EXAME DE QUALIFICAÇÃO

MESTRADO EM CIÊNCIA DA COMPUTAÇÃO

Caminhos Mínimos com Recursos Limitados

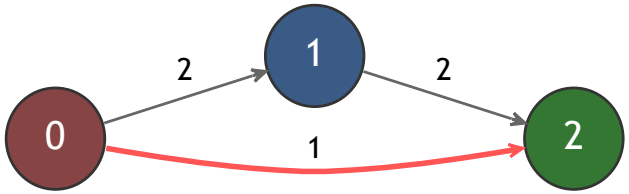
Joel Silva Uchoa
Orientador: Carlos Eduardo Ferreira

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

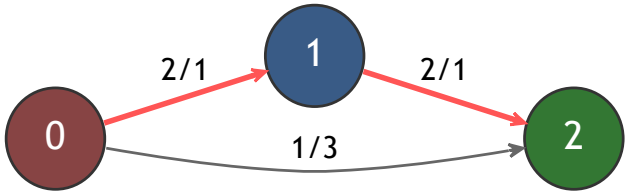
Sumário

Introdução

- Caminho Mínimo



- E se existir outro parâmetro a considerar?



- Caminho Mínimo com Recursos Limitados (*resource constrained shortest path* - **RCSP**) [2]

Definição Formal

Problema RCSP(G, s, t, k, r, l, c): *Dados*

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

Para um st-caminho P*:*

- *Seu consumo é* $r^i(P) = \sum_{a \in P} r_a^i$, $i = 1, \ldots, k$.
- *É dito limitado pelos recursos se* $r^i(P) \leq l^i$, $i = 1, \ldots, k$.
- *Seu custo é* $c(P) = \sum_{a \in P} c_a$.

O problema RCSP *consiste em encontrar um caminho limitado pelos recursos de menor custo.*

Variações

- *the vertex constrained shortest path problem (VCSP)* - encontrar o menor caminho que contem um subconjunto específico de vértices.
- *the time constrained shortest path problem (TCSP)* - encontrar o menor caminho quando os tamanho dos arcos são dependentes do tempo e/ou existem janelas de tempo para o uso dos vértices.

Histórico

- Witzgall e Gouldman 1965 [8], Joksch 1966 [5], e Lawler 1976 [6] - algoritmo baseado em programação dinâmica pseudopolinomial.
- Aneja e Nair 1978 - algoritmo baseado em uma abordagem paramétrica (algoritmo incorreto [1]).
- Handler e Zang 1980 - algoritmo baseado em uma relaxação lagrangiana de uma formulação do problema, resolvida com caminhos mínimos irestritos [4]. Resolveram instâncias com 500 vértices e 2500 arcos (apenas um único recurso).

Histórico

- Beasley e Christofides 1989 - algoritmo baseado em uma relaxação lagrangiana de uma formulação do problema, usada como limite inferior para procedimento de árvore de busca [2] (com duas inequações para cada recurso). Resolveram instâncias com 500 vértices, 5000 arcos e 10 recursos.
- Hassin 1992 - algoritmo de aproximação (FPTAS) $O(\log\log B(|A|(n/\epsilon) + \log\log B))$, baseado na técnica de "escalar e arredondar".
- Mehlhorn e Ziegelmann 2000 - algoritmo baseado em uma relaxação de uma formulação do problema por programação linear, técnica denominada de "abordagem do envoltório" [7].

NP-difícil

- O problema RCSP é *NP*-difícil mesmo para o caso em que existe apenas um único recurso ([4] [3]).
- A prova se dá pela redução do problema Mochila ao RCSP.

Problema Mochila(N,w,v,d): *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.

NP-difícil

Teorema 1: *O* RCSP *é NP-difícil.*

Demonstração: Vamos tomar uma instância $\mathcal{I}$ do problema Mochila. Nós podemos construir uma instância $\mathcal{I}'$ 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 $a \in A$.

NP-difícil

Demonstração: Continuação ...

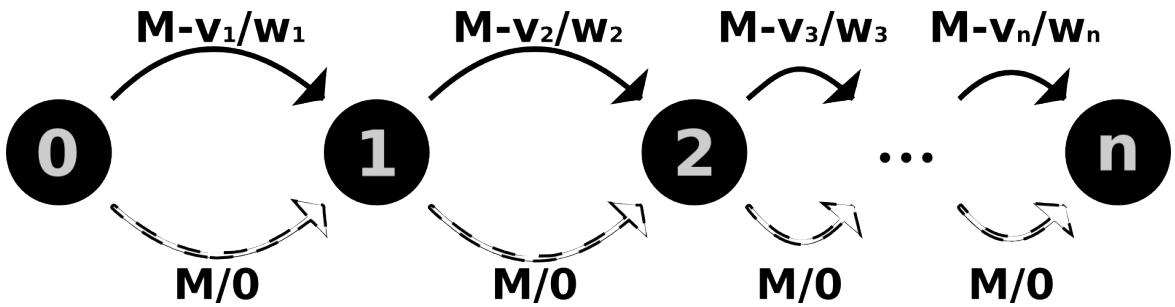


Figura: *Os arcos preenchidos são os arcos de A_1 e os tracejados de A_2. O rótulo de cada arco a representa c_a/r_a.*

- A constante M deve ser definida de tal forma que $M - v_i \geq 0$, $1 \leq i \leq n$.
- Por questão de praticidade vamos representar 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.

NP-difícil

Demonstração: Continuação ...

- 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 subconjuntos 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 $a_i^1 \in X$.

NP-difícil

Demonstração: Continuação ...

Com isso,

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

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

NP-difícil

Demonstração: Continuação ...

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:

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

□

Algoritmos Exatos

Por praticidade, para os algoritmos a seguir,

- vamos assumir que $s = 1$ e $t = n$;
- vamos descrever para a versão com um único recurso.

Mas os algoritmos podem ser facilmente generalizados.

Algoritmo A

Definimos $f_j(r)$ como sendo o custo de um caminho de 1 a j com menor custo, que consome no máximo r unidades de recurso, e assim temos a recorrência:

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

Podemos implementar um algoritmo que computa o valor de um caminho ótimo $OPT = f_n(l)$ em tempo $O(nml)$.

Algoritmo B

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:

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

OPT pode ser expresso como $\min\{c \mid g_n(c) \leq l\}$. Devemos computar g iterativamente, 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 $O(nmOPT)$.

Relaxação Lagrangeana - Primal

- Algoritmo proposto por Handler e Zang [4].

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.

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

Relaxação Lagrangeana - Primal

- 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.
- A restrição (2) nos garante que o conjunto de arcos induzido por um vetor x viável, não excede os recursos disponíveis.

Relaxação Lagrangeana - Primal

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.

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

Relaxação Lagrangeana - Dual

O problema é 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 usar-lá 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 $c_{ij} + ur_{ij}$, $(i, j) \in A$.

Relaxação Lagrangeana - Dual

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.

Relaxação Lagrangeana - Algoritmo

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 $L(u)$.

Relaxação Lagrangeana - Algoritmo (Inicialização)

O mais natural é que, como primeiro passo, verifiquemos se o menor caminho (não limitado, $\min_{x \in \mathcal{X}} c(x)$) 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.

Relaxação Lagrangeana - Algoritmo (Inicialização)

Como segundo passo, devemos verificar se o caminho que consome menor quantidade de recursos ($\min_{x \in \mathcal{X}} g(x)$) 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.

Relaxação Lagrangeana - Algoritmo

Com os resultados dos passos anteriores,

- temos a solução ou a prova de que a instância é inviável,
- ou temos 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 $g(x(\infty)) \leq 0$.

Relaxação Lagrangeana - Algoritmo

- Podemos interpretar cada caminho no grafo como uma reta no espaço (u, L) da forma $L = c(x) + ug(x)$.
- Isso nos permite dar uma interpretação geométrica para a função $L(u)$, que 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.

Relaxação Lagrangeana - Algoritmo

Como estamos procurando o valor de L^* 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 $g(x(0)) > 0$.

- Se existem apenas dois caminhos o ponto (u', L') é o que maximiliza $L(u)$.
- O mesmo acontece quando existem vários caminhos e $L(u') = L'$.
- 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).

Relaxação Lagrangeana - Algoritmo (Resolvendo o Dual)

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, das retas associadas a 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^+.

Relaxação Lagrangeana - Algoritmo (Resolvendo o Dual)

- O procedimento se repete até determinarmos a solução do problema (D).
- Ao final 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 UB é o valor do último c^-.

Relaxação Lagrangeana - Algoritmo (Eliminando a Folga)

- 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 vamos usar um algoritmo de k-ésimo menor caminho (*k-shortest path*) em relacão a função lagrangeana $L(u^*, x)$ (o que é equivalente a usar a função c' como custo, $c'_{ij} = c_{ij} + u^* \cdot r_{ij}$, $(i,j) \in A$).

Relaxação Lagrangeana - Algoritmo (Eliminando a Folga)

- 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 $UB = c(x_k)$ 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.

Relaxação Lagrangeana - Pseudocódigo

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

▷ Inicialização

01 $x_0, c_0, g_0 \leftarrow L(0)$
02 se $g_0 \leq 0$
03 então $x^*, c^* \leftarrow x_0, c_0$
04 senão $x^+, c^+, g^+ \leftarrow x_0, c_0, g_0$
05 $x_\infty, c_\infty, g_\infty \leftarrow L(\infty)$
06 se $g_\infty > 0$
07 então $x^*, c^* \leftarrow NULL, NULL$ ▷ Não tem solução!
08 senão $x^-, c^-, g^- \leftarrow x_\infty, c_\infty, g_\infty$

Relaxação Lagrangeana - Pseudocódigo

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

▷ Resolvendo o Dual

09 se $x^+ \neq NIL$ e $x^- \neq NIL$ ▷ Se setou os valores realativos a x^+ e x^- acima
10 $LB \leftarrow 0$; $UB \leftarrow c^-$
11 enquanto $LB < UB$ faça
12 $u' \leftarrow (c^- - c^+)/(g^+ - g^-)$; $L' \leftarrow c^+ + u'g^+$; $x', c', g' \leftarrow L(u')$
13 se $g' = 0$
14 então $x^*, c^* \leftarrow x', c'$; $LB \leftarrow UB \leftarrow c'$
15 PÁRA o enquanto
16 se $L(u') = L'$ e $g' < 0$
17 então $LB \leftarrow L'$; $UB \leftarrow \min\{UB, c'\}$; $x^- \leftarrow x'$; $u^* \leftarrow u'$
18 PÁRA o enquanto
19 se $L(u') = L'$ e $g' > 0$
20 então $LB \leftarrow L'$; $u^* \leftarrow u'$
21 PÁRA o enquanto
22 se $L(u') < L'$ e $g' > 0$
23 então $x^+, c^+, g^+ \leftarrow x', c', g'$
24 se $L(u') < L'$ e $g' < 0$
25 então $x^-, c^-, g^- \leftarrow x', c', g'$; $UB \leftarrow \min\{UB, c'\}$

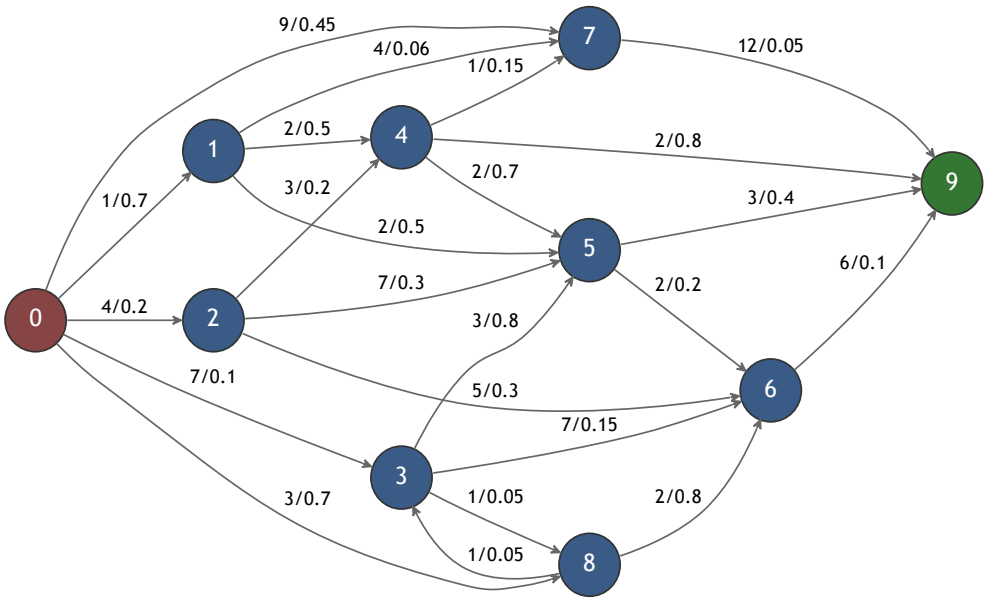
Relaxação Lagrangeana - Pseudocódigo

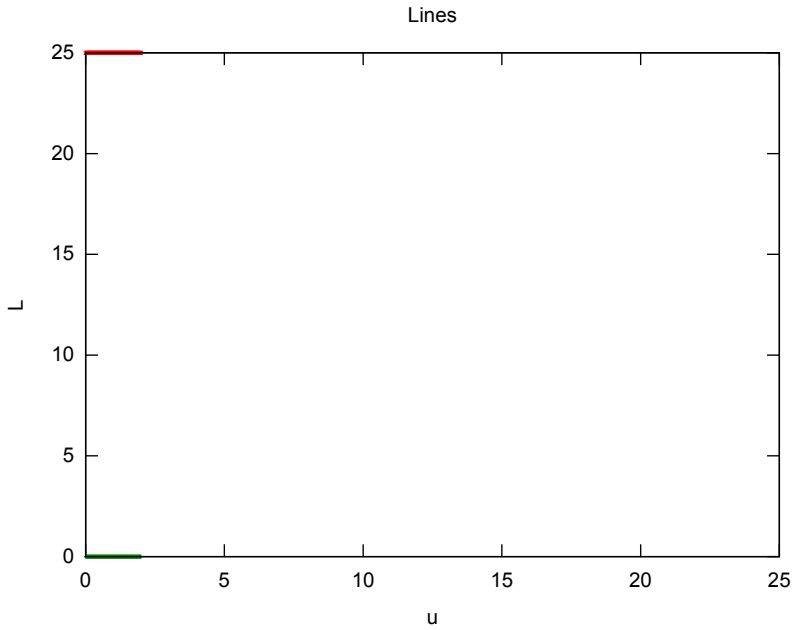
Algoritmo RCSP-LANGRANGIANA(G, $s = 1$, $t = n$, $k = 1$, r, l, c)

▷ Eliminando a folga da dualidade

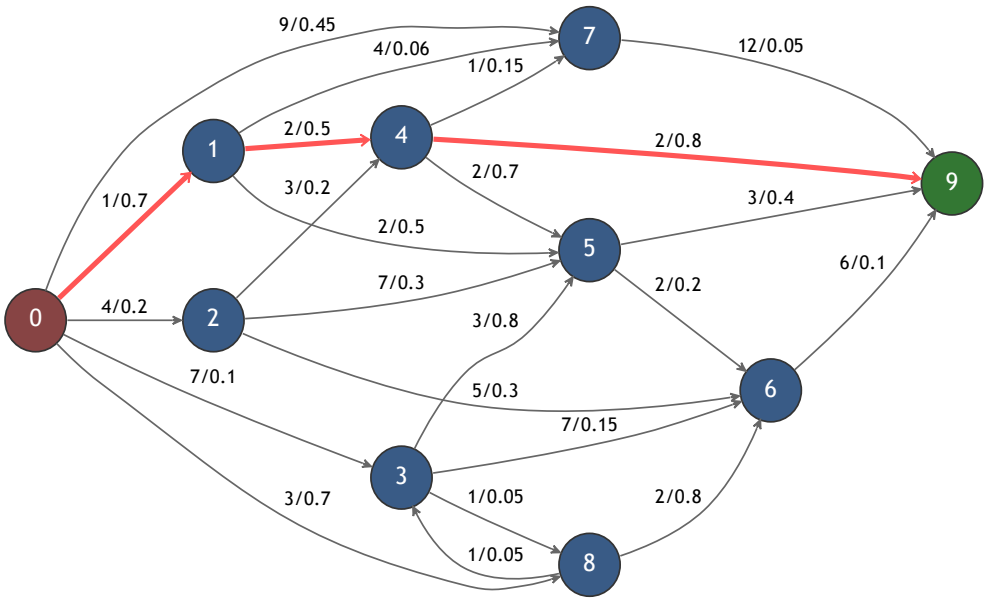
$x_1, x_2 \leftarrow x^+, x^-$; $k \leftarrow 2$
enquanto $LB < UB$ faça
  $k \leftarrow k + 1$; $x_k, c_k, g_k \leftarrow L_k(u^*)$; $LB \leftarrow L_k(u^*)$
  se $g_k \leq 0$ e $c_k < UB$
    então $x^-, UB \leftarrow x_k, c_k$
  se $LB \geq UB$
    então $x^*, c^* \leftarrow x^-, UB$
devolva x^*, c^*

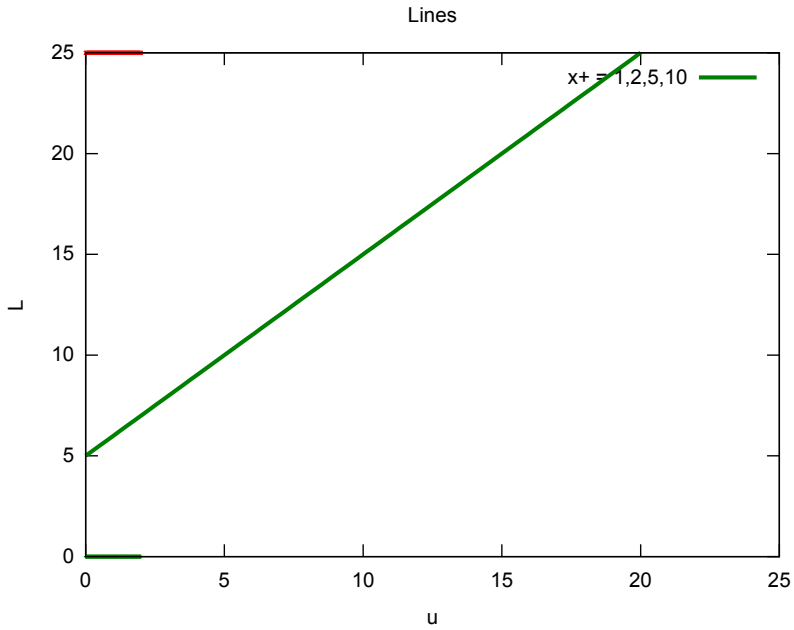
Relaxação Lagrangeana - Exemplo



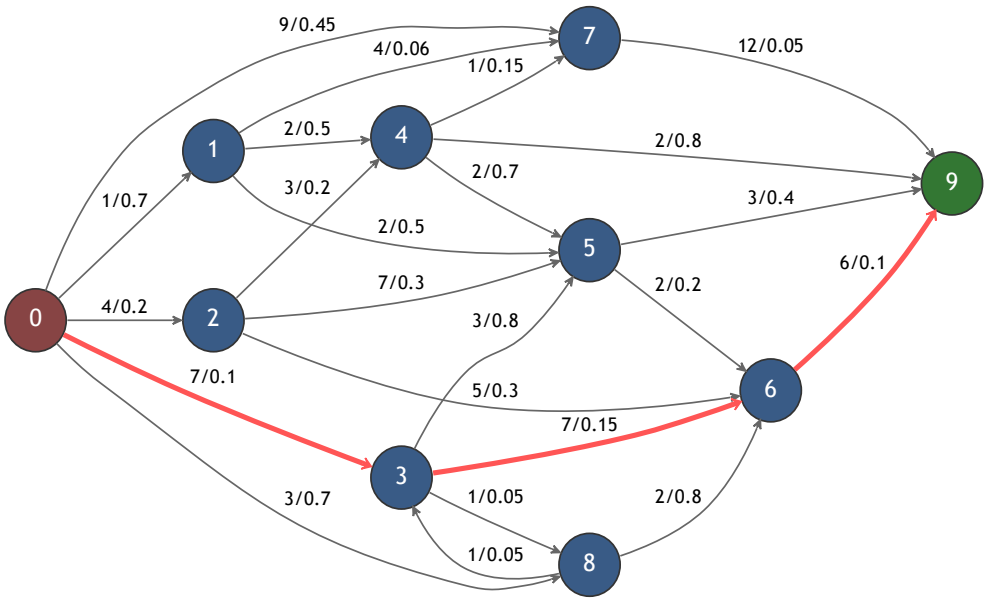


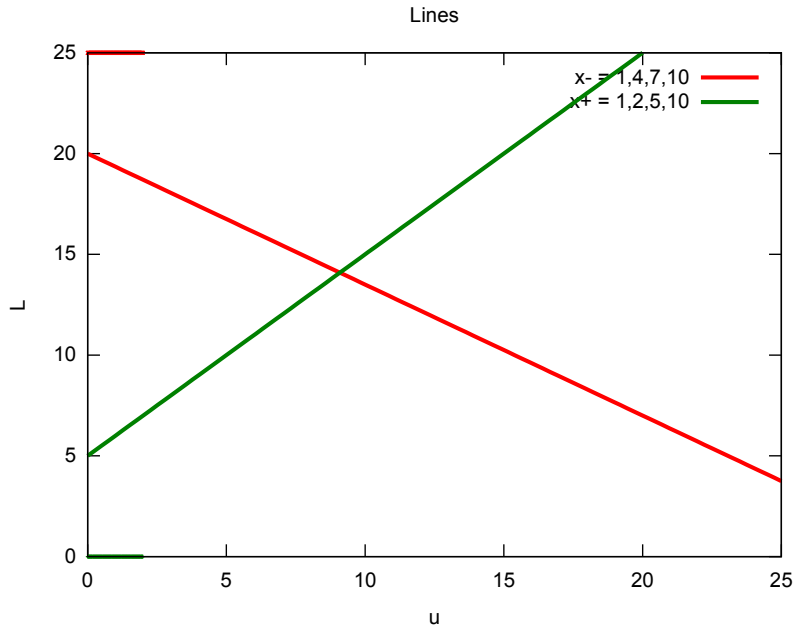
Relaxação Lagrangeana - Exemplo



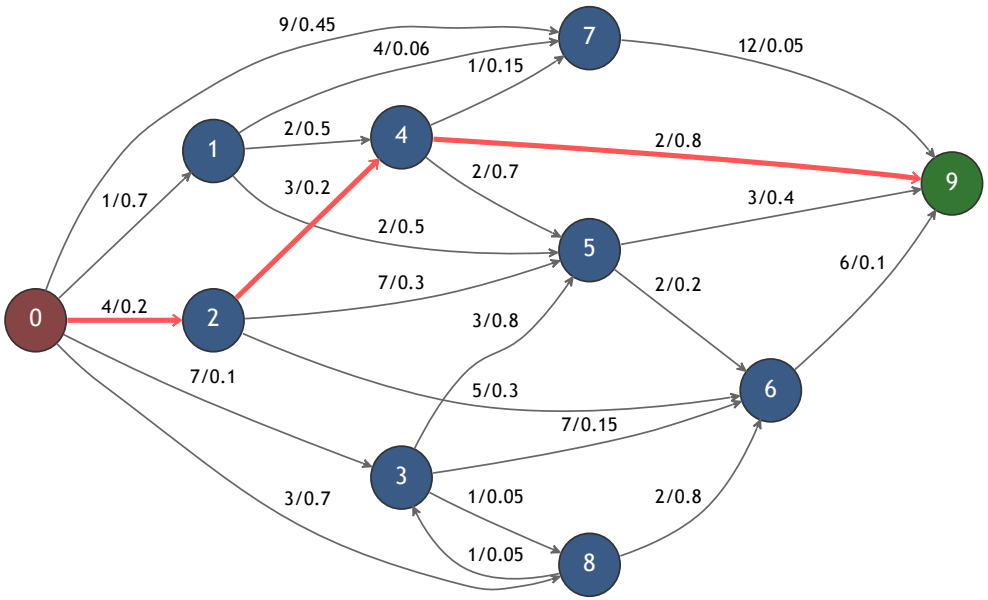


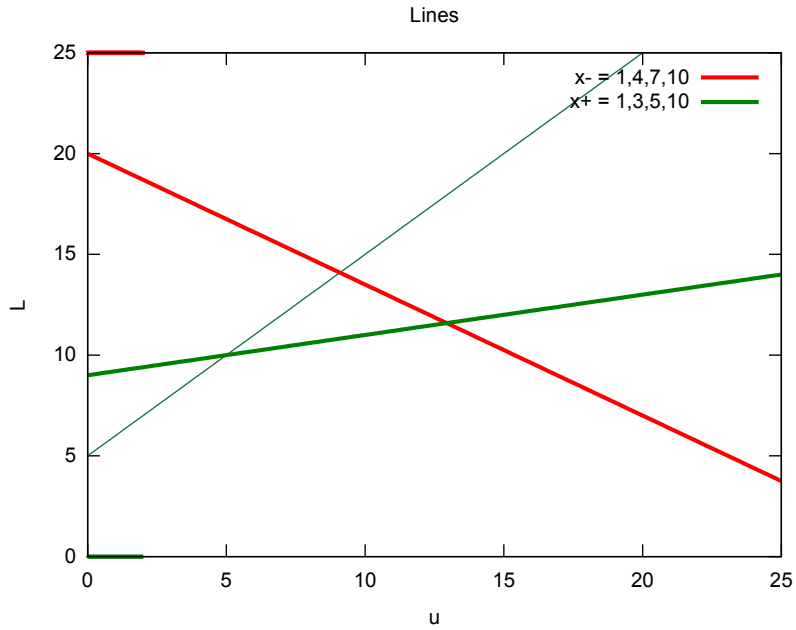
Relaxação Lagrangeana - Exemplo



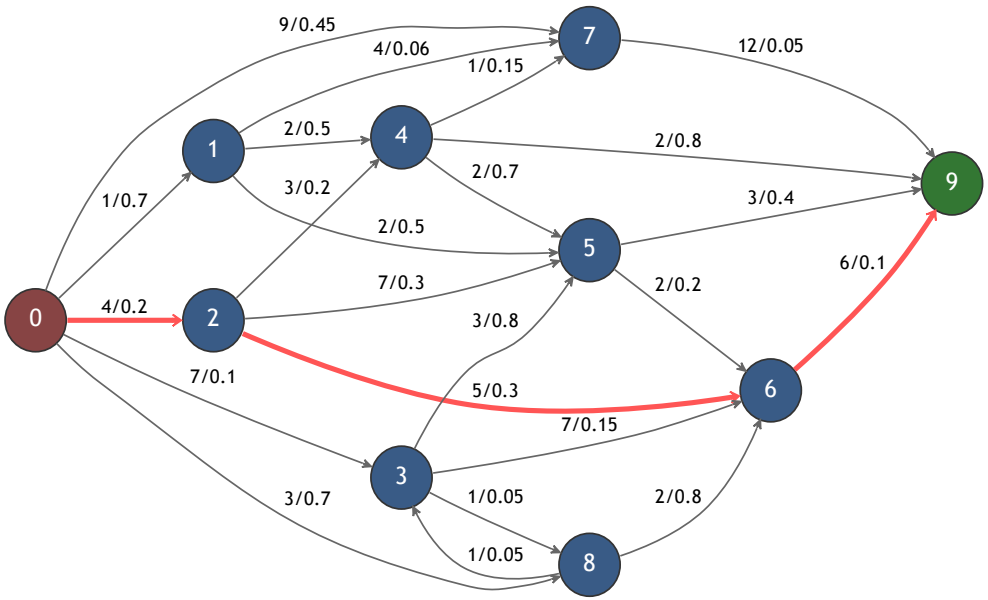


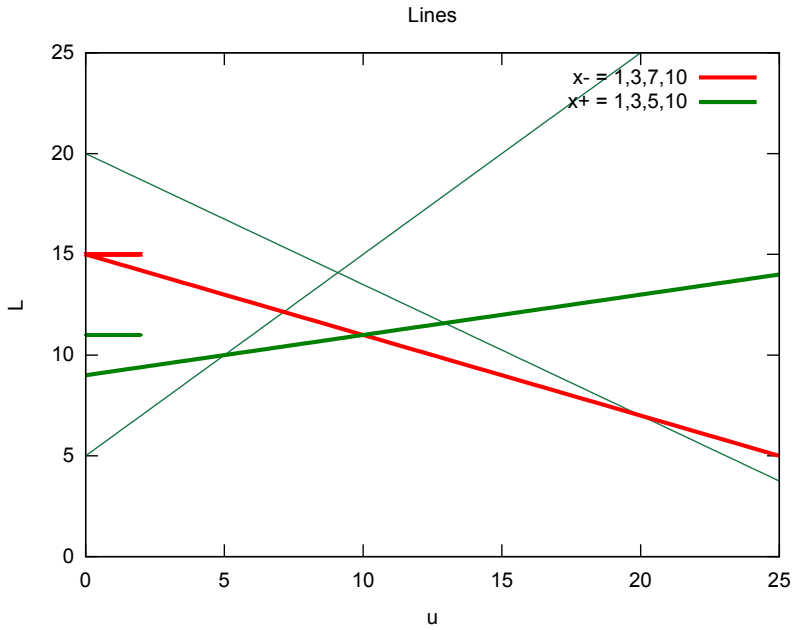
Relaxação Lagrangeana - Exemplo



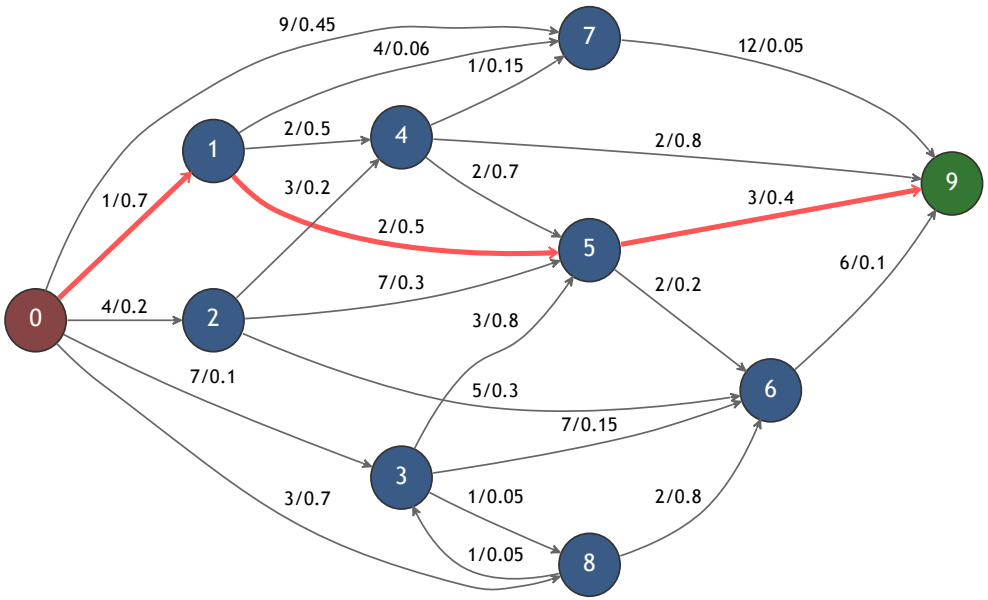


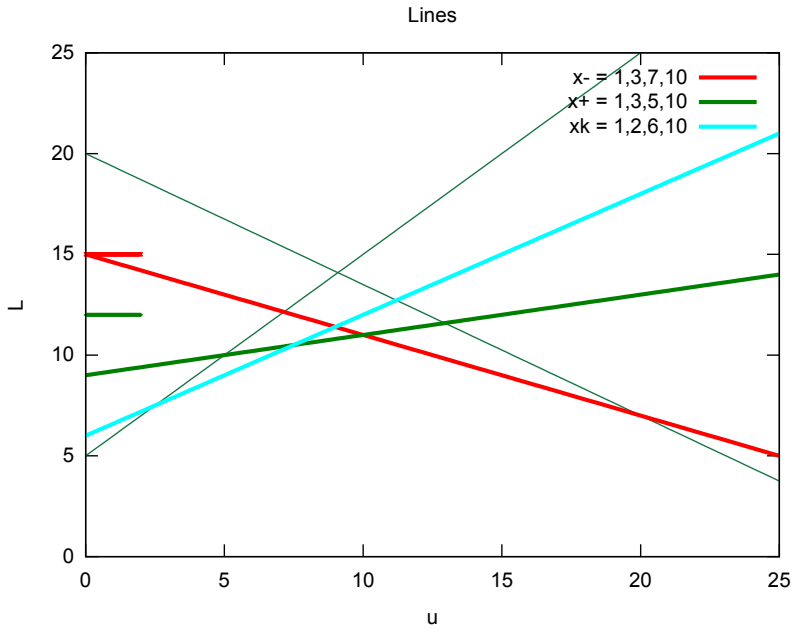
Relaxação Lagrangeana - Exemplo



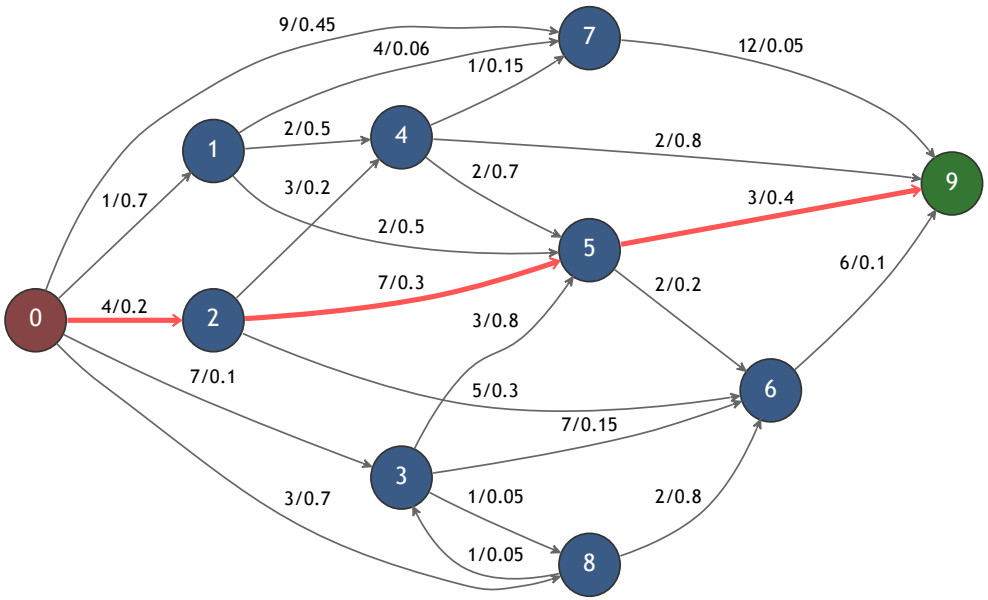


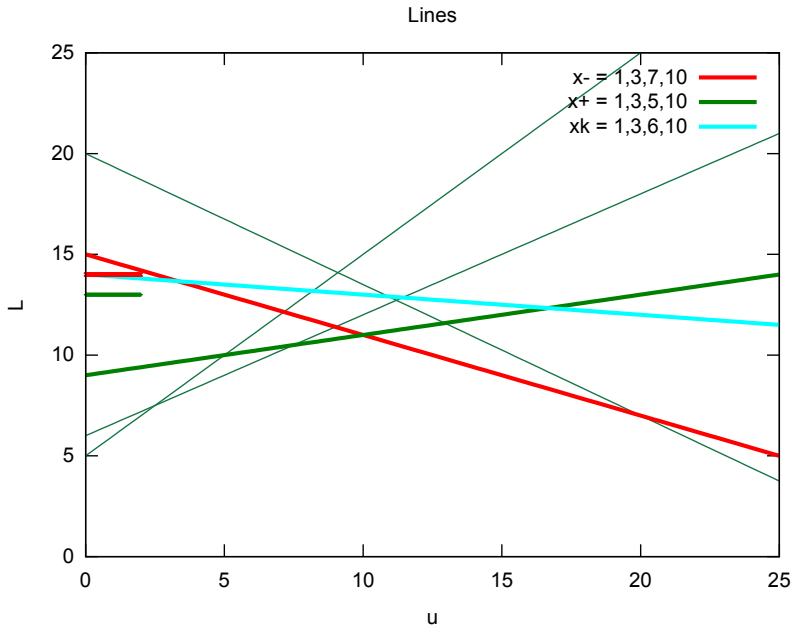
Relaxação Lagrangeana - Exemplo



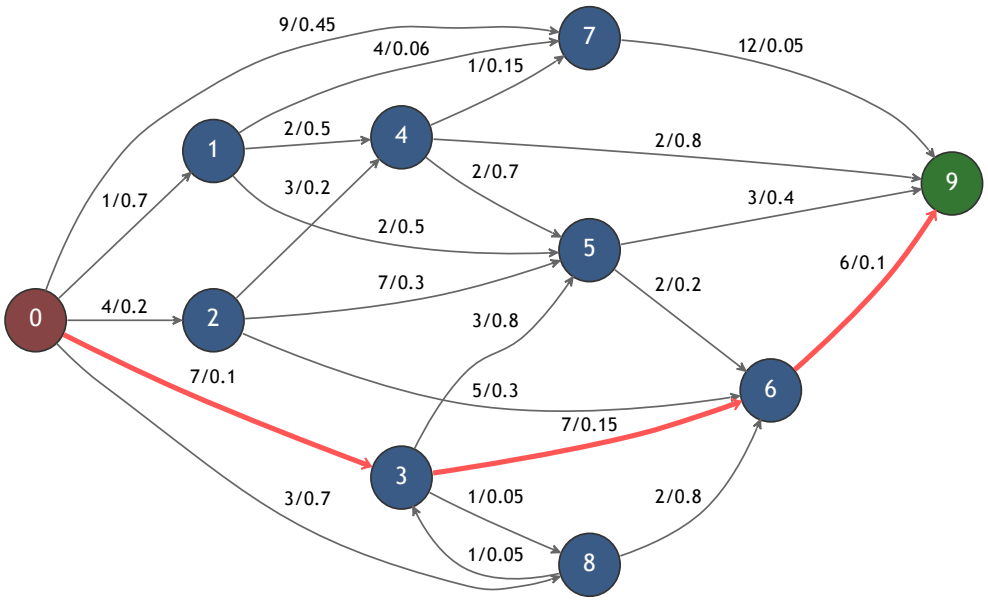


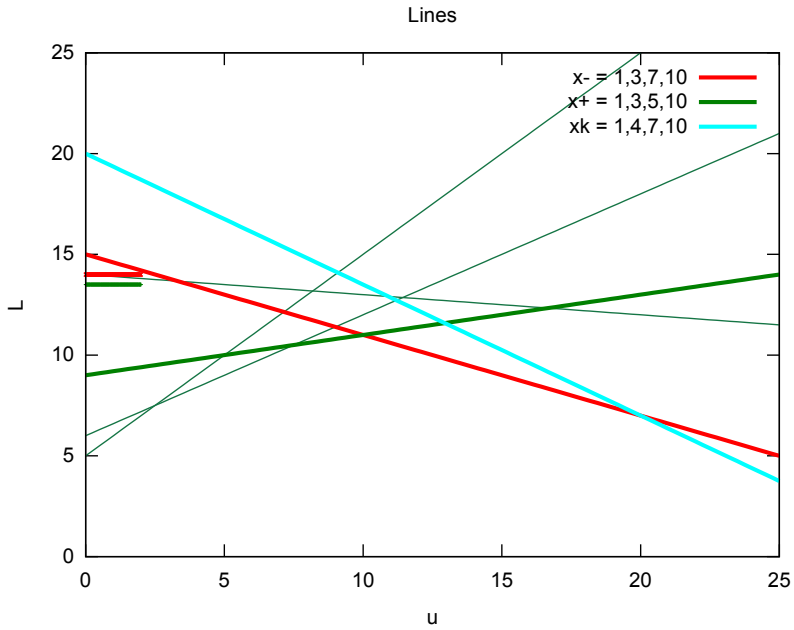
Relaxação Lagrangeana - Exemplo



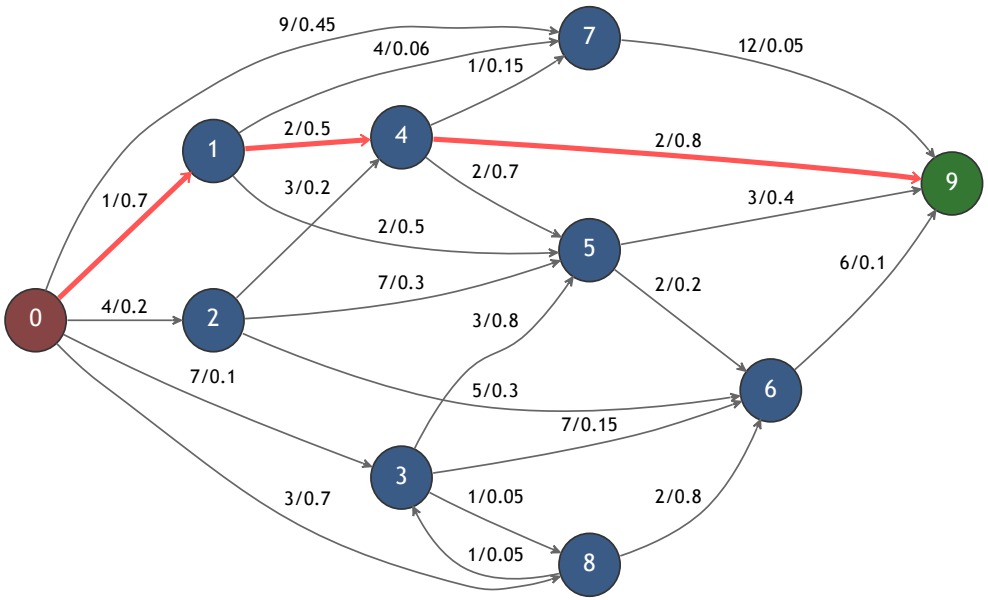


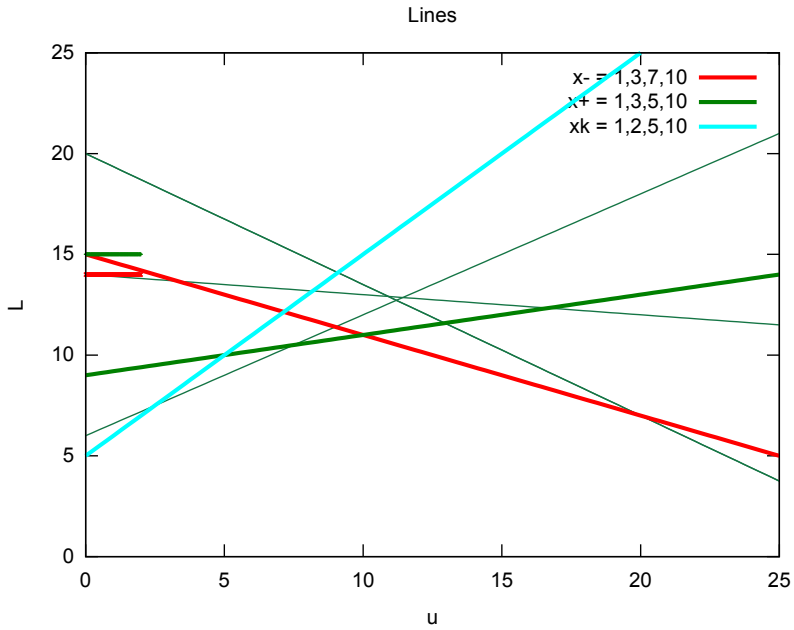
Relaxação Lagrangeana - Exemplo



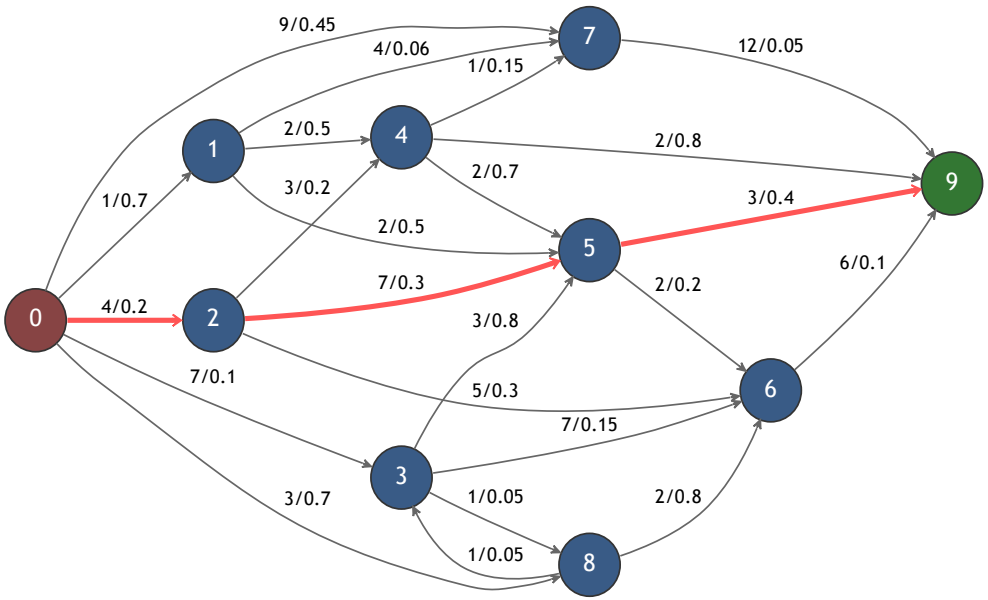


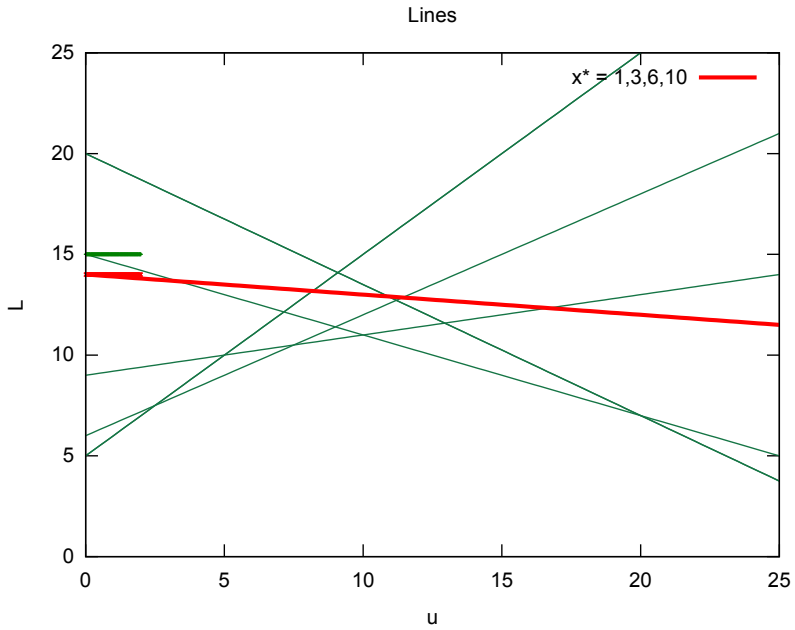
Relaxação Lagrangeana - Exemplo





Relaxação Lagrangeana - Exemplo





Projeto

Nesse projeto, pretendemos

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

Projeto

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

- Introdução: definição, histórico e aplicações para o problema
- Preliminares: notação, terminologias e conceitos básicos
- Complexidade
- Algoritmos exatos
- Algoritmos de aproximação
- Implementação e Resultados Computacionais

Projeto

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: *Distribuição do tempo por atividade*

Atividade	Set	Out	Nov	Dez	Jan	Fev	Mar
[1]	√	√					
[2]	√	√	√				
[3]		√	√	√			
[4]			√	√	√	√	
[5]							√

Referência Bibliográfica

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

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

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

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

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

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

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

C. Witzgall and A. J. Goldman.