Um algoritmo GRASP para o Problema da Árvore de Steiner com Coleta de Prêmios

Matheus Simões[1], Vinícius S. Ribeiro[1], Vitor Mazal Krauss[1],

[1]Programa de Engenharia de Sistemas e Computação
Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa em Engenharia
Universidade Federal do Rio de Janeiro

1. Introdução

Nesta seção, começamos introduzindo o problema de otimização combinatória conhecido como *problema da árvore de Steiner com coleta de prêmios* ou, em inglês, prize-collecting Steiner Tree problem (PCSTP). O PCSTP é descrito da seguinte maneira: é dado um grafo $G = (V, E)$, um *custo* $c_e \geq 0$ para cada aresta $e \in E$ e um *prêmio* $p_v \geq 0$ para cada vértice $v \in V$. Os vértices $v \in V$ tais que $p_v > 0$ são denominados *terminais*, e o conjunto $T = \{v \in V : p_v > 0\}$ denota o conjunto dos vértices terminais. Uma solução viável para o PCSTP é uma árvore $S = (V_S, E_S)$, onde $V_S \subseteq V$ e $E_S \subseteq E$. Ou seja, uma solução viável para o PCSTP é um subgrafo S de G tal que S é uma árvore. O custo de uma solução $S = (V_S, E_S)$ é dado por

$$c(S) = \sum_{e \in E_S} c_e + \sum_{v \in V \setminus V_E} p_v \tag{1}$$

Em palavras, o custo de uma solução $S = (V_S, E_S)$ é a soma dos custos das arestas em S mais a soma dos prêmios dos vértices de G não gerados por S. O problema da árvore de Steiner com coleta de prêmios consiste em encontrar a solução $S^* = (V_{S^*}, E_{S^*})$ de menor custo.

O PCSTP está relacionado ao conhecido *problema da árvore de Steiner em grafos* (STP). O STP é descrito da seguinte maneira: é dado um grafo $G = (V, E)$, um *custo* $c_e \geq 0$ para cada aresta $e \in E$ e um conjunto $T \subseteq V$ de *terminais*. Uma solução viável para o STP é uma árvore $S = (V_S, E_S)$, onde $T \subseteq V_S \subseteq V$ e $E_S \subseteq E$. Ou seja, uma solução viável para o STP é um subgrafo S de G tal que S é uma árvore e S gera todos os vértices terminais $v \in T$. O custo de uma solução $S = (V_S, E_S)$ do STP é $\sum_{e \in E_S} c_e$. O problema da árvore de Steiner consiste em encontrar a solução de menor custo.

O STP é um problema bem estudado na literatura. De fato, a versão de decisão do STP é um dos 21 problemas $\mathcal{NP}$-completos propostos por Karp em [1], enquanto a versão de otimização do STP—como apresentado acima—é provada ser $\mathcal{NP}$-hard. Diferentes métodos exatos e heurísticos foram propostos para o STP, por exemplo em [2, 3, 4, 5, 6].

Conceitualmente, a diferença entre o STP e o PCSTP é que no STP todos os vértices terminais devem necessariamente ser gerados pela solução, enquanto no PCSTP um vértice terminal $v \in T$ não precisa necessariamente ser gerado, mas neste caso é arcado o valor do seu prêmio p_v. Com efeito, dado uma instância do STP com $G = (V, E)$, $\{c_e \geq 0 : e \in E\}$ e $T \subseteq V$, esta instância pode ser resolvida pela resolução da instância do PCSTP definida da seguinte maneira: consideramos $G = (V, E)$ e $\{c_e : e \in E\}$. Além disso, seja $P > 0$ suficientemente grande—por exemplo $P > \sum_{e \in E} c_e$. Para

cada terminal $v \in T$, seja $p_v = P$. Para $v \notin T$, seja $p_v = 0$. Então a solução ótima desta instância do PCSTP corresponde a uma solução ótima da instância original do STP. Sendo assim, o PCSTP é, no sentido matemático, tão difícil quanto o STP e, portanto, o PCSTP é $\mathcal{NP}$-hard.

Dada uma instância do PCSTP, seja $X \subseteq V$ um subconjunto de vértices e $E_X = \{(u, v) \in E : u, v \in X\}$ o conjunto de arestas entre vértices de X. Seja MST(X) a arvore geradora mínima de (X, E_X). Note que $c(\text{MST}(X)) \leq c(S(x))$ para qualquer outra solução $S(X) = (X, E')$, onde $E' \subseteq E_X$. Ou seja, para cada subconjunto de vértices X, a árvore geradora mínima de (X, E_X) é a solução de menor custo dentre todas as solução que geram X. Sendo assim, a complexidade combinatória do PCSTP está, efetivamente, na seleção dos vértices da solução. E note que há $\Omega(2^{|T|})$ subconjuntos contendo pelo menos um terminal.

Levando-se em conta a dificuldade do PCSTP, naturalmenete se torna relevante o estudo e desenvolvimento de métodos computacionais para sua resolução. Exemplos de métodos exatos e heurísticos para o PCSTP são apresentados em [7, 8, 9, 10].

Além do desafio matemático e algorítmico, a importância da classe dos problemas de Steiner—o que inclui o STP e o PCSTP—vem também da relevância prática destes problemas. Em [11] é apresentada uma revisão de problemas de Steiner e aplicações na área de telecomunicações, enquanto [12] é um livro sobre aplicação de problemas de Steiner na indústria. O PCSTP, por exemplo, encontra aplicação no design de redes de fibra ótica [7]: neste caso, o grafo de entrada representa a topologia de ruas de uma cidade. Cada aresta representa uma rua e o custo de uma aresta representa o custo de instalação de um segmento de fibra ótica nesta rua. Os vértices do grafo representam as interseções entre ruas e localizações de clientes. O prêmio associado a cada vértice é uma estimativa da receita a ser obtida por servir o cliente naquele vértice. O objetivo é determinar uma rede que minimize o custo de instalação e a perda de potencial receita. Este problema é naturalmente modelado como uma instância do PCSTP.

Neste trabalho, apresentamos um algoritmo GRASP para o PCSTP. Na Seção 2, revisamos as etapas de um procedimento GRASP em um contexto geral [13] e apresentamos um procedimento GRASP para o PCSTP [9]. Na Seção 3, propomos uma heurística construtiva, utilizada para obtenção de soluções iniciais. Na Seção 4 propomos um método de busca local, baseada na inserção e remoção de vértices na solução inicial. Na Seção 5 propomos um procedimento de *path relinking* [9]. Na Seção 6 apresentamos os resultados computacionais obtidos em instâncias do PCSTP usuais da literatura. Na Seção 7 apresentamos as conclusões do trabalho e linhas de trabalho futuro.

2. GRASP

O termo GRASP refere-se a um procedimento para solução heurística de problemas de otimização combinatória (COP) originalmente proposto por Feo e Resende em [13]. O GRASP é um método heurístico—e não garante certificado de otimalidade—e é geralmente utilizado na solução do COP cuja resolução por métodos exatos demanda tempos intratáveis de execução. Desde sua apresentação em 1995, o GRASP foi aplicado a diferentes COP e, devido ao recorrente sucesso, se estabeleceu como poderosa ferramenta para resolução de problemas de otimzação combinatória em geral [5, 9, 13].

Agora descreveremos, de maneira geral, um procedimento GRASP para um pro-

blema de minimização. O GRASP começa inicializando uma solução S^* como vazia, e um custo $c^* = \infty$. O propósito de S^* é armazenar a solução de menor custo encontrada, e c^* o respectivo custo desta solução. O GRASP consiste de um número pré-determinado de iterações, e cada iteração consiste das seguidas duas etapas: na primeira etapa é construída uma solução viável S para o COP em questão. Esta construção é feita de maneira heurística e, em geral, preza-se por um procedimento heurístico rápido. A segunda etapa consiste de um procedimento de busca-local aplicado a solução inicial S. O propósito de uma busca-local é procurar soluções possivelmente melhores na vizinhança de uma dada solução. A vizinhança de uma solução varia para cada problema e possivelmente diferentes vizinhanças podem ser definidas para o mesmo problema. A busca-local encontra uma solução S' tal que $c(S') \leq c(S)$. Se $c(S') < c^*$, então é feito $S^* \leftarrow S'$ e $c^* \leftarrow c(S')$. Ao final do GRASP, S^* é a solução de menor custo encontrada, com custo c^*.

O ponto chave do sucesso do GRASP é a construção de soluções inicias distintas em cada iteração. A ideia é que gerando diferentes soluções iniciais, o procedimento GRASP explora uma região mais ampla do espaço de soluções. Sendo assim, a variabilidade das soluções iniciais é crucial para o sucesso do GRASP. A variabilidade das soluções iniciais é obtida incorporando-se algum fator aleatório a heurística construtiva da primeira etapa. Uma maneira de fazer isso é considerar uma função de avaliação f, que estima a qualidade $f(A)$ de cada escolha A dentro da heurística construtiva. Uma heurística construtiva *gulosa*, em cada iteração, faria a escolha de melhor qualidade. Uma heurística construtiva gulosa aleatória, em cada iteração, determinaria uma *lista de candidatos restritos* (RCL) das escolhas com estimativa no intervalo $[\max_A f(A) - \alpha(\max_A f(A) - \min_A f(A)), \max_A f(A)]$ e aleatoriamente selecionaria uma escolha da RCL. Esta é a maneira originalmente apresentada em [13]. No caso deste trabalho, introduziremos variabilidade na construção inicial por meio de perturbações nos dados de entrada do problema, mais especificamente, incluiremos perturbações nos valores dos prêmios.

Em [7, 9] é proposto um procedimento conhecido como *path relinking*. Assim como a busca local, o path relinking é um procedimento que busca explorar o espaço de soluções a partir de soluções iniciais já conhecidas. O path relinking pode ser incluído como uma etapa a ser executada em cada iteração do GRASP, ou como uma única etapa após o término do GRASP. Neste trabalho, optamos por incluir o path relinking como uma única etapa após o término do GRASP.

Além disso, neste trabalho, em alternativa a especificar um número máximo de iterações para o GRASP, optamos por especificar um número de iterações sem melhoria.

O procedimento GRASP para o PCSTP proposto é apresentado no Algoritmo 1.

Algoritmo 1 Procedimento GRASP para o PCSTP.

$S^* \leftarrow \emptyset$
$c(S^*) \leftarrow \infty$
$i \leftarrow 0$
while i $\leq$ Max Unimproving Iterations **do**
    $S \leftarrow$ ConstructionHeuristic
    $S' \leftarrow$ LocalSearch(S)
    if $c(S') < c(S^*)$ **then**
        $S^* \leftarrow S'$
        $c(S^*) \leftarrow c(S')$
        $i \leftarrow 0$
    else
        $i \leftarrow i + 1$
    end if
end while
Path Relinking

3. Heurística Construtiva

Considere uma instância do PCSTP com dados de entrada $G = (V, E)$, custos $\{c_e : e \in E\}$, conjunto de terminais T e prêmios $\{p_v : v \in V\}$. Para cada par de vértices $u, v \in V$, seja p_{uv} o caminho mínimo u e v e seja $\ell(p_{uv}) = \sum_{e \in p_{uv}} c_e$ o tamanho do caminho p_{uv}. Definimos o *grafo de terminais* $G_T = (T, E_T)$ relacionado a esta instância da seguinte maneira: para cada par de terminais $t_1, t_2 \in T$, $(t_1, t_2) \in E_T$ com peso $\ell(p_{t_1 t_2})$. Portanto G_T é um grafo completo, cujos vértices representam os terminais, e cada aresta representa o caminho mínimo entre cada par de terminais. A heurística construtiva proposta neste trabalho é baseada em [14] e consiste das seguintes duas etapas: na primeira etapa, é determinada a árvore geradora mínima S_0 do grafo de terminais G_T. Na segunda etapa, uma solução viável para o PCSTP é construída da seguinte maneira. Para cada aresta $(t_1, t_2) \in S_0$, (t_1, t_2) representa o caminho mínimo $p_{t_1 t_2}$ entre t_1 e t_2. Então, para cada cada aresta $(t_1, t_2) \in S_0$, incluímos em S os vértices e arestas em $p_{t_1 t_2}$. Com isso, obtemos um subgrafo S do grafo original G. Para certificar que S é de fato uma árvore e, portanto, uma solução viável para o PCSTP, propomos duas possibilidades: a primeira, de menor complexidade, é armazenar, em cada iteração, os componentes conexos de S por meio da estrutura de dados *union find* [15],também conhecida como *disjoint set*. Neste caso, uma aresta só é incluída em S se ela liga dois vértices em componentes conexos distintos. A segunda possibilidade é computar a arvore geradora mínima S' de S. Assim, S' é solução viável para o PCSTP.

Na Figura 1 apresentamos uma ilustração do grafo de uma instância do PCSTP. Na Figura 2 apresentamos uma ilustração do grafo de terminais para a instância ilustrada na Figura 1. Note que os pesos das arestas denotam o tamanho do caminho mínimo entre terminais no grafo original. Na Figura 3 apresentamos uma ilustração da árvore geradora mínima do grafo de terminais. Na Figura 4 apresentamos uma ilustração da solução inicial encontrada pela heurística construtiva. Na figura, os vértices e arestas em vermelho são aqueles que fazem parte da solução.

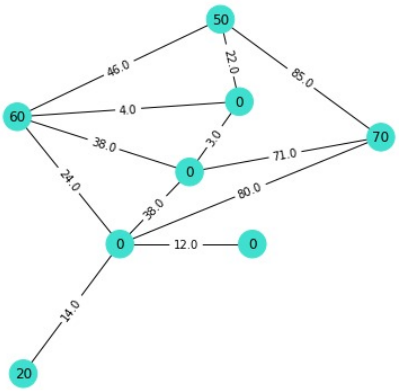

Figura 1. Grafo de entrada de uma instância do PCSTP.

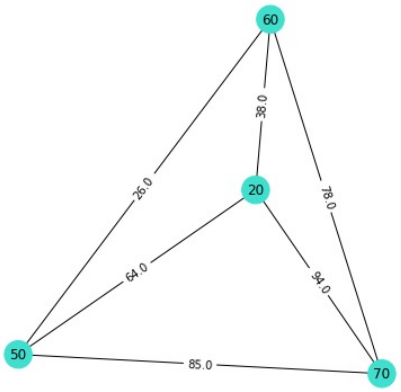

Figura 2. Grafo de terminais para instância do PCSTP.

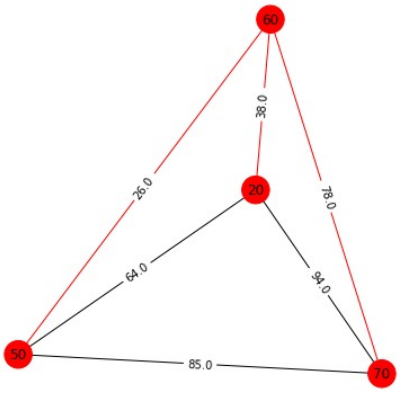

Figura 3. Árvore geradora mínima do grafo de terminais.

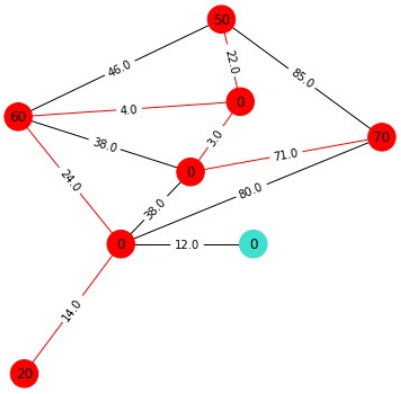

Figura 4. Solução inicial pela heurística construtiva para instância do PCSTP.

3.1. Poda

O procedimento de *poda* é uma etapa que sucede a heurística construtiva e tem como objetivo melhorar a solução inicial por meio da remoção de folhas da árvore cujo prêmio é menor que o custo da aresta incidente a folha. Seja $S = (V_S, E_S)$ uma solução viável para o PCSTP. O algoritmo de poda é apresentado no Algoritmo 2.

Algoritmo 2 Procedimento de poda.

Require: $S = (V_S, E_S)$
  $P \leftarrow$ True
  while P **do**
    $P \leftarrow$ False
    Seja $u \in V$ com grau 1 e vizinho v
    if $p_u < c_{uv}$ **then**
      $S \leftarrow S \setminus \{u\}$
      $P \leftarrow$ True
    end if
  end while

Nas Figuras 5 e 6 apresentamos a ilustração do procedimento de poda aplicado a solução da Figura 4. Note que neste caso, o procedimento de poda primeiro remove um vértice folha. A remoção deste vértice leva a uma nova folha, que é então removida no segundo passo da poda.

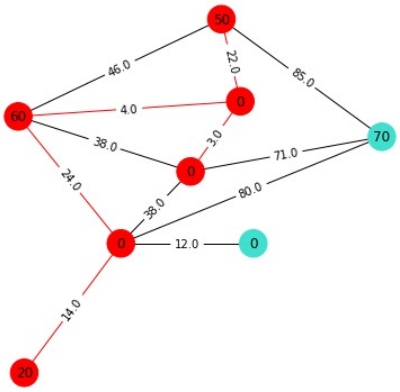

Figura 5. Solução após poda de um vértice.

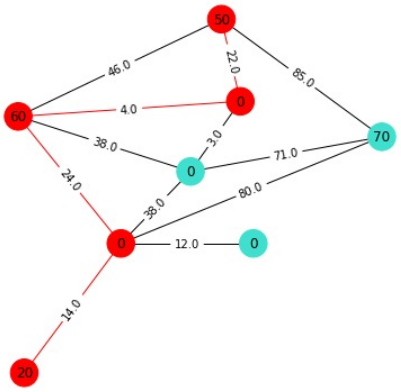

Figura 6. Solução após poda de dois vértices.

3.2. Perturbações

Como mencionado na Seção 2, a geração de soluções iniciais distintas em cada iteração do GRASP é ponto chave para o sucesso do método, pois a geração de soluções distintas leva a uma busca mais ampla no espaço de soluções. Neste trabalho, para fazer com que a heurística construtiva leve a soluções distintas, em cada iteração do GRASP, fazemos os seguintes dois procedimentos:

- Pertubações nos prêmios dos terminais: para cada terminal $t \in T$, perturbamos o prêmio p_t multiplicando p_v por um fator escolhido aleatoriamente no intervalo $[1-\alpha, 1+\alpha]$. Esta perturbação é acumulativa, no seguinte sentido: seja p_t^k o prêmio do terminal t na k-ésima iteração. Vale que $p_t^0 \leftarrow p_t$ e, para $k > 0$, $p_t^{k-1} \leftarrow p_t^k \cdot \beta_t^k$, onde β_t^k tem distribuição uniforme em $[1-\alpha, 1+\alpha]$ para todo $k > 0$.

- Inclusão de novos terminais: seja $u \in V \setminus T$ um vértice que não é originalmente terminal. Em cada iteração ímpar do GRASP, u é feito um vértice terminal naquela iteração com probabilidade $p \in (0, 1)$. Neste caso, fazemos $p_u \leftarrow \max\{p_v : v \in \delta(u)\}$.

4. Busca Local

A *busca local* é um procedimento comum e bem estabelecido na resolução de problemas de otimização combinatória [16]. O objetivo da busca local é procurar por soluções melhores a partir de uma dada solução inicial. Em geral, suponha que S é uma solução viável para um determinado problema de otimização combinatória. Cada iteração da busca local explora uma vizinhança $N(S)$ de S à procura de uma solução melhor. Se uma solução S' melhor que S é encontrada, a busca local então passa a explorar $N(S')$. O procedimento termina quando é encontrado uma solução S^* que é ótimo local, ou seja, S^* é tal que não existe $S' \in N(S^*)$ e S' é melhor solução que S^*. A definição da vizinhança $N(S)$ é geralmente feita para cada problema e pode mesmo variar entre diferentes métodos para o mesmo problema. Neste trabalho, assim como é comum na literatura, definimos a vizinhança $N(S)$ de uma solução viável S do PCSTP por meio de operadores que operam sobre S. Seja $S = (V_S, E_S)$ uma solução viável para o PCSTP. A vizinhança $N(S)$ é conjunto de soluções S' que podem ser obtidas pelos seguintes dois operadores:

- Inserção: seja $u \in V \setminus V_S$ e seja $v \in V_S$. Seja p_{uv} o caminho mínimo entre u e v. $S' = S \cup p_{uv}$ é a solução obtida incluindo-se os vértices e arestas de p_{uv} a S.
- Remoção: seja $u \in V_S$. Seja $\Delta(u) = \{v \in V_S : (u, v) \in E_S\}$. S' é obtida da seguinte maneira: remova de S o vértice u e as arestas incidentes a u, ou seja, remova as arestas em $\Delta(u)$. Enquanto S' for desconexo, selecione $w, x \in V_{S'}$ e inclua os vértices e arestas do caminho p_{wx} a S'.

Neste trabalho, consideramos uma estratégia de primeira melhoria, i.e, a cada iteração da bsuca local, a vizinhança é $N(S)$ explorada até ser encontrada, pela primeira vez, uma solução melhor S'. Daí, o procedimento é reiniciado na vizinhança $N(S')$, e termina quando for encontrada uma solução que é ótimo local. Nos testes computacionais realizados para instâncias do PCSTP, esta estratégia levou a resultados melhores do que a estratégia de melhor melhoria, na qual toda a vizinhança é explorada a procura da melhor solução na vizinhança. Além disso, outras duas estratégias que se demonstraram melhores no testes e foram então adotadas foram: primeiro, na inserção, a escolha de um vértice $v \in V_S$ aleatório para conexão com dado $u \in V \setminus V_S$. Esta estratégia se provou melhor do que a estratégia de escolher o vértice $v \in V_S$ com menor caminho até u. Segundo, na remoção, para reconectar a solução S', escolhendo, a cada iteração, aleatoriamente dois vértices u e v em componentes conexos diferentes e os conectando pelo caminho mínimo. Esta estratégia se provou melhor do que a estratégia de selecionar os componentes que minimizam a distânicia entre componentes e conectá-los pelo caminho mínimo.

No Algoritmo 3 apresentamos o operador de inserção para o PCSTP. No Algoritmo 4 apresentamos o operador de remoção para o PCSTP. No Algoritmo 5 apresentamos o procedimento de busca local para o PCSTP.

Algoritmo 3 Operador de inserção da busca local.

Require: Solution $S = (V_S, E_S)$
  $A \leftarrow V \setminus V_S$
  while $A \neq \emptyset$ **do**
    $u \leftarrow$ Sample(A)
    $v \leftarrow$ Sample(V_S)
    p_{uv} shortest uv path in G
    $S' \leftarrow S \cup p_{uv}$
    if $c(S') < c(S)$ **then**
      return True, S'
    end if
    $A \leftarrow A \setminus \{u\}$
  end while
  return False, S

Algoritmo 4 Operador de remoção da busca local.

Require: Solution $S = (V_S, E_S)$
  $A \leftarrow V_S$
  while $A \neq \emptyset$ **do**
    $u \leftarrow$ Sample(A)
    $S' \leftarrow S \setminus (u, \Delta(u))$
    while S' unconnected **do**
      $V_1, V_2, \ldots, V_k$ connected components of S'
      Sample $u \in V_i$ and $v \in V_j$
      p_{uv} shortest uv path in G
      $S' \leftarrow S' \cup p_{uv}$
    end while
    if $c(S') < c(S)$ **then**
      return True, S'
    end if
    $A \leftarrow A \setminus \{u\}$
  end while
  return False, S

Algoritmo 5 Busca local para o PCSTP.

Require: Solution $S = (V_S, E_S)$
  $S' \leftarrow S$
  $S^* \leftarrow S$
  $c^* \leftarrow c(S)$
  $P \leftarrow$ True
  while P **do**
    $P, S' \leftarrow$ Insertion(S')
    $P, S' \leftarrow$ Removal(S')
    if $c(S') < c^*$ **then**
      $S^* \leftarrow S'$
      $c^* \leftarrow c(S')$
    end if
  end while
  return S^*

Na Figura 7 apresentamos a ilustração antes de uma solução viável do PCSTP da inserção de um vértice. Na Figura 8 apresentamos a ilustração da solução após inserção de um vértice de prêmio 40. Note que a inserção inclui todo o caminho mínimo até o vértice.

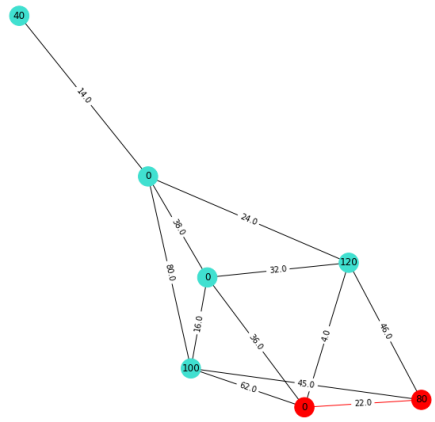

Figura 7. Solução antes da inserção.

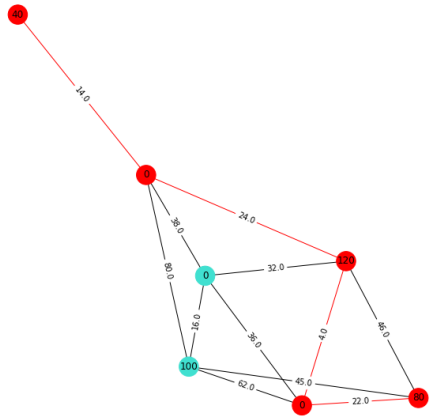

Figura 8. Solução após inserção de um vértice.

Na Figura 9 apresentamos a ilustração de uma solução viável antes da remoção de um vértice. Na Figura 10 apresentamos a ilustração da solução após remoção de um vértice e as arestas incidentes a ele. Na Figura 11 apresentamos a ilustração da solução após a conexão de duas componentes conexas. Na Figura 12 apresentamos a ilustração da solução após a conexão de outro componente conexo. Note que após as conexões a solução fica novamente conexo e é portanto uma solução viável.

5. Path Relinking

O procedimento de *path relinking* foi proposto originalmente em [9] e aplicado ao PCSTP em [7]. A ideia do path relinking é de ampliar a busca no espaço de soluções. A cada

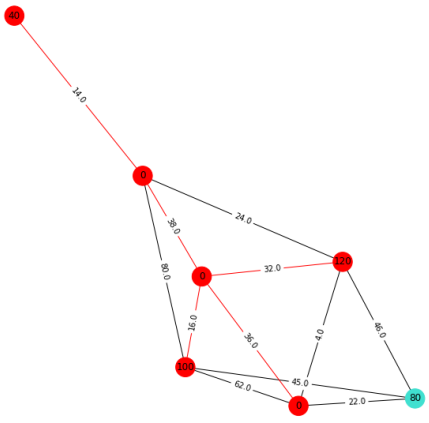

Figura 9. Solução antes da inserção.

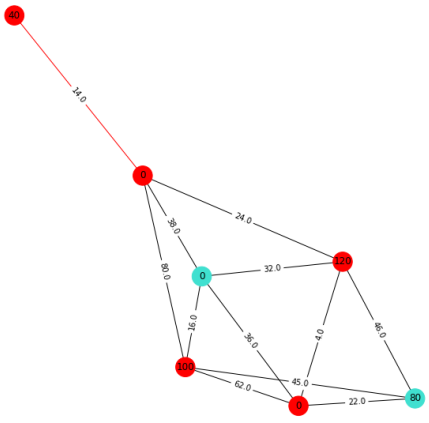

Figura 10. Solução após inserção de um vértice.

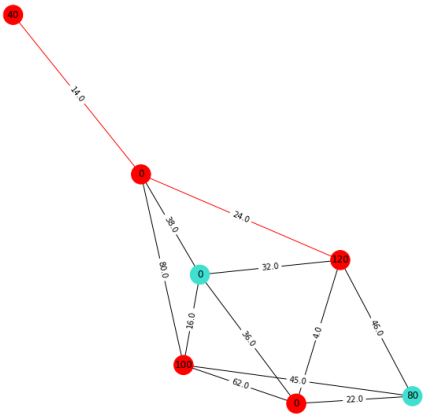

Figura 11. Solução antes da inserção.

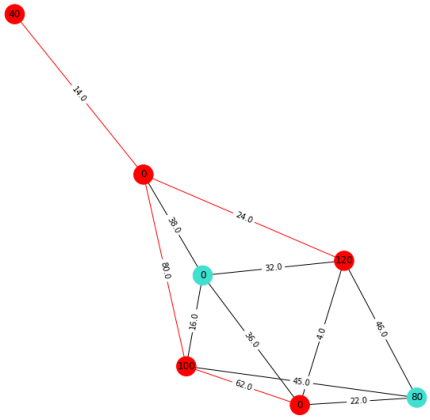

Figura 12. Solução após inserção de um vértice.

iteração do GRASP a suloção corrente é inserida em uma conjunto de soluções elite se satisfazer uma das condições:

- Uma solução equivalente está no conjunto mas com custo maior (substitui)
- A solução tem custo melhor que todas as já existentes no conjunto
- A solução é suficientemente diferente das demais e o conjunto não está cheio
- A solução tem custo menor que a pior solução no conjunto e a solução é diferente das demais (a pior solução é removida)

Sejam $S_1, S_2, \ldots, S_k$ soluções viáveis para o PCSTP. Para S_i e S_j duas soluções viáveis do PCSTP, seja $\Delta^+(S_i, S_j) = V_{S_j} \setminus V_{S_i}$ e $\Delta^-(S_i, S_j) = V_{S_i} \setminus V_{S_j}$. O procedimento de path relinking segue de seguinte forma: para cada par S_i, S_j de soluções elite, definimos S_i como *origem* e S_j como *destino*. Inicializamos uma solução corrente $S \leftarrow S_i$. Além disso, é mantida a melhor solução $S*$ no caminho de S_i até S_j. O seguinte laço é executado até que todas as operações determinadas pelo calculo de $\Delta^+(S_i, S_j)$ e $\Delta^-(S_i, S_j)$ sejam realizadas. Para cada $u \in \Delta^+(S, S_j)$, analisamos a solução S' obtida pela inserção de u a S e comparamos com a atual melhor solução para a iteração. E para

cada $u \in \Delta^{-}(S, S_j)$, analisamos a solução S' obtida pela remoção de u a S e comparamos com a atual melhor solução para a iteração. Após avaliar todas as possibilidades de adição e remoção de vértice, a solução de menor custo é colocada como solução corrente S e verificamos se ela é melhor que a solução $S*$. Considerando todos os pares ordenados $(S_i, S_j) \in \{S_1, S_2, \ldots, S_k\}^2$, realizamos o forward (origem na solução de pior custo) e backward (origem na solução de melhor custo) path-relinking. Ao término do procedimento, a melhor solução $S*$ conhecida é retornada.

Sendo assim, o path relinking descreve mais um procedimento de exploração do espaço de soluções do problema. Neste trabalho, o método é utilizado como pós-otimização para melhorar a solução obtida pelo GRASP. Para cada par (S_i, S_j) de soluções elite, o path relinking "cruza" o caminho no espaço de soluções partindo da solução S_i até a solução S_j. As soluções neste caminho são comparadas com a melhor solução conhecida e, se necessário, a melhor solução conhecida é atualizada.

Como é apresentado na Seção 6 e no Apêndice, o procedimento de path relinking teve sucesso em encontrar soluções melhores em diversas instâncias.

6. Resultados

Na Tabela 1 apresentamos, para cada classe de instâncias, o número acumulativo de instâncias com gap de otimalidade abaixo de um dado valor percentual. Na Tabela 1, cada linha corresponde a uma classe de instâncias. A coluna Classe denota a classe de instâncias, a coluna Nº Instancias denota o número de instâncias consideradas daquela classe, a coluna Ótimas denota o número de instâncias para quais o GRASP encontrou o valor ótimo, a coluna $< 1\%$ denota o número de instâncias para quais o GRASP encontrou solução com gap de otimalidade menor que 1%, e assim por diante. Note que as colunas são acumulativas. A coluna Tempo(s) denota o tempo, em segundos, de execução do GRASP.

Para as classes C e D, os gaps de otimalidade são feitos em comparação com os valores ótimos exatos para cada instância, estes valores estão disponíveis em [10]. Para a classe H, os gaps de otimalidade são feitos em comparação com ao melhor limitante superior apresentado em [17]. Para as classes H2, i640 e PUCNU, os gaps de otimalidade são feitos, para cada instância, em comparação com o melhor valor apresentado em [8], no qual é apresentado uma variante do algoritmo de Goemans-Williamson para o PCSTP. Note que o algoritmo proposto em [8] é heurístico e suas soluções não tem certificado de otimalidade.

Na Tabela 2 apresentamos os resultados para algumas instâncias da classe D com números diferentes de terminais.

Demais tabelas com informações detalhadas para cada instância de cada classe são apresentados no Apêndice ao final do texto.

Classe	Nº Instancias	Ótimas	$< 1\%$	$< 2\%$	$< 5\%$	Tempo(s)
C	8	3	4	7	8	181.53
D	6	3	5	5	6	1068.67
H	4	0	0	3	4	1731.35
H2	4	4	4	4	4	1492.94
i640	15	15	15	15	15	167.30
PUCNU	11	11	11	11	11	629.93

Tabela 1. Número cumulativo de instâncias abaixo de um dado gap de otimalidade.

Instância	Nº Terminais	GAP_GRASP	GAP_PR	Tempo(s)
D01-A.stp	5	0.00	0.00	36.5
D06-A.stp	5	0.00	0.00	38.6
D03-A.stp	167	0.35	0.00	520.5
D08-A.stp	167	1.56	0.13	972.3
D13-A.stp	167	4.67	0.90	2363.8
D18-A.stp	167	12.11	4.13	2480.5

Tabela 2. Alguns resultados para instâncias da classe D. Todas as instancias têm 1000 vértices e 1250 arestas.

7. Conclusão

Dos resultados apresentados na Seção 6 e no Apêndice, concluímos que o procedimento GRASP proposto levou a resultados satisfatórios em todas as classes de instâncias e para instâncias com variados números de vértices e terminais. De fato, para as classes C e D—aquelas em que o valor ótimo exato é conhecido—o maior gap de otimalidade foi de 4.12% após o path relinking. Em particular, para as instâncias C06, D01 e D06, as soluções ótimas consistem de um grafo com único vértice e nenhuma aresta. Por conta disso, fizemos um ajuste no procedimento de poda para levar em conta estes possíveis casos: se o procedimento de poda leva a uma árvore vazia—que é o caso para estas instâncias—fazemos com que o procedimento de poda retorne uma árvore com único um único vértice que é o terminal de maior prêmio. Este ajuste ao procedimento de poda fez com que o GRASP encontrasse o valor ótimo para estas três instancias. Antes do ajuste, o GRASP obtinha como melhor solução o grafo vazio, cujo custo é a soma de todos os prêmios e não é ótimo.

Além disso, consideramos que, em particular, o procedimento de path relinking proposto também foi satisfatório. Como é apresentado no Apêndice, o procedimento de path relinking em todas as instâncias teve sucesso em encontrar soluções melhores. Note que para as instâncias C06, D01 e D06 o path reliking não foi acionado, pois o GRASP não levou a um número suficiente de soluções distintas—aqui determinado como 10—para dar início ao path relinking. Na instância C18-A, por exemplo, o gap de otimalidade antes do path relinking foi de 12.61%, e de 1.8% após o path relinking. Na instância D18-A, por exemplo, o gap de otimalidade antes do path relinking foi de 12.11%, e de 4.12% após o path relinking. Esta foi a instância com maior gap de otimalidade após o path relinking.

Para as classes H2, i640 e PUCNU, para as quais os valores ótimos exatos não são conhecidos, os resultados apresentados neste trabalho podem proporcionar melhores limitantes superiores em comparação aos já conhecidos. Como apresentado no Apêndice, para todas as instâncias destas três classes, o GRASP aqui proposto levou a soluções melhores do que as soluções encontradas em [8]. Por outro lado, a confirmação das soluções aqui apresentadas como as melhores conhecidas necessita de uma revisão exaustiva da literatura para comparação dos resultados.

A complexidade e o tempo de execução do GRASP cresce em função do número de vértices e do número de terminais. De fato, note da Tabela 2, que para instâncias com mesmo número de vértices e arestas, o tempo de execução do GRASP cresce com o número de terminais. Além disso, note que o gap de otimalidade também cresce com o número de terminais.

Sendo assim, apesar dos bons resultados com relação a qualidade das soluções nas instâncias testadas, uma linha de trabalho futuro é buscar reduzir o tempo de execução do método proposto. Por exemplo, para instância hc9p2.stp, que das instâncias consideradas neste trabalho é aquela com maior número de terminais, o tempo de execução foi de 5324 segundos. É claro que a resolução das maiores instâncias por meio de métodos exatos também exige uma grande quantidade de tempo, sendo impraticável em algumas instâncias. Em [10] por exemplo, para muitas das instâncias da classe H são apenas apresentados limitantes superiores para o valor ótimo. De qualquer forma, uma linha de trabalho futuro seria estudar maneiras de acelerar o procedimento GRASP proposto aqui. Neste caso, há naturalmente um *trade off* entre a qualidade das soluções e o tempo de execução.

Além disso, no que diz respeito à qualidade das soluções, apontamos duas possibilidades de trabalho futuro que vemos como promissoras na obtenção de soluções com menor gap de otimalidade: primeiro, a inclusão do path-relinking como etapa em cada iteração do GRASP. Segundo, a implementação de variable neighborhood search. Ambos têm o intuito de tornar mais ampla a busca pelo espaço de soluções e possivelmente levar a soluções melhores. Como contraponto, a implementação destas duas estratégias provavelmente levaria a tempo de execução ainda maiores.

Referências

[1] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*, pp. 85–103, Springer, 1972.

[2] M. Chlebík and J. Chlebíková, "The steiner tree problem on graphs: Inapproximability results," *Theoretical Computer Science*, vol. 406, no. 3, pp. 207–214, 2008.

[3] N. Maculan, "The steiner problem in graphs," *North-Holland Mathematics Studies*, vol. 132, pp. 185–211, 1987.

[4] A. Lucena and J. E. Beasley, "A branch and cut algorithm for the steiner problem in graphs," *Networks: An International Journal*, vol. 31, no. 1, pp. 39–59, 1998.

[5] S. L. Martins, P. M. Pardalos, M. G. Resende, C. C. Ribeiro, *et al.*, "Greedy randomized adaptive search procedures for the steiner problem in graphs.," in *Randomization methods in algorithm design*, pp. 133–145, Citeseer, 1997.

[6] C. C. Ribeiro and M. C. De Souza, "Tabu search for the steiner problem in graphs," *Networks: An International Journal*, vol. 36, no. 2, pp. 138–146, 2000.

[7] S. A. Canuto, M. G. Resende, and C. C. Ribeiro, "Local search with perturbations for the prize-collecting steiner tree problem in graphs," *Networks: An International Journal*, vol. 38, no. 1, pp. 50–58, 2001.

[8] C. Hegde, P. Indyk, and L. Schmidt, "A fast, adaptive variant of the goemans-williamson scheme for the prize-collecting steiner tree problem," in *Workshop of the 11th DIMACS Implementation Challenge*, vol. 2, p. 3, Workshop of the 11th DIMACS Implementation Challenge, 2014.

[9] M. G. Resende and C. C. Ribeiro, "Grasp with path-relinking: Recent advances and applications," *Metaheuristics: progress as real problem solvers*, pp. 29–63, 2005.

[10] A. S. da Cunha, A. Lucena, N. Maculan, and M. G. Resende, "A relax-and-cut algorithm for the prize-collecting steiner problem in graphs," *Discrete Applied Mathematics*, vol. 157, no. 6, pp. 1198–1217, 2009.

[11] S. Voß, "Steiner tree problems in telecommunications," in *Handbook of optimization in telecommunications*, pp. 459–492, Springer, 2006.

[12] X. Cheng and D.-Z. Du, *Steiner trees in industry*, vol. 11. Springer Science & Business Media, 2013.

[13] T. A. Feo and M. G. Resende, "Greedy randomized adaptive search procedures," *Journal of global optimization*, vol. 6, no. 2, pp. 109–133, 1995.

[14] K. Mehlhorn, "A faster approximation algorithm for the steiner problem in graphs," *Information Processing Letters*, vol. 27, no. 3, pp. 125–128, 1988.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*. MIT press, 2009.

[16] E. Aarts, E. H. Aarts, and J. K. Lenstra, *Local search in combinatorial optimization*. Princeton University Press, 2003.

[17] M. Leitner, I. Ljubić, M. Luipersbeck, and M. Sinnl, "A dual ascent-based branch-and-bound framework for the prize-collecting steiner tree and related problems," *INFORMS journal on computing*, vol. 30, no. 2, pp. 402–420, 2018.

8. Apêndice

As tabelas abaixo apresentam de forma mais detalhada os resultados obtidos pelo procedimento GRASP proposto. Para realização dos testes, cada instância foi resolvida pelo GRASP cinco vezes. Assim, as tabelas abaixo apresentam dados como o melhor custo e a média dos custos para cada instância, além dos tempos de execução e gap de otimalidade. O gap para instâncias das classes C e D foram calculados comparando os resultados em [7], enquanto para as demais classes foram utilizados os custos das soluções em [8].

Classe	Instância	$\|V\|$	$\|E\|$	$\|T\|$
C	C01-A.stp	500	625	5
C	C03-A.stp	500	625	83
C	C04-A.stp	500	625	125
C	C06-A.stp	500	1000	5
C	C08-A.stp	500	1000	83
C	C13-A.stp	500	2500	83
C	C18-A.stp	500	12500	83
C	C19-A.stp	500	12500	125
D	D01-A.stp	1000	1250	5
D	D03-A.stp	1000	1250	167
D	D06-A.stp	1000	2000	5
D	D08-A.stp	1000	2000	167
D	D13-A.stp	1000	5000	167
D	D18-A.stp	1000	25000	167
H	hc6p.stp	64	192	64
H	hc7p.stp	128	448	128
H	hc8p.stp	256	1024	256
H	hc9p.stp	512	2304	512
H2	hc6p2.stp	64	192	64
H2	hc7p2.stp	128	448	128
H2	hc8p2.stp	256	1024	256
H2	hc9p2.stp	512	2304	512
i640	i640-001.stp	640	960	9
i640	i640-014.stp	640	4135	9
i640	i640-023.stp	640	204480	9
i640	i640-034.stp	640	1280	9
i640	i640-045.stp	640	40896	9
i640	i640-102.stp	640	960	25
i640	i640-114.stp	640	4135	25
i640	i640-124.stp	640	204480	25
i640	i640-133.stp	640	1280	25
i640	i640-142.stp	640	40896	25
i640	i640-201.stp	640	960	50
i640	i640-212.stp	640	4135	50
i640	i640-223.stp	640	204480	50
i640	i640-233.stp	640	1280	50
i640	i640-243.stp	640	40896	50
PUCNU	bip42nu.stp	1200	3982	200
PUCNU	bip62nu.stp	1200	10002	200
PUCNU	bipe2nu.stp	550	5013	50
PUCNU	cc10-2nu.stp	1024	5120	135
PUCNU	cc3-10nu.stp	1000	13500	50
PUCNU	cc3-4nu.stp	64	288	8
PUCNU	cc3-5nu.stp	125	750	13
PUCNU	cc5-3nu.stp	243	1215	27
PUCNU	cc6-2nu.stp	64	192	12
PUCNU	cc6-3nu.stp	729	4368	76
PUCNU	cc9-2nu.stp	512	2304	64

Instância	HC	LS_mean	LS_best	GRASP_mean
C01-A.stp	62.0	18.0	18.0	18.0
C03-A.stp	649.0	450.8	444.0	425.0
C04-A.stp	808.0	639.4	635.0	631.0
C06-A.stp	52.0	18.0	18.0	18.0
C08-A.stp	449.0	375.0	370.0	364.4
C13-A.stp	266.0	244.4	243.0	243.2
C18-A.stp	128.0	125.2	123.0	125.0
C19-A.stp	158.0	156.0	156.0	155.6
D01-A.stp	44.0	18.0	18.0	18.0
D03-A.stp	1145.0	825.4	821.0	809.8
D06-A.stp	55.0	18.0	18.0	18.0
D08-A.stp	928.0	777.6	770.0	766.8
D13-A.stp	501.0	468.6	466.0	465.8
D18-A.stp	253.0	244.4	242.0	244.4
hc6p.stp	4874.0	4333.2	4210.0	4117.0
hc7p.stp	10011.0	8787.4	8523.0	8288.0
hc8p.stp	20802.0	17401.4	17247.0	16716.2
hc9p.stp	40760.0	34657.6	34013.0	32793.4
hc6p2.stp	4874.0	4346.2	4264.0	4136.0
hc7p2.stp	10112.0	8825.4	8527.0	8328.6
hc8p2.stp	20800.0	17703.6	17430.0	16627.8
hc9p2.stp	40658.0	35149.8	34561.0	33177.2
i640-001.stp	3569.0	3013.2	2932.0	2932.0
i640-014.stp	2667.0	2373.6	2268.0	2274.4
i640-023.stp	2307.0	2014.0	2010.0	1744.0
i640-034.stp	3573.0	2968.2	2766.0	2757.0
i640-045.stp	1932.0	1822.6	1748.0	1614.8
i640-102.stp	9497.0	7845.4	7800.0	7801.6
i640-114.stp	7738.0	6556.2	6294.0	6326.4
i640-124.stp	6516.0	6135.0	6122.0	5288.8
i640-133.stp	8758.0	7135.0	7028.0	6909.0
i640-142.stp	6450.0	6207.0	6094.0	5475.0
i640-201.stp	17520.0	14723.0	14517.0	14533.4
i640-212.stp	15755.0	12524.6	12035.0	11519.6
i640-223.stp	13472.0	12860.0	12850.0	11100.0
i640-233.stp	16980.0	14194.0	13669.0	13398.0
i640-243.stp	13042.0	12665.8	12557.0	10974.8
bip42nu.stp	294.0	275.6	274.0	265.0
bip62nu.stp	257.0	246.8	246.0	239.2
bipe2nu.stp	59.0	55.0	55.0	54.8
cc10-2nu.stp	216.0	194.8	192.0	194.2
cc3-10nu.stp	91.0	70.0	70.0	69.2
cc3-4nu.stp	13.0	12.0	12.0	10.8
cc3-5nu.stp	22.0	20.0	20.0	18.2
cc5-3nu.stp	51.0	44.8	42.0	39.6
cc6-2nu.stp	22.0	17.4	16.0	15.2
cc6-3nu.stp	120.0	107.6	106.0	107.6
cc9-2nu.stp	123.0	105.8	101.0	95.2

Instância	GRASP_best	PR_mean	Cost_best	Cost_mean
C01-A.stp	18.0	NaN	18.0	18.0
C03-A.stp	419.0	422.2	419.0	422.2
C04-A.stp	629.0	630.8	629.0	630.8
C06-A.stp	18.0	NaN	18.0	18.0
C08-A.stp	362.0	361.6	361.0	361.6
C13-A.stp	242.0	239.0	238.0	239.0
C18-A.stp	123.0	115.2	113.0	115.2
C19-A.stp	154.0	150.2	149.0	150.2
D01-A.stp	18.0	NaN	18.0	18.0
D03-A.stp	807.0	807.0	807.0	807.0
D06-A.stp	18.0	NaN	18.0	18.0
D08-A.stp	763.0	758.8	756.0	758.8
D13-A.stp	463.0	450.2	449.0	450.2
D18-A.stp	242.0	228.2	227.0	228.2
hc6p.stp	4074.0	3982.4	3968.0	3982.4
hc7p.stp	8162.0	7903.8	7875.0	7903.8
hc8p.stp	16593.0	15748.4	15689.0	15748.4
hc9p.stp	32516.0	31263.8	31034.0	31263.8
hc6p2.stp	4100.0	3978.2	3946.0	3978.2
hc7p2.stp	8161.0	7867.8	7819.0	7867.8
hc8p2.stp	16443.0	15805.4	15674.0	15805.4
hc9p2.stp	33114.0	31594.6	31479.0	31594.6
i640-001.stp	2932.0	2932.0	2932.0	2932.0
i640-014.stp	2242.0	2215.2	2171.0	2215.2
i640-023.stp	1716.0	1640.8	1627.0	1640.8
i640-034.stp	2757.0	2757.0	2757.0	2757.0
i640-045.stp	1569.0	1576.2	1569.0	1576.2
i640-102.stp	7791.0	7792.8	7791.0	7792.8
i640-114.stp	6178.0	5929.8	5875.0	5929.8
i640-124.stp	5184.0	4763.0	4631.0	4763.0
i640-133.stp	6867.0	6851.6	6814.0	6851.6
i640-142.stp	5350.0	4926.8	4848.0	4926.8
i640-201.stp	14505.0	14404.0	14381.0	14404.0
i640-212.stp	11280.0	10827.2	10710.0	10827.2
i640-223.stp	10739.0	9645.2	9546.0	9645.2
i640-233.stp	13240.0	13198.4	13009.0	13198.4
i640-243.stp	10821.0	9579.4	9526.0	9579.4
bip42nu.stp	262.0	250.2	245.0	250.2
bip62nu.stp	235.0	229.6	226.0	229.6
bipe2nu.stp	54.0	54.2	54.0	54.2
cc10-2nu.stp	192.0	180.0	178.0	180.0
cc3-10nu.stp	68.0	64.2	63.0	64.2
cc3-4nu.stp	10.0	10.4	10.0	10.4
cc3-5nu.stp	18.0	17.6	17.0	17.6
cc5-3nu.stp	39.0	38.6	38.0	38.6
cc6-2nu.stp	15.0	15.2	15.0	15.2
cc6-3nu.stp	106.0	100.0	99.0	100.0
cc9-2nu.stp	93.0	90.6	90.0	90.6

Instância	Optimal	Time	GAP_GRASP	GAP_PR
C01-A.stp	18	9.6196	0.000000	0.000000
C03-A.stp	414	176.6976	2.657005	1.207729
C04-A.stp	618	216.2774	2.103560	1.779935
C06-A.stp	18	8.9268	0.000000	0.000000
C08-A.stp	361	174.5736	0.941828	0.000000
C13-A.stp	236	276.7054	3.050847	0.847458
C18-A.stp	111	307.8318	12.612613	1.801802
C19-A.stp	146	281.5990	6.575342	2.054795
D01-A.stp	18	36.4524	0.000000	0.000000
D03-A.stp	807	520.4748	0.346964	0.000000
D06-A.stp	18	38.5768	0.000000	0.000000
D08-A.stp	755	972.2610	1.562914	0.132450
D13-A.stp	445	2363.7848	4.674157	0.898876
D18-A.stp	218	2480.4688	12.110092	4.128440
hc6p.stp	3908	19.4470	5.348004	1.535312
hc7p.stp	7721	83.7120	7.343608	1.994560
hc8p.stp	15422	711.4460	8.391908	1.731293
hc9p.stp	30318	6110.7980	8.164787	2.361633
hc6p2.stp	4468	12.1414	0.000000	0.000000
hc7p2.stp	9163	72.4612	0.000000	0.000000
hc8p2.stp	18915	562.7220	0.000000	0.000000
hc9p2.stp	37379	5324.4350	0.000000	0.000000
i640-001.stp	3414	33.1968	0.000000	0.000000
i640-014.stp	2667	70.4866	0.000000	0.000000
i640-023.stp	2072	41.9648	0.000000	0.000000
i640-034.stp	3188	61.9300	0.000000	0.000000
i640-045.stp	1932	70.8644	0.000000	0.000000
i640-102.stp	8575	124.5988	0.000000	0.000000
i640-114.stp	7271	128.1970	0.000000	0.000000
i640-124.stp	6317	57.6012	0.000000	0.000000
i640-133.stp	8253	160.0352	0.000000	0.000000
i640-142.stp	6239	87.5006	0.000000	0.000000
i640-201.stp	15953	289.7386	0.000000	0.000000
i640-212.stp	14377	529.8878	0.000000	0.000000
i640-223.stp	13065	123.1754	0.000000	0.000000
i640-233.stp	15172	472.9152	0.000000	0.000000
i640-243.stp	12843	257.3400	0.000000	0.000000
bip42nu.stp	280	3247.5382	0.000000	0.000000
bip62nu.stp	254	1189.0948	0.000000	0.000000
bipe2nu.stp	57	44.7682	0.000000	0.000000
cc10-2nu.stp	188	1697.8998	3.297872	0.000000
cc3-10nu.stp	65	151.2462	6.461538	0.000000
cc3-4nu.stp	11	1.6178	0.000000	0.000000
cc3-5nu.stp	19	4.4976	0.000000	0.000000
cc5-3nu.stp	38	24.4684	4.210526	0.000000
cc6-2nu.stp	15	2.1410	1.333333	0.000000
cc6-3nu.stp	104	394.0728	3.461538	0.000000
cc9-2nu.stp	90	171.8398	5.777778	0.000000