3
Distribuição de probabilidade do tempo para valor alvo

3.1
Introdução

Assim como qualquer heurística para otimização combinatória baseada em partidas múltiplas, um algoritmo GRASP pode ser implementado em paralelo dividindo-se suas k iterações entre ρ processadores. Heurísticas paralelas que seguem essa estratégia são exemplos de abordagens denominadas de *múltiplas trajetórias independentes* de acordo com as taxonomias propostas em [38, 129]. Várias implementações paralelas do algoritmo GRASP que seguem esse esquema têm sido apresentadas na literatura [83, 81, 86, 90, 91]. Na maior parte desses trabalhos, um fato comum é observado. A aceleração nos tempos de execução medidos, isto é, o tempo de execução do problema seqüencial dividido pelo tempo de execução do programa paralelo, é proporcional ao número de processadores usados. Esse comportamento é chamado de aceleração linear. Um exemplo típico pode ser visto em Pardalos, Pitsoulis e Resende [91], onde em uma implementação paralela do método para o problema de satisfabilidade máxima, usando a biblioteca PVM (*Parallel Virtual Machine*), acelerações médias aproximadamente lineares foram observadas (vide Tabela 3.1). Esse comportamento pode ser explicado de forma intuitiva devido ao fato de que, em geral, um determinado número de iterações do algoritmo levará aproximadamente o mesmo tempo para ser executado, independente das sementes do gerador de números aleatórios usadas para iniciá-las, considerando-se uma carga relativamente homogênea nos processadores usados no experimento. Caso as mesmas sementes usadas em cada iteração do programa seqüencial sejam usadas no programa paralelo, os dois programas obterão a mesma solução.

O comportamento de implementações paralelas de GRASP em relação à qualidade da solução produzida durante as execuções paralelas foi estudado em [8]. Para isso, cada processador executa uma cópia do algoritmo GRASP seqüencial. É usado um critério de parada que consiste em forne-

Tabela 3.1: Tempo de processamento (em segundos) e aceleração para problemas MAX-SAT [91]. A média das acelerações é mostrada para 5, 10 e 15 processadores.

	1 processador	5 processadores		10 processadores		15 processadores	
problema	tempo	tempo	acel.	tempo	acel.	tempo	acel.
jnh201	310.4	62.8	4.9	30.5	10.2	22.2	14.0
jnh202	312.2	59.8	5.2	31.2	10.0	23.4	13.3
jnh203	351.2	72.3	4.9	35.2	10.0	23.2	15.1
jnh205	327.8	63.4	5.2	32.1	10.2	22.5	14.6
jnh207	304.7	56.7	5.4	29.6	10.3	19.8	15.4
jnh208	355.2	65.6	5.4	33.2	10.7	21.0	16.9
jnh209	339.0	60.5	5.6	33.6	10.1	21.6	15.7
jnh210	318.5	57.6	5.5	32.5	9.8	20.8	15.3
jnh301	414.5	85.3	4.9	45.2	9.2	28.3	14.6
jnh302	398.7	88.6	4.5	48.2	8.3	27.0	14.7
média das acelerações:			5.2		9.9		15.0

cer um valor alvo τ da função objetivo para todos os processos e executar o algoritmo até que o primeiro processo encontre uma solução com valor pelo menos tão bom quanto τ. Nessa abordagem, não há a necessidade de se utilizar as mesmas sementes nos programas seqüenciais e paralelos para garantir uma relação entre a qualidade das soluções produzidas pelos programas. Basta utilizar o critério de parada por valor alvo nos programas seqüenciais e paralelos para o mesmo valor de τ. Alternativamente, pode-se executar k iterações do programa seqüencial e usar o custo da solução obtida como valor alvo no programa paralelo. O critério de parada por valor alvo permite comparar os tempos de execução de dois programas diferentes, considerando-se que as soluções obtidas não precisam ser necessariamente idênticas, mas sim de mesma qualidade.

Acelerações aproximadamente lineares são observadas para estratégias de paralelização que usam o critério de parada por valor alvo [7, 5]. Isso significa que a qualidade da solução obtida em tempo ρt usando-se um processador será a mesma da solução obtida em tempo t usando-se ρ processadores. Esse fato pode ser explicado caso a variável aleatória *tempo para valor alvo* seja exponencialmente distribuída (deslocada ou não), conforme indicado pela proposição a seguir [129].

Proposição 3.1 *Seja $P_\rho(t)$ a probabilidade de uma solução com um certo valor alvo não ter sido encontrada em t unidades de tempo com ρ processos independentes. Se $P_1(t) = e^{-t/\lambda}$ com $\lambda \in \mathbb{R}^+$, isto é, $P_1(t)$ corresponde a uma distribuição exponencial, então $P_\rho(t) = e^{-\rho t/\lambda}$.*

Prova:

Seja $F_\rho(t)$ a função cumulativa de distribuição que representa a probabilidade de um dado valor alvo ter sido encontrado em até t unidades

de tempo com ρ processos independentes. Analogamente, seja $F_1(t)$ a função cumulativa de distribuição que representa a probabilidade de um dado valor alvo ter sido encontrado em t unidades de tempo com um processo, dada por $F_1(t) = 1 - e^{-t/\lambda}$, com $\lambda \in \mathbb{R}^+$.

Da definição da função cumulativa de distribuição tem-se que:

$$\begin{aligned} F_1(t) &= P(X \leq t) = 1 - P(X > t) \\ F_\rho(t) &= P(Y \leq t) \end{aligned}$$

Por definição $Y = min(X_1, X_2, \cdots, X_\rho)$, onde $(X_1, X_2, \cdots, X_\rho)$ é uma amostragem de tamanho ρ da distribuição $F_1(t)$. Logo:

$$\begin{aligned} F_\rho(t) &= P(X_1 < t \vee X_2 < t \vee \cdots \vee X_\rho < t) \\ &= 1 - P(X_1 > t \wedge X_2 > t \wedge \cdots \wedge X_\rho > t) \\ &= 1 - \prod_{i=1}^{\rho} P(X_i > t) \end{aligned} \tag{3-1}$$

Calculando-se cada termo do produtório em 3-1, tem-se que:

$$\begin{aligned} P(X_i > t) &= 1 - F_1(t) \\ &= 1 - (1 - e^{-t/\lambda}) \\ &= e^{-t/\lambda} \end{aligned} \tag{3-2}$$

Substituindo 3-2 em 3-1:

$$F_\rho(t) = 1 - e^{-\rho t/\lambda} = F_1(\rho t)$$

Portanto, $P_\rho(t) = e^{-\rho t/\lambda} = P_1(\rho t)$. □

A proposição acima vem da definição da distribuição exponencial. Ela implica em que a probabilidade de se encontrar uma solução com um determinado valor alvo em tempo ρt, usando-se um processo seqüencial, é igual à probabilidade de se encontrar uma solução com custo pelo menos tão bom quanto o valor alvo em tempo t, usando-se ρ processos paralelos independentes. Portanto, é possível atingir uma aceleração linear no tempo de obtenção do valor alvo, usando-se vários processos independentes.

Uma proposição análoga pode ser definida para a distribuição exponencial de dois parâmetros (ou distribuição exponencial deslocada).

Proposição 3.2 *Seja $P_\rho(t)$ a probabilidade de um dado valor alvo não ter sido encontrado em t unidades de tempo com ρ processos independentes. Se*

$P_1(t) = e^{-(t-\mu)/\lambda}$ com $\lambda \in \mathbb{R}^+$ e $\mu \in \mathbb{R}$, isto é, $P_1(t)$ corresponde a uma distribuição exponencial de dois parâmetros, então $P_\rho(t) = e^{-\rho(t-\mu)/\lambda}$.

Prova:

Seja $F_\rho(t)$ a função cumulativa de distribuição que representa a probabilidade de um dado valor alvo ter sido encontrado em até t unidades de tempo com ρ processos independentes. Analogamente, seja $F_1(t)$ a função cumulativa de distribuição que representa a probabilidade de um dado valor alvo ter sido encontrado em t unidades de tempo com um processo, dada por $F_1(t) = 1 - e^{-(t-\mu)/\lambda}$, com $\lambda \in \mathbb{R}^+$ e $\mu \in \mathbb{R}$.

Da definição da função cumulativa de distribuição tem-se que:

$$\begin{aligned} F_1(t) &= P(X \leq t) = 1 - P(X > t) \\ F_\rho(t) &= P(Y \leq t) \end{aligned}$$

Por definição $Y = min(X_1, X_2, \cdots, X_\rho)$, onde $(X_1, X_2, \cdots, X_\rho)$ é uma amostragem de tamanho ρ da distribuição $F_1(t)$. Logo:

$$\begin{aligned} F_\rho(t) &= P(X_1 < t \vee X_2 < t \vee \cdots \vee X_\rho < t) \\ &= 1 - P(X_1 > t \wedge X_2 > t \wedge \cdots \wedge X_\rho > t) \\ &= 1 - \prod_{i=1}^{\rho} P(X_i > t) \end{aligned} \tag{3-3}$$

Calculando-se cada termo do produtório em 3-3, tem-se que:

$$\begin{aligned} P(X_i > t) &= 1 - F_1(t) \\ &= 1 - (1 - e^{-(t-\mu)/\lambda}) \\ &= e^{-(t-\mu)/\lambda} \end{aligned} \tag{3-4}$$

Substituindo 3-4 em 3-3:

$$F_\rho(t) = 1 - e^{-\rho(t-\mu)/\lambda}$$

Portanto, $P_\rho(t) = e^{-\rho(t-\mu)/\lambda}$ e $P_1(\rho t) = e^{-(\rho t-\mu)/\lambda}$. □

Analogamente, essa proposição vem da definição da distribuição exponencial de dois parâmetros. Ela implica em que a probabilidade de se encontrar uma solução com um determinado valor alvo em tempo ρt usando-se um processo seqüencial é $1 - e^{-(\rho t-\mu)/\lambda}$, enquanto que a probabilidade de se encontrar uma solução com custo pelo menos tão bom quanto o valor alvo em tempo t usando-se ρ processos paralelos independentes é $1 - e^{-\rho(t-\mu)/\lambda}$.

Deve-se notar que para $\mu = 0$ as probabilidades são iguais e correspondem à distribuição exponencial não deslocada. Além disso, quando $\rho|\mu| \ll \lambda$, as duas probabilidades serão aproximadamente iguais e será possível atingir uma aceleração no tempo de obtenção de solução alvo próxima à linear, usando-se vários processos independentes. Esse comportamento tem sido observado em implementações de várias metaheurísticas. Entre elas, pode-se citar: *simulated annealing* [41, 89]; algoritmos iterativos de busca local para o problema do caixeiro viajante [42], onde mostra-se que a probabilidade de se encontrar uma solução sub-ótima é dada por uma distribuição exponencial deslocada, descontando-se o tempo para encontrar o primeiro ótimo local; busca tabu, desde que a busca seja iniciada a partir de um ótimo local [14, 125]; e WalkSAT [123] (um algoritmo de busca local para problemas de satisfabilidade), para obter soluções para problemas 3-SAT [65] difíceis, gerados aleatoriamente.

O objetivo desse capítulo é verificar se a metaheurística GRASP também possui essa propriedade, isto é, se apresenta uma distribuição exponencial da variável tempo para valor alvo. Para isso, foram consideradas cinco heurísticas GRASP que foram publicadas na literatura cujos códigos fontes estavam disponíveis:

1. conjunto máximo independente [44, 110];
2. problema quadrático de atribuição [76, 106];
3. planarização de grafos [108, 116];
4. satisfabilidade valorada máxima (MAX-SAT) [109, 112];
5. recobrimento máximo [111].

Para cada heurística testada, foram selecionados quatro problemas testes da literatura. Para cada um desses problemas, foram determinados três valores alvos para a função objetivo: um valor facilmente obtido pelo algoritmo, um valor de dificuldade intermediária e um valor de difícil obtenção. Para cada par problema teste/valor alvo, mediu-se o tempo de execução necessário para se obter uma solução com custo pelo menos tão bom quanto o valor alvo e analisou-se a distribuição desses tempos.

Na Seção 3.2 cada um dos cinco algoritmos GRASP analisados será brevemente descrito. O projeto dos experimentos é descrito na Seção 3.3. Na Seção 3.4 são mostrados os resultados dos experimentos.

Tabela 3.2: Problemas testes para o problema de conjunto máximo independente, com o valor da melhor solução conhecida (msc), valores alvos e parâmetros estimados.

			parâmetros estimados	
problema	msc	alvo	$\hat{\mu}$	$\hat{\lambda}$
270002	15	13	0.015	0.146
		14	-0.018	1.612
		15	25.560	291.715
270003	15	13	0.068	0.147
		14	0.142	5.797
		15	1.849	223.491
270004	15	13	0.034	0.092
		14	-0.035	2.024
		15	1.339	30.248
270006	15	13	0.021	0.137
		14	-0.013	1.383
		15	38.909	516.965

3.2 As implementações analisadas

Nessa seção os cinco algoritmos GRASP usados nos experimentos serão brevemente descritos. Para cada um deles, serão apresentados o problema resolvido e as fases construtiva e de busca local do algoritmo.

3.2.1 Heurística GRASP para o problema de conjunto máximo independente

Um método GRASP para o problema de conjunto máximo independente foi apresentado por Feo, Resende e Smith [44]. As subrotinas Fortran da implementação desse algoritmo podem ser encontradas em [110].

Definição do problema

Seja $G = (V, E)$ um grafo não direcionado com conjunto de vértices V e conjunto de arestas E. Os vértices $u, v \in V$ são *não-adjacentes* se $(u, v) \notin E$. Um subconjunto dos vértices $S \subseteq V$ é *independente* se todos seus vértices são não-adjacentes par a par. No *problema de conjunto máximo independente*, tem-se como objetivo encontrar um conjunto independente de cardinalidade máxima.

Fase construtiva

O algoritmo inicializa um grafo de trabalho $\tilde{G} = (\tilde{V}, \tilde{E})$ como sendo o grafo original e o conjunto independente S como sendo o conjunto vazio. O conjunto independente é construído inserindo-se um elemento de cada vez. A função gulosa usada na fase construtiva é o grau do vértice em relação ao grafo de trabalho. O algoritmo guloso seleciona entre os vértices do grafo de trabalho, aquele de menor grau. Um vértice selecionado será retirado do grafo de trabalho juntamente com os vértices adjacentes a ele. A função gulosa é adaptativa, pois é alterada após a seleção de cada vértice.

Sejam $\underline{d}$ e $\overline{d}$ respectivamente o menor e o maior grau entre todos os vértices do grafo de trabalho, isto é, $\underline{d} = \min_{v \in \tilde{V}}\{d(v, \tilde{G})\}$ e $\overline{d} = \max_{v \in \tilde{V}}\{d(v, \tilde{G})\}$, onde $d(v, \tilde{G})$ é o grau do vértice v em relação a $\tilde{G}$. A *lista restrita de candidatos* é o conjunto de vértices

$$\text{LRC} = \{v \in \tilde{V} \mid d(v, \tilde{G}) \leq \underline{d} + \alpha(\overline{d} - \underline{d})\},$$

onde o parâmetro $0 \leq \alpha \leq 1$ controla o tamanho da LRC. A seleção de um vértice na fase construtiva é feita aleatoriamente, restrita aos vértices da LRC.

Fase de busca local

A heurística de busca local para o problema de conjunto máximo independente considera somente movimentos de trocas-(1,2). Na tentativa de aumentar o conjunto independente, um único vértice x do conjunto independente S é removido, sendo substituído por dois vértices não-adjacentes u e v, tais que u e v são não-adjacentes a todos os vértices em $S \setminus \{x\}$. Caso seja possível realizar uma troca como essa, o procedimento é recursivamente aplicado ao novo conjunto independente aumentado. Uma solução localmente ótima é encontrada quando não é mais possível aumentar o conjunto independente através de movimentos de trocas-(1,2).

3.2.2 Heurística GRASP para o problema quadrático de atribuição

Um algoritmo GRASP para o problema quadrático de atribuição (QAP) foi proposto por Li, Pardalos e Resende em [76]. Em [106] as subrotinas em Fortran para esse algoritmo são descritas. Uma especialização

Tabela 3.3: Problemas testes para o problema quadrático de atribuição, com o valor da melhor solução conhecida (msc), valores alvos e parâmetros estimados.

			parâmetros estimados	
problema	msc	alvo	$\hat{\mu}$	$\hat{\lambda}$
chr25a	3796	5023	0.016	0.221
		4721	0.012	1.147
		4418	0.460	8.401
kra30b	91420	94675	0.021	0.076
		93590	0.024	0.261
		92505	-0.041	2.045
sko42	15812	16389	0.051	0.049
		16222	0.044	0.244
		16055	0.173	1.955
tho40	240516	247160	0.105	0.419
		245396	0.255	2.163
		243632	3.397	19.413

para QAPs esparsos, da heurística proposta em [76] é apresentada em [92]. Em [90] é apresentada uma versão paralela para a heurística proposta em [76]. Melhorias feitas nas fases construtiva e de busca local são descritas em [49, 104, 103].

Definição do problema

Dado um conjunto $\mathcal{N} = \{1, 2, \ldots, n\}$ e $n \times n$ matrizes $F = (f_{ij})$ (correspondente aos fluxos entre facilidades) e $D = (d_{kl})$ (correspondente às distâncias entre localidades), o problema de atribuição quadrática (QAP) pode ser apresentado como

$$\min_{p \in \Pi_{\mathcal{N}}} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{p(i)p(j)},$$

onde $\Pi_{\mathcal{N}}$ é o conjunto de todas as permutações de $\mathcal{N}$.

Fase construtiva

Essa fase é formada por duas etapas. Na etapa 1, duas atribuições são produzidas, isto é, uma facilidade i é atribuída a uma localidade k e uma facilidade j é atribuída a uma localidade l. A idéia é atribuir facilidades com muita interação (pares de facilidades que possuam valores altos para f_{ij}) a localidades próximas (pares de localidades com valores baixos para d_{kl}). Para facilitar essa operação, ordena-se as distâncias entre localidades em ordem crescente e os fluxos entre facilidades em ordem decrescente.

Sejam $d_{k_1,l_1} \leq d_{k_2,l_2} \leq \cdots \leq d_{k_p,l_p}$ e $f_{i_1,j_1} \geq f_{i_2,j_2} \geq \cdots \geq f_{i_p,j_p}$ os valores ordenados, onde $p = n^2 - n$. Sejam $d_{k_1,l_1} \cdot f_{i_1,j_1}, d_{k_2,l_2} \cdot f_{i_2,j_2}, \ldots, d_{k_p,l_p} \cdot f_{i_p,j_p}$ os produtos das distâncias multiplicadas pelos fluxos na ordem estabelecida pela ordenação. Sejam $n_\beta = \beta p$ e $0 < \beta \leq 1$ um parâmetro do algoritmo. A ordenação de $p = n^2 - n$ produtos de distâncias por fluxos além de ser ineficiente, não é muito vantajosa para o algoritmo e por isso, apenas os primeiros n_β valores são ordenados em ordem crescente. Entre esses n_β pares de atribuições, um par é selecionado aleatoriamente do conjunto de αn_β atribuições que possuem os menores produtos $d_{kl} \cdot f_{ij}$, onde α é um parâmetro do algoritmo, tal que $0 < \alpha \leq 1$. Esse par selecionado corresponde às atribuições feitas na etapa 1 da fase construtiva.

Na segunda etapa da fase construtiva, as $n-2$ atribuições de facilidades às localidades restantes são efetuadas. Atribuições que possuam baixo custo de interação com o conjunto de atribuições já efetuadas são favorecidas. Seja Γ o conjunto das q atribuições já efetuadas em um dado ponto da fase construtiva, isto é, $\Gamma = \{(i_1,k_1),(i_2,k_2),\ldots,(i_q,k_q)\}$. O custo de atribuir a facilidade j à localidade l, considerando-se as atribuições já efetuadas, é definido como sendo

$$c_{jl} = \sum_{(i,k)\in\Gamma} f_{ij} d_{kl}.$$

Os custos de todos os pares facilidade/localidade (j,l) são ordenados em ordem crescente. Entre os pares com os menores $\alpha \cdot |\Gamma|$ custos, um é selecionado aleatoriamente e incluído no conjunto Γ. Esse procedimento é repetido até que $n-3$ atribuições tenham sido realizadas. Ao final do processo, a facilidade restante é atribuída à última localidade.

Fase de busca local

Na fase de busca local desse algoritmo, uma vizinhança de trocas-(2,2) é analisada a partir da solução construída. Dessa forma, todas as possíveis trocas-(2,2) de pares facilidade/localidade são consideradas. Caso uma troca melhore o custo da solução, a troca é efetuada. O procedimento continua até que não existam mais trocas que melhorem o valor da solução.

3.2.3 Heurística GRASP para planarização de grafos

Um algoritmo GRASP para planarização de grafos foi apresentado por Resende e Ribeiro em [108]. As subrotinas em Fortran deste algoritmo são

Tabela 3.4: Problemas testes para o problema de planarização de grafos, com o valor da melhor solução conhecida (msc), valores alvos e parâmetros estimados.

			parâmetros estimados	
problema	msc	alvo	$\hat{\mu}$	$\hat{\lambda}$
g17	236	222	2.564	8.723
		227	-0.738	72.498
		231	-19.991	763.081
tg100.10	277	215	0.575	0.690
		226	0.747	5.120
		236	-10.540	181.038
rg100.1	162	154	0.135	0.563
		157	0.062	3.983
		159	-0.148	25.954
rg150.1	231	215	0.531	0.722
		220	0.368	6.961
		225	3.495	286.668

descritas em [116].

Definição do problema

Um grafo é dito *planar* caso ele possa ser desenhado em um plano de forma que nenhum par de arcos se cruzem.

Dado um grafo $G = (V, E)$ com conjunto de vértices V e conjunto de arestas E, o objetivo do problema de planarização de grafos é encontrar um subconjunto de cardinalidade mínima de arestas $F \subseteq E$ de tal forma que o grafo $G' = (V, E \setminus F)$, resultante da remoção das arestas F de G, seja planar. Esse problema também é conhecido como problema do *sub-grafo planar máximo*.

Heurística de duas fases

O algoritmo GRASP descrito nessa seção é dividido em duas fases [62]. A primeira fase consiste em estabelecer uma permutação linear dos nós do grafo de entrada e, em seguida, colocá-los dispostos em uma linha reta.

A segunda fase determina dois conjuntos de arestas. Esses dois conjuntos são formados, respectivamente, por arestas que podem ser representadas sem nenhum cruzamento acima nem abaixo da linha em que os nós estão dispostos.

As fases construtiva e de busca local são aplicadas à primeira fase dessa estratégia, onde uma permutação linear dos nós é determinada.

Fase construtiva

O primeiro nó da permutação é selecionado aleatoriamente entre os nós de mais baixo grau em G. Após os k primeiros nós $v_1, v_2, \cdots, v_k$ da permutação terem sido determinados, o próximo nó v_{k+1} a ser incluído é selecionado aleatoriamente entre os nós adjacentes a v_k em G que possuam os menores graus no sub-grafo G_k de G induzido por $V \setminus \{v_1, v_2, \cdots, v_k\}$. Caso não exista nenhum nó de G_k adjacente a v_k em G, então v_{k+1} é selecionado aleatoriamente do conjunto de nós que possuam baixo grau em G_k.

Fase de busca local

A fase de busca local tem como objetivo reduzir o número de arestas que se cruzam na permutação de nós construída. Para isso, a vizinhança da permutação corrente é explorada realizando-se trocas entre as posições ocupadas por nós dois a dois.

Pós-otimização

Nessa etapa, são produzidos três conjuntos de arestas: $\mathcal{A}$ (arestas azuis que podem ser desenhadas acima da linha onde os nós estão dispostos sem se cruzarem), $\mathcal{V}$ (arestas vermelhas, que podem ser desenhadas abaixo da linha onde os nós estão dispostos sem se cruzarem) e $\mathcal{P}$ (arestas pálidas, que são as arestas restantes). Para isso, constrói-se um grafo $H = (E, I)$ onde cada um de seus nós corresponde a uma aresta de G. Os nós e_1 e e_2 de H são conectados por um arco caso as arestas correspondentes em G se cruzem na seqüência estabelecida pelo algoritmo GRASP. Os conjuntos de arestas $\mathcal{A}$, $\mathcal{V}$ e $\mathcal{P}$ são determinados a partir desse grafo. O conjunto $\mathcal{A}$ é construído computando-se o conjunto máximo independente em H. Em seguida, o conjunto $\mathcal{V}$ é obtido calculando-se o conjunto máximo independente no grafo induzido em H por $E \setminus A$. Por construção, $\mathcal{A}$, $\mathcal{V}$ e $\mathcal{P}$ são tais que nenhuma aresta vermelha ou pálida pode ser colorida de azul. Da mesma forma, arestas pálidas não podem ser coloridas de vermelho. Porém, caso exista uma aresta pálida $p \in \mathcal{P}$ tal que todas as arestas azuis que cruzam com p (seja $\hat{A}_p \subseteq \mathcal{A}$ o conjunto formado por esses arcos azuis) não cruzam com nenhuma aresta vermelha $v \in \mathcal{V}$, então todas as arestas azuis $a \in \hat{A}_p$ podem ser coloridas de vermelho e p pode ser colorido com azul. Caso essa nova atribuição de cores seja possível, então o tamanho do sub-grafo planar é

Tabela 3.5: Problemas testes para o problema de satisfabilidade valorada máxima, com o valor da melhor solução conhecida (msc), valores alvos e parâmetros estimados.

			parâmetros estimados	
problema	msc	alvo	$\hat{\mu}$	$\hat{\lambda}$
`jnh11`	420753	418851	0.061	0.136
		419485	0.063	0.876
		420119	0.860	24.903
`jnh12`	420925	417664	0.053	0.046
		418751	0.044	0.233
		419838	0.064	2.797
`jnh212`	394238	393145	0.033	0.707
		393506	-0.226	4.148
		393866	-0.261	46.058
`jnh306`	444838	441720	0.059	0.058
		442759	0.062	0.219
		443798	-0.198	3.509

aumentado de um arco. Esse procedimento de pós-otimização é incorporado ao final de cada iteração do algoritmo GRASP.

3.2.4 Heurística GRASP para o problema MAX-SAT

Um algoritmo GRASP para o problema de satisfabilidade foi inicialmente proposto por Resende e Feo em [107]. Esse algoritmo foi generalizado para obter soluções para problemas de satisfabilidade máxima valorada (MAX-SAT) por Resende, Pitsoulis e Pardalos [109] e suas subrotinas em Fortran são apresentadas em [112].

Definição do problema

Sejam m cláusulas $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m$, envolvendo n variáveis booleanas $x_1, x_2, \ldots, x_n$ que podem receber somente os valores `verdadeiro` ou `falso` (1 ou 0). A cada cláusula $\mathcal{C}_k$ é associado um custo não negativo w_k. Define-se a cláusula $\mathcal{C}_k$ como

$$\mathcal{C}_k = \bigvee_{j=1}^{n_k} l_{kj},$$

onde n_k é o número de literais na cláusula $\mathcal{C}_k$ e o literal $l_{kj} \in \{x_i, \bar{x}_i \mid i = 1, \ldots, n\}$. Uma cláusula é dita *satisfeita* se ela é avaliada como `verdadeira`. No *problema de satisfabilidade valorada máxima* (MAX-SAT), procura-se

determinar uma atribuição de valores booleanos às n variáveis do problema que maximize a soma dos custos das cláusulas satisfeitas.

Fase construtiva

Uma solução viável para o problema MAX-SAT é dada por $x \in \{0,1\}^n$ e $w(x)$ é a soma dos custos das cláusulas satisfeitas por x. Nessa fase, uma solução é construída incorporando-se a ela um elemento a cada passo. Esse processo é guiado por uma função gulosa e por randomização. Como no problema MAX-SAT deve-se atribuir valores booleanos a n variáveis, cada fase construtiva do problema MAX-SAT consiste de n iterações.

A idéia associada à função gulosa é, a cada passo da fase construtiva, maximizar o custo total das cláusulas que tornam-se satisfeitas após a atribuição feita. Seja Γ_i^+ o conjunto de cláusulas que serão satisfeitas caso o valor `verdadeiro` seja atribuído à variável x_i. Da mesma forma, seja Γ_i^- o conjunto de cláusulas que serão satisfeitas caso seja atribuído o valor `falso` à variável x_i. Define-se

$$\gamma_i^+ = \sum_{j \in \Gamma_i^+} w_j \text{ e } \gamma_i^- = \sum_{j \in \Gamma_i^-} w_j.$$

A escolha gulosa consiste em selecionar a variável x_k com o maior valor de γ_k^+ ou γ_k^- e atribuir a ela o valor booleano correspondente. Caso $\gamma_k^+ > \gamma_k^-$, então a atribuição $x_k = 1$ é feita, caso contrário, faz-se $x_k = 0$. Nota-se que a cada atribuição feita, os conjuntos Γ_i^+ e Γ_i^- são alterados para conter apenas cláusulas que ainda não foram satisfeitas. Conseqüentemente, essa operação altera os valores de γ_i^+ e γ_i^-, caracterizando a componente adaptativa da heurística.

Sejam

$$\gamma^* = \max\{\gamma_i^+, \gamma_i^- \mid x_i \text{ não atribuído}\}$$

e

$$\gamma_* = \min\{\gamma_i^+, \gamma_i^- \mid x_i \text{ não atribuído}\},$$

e seja $\alpha \in [0,1]$ o parâmetro da lista restrita de candidatos. Um novo valor para α é selecionado aleatoriamente da distribuição uniforme $U[0,1]$ em cada iteração. Um candidato $x_i =$ `verdadeiro` é inserido na LRC caso $\gamma_i^+ \geq \gamma_* + \alpha \cdot (\gamma^* - \gamma_*)$. Da mesma forma, um candidato $x_i =$ `falso` é inserido na LRC caso $\gamma_i^- \geq \gamma_* + \alpha \cdot (\gamma^* - \gamma_*)$.

Tabela 3.6: Problemas testes para recobrimento máximo, com o valor da melhor solução conhecida (msc), valores alvos e parâmetros estimados.

			parâmetros estimados	
problema	msc	alvo	$\hat{\mu}$	$\hat{\lambda}$
r24-500	33343542	33330441	-2.257	109.827
		33334808	11.960	229.850
		33339175	2.273	669.501
r25-250	20606926	20567005	1.042	3.319
		20580312	0.716	14.555
		20593619	4.279	101.40
r54-100	39684669	39332719	6.272	42.187
		39450036	17.803	272.29
		39567352	73.320	3978.427
r55-100	39504338	39037219	6.917	9.063
		39192925	3.190	37.672
		39348631	-10.888	279.470

Fase de busca local

Para apresentar o procedimento de busca local, algumas definições preliminares precisam ser feitas. Dada uma atribuição de valores booleanos $x \in \{0,1\}^n$, define-se a vizinhança $N(x)$ como sendo o conjunto de todos os vetores $y \in \{0,1\}^n$ tais que $\| x - y \|_2 = 1$. Interpretando-se x como um vértice de um hipercubo n-dimensional, a sua vizinhança consiste dos n vértices a ele adjacentes.

Seja $w(x)$ o custo total das cláusulas satisfeitas pela atribuição booleana x, então a atribuição x é um *máximo local* se e somente se $w(x) \geq w(y)$, para todo $y \in N(x)$. A busca local é iniciada a partir de uma atribuição booleana x e é executada até encontrar uma solução y em $N(x)$ tal que $w(y) > w(x)$, fazendo-se então, $x = y$. Esse processo é repetido até que não seja mais possível melhorar a solução corrente.

3.2.5 Heurística GRASP para o problema de recobrimento máximo

Um algoritmo GRASP para o problema de recobrimento máximo foi descrito em Resende [111].

Definição do problema

O problema de recobrimento máximo pode ser especificado da seguinte forma. Seja $J = \{1, 2, \ldots, n\}$ um conjunto com n localizações de recursos. São definidos n conjuntos finitos $P_1, P_2, \ldots, P_n$, um para cada localização de

recurso em J. Seja $I = \cup_{j \in J} P_j = \{1, 2, \ldots, m\}$ o conjunto dos m pontos de demanda que podem ser cobertos pelos n recursos em J. A cada ponto de demanda $i \in I$, associa-se um peso $w_i \geq 0$. Um *recobrimento* $J^* \subseteq J$ cobre os pontos de demanda no conjunto $I^* = \cup_{j \in J^*} P_j$ e possui custo $w(J^*) = \sum_{i \in I^*} w_i$. Dado um número $p > 0$ de localizações de recursos que devem ser selecionadas, deseja-se encontrar um conjunto $J^* \subseteq J$ que maximize $w(J^*)$ sujeito à restrição $|J^*| = p$.

Fase construtiva

Como no problema de recobrimento máximo, deve-se escolher p localizações de recursos. Cada fase construtiva é formada por p iterações. Em cada iteração uma nova localização é escolhida para fazer parte da solução.

Para definir uma lista restrita de candidatos, as localizações que ainda não foram escolhidas são ordenadas de acordo com a função gulosa adaptativa. Seja J^* o conjunto (inicialmente vazio) das localizações já escolhidas. A cada iteração da fase construtiva, seja Γ_j o conjunto de pontos de demanda que se tornarão cobertos caso a localização j (para $j \in J \setminus J^*$) seja adicionada a J^*. A *função gulosa*

$$\gamma_j = \sum_{i \in \Gamma_j} w_i$$

é definida como sendo o custo adicional resultante da escolha da localização $j \in J \setminus J^*$. A escolha gulosa consiste em selecionar a localização k que possui o maior valor de γ_k. O conjunto Γ_j é atualizado para incorporar as mudanças causadas por cada nova seleção feita, para cada índice $j \in J \setminus J^*$ de localização de recurso ainda não escolhida. Como conseqüência, essa atualização muda o valor da função gulosa γ_j, caracterizando a componente adaptativa da heurística. Sejam

$$\gamma^* = \max\{\gamma_j \mid \text{localização de recurso } j \text{ ainda não selecionada, isto é, } j \in J \setminus J^*\}$$

e α o parâmetro da lista restrita de candidatos ($0 \leq \alpha \leq 1$). Uma localização j será adicionada à LRC caso $\gamma_j \geq \alpha \cdot \gamma^*$. Uma localização é selecionada aleatoriamente da LRC e adicionada à solução.

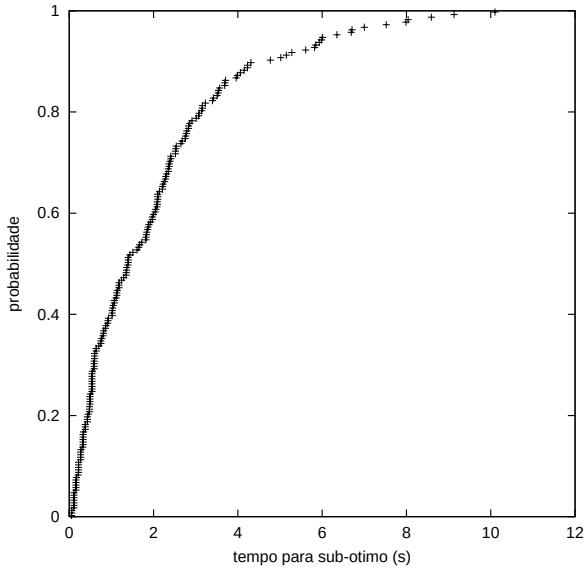

Figura 3.1: Gráfico da distribuição de probabilidade para os dados medidos.

Fase de busca local

Duas soluções (conjuntos de localizações de recursos) J^1 e J^2 são ditas vizinhas em uma vizinhança de trocas-(1,1) se a diferença entre elas for de exatamente um elemento, isto é, $\mid J^1 \cap \Delta J \mid \; = \; \mid J^2 \cap \Delta J \mid \; = 1$, onde $\Delta J = (J^1 \cup J^2) \setminus (J^1 \cap J^2)$. A busca local é iniciada a partir de um conjunto J^* de p localizações de recursos e a cada iteração procura-se encontrar um par de localizações $s \in J^*$ e $t \in J \setminus J^*$ tal que $w(J^* \setminus \{s\} \cup \{t\}) > w(J^*)$. Caso um par com essa característica seja encontrado, a localização s é substituída pela localização t em J^*. Uma solução é localmente ótima em relação a essa vizinhança caso não existam mais trocas que aumentem o custo total de J^*.

3.3 Projeto dos experimentos

Nessa seção os experimentos realizados serão descritos. O objetivo desses experimentos é mostrar que os tempos de processamento medidos seguem uma distribuição exponencial de dois parâmetros. Para cada um dos cinco algoritmos GRASP apresentados na seção anterior, a distribuição da probabilidade do tempo para valor alvo foi estudada para quatro problemas testes. Foram considerados três valores alvos para cada problema teste. Os valores alvos são distribuídos entre um valor de função objetivo distante do ótimo e o melhor valor produzido pelo algoritmo GRASP. Foram realizadas 200 execuções do programa para cada par problema teste/valor alvo. Em cada execução, o programa foi interrompido após encontrar uma solução

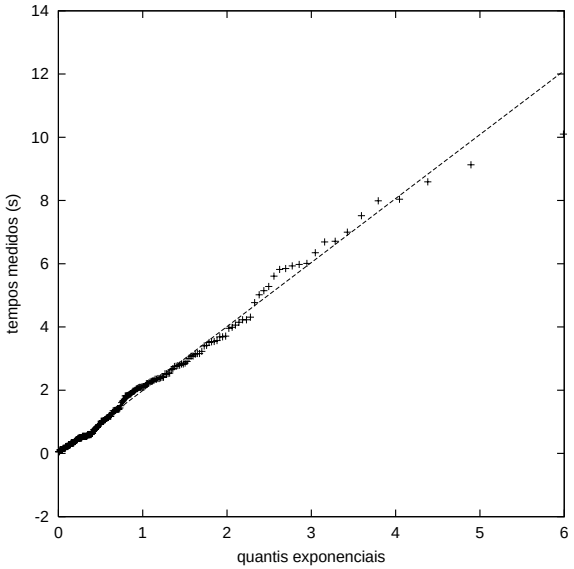

Figura 3.2: Gráfico Q-Q mostrando reta estimada.

de custo pelo menos tão bom quanto o valor alvo, e o seu tempo total de CPU (em segundos) foi registrado. Para cada uma das 200 execuções de cada par, o gerador de números aleatórios foi inicializado com uma semente diferente e portanto, as execuções são independentes. Para comparar as distribuições empíricas à suas distribuições teóricas correspondentes, usou-se uma metodologia gráfica padrão para análise de dados [30]. No restante dessa seção essa metodologia será descrita.

Para cada par problema teste/valor alvo, os tempos de execução medidos são ordenados de forma crescente. Ao i-ésimo tempo de execução ordenado (t_i) é associada uma probabilidade $p_i = (i - \frac{1}{2})/n$. Os pontos $z_i = (t_i, p_i)$ para $i = 1, \ldots, n$ são plotados em um gráfico. A expressão usada para determinar p_i será comentada mais adiante nessa seção. A Figura 3.1 mostra um gráfico da distribuição de probabilidade do tempo para valor alvo do algoritmo GRASP para um determinado par problema teste/valor alvo.

Um gráfico quantil-quantil (ou gráfico Q-Q) é usado para estimar os parâmetros da distribuição exponencial de dois parâmetros. Para descrever o gráfico Q-Q, é necessário relembrar que a função cumulativa de distribuição para a distribuição exponencial de dois parâmetros é dada por

$$F(t) = 1 - e^{-(t-\mu)/\lambda},$$

onde λ é a média da distribuição (e indica o espalhamento dos dados) e μ é o afastamento da distribuição em relação ao eixo das ordenadas.

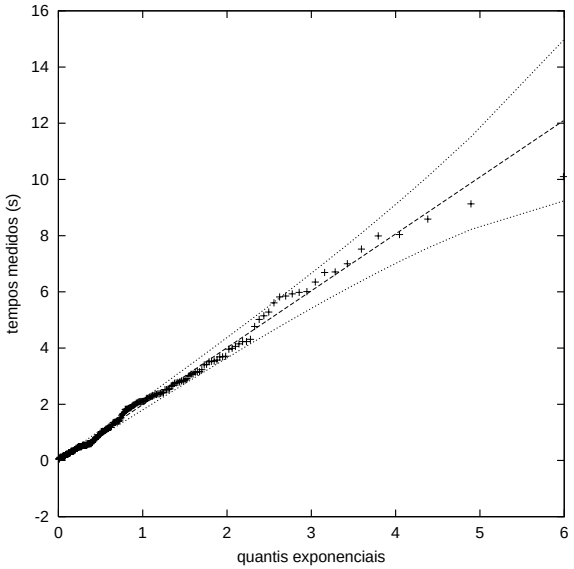

Figura 3.3: Gráfico Q-Q com informação de variabilidade.

A cada valor de p_i, para $i = 1, \ldots, n$, é associado o valor do seu quantil $Qt(p_i)$ na distribuição teórica. Tem-se que o quantil de p_i é dado por:

$$F((Qt(p_i)) = p_i.$$

Portanto, $Qt(p_i) = F^{-1}(p_i)$ e dessa forma, para a distribuição exponencial de dois parâmetros, tem-se:

$$Qt(p_i) = -\lambda \ln(1 - p_i) + \mu.$$

Os quantis dos dados de uma distribuição empírica são simplesmente os dados medidos ordenados. Deve-se notar que a probabilidade p_i não é calculada fazendo-se $p_i = i/n$, para $i = 1, \ldots, n$, porque dessa forma, $Qt(p_n)$ seria indefinido. Um gráfico quantil-quantil teórico (ou gráfico Q-Q teórico) é obtido plotando-se os quantis dos dados de uma distribuição empírica contra os quantis de uma distribuição teórica. Essa operação envolve três passos. Primeiro, os dados (no caso, os tempos medidos) são ordenados em ordem crescente. Em seguida, os quantis da distribuição exponencial de dois parâmetros teórica são calculados. Finalmente, os dados são plotados em função dos quantis teóricos.

Em uma situação em que a distribuição teórica é uma boa aproximação para a distribuição empírica, os pontos do gráfico Q-Q aparecerão dispostos de forma aproximadamente linear. Caso os parâmetros λ e μ da distribuição teórica que melhor representa os dados medidos pudessem ser estimados a priori, os pontos do gráfico Q-Q apareceriam dispostos próximos à reta

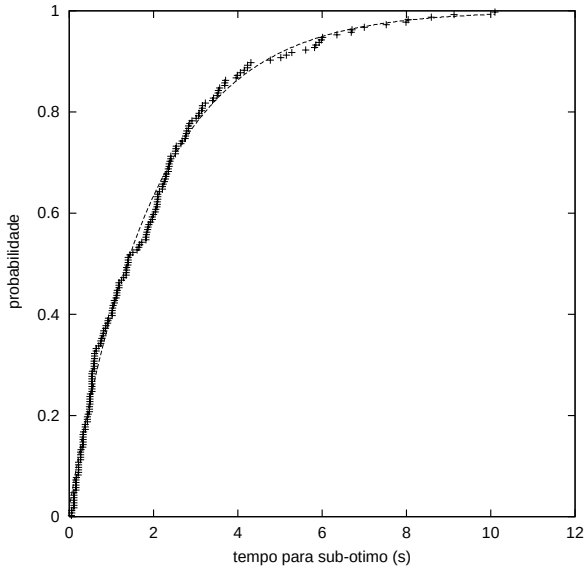

Figura 3.4: Distribuições empírica e teórica sobrepostas.

$x = y$. Alternativamente, em um gráfico onde os dados são plotados contra uma distribuição exponencial de dois parâmetros onde $\lambda' = 1$ e $\mu' = 0$, os pontos aparecerão dispostos próximo à reta $y = \lambda x + \mu$. Isso significa que um único gráfico Q-Q teórico não compara um conjunto de dados com apenas uma distribuição teórica, mas sim com uma família inteira de distribuições teóricas. Conseqüentemente, os parâmetros λ e μ da distribuição exponencial de dois parâmetros podem ser estimados, respectivamente, pela inclinação e pelo afastamento da reta traçada no gráfico Q-Q.

O gráfico Q-Q mostrado na Figura 3.2 é obtido plotando-se os tempos medidos em função dos quantis da distribuição exponencial de dois parâmetros com $\lambda' = 1$ e $\mu' = 0$, dados por $-\ln(1 - p_i)$, para $i = 1, \ldots, n$. Para evitar possíveis distorções causadas por exceções, os parâmetros da distribuição teórica não são calculados através de uma regressão linear nos pontos do gráfico Q-Q. Ao invés, a inclinação estimada $\hat{\lambda}$ da reta $y = \lambda x + \mu$ é calculada usando-se os quartis superior q_u e inferior q_l dos dados e os valores correspondentes na abscissa [30]. Os quartis superior e inferior são, respectivamente, os quantis $Q(\frac{1}{4})$ e $Q(\frac{3}{4})$. Calcula-se então

$$\hat{\lambda} = (q_u - q_l)/(z_u - z_l)$$

como um estimativa da inclinação da reta, onde z_u e z_l são os valores na abscissa correspondentes aos quantis superior q_u e inferior q_l, respectivamente.

Para analisar a linearidade dos dados nos gráficos Q-Q, esses gráficos

foram sobrepostos com informações de variabilidade. Para cada ponto medido, são mostrados no gráfico os desvios padrões na direção vertical da reta adaptada aos dados. Uma estimativa do desvio padrão dos pontos z_i, $i = 1, \ldots, n$, do gráfico Q-Q é

$$\hat{\sigma} = \hat{\lambda}\sqrt{\frac{p_i}{(1-p_i)n}}.$$

A Figura 3.3 mostra um exemplo de gráfico Q-Q com informação de variabilidade sobreposta.

A análise de um gráfico Q-Q com informação de desvio padrão sobreposta não deve ser considerada como um teste formal. Um fato que deve ser sempre lembrado ao analisar esses gráficos é que a variabilidade natural de dados reais gera desvios em relação à linearidade, mesmo que o modelo de distribuição seja válido. A razão mais importante para exibir desvios padrões é que eles permitem verificar a variabilidade dos pontos em relação à reta traçada, para as diferentes regiões do gráfico. Porém, como pretende-se fazer inferências simultâneas a partir de várias inferências individuais, torna-se difícil usar desvios padrões para julgar deslocamentos em relação à distribuição referente. Por exemplo, a probabilidade de que um ponto particular desvie da reta por mais do que dois desvios padrões é pequena. Porém, a probabilidade de que pelo menos um dos pontos desvie da reta por dois desvios padrões é muito mais alta. Em estatística de ordem, essa análise é ainda mais difícil devido à alta correlação que existe entre pontos vizinhos. Caso um ponto plotado esteja afastado da reta por mais do que um desvio padrão, existe uma boa chance de que vários outros pontos também estarão. Um outro fator a ser considerado é que desvios padrões variam substancialmente nos gráficos Q-Q. Esse fato pode ser observado nos gráficos Q-Q da Figura 3.3, onde o desvio padrão dos pontos próximos à extremidade mais alta são substancialmente maiores do que os desvios padrões na outra extremidade.

Uma vez que os dois parâmetros da distribuição tenham sido estimados, pode-se plotar um gráfico onde uma distribuição teórica é sobreposta à distribuição empírica. A Figura 3.4 exibe um gráfico com a distribuição teórica calculada a partir do gráfico Q-Q da Figura 3.2 sobreposta à distribuição empírica dos tempos medidos.

3.4 Resultados computacionais

Nessa seção os resultados computacionais serão apresentados. O ambiente computacional usado para conduzir os experimentos será descrito, assim como os problemas testes selecionados para cada uma das cinco heurísticas GRASP. Serão apresentados para cada tripla heurística GRASP/problema teste/valor alvo, o gráfico Q-Q com informação de variabilidade, os dois parâmetros estimados e o gráfico com as distribuições teórica e empírica sobrepostas.

3.4.1 Ambiente computacional

Os experimentos foram feitos 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. A cada execução apenas um processador foi usado.

Os programas foram escritos em Fortran e foram modificados minimamente para produzirem os dados necessários para o experimento. Os tempos de CPU foram medidos com a função `etime` do sistema operacional. Os códigos foram compilados com o compilador SGI MPISpro F77. Os programas para conjunto máximo independente e para atribuição quadrática foram compilados usando as opções `-Ofast -u` e o programa para satisfabilidade máxima foi compilado usando as opções `-Ofast -static`. O programa para recobrimento máximo foi compilado usando as opções `-O3 -r4 -64` e o programa para planarização foi compilado usando-se `-O3 -static`. O gerador de números pseudo-aleatórios proposto em [124] foi usado para gerar a seqüência de números usados para iniciar cada iteração do algoritmo GRASP.

3.4.2 Problemas testes

O nome de cada problema teste e seus respectivos valores alvos são mostrados nas Tabelas 3.2, 3.3, 3.4, 3.5 e 3.6.

Os quatro problemas usados para estudar o algoritmo GRASP para o conjunto máximo independente foram escolhidos da classe $G_{n,p}$ de grafos aleatórios [19]. Tais grafos possuem n nós e cada arco (i, j), para $i, j = 1, \ldots, n$ e $i \neq j$, existe segundo uma probabilidade p. O programa foi

executado para quatro problemas testes de $G_{n,p}$ ($n = 1000, p = 0.5$) gerados aleatoriamente usando-se o código Fortran na distribuição [110]. As sementes iniciais do gerador de números aleatórios usadas para gerar os problemas testes foram 27002, 27003, 27004 e 27006.

Para a heurística GRASP para o QAP, os problemas testes `chr25a`, `kra30b`, `sko42` e `tho40` foram selecionados da coleção QAPLIB [23]. Esses problemas testes são problemas de atribuição quadrática puros que possuem pelo menos uma das matrizes de distância ou de fluxo simétricas. Suas dimensões (n) vão de 25 a 42.

O algoritmo GRASP para planarização de grafos foi testado para os problemas testes `g17`, `tg100.10`, `rg100.1` e `rg150.1`, selecionados de um conjunto de 75 problemas testes descritos na literatura [33, 62]. As dimensões dos problemas selecionados variam de 100 a 150 vértices e de 261 a 742 arestas.

Os problemas testes para o problema de MAX-SAT foram escolhidos entre as instâncias apresentadas em [109]. Os problemas `jnh11` e `jnh12` possuem 100 variáveis e 800 cláusulas, o problema `jnh212` possui 100 variáveis e 850 cláusulas e o problema `jnh306` possui 100 variáveis e 900 cláusulas.

Os problemas testes usados nos experimentos com o algoritmo GRASP para recobrimento máximo (`r24-500`, `r25-250`, `r54-100` e `r55-100`) foram gerados aleatoriamente usando o gerador descrito em [111]. Todos os problemas testes possuem 1000 localidades em potencial e pontos de demanda variando de 7425 a 9996. O número de facilidades para serem alocadas varia de 100 a 500.

3.4.3 Gráficos Q-Q e distribuições teóricas

Nessa subseção, serão apresentados os gráficos Q-Q com informação de variabilidade e os gráficos correspondentes com as distribuições teórica e empírica sobrepostas.

As figuras que contém os gráficos Q-Q são formadas por 12 gráficos desse tipo, um para cada par problema teste/valor alvo. Da mesma forma, as figuras que contém os gráficos das distribuições empírica e teórica sobrepostas são formadas por 12 gráficos, um para cada par problema teste/valor alvo. Cada gráfico foi gerado a partir de 200 pontos de dados. Cada figura é formada por quatro linhas onde são dispostos os gráficos. Cada linha corresponde a um dos quatro problemas testes escolhidos para

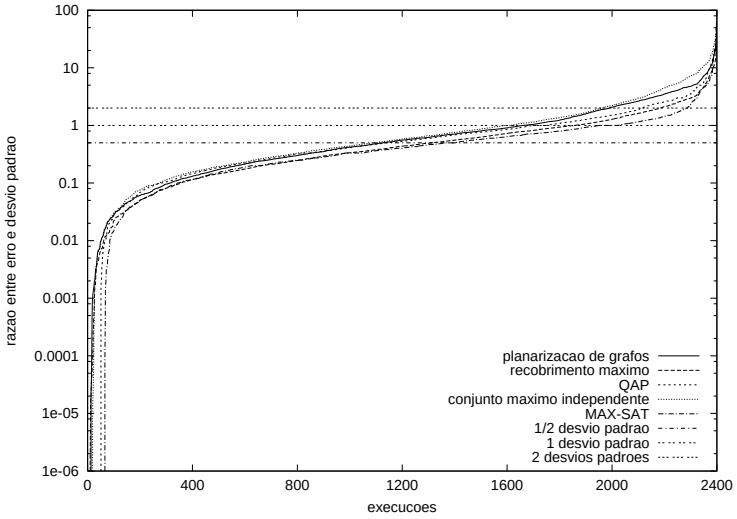

Figura 3.5: Razão entre o erro e o desvio padrão para tempos medidos (todos os pares problema teste/valor alvo foram considerados).

cada algoritmo. Para cada problema teste, três valores alvos são usados. Em cada linha das figuras, a dificuldade de se encontrar uma solução para um dado valor alvo aumenta da esquerda para a direita.

As Figuras 3.7 e 3.8 mostram, respectivamente, os gráficos Q-Q e os gráficos com as distribuições teórica e empírica sobrepostas para os problemas testes para o problema de conjunto máximo independente. Para os problemas testes para atribuição quadrática, os gráficos Q-Q e os gráficos com as distribuições teórica e empírica sobrepostas são mostrados, respectivamente, nas Figuras 3.9 e 3.10. As Figuras 3.11 e 3.12 mostram, respectivamente, os gráficos Q-Q e os gráficos com as distribuições sobrepostas para os problemas testes para planarização de grafos. Para os problemas testes para satisfabilidade máxima, os gráficos Q-Q e os gráficos com as distribuições sobrepostas são mostrados respectivamente, nas Figuras 3.13 e 3.14. As Figuras 3.15 e 3.16 mostram, respectivamente, os gráficos Q-Q e as distribuições teórica e empírica sobrepostas para os problemas testes para recobrimento máximo. Os parâmetros estimados para cada heurística GRASP são mostrados nas Tabelas 3.2, 3.3, 3.4, 3.5 e 3.6.

Um total de 12000 execuções independentes foram executadas, cada uma encontrando uma solução pelo menos tão boa quanto o seu valor alvo correspondente.

Em geral, não houve nenhum afastamento substancial ou sistemático da linearidade nos gráficos Q-Q. Porém, o fato de que amostras de dados reais são freqüentemente contaminadas por uma pequena fração de observações anormais, que aparecem fora do intervalo típico de dados é bastante

conhecido em estatística. Isso pode ser observado nos experimentos apresentados nesse capítulo.

Observa-se que a linearidade nos gráficos Q-Q geralmente aumenta com a dificuldade do problema. Dessa forma, quanto mais o valor alvo aproxima-se do ótimo, a distribuição teórica adapta-se melhor à distribuição dos dados medidos, o que indica que os parâmetros calculados para a distribuição teórica foram boas estimativas. Esse fato ocorre porque o número de iterações necessárias para o algoritmo GRASP encontrar uma solução satisfatória varia mais, de acordo com a semente inicial, para problemas mais difíceis. Em alguns dos problemas testes mais fáceis, muitas execuções precisaram de apenas poucas iterações. Esse problema é discutido mais adiante nessa seção.

De acordo com a Seção 3.3, os pontos plotados em cada figura podem ser vistos como estatísticas de ordem. Devido à alta correlação que existe entre estatísticas de ordem vizinhas, a probabilidade de que um ponto particular desvie da reta por mais do que dois desvios padrões é pequena, mas a probabilidade de que pelo menos um dos pontos desvie da reta por dois desvios padrões é, sem dúvida, muito maior. Nas Figuras 3.5 e 3.6 são plotadas as razões entre o afastamento de cada ponto em relação à reta estimada no gráfico Q-Q correspondente e o valor do desvio padrão nesse ponto, para o total de 2400 execuções de cada programa GRASP e para as 800 execuções feitas para os valores alvos mais difíceis dos quatro problemas testes de cada problema, respectivamente. Os valores alvos mais difíceis para cada problema teste correspondem à coluna mais à direita nas figuras que mostram os gráficos Q-Q, assim como nas figuras que mostram as distribuições empírica e teórica sobrepostas. As razões foram ordenadas em ordem crescente. Aproximadamente 75% de todos os pontos dos gráficos Q-Q estão afastados da reta estimada de no máximo um desvio padrão e aproximadamente 88% estão afastados da reta estimada de no máximo dois desvios padrões. Quando essa análise é limitada apenas aos pares problema teste/valor alvo mais difíceis, aproximadamente 80% dos pontos dos gráficos Q-Q estão afastados da reta estimada por no máximo um desvio padrão e 93% dos pontos estão afastados da reta estimada por no máximo dois desvios padrões.

Como pode ser observado nos gráficos Q-Q, poucos pontos relativos a tempos de processamento elevados estão fora do intervalo de um desvio padrão. Portanto, a maior parte dos pontos que estão fora do intervalo de dois desvios padrões são pontos relativos a tempos de processamento pequenos e que, por sua vez, possuem desvio padrão pequeno. De fato,

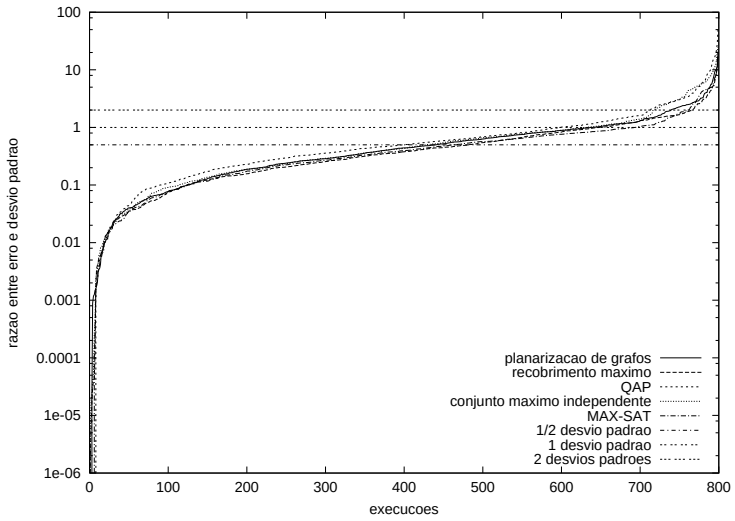

Figura 3.6: Razão entre o erro e o desvio padrão para tempos medidos (apenas pares problema teste/valor alvo mais difíceis foram considerados).

caso sejam considerados apenas os 180 maiores tempos de processamento (de um total de 200) para cada par problema teste/valor alvo, observa-se que 93% dos pontos dos gráficos Q-Q estão afastados de no máximo dois desvios padrões da reta estimada. Restringindo-se os dados para apenas os 180 maiores tempos de processamento para os pares de problema teste/valor alvo mais difíceis, tem-se que 98% dos pontos estão dentro do intervalo de dois desvios padrões da reta estimada.

Gráficos Q-Q com segmentos horizontais, como observados nas Figuras 3.7 e 3.11, são freqüentes na prática [30]. Essa granularidade discreta pode significar que os dados foram arredondados em alguma etapa do experimento antes de serem plotados. Nos experimentos realizados nesse trabalho, isso é observado apenas para os casos de pares problema teste/valor alvo mais fáceis. Isso ocorre porque em várias execuções, uma solução satisfatória foi encontrada na mesma iteração. Nesses exemplos, o i-ésimo segmento horizontal representa os tempos para valor alvo encontrados na i-ésima iteração. Apesar desse fato ser uma anomalia, caso essas repetições sejam eliminadas e cada segmento seja transformado em um ponto, representado pela sua mediana, esses pontos estarão dispostos aproximadamente em linha reta.

3.5 Conclusão

Nesse capítulo foi mostrada uma investigação empírica da distribuição do tempo para valor alvo do método GRASP. Um total de 12000 execuções do algoritmo foram feitas no experimento. Para estudar os resultados, foi usada uma metodologia gráfica padrão para análise de dados. As distribuições teóricas foram estimadas usando-se informações de gráficos Q-Q, plotados a partir dos dados medidos. Informações de variabilidade foram plotadas nos gráficos Q-Q para determinar se a distribuição teórica estimada adapta-se bem a distribuição empírica dos dados.

A conclusão principal desses experimentos é que a distribuição do tempo para valor alvo da metaheurística GRASP ajusta-se corretamente a uma distribuição exponencial de dois parâmetros. A distribuição teórica estará tão melhor adaptada à distribuição empírica dos dados, quanto maior for a dificuldade de se encontrar uma solução com um dado valor alvo.

Apesar desse estudo ter sido limitado a cinco implementações de GRASP, acredita-se que essa característica esteja presente em qualquer algoritmo GRASP implementado da maneira básica. O tempo para valor alvo adapta-se corretamente a uma distribuição exponencial de dois parâmetros, porém, em geral, a razão $|\mu|/\lambda$ é baixa. Observou-se na Seção 3.1 que quando $\rho|\mu| \ll \lambda$, acelerações aproximadamente lineares podem ser atingidas. Portanto, esse será o caso para um algoritmo GRASP básico que usa o critério de parada por qualidade de solução, para um número não muito elevado de processadores ρ. Deve-se notar que artifícios usados para reduzir o tempo de execução de um algoritmo GRASP seqüencial, tais como tabelas *hash* usadas para evitar a repetição de buscas locais iniciadas de soluções idênticas, podem levar a aceleração do algoritmo paralelo a ser sublinear em relação ao número de processadores. Um exemplo disso pode ser visto em [83], onde o uso de uma tabela *hash* reduz o tempo de um programa GRASP seqüencial em problemas testes onde a fase construtiva gera muitas soluções repetidas. A versão paralela desse programa repete muitas dessas busca locais desnecessariamente, porque utiliza uma tabela *hash* diferente para cada processo e, dessa forma, a informação colhida durante a execução do algoritmo não é vista de forma centralizada.

No próximo capítulo algumas variantes do algoritmo GRASP serão apresentadas para o problema de atribuição de três índices. Nessas variantes, o religamento de caminhos é usado no contexto do algoritmo GRASP para gerar soluções que incorporam atributos de soluções de elite. Será mostrado

que a variável aleatória "tempo para valor alvo" também se encaixa em uma distribuição exponencial de dois parâmetros, para todas as variantes propostas.

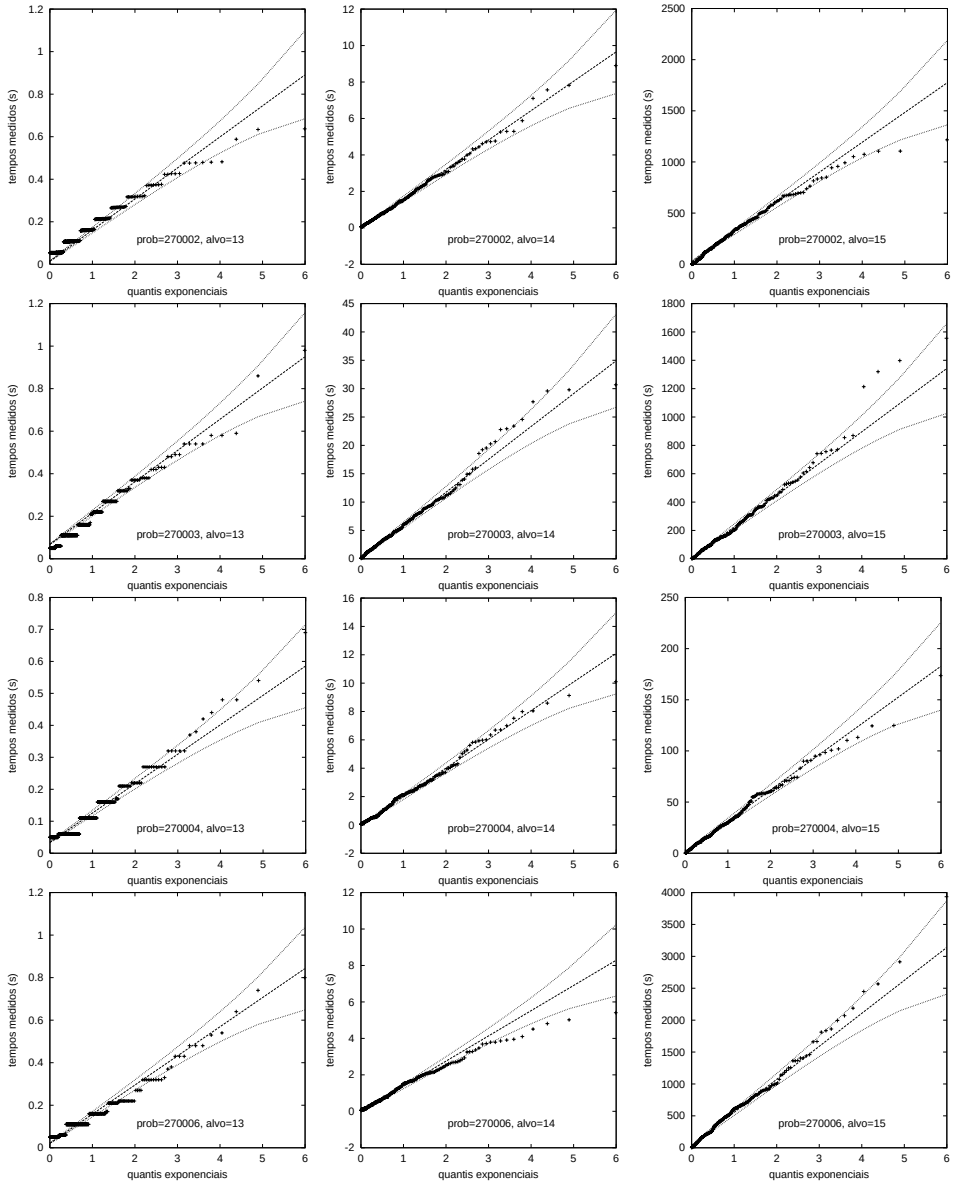

Figura 3.7: Gráficos Q-Q para o problema de conjunto máximo independente.

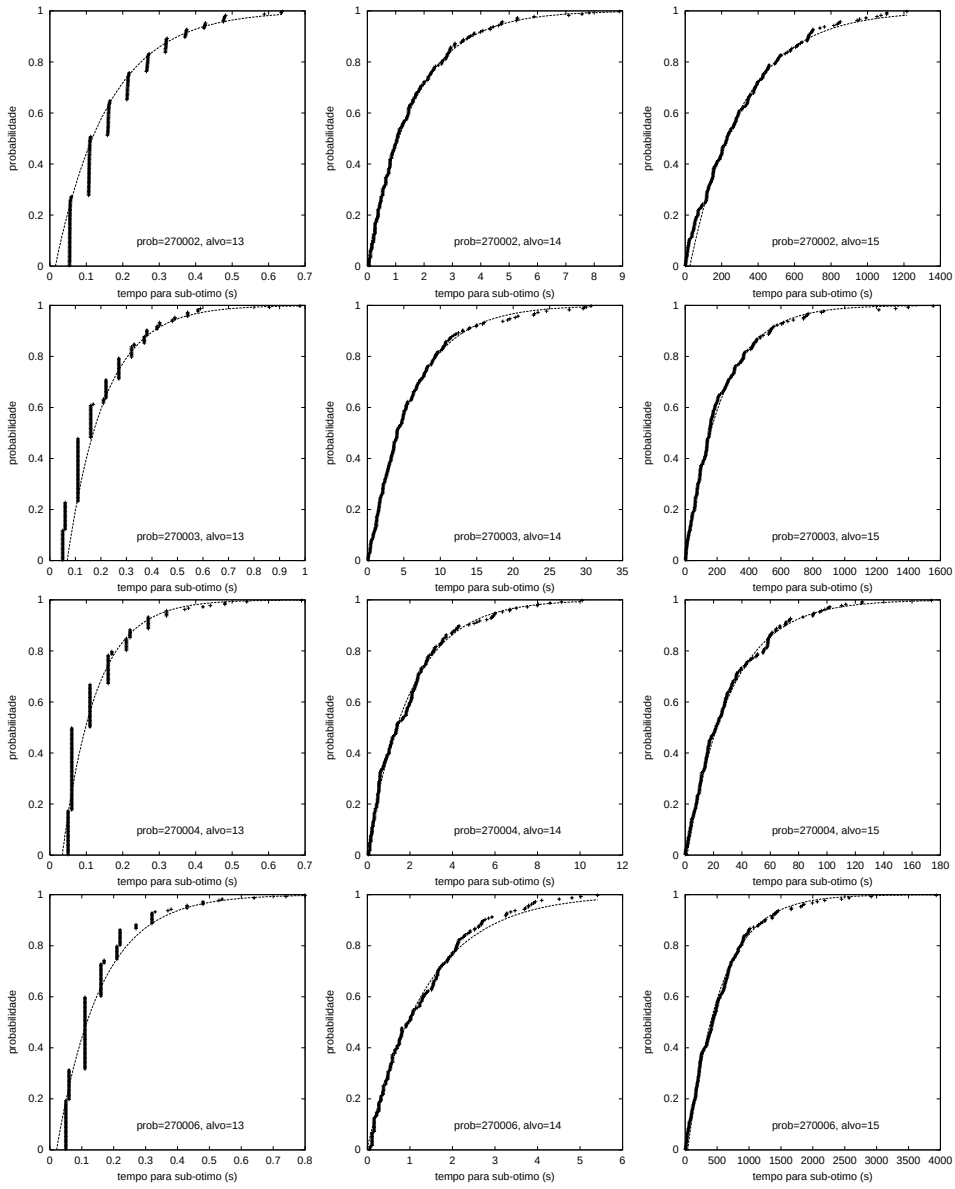

Figura 3.8: Gráficos das distribuições exponenciais para o problema de conjunto máximo independente.

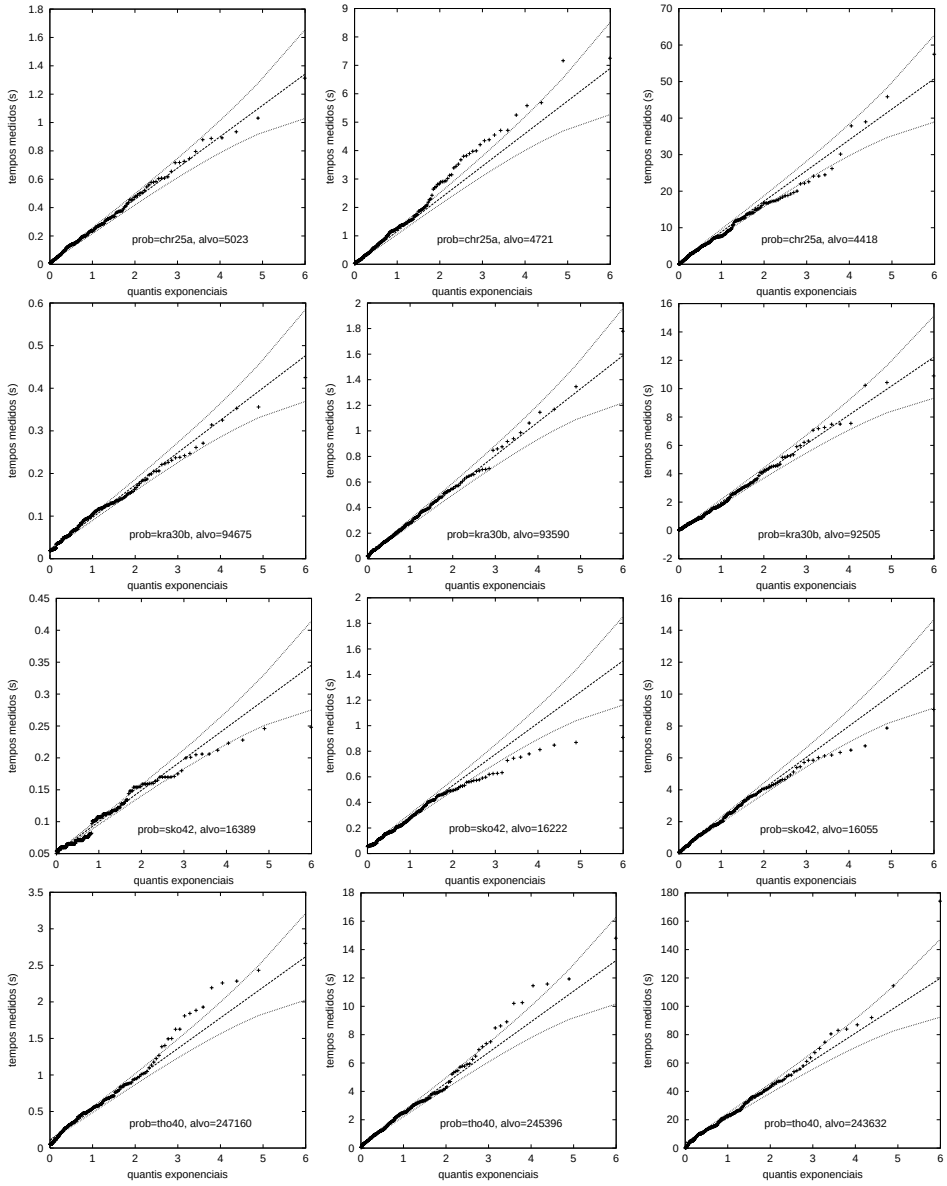

Figura 3.9: Gráficos Q-Q para o problema quadrático de atribuição.

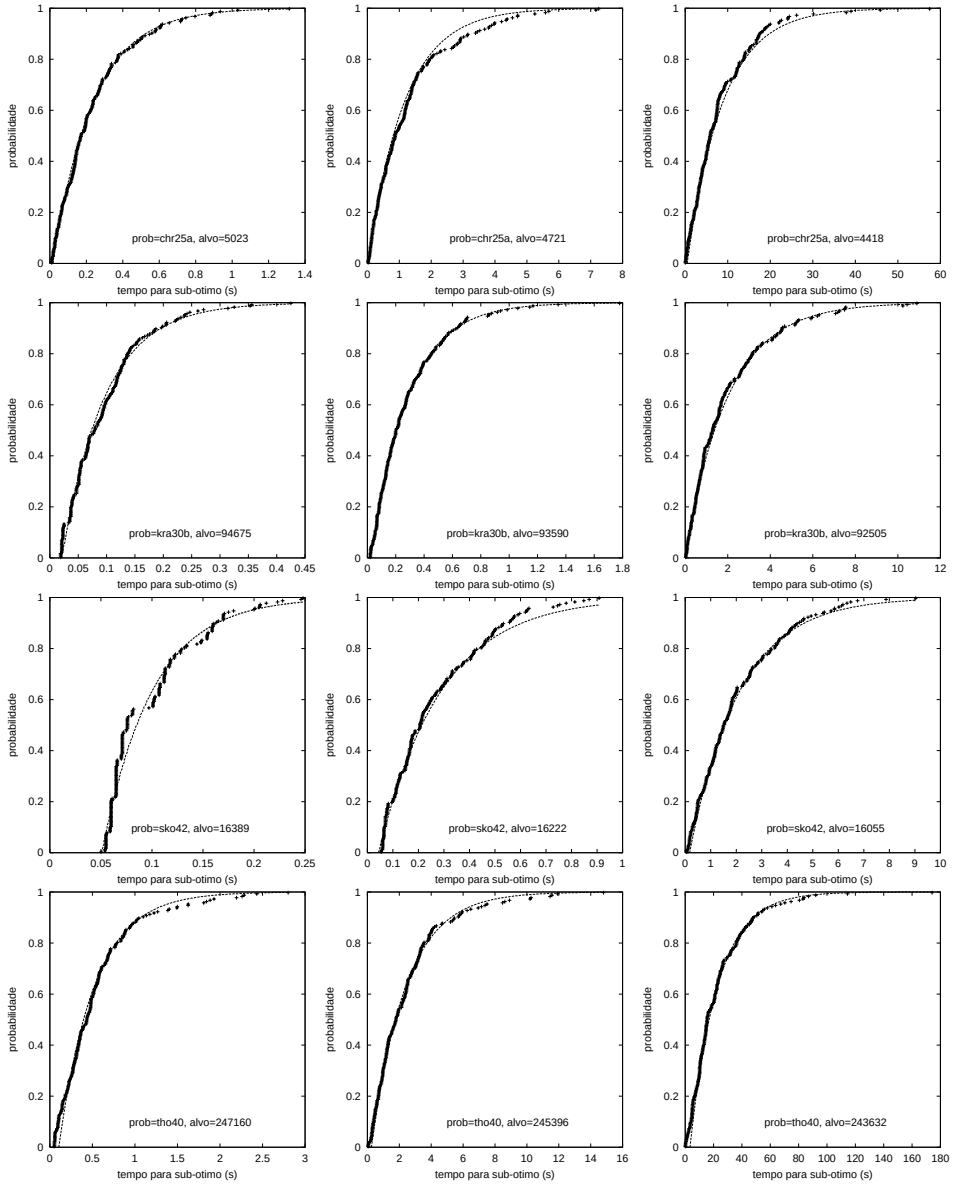

Figura 3.10: Gráficos das distribuições exponenciais para o problema quadrático de atribuição.

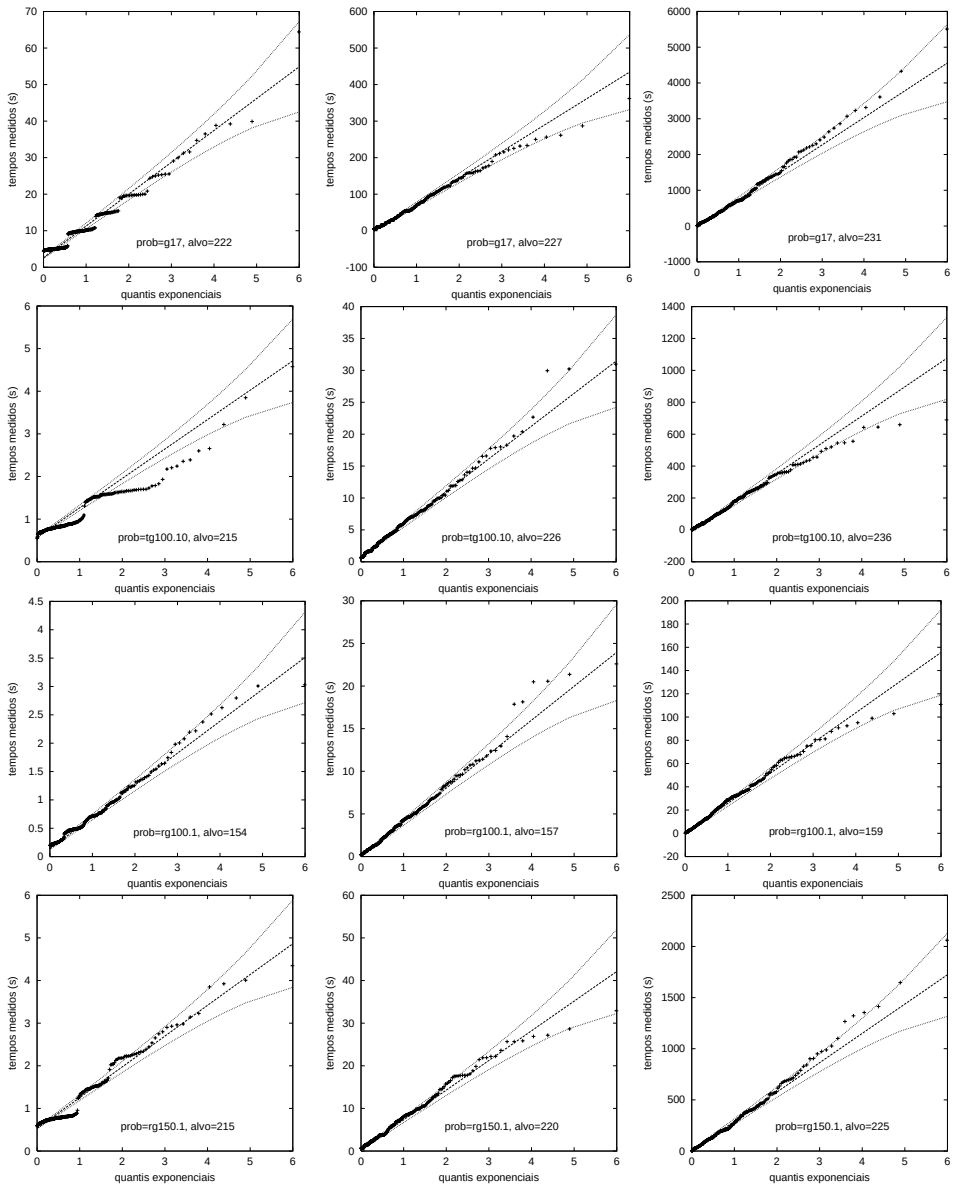

Figura 3.11: Gráficos Q-Q para o problema de planarização de grafos.

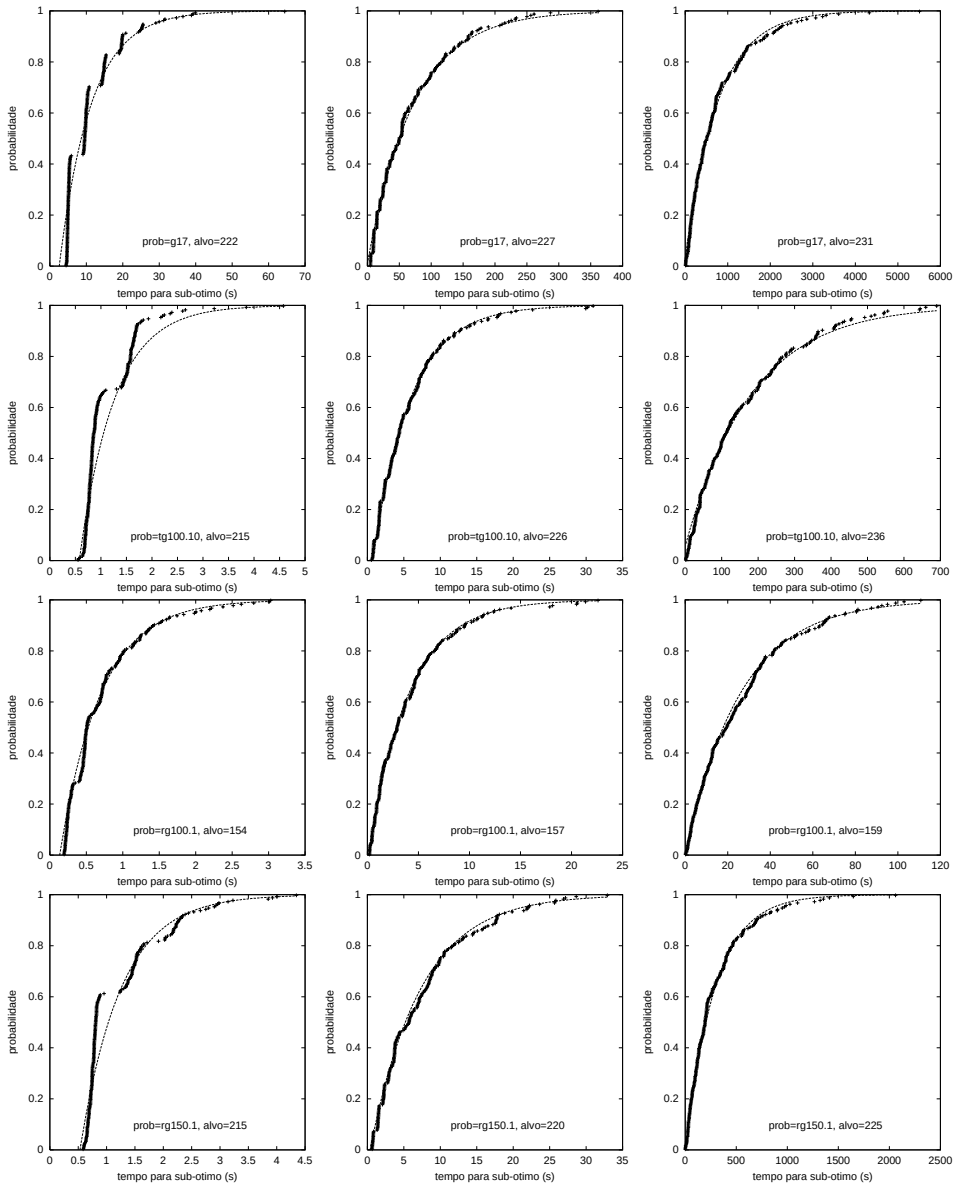

Figura 3.12: Gráficos das distribuições exponenciais para o problema de planarização de grafos.

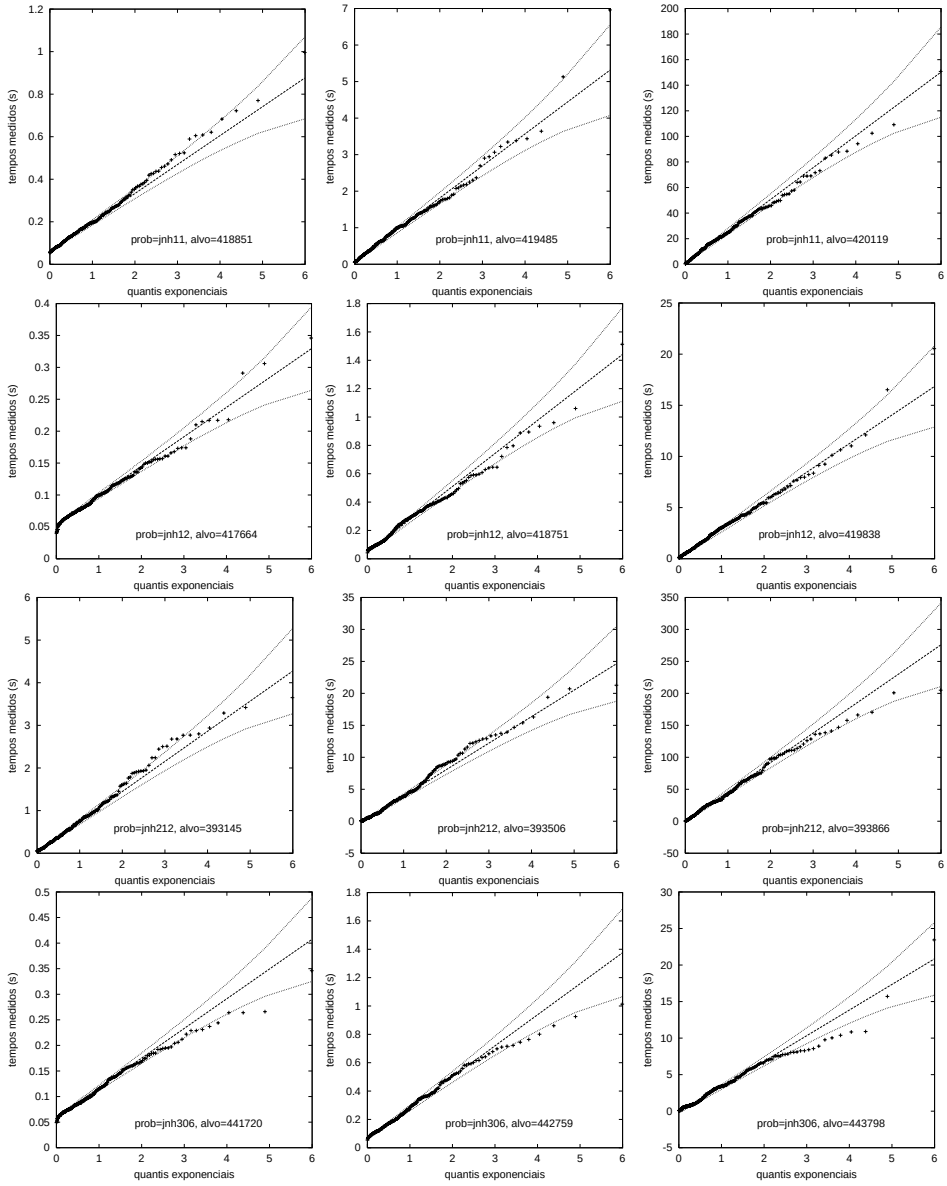

Figura 3.13: Gráficos Q-Q para o problema de satisfabilidade valorada máxima.

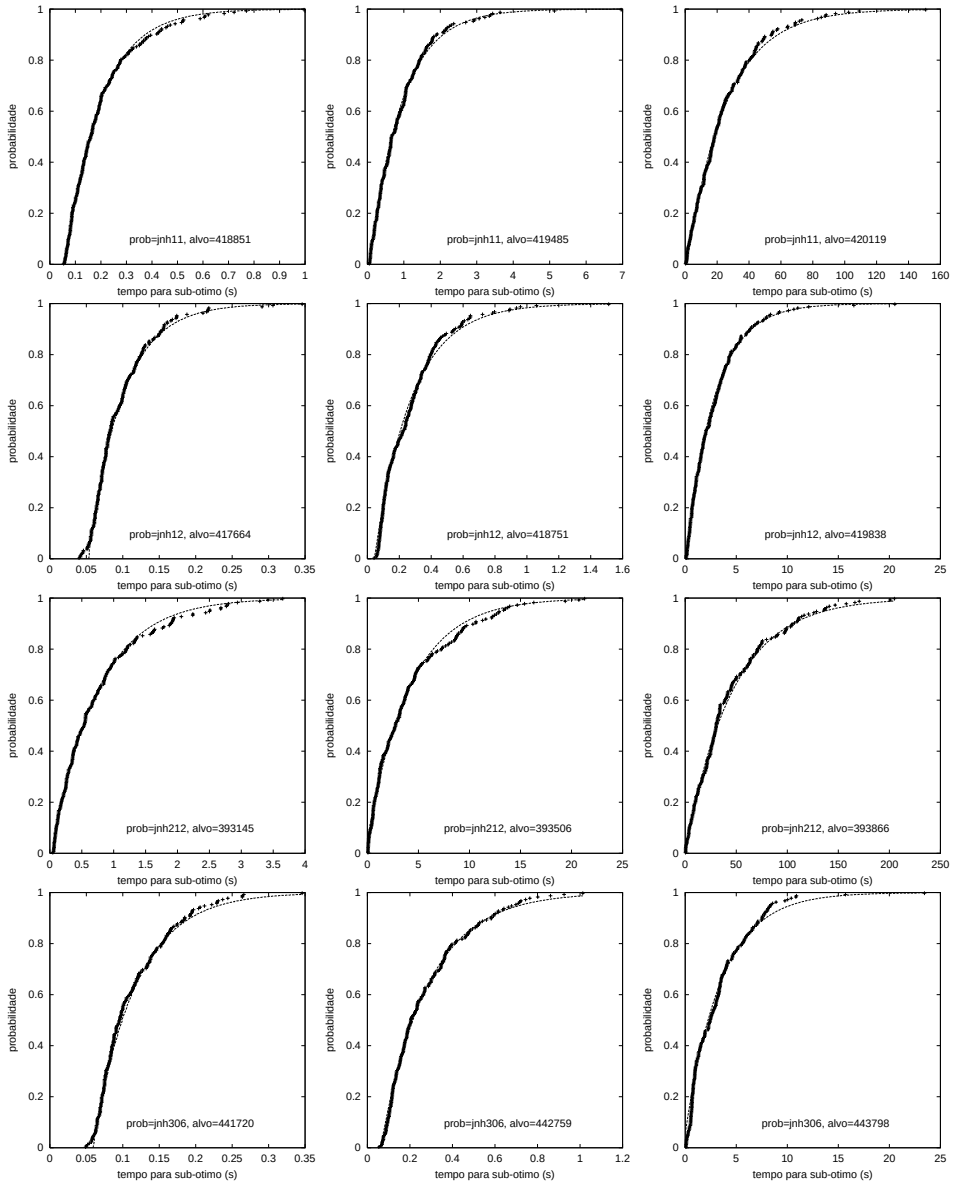

Figura 3.14: Gráficos das distribuições exponenciais para o problema de satisfabilidade valorada máxima.

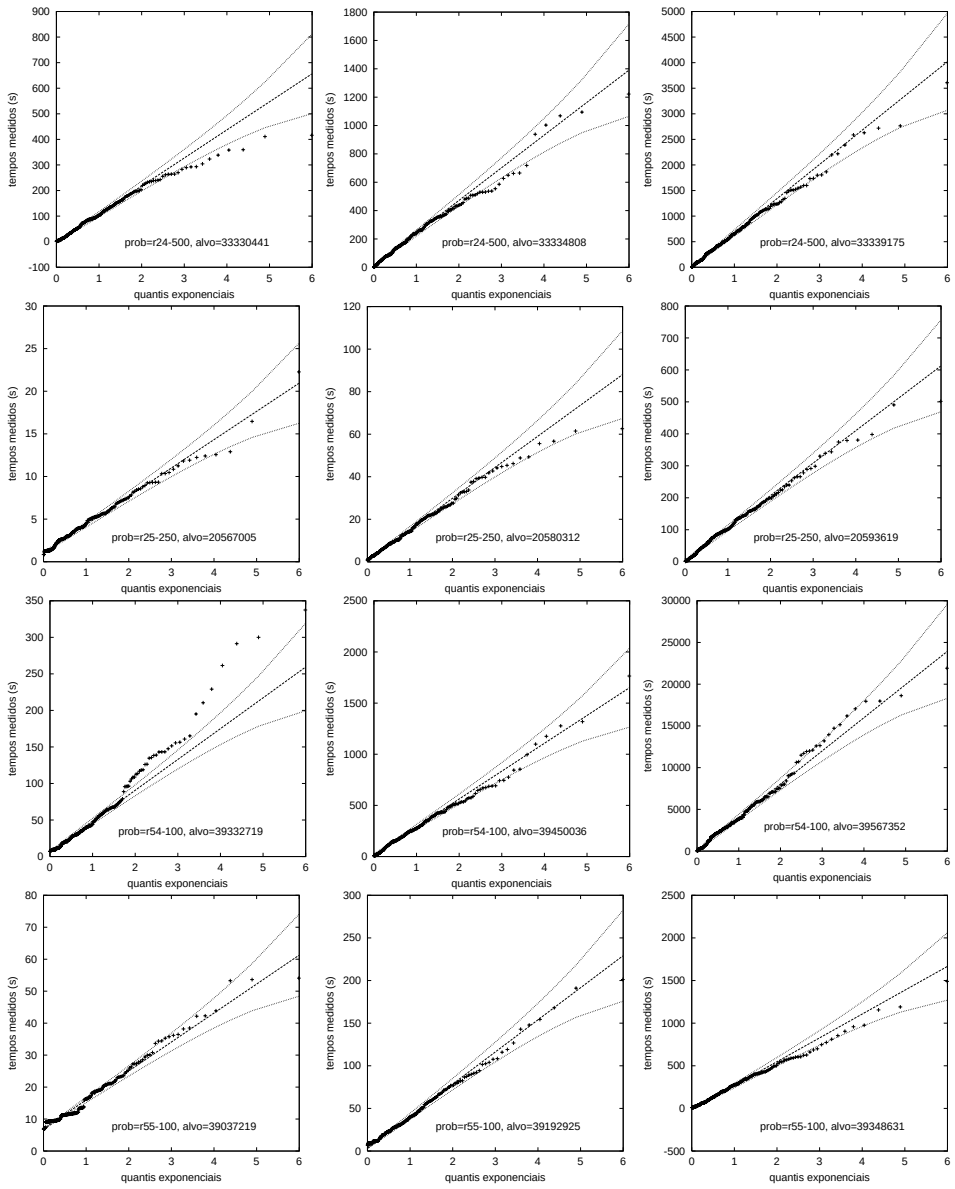

Figura 3.15: Gráficos Q-Q para o problema de recobrimento máximo.

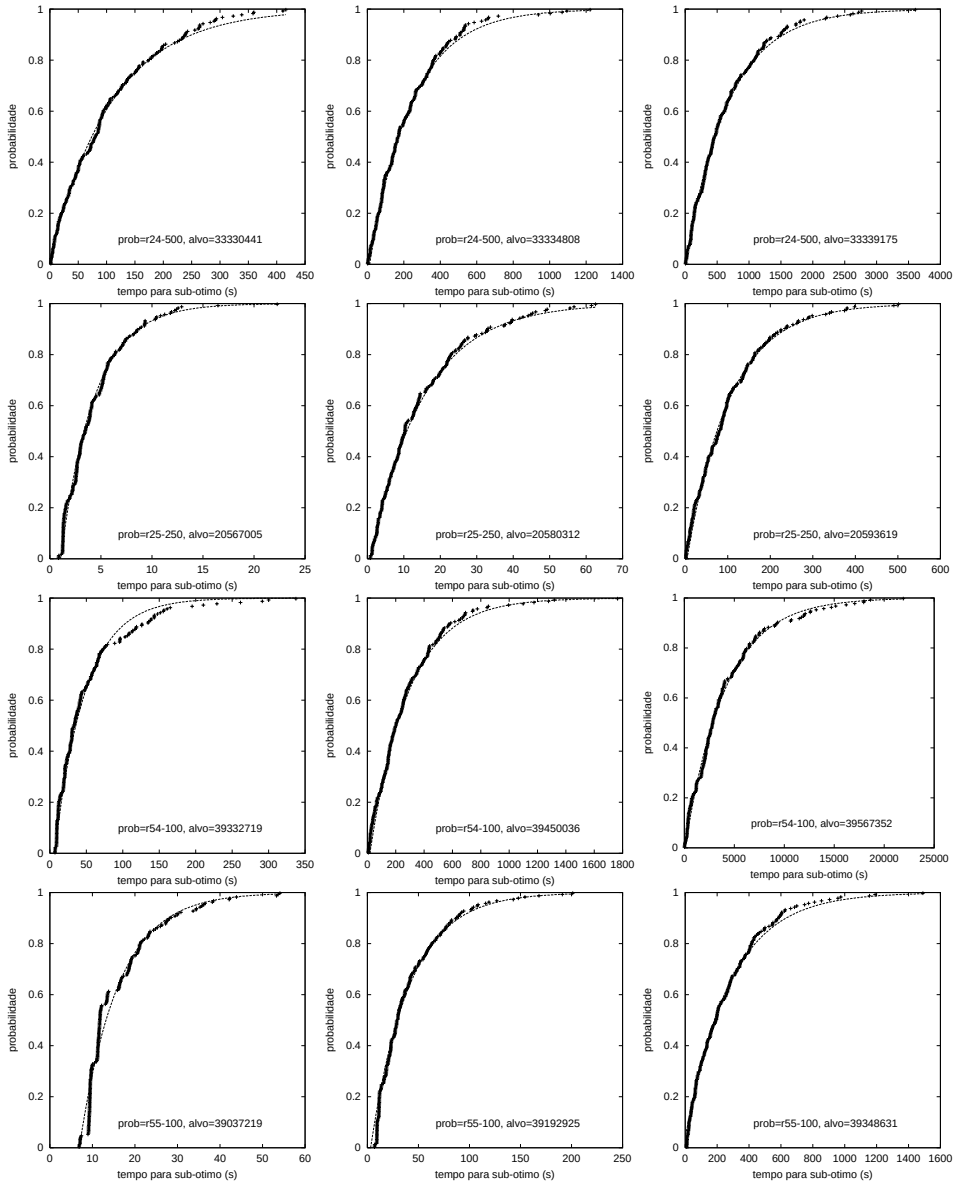

Figura 3.16: Gráficos das distribuições exponenciais para o probema de recobrimento máximo.