grupo `não-uniforme`.

Em todas estas tabelas, a primeira linha identifica a heurística e as demais linhas mostram resultados médios de dez instâncias para cada uma das 13 classes determinadas pela combinação do número de processadores

				fase			heurística			no. máximo iterações			
grupo	w_i	ótimos	I	P1	P2	P3	H1	H2	H3	M	R	E	D
uniforme	$[1, 100]$	130	130	0	0	0	0	0	0	2069	3	36	268
	$[1, 1000]$	126	122	1	1	2	4	0	0	1631	3	301	390
	$[1, 10000]$	110	101	2	7	0	6	2	1	3182	5	494	594
não	$[1, 100]$	120	71	0	0	49	37	11	1	1708	5	90	15
uniforme	$[1, 1000]$	128	65	0	4	59	44	19	0	1709	5	288	10
	$[1, 10000]$	121	68	0	1	52	34	19	0	3999	10	227	23
totais		735	557	3	13	162	125	51	2				

Tabela 5.3: Estatísticas das fases, heurísticas e número de iterações de uma execução de HI_PCmax

(m) e número de tarefas (n) para o dado intervalo de tempos de processamento possíveis. Para os algoritmos `LPT`, `B&B` e `HI_PCmax`, executados no mesmo ambiente, apresenta-se o valor médio dos erros relativos $(x - y)/y$, onde x é o valor da solução obtida pela heurística correspondente e y é o valor do melhor limite inferior conhecido (L_3, L_{HS}, L_ϑ ou o limite obtido pelo método exato `B&B`), o número de soluções ótimas e os tempos médios de execução em segundos. Esta última coluna foi omitida para a heurística `LPT` pois seus tempos de processamento são desprezíveis em relação às demais.

Na primeira parte deste estudo, a informação do erro relativo da heurística `3-PHASE` foi obtida de [26], calculado em relação ao valor da solução ótima, segundo os autores. Os resultados relacionados à coluna (`ME`), na segunda parte, correspondem aos valores do melhor resultado de três algoritmos *multi-exchange* (colunas 1-SPT, 1-BPT e K-SPT) extraídos das Tabelas 3, 4 e 5 do trabalho relatado em [27, págs. 19, 20 e 21]. A coluna (H) identifica qual das três heurísticas obteve a melhor solução. Estes algoritmos foram executados em um PC com processador Pentium II 400 MHz e 256 Mbytes de memória RAM.

As tabelas do documento [1] mostram os resultados detalhados dos experimentos.

Muitos autores usaram instâncias uniformes para testar seus algoritmos. A maioria das instâncias uniformes são resolvidas em poucos segundos. Os resultados das Tabelas 5.4, 5.5 e 5.6 revelam que `HI_PCmax` possui o melhor desempenho em relação à qualidade da solução para todas as classes de problemas. Apenas quatro instâncias do intervalo $[1, 1000]$ e 20 do intervalo $[1, 10000]$ não foram resolvidas otimamente pelo algoritmo `HI_PCmax`. Entre os três outros métodos, `B&B` é o melhor. Comparando-se com o algoritmo `B&B` este encontrou praticamente a mesma quantidade de soluções ótimas (menos três instâncias do intervalo $[1, 10000]$), porém o erro relativo nunca é inferior aquele obtido pelo `HI_PCmax`. Para um grupo similar de problemas testes, Frangioni et al. [27] observam que a medida que a amplitude dos tempos de processamento cresce as instâncias ficam mais difíceis, o que também pode ser constatado nos resultados aqui apresentados. Além disso, eles identificam que instâncias onde a razão n/m pertence ao intervalo $[2, 10]$ (em destaque na tabela) são especialmente difíceis. Entretanto, o algoritmo `HI_PCmax` proposto resolveu otimamente todas as instâncias com a relação $n/m = 10$, encontrando dificuldade apenas para o caso $2 < n/m < 10$.

As instâncias do grupo `não-uniforme` são mais difíceis do que as do grupo `uniforme`, como pode ser observado nas Tabelas 5.7, 5.8 e 5.9. Mais uma vez, Frangioni et al. [27] consideram como as instâncias mais difíceis

aquelas em que $2 < n/m \leq 10$ (em destaque na tabela) e observaram que, para valores grandes de n, instâncias em que $n/m \leq 40$ (nas tabelas, as duas últimas linhas) também são difíceis. De fato, constata-se esta dificuldade para os métodos `LPT`, `B&B` e `ME`, mas não para `HI_PCmax`. Observando-se as duas últimas linhas das três tabelas, percebe-se que `HI_PCmax` obteve a solução exata de todas estas instâncias. A comparação de `B&B` com `ME` não dá para estabelecer dominância de um método em relação ao outro. Por exemplo, `B&B` obteve os melhores resultados para as classes caracterizadas por $m = 5, n = 50$ (segunda linha da tabela), $m = 10$, $n = 50$ (sexta linha da tabela) e $m = 25$, $n = 100$ (décima primeira linha da tabela); e `ME` conseguiu os melhores resultados para as classes com $m = 5, n = 100$ e $m = 25, n = 500$ dos intervalos [1,1000] e [1,10000] e $m = 10$ e $n = 100$ dos três intervalos. Novamente, a heurística `HI_PCmax` superou todos os outros métodos em termos de qualidade de solução. Esta encontrou dificuldades apenas no intervalo restrito em que $2 < n/m < 10$. Diferentemente do grupo `uniforme`, instâncias com tempos de processamento no intervalo $[1, 100]$ são mais difíceis do que as no intervalo $[1, 1000]$.

A Tabela 5.3 resume as estatísticas para os métodos `HI_PCmax` e `B&B`, ambos executados no mesmo ambiente computacional. Para cada grupo, para cada intervalo e para cada um dos métodos, a tabela mostra o número de ótimos, os valores médio e máximo encontrados para o erro absoluto, os valores médio e máximo encontrados para o erro relativo e os valores médio e máximo do tempo de processamento em segundos. Os melhores resultados estão em destaque na tabela. A superioridade de `HI_PCmax` é mais evidente para as classes do grupo `não-uniforme`. Esta heurística, além de ter obtido um número maior de melhores soluções, obteve também os melhores tempos.

Os resultados detalhados disponíveis em [1] ajudam a perceber melhor essas tendências. `HI_PCmax` é uma heurística iterativa de múltiplas partidas e três fases, além da etapa inicial de pré-processamento. Quando a etapa inicial não encontra solução comprovadamente ótima, para um dado valor de $C_{\max}$, a melhor solução pode ser encontrada na primeira tentativa (heurística construtiva `DWSFD`), na segunda tentativa (heurística construtiva `AHS`) ou na última tentativa (heurística construtiva `LPT`). Em cada tentativa, isso pode acontecer em uma das três fases distintas de `C+R+M_BP`. O grupo `uniforme` é considerado fácil e, nas tabelas disponíveis em [1], isso pode ser verificado observando-se que a grande maioria das melhores soluções foi encontrada na etapa inicial (coluna F, valor `I`). Do total de 390 instâncias do grupo `não-uniforme`, em 369 instâncias a melhor solução é comprovadamente ótima: 61 foram resolvidas pela heurística `LPT` e 143 por `DWSFD` na etapa

inicial. Das 165 instâncias restantes, 50 não foram resolvidas na primeira tentativa. Destas, 41 instâncias pertencem às classes em que $2 < n/m \leq 10$, justamente as consideradas mais difíceis. Este conjunto de classes difíceis engloba 120 instâncias e em todas elas a melhor solução encontrada foi obtida na fase de `Melhoria`, qualquer que seja a heurística inicial. Estes dados podem ser observados nos resultados detalhados nas tabelas disponíveis em [1] e mostram a importância da diversidade de soluções iniciais e das três fases do algoritmo.

A análise dos resultados é favorável à heurística `HI_PCmax`. Para os dois grupos `uniforme` e `não-uniforme`, em todas as classes de problemas, a heurística `HI_PCmax` obteve resultados superiores (no número de soluções ótimas e na média do erro relativo) aos dos outros quatro algoritmos. A abordagem dual para o problema de BP, usada inicialmente para resolver instâncias de BP, também mostrou ser muito eficiente quando aplicada ao problema $P||C_{\max}$.

O próximo e último capítulo ressalta as principais contribuições desta tese.

		LPT		B&B			HI_PCmax			3-PHASE
m	n	erro	ót.	erro	ót.	tempo	erro	ót.	tempo	erro
5	10	3.54e-03	9	0	10	0.00	0	10	0.00	0.018
5	50	4.58e-03	1	0	10	0.00	0	10	0.00	0.000
5	100	8.81e-04	4	0	10	0.00	0	10	0.00	0.000
5	500	0	10	0	10	0.00	0	10	0.00	0.000
5	1000	0	10	0	10	0.00	0	10	0.00	0.000
10	50	1.56e-02	0	0	10	0.00	0	10	0.00	0.002
10	100	3.64e-03	1	0	10	0.00	0	10	0.00	0.002
10	500	1.20e-04	7	0	10	0.00	0	10	0.00	0.000
10	1000	0	10	0	10	0.00	0	10	0.00	0.000
25	50	8.61e-03	6	0	10	0.00	0	10	0.02	0.011
25	100	2.37e-02	0	0	10	0.00	0	10	0.00	0.003
25	500	9.04e-04	4	0	10	0.00	0	10	0.00	0.000
25	1000	0	10	0	10	0.00	0	10	0.00	0.000

Tabela 5.4: Estatísticas, grupo `uniforme`, intervalo [1,100]

		LPT		B&B			HI_PCmax			3-PHASE
m	n	erro	ót.	erro	ót.	tempo	erro	ót.	tempo	erro
5	10	0	10	0	10	0.00	0	10	0.00	0.010
5	50	3.33e-03	0	0	10	0.00	0	10	0.00	0.001
5	100	1.02e-03	0	0	10	0.00	0	10	0.00	0.000
5	500	4.63e-05	1	0	10	0.00	0	10	0.03	0.000
5	1000	7.97e-06	5	0	10	0.00	0	10	0.07	0.000
10	50	1.61e-02	0	8.17e-05	8	0.03	7.74e-05	8	0.03	0.002
10	100	4.04e-03	0	0	10	0.00	0	10	0.00	0.000
10	500	2.21e-04	0	0	10	0.00	0	10	0.01	0.000
10	1000	4.42e-05	3	0	10	0.00	0	10	0.05	0.000
25	50	1.06e-02	4	0	10	0.00	0	10	0.03	0.011
25	100	3.15e-02	0	2.07e-04	8	0.31	9.93e-05	8	0.06	0.003
25	500	1.41e-03	0	0	10	0.00	0	10	0.01	0.000
25	1000	2.75e-04	0	0	10	0.00	0	10	0.03	0.000

Tabela 5.5: Estatísticas, grupo `uniforme`, intervalo [1,1000]

		LPT		B&B			HI_PCmax			3-PHASE
m	n	erro	ót.	erro	ót.	tempo	erro	ót.	tempo	erro
5	10	6.48e-03	9	0	10	0.00	0	10	0.04	0.013
5	50	5.92e-03	0	9.26e-06	7	0.03	0	10	0.04	0.000
5	100	1.41e-03	0	0	10	0.00	0	10	0.00	0.000
5	500	4.87e-05	0	0	10	0.00	0	10	0.02	0.000
5	1000	1.02e-05	0	0	10	0.00	0	10	0.18	0.000
10	50	2.45e-02	0	1.14e-03	0	0.35	1.03e-04	0	0.45	0.004
10	100	4.82e-03	0	0	10	0.00	0	10	0.00	0.000
10	500	2.34e-04	0	0	10	0.00	0	10	0.01	0.000
10	1000	6.31e-05	0	0	10	0.00	0	10	0.07	0.000
25	50	7.25e-03	6	0	10	0.00	0	10	0.06	0.004
25	100	2.76e-02	0	4.49e-03	0	1.79	3.47e-04	0	1.01	0.001
25	500	1.00e-03	0	0	10	0.00	0	10	0.01	-[a]
25	1000	3.12e-04	0	0	10	0.00	0	10	0.04	0.000

[a] Informação não disponibilizada em [26].

Tabela 5.6: Estatísticas, grupo `uniforme`, intervalo [1,10000]

		LPT		B&B			HI_PCmax			ME		
m	n	erro	ót.	erro	ót.	tempo	erro	ót.	tempo	erro	tempo	melhor
5	10	0	10	0	10	0.00	0	10	0.00	0	0	1-SPT
5	50	9.37e-03	0	7.24e-03	0	0.24	0	10	0.03	8.58e-03	0.01	1-SPT
5	100	1.67e-02	0	0	10	0.00	0	10	0.02	5.31e-05	0.02	1-SPT
5	500	5.86e-04	0	0	10	0.02	0	10	0.01	0	1.01	1-BPT
5	1000	1.44e-04	0	0	10	0.00	0	10	0.04	0	9.71	1-BPT
10	50	1.13e-02	0	8.34e-03	2	0.81	7.28e-03	4	0.46	1.57e-02	0.00	1-SPT
10	100	9.02e-03	0	7.96e-03	0	0.48	2.13e-04	8	0.34	5.09e-03	0.04	1-SPT
10	500	1.00e-02	0	0	10	0.04	0	10	0.00	2.13e-05	3.26	1-SPT
10	1000	3.83e-04	1	0	10	0.02	0	10	0.01	0	17.14	1-BPT
25	50	0	10	0	10	0.00	0	10	0.00	0	0.01	1-SPT
25	100	4.77e-03	0	2.65e-03	3	3.82	1.34e-03	8	0.61	9.85e-03	0.04	1-SPT
25	500	9.79e-03	0	1.60e-04	8	3.37	0	10	0.09	2.12e-04	3.29	1-BPT
25	1000	9.30e-03	0	1.17e-03	4	18.80	0	10	0.20	7.97e-05	36.59	1-BPT

Tabela 5.7: Estatísticas, grupo `não-uniforme`, intervalo $[1, 100]$

		LPT		B&B			HI_PCmax			ME		
m	n	erro	ót.	erro	ót.	tempo	erro	ót.	tempo	erro	tempo	melhor
5	10	0	10	0	10	0.00	0	10	0.00	0	0	1-SPT
5	50	8.82e-03	0	7.94e-03	0	0.37	0	10	0.05	8.92e-03	0.01	1-SPT
5	100	1.70e-02	0	1.55e-04	9	0.05	0	10	0.02	7.43e-05	0.06	1-SPT
5	500	6.35e-04	0	0	10	0.01	0	10	0.01	0	1.39	1-BPT
5	1000	1.78e-04	0	0	10	0.00	0	10	0.04	0	14.80	1-BPT
10	50	4.08e-03	0	1.57e-03	0	1.02	0	10	0.01	1.46e-02	0.01	1-SPT
10	100	8.66e-03	0	8.18e-03	0	0.74	0	10	0.44	4.64e-03	0.07	1-SPT
10	500	1.01e-02	0	0	10	0.02	0	10	0.11	0	5.61	1-SPT
10	1000	4.12e-04	0	0	10	0.21	0	10	0.07	0	14.61	1-BPT
25	50	0	10	0	10	0.00	0	10	0.00	0	0.01	1-SPT
25	100	4.67e-03	0	3.80e-03	0	2.66	1.34e-03	8	1.18	7.93e-03	0.08	1-SPT
25	500	1.00e-02	0	1.39e-03	1	13.04	0	10	0.13	4.25e-05	15.39	1-SPT
25	1000	9.38e-03	0	7.97e-06	9	1.59	0	10	0.56	7.97e-06	138.21	1-SPT

Tabela 5.8: Estatísticas, grupo `não-uniforme`, intervalo $[1, 1000]$

		LPT		B&B			HI_PCmax			ME		
m	n	erro	ót.	erro	ót.	tempo	erro	ót.	tempo	erro	tempo	melhor
5	10	0	10	0	10	0.00	0	10	0.00	0	0.00	1-SPT
5	50	8.79e-03	0	8.11e-03	0	0.45	0	10	0.03	8.95e-03	0.01	1-BPT
5	100	1.70e-02	0	1.00e-04	8	0.12	0	10	0.02	5.78e-05	0.09	1-SPT
5	500	6.42e-04	0	0	10	0.00	0	10	0.01	0	1.97	1-SPT
5	1000	1.78e-04	0	0	10	0.00	0	10	0.06	0	13.88	1-BPT
10	50	4.12e-03	0	2.02e-03	0	1.11	4.22e-06	8	0.42	1.46e-02	0.01	1-BPT
10	100	8.61e-03	0	8.28e-03	0	0.88	0	10	0.30	4.63e-03	0.15	1-SPT
10	500	1.02e-02	0	0	10	0.04	0	10	0.01	1.06e+00	7.99	1-BPT
10	1000	4.10e-04	0	0	10	0.03	0	10	0.03	0	15.57	1-SPT
25	50	0	10	0	10	0.00	0	10	0.00	0	0.01	1-SPT
25	100	4.73e-03	0	4.16e-03	0	2.90	1.35e-03	3	4.84	7.76e-03	0.14	1-SPT
25	500	1.01e-02	0	7.23e-04	1	32.77	0	10	3.43	1.91e-05	20.37	1-BPT
25	1000	9.40e-03	0	1.86e-06	8	13.60	0	10	0.26	5.31e-06	195.88	1-SPT

Tabela 5.9: Estatísticas, grupo `não-uniforme`, intervalo $[1, 10000]$

		HI_PCmax						B&B					
		ótimos	erro abs.		erro rel.		tempo		ótimos	erro abs.		erro rel.		tempo	
grupo	$w_i \in$		méd.	máx.	méd.	máx.	méd	máx.		méd.	máx.	méd.	máx.	méd	máx.
uniforme	$[1, 100]$	130	0.00	0	0	0	0	0.16	130	0.00	0	0	0	0	0
	$[1, 1000]$	126	0.03	1	1.36e-05	5.15e-04	0.02	0.33	126	0.05	3	2.22e-05	1.55e-03	0.03	2.26
	$[1, 10000]$	110	0.75	12	3.46e-05	5.83e-04	0.15	1.36	107	9.31	173	4.33e-04	8.30e-03	0.17	3.20
não	$[1, 100]$	120	0.32	7	6.79e-04	1.50e-02	0.14	4.10	87	1.84	20	2.12e-03	1.50e-02	2.12	68.03
uniforme	$[1, 1000]$	128	0.38	25	1.03e-04	6.71e-03	0.20	10.07	79	15.59	152	1.77e-03	9.42e-03	1.52	22.43
	$[1, 10000]$	121	3.90	253	1.04e-04	6.75e-03	0.72	21.89	77	150.01	880	1.80e-03	9.73e-03	3.99	57.33

Tabela 5.10: Comparação dos métodos HI_PCmax e B&B