technische universität dortmund

SFB 531

When does it pay to parallelize stochastic optimization algorithms?

Günter Rudolph
Technische Universität Dortmund
Fakultät für Informatik

SFB 559

10-APR-2008 @ Lorentz center

Outline

a) Real-world Background

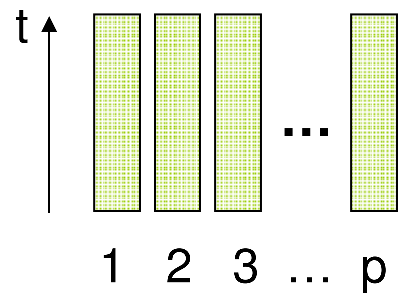
b) Model

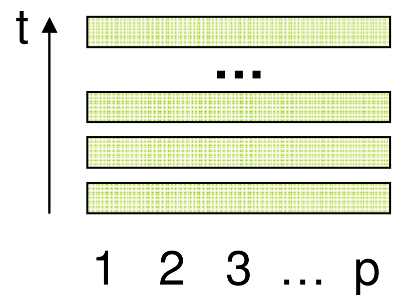
c) 1st Scenario

d) 2nd Scenario

e) Conclusions

stochastic optimizer must be run more than once!



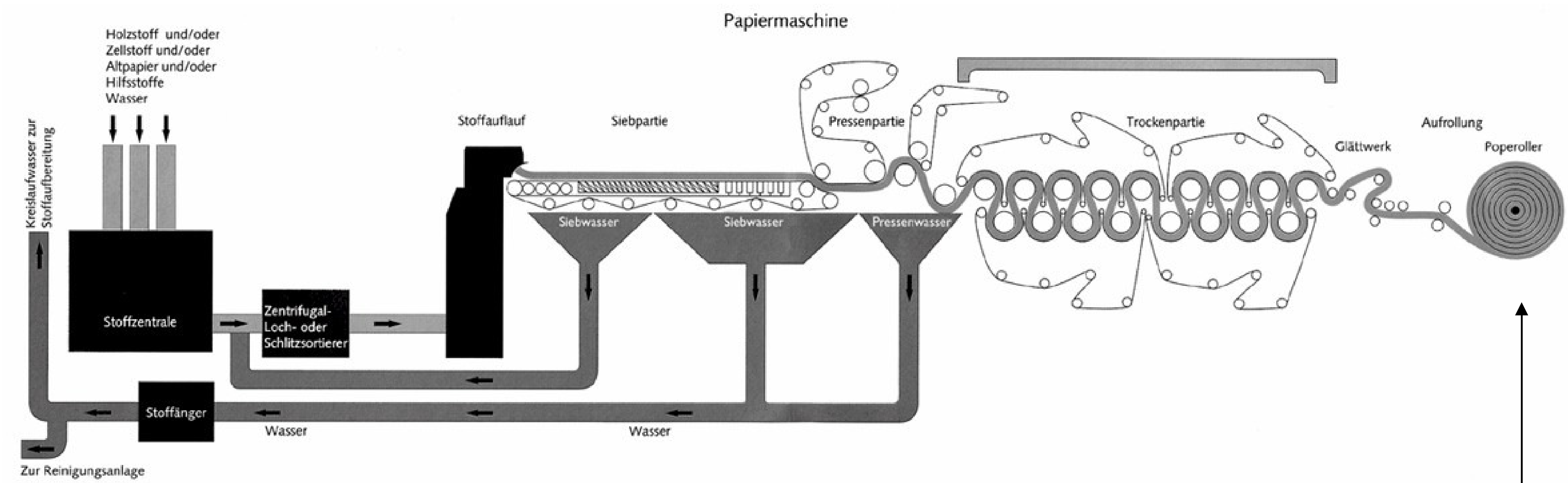


<u>two options:</u>

1. p runs of sequential code in parallel on p processors or
2. p consecutive runs of parallelized code on p processors

Consider continuous production process (24 hours a day, 365 days per year)

here: paper production

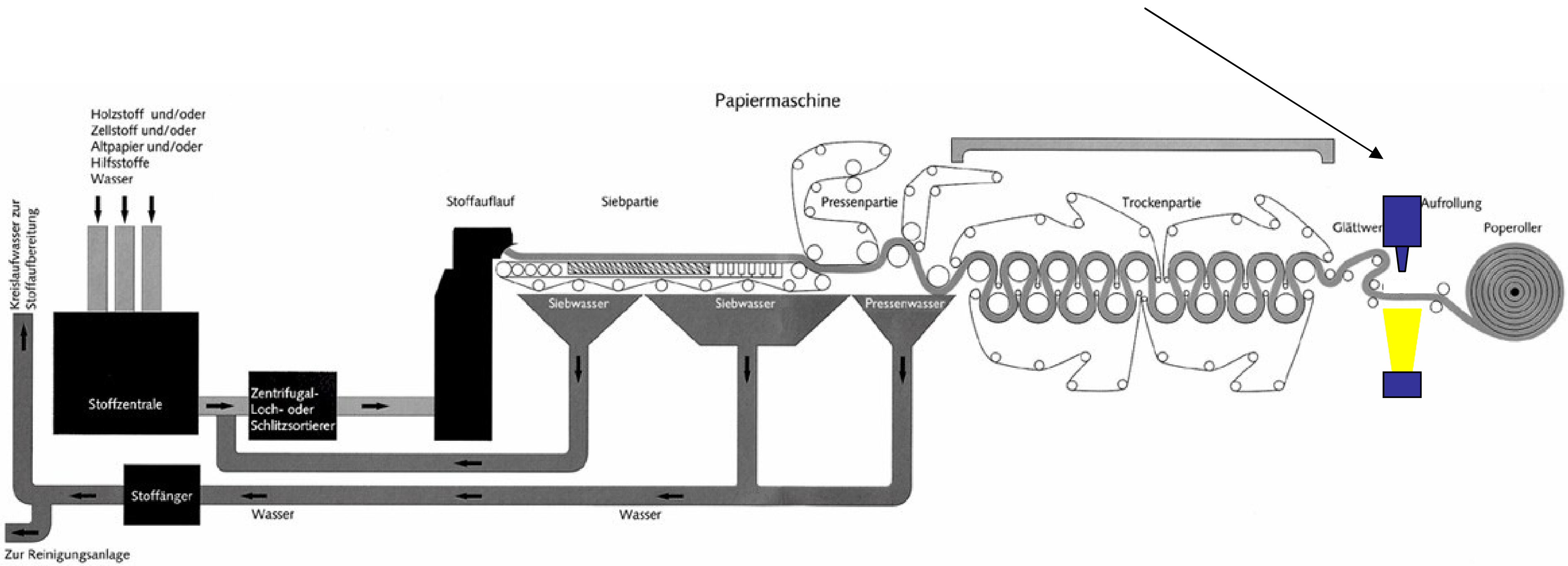


up to 2000 m/min.

weight up to 40 tons, length up to 80 km, width 2 to 10 meters

?

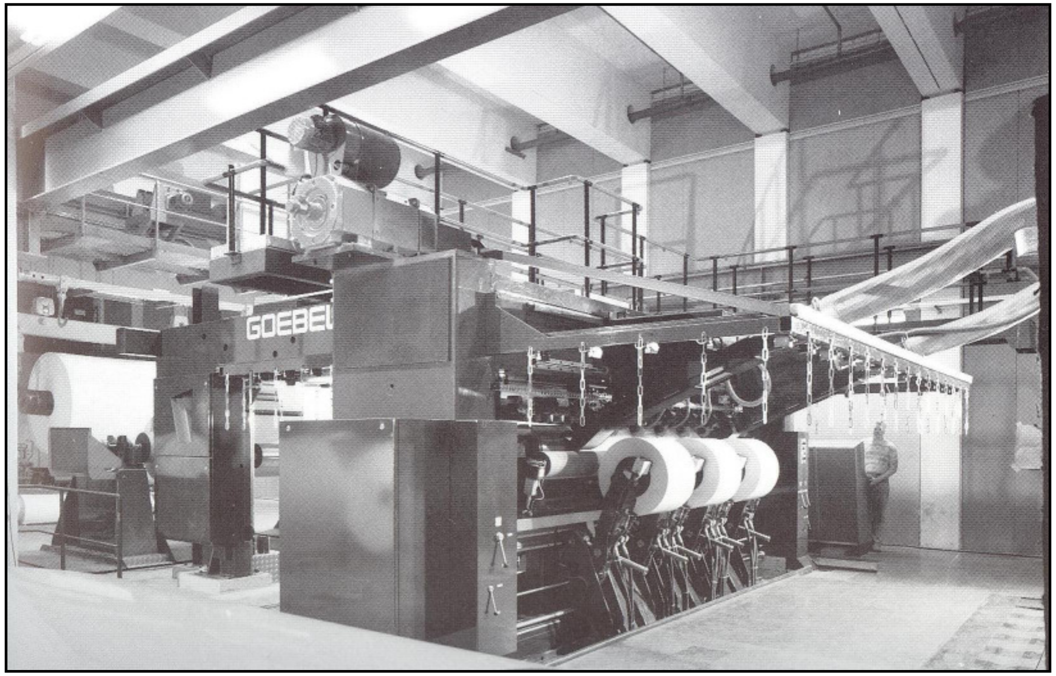
⇒ must be slit into rolls according to customers‘ requests (size & quality) optical defect inspection



high-speed cameras take pictures of every piece of paper

⇒ parallel processing for defect detection & classification

⇒ position and classification of defect stored in database time slot of few minutes until mother roll is moved to the slitter



optimization?

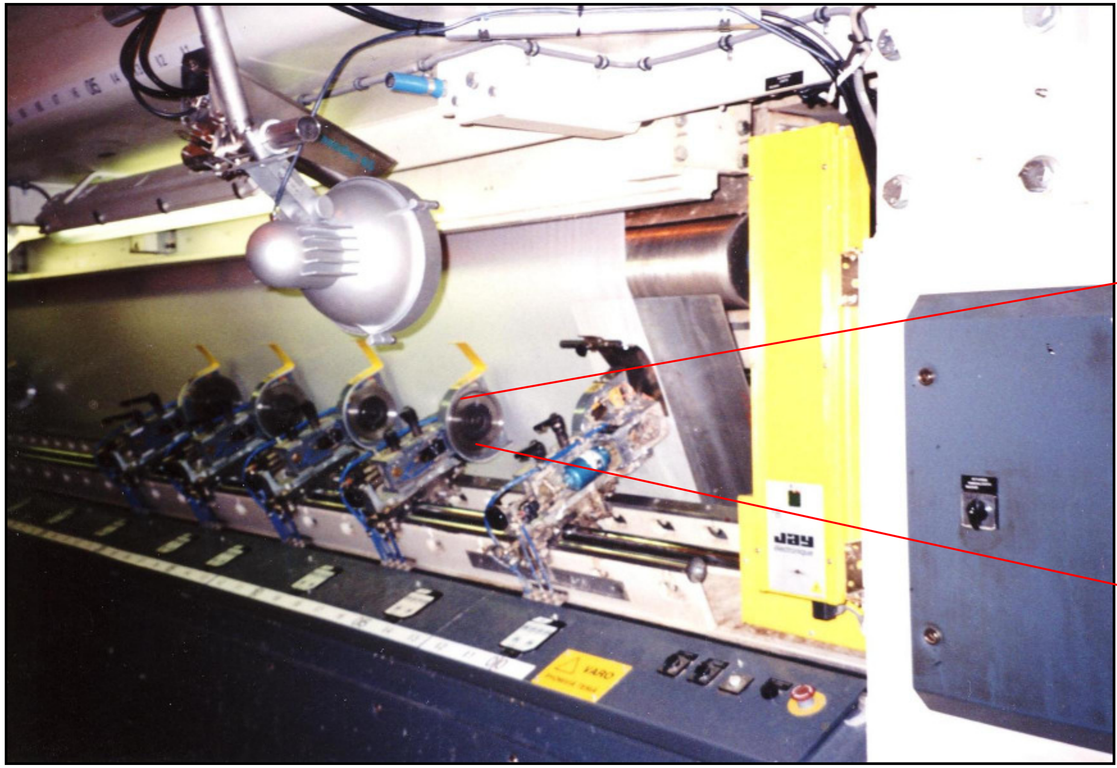
get orders from order database

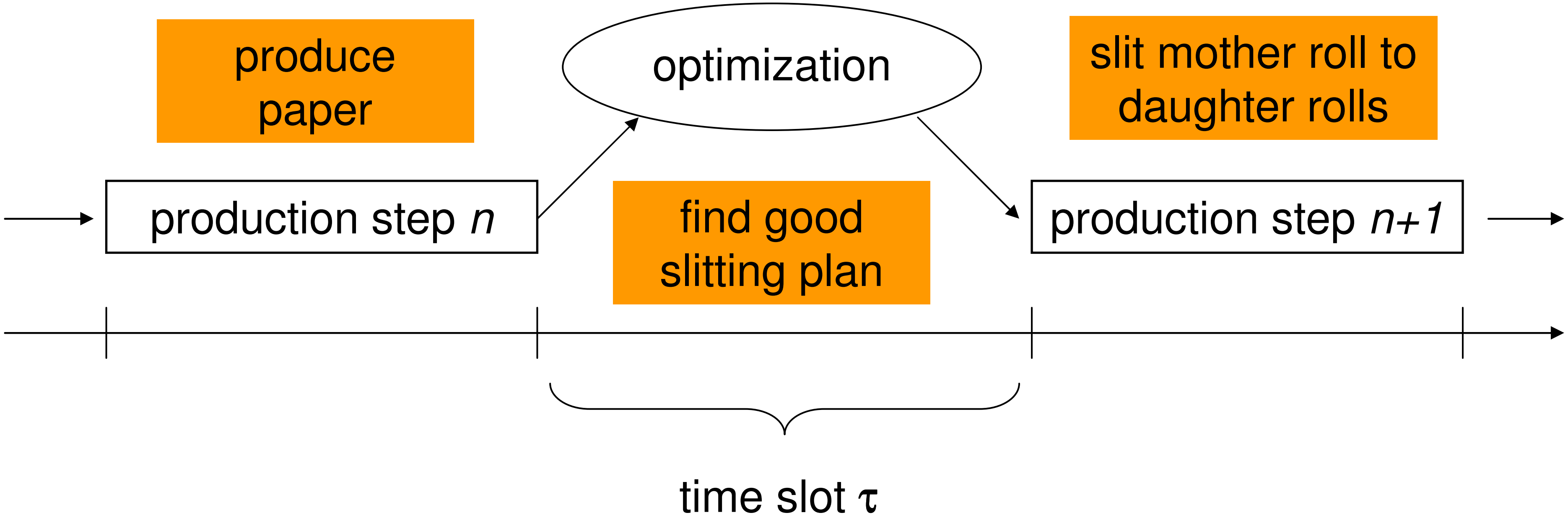
get defects from defect database

find optimal slitting plan: minimize waste and interim storage costs

⇒ you must have a slitting plan as soon as mother roll arrives at slitter!

blades can be positioned automatically by plant automation





<u>assumption:</u> 1 run of randomized optimizer requires t time units

if $t \leq \tau < 2\,t$ then you can run optimizer only once!

⇒ **use parallel hardware!**

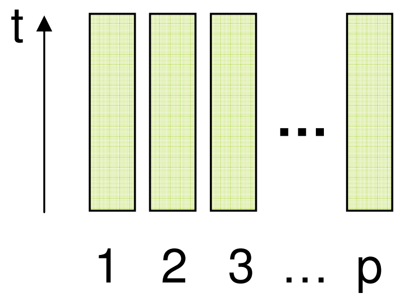
how?

- periodically occurring optimization task
- fixed time slot available for optimization
- best solution found within fixed time slot is used in production process

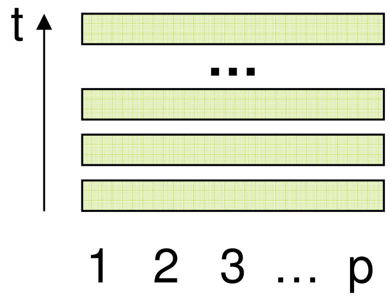
p processors available

<u>two options:</u>

1. p runs of sequential code in parallel on p processors or



2. p consecutive runs of parallelized code on p processors



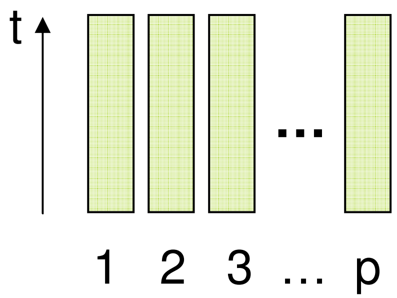
t	running time of sequential algorithm
$t_p = c \times \dfrac{t}{p}$	running time of parallelized algorithm
$c > 1$	aggregated communication costs etc
p	number of processors
n	maximum number of runs

assumption: $n = p$ since $t \le \tau < 2\,t$

total running time r

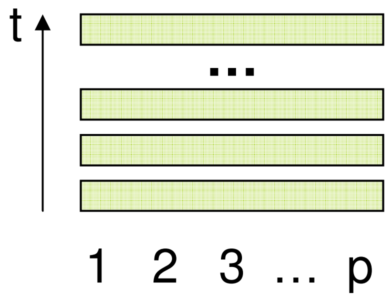
- p runs of sequential code:

$$r = t$$



- p runs of parallelized algorithm:

$$r_p = p \times t_p = p \times c \times \frac{t}{p} = c\,t$$



$\Rightarrow r = t < c \cdot t = r_p$ **⇒ don‘t parallelize your code!**

T	random running time of seq. algorithm
$T_p = c \times \dfrac{T}{p}$	random running time of par. algorithm
$c > 1$	aggregated communication costs etc.
p	number of processors
n	maximum number of runs

<u>modified scenario:</u> wait until n runs are completed (n = p)

total running time:

- p runs of sequential code:

$$R = \max\{\, T(1), T(2), \ldots, T(p) \,\} = T_{p:p}$$

comparison via expectation

- p runs of parallelized algorithm:

$$R_p = \sum_{i=1}^{p} T_p(i) = \frac{c}{p} \sum_{i=1}^{p} T(i)$$

$T(i)$ = random running time on processor i

$T_p(i)$ = random running time of run i

expected total running time:

ass. $T(i) \sim N(t, \sigma^2)$

- p runs of sequential code:

$$\mathsf{E}[\,R\,] = \mathsf{E}[\,T_{p:p}\,] \approx \mathsf{E}[\,T\,] + \mathsf{D}[\,T\,]\,\sqrt{2\,\log\,p}\,.$$

- p runs of parallelized algorithm:

$$\mathsf{E}[\,R_p\,] = \frac{c}{p}\,\mathsf{E}\left[\,\sum_{i=1}^{p} T(i)\,\right] = c\,\mathsf{E}[\,T\,]\,.$$

$$\mathsf{E}[\,R_p\,] < \mathsf{E}[\,R\,] \Leftrightarrow c < 1 + \frac{\mathsf{D}[\,T\,]}{\mathsf{E}[\,T\,]} \times \sqrt{2\,\log\,p}\,.$$

$$\mathsf{E}[\,R_p\,] < \mathsf{E}[\,R\,] \Leftrightarrow c < 1 + \frac{\mathsf{D}[\,T\,]}{\mathsf{E}[\,T\,]} \times \sqrt{2\ \log\ p}\,.$$

What does that mean?

Speedup $\mathcal{S}_p := \dfrac{t_1}{t_p}$

Efficiency $\mathcal{E}_p := \dfrac{\mathcal{S}_p}{p} = \dfrac{t_1}{p\,t_p} = \dfrac{t_1}{p\,c\,t_1\,/p} = \dfrac{1}{c}$

⇒ parallelized code is quicker in total the smaller is c

or: the better is efficiency of parallelization

or: the larger is variation of random running time T

Is result artifact of normal distribution?

Assumption: $T(i) \sim U(t - \varepsilon, t + \varepsilon)$

$$\mathsf{E}[\,R_p\,] < \mathsf{E}[\,R\,] \Leftrightarrow c < 1 + \frac{\mathsf{D}[\,T\,]}{\mathsf{E}[\,T\,]} \times \left(1 - \frac{2}{p+1}\right) \sqrt{3}$$

Example:

running time 40 to 60 sec., 9 processors

$$\Rightarrow \; c < 1 + \tfrac{4}{25} = 1{,}16$$

$$\Rightarrow \; \text{Efficiency} = \tfrac{1}{c} > \tfrac{25}{29} = 0{,}862\ldots \text{ required!}$$

Generalization:

$$\mathsf{E}[\,T\,] \le \mathsf{E}[\,T_{p:p}\,] \le \mathsf{E}[\,T\,] + \frac{p-1}{\sqrt{2\,p-1}}\,\mathsf{D}[\,T\,]$$

(David 80, p. 59 + 63)

$\Rightarrow \exists$ sublinear $g(\cdot)$:

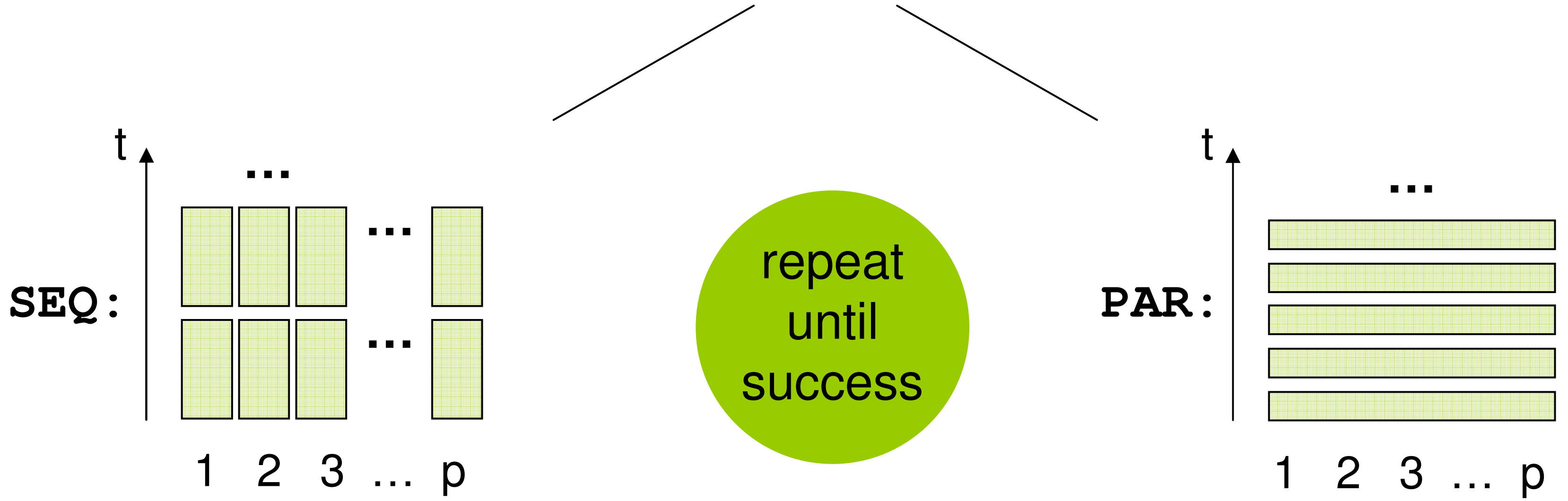
$$\mathsf{E}[\,T_{p:p}\,] = \mathsf{E}[\,T\,] + g(p)\,\mathsf{D}[\,T\,]$$

and hence

$$\mathsf{E}[\,R_p\,] < \mathsf{E}[\,R\,] \Leftrightarrow c < 1 + \frac{\mathsf{D}[\,T\,]}{\mathsf{E}[\,T\,]} \times g(p)$$

- periodically occurring optimization task
- no *hard* real-time constraints
- target quality bound must be exceeded
- repeat randomized optimizer until target quality is exceeded

p processors available ⇒ two options:



success probability (target exceeded) $s \in (0,1)$

r.v. G: #runs until first successful run

geometric distrib.: $\mathsf{P}\{\, G = k\,\} = s\,(1-s)^{k-1}$

$$\mathsf{E}[\,G\,] = \frac{1}{s} \quad \text{and} \quad \mathsf{D}^2[\,G\,] = \frac{1-s}{s^2}\,.$$

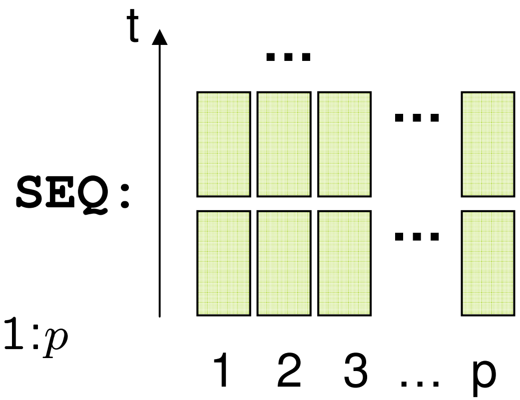
single processor:
random time until 1st successful run: $S = t\,G$

t	running time of sequential algorithm
$t_p = c \times \frac{t}{p}$	running time of parallelized algorithm
$c > 1$	aggregated communication costs etc
p	number of processors
s	success probability

random total running time:

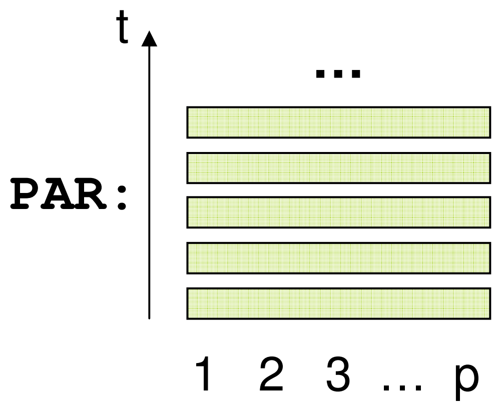
- repeated runs of sequential code on p processors:

$$R = \min\{\, S(1), S(2), \ldots, S(p)\,\} = S_{1:p} = t\, G_{1:p}$$



- G runs of parallelized algorithm:

$$R_p = t_p\, G = \frac{c}{p}\, t\, G$$



expected total running time:

- repeated runs of sequential code on p processors:

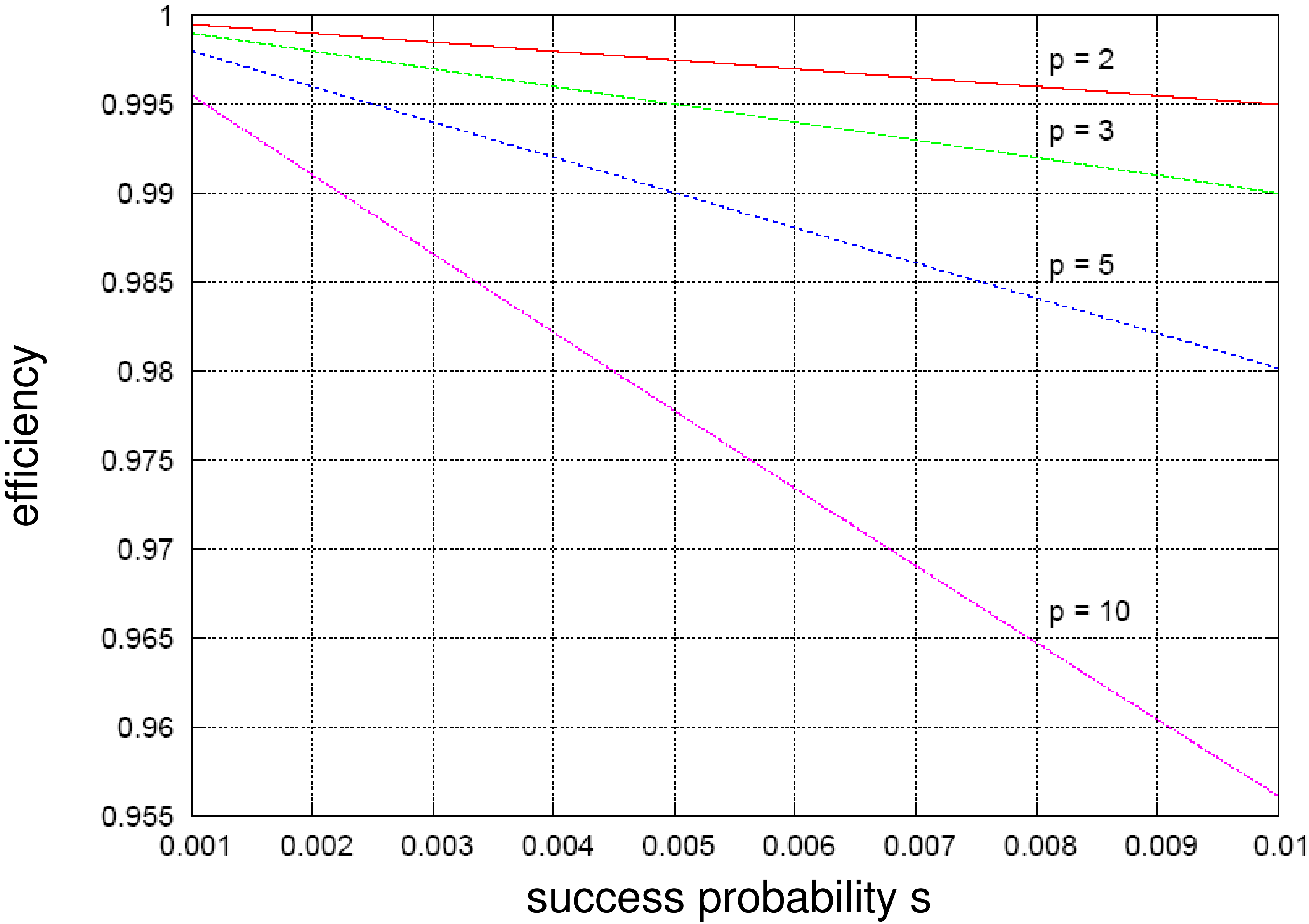
$$\mathsf{E}[\,R\,] = t\,\mathsf{E}[\,G_{1:p}\,] = \frac{t}{1-(1-s)^p}\,.$$

- G runs of parallelized algorithm:

$$\mathsf{E}[\,R_p\,] = \frac{c}{p}\,t\,\mathsf{E}[\,G\,] = \frac{c\,t}{s\,p}\,.$$

$$\boxed{\mathsf{E}[\,R_p\,] < \mathsf{E}[\,R\,] \Longleftrightarrow c < \frac{s\,p}{1-(1-s)^p}}$$

Constant Running Time



first successful run on single processor system: $S = \sum_{i=1}^{G} T(i)$

random total running time:

- repeated runs of sequential code on p processors:

$$R = \min\{ S(1), S(2), \ldots, S(p) \} = S_{1:p}$$

- G runs of parallelized algorithm:

$$R_p = \sum_{i=1}^{G} T_p(i) = \frac{c}{p} \sum_{i=1}^{G} T(i)$$

Theorem:

N positive, integer-values random variable

X_1, X_2, … sequence of i.i.d. random variables

a) $$\mathsf{E}\left[\sum_{k=1}^{N} X_k\right] = \mathsf{E}[\,N\,]\cdot\mathsf{E}[\,X_1\,]$$

b) $$\mathsf{V}\left[\sum_{k=1}^{N} X_k\right] = \mathsf{E}[\,N\,]\cdot\mathsf{V}[\,X_1\,] + \mathsf{V}[\,N\,]\cdot\mathsf{E}[\,X_1\,]^2$$

expected total running time:

- repeated runs of sequential code on p processors:

$$\mathsf{E}[\,R\,] = \mathsf{E}[\,S_{1:p}\,] < \mathsf{E}[\,S\,] = \mathsf{E}[\,T\,]\,\mathsf{E}[\,G\,]\,.$$

- G runs of parallelized algorithm:

$$\mathsf{E}[\,R_p\,] = \frac{c}{p}\,\mathsf{E}[\,T\,]\,\mathsf{E}[\,G\,] = \frac{c}{p}\,\mathsf{E}[\,S\,] = \frac{c\,t}{s\,p}\,.$$

$$\mathsf{E}[\,R_p\,] < \mathsf{E}[\,R\,] \quad \Leftrightarrow \quad \frac{c}{p}\,\mathsf{E}[\,S\,] < \mathsf{E}[\,S_{1:p}\,]$$

First approach:

$$\mathsf{E}[\,S\,] \geq \mathsf{E}[\,S_{1:p}\,] \geq \mathsf{E}[\,S\,] - \frac{p-1}{\sqrt{2\,p-1}}\mathsf{D}[\,S\,]$$ (David 80, p. 59 + 63)

$$\exists h : \mathbb{N} \to \mathbb{R}^{\geq 0} \text{ with } h(p) \geq \frac{p-1}{\sqrt{2\,p-1}} :$$

$$\mathsf{E}[\,S_{1:p}\,] = \mathsf{E}[\,S\,] - h(p)\,\mathsf{D}[\,S\,]$$

Wald: $\mathsf{E}[\,G\,] \cdot \mathsf{E}[\,T\,]$

Wald: $\sqrt{\mathsf{E}[\,G\,] \cdot \mathsf{V}[\,T\,] + \mathsf{V}[\,G\,] \cdot \mathsf{E}[\,T\,]^2}$

First approach:

$$\mathsf{E}[\,R_p\,] < \mathsf{E}[\,R\,] \quad\Leftrightarrow\quad \frac{c}{p}\,\mathsf{E}[\,S\,] < \mathsf{E}[\,S_{1:p}\,]$$

$$\Leftrightarrow\quad \frac{c}{p}\,\mathsf{E}[\,S\,] < \mathsf{E}[\,S\,] - h(p)\,\mathsf{D}[\,S\,]$$

$$\Leftrightarrow\quad c < p\left(1 - \frac{\mathsf{D}[\,S\,]}{\mathsf{E}[\,S\,]} \times h(p)\right)$$

Interpretation difficult!

Second approach:

<u>assumption:</u> each run T_i has minimum runtime a > 0, i.e., $T_i \geq a > 0$ w.p. 1

$$\mathsf{E}[\,S_{1:p}\,] = \min\{T_1\,G_1, \ldots, T_p\,G_p\} \geq a\,\mathsf{E}[\,G_{1:p}\,] \quad \Rightarrow$$

$$\mathsf{E}[\,R\,] = \mathsf{E}[\,S_{1:p}\,] \geq \frac{a}{1-(1-s)^p} \to a > 0 \quad \text{as} \quad p \to \infty$$

but:

$$\mathsf{E}[\,R_p\,] = \frac{c}{p}\,\mathsf{E}[\,G\,]\,\mathsf{E}[\,T\,] = \frac{c\,t}{s\,p} \to 0 \quad \text{as } p \to \infty$$

Second approach:

$$E[\,R_p\,] < E[\,R\,]$$

$$p \longrightarrow \infty \qquad E[\,R_p\,] \to 0, \quad E[\,R\,] \to a \qquad (a > 0)$$

As a consequence,

$$\exists\, p_0 < \infty : \forall p > p_0 : E[\,R_p\,] < E[\,R\,]$$

∃ situations in which parallelized code is advisable

1. fixed time slot & constant running time
   $\Rightarrow$ waste of ressources!

2. wait until completion & random running time
   $\Rightarrow$ high efficiency + large V[T] required

3. repeat until success & constant running time
   `if` success probability $\searrow$ $\Rightarrow$ necessary: efficiency $\nearrow$
   `if` #processors $\nearrow$ $\Rightarrow$ success probability may $\searrow$

4. repeat until success & random running time
   $\Rightarrow$ ∃ threshold on #processors: parallelized code faster in total (if >)

10th International Conference on

Parallel Problem Solving from Nature

PPSN X

September 13-17, 2008

Technische Universität Dortmund, Germany

www.ppsn2008.org

Conference Site:

Congress Center Westfalenhallen (same place as 1st PPSN 1990)

Important Dates:

Deadline	~~14.04.2008~~	**28.04.2008**
Conference	13.09.2008	Workshops
	14.09.2008	Tutorials
	15.09.2008	Technical Sessions
	16.09.2008	
	17.09.2008	