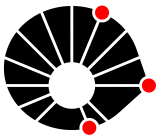

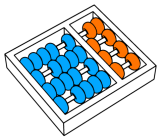

Uma implementação do algoritmo de Chudak e Shmoys para o Problema da Localização de Instalações

M. A. M. Alarcón *L. L. C. Pedrosa*

Relatório Técnico - IC-PFG-19-27
Projeto Final de Graduação
2019 - Julho

UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTO DE COMPUTAÇÃO

Uma implementação do algoritmo de Chudak e Shmoys para o Problema da Localização de Instalações

Miguel Ángel Marfurt Alarcón Lehilton Lelis Chaves Pedrosa*

Resumo

Este trabalho implementa o algoritmo de Chudak e Shmoys [1] para resolver o Problema da Localização de Instalações [11]. Neste problema, dado um conjunto de clientes e um conjunto de instalações o objetivo é encontrar um conjunto de facilidades para abrir de forma a minimizar o custo de abertura das instalações e o custo de conexão entre clientes às facilidades abertas mais próximas.

1 Introdução

Algoritmos de aproximação são algoritmos com complexidade polinomial para problema de otimização que asseguram uma solução com valor a no máximo um fator da solução ótima. Para lidar com problemas NP-difíceis, que não podem ser resolvidos de forma exata por algoritmos polinomiais (a não ser que P = NP), algoritmos de aproximação são muito úteis: a ideia é construir uma implementação de algoritmos para esses problemas, apesar da solução devolvida ter a qualidade possivelmente comprometida. Encontrar aproximações boas para problemas NP-difíceis pode ser trabalhoso; contudo, diversas técnicas já foram desenvolvidas, uma técnica em particular consiste em formular o problema como um programa linear inteiro ou misto, obter uma solução da relaxação correspondente e encontrar uma solução inteira para o problema original com custo limitado por um fator do valor da solução do programa linear.

Para resolver programas lineares, o algoritmo padrão da indústria é o algoritmo Simplex [3], que é exponencial no pior caso, e para instâncias práticas, ele é bem rápido. Muitas vezes, os pacotes para resolver PLs já possuem otimizações para detectar um tipo específico de PL e resolver com um algoritmo mais rápido (pode ser

*Instituto de Computação, Universidade Estadual de Campinas, 13081-970 Campinas, SP.

até polinomial). Além do mais, alguns PLs podem ser formulados em fluxo de rede [6] o que permite resolver o problema em tempo polinomial utilizando, por exemplo, algoritmos de resolução de fluxo máximo.

O Problema da Localização de Instalações [11] (FLP) é um problema bem conhecido e estudado na computação e enfrentado com alta recorrência na indústria. Nesse problema, um atacadista tem um conjunto de lojas espalhadas na cidade e precisa decidir quais armazéns contratar de forma a minimizar o custo de aluguel e transporte de mercadorias. A análise da localização de instalações é aplicada a vários problemas na indústria; por exemplo, a localização de aeroportos, escolas, armazéns, fábricas, postos de correios, hospitais, entre outros. O FLP é um problema NP-difícil, portanto, não existe algoritmo polinomial que o resolva (a não ser que P = NP). Contudo, existem algoritmos de aproximação na literatura que o resolvem, alguns dos quais são baseados na resolução de um programa linear inteiro relaxado[1].

2 Definições e métodos

2.1 Algoritmo de aproximação

Um algoritmo é uma α-aproximação para um problema de minimização se for um algoritmo polinomial que, para toda instância do problema produz uma solução cujo valor é no máximo α vezes o valor da solução ótima para o problema. Chamamos de α a performance do algoritmo, também chamada de fator de aproximação ou razão de aproximação.

2.2 Programação Linear e Programação Linear Inteira

Programação linear é um problema em que uma *instância* é formada por um vetor $c \in \mathbb{R}^n$, um vetor $b \in \mathbb{Q}^m$ e uma matriz $A = (a_{ij}) \in \mathbb{Q}^{m \times n}$ e a *tarefa* é devolver exatamente um dos seguintes resultados possíveis:

1. encontrar um vetor $x \in \mathbb{R}^n$ tal que $Ax \leq b$ e $c^T x$ é máximo, ou
2. verificar que $\{x \in \mathbb{R}^n : Ax \leq b\}$ é vazio, ou
3. verificar que para todo $\alpha \in \mathbb{R}$ existe um $x \in \mathbb{R}^n$ com $Ax \leq b$ e $c^T x > \alpha$.

Uma instância do problema acima é denominado Programa Linear. O vetor x é chamado de variável do problema. Qualquer x que satisfaça as restrições é chamado de solução factível e, se existir pelo menos algum x no problema, o problema é dito factível; caso contrário, é dito infactível. O termo programa linear é frequentemente abreviado para LP.

Algoritmo Simplex: O algoritmo Simplex é o mais velho e conhecido algoritmo para resolver problemas lineares criado por George Dantzig. Recebe como entrada os mesmos vetores da entrada de um problema de programação linear e retorna, por sua vez, uma solução factível ótima, caso exista, ou uma indicação que o problema é infactível, caso contrário.

O algoritmo consiste em inicializar a solução inicial como 0 e incrementar um pouco a variável que tem maior interferência positiva no resultado da função objetivo. O procedimento passa então para outra variável que nos aproxima da solução ótima e, em cada etapa, são convertidos todos os coeficientes das restrições de acordo com os limites encontrados nas sucessivas restrições.

O algoritmo possui complexidade exponencial no pior caso, devido a quantidade de iterações que são necessárias para atingir o critério de parada. Contudo, na prática, ele é bem eficiente e já foi relatado que são raras as instâncias em que atinge essa complexidade exponencial [3] e, por isso, o algoritmo é muito utilizado na indústria.

Programa Linear Inteiro: Também, é possível definir um programa linear com variáveis inteiras. Na programação linear inteira, uma *instância* é composta de um vetor $c \in \mathbb{Q}^n$, um vetor $b \in \mathbb{Q}^m$ e uma matriz $A = (a_{ij}) \in \mathbb{Q}^{m \times n}$. A *tarefa* é encontrar um vetor $x \in \mathbb{Z}^n$ tal que $Ax \leq b$ e $c^T x$ é máximo, ou verificar que $\{x \in \mathbb{Z}^n : Ax \leq b\}$ é vazio, ou verificar que $\sup\{cx : x \in \mathbb{Z}^n, Ax \leq b\} = \infty$.

Uma instância do problema acima é denominado Programa Linear Inteiro, ou abreviando, PLI. Diferentemente de um PL, PLI é um problema NP-difícil [6].

2.3 Problema da Cobertura de Conjuntos

No problema da cobertura de conjuntos (também chamado de set cover), são dados um conjunto de elementos $E = \{e_1, e_2, ..., e_n\}$, um conjunto de subconjuntos $S_1, S_2, ..., S_m$ onde cada $S_j \subseteq E$, $j = 1, ..., m$ e pesos não negativos $w_j \geq 0$ associados a cada subconjunto. A *tarefa* é encontrar uma coleção de subconjuntos de custo mínimo que cobre todos os elementos de E. O problema pode ser formulado como o seguinte PLI: Minimizar:

$$\sum_{j=1}^{m} w_j x_j$$

Sujeito a:

$$\sum_{j:e_i \in S_j} x_j \geq 1, \forall i = 1, ..., n$$

$$x_j \in \{0, 1\}, \forall j = 1, ..., m$$

onde foi adicionada uma variável de decisão x_j para indicar se um conjunto S_j faz parte da cobertura, ou não.

2.4 Problema da Localização de Instalações FLP

Um atacadista tem um conjunto de lojas espalhadas na cidade e precisa decidir que armazéns contratar de forma a minimizar o custo de aluguel e transporte de mercadorias. Esse problema tem motivação principalmente em cadeias de produção de indústrias e tem aplicações em projeto de rede em várias situações. Uma versão para o problema acima pode ser formulada da seguinte maneira [11]:

Seja um conjunto de clientes ou demandas D e um conjunto de facilidades F. Para cada cliente $j \in D$ e facilidade $i \in F$ existe um custo c_{ij} para atribuir um cliente j à facilidade i. Além do mais, existe um conjunto f_i associado à abertura de uma facilidade $i \in F$. Assim, queremos encontrar um subconjunto de facilidades $F' \subseteq F$ que minimize o custo de abrir as facilidades e atribuir cada cliente (um cliente é associado com apenas uma facilidade). Logo, o problema a seguir pode ser formulado no seguinte PLI:
Minimizar:

$$\sum_{i \in F} f_i y_i + \sum_{i \in F, j \in D} c_{ij} x_{ij}$$

Sujeito a:

$$\sum_{i \in F} x_{ij} = 1, \forall j \in D,$$

$$x_{ij} \leq y_i, \forall i \in F, j \in D,$$

$$x_{ij} \in \{0, 1\}, \forall i \in F, j \in D,$$

$$y_i \in \{0, 1\}, \forall i \in F,$$

onde foram adicionadas variáveis de decisão y_i para cada facilidade $i \in F$ para indicar se a facilidade é aberta ou não e x_{ij} para todo $i \in F$ e $j \in D$ para indicar se a facilidade i está atribuída ao cliente j.

2.5 Redução do Problema de Localização de Instalações para o Problema da Cobertura de Conjuntos

A seguinte redução foi mencionada por Young em um artigo que propõe algoritmos para resolução de PLs de instâncias do Problema da Localização de Instalações[13]:

Para cada facilidade j, crie um conjunto F_j com custo igual ao custo de abertura da facilidade j. Para cada par cliente i e facilidade j crie um elemento (i, j) e um conjunto $S_{ij} = \{(i, j)\}$. Para cada i, seja $\prec_i$ uma ordenação de facilidades j por distância até i crescente (para distâncias iguais a ordem é arbitrária) e faça F_j ser $\{(i, k) : j \prec_i k\}$. O custo do conjunto S_{ij} é igual a distância $d(i, j)$ de i até j menos a distância $d(i, j')$ da próxima facilidade i' na ordenação, se existir. O LP resultante pode ter $\Omega(nm^2)$ entradas iguais a 0, onde n é o número de facilidades e m o número de clientes.

3 Revisão bibliográfica

Para resolução de programas lineares, o Método Simplex é o mais conhecido na indústria, porém, dependendo da aplicação, é possível utilizar outras abordagens mais eficientes e que já cumprem com o seu propósito. Por exemplo, Young [12] propôs uma solução utilizando um algoritmo de aproximação para resolver problemas mistos de packing e covering (programas lineares com restrições da forma $Px \leq p$ e $Cx \geq c$ com matrizes P e C com coeficientes positivos). Em sua solução, ele conseguiu elaborar um algoritmo de aproximação de complexidade $O(md\log(m)/\epsilon^2)$ para satisfatibilidade de problemas mistos de packing e covering (onde m é o número total de restrições e d é o número máximo de restrições que uma variável aparece), verificando se existe um vetor $x \geq 0$ sujeito a $Px \leq (1+\epsilon)p$ e $Cx \geq c$, garantindo assim, a cobertura das restrições de covering e garantindo que as restrições de packing fossem no máximo um fator $1+\epsilon$ do desejado, para qualquer $\epsilon > 0$ constante.

O algoritmo de Young é bem simples e consistiu em construir a solução incrementalmente, com auxílio de derivadas parciais e observando a taxa de variação que cada incremento de alguma variável causava nas restrições de packing e covering, de forma a garantir que a primeira não ultrapasse um fator $1+\epsilon$ para cada restrição, garantindo o objetivo do algoritmo proposto. As computações realizadas no algoritmo se baseiam em produto de matrizes com vetores e somas, multiplicações, divisões e exponenciais termo a termo dos elementos das matrizes, gerando um algoritmo altamente paralelizável (versões de algoritmos paralelos são propostas no artigo). Vale ressaltar que muitos problemas que existem possuem a propriedade de serem puramente de packing (restrições na forma $Ax \leq b$), puramente de covering (restrições na forma $Ax \geq b$), ou ambos, como o problema da mochila, cobertura de conjuntos, entre outros. Young também conseguiu utilizar seu algoritmo para a resolução de problemas de otimização do tipo $max\{\lambda : Px \leq \lambda p, Cx \geq c\}$. A ideia consistiu basicamente em resolver vários sub-problemas de satisfatibilidade adicionando uma restrição de packing com um limitante variável utilizando pesquisa binária até encontrar a solução ótima. Além do mais, Young utiliza seu algoritmo para resolver problemas com número exponencial de variáveis, mais especificamente, o problema do fluxo mínimo multicommodities, implementável com uma complexidade proporcional a resolver $O(m\log(m)/\epsilon^2)$ sub-problemas de caminho mínimo, sendo m o número de arestas mais o número de commodities.

Com relação a algoritmos baseados sobre fluxo em rede, Garg et al. [4] buscaram solucionar problemas de fluxo multicommodities e problemas de packing de forma rápida com algoritmos de fácil implementação, aproximados e combinacionais. A abordagem deles baseou-se em explorar as versões duais dos problemas de fluxo máximo e mínimo multicommodities, problema de fluxo concorrente máximo, entre outros problemas. A ideia do algoritmo foi interpretar as variáveis do problema dual como comprimentos nas arestas do grafo (pois cada variável era relacionada a uma aresta)

iniciando cada comprimento com um valor muito pequeno γ. Logo, para cada iteração do algoritmo, introduzia-se um fluxo no caminho mínimo do grafo em função dos comprimentos das aresta e garantia-se que cada comprimento não fosse aumentado por um fator de no máximo $1 + \epsilon$, fazendo isso até atingir o valor máximo possível de fluxo, respeitando as restrições de capacidade. Como resultado dessa abordagem, eles obtiveram uma $1+\omega$ aproximação em tempo $O(\omega^{-2}(k \log m+m) \log m \cdot T_{sp})$ para o problema do fluxo concorrente máximo, onde k é o número de commodities, m o número de arestas mais vértices no grafo e T_{sp} é o custo para computar o caminho mínimo entre uma fonte e um ralo. Para atingir esse resultado foi necessário realizar diversas otimizações no algoritmo original, dentre elas, minimizar o número de vezes que o algoritmo de caminho mínimo é utilizado e garantir um número fixo de iterações na qual o algoritmo finalizava. Questões de integralidade também foram provadas que podem ser obtidas realizando pequenas modificações no algoritmo original.

Acerca do Problema de Localização de Instalações, o livro de Williamson e Shmoys [11] revisa a literatura sobre o problema, desde formulação como PLI aos algoritmos de aproximação, refinando cada solução a fim de melhorar o fator de aproximação dos algoritmos. Um resultado que também é demonstrado no livro é que não existe um algoritmo de aproximação para o Problema da Localização de Instalações com fator $\alpha \leq 1{,}463$, a não ser que cada problema em NP tenha um algoritmo com tempo de execução $O(n^{O(\log \log n)})$. As soluções apresentadas pelo livro se basearam em analisar a versão primal e dual do problema, definindo uma noção de "vizinhança" entre instalações e clientes, de forma que o algoritmo definia as vizinhanças a partir dos valores das variáveis do problema relaxado, olhando para esses valores como probabilidades do cliente j estar associado à instalação i [5]. O melhor algoritmo proposto no livro (desenvolvido por Chudak e Shmoys) possuía um fator de aproximação $\alpha = 1 + \frac{2}{\epsilon} \approx 1.736$. O algoritmo em questão consiste em: resolver a relaxação do PLI para FLP; escalar as variáveis de abertura de facilidades da solução do problema por um fator γ e transformar a solução relaxada incompleta em uma solução completa; dividir os clientes em clusters minimizando uma função de custo; escolher as facilidades que serão abertas (uma para cada cluster, escolhidas com uma certa probabilidade dentre as facilidades que estão mais próximas do centro do cluster); abrir demais facilidades que estão longe dos clusters com uma certa probabilidade (dada pela solução relaxada completa do PL); associar os clientes às facilidades abertas mais próximas.

Modificando o algoritmo de Chudak e Shmoys [2], que é um algoritmo com fator $(1 + 2/e)$, Byrka e Aardal[1] implementaram um algoritmo obtendo uma 1,5-aproximação, melhorando o resultado anterior que era de 1,52-aproximação feito por Mahdian, Ye, and Zhang comentado por eles. Nesse artigo, é definido o conceito de algoritmo de aproximação com bi-fatores, ou seja, uma (λ_f, λ_c)-aproximação, onde queremos encontramos um algoritmo cujo custo das facilidades seja no máximo $\lambda_f \cdot F^*$ do valor ótimo F^* e o custo das conexões entre facilidades e clientes seja no máximo $\lambda_c \cdot C^*$ do valor ótimo C^*. O algoritmo proposto por Byrka e Aardal é uma

(1,6774, 1,3738)-aproximação. O melhor algoritmo conhecido para o problema tem fator de aproximação 1,488 [7].

4 Metodologia e Implementação

O projeto foi dividido em três etapas.

1. Implementação do algoritmo proposto por Chudak e Shmoys[1] para resolver o FLP dada a solução do programa linear relaxado retornado pelo Simplex. Comparar aproximações obtidas com relação à solução ótima.

2. Desenvolvimento de uma adaptação do algoritmo de aproximação de Young para resolver programas lineares mistos de packing e covering[12]. Otimizar algoritmo para resolver o problema da cobertura de conjuntos[11]. Comparar qualidade das aproximações e tempo de execução com relação ao Simplex.

3. Reduzir o FLP para cobertura de conjuntos a partir da redução proposta por Young[13] e integrar os algoritmos de Chudak e Shmoys e Young implementados. Comparar aproximações e tempos de execução com relação à solução do programa linear inteiro original resolvido via Gurobi/Simplex.

4.1 Algoritmo de Chudak e Shmoys + Simplex

As instâncias utilizadas para testar o algoritmo são provenientes de dois datasets diferentes. O primeiro dataset (Beasley) foi obtido da OR-Library da Brunel University London[9], consistindo de 36 instâncias de dimensões variadas (as primeiras 24 instâncias têm tamanho médio variando de 16 a 50 facilidades e 50 clientes, e as 12 últimas instâncias são maiores, com 100 facilidades e 1000 clientes). O segundo dataset (Holmberg) foi obtido no site do Departamento de Ciência da Computação da Universidade de Pisa[10], consistindo de 71 instâncias de dimensões variando de 50 a 200 clientes e de 10 a 30 facilidades. Ambos datasets passaram por padronizações de formatação e simplificações dos dados para facilitar a automatização dos testes (muitas instâncias estavam em formatos diferentes e com informações desnecessárias que foram descartadas).

Para computar as soluções ótimas das instâncias do FLP via formulação do PLI relacionado e para obter as soluções relaxadas correspondentes, foi utilizada a ferramenta Gurobi. Como o Gurobi utiliza por padrão diversas otimizações automáticas e possui diversos algoritmos diferentes para resolver tipos específicos de PL, foi necessário desabilitar todas as flags de otimização para fazer uma comparação justa. Assim, deixamos o algoritmo de resolução do PL sendo o Simplex e o número de threads utilizadas como 1.

A implementação do algoritmo de Chudak e Shmoys foi feita inteiramente na linguagem C utilizando somente as bibliotecas básicas da linguagem. Optou-se por realizar uma implementação mais simples do algoritmo (não foram utilizadas estruturas de dados muito complexas) a fim de analisar apenas a qualidade das aproximações, e somente se a performance do algoritmo fosse muito comprometida seriam explorados novos meios de implementar o algoritmo. Por exemplo, para calcular a vizinhança entre clientes e facilidades poderia ter sido usado um grafo auxiliar utilizando listas de adjacência pois melhoraria o custo de acessar a vizinhança de um nó (muito utilizado no algoritmo). Contudo, a versão inicial utilizando matriz de adjacência já satisfez as necessidades do algoritmo sem muitas perdas de performance.

O algoritmo de Chudak e Shmoys possui um parâmetro γ que é utilizado no começo do algoritmo que escala as variáveis de abertura de facilidades do PL relaxado. Como nas fases seguintes do algoritmo a escolha das facilidades dependem de uma probabilidade relacionada às variáveis da solução do PL, escolher um γ muito alto apenas abriria todas as facilidades com probabilidades diferente de zero, pois escalando as variáveis a probabilidade de abertura seria muito próxima ou até mesmo maior que 1 (o que foi observado nos testes). Por isso, optou-se por utilizar apenas fatores γ próximos a 1. Nos resultados apresentados serão exibidos testes com γ entre 1 e 1.5.

4.2 Algoritmo de Young

As instâncias utilizadas para testar o algoritmo de Young são instâncias do problema da cobertura de conjuntos. As instâncias foram obtidas da mesma fonte do primeiro dataset do UFL (OR-Library [8]) e são compostas de 87 instâncias de tamanhos variados. Para facilitar a análise, as instâncias foram dividias em 3 datasets: small (50 instâncias menores do dataset original, de 500 a 4000 elementos e 50 a 400 subconjuntos), medium (30 instâncias médias do dataset original, também possui instâncias pequenas, mas grande parte das instâncias tem 5000 a 10000 elementos ou 500 a 1000 subconjuntos) e big (7 instâncias grandes do dataset original, possuindo 3 instâncias com 500 a 600 elementos e 50 a 60 mil subconjuntos e 4 instâncias com 2500 a 5000 elementos e por volta de 1 milhão de subconjuntos). Observação: No dataset medium, as 10 primeiras instâncias começam de instâncias pequenas (200 elementos e 500 subconjuntos) até instâncias médias (11264 elementos e 28160 subconjuntos) com grau de esparcidade aumentando crescentemente até a décima instância (poucos elementos nos subconjuntos). As demais instâncias possuem 5000 elementos e 500 subconjuntos ou 10000 elementos e 1000 subconjuntos. Contudo, tratam-se de instâncias mais densas (muitos elementos nos subconjuntos) tornando o tamanho dos arquivos maiores que as 10 primeiras instâncias. Infelizmente não foi possível utilizar as instâncias do dataset big pois o Gurobi não suportou carregar restrições e variáveis na ordem de 1 milhão (não foi possível concluir se foi incapacidade do Gurobi

ou da memória da máquina utilizada, acredita-se ser a segunda razão), além do mais, os tempos de execução para as menores instâncias do dataset big demoravam muito tempo para executar (mesmo no Gurobi), logo, pela inviabilidade na obtenção dos valores de referência das instâncias, optou-se por não utilizar este dataset.

Para computar as soluções ótimas das instâncias do set cover e obter os tempos de execução de referência foi utilizado o Gurobi com as mesmas configurações de execução do algoritmo de Chudak e Shmoys (algoritmo utilizado Simplex e número de threads igual a 1).

A implementação do algoritmo de Young foi feita inteiramente na linguagem C utilizando apenas as funções da biblioteca padrão da linguagem C. A fim de evitar multiplicações (que possuem custo muito alto nesse algoritmo), foi levado em consideração que as entradas do algoritmo são apenas instâncias da cobertura de conjuntos, ou seja, com a característica de possuir apenas zeros ou uns nas restrições de covering (já que as variáveis de pertinência nos subconjuntos são binárias), o que permitiu efetuar produtos de matrizes apenas com somas. Além do mais, os produtos de matrizes foram adaptados utilizando uma estrutura de dados compacta que salva apenas as entradas diferentes de zero das matrizes, a fim de efetuar produtos de matrizes mais eficientes para matrizes esparsas, encontradas com grande frequência em instâncias do problema da cobertura de conjuntos.

O algoritmo de Young implementado possui 2 parâmetros na execução, a precisão ϵ da aproximação desejada e o step size factor κ, que indica o tamanho do passo que o algoritmo realiza em cada iteração. O step size factor é um multiplicador de um valor α calculado no algoritmo que garante que a solução retornada sempre é um fator $1+\epsilon$ da solução ótima caso seja utilizado o valor α como step size, ou seja, caso seja utilizado um passo maior que α não é garantido que a solução devolvida seja uma $1+\epsilon$ aproximação. Contudo, foi observado que o valor α é um limitante muito abaixo do step size ideal e que é possível utilizar valores bem maiores que α ainda garantindo aproximações boas. Nos testes realizados, foram usados ϵ no intervalo de 0.08 e 1.00 e κ no intervalo de 1 a 100.

4.3 Algoritmo de Chudak e Shmoys + Algoritmo de Young

Nesta etapa final, testamos a nossa implementação do algoritmo de Chudak e Shmoys utilizando as soluções do PLI relaxado resolvidas pela nossa implementação do algoritmo de Young. As instâncias utilizadas para testar o algoritmo são as mesmas utilizadas na primeira subseção (dataset Holmberg e Beasley).

Como o PL original do FLP não é um programa linear misto de packing e covering, foi necessário reduzir o FLP para cobertura de conjuntos e assim utilizar a nossa implementação do algoritmo de Young para resolver o PL correspondente à instância fornecida.

Para computar as soluções ótimas das instâncias do FLP via PLI e os tempos de

execução de referência foi utilizada a ferramenta Gurobi com as mesmas configurações de execução das subseções anteriores (algoritmo utilizado Simplex e número de threads igual a 1).

Como a variação do fator γ do algoritmo de Chudak e Shmoys não comprometeu a qualidade das aproximações obtidas (ver resultados do algoritmo de Chudak e Shmoys), este parâmetro foi mantido constante (utilizou-se $\gamma = 1.2$). Além disso, os resultados do algoritmo de Young serviram como base para escolher as combinações de κ e ϵ para serem testados sem comprometer a qualidade das aproximações (vale ressaltar que Chudak e Shmoys não comentam nada sobre a qualidade das aproximações utilizando uma solução fracionária aproximada já que a 1.736-aproximação vale para a solução ótima do PL relaxado). Nos testes foram utilizados κ entre 10 e 80 e ϵ entre 0.57 e 1.

5 Resultados

5.1 Algoritmo de Chudak e Shmoys + Simplex

A figura 1 apresenta os fatores de aproximação e razão de tempo resolvendo o algoritmo de Chudak Shmoys com relação ao tempo total de execução obtidos para as instâncias do dataset Holmberg e a figura 2 apresenta os mesmos resultados para as instâncias do dataset Beasley (para todas as instâncias, o fator de aproximação foi calculado dividindo o custo total da solução obtida com o algoritmo de Chudak e Shmoys pelo custo da solução ótima obtida com o Gurobi).

Como observado nos gráficos das figuras 1 e 2, os fatores de aproximação obtidos em todas as instâncias ficaram bem menores que o valor máximo teórico de 1.736 que Chudak e Shmoys demonstraram no seu livro[11]. Para ambos datasets, valores de aproximação iguais a 1 vieram de instâncias cuja solução relaxada é igual à solução inteira (no qual ocorre em mais da metade dos casos), o que é de se esperar, já que a probabilidade de abertura das localizações é sempre 0 ou 1, que por sua vez não seriam alterados nas etapas do algoritmo de Chudak e Shmoys.

No dataset Holmberg, todos os testes que apresentaram fatores de aproximação diferente de 1 ficaram com aproximação entre 1.04 e 1.10 ($\gamma = 1$, $\gamma = 1.1$ e $\gamma = 1.2$) e 1.02 e 1.08 ($\gamma = 1.3$ e $\gamma = 1.5$), com exceção da instância 56 que apresentou fator de aproximação próximo de 1.13 para $\gamma > 1$. No dataset Beasley, para todos os valores de γ testados, exceto 1.5, os testes que apresentaram fatores de aproximação diferente de 1 ficaram com aproximação observada menores que 1.12 (alguns testes apresentaram fator de aproximação 1.05 e outros por volta de 1.03) e para $\gamma = 1.5$, por volta de 1.05 ou menores. Como observado em ambos datasets, o algoritmo de Chudak e Shmoys apresentou soluções muito próximas da solução ótima para instâncias cuja solução do PLI relaxado é fracionária.

A alteração do fator γ para uma mesma instância do problema demonstrou modificar levemente os fatores de aproximação obtidos nas mesmas, mas nada que comprometesse gravemente o custo total da solução. Para algumas instâncias, aumentar levemente o fator γ diminuía o fator de aproximação, e para outras, aumentava o fator de aproximação, contudo, foi constatado em poucas instâncias. Além do mais, por se tratar de um algoritmo aleatorizado é difícil afirmar se a alteração do γ melhorava o fator de aproximação do algoritmo ou se apenas alterava o caminho que a solução percorria até chegar na solução final.

Com relação ao tempo de execução, a performance do algoritmo de Chudak e Shmoys se limita pela obtenção da solução do PLI relaxado. A fração de tempo que o algoritmo leva para resolver o PLI relaxado (utilizando o Gurobi) com relação ao tempo total de execução é muito maior que a fração de tempo que o algoritmo de Chudak e Shmoys utiliza para efetuar todas as suas etapas (escalar solução do PLI relaxado, clusterização de clientes, cálculo de vizinhança entre clientes e facilidades, abertura aleatorizada de facilidades e conexão de clientes mais próximos às facilidades abertas). Para as instâncias do dataset Holmberg a razão fica por volta de 0.1 ou menor, e para instâncias maiores (observando o dataset Beasley) a razão diminui ainda mais (nas primeiras 24 instâncias que são menores a razão fica entre 0.03 e 0.06 e nas ultimas 12 instâncias que são maiores a razão fica por volta de 0.01). Por causa desse resultado, optou-se por manter a implementação do algoritmo de Chudak e Shmoys como está (apesar de ainda ser possível otimizar algumas etapas do algoritmo, por exemplo, utilizando estruturas de dados mais complexos para computar as vizinhanças entre clientes e facilidades na hora de fazer a clusterização dos clientes) e dar mais foco no algoritmo de Young para tentar bater o tempo de execução do Gorobi.

5.2 Algoritmo de Young

As figuras 3 e 4 apresentam os fatores de aproximação e speedup obtidos para as instâncias do dataset small e as figuras 5 e 6 apresentam os fatores de aproximação e speedup obtidos para as instâncias do dataset medium.

Para todas as instâncias, o fator de aproximação foi calculado dividindo o custo total da solução obtida com custo da solução ótima obtida pelo Gurobi resolvendo o PL correspondente à instância testada. Os speedups foram calculados dividindo o tempo de execução do Gurobi resolvendo o PL do setcover pelo tempo de execução do algoritmo de Young resolvendo o mesmo PL.

Optou-se por exibir os gráficos de speedup aplicando log na base 10, pois observou-se que conforme as dimensões da entrada aumentavam , o speedup aumentava muito, chegando na casa dos milhares, dificultando a leitura do gráfico.

5.2.1 Fatores de aproximação e speedups observados, $\kappa = 1$

Observando primeiramente as aproximações obtidas utilizando o $\kappa = 1$ (ou seja, utilizando o step size = α que garante a aproximação desejada fornecida no algoritmo) vemos que tanto no dataset small e medium, as aproximações são todas menores que as fornecidas no algoritmo (o que é esperado).

No dataset small, utilizando um $\epsilon = 1$, a instância que gerou a pior aproximação tem um fator de aproximação observado de 1.35 (bem menor que a maior possível que é uma 2-aproximação). Ainda utilizando a mesma precisão, em algumas instâncias foram encontradas aproximações muito boas (por volta de 1.04) mesmo utilizando $\epsilon = 1$. No dataset medium, utilizando a mesma precisão, a pior aproximação foi de 1.4, onde também foram encontradas aproximações muito boas em algumas instâncias (por volta de 1.05).

Utilizando $\epsilon = 0.43$, todas as instâncias dos datasets small e medium possuem fator de aproximação observados menores que 1.15, e utilizando $\epsilon = 0.18$, todas as instâncias dos datasets small e medium possuem fator de aproximação observados menores que 1.05. Estes resultados demonstraram que o step size utilizado de fato garante a aproximação escolhida na execução o algoritmo, garantindo aproximações muito melhores na prática. Por causa disso, foi considerado testar step size maiores posteriormente.

Com relação aos speedups obtidos, no dataset small, apenas utilizando $\epsilon = 1$ (na maioria dos testes) e $\epsilon = 0.76$ (na minoria dos testes) obtivemos performances melhores que o Gurobi, contudo, não foram muito elevados (por volta de no máximo 4x mais rápido que o Gurobi).

Já no dataset medium, observamos melhoras de performance na maioria das instâncias até $\epsilon = 0.43$ (onde obtivemos aproximações menores que 1.15 em todas as instâncias). Vale ressaltar que para uma instância (instância 10) que teve performance melhor que o Gurobi com todos os ϵ testados (até 0.08) e além do mais, utilizando $\epsilon = 1$ foi obtido speedup por volta de 1000 com aproximação observada de 1.025 (melhor resultado encontrado). Um outro conjunto de instâncias (1 a 9 por exemplo) também obtiveram performances muito boas (10x a 100x mais rápidas que o Gurobi) utilizando $\epsilon = 1$ e com aproximações observadas por volta de 1.05. A maioria das instâncias apresentaram fatores de aproximação menores que 1.4 com speedups obervados entre 10 e 30 (utilizando $\epsilon = 1$), fatores de aproximação menores que 1.3 com speedups entre 3 e 10 (para $\epsilon = 0.76$) e fatores de aproximação menores que 1.2 com speedups entre 2 e 9 (para $\epsilon = 0.57$).

5.2.2 Fatores de aproximação e speedups observados, $\kappa > 1$

Observando os resultados dos datasets small e medium, temos que as aproximações pioram claramente conforme aumentamos o valor do step size, sendo que no dataset small as aproximações observadas ficam maiores que as esperadas a partir de um

step size maior que 80x do step size original e para o dataset medium os valores das aproximações são maiores a partir de um step size maior que 30x do step size original.

Contudo, ainda notamos uma melhora na precisão da aproximação conforme diminuimos o ϵ com todos os step size utilizados, de forma que a diminuição do ϵ em função do aumento do step size compensa a perda de precisão do algoritmo. Por causa disso, foi necessário ponderar o trade-off entre precisão e performace de forma a encontrar combinações dos parâmetros κ e ϵ que gerem speedups bons com fatores de aproximação pequenos.

Segue alguns dos melhores resultados observados nos gráficos das figuras 3 a 6 separados por dataset. Para todos os resultados foram selecionados combinações de κ e ϵ de forma que para todas as instâncias o speedup seja maior que 1 (melhor que o Gurobi) e cujas aproximações observadas sempre sejam menores que o ϵ fornecido.

No dataset small, alguns resultados dentre os melhores (fator de aproximação observado e speedup) foram:

- $\kappa = 10$, $\epsilon = 1.00$: speedup por volta de 10 a 50, aproximações observadas menores que 1.4 (cerca da metade das instâncias menores que 1.2)
- $\kappa = 10$, $\epsilon = 0.76$: maioria das instâncias com speedup entre 3 e 30, aproximações observadas por volta de 1.3 (cerca da metade das instâncias por volta de 1.1)
- $\kappa = 20$, $\epsilon = 1.00$: maioria das instâncias com speedup entre 5 a 40, aproximações observadas menores que 1.5 (cerca da metade das instâncias menores que 1.2)
- $\kappa = 20$, $\epsilon = 0.76$: maioria das instâncias com speedup entre 5 e 40, aproximações observadas por volta de 1.3 (mais da metade das instâncias menores que 1.15)
- $\kappa = 30$, $\epsilon = 0.57$: maioria das instâncias com speedup entre 3 e 30, aproximações observadas menores que 1.2 (metade das instâncias por volta de 1.1)
- $\kappa = 50$, $\epsilon = 0.57$: maioria das instâncias com speedup entre 10 e 40, aproximações observadas por volta de 1.3 (cerca da metade das instâncias por volta de 1.1)
- $\kappa = 100$, $\epsilon = 0.43$: maioria das instâncias com speedup entre 3 e 30, aproximações observadas por volta de 1.3 (cerca da metade das instâncias menores que 1.18)

No dataset medium, alguns resultados dentre os melhores (fator de aproximação observado e speedup) foram:

- $\kappa = 10$, $\epsilon = 0.57$: maioria das instâncias com speedup maior que 20 (algumas maiores que 100 e uma por volta de 1000), aproximações observadas por volta de 1.2

- $\kappa = 10$, $\epsilon = 0.43$: maioria das instâncias com speedup maior que 10 (algumas maiores que 100), aproximações observadas por volta de 1.15 (algumas instâncias menores que 1.05)
- $\kappa = 20$, $\epsilon = 0.32$: maioria das instâncias com speedup maior que 10 (algumas instâncias maiores que 100), aproximações observadas por volta de 1.15 (maioria das instâncias)
- $\kappa = 30$, $\epsilon = 0.32$: instâncias com speedup maior que 10 (algumas instâncias maiores que 100, uma por volta de 1000), aproximações observadas menores que 1.25
- $\kappa = 50$, $\epsilon = 0.24$: instâncias com speedup maior que 10 (algumas instâncias maiores que 100, uma por volta de 1000), aproximações observadas menores que 1.15 exceto uma igual a 1.21 (cerca de metade menor que 1.1)
- $\kappa = 50$, $\epsilon = 0.18$: maioria das instâncias com speedup entre 10 e 30 (algumas instâncias maiores que 100), aproximações observadas menores que 1.1
- $\kappa = 80$, $\epsilon = 0.14$: maioria das instâncias com speedup entre 5 e 30 (algumas instâncias maiores que 100), aproximações observadas menores que 1.1 exceto uma igual a 1.11
- $\kappa = 100$, $\epsilon = 0.11$: maioria das instâncias com speedup entre 3 e 10 (algumas instâncias maiores que 100), aproximações observadas por volta de 1.05

Como pressuposto, aumentar o step size de fato aumentou o speedup e manteve a precisão das aproximações (regulando o κ e ϵ é claro). No geral, nas instâncias pequenas (dataset small), o speedup foi observado apenas para ϵ maiores que 0.24 (precisando de step sizes menores para garantir a precisão desejada), enquanto que nas instâncias médias (dataset medium), o speedup foi aparente em praticamente todas as combinações de κ e ϵ que garantisse a precisão nas aproximações.

Outra observação no dataset medium é que conforme a esparcidade e as dimensões das entradas aumenta (instâncias de 1 até 10) o speedup também aumenta consideravelmente (são os melhores resultados onde encontramos speedups maiores que 100 a partir da instância 7 e no caso da instância 10 por volta de 1000 ou mais). Esse comportamente é esperado já que a implementação do algoritmo de Young proposta foi otimizada para matrizes esparsas, cujo produto de matrizes (custo principal do algoritmo) tem complexidade de acordo com o número de entradas diferentes de zero das matrizes das restrições de packing e covering. O Simplex utilizado pelo Gurobi, por outro lado, não utiliza essa otimização, logo, a performance dele depende somente das dimensões das matrizes, independente das características específicas delas, o que torna o algoritmo de Young implementado muito superior ao Simplex para matrizes esparsas.

5.3 Algoritmo de Chudak e Shmoys + Algoritmo de Young

As figuras 7 a 9 apresentam os fatores de aproximação e speedup obtidos para as instâncias do dataset Holmberg e as figuras 10 a 12 apresentam os fatores de aproximação e speedup obtidos para as instâncias do dataset Beasley.

De modo geral, as aproximações ficaram inferiores que o algoritmo de Chudak e Shmoys utilizando a solução fracionária ótima do PL relaxado, contudo, as aproximações observadas ainda são menores que a 1.736-aproximação garantida do algoritmo de Chudak e Shmoys, sendo que no dataset de Holmberg todas as combinações de κ e ϵ testadas possuem em média um fator de aproximação por volta de 1.1 enquanto que no dataset Beasley o mesmo valor fica por volta de 1.2 (valores ainda muito abaixo da 1.736-aproximação).

No dataset Holmberg, alguns resultados dentre os melhores (fator de aproximação observado e speedup) foram:

- $\kappa = 30$, $\epsilon = 1.00$: maioria das instâncias com speedup por volta de 5 a 50 (algumas instâncias pouco maiores que 100), aproximações observadas por volta de 1.3 ou menores sendo que maior parte se encontra entre 1 e 1.2 (fator de aproximação médio de 1.11).

- $\kappa = 80$, $\epsilon = 1.00$: maioria das instâncias com speedup por volta de 7 a 50 (algumas instâncias próximas de 200), aproximações observadas menores que 1.5 sendo que maior parte se encontra entre 1 e 1.2 (fator de aproximação médio de 1.15).

- $\kappa = 50$, $\epsilon = 0.76$: maioria das instâncias com speedup por volta de 3 a 50 (algumas instâncias entre 100 e 200), aproximações observadas menores que 1.4 sendo que maior parte se encontra entre 1 e 1.2 (fator de aproximação médio de 1.15).

- $\kappa = 80$, $\epsilon = 0.76$: maioria das instâncias com speedup por volta de 3 a 50, aproximações observadas por volta de 1.4 ou menores, sendo que maior parte se encontra entre 1 e 1.14 (fator de aproximação médio de 1.15).

- $\kappa = 50$, $\epsilon = 0.57$: maioria das instâncias com speedup por volta de 2 a 30, aproximações observadas por volta de 1.3 ou menores sendo que maior parte se encontra entre 1 e 1.15 (fator de aproximação médio de 1.1).

No dataset Beasley, alguns resultados dentre os melhores (fator de aproximação observado e speedup) foram:

- $\kappa = 10$, $\epsilon = 1.00$: boa parte das instâncias com speedup entre 1 e 9, aproximações observadas menores que 1.6 sendo que a maioria se encontra entre 1 e 1.25 (fator de aproximação médio de 1.15).

- $\kappa = 50$, $\epsilon = 1.00$: maioria das instâncias com speedup entre 3 e 13 (algumas pouco maiores que 20), aproximações observadas menores que 1.6 sendo que a maioria se encontra entre 1 e 1.3 (fator de aproximação médio de 1.21).

- $\kappa = 80$, $\epsilon = 1.00$: maioria das instâncias com speedup entre 3 e 14 (algumas pouco maiores que 20), aproximações observadas por volta de 1.6 ou menores sendo que a maioria se encontra entre 1 e 1.4 (fator de aproximação médio de 1.31).

- $\kappa = 80$, $\epsilon = 0.76$: maioria das instâncias com speedup entre 2 e 10, aproximações observadas menores que 1.6 sendo que a maioria se encontra entre 1 e 1.4 (fator de aproximação médio de 1.24).

- $\kappa = 80$, $\epsilon = 0.57$: boa parte das instâncias com speedup entre 1 e 8, aproximações observadas menores que 1.6 sendo que a maioria se encontra entre 1 e 1.3 (fator de aproximação médio de 1.2).

Em ambos datasets, observou-se que no geral o aumento do κ e do ϵ não piorou as aproximações geradas pelo algoritmo de Chudak e Shmoys, ou seja, uma solução do PLI relaxado aproximada já é suficiente para garantir aproximações boas no algoritmo em questão, e além do mais, esta solução não precisa ter uma precisão muito alta para atingir precisões dentro da 1.736-aproximação para todas as instâncias testadas (formalmente isso não é garantido para qualquer instância). Esse ótimo resultado permitiu escolher valores de κ e ϵ mais elevados a fim de melhorar a performance do algoritmo e tentar obter speedups maiores com relação ao Simplex.

Os speedups em geral ficaram melhores para o dataset Holmberg (que apresenta instâncias menores) onde foi possível obter speedups acima de 100x o tempo de execução do Gurobi resolvendo o PL. No dataset Beasley (que apresenta instâncias maiores) o speedup foi inferior, obtendo no máximo speedups pouco acima de 20x o tempo de execução do Gurobi. Vale lembrar que foi necesssário aplicar uma redução no PLI original para resolver o FLP reduzido a cobertura de conjuntos (assim foi possível utilizar o algoritmo de Young), e esta redução aumenta muito o número de variáveis e restrições (por volta de $O(nm)$ variáveis e restrições, onde n é o número de facilidades e m o número de clientes). Contudo, a esparcidade do PLI reduzido aliado a ótima performance do algoritmo de Young para instâncias esparsas compensou a penalização da aplicação da redução, já que foi possível para algumas combinações de κ e ϵ obter speedups maiores que 1 para todas as instâncias em ambos datasets obtendo bons fatores de aproximações observados médios ($\kappa = 80$ e $\epsilon = 1.00$ para o dataset Holmberg e $\kappa = 50$ e $\epsilon = 1.00$ para o dataset Beasley, por exemplo).

Ademais, para algumas instâncias foram obtidos fatores de aproximação iguais a 1, ou seja, solução igual à solução ótima (no dataset Holmberg utilizando $\kappa = 10$ e $\epsilon = 1.00$ nas instâncias 17, 30, 32, 61, e 71 e no dataset Beasley utilizando a mesma

combinação de parâmetros nas instâncias 2 e 19). Esse resultado é bem interessante visto que testando o algoritmo de Chudak e Shmoys utilizando as soluções relaxadas retornadas pelo Simplex isso não ocorreu para nenhuma instância do FLP que gerava uma solução relaxada fracionária.

6 Conclusão

Neste projeto, investigamos algoritmos de aproximação polinomiais para resolver o Problema da Localização de Instalações [11] (FLP), tentando atingir tempos de execução e aproximações próximas ou até melhores, em casos específicos, de algoritmos projetados até hoje. Para atingir este objetivo, o projeto foi dividido em três etapas.

Na primeira etapa, implementamos o algoritmo proposto por Chudak e Shmoys[1] para resolver o FLP dada a solução do programa linear relaxado retornado pelo Simplex, comparando as aproximações obtidas com relação à solução ótima. Nela, pudemos concluir que o limitante em performance do nosso algoritmo se dá na resolução do PL já que o tempo de execução do algoritmo de Chudak e Shmoys é muito inferior ao tempo de execução do Gurobi resolvendo um PL. Também pudemos observar fatores de aproximação muito melhores que a 1.736-aproximação garantida pelo algoritmo de Chudak e Shmoys, apresentando na prática soluções muito próximas da solução ótima (utilizando $\gamma = 1.2$ obtivemos fatores de aproximação menores que 1.13).

Na segunda etapa, desenvolvemos uma adaptação do algoritmo de aproximação de Young[12] para resolver programas lineares mistos de packing e covering, otimizado para resolver o problema da cobertura de conjuntos[11], comparando a qualidade das aproximações e o tempo de execução com relação ao Simplex. Nesta etapa verificamos que o algoritmo implementado consegue ser mais eficiente que o Gurobi para instâncias pequenas, contudo, o ganho que não é muito elevado (é possível garantir aproximações entre 1.1 e 1.3 com speedup entre 10 e 40 utilizando $\kappa = 50$ e $\epsilon = 0.57$). Já para instâncias médias o algoritmo implementado se mostra mais escalável que o Gurobi para problemas com instâncias esparsas (número de elementos nos subconjuntos reduzido) cuja performance ficou muito superior, com speedups na casa das centenas e até milhares. Nas instâncias médias e densas (número de elementos nos subconjuntos elevado) o algoritmo implementado também conseguiu atingir performances melhores que o Gurobi (é possível garantir fatores de aproximação menores que 1.1 com speedup entre 10 e 30 utilizando $\kappa = 50$ e $\epsilon = 0.18$).

Na terceira etapa, utilizamos a redução proposta por Young para resolver o FLP reduzido a cobertura de conjuntos[13], e assim, comparar aproximações e tempos de execução com relação à resolução do programa linear inteiro original utilizando o Gurobi. A penalização na redução implicou uma perda severa na performance, contudo, ainda assim foi possível bater o tempo de execução do Gurobi devido às otimizações no algoritmo de Young para instâncias esparsas, sendo possível garantir

em todas as instâncias testadas a 1.736-aproximação do algoritmo de Chudak Shmoys com speedups maiores que 1. Além do mais, para instâncias menores, é possível obter fatores de aproximação observados médios de 1.11 com speedups entre 5 e 50 (utilizando $\kappa = 30$, $\epsilon = 1.00$ e $\gamma = 1.2$) e para instâncias maiores, é possível obter fatores de aproximação observados médios de 1.21 com speedups entre 3 e 13 (utilizando $\kappa = 50$, $\epsilon = 1.00$ e $\gamma = 1.2$).

7 Trabalhos Futuros

Durante o projeto, foram estudados algoritmos baseados sobre fluxo de rede, por exemplo, o algoritmo de Garg et al. [4] para resolução de programas lineares de packing. Contudo, o FLP não é um algoritmo de packing puro, portanto, o algoritmo descrito por Garg et al. não pode ser utilizado (da maneira que foi proposto) para resolver o FLP. Como a eficiência do algoritmo para resolver o FLP proposto neste projeto foi limitado pelo algoritmo de resolução do PLI relaxado, talvez seria interessante estudar outras formas de resolver programas lineares mistos de packing e covering, talvez, baseados em fluxo de rede como o algoritmo de Garg et al, adaptado para resolver também problemas de covering (se existir tal maneira).

Outra possibilidade é estudar os algoritmos de Young em [13]. Neste artigo, são apresentados algoritmos de aproximação com complexidade linear voltados para a resolução de PLs do FLP sem a necessidade de reduzir explicitamente o problema à cobertura de conjuntos (causa da perda de performance do algoritmo implementado na última etapa do projeto). Esse artigo somente foi estudado após a implementação da redução do FLP para cobertura de conjuntos (fase final do projeto), e por falta de tempo, optou-se por realizar a redução já que era a solução mais simples e de fácil implementação. Aqueles algoritmos unidos ao algoritmo de Chudak Shmoys aparentam ter resultados promissores e acreditamos valer a pena investigá-los futuramente.

Referências

[1] J. Byrka and K. Aardal. An Optimal Bifactor Approximation Algorithm for the Metric Uncapacitated Facility Location Problem. *SIAM Journal on Computing*, 39(6):2212–2231, 2010.

[2] Fabián A. Chudak and David B. Shmoys. Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal on Computing*, 33(1):1–25, 2004.

[3] George B. Dantzig. Reminiscences about the origins of linear programming. *Operations Research Letters*, 1(2):43 – 48, 1982.

[4] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing*, 37(2):630–652, 2007.

[5] Kamal Jain and Vijay V. Vazirani. Approximation Algorithms for Metric Facility Location and k-Median Problems Using the Primal-dual Schema and Lagrangian Relaxation. *J. ACM*, 48(2):274–296, March 2001.

[6] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer Publishing Company, Incorporated, 5th edition, 2012.

[7] Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Information and Computation*, 222(0):45–58, 2013. 38^{th} International Colloquium on Automata, Languages and Programming (ICALP 2011).

[8] Brunel University London. Setcover instances from or-library.

[9] Brunel University London. Ufl instances from or-library.

[10] Computer Science Departament University of Pisa. Ufl instances from computer science departament.

[11] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, Cambridge, 2011.

[12] N. E. Young. Sequential and parallel algorithms for mixed packing and covering. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 538–546, Oct 2001.

[13] Neal E. Young. Nearly linear-work algorithms for mixed packing/covering and facility-location linear programs. 2016.

Figura 1: Fatores de aproximação e razão de tempo resolvendo algoritmo de Chudak e Shmoys com relação ao tempo total de execução para as instâncias do dataset Holmberg para diferentes fatores γ

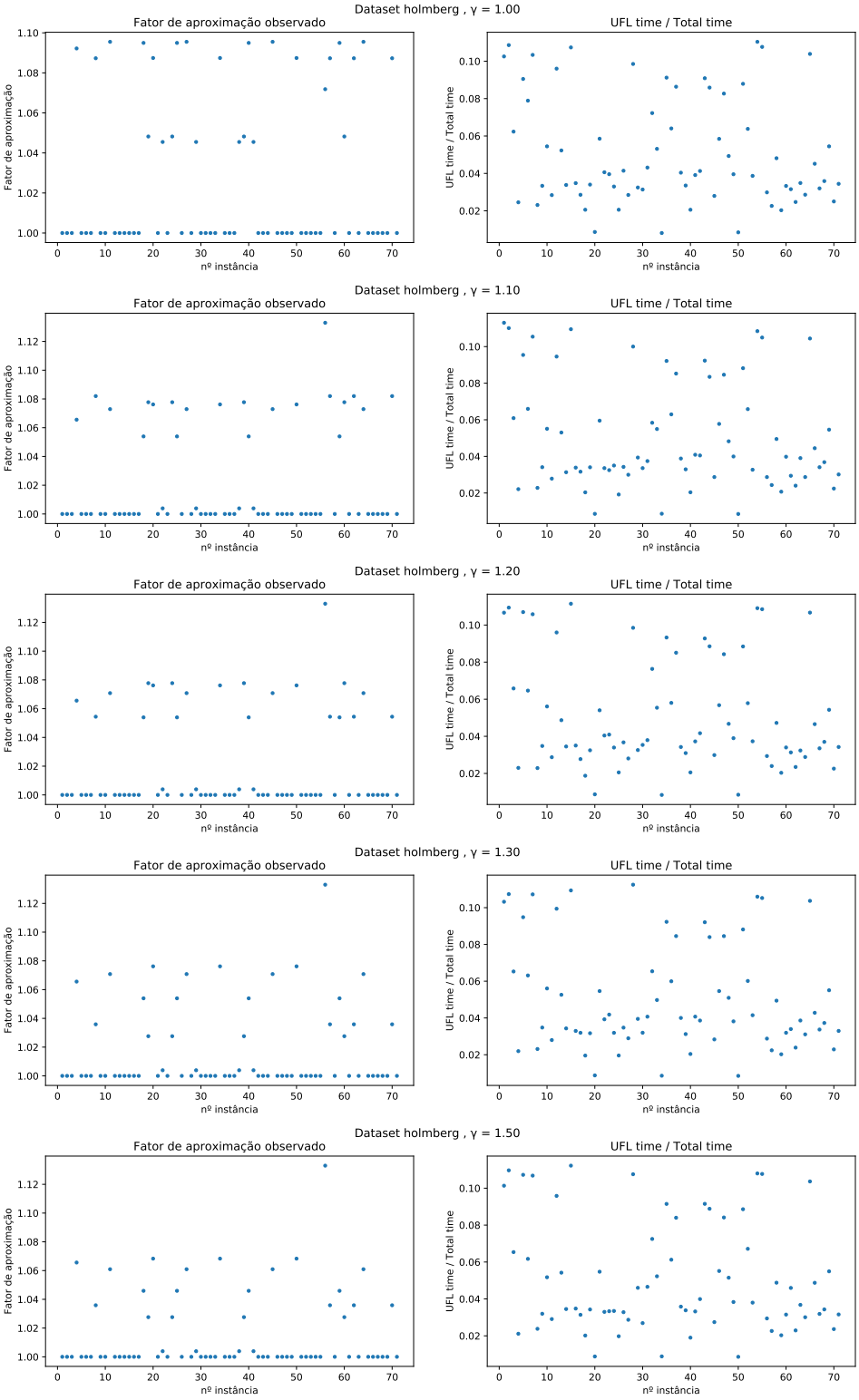

Figura 2: Fatores de aproximação e razão de tempo resolvendo algoritmo de Chudak e Shmoys com relação ao tempo total de execução para as instâncias do dataset Beasley para diferentes fatores γ

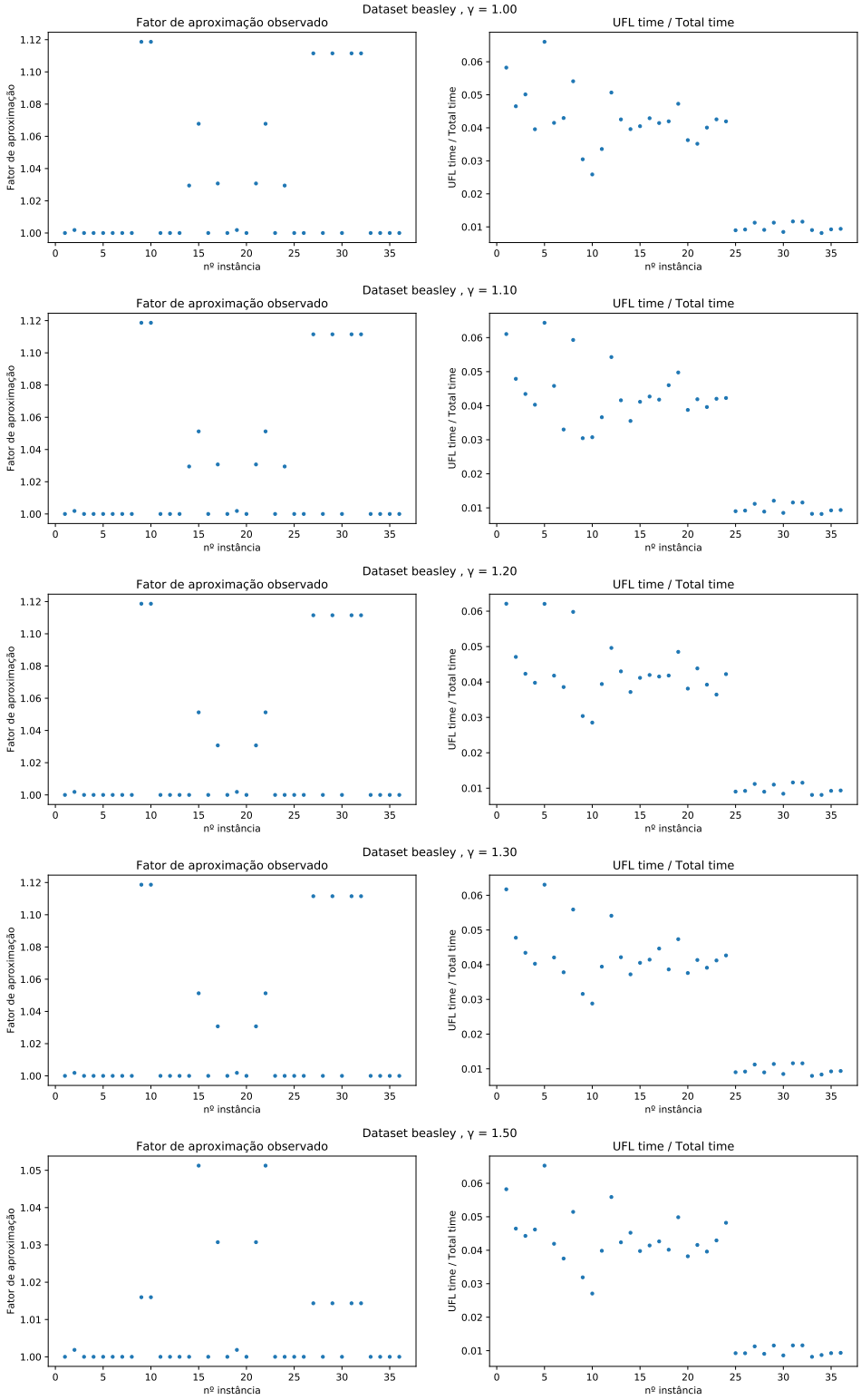

Figura 3: Fatores de aproximação e Speedup (log 10) para as instâncias do dataset small para κ entre 1 e 100 e ϵ entre 0.32 e 1.00

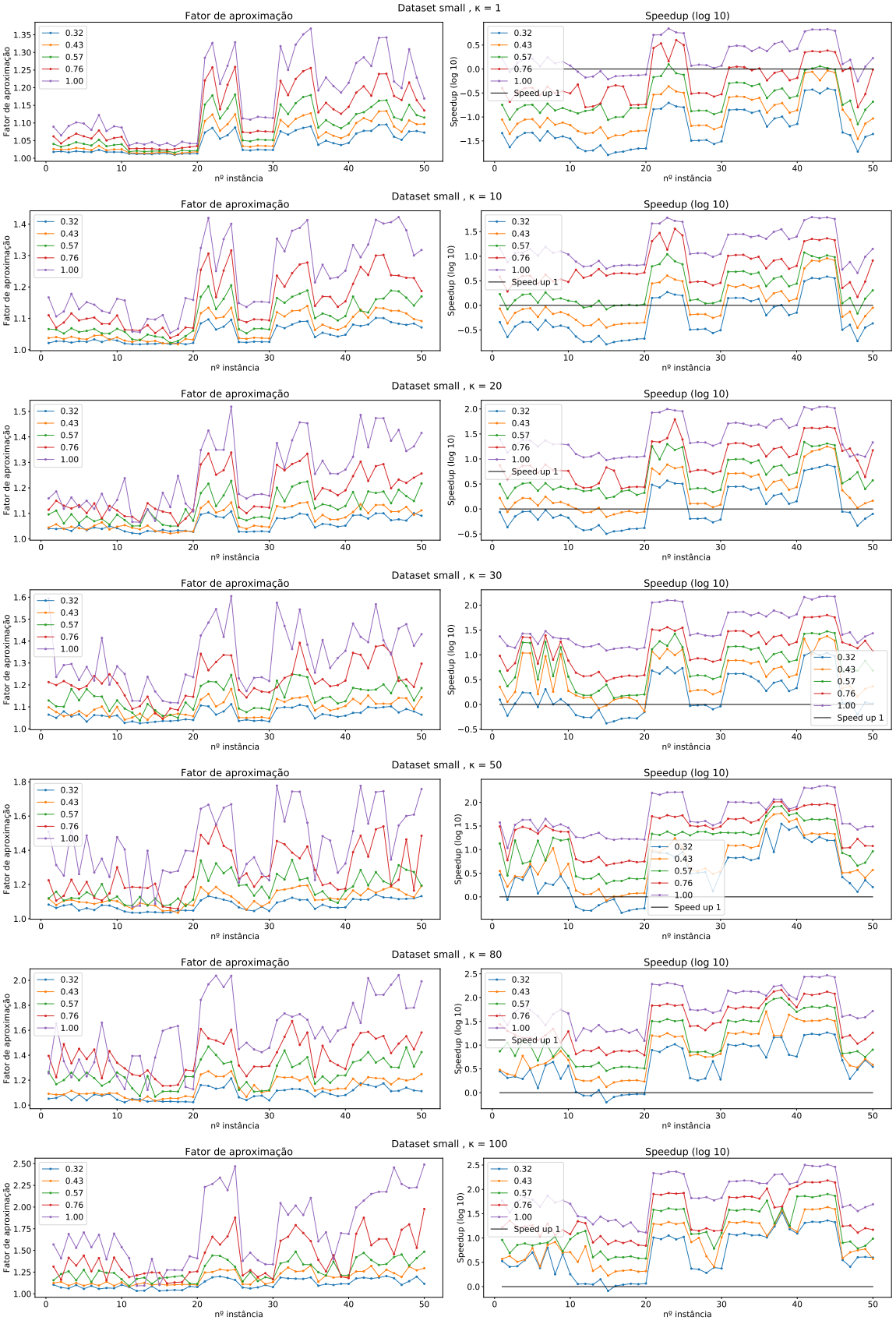

Figura 4: Fatores de aproximação e Speedup (log 10) para as instâncias do dataset small para κ entre 1 e 100 e ϵ entre 0.08 e 0.24

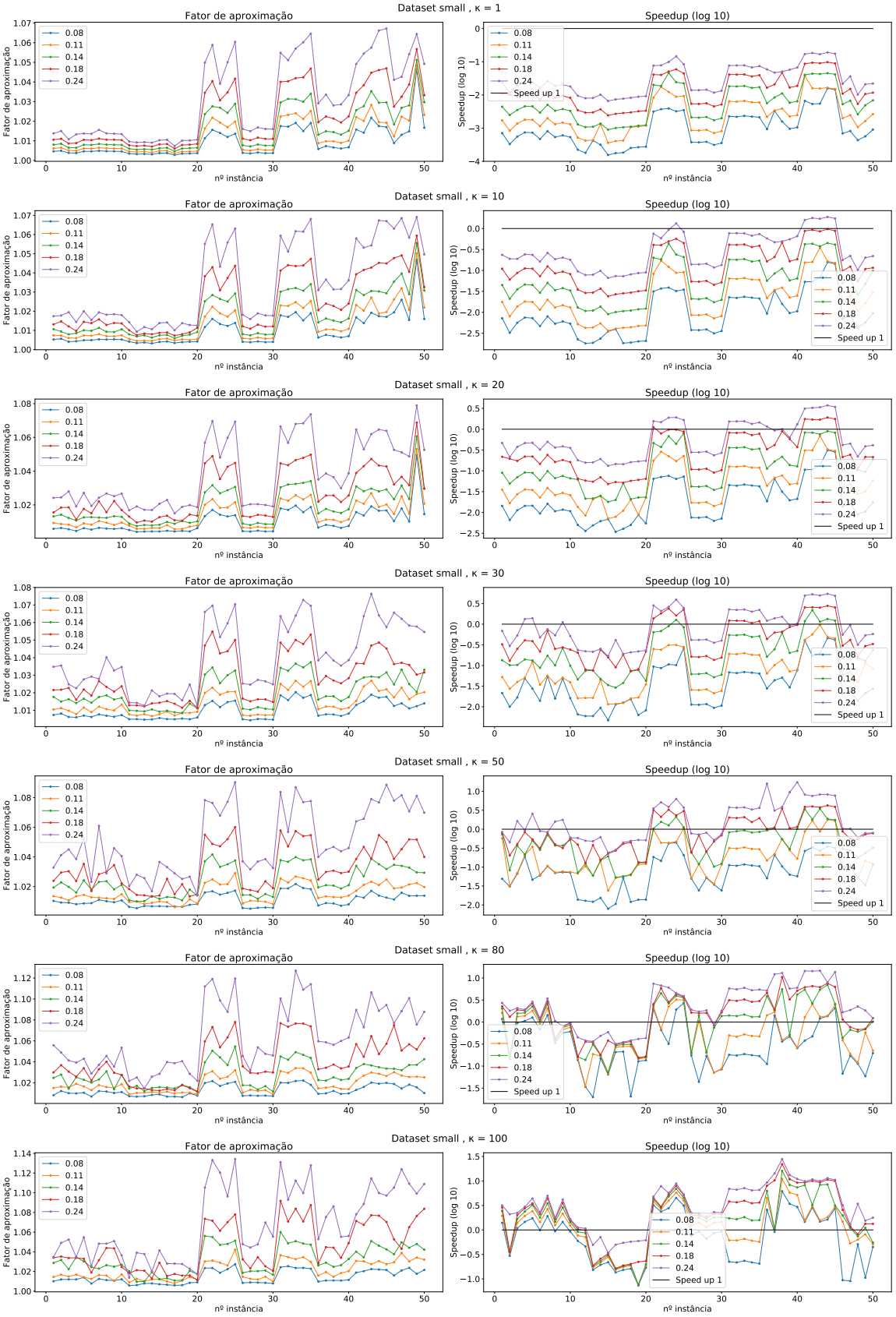

Figura 5: Fatores de aproximação e Speedup (log 10) para as instâncias do dataset medium para κ entre 1 e 100 e ϵ entre 0.32 e 1.00

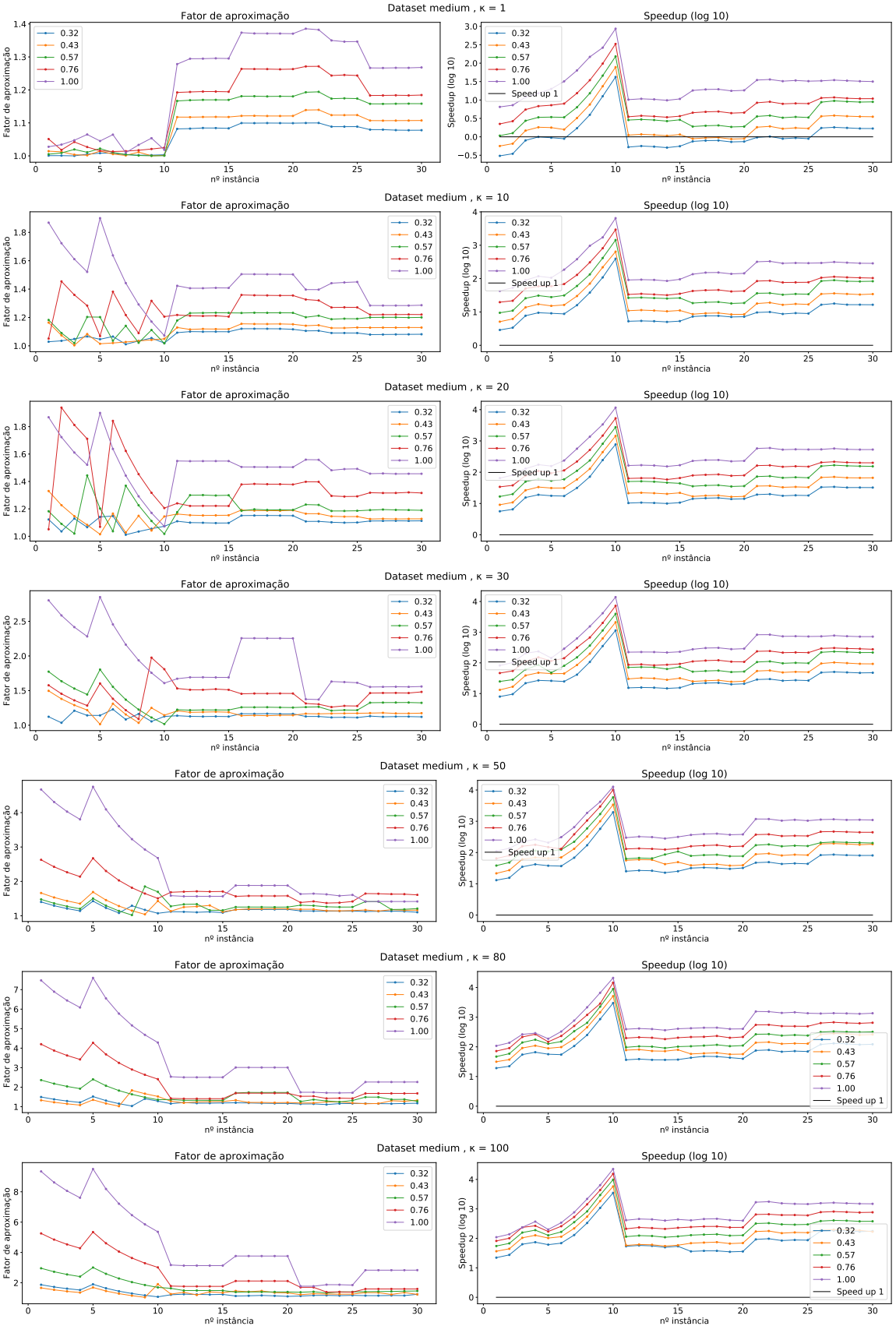

Figura 6: Fatores de aproximação e Speedup (log 10) para as instâncias do dataset medium para κ entre 1 e 100 e ϵ entre 0.08 e 0.24

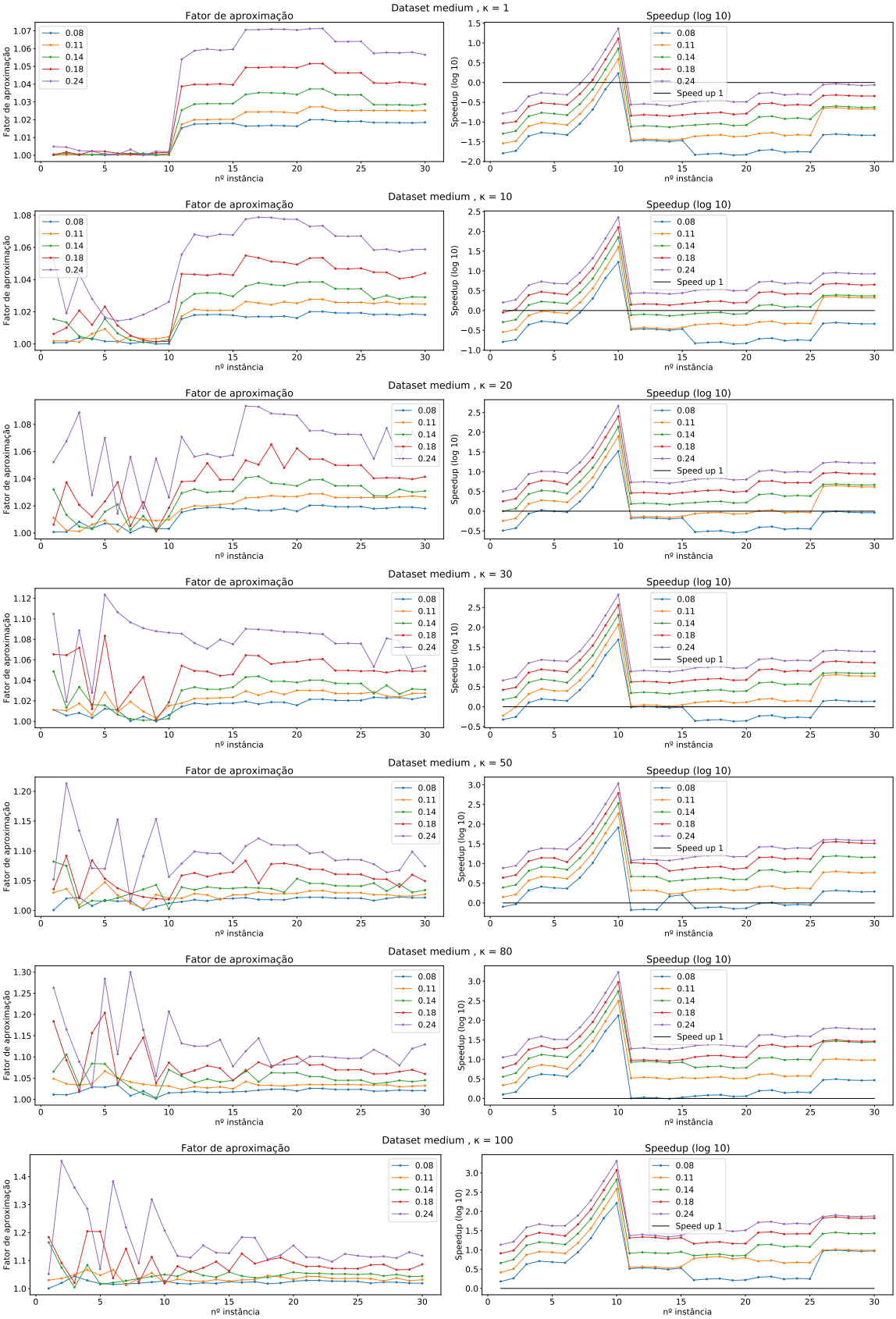

Figura 7: Fatores de aproximação e Speedup para as instâncias do dataset Holmberg resolvidas pelo algoritmo de Chudak Shmoys + algoritmo de Young para $\epsilon = 1.00$

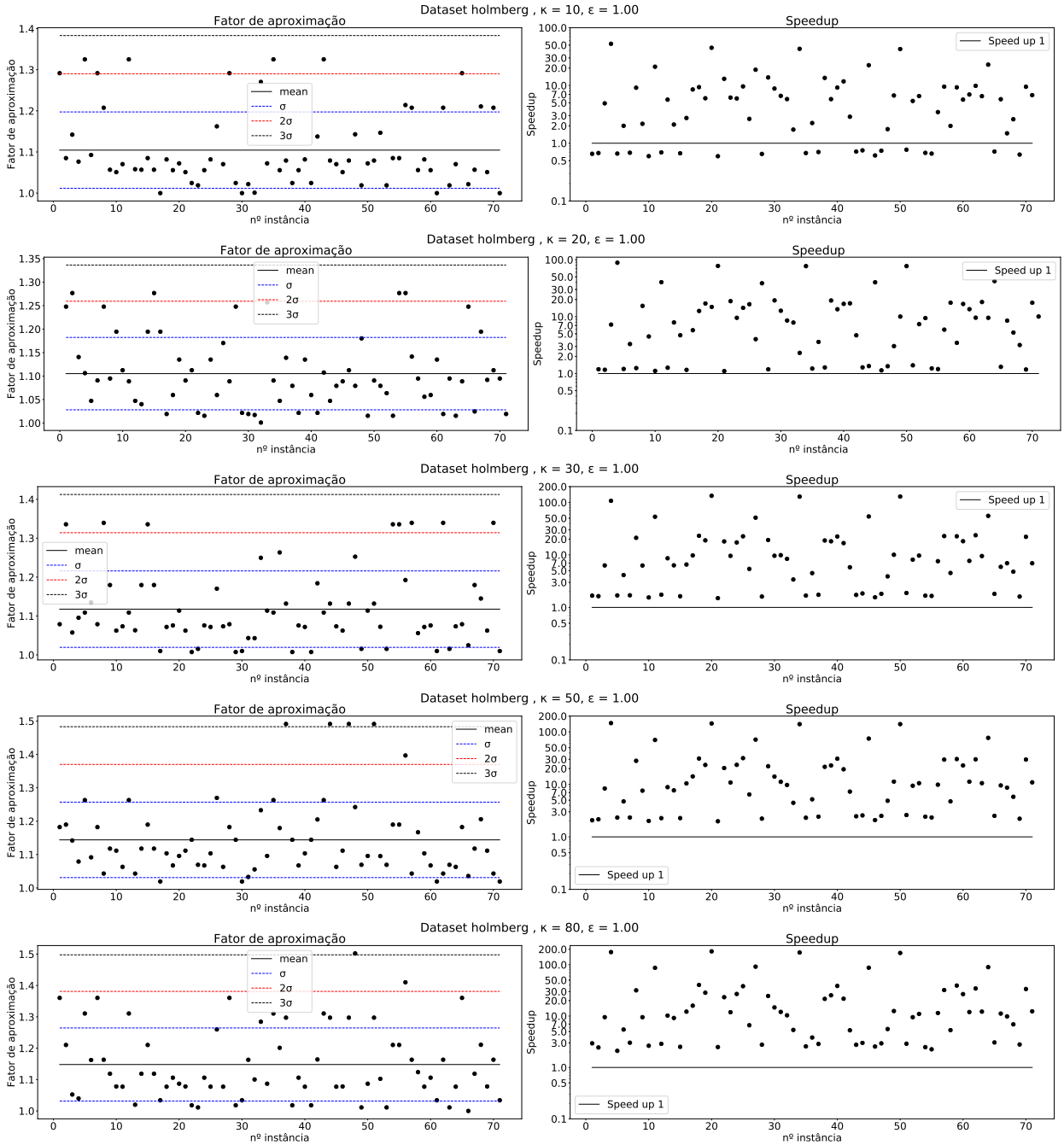

Figura 8: Fatores de aproximação e Speedup para as instâncias do dataset Holmberg resolvidas pelo algoritmo de Chudak Shmoys + algoritmo de Young para $\epsilon = 0.76$

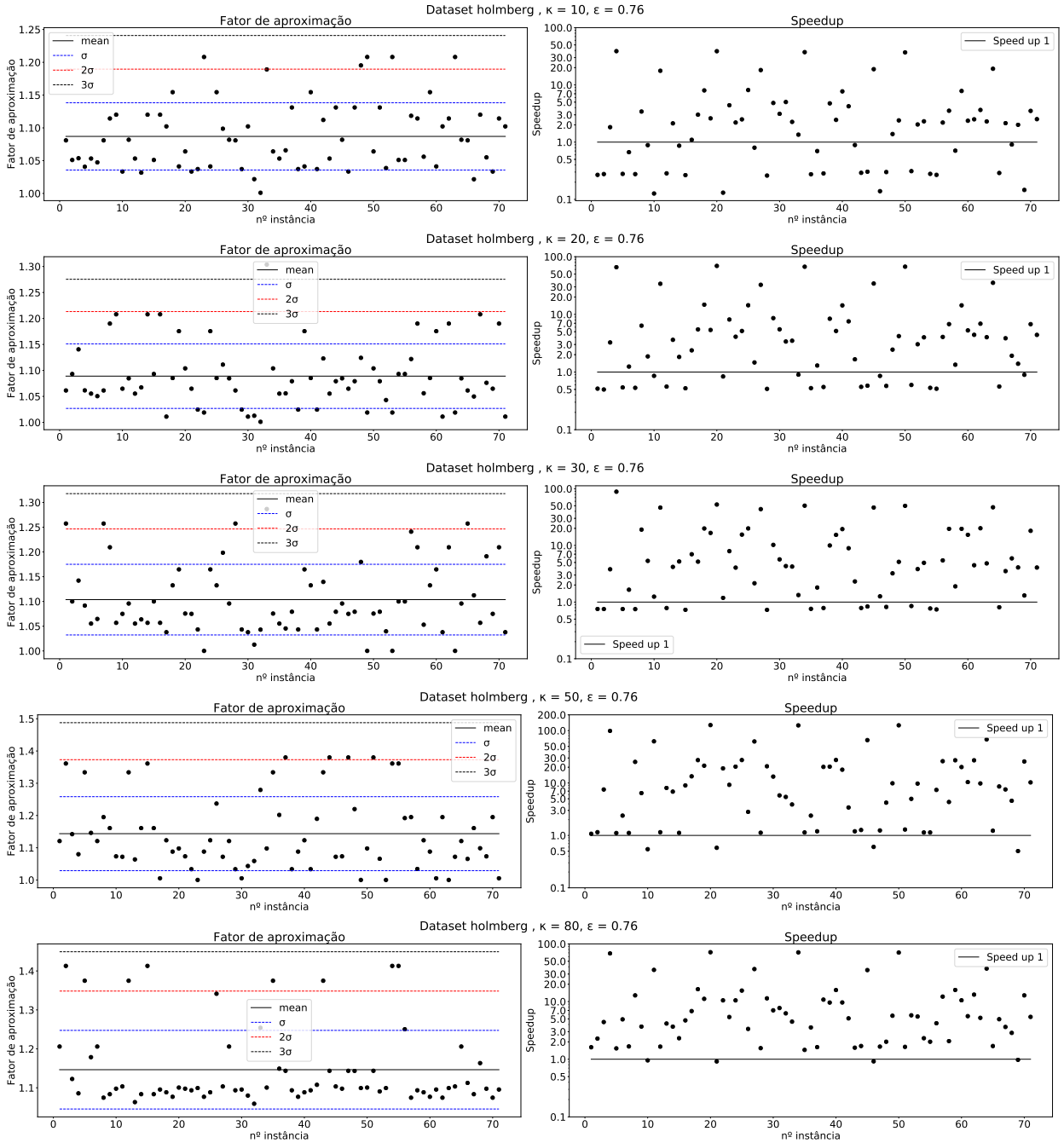

Figura 9: Fatores de aproximação e Speedup para as instâncias do dataset Holmberg resolvidas pelo algoritmo de Chudak Shmoys + algoritmo de Young para $\epsilon = 0.56$

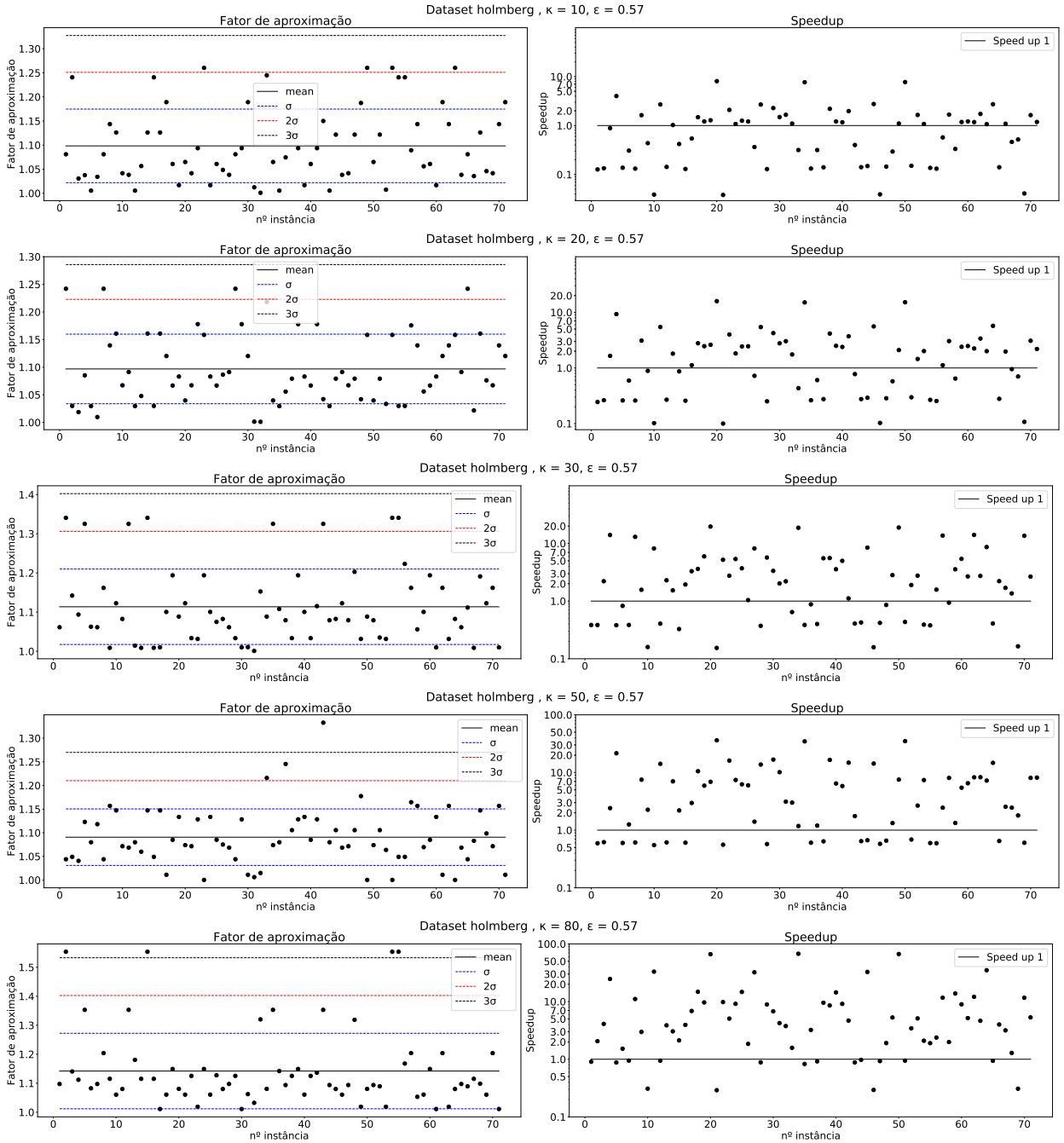

Figura 10: Fatores de aproximação e Speedup para as instâncias do dataset Beasley resolvidas pelo algoritmo de Chudak Shmoys + algoritmo de Young para $\epsilon = 1.00$

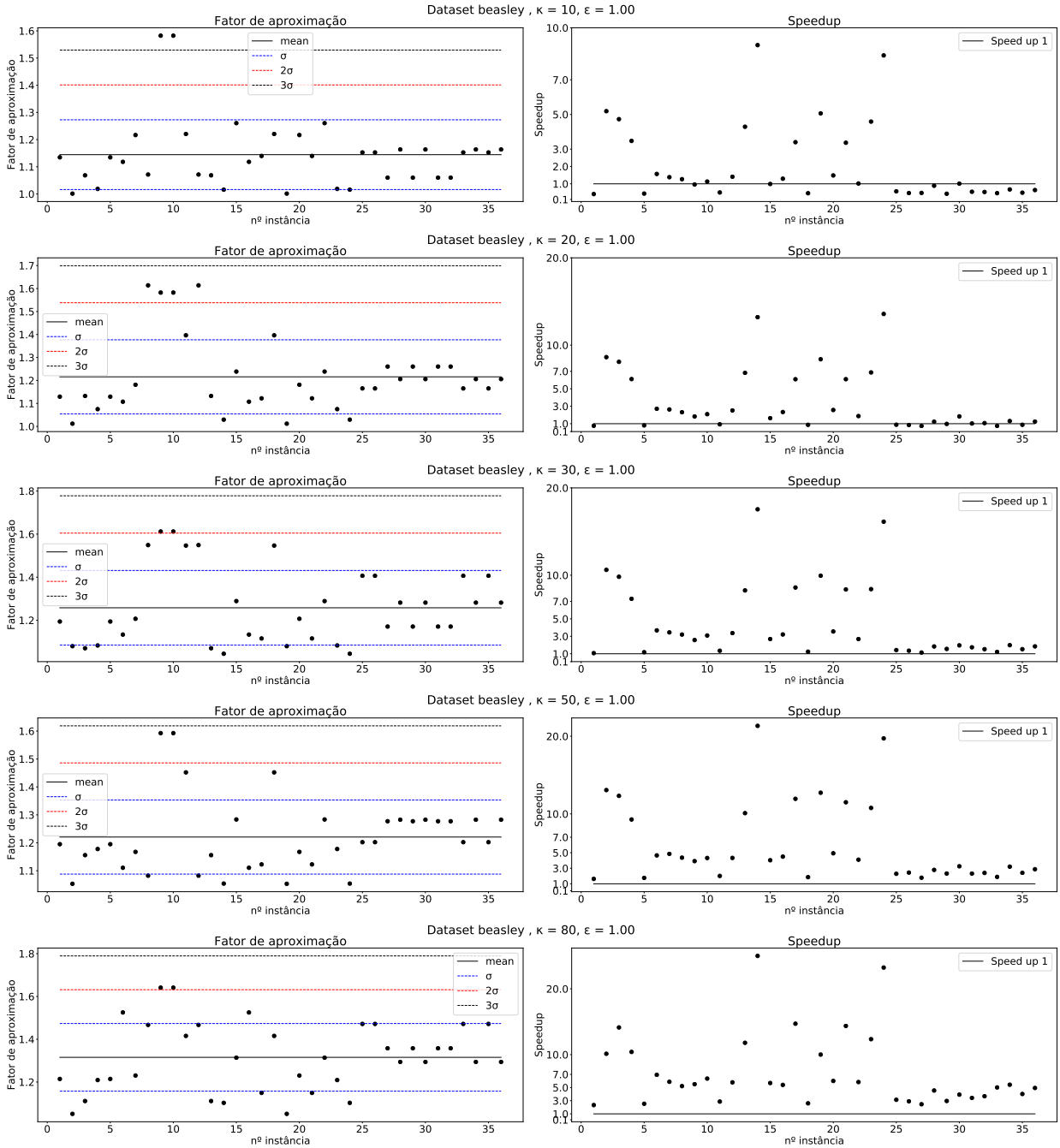

Figura 11: Fatores de aproximação e Speedup para as instâncias do dataset Beasley resolvidas pelo algoritmo de Chudak Shmoys + algoritmo de Young para $\epsilon = 0.76$

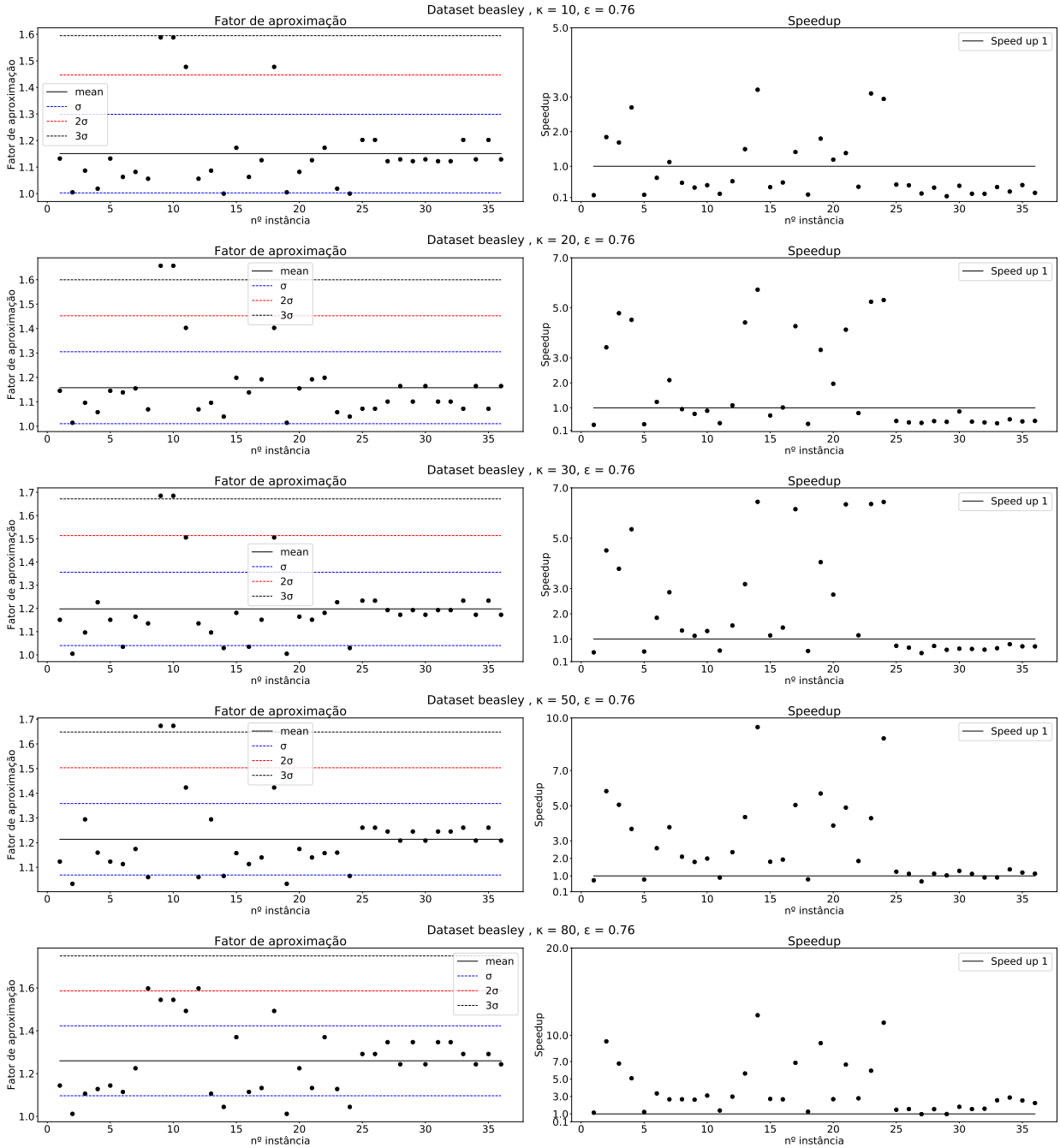

Figura 12: Fatores de aproximação e Speedup para as instâncias do dataset Beasley resolvidas pelo algoritmo de Chudak Shmoys + algoritmo de Young para $\epsilon = 0.56$