4
Estratégias de Paralelização

Embora metaheurísticas sejam estratégias eficientes para solucionar problemas de otimização combinatória, os tempos de exploração do espaço de soluções podem ser muito altos. Com a proliferação de computadores paralelos, estações de trabalho e computadores pessoais cada vez mais velozes e redes de alta velocidade de comunicação, implementações paralelas surgem como uma alternativa bastante promissora para acelerar a busca por soluções aproximadas.

Além disto, implementações paralelas permitem resolver problemas maiores ou encontrar melhores soluções, devido ao particionamento do espaço de busca e às maiores possibilidades de intensificar e diversificar a busca. A robustez das estratégias paralelas é uma das mais importantes contribuições do paralelismo em heurísticas. Implementações robustas podem ser obtidas pelo uso de diferentes combinações de estratégias, levando a boas soluções aproximadas para diferentes classes de instâncias do mesmo problema, sem grande esforço no ajuste de parâmetros.

O objetivo deste capítulo é desenvolver heurísticas GRASP paralelas com reconexão por caminhos bidirecional para 2PNDP. Estas estratégias paralelas serão avaliadas e comparadas para verificar qual o papel que a colaboração entre os processadores desempenha em termos da convergência para as melhores soluções e do encontro das melhores soluções. Procura-se também estudar a melhor maneira de desenvolver implementações paralelas, para se utilizar da melhor forma possível dos recursos computacionais e reduzir conflitos de memória e comunicação. Na Seção 4.1, é realizado um levantamento dos principais resultados das implementações GRASP paralelas existentes na literatura. Na Seção 4.2, é apresentada a heurística GRASP seqüencial com reconexão por caminhos bidirecional para um problema de minimização. Na Seção 4.3, a estratégia paralela independente para 2PNDP é discutida. Na Seção 4.4, a estratégia paralela colaborativa centralizada para 2PNDP é descrita. Na Seção 4.5, é apresentada a estratégia paralela colaborativa distribuída para 2PNDP. Na Seção 4.6, são realizadas análises

das diferentes variantes paralelas de GRASP com reconexão por caminhos bidirecional. Finalmente, são feitas na Seção 4.7 algumas considerações finais sobre paralelismo em metaheurística GRASP.

4.1 Revisão Bibliográfica

Embora estratégias de paralelização nem sempre permitam acelerar ou melhorar a eficácia de implementações seqüenciais de metaheurísticas, as implementações paralelas são muito robustas. Um levantamento recente sobre o estado-da-arte foi publicado por Cung et al. [23].

A maior parte das implementações paralelas da metaheurística GRASP seguem uma estratégia do tipo *multiple-walk-independent-thread* (ou, simplesmente, independente, não-cooperativa ou não-colaborativa), baseada na distribuição das iterações pelos processadores [6, 7, 32, 57, 60, 62, 67, 70, 71, 89]. Em geral, cada processador executa `MaxIter/p` iterações, onde `MaxIter` e `p` são, respectivamente, o número total de iterações do procedimento GRASP e o número de processadores. Cada processador possui uma cópia do algoritmo seqüencial, uma cópia dos dados do problema a ser resolvido e uma semente independente para gerar sua própria seqüência de números pseudo-aleatórios. Para evitar que os processadores encontrem a mesma solução, cada um deles usa uma seqüência diferente de números pseudo-aleatórios. Uma única variável global é necessária para armazenar a melhor solução dentre aquelas encontradas por todos os processadores. Um dos processadores atua como mestre, lendo e distribuindo os dados do problema, gerando as sementes que serão usadas pelo gerador de números pseudo-aleatórios em cada processador, distribuindo as iterações pelos processadores e coletando a melhor solução obtida por cada um deles. Como as iterações são completamente independentes e poucas informações são trocadas entre os processadores, acelerações lineares podem ser obtidas, desde que não exista um fator forte de desbalanceamento de carga. As iterações podem ser desigualmente distribuídas pelos processadores ou de acordo com suas demandas, de modo a melhorar o balancemanto de carga.

Martins et al. [62] implementaram uma heurística GRASP paralela para o problema de Steiner em grafos. A paralelização realizada era baseada na distribuição balanceada de 512 iterações pelos processadores, com o valor do parâmetro α que define a LRC escolhido aleatoriamente de forma uniforme dentro do intervalo [0.0,0.3] em cada iteração. O algoritmo foi implementado em C em um sistema IBM SP2 com 32 processadores, usando-

se a biblioteca MPI [63] para comunicação. As 60 instâncias das séries C, D e E da OR-Library [14] foram utilizadas nos experimentos computacionais. A implementação paralela obteve 45 soluções ótimas sobre as 60 instâncias. O erro em relação ao valor ótimo nunca foi superior a 4%. Acelerações quase lineares em relação à implementação seqüencial foram observadas para 2, 4, 8 e 16 processadores.

A técnica de reconexão por caminhos também pode ser usada em conjunto com implementações paralelas de GRASP. No caso da implementação da estratégia independente descrita por Aiex et al. [3, 4] para o problema de alocação com três índices, cada processador aplicava uma estratégia de pós-otimização por reconexão por caminhos a pares de soluções de elite armazenadas em um conjunto de elite centralizado no processador mestre. Resultados computacionais obtidos com o uso de MPI em um computador SGI Challenge com 28 processadores mostraram acelerações lineares.

Alvim e Ribeiro [6, 7] mostraram que estratégias independentes para a paralelização da metaheurística GRASP podem beneficiar-se sensivelmente de técnicas de balanceamento de carga, sempre que processadores heterogêneos são usados ou que o sistema paralelo seja compartilhado simultaneamente por diversos usuários. Neste caso, acelerações quase lineares podem ser obtidas pela distribuição heterogênea das iterações sobre os $\mathtt{p}$ processadores em $\mathtt{q} \geq \mathtt{p}$ blocos. Cada processador executa blocos de $\lceil \mathtt{MaxIter/q} \rceil$ iterações e informa o mestre cada vez que um deles termina de ser executado. O mestre interrompe os processadores escravos quando não há mais iterações a serem executadas e, em seguida, coleta a melhor solução encontrada. Processadores mais rápidos ou menos carregados executarão mais iterações que os demais. No caso da implementação paralela de GRASP para o problema da alocação de tráfego descrito em [75], esta estratégia dinâmica de balanceamento de carga permitiu reduções de até 15% em relação aos tempos observados para a estratégia estática, na qual as iterações são distribuídas igualmente pelos processadores.

A eficiência de implementações paralelas de metaheurísticas baseadas em estratégias independentes, que executam múltiplas cópias do mesmo algoritmo seqüencial, vem sendo analisada por diversos autores. Nestes estudos, um determinado valor alvo τ para a função objetivo é comunicado a todos os processadores que executam independentemente o algoritmo seqüencial. Todos os processadores são interrompidos logo que um deles encontra uma solução com valor tão bom quanto o alvo τ. A aceleração é dada pela razão entre os tempos necessários para encontrar uma solução com valor pelo menos tão bom quanto τ, usando o algoritmo seqüencial e a

implementação paralela com p processadores. É necessário algum cuidado, para evitar que duas iterações em processadores diferentes comecem com a mesma semente para o gerador de números pseudo-aleatórios. As acelerações são lineares para diferentes metaheurísticas, incluindo *simulated annealing* [28, 69]; algoritmos de busca local para o problema de caixeiro viajante [29]; busca tabu, desde que a busca comece de um ótimo local [13, 96]; e WalkSAT [93] em problemas difíceis da classe 3-SAT [50]. Segundo [98], esta observação pode ser explicada pelo fato de que a variável aleatória tempo para encontrar uma solução pelo menos tão boa quanto determinado valor alvo (ou, simplesmente, tempo para o alvo) segue uma distribuição exponencial.

Aiex et al. [3, 5] mostraram experimentalmente que os tempos de processamento observados para implementações de metaheurística GRASP também satisfazem a essa propriedade, mostrando que eles se ajustam a uma distribuição exponencial com dois parâmetros. Eles apresentaram as distribuições de probabilidade empírica e teórica observadas para cinco problemas de otimização combinatória distintos, incluindo conjunto independente máximo [32, 80]; problema quadrático da alocação [57, 81]; planarização de grafos [83, 87]; satisfabilidade ponderada máxima [82]; e recobrimento máximo [79]. Os mesmos resultados continuam válidos quando o GRASP é implementado em conjunto com um procedimento de pós-otimização por reconexão por caminhos para o problema de alocação com três índices [3, 4]. Seus experimentos computacionais envolveram 2400 execuções de procedimentos GRASP.

No caso de estratégias do tipo *multiple-walk-cooperative-thread* (ou, simplesmente, cooperativas ou colaborativas), os processadores executando em paralelo trocam e compartilham informações coletadas ao longo da trajetória que cada um deles investiga. Espera-se não apenas acelerar a convergência para a melhor solução, mas, também, encontrar melhores soluções do que através de estratégias independentes. O aspecto mais difícil de ser ajustado é a determinação da natureza das informações que serão compartilhadas ou trocadas para melhorar a busca, sem consumir muita memória ou tempo adicional para serem coletadas. Estratégias colaborativas podem ser implementadas usando reconexão por caminhos, combinando soluções de elite armazenadas em um conjunto de elite centralizado com os ótimos locais encontrados por cada processador ao final de cada iteração GRASP.

Canuto et al. [19] desenvolveram uma implementação paralela da estratégia de reconexão por caminhos, como uma etapa de pós-otimização

para melhorar uma heurística GRASP híbrida para o problema da árvore de Steiner com prêmios. Pares de soluções de elite provenientes de um conjunto de elite único e centralizado são distribuídos aos processadores, que realizam a reconexão por caminhos em paralelo. Resultados computacionais obtidos com uma implementação em MPI em um *cluster* de 32 processadores Pentium II 400 MHz mostraram acelerações lineares e melhorias significativas nas soluções encontradas pela heurística original.

4.2 Estratégia GRASP Seqüencial com Reconexão por Caminhos Bidirecional

A variante de GRASP com reconexão por caminhos bidirecional para um problema de minimização discutida no Capítulo 3 pode ser descrita conforme ilustrado na Figura 4.1.

O laço da estratégia GRASP com reconexão por caminhos (linhas 2 a 17) é executado até que uma solução com custo menor ou igual a solução alvo seja encontrada (linha 16).

Nas linhas 3 e 4 as fases de construção e busca local são realizadas. Se a solução y obtida pela busca local pode vir a pertencer ao conjunto das melhores soluções já encontradas, ela é inserida no conjunto de soluções de elite (linhas 5 e 6). Uma solução z é escolhida aleatoriamente do conjunto de soluções de elite (linha 7) e a reconexão por caminhos unidirecional é aplicada a solução inicial z até alcançar a solução guia y (linha 8). Se a solução y' encontrada pela reconexão por caminhos é melhor que a pior solução do conjunto de elite e diferente das demais soluções de elite, ela é inserida no conjunto de soluções de elite (linhas 9 e 10). Se y' é melhor que a melhor solução encontrada x^*, ela é atribuída a x^* (linha 11). Em seguida, a reconexão por caminhos unidirecional é aplicada a solução inicial y até alcançar a solução guia z (linha 12). Se a solução y'' encontrada pela reconexão por caminhos é melhor que a pior solução do conjunto de elite e diferente das demais soluções de elite, ela é inserida no conjunto de soluções de elite (linhas 13 e 14). Se y'' é melhor que a melhor solução encontrada x^*, ela é atribuída a x^* (linha 15).

```
procedimento GRASP+RC_2PNDP;
1   f* ← ∞; Pool ← ∅; achou_alvo ← falso;
2   enquanto achou_alvo = falso faça
3       Construir uma solução gulosa randomizada x (fase de construção);
4       Encontrar y aplicando busca local a x (fase de busca local);
5       se y deve pertencer ao conjunto de soluções de elite então
6           Inserir y no Pool;
7       Escolher aleatoriamente z ∈ Pool (z ≠ y) com probabilidade
        uniforme;
8       Atribuir a y' a melhor solução obtida aplicando reconexão por
        caminhos ao par (z, y);
9       se y' deve pertencer ao conjunto de soluções de elite então
10          Inserir y' no Pool;
11      se f(y') < f* então x* ← y'; f* ← f(y'); fim-se;
12      Atribuir a y'' a melhor solução obtida aplicando reconexão por
        caminhos ao par (y, z);
13      se y'' deve pertencer ao conjunto de soluções de elite então
14          Inserir y'' no Pool;
15      se f(y'') < f* então x* ← y''; f* ← f(y''); fim-se;
16      se f* ≤ alvo então achou_alvo ← verdadeiro;
17  fim-enquanto;
18  retornar x*;
fim GRASP+RC_2PNDP;
```

Figura 4.1: Pseudo-código da heurística GRASP seqüencial com reconexão por caminhos bidirecional.

4.3 Estratégia Paralela Independente

O esquema básico de paralelização do procedimento GRASP seqüencial com reconexão por caminhos bidirecional discutido na Seção 4.2 pode ser descrito conforme ilustrado na Figura 4.2. A implementação desenvolvida nesta tese baseou-se no paradigma de memória distribuída e usa passagem de mensagens para realizar a comunicação entre os p processadores utilizados no sistema.

De acordo com as taxonomias de [23, 98], esta abordagem paralela segue a estratégia independente. A estratégia GRASP paralela independente com reconexão por caminhos bidirecional é desenvolvida a partir da descrição da Figura 4.1. Cada processador participante do ambiente paralelo executa o procedimento da Figura 4.2. Cada um possui uma cópia dos dados do problema a ser resolvido e uma semente independente para gerar sua própria seqüência de números pseudo-aleatórios. Para evitar que os processadores encontrem a mesma solução, cada um deles usa uma seqüência diferente de números pseudo-aleatórios.

A comunicação é limitada à inicialização e ao término da execução em cada processador. Um dos processadores atua como mestre, gerando e distribuindo as sementes que serão usadas pelo gerador de números pseudo-aleatórios em cada processador e retornando uma solução tão boa quanto o valor alvo.

Nas linhas 1 e 2, são determinados os identificadores únicos de cada processo e o número de processadores. Na linha 3, o conjunto de soluções elite (*Pool*) é inicializado como vazio e as variáveis que testam se o custo de uma solução é melhor que o alvo ($achou\_alvo$) e se chegou alguma mensagem com o rótulo `Fim` (msg_fim) são inicializados com o valor booleano **falso**. Nas linhas 4 a 8, o processador identificado como `Mestre` gera todas as sementes que os demais processadores usarão para gerar as suas próprias seqüências de números pseudo-aleatórios (linha 9). O laço das linhas 10 a 27 realiza as iterações. Este laço é praticamente idêntico ao laço das linhas 2 a 17 da estratégia seqüencial. Os métodos de construção, busca local e reconexão por caminhos são utilizados da mesma forma que na heurística seqüencial.

Se pelo menos um processador encontra uma solução com custo menor ou igual ao alvo, ele sai do laço das linhas 10 a 27 e envia uma mensagem para todos os demais, indicando que uma solução tão boa quanto o valor alvo foi encontrada. Esta mensagem é identificada com o rótulo `Fim` (linhas 28 e 29). Além disto, ele envia esta solução para o processador identificado como `Mestre` (linhas 30 e 31).

Caso uma mensagem chegue a um processador com o rótulo `Fim`, as iterações deste processador são finalizadas (linhas 25 e 26).

O processador mestre recebe a solução tão boa quanto o alvo e a retorna como solução do problema (linhas 32 a 35).

4.4 Estratégia Paralela Colaborativa Centralizada

No caso de estratégias independentes, cada processador mantém seu próprio conjunto de soluções de elite. Já nas estratégias colaborativas centralizadas, o processador mestre mantém um conjunto de elite centralizado, coletando e distribuindo as soluções de elite quando solicitado a fazê-lo, além de executar as demais tarefas de distribuição de iterações, de geração e fornecimento de sementes, e de coleta da melhor solução.

Apesar de se perder o poder computacional de um processador que fica dedicado à gerência do conjunto de soluções de elite, existe o interesse

```
procedimento PAR_GRASP+RC_2PNDP_IND;
1  Obter o identificador de cada processador;
2  Obter a quantidade de processadores p participantes do sistema;
3  f* ← ∞; Pool ← ∅; achou_alvo ← falso; msg_fim ← falso;
4  se identificador = Mestre então
5      para i = 1, ..., p − 1 faça
6          Gerar uma Semente diferente das demais;
7          Enviar a Semente gerada para o processador i;
8      fim-para;
9  senão Receber a Semente enviada pelo processador Mestre; fim-se;
10 enquanto achou_alvo = falso e msg_fim = falso faça
11     Construir uma solução gulosa randomizada x (fase de construção);
12     Encontrar y aplicando busca local a x (fase de busca local);
13     se y deve pertencer ao conjunto de soluções de elite então
14         Inserir y no Pool;
15     Escolher aleatoriamente z ∈ Pool (z ≠ y) com probabilidade
       uniforme;
16     Atribuir a y' ser a melhor solução obtida aplicando reconexão por
       caminhos ao par (z, y);
17     se y' deve pertencer ao conjunto de soluções de elite então
18         Inserir y' no Pool;
19     se f(y') < f* então x* ← y'; f* ← f(y'); fim-se;
20     Atribuir a y'' ser a melhor solução obtida aplicando reconexão por
       caminhos ao par (y, z);
21     se y'' deve pertencer ao conjunto de soluções de elite então
22         Inserir y'' no Pool;
23     se f(y'') < f* então x* ← y''; f* ← f(y''); fim-se;
24     se f* ≤ alvo então achou_alvo ← verdadeiro;
25     se chegar alguma mensagem com o rotulo Fim então
26         msg_fim ← verdadeiro;
27 fim-enquanto;
28 se achou_alvo = verdadeiro então
29     Enviar uma mensagem para todos os processadores com o rotulo Fim;
30     Enviar a melhor solução x* para o processador Mestre;
31 fim-se;
32 se identificador = Mestre então
33     Receber a solução x*;
34     retornar x*;
35 fim-se;
fim PAR_GRASP+RC_2PNDP_IND;
```

Figura 4.2: Pseudo-código da heurística GRASP paralela independente com reconexão por caminhos bidirecional executada em cada processador.

de estudar como a troca de informações entre os processadores interfere no tempo de obtenção da melhor solução ou de uma solução com custo menor ou igual ao valor alvo.

Nesta seção, serão discutidas três abordagens distintas de estratégias colaborativas com a centralização do conjunto de soluções de elite: mestre gerenciando o conjunto de soluções de elite; mestre com dois processos; e mestre fazendo iterações após tratar um número pré-determinado de mensagens.

4.4.1 Mestre Gerenciando o Conjunto de Soluções de Elite

Nesta abordagem, chamada de estratégia colaborativa básica no restante desta tese, o processador mestre controla a entrada de soluções no conjunto de soluções de elite, a distribuição das soluções de elite, a geração e o fornecimento de sementes, e a coleta da melhor solução. Considerando-se que `p` processadores façam parte do ambiente paralelo, apenas `p - 1` processadores executam iterações.

Esta abordagem é ilustrada na Figura 4.3. A sua discussão será limitada apenas às suas diferenças em relação à estratégia independente, descrita na Seção 4.3.

As linhas 4 a 24 mostram o procedimento executado pelos processadores cujos identificadores são diferentes do `Mestre`, chamados processadores escravos. O laço das linhas 6 a 20 é praticamente idêntico ao laço das linhas 12 a 29 da estratégia independente, exceto pelos envios ao processo mestre das três soluções encontradas durante a computação de uma iteração, ocorridos nas linhas 9, 12 e 15. Na linha 9, o processador envia a solução obtida pelo procedimento de busca local e solicita uma solução de conjunto de elite na mensagem com rótulo `Pedido`. Na linha 10, o processador armazena a solução de elite recebida do processador mestre como resposta da mensagem com o rótulo `Pedido`. Nas linhas 12 e 15, o processador escravo envia as duas soluções encontradas pelo procedimento de reconexão por caminhos bidirecional nas mensagens com o rótulo `Insere`.

O critério de parada dos processadores escravos é o mesmo da estratégia independente, ilustrada na Figura 4.2.

As linhas 25 a 42 ilustram o procedimento executado pelo processo mestre. Este processo é, na verdade, um tratador de mensagens. De acordo com o rótulo da mensagem que chega ao mestre, alguma operação é realizada. Se o rótulo da mensagem for `Pedido`, isto indica que algum

processador está necessitando de uma solução de elite. Então, o mestre seleciona aleatoriamente uma solução do conjunto de elite e a envia ao processo requisitante (linhas 35 a 38). Além disto, o mestre verifica se a solução recebida nesta mensagem pode ser inserida no conjunto de soluções de elite. Se esta solução satisfaz os critérios de entrada no conjunto de elite, ela é inserida. Senão, é descartada (linhas 31 a 34). O processo de inserção de uma solução recebida no conjunto de soluções de elite é realizado, também, quando o rótulo da mensagem for `Insere`.

Caso uma mensagem chegue ao processador mestre com o rótulo `Fim` (linha 39), o laço das linhas 30 a 40 pára. Ele recebe, então, a solução tão boa quanto o alvo enviada por algum processador escravo, e a retorna como solução do problema (linhas 39 a 41).

4.4.2 Diminuição no Número de Soluções Enviadas ao Mestre

No início da computação da estratégia colaborativa básica, muitas soluções enviadas ao processador mestre serão inseridas no conjunto de soluções de elite, já que este conjunto está inicialmente vazio. Depois que estiver completo, será mais raro que uma solução seja inserida no conjunto de soluções de elite, fazendo com que o processador mestre trate muitas soluções que serão simplesmente descartadas.

Nesta seção serão discutidas duas maneiras distintas de diminuir o número de soluções enviadas ao processador mestre, que têm o objetivo de reduzir o processamento neste processador: a redução do número de soluções enviadas ao processador mestre e a verificação do custo de uma solução antes do seu envio.

4.4.2.1 Redução do Número de Soluções Enviadas

Uma das maneiras de tentar diminuir a quantidade de soluções que chegam ao processador mestre na estratégia colaborativa básica, ilustrada na Figura 4.3, e que não melhoram o seu conjunto de soluções de elite consiste em reduzir o número de soluções enviadas ao mestre. Duas possibilidades de redução foram estudadas:

- cada processo escravo envia, ao fim de uma iteração, a melhor solução encontrada por ele; e

```
procedimento PAR_GRASP+RC_2PNDP_COL;
1  Obter o identificador de cada processador;
2  Obter a quantidade de processadores p participantes do sistema;
3  f* ← ∞; Pool ← ∅; achou_alvo ← falso; msg_fim ← falso;
4  se identificador ≠ Mestre então
5      Receber a Semente enviada pelo processador Mestre;
6      enquanto achou_alvo = falso e msg_fim = falso faça
7          Construir uma solução gulosa randomizada x (fase de construção);
8          Encontrar y aplicando busca local a x (fase de busca local);
9          Enviar a solução y para o processador Mestre com o rotulo
           Pedido;
10         Atribuir a z a solução de elite enviada pelo Mestre;
11         Atribuir a y' a melhor solução obtida aplicando reconexão por
           caminhos ao par (z, y);
12         Enviar a solução y' para o processador Mestre com o rotulo
           Insere;
13         se f(y') < f* então x* ← y'; f* ← f(y'); fim-se;
14         Atribuir a y'' a melhor solução obtida aplicando reconexão por
           caminhos ao par (y, z);
15         Enviar a solução y'' para o processador Mestre com o rotulo
           Insere;
16         se f(y'') < f* então x* ← y''; f* ← f(y''); fim-se;
17         se f* ≤ alvo então achou_alvo ← verdadeiro;
18         se chegar alguma mensagem com o rotulo Fim então
19             msg_fim ← verdadeiro;
20     fim-enquanto;
21     se achou_alvo = verdadeiro então
22         Enviar uma mensagem para todos os processadores com o rotulo
23         Fim; Enviar a melhor solução x* para o processador Mestre;
24     fim-se;
25 senão
26     para i = 1, ..., p − 1 faça
27         Gerar uma Semente diferente das demais;
28         Enviar a Semente gerada para o processador i;
29     fim-para;
30     enquanto msg_fim = falso faça
31         se a mensagem recebida tiver o rotulo Insere ou Pedido então
32             Atribuir a z a solução recebida na mensagem;
33             se z deve pertencer ao conjunto de soluções de elite então
34                 Inserir z no Pool;
35             se a mensagem recebida tiver o rotulo Pedido então
36                 Escolher aleatoriamente uma solução do Pool;
37                 Enviar esta solução para o processador requisitante;
38             fim-se;
39         senão msg_fim ← verdadeiro; Receber a solução x*; fim-se;
40     fim-enquanto;
41     retornar x*;
42 fim-se;
fim PAR_GRASP+RC_2PNDP_COL;
```

Figura 4.3: Pseudo-código da estratégia paralela colaborativa com reconexão bidirecional por caminhos com o mestre gerenciando o conjunto de soluções de elite.

– cada processo escravo envia, durante uma iteração, duas soluções encontradas por ele. A primeira é a solução obtida após o procedimento de busca local. A segunda é a melhor solução encontrada ao final do procedimento bidirecional de reconexão por caminhos.

Estas alternativas têm duas vantagens em relação ao método de envio utilizado pela estratégia colaborativa básica da Figura 4.3: (a) a diminuição do número de mensagens trafegando na rede de comunicação e (b) a diminuição do tempo de resposta do processador mestre para requisições feitas pelos escravos (isto é, a diminuição do tempo de espera de cada escravo pelo processamento no mestre).

4.4.2.2 Verificação do Custo antes do Envio da Solução

Outra maneira de diminuir o número de soluções que chegam ao processador mestre na estratégia colaborativa básica, ilustrada na Figura 4.3, e que não melhoram o seu conjunto de soluções de elite é analisar, no processador escravo, o custo de uma solução antes do seu envio ao processador mestre.

Nem todas soluções encontradas pelos escravos são enviadas para o mestre, como na estratégia colaborativa básica descrita na Seção 4.4.1. Uma solução somente é enviada, se o seu custo for menor que o da pior solução de elite [12]. Para saber se o custo de uma solução é menor que o das soluções do conjunto de elite, cada processo escravo deve manter uma variável própria, informando qual é o maior custo dentre as soluções de elite. As comparações entre estes custos são realizadas no próprio processo escravo, e a cada comunicação feita entre os processos mestre e escravo, esta variável deve ser atualizada, a fim de reduzir problemas de inconsistência a respeito do custo da pior solução existente no conjunto de elite.

A inclusão de uma solução no conjunto de soluções de elite é um procedimento que envolve bastante troca de mensagens entre os processos mestre e escravo: se o custo da solução encontrada pelo processador escravo for maior que o valor da sua variável que informa qual é o maior custo das soluções de elite, esta solução não é enviada ao processador mestre. Senão, o processador escravo envia o custo desta solução para o processador mestre. Se o processador mestre verificar que, realmente, o custo da solução é menor que o maior custo das soluções de elite (este passo deve ser realizado para reduzir possíveis inconsistências nas variáveis próprias dos processadores escravos), o processador mestre envia uma mensagem pedindo

ao processador escravo que lhe envie a solução. Senão, o mestre envia uma mensagem ao processador escravo a rejeitando. O processador escravo, por sua vez, verifica o rótulo da mensagem recebida do processador mestre. Ele somente envia a solução ao mestre se receber uma mensagem cujo rótulo é uma solicitação de solução.

As vantagens discutidas na Seção 4.4.2.1 também são válidas para o método de verificação do custo da solução antes do seu envio:

(a) diminuição do gargalo existente no processador mestre em estratégias paralelas centralizadas: no decorrer da execução de uma estratégia colaborativa com verificação do custo da solução, menos mensagens devem ser tratadas pelo processador mestre, diminuindo, assim, o tempo de resposta das requisições feitas pelos processadores escravos; e

(b) diminuição do número de mensagens trafegando na rede de comunicação: se uma solução encontrada pelo processador escravo tem custo maior do que o valor da sua variável local, o processador escravo não troca mensagens com o processador mestre.

4.4.3 Mestre com Dois Processos

Nesta abordagem paralela colaborativa centralizada, o processador mestre executa dois processos distintos [100]. O processo `P`, responsável por gerenciar o conjunto de soluções de elite, é análogo ao processo mestre discutido na Seção 4.4.1. O processo `Q`, responsável pela tentativa de computação da melhor solução, é análogo aos processos escravos descritos na Seção 4.4.1. Os demais processadores executam processos escravos, semelhantes ao processo escravo da estratégia da Figura 4.3. É importante enfatizar que o processo `Q` é um processo escravo no processador mestre.

A fim de diminuir o número de mensagens chegando ao processo `P`, os processos escravos e o processo `Q` utilizam o método de verificação do custo das soluções obtidas ao longo de uma iteração GRASP, antes de seu envio ao processador `P`, como discutido na Seção 4.4.2.2.

Pelo fato do processo mestre ser um tratador de mensagens em estratégias colaborativas centralizadas, a prioridade de execução do processo `P` é maior que a prioridade de execução de `Q`. Esta desigualdade de prioridades faz com que se uma mensagem chegar ao processo `P` e este não estiver

sendo executado no processador, então `Q` será interrompido e a execução de `P` será reinicializada.

Conceitualmente, ter dois processos distintos com prioridades diferentes executando em um único processador é uma tentativa de não deixar que este processador fique ocioso. No início da computação, o processo `P` receberá muitas mensagens de inserção de soluções no conjunto de soluções de elite, já que este conjunto está inicialmente vazio. Conseqüentemente, nesta fase este processo executará muito mais tempo que o processo `Q`, devido à sua maior prioridade em relação a `Q`. Depois que este conjunto estiver completo, será mais raro que um processo escravo envie uma solução a ser inserida no conjunto de soluções de elite, fazendo com que o processo `P` fique à espera de mensagens e o processo `Q` comece a executar mais freqüentemente.

4.4.4
Mestre Fazendo Iterações em um Único Processo

Nesta abordagem, o processador mestre é responsável por realizar todas as tarefas descritas na Seção 4.4.1. Além disto, ele também faz iterações GRASP computando soluções. A descrição desta estratégia é simples: após o processo mestre tratar um número pré-determinado de mensagens, ele realiza uma iteração GRASP. Se a melhor solução encontrada por ele nesta iteração tiver o custo menor ou igual ao alvo, ele pára. Senão, ele volta a tratar mensagens.

Os demais processadores do sistema paralelo executam processos escravos, análogos ao processo escravo da Seção 4.4.1. Contudo, diferentemente da estratégia da Seção 4.4.1, cada processo escravo envia, somente, a sua melhor solução encontrada ao longo de uma iteração para o mestre, a fim de diminuir o número de mensagens que chegam ao processador mestre, conforme discutido na Seção 4.4.2.1.

4.5
Estratégia Paralela Colaborativa Distribuída

Neste tipo de estratégia, cada processador mantém sua própria cópia atualizada do conjunto de soluções de elite. Toda vez que um processador encontra uma solução que pode fazer parte do seu conjunto de soluções de elite, ele divulga esta solução para os demais processadores.

Existe um processador identificado como mestre com as mesmas funções do processador mestre da estratégia discutida na Seção 4.3: ele realiza iterações, além de gerar e fornecer as sementes, de distribuir iterações e de coletar as melhores soluções. A descrição desta estratégia é mostrada na Figura 4.4. Ela é baseada na estratégia paralela independente descrita na Seção 4.3. Sua discussão será limitada, apenas, às diferenças em relação à estratégia independente.

Nas linhas 13 a 16, é realizada a distribuição de uma solução encontrada por um processador, que deve fazer parte de seu conjunto de elite, para os demais processadores. Esta operação se repete nas linhas 19 a 22 e nas linhas 25 a 28. Nas linhas 33 a 36, ocorre o tratamento das mensagens com as soluções que devem ser guardadas nos conjuntos de elite dos processadores. É importante salientar que, apesar dos conjuntos de soluções de elite possuírem as mesmas soluções, não há garantia de que estes conjuntos apresentem todas as soluções na mesma ordem. A ordem é relevante porque duas execuções desta estratégia sobre um problema com os mesmos parâmetros podem levar a diferentes soluções.

Esta estratégia causa um aumento considerável de tráfego de mensagens entre os processadores, quando comparada à estratégia colaborativa centralizada básica.

4.6 Resultados Computacionais

A fim de analisar o comportamento das estratégias paralelas descritas nas seções anteriores, serão ilustrados os tempos gastos nas execuções das estratégias independente e colaborativas. Estas estratégias foram implementadas em C, usando a versão egcs-2.91.66 do compilador gcc e a implementação do MPI LAM 6.3.2 para a comunicação. Todas as medidas de tempo foram realizadas pela rotina de tempo MPI_WTIME, pertencente ao conjunto de procedimentos padrões MPI.

O comportamento das estratégias paralelas é bastante uniforme sobre diferentes instâncias. Ilustram-se os resultados obtidos em uma instância de 100 nós, 4950 arestas (isto é, um grafo completo) e 1000 pares origem-destino. O valor alvo 683 foi escolhido como sendo a média de 15 soluções encontradas em 15 execuções de GPRfb com 3200 iterações com 15 sementes distintas.

Os experimentos computacionais foram realizados em um *cluster* de 32 processadores Pentium II 400MHz com 32 Mbytes, usando a implementação

```
procedimento PAR_GRASP+RC_2PNDP_DIST;
1  Obter o identificador de cada processador;
2  Obter a quantidade de processadores p participantes do sistema;
3  f* ← ∞; Pool ← ∅; achou_alvo ← falso; msg_fim ← falso;
4  se identificador = Mestre então
5      para i = 1, ..., p − 1 faça
6          Gerar uma Semente diferente das demais;
7          Enviar a Semente gerada para o processador i;
8      fim-para;
9  senão Receber a Semente enviada pelo processador Mestre; fim-se;
10 enquanto achou_alvo = falso e msg_fim = falso faça
11     Construir uma solução gulosa randomizada x (fase de construção);
12     Encontrar y aplicando busca local a x (fase de busca local);
13     se y deve pertencer ao conjunto de soluções de elite então
14         Inserir y no Pool;
15         Enviar y para todos os processadores com o rotulo Insere;
16     fim-se;
17     Escolher aleatoriamente z ∈ Pool (z ≠ y) com probabilidade
       uniforme;
18     Atribuir a y' a melhor solução obtida aplicando reconexão por
       caminhos ao par (z, y);
19     se y' deve pertencer ao conjunto de soluções de elite então
20         Inserir y' no Pool;
21         Enviar y' para todos os processadores com o rotulo Insere;
22     fim-se;
23     se f(y') < f* então x* ← y'; f* ← f(y'); fim-se;
24     Atribuir a y'' a melhor solução obtida aplicando reconexão por
       caminhos ao par (y, z);
25     se y'' deve pertencer ao conjunto de soluções de elite então
26         Inserir y'' no Pool;
27         Enviar y'' para todos os processadores com o rotulo Insere;
28     fim-se;
29     se f(y'') < f* então x* ← y''; f* ← f(y''); fim-se;
30     se f* ≤ alvo então achou_alvo ← verdadeiro;
31     se chegar alguma mensagem com o rotulo Fim então
32         msg_fim ← verdadeiro;
33     se chegar alguma mensagem com o rotulo Insere então
34             Atribuir a z a solução recebida na mensagem;
35             Inserir z no Pool;
36     fim-se;
37 fim-enquanto;
38 se achou_alvo = verdadeiro então
39     Enviar uma mensagem para todos os processadores com o rotulo
40     Fim; Enviar a melhor solução x* para o processador Mestre;
41 fim-se;
42 se identificador = Mestre então
43     Receber a solução x*;
44     retornar x*;
45 fim-se;
fim PAR_GRASP+RC_2PNDP_DIST;
```

Figura 4.4: Pseudo-código da estratégia paralela distribuída com reconexão por caminhos bidirecional.

Red Hat 6.2 do sistema operacional Linux 2.2.14-5.0 e o *switch* IBM 8274 com 96 portas com velocidade de 10 Mbits/s para a comunicação entre os processadores. Os experimentos foram realizados em modo exclusivo, isto é, as execuções foram obtidas sem o compartilhamento do poder computacional dos nós com outra aplicação.

4.6.1 Comparação da Estratégia Colaborativa Básica com a Independente

As análises realizadas no restante deste capítulo são baseadas em distribuições de probabilidade empíricas do tempo para alcançar um valor alvo. Sempre que a curva relativa à distribuição empírica de uma estratégia estiver à esquerda da curva relacionada à distribuição empírica de uma outra estratégia, isto ilustra que a primeira estratégia possui um desempenho melhor do que da outra, porque ela obtém melhores soluções com maior probabilidade em tempos de processamento menores.

Os resultados obtidos sobre 2, 4 e 8 processadores para a instância e o valor alvo pré-definidos são ilustrados nas Figuras 4.5, 4.6 e 4.7, respectivamente, considerando-se 200 execuções com sementes iniciais distintas em cada caso. Na Figura 4.5, com apenas dois processadores, observa-se que a estratégia independente (a curva rotulada independente) tem um desempenho melhor que a estratégia colaborativa (a curva rotulada colaborativa (3 solucoes)). Esta predominância da estratégia independente sobre a colaborativa para dois processadores é facilmente explicável: a estratégia independente faz uso de dois processadores para executar as iterações. Já a estratégia colaborativa utiliza, na realidade, um único processador na computação de soluções.

A medida que o número de processadores aumenta, esta predominância se inverte. Tempos decorridos progressivamente menores são observados, com o deslocamento das duas curvas para a esquerda. Mesmo contando com apenas `p - 1` dos `p` processadores para executar iterações, o desempenho da estratégia colaborativa melhora progressivamente, graças à execução de um menor número de iterações e à cooperação entre os processadores. Observa-se que, já com quatro processadores, as duas curvas se interceptam. O desempenho da estratégia colaborativa com oito processadores já é sensivelmente superior à independente.

Quando são utilizados 16 processadores, a quantidade de memória necessária para a execução da estratégia colaborativa básica ultrapassa a quantidade de memória disponível no *cluster*. Isto se deve ao elevado

número de mensagens que chegam ao processador mestre. Na estratégia colaborativa básica (Seção 4.4.1), cada processador escravo envia, em uma única iteração, três soluções completas para o mestre. Com o aumento do número de processadores no ambiente paralelo, o número de mensagens aumenta consideravelmente. O processador mestre não consegue tratar todas as mensagens que chegam a ele. A biblioteca MPI armazena em memória estas mensagens. Este processo de armazenamento é realizado até que o processador mestre esteja pronto para tratar estas mensagens guardadas ou até que haja uma falha no computador.

Novas variantes desta estratégia para tentar diminuir o número de soluções enviadas ao mestre foram implementadas e serão discutidas nas próximas seções. Para a análise destas variantes foram usados oito processadores, a instância e o valor alvo pré-determinados. Nos resultados apresentados ao longo das próximas seções, são consideradas, sempre, 200 execuções com sementes iniciais distintas em cada estratégia implementada.

4.6.2 Comparação de Estratégias Colaborativas Diminuindo o Número de Soluções Enviadas

A fim de diminuir a quantidade de soluções enviadas ao mestre na estratégia colaborativa da Seção 4.4.1, as duas variantes descritas na Seção 4.4.2.1 (chamadas de colaborativa (1 solucao) e colaborativa (2 solucoes) na Figura 4.8, respectivamente) foram implementadas e avaliadas. Os resultados computacionais mostrados na Figura 4.8 ilustram que a diminuição do número de soluções enviadas ao mestre não é uma alternativa satisfatória. A queda na qualidade das soluções, devida à redução do envio das soluções obtidas pelo procedimento de reconexão por caminhos, faz com que estas variantes tenham um desempenho pior do que o desempenho da estratégia colaborativa básica.

4.6.3 Estudo do Envio do Custo e da Solução Separadamente

Nesta seção, a idéia de verificação do custo da solução antes do seu envio ao mestre (Seção 4.4.2.2) será implementada juntamente com o conceito de diminuição de soluções enviadas ao mestre (Seção 4.4.2.1). Isto é feito com o propósito de tentar melhorar o desempenho das variantes com envio de uma, duas e três soluções, além de evitar os problemas de

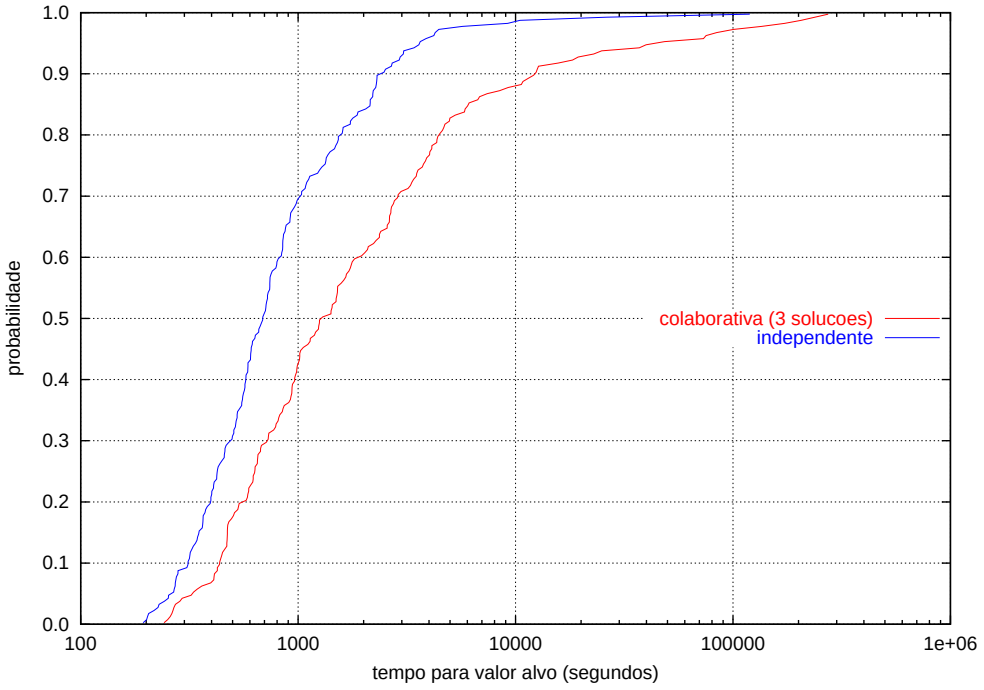

Figura 4.5: Comparação entre as estratégias colaborativa e independente usando 2 processadores.

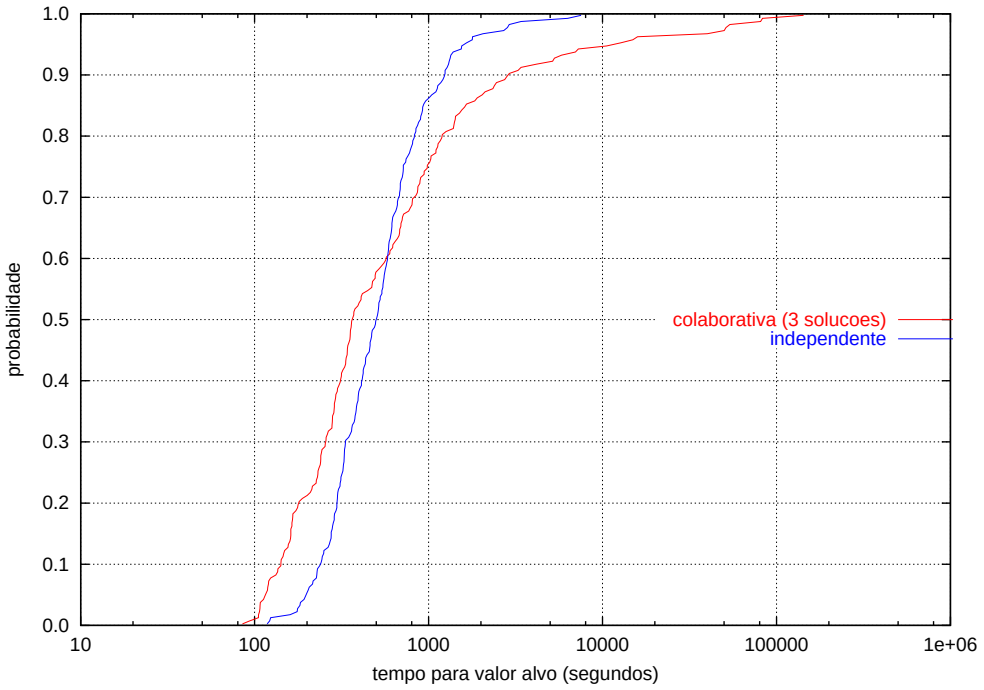

Figura 4.6: Comparação entre as estratégias colaborativa e independente usando 4 processadores.

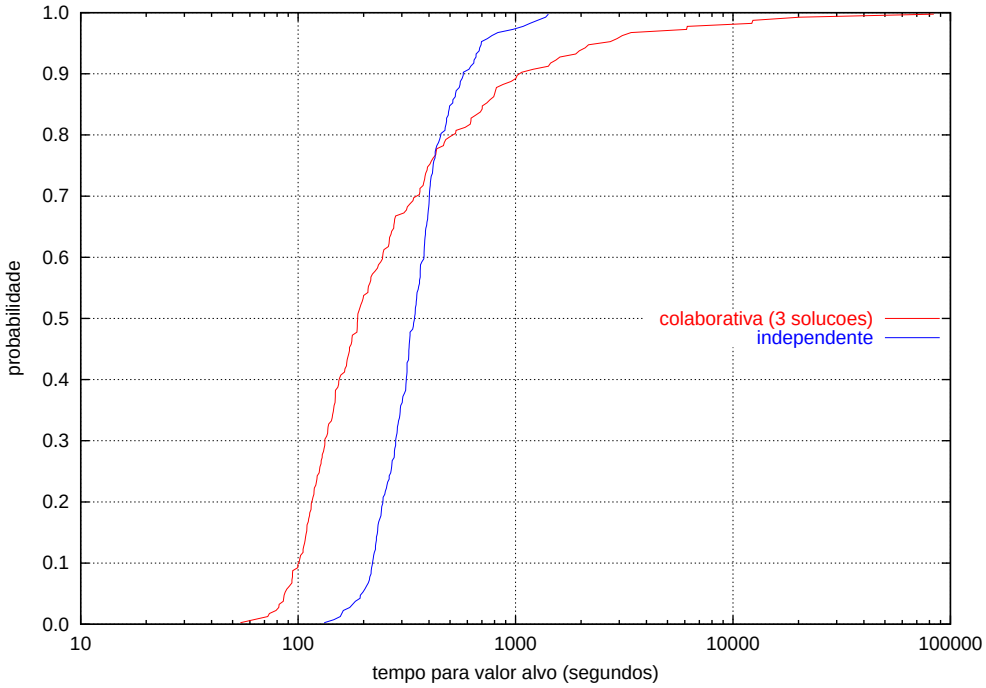

Figura 4.7: Comparação entre as estratégias colaborativa e independente usando 8 processadores.

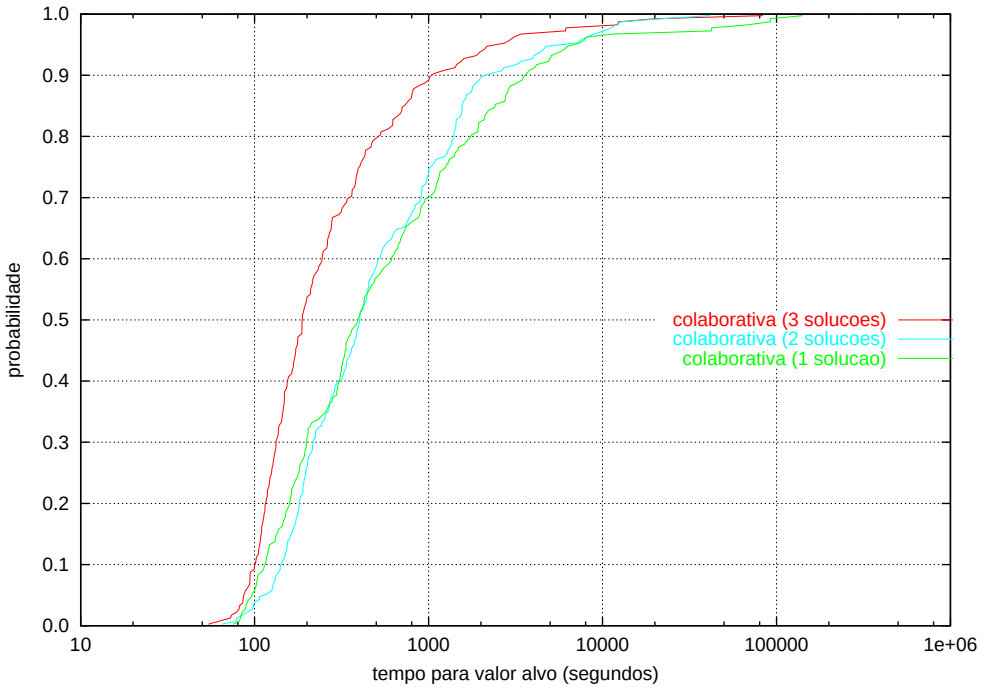

Figura 4.8: Comparação entre as estratégias colaborativas variando o número de soluções enviadas ao mestre.

memória obtidos pela estratégia colaborativa básica quando foram usados 16 processadores.

4.6.3.1 Análise das Estratégias Colaborativas com o Envio de Três Soluções

Uma outra alternativa para diminuir a quantidade de soluções enviadas ao mestre na estratégia colaborativa básica é verificar o custo da solução antes de enviá-la, conforme discutido na Seção 4.4.2.2: só ocorre o envio de uma solução para o processo mestre quando seu custo for menor que o custo da pior solução de elite.

Pelos resultados computacionais mostrados na Figura 4.9, a verificação do custo da solução antes do seu envio ao mestre é uma alternativa satisfatória. O desempenho da nova variante (rotulada colaborativa (3 solucoes + 3 custos)) é melhor do que o desempenho da estratégia original (colaborativa (3 solucoes)). Neste caso, a variante colaborativa básica será descartada e substituída pela nova variante nos estudos que se seguem.

Nas próximas análises, toda vez que o desempenho de uma estratégia for melhor do que de outra, a pior delas será descartada.

4.6.3.2 Análise das Estratégias Colaborativas com o Envio de Duas Soluções

Como foi ilustrado na Figura 4.8, a redução do número de soluções enviadas ao mestre pelos processos escravos não é uma boa estratégia, porque o desempenho da estratégia colaborativa básica é melhor do que o desempenho das variantes com envio de uma ou duas soluções.

A fim de tentar melhorar o desempenho da estratégia colaborativa com envio de duas soluções, foi implementada uma nova estratégia, baseada na idéia de verificação do custo da solução antes seu do envio (rotulada colaborativa (2 solucoes + 2 custos) na Figura 4.10) como discutido na Seção 4.4.2.2.

Os resultados computacionais mostrados na Figura 4.10 ilustram que as duas variantes com envio de duas soluções não são estratégias satisfatórias. Seus desempenhos são sempre piores que o desempenho da estratégia colaborativa com a verificação do custo antes do envio das três soluções. Então, estas variantes com envio de duas soluções serão descartadas dos próximos estudos.

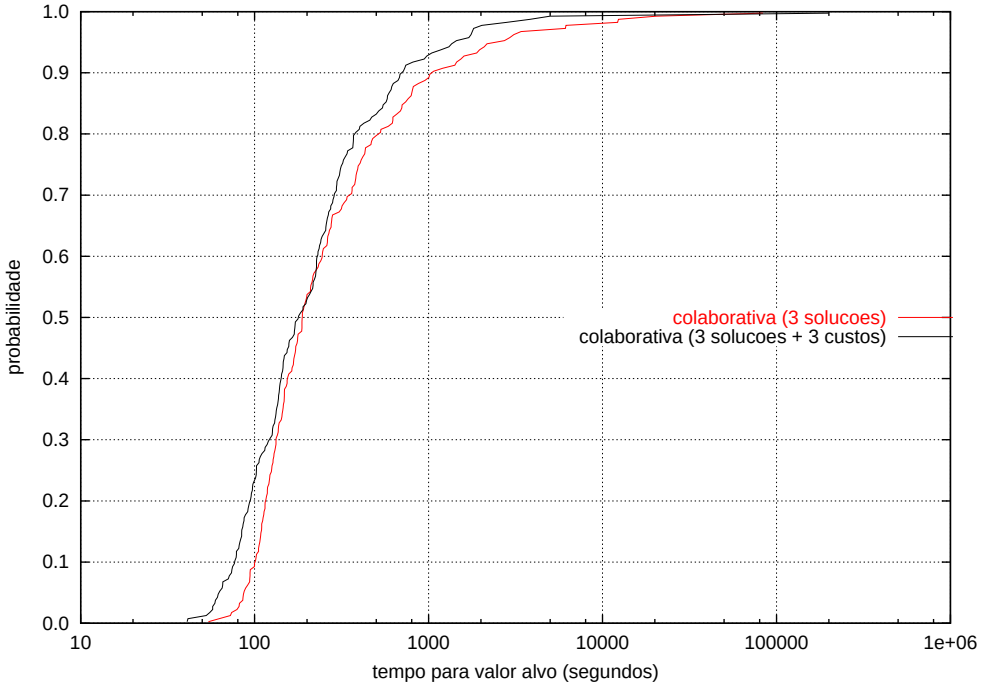

Figura 4.9: Comparação entre as estratégias colaborativas com envio de três soluções a cada iteração, com custos e soluções enviadas separadamente.

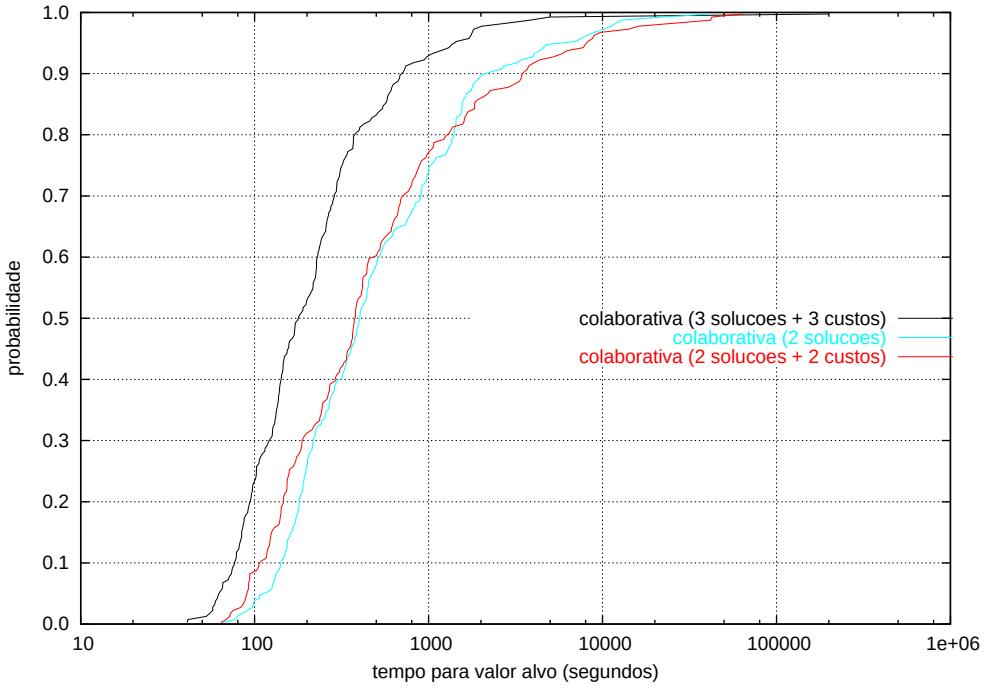

Figura 4.10: Comparação entre as estratégias colaborativas com envio de duas soluções a cada iteração, com custos e soluções enviadas separadamente.

4.6.3.3 Análise das Estratégias Colaborativas com o Envio de Uma Solução

Com o objetivo de tentar melhorar o desempenho da estratégia colaborativa com o envio, ao mestre, da melhor solução encontrada ao fim de uma iteração, quatro estratégias foram implementadas e avaliadas:

(1) estratégia baseada na idéia de verificação do custo da solução antes de seu envio, como discutido na Seção 4.4.2.2 (rotulada colaborativa (1 solucao + 1 custo) na Figura 4.11);

(2) estratégia em que cada processo escravo aplica o procedimento de busca local sobre a melhor solução encontrada antes de enviá-la ao mestre (rotulada colaborativa (1 solucao + bl) na Figura 4.11);

(3) estratégia similar à que foi discutida na Seção 4.4.2.1, acrescida do conceito de tempo de corte de execução, descrito na Seção 2.2.4.4: se o processador mestre não inserir uma solução no conjunto de soluções de elite, após um número limitado de soluções depois da última inserção, o processador mestre envia uma mensagem a todos os escravos ordenando que recomecem suas execuções com as sementes iniciais e com o conjunto de soluções de elite atual (rotulada colaborativa (1 solucao + tc) na Figura 4.11). Para implementar esta estratégia, é necessário que no processador mestre exista uma variável que indique quantas soluções foram rejeitadas desde a última aceita. Os processos escravos desta estratégia são idênticos aos da estratégia discutida na Seção 4.4.2.1;

(4) estratégia baseada na idéia de verificação do custo da solução antes de seu envio, como discutido na Seção 4.4.2.2, acrescida do conceito de tempo de corte de execução, descrito na Seção 2.2.4.4 (rotulada colaborativa (1 solucao + 1 custo + tc) na Figura 4.11). O controle de parada da execução nesta estratégia é um pouco diferente do mecanismo descrito no item (3), uma vez que, nesta estratégia, já existe uma filtragem de soluções enviadas ao mestre. Todos os processos escravos têm variáveis próprias que contam o número de vezes que eles não obtiveram sucesso ao tentar enviar suas melhores soluções ao mestre.

Os resultados computacionais mostrados na Figura 4.11 ilustram que o conceito de tempo de corte é uma técnica bastante promissora porque melhora, sensivelmente, os resultados das estratégias com envio de uma solução que não utilizam este conceito. Além disto, estes resultados ilustram

que a estratégia que usa o procedimento de busca local como um método de intensificação de soluções não é uma boa estratégia, porque todas as outras quatro estratégias com envio de uma solução têm desempenhos melhores do que o dela.

Estes resultados ilustram, também, que as cinco variantes com envio de uma solução não são estratégias satisfatórias. Os seus desempenhos são sempre piores que o desempenho da estratégia colaborativa com a verificação do custo antes do envio das três soluções. Então, estas variantes com envio de uma solução serão descartadas dos estudos que se seguem.

4.6.4 Comparação das Melhores Estratégias Colaborativas com a Independente

Para verificar como se comportam as estratégias colaborativas quando o processo mestre também executa iterações, quatro variantes foram implementadas e avaliadas:

(1) estratégia baseada na idéia do processador mestre ter dois processos, discutida na Seção 4.4.3 (rotulada processos pq na Figura 4.12);

(2) estratégia baseada na idéia do processador mestre ter dois processos, discutida na Seção 4.4.3, acrescida do conceito de tempo de corte de execução, como descrito no item (4) da Seção 4.6.3.3 (rotulada processos pq (tc) na Figura 4.12);

(3) estratégia baseada na idéia do processador mestre realizar iterações depois de tratar um número limitado de mensagens, descrita na Seção 4.4.4 (rotulada colaborativa (iteracoes GRASP no mestre) na Figura 4.12);

(4) estratégia colaborativa distribuída, descrita na Seção 4.5 (rotulada distribuida na Figura 4.12).

Os resultados computacionais mostrados na Figura 4.12 ilustram que a estratégia em que o processo mestre faz uma iteração toda vez que trata um número limitado de mensagens (rotulada colaborativa (iteracoes GRASP no mestre)) não é uma boa alternativa, porque a estratégia independente tem um desempenho melhor que esta variante. Esta variante será descartada.

Os resultados anteriores, resumidos na Figura 4.13, ilustram que as estratégias colaborativas com envio de três soluções têm desempenhos sensivelmente melhores que a estratégia independente, graças à cooperação

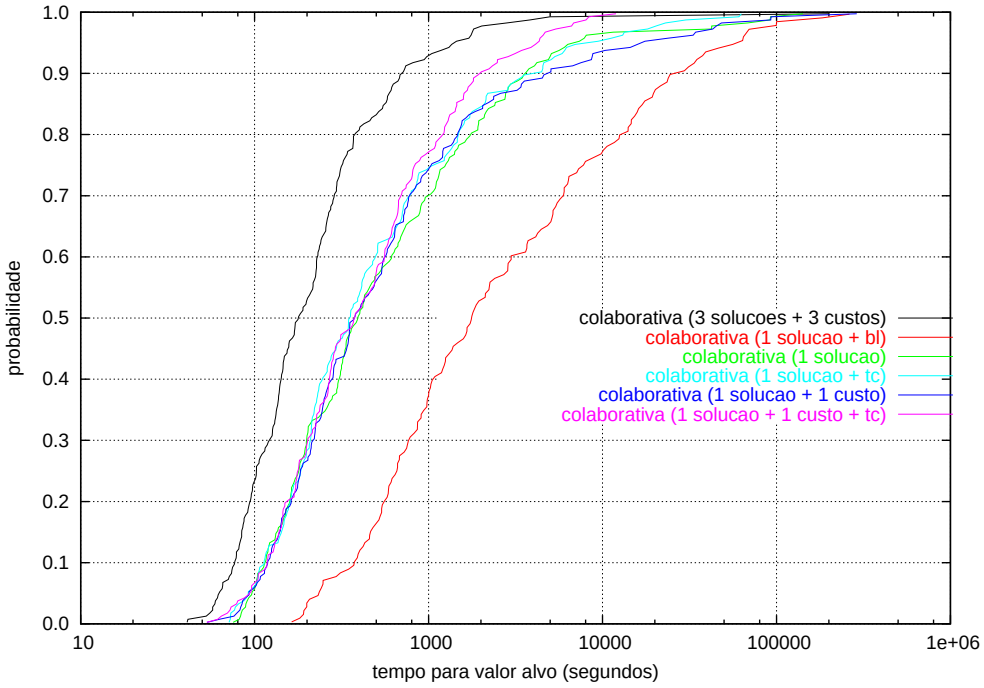

Figura 4.11: Comparação entre as estratégias colaborativas com envio de uma solução a cada iteração.

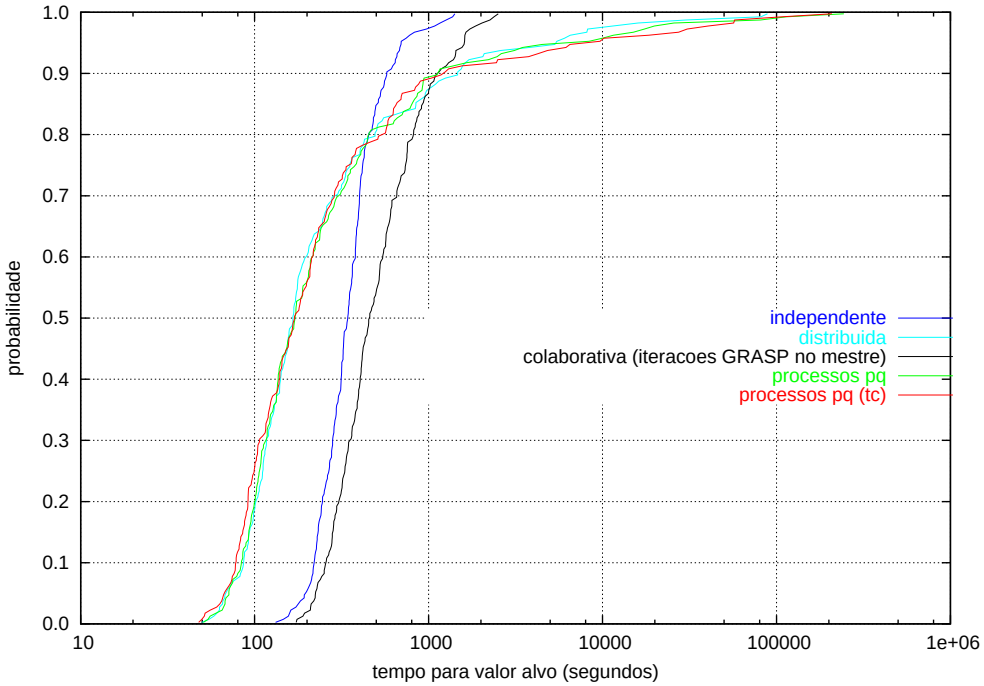

Figura 4.12: Comparação entre as estratégias colaborativas em que o processador mestre trabalha.

entre os processadores. Estes resultados ilustram, também, a impossibilidade de garantir superioridade entre todas as estratégias colaborativas que enviam três soluções ao mestre, porque as curvas de execução são bastante similares, indicando que os tempos de processamento para o encontro de uma solução com valor de custo tão bom quanto o alvo são bastante parecidos.

A estratégia colaborativa com envio de três custos e três soluções separadamente em que o processador mestre realiza, unicamente, a gerência do conjunto de soluções de elite, será usada nas próximas comparações com a estratégia independente.

4.6.5 Comparação entre a Estratégia Independente e a Colaborativa com Envio de Três Custos e Três Soluções Separadamente

Os resultados obtidos por estas duas estratégias sobre 2, 4, 8, 16 e 32 processadores para a instância e o valor alvo pré-definidos são apresentados nas Figuras 4.14, 4.15, 4.16, 4.17 e 4.18, respectivamente. As observações feitas para as curvas relativas à estratégia colaborativa básica nas Figuras 4.5, 4.6 e 4.7 da Seção 4.6.1 são válidas para as curvas relativas à estratégia colaborativa com envio de três custos e três soluções separadamente nas Figuras 4.14, 4.15 e 4.16, respectivamente.

O desempenho da estratégia colaborativa para 16 processadores é sobremaneira superior ao da independente. Com 32 processadores, esta superioridade se mantém, mas há uma queda de desempenho da estratégia colaborativa. Este fato pode ser melhor observado analisando as Figuras 4.19 e 4.20. Esta queda de desempenho se deve ao aumento do número de mensagens trafegando na rede, devido ao aumento considerável do número de processadores no ambiente paralelo.

Examinando a Figura 4.19, observa-se que quanto maior é o número de processadores no sistema, menor é o tempo de processamento gasto pela estratégia independente para obter soluções tão boas quanto o alvo. Isto também acontece com a estratégia colaborativa utilizando até 16 processadores, como indicado na Figura 4.20. Entretanto, o desempenho da estratégia colaborativa com envio de três custos e três soluções com 32 processadores é pior que o desempenho da mesma estratégia com 16 processadores.

Aceleração pode ser definida como sendo o resultado da divisão do tempo da estratégia independente sobre um único processador pelo tempo

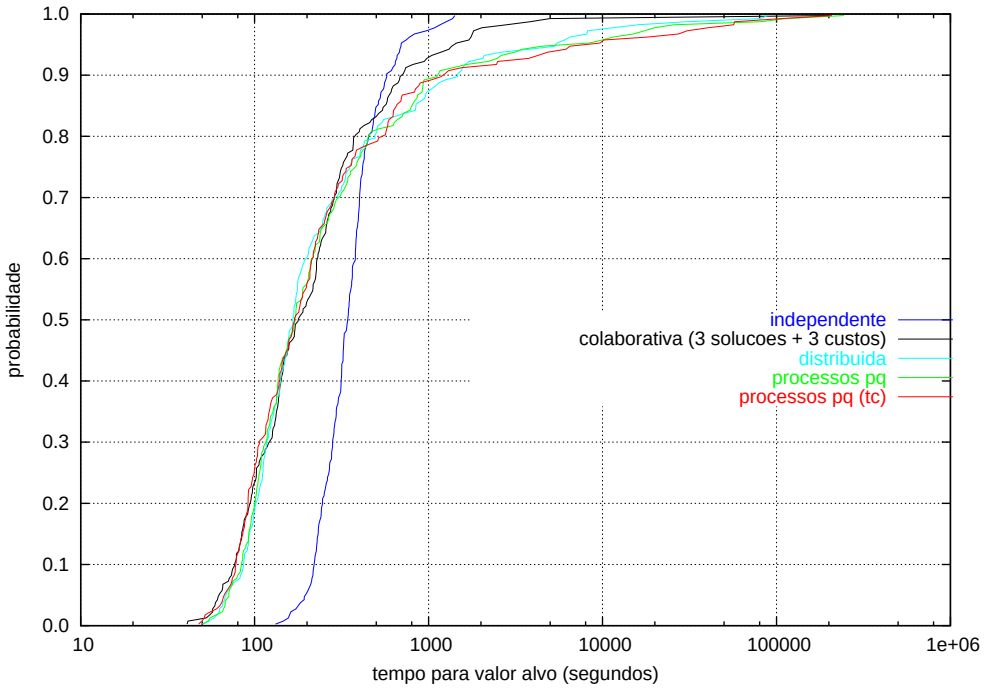

Figura 4.13: Comparação entre as melhores estratégias colaborativas e a independente.

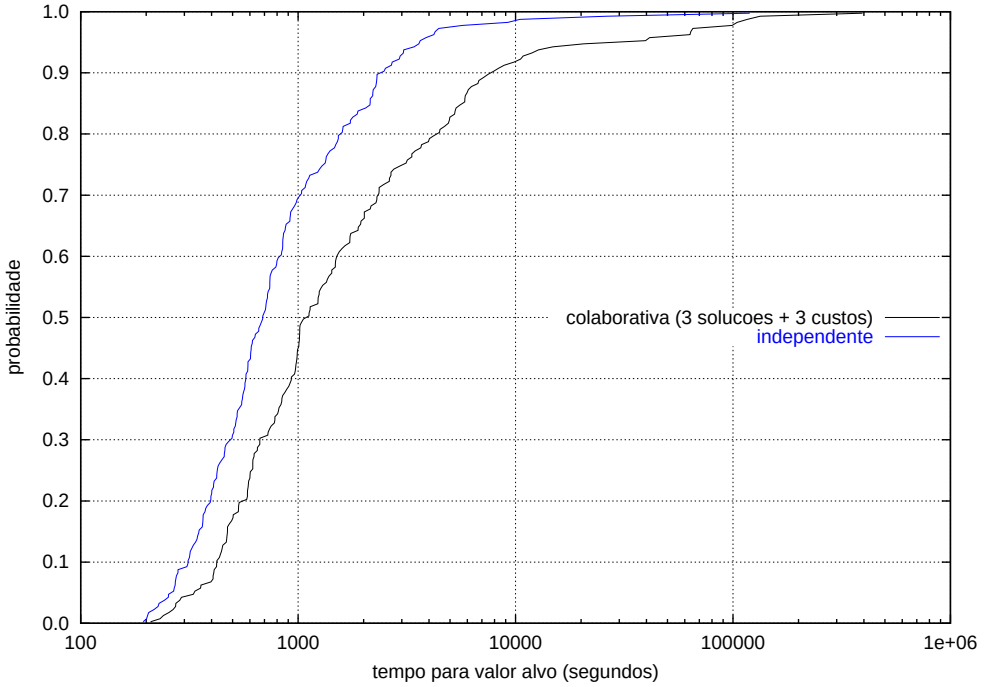

Figura 4.14: Comparação entre a estratégia colaborativa com envio de três custos e três soluções separadamente e a estratégia independente usando 2 processadores.

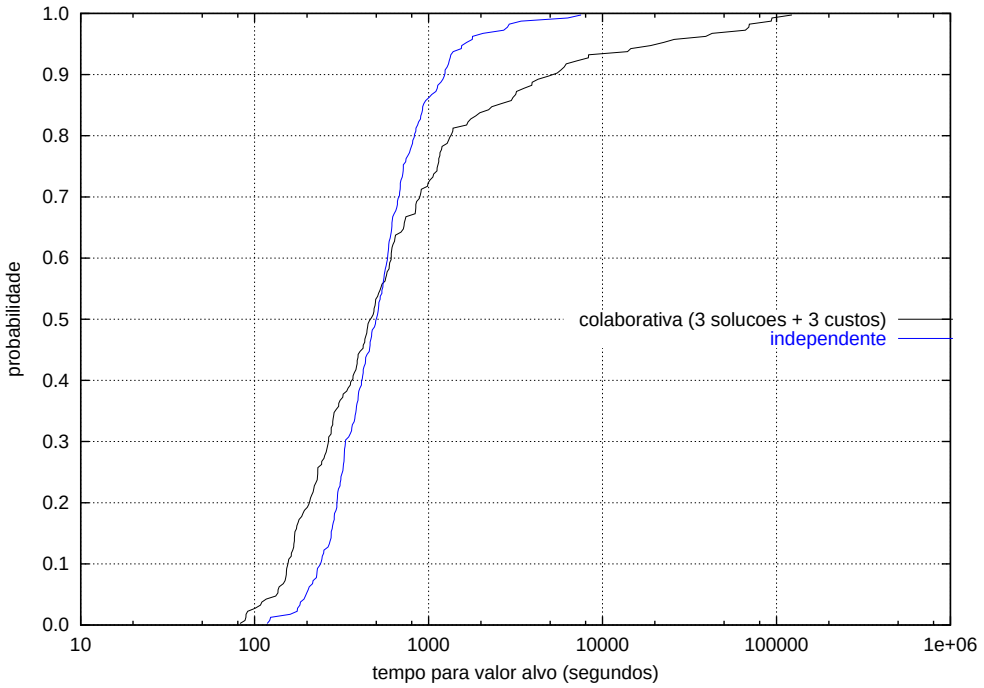

Figura 4.15: Comparação entre a estratégia colaborativa com envio de três custos e três soluções separadamente e a estratégia independente usando 4 processadores.

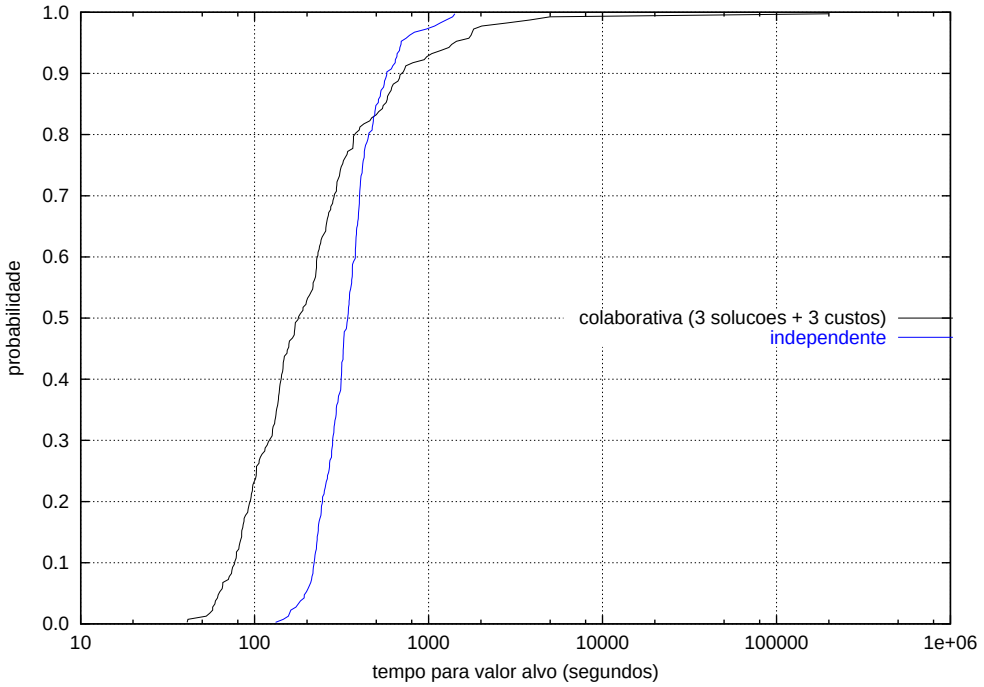

Figura 4.16: Comparação entre a estratégia colaborativa com envio de três custos e três soluções separadamente e a estratégia independente usando 8 processadores.

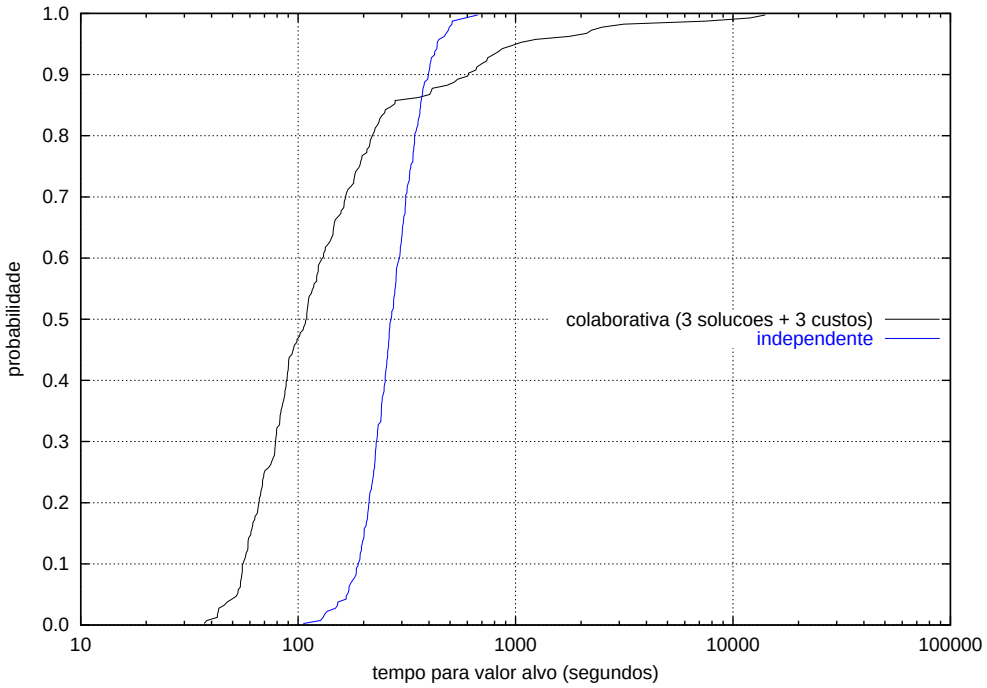

Figura 4.17: Comparação entre a estratégia colaborativa com envio de três custos e três soluções separadamente e a estratégia independente usando 16 processadores.

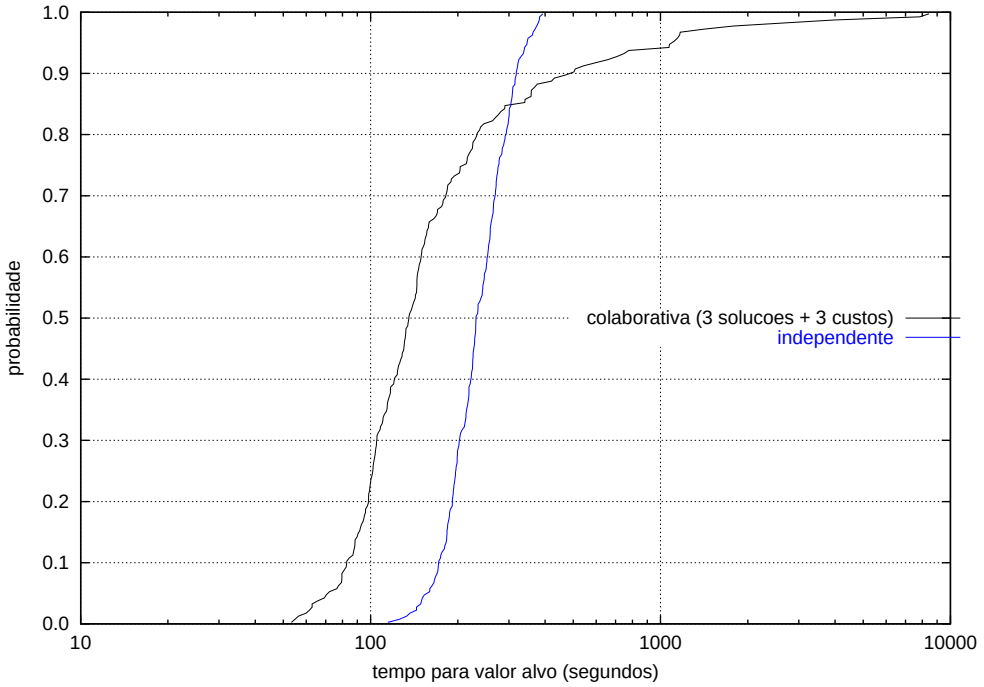

Figura 4.18: Comparação entre a estratégia colaborativa com envio de três custos e três soluções separadamente e a estratégia independente usando 32 processadores.

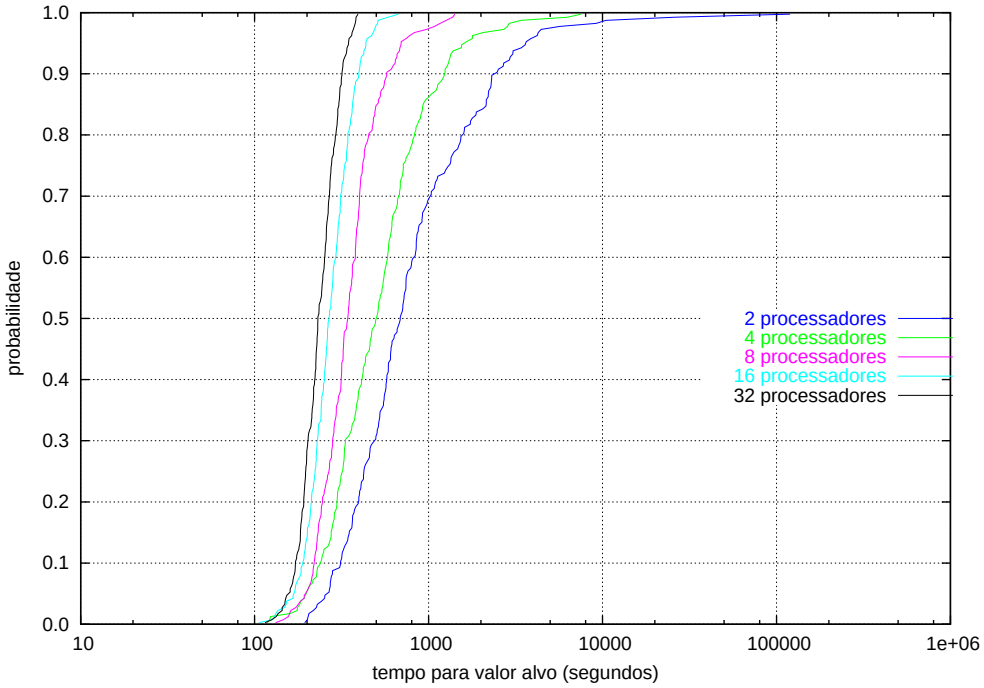

Figura 4.19: Variação do número de processadores para a estratégia independente.

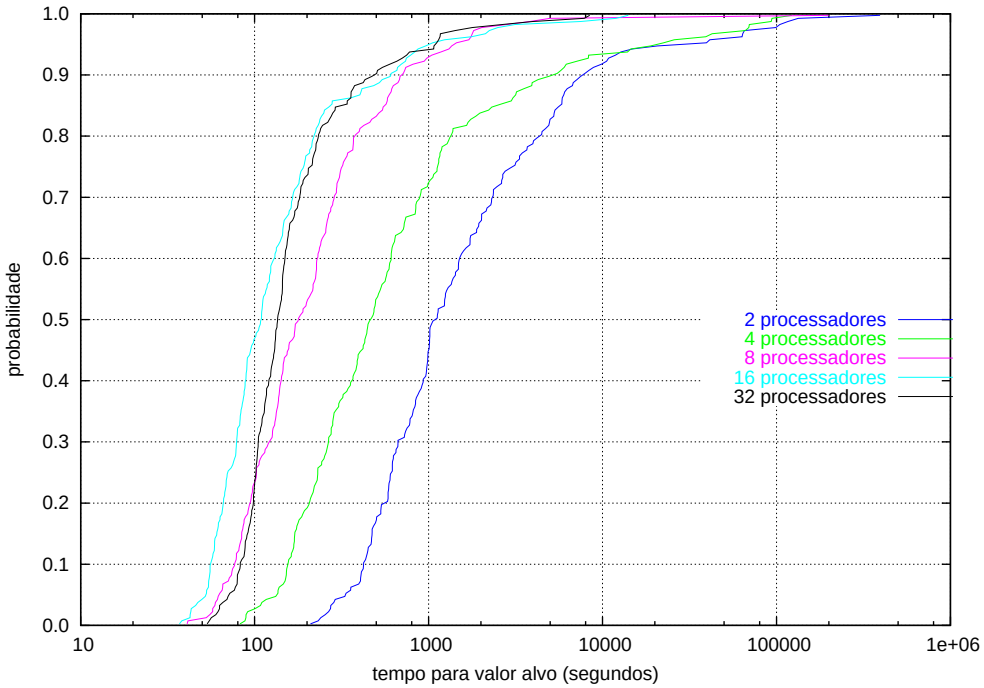

Figura 4.20: Variação do número de processadores para a estratégia colaborativa com envio de três custos e três soluções separadamente.

Processadores	Independente	Colaborativo
1	1310.1	—
2	686.8	1380.9
4	332.7	464.1
8	164.1	200.9
16	81.7	97.5
32	41.3	74.6

Tabela 4.1: Tempos médios em segundos de 10 execuções das estratégias independente e colaborativa com envio de três custos e três soluções separadamente com 3200 iterações.

de processamento de uma estratégia paralela sobre um número p de processadores. Acelerações lineares para a estratégia independente são observadas nas Figuras 4.21 e 4.22. Para plotar estes gráficos, foram considerados os tempos médios de dez execuções (com dez sementes distintas) das estratégias independente e colaborativa sobre a instância pré-definida com 3200 iterações para 2, 4, 8, 16 e 32 processadores. A estratégia independente também foi executada para um único processador. Os tempos médios são resumidos na Tabela 4.1. Os resultados obtidos na Figura 4.21 mostram que a aceleração da estratégia colaborativa é linear quando são utilizados, no máximo, 16 processadores.

Na Tabela 4.2 são mostrados os valores médios e as melhores soluções obtidas pelas estratégias independente e colaborativa para 2, 4, 8, 16 e 32 processadores. O valor em itálico indica qual a melhor solução obtida pela estratégia independente com 3200 iterações, usando somente um processador (neste caso, a estratégia independente equivale a GPRfb). Observa-se uma queda na qualidade das soluções encontradas pela estratégia independente quando o número de processadores participantes do sistema paralelo é aumentado. Esta queda se deve ao fato de que o conjunto de soluções de elite só passa a ser composto por boas soluções após um número considerável de iterações. Como o número de iterações é dividido pelo número de processadores, quanto mais processadores fazem parte do sistema, menos iterações cada um deles executa. Como esta estratégia não troca informações entre os processadores, o conjunto de elite fica empobrecido e permanece com poucas soluções de elite de qualidade.

Os resultados obtidos na Tabela 4.2 mostram que a estratégia colaborativa atinge a melhor solução seqüencial com 4 processadores. Estes dados indicam que a qualidade das soluções permanece ou melhora quando um número maior de processadores é utilizado, graças à cooperação existente entre eles.

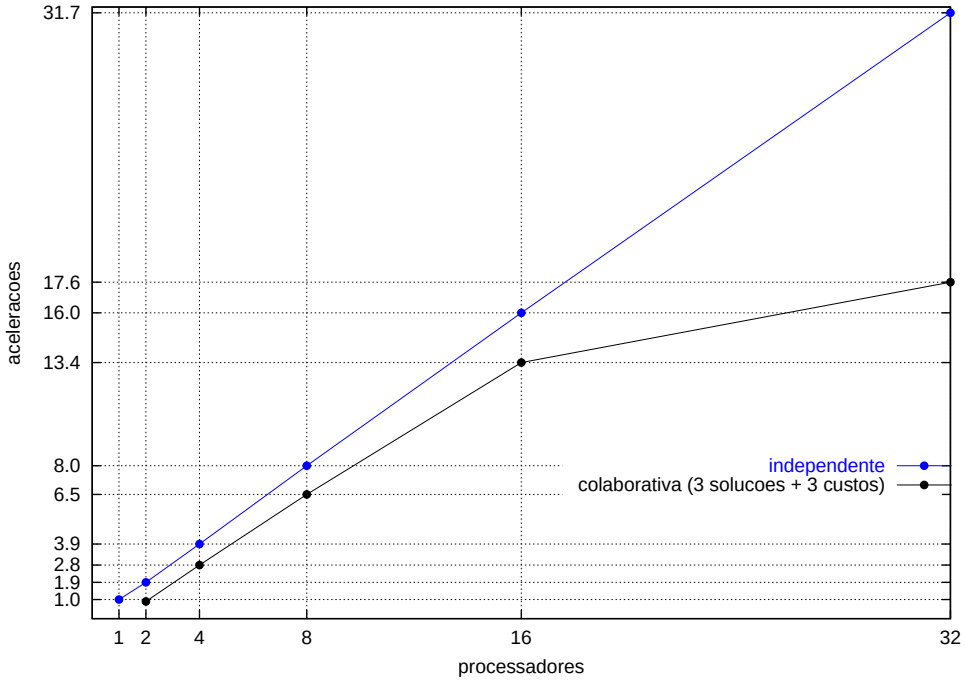

Figura 4.21: Acelerações das estratégias independente e colaborativa com envio de três custos e três soluções separadamente.

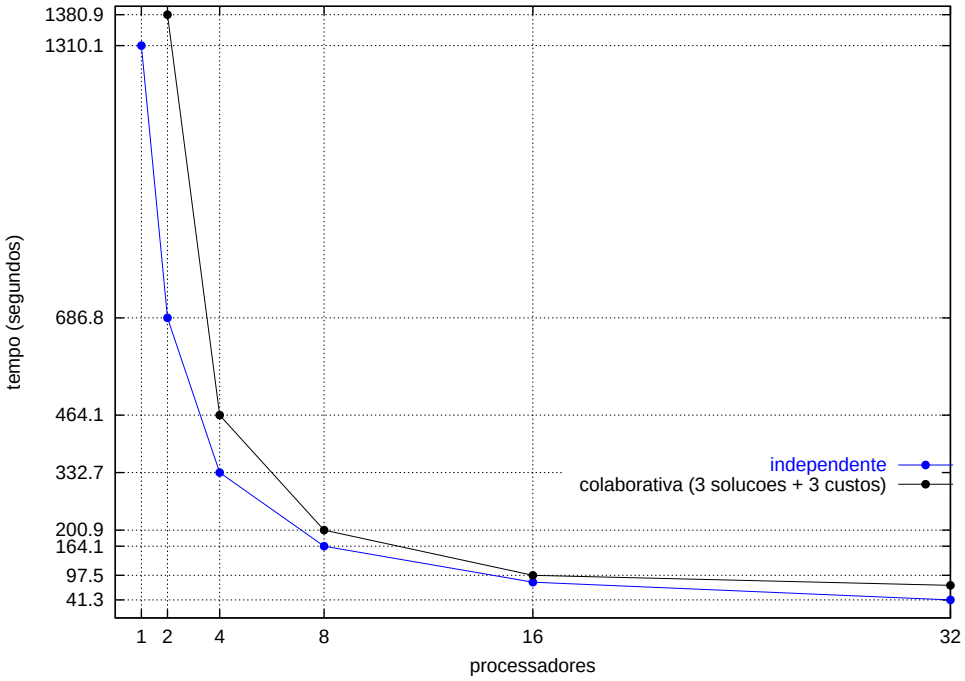

Figura 4.22: Médias dos tempos de 10 execuções das estratégias independente e colaborativa com envio de três custos e três soluções separadamente com 3200 iterações.

Processadores	Independente		Colaborativo	
	melhor	média	melhor	média
1	*673*	678.6	—	—
2	676	680.4	676	681.6
4	680	685.1	*673*	681.2
8	687	690.3	676	683.1
16	692	699.1	674	682.3
32	702	708.5	678	684.8

Tabela 4.2: Médias das soluções de 10 execuções das estratégias independente e colaborativa com envio de três custos e três soluções separadamente com 3200 iterações.

Para comprovar que a estratégia colaborativa se beneficia do aumento do número de processadores e da troca de informações entre os processadores, foi usado o teste estatístico *Mann-Whitney*, descrito na Seção 3.3. Para comparar a estratégia independente com a colaborativa com envio de três custos e três soluções separadamente para 16 processadores, as seguintes suposições foram feitas:

- hipótese nula: não existem diferenças de desempenho significativas entre a estratégia independente e a colaborativa;
- hipótese alternativa: existem diferenças de desempenho significativas entre a estratégia independente e a colaborativa;

O nível de significância escolhido foi $\alpha = 0.05$ e o número de execuções da estratégia independente e da colaborativa com envio de três custos e três soluções separadamente é igual a 200.

Os tempos gastos nas execuções das estratégias independente e da colaborativa são apresentadas nas Tabelas 4.3 e 4.4, respectivamente. A soma dos valores atribuídos aos tempos de execução da estratégia independente é 53072. A soma dos valores atribuídos aos tempos da estratégia colaborativa é 27128. Para obter o valor de T, tem-se que

$$T = R_{ind} - [n_{ind}(n_{ind} + 1)]/2 = 32972.$$

Calculando w_p, tem-se que

$$w_{0.95} = n_{ind}n_{col}/2 + s_{0.95}\sqrt{[n_{ind}n_{col}(n_{ind} + n_{col} + 1)]/12} = 21901.85525,$$

onde $s_{0.95} = 1.645$. Como $T > w_{0.95}$, a hipótese nula deve ser rejeitada em favor da hipótese alternativa. Assim, com uma probabilidade de 95%, pode-se afirmar que existem diferenças de desempenho significativas entre

tempo (s)	Semente	tempo (s)	Semente	tempo (s)	Semente	tempo (s)	Semente
228.490358	1	211.353787	51	256.087484	101	363.208421	151
306.243757	2	188.444381	52	166.482902	102	490.845968	152
205.442947	3	255.765064	53	274.031851	103	262.973957	153
246.255916	4	585.957084	54	401.084373	104	224.88213	154
336.883477	5	302.572856	55	132.48089	105	251.600009	155
195.565596	6	240.217053	56	252.50251	106	196.966328	156
230.146082	7	259.493944	57	105.721796	107	189.903799	157
281.910061	8	254.798682	58	280.923341	108	195.634746	158
240.970084	9	260.623743	59	233.121612	109	242.36038	159
171.634978	10	317.105356	60	217.704934	110	181.811369	160
226.824661	11	317.70689	61	310.317296	111	278.254916	161
201.2591	12	304.540611	62	185.176933	112	192.766031	162
408.161794	13	253.847683	63	249.138578	113	436.061841	163
241.66139	14	231.766151	64	404.451297	114	192.395461	164
219.204565	15	297.300875	65	312.501658	115	295.189464	165
209.023472	16	210.062205	66	470.653618	116	424.13954	166
210.749399	17	511.146005	67	283.27057	117	321.005002	167
177.51402	18	227.615235	68	212.431048	118	262.025347	168
355.559776	19	396.230442	69	434.236975	119	185.208524	169
241.394897	20	312.422977	70	222.476039	120	208.702141	170
248.711851	21	368.105817	71	213.471256	121	151.777348	171
268.378281	22	226.377543	72	293.418223	122	200.927407	172
264.406782	23	301.72948	73	326.139057	123	291.045069	173
269.545076	24	148.503009	74	262.431761	124	280.030458	174
444.171528	25	343.287289	75	231.251247	125	225.878742	175
398.168559	26	356.278508	76	207.773214	126	509.78062	176
126.810593	27	479.472844	77	325.942915	127	282.383083	177
381.100822	28	298.772363	78	152.358181	128	379.480524	178
374.019334	29	375.044332	79	329.517634	129	324.989734	179
340.990572	30	214.235098	80	349.041008	130	257.763349	180
288.069145	31	495.890373	81	184.766294	131	171.220435	181
243.459767	32	351.068334	82	310.157772	132	372.842958	182
403.179508	33	299.130401	83	337.729655	133	201.183182	183
193.214547	34	357.555642	84	212.463109	134	329.875051	184
670.508725	35	260.818421	85	282.128229	135	241.446496	185
337.732097	36	340.359214	86	337.222367	136	130.297602	186
222.699376	37	221.533109	87	233.579699	137	394.152169	187
136.047027	38	197.978859	88	206.174776	138	251.658087	188
262.334023	39	274.781609	89	343.443431	139	301.148009	189
248.471277	40	269.703111	90	283.372643	140	370.080515	190
251.466362	41	169.702101	91	311.978124	141	316.955671	191
217.500101	42	258.291127	92	243.74637	142	343.320512	192
410.21499	43	278.58887	93	294.333321	143	361.600753	193
436.206595	44	300.68234	94	166.294214	144	226.005521	194
366.216632	45	295.085977	95	224.994039	145	305.456637	195
275.133095	46	312.294692	96	258.153602	146	264.253039	196
365.836623	47	424.954483	97	274.749906	147	343.743581	197
231.700944	48	200.651019	98	228.349212	148	285.94396	198
276.513852	49	240.485325	99	174.06898	149	220.432978	199
324.665621	50	266.577409	100	363.327836	150	310.36395	200

Tabela 4.3: Tempos das 200 execuções da estratégia independente para 16 processadores.

tempo (s)	Semente	tempo (s)	Semente	tempo (s)	Semente	tempo (s)	Semente
83.669338	1	82.254387	51	54.889541	101	56.775396	151
82.164658	2	237.403312	52	68.597974	102	90.952585	152
142.179356	3	65.525426	53	279.800435	103	121.705927	153
54.450409	4	45.787914	54	90.500894	104	102.180896	154
89.056544	5	117.891791	55	180.436592	105	1056.46439	155
70.42532	6	521.967541	56	175.208297	106	78.460029	156
52.898737	7	169.753149	57	192.585612	107	73.197305	157
110.466044	8	82.506337	58	121.692324	108	656.810298	158
123.792821	9	65.89215	59	88.173088	109	106.114482	159
109.909987	10	133.157627	60	869.801176	110	57.383031	160
57.656707	11	144.491004	61	133.355042	111	207.074755	161
54.28883	12	53.018204	62	663.744331	112	249.941586	162
961.326026	13	123.890559	63	162.1901	113	181.639882	163
49.706302	14	788.314113	64	137.163103	114	89.414676	164
95.380395	15	111.748601	65	79.628958	115	214.90037	165
43.184904	16	3140.140049	66	58.895904	116	146.822507	166
55.397717	17	164.910328	67	75.684761	117	60.762709	167
79.537318	18	56.072235	68	67.962102	118	226.119123	168
87.789986	19	125.909592	69	38.093394	119	61.363548	169
69.387549	20	737.256613	70	152.541621	120	84.511106	170
98.983953	21	55.392731	71	148.070167	121	61.967995	171
59.216714	22	76.66881	72	14080.1489	122	1765.08105	172
414.428441	23	68.716869	73	1235.80544	123	251.915106	173
66.340019	24	111.299152	74	63.176416	124	162.698954	174
220.821305	25	744.606745	75	68.485252	125	109.643661	175
408.935983	26	140.26401	76	42.606193	126	403.14253	176
189.864999	27	85.81469	77	483.721215	127	78.245305	177
108.323199	28	193.952026	78	697.229892	128	47.149036	178
2235.78523	29	42.496468	79	101.662983	129	266.949444	179
179.862592	30	213.863606	80	7488.20254	130	121.988454	180
110.807686	31	94.52144	81	78.09286	131	130.856989	181
77.76283	32	58.618125	82	109.603823	132	58.822608	182
93.621872	33	115.157839	83	242.217482	133	608.944292	183
82.643139	34	96.154147	84	224.56243	134	11951.706	184
74.899873	35	55.098958	85	64.832093	135	157.08497	185
79.132029	36	107.832236	86	128.10853	136	833.155665	186
88.128035	37	86.9685	87	78.838362	137	231.789367	187
61.787807	38	182.741056	88	356.931869	138	112.450498	188
58.769784	39	90.021919	89	90.388734	139	67.209591	189
541.180748	40	118.296133	90	167.298766	140	105.37063	190
162.96765	41	131.02708	91	36.95424	141	184.681988	191
79.948834	42	90.241956	92	69.625044	142	278.885967	192
78.63726	43	235.658721	93	144.962689	143	66.293256	193
145.312129	44	84.941538	94	115.917665	144	89.196749	194
212.837562	45	86.96597	95	165.893498	145	67.295504	195
42.943473	46	157.8927	96	86.305828	146	146.199422	196
123.697159	47	2124.413801	97	55.421471	147	98.499971	197
63.301536	48	195.967895	98	52.043589	148	54.31413	198
2501.12991	49	65.229937	99	83.281002	149	217.891412	199
196.968968	50	60.55005	100	602.433488	150	207.263049	200

Tabela 4.4: Tempos das 200 execuções da estratégia colaborativa para 16 processadores.

a estratégia independente e colaborativa com envio de três custos e três soluções separadamente.

As comparações dos tempos das 200 execuções das estratégias independente e colaborativa com envio de três custos e três soluções obtidos sobre 2 e 16 processadores para a instância e o valor alvo pré-definidos, apresentadas nas Figuras 4.23 e 4.24, respectivamente, ilustram a conclusão de que a estratégia colaborativa se beneficia do aumento do número de processadores e da troca de informações entre eles. Para gerar estes gráficos, os tempos de execução das estratégias independente e colaborativa com envio de três custos e três soluções são ordenadas em ordem crescente separadamente. Constroem-se pares (col_i, ind_i), para $i = 1, \ldots, 200$, onde col_i e ind_i são os i-ésimos tempos ordenados de execução das estratégias colaborativa e independente, respectivamente. Os 200 pontos gerados são plotados. Em relação à Figura 4.23 (2 processadores), como todos pontos estão abaixo da reta $y = x$, isto ilustra que a estratégia colaborativa tem um desempenho inferior quando comparado ao da estratégia independente. Já na Figura 4.24 (16 processadores), como a maioria dos pontos está acima da reta $y = x$, isto ilustra que a estratégia colaborativa tornou-se superior.

4.7 Considerações Finais

Neste capítulo foram descritas, implementadas e avaliadas heurísticas paralelas para 2PNDP. Os experimentos computacionais foram realizados em um *cluster* de 32 processadores Pentium II 400MHz com 32 Mbytes, usando a implementação Red Hat 6.2 do sistema operacional Linux 2.2.14-5.0 e o *switch* IBM 8274 com 96 portas com velocidade de 10 Mbits/s para a comunicação entre os processadores.

Inicialmente, foram comparadas as estratégias independente e colaborativa básica, variando-se o número de processadores no ambiente paralelo. Com apenas dois processadores, observou-se que a estratégia independente tem um desempenho melhor que a estratégia colaborativa. Esta predominância da estratégia independente sobre a colaborativa para dois processadores é facilmente explicável: a estratégia independente faz uso de dois processadores para executar as iterações, enquanto a estratégia colaborativa utiliza, na realidade, um único processador. A medida que o número de processadores aumenta, esta predominância se inverte. Mesmo contando com apenas `p - 1` dos `p` processadores para executar iterações, o desempenho da estratégia colaborativa melhora progressivamente, graças

à execução de um menor número de iterações e à cooperação entre os processadores. Entretanto, quando foram utilizados 16 processadores, a quantidade de memória necessária para a execução da estratégia colaborativa básica ultrapassou a quantidade de memória disponível no *cluster*. Isto se deve ao elevado número de mensagens que chegam ao processador mestre, sem que este tenha capacidade de tratá-las.

Foram discutidas e implementadas variantes da estratégia colaborativa básica que diminuem o número de soluções enviadas ao mestre. Observou-se que as melhores estratégias são as que enviam a mesma quantidade de soluções ao mestre que a estratégia colaborativa básica, mas que verificam o custo da solução antes de enviá-la. Foi mostrado que a verificação do custo antes do envio da solução é imprescindível para prevenir problemas de memória no *cluster* utilizado.

O comportamento das estratégias paralelas é bastante uniforme sobre diferentes instâncias. Foram ilustrados na seção anterior os resultados obtidos em uma instância de 100 nós, 4950 arestas e 1000 pares origem-destino. Nesta seção, apresentam-se os resultados obtidos pelas melhores estratégias paralelas em outra instância, esta com 80 nós, 3160 arestas e 800 pares origem-destino. Para estes novos experimentos, o valor alvo 525 foi escolhido como sendo a média de dez soluções encontradas em dez execuções de GPRfb com 3200 iterações com dez sementes distintas. Os resultados obtidos sobre 8 processadores são ilustrados na Figura 4.25, considerando-se 50 execuções com sementes iniciais distintas em cada caso. Este exemplo ilustra que as observações feitas sobre os desempenhos relativos destas estratégias continuam válidas.

A estratégia colaborativa com envio de três custos e três soluções separadamente foi comparada à estratégia independente, usando 2, 4, 8, 16 e 32 processadores. As mesmas observações feitas para a comparação entre as estratégias independente e colaborativa básica para 2, 4, e 8 processadores continuaram válidas. O desempenho da estratégia colaborativa para 16 processadores foi muito superior ao da independente, o que não era alcançado pela estratégia colaborativa básica. Com 32 processadores, ocorre uma sensível queda de desempenho da estratégia colaborativa (Figura 4.20). Esta queda de desempenho se deve ao aumento do número de mensagens trafegando na rede, devido ao aumento considerável do número de processadores no ambiente paralelo.

Para mostrar que esta queda de desempenho foi devida exclusivamente às limitações do *cluster* e que o estudo realizado neste capítulo mostra como superar estas limitações através de estratégias mais eficientes de im-

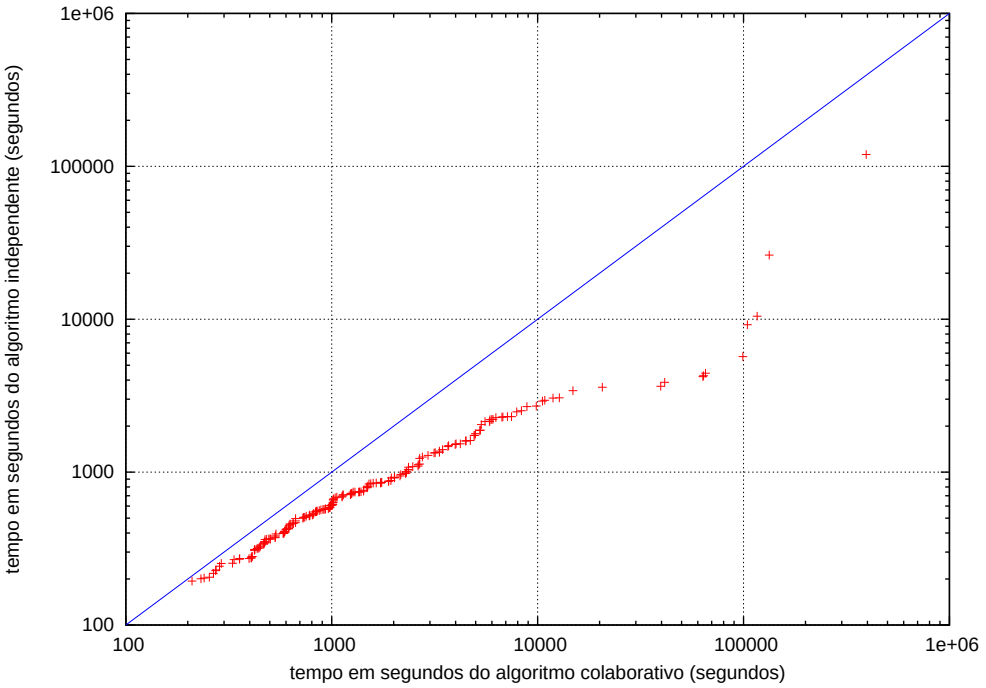

Figura 4.23: Comparação dos tempos das 200 execuções das estratégias independente e colaborativa com envio de três custos e três soluções separadamente usando 2 processadores.

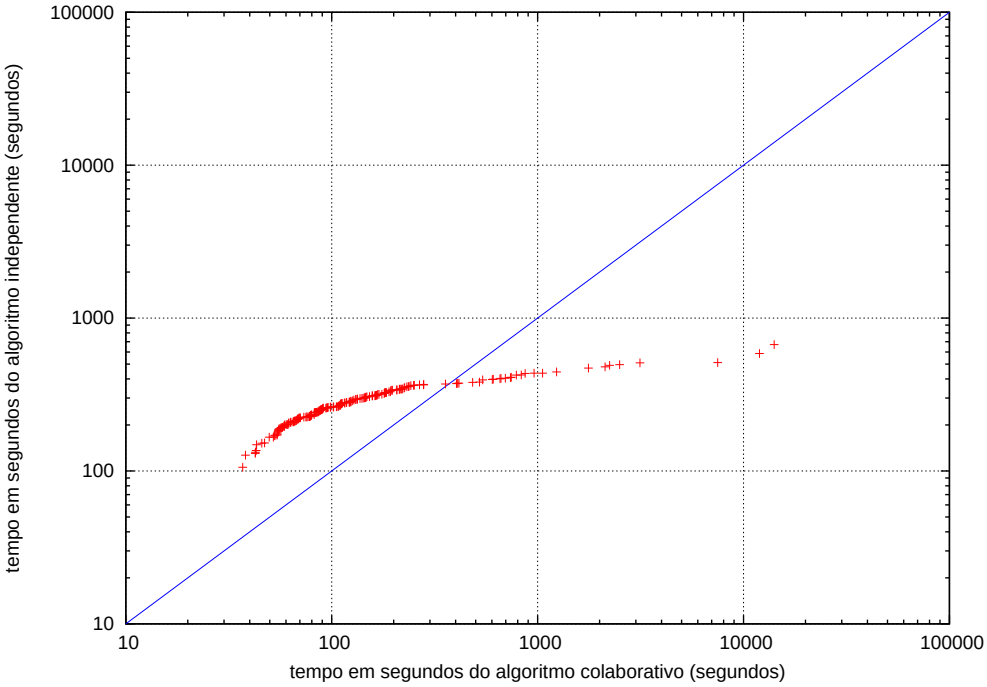

Figura 4.24: Comparação dos tempos das 200 execuções das estratégias independente e colaborativa com envio de três custos e três soluções separadamente usando 16 processadores.

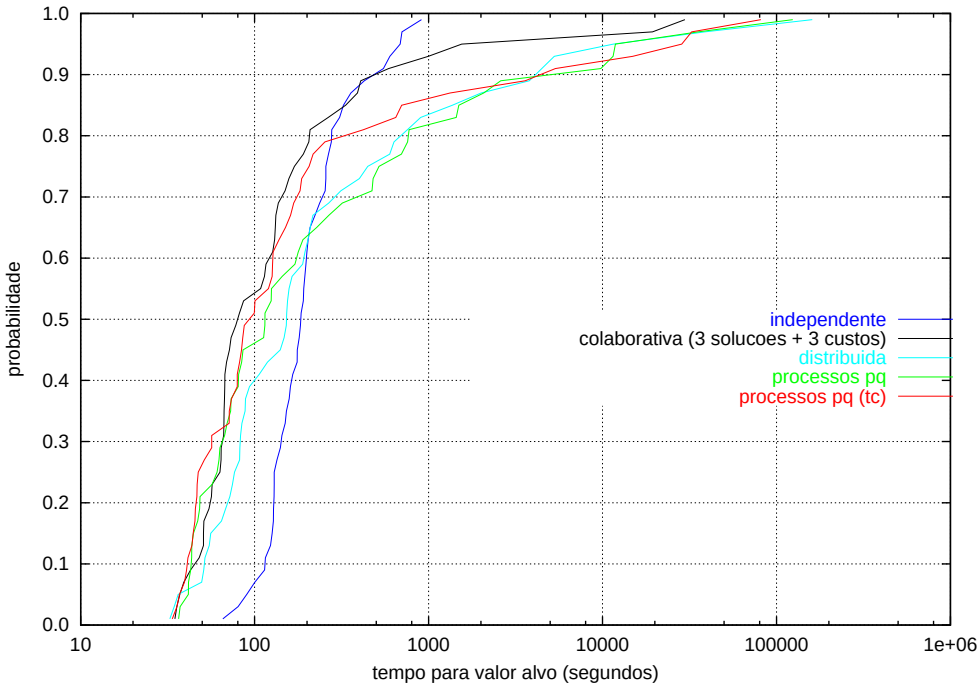

Figura 4.25: Comparação entre as melhores estratégias colaborativas e a independente sobre a instância de 80 nós.

plementação, as duas estratégias foram testadas em um *cluster* de 32 processadores Pentium IV 1.7 GHz com 256 Mbytes, usando a implementação Red Hat 7.2 do sistema operacional Linux e o *switch* Extreme Networks com 48 portas 10/100 Mbits/s e com 2 portas 1 Gbits/s para a comunicação entre os processadores. Analisando os resultados ilustrados nas Figuras 4.26 e 4.27, foi possível observar que não houve queda de desempenho da estratégia colaborativa quando foram usados 32 processadores. Este fato ilustra que a quantidade de memória disponível nos processadores e a velocidade da rede de comunicação influíram no desempenho dos algoritmos.

Segundo [53, 78], um algoritmo paralelo é escalável se o seu desempenho é proporcional ao número de processadores no ambiente paralelo. Assim, pode-se afirmar que a estratégia colaborativa com envio de três custos e três soluções separadamente não é escalável, mesmo quando não houve queda de desempenho desta estratégia, devido ao uso de um outro *cluster* com mais recursos. Se fosse possível aumentar o número de processadores deste *cluster* e executar o algoritmo colaborativo com mais de 32 processadores, quedas de desempenho desta implementação certamente seriam observadas.

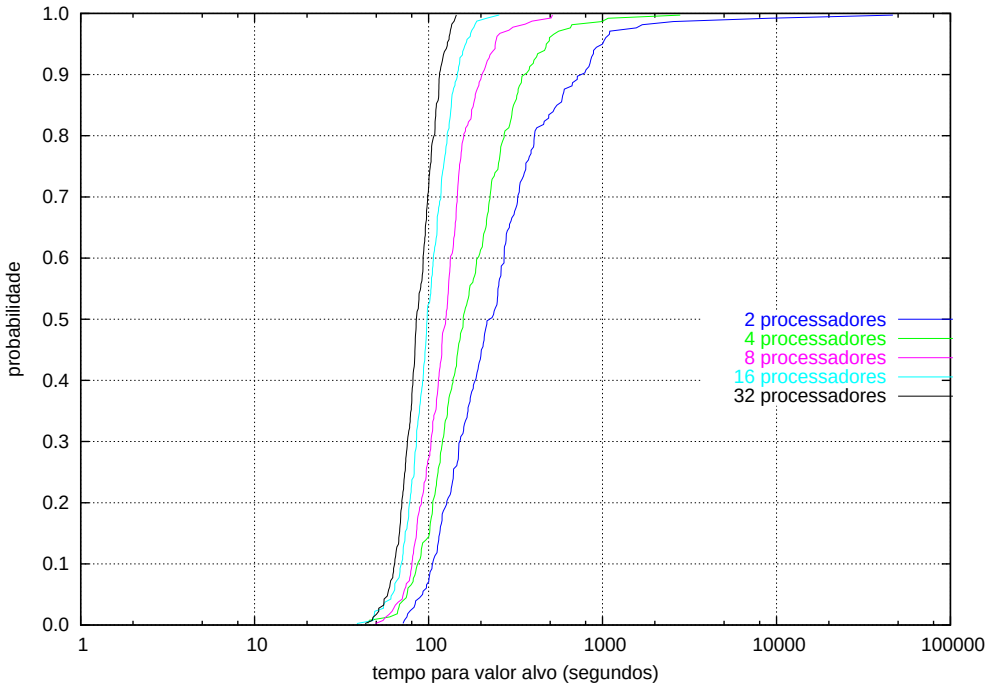

Figura 4.26: Variação do número de processadores para a estratégia independente usando um *cluster* de 32 processadores Pentium IV 1.7 GHz com 256 Mbytes.

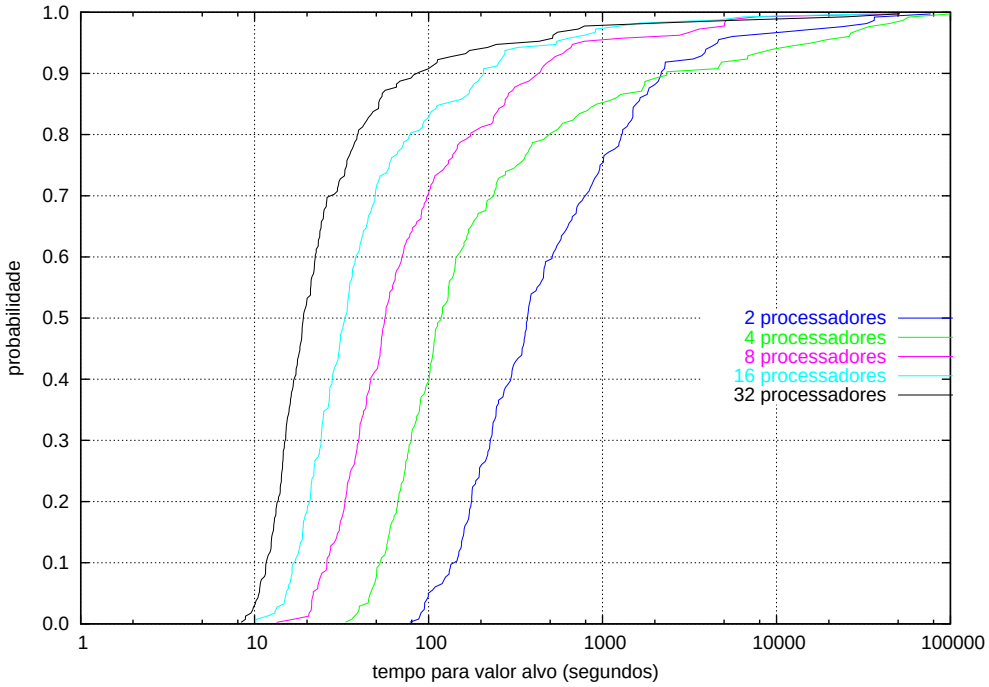

Figura 4.27: Variação do número de processadores para a estratégia colaborativa com envio de três custos e três soluções separadamente *cluster* de 32 processadores Pentium IV 1.7 GHz com 256 Mbytes.