6
Estratégias de paralelização para heurísticas GRASP

Tempos computacionais elevados são observados para os algoritmos GRASP com religamento de caminhos propostos para o AP3 e para o JSP nos Capítulos 4 e 5, respectivamente. Além disso, esses tempos computacionais crescem consideravelmente com o tamanho do problema testado. Por exemplo, a Figura 6.1 mostra em escala log-log, os tempos de processamento para 10000 iterações da variante `GRC(ALL)` do método GRASP com religamento de caminhos para o AP3, para dimensões de problemas testes dispostas em ordem crescente. O modelo de regressão $T = 10^{-2.56}n^{2.71}$, onde T é o tempo de processamento em segundos (em um computador SGI Challenge com 16 processadores MIPS R10000 de 196 MHz, 12 processadores MIPS R10000 de 194 MHz e 7.6 Gb de memória) e n é a dimensão do problema, é mostrado na figura como uma reta. O valor do coeficiente de determinação múltipla (R^2) é 0.968. Verifica-se que para a implementação seqüencial de `GRC(ALL)`, os tempos de processamento aumentam super-quadraticamente com a dimensão do problema. Dessa forma, é natural que implementações paralelas sejam consideradas para acelerar os algoritmos propostos nos capítulos anteriores.

Nesse capítulo, as estratégias seqüenciais com religamento de caminhos propostas nos Capítulos 4 e 5 serão generalizadas em um único algoritmo. Em seguida, serão descritas duas estratégias de paralelização para esse algoritmo. Na primeira estratégia, chamada de *independente*, a comunicação entre processos é limitada à entrada dos dados, à detecção do término do programa e à determinação da melhor solução global. A segunda abordagem, chamada de *cooperativa*, além de realizar a comunicação feita na abordagem independente, permite a troca de informações armazenadas nos conjuntos de elite.

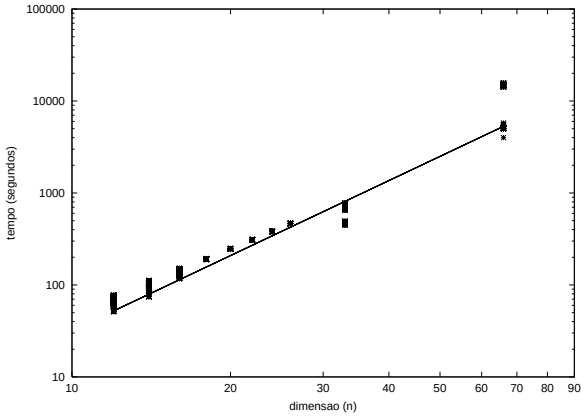

Figura 6.1: Tempo de processamento em função da dimensão do problema para 10000 iterações da variante do algoritmo GRASP com religamento de caminhos `GRC(ALL)` para o AP3.

6.1 Algoritmo GRASP com religamento de caminhos

As estratégias de GRASP com religamento de caminhos descritas nos capítulos anteriores podem ser generalizadas conforme o algoritmo descrito na Figura 6.2. Seja *maxpool* o tamanho máximo que o conjunto de elite pode assumir. As primeiras *maxpool* iterações do algoritmo contribuem cada uma com uma solução para o conjunto de elite (linha 16). O religamento de caminhos será realizado somente quando o *pool* de soluções estiver completo, ou seja, $|P| = maxpool$.

Nas linhas 3 e 4, as fases construtiva e de busca local são realizadas. Cada uma dessas fases pode ser substituída por mecanismos mais elaborados, tais como a fase construtiva desenvolvida para o JSP que usa de forma alternada duas funções gulosas.

Uma vez que o conjunto de soluções de elite esteja completo, a qualidade de uma solução S produzida pela busca local é testada (linha 6). Caso S seja aceita de acordo com o critério de qualidade usado, o religamento de caminhos bidirecional é feito entre S e todos os elementos de um subconjunto $P' \subseteq P$ (linhas 7 a 15). O grau de restritividade do critério de qualidade é proporcional ao tempo computacional necessário para realizar a etapa de religamento de caminhos. Por exemplo, no AP3 a etapa de religamento de caminhos é feita de forma eficiente e, por isso, todas as soluções geradas pela busca local participam dessa etapa. Após cada etapa de religamento de caminhos, a melhor solução obtida na trajetória percorrida é testada para inserção no conjunto de elite (linhas 11 e 13).

O procedimento de intensificação no conjunto de elite é realizado a

```
procedimento GRASP_RC(seed, alvo, maxitr, maxpool, ifreq, dados_problema)
1     P = ∅;
2     para i = 1, ..., maxitr faça
3           CONSTRUTIVA(seed, S, dados_problema);
4           LOCAL(S, dados_problema);
5           se |P| == maxpool então
6                 aceito = VERIFICA_FAZER_RELIGAMENTO(S);
7                 se aceito então
8                       selecione P' ⊆ P;
9                       para T ∈ P' faça
10                            S_gmin = RELIGAMENTO(c_S, S, T, dados_problema);
11                            ATUALIZAR_POOL(S_gmin, c_gmin, P);
12                            S_gmin = RELIGAMENTO(c_T, T, S, dados_problema);
13                            ATUALIZAR_POOL(S_gmin, c_gmin, P);
14                      fim-para;
15                fim-se;
16          senão P = P ∪ {S} fim-se;
17          se mod (i,ifreq) == 0 então
18                INTENSIFICAÇÃO(P);
19          fim-se;
20          S_melhor = POOLMIN(P);
21          se c_melhor ≤ alvo então
22                Saia do laço;
23          fim-se;
24    fim-para;
25    POS-OTIMIZAÇÃO(P);
26    S_melhor = POOLMIN(P);
27    retorne (S_melhor);
fim GRASP_RC;
```

Figura 6.2: Algoritmo GRASP com religamento de caminhos.

cada intervalo de *ifreq* iterações (linhas 17 a 19). Esse procedimento realiza o religamento de caminhos entre cada par de soluções armazenadas no *pool*, conforme explicado nas Seções 4.3 e 5.3.

O laço do algoritmo GRASP com religamento de caminhos nas linhas 2 a 24 é executado por no máximo *maxitr* iterações, mas pode ser terminado caso uma solução com custo inferior a *alvo* seja encontrada (linhas 21 a 23).

Finalmente, o procedimento de pós-otimização é aplicado ao conjunto de elite na linha 26.

6.2 Abordagem paralela independente

Um esquema básico de paralelização do método GRASP com religamento de caminhos é mostrado nessa seção. A Figura 6.3 mostra o pseudo-código dessa abordagem. De acordo com as taxonomias propostas

em [38, 129], essa abordagem paralela é classificada como *múltiplas trajetórias independentes*.

A implementação foi desenvolvida sobre o paradigma de memória distribuída e usa passagem de mensagens para fazer a comunicação entre os ρ processos usados. A comunicação é limitada à inicialização e ao término do programa. Um único processador faz a leitura dos dados, que são enviados aos $\rho - 1$ processos restantes. Um processo envia uma mensagem a todos os outros após encontrar uma solução tão boa quanto a solução alvo, ou após completar o seu número máximo de iterações.

O método GRASP com religamento de caminhos paralelo independente é construído a partir do algoritmo seqüencial mostrado na Figura 6.2. Cada processo executa uma cópia do programa. As diferenças entre o algoritmo seqüencial e o algoritmo paralelo independente são descritas a seguir. Na linha 1 da Figura 6.3 o identificador do processo *meu_id* e o número de processos usados ρ são determinados. Cada fase construtiva é iniciada com uma semente do gerador de números aleatórios. Para assegurar que os processos irão computar iterações independentes, sementes idênticas do gerador de números aleatórios (`rand()`) não podem ser usadas por mais de um processo para iniciar uma fase construtiva. A semente inicial usada pelo processo *meu_id* é computada nas linhas 2 a 4. Dessa forma, cada processo possui uma seqüência disjunta de $maxitr/\rho$ sementes que serão usadas para iniciar cada iteração.

O laço das linhas 6 a 36 computa as iterações. Os procedimentos de construção, de busca local e de religamento de caminhos são idênticos aos realizados no algoritmo seqüencial.

Na linha 23, caso um processo encontre uma solução com custo menor ou igual a *alvo*, esse processo envia uma mensagem para todos os demais indicando que uma solução satisfatória foi encontrada. Da mesma forma, caso um processo complete o seu número máximo de iterações $maxitr/\rho$, ele envia uma mensagem a todos os outros comunicando esse fato (linhas 24 a 27).

Na linha 28, o processo verifica se alguma mensagem foi enviada a ele. Caso exista uma mensagem indicando que uma solução com custo menor ou igual a *alvo* foi encontrada, as iterações são finalizadas na linha 30. Caso uma mensagem indicando que o número máximo de iterações foi alcançado por algum processo tenha sido recebida, o contador *num_terminados* do número de processos prontos para terminar é incrementado na linha 32. Após todos os processos terem completado o seu número máximo de iterações, a execução do laço principal é terminada na linha 35.

```
procedimento GRASP_RC_INDEPENDENTE(seed, alvo, maxitr, maxpool, ifreq, dados_problema)
1  meu_rank = OBTER_RANK(); ρ = OBTER_NUM_PROCS();
2  para i = 1, ..., (maxitr/ρ) * meu_rank faça
3      seed = rand(seed);
4  fim-para;
5  P = ∅; num_terminados = 0;
6  para i = 1, ..., ∞ faça
7      CONSTRUTIVA(seed, S, dados_problema);
8      LOCAL(S, dados_problema);
9      se |P| == maxpool então
10         aceito = VERIFICAR_FAZER_RELIGAMENTO(S);
11         se aceito então
12             selecione P' ⊆ P;
13             para T ∈ P' faça
14                 S_gmin = RELIGAMENTO(c_S, S, T, dados_problema);
15                 ATUALIZAR_POOL(S_gmin, c_gmin, P);
16                 S_gmin = RELIGAMENTO(c_T, T, S, dados_problema);
17                 ATUALIZAR_POOL(S_gmin, c_gmin, P);
18             fim-para;
19         fim-se;
20     senão P = P ∪ {S} fim-se;
21     se  mod (i,ifreq) == 0 então INTENSIFICAÇÃO(P); fim-se;
22     S_melhor = POOLMIN(P);
23     se c_melhor ≤ alvo então ENVIAR_PARA_TODOS(achou_alvo) fim-se;
24     se i == maxitr/ρ então
25         num_terminados = num_terminados + 1;
26         ENVIAR_PARA_TODOS(maxitr_terminadas);
27     fim-se;
28     recebido = VERIFICAR_RECEBIMENTO(flag);
29     se recebido então
30         se flag == achou_alvo então break;
31         senão se flag == maxitr_terminadas então
32             num_terminados = num_terminados + 1;
33         fim-se;
34     fim-se;
35     se num_terminados == ρ então break fim-se;
36 fim-para;
37 POS-OTIMIZAÇÃO(P);
38 S_MelhorGlobal = OBTER_MELHOR_GLOBAL(S_melhor);
39 retorne (S_MelhorGlobal);
fim GRASP_RC_INDEPENDENTE;
```

Figura 6.3: Estratégia paralela independente de GRASP com religamento de caminhos.

Cada processo executa a fase de pós-otimização no *pool* de soluções de elite (linha 37) após sair do laço principal (linhas 6 a 36). Em seguida, um operador de redução (`OBTER_MELHOR_GLOBAL`) é usado para determinar a melhor solução obtida globalmente entre todos os processadores (linha 38) e essa solução é retornada (linha 39).

Uma estratégia paralela de GRASP puro pode ser obtida a partir do algoritmo da Figura 6.3, caso as linhas 9 a 19 não sejam executadas. Em um algoritmo GRASP puro é necessário armazenar apenas a melhor solução encontrada por cada processo. Um *pool* de tamanho $|P| = 1$ é usado nessa estratégia. Por esse motivo, os procedimentos de intensificação e de pós-otimização não são executados em uma abordagem paralela do método puro.

6.3 Abordagem paralela cooperativa

Na abordagem paralela cooperativa de GRASP com religamento de caminhos, os processos comunicam informações armazenadas no conjunto de elite. Um algoritmo para essa abordagem é mostrado na Figura 6.4. Esse algoritmo é construído a partir do esquema independente apresentado na seção anterior. A discussão sobre o algoritmo cooperativo será limitada às suas diferenças em relação ao algoritmo independente.

As diferenças entre as abordagens independente e cooperativa ocorrem na fase de religamento de caminhos. Antes de executar o religamento de caminhos entre duas soluções S e T, cada processo verifica se um ou mais processos enviaram novas soluções de elite para ele. Caso existam soluções de elite para serem recebidas, o procedimento `RECEBER_SOLUÇÕES` (linhas 14 e 18) faz o recebimento das soluções, testa se cada solução recebida pode ser inserida no conjunto local de soluções de elite e insere as soluções aceitas. Após percorrer cada trajetória de religamento de caminhos, caso o conjunto local de soluções de elite tenha sido atualizado, o processo escreve as novas soluções do conjunto de elite em um *buffer* local de envio (linhas 17 e 21). Na linha 23, se o *buffer* local de envio não estiver vazio, o processo envia o *buffer* para os outros processos.

Outra diferença entre os esquemas independente e cooperativo diz respeito ao procedimento de intensificação. No esquema cooperativo, toda vez que o conjunto local de soluções de elite é atualizado, as novas soluções de elite são escritas no *buffer* local de envio. As soluções armazenadas

```
procedimento GRASP_RC_COOPERATIVO(seed, alvo, maxitr, maxpool, ifreq, dados_problema)
1  meu_rank = OBTER_RANK(); ρ = OBTER_NUM_PROCS();
2  para i = 1, ..., (maxitr/ρ) * meu_rank faça
3      seed = rand(seed);
4  fim-para;
5  P = ∅; num_terminados = 0;
6  para i = 1, ..., ∞ faça
7      CONSTRUTIVA(seed, S, dados_problema);
8      LOCAL(S, dados_problema);
9      se |P| == maxpool então
10         aceito = VERIFICAR_FAZER_RELIGAMENTO(S);
11         se aceito então
12             selecione P' ⊆ P;
13             para T ∈ P' faça
14                 RECEBER_SOLUÇÕES(P);
15                 S_gmin = RELIGAMENTO(c_S, S, T, dados_problema);
16                 atualizado = ATUALIZAR_POOL(S_gmin, c_gmin, P);
17                 se (atualizado) então INSERIR_BUFFER_ENVIO(S_gmin, c_gmin, buffer) fim-se;
18                 RECEBER_SOLUÇÕES(P);
19                 S_gmin = RELIGAMENTO(c_T, T, S, dados_problema);
20                 atualizado = ATUALIZAR_POOL(S_gmin, c_gmin, P);
21                 se (atualizado) então INSERIR_BUFFER_ENVIO(S_gmin, c_gmin, buffer) fim-se;
22             fim-para;
23             ENVIAR_SOLUÇÕES(buffer);
24         fim-se;
25     senão P = P ∪ {S} fim-se;
26     se  mod (i,ifreq) == 0 então INTENSIFICAÇÃO(P) fim-se;
27     S_melhor = POOLMIN(P);
28     se c_melhor ≤ alvo então ENVIAR_PARA_TODOS(achou_alvo) fim-se;
29     se i == maxitr/ρ então
30         num_terminados = num_terminados + 1;
31         ENVIAR_PARA_TODOS(maxitr_terminadas)
32     fim-se;
33     recebido = VERIFICAR_RECEBIMENTO(flag);
34     se recebido então
35         se flag == achou_alvo então break;
36         senão se flag == maxitr_terminadas então
37             num_terminados = num_terminados + 1;
38         fim-se;
39     fim-se;
40     se num_terminados == ρ então break fim-se;
41 fim-para;
42 POS-OTIMIZAÇÃO(P);
43 S_MelhorGlobal = OBTER_MELHOR_GLOBAL(S_melhor);
44 retorne (S_MelhorGlobal);
fim GRASP_RC_COOPERATIVO;
```

Figura 6.4: Estratégia paralela cooperativa de GRASP com religamento de caminhos.

no *buffer* são enviadas para os outros processos na próxima vez que o procedimento `ENVIAR_SOLUÇÕES` for executado.

6.4 Resultados computacionais

Nessa seção serão descritos e analisados os resultados dos experimentos computacionais feitos com as estratégias paralelas pura e com religamento de caminhos propostas. As estratégias paralelas foram implementadas para o AP3 e para o JSP a partir dos algoritmos GRASP descritos nos Capítulos 4 e 5, respectivamente.

6.4.1 Ambiente computacional

Os experimentos foram realizados em um computador SGI Challenge com 16 processadores MIPS R10000 de 196 MHz, 12 processadores MIPS R10000 de 194 MHz e 7.6 Gb de memória.

Os programas foram escritos em Fortran e foram compilados com o compilador SGI MIPSpro F77 usando as opções `-O3 -static -u`. A especificação MPI (*Message Passing Interface*) é considerada como sendo um padrão para bibliotecas de envio de mensagem para computação paralela [132]. Os códigos paralelos usam o *Message Passing Toolkit* 1.4 da SGI, que contém uma implementação da especificação do MPI 1.2. Foram medidos os tempos corridos dos programas, usando-se a função `MPI_WT` do MPI. Os tempos medidos excluem o tempo necessário para fazer a entrada dos dados, inicializar o gerador de números aleatórios e imprimir a saída dos dados.

As implementações paralelas foram executadas em um, dois, quatro, oito e 16 processadores. A carga da máquina foi medida com a função `top` do sistema operacional e manteve-se baixa durante todos os experimentos. Isso indica que, durante a execução de um programa paralelo, todos os seus processos estavam ativos em uma das CPUs da máquina.

Cada aceleração (tempo de execução do programa seqüencial dividido pelo tempo de execução do programa paralelo) foi calculada comparando-se os tempos de execução em dois, quatro, oito ou 16 processadores com o tempo necessário para o programa paralelo independente ser executado em um único processador. Os tempos de execução do programa paralelo independente executando em um processador e do programa seqüencial

são aproximadamente iguais. Optou-se por usar o tempo de execução do programa paralelo independente em um processador, ao invés do tempo de execução do programa seqüencial, para manter a homogeneidade na maneira como os tempos de execução são medidos. As acelerações e eficiências médias mostradas nas tabelas e gráficos desse capítulo são calculadas usando-se as médias dos tempos de execução do programa paralelo para dois, quatro, oito ou 16 processadores e a média dos tempos do programa paralelo independente em um único processador, para 60 execuções independentes realizadas para cada caso testado.

6.4.2 Heurísticas GRASP paralelizadas

Os algoritmos paralelos usados nesses experimentos são:

1. o algoritmo GRASP puro;
2. o algoritmo GRASP com religamento de caminhos independente;
3. o algoritmo GRASP com religamento de caminhos cooperativo.

Para o AP3, os algoritmos paralelos testados correspondem à paralelização das variantes `GRASP` e `GRC(ALL)`, descritas no Capítulo 4. Para o JSP, os programas usados nos experimentos desse capítulo correspondem à paralelização das variantes `GP` e `GP+RC` descritas no Capítulo 5. Os parâmetros dos procedimentos usados nas abordagem paralelas foram os mesmos usados para testar cada programa seqüencial nos Capítulos 4 e 5. Os procedimentos de intensificação e a pós-otimização não são realizados durante esses experimentos.

6.4.3 Experimentos com estratégias de paralelização

O objetivo desses experimentos é verificar computacionalmente o resultado da Proposição 3.2 mostrada no Capítulo 3. Nos Capítulos 3, 4 e 5 verificou-se que os tempos para valor alvo do algoritmo GRASP e de suas variantes com religamento de caminhos adaptam-se a uma distribuição exponencial de dois parâmetros. A seguinte afirmação pode ser feita sobre essa distribuição [8, 129]. Seja $P_\rho(t)$ a probabilidade de uma solução alvo não ter sido encontrada em t unidades de tempo usando-se ρ processos independentes. Se $P_1(t) = e^{-(t-\mu)/\lambda}$ com $\lambda \in \mathbb{R}^+$ e $\mu \in \mathbb{R}$, isto é, P_1

corresponde a uma distribuição exponencial de dois parâmetros, então $P_\rho(t) = e^{-\rho(t-\mu)/\lambda}$. Isso vem da definição da distribuição exponencial de dois parâmetros e foi demonstrado na Seção 3.1. Essa propriedade implica em que a probabilidade de encontrar uma solução com um dado valor da função objetivo em tempo ρt usando-se um processo seqüencial é $1 - e^{-(\rho t-\mu)/\lambda}$, enquanto que a probabilidade de se obter uma solução com custo pelo menos tão bom quanto um dado valor em tempo t com ρ processos paralelos independentes é $1 - e^{-\rho(t-\mu)/\lambda}$. Deve-se notar que caso $\mu = 0$, então as duas probabilidades são idênticas e correspondem a uma distribuição exponencial simples. Além disso, caso $\rho|\mu| \ll \lambda$, então as duas probabilidades são aproximadamente iguais e é possível alcançar uma aceleração aproximadamente linear no tempo para solução alvo usando-se múltiplos processos independentes.

No contexto do algoritmo GRASP, o parâmetro μ é uma estimativa do tempo mínimo necessário para encontrar o valor alvo para o par problema teste/valor alvo testado. O parâmetro λ é uma estimativa do espalhamento dos tempos medidos para o par problema teste/valor alvo testado.

Nesses experimentos serão estudadas as acelerações da abordagem paralela do algoritmo GRASP puro e da abordagem independente do algoritmo com religamento de caminhos implementadas para o AP3 e para o JSP. Pretende-se também comparar as abordagens cooperativa e independente com religamento de caminhos segundo o tempo para valor alvo. Para isso, o critério de parada por número máximo de iterações é desativado. Portanto, os programas são terminados apenas quando uma solução com custo tão bom quanto o valor alvo é encontrada.

Resultados dos experimentos paralelos para o AP3

Para estudar a implementação paralela de `GRASP` foram usados os problemas `B-S 20.1`, `B-S 22.1`, `B-S 24.1` e `B-S 26.1` de Balas e Saltzman [13], com valores alvos 16, 16, 16 e 17, respectivamente. As implementações paralelas independente e cooperativa de `GRC(ALL)` foram estudadas para os problemas `B-S 20.1`, `B-S 22.1`, `B-S 24.1` e `B-S 26.1`, com os valores alvos 7, 8, 7 e 8, respectivamente. Na Figura 6.5 são mostradas as acelerações e distribuições empíricas para a implementação paralela de `GRASP`. Analogamente, as acelerações e as distribuições empíricas para as implementações paralelas independente e cooperativa de `GRC(ALL)` são mostradas nas Figuras 6.6, 6.7, 6.8 e 6.9. Os gráficos com as acelerações e as distribuições empíricas foram gerados a partir de 60 execuções independentes para cada

Tabela 6.1: Aceleração e eficiência para problemas testes do AP3. A implementação testada foi a estratégia de paralelização para `GRASP`. Os problemas são `B-S 20.1`, `B-S 22.1`, `B-S 24.1` e `B-S 26.1`, com valores alvos 16, 16, 16 e 17, respectivamente. São mostrados os valores dos parâmetros da distribuição exponencial de dois parâmetros para cada par problema teste/valor alvo.

	parâmetros da exponencial			número de processadores							
				2		4		8		16	
problema	μ	λ	$\lvert\mu\rvert/\lambda$	acel.	efic.	acel.	efic.	acel.	efic.	acel.	efic.
`B-S 20.1`	1.28	90.18	0.014	2.02	1.01	3.66	0.91	6.05	0.75	16.30	1.01
`B-S 22.1`	-2.607	185.21	0.014	2.03	1.01	4.58	1.14	10.33	1.29	17.88	1.11
`B-S 24.1`	-2.890	246.55	0.011	2.16	1.08	4.27	1.06	7.89	0.98	13.91	0.86
`B-S 26.1`	26.36	252.90	0.100	1.62	0.81	3.22	0.80	6.23	0.77	11.72	0.73
média			**0.034**	**1.95**	**0.97**	**3.93**	**0.97**	**7.62**	**0.95**	**14.95**	**0.93**

Tabela 6.2: Estimativas da probabilidade de encontrar uma solução de custo pelo menos tão bom quanto o valor alvo em um dado tempo de execução, em função do número de processadores, para problemas testes do AP3. A implementação testada foi a estratégia de paralelização para `GRASP`. Os problemas são `B-S 20.1`, `B-S 22.1`, `B-S 24.1` e `B-S 26.1`, com valores alvos 16, 16, 16 e 17, respectivamente.

		probabilidade GRASP paralelo				
	tempo	número de processadores				
problema	(s)	1	2	4	8	16
`B-S 20.1`	10	0.11	0.22	0.37	0.05	0.87
	50	0.52	0.77	0.94	0.98	1.00
	100	0.72	0.92	1.00	1.00	1.00
`B-S 22.1`	10	0.04	0.07	0.18	0.41	0.62
	50	0.22	0.40	0.70	0.95	1.00
	100	0.45	0.69	0.90	1.00	1.00
`B-S 24.1`	10	0.05	0.10	0.19	0.33	0.46
	50	0.18	0.35	0.56	0.80	0.97
	100	0.27	0.54	0.82	0.95	1.00
`B-S 26.1`	10	0.04	0.05	0.13	0.24	0.44
	100	0.27	0.45	0.82	0.95	0.98
	500	0.91	0.98	1.00	1.00	1.00

Tabela 6.3: Aceleração com dois, quatro, oito e 16 processadores para problemas testes do AP3. As implementações testadas foram as estratégias de paralelização independente e cooperativa para `GRC(ALL)`. Os problemas são `B-S 20.1`, `B-S 22.1`, `B-S 24.1` e `B-S 26.1`, com valores alvos 7, 8, 7 e 8, respectivamente.

	parâmetros da exponencial			aceleração independente				aceleração cooperativo			
				número de processadores				número de processadores			
problema	μ	λ	$\lvert\mu\rvert/\lambda$	2	4	8	16	2	4	8	16
`B-S 20.1`	-26.46	1223.80	0.021	1.67	3.34	6.22	10.82	1.56	3.47	7.37	14.36
`B-S 22.1`	-135.12	3085.32	0.043	2.25	4.57	9.01	14.37	1.64	4.22	8.83	18.78
`B-S 24.1`	-16.76	4004.11	0.004	1.71	4.00	7.87	12.19	2.16	4.00	9.38	19.29
`B-S 26.1`	32.12	2255.55	0.014	2.11	3.89	6.10	11.49	2.16	5.30	9.55	16.00
média			**0.020**	**1.935**	**3.95**	**7.3**	**12.21**	**1.88**	**4.24**	**8.78**	**17.10**

Tabela 6.4: Estimativas da probabilidade de encontrar uma solução de custo pelo menos tão bom quanto o valor alvo em um dado tempo de execução, em função do número de processadores, para problemas testes do AP3. As implementações testadas foram as estratégias de paralelização independente e cooperativa para `GRC(ALL)`. Os problemas são `B-S 20.1`, `B-S 22.1`, `B-S 24.1` e `B-S 26.1`, com valores alvos 7, 8, 7 e 8, respectivamente. São mostrados os valores dos parâmetros da distribuição exponencial de dois parâmetros para cada par problema teste/valor alvo.

		probabilidade independente					probabilidade cooperativo				
	tempo	número de processadores					número de processadores				
problema	(s)	1	2	4	8	16	1	2	4	8	16
`B-S 20.1`	100	0.08	0.09	0.16	0.40	0.54	0.08	0.17	0.24	0.52	0.74
	500	0.36	0.53	0.75	0.93	1.00	0.36	0.57	0.70	0.93	0.96
	1000	0.55	0.77	0.95	0.98	1.00	0.55	0.74	0.97	1.00	1.00
`B-S 22.1`	100	0.03	0.05	0.14	0.24	0.40	0.03	0.09	0.15	0.32	0.62
	500	0.25	0.42	0.60	0.81	0.90	0.25	0.37	0.56	0.83	0.91
	1000	0.43	0.61	0.86	0.96	1.00	0.43	0.57	0.81	0.92	1.00
`B-S 24.1`	100	0.01	0.01	0.03	0.07	0.22	0.01	0.04	0.08	0.20	0.45
	500	0.18	0.26	0.33	0.62	0.84	0.18	0.28	0.46	0.77	0.94
	1000	0.29	0.41	0.69	0.95	0.98	0.29	0.50	0.68	0.94	1.00
`B-S 26.1`	100	0.03	0.03	0.16	0.19	0.31	0.03	0.08	0.18	0.36	0.48
	500	0.12	0.30	0.54	0.81	0.96	0.12	0.34	0.68	0.88	0.98
	1000	0.39	0.64	0.86	0.93	1.00	0.39	0.70	0.95	0.98	1.00

número de processadores considerado (um, dois, quatro, oito e 16 processadores). Para plotar as distribuições empíricas, os tempos medidos são ordenados, ao i-ésimo tempo de execução (t_i) é associada uma probabilidade $p_i = (i - \frac{1}{2})/60$ e, finalmente, os pontos $z_i = (t_i, p_i)$ são plotados para $i = 1, \ldots, 60$.

As acelerações e as eficiências (aceleração dividida pelo número de processadores usados) observadas nos gráficos da Figura 6.5 são mostradas na Tabela 6.1. Também são mostrados os valores dos parâmetros μ e λ da distribuição exponencial de dois parâmetros e de $|\mu|/\lambda$ para cada par problema teste/valor alvo testado. Esses parâmetros foram estimados na Subseção 4.5.5 a partir de 200 execuções independentes do `GRASP` seqüencial.

As acelerações observadas para o `GRASP` paralelo são aproximadamente lineares. São observadas algumas acelerações super-lineares, como por exemplo, para o problema `B-S 22.1`. Deve-se lembrar que os resultados exibidos nas tabelas e figuras desse capítulo são resultados de experimentos computacionais. Caso um número maior de execuções independentes fossem feitas, essas acelerações tenderiam em média para acelerações lineares. Da mesma forma, acelerações abaixo da linear, como observado para `B-S 24.1` com 16 processadores, deverão eventualmente aproximar-se da linear, caso um número maior de execuções independentes sejam realizadas. Deve-se notar

que o objetivo desses experimentos é ilustrar com resultados computacionais o que foi provado no Capítulo 3.

Observa-se na Tabela 6.1 que os valores de $|\mu|/\lambda$ para os problemas testes estudados para o `GRASP` paralelo são pequenos e que a média é 0.034. Portanto, acelerações aproximadamente lineares eram esperadas para o `GRASP` paralelo. Na Tabela 6.1 são observados alguns valores negativos para o parâmetro μ. Deve-se lembrar que λ e μ são estimativas dos parâmetros da exponencial de dois parâmetros e são calculados a partir de dados medidos. Valores negativos de μ ocorrem principalmente nas execuções realizadas para pares problemas testes/valores alvos onde é grande o intervalo entre os menores e os maiores tempos medidos. Dessa forma, para os casos em que o valor estimado do parâmetro μ for negativo, a exponencial que melhor descreve o comportamento do programa deverá ter um deslocamento real $\mu' > 0$, onde $\mu' \ll \lambda$.

As curvas das distribuições empíricas mostradas na Figura 6.5 permitem observar o comportamento das execuções realizadas para cada número de processadores testado. Na Tabela 6.2 mostra-se a probabilidade estimada para se obter uma solução de custo pelo menos tão bom quanto o valor alvo, em função do número de processadores e do tempo de execução. Esta tabela mostra que, por exemplo, para o problema `B-S 22.1`, a probabilidade de obter uma solução com valor no máximo igual a 16 em no máximo 50 segundos cresce de 22% com um processador, para 70% com quatro processadores e para 100% com 16 processadores.

Na Tabela 6.3 são resumidas as acelerações observadas nos gráficos das Figuras 6.6, 6.7, 6.8 e 6.9 para as abordagens paralelas independente e cooperativa de `GRC(ALL)`. São mostrados também os valores de μ, λ e $|\mu|/\lambda$, estimados na Subseção 4.5.5 a partir de 200 execuções independentes do `GRC(ALL)` seqüencial. Observa-se uma aceleração aproximadamente linear para a abordagem paralela independente de `GRC(ALL)` para até 8 processadores. Para 16 processadores, observa-se uma redução na aceleração, porém essa aceleração ainda pode ser considerada satisfatória. De fato, como observado na Tabela 6.3, os valores de $|\mu|/\lambda$ para os problemas testes estudados para a abordagem paralela independente de `GRC(ALL)` são pequenos (sua média é 0.020). Portanto, acelerações aproximadamente lineares também eram esperadas para o `GRC(ALL)` paralelo independente. Além disso, de acordo com a Proposição 3.2, espera-se uma redução na aceleração com o aumento do número de processadores ρ, como observado. Intuitivamente, isso pode ser explicado visto que o religamento de caminhos para o AP3 é feito de forma bastante eficiente, pois o custo de cada solução visitada é

calculado em tempo $O(1)$ a partir do custo da solução anterior na trajetória percorrida. Dessa forma, cada iteração de `GRC(ALL)` precisará de mais tempo computacional do que uma iteração de `GRASP`, porém o aumento no tempo computacional de cada iteração de `GRC(ALL)` não é o suficiente para reduzir significativamente a aceleração do programa.

A Proposição 3.2 não leva em conta a troca de informações entre os processos durante a execução do programa. Dessa forma, nenhuma conclusão sobre a aceleração da abordagem cooperativa pode ser tirada a partir das distribuições plotadas para o `GRC(ALL)` seqüencial. Em uma abordagem cooperativa, deve-se sempre levar em conta o fato de que a comunicação entre processos aumenta o tempo computacional do programa. Quanto maior for o número de processos usados, maior será o aumento no tempo computacional devido à comunicação. Portanto, a aplicação deve ser projetada de maneira que as informações trocadas entre processos sejam usadas para acelerar a heurística, de forma a compensar o tempo gasto com comunicação. Observa-se que para $\rho > 2$ processadores, as acelerações da abordagem cooperativa são em média superiores às acelerações da abordagem independente. Dessa forma, para o `GRC(ALL)` paralelo cooperativo a partir de $\rho = 4$ processadores para os problemas testes estudados, conclui-se que o tempo computacional gasto com a comunicação entre processos passa a ser compensado pelo ganho devido ao uso das informações recebidas.

Algumas acelerações super-lineares são observadas para o `GRC(ALL)` paralelo cooperativo na Tabela 6.3. De fato, acelerações super-lineares podem ocorrer para a abordagem cooperativa. A troca de informações entre processos melhora a qualidade das soluções do *pool* durante a execução do método GRASP. Verificou-se, durante os experimentos, que quanto melhor for a qualidade das soluções armazenadas no *pool*, maior será a probabilidade de que novas soluções de qualidade sejam geradas. Por exemplo, a melhor solução obtida pela estratégia muitas vezes foi gerada pelo procedimento de religamento de caminhos, logo após a inserção de uma solução de qualidade no *pool*. Portanto, as soluções de qualidade recebidas em um processo permitem reduzir o tempo de obtenção de uma solução com custo melhor ou igual ao valor alvo.

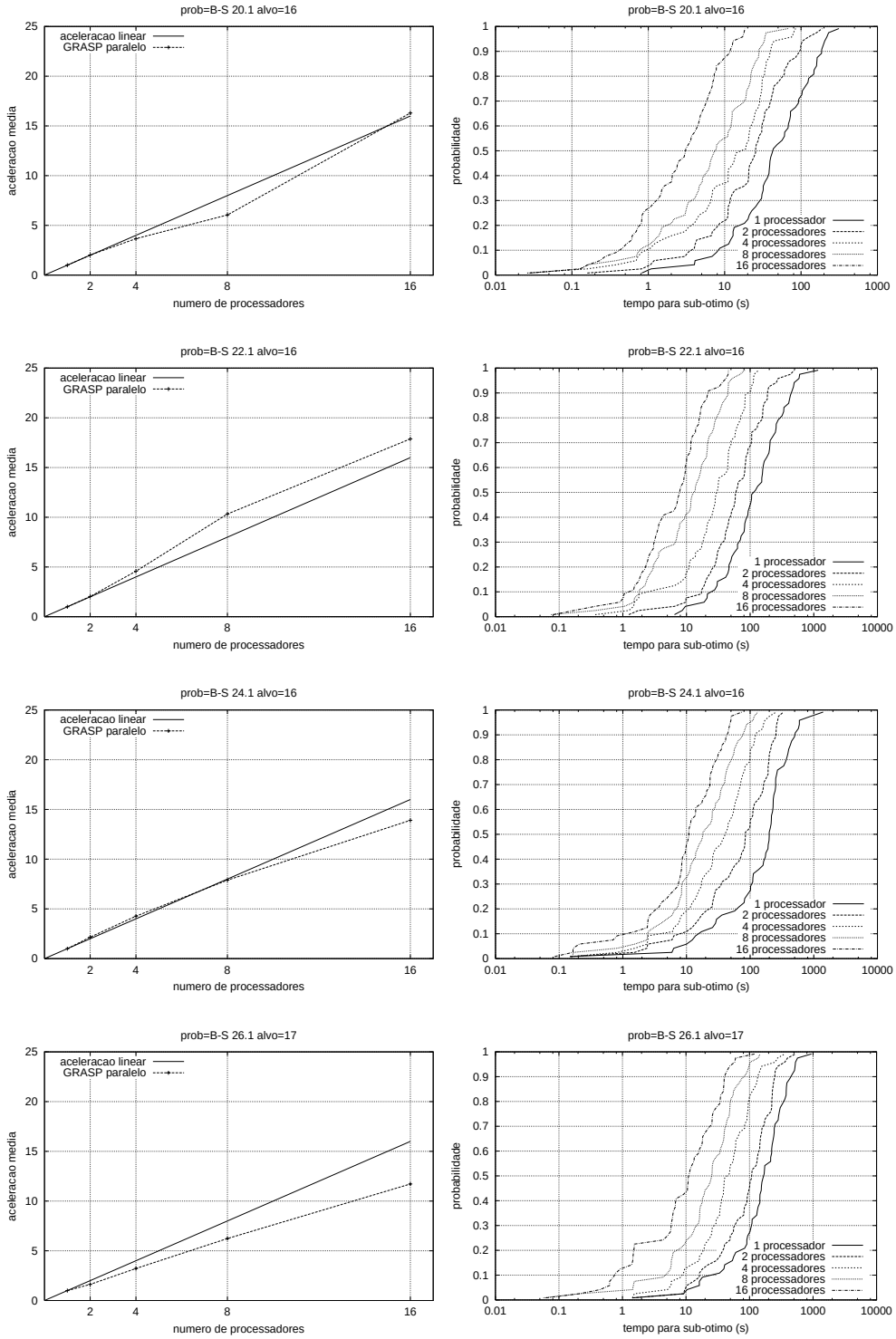

Figura 6.5: Acelerações e distribuições empíricas para a implementação paralela de GRASP para o AP3, problemas B-S 20.1, B-S 22.1, B-S 24.1 e B-S 26.1.

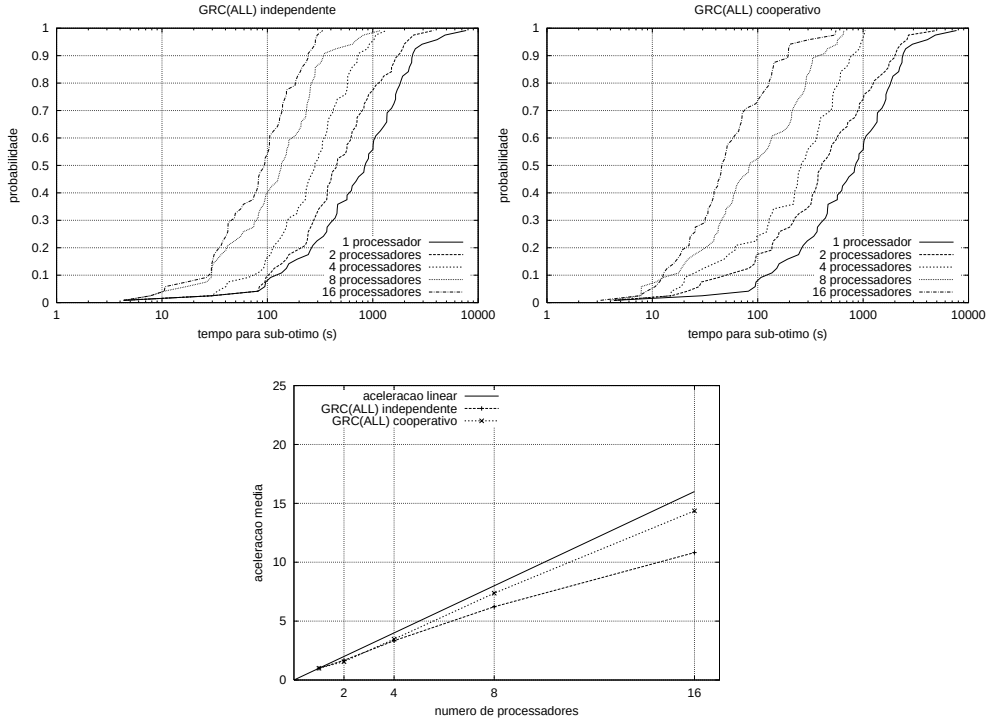

Figura 6.6: Acelerações e distribuições empíricas para as implementações paralelas de `GRC(ALL)` para o AP3, problema `B-S 20.1` com valor alvo 7.

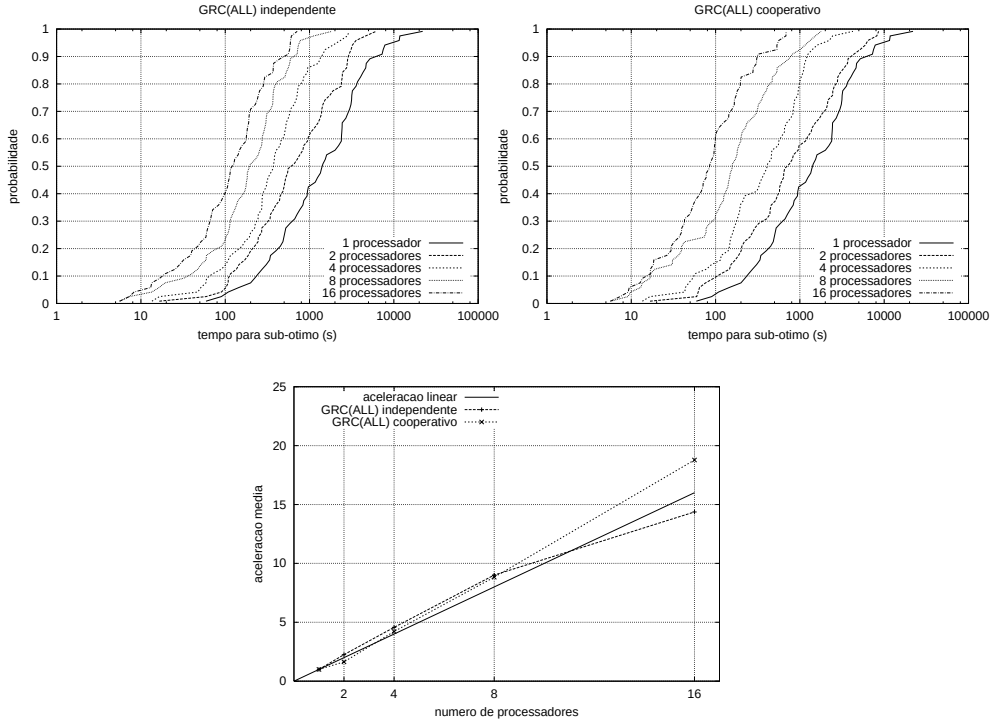

Figura 6.7: Acelerações e distribuições empíricas para as implementações paralelas de `GRC(ALL)` para o AP3, problema `B-S 22.1` com valor alvo 8.

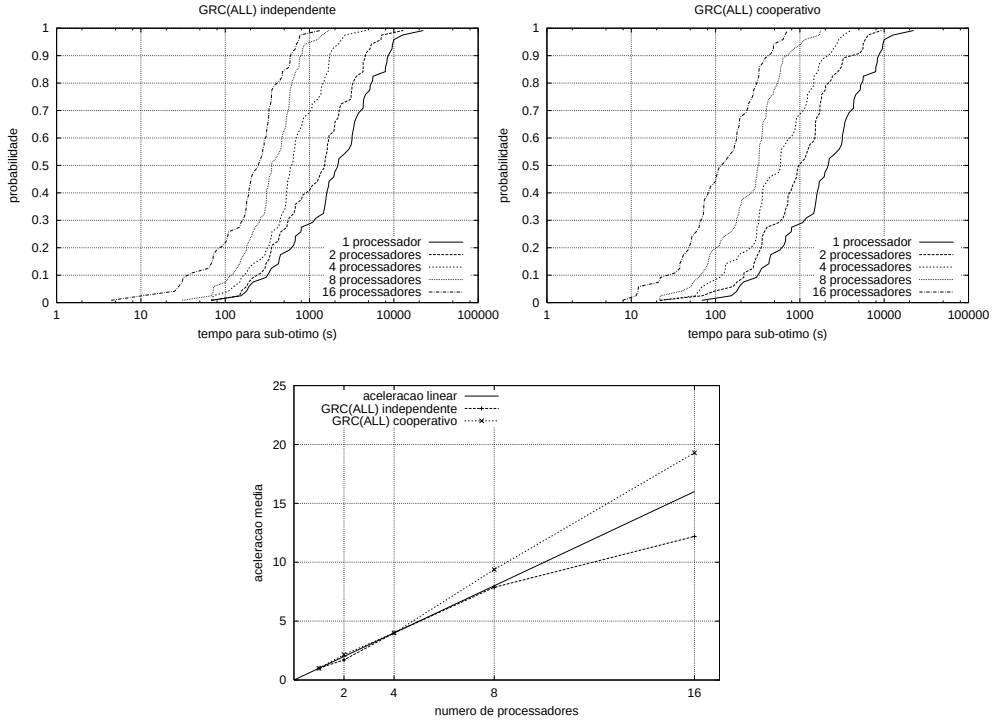

Figura 6.8: Acelerações e distribuições empíricas para as implementações paralelas de `GRC(ALL)` para o AP3, problema `B-S 24.1` com valor alvo 7.

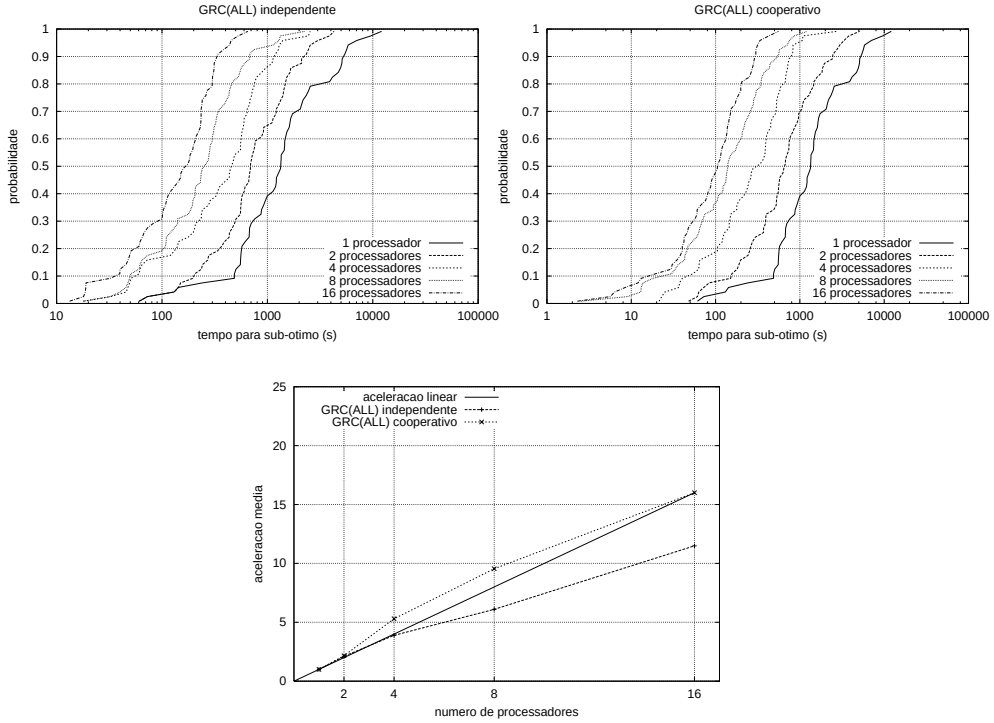

Figura 6.9: Acelerações e distribuições empíricas para as implementações paralelas de `GRC(ALL)` para o AP3, problema `B-S 26.1` com valor alvo 8.

Tabela 6.5: Aceleração e eficiência (aceleração dividida pelo número de processos) para problemas testes do JSP. A implementação testada foi a estratégia de paralelização para GP. Os problemas são abz6, mt10, orb5 e la21, com valores alvos 960, 960, 920 e 1120, respectivamente. São mostrados os valores dos parâmetros da distribuição exponencial de dois parâmetros para cada par problema teste/valor alvo.

	parâmetros da exponencial			número de processadores							
				2		4		8		16	
problema	μ	λ	$\lvert\mu\rvert/\lambda$	acel.	efic.	acel.	efic.	acel.	efic.	acel.	efic.
abz6	0.42	15.56	0.027	2.04	1.02	4.75	1.18	8.87	1.10	19.17	1.19
mt10	11.72	885.03	0.013	1.62	0.81	4.07	1.01	7.34	0.91	14.81	0.92
orb5	1.24	38.27	0.032	2.12	1.06	3.97	0.99	7.63	0.95	14.10	0.88
la21	-1.01	206.83	0.004	1.94	0.97	4.98	1.24	8.13	1.01	19.63	1.22
média			**0.019**	**1.93**	**0.96**	**4.44**	**1.10**	**7.99**	**0.99**	**16.92**	**1.05**

A probabilidade estimada para as abordagens paralelas independente e cooperativa de GRC(ALL) encontrarem uma solução cujo custo é pelo menos tão bom quanto o valor alvo antes de um tempo especificado é mostrada em função do número de processadores na Tabela 6.4. Por exemplo, a tabela mostra que a probabilidade de encontrar uma solução com custo no máximo igual a 7 em no máximo 100 segundos para o problema B-S 20.1 cresce de 16% com quatro processadores, para 40% com oito processadores e para 54% com 16 processadores, para a abordagem paralela independente. Para a abordagem paralela cooperativa, esses valores aumentam para 24%, 52% e 74%, para quatro, oito e 16 processadores, respectivamente.

Resultados dos experimentos paralelos para o JSP

A paralelização de GP foi estudada para os problemas abz6, mt10, orb5 e la21, com valores alvos 960, 960, 920 e 1120, respectivamente. As implementações paralelas independente e cooperativa de GP+PR também foram testadas para os problemas testes abz6, mt10, orb5 e la21, porém foram usados valores alvos mais difíceis de obter: 943, 938, 895 e 1100, respectivamente. A Figura 6.10 mostra as acelerações e as distribuições empíricas para a implementação paralela de GP. Analogamente, as Figuras 6.11, 6.12, 6.13 e 6.14 mostram as acelerações e as distribuições empíricas para as implementações paralelas independente e cooperativa de GP+PR. Esses gráficos foram gerados a partir de 60 execuções independentes para cada número de processadores considerado (um, dois, quatro, oito e 16 processadores). As distribuições empíricas são plotadas da mesma forma como é feito para o AP3 na subseção anterior.

Tabela 6.6: Estimativas da probabilidade de encontrar uma solução pelo menos tão boa quanto a solução alvo em um dado tempo de execução, em função do número de processadores, para problemas testes do JSP. A implementação testada foi a estratégia de paralelização para `GP`. Os problemas são `abz6`, `mt10`, `orb5` e `la21`, com valores alvos 960, 960, 920 e 1120, respectivamente.

		probabilidade GRASP paralelo				
	tempo	número de processadores				
problema	(s)	1	2	4	8	16
`abz6`	10	0.34	0.67	0.93	1.00	1.00
	20	0.61	0.90	0.98	1.00	1.00
	50	0.90	0.98	1.00	1.00	1.00
`mt10`	10	0.04	0.02	0.04	0.07	0.19
	100	0.16	0.20	0.45	0.64	0.82
	500	0.46	0.60	0.92	1.00	1.00
`orb5`	10	0.20	0.37	0.62	0.87	0.99
	20	0.36	0.62	0.84	0.96	1.00
	50	0.70	0.94	1.00	1.00	1.00
`la21`	10	0.04	0.08	0.18	0.30	0.54
	100	0.29	0.57	0.87	0.96	1.00
	500	0.89	0.97	1.00	1.00	1.00

Na Tabela 6.5 são resumidas as acelerações mostradas nos gráficos da Figura 6.10. A tabela mostra também os valores da eficiência, dos parâmetros μ, λ e de $|\mu|/\lambda$ para cada par problema teste/valor alvo testado. Esses parâmetros foram estimados na Subseção 5.6.4 a partir de 200 execuções independentes do `GP` seqüencial.

As acelerações médias observadas (calculadas como explicado na Subseção 6.4.1) são aproximadamente lineares. De fato, os valores de $|\mu|/\lambda$ observados na Tabela 6.5 são baixos e sua média é 0.019. Portanto, acelerações aproximadamente lineares eram esperadas para o `GP` paralelo.

A Tabela 6.6 mostra, para alguns tempos de execução, a probabilidade estimada para se obter uma solução de custo pelo menos tão bom quanto o valor alvo, em função do número de processadores. A tabela mostra que, por exemplo, a probabilidade de obter uma solução com valor no máximo igual a 920 para o problema `orb5` em no máximo 10 segundos cresce de 20% com um processador, para 62% com quatro processadores e para 99% com 16 processadores.

A Tabela 6.7 resume as acelerações mostradas nos gráficos das abordagens paralelas independente e cooperativa de `GP+PR`. Também são mostrados os valores de μ, λ e $|\mu|/\lambda$, estimados na Subseção 5.6.4, a partir de 200 execuções independentes do `GP+PR` seqüencial. São observadas acelerações sublineares para a abordagem paralela independente de `GP+PR`. Observa-se que as razões $|\mu|/\lambda$ calculadas para os parâmetros na Tabela 6.7 são muito

Tabela 6.7: Aceleração com dois, quatro, oito e 16 processadores para problemas testes do JSP. As implementações testadas foram as estratégias de paralelização independente e cooperativa para `GP+PR`. Os problemas são `abz6`, `mt10`, `orb5` e `la21`, com valores alvos 943, 938, 895 e 1100, respectivamente. São mostrados os valores dos parâmetros da distribuição exponencial de dois parâmetros para cada par problema teste/valor alvo.

	parâmetros da exponencial			aceleração independente				aceleração cooperativo			
				número de processadores				número de processadores			
problema	μ	λ	$\lvert\mu\rvert/\lambda$	2	4	8	16	2	4	8	16
`abz6`	47.67	756.56	0.06	2.00	3.36	6.44	10.51	2.40	4.21	11.43	23.58
`mt10`	305.27	524.23	0.58	1.57	2.12	3.03	4.05	1.75	4.58	8.36	16.97
`orb5`	130.12	395.41	0.32	1.95	2.97	3.99	5.36	2.10	4.91	8.89	15.76
`la21`	175.20	407.73	0.42	1.64	2.25	3.14	3.72	2.23	4.47	7.54	11.41
média			**0.34**	**1.79**	**2.67**	**4.15**	**5.91**	**2.12**	**4.54**	**9.05**	**16.93**

Tabela 6.8: Estimativas da probabilidade de encontrar uma solução pelo menos tão boa quanto a solução alvo em um dado tempo de execução, em função do número de processadores, para problemas testes do JSP. As implementações testadas foram as estratégias de paralelização independente e cooperativa para `GP+PR`. Os problemas são `abz6`, `mt10`, `orb5` e `la21`, com valores alvos 943, 938, 895 e 1100, respectivamente.

		probabilidade independente					probabilidade cooperativo				
	tempo	número de processadores					número de processadores				
problema	(s)	1	2	4	8	16	1	2	4	8	16
`abz6`	100	0.01	0.03	0.12	0.25	0.47	0.01	0.12	0.24	0.64	0.94
	500	0.30	0.59	0.79	0.95	1.00	0.30	0.61	0.79	1.00	1.00
	1000	0.57	0.85	0.97	0.99	1.00	0.57	0.92	0.98	1.00	1.00
`mt10`	100	0.00	0.00	0.00	0.02	0.05	0.00	0.00	0.05	0.34	0.82
	500	0.17	0.32	0.55	0.82	0.98	0.17	0.49	0.95	1.00	1.00
	1000	0.54	0.79	0.95	1.00	1.00	0.54	0.85	1.00	1.00	1.00
`orb5`	100	0.00	0.00	0.02	0.03	0.19	0.00	0.07	0.42	0.74	0.92
	500	0.35	0.75	0.93	1.00	1.00	0.35	0.80	0.97	1.00	1.00
	1000	0.75	0.97	1.00	1.00	1.00	0.75	0.95	1.00	1.00	1.00
`la21`	100	0.00	0.00	0.00	0.02	0.06	0.00	0.02	0.08	0.44	0.87
	500	0.29	0.52	0.82	0.98	1.00	0.29	0.79	1.00	1.00	1.00
	1000	0.75	0.98	1.00	1.00	1.00	0.75	0.98	1.00	1.00	1.00

maiores do que os valores de $|\mu|/\lambda$ dos parâmetros estimados para os pares problemas testes/valores alvos usados para estudar GP (mostrados na Tabela 6.5), assim como para os pares problemas testes/valores alvos usados para estudar o AP3 na subseção anterior. Conforme visto no Capítulo 5, embora GP+PR atinja o valor alvo mais rapidamente do que GP, as suas iterações precisam de mais tempo de processamento, o que corresponde a valores mais altos de μ. Além disso, o religamento de caminhos reduz o tempo total de execução (com critério de parada por valor alvo), reduzindo o espalhamento dos tempos de solução, dados pelo parâmetro λ. Portanto, para GP+PR os valores de μ são maiores e os valores de λ são menores em relação aos parâmetros de GP. Por essas razões, as distribuições plotadas para GP+PR não podem ser aproximadas por uma distribuição exponencial simples. Como observado na Proposição 3.2, verifica-se para o GP+PR paralelo independente que, à medida que o número de processadores ρ aumenta, a aceleração do algoritmo se deteriora.

Na Figura 6.15 são mostradas as distribuições empíricas para as implementações paralelas de GP, GP+RC independente e GP+RC cooperativo usando 16 processadores. Os programas foram testados para os problemas abz6, mt10, orb5 e la21, com valores alvos 943, 938, 895 e 1100, respectivamente. Observa-se que para 16 processadores, apesar do GP paralelo apresentar maiores valores de aceleração do que os obtidos por GP+RC independente, o tempo decorrido de GP+RC independente é em geral inferior ao tempo decorrido de GP paralelo. Observa-se também que o tempo decorrido de GP+RC cooperativo é bastante inferior aos tempos decorridos de GP paralelo e de GP+RC independente.

Na Tabela 6.7 são observadas acelerações lineares e superlineares para a maior parte dos problemas testes estudados para a abordagem cooperativa. Essas acelerações são muito maiores do que as acelerações observadas para a abordagem paralela independente. Por exemplo, para 16 processadores, a aceleração da abordagem paralela cooperativa é quase três vezes maior do que a aceleração observada para a abordagem paralela independente. No GP+PR, o religamento de caminhos não é realizado a cada iteração, apenas soluções de qualidade produzidas pela busca local serão religadas com soluções do *pool* e irão, eventualmente, alterar o conteúdo do *pool* de soluções. Dessa forma, as soluções recebidas através da troca de mensagens na abordagem paralela cooperativa de GP+PR tornam-se essenciais para renovar e melhorar o custo das soluções armazenadas no *pool* de cada processo. Deve-se lembrar também que, no caso de GP+PR, uma solução produzida pela busca local somente participará do religamento

de caminhos caso ela seja considerada satisfatória segundo um critério de qualidade. Conforme explicado nas Seções 5.3 e 5.6.3, nas primeiras 10000 iterações do `GP+PR`, o religamento de caminhos será realizado entre a solução S produzida pelo algoritmo GRASP e um subconjunto de soluções do *pool*, caso o custo de S seja melhor ou igual ao custo da pior solução armazenada no *pool*. Portanto, a troca de informações reduz o número de vezes em que o religamento de caminhos é realizado durante as 10000 primeiras iterações do `GP+PR` cooperativo, pois melhora o valor da pior solução armazenada no *pool*. Esse fato, além da melhoria na qualidade das soluções armazenadas no *pool* devido à troca de mensagens, é responsável pelas acelerações superlineares observadas na Tabela 6.7.

Na Tabela 6.8, a probabilidade estimada de encontrar uma solução de custo pelo menos tão bom quanto o valor alvo dentro de um tempo especificado é mostrada em função do número de processadores. Por exemplo, a tabela mostra que, para o problema `mt10`, a probabilidade de encontrar uma solução com custo no máximo 938 em no máximo 500 segundos cresce de 32% com dois processadores, para 55% com quatro processadores, para 82% com oito processadores e para 98% com 16 processadores, para a abordagem independente. Para a abordagem cooperativa, esses valores aumentam para 49%, 95%, 100% e 100%, para dois, quatro, oito e 16 processadores, respectivamente.

Considerações sobre o critério de parada por número máximo de iterações

A análise do comportamento das estratégias paralelas feitas a partir das distribuições do tempo para valor alvo dos algoritmos seqüenciais proposta nesse trabalho pode ser aplicada apenas a estratégias paralelas que usam o critério de parada por valor alvo. Quando um algoritmo GRASP paralelo apresenta uma aceleração aproximadamente linear usando-se o critério de parada por valor alvo, significa que a qualidade da solução obtida em um tempo ρt usando-se um processador, será semelhante à qualidade da solução obtida em tempo t, usando-se ρ processadores.

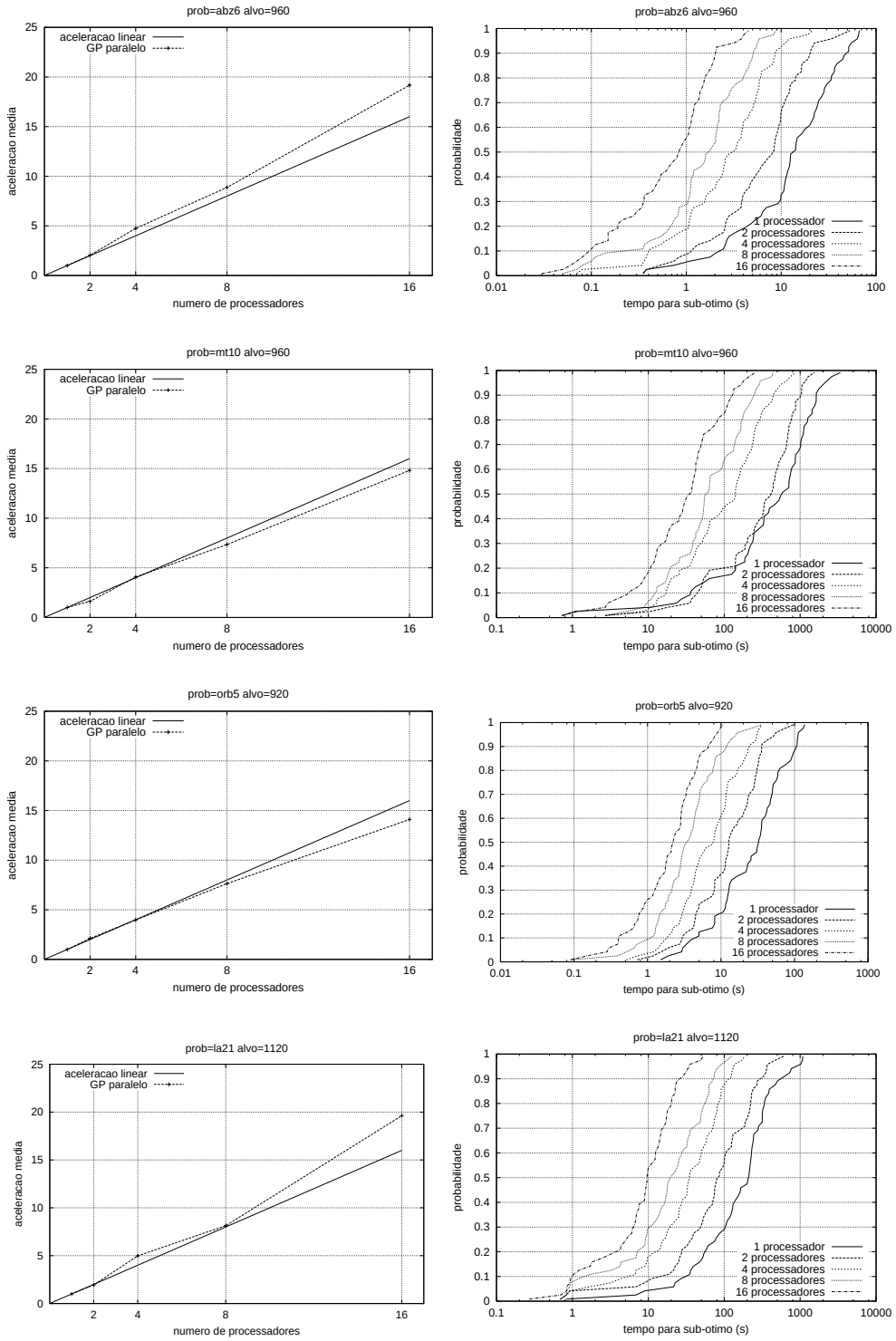

Figura 6.10: Acelerações e distribuições empíricas para a implementação paralela de `GP` para o JSP, problemas `abz6`, `mt10`, `orb5` e `la21`.

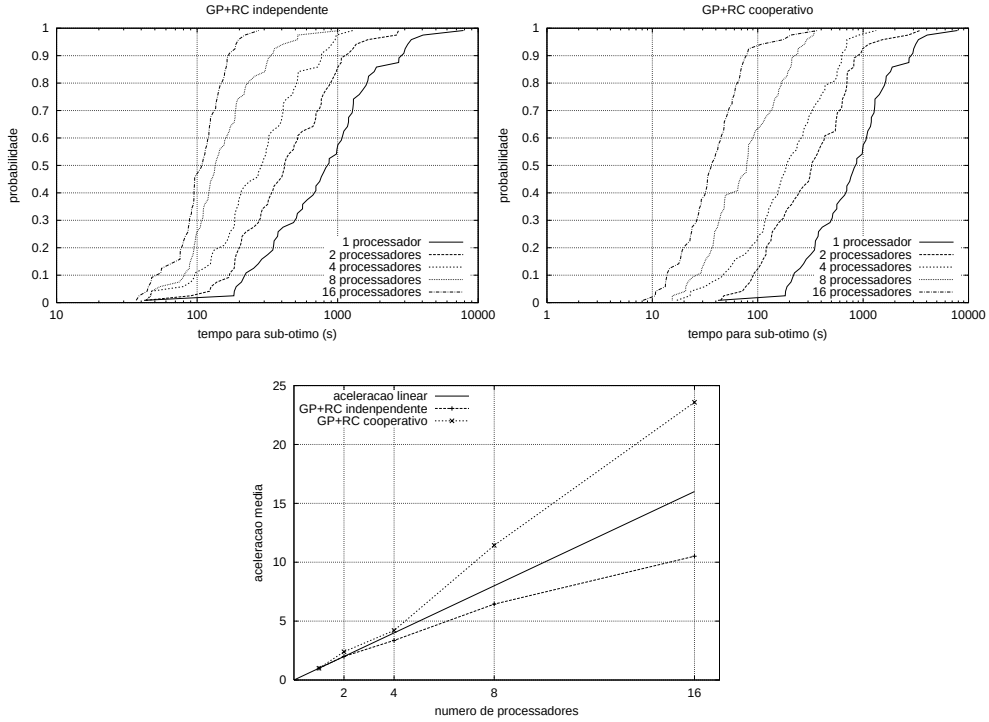

Figura 6.11: Acelerações e distribuições empíricas para as implementações paralelas de `GP+PR` para o JSP, problema `abz6` com valor alvo 943.

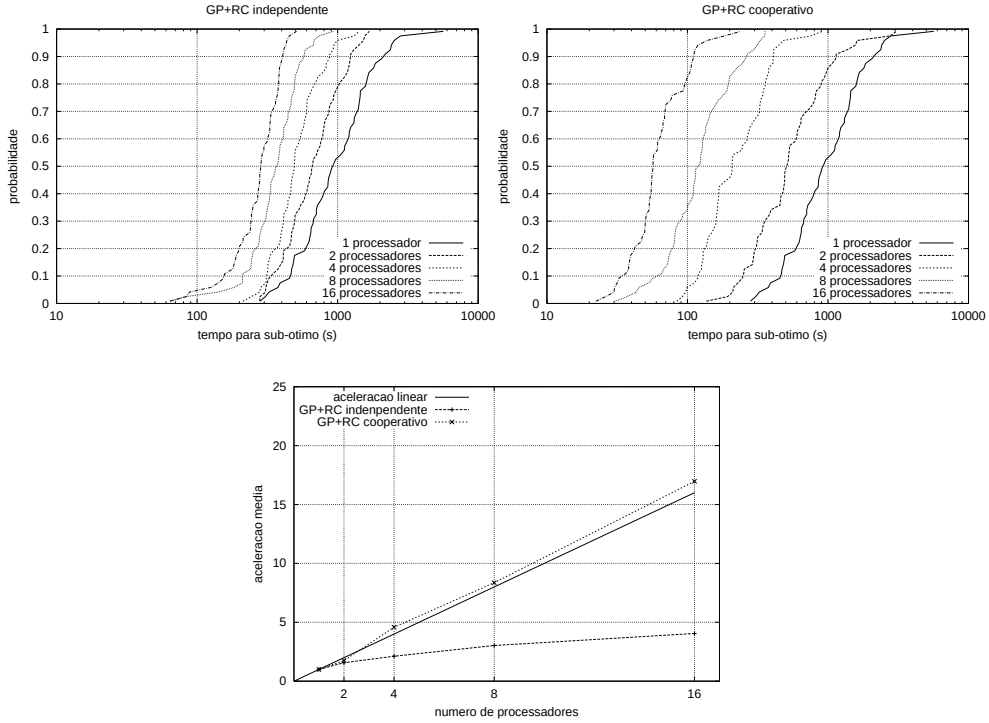

Figura 6.12: Acelerações e distribuições empíricas para as implementações paralelas de `GP+PR` para o JSP, problema `mt10` com valor alvo 938.

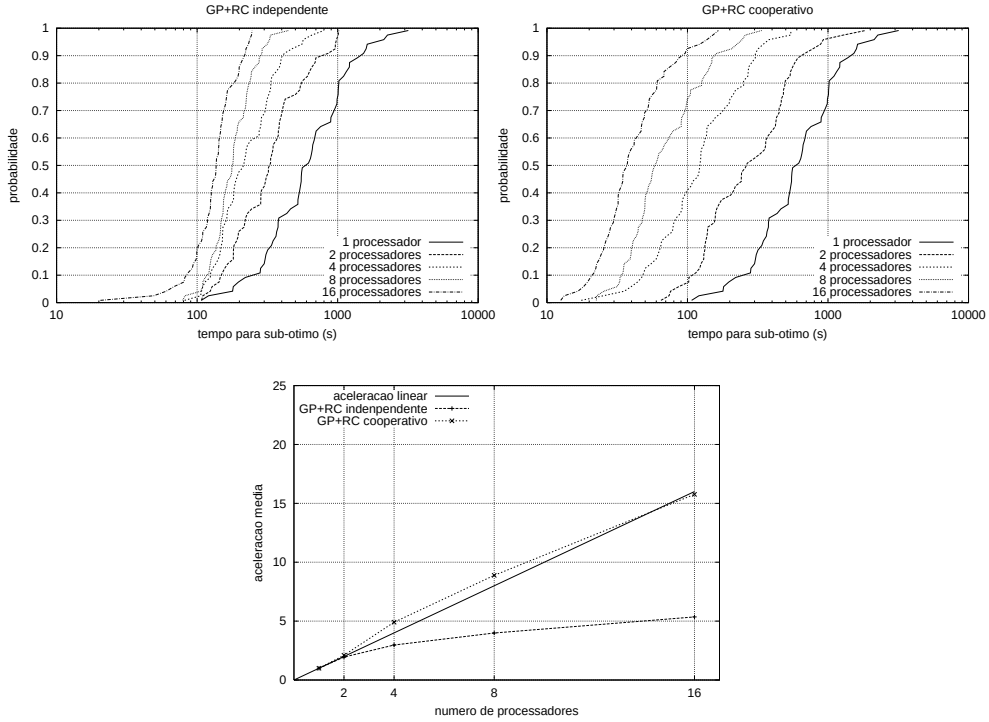

Figura 6.13: Acelerações e distribuições empíricas para as implementações paralelas de `GP+PR` para o JSP, problema `orb5` com valor alvo 895.

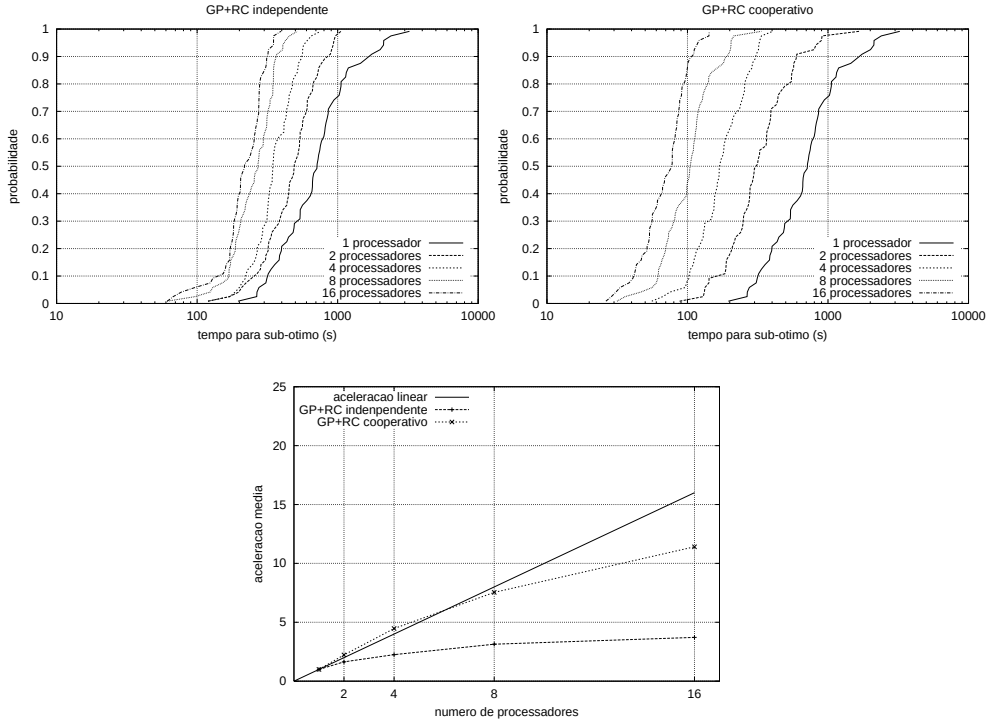

Figura 6.14: Acelerações e distribuições empíricas para as implementações paralelas de `GP+PR` para o JSP, problema `la21` com valor alvo 1100.

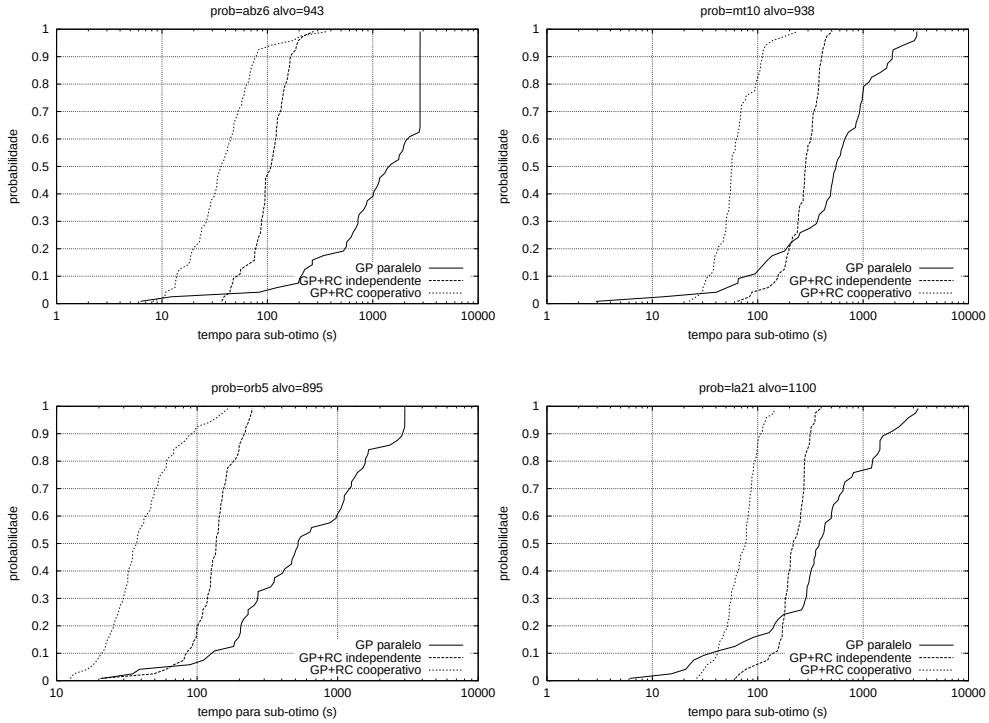

Figura 6.15: Distribuições empíricas para as implementações paralelas de `GP`, `GP+RC` independente e `GP+RC` cooperativo usando 16 processadores. Os problemas testados foram `abz6`, `mt10`, `orb5` e `la21`, com valores alvos 943, 938, 895 e 1100, respectivamente.

Tabela 6.9: Acelerações para a implementação paralela de GRASP para o AP3 para os problemas B-S 20.1, B-S 22.1, B-S 24.1 e B-S 26.1. O critério de parada por número de iterações foi testado para 4096 iterações. O critério de parada por valor alvo foi testado para os valores alvos 20, 21, 21, 22 para os problemas B-S 20.1, B-S 22.1, B-S 24.1 e B-S 26.1, respectivamente.

	parada por valor alvo				parada por total de iterações			
	número de processadores				número de processadores			
problema	2	4	8	16	2	4	8	16
B-S 20.1	1.91	3.48	7.85	16.60	1.98	3.93	8.00	15.07
B-S 22.1	1.72	3.90	7.26	17.38	1.98	3.93	7.76	15.11
B-S 24.1	1.73	4.33	7.68	13.22	1.98	3.93	7.75	15.12
B-S 26.1	1.93	4.41	7.59	15.42	1.98	3.93	7.76	15.12
médias	**1.82**	**4.03**	**7.59**	**15.65**	**1.98**	**3.93**	**7.81**	**15.10**

Tabela 6.10: Qualidade da solução para execuções paralelas com um, dois, quatro, oito e 16 processadores para problemas testes do AP3. A implementação testada foi o GRASP paralelo, para 4096 iterações. Os problemas testes são B-S 20.1, B-S 22.1, B-S 24.1 e B-S 26.1.

	média dos custos para GRASP				
	número de processadores				
problema	1	2	4	8	16
B-S 20.1	19.63	19.71	19.80	19.35	20.05
B-S 22.1	20.85	20.55	20.68	20.86	20.43
B-S 24.1	20.30	20.23	20.46	21.41	20.48
B-S 26.1	21.20	21.33	20.71	20.90	20.51

Pode-se estabelecer a seguinte relação entre os dois critérios de parada usados nas implementações descritas nesse capítulo. Seja um método GRASP paralelo em que as acelerações médias medidas para o critério de parada por valor alvo é aproximadamente linear. Caso para esse método GRASP paralelo, as acelerações médias medidas para o critério de parada por número de iterações também seja aproximadamente linear, então a qualidade média das soluções será preservada durante essas execuções paralelas.

Para ilustrar essa propriedade, seja G um método GRASP paralelo em que as acelerações médias medidas usando-se o critério de parada por valor alvo para um dado par de problema teste p e valor alvo q seja linear. Dessa forma, uma solução com custo pelo menos tão bom quanto q será obtida por G em tempo médio ρt usando-se um processador, e em tempo médio t usando-se ρ processadores. Seja i o número médio de iterações que são executadas durante o intervalo de tempo ρt em um processador. Caso as acelerações médias de G para o critério de parada por número de iterações (para i iterações) também seja aproximadamente linear, então o seu tempo de execução paralela em ρ processadores será em média t. Portanto, as execuções que usam o critério de parada por valor alvo e as execuções que usam o critério de parada por número de iterações serão aproximadamente

equivalentes, visto que, em ambos os casos, G será executado em um processador em média em tempo ρt e em ρ processadores em média em tempo t. Logo, a qualidade média das soluções será preservada durante as execuções paralelas, e será aproximadamente q.

A propriedade acima é exemplificada para o algoritmo GRASP puro implementado para o AP3. Os problemas testados foram `B-S 20.1`, `B-S 22.1`, `B-S 24.1` e `B-S 26.1`. A Tabela 6.9 mostra as acelerações obtidas para execuções paralelas para dois, quatro, oito e 16 processadores, usando-se os critérios de parada por valor alvo e por número de iterações. Para cada número de processadores e critério de parada considerados, foram feitas 60 execuções do algoritmo GRASP paralelo. Os valores alvos usados nas execuções que usam o critério de parada por valor alvo foram 20, 21, 21 e 22, para os problemas `B-S 20.1`, `B-S 22.1`, `B-S 24.1` e `B-S 26.1`, respectivamente. As execuções feitas para testar o critério de parada por número de iterações fazem um total de 4096 iterações. Verifica-se que a implementação paralela de `GRASP` alcançou uma aceleração aproximadamente linear para todos os problemas testes estudados para os dois critérios de parada.

Na Tabela 6.10 são mostrados os custos médios das soluções obtidas para um, dois, quatro, oito e 16 processadores para as execuções que usam o critério de parada por número de iterações. Nas execuções de um algoritmo GRASP puro que usa o critério de parada por número de iterações, caso a mesma seqüência de números aleatórios usada em uma execução com um processador também seja usada em execuções com dois, quatro, oito e 16 processadores (dividindo-se a seqüência em dois, quatro, oito e 16 segmentos, respectivamente) as soluções produzidas por essas execuções serão idênticas, visto que o mesmo número de iterações estará sendo realizado para os mesmos números aleatórios. Para evitar que isso ocorra, seqüências disjuntas são usadas para cada número de processadores utilizado. Observa-se que são obtidos custos médios muito semelhantes para as execuções paralelas efetuadas. Além disso, esses custos médios são aproximadamente iguais (sendo na maior parte dos casos, inferiores) aos valores alvos usados nas execuções com o critério de parada por valor alvo.

6.5 Conclusão

Nesse capítulo as estratégias de GRASP com religamento de caminhos propostas nos Capítulos 4 e 5 foram generalizadas e paralelizadas. Foram

propostas duas abordagens para paralelização: uma independente e outra cooperativa. Na abordagem independente, a única comunicação feita entre os processos durante as iterações do algoritmo tem como objetivo a detecção do término do programa. Na abordagem cooperativa, além de fazer a comunicação realizada em uma abordagem independente, um processo compartilha com os demais soluções de qualidade armazenadas no seu *pool* local. Essas soluções compartilhadas são usadas pelo religamento de caminhos em cada processo, para melhorar a qualidade das soluções produzidas.

Observou-se que as acelerações da abordagem paralela cooperativa são superiores às acelerações da abordagem independente na maior parte dos casos testados. Ou seja, o tempo para valor alvo da estratégia cooperativa é em média inferior ao tempo para valor alvo da estratégia independente. Portanto, para um tempo de execução fixo, a qualidade das soluções obtidas pela abordagem cooperativa será em média superior à das soluções obtidas pela abordagem independente.

Os parâmetros estimados λ e μ das distribuições exponenciais de dois parâmetros para os pares problema teste/valor alvo testados são usados para explicar as acelerações obtidas pelas abordagens paralelas que usam múltiplos processos independentes. Observou-se que, de acordo com o estudo feito no Capítulo 3, as implementações de GRASP com valores pequenos de $|\mu|/\lambda$, apresentam uma aceleração próxima da linear quando paralelizadas. Deve-se notar que os valores de $|\mu|/\lambda$ não devem ser usados para explicar de forma quantitativa as acelerações medidas, visto que esses parâmetros são estimativas calculadas a partir de dados reais. Alternativamente, esses parâmetros podem ser considerados como ferramentas úteis para prever, de forma qualitativa, o comportamento de uma estratégia GRASP paralela, a partir da sua versão seqüencial.

No próximo capítulo será mostrado um esquema de geração de restrições violadas para alocação de custos de sistemas de transmissão. Uma abordagem paralela é proposta para esse esquema. Será mostrado que os tempos para valor alvo de uma etapa desse algoritmo também se ajustam a uma distribuição exponencial de dois parâmetros.