minimum(Value, List)

maximum(Value, List)

cumulative(Tasks, Options)

limit(L)

precedences(List) Ti-Tj #= Diff

List of task(Oi, Di, Ei, Hi, Ti)

cumulatives(Tasks, Machines, Options)

bound(lower/upper)

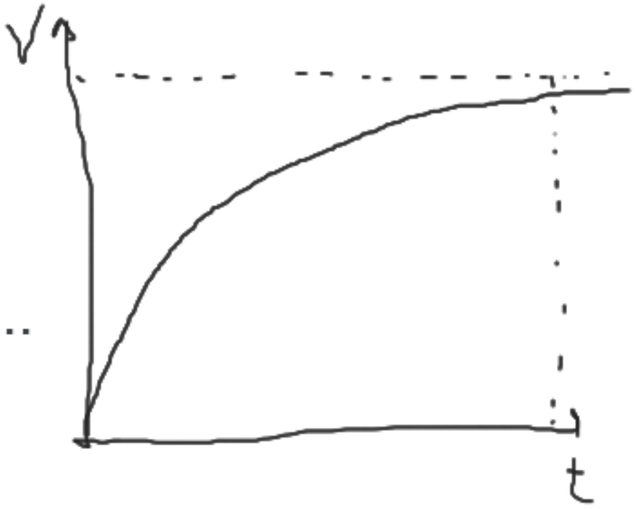
...

List of machine(Mi, Li)

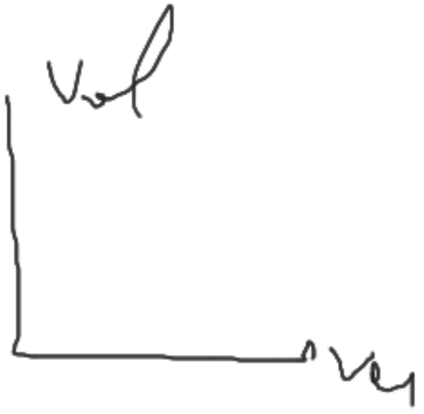
List of task(Oi, Di, Ei, Hi, Mi)

labeling(Options, Vars)

Selecção de próxima variável
leftmost, min, max, ff, ffc, anti_first_fail, occurrence, max_regret, ...



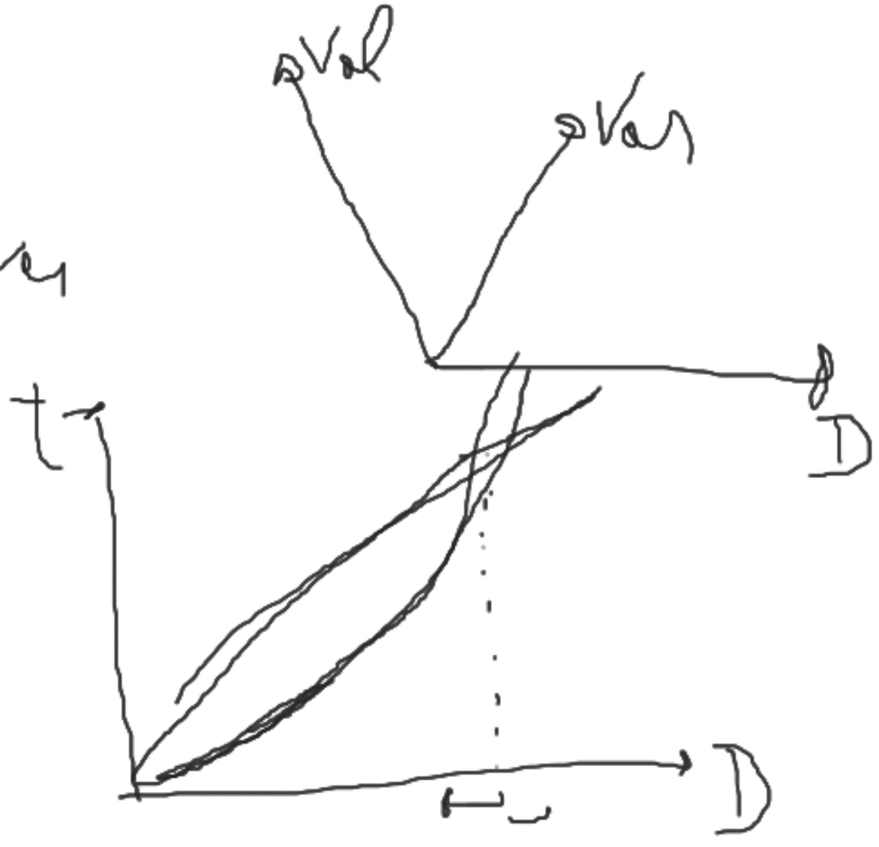
Selecção do valor a atribuir
step, enum, bisect, median, middle, ...



Ordenação
up, down

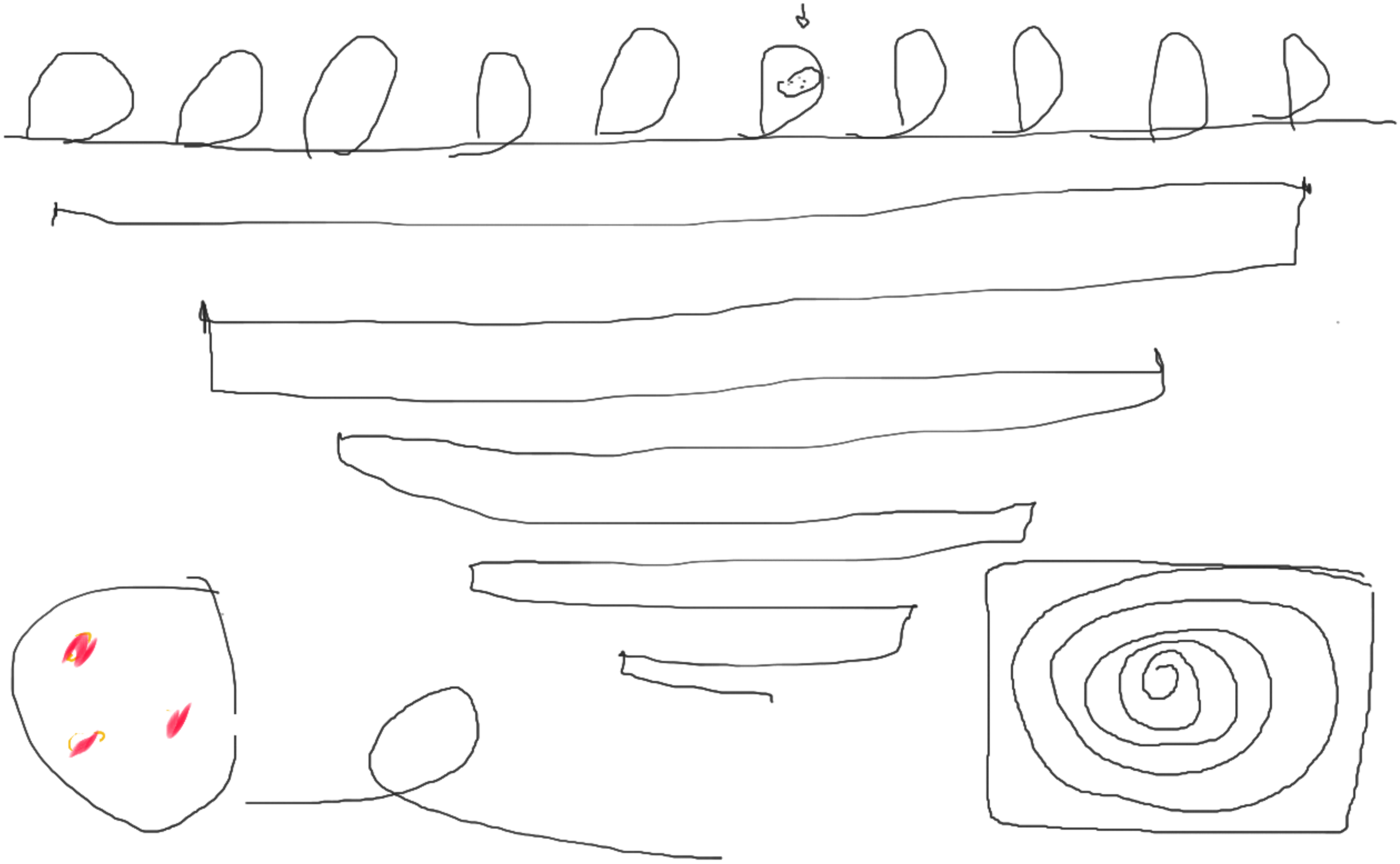
O que encontrar
satisfy, minimize(Var), maximize(Var), best, all

Timeout
time_out(Time, Flag)
(ms) time_out / success / optimality



Pretendem-se escalonar 7 tarefas com as características abaixo indicadas, devendo completar a execução o mais cedo possível, e nunca excedendo a capacidade máxima do recurso: 13.

Tarefa	Duração	Recursos
T1	16	2
T2	6	9
T3	13	3
T4	7	7
T5	5	10
T6	18	1
T7	4	11



```
houses(N):-
    length(Houses, N),    %ordem de visita às casas
    domain(Houses, 1, N),
    element(N, Houses, 6),
    all_distinct(Houses),
    times(Houses, Time),
    labeling( [maximize(Time) ], Houses), write(Houses).

times( [ _ ], 0).
times( [A, B | T ], Time):-
    Time #= Time2 + abs(B-A),
    times( [B | T], Time2).
```