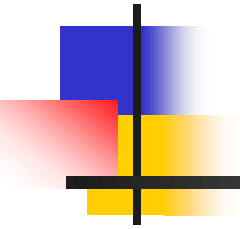
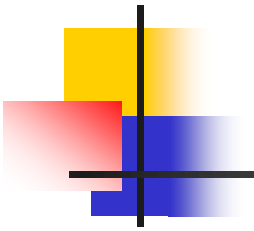


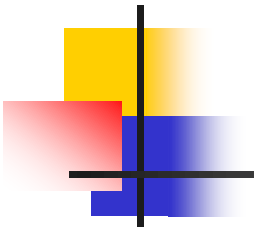
Escalonadores para HW/SW Codesign

Remy Eskinazi Sant´Anna
GRECO – Cin - UFPE



Agenda

- Algoritmos de escalonamento
- Instancias Básicas do problema de escalonamento
  - Escalonamento com restrição de recursos
    - ASAP Scheduling
    - ALAP Scheduling
    - Algorítmo de Hu (List Scheduling)
  - Escalonamento com restrição de tempo
    - Force-Directed Scheduling:
  - Advanced Scheduling Issues
    - As fast as possible scheduling (AFAP)
    - Pipeline path based scheduling

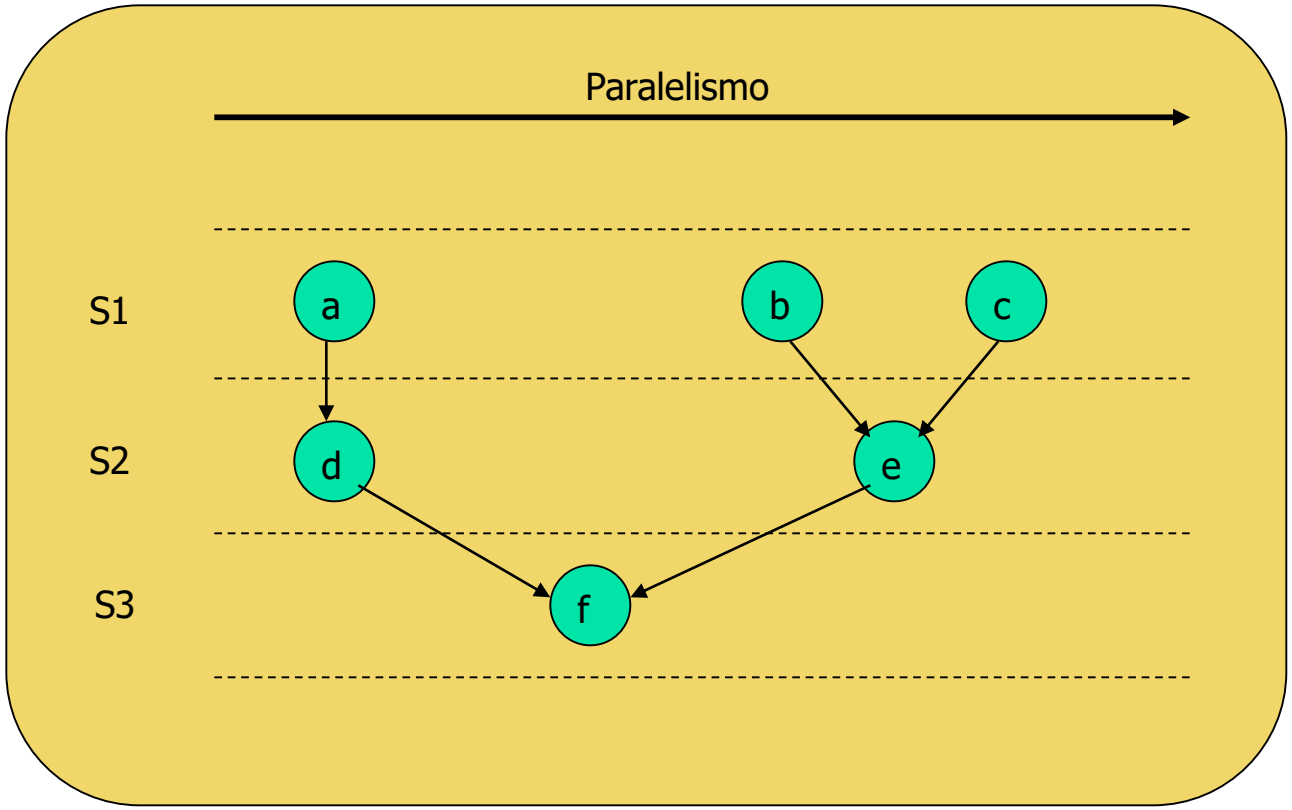


Conceitos básicos de escalonamento

- Algorítmos de escalonamento:
  - Data flow-based scheduling (DFBS)
  - Control flow-based scheduling (CDFS)

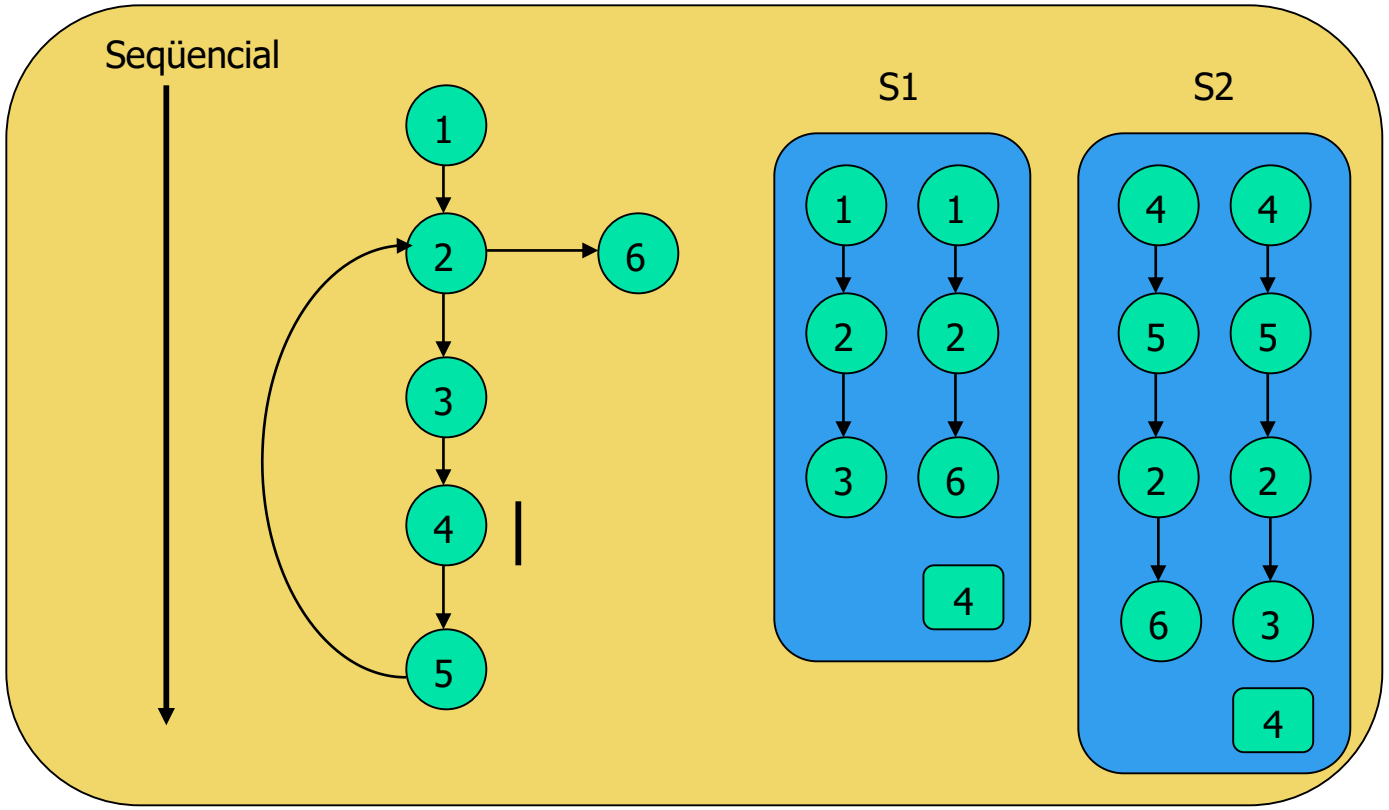
Escalonamento DFBS

- Paralelismo dado pela não dependência de dados entre operações
- Restrições de recursos podem diminuir paralelismo de operações



Escalonamento CFBS

- Representação seqüencial de uma descrição comportamental



Escalonamento com restrição de precedência

- Definição:

Considerando:

T => Conjunto de tarefas de duração unitária
η => Ordenação parcial em T
M => Número de processadores ($m \in Z^+$)
D => Tempo (limite) total, então...

Existe um escalonamento $\theta : T \rightarrow \{0,1,...D\}$, tal que

$$|\{ t \in T : \theta(t) = s \}| \leq m \ \forall \ s \in \{0,1,...D\} \quad e,$$

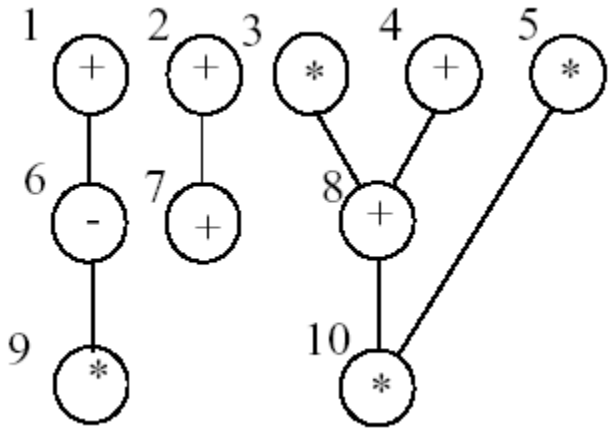
$$t_i \ \eta \ t_j \Rightarrow \ \theta(t_i) < \theta(t_j)$$

Escalonamento com restrição de recursos

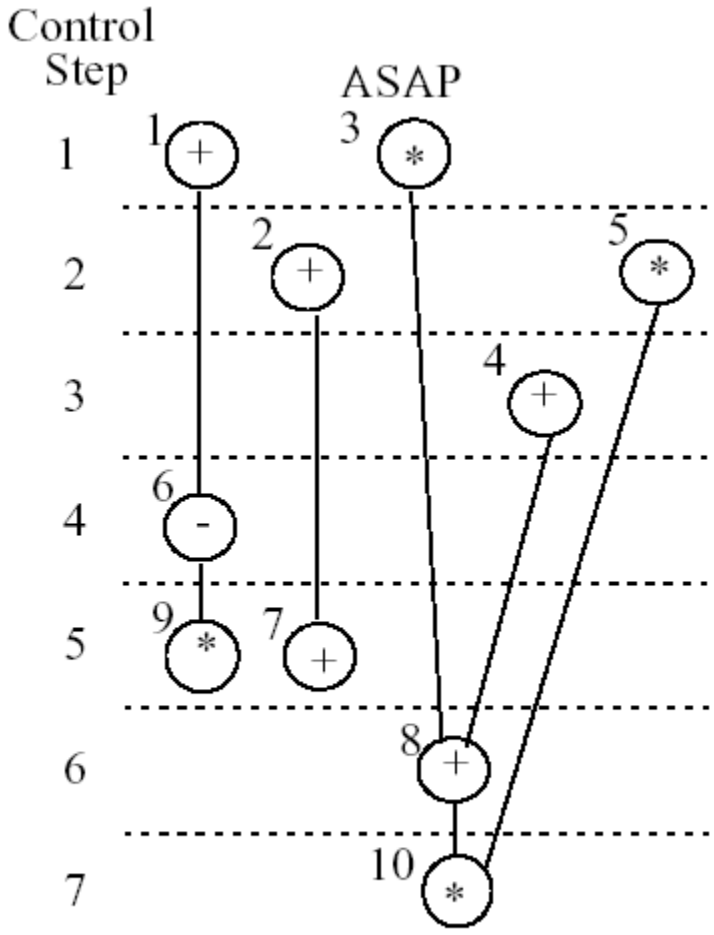
- ASAP – As soon as possible
  - Faz a ordenação das operações topologicamente, de acordo com seu fluxo de dados/controle
  - Faz o escalonamento das operações já ordenadas colocando-as no passo de controle o mais breve possível

Escalonamento com restrição de recursos

- ASAP – As soon as possible



(a) Sorted DFG



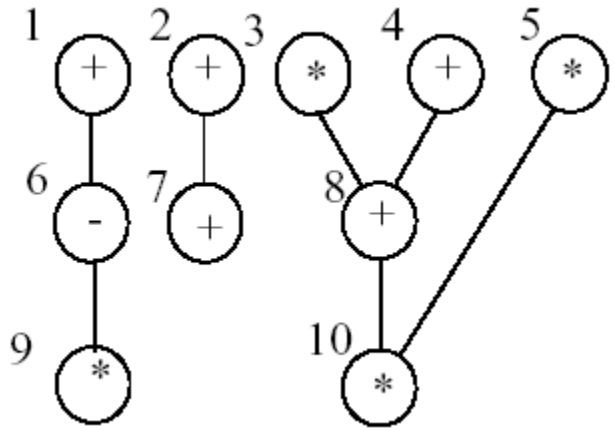
(b) ASAP schedule

Escalonamento com restrição de recursos

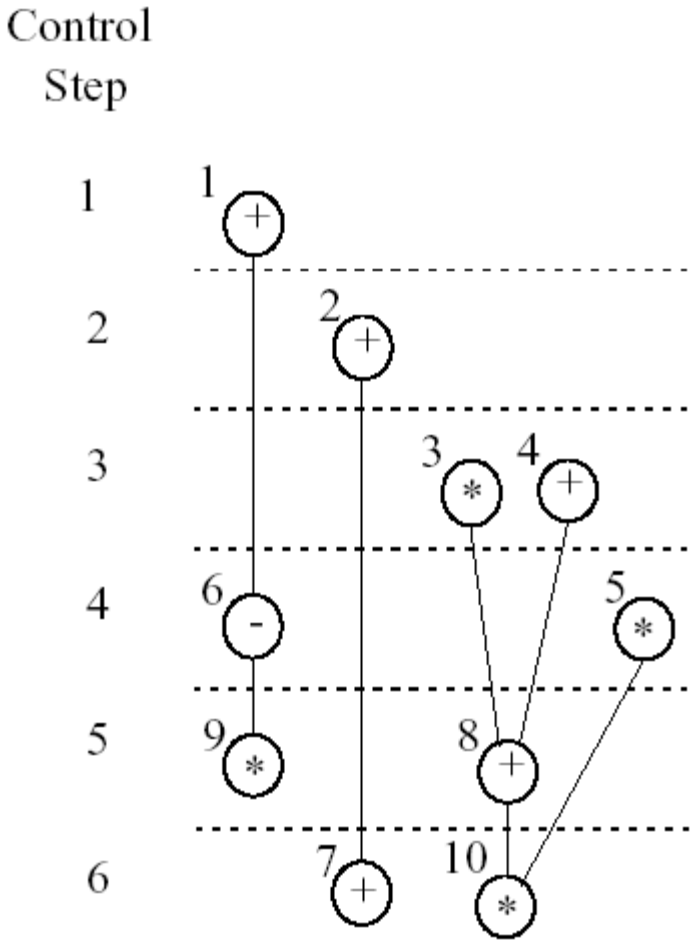
- ALAP – As late as possible
  - Faz a ordenação das operações topologicamente, de acordo com seu fluxo de dados/controle
  - Faz o escalonamento das operações já ordenadas colocando-as no passo de controle o mais atrasado possível

Escalonamento com restrição de recursos

- ALAP – As late as possible



(a) Sorted DFG



(b) ALAP schedule

Função de Priorização (Caddy Priority Function)

- Prioriza as funções a serem executadas em um list scheduling
- Calcula a probabilidade da operação ser executada em 1 step;
- Time Frame $\Delta T_{(O)}$ => Numero do steps de controle em que uma operação pode iniciar a execução
- $\Delta T_{(O)} = \sigma_{ALAP(O)} - \sigma_{ASAP(O)} + 1$

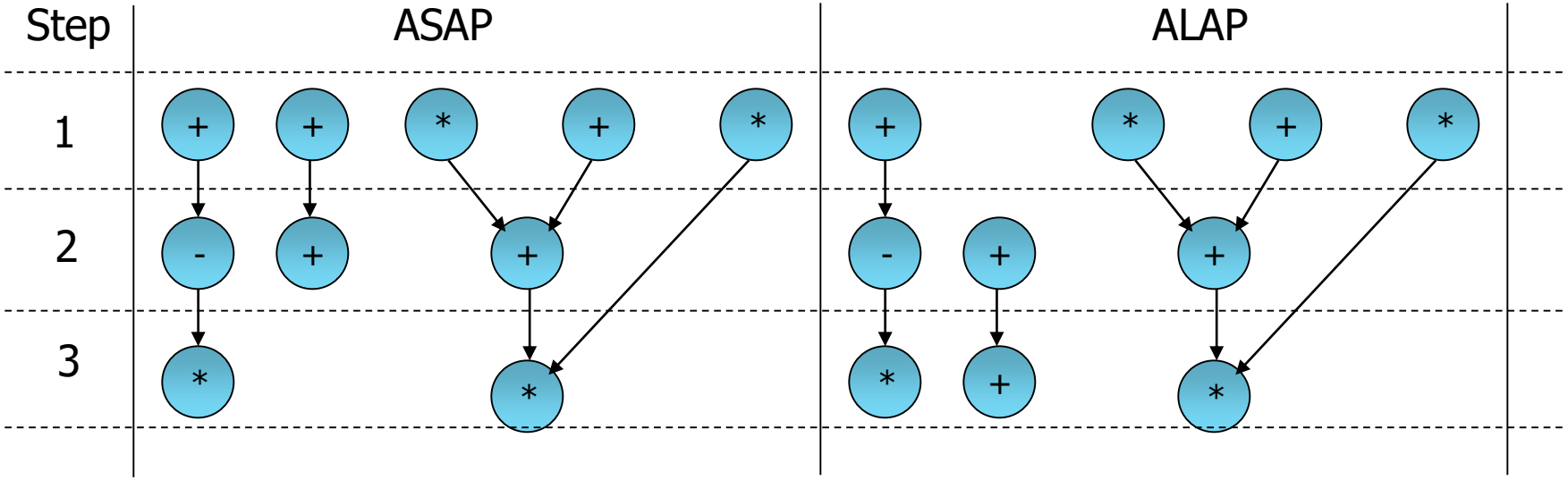
Função de Priorização (Caddy Priority Function)

- Probabilidade da operação ocorrer em um determinado step

$$P_O(s) = \begin{cases} 1/\ \Delta T_{(O)}\ ,\ \sigma_{ASAP(O)} < s < \sigma_{ALAP(O)} \\ 0,\ \text{Caso contrário} \end{cases}$$

Função de Priorização (Caddy Priority Function)

- Exemplo considerando um escalonamento sem restrição de recursos:



$\Delta T_{(1)} = 1 \Rightarrow P_1 = 1$

$\Delta T_{(2)} = 2 \Rightarrow P_2 = 0,5$

$\Delta T_{(3)} = 1 \Rightarrow P_3 = 1$

$\Delta T_{(4)} = 1 \Rightarrow P_4 = 1$

$\Delta T_{(5)} = 1 \Rightarrow P_5 = 1$

$\Delta T_{(6)} = 1 \Rightarrow P_6 = 1$

$\Delta T_{(7)} = 2 \Rightarrow P_7 = 0,5$

$\Delta T_{(8)} = 1 \Rightarrow P_8 = 1$

$\Delta T_{(9)} = 1 \Rightarrow P_9 = 1$

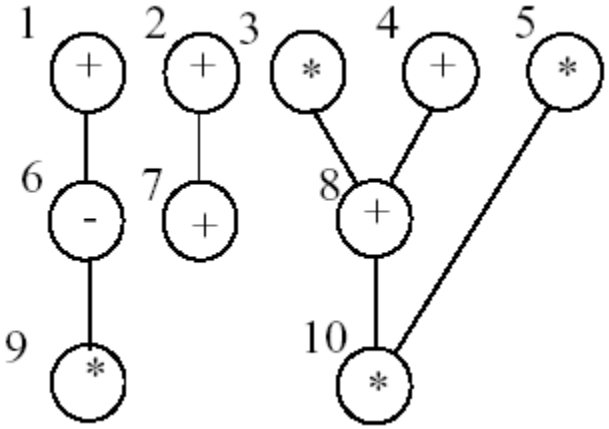
$\Delta T_{(10)} = 1 \Rightarrow P_{10} = 1$

Escalonamento com restrição de recursos

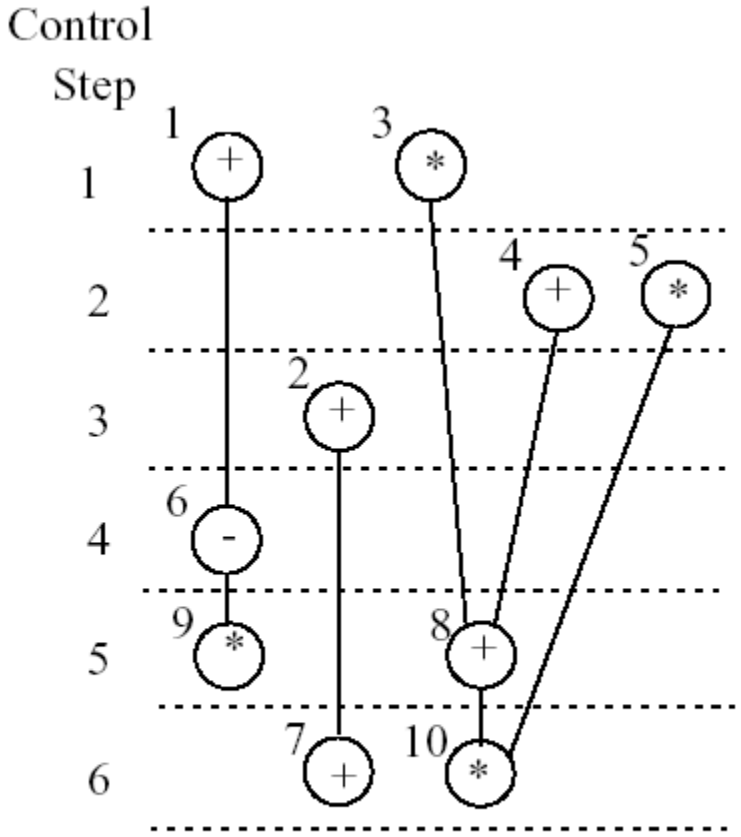
- Algoritmo de Hu [1] (List scheduling)
  - Utiliza um List scheduling (Lista de operações "prontas"): Para cada step de controle, as operações disponíveis para serem escalonadas são mantidas em uma lista;
  - Utiliza Função de priorização => (A operação pronta de maior prioridade é escolhida para no step de controle):
    - A duração do path partindo da operação até o fim do bloco;
    - Mobilidade: O nº de steps de controle partindo do mais recente ao último step de controle possível.
  - Operação "pronta" => Representa uma atividade não escalonada, cujas ações precedentes já estejam escalonadas e desta forma pode entrar no step de controle do escalonamento.

Escalonamento com restrição de recursos

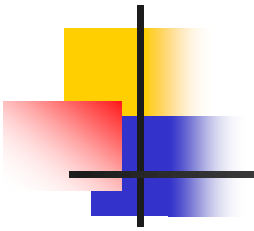
- Algoritmo de Hu [1] (List scheduling)
  - Cada operação na lista é escalonada uma a uma se os recursos necessários estão livres; Caso contrário, é deixado para o próximo step de controle



(a) Sorted DFG

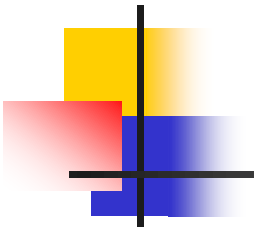


(b) List schedule



Escalonamento com restrição de Tempo

- Restrição na duração do escalonamento (ex. n_o de steps)
- Padrões de restrição de tempo
  - Aplicações em processamento digital de sinais
    - Restrições nas taxas de amostragem de entrada de sinal ou na saída de dados;
  - Aplicações no domínio do controle
    - Restrições de tempo entre operações que são distribuídas na descrição algorítmica (sobre o fluxo de dados ou controle)



Escalonamento com restrição de Tempo

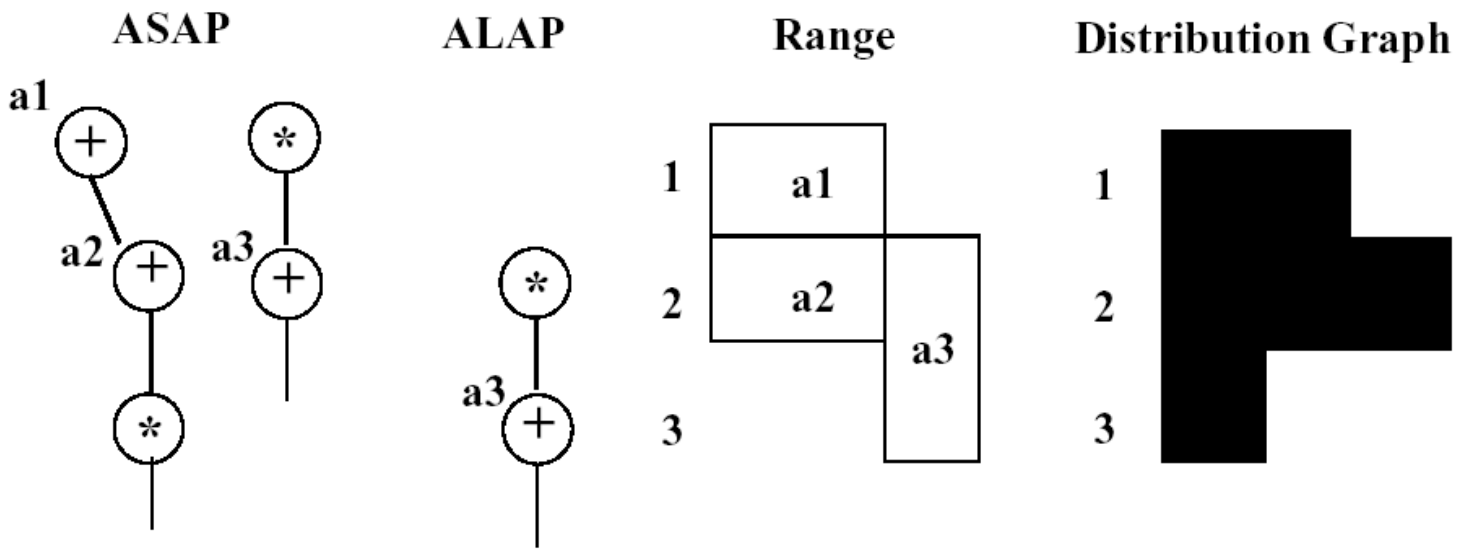
- Force-Direct scheduling
  - O objetivo desta técnica é distribuir uniformemente operações do mesmo tipo para todos os steps de controle disponíveis;
  - Esta distribuição uniforme resulta em um uso funcional mais apropriado da unidade, de forma a atender as restrições temporais

Force-Direct scheduling

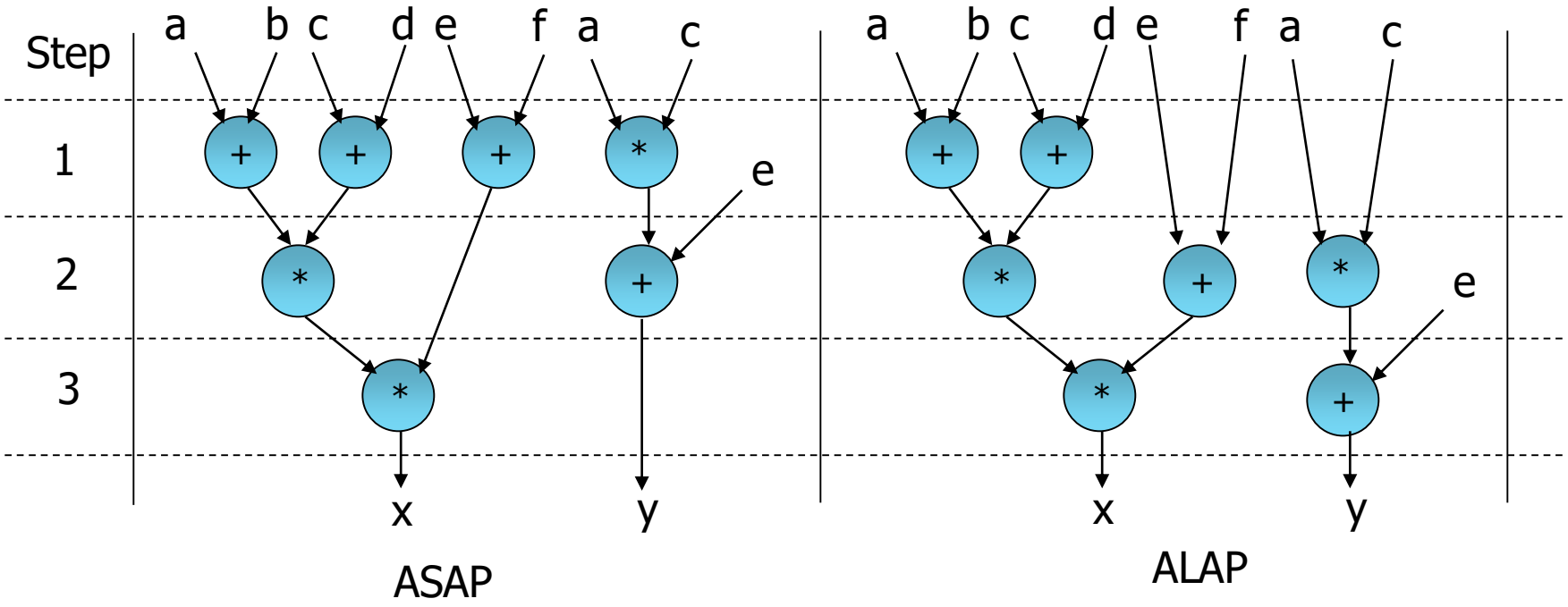
- Force-Direct scheduling
  - Escalonamentos ASAP e ALAP são calculados de forma a determinar os intervalos do escalonamento de cada operação;
  - Para cada tipo de operações (unidade funcional), um gráfico de distribuição de probabilidade é construído para denotar os possíveis steps de controle para cada operação.
  - O algoritmo tenta balancear o gráfico de distribuição calculando a força (influencia ?) de cada direcionamento de step de controle para operação e seleciona a menor força

Escalonamento com restrição de Tempo

An example:



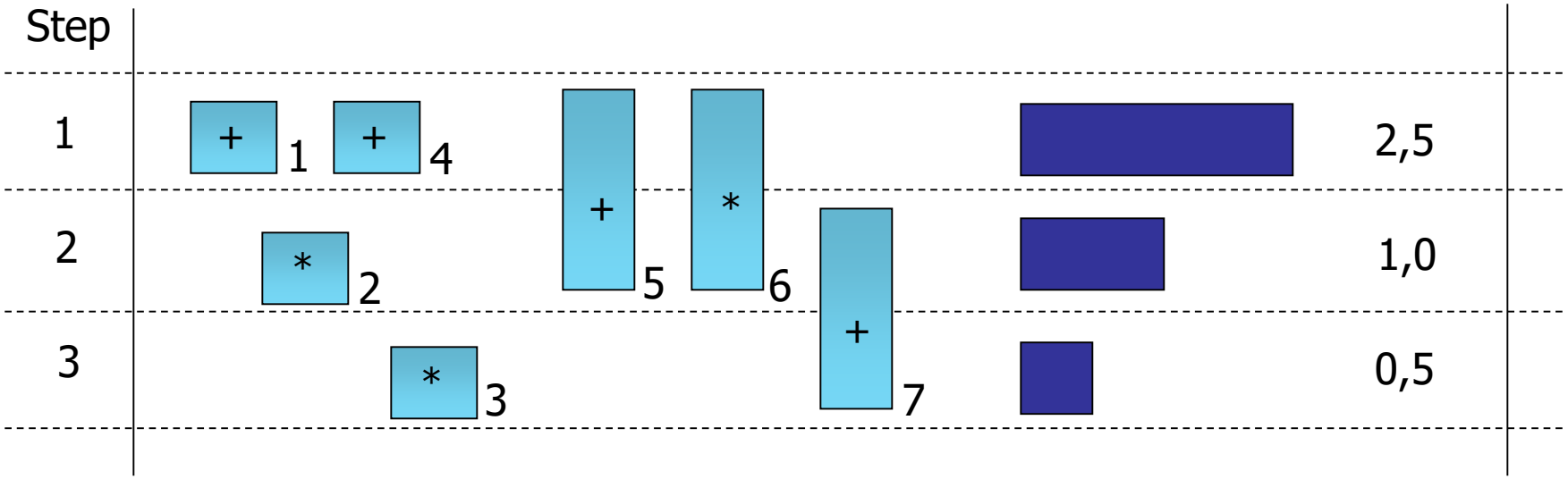
Escalonamento com restrição de Tempo



A probabilidade P_{ij} de uma operação o_i ser escalonada em um step particular de controle s_j tal que $E_i \leq j \leq L_i$ é dada por:

$$1/\Delta T_{(O)} = 1/\sigma_{ALAP(O)} - \sigma_{ASAP(O)} + 1$$

Gráfico de distribuição para Unidades de soma



Função de custo das unidades:

$$Fcost_{k,j} = C_k \sum_{i,\, j \in range(i)} P_{i,j}$$

$P_{i,j}$ = Prob. de ocorrência de O_i no intervalo de escalonamento;

C_k = Custo funcional da unidade

i = (Numero da) operação no step

j = Step em questão

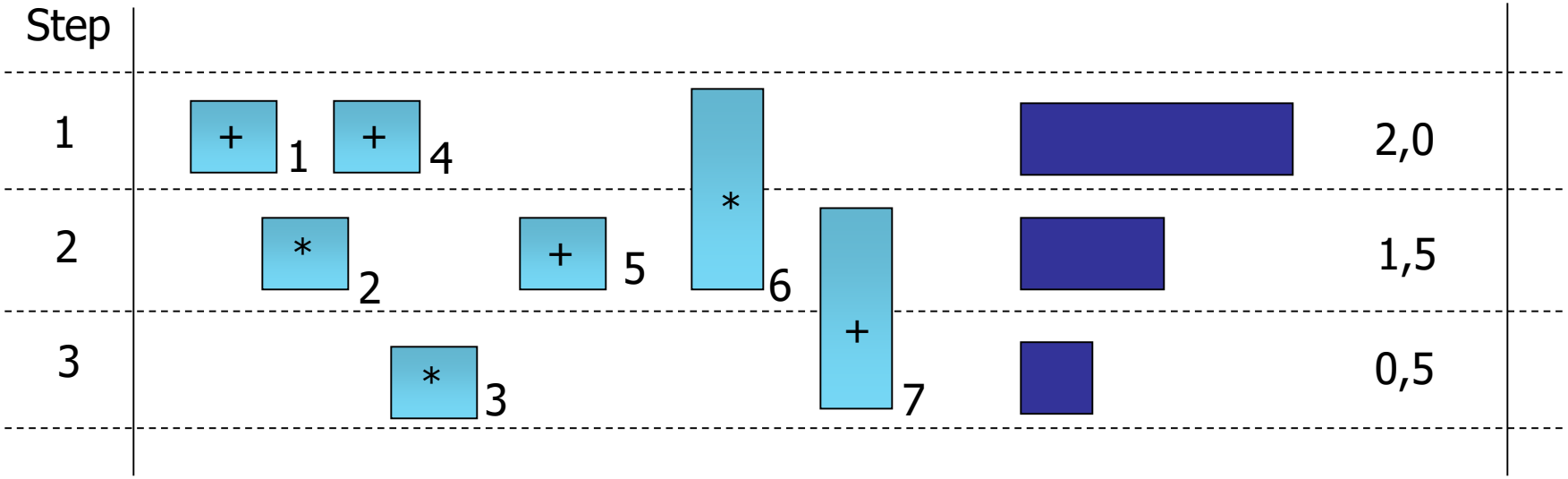
Range (i) = Intervalo de escalonamento de (i)

Custo soma step1 -> $Fcost_{add,1} = P_{1,1} + P_{4,1} + P_{5,1}$
$= 1+1+0.5 =$ **2.5**

Custo soma step2 -> $Fcost_{add,2} = P_{5,2} + P_{7,2}$
$= 0.5+0.5 =$ **1.0**

Custo soma step3 -> $Fcost_{add,3} = P_{7,3}$
= **0.5**

Gráfico de distribuição para Unidades de soma – Relocação da soma 5



Função de custo das unidades:

$Fcost_{k,j} = C_k \sum P_{i,j}$

$Fcost_{add,1} = 1+1 = 2.0$

$Fcost_{add,2} = 1.5+0.5 = 1.5$

$Fcost_{add,3} = 0.5$

Escalonamento com restrição de Tempo

- Force-Direct scheduling
  - A influência (força) de direcionamento de uma operação para um step particular de controle é a diferença entre o custo esperado da unidade funcional e a média do custo da unidade funcional esperada sobre o intervalo de escalonamento, e é dada por:

$$Dforce_{i,k,j} = Fcost_{k,j} - \sum_{s \in range(i)} Fcost_{k,s} / \Delta T_{(O)}$$

S $\in$ range das operações de i

EX.: i = O_5 tem range j=1 e j=2

Escalonamento com restrição de Tempo

- Exemplo: Considerando a operação i = O_5 teremos

$$\Delta T_{(O)} = L_i - E_i + 1 \Rightarrow \Delta T_{(O)} = 2 - 1 + 1 \Rightarrow \boxed{\Delta T_{(O)} = 2}$$

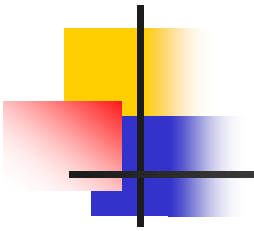
Temos: $\text{Dforce}_{i,k,j} = \text{Fcost}_{k,j} - \sum_{s \in \text{range}(i)} \text{Fcost}_{k,s} / \Delta T_{(O)}$

Para O_5 designado para o step (s) = 1, temos:

2,5 – (2.5+1)/2 = 0,75

Para O_5 designado para o step (s) = 2, temos:

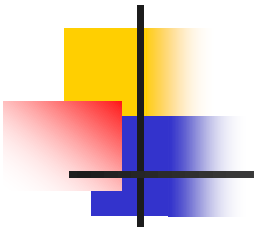
1 – (2,5+1)2 = - 0,75



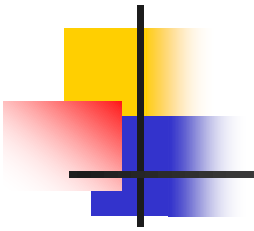
Conclusões:

- Problemas de escalonamento em sintese de sistemas digitais são problemas do tipo N-P completos;
- Técnicas considerando restrições de recursos e tempo podem ser utilizadas para minimizar o problema;
- Técnicas de escalonamento situam-se em dois maiores grupos: DFBS e CFBS

Escalonamento AFAP (As fast as possible):



- Objetivo básico: Minimizar o número de estados de controle sobre as restrições de execução
- Cálculo dos paths e eliminação dos feed-backs dos loops da descrição;
- Determinação dos pontos de loops em CFG acíclicos
- Cálculo das restrições para cada path (Ex. dependência de dados)



Pipeline path based scheduling

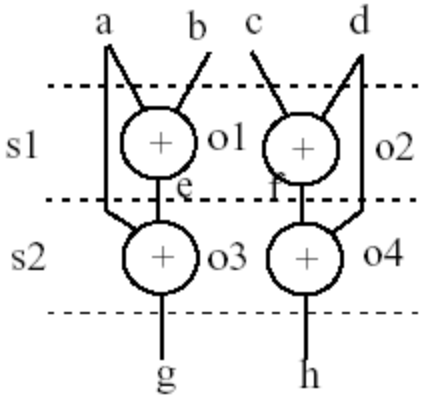
- Objetivo básico: Melhorar a performance do escalonamento para algorítmos AFAP com muitos loops
- Otimização do scheduling através de uma FSM melhorada;
- Fazer o pipeline de loops de forma a minimizar o número de ciclos de clock necessários para a execução de uma descrição

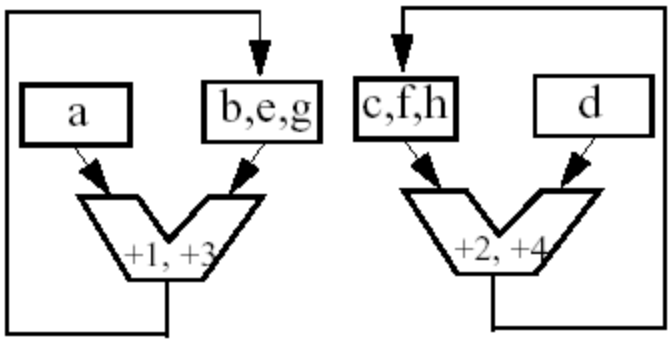
Allocation and Binding

- Allocation (Seleção de Unidade) – Determinação do número e tipo de recursos requeridos:
  - Unidades funcionais;
  - Tipos de elementos de armazenamento;
  - Barramentos;
  - ...

Allocation and Binding

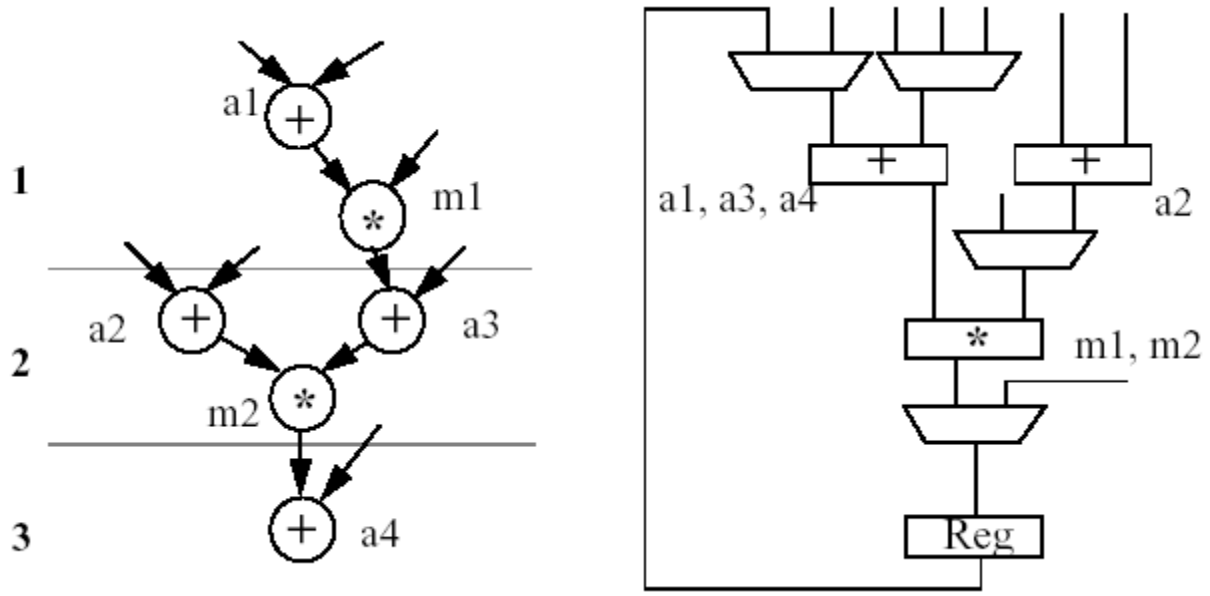
- Binding: Designação para os recursos (instancias)
  - Operações para instancias funcionais;
  - Valores a serem armazenados em instancias de elementos de armazenamento;
  - Transferências de dados para instancias de barramentos;





Técnicas para Allocation/Binding

- Construtiva: Começa com um datapath vazio e adciona unidades funcionais, armazenamento e interconexão a medida que for necessário
  - Greedy Algorithms – Realizam alocação de recursos em um step de controle por vez.



Técnicas para Allocation/Binding

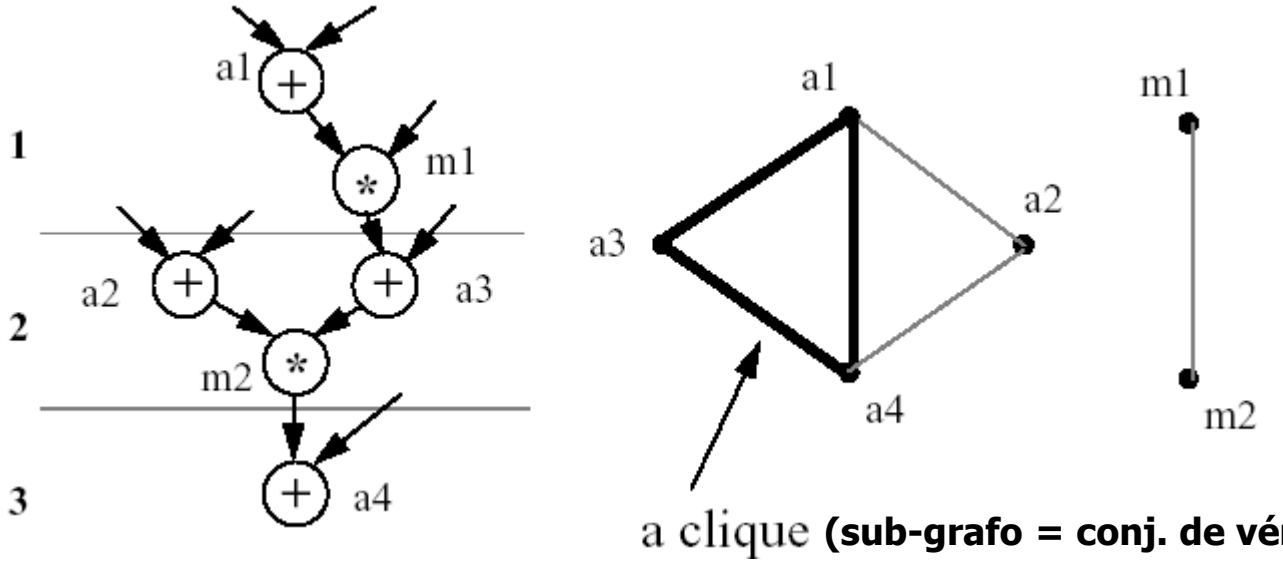
- Construtiva: (Cont.)
  - Rule Based – Usado para seleção de tipo e número de unidades de funções, especialmente previamente ao escalonamento.
- Graph-Theorical – Sub tarefas são mapeadas em problemas bem modelados na teoria dos grafos
  - Clique partitioning;
  - Left Edge algorithm
  - Graph Coloring
- Transformational Allocation

Clique Partitioning

- Seja G = (V,E) um grafo não direcional, com um conjunto de V de vértices e um conjunto E de bordas (Edges);
- Clique => Conjunto de vértices que formam um sub-grafo completo de G;
- Clique Partitioning =>Particionamento de um grafo em um número mínimo de cliques , tal que cada vértice pertença a exatamente um clique.

Clique Partitioning

- Formulação da alocação de unidade funcional como um problema Clique Partitioning:
  - Cada vértice representa uma operação;
  - Uma Borda (edge) conecta com dois vértices sse:
    - As duas operações são escalonadas em diferentes passos de controle e,
    - Existe uma unidade funcional que é capaz de carregar ambas as operações.



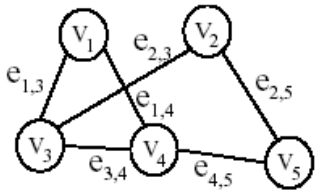
Clique Partitioning

- Formulação da alocação de armazenamento como um problema Clique Partitioning:
  - Cada valor necessário para ser armazenado é mapeado em um vértice
  - Dois vértices estão conectados sse, o tempo de vida dos dois valores não intersectam.
- OBS:
  - Clique partitioning é um problema NP-Completo
  - Heurísticas eficientes foram desenvolvidas => Algoritmo de tempo polinomial de Tseng

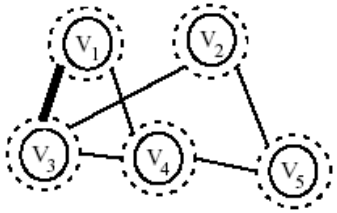
Algoritmo de Tseng

- Um super grafo é derivado do grafo original;
- Encontra-se dois super-nós conectados, tais que, eles tenham o número máximo de vizinhos em comum;
- Faz-se o merge dos dois nós e repete-se o processo, até não ser mais possível se fazer nenhum merge

Algoritmo de Tseng

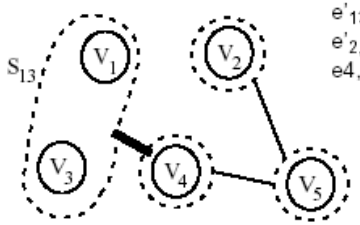


(a)



(b)

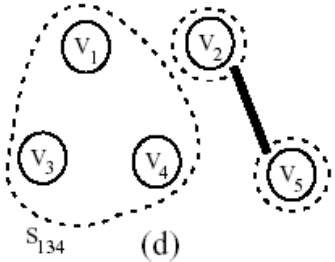
Edge	Common neighbors
$e'_{1,3}$	1
$e'_{1,4}$	1
$e'_{2,3}$	0
$e'_{2,5}$	0
$e'_{3,4}$	1
$e'_{4,5}$	0



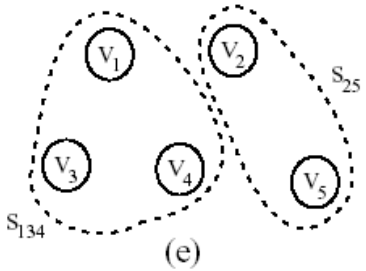
(c)

Edge	Common neighbors
$e'_{13,4}$	0
$e'_{2,5}$	0
e4,5	0

Edge	Common neighbors
$e'_{2,5}$	0



(d)



(e)

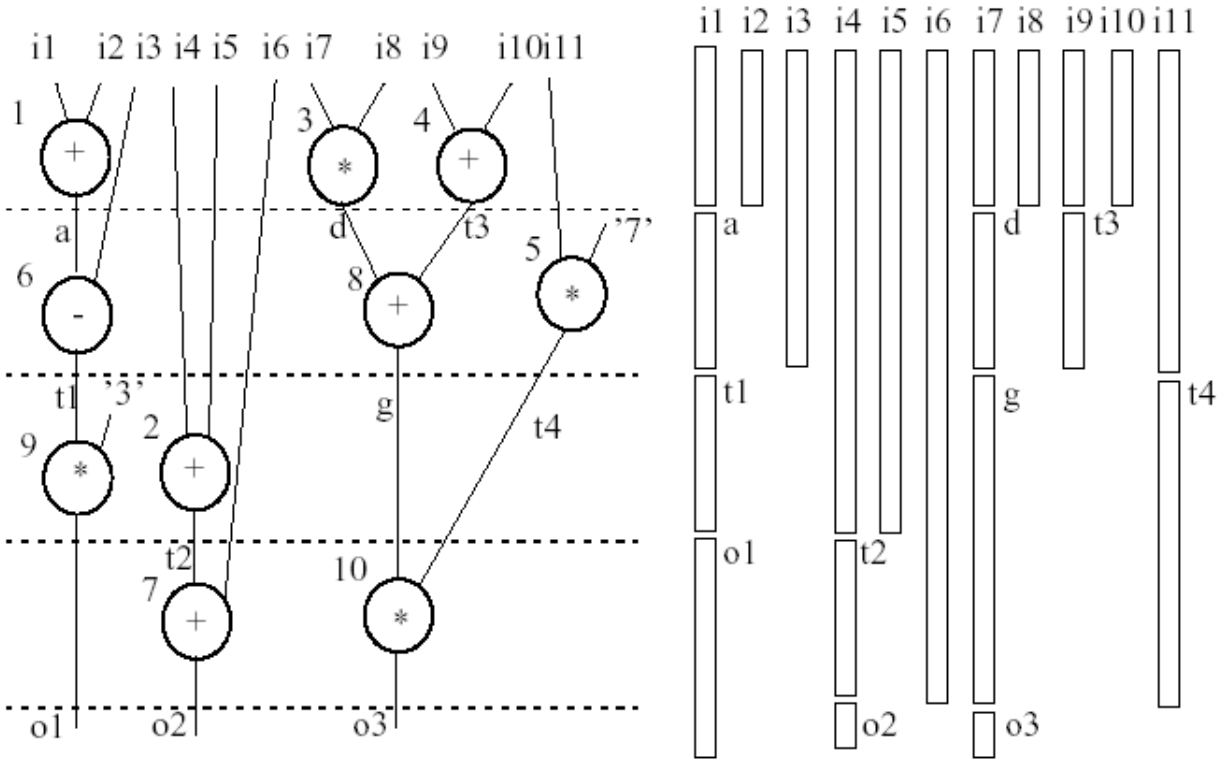
Cliques:

$S_{134} = (V_1, V_3, V_4)$

$S_{25} = (V_2, V_5)$

Left-Edge (LE) Algorithm

- O problema de alocação de registrador pode ser resolvido pelo algoritmo LE através do mapeamento do tempo de "nascimento" de um determinado valor

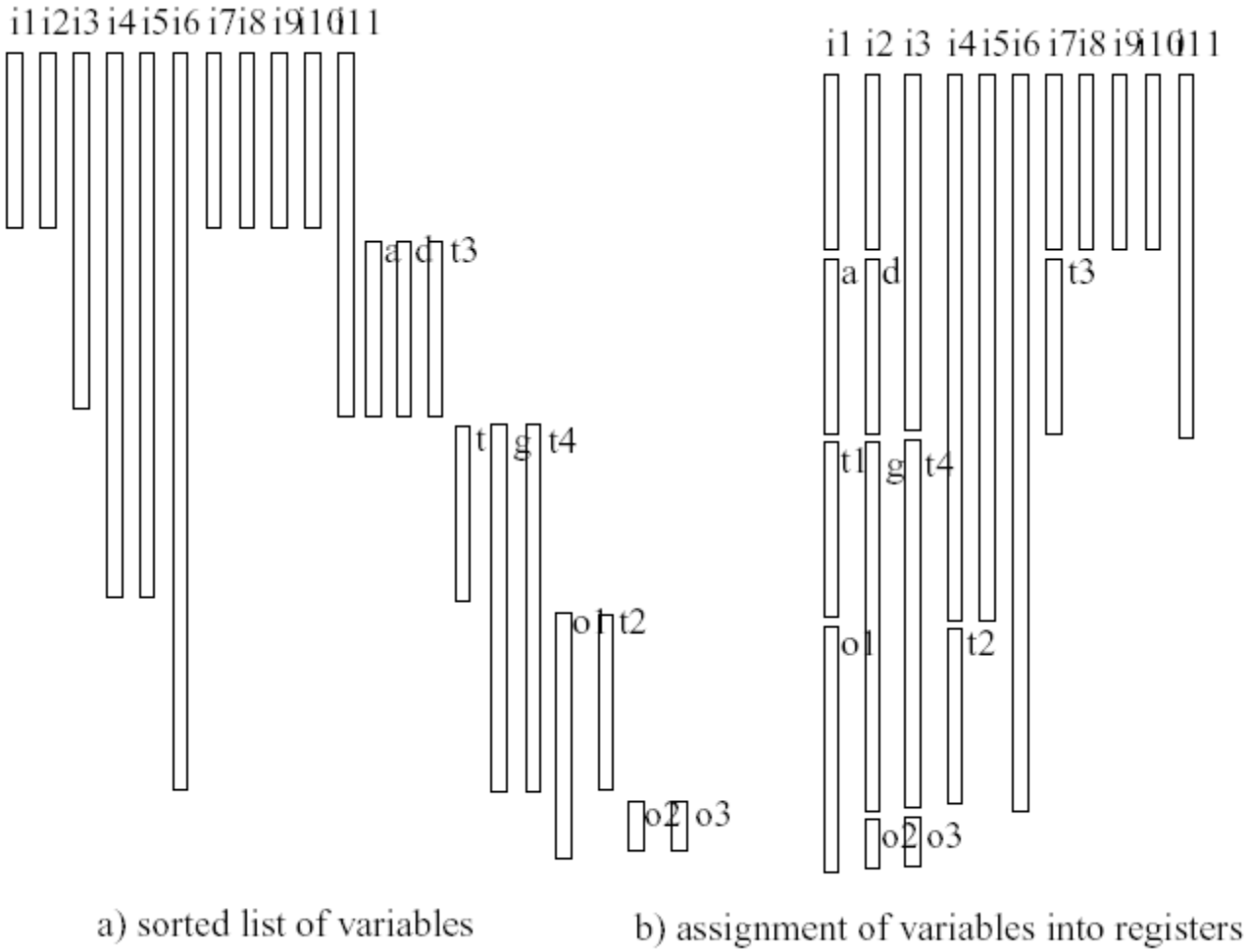


Variable life-times

Left-Edge (LE) Algorithm

- Funcionamento do algoritmo:
  - Os valores são ordenados em ordem crescente do seu momento de nascimento;
  - O primeiro valor é designado para o primeiro registrador;
  - A lista é então varrida para o próximo valor cujo tempo de "nascimento" é maior ou igual ao tempo de "morte" do valor anterior;
  - Este valor é designado para o registrador corrente;
  - A lista é varrida até nenhum valor poder partilhar o mesmo registrador. Um novo registrador é então introduzido, e repete-se o processo.

Left-Edge (LE) Algorithm



a) sorted list of variables

b) assignment of variables into registers

Referências Bibliográficas:

[1] T. C. Hu "Paralell sequencing and assembly line problems" (Operations research) pages 841-848.

[2] Petra M., U. Lauther, P. Duzzy - The synthesis Approach to Digital System Design

[3] Ahmed Jerraya,H. Ding, P. Kission, M. Rahmouni - Behavioral Synthesis and Component Reuse with VHDL