EXAME DE QUALIFICAÇÃO
MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

Caminhos Mínimos com Recursos Limitados

Joel Silva Uchoa
Orientador: Carlos Eduardo Ferreira

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO

1 de julho de 2012

1 Introdução

Um problema bastante conhecido é o de escolher uma rota para se fazer uma viagem, tal que a rota minimize a distância do percurso. Nesta forma básica, esse problema é o problema de caminho mínimo em grafos onde as arestas são possíveis trechos, valorados por seu comprimento. Algumas vezes um caminho mínimo desta forma é bom, outras vezes não. Existem ocasiões, onde tal caminho possui propriedades indesejáveis. Por exemplo, alguns trechos podem ter tráfego denso e nos fazer perder muito tempo na travessia, ou existem muitos pedágios com taxas que, acumuladas pelo caminho, vão exceder o dinheiro que temos disponível. Isso nos leva a considerar um ou mais parâmetros adicionais para a escolha do caminho. Os casos mais comuns de parâmetros a considerar envolvem o consumo de recursos em um orçamento que limita a quantidade disponível desses recursos. Um caminho mínimo com essas limitações adicionais é chamado de **caminho mínimo com recursos limitados** (*resource constrained shortest path - RCSP*) [2].

Problema RCSP(G, s, t, k, r, l, c): *Como parâmetros do problema são dados:*

- *um grafo dirigido* $G = (V, A)$,
- *um vértice origem* $s \in V$ *e um vértice destino* $t \in V$, $s \neq t$,
- *um número* $k \in \mathbb{N}$ *de recursos disponíveis* $\{1, \dots, k\}$,
- *o consumo de recursos* $r_a^i \in \mathbb{N}_0$ *de cada arco de* G *sobre os* k *recursos disponíveis,* $i = 1, \dots, k$, $a \in A$,
- *o limite* $l^i \in \mathbb{N}_0$ *que dispomos de cada recurso,* $i = 1, \dots, k$,
- *o custo* $c_a \in \mathbb{N}_0$, *para cada arco,* $a \in A$.

O consumo de um recurso i, $i = 1, \dots, k$ *em um st-caminho* P *é* $r^i(P) = \sum_{a \in P} r_a^i$. *Um st-caminho* P *é limitado pelos recursos* $1, \dots, k$ *se este consome não mais que o limite disponível de cada recurso, ou seja, se* $r^i(P) \leq l^i$, $i = 1, \dots, k$. *O custo de um st-caminho* P *é* $c(P) = \sum_{a \in P} c_a$. *O problema* RCSP *consiste em encontrar o caminho limitado pelos recursos de menor custo.*

Usaremos no decorrer deste trabalho $n = |V|$ e $m = |A|$. Quando estivermos tratando de um contexto onde existe apenas um recurso, ou seja, $k = 1$, usaremos apenas l para representar l^1 e apenas r_a para representar r_a^1.

2 Histórico do Problema

Os primeiros a apresentarem resultados sobre o problema foram Witzgall e Goldman 1965 [14]. Joksch 1966 [9], trabalhando de forma independente, apresentou um resultado similar ao resultado de Witzgall e Goldman 1965 [14] (veja também Lawler 1976 [10]). Todos esses autores apresentaram algoritmos baseados em programação dinâmica com complexidade pseudopolinomial, com recorrências similares. A partir de então, o problema tem recebido grande atenção de diversos pesquisadores.

Temos vários algoritmos exatos propostos ao longo do tempo para o RCSP. Dentre eles, cronologicamente, podemos citar Joskch 1966 [9], Lawler 1976 [10], Handler e Zang 1980 [6], Aneja, Aggarwal e Nair 1983 [1], Henig 1985 [8], Beasley e Christofides 1989 [2], Hassin 1992 [7]. Os algoritmos descritos em [6] e [2] usam uma relaxação lagrangeana da formulação mais usual do RCSP por programação linear. Vale ressaltar ainda que [2] usa o método de subgradiente para resolver, aproximadamente, a relaxação lagrangeana. Temos ainda Mehlhorn e Ziegelmann 2000 [11], que apresenta a abordagem do envoltório (*hull approach*), um algoritmo combinatório para resolver uma relaxação de uma outra representação do RCSP como um problema de programação linear.

Além dos algoritmos exatos, também surgiram algoritmos de aproximação. Warburton 1987 [13] desenvolveu um FPTAS (*fully polynomial-time approximation scheme*) para o problema. Hassin 1992 [7], além de apresentar um algoritmo pseudo-polinomial, apresentou melhoras ao resultado de Warburton e também propôs um outro FPTAS para o problema. Temos ainda mais um FPTAS proposto por Phillips 1993 [12].

3 Complexidade

O problema RCSP é NP-difícil mesmo para o caso em que existe apenas um único recurso ([6] [4]). Nós podemos reduzir um problema NP-difícil bem conhecido a ele, o **problema da mochila** (*knapsack*), definido a seguir.

Problema Mochila(N,w,v,d): *Como parâmetros do problema são dados:*

- *Um conjunto de itens* $N = \{1, \ldots, n\}$,
- *pesos* $w_i \in \mathbb{N}$, $i = 1, \ldots, n$, *para esses itens,*
- *valores* $v_i \in \mathbb{N}$, $i = 1, \ldots, n$, *para esse itens,*
- *um peso limite* $d \in \mathbb{N}_0$.

O peso de um subconjunto $I \subseteq N$ *é* $w(I) = \sum_{i \in I} w_i$, *e seu valor é* $v(I) = \sum_{i \in I} v_i$. *O problema* Mochila *consiste em encontrar um subconjunto de itens com valor máximo, cujo peso não excede o limite* d.

Teorema 1: *O* RCSP *é NP-difícil.*

Demonstração: A prova se dá pela redução do problema Mochila ao RCSP. Vamos tomar uma instância I do problema Mochila. Nós podemos construir uma instância I' para o RCSP como se segue:

- $V := N \cup \{0\}$.
- $A := A_1 \cup A_2$.
  - $A_1 := \{(i-1, i) : i = 1, \ldots, n\}$,
  - $A_2 := \{(i-1, i) : i = 1, \ldots, n\}$.
- $s := 0$, $t := n$.
- $k = 1$.

- $l := d$.
- $r_a := \begin{cases} w_i, & \text{se } a \in A_1, \\ 0, & \text{caso contrário} \end{cases}$ para todo $a \in A$.
- $c_a := \begin{cases} M - v_i, & \text{se } a \in A_1, \\ M, & \text{caso contrário} \end{cases}$ para todo $a \in A$.

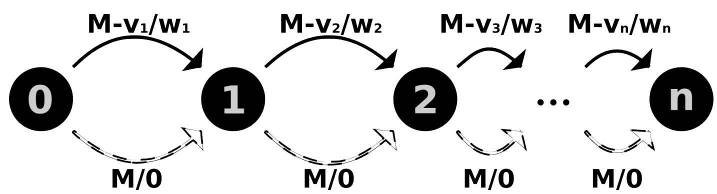

Figura 1: *Os arcos preenchidos são os arcos de A_1 e os tracejados de A_2. O rótulo de cada arco a representa c_a/r_a.*

A constante M pode ser definida como um grande inteiro de tal forma que $M - v_i$, para qualquer i, seja não negativo. Por questão de praticidade, vamos convencionar que representaremos um arco $(i-1, i) \in A_1$ como a_i^1 e um arco $(i-1, i) \in A_2$ como a_i^2.

Como $s = 0$, $t = n$; podemos ver que qualquer st-caminho P em $G = (V, A)$ contém ou a_i^1 ou a_i^2, $i = 1, \ldots, n$. Vamos dividir os arcos de P em dois conjuntos X e Y, onde X contém os arcos em P que estão em A_1, e Y contém os demais arcos. A partir de X, vamos definir um subconjunto $S \subseteq N$, tal que $i \in S$ se e somente se $a_i^1 \in X$. Com isso,

$$\begin{aligned} c(P) &= \sum_{a_i^1 \in X} (M - v_i) + \sum_{a_i^2 \in Y} M \\ &= n \cdot M - \sum_{a_i^1 \in X} v_i \\ &= n \cdot M - v(S) \end{aligned} \tag{1}$$

$$\begin{aligned} r(P) &= \sum_{a_i^1 \in X} w_i + \sum_{a_i^2 \in Y} 0 \\ &= \sum_{a_i^1 \in X} w_i \\ &= w(S) \end{aligned} \tag{2}$$

Daí, concluímos que todo subconjunto $S \subseteq N$ contém um st-caminho P associado, e vice-versa, por meio da equivalência $i \in S \Leftrightarrow a_i^1 \in P$. Pelas equações (1) e (2), um conjunto S e um caminho P associados, possuem $r(P) = w(S)$ e $c(P) = n \cdot M - v(S)$. E daí temos dois resultados:

$$\begin{aligned} r(P) \leq l \quad &\Longleftrightarrow \quad w(S) \leq d \\ \text{minimizar } c(P) \quad &\Longleftrightarrow \quad \text{maximizar } v(S) \end{aligned}$$

$\square$

4 Algoritmos Exatos

4.1 Programação Dinâmica

O RCSP é um problema que pode ser resolvido por algoritmos pseudopolinomiais [7]. Para os algoritmos a seguir, vamos assumir que $s = 1$ e $t = n$ Por praticidade, vamos descrever para versão com um único recurso, mas os algoritmos podem ser facilmente generalizados.

4.1.1 Algoritmo A

Primeiro vamos descrever a recorrência mais comumente citada na literatura [9], [10]. Definimos $f_j(r)$ como sendo o custo do caminho com menor custo de 1 a j, que consome no máximo r unidades de recurso, e assim temos a recorrência:

$$f_j(r) = \begin{cases} 0, & \text{se } j = 1 \\ & \text{ e } r = 0, \ldots, l \\ \\ \infty, & \text{se } j = 2, \ldots, n \\ & \text{ e } r = 0 \\ \\ \min\left\{ f_j(r-1), \min\limits_{k \mid r_{kj} \leq r} \{ f(r - r_{kj}) + c_{kj} \} \right\}, & \text{se } j = 2, \ldots, n \\ & \text{ e } r = 1, \ldots, l \end{cases}$$

Podemos implementar um algoritmo que computa o valor de um caminho ótimo $OPT = f_n(l)$ em tempo $O(nml)$. Joksch [9] apresentou melhoras práticas para este algoritmo, contudo a complexidade de pior caso é não melhor que a obtida com a ideia básica.

4.1.2 Algoritmo B

Podemos ainda fazer um algoritmo de programação dinâmica para resolver uma outra recorrência, que também resolve o problema em tempo pseudopolinomial [7].

Definimos $g_j(c)$ como sendo o custo do caminho que consome menos recurso de 1 a j, e tem custo máximo c. Assim, temos a recorrência:

$$g_j(c) = \begin{cases} 0, & \text{se } j = 1 \\ & \text{ e } c = 0, \ldots, OPT \\ \\ \infty, & \text{se } j = 2, \ldots, n \\ & \text{ e } c = 0 \\ \\ \min\left\{ g_j(c-1), \min\limits_{k \mid c_{kj} \leq c} \{ f(c - c_{kj}) + r_{kj} \} \right\}, & \text{se } j = 2, \ldots, n \\ & \text{ e } c = 1, \ldots, OPT \end{cases}$$

Observe que OPT não é um valor conhecido no inicio da execução, mas ele pode ser expresso como $OPT = \min\{c \mid g_n(c) \leq l\}$. Para contornar isso, devemos computar a função g iterativamente, primeiro para $c = 1$ e $j = 2, \ldots, n$, então para $c = 2$ e $j = 2, \ldots, n$, e assim sucessivamente, até o primeiro valor c' tal que $g_n(c') \leq l$. Só então teremos o conhecimento do valor $OPT = c'$. A complexidade do algoritmo sugerido acima é $O(nmOPT)$.

4.2 Relaxação Lagrangeana

A seguir vamos apresentar o algoritmo proposto por Handler e Zang [6], que se utiliza de uma relaxação de um problema de programação linear que modela o RCSP. A descrição que vamos fazer será para o problema com um único recurso, mas o método é perfeitamente aplicável ao problema com um número arbitrário de recursos.

Inicialmente, vamos apresentar uma formulação para o problema RCSP usando programação linear. Nela teremos uma variável x_{ij} para cada $(i,j) \in A$, $x_{ij} = 1$ significa dizer que o arco (i,j) está na solução e $x_{ij} = 0$ o contrário.

$$
\text{(P)} \qquad
\begin{array}{llcll}
\text{minimize} & c(x) & = & \displaystyle\sum_{(i,j)\in A} c_{ij}x_{ij} & \\
\text{sujeito a} & \displaystyle\sum_j x_{ij} - \sum_k x_{ki} & = & \begin{cases} 1, & \text{se } i = 1 \\ 0, & \text{se } i = 2, \ldots, n-1 \\ -1, & \text{se } i = n \end{cases} & (1) \\
& \displaystyle\sum_{(i,j)\in A} r_{ij}x_{ij} & \leq & l & (2) \\
& x_{ij} & \in & \{0,1\},\ (i,j) \in A & (3)
\end{array}
$$

Na formulação acima, a restrição (3) é responsável por delimitar os possíveis valores que um componente do vetor x pode assumir. A restrição (1), por sua vez, é responsável por garantir que para um vetor x ser solução viável do problema, ele deve "conter" um caminho do vértice 1 ao vértice n. Por fim, a restrição (2) nos garante que o conjunto de arcos induzido por um vetor x viável, não excede os recursos disponíveis.

Por conveniência de notação, nós iremos definir os seguintes termos. Vamos definir $\mathcal{X}$ denotando o conjunto de vetores x que satisfazem as equações (1) e (3), ou seja, vetores x que contêm um caminho de 1 a n. Vamos definir também a seguinte função.

$$g(x) = \sum_{(i,j)\in A} r_{ij}x_{ij} - l$$

Com as definições acima, resolver (P) é equivalente a resolver o seguinte.

$$c^* = c(x^*) = min\,\{\ c(x) \mid x \in \mathcal{X} \text{ e } g(x) \leq 0\ \}$$

Agora iremos aplicar a teoria da dualidade lagrangeana (como apresentada, por exemplo, em [5], [3]) como primeiro passo para resolver o RCSP. Tendo em vista que o problema é relativamente mais simples de resolver quando a restrição $g(x) \leq 0$ é relaxada (sem essa restrição, o problema se reduz a caminho mínimo simples), nossa estratégia será justamente retirar essa "restrição complicada" do conjunto de restrições e a usarmos como penalidade na função objetivo (técnica essa que é a essência da relaxação lagrangeana).

Para qualquer $u \in \mathbb{R}$, definimos a função lagrangeana.

$$L(u) = \min_{x \in \mathcal{X}} L(u, x), \text{ onde } L(u, x) = c(x) + ug(x)$$

Perceba que encontrar a solução de $L(u)$ é o problema de caminho mínimo no grafo original, porém com os custos dos arcos alterados para $c_{ij} + ur_{ij}$, $(i, j) \in A$. Temos que $L(u) \leq c^*$ para qualquer $u \geq 0$ (teorema fraco da dualidade), pois

$$g(x^*) \leq 0 \Rightarrow L(u) \leq c(x^*) + ug(x^*) \leq c(x^*) = c^*,$$

o que nos permite usar $L(u)$ como um limite inferior para o problema original. Para encontrarmos o limite inferior mais justo possível, resolvemos o problema dual a seguir.

$$\text{(D)} \qquad L^* = L(u^*) = \max_{u \geq 0} L(u)$$

Pode ser que exista uma folga na dualidade (*duality gap*), ou seja, pode ser que L^* seja estritamente menor que c^*. Nos casos que existir essa folga, teremos que trabalhar um pouco mais para eliminá-la.

Vamos, agora, descrever um método para resolver o programa (P), que usa como passo, resolver o problema (D). Por praticidade vamos denotar $x(u)$ como um caminho que possui valor ótimo associado à função $L(u)$.

O mais natural é que, como primeiro passo, verifiquemos se o menor caminho (não limitado, $\min\limits_{x \in \mathcal{X}} c(x)$) respeita nossas restrições. Vamos chamar esse caminho de $x(0)$, pois $L(0) = c^*$.

- Se $g(x(0)) \leq 0$, então $x(0)$ é claramente uma solução ótima de (P).
- Senão, $x(0)$ nos serve, pelo menos, como limite inferior para a solução.

Como segundo passo, devemos verificar se o caminho que consome menor quantidade de recursos ($\min\limits_{x \in \mathcal{X}} g(x)$) respeita nossas restrições. Vamos chamar esse caminho de $x(\infty)$, pois para valores muito grandes de u, o parâmetro $c(x)$ na função $L(u)$ é "dominado" por $ug(x)$.

- Se $g(x(\infty)) > 0$, o problema não tem solução, pois o caminho que consume a menor quantidade de recursos consome uma quantidade maior do que o limite.
- Senão, $x(\infty)$ é uma solução viável para a instância e nos serve de limite superior para problema.

Agora com os resultados dos passos anteriores, se não temos ainda a solução ou a prova de que a instância é inviável, temos a seguinte situação: Dois caminhos, $x(0)$, que **não é solução** e é um **limite inferior** e $x(\infty)$, que **é solução viável** e é um **limite superior**, $g(x(0)) > 0$ e $g(x(\infty)) \leq 0$.

Da forma como desenvolvemos a solução até então, podemos interpretar cada caminho no grafo como uma reta no espaço (u, L) da forma $L = c(x) + ug(x)$, onde u é nossa variável, $c(x)$ é nosso termo independente (ponto onde a reta corta o eixo L) e $g(x)$ é nosso coeficiente angular. Isso nos permite dar uma interpretação geométrica para a função $L(u)$, que será o envolope inferior do conjunto de retas (caminhos), ou seja, $L(u)$ será um conjunto de segmentos de retas, tal que cada ponto (u, L) nesses segmentos está abaixo ou na mesma altura de qualquer ponto (u, L') pertencente as retas associadas aos caminhos.

Com a interpretação geométrica dos caminhos, temos a informação que retas **crescentes** são associadas a caminhos **não viáveis** para o nosso problema, enquando as retas **não crescentes** são **soluções viáveis**. Como estamos procurando o valor de L^* (o ponto "mais alto" da função $L(u)$) vamos analisar o ponto (u', L') que é a intercessão das retas associadas a $x(0)$ e $x(\infty)$.

$$u' = (c(x(\infty)) - c(x(0)))/(g(x(0)) - g(x(\infty)))$$

$$L' = c(x(0)) + u' \cdot g(x(0))$$

É fato que $u' \geq 0$, pois $c(x(0))$ é mínimo, $g(x(\infty)) \leq 0$ e $g(x(0)) > 0$. Claramente, se existem apenas dois caminhos o ponto (u', L') é o que maximiliza $L(u)$. O mesmo acontece quando existem vários caminhos e $L(u') = L'$, ou seja, $L(u', x) \geq L'$ para qualquer $x \in \mathcal{X}$. Um último caso "especial" é quando existe um caminho $x_h \in \mathcal{X}$ tal que $g(x_h) = 0$ e $L(u') = L(u', x_h) < L'$. Como a reta associada a x_h é horizontal, ela limita superiomente $L(u)$, e como temos o ponto $(u', L(u'))$ sobre ela, $c^* = c(x_h) = L^* = L(u')$ (neste caso não existe folga na dualidade).

Falamos, especificamente, sobre os caminhos $x(0)$ e $x(\infty)$ no parágrafo anterior, mas o que foi dito vale no caso geral, onde temos dois caminhos disponíveis $x^+, x^- \in \mathcal{X}$, tal que $g^+ \equiv g(x^+) > 0$, $g^- \equiv g(x^-) \leq 0$ e $c^- \equiv c(x^-) \geq c^+ \equiv c(x^+)$. Então, temos que $u' = (c^- - c^+)/(g^+ - g^-)$ e $L' = c^+ + u'g^+$ definem o ponto de intercessão, no espaço (u, L), das retas associadas aos caminhos x^+ e x^-. Se $L(u') = L'$ ou se $g(x(u')) = 0$, então $L(u^*) = L(u')$ é a solução do nosso problema dual (D). Caso contrário, se $g(x(u')) < 0$, então $x(u')$ é o nosso novo caminho x^-, e se $g(x(u')) > 0$, então $x(u')$ é o nosso novo caminho x^+. O procedimento se repete até determinarmos a solução do problema (D). Com a realização do procedimento temos disponíveis um limite inferior LB (*lower bound*) e um limite superior UB (*upper bound*) para o valor de c^*. Nós temos que $LB = L(u^*) \leq c(x^*)$ (pelo teorema fraco da dualidade); e por definição segue que qualquer x^- usado durante o procedimento é uma solução viável, assim UB é o valor do último c^- ou o valor de $c(x(u'))$ associado com o último caminho $x(u')$ **se** $g(x(u')) \leq 0$.

Tendo resolvido o problema (D), temos limites $LB \leq c^* \leq UB$ e uma solução viável associada a UB para o RCSP. Quando $LB = UB$, esta solução é ótima. Porém, quando $LB < UB$ temos um folga na dualidade. Para eliminarmos essa folga poderiamos considerar usar um algoritmo de k-ésimo menor caminho (*k-shortest path*) a partir do primeiro caminho x tal que $c(x) \geq LB$ até o primeiro x_k tal que $g(x_k) \leq 0$. Como esse algoritmo precisa do conhecimento de todos os caminhos anteriores para gerar o próximo, essa abordagem não tomaria nenhum proveito da resolução do dual. Em contraste, determinar o k-ésimo menor caminho em relacão a função lagrangeana $L(u^*, x)$ (o que é equivalente a usar a função c' como custo, $c'_{ij} = c_{ij} + u^* \cdot r_{ij}$, $(i, j) \in A$) é perfeitamente aplicável a partir da solução dual.

Vamos denotar $L_k(u^*)$, para $k = 1, 2, \ldots$, como sendo o valor do k-ésimo menor caminho $x_k \in \mathcal{X}$ em relação a função de custo $L(u^*, x)$. Os caminhos x_1 e x_2 já são conhecidos, eles são x^+ e x^- respectivamente, pois se interceptam no ponto $(u^*, L(u^*))$, o que significa que possuem valor mínimo em relação a função $L(u^*, x)$. Iterando sobre o k-ésimo caminho, $k \geq 3$, nós atualizamos UB quando $g(x_k) \leq 0$ e $c(x_k) < UB$; e atualizamos $LB = L_k(u^*)$, pois essa é uma sequência não decrescente $(L_{k-1}(u^*) \leq L_k(u^*))$. O procedimento continua até que $LB \geq UB$, e então temos a solução do problema (P), associada a $UB = c^*$, solução do RCSP.

Algoritmo RCSP-LANGRANGEANA(G, $s = 1$, $t = n$, $k = 1$, r, l, c)

$\triangleright$ Inicialização
    $x_0, c_0, g_0 \leftarrow L(0)$
    se $g_0 \leq 0$
        então $x^*, c^* \leftarrow x_0, c_0$
        senão $x^+, c^+, g^+ \leftarrow x_0, c_0, g_0$
    $x_\infty, c_\infty, g_\infty \leftarrow L(\infty)$
    se $g_\infty > 0$
        então $x^*, c^* \leftarrow NULL, NULL$ $\triangleright$ Não tem solução!
        senão $x^-, c^-, g^- \leftarrow x_\infty, c_\infty, g_\infty$
$\triangleright$ Resolvendo o Dual
    se $x^+ \neq NIL$ e $x^- \neq NIL$ $\triangleright$ Se entrou nos dois "então" acima
        $LB \leftarrow 0$; $UB \leftarrow c^-$
        enquanto $LB < UB$ faça
            $u' \leftarrow (c^- - c^+)/(g^+ - g^-)$; $L' \leftarrow c^+ + u'g^+$; $x', c', g' \leftarrow L(u')$
            se $g' = 0$
                então $x^*, c^* \leftarrow x', c'$; $LB \leftarrow UB \leftarrow c'$
                        PÁRA o enquanto
            se $L(u') = L'$ e $g' < 0$
                então $LB \leftarrow L'$; $UB \leftarrow \min\{UB, c'\}$; $x^- \leftarrow x'$; $u^* \leftarrow u'$
                        PÁRA o enquanto
            se $L(u') = L'$ e $g' > 0$
                então $LB \leftarrow L'$; $u^* \leftarrow u'$
                        PÁRA o enquanto
            se $L(u') < L'$ e $g' > 0$
                então $x^+, c^+, g^+ \leftarrow x', c', g'$
            se $L(u') < L'$ e $g' < 0$
                então $x^-, c^-, g^- \leftarrow x', c', g'$; $UB \leftarrow \min\{UB, c'\}$
$\triangleright$ Eliminando a folga da dualidade
        $x_1, x_2 \leftarrow x^+, x^-$; $k \leftarrow 2$
        enquanto $LB < UB$ faça
            $k \leftarrow k + 1$; $x_k, c_k, g_k \leftarrow L_k(u^*)$; $LB \leftarrow L_k(u^*)$
            se $g_k \leq 0$ e $c_k < UB$
                então $x^-, UB \leftarrow x_k, c_k$
            se $LB \geq UB$
                então $x^*, c^* \leftarrow x^-, UB$

    devolva x^*, c^*

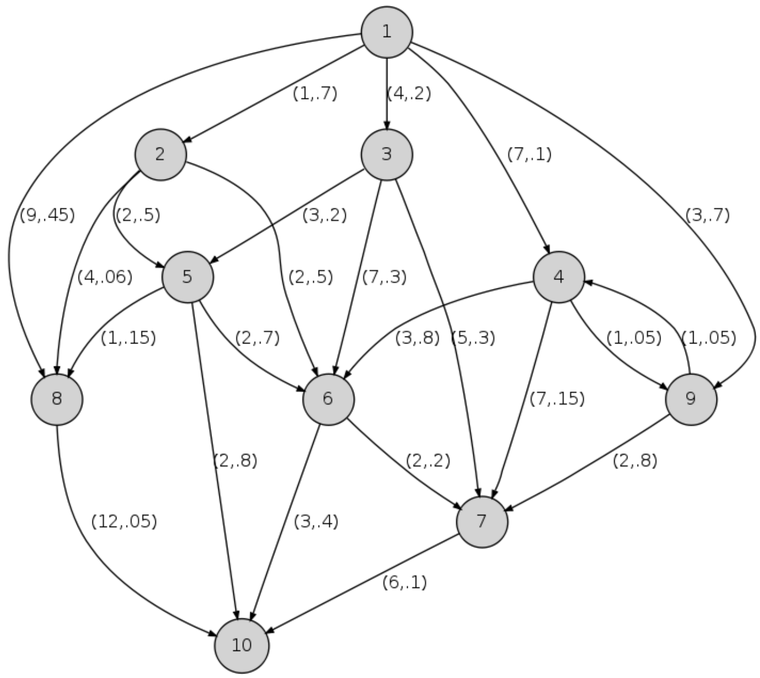

Figura 2: *Grafo exemplo; os rótulos dos arcos representam* (c_{ij}, r_{ij}).

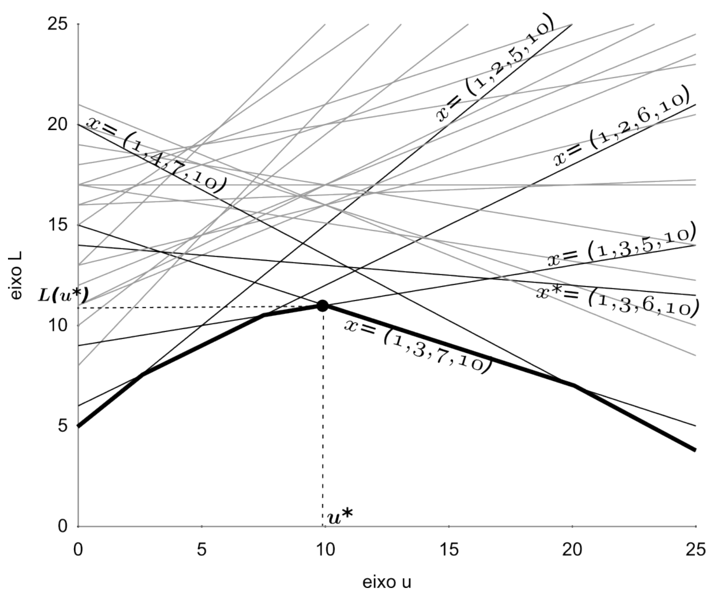

Figura 3: *Representação geométrica do grafo da Figura 2. As retas pretas represetam os caminhos que são relevantes ao algoritmo. A "curva" de segmentos mais espessos representa o função* $L(u)$.

5 Projeto de mestrado

Nesse projeto, pretendemos desenvolver um estudo detalhado sobre o problema RCSP. Exporemos os resultados mais relevantes, mantendo uma notação padronizada, tomando o cuidado de fazer provas detalhadas sobre corretude, complexidade de espaço e tempo de algoritmos. Por fim, vamos fazer implementações dos algoritmos e analisar seus comportamentos na prática.

5.1 Estrutura da dissertação

Inicialmente pretendemos que a dissertação siga uma estrutura semelhante a deste projeto, apresentando os seguintes capítulos.

- Introdução: Neste capítulo, descreveremos e definiremos de forma detalhada o problema. Apresentaremos também um histórico, falando sobre os primeiros trabalhos a abordar o problema e descrevendo de forma sucinta os principais resultados até os dias atuais. Além disso, apresentaremos exemplos e aplicações práticas.
- Preliminares: Aqui, trataremos de definir e apresentar algumas notações e terminologias que usaremos com frequência no decorrer do trabalho. Ainda neste capítulo, disponibilizaremos os conceitos básicos necessários para o entendimento de todo o conteúdo a ser abordado. Como exemplo de conceitos necessários podemos citar: Complexidade e Análise de Algoritmos, Grafos, Programa Linear, Algoritmos de Aproximação, etc.
- Complexidade: Discutiremos aqui, a complexidade computacional do problema RCSP, provando de forma cuidadosa e detalhada que ele se trata de um problema NP-difícil. Falaremos de casos especiais, onde são conhecidos algoritmos eficientes.
- Algoritmos exatos: Neste capítulo, exibiremos algoritmos propostos ao problema que encontram uma solução ótima para o mesmo. Basicamente algoritmos baseados em programação dinâmica, programação linear e também algoritmos combinatórios.
- Algoritmos de aproximação: Aqui, mostraremos algoritmos de aproximação propostos para o RCSP.
- Implementação e experimentos: Por fim, exibiremos informações importantes sobre as dificuldades de implementação dos algoritmos, além de apresentar com detalhes implementações eficientes dos principais deles. Também faremos experimentos e análises comparativas com essas implementações para determinar quais dos algoritmos são melhores, e em que condições tais algoritmos são aplicáveis de forma satisfatória.

5.2 Planejamento

Dentre as atividades que devem ser desempenhadas para o desenvolvimento desse projeto, as principais são as seguintes.

1. Leitura de material sobre o problema e sobre conceitos afins, necessários para o entendimento de todo o conteúdo.

2. Implementação dos principais algoritmos e testes de tais implementações.

3. Realização de testes e experimentos práticos com instâncias geradas para o problema e análise de seus resultados.

4. Escrita, revisão e correção do texto da tese.

5. Defesa da dissertação.

Tabela 1: *Distribuição do tempo por atividade*

Atividade	Set	Out	Nov	Dez	Jan	Fev	Mar
[1]	$\surd$	$\surd$					
[2]	$\surd$	$\surd$	$\surd$				
[3]		$\surd$	$\surd$	$\surd$			
[4]			$\surd$	$\surd$	$\surd$	$\surd$	
[5]							$\surd$

Referências

[1] Y. P. Aneja. Shortest chain subject to sided constraints. *Networks*, 13:295–302, 1983.

[2] J. E. Beasley and N. Christofides. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989.

[3] M. L. Fisher. Lagrangian relaxation methods for combinatorial optimization. *Research Paper, Department of Decision Sciences*, 1978.

[4] M. Garey and D. Johnson. *Computers and Intractability: A Guide of the Theory of NP Completeness*. W. H. Freeman, San Francisco, 1979.

[5] A. M. Geoffrion. Lagrangian relaxation for integer programming. *Math Program Study*, 2:82–114, 1974.

[6] G. Handler and I. Zang. A dual algorithm fot the constrained shortest path problem. *Networks*, 10:281–291, 1980.

[7] Refael Hassin. Approximation schemes for the restricted shortest path problem. *Mathmatics of Operations Research*, 17:36–42, 1992.

[8] M. Henig. The shortest path problem with two objective functions. *European J. Oper. Res.*, 25:295–302, 1985.

[9] H. C. Joksch. The shortest route problem with constraints. *J. Math. Anal. Appl.*, 14:191–197, 1966.

[10] E. L. Lawler. *Combinatotial Optimization: Networks and Matroids*. Holt, Rinehart and Winston, New York, 1976.

[11] Kurt Mehlhorn. Resource constrained shortest paths. *Lectures Notes in Computer Science*, 1879:326–337, 1985.

[12] C. Phillips. The network inhibition problem. *In 25th ACM STOC*, pages 776–785, 1993.

[13] A. Warburton. Approximation of pareto-optima in multiple-objective shortest path problems. *Operations Research*, 35:70–79, 1987.

[14] C. Witzgall and A. J. Goldman. Most profitable routing before maintenance. *Paper presented at the 27th National ORSA Meeting*, B-82, 1965.