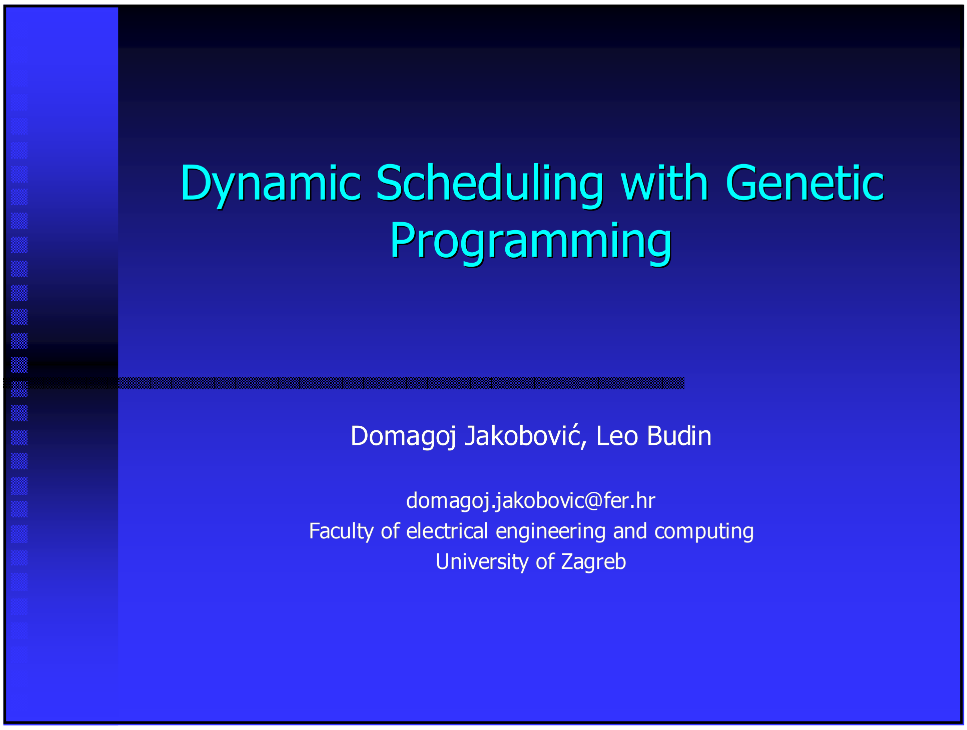
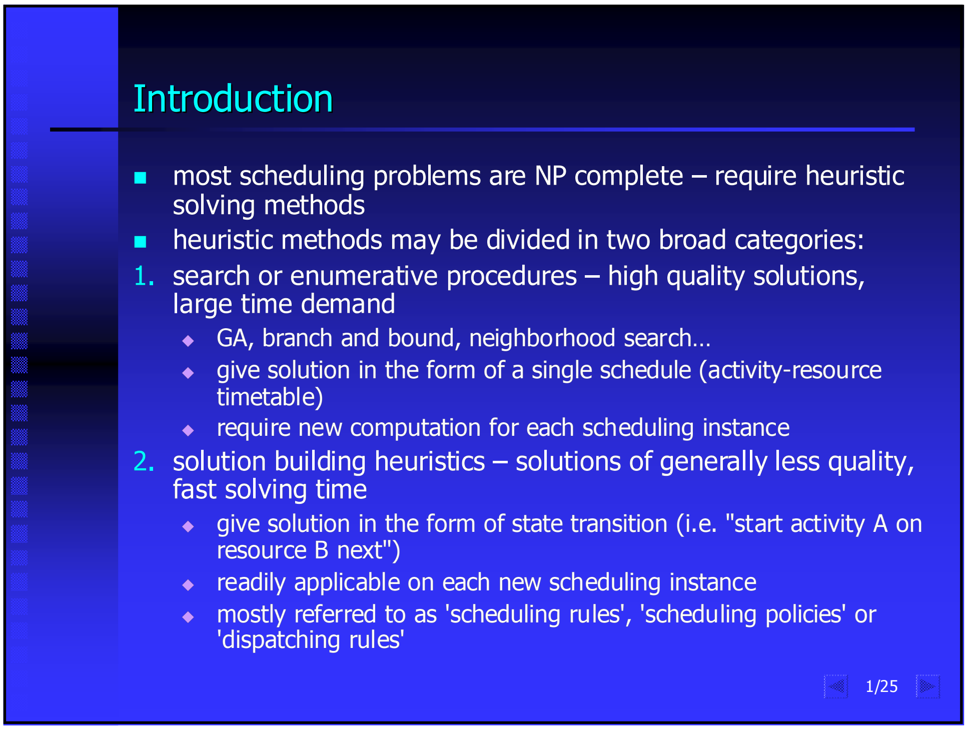


Dynamic Scheduling with Genetic Programming

Domagoj Jaković, Leo Budin

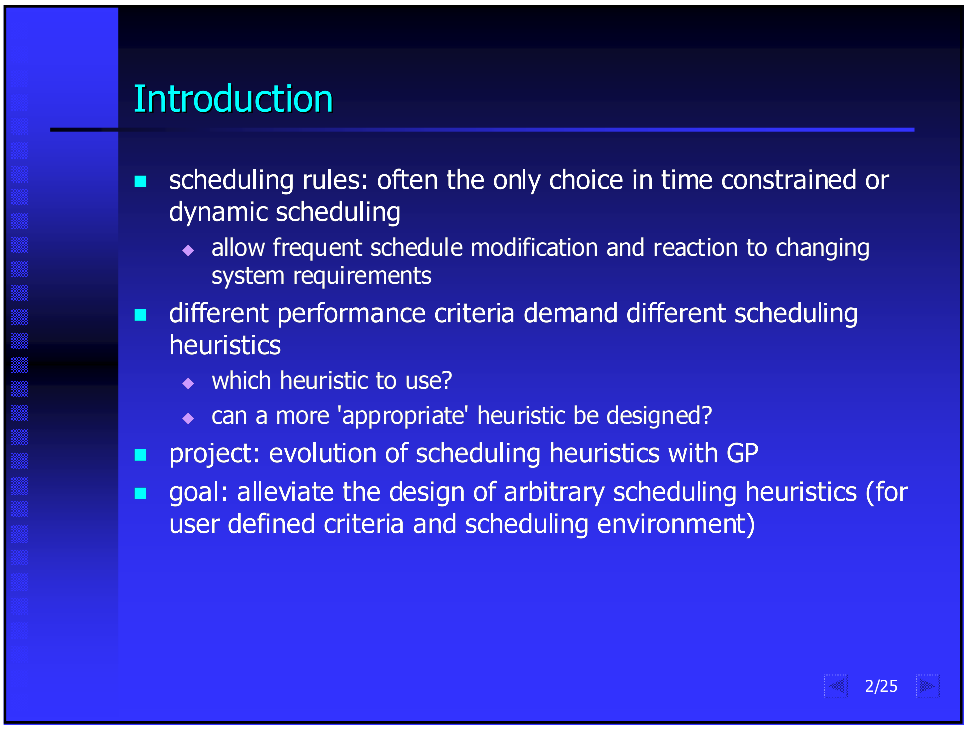
domagoj.jakobovic@fer.hr
Faculty of electrical engineering and computing
University of Zagreb



Introduction

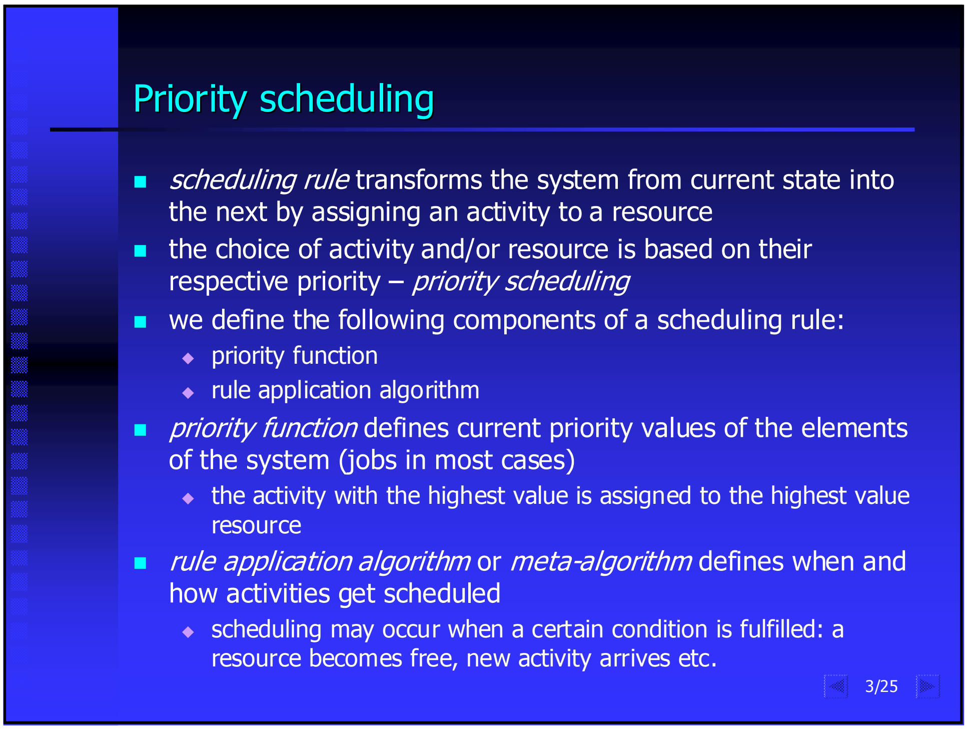
- most scheduling problems are NP complete – require heuristic solving methods
- heuristic methods may be divided in two broad categories:

1. search or enumerative procedures – high quality solutions, large time demand
   - GA, branch and bound, neighborhood search…
   - give solution in the form of a single schedule (activity-resource timetable)
   - require new computation for each scheduling instance
2. solution building heuristics – solutions of generally less quality, fast solving time
   - give solution in the form of state transition (i.e. "start activity A on resource B next")
   - readily applicable on each new scheduling instance
   - mostly referred to as 'scheduling rules', 'scheduling policies' or 'dispatching rules'



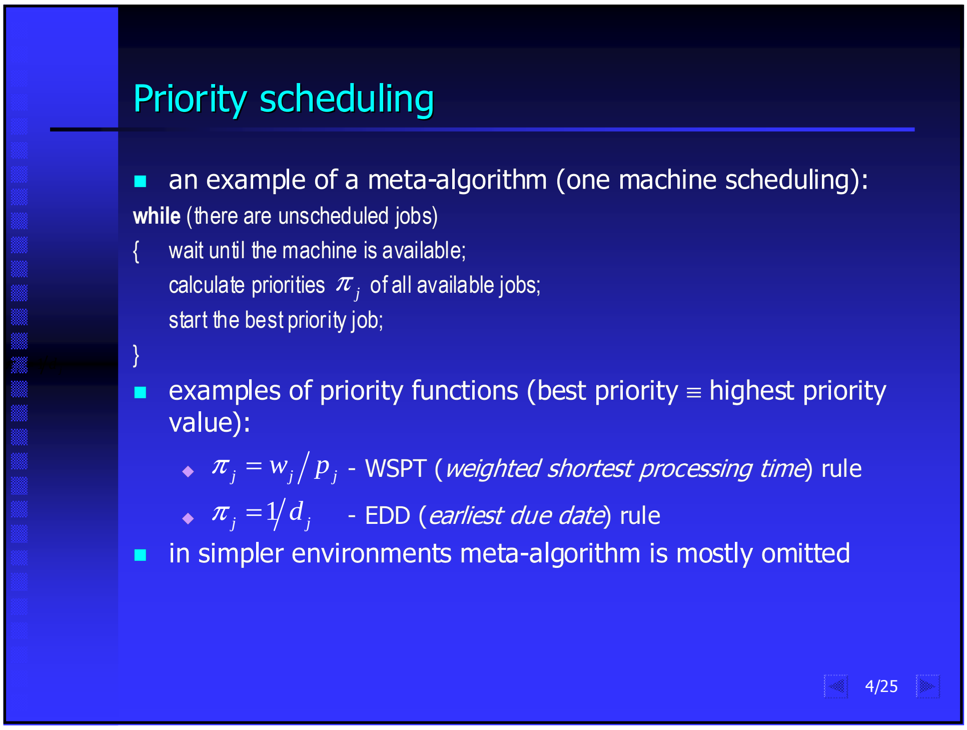
Introduction

- scheduling rules: often the only choice in time constrained or dynamic scheduling
  - allow frequent schedule modification and reaction to changing system requirements
- different performance criteria demand different scheduling heuristics
  - which heuristic to use?
  - can a more 'appropriate' heuristic be designed?
- project: evolution of scheduling heuristics with GP
- goal: alleviate the design of arbitrary scheduling heuristics (for user defined criteria and scheduling environment)



Priority scheduling

- *scheduling rule* transforms the system from current state into the next by assigning an activity to a resource
- the choice of activity and/or resource is based on their respective priority – *priority scheduling*
- we define the following components of a scheduling rule:
  - priority function
  - rule application algorithm
- *priority function* defines current priority values of the elements of the system (jobs in most cases)
  - the activity with the highest value is assigned to the highest value resource
- *rule application algorithm* or *meta-algorithm* defines when and how activities get scheduled
  - scheduling may occur when a certain condition is fulfilled: a resource becomes free, new activity arrives etc.

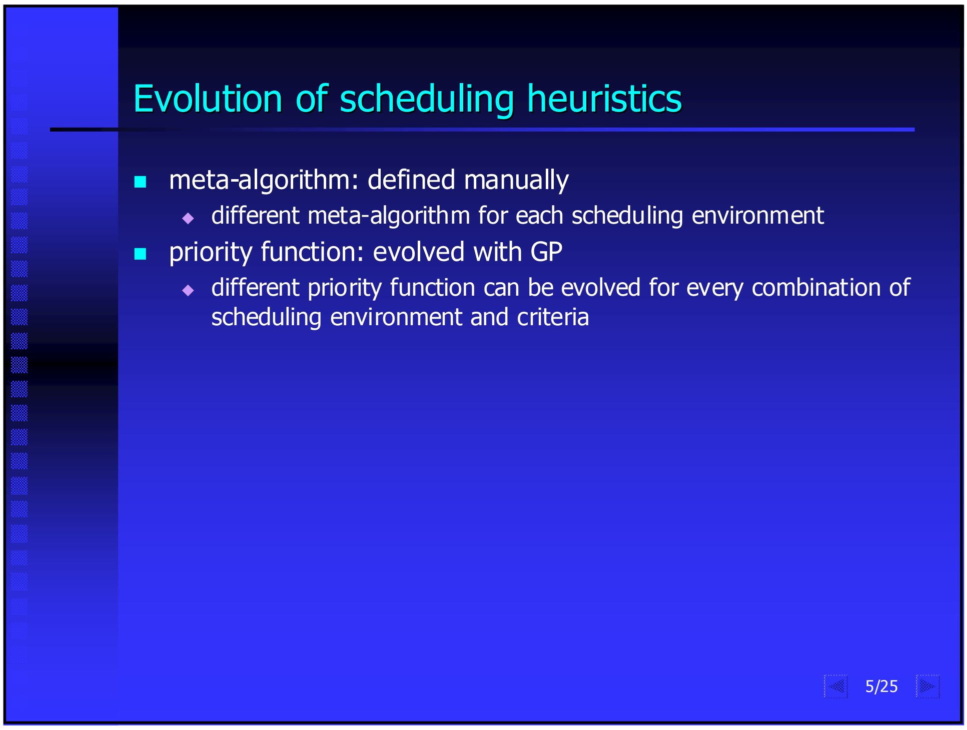


Priority scheduling

- an example of a meta-algorithm (one machine scheduling):

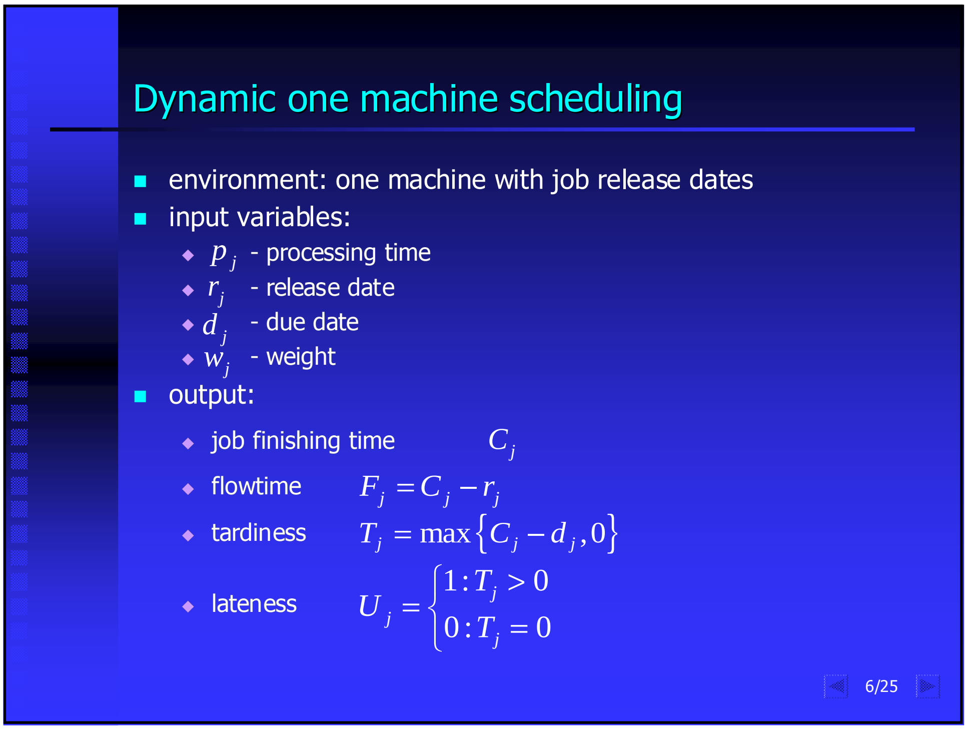
```
while (there are unscheduled jobs)
{    wait until the machine is available;
     calculate priorities $\pi_j$ of all available jobs;
     start the best priority job;
}
```

- examples of priority functions (best priority ≡ highest priority value):
  - $\pi_j = w_j / p_j$ - WSPT (*weighted shortest processing time*) rule
  - $\pi_j = 1/d_j$ - EDD (*earliest due date*) rule
- in simpler environments meta-algorithm is mostly omitted



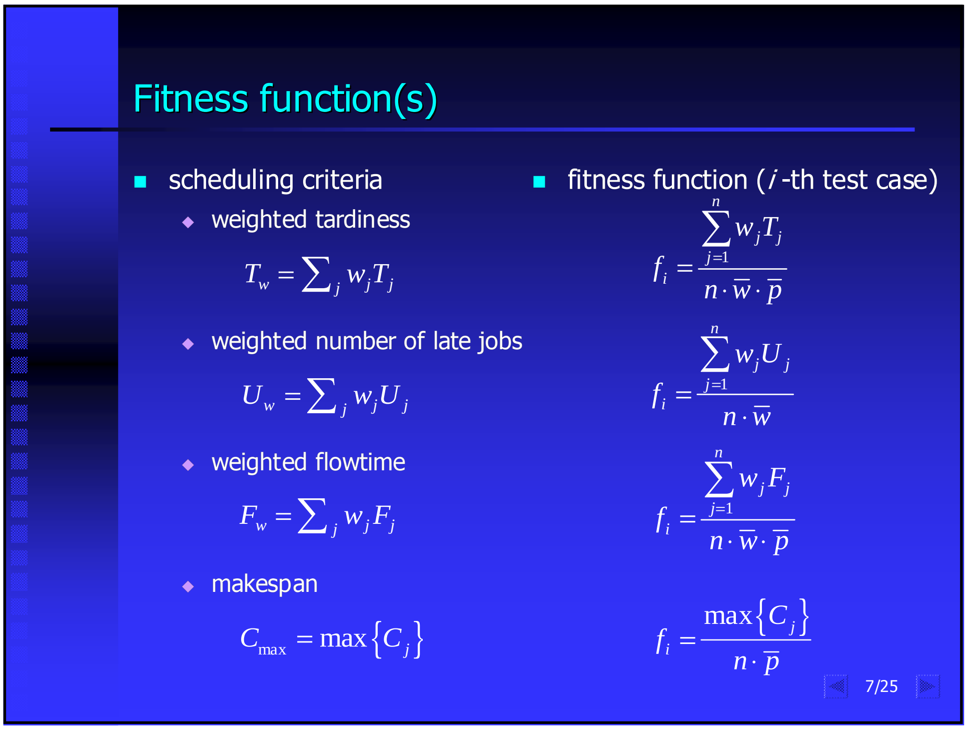
Evolution of scheduling heuristics

- meta-algorithm: defined manually
  - different meta-algorithm for each scheduling environment
- priority function: evolved with GP
  - different priority function can be evolved for every combination of scheduling environment and criteria



Dynamic one machine scheduling

- environment: one machine with job release dates
- input variables:
  - p_j - processing time
  - r_j - release date
  - d_j - due date
  - w_j - weight
- output:
  - job finishing time C_j
  - flowtime $F_j = C_j - r_j$
  - tardiness $T_j = \max\{C_j - d_j, 0\}$
  - lateness $U_j = \begin{cases} 1 : T_j > 0 \\ 0 : T_j = 0 \end{cases}$



Fitness function(s)

- scheduling criteria
  - weighted tardiness

    $T_w = \sum_j w_j T_j$
  - weighted number of late jobs

    $U_w = \sum_j w_j U_j$
  - weighted flowtime

    $F_w = \sum_j w_j F_j$
  - makespan

    $C_{\max} = \max\{C_j\}$

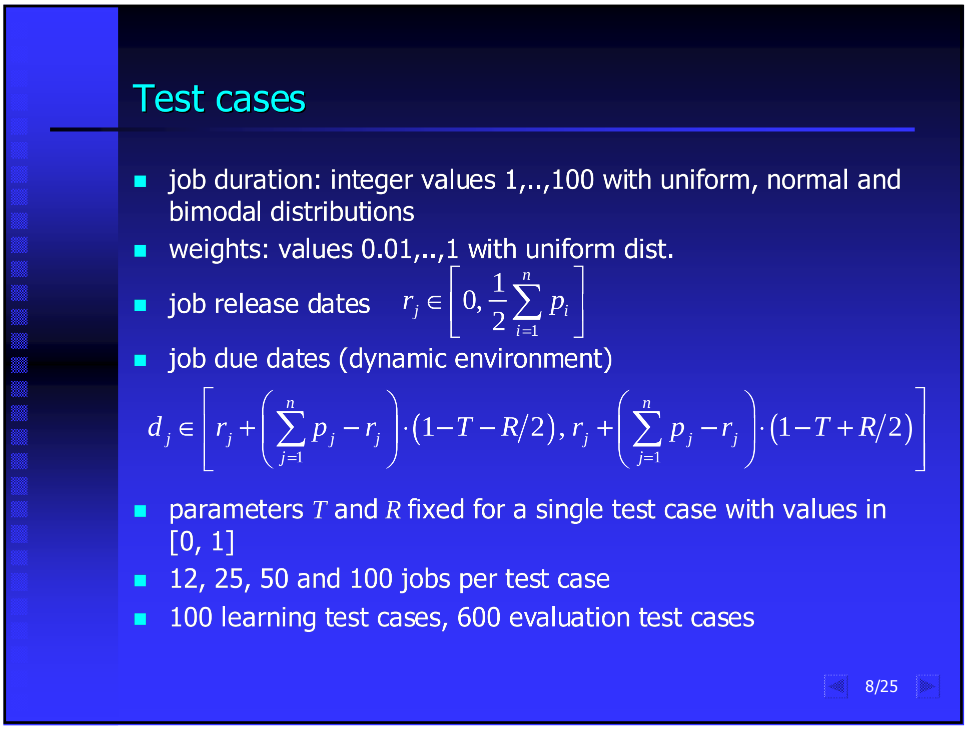
- fitness function (i-th test case)

  $$f_i = \frac{\sum_{j=1}^{n} w_j T_j}{n \cdot \overline{w} \cdot \overline{p}}$$

  $$f_i = \frac{\sum_{j=1}^{n} w_j U_j}{n \cdot \overline{w}}$$

  $$f_i = \frac{\sum_{j=1}^{n} w_j F_j}{n \cdot \overline{w} \cdot \overline{p}}$$

  $$f_i = \frac{\max\{C_j\}}{n \cdot \overline{p}}$$

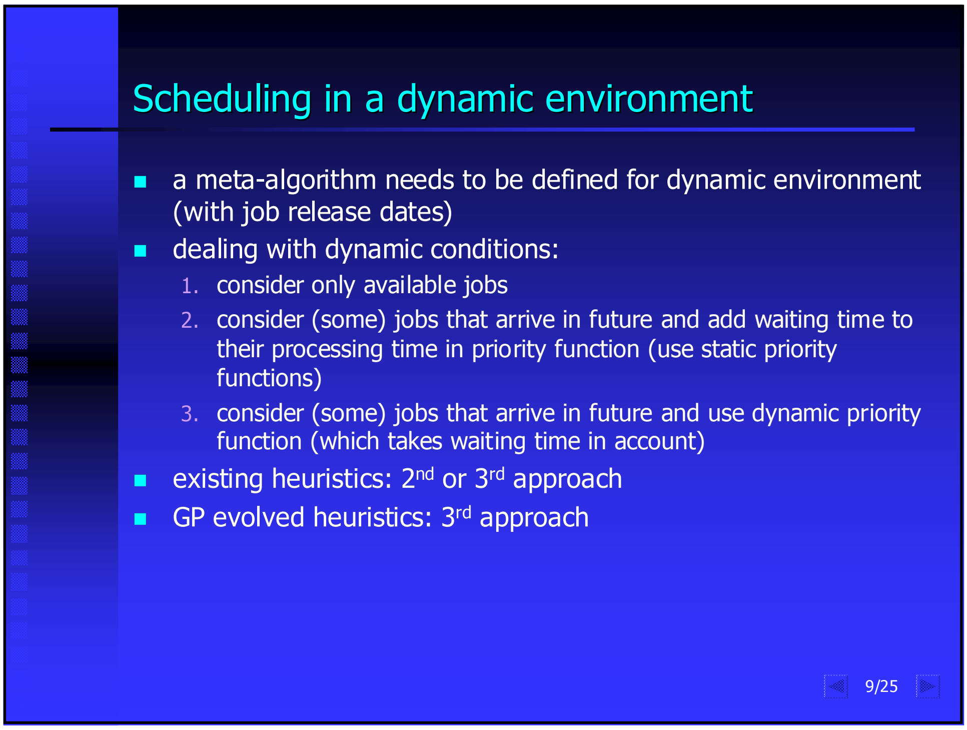


Test cases

- job duration: integer values 1,..,100 with uniform, normal and bimodal distributions
- weights: values 0.01,..,1 with uniform dist.
- job release dates $r_j \in \left[0, \frac{1}{2}\sum_{i=1}^{n} p_i\right]$
- job due dates (dynamic environment)

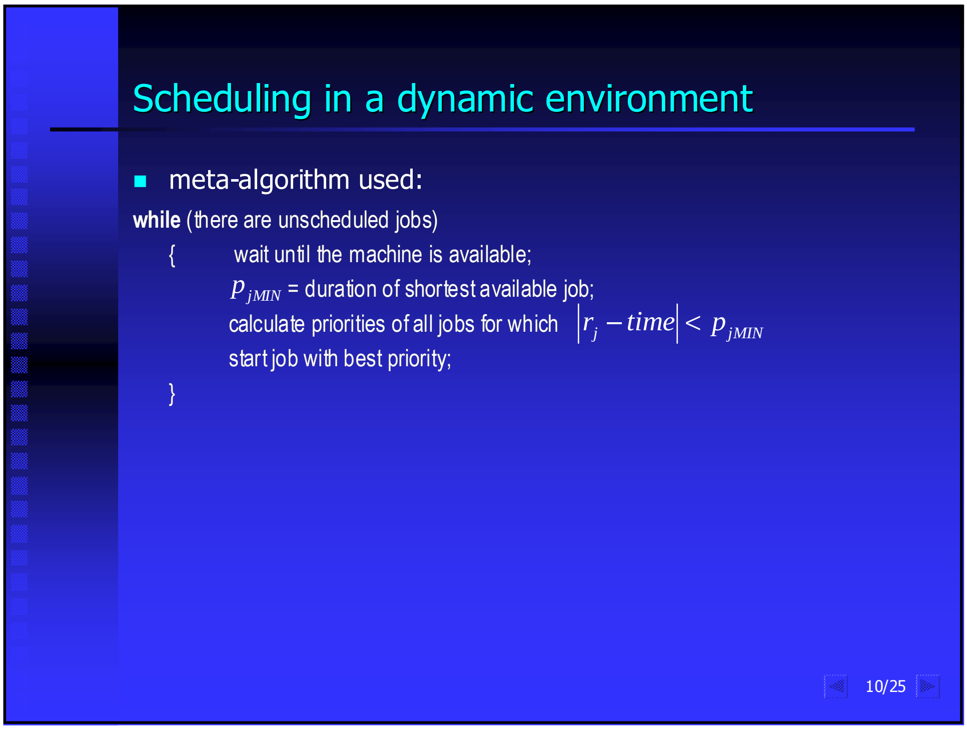
$$d_j \in \left[r_j + \left(\sum_{j=1}^{n} p_j - r_j\right)\cdot(1 - T - R/2),\, r_j + \left(\sum_{j=1}^{n} p_j - r_j\right)\cdot(1 - T + R/2)\right]$$

- parameters T and R fixed for a single test case with values in [0, 1]
- 12, 25, 50 and 100 jobs per test case
- 100 learning test cases, 600 evaluation test cases



Scheduling in a dynamic environment

- a meta-algorithm needs to be defined for dynamic environment (with job release dates)
- dealing with dynamic conditions:
  1. consider only available jobs
  2. consider (some) jobs that arrive in future and add waiting time to their processing time in priority function (use static priority functions)
  3. consider (some) jobs that arrive in future and use dynamic priority function (which takes waiting time in account)
- existing heuristics: 2nd or 3rd approach
- GP evolved heuristics: 3rd approach

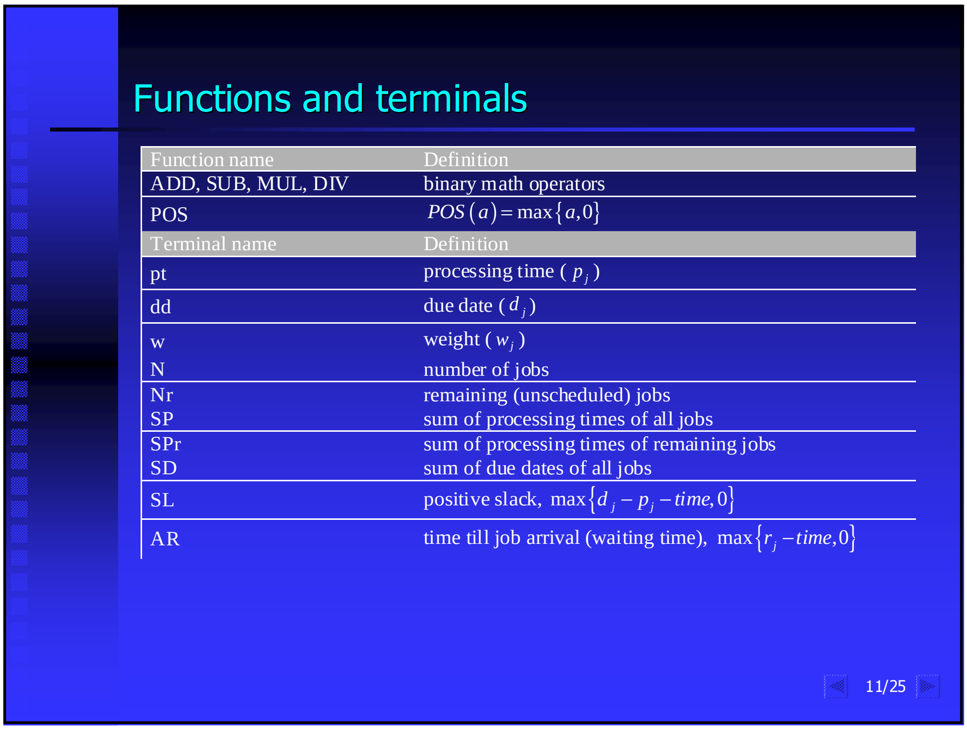


Scheduling in a dynamic environment

- meta-algorithm used:

```
while (there are unscheduled jobs)
    {       wait until the machine is available;
            $p_{jMIN}$ = duration of shortest available job;
            calculate priorities of all jobs for which $|r_j - time| < p_{jMIN}$
            start job with best priority;
    }
```

10/25



Functions and terminals

Function name	Definition
ADD, SUB, MUL, DIV	binary math operators
POS	$POS(a) = \max\{a, 0\}$
Terminal name	Definition
pt	processing time (p_j)
dd	due date (d_j)
w	weight (w_j)
N	number of jobs
Nr	remaining (unscheduled) jobs
SP	sum of processing times of all jobs
SPr	sum of processing times of remaining jobs
SD	sum of due dates of all jobs
SL	positive slack, $\max\{d_j - p_j - time, 0\}$
AR	time till job arrival (waiting time), $\max\{r_j - time, 0\}$

11/25

Results

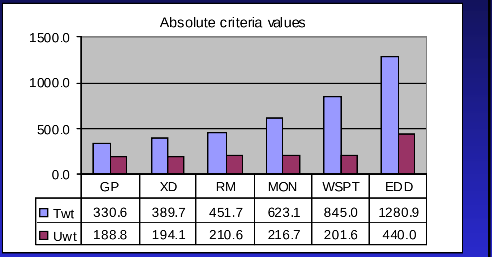
- an example of evolved priority function:

job priority =
((SL+((SPr/(((((pt/w)+((Nr+(Nr/w))*AR))+((SPr/(N/((N-SPr)+(SPr+SL))))/(SPr*((SL+(pt+SL))/SPr))))/((pt-SPr)+(SPr+SL)))+w))/(N+N)))+((((pt/w)+((Nr+(Nr/w))*AR))+((SPr/((N/((pt-SPr)+(SPr+SL)))+w))/((pos((N*(dd+((N/(N+((N+N)*Nr)))/w))))*(N/(w/(SL+((SD-SPr)+SL)))))*((SL+(pt/N))/SPr))))-(((SL/(((N+N)*Nr)/((N*(dd+(AR/w)))-((SL+(Nr/w))+((SP+w)-pos(((pt+SL)*(dd+((N+N)/w)))))))))-(Nr/w))/((pt-SPr)+(((pt+SL)/pt)*w)))))

- GP evolved solution is compared to existing heuristics (for a given environment)
- results shown in two forms:
  1. total normalized criteria values (less is better)
  2. percentage of test cases in which the heuristic achieved the best known result – dominance percentage (more is better)
- results generated on unseen set of evaluation test cases

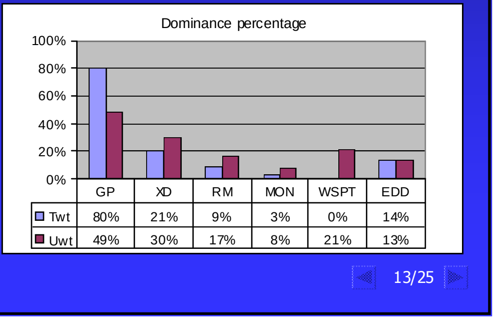
Results

- one machine, job release dates, weighted tardiness problem



Absolute criteria values

	GP	XD	RM	MON	WSPT	EDD
▨ Twt	330.6	389.7	451.7	623.1	845.0	1280.9
■ Uwt	188.8	194.1	210.6	216.7	201.6	440.0

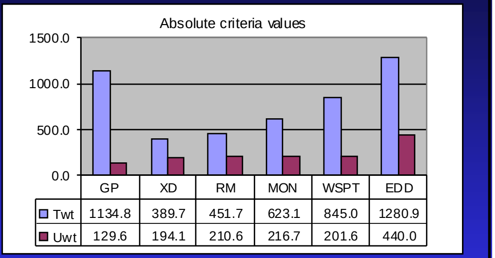


Dominance percentage

	GP	XD	RM	MON	WSPT	EDD
▨ Twt	80%	21%	9%	3%	0%	14%
■ Uwt	49%	30%	17%	8%	21%	13%

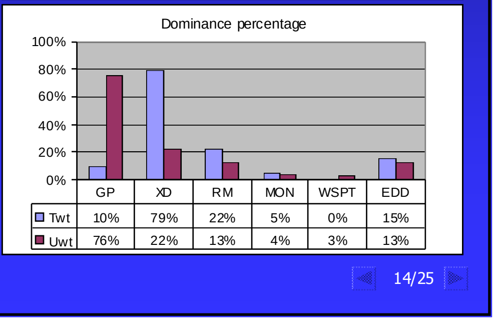
Results

- one machine, job release dates, weighted number of late jobs



Absolute criteria values

	GP	XD	RM	MON	WSPT	EDD
▨ Twt	1134.8	389.7	451.7	623.1	845.0	1280.9
■ Uwt	129.6	194.1	210.6	216.7	201.6	440.0

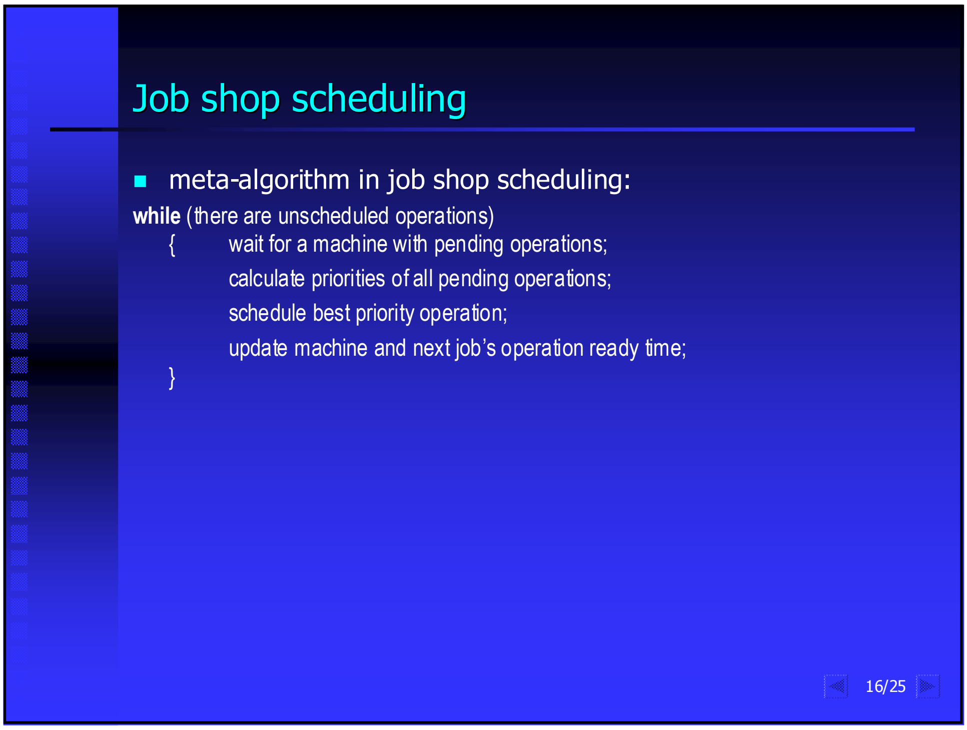


Dominance percentage

	GP	XD	RM	MON	WSPT	EDD
▨ Twt	10%	79%	22%	5%	0%	15%
■ Uwt	76%	22%	13%	4%	3%	13%

Job shop scheduling

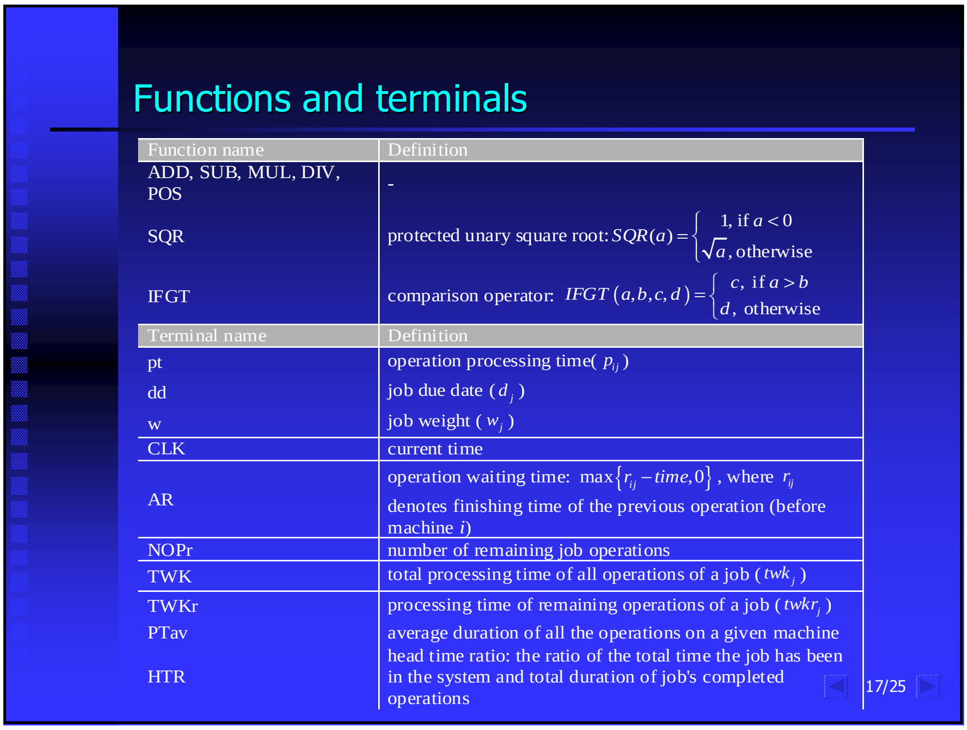
- jobs consist of series of operations
- each operation executed on a predefined machine in a predefined sequence (job dependant)
- input variables:
  - p_{ij} - operation processing time of job *j* on machine *i*
  - w_j - job weight
  - d_j - job due date
- what operations can be scheduled at some moment in time?
  - available operations
  - operations with known ready time in future (the job's previous operation is currently executing)
    - time till the operation can start is smaller than the duration of the shortest available operation
- above categories denoted as *pending operations*



Job shop scheduling

- meta-algorithm in job shop scheduling:

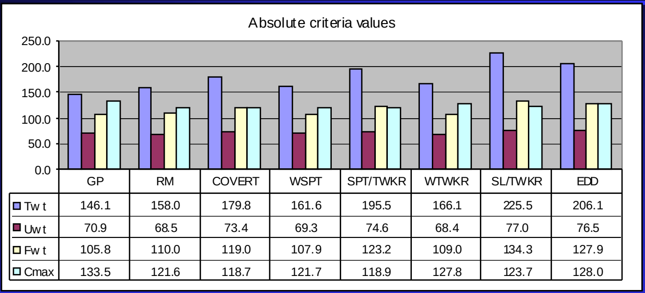
```
while (there are unscheduled operations)
    {       wait for a machine with pending operations;
            calculate priorities of all pending operations;
            schedule best priority operation;
            update machine and next job's operation ready time;
    }
```



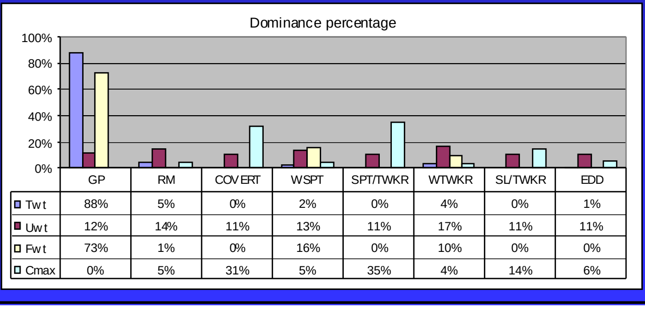
Functions and terminals

Function name	Definition
ADD, SUB, MUL, DIV, POS	-
SQR	protected unary square root: $SQR(a) = \begin{cases} 1, \text{ if } a < 0 \\ \sqrt{a}, \text{otherwise} \end{cases}$
IFGT	comparison operator: $IFGT(a,b,c,d) = \begin{cases} c, \text{ if } a > b \\ d, \text{ otherwise} \end{cases}$
Terminal name	Definition
pt	operation processing time(p_{ij})
dd	job due date (d_j)
w	job weight (w_j)
CLK	current time
AR	operation waiting time: $\max\{r_{ij} - time, 0\}$, where r_{ij} denotes finishing time of the previous operation (before machine *i*)
NOPr	number of remaining job operations
TWK	total processing time of all operations of a job (twk_j)
TWKr	processing time of remaining operations of a job ($twkr_j$)
PTav	average duration of all the operations on a given machine
HTR	head time ratio: the ratio of the total time the job has been in the system and total duration of job's completed operations

Results – weighted tardiness optimization



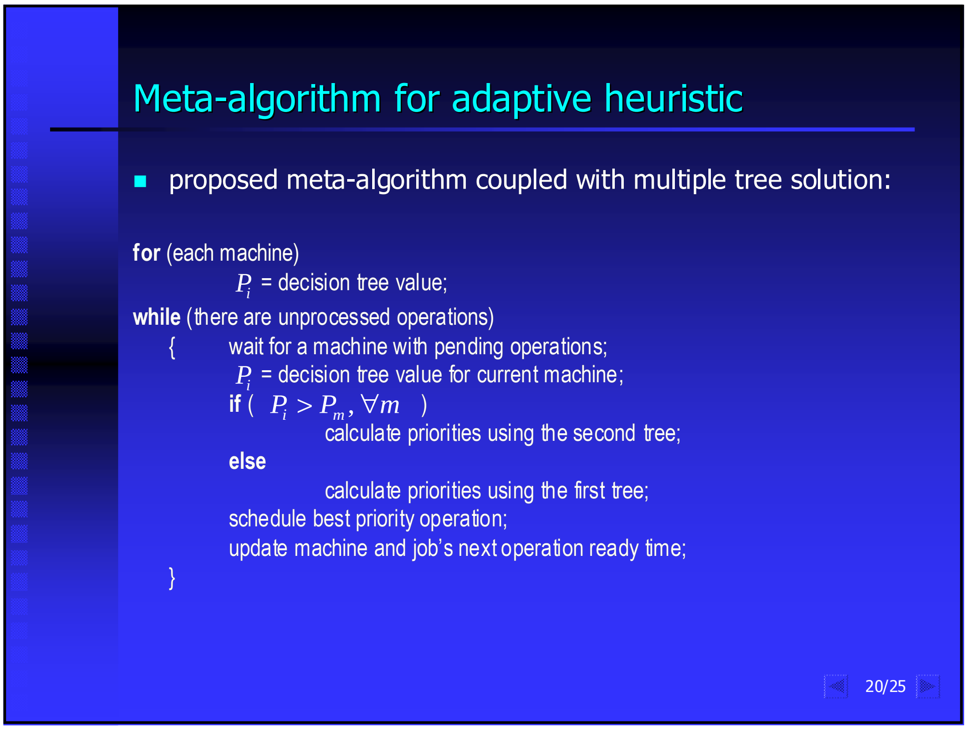
	GP	RM	COVERT	WSPT	SPT/TWKR	WTWKR	SL/TWKR	EDD
▨ Tw t	146.1	158.0	179.8	161.6	195.5	166.1	225.5	206.1
■ Uw t	70.9	68.5	73.4	69.3	74.6	68.4	77.0	76.5
□ Fw t	105.8	110.0	119.0	107.9	123.2	109.0	134.3	127.9
□ Cmax	133.5	121.6	118.7	121.7	118.9	127.8	123.7	128.0



	GP	RM	COV ERT	W SPT	SPT/TWKR	WTWKR	SL/TWKR	EDD
▨ Tw t	88%	5%	0%	2%	0%	4%	0%	1%
■ Uw t	12%	14%	11%	13%	11%	17%	11%	11%
□ Fw t	73%	1%	0%	16%	0%	10%	0%	0%
□ Cmax	0%	5%	31%	5%	35%	4%	14%	6%

Adaptive scheduling heuristic

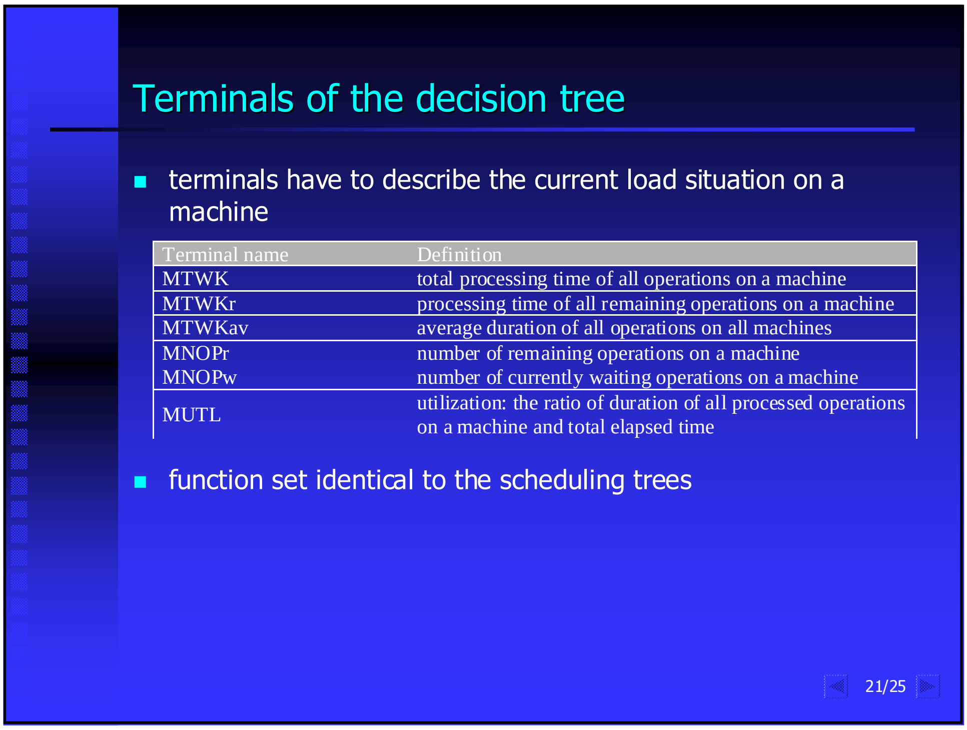
- often a small number of resources present *bottlenecks* in a system (relatively much greater load)
- scheduling efficiency can be improved with *a priori* load data
  - generally not available in advance
- usage of a 'load-aware' meta-heuristic with bottleneck identification may be useful
- let GP develop such a heuristic
- proposed GP solution structure in 3 independent trees:
  - two scheduling trees – priority functions for 'high load' and 'average load' conditions/resources
  - decision tree – determines which of the other two is used at a given moment



Meta-algorithm for adaptive heuristic

- proposed meta-algorithm coupled with multiple tree solution:

```
for (each machine)
        $P_i$ = decision tree value;
while (there are unprocessed operations)
    {   wait for a machine with pending operations;
        $P_i$ = decision tree value for current machine;
        if ( $P_i > P_m, \forall m$ )
                calculate priorities using the second tree;
        else
                calculate priorities using the first tree;
        schedule best priority operation;
        update machine and job's next operation ready time;
    }
```

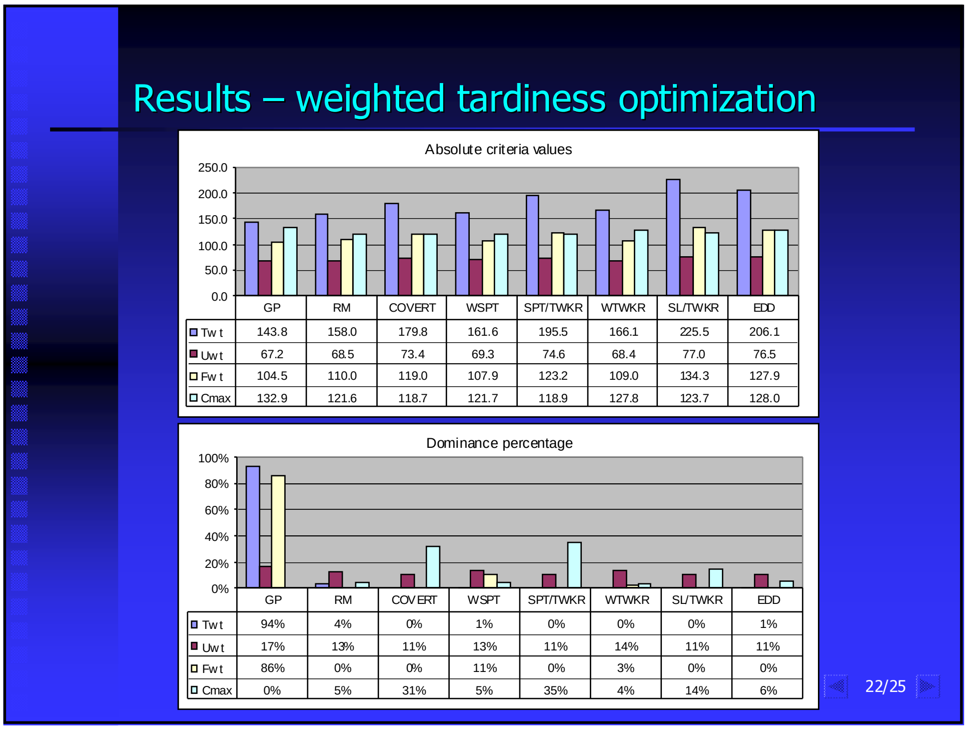


Terminals of the decision tree

- terminals have to describe the current load situation on a machine

Terminal name	Definition
MTWK	total processing time of all operations on a machine
MTWKr	processing time of all remaining operations on a machine
MTWKav	average duration of all operations on all machines
MNOPr	number of remaining operations on a machine
MNOPw	number of currently waiting operations on a machine
MUTL	utilization: the ratio of duration of all processed operations on a machine and total elapsed time

- function set identical to the scheduling trees



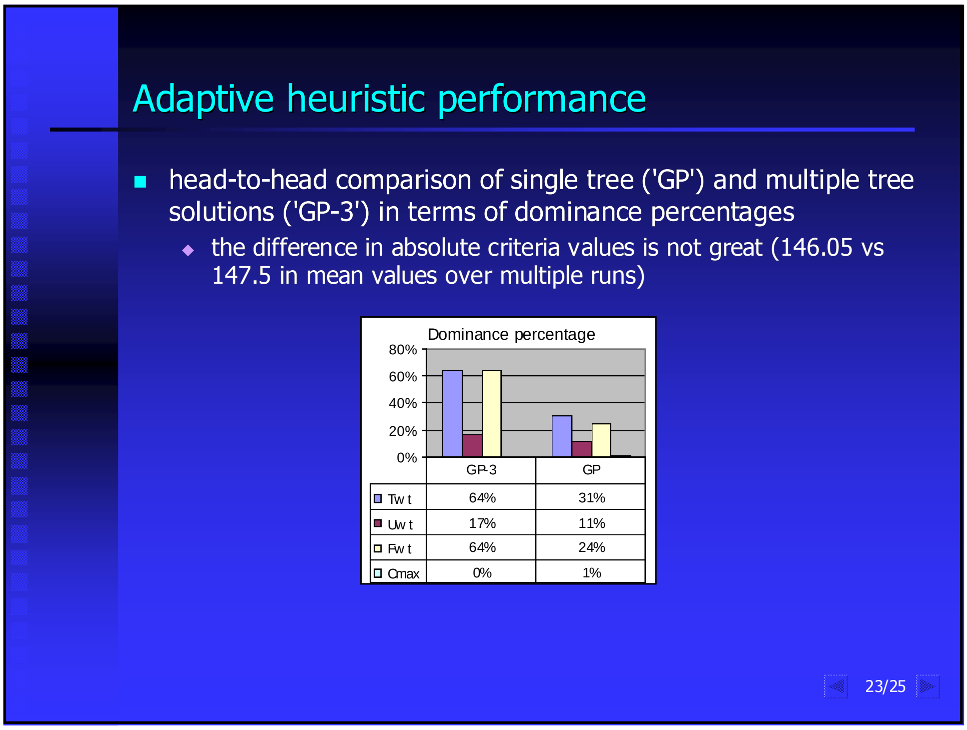
Results – weighted tardiness optimization

Absolute criteria values

	GP	RM	COVERT	WSPT	SPT/TWKR	WTWKR	SL/TWKR	EDD
□ Tw t	143.8	158.0	179.8	161.6	195.5	166.1	225.5	206.1
□ Uw t	67.2	68.5	73.4	69.3	74.6	68.4	77.0	76.5
□ Fw t	104.5	110.0	119.0	107.9	123.2	109.0	134.3	127.9
□ Cmax	132.9	121.6	118.7	121.7	118.9	127.8	123.7	128.0

Dominance percentage

	GP	RM	COV ERT	W SPT	SPT/TWKR	WTWKR	SL/TWKR	EDD
□ Tw t	94%	4%	0%	1%	0%	0%	0%	1%
□ Uw t	17%	13%	11%	13%	11%	14%	11%	11%
□ Fw t	86%	0%	0%	11%	0%	3%	0%	0%
□ Cmax	0%	5%	31%	5%	35%	4%	14%	6%

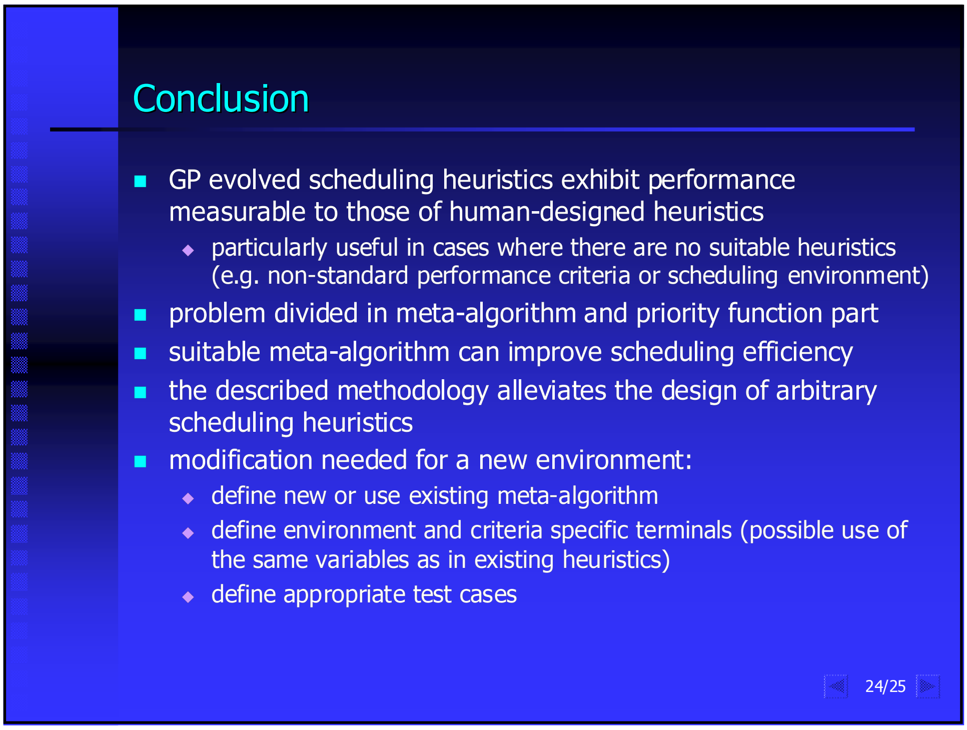


Adaptive heuristic performance

- head-to-head comparison of single tree ('GP') and multiple tree solutions ('GP-3') in terms of dominance percentages
  - the difference in absolute criteria values is not great (146.05 vs 147.5 in mean values over multiple runs)

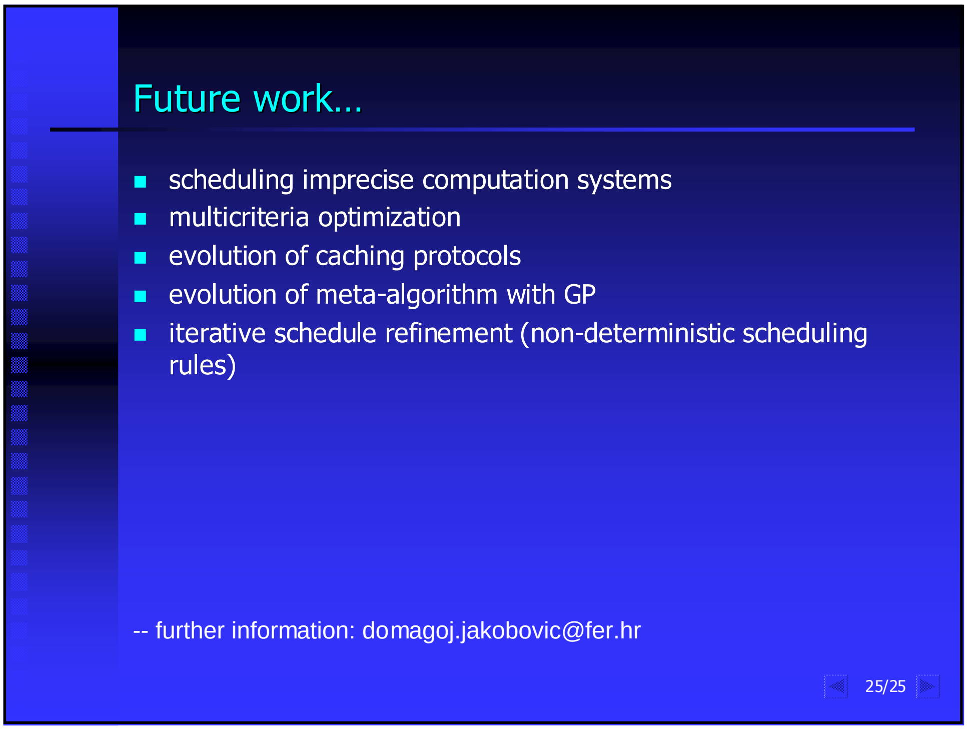
Dominance percentage

	GP-3	GP
□ Tw t	64%	31%
□ Uw t	17%	11%
□ Fw t	64%	24%
□ Cmax	0%	1%



Conclusion

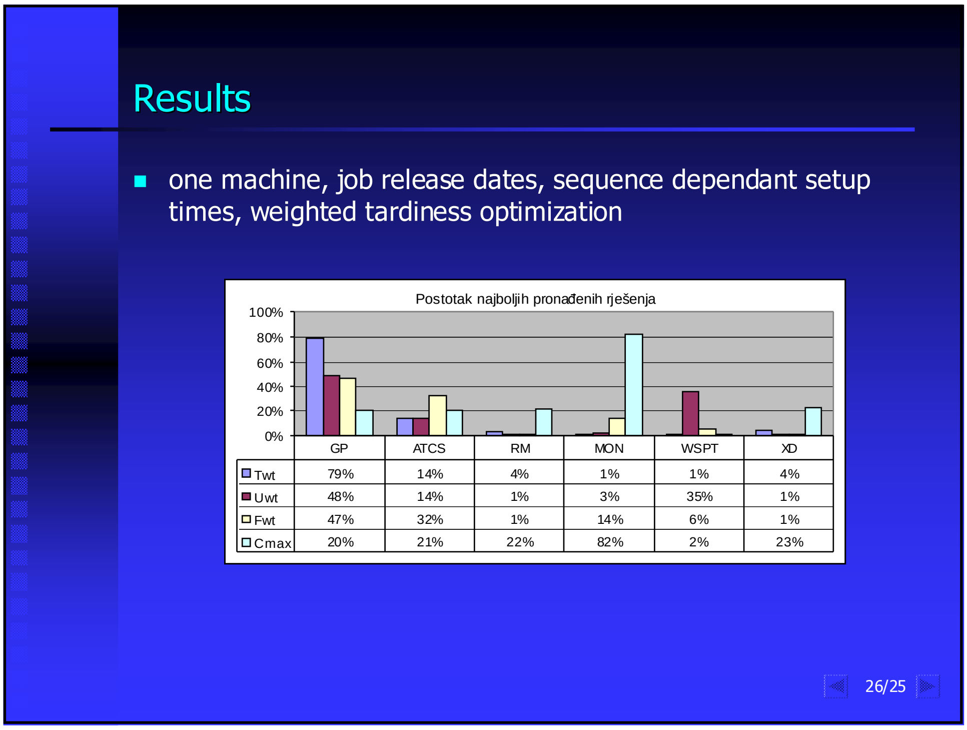
- GP evolved scheduling heuristics exhibit performance measurable to those of human-designed heuristics
  - particularly useful in cases where there are no suitable heuristics (e.g. non-standard performance criteria or scheduling environment)
- problem divided in meta-algorithm and priority function part
- suitable meta-algorithm can improve scheduling efficiency
- the described methodology alleviates the design of arbitrary scheduling heuristics
- modification needed for a new environment:
  - define new or use existing meta-algorithm
  - define environment and criteria specific terminals (possible use of the same variables as in existing heuristics)
  - define appropriate test cases



Future work…

- scheduling imprecise computation systems
- multicriteria optimization
- evolution of caching protocols
- evolution of meta-algorithm with GP
- iterative schedule refinement (non-deterministic scheduling rules)

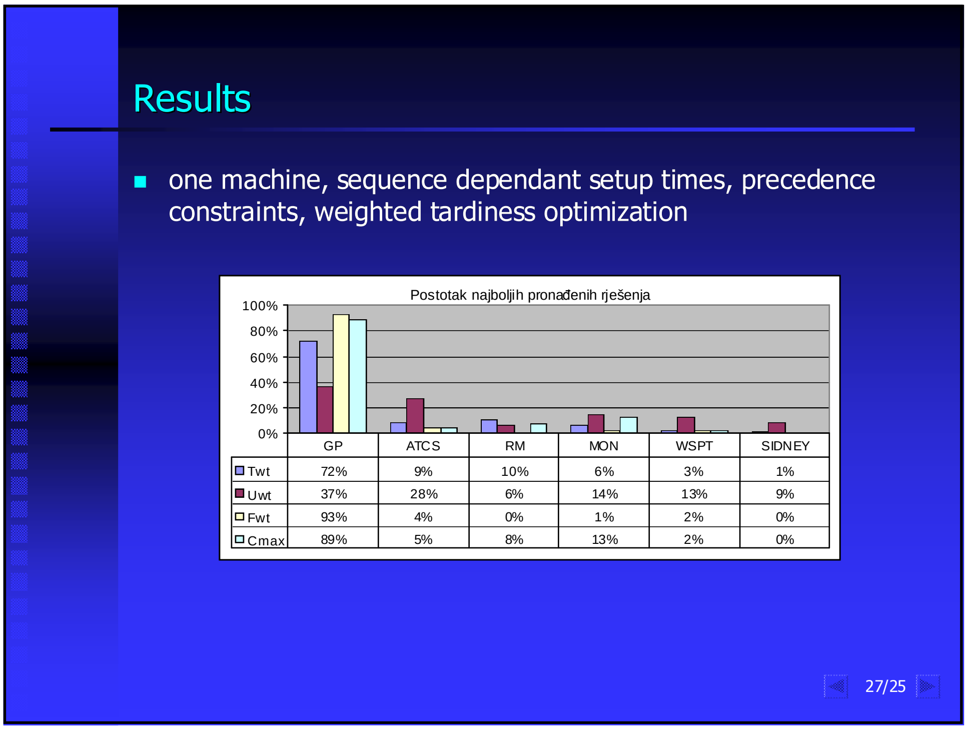
-- further information: domagoj.jakobovic@fer.hr



Results

- one machine, job release dates, sequence dependant setup times, weighted tardiness optimization

	GP	ATCS	RM	MON	WSPT	XD
Twt	79%	14%	4%	1%	1%	4%
Uwt	48%	14%	1%	3%	35%	1%
Fwt	47%	32%	1%	14%	6%	1%
Cmax	20%	21%	22%	82%	2%	23%



Results

- one machine, sequence dependant setup times, precedence constraints, weighted tardiness optimization

	GP	ATCS	RM	MON	WSPT	SIDNEY
Twt	72%	9%	10%	6%	3%	1%
Uwt	37%	28%	6%	14%	13%	9%
Fwt	93%	4%	0%	1%	2%	0%
Cmax	89%	5%	8%	13%	2%	0%