



Job Scheduling

Uwe Schwiegelshohn

EPIT 2007, June 5
Ordonnancement



Content of the Lecture



- What is job scheduling?
- Single machine problems and results
- Makespan problems on parallel machines
- Utilization problems on parallel machines
- Completion time problems on parallel machines
- Exemplary workload problem



- Processor scheduling
 - ➔ Jobs are executed on a CPU in a multitasking operating system.
 - ➔ Users submit jobs to web servers and receive results after some time.
 - ➔ Users submit batch computing jobs to a parallel processor.
- Bandwidth scheduling
 - ➔ Users call other persons and need bandwidth for some period of time.
- Airport gate scheduling
 - ➔ Airlines require gates for their flights at an airport.
- Repair crew scheduling
 - ➔ Customer request the repair of their devices.





- Independent jobs
 - ➔ No known precedence constraints
 - Difference to task scheduling
- Atomic jobs
 - ➔ No job stages
 - Difference to job shop scheduling
- Batch jobs
 - ➔ No deadlines or due dates
 - Difference to deadline scheduling

p_j	processing time of job j	
r_j	release date of job j	earliest starting time
w_j	weight of job j	importance of the job
m_j	size of job j	parallelism of the job



- 1: single machine
 - ➔ Many job scheduling problems are easy.
- P_m : **m parallel identical machines**
 - ➔ Every job requires the same processing time on each machine.
 - ➔ Use of machine eligibility constraints M_j if job j can only be executed on a subset of machines
 - Airport gate scheduling: wide and narrow body airplanes
- Q_m : **m uniformly related machines**
 - ➔ The machines have different speeds v_i that are valid for all jobs.
 - ➔ In deterministic scheduling, results for P_m and Q_m are related.
 - ➔ In online scheduling, there are significant differences between P_m and Q_m .
- R_m : **m unrelated machines**
 - ➔ Each job has a different processing time on each machine.



- Release dates r_j
 - Parallelism m_j
 - ➔ Fixed parallelism: m_j machines must be available during the whole processing of the job.
 - ➔ Malleable jobs: The number of allocated machines can change before or during the processing of the job.
 - Preemption
 - ➔ The processing of a job can be interrupted and continued on another machine.
 - ➔ Gang scheduling: The processing of a job must be continued on the same machines.
 - Machine eligibility constraints M_j
 - Breakdown of machines
 - ➔ $m(t)$: time dependent availability
- } rarely discussed in the literature



- Completion time of job j : C_j
- Owner oriented:
 - ➔ Makespan: $C_{\max} = \max (C_1, \dots, C_n)$
 - completion time of the last job in the system
 - ➔ Utilization U_t : Average ratio of busy machines to all machines in the interval $(0, t]$ for some time t .
- User oriented:
 - ➔ Total completion time: $\sum C_j$
 - ➔ Total weighted completion time: $\sum w_j C_j$
 - ➔ Total weighted waiting time: $\sum w_j (C_j - p_j - r_j) = \sum w_j C_j - \underbrace{\sum w_j (p_j + r_j)}_{\text{const.}}$
 - ➔ Total weighted flow time: $\sum w_j (C_j - r_j) = \sum w_j C_j - \underbrace{\sum w_j r_j}_{\text{const.}}$
- Regular objective functions:
 - ➔ non decreasing in C_1, \dots, C_n



- **Deterministic scheduling problems**
 - ➔ All problem parameters are available at time 0.
 - ➔ Optimal algorithms,
 - ➔ Simple individual approximation algorithms
 - ➔ Polynomial time approximation schemes
- **Online scheduling problems**
 - ➔ Parameters of job j are unknown until r_j (submission over time).
 - ➔ p_j is unknown C_j (nonclairvoyant scheduling).
 - ➔ Competitive analysis
- **Stochastic scheduling**
 - ➔ Known distribution of job parameters
 - ➔ Randomized algorithms
- **Workload based scheduling**
 - ➔ An algorithm is parameterized to achieve a good solution for a given workload.



- No machine is kept idle while a job is waiting for processing.
An optimal schedule need not be nondelay!

Example: 1 || $\sum w_j C_j$

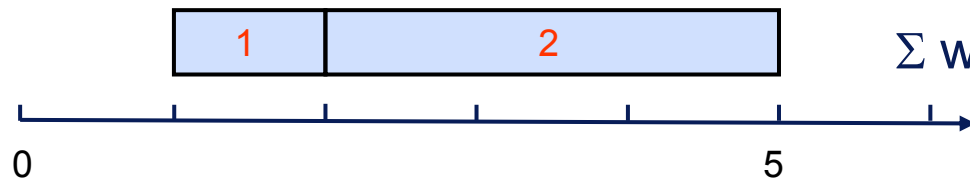
jobs	1	2
p_j	1	3
r_j	1	0
w_j	2	1

Nondelay schedule



$$\sum w_j C_j = 11$$

Optimal schedule



$$\sum w_j C_j = 9$$

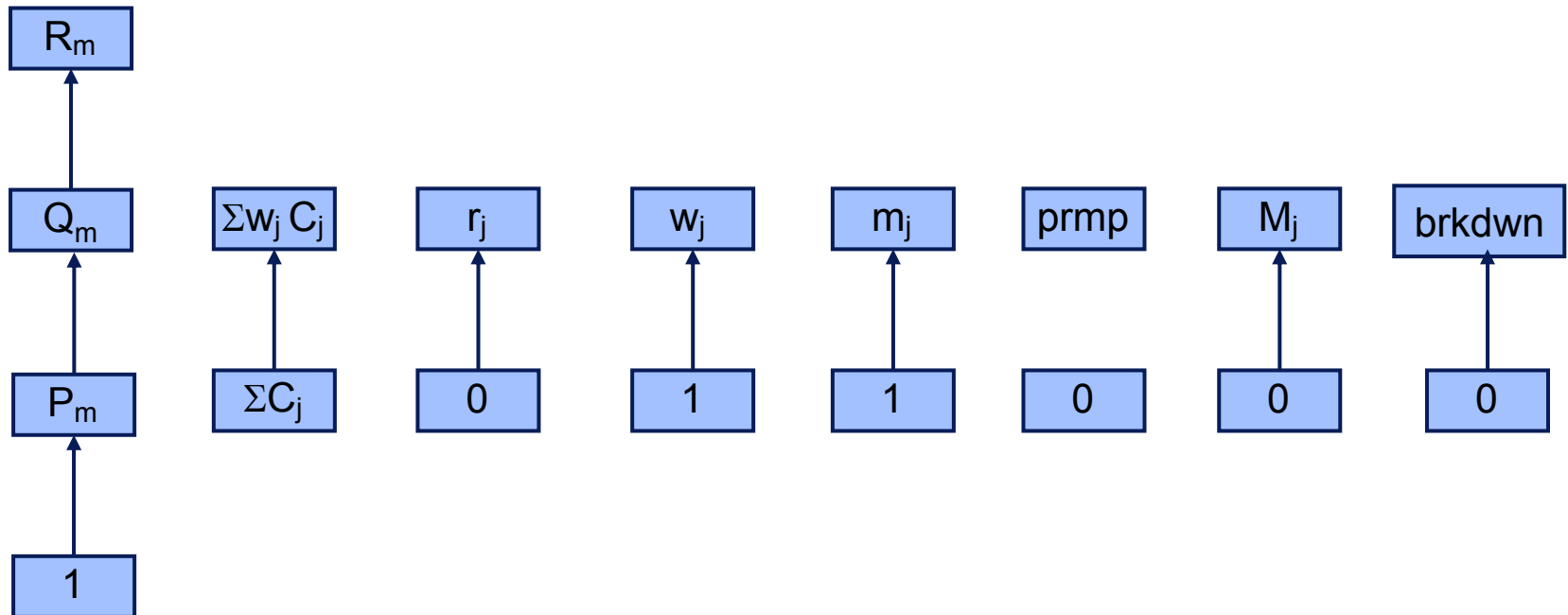


Some problems are special cases of other problems:

Notation: $\alpha_1 | \beta_1 | \gamma_1 \propto$ (reduces to) $\alpha_2 | \beta_2 | \gamma_2$

Examples:

$1 \parallel \Sigma C_j \propto 1 \parallel \Sigma w_j C_j \propto P_m \parallel \Sigma w_j C_j \propto P_m | m_j | \Sigma w_j C_j$





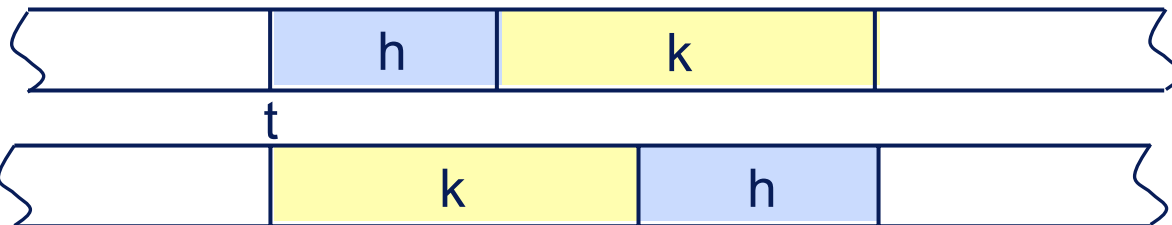
- What is job scheduling?
- Single machine problems and results
- Makespan problems on parallel machines
- Utilization problems on parallel machines
- Completion time problems on parallel machines
- Exemplary workload problem



- 1 || $\sum w_j C_j$ is easy and can be solved by sorting all jobs in decreasing Smith order w_j/p_j (weighted shortest processing time first (WSPT) rule, Smith, 1956).
 - ➔ Nondelay schedule
 - ➔ Proof by contradiction and localization:

If the WSPT rule is violated then it is violated by a pair of neighboring task h and k .

$$S_1: \sum w_j C_j = \dots + w_h(t+p_h) + w_k(t + p_h + p_k)$$



$$S_2: \sum w_j C_j = \dots + w_k(t+p_k) + w_h(t + p_k + p_h)$$

$$S_1 - S_2:$$

$$w_k p_h - w_h p_k > 0$$

$$w_k/p_k > w_h/p_h$$



- Every nondelay schedule has
 - ➔ optimal makespan and
 - ➔ optimal utilization for any interval starting at time 0.
- WSPT requires knowledge of the processing times
 - ➔ No direct application to nonclairvoyant scheduling
- $1 \mid \text{prmp} \mid \sum C_j$ is easy.
 - ➔ The online nonclairvoyant version (Round Robin) has a competitive factor of $2 - 2/(n+1)$ (Motwani, Phillips, Torng, 1994).
- $1 \mid r_j, \text{prmp} \mid \sum C_j$ is easy.
 - ➔ The online, clairvoyant version is easy.
- $1 \mid r_j \mid \sum C_j$ is strongly NP hard.
- $1 \mid r_j, \text{prmp} \mid \sum w_j C_j$ is strongly NP hard.
 - ➔ The WSRPT (remaining processing time) rule is not optimal.



- $1 \mid r_j, \text{prmp} \mid \sum w_j (C_j - r_j)$ and $1 \mid r_j, \text{prmp} \mid \sum w_j C_j$
 - ➔ Same optimal solution
 - ➔ Larger approximation factor for $1 \mid r_j, \text{prmp} \mid \sum w_j (C_j - r_j)$.
 - ➔ No constant approximation factor for the total flowtime objective (Kellerer, Tautenhahn, Wöginger, 1999)

$$\begin{aligned}
 & \frac{\sum w_j \cdot (C_j(S) - r_j)}{\sum w_j \cdot (C_j(\text{OPT}) - r_j)} = \\
 & = \frac{\sum w_j \cdot C_j(S)}{\sum w_j \cdot C_j(\text{OPT})} \sum w_j \cdot (C_j(\text{OPT}) - r_j) + \left(\frac{\sum w_j \cdot C_j(S)}{\sum w_j \cdot C_j(\text{OPT})} - 1 \right) \sum w_j \cdot r_j \\
 & = \frac{\sum w_j \cdot C_j(S)}{\sum w_j \cdot (C_j(\text{OPT}) - r_j)} + \left(\frac{\sum w_j \cdot C_j(S)}{\sum w_j \cdot C_j(\text{OPT})} - 1 \right) \cdot \frac{\sum w_j \cdot r_j}{\sum w_j \cdot (C_j(\text{OPT}) - r_j)}
 \end{aligned}$$



- $1 \mid r_j \mid \Sigma C_j$
 - ➔ Approximation factor $e/(e-1)=1.58$ (Chekuri, Motwani, Natarajan, Stein, 2001)
 - ➔ Clairvoyant online scheduling: competitive factor 2 (Hoogeveen, Vestjens, 1996)
- $1 \mid r_j \mid \Sigma w_j C_j$
 - ➔ Approximation factor 1.6853 (Goemans, Queyranne, Schulz, Skutella, Wang, 2002)
 - ➔ Clairvoyant online scheduling: competitive factor 2 (Anderson, Potts, 2004)
- $1 \mid r_j, \text{prmp} \mid \Sigma w_j C_j$
 - ➔ Approximation factor 1.3333,
 - ➔ Randomized online algorithm with the competitive factor 1.3333
 - ➔ WSPT online algorithm with competitive factor 2 (all results: Schulz, Skutella, 2002)



- What is job scheduling?
- Single machine problems and results
- Makespan problems on parallel machines
- Utilization problems on parallel machines
- Completion time problems on parallel machines
- Exemplary workload problem



- A scheduling problem for parallel machines consists of 2 steps:
 - ➔ Allocation of jobs to machines
 - ➔ Generating a sequence of the jobs on a machine
- A minimal makespan represents a balanced load on the machines if no single job dominates the schedule.

$$C_{\max}(\text{OPT}) \geq \max \left\{ \max \{ p_j \}, \frac{1}{m} \cdot \sum p_j \right\}$$

- Preemption may improve a schedule even if all jobs are released at the same time.

$$C_{\max}(\text{OPT}) = \max \left\{ \max \{ p_j \}, \frac{1}{m} \cdot \sum p_j \right\}$$

- Optimal schedules for parallel identical machines are nondelay.



- $P_m \parallel C_{\max}$ is strongly NP-hard (Garey, Johnson 1979).
- Approximation algorithm: Longest processing time first (LPT) rule (Graham, 1966)
 - ➔ Whenever a machine is free, the longest job among those not yet processed is put on this machine.

➔ Tight approximation factor:
$$\frac{C_{\max}(\text{LPT})}{C_{\max}(\text{OPT})} \leq \frac{4}{3} - \frac{1}{3m}$$

- ➔ The optimal schedule $C_{\max}(\text{OPT})$ is not necessarily known but a simple lower bound can be used:

$$C_{\max}(\text{OPT}) \geq \frac{1}{m} \sum_{j=1}^n p_j$$



- If the claim is not true, then there is a counterexample with the smallest number n of jobs.
- The shortest job n in this counterexample is the last job to start processing (LPT) and the last job to finish processing.
 - ➔ If n is not the last job to finish processing then deletion of n does not change C_{\max} (LPT) while C_{\max} (OPT) cannot increase.
 - ➔ A counter example with $n - 1$ jobs
- Under LPT, job n starts at time $C_{\max}(\text{LPT}) - p_n$.
 - ➔ In time interval $[0, C_{\max}(\text{LPT}) - p_n]$, all machines are busy.

$$C_{\max}(\text{LPT}) - p_n \leq \frac{1}{m} \sum_{j=1}^{n-1} p_j$$



$$C_{\max}(\text{LPT}) \leq p_n + \frac{1}{m} \sum_{j=1}^{n-1} p_j = p_n \left(1 - \frac{1}{m}\right) + \frac{1}{m} \sum_{j=1}^n p_j$$

$$\frac{4}{3} - \frac{1}{3m} < \frac{C_{\max}(\text{LPT})}{C_{\max}(\text{OPT})} \leq \frac{p_n \left(1 - \frac{1}{m}\right)}{C_{\max}(\text{OPT})} + \frac{\frac{1}{m} \sum_{j=1}^n p_j}{C_{\max}(\text{OPT})} \leq \frac{p_n \left(1 - \frac{1}{m}\right)}{C_{\max}(\text{OPT})} + 1$$

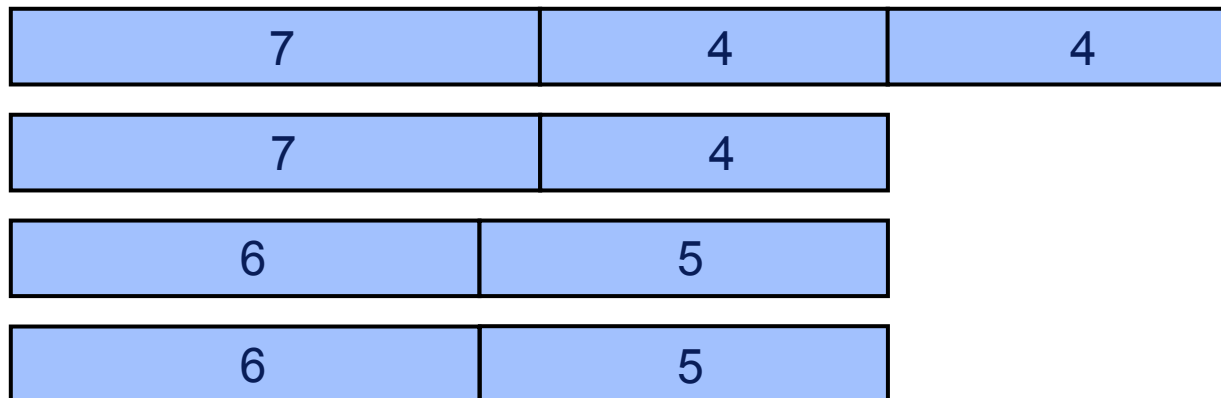
$$C_{\max}(\text{OPT}) < 3p_n$$

At most two jobs are scheduled on each machine.
For such a problem, LPT is optimal.



jobs	1	2	3	4	5	6	7	8	9
p_j	7	7	6	6	5	5	4	4	4

- 4 parallel machines: $P4||C_{\max}$
- $C_{\max}(\text{OPT}) = 12 = 7+5 = 6+6 = 4+4+4$
- $C_{\max}(\text{LPT}) = 15 = 11+4 = (4/3 - 1/(3 \cdot 4)) \cdot 12$

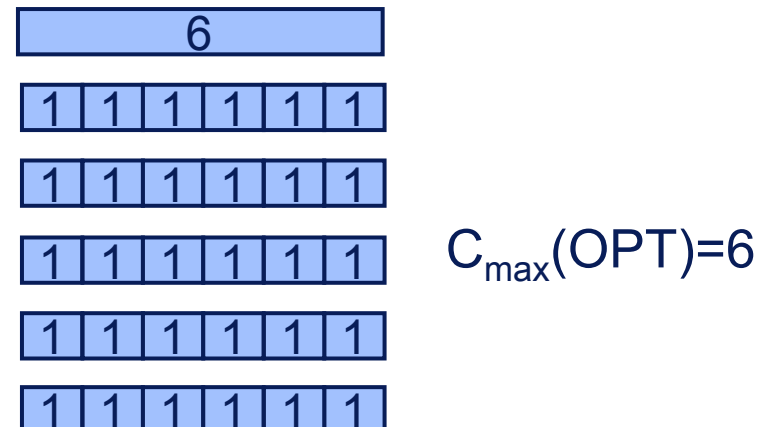
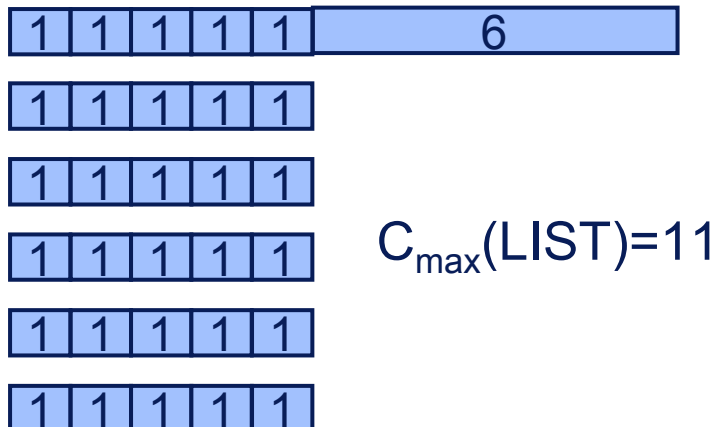




- LPT requires knowledge of the processing times.
 - ➔ No direct application to nonclairvoyant scheduling
- Arbitrary nondelay schedule (List Scheduling, Graham, 1966)

➔ Tight approximation factor:

$$\frac{C_{\max}(\text{LIST})}{C_{\max}(\text{OPT})} \leq 2 - \frac{1}{m}$$





Let A be an algorithm for a job scheduling problem without release dates and with

$$\frac{C_{\max}(A)}{C_{\max}(\text{OPT})} \leq k$$

Then there is an algorithm A' for the corresponding online job scheduling problem with

$$\frac{C_{\max}(A')}{C_{\max}(\text{OPT})} \leq 2k$$

(Shmoys, Wein, Williamson, 1995)



- S_0 : Jobs available at time $0 = F_{-1} = F_{-2}$
- $F_0 = C_{\max}(A, S_0)$
- S_{i+1} : Jobs released in $(F_{i-1}, F_i]$
- $F_i = C_{\max}(A, S_i)$ such that no job from S_i starts before F_{i-1} .
- Assume that all jobs in S_i are released at time F_{i-2}
 - ➔ $C_{\max}(\text{OPT})$ cannot increase while $C_{\max}(A')$ remains unchanged.
- Proof

$$F_{i-2} + F_i - F_{i-1} \leq k \cdot C_{\max}(A, S_i) = k \cdot C_{\max}(A')$$

$$F_{i-1} - F_{i-2} \leq F_{i-3} + F_{i-1} - F_{i-2} \leq k \cdot C_{\max}(A, S_{i-1}) < k \cdot C_{\max}(A')$$

$$F_i < 2k \cdot C_{\max}(A')$$



- The List scheduling bound $2-1/m$ also applies to $P_m|r_j|C_{\max}$ (Hall, Shmoys, 1989).
- Online extension of List scheduling to parallel jobs:
 - ➔ No machine is kept idle while there is at least one job waiting and there are enough machines idle to start this job (nondelay).
- The List scheduling bound $2-1/m$ also applies to $P_m|m_j|C_{\max}$ (Feldmann, Sgall, Teng, 1994).
- The List scheduling bound $2-1/m$ also applies to $P_m|m_j,r_j|C_{\max}$ (Naroska, Schwiegelshohn, 2002).
 - ➔ $2-1/m$ is a competitive factor for the corresponding online nonclairvoyant scheduling problem.
 - ➔ Proof by induction on the number of different release dates



- The bound holds if during the whole schedule there is no interval with at least $m/2$ idle machines.

$$C_{\max}(\text{OPT}) \geq \frac{1}{m} \sum m_j \cdot p_j \geq \frac{m+1}{2m} \cdot C_{\max}(\text{S}) \geq$$

$$\frac{m}{2m-1} \cdot C_{\max}(\text{S}) = \frac{1}{2 - \frac{1}{m}} \cdot C_{\max}(\text{S})$$

- The sum of machines used in any two intervals is larger than m unless the jobs executed in one interval are a subset of the jobs executed in the other interval.

$$C_{\max}(\text{S}) \leq \max \left\{ \left(2 - \frac{1}{m} \right) \cdot \sum m_j \cdot p_j, \left(2 - \frac{1}{m} \right) \cdot \max \{ p_j \} \right\}$$



- $P_m |prmp| C_{max}$ is easy.
 - ➔ Transformation of a nonpreemptive single machine schedule in a preemptive parallel schedule (McNaughton, 1959)
 - The single machine schedule is split into at most m schedules of length $C_{max}(OPT)$.
 - Each schedule is executed on a different machine.
 - There are at most $m-1$ preemptions.
- $P_m |r_j, prmp| C_{max}$ is easy.
 - ➔ Longest remaining processing time algorithm.
 - ➔ Clairvoyant online scheduling
 - Competitive factor 1 for allocation as late as possible.
 - Competitive factor $e/(e-1)=1.58$ for allocation of machine slots at submission time (Chen, van Vliet, Wöginger, 1995)
 - ➔ Nonclairvoyant online scheduling: same competitive factor $2-1/m$ as for the nonpreemptive case (Shmoys, Wein Williamson, 1995)



- What is job scheduling?
- Single machine problems and results
- Makespan problems on parallel machines
- Utilization problems on parallel machines
- Completion time problems on parallel machines
- Exemplary workload problem



- Utilization U_t is closely related to the makespan C_{\max} if $t=C_{\max}$.
 - ➔ In online job scheduling problems, there is no last submitted job.
 - ➔ U_t with t being the actual time is better suited than the makespan objective.
- $P_m |r_j| U_t$
 - ➔ Nonclairvoyant online scheduling: tight competitive factor for any nondelay schedule 1.3333 (Hussein, Schwiegelshohn, 2006)
 - ➔ Proof by induction on the different release dates.



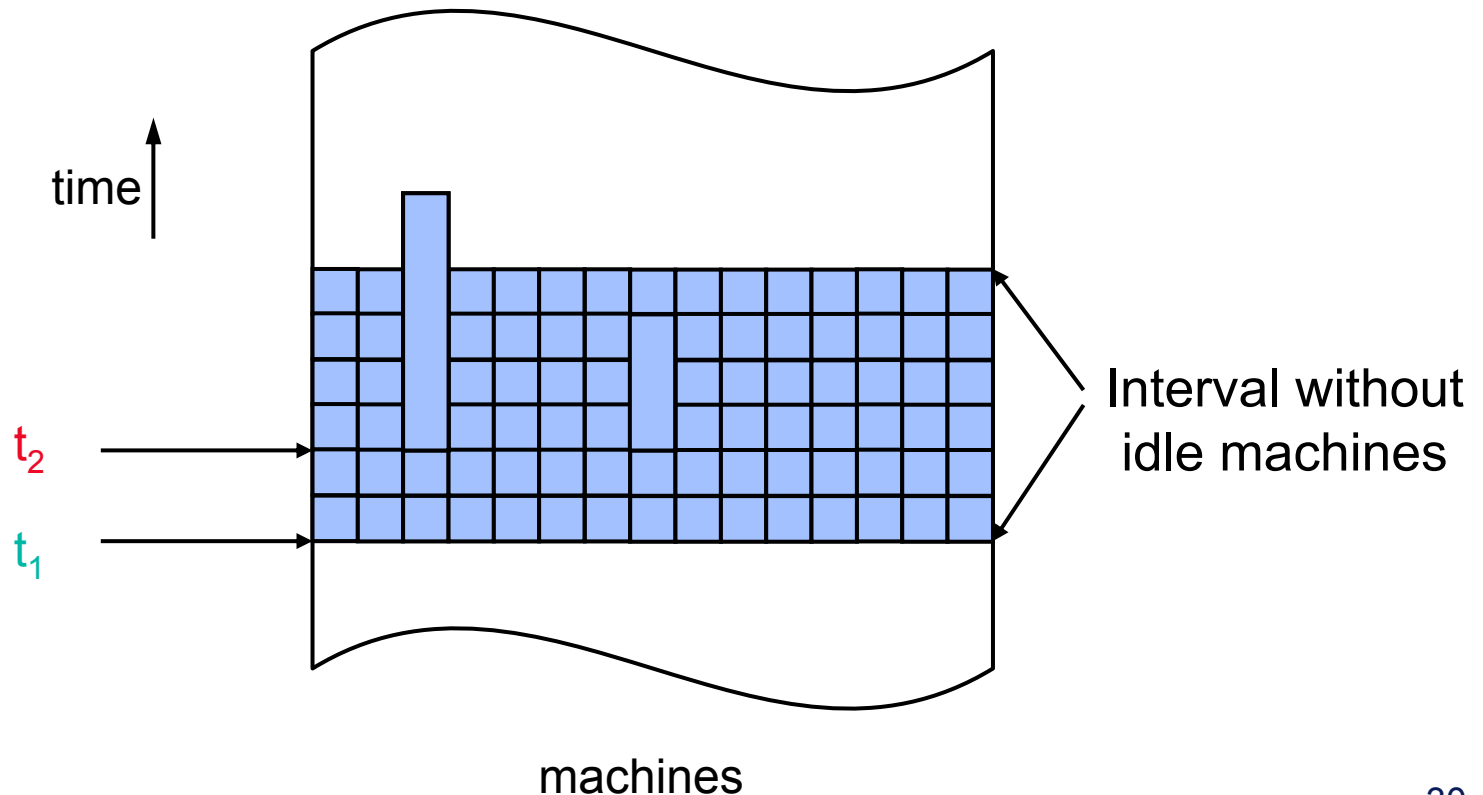
$$U_2(\text{LIST})=0.75$$



$$U_2(\text{OPT})=1$$



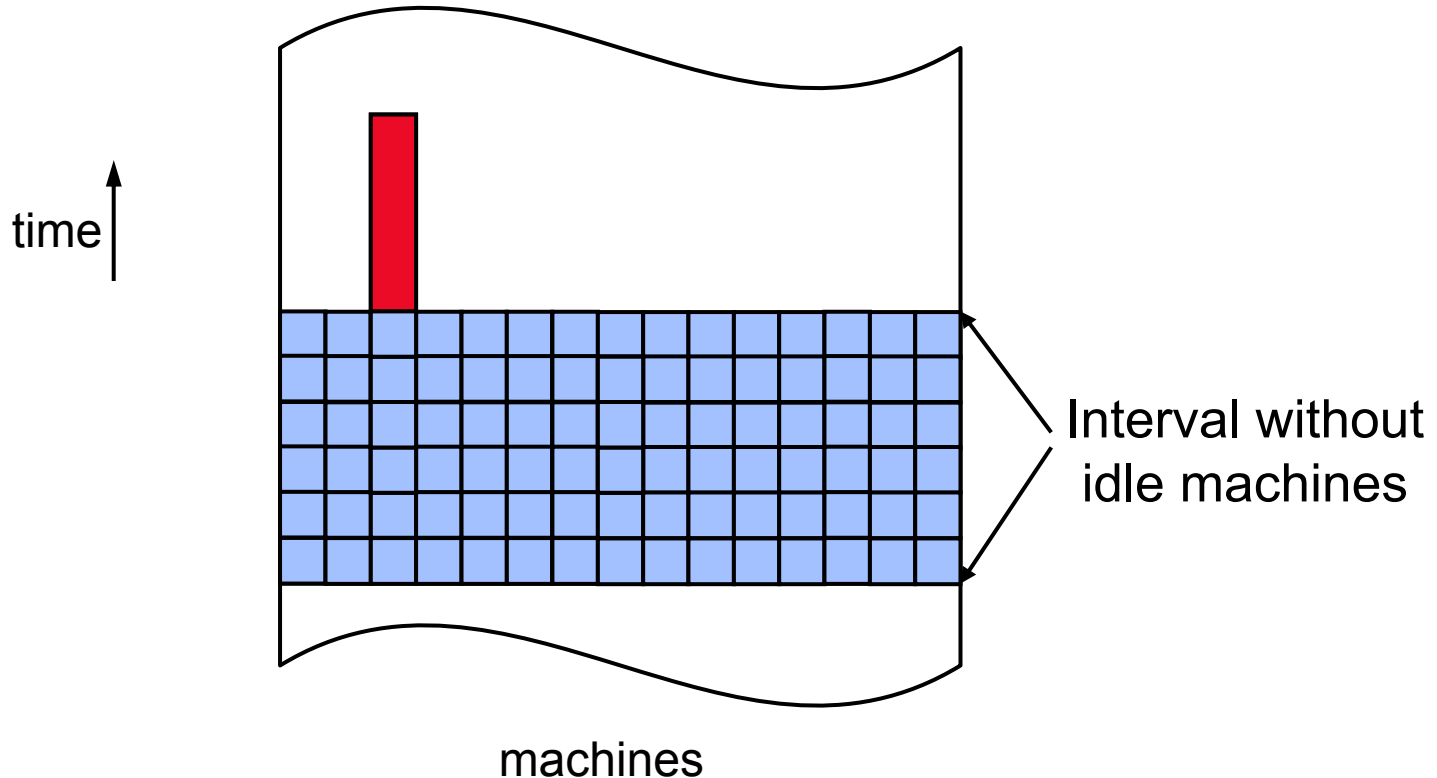
- Transformation of the job system
 - ➔ Reduction of the release dates





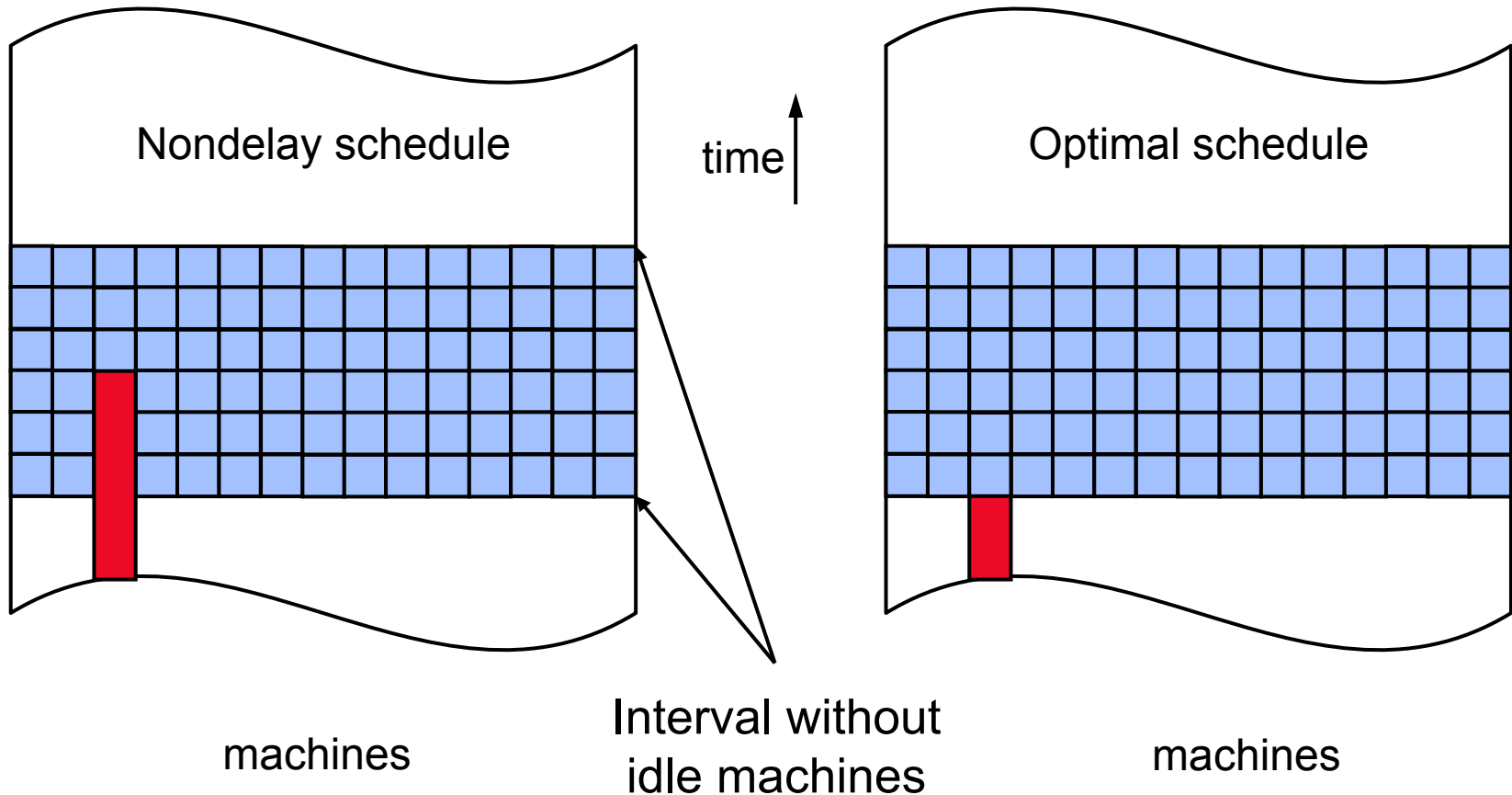
■ Transformation of the job system

- ➔ Splitting of jobs
- ➔ The system only contains short and long jobs.
 - All long jobs start at the end of an interval.



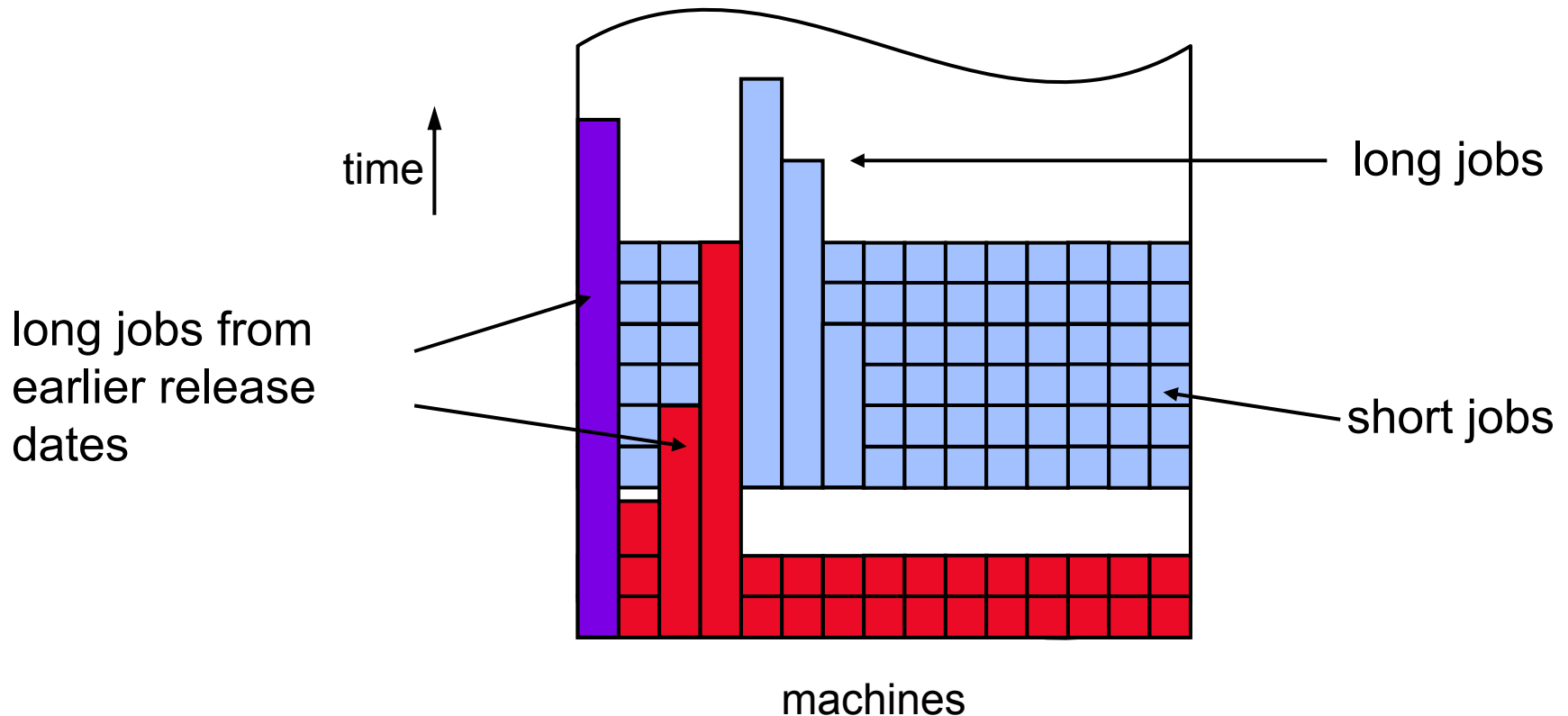


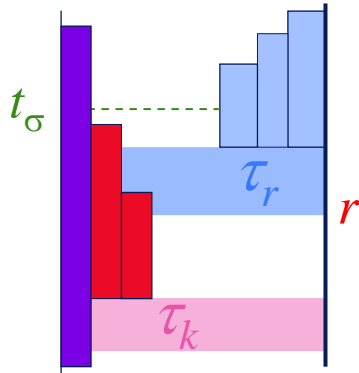
- Transformation of the job system
 - ➔ Modification of jobs with earlier release dates



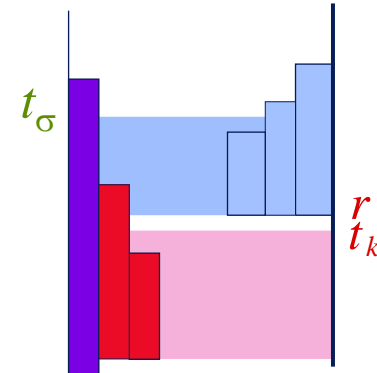


If all long jobs of a transformed job system start at their release date, then the utilization is maximal for all t and the equal priority completion time is minimal.

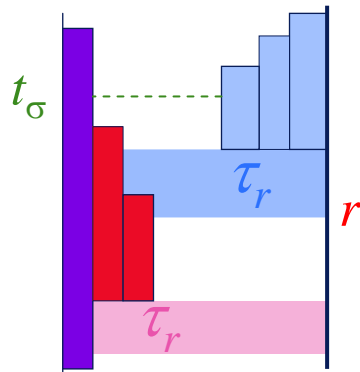




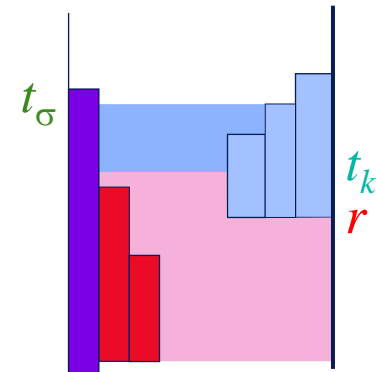
Nondelay schedule S



Optimal schedule



Nondelay schedule S

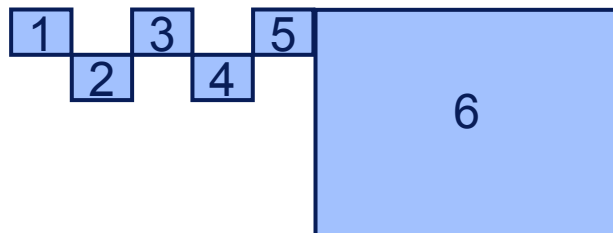


Optimal schedule



- Parallel jobs may cause intermediate idle time even if all jobs are released at time 0.
- Nonclairvoyant online scheduling:
 - ➔ Competitive factor $\rightarrow m$ in the worst case
 - ➔ Competitive factor $\rightarrow 2$ if the actual time $\gg \max\{p_j\}$

Jobs	1	2	3	4	5	6
p_j	$1+\varepsilon$	$1+\varepsilon$	$1+\varepsilon$	$1+\varepsilon$	1	5
r_j	0	1	2	3	4	0
m_j	1	1	1	1	1	5



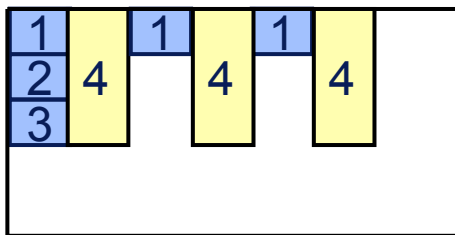
$$U_5(\text{LIST}) = 0.2 + 0.16\varepsilon$$



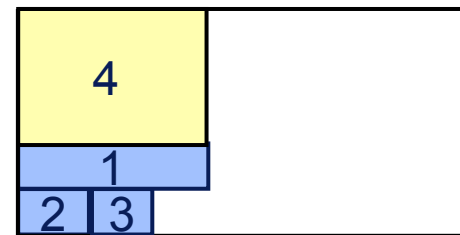
$$U_5(\text{OPT}) = 1$$



- Here, preemption of parallel jobs is based on gang scheduling.
 - ➔ All allocated machines concurrently start, interrupt, resume, and complete the execution of a parallel job.
 - ➔ There is no migration or change of parallelism.
- Nonclairvoyant online scheduling: competitive factor 4 (Schwiegelshohn, Yahyapour, 2000)



$$U_3(A) = 7/15$$



$$U_3(OPT) = 14/15$$



- What is job scheduling?
- Single machine problems and results
- Makespan problems on parallel machines
- Utilization problems on parallel machines
- Completion time problems on parallel machines
- Exemplary workload problem



■ $P_m \parallel \sum C_j$ is easy.

➔ Shortest processing time (SPT) (Conway, Maxwell, Miller, 1967)

➔ Single machine proof:

- $\sum C_j = n p_{(1)} + (n-1) p_{(2)} + \dots + 2 p_{(n-1)} + p_{(n)}$

- $p_{(1)} \leq p_{(2)} \leq p_{(3)} \leq \dots \leq p_{(n-1)} \leq p_{(n)}$ must hold for an optimal schedule.

➔ Parallel identical machines proof:

- Dummy jobs with processing time 0 are added until n is a multiple of m .

- The sum of the completion time has n additive terms with one coefficient each:

m coefficients with value n/m

m coefficients with value $n/m - 1$

⋮

m coefficients with value 1

- If there is one coefficient $h > n/m$ then there must be a coefficient $k < n/m$.

- Then we replace h with $k+1$ and obtain a smaller $\sum C_j$.

■ $P_m |prmp| \sum C_j$ is easy (Shortest remaining processing time).



- $P_m \parallel \sum w_j C_j$ is strongly NP-hard.
 - ➔ The WSPT algorithm has a tight approximation factor of 1.207 (Kawaguchi, Kyan, 1986)
 - ➔ It is sufficient to consider instances where all jobs have the same ratio w_j/p_j .
 - ➔ Proof by induction on the number of different ratios.
 - J is the set of all jobs with the largest ratio in an instance I .
 - The weights of all jobs in J are multiplied by a positive factor $\varepsilon < 1$ such that those jobs now have the second largest ratio.
 - This produces instance I' .
 - The WSPT order is still valid.
 - The WSPT schedule remains unchanged.
 - The optimal schedule may change.



■ Induction Proof

- $\sum w_j C_j(\text{WSPT}, I') \leq \lambda \cdot \sum w_j C_j(\text{OPT}, I')$ (induction assumption)
- x : contribution of all jobs in J to $\sum w_j C_j(\text{WSPT}, I)$
- y : contribution of all jobs not in J to $\sum w_j C_j(\text{WSPT}, I)$
- x' : contribution of all jobs in J to $\sum w_j C_j(\text{OPT}, I)$
- y' : contribution of all jobs not in J to $\sum w_j C_j(\text{OPT}, I)$
- $x \leq \lambda \cdot x'$ (induction assumption)
- $\sum w_j C_j(\text{WSPT}, I) = x + y$ and $\sum w_j C_j(\text{WSPT}, I') = \varepsilon \cdot x + y$,
- $\sum w_j C_j(\text{OPT}, I) = x' + y'$ and $\sum w_j C_j(\text{OPT}, I') \leq \varepsilon \cdot x' + y'$
- $y \leq \lambda \cdot y' \rightarrow \sum w_j C_j(\text{WSPT}, I) \leq \lambda \cdot \sum w_j C_j(\text{OPT}, I)$
- $y > \lambda \cdot y' \rightarrow \lambda \cdot x' y > x \cdot \lambda \cdot y' \rightarrow x'/y' > x/y \rightarrow x'y - xy' > 0 \rightarrow x'y - xy' > \varepsilon(x'y - xy')$
- $\frac{\sum w_j C_j(\text{WSPT}, I') \cdot \sum w_j C_j(\text{OPT}, I)}{\sum w_j C_j(\text{OPT}, I') \cdot \sum w_j C_j(\text{WSPT}, I)} = \frac{(\varepsilon \cdot x + y)(x' + y')}{(\varepsilon \cdot x' + y')(x + y)} \geq 1$
- $\sum w_j C_j(\text{WSPT}, I) \leq \lambda \cdot \sum w_j C_j(\text{OPT}, I)$

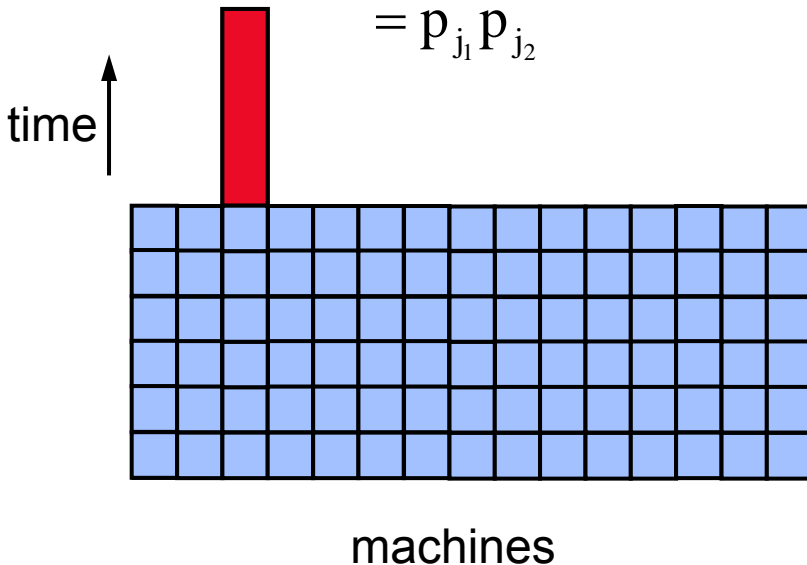
■ Assumption: $w_j = p_j$ holds for all jobs j .



■ Transformation of the job system

- ➔ Splitting of job j into jobs j_1 and j_2 .
- ➔ The system only contains short and long jobs.
 - All long jobs start at the end of busy interval in the list schedule.

$$\begin{aligned} \sum w_j C_j(S') - \sum w_j C_j(S) &= p_j C_j(S') - p_{j_1} C_{j_1}(S) - p_{j_2} C_{j_2}(S) = \\ &= (p_{j_1} + p_{j_2}) \cdot C_j(S') - p_{j_1} \cdot (C_j(S') - p_{j_2}) - p_{j_2} C_j(S') = \\ &= p_{j_1} p_{j_2} \end{aligned}$$



$$\begin{aligned} \frac{\sum w_j C_j(S)}{\sum w_j C_j(\text{OPT})} &\geq \frac{\sum w_j C_j(S') - p_{j_1} p_{j_2}}{\sum w_j C_j(\text{OPT}') - p_{j_1} p_{j_2}} \geq \\ &\geq \frac{\sum w_j C_j(S')}{\sum w_j C_j(\text{OPT}')} \end{aligned}$$



■ Single machine without intermediate idle time

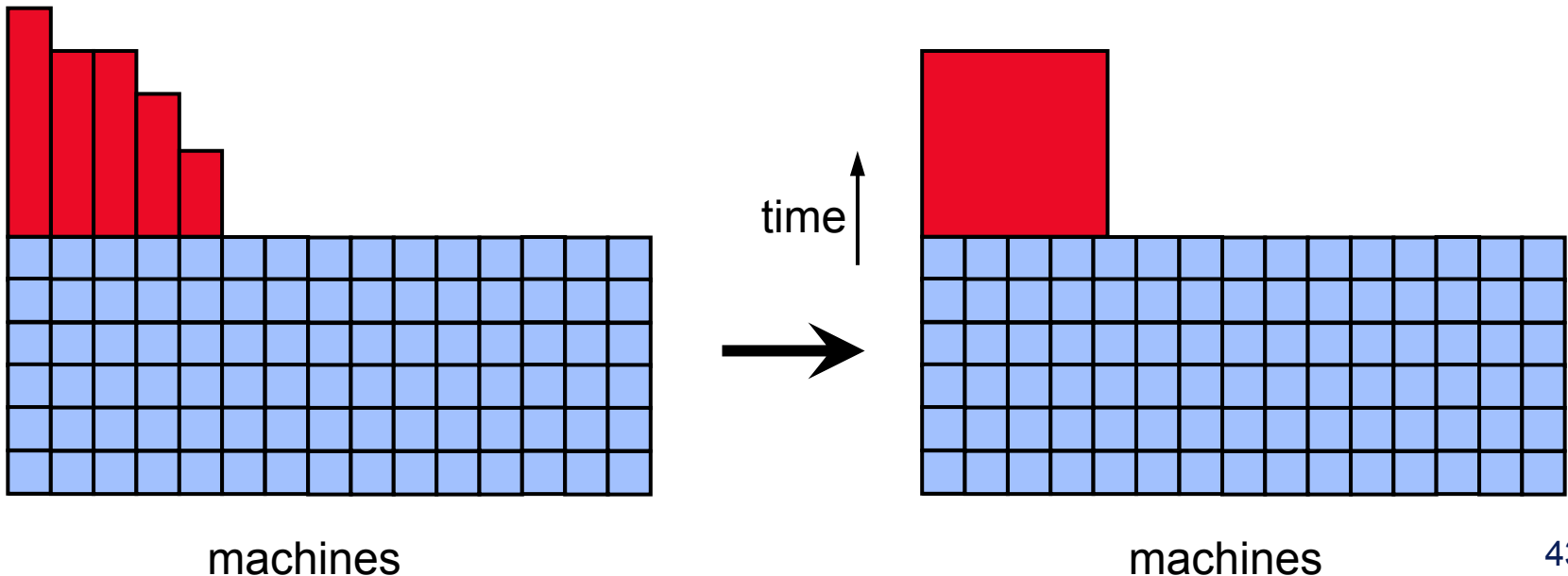
- $w_j = p_j$ holds for all jobs.
- $\sum w_j C_j(S) = \sum w_j C_j(OPT) = 0.5((\sum p_j)^2 + \sum p_j^2)$
- Proof by induction on the number of jobs

$$\begin{aligned}\sum w_j C_j(S) &= \frac{1}{2} \left((\sum p_j)^2 + \sum p_j^2 \right) + p_{j'} (p_{j'} + \sum p_j) = \\ &= \frac{1}{2} \left((\sum p_j)^2 + 2p_{j'} \sum p_j + p_{j'}^2 \right) + \frac{1}{2} (\sum p_j^2 + p_{j'}^2)\end{aligned}$$



■ Equalization of the long jobs

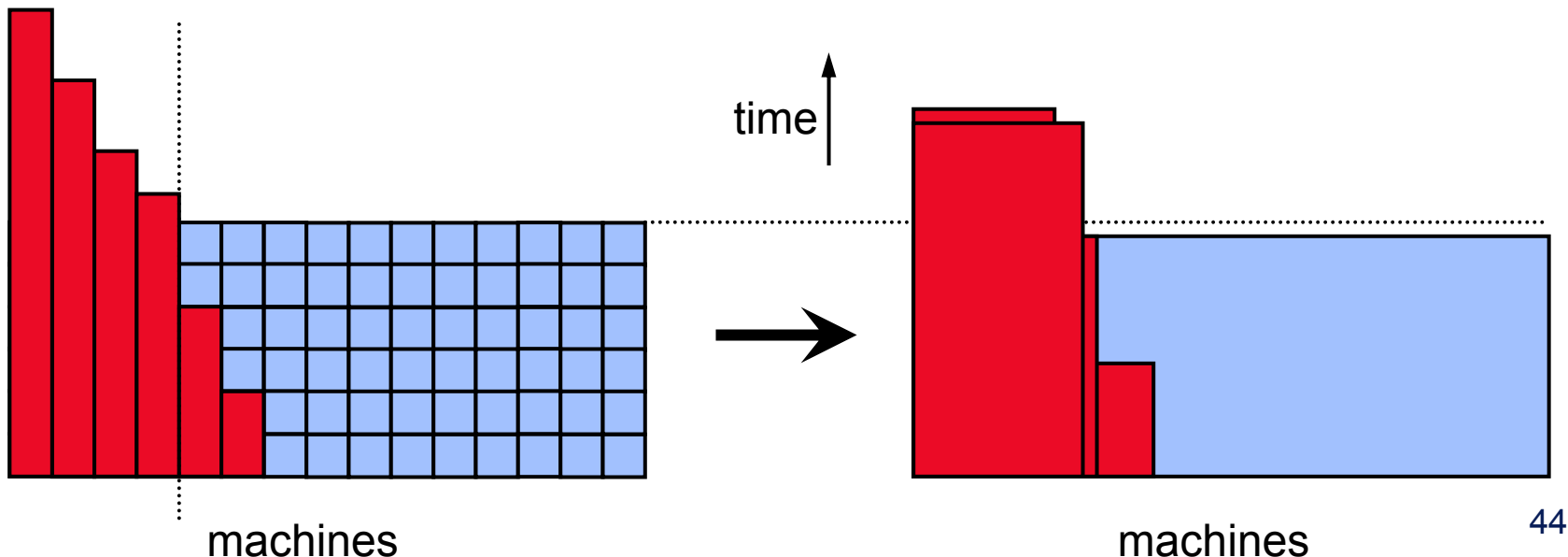
- Assumption of a continuous model (fraction of machines)
- k long jobs with different processing times are transformed into $n(k)$ jobs with the same processing time $p(k)$ such that $\sum p_j = n(k) \cdot p(k)$ and $\sum p_j^2 = n(k) \cdot (p(k))^2$ hold.
- $p(k) = \frac{\sum p_j^2}{\sum p_j}$ and $n(k) = \frac{(\sum p_j)^2}{\sum p_j^2}$
- Then we have $k \geq n(k)$ for reasons of convexity.





■ Modification of the job system

- ➔ Partitioning of the long jobs into two groups
- ➔ Equalization of the both groups separately
- ➔ The maximum completion time of the small jobs decreases due to the large rectangle.
- ➔ The jobs of the small rectangle are rearranged.
- ➔ New equalization of the large rectangle
- ➔ Determination of the size of the large rectangle





- $P_m |r_j| \Sigma C_j$
 - ➔ Approximation factor 2
 - ➔ Clairvoyant, randomized online scheduling: competitive factor 2
- $P_m |r_j, prmp| \Sigma C_j$
 - ➔ Approximation factor 2
 - ➔ Clairvoyant, randomized online scheduling: competitive factor 2
- $P_m |r_j| \Sigma w_j C_j$
 - ➔ Approximation factor 2
 - ➔ Clairvoyant, randomized online scheduling: competitive factor 2
- $P_m |r_j, prmp| \Sigma w_j C_j$
 - ➔ Approximation factor 2
 - ➔ Clairvoyant, randomized online scheduling: competitive factor 2
(all results Schulz, Skutella, 2002)



- $P_m |m_j, prmp| \Sigma w_j C_j$
 - ➔ Use of gang scheduling without any task migration
 - ➔ Approximation factor 2.37 (Schwiegelshohn, 2004)
- $P_m |m_j, prmp| \Sigma C_j$
 - ➔ Nonclairvoyant approximation factor $2 - 2/(n+1)$ if all jobs are malleable with linear speedup (Deng, Gu, Brecht, Lu, 2000).
- $P_m |m_j| \Sigma w_j C_j$
 - ➔ Approximation factor 7.11 (Schwiegelshohn, 2004)
 - ➔ Approximation factor 2 if $m_j \leq 0.5m$ holds for all jobs (Turek et al., 1994)
- $P_m |m_j| \Sigma C_j$
 - ➔ Approximation factor 2 if the jobs are malleable without superlinear speedup (Turek et al., 1994)



- $P_m |m_j, r_j, prmp| \sum w_j C_j$
 - ➔ Nonclairvoyant online scheduling with gang scheduling and $w_j = m_j \cdot p_j$: competitive factor 3.562 (Schwiegelshohn, Yahyapour, 2000)
 - $w_j = m_j \cdot p_j$ guarantees that no job is preferred over another job regardless of its resource consumption as all jobs have the same (extended) Smith ratio.
 - All jobs are started in order of their arrival (FCFS).
 - Any job started after a job j can increase the flow time $C_j - r_j$ by at most a factor of 2
 - ➔ Clairvoyant online scheduling with malleable jobs and linear speedup:
 - Competitive factor $12 + \epsilon$ for a deterministic algorithm
 - Competitive factor 8.67 for a randomized algorithm (both results Chakrabarti et al., 1996)



- What is job scheduling?
- Single machine problems and results
- Makespan problems on parallel machines
- Utilization problems on parallel machines
- Completion time problems on parallel machines
- Exemplary workload problem



■ Machine model

- ➔ *Massively parallel processor (MPP):* m parallel identical machines

■ Job model

- ➔ Multiple independent users
- ➔ Nonclairvoyant (unknown processing time p_j) with estimates
- ➔ Online (submission over time r_j)
- ➔ Fixed degree of parallelism m_j during the whole processing
- ➔ No preemption

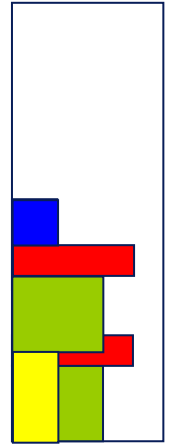
■ Objective

- ➔ Machine utilization
- ➔ Average weighted response time (*AWRT*): $p_j \cdot m_j \cdot (C_j - r_j)$
- ➔ Based on user groups



- Reordering of the waiting queue
 - ➔ Parameters of jobs in the waiting queue
 - ➔ Actual time
 - ➔ Scheduling situations: weekdays daytime (8am – 6pm), weekdays nighttime (6pm – 8am), weekends
- Selected sorting criteria
- Selected objective
 - ➔ Consideration of 2 user groups: $10 AWRT_1 + 4 AWRT_2$
- Parameter training with Evolution Strategies
 - ➔ Recorded workloads and simulations
 - ➔ Workload scaling for comparison

Waiting queue





Workloads and User Groups



User Group	1	2	3	4	5
RC _u /RC	> 8%	2 – 8 %	1 – 2 %	0.1 – 1 %	< 0.1 %

User group definition

Identifier	CTC	KTH	LANL	SDSC 00	SDSC 95	SDSC 96
Machine	SP2	SP2	CM-5	SP2	SP2	SP2
Period	06/26/96 – 05/31/97	09/23/96 – 08/29/97	04/10/94 – 09/24/96	04/28/98 – 04/30/00	12/29/94 – 12/30/95	12/27/95 – 12/31/96
Processors (<i>m</i>)	1024	1024	1024	1024	1024	1024
Jobs (<i>n</i>)	136471	167375	201378	310745	131762	66185

Workload scaling



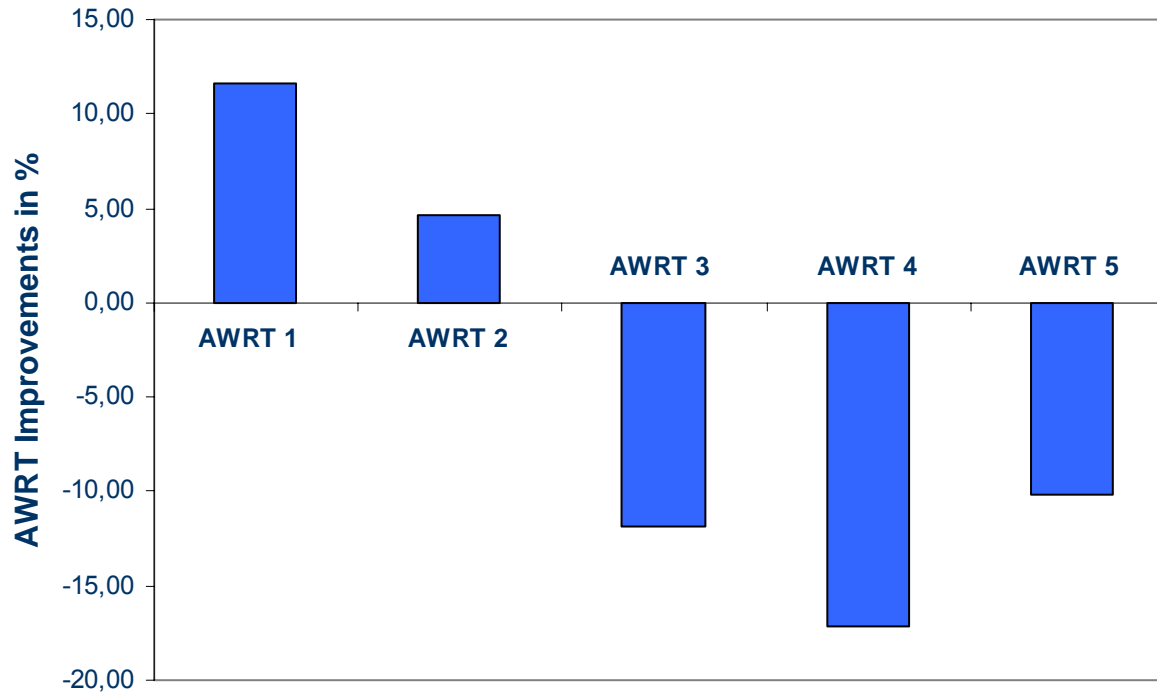
$$f_1(\text{Job}) = \sum_{i=1}^{|\text{Groups}|} w_i \cdot \left(K_i + a \cdot \frac{\text{waitTime}}{\text{requestedTime}} + b \cdot \frac{\text{requestedTime}}{\text{processors}} \right)$$

$$f_2(\text{Job}) = \sum_{i=1}^{|\text{Groups}|} w_i \cdot \left(K_i + a \cdot \text{waitTime} + b \cdot \frac{\text{requestedTime}}{\text{processors}} \right)$$

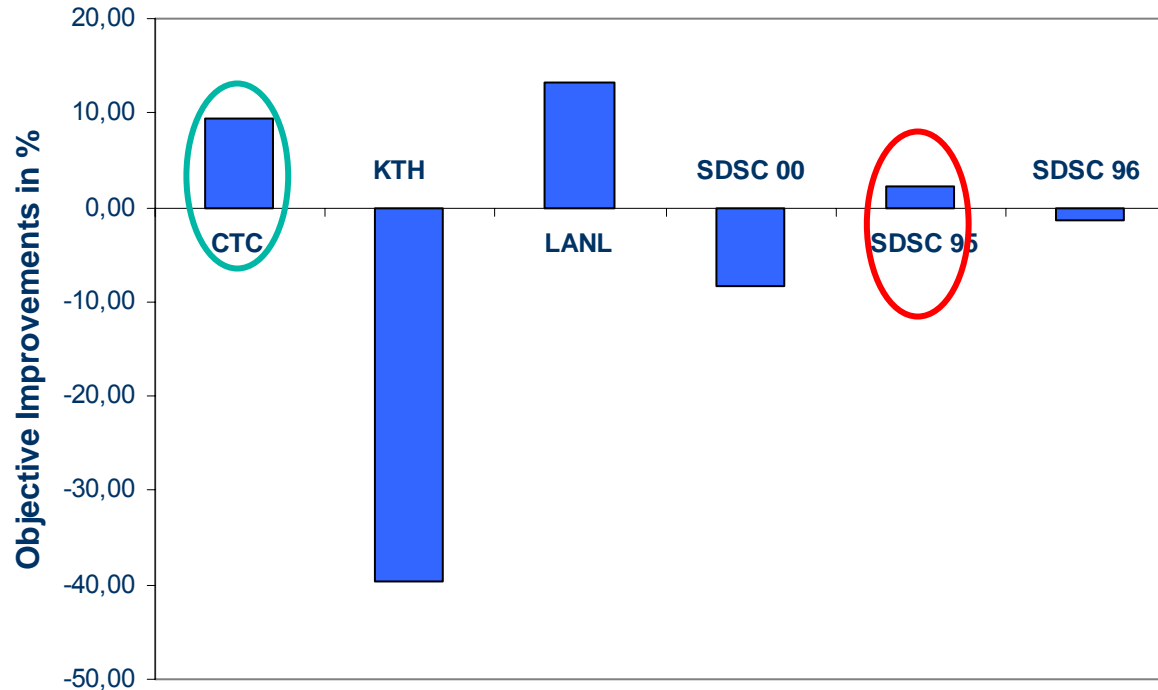
$$f_3(\text{Job}) = \sum_{i=1}^{|\text{Groups}|} w_i \cdot \left(K_i + a \cdot \frac{\text{waitTime}}{\text{requestedTime} \cdot \text{processors}} \right)$$

$$f_4(\text{Job}) = \sum_{i=1}^{|\text{Groups}|} w_i \cdot (K_i + a \cdot \text{waitTime} + b \cdot \text{requestedTime} \cdot \text{processors})$$

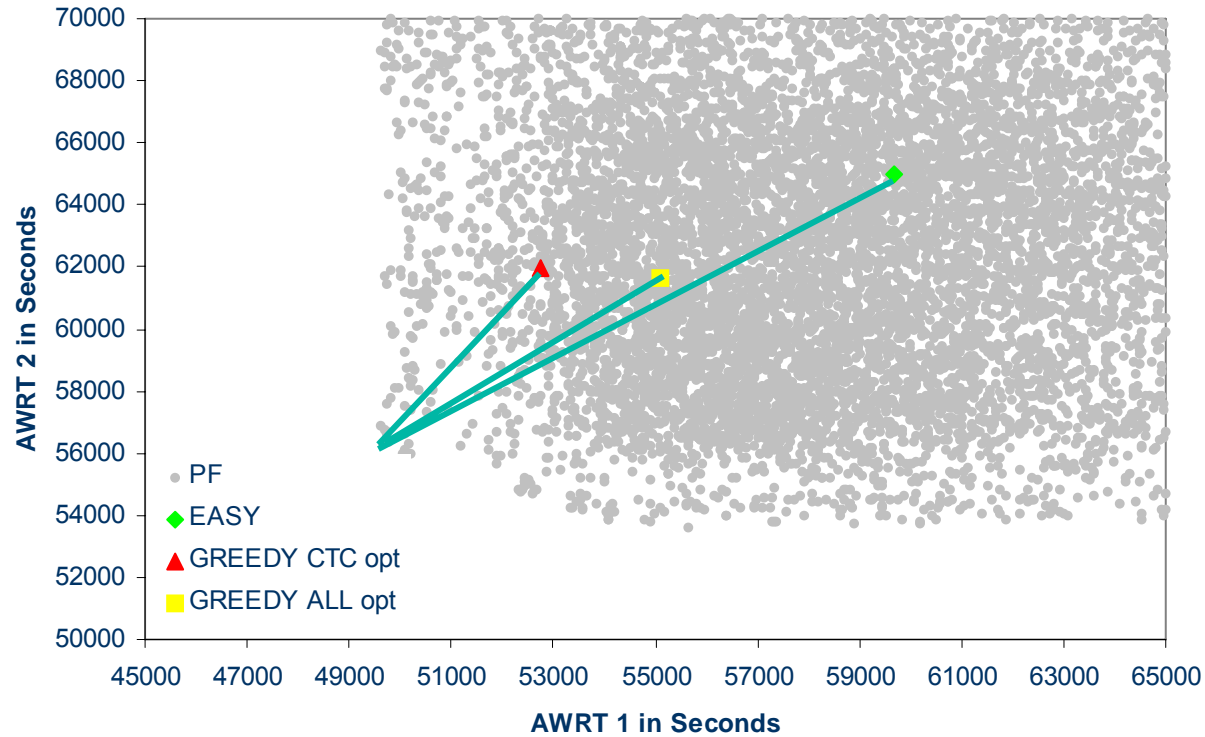
Training of parameters w_i , K_i , a , b with Evolution Strategies

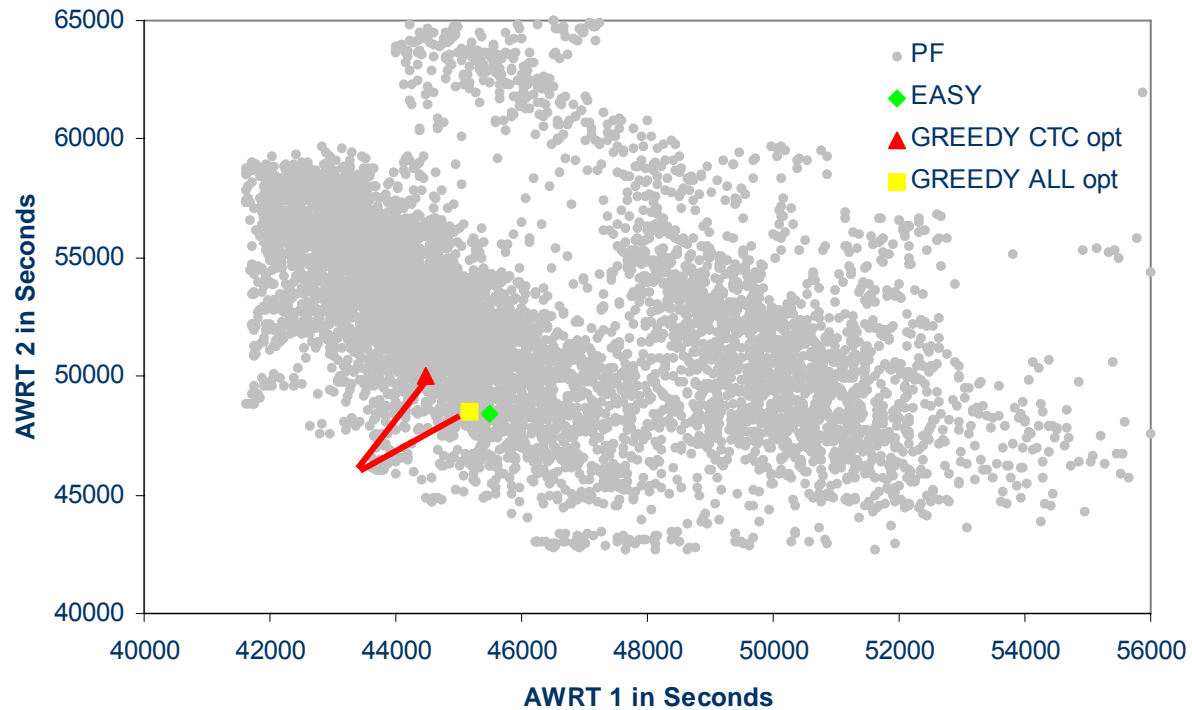


Method	AWRT 1	AWRT 2	AWRT 3	AWRT 4	AWRT 5	UTIL
GREEDY	52755.80 s	61947.65 s	56275.18 s	54017.23 s	35085.84 s	66.99 %
EASY	59681.28 s	64976.07 s	50317.47 s	46120.02 s	31855.68 s	66.99 %



- Some workloads are similar (CTC, LANL).
- Some workloads are significantly different (CTC, KTH).







- Most deterministic job scheduling problems are NP hard.
 - ➔ Approximation algorithms
 - Polynomial time approximation schemes
 - Simple algorithms
- Complete problem knowledge is rare in practice.
 - ➔ Online algorithms
 - Competitive analysis
 - ➔ Stochastic scheduling
 - Randomized algorithms
- Challenges
 - ➔ Partial information
 - Recorded workloads
 - User estimates
 - ➔ Scheduling objectives and constraints