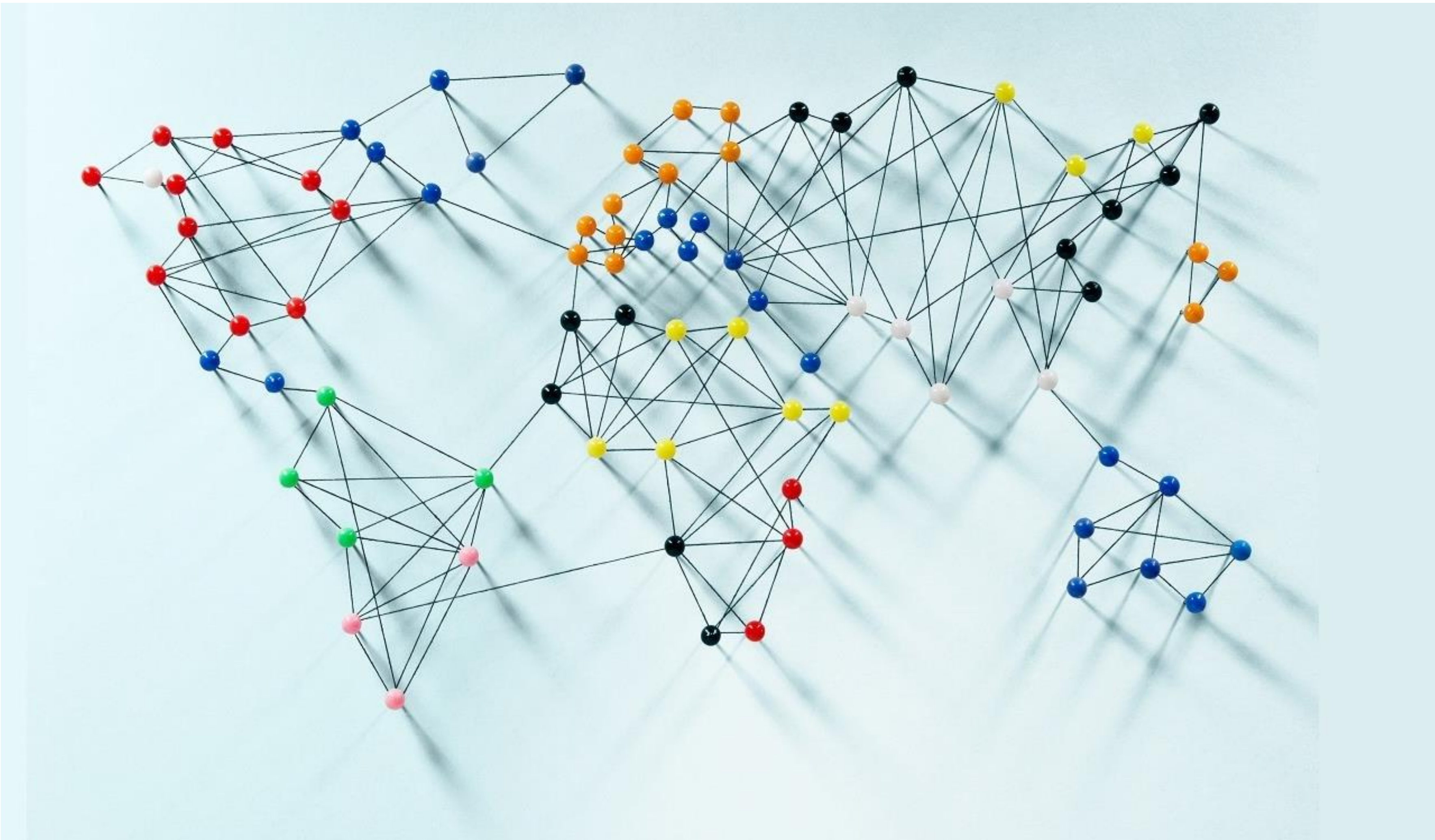
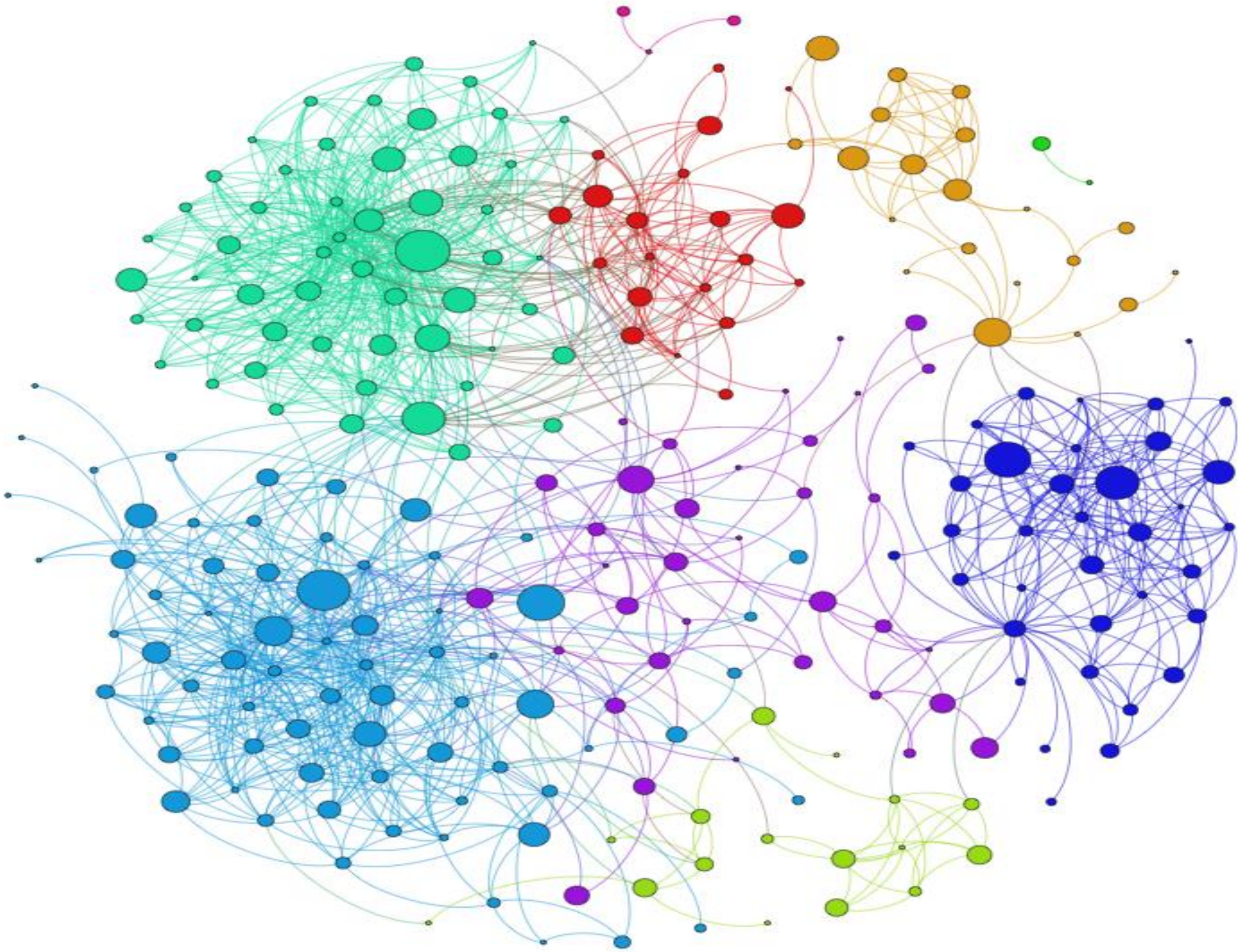
Programação Linear e Grafos

Sistemas de Informação - UNISUL
Aran Bey Tcholakian Morales, Dr. Eng.
(Apostila 2)

Teoria de Grafos



Problemas de Caminhos



Problemas de Caminhos

Caminho: é qualquer seqüência de **arestas orientadas** onde o vértice final de uma aresta é o vértice inicial da próxima.

Um caminho de **k** vértices é formado pôr **(k-1)** arestas (chamado de **comprimento (k-1)**)

Se todos os vértices do caminho são distintos, então temos um **caminho simples**.

Se todas as arestas do caminho são distintas, então temos um **caminho elementar**.

Quando o grafo **não é orientado** temos uma **cadeia** (e não mais um caminho) e as mesmas definições continuam válidas.

Problemas de Caminhos

Menor Caminho de um vértice a qualquer outro vértice:

Algoritmo de Dijkstra

Considere um grafo valorado **G(V, A)**, um vértice chamado origem v_0 e uma função **L** que associe cada aresta a um número real não negativo, isto é: $L(v_i,v_j) = \infty$, se não existe a aresta (v_i,v_j)

$= 0$, se $v_i = v_j$

$=$ **custo**, se $v_i \neq v_j$ e existe a aresta (v_i,v_j)

A matriz formada pela função **L** é chamada de matriz de distancia **D**.

O problema consiste em se determinar os caminhos (ou cadeias) do **vértice inicial** v_0 para cada vértice v_i de **G**, de tal forma que a somatória das distâncias das arestas envolvidas em cada caminho seja mínima.

Problemas de Caminhos

Algoritmo de Dijkstra

Inicio

$S \leftarrow \{v_0\}$; $w \leftarrow v_0$; $D[v0] \leftarrow 0$;

Para cada $v \in V - \{ S\}$ faça $D[v] \leftarrow L(v_0,v)$;

Enquanto $S \neq V$ faça

Escolha o vértice $w \in V\text{-}S$ tal que $D[w]$ seja mínimo;

Coloque w em S, isto é, faça $S \leftarrow S \cup \{w\}$;

Para cada $v \in V\text{-}S$ faça $D[v] \leftarrow \text{MIN}(D[v], D[w] + L(w,v))$

FimEnquanto

Fim

Problemas de Caminhos

Algoritmo de Dijkstra: Exemplo

	V_0	V_1	V_2	V_3	V_4
V_0		2			10
V_1			3		7
V_2				4	
V_3					
V_4			8	5	

Iter	S	W	D[W]	$D[V_1]$	$D[V_2]$	$D[V_3]$	$D[V_4]$
0	V_0	-	0	2	∞	∞	10
1	V_0,V_1	V_1	2	-	$(\infty,5)$	∞	(10,9)
2	V_0,V_1,V_2	V_2	5	-	-	$(\infty,9)$	9
3	V_0,V_1,V_2,V_3	V_3	9	-	-	-	9
4	V_0,V_1,V_2,V_3,V_4	V_4	9	-	-	-	-

Iter	S	W	D[W]	$D[V_1]$	$D[V_2]$	$D[V_3]$	$D[V_4]$
0	V_0	-	0	V_0			V_0
1	V_0,V_1	V_1	2	-	V_1		V_1
2	V_0,V_1,V_2	V_2	5	-	-	V_2	V_1
3	V_0,V_1,V_2,V_3	V_3	9	-	-	-	V_1
4	V_0,V_1,V_2,V_3,V_4	V_4	9	-	-	-	-

Caminhos: A V4 chego de V1 que chego de V0;
A V3 chego de V2 que chego de V1 que chego de V0
A V2 chego de V1 que chego de V0

Problemas de Caminhos

Algoritmo de Dijkstra: Exemplo

	V_0	V_1	V_2	V_3	V_4
V_0		1		0.5	
V_1			1		
V_2					5
V_3	1	3			1
V_4				1	

Iter	S	W	D[W]	D[V_1]	D[V_2]	D[V_3]	D[V_4]
0	V_0	-	0	1	∞	0,5	∞
1	V_0,V_3	V_3	0,5	(1, 3.5)	∞	-	(∞,1.5)
2	V_0,V_3,V_1	V_1	1	-	(∞,2)	-	1.5
3	V_0,V_3,V_1,V_4	V_4	1.5	-	2	-	-
4	V_0,V_3,V_1,V_4,V_2	V_2	2	-	-	-	-

Iter	S	W	D[W]	D[V_1]	D[V_2]	D[V_3]	D[V_4]
0	V_0	-	0	V_0		V_0	
1	V_0,V_3	V_3	0,5	V_0		-	V_3
2	V_0,V_3,V_1	V_1	1	-	V_1	-	V_3
3	V_0,V_3,V_1,V_4	V_4	1.5	-	V_1	-	-
4	V_0,V_3,V_1,V_4,V_2	V_2	2	-	-	-	-

Caminhos: A V3 chego de V0; a V1 chego de V0;
A V4 chego de V3 que chego de V0
A V2 chego de V1 que chego de V0

Problemas de Caminhos

Menor Caminho entre dois vértices: Algoritmo de Floyd

Idéia do Algoritmo

Construir uma matriz D_0 de custos de arestas, onde os laços possuem custo zero e à não existência de arestas atribui-se o custo infinito.

O algoritmo constrói, sucessivamente, **n** matrizes a partir de D_0, através de modificações efetuadas de acordo com o seguinte expressão:

$$d_{ij}(k) = \text{Min}\ [d_{ij}(k-1), (d_{iw}(k-1) + d_{wj}(k-1))] \quad \text{com } w = 0\ ..\ n-1$$

Para a determinação do caminho, parte-se do final para o início, levando-se em conta os vértices intermediários incluídos durante o processo.

Problemas de Caminhos

Algoritmo de Floyd

Inicio

1. Iniciar a matriz D_1 tal que $d_{ii} = 0$ e $d_{ij} = \infty$ quando não existe aresta (v_i,v_j), inicializar $k \leftarrow 1$; { o k é o passo ou número de iterações }
2. Para todo $w = 1 .. n$ faça $d_{ij}(k+1) = Min\ [d_{ij}(k), (d_{iw} + d_{wj}(k))]$
3. Se o número de iterações (k) é igual a $|V| - 1$ então parar

   Senão

   Se a matriz de distancia $D_k = D_{k-1}$ então parar

   Senão Faça $k \leftarrow k + 1$;
4. Retornar ao passo 2.

Fim

Problemas de Caminhos

Algoritmo de Floyd: Exemplo

0	1	∞	0.5	∞
∞	0	1	∞	1
∞	∞	0	∞	5
1	3	∞	0	1
∞	∞	∞	1	0

Caminhos de até 2 passos

0	1	∞	0.5	∞		0	1	2*	0.5	1.5*
∞	0	1	∞	1		∞	0	1	2*	1
∞	∞	0	∞	5		∞	∞	0	6 *	5
1	3	∞	0	1		1	2*	4*	0	1
∞	∞	∞	1	0		2*	4*	∞	1	0

Caminhos de até 3 passos

0	1	∞	0.5	∞		0	1	2	0.5	1.5
∞	0	1	∞	1		3*	0	1	2	1
∞	∞	0	∞	5		7*	9*	0	6	5
1	3	∞	0	1		1	2	3*	0	1
∞	∞	∞	1	0		2	3	5*	1	0

Caminhos de até 4 passos

0	1	∞	0.5	∞		0	1	2	0.5	1.5
∞	0	1	∞	1		3	0	1	2	1
∞	∞	0	∞	5		7	8*	0	6	5
1	3	∞	0	1		1	2	3	0	1
∞	∞	∞	1	0		2	3	4*	1	0

Problemas de Caminhos

Menor Caminho entre dois vértices: Algoritmo de Floyd

Matriz de Roteamento

Em diversas situações deseja-se saber qual é o menor caminho de um vértice a outro.

Uma maneira de conseguir esta informação é utilizar uma matriz **R**, onde em **R[i,j]** contém aquele vértice **k** que permite ao algoritmo de Floyd achar o menor valor **A[i,j]**. Se **R[i,j] = 0**, então o menor caminho de **i** para **j** é direto, seguindo a aresta (v_i,v_j) .

A versão do algoritmo de Floyd modificado fornece estes caminhos

Problemas de Caminhos

Algoritmo de Floyd: Matriz de Roteamento

```
Inicio
    Para  i = 1 até n faça
        Para  j = 1 até n faça
            A [ i, j ] ← D( i, j );
            R [ i, j ] ← 0;
    Para  i = 1 até n faça
            A [ i, j ] ← 0;
    Para  k = 1 até n faça
        Para  i = 1 até n faça
            Para  j = 1 até n faça
                Se A [ i, k ] + A [ k, j ] < A [ i,j ] então faça
                    A [ i,j ] ← A [ i, k ] + A [ k, j ] ;
                    R [ i, j ] ← k;
Fim.
```

Problemas de Caminhos

Algoritmo de Floyd: Roteamento

Caminhos de até 1 passo

Caminhos de até 2 passos

V0	V1		V3				V0	**V1**	V1	V3	V3
	V1	V2		V4				V1	V2	V4	V4
		V2		V4					V2	V4	V4
V0	V1		V3	V4			V0	V0	V1	V3	V4
			V3	V4			V3	V3		V3	V4

O valor **V1** foi obtido: (1+1: Vo => V1 e V1=>V2) sempre coloco o valor do meio

Caminhos de até 3 passos

Caminhos de até 4 passos

V0	V1	V1	V3	V3			V0	V1	V1	V3	V3
V4	V1	V2	V4	V4			V4	V1	V2	V4	V4
V4	V4	V2	V4	V4			V4	V4	V2	V4	V4
V0	V0	V0	V3	V4			V0	V0	V0	V3	V4
V3	V3	V3	V3	V4			V3	V3	V3	V3	V4

Caminhos V2 ao V1 (olhar na coluna do V1)

De V2 ao V4, do V4 ao V3 do V3 ao V0 do V0 ao V1