planeamento mais realista

outras abordagens ao planeamento

- grafos de planeamento
- planeamento por satisfação booleana

  ambos só com representação proposicional (sem variáveis)
- planeamento por dedução em LPO: cálculo de situações

  dificuldade em escalar para larga escala
- planeamento por satisfação de restrições
- planeamento por refinamento de planos de ordem parcial

  facilitam a verificação por humanos (ex: Mars rover)

tempo, agendamento e recursos

planos em muitos casos envolvem tempo

- quando começa a ação, quanto demora → agendas, horários

planos normalmente envolvem restrições

- capacidade de um avião é limitada
- tripulações específicas para cada tipo de avião

abordagem simples: planeia 1º, calendariza depois

- normalmente é simples fazer a calendarização de um plano

planeamento com tempo e recursos

ações têm duração

recursos têm duas variantes: consumíveis, reutilizáveis

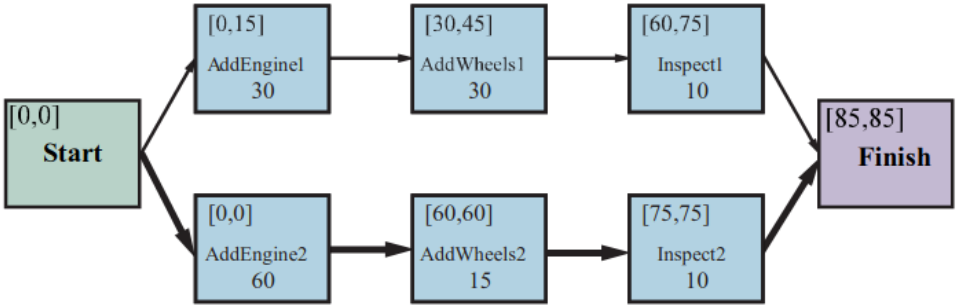
- pode haver ações com consumo negativo, ex: fabrico de…

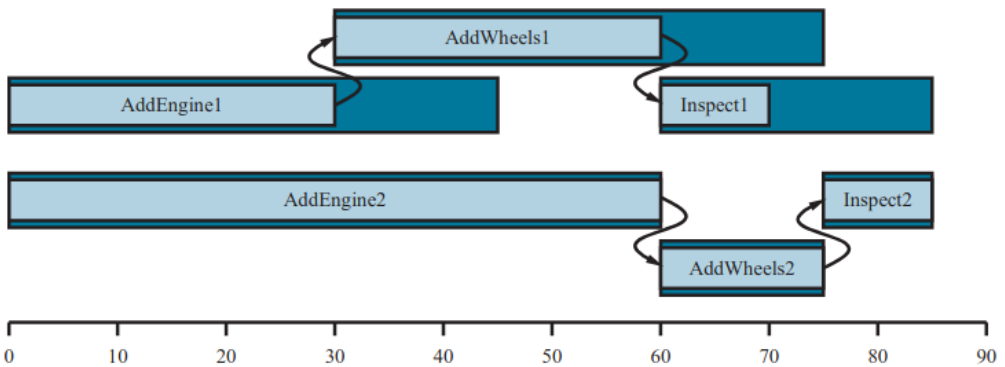
a representação de recursos pode usar **agregação**

- ex: *professores*(4) em vez de individualizar quando não é necessário – teste da quantidade evita 4! testes na procura

simplificação: custo = duração total do plano (*makespan*)

planos com o tempo
ex. uma fábrica





caminho crítico: o de maior duração
(setas carregadas)

[*ES*, *LS*]
ES: tempo de início mais cedo possível
LS: tempo de início mais tade possível

LS-*ES*: tolerância

Ciências ULisboa

cálculo da ordenação de ações

começa por definir *ES*(*Start*)=0

dada uma ação *B*

que tenha todas as ações precedentes com *ES* definidos,

calcula-se *ES*(*B*) como o máximo dos *ES* das ações que precedem *B*

repete-se o processo até todas as ações terem *ES* definido

---

os valores dos *LS* são calculados de modo semelhante a partir da ação *Finish*

formalmente

sendo A e B ações, $A \prec B$ significa "A precede B"

modelo de calendarização

$ES(Start) = 0$

$ES(B) = \max_{A \prec B} ES(A) + Duration(A)$

$LS(Finish) = ES(Finish)$

$LS(A) = \min_{B \succ A} LS(B) - Duration(A)$

subtrai a partir do início de B, para determinar início de A

complexidade do caminho crítico O(Nb)
N: nº de ações
b: fator de ramificação máx

planeamento hierárquico

hierarchical task networks (**HTN**) *planning*

facilita o planeamento a organização deste em macro-ações, que podem posteriormente ser expandidas em planos detalhados

pressupõe uma decomposição hierárquica do plano

- ver engª de software para desenvolvimento de aplicações complexas

ações de alto nível

distinguem-se das ações primitivas

- que estão bem definidas (atómicas)

as ações de alto nível (HLA) são uma composição de outras ações

- primitivas ou HLAs
- e têm uma ou mais expansões possíveis

expansão de HLA, ex.

expansão(*vai*(*casa*, *LIS*),

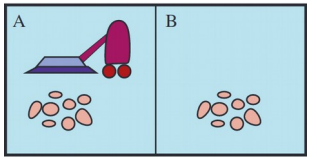
PASSOS: [*conduz*(*casa*, *parqueAeroportoLIS*),

shuttle(*parqueAeroportoLIS*, *LIS*)])

expansão(*vai*(*casa*, *LIS*),

PASSOS: [*táxi*(*casa*, *LIS*)])

o robô aspirador



mas em 2D

expansão(*navega*([*a*, *b*], [*x*, *y*]),

PRÉ-COND: $a = x \wedge b = y$

PASSOS: [ ])

recursiva

expansão(*navega*([*a*, *b*], [*x*, *y*]),

PRÉ-COND: *ligado*([*a*, *b*], [*a* − 1, *b*])

PASSOS: [*esquerda*, *navega*([*a* − 1, *b*], [*x*, *y*])])

expansão(*navega*([*a*, *b*], [*x*, *y*]),

PRÉ-COND: *ligado*([*a*, *b*], [*a* + 1, *b*])

PASSOS: [*direita*, *navega*([*a* + 1, *b*], [*x*, *y*])])

...

HLA: *navega*([1,3], [3,2])
tem 3 implementações

implementação: expansão apenas com ações primitivas

Ciências ULisboa

plano de alto nível e o objetivo

um plano de alto nível atinge o objetivo a partir de um estado se pelo menos uma das suas implementações atinge o objetivo a partir desse estado

semântica angélica

não é necessário que todas as implementações atinjam o objetivo, porque o planeador pode escolher uma que atinge

começa com uma única HLA, *act*, e resolve recursivamente

para cada ação primitiva a_i expande *act* com os passos [a_i, *act*]

HTN como procura em largura

```
function HIERARCHICAL-SEARCH(problem, hierarchy) returns a solution or failure
    frontier ← a FIFO queue with [Act] as the only element
    while true do
        if IS-EMPTY(frontier) then return failure
        plan ← POP(frontier)          // chooses the shallowest plan in frontier
        hla ← the first HLA in plan, or null if none
        prefix,suffix ← the action subsequences before and after hla in plan
        outcome ← RESULT(problem.INITIAL, prefix)
        if hla is null then          // so plan is primitive and outcome is its result
            if problem.IS-GOAL(outcome) then return plan
        else for each sequence in REFINEMENTS(hla, outcome, hierarchy) do
            add APPEND(prefix, sequence, suffix) to frontier
```

complexidade

um planeador com procura adiante com *d* ações primitivas e *b* ações possíveis em cada estado tem custo $O(b^d)$

um planeador HTN com ações não primitivas com *r* expansões em *k* ações no nível inferior tem custo $O(r^{d/k})$

ideal:

pequeno *r* – pequeno nº de expansões

grande *k* – cada uma com uma longa sequência de ações

não é fácil de conseguir...

procura com semântica angélica

uma descrição otimista $\textsc{Reach}^{+}(s, h)$ de uma HLA, h, pode sobrestimar a possibilidade de chegar ao objetivo

uma descrição pessimista $\textsc{Reach}^{-}(s, h)$ de uma HLA, h, pode subestimar a possibilidade de chegar ao objetivo

$$\textsc{Reach}^{-}(s, h) \subseteq \textsc{Reach}(s, h) \subseteq \textsc{Reach}^{+}(s, h)$$

expansão (refinamento) do plano

- se o conjunto pessimista de estados atingíveis interseta o objetivo ⇒ o plano tem solução

- se o conjunto otimista de estados atingíveis interseta o objetivo

  e

  o conjunto pessimista de estados atingíveis não interseta

  ⇒ tem de se refinar (expandir) o plano

procura angélica

function ANGELIC-SEARCH($problem, hierarchy, initialPlan$) **returns** solution or *fail*

```
frontier ← a FIFO queue with initialPlan as the only element
while true do
    if EMPTY?(frontier) then return fail
    plan ← POP(frontier)          // chooses the shallowest node in frontier
    if REACH+(problem.INITIAL, plan) intersects problem.GOAL then
        if plan is primitive then return plan          // REACH+ is exact for primitive plans
        guaranteed ← REACH−(problem.INITIAL, plan) ∩ problem.GOAL
        if guaranteed≠{ } and MAKING-PROGRESS(plan, initialPlan) then
            finalState ← any element of guaranteed
            return DECOMPOSE(hierarchy, problem.INITIAL, plan, finalState)
        hla ← some HLA in plan
        prefix,suffix ← the action subsequences before and after hla in plan
        outcome ← RESULT(problem.INITIAL, prefix)
        for each sequence in REFINEMENTS(hla, outcome, hierarchy) do
            frontier ← Insert(APPEND(prefix, sequence, suffix), frontier)
```

function DECOMPOSE($hierarchy, s_0, plan, s_f$) **returns** a solution

```
solution ← an empty plan
while plan is not empty do
    action ← REMOVE-LAST(plan)
    s_i ← a state in REACH−(s_0, plan) such that s_f ∈ REACH−(s_i, action)
    problem ← a problem with INITIAL = s_i and GOAL = s_f
    solution ← APPEND(ANGELIC-SEARCH(problem, hierarchy, action), solution)
    s_f ← s_i
return solution
```

ambientes mais complexos

- planeamento de contingência

  para ambientes parcialmente observáveis

  para ambientes não-deterministas

  estados de crença (*belief*)

- planeamento *online* e replaneamento

  para ambientes desconhecidos

  planear a perceção