

Algoritmos para r-Arborescências Geradoras Mínimas em Digrafos: Uma Aplicação Web Interativa

Lorena Sampaio, Samira Haddad

Orientador: Prof. Dr. Mário Leston Rey

Universidade Federal do ABC
Centro de Matemática, Computação e Cognição

25 de novembro de 2025



Sumário

- 1 Introdução
- 2 Algoritmo de Chu-Liu-Edmonds
- 3 Algoritmo de András Frank
- 4 Resultados Experimentais
- 5 Aplicação Web
- 6 Conclusões



O Problema

Encontrar uma r -Arborescência Geradora de Custo Mínimo

Dado um r -digafo ponderado (D, w, r) :

- Encontrar uma r -arborescência geradora de custo mínimo de D

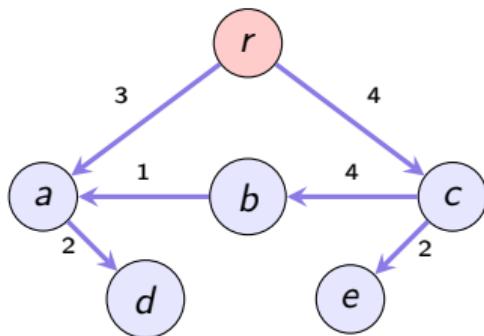
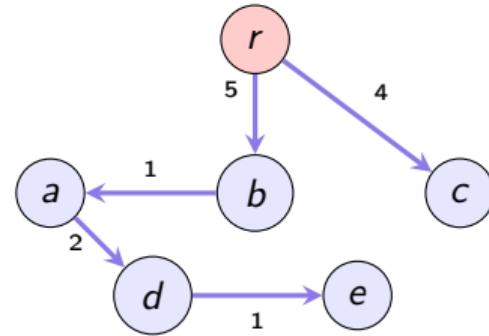
Algoritmos estudados:

- ① Chu-Liu-Edmonds (1965-67)
- ② András Frank (1981-2014)

Exemplo: r -Arborescência Geradora Mínima



Digrafo Original

 r -Arborescência Geradora

Geradora Mínima

Custo: 16

Custo: 13

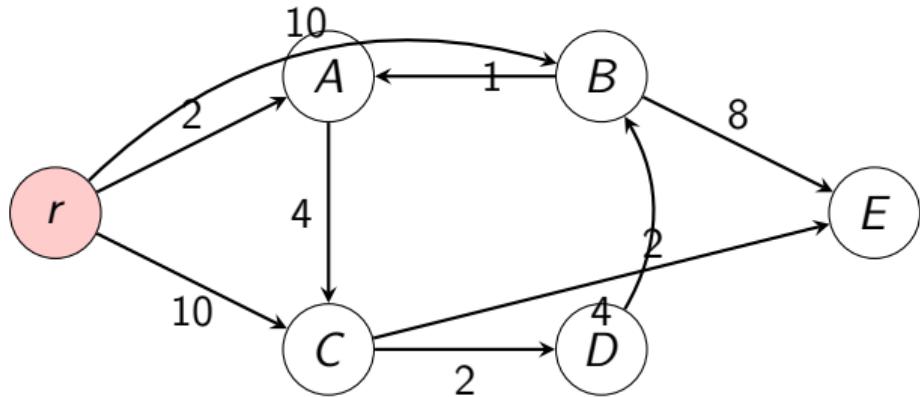
Chu-Liu-Edmonds: Ideia Principal

Algoritmo Recursivo: dado um r-digrafo ponderado (D, w, r)

chu-liu-edmonds((D, w, r)):

- ① **Reduzir custos:** para cada vértice $v \neq r$, subtrair $\lambda(v) = \min\{w(a) : a \in \delta^-(v)\}$
- ② **Construir D_0 :** escolhendo um arco a_v de custo reduzido zero para cada $v \neq r$
- ③ **Verificar:** se D_0 é uma r -arborescência \Rightarrow **devolver** D_0
Caso contrário:
- ④ **Contração:** encontrar ciclo C em D_0 e contrair
- ⑤ **Chamada recursiva:** Seja $D' = D/C$ e $w' = w_\lambda/C$. Calcular $T' =$
chu-liu-edmonds(D', w', r)
- ⑥ **Expandir:** **devolver** $\text{expand}(T')$

Digrafo de Exemplo



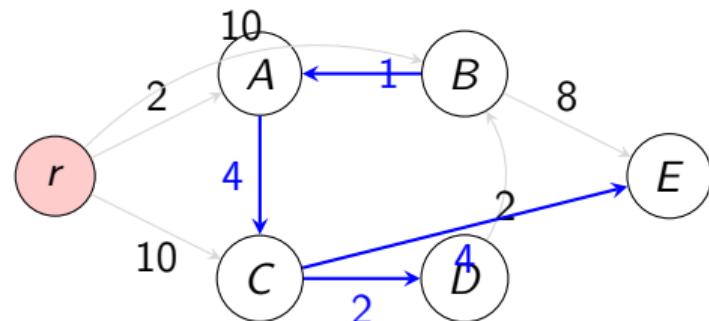
Objetivo

Encontrar a r -arborescência geradora de custo mínimo

Passo 1: Redução de Custos e Construção de D_0

Redução de custos λ :

- $\lambda(A) = 1 \Rightarrow (B, A): 1 - 1 = 0$
- $\lambda(B) = 2 \Rightarrow (D, B): 2 - 2 = 0$
- $\lambda(C) = 4 \Rightarrow (A, C): 4 - 4 = 0$
- $\lambda(D) = 2 \Rightarrow (C, D): 2 - 2 = 0$
- $\lambda(E) = 4 \Rightarrow (C, E): 4 - 4 = 0$



Escolhemos arcos de custo zero para formar D_0

D_0 não é uma r -arborescência

Ciclo detectado: $A \rightarrow C \rightarrow D \rightarrow B \rightarrow A$

Em outras palavras: há vértices não alcançáveis a partir de r

Passo 2: Contração do Ciclo

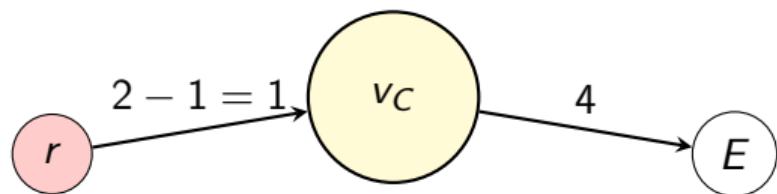
Ciclo C encontrado: $\{A, B, C, D\}$

Contraímos C em supervértice x_C

Em outras palavras: todos os vértices de C viram um único vértice

Custos já foram reduzidos por λ :

$$w_\lambda(r, A) = 2 - 1 = 1$$

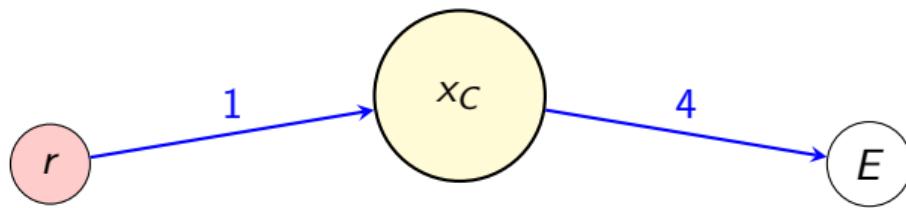


Chamada Recursiva

Resolvemos chu-liu-edmonds($D/C \mapsto x_C, w_\lambda/C \mapsto x_C, r$)

Em outras palavras: aplicamos o mesmo algoritmo no digrafo menor

Passo 3: Solução Recursiva no Digrafo Contraído



O algoritmo devolve $T' = \{(r, xc), (xc, E)\}$

Em outras palavras: encontramos a r -arborescência ótima no grafo contraído

Próximo passo: expandir T' de volta para obter T no digrafo original

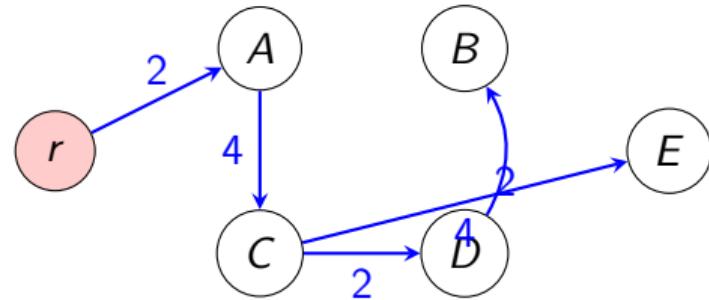
Passo 4: Expansão da Solução

Função expand(T'):

Arco (r, x_C) em T' corresponde a (r, A) em D

Em outras palavras:

- Incluir (r, A) que entra no ciclo
- Incluir todos os arcos do ciclo C exceto (B, A)
- Resultado: cada vértice tem grau de entrada 1



Solução Ótima

Custo total: $2 + 4 + 2 + 2 + 4 = 14$

András Frank: Visão Geral

Abordagem em Duas Fases

Fase I: Construir cobertura de subconjuntos minimais via redução de custos

Fase II: Extrair arborescência da cobertura

Diferencial:

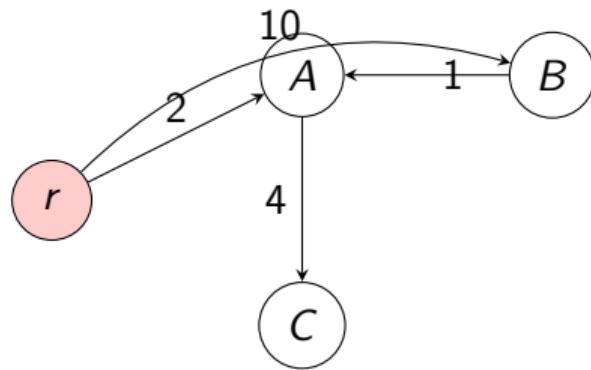
- Trabalha com múltiplos vértices simultaneamente
- Usa componentes fortemente conexas
- Redução sistemática de custos

Complexidade:

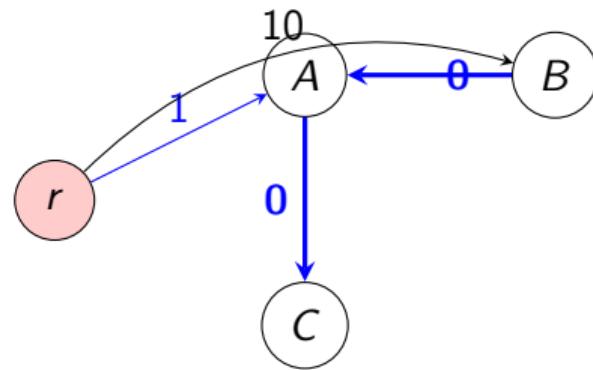
- Fase I: $O(nm)$
- Fase II v1 (lista): $O(n^2)$
- Fase II v2 (heap): $O(n \log n)$

Fase I: Redução de Custos

Para cada vértice $v \neq r$: subtrair o mínimo de entrada



Original



Após Redução

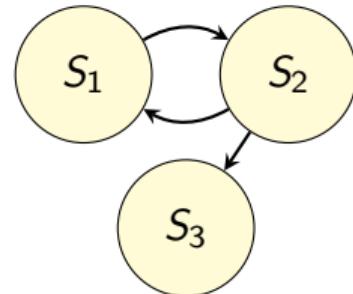
Arcos com custo zero formam o digrafo D_0

Fase I: Componentes Fortemente Conexas

Identificar componentes fortemente conexas (CFCs) em D_0

Cada CFC forma um **subconjunto minimal**

Construir sequência laminar de subconjuntos



Condição de Optimalidade

Sequência λ satisfaz: $|\delta^-(X)| = 1$ para cada X em λ

Fase II: Construção da Arborescência

Objetivo: Extrair arborescência de D_0 respeitando λ

- ① Iniciar com conjunto $R = \{r\}$
- ② Para cada v fora de R :
 - Selecionar arco (u, v) com $u \in R$ e custo reduzido zero
 - Adicionar v a R
- ③ Repetir até incluir todos os vértices

Resultado

Arborescência ótima com mesma solução: custo 14

Comparação de Desempenho

Experimentos: 2000 digrafos aleatórios, $|V| \in [101, 4996]$

Algoritmo	Tempo Mediano	Tempo Médio
Chu-Liu-Edmonds	0,25 s	0,58 s
Frank Fase I	8,93 s	12,40 s
Frank Fase II (lista)	0,98 s	1,34 s
Frank Fase II (heap)	0,016 s	0,020 s

Speedup Fase II

Heap vs Lista: aceleração de 58,12 vezes (mediana)

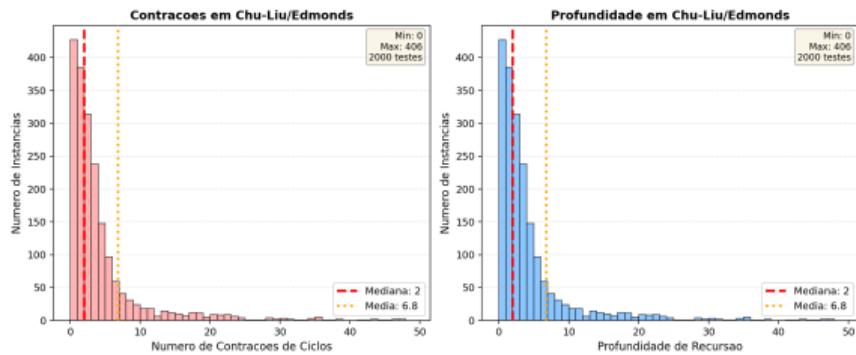
Características Estruturais

Contrações (Chu-Liu):

- Mediana: 2 contrações
- Média: 6,82
- Máximo: 406
- 93,8% com < 20

Muito abaixo do limite teórico $O(n)$

Consumo de memória: mediana 11,5 MB (Fase I)



Motivação Didática

Desafio

Algoritmos de grafos são **abstratos e difíceis de visualizar**

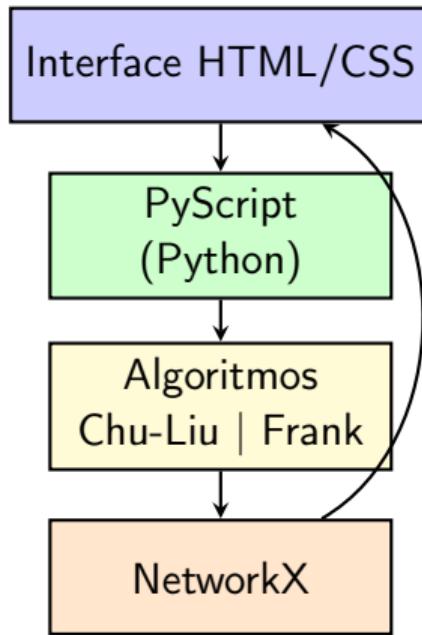
Solução Proposta:

- Visualização interativa
- Execução passo a passo
- Feedback imediato
- Acessível via navegador

Tecnologias:

- PyScript (Python no browser)
- JavaScript
- HTML5/CSS3
- NetworkX

Arquitetura da Aplicação



Interface: Página Principal

 ArboGraph

Algoritmos para o problema da arborescência geradora mínima: uma aplicação didática interativa

-  Home
-  Chu-Liu-Edmonds
-  András Frank (V1)
-  András Frank (V2)
-  Desenhe um digrafo
-  Nossa dissertação

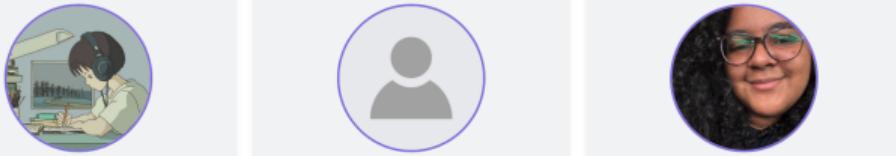
Dúvidas ?
Quer aprender mais sobre esses algoritmos, leia nossa tese :)

[Link](#)

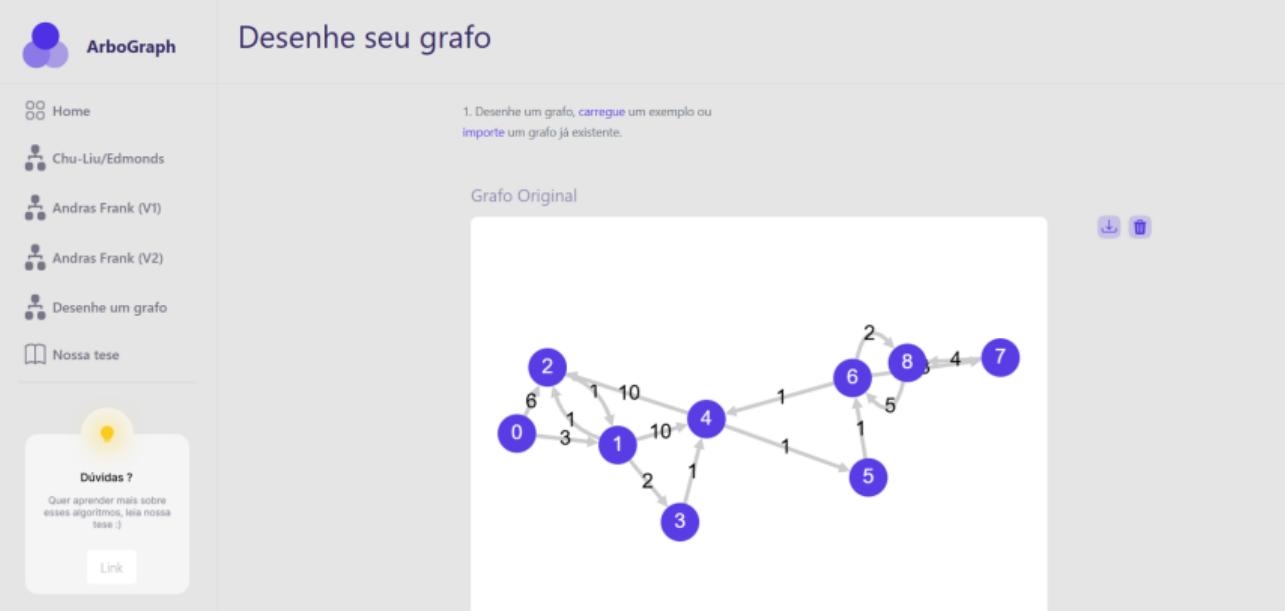
Resumo

Este trabalho investiga e implementa algoritmos de busca de uma r -arborescência geradora mínima em digrafos. A partir da formulação clássica e da literatura de Chu-Liu-Edmonds e também da formulação de András Frank, desenvolvemos uma aplicação web que permite: (i) desenhar ou importar um digrafo ponderado, (ii) escolher o nó raiz r , (iii) executar o algoritmo passo a passo com visualização das contrações, seleção de arcos de custo mínimo e reconstrução da arborescência, e (iv) exportar resultados e logs. A solução combina PyScript e NetworkX para a lógica algorítmica, Cytoscape para edição e visualização interativa, e Tailwind/Flowbite na interface. Como contribuição, o sistema oferece um ambiente didático que torna transparentes as decisões do algoritmo e facilita a análise e comparação de soluções em diferentes instâncias, apoiando ensino, experimentação e validação.

Integrantes do Projeto



Interface: Desenho de Grafos



The screenshot shows the ArboGraph web application interface. On the left, there's a sidebar with a logo and several menu items: Home, Chu-Liu/Edmonds, Andras Frank (V1), Andras Frank (V2), Desenhe um grafo, and Nossa tese. Below these is a 'Dúvidas?' section with a yellow lightbulb icon, a text box about a thesis, and a 'Link' button. The main area is titled 'Desenhe seu grafo' and contains instructions: '1. Desenhe um grafo, carregue um exemplo ou importe um grafo já existente.' A sub-section titled 'Grafo Original' displays a directed graph with 9 nodes (0-8) and edges with weights: (0,1):3, (0,2):6, (1,0):1, (1,2):1, (1,3):2, (1,4):10, (1,5):1, (2,1):1, (2,3):1, (3,4):10, (3,5):1, (4,1):2, (4,5):1, (4,6):1, (5,6):1, (5,8):5, (6,7):2, (6,8):4, (7,8):3. There are also save and delete icons.

Funcionalidades:

- Adicionar vértices e arestas
- Definir pesos

Interface: Chu-Liu-Edmonds



ArboGraph

- Home
- Chu-Liu/Edmonds
- ▢▢ Andras Frank (V1)
- ▢▢ Andras Frank (V2)
- ▢▢ Desenhe um grafo
- ▢▢ Nossa tese

Dúvidas ?

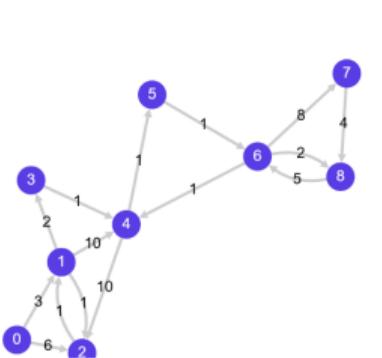
Quer aprender mais sobre esses algoritmos, leia nossa tese :)

[Link](#)

Chu-Liu / Edmonds

1 Crie um grafo
Desenhe um grafo, [carregue](#) um exemplo ou [importe](#) um grafo já existente.
2 Escolha o nó raiz
3 Execute o algoritmo
[»](#)

Grafo Original



```

graph LR
    0 -- 6 --> 1
    0 -- 1 --> 2
    1 -- 3 --> 3
    1 -- 10 --> 4
    1 -- 10 --> 5
    2 -- 1 --> 4
    3 -- 2 --> 4
    4 -- 1 --> 5
    4 -- 1 --> 6
    4 -- 10 --> 0
    5 -- 1 --> 6
    6 -- 8 --> 7
    6 -- 2 --> 8
    6 -- 5 --> 8
    7 -- 4 --> 8
    8 -- 2 --> 6
    8 -- 4 --> 7
  
```

» Passo A Passo
Execute o algoritmo para visualizar o passo-a-passo

- Visualização passo a passo
- Destacamento de ciclos detectados
- Log detalhado das operações

Interface: András Frank

The screenshot shows the ArboGraph web application interface. On the left, there is a sidebar with a purple logo and the text "ArboGraph". Below the logo are several menu items: "Home", "Chu-Liu/Edmonds", "Andras Frank (V1)", "Andras Frank (V2)", "Desenhe um grafo", and "Nossa tese". A yellow button labeled "Dúvidas ?" with a lightbulb icon is also present. The main area is titled "Andras Frank v1" and contains three numbered steps: 1. "Crie um grafo" (Create a graph), which includes instructions to draw a graph, upload an example, or import an existing one; 2. "Escolha o nó raiz" (Select the root node), with a text input field containing the word "node"; 3. "Execute o algoritmo" (Run the algorithm), with a blue play button icon. To the right of these steps is a button labeled "Passo A Passo" (Step by Step). Below the steps is a section titled "Grafo Original" (Original Graph) which is currently empty. A tooltip on the right says "Execute o algoritmo para visualizar o passo-a-passo" (Run the algorithm to visualize step-by-step).

- Exibição das duas fases
- Visualização de CFCs
- Comparação entre versões (lista vs heap)

Princípios de Design

Teoria dos Registros de Representação (Duval)

Transitar entre diferentes representações:

- **Visual:** diagramas do grafo
- **Simbólico:** código Python
- **Textual:** log das operações

Feedback Imediato

Validação em tempo real das operações do usuário



UFABC

Contribuições do Trabalho

① Implementação completa de dois algoritmos clássicos

- Chu-Liu-Edmonds: recursivo com contração
- András Frank: duas fases com otimização heap

② Análise experimental detalhada

- 2000 instâncias aleatórias
- Comparação de desempenho e características estruturais

③ Aplicação web interativa

- Ferramenta didática para visualização
- Execução passo a passo dos algoritmos
- Design centrado no usuário



Principais Resultados

- **Corretude validada:** custos idênticos em todas as instâncias
- **Chu-Liu-Edmonds** mais rápido para construção direta
 - Mediana: 0,25 s vs 8,93 s (Fase I Frank)
- **Otimização heap fundamental** na Fase II
 - Speedup: 58× (mediana), 61× (média)
- **Comportamento prático** muito melhor que limites teóricos
 - Contrações: mediana 2 (limite $O(n)$)
 - Memória modesta: 11,5 MB



UFABC

Trabalhos Futuros

Extensões Possíveis

- Implementar outras variantes (Tarjan, Gabow)
- Análise em grafos com estruturas especiais
- Paralelização dos algoritmos
- Extensão para grafos dinâmicos

Melhorias na Aplicação

- Modo de edição visual de grafos
- Geração automática de casos de teste
- Exercícios interativos com correção automática
- Integração com plataformas de ensino (Moodle, Jupyter)

Obrigado!

Perguntas?

<https://github.com/lorenypsum/graph-visualizer>