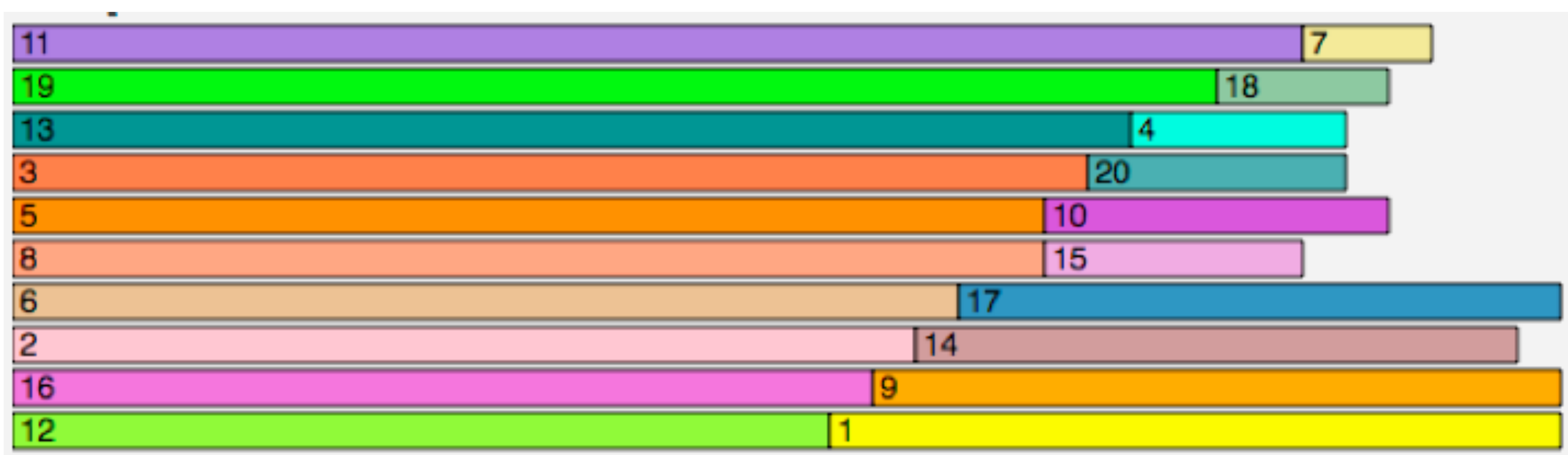
Approximating the Throughput by Coolest First Scheduling

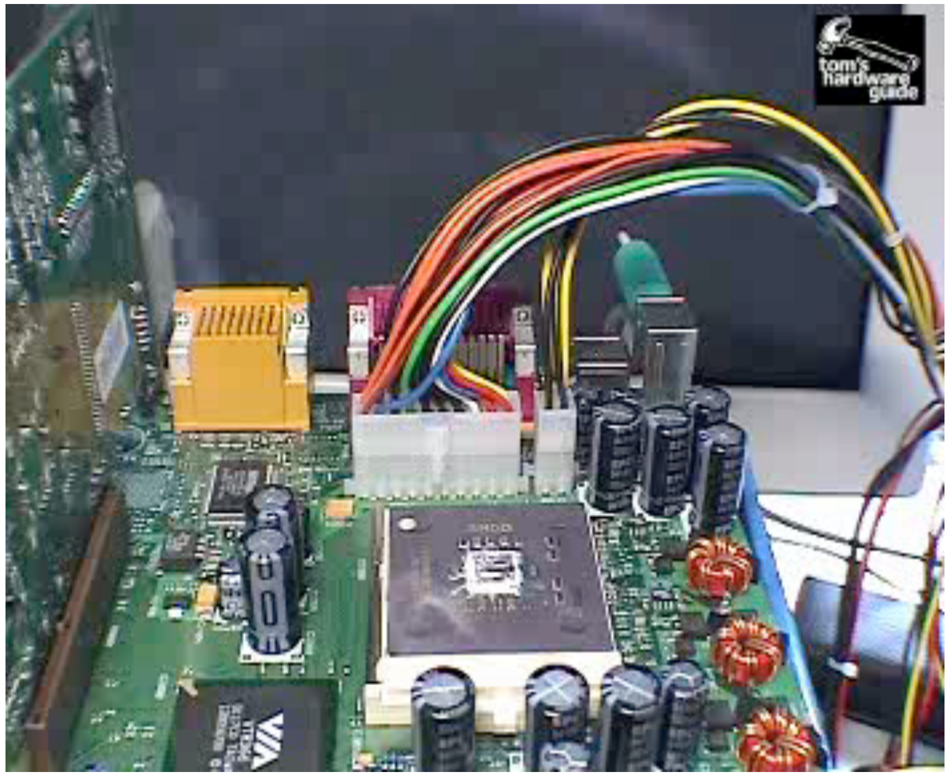
Christoph Dürr	CNRS, LIP6, U. Pierre et Marie Curie
Ioannis Milis	Athens Univ. of Eco. and Business
Julien Robert	LIFO, Univ. d'Orléans
Georgios Zois	LIP6 and AUEB

sep 2012 — WAOA


11
7
19
18
13
4
3
20
5
10
8
15
6
17
2
14
16
9
12
1


KENWOOD
KENWOOD
KENWOOD

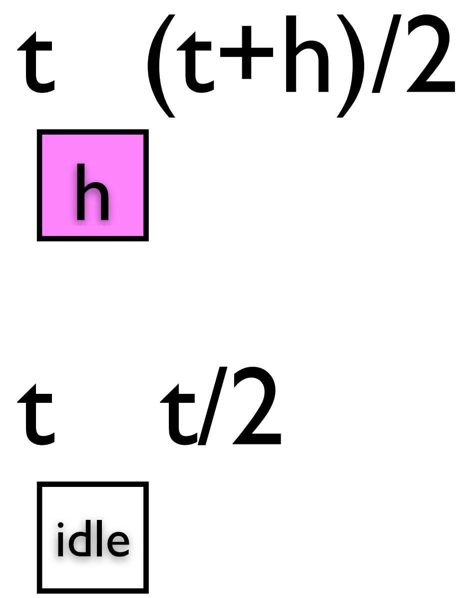
Motivation



When the CPU temperature exceeds some threshold

- some models freeze until CPU cooled down enough
- some models just start melting

The model



jobs have

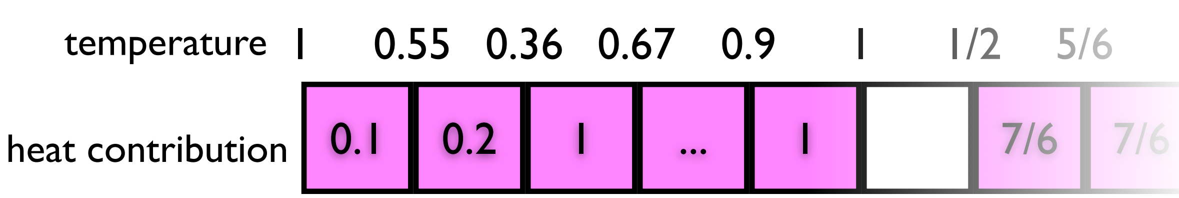
- unit processing time
- distinct heat contribution in [0,2]

CPU temperature

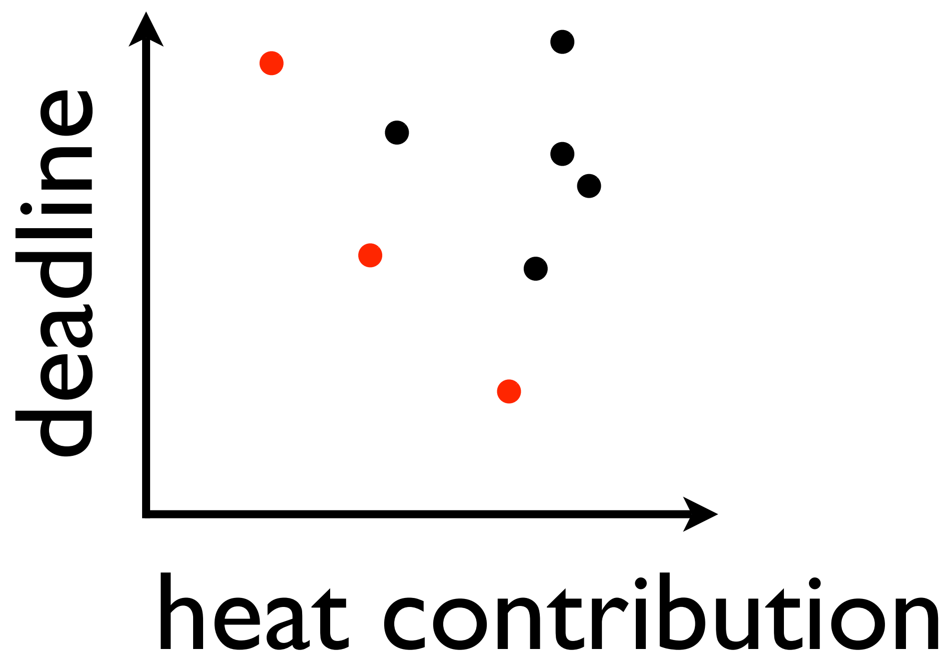
- is initially 1
- should never exceed 1

goal

- schedule a maximum number of jobs before (common) deadline D
- problem is NP-hard



What was known



more general on-line setting

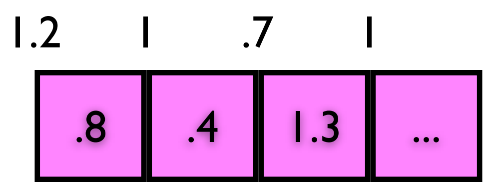
- jobs arrive online and have distinct deadlines, heat-contributions
- goal: maximize number of scheduled jobs respecting deadlines

[Chrobak,D,Hurand,Robert'08]

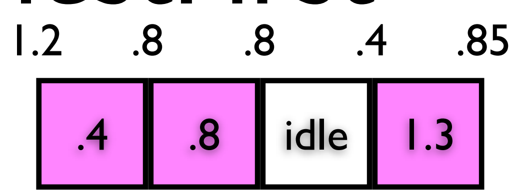
- scheduling any non-dominated job yields a 2-competitive on-line algorithm
- **how good are these algorithms in the offline setting?**

Minimizing makespan

HottestFirst



CoolestFirst



offline setting

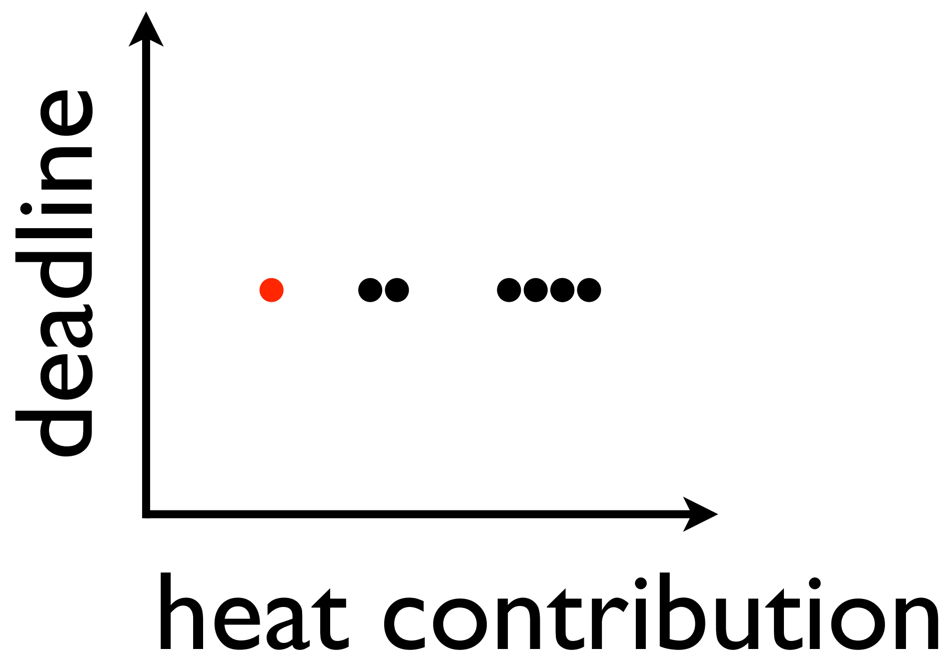
- all jobs are available at time 0 and have no deadline
- goal: produce a schedule respecting temperature threshold with smallest makespan

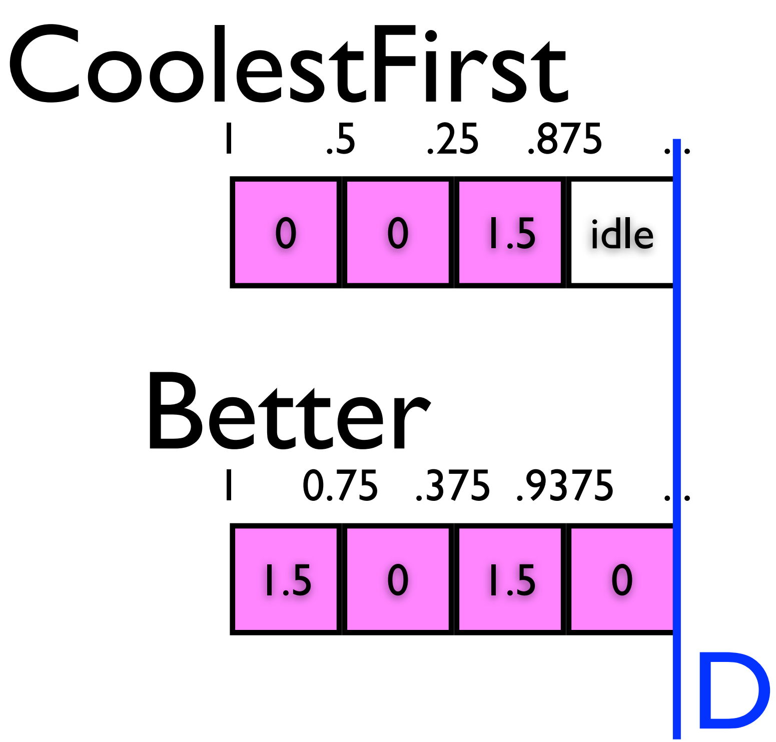
[Yang,Zhou,Chrobak,Zhang'08]

[Bampis,Letsios,Lucarelli,Markakis,Milis'12]

- *HottestFirst*: schedule at any moment the job with the largest heat contribution among jobs that would respect the temperature threshold
- seems to be a good heuristic to keep temperature low at later point and so avoid idle times

Maximizing throughput





offline setting

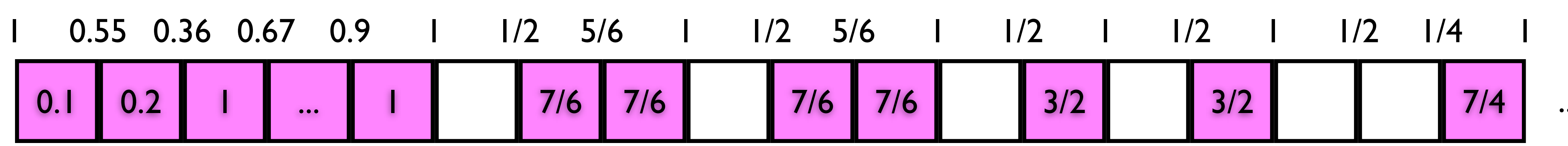
- all jobs are available at time 0
- all jobs have the same deadline D
- goal: schedule as many jobs before deadline D

Only the coolest job is non-dominated

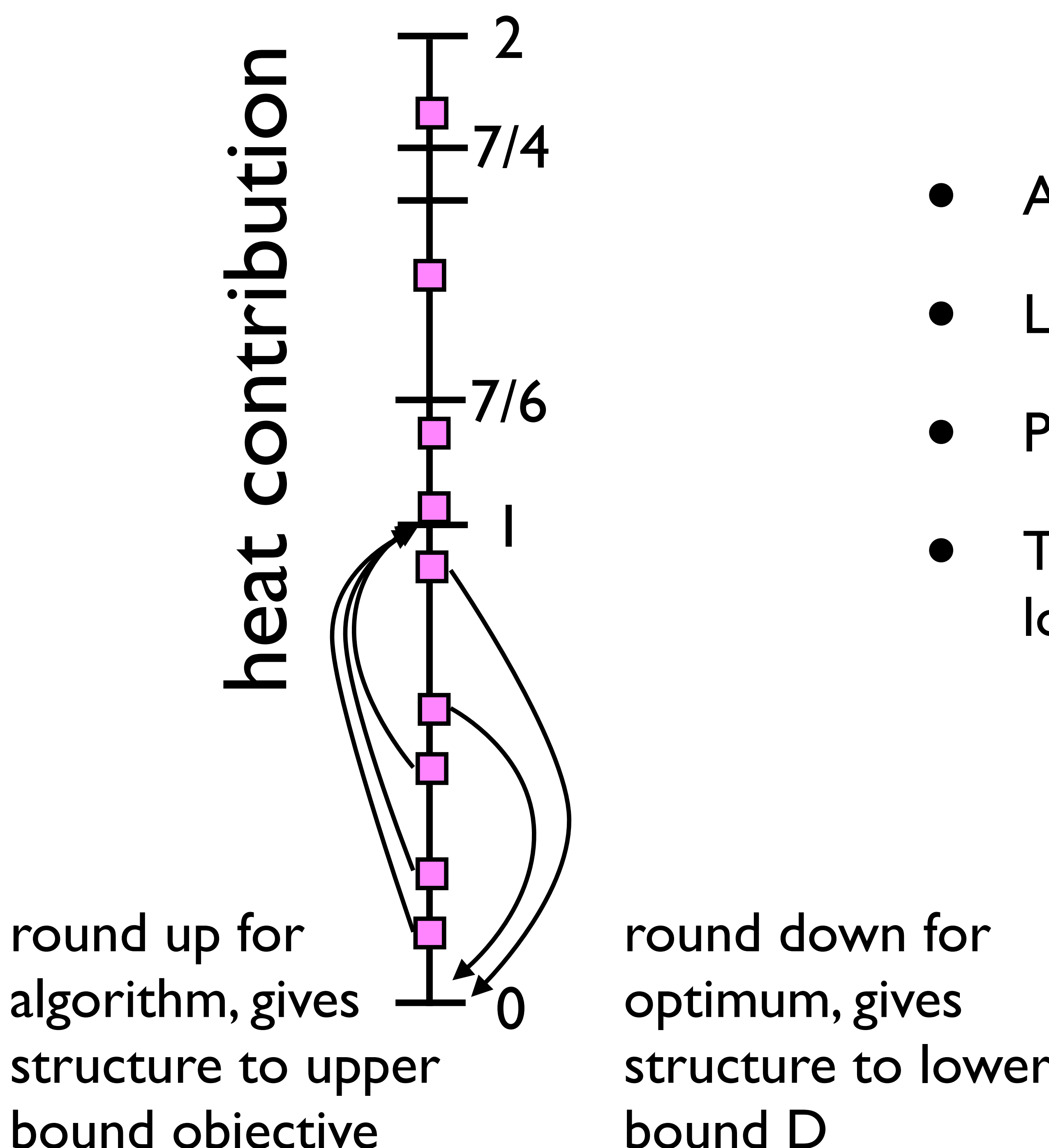
- *CoolestFirst*: schedule at any moment the job with the smallest heat contribution among allowed jobs
- It would be more clever to give priority to jobs with heat contribution ∈(1,2] since jobs from [0,1] can be scheduled anytime. But then we don't know how to analyse.

Analyze Coolest First

- **Input:** n jobs with different heat contributions ∈[0,2] and a common deadline D
- **CoolestFirst:** At any time schedule the one with smallest heat contribution among the jobs that would respect the temperature threshold
- **typical behavior:** schedule get’s less and less dense



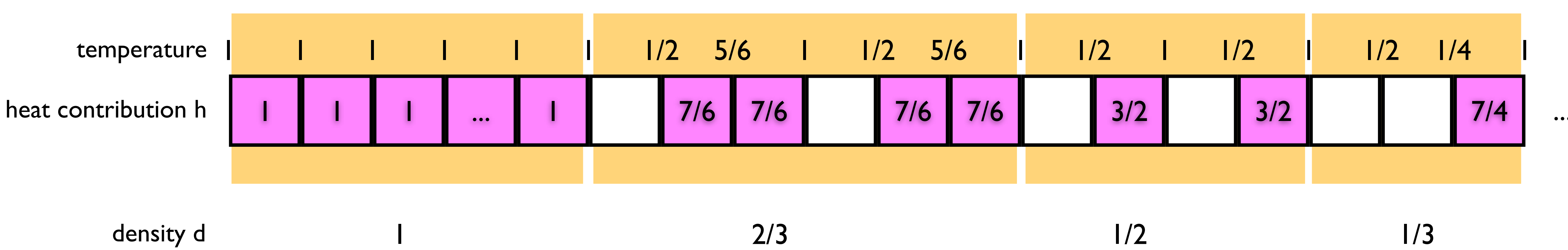
Method of analysis



- Assume optimum schedules *all* jobs
- Let D be the optimal makespan
- Partition [0,2] into job classes
- This pessimistic rounding permits to lower bound approximation ratio

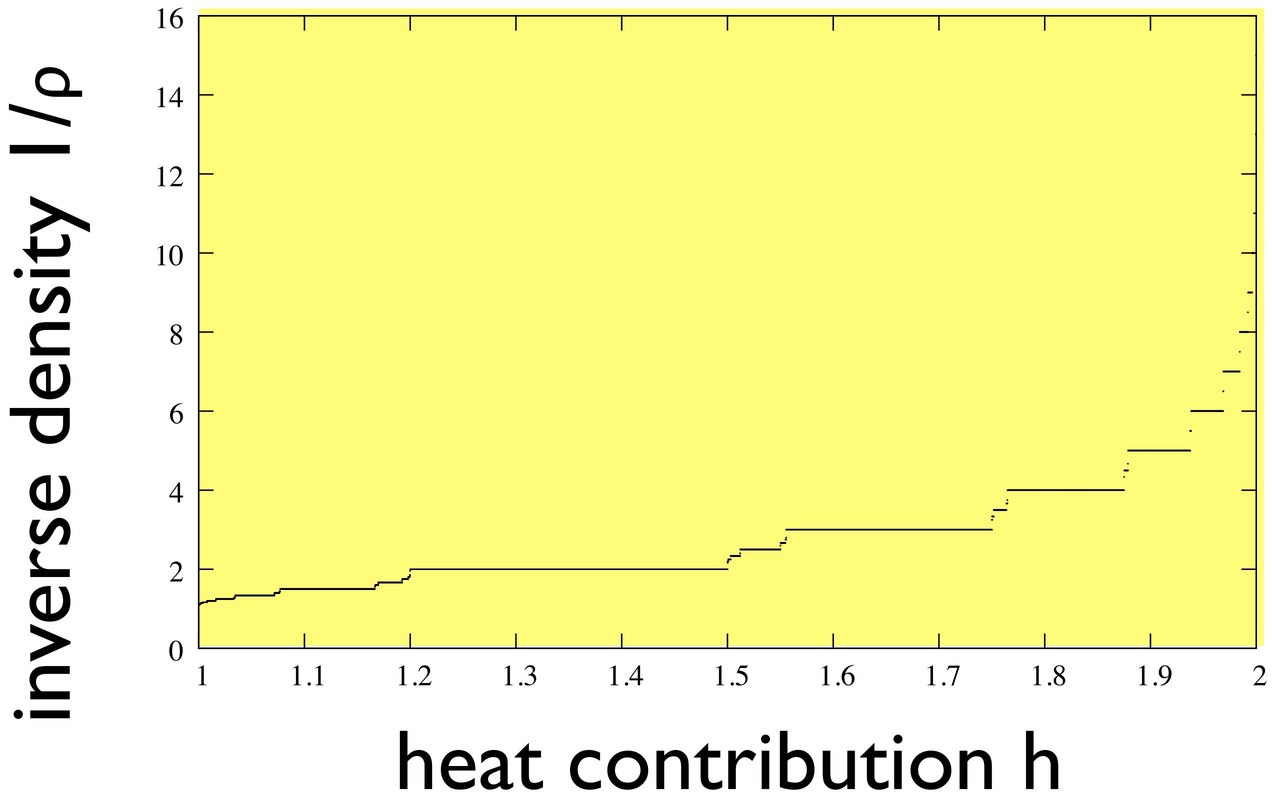
Density

- the output for the rounded instance, consist of blocks of jobs of same heat contribution **h**
- every block has some density **ρ**:=average #jobs per slot
- what is the relation between **h** and **ρ**?



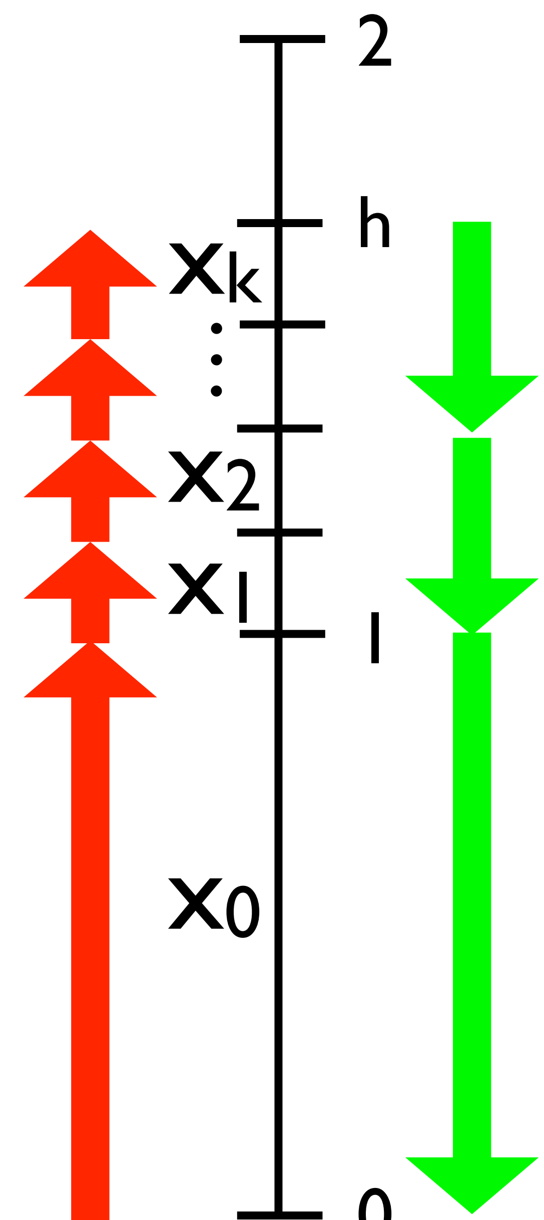
Relation between h and ρ

- experiments indicate it's monotone
  (as one would expect)
- confirmed by a proof:
  ∀rational ρ ∃h_ρ:
  $G(h_\rho)$ has density ρ
  and h_ρ is decreasing in ρ



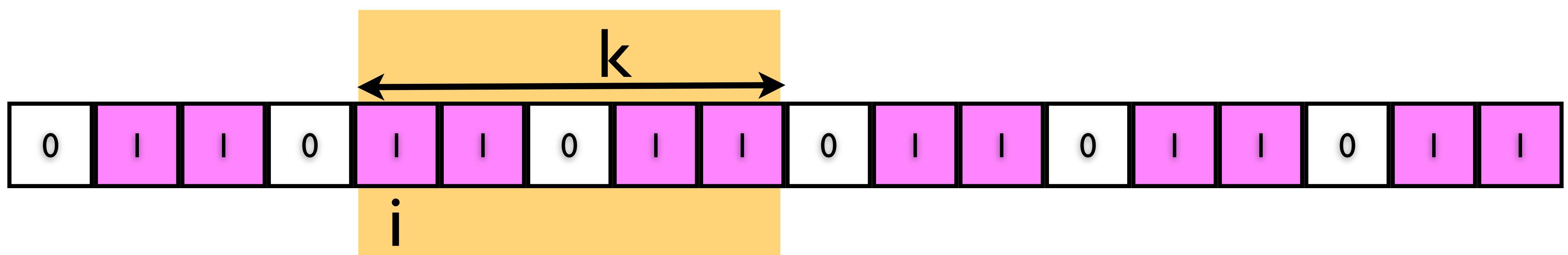
Structure of the analysis

- Suppose last job scheduled by CoolestFirst has heat contribution h
- Partition [0,2] into s classes
- Round heat contributions up for algorithm
- For every q=1,…,k-1 round jobs down for the optimum
- Find worst case instance described by number of jobs from each class $x_1,\dots,x_k$ boils down in solving a linear program
- We provide a solution to the dual linear program, bounding the primal
- For the worst choice of h, this shows approximation ratio of CoolestFirst ≥ 0.72
- In contrast worst case example has ratio=0.75



Properties of sequence

- $G : h \mapsto$ infinite 01-sequence w describing output of CoolestFirst on infinite h-jobs
- exists ρ, such that for all i,k *sliding window property*
  $\mathrm{floor}(\rho k) \leq \sum_{j=i}^{i+k-1} w_j \leq \mathrm{ceil}(\rho k)$

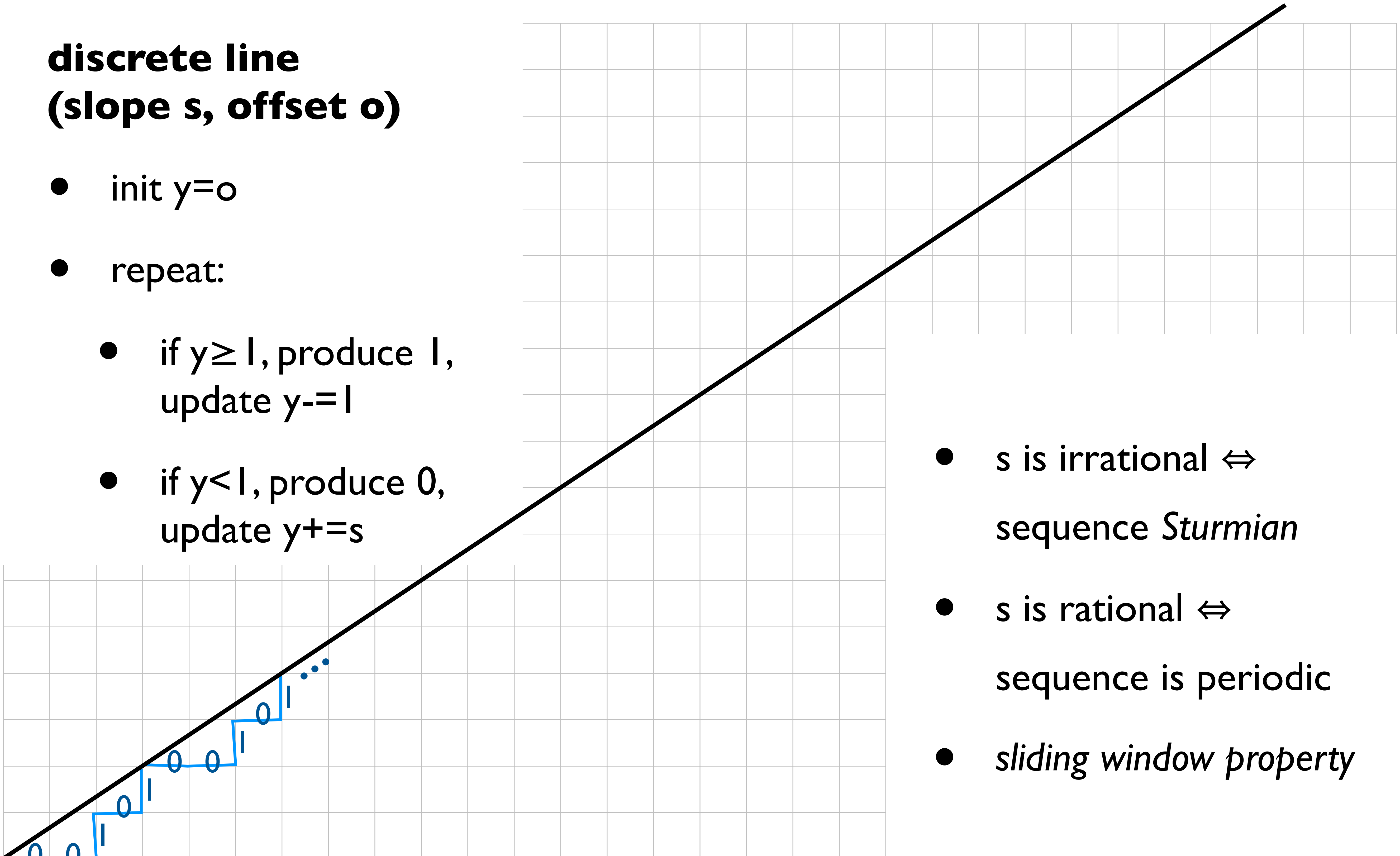


- is periodic (if h is rational)
- sounds familiar?

Discretization of line

discrete line
(slope s, offset o)

- init y=o
- repeat:
  - if y≥1, produce 1, update y-=1
  - if y<1, produce 0, update y+=s



- s is irrational ⇔ sequence *Sturmian*
- s is rational ⇔ sequence is periodic
- *sliding window property*

Seem so different

discrete line
(slope s, offset o)

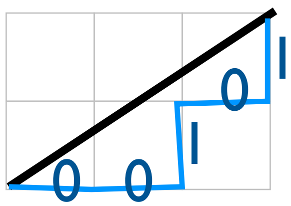
- init y=o
- repeat:
  - if $y \geq 1$, produce 1, update y-=1
  - if $y < 1$, produce 0, update y+=s
- different slopes produce different sequences
- add

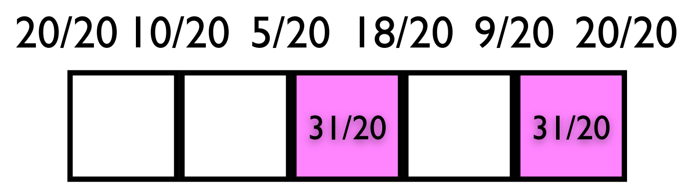
Greedy
(init temperature t, h)

- init t
- repeat:
  - if $t+h \leq 2$, produce 1, update $t=(t+h)/2$
  - if $t+h > 2$, produce 0, update t/=2
- several h produce same sequence
- add and divide

ok, but all we need $\forall \rho\ \exists h_\rho$ Greedy($1, h_\rho$) produces density ρ, and $\rho < \rho'$ implies $h_\rho > h_{\rho'}$

$\forall\rho\ \exists h_\rho$





- density p/q — 2/5
- slope p/(q-p) — 2/3
- produces w* — (00101)*
- $\underline{w} := \sum_{i=0..q-1} 2^i w_i$ — 4+16=20
- $h=(2^q-1)/\underline{w}$ — 31/20
- $\text{CircShift}(x_1x_2\ldots x_k) := x_2\ldots x_kx_1$
- Circ(x) := closure of {x} under CircShift
- Properties $\forall u \in \text{Circ}(w)$
  - $u \preceq w$ reverse lexicographically: $\underline{u} \leq \underline{w}$
  - $u1 \leq 1w$ iff u starts with 1

00101
01010

010101
100101
101001

procedure (w)

- init u=w
- repeat:
  - if u starts with 1, produce 1
  - else produce 0
  - update u=CircShift(u)

equivalences (let $t=\underline{u}/\underline{w}$)

$t+h \leq 2$
$\underline{u}+2^q-1 \leq 2\underline{w}$
$\underline{u1}-1 \leq \underline{0w}$
$\underline{u1} \leq \underline{1w}$
u starts with 1

update

- if u starts with 0 then $\underline{\text{CircShift}}(u)=\underline{u}/2$
- if u starts with 1 then u=1v and $(\underline{u}+2^q-1)/2=\underline{v}+2^{q-1}=\underline{v1}$ $=\underline{\text{CircShift}}(u)$

conclusion

- CoolestFirst(h) produces same sequence as DiscreteLine(p/q)

Future directions

- Use this technique to analyse scheduling problems with different renewable resources
- Develop a new technique to analyse the Better Algorithm (priority to jobs $\in [0,1]$)
- Come up with better bounds for the optimum schedule by rounding jobs $\in [0,1]$ as well