Um algoritmo GRASP para o problema de localização de instalação conectada

Alessandro Tomazic * Ivana Ljubić´ †
Faculdade de Administração, Economia e Estatística
Universidade de viena
Brü¨nnerstr. 72, 1210 Viena, Áustria
{alessandro.tomazic, ivana.ljubic}@univie.ac.at

Abstrato

Aplicamos um Greedy Randomized Adaptive Search Procedure (GRASP) para resolver heuristicamente o problema de localização da instalação conectada. A propriedade de diversificação é assegurada pela aplicação de um algoritmo guloso aleatório para construir soluções viáveis de uma forma multi-start. Os elementos de intensificação são garantidos devido a duas técnicas de busca local baseadas em instalações. O estudo computacional é conduzido em um conjunto parametrizado de instâncias de benchmark geradas aleatoriamente. Os resultados obtidos refletem a qualidade da abordagem proposta no que diz respeito à qualidade das soluções e ao esforço computacional, por comparação com os limites inferiores obtidos a partir de um framework Branch-and-Cut.

1. Introdução

Dado um gráfico conectado não direcionado G com vértices V e o conjunto de arestas E, uma partição não trivial $P = (F, C)$ de V identificando *instalações* e *clientes, custos de borda* $c: E 7 \rightarrow$ Q + e uma função $p: F 7 \rightarrow$ Q + que define o *custos de abertura* para as instalações, o *Problema de localização de instalação conectada* é definido como encontrar uma subárvore conectada $T = (V [T], E [T])$, de tal modo que $C \subset V [T]$ e $G [F \cap V [T]]$ está conectado, o que minimiza a função objetivo

$$\sum_{v \in V[T] \cap N_T(C)} p(v) + \sum_{e \in E[T]} c(e).$$

Com $N_T(C)$, denotamos os nós vizinhos de C dentro T. Em uma solução T com uma borda conectando uma instalação v com cliente *você*, nós dizemos isso v é *abrir* e *você* é *fornecido* de v. *Fornecedores potenciais* F_p são aqueles nós de F que tem em pelo menos um vizinho em C, ie $F_p = N_G(C) \cap F$.

Resulta imediatamente da definição que em cada nós de clientes de solução viável devem ser deixados. Lá-antes, dado um gráfico G, as bordas entre dois clientes podem ser excluídas, sem perda de generalidade.

No projeto de redes de telecomunicações, o seguinte problema pode ser modelado como ConFL: construir uma rede de última milha substituindo conexões desatualizadas de cobre por fibra óptica, colocando assim multiplexadores para alternar entre elas. Conecte os multiplexadores entre si, de modo que os custos de conexão e instalação sejam minimizados.

O problema ConFL foi introduzido em [4], e a razão de aproximação mais conhecida de 4,23 foi obtida recentemente por Eisenbrand et al. [2]. Ljubić´ [5] concentrou-se na versão enraizada do problema, em que uma instalação r está aberto sem nenhum custo.

Na próxima seção, mostramos como integrar uma heurística gananciosa e dois métodos de pesquisa local em um APERTO estrutura. A seção 3 descreve como o problema pode ser transformado no Problema de Arborescência de Steiner Mínimo (SA) em gráficos. Também propomos resolver o SA usando um algoritmo branch-and-cut (B&C) cujos limites são então usados na seção 4 para medir a qualidade de nossa abordagem heurística. A seção 5 fornece algumas conclusões e idéias para melhorar o algoritmo.

2 O Algoritmo GRASP

UMA APERTO [3] é uma abordagem iterativa multi-start para problemas de otimização combinatória onde cada iteração consiste em duas fases: construção gananciosa e melhoria local. A melhor solução geral é relatada como a final.

Em cada iteração do APERTO algoritmo, usamos um algoritmo aleatório *Ambicioso* procedimento para construir uma solução viável. Como um mecanismo de intensificação de pesquisa, aplicamos iterativamente *abrir-* e *fechar instalação* move, seguido por uma heurística de árvore de Steiner de caminho mais curto para encontrar soluções localmente ideais. Todo o processo é iterado de forma multi-start até que um número pré-especificado de iterações seja alcançado.

A propriedade de diversificação é assegurada pelo aumento dinâmico do número de candidatos potenciais a serem inseridos no procedimento guloso aleatório.

* Apoiado pela Agência Austríaca de Promoção de Pesquisa, FFG, dentro do programa Bridge 2 (812973)
† Apoiado pela Hertha-Firnberg Fellowship da Austrian Science Foundation (FWF)

978-0-7695-3297-4 / 08 $ 25,00 © 2008 IEEE
DOI 10.1109 / SAINT.2008.64

2.1. A construção gananciosa aleatória

Para a fase de construção um determinístico *Ambicioso* heurística apresentada a seguir é modi fi cada. O algoritmo abre incrementalmente fornecedores em potencial e os conecta em uma árvore *T,* até que todos os clientes sejam atendidos.

Algoritmo ganancioso As instalações de abastecimento potenciais F_p são classificados em ordem crescente de acordo com o seguinte critério terion:

$$\frac{p\,(v)\,dist\,(v,\,T)}{deg_{C'(}\,v)+1},\ v \in F_p \qquad (1)$$

Onde $dist\,(v,\,T)$ é o comprimento de um caminho mais curto de v para um vértice em T em relação a c. Se $T = \varnothing$ ou $v \in V\,[T]$, $dist\,(v,\,T) = \min_{u,\,v \in F} c\,(u,\,v)$. $deg_{C'(}\,v)$ é o número de clientes não fornecidos que podem ser atendidos por v. Por meio deste, uma prioridade mais alta é dada aos vértices próximos à subárvore parcialmente construída T, incidente para muitos clientes não fornecidos ou com baixos custos de abertura.

Entre todos os fornecedores potenciais fechados, deixe v seja o melhor de acordo com (1). Nós abrimos v e conectar todos os clientes vizinhos a ele se já não for fornecido mais barato. Então nós estendemos T adicionando o caminho mais curto de v para T. Posteriormente, as instalações são reatribuídas onde apropriado, e se algumas das instalações abertas anteriormente forem fechadas, a rede da instalação é redesenhada pela aplicação do *Heurística de caminho mais curto [* 7] descrito abaixo.

O algoritmo abre instalações iterativamente dessa maneira até que todos os clientes sejam atendidos.

Heurística de árvore de Steiner de caminho mais curto (SPH) Para um conjunto de instalações abertas, usamos uma heurística para redesenhar a rede de instalações. O subproblema básico neste estágio é o Problema da Árvore de Steiner com o conjunto de instalações abertas escolhidas como terminais e as instalações restantes como nós de Steiner. Portanto, aplicamos a conhecida Heurística de Caminho Mais Curto, que retorna com eficiência uma aproximação da conexão ótima entre instalações abertas. Assim, a árvore é estendida iterativamente adicionando o caminho mais curto para o nó terminal não conectado mais próximo, começando com um arbitrário.

Denotar com $G\,[F]$ o subgrafo induzido por F, e com $E\,[F]$ o conjunto correspondente de arestas. A complexidade computacional do SPH é $O\,(|\,F\,|\,(|\,E\,[F]\,|\,\log\,|\,F\,|)\,($ veja [1] para a explicação de uma implementação eficiente usando o algoritmo de caminho mais curto de Dijkstra e pilhas binárias).

Ganancioso aleatório Em uma versão multistart com alternativas iniciais, adicionamos uma tolerância à seleção de instalações a serem abertas. Nossa lista de candidatos de comprimento k simplesmente consistirá em o primeiro k vértices em F_p - classificados em relação a (1) - que ainda não foram abertos. Então, em cada etapa do método, nós aleatoriamente escolha um entre aqueles k instalações. Configurando o parâmetro k para 1 corresponde à versão não aleatória do algoritmo. Em nossa implementação, o número de candidatos é definido dinamicamente dentro do APERTO procedimento e aplica-se para um completo *Ambicioso* corre. Começamos com $k = 1$ e linearmente (arredondado para inteiros) aumente até 20/03 do número de vértices disponíveis após cada *Ambicioso* inserção.

Uma vez que em cada iteração o algoritmo de Dijkstra e o SPH são chamados, a complexidade total do tempo de execução de *Ambicioso* é $O\,(|\,F\,|\,(|\,F\,|\,|\,E\,[F]\,|\,\log\,|\,F\,|\,+\,|\,C\,|))$.

2.2. Técnicas de Pesquisa Local

Para intensificar a busca nas regiões locais de tal solução gananciosa construída, exploramos *abrir-* e *closefacility* bairros em duas fases: primeiro procuramos a solução ideal localmente com respeito à vizinhança de instalação aberta, na segunda fase procuramos a melhoria aplicando sequencialmente *fechar instalação* movimentos. No caso de fechamento de instalação, o *Heurística de caminho mais curto* é reaplicado.

Nossos estudos computacionais mostraram que outras repetições das duas fases seriam mais demoradas do que benéficas. Portanto, decidimos aplicá-los apenas uma vez por APERTO iteração.

Um conjunto de referências relacionadas às técnicas de busca de vizinhança para o Problema da Árvore de Steiner pode ser encontrado em [1], por exemplo.

Movimentos de instalação aberta Um potencial fornecedor fechado $v \in F_p$ é aberto para verificar se o fornecimento de clientes resulta em redução de custos. Assim, consideramos os custos de abertura de v, os custos de conexão de clientes e a economia de custos de conexão pagos anteriormente para cada cliente reconectado. Se isso reduzir o valor objetivo, realizamos as operações. Ao reconectar os clientes, podemos acabar encontrando uma instalação aberta *você* não fornecendo quaisquer clientes. Nesse caso *você* pode ser fechado e seus custos de abertura são levados em consideração ao avaliar a abertura de v. A complexidade computacional de um único movimento é $O\,(|\,F\,|\,|\,E\,[F]\,|\,\log\,|\,F\,|\,+\,|\,C\,|)$.

Para a exploração total do bairro, precisamos $O\,(|\,F\,|\,(|\,F\,|\,|\,E\,[F]\,|\,\log\,|\,F\,|\,+\,|\,C\,|))$ Tempo.

Movimentos de fechamento de instalação Fechando uma instalação v só pode resultar em uma solução viável quando todos os clientes puderem ser abastecidos por outras instalações abertas. Aqui nos concentramos nos abertos e tentamos reconectar seus clientes de forma que nos custe menos do que ganhamos fechando v. O fechamento de uma instalação também pode ter impacto na estrutura de nossa rede de instalações. Desde a v não é mais indispensável por causa de sua função de abastecimento, resolver o problema da árvore de Steiner correspondente retornaria uma rede de instalações ótima para a situação de abastecimento atual. Isso é aproximado executando a Heurística de árvore de Steiner de caminho mais curto descrita anteriormente. Um único movimento, nesse caso, tem uma complexidade computacional de $O\,(|\,F\,|\,(|\,E\,[F]\,|\,\log\,|\,F\,|\,+\,|\,C\,|))$. Para a exploração total do bairro, precisamos $O\,(|\,F\,|_{2(|}\,E\,[F]\,|\,\log\,|\,F\,|\,+\,|\,C\,|))$

Tempo. Ao usar filas prioritárias, podemos melhorar essa complexidade computacional. Ao explorar os bairros, consideramos *primeira melhoria* estratégia onde as instalações a serem abertas / fechadas são sempre processadas em uma ordem aleatória

para evitar que o algoritmo fique preso no mesmo ótimo local.

2.3. Integração no GRASP-Framework

O *Ambicioso-* heurística junto com as técnicas de busca local são embutidas em uma estrutura GRASP conforme descrito no Algoritmo 1.

Algoritmo 1 O GRASP-Framework

Entrada: ($G = (V, E), c, p$)
Resultado: Uma solução ConFL viável (T, A)
1: para $iter = 0$ para $iter = | F_p | / 5$ Faz
2: (T, A) = *Ambicioso*(G, c, p)
3: *OpenFacilityLocalSearch* (G, c, p, T, A)
4: *CloseFacilityLocalSearch* (G, c, p, T, A)
5: fim para

3 Relação com o Steiner mínimo Ar-

Problema de boroscena

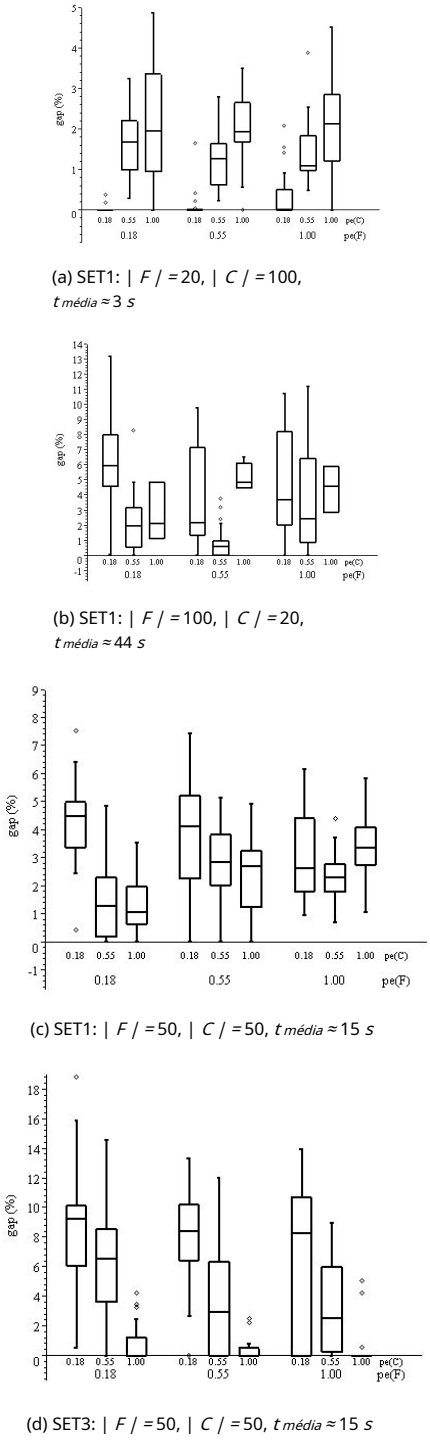
Dado um gráfico conectado direcionado $G_A = (V_{UMA}, A)$ com função de custos nos arcos $w: A 7 \rightarrow$ Q +, com uma raiz $r \in V_{UMA}$ e um conjunto de *terminais* $T_{UMA} \subset V_{UMA}$, a *Steiner mínimo Problema de Arborescência* procura por uma subárvore enraizada de G_{UMA} de custos mínimos, de modo que haja um caminho direcionado de r a cada $v \in T_{UMA}$.

Cada problema ConFL em um gráfico $G = (V, E)$ pode ser transformado em um problema de SA da seguinte maneira:

1. Apresente uma raiz artificial r e para todo potencial fornecedor $v \in F_p$ sua contraparte v': Definir $V_A = V \cup \{r\} \cup \{v' | v \in F_p\}$.
2. Conecte todos os fornecedores potenciais para r, ie definido $A = \{(r, v) | v \in F_p\}$ e $w(r, v) = 0 \ \forall v \in F_p$.
3. Conecte todas as instalações entre si: $A = A \cup \{(u, v), (v, u) | u, v \in F\}$ e $w(u, v) = w(v, u) = c(u, v), u, v \in F$.
4. Divida os fornecedores potenciais: $A = A \cup \{(v, v') | v \in F_p\}$. Definir $w(v, v') = p(v), v \in F_p$.
5. Conecte instalações e clientes apenas em uma direção: $A = A \cup \{(v, v_c) | v \in F_p, v_c \in C\}$ e $w(v, v_c) = c(v, v_c)$.

Em tal caso obtido, obviamente não há despesas arcos de qualquer nó do cliente $v_c \in C$, e há apenas arcos saindo da raiz r. Para garantir a viabilidade de uma solução ConFL que é obtida removendo r e seus arcos adjacentes, e contraindo nós divididos novamente, solicitamos que o grau de saída de r deve ser igual a um.

Para resolver o problema mínimo de SA em G_{UMA} para a otimização, usamos uma adaptação do algoritmo branch-and-cut pro colocado em [6]. Os limites obtidos com este algoritmo são apresentados na próxima seção.



(a) SET1: | F | = 20, | C | = 100, $t_{média} \approx 3$ s

(b) SET1: | F | = 100, | C | = 20, $t_{média} \approx 44$ s

(c) SET1: | F | = 50, | C | = 50, $t_{média} \approx 15$ s

(d) SET3: | F | = 50, | C | = 50, $t_{média} \approx 15$ s

Figura 1. Lacunas percentuais entre as soluções GRASP e os valores ideais.

4 resultados computacionais

O procedimento foi testado em um conjunto de gráficos gerados aleatoriamente com pesos inteiros aleatórios [1] Bordas da rede

[1] Todas as instâncias estão disponíveis em homepage.univie.ac.at/alessandro.tomazic.

G [F] são gerados com probabilidade *pe (F)* ∈ { 0,1, 0,5, 1}, Considerando que as conexões entre instalações e clientes são estabelecidas com probabilidade *pe (C)* ∈ { 0,18, 0,55, 1}.

Os limites inferior e superior para os pesos das bordas foram definidos para 50 e 100 Geramos dois conjuntos, SET1 (| *F* | × | *C* | ∈ { 100 × 20, 20 × 100, 50 × 50}) e SET2 (100 × 100) de 81 e 27 instâncias, respectivamente, com custos de abertura de instalação atribuídos aleatoriamente a valores entre 150 e 200 Por 27 adicionais 50 × 50 instâncias (SET3) atribuímos aleatoriamente os custos de abertura da instalação a valores entre 800 e 1600.

Depois de deixar MAPLE gerar uma rede de instalação aleatória *G [F],* vinculamos clientes aleatoriamente aos vértices existentes usando os parâmetros fornecidos acima. Isso resultou em casos em que quase sempre todas as instalações eram fornecedores em potencial.

Nossa implementação C ++ do GRASP foi testada em um Intel Core 2 Duo E4300 com 1,8 GHz, 3,25 GB de RAM.

SET1 e SET3: Cada caixa nos diagramas da Figura 1 reflete os resultados de 10 APERTO é executado em 3 instâncias aleatórias com as mesmas propriedades de construção (30 resultados). O tempo de cálculo que nosso método necessitou não excedeu dois minutos. O método exato completou os cálculos em 5 minutos em média para as instâncias de SET1 e SET3, mas precisou de quase 30 minutos para três delas. A Figura 1 mostra as lacunas percentuais entre APERTO soluções e soluções ótimas (OPTAR) calculado como: *gap =* (APERTO - OPT) / OPT [%].

Observamos que GRASP facilmente encontra soluções ótimas se | *F* | é pequeno quando comparado a | *C* | e as conexões entre *F* e *C* são esparsos (Figura 1 (a)). Com densidade crescente *pe (C),* o desempenho piora, mas os gaps médios ainda estão dentro de 2% do ideal.

Para casos cujo número de instalações é grande quando comparado ao número de clientes (Figura 1 (c)), observamos que GRASP tem di fi culdades de lidar, fornecendo soluções cujas lacunas médias estão entre 1% e 6% de ótimo.

Dada a mesma topologia de gráfico, também testamos a influência da estrutura de custos no desempenho GRASP, comparando 50 × 50 instâncias do SET1 e do SET3 (Figuras 1 (c), 1 (d)). Obviamente, não há dependência direta entre a qualidade das soluções obtidas e os parâmetros *pe (F)* e *pe (C)* do grupo SET1. No entanto, quando os custos médios de abertura de instalação são em uma ordem de magnitude mais elevados do que os custos médios de conexão (SET3), GRASP resolve gráficos com estrutura bipartida completa entre *F* e *C (pe (C) =* 1) muito e fi ciente (gap mediano inferior a 1%), e teve dificuldades com estruturas esparsas (*pe (C)* ∈ 0,18, 0,55).

SET2: Para 27 instâncias de tamanho 100 × 100, depois de rodar o B&C por uma hora, apenas uma instância foi resolvida de forma otimizada, e para três delas nem mesmo uma solução viável foi encontrada. Pelo contrário, o APERTO todos os resultados foram obtidos em menos de 5 minutos e em menos de 2 minutos na média.

Grupo		LIBRA- *Gap = Vão*		UB- *Gap = Vão*		APERTO B & C	*Gap = Vão*
pe (F)	*pe (C)*	*média*	*med*	*média*		*t [s]*	*média*
0,1	0,18	8,6	8,8	-	-	41,5	-
0,1	0,55	5,5	5,8	3,2	1,9	123,9	9,1
0,1	1.0	5,7	5,5	- 2,2	- 2.0	69,7	3,4
0,5	0,18	6,1	6,3	- 0,6	- 0,5	106,0	5,5
0,5	0,55	5,8	6,1	- 0,4	0,2	27,9	5,4
0,5	1.0	5,8	5,9	0,0	- 0,2	185,5	5,8
1.0	0,18	4,7	4,4	- 1,7	- 1,7	275,1	3,0
1.0	0,55	4,9	4,7	- 1,1	- 0,7	46,8	3,8
1.0	1.0	3,4	3,2	- 2.0	- 2.0	15,6	1,4

Tabela 1. GRASP vs. resultados de ramificação e corte para instâncias SET2 (100 × 100).

Na Tabela 4, para três instâncias dentro de um grupo e para 10 execuções por instância, relatamos os seguintes valores: LB-*gap = (* APERTO - LB) / LB, as lacunas média e mediana entre a solução GRASP e o limite inferior de B & C (LIBRA); UB-*gap = (* UB - GRASP) / UB, as lacunas média e mediana entre a solução GRASP e o limite superior de B & C (UB); tempo médio de execução GRASP em segundos (*t [s])* e a lacuna de otimalidade do B & C obtida como (UB - LB) / LB.

Os exemplos mais difíceis para P&C são os gráficos esparsos, onde GRASP supera até mesmo os limites superiores encontrados por P&C (valores UB-gap positivos) após uma hora.

Concluímos que, em três conjuntos de instâncias geradas aleatoriamente com topologia uniforme e diferentes estruturas de custo, GRASP tem um desempenho rápido e fornece resultados estáveis cujo gap médio para ótimo varia entre 0% e 10%. Para ambas as abordagens, B&C e GRASP, as instâncias mais difíceis (mais fáceis) parecem ser aquelas com topologias de instalação do cliente esparsas (densas) quando | *F* | = | *C* |.

Referências

[1] MP de Aragão, CC Ribeiro, E. Uchoa e R. Werneck. Pesquisa local híbrida para o problema de Steiner em gráficos. Dentro *Extended Abstracts of MIC 2001,* páginas 429-433, Porto, Portugal, 2001.

[2] F. Eisenbrand, F. Grandoni, T. Rothvoß e G. Schä¨fer. Aproximando problemas de localização de instalações conectadas por meio de amostragem aleatória de instalações e desvio de núcleo. Dentro *Proceedings of SODA,* páginas 1174–1183, 2008.

[3] T. Feo e M. Resende. Busca adaptativa aleatória gananciosa procedimentos. *Jornal de otimização global,* 6 (2): 109-133, 1995.

[4] DR Karger e M. Minkoff. Construindo árvores Steiner com conhecimento global completo. Dentro *FOCS,* páginas 613-623, 2000. [5] I. Ljubić´. Um VNS híbrido para localização de instalação conectada. Dentro *Hybrid Metaheuristics 2007,* páginas 157–169, 2007.

[6] I. Ljubić´, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, e M. Fischetti. Uma estrutura algorítmica para a solução exata do problema da árvore de Steiner com coleta de prêmios. *Programação Matemática, Série B,* 105 (2-3): 427–449, 2006.

[7] H. Takahashi e A. Matsuyama. Uma solução aproximada para o problema de Steiner em gráficos. *Matemática. Japonica,* 6: 573-577, 1990.