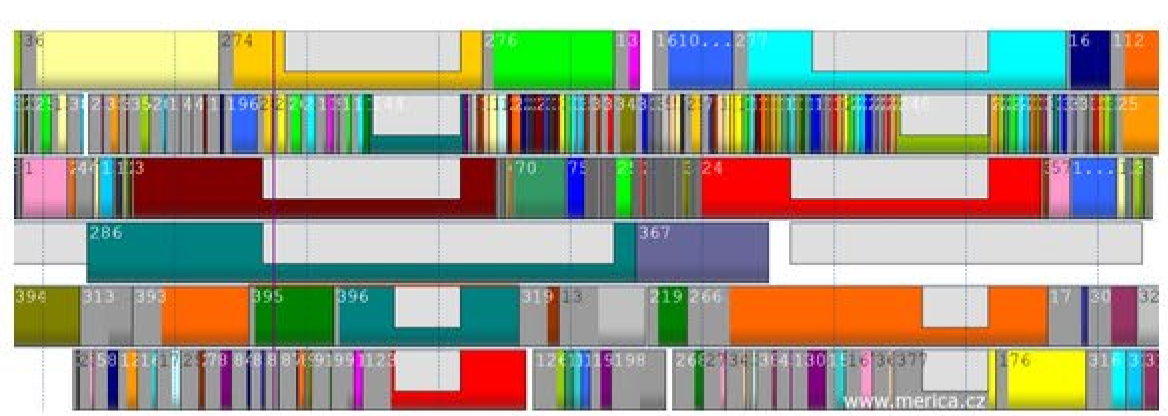
# State-of-the-art flowshop scheduling heuristics: Dos and Don'ts

Rubén Ruiz



## Scheduling seminar

Objective of a virtual seminar on scheduling research and applications is to discuss both the field's newest advancements and survey traditional areas. Seminars take place typically on every second Wednesday through three different time zones (Europe, the Middle East & Africa, North America & South America, and Asia, Australia & Oceania).

www.schedulingseminar.com

December 2021

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

SISTEMAS DE OPTIMIZACIÓN APLICADA


# Rubén Ruiz García

rruiz@eio.upv.es

Departamento de Estadística e Investigación Operativa Aplicadas y Calidad

SISTEMAS DE OPTIMIZACIÓN APLICADA

Universitat Politècnica de València

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ruben.ruizgarcia.79

@RubRubRuiz

RG Ruben_Ruiz4

# Outline

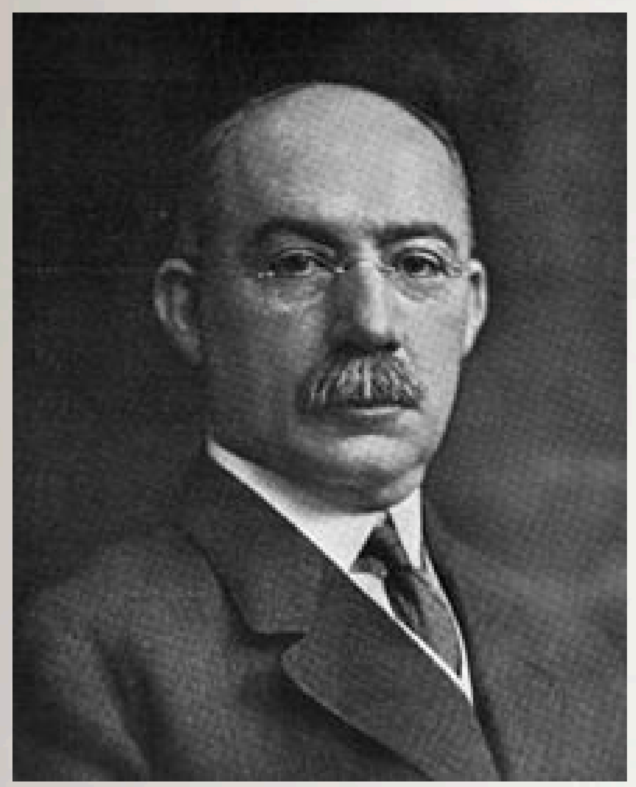
1. Introduction
2. The flowshop problem
3. Basic IG algorithm
4. Results for other flowshop problems
5. Complex hybrid problems
6. Conclusions

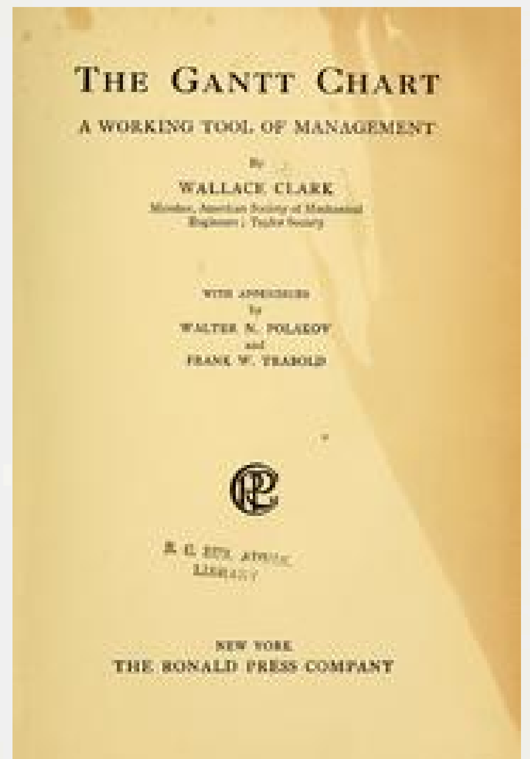
# 1. Introduction

”Scheduling is a decision-making process that is used on a regular basis in many manufacturing and services industries. It deals with the allocation of resources to tasks over given time periods and its goal is to optimize one or more objectives”

*Scheduling. Theory, Algorithms and Systems. Michael Pinedo. Springer (2016). Fifth Edition*

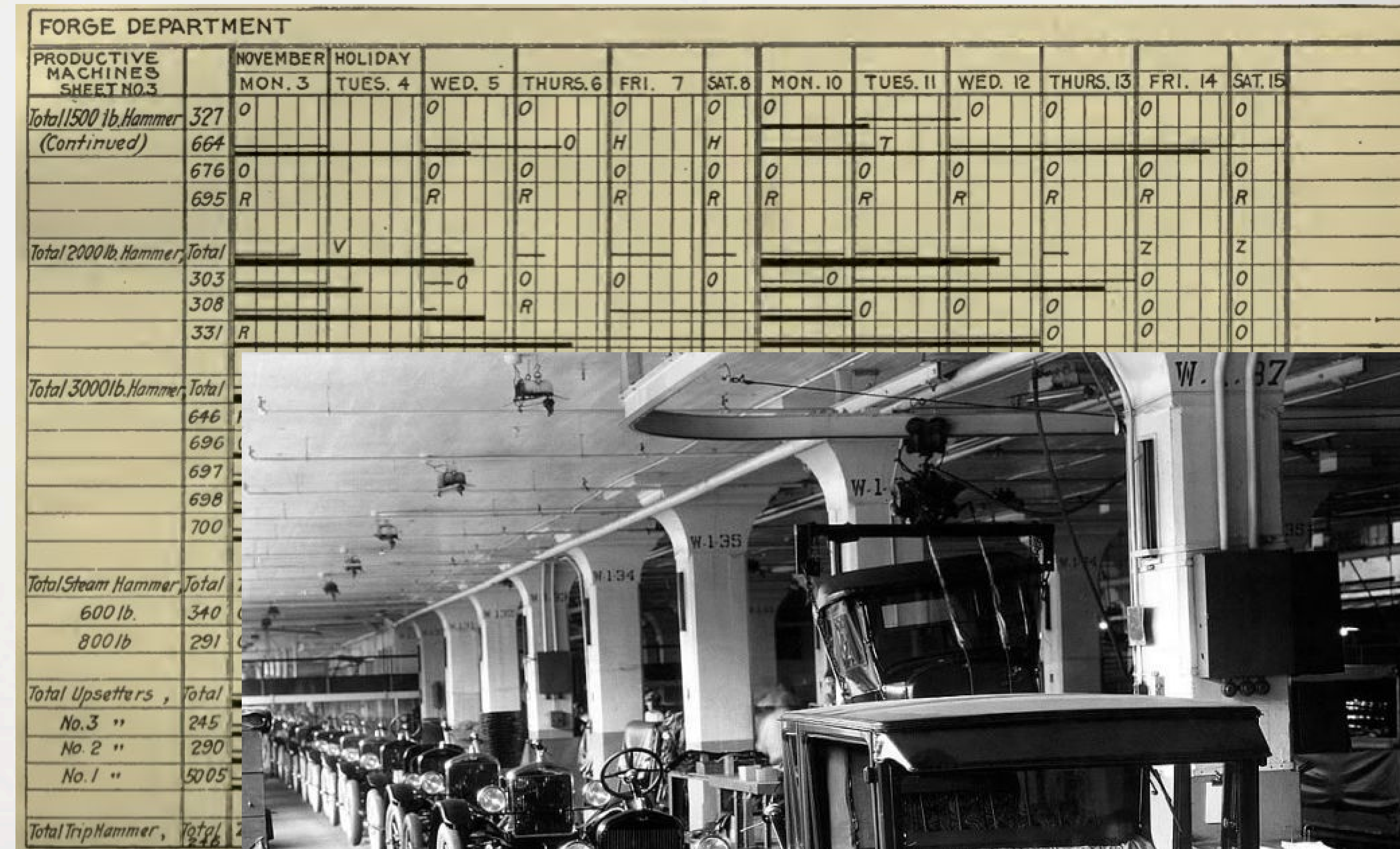
# Introduction



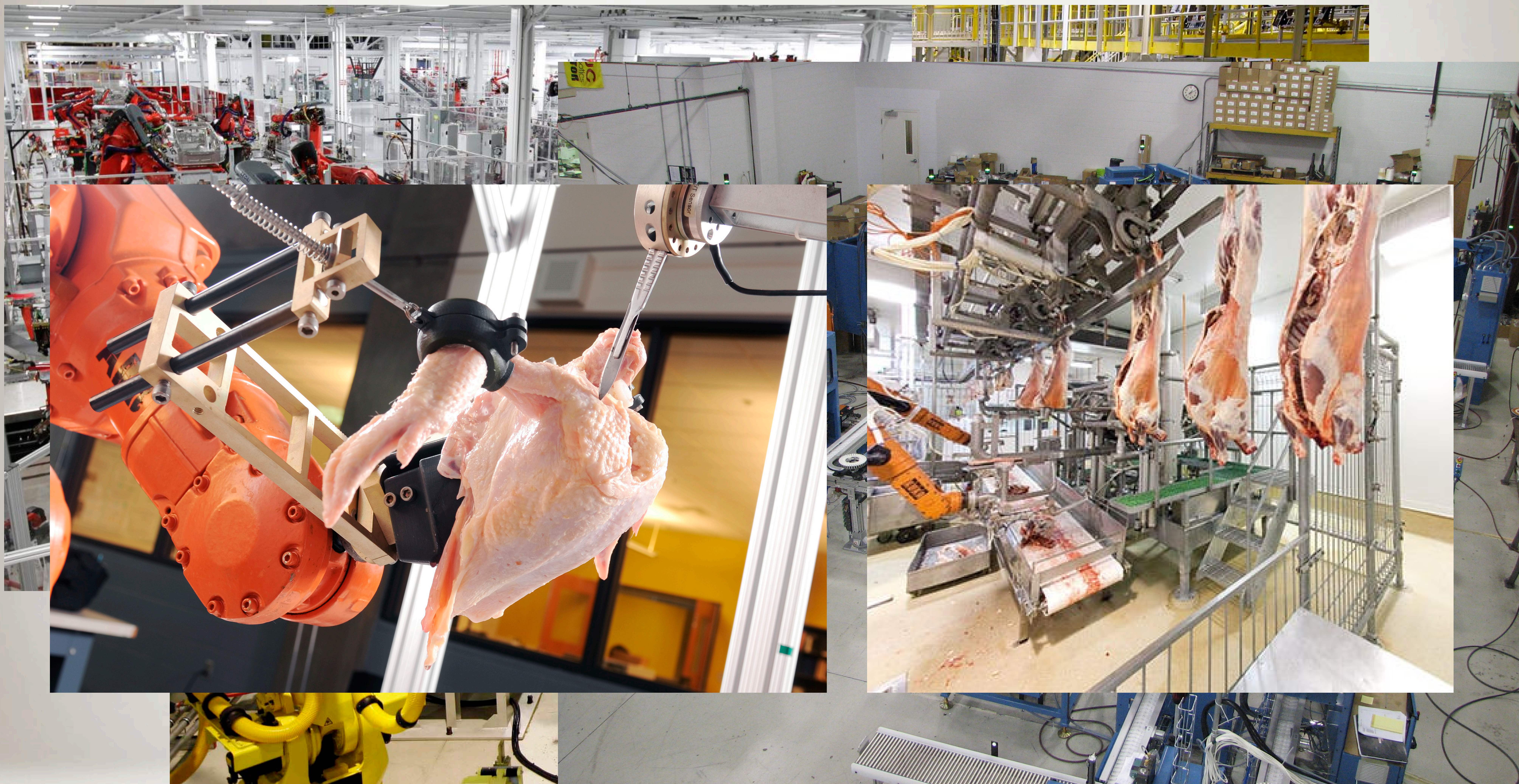



Henry Lawrence Gantt

(1861-1919)




# Introduction



# Introduction

Scheduling today is notoriously difficult and complicated

Production processes vary a lot from industry to industry:

- Not the same producing an LCD panel
- Than a ceramic tile
- Ad-hoc specific algorithms for each process/product is not a viable approach, as we would need thousands of different algorithms with huge maintenance costs

# Introduction

We need general optimization methods

Context independent

Flexible

But at the same time powerful

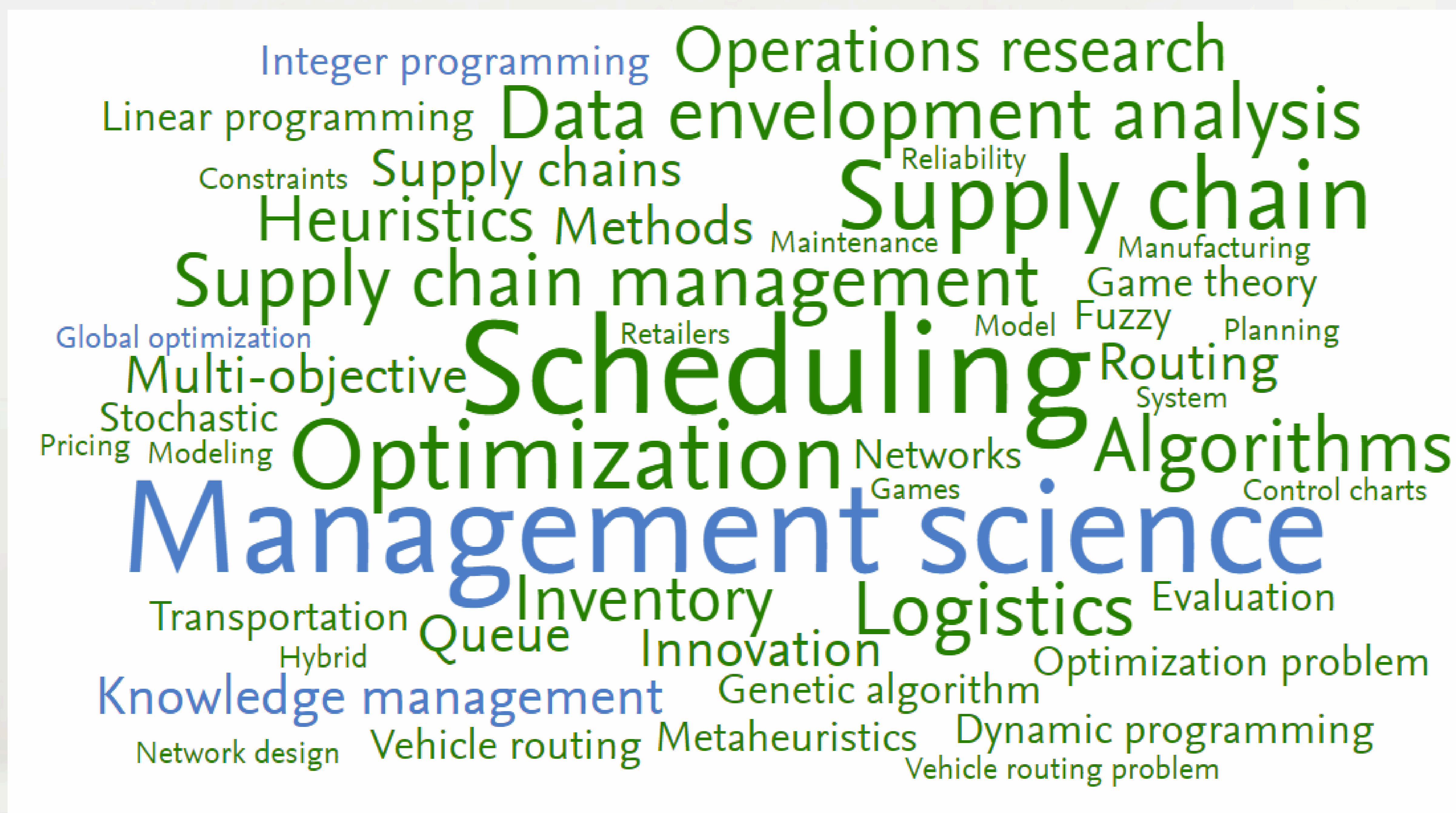
Optimality is a panacea for real complex problems

We have to resort to heuristics

# Introduction

## Scheduling

SciVal Trends. 50 most frequent keywords in the Management Science and Operations Research area between 2009 and 2021 (147,471 analyzed papers)



# Introduction

## Metaheuristics

"…higher level procedure or heuristic designed to find, generate, or select a lower-level procedure or heuristic (partial search algorithm) that may provide a sufficiently good solution to an optimization problem…"

(wikipedia)

# Introduction

## Metaheuristics is a very prolific field

1. Genetic algorithms (Holland, 1975)
2. Simulated Annealing (Kirkpatrick et al., 1983)
3. Tabu Search (Glover, 1986)
4. GRASP (Feo and Resende, 1989)
5. Ant Colony Optimization (Dorigo, 1992)
6. Iterated Local Search (Stützle, 1998)
7. Particle Swarm Optimization (Kennedy, 1995)
8. VNS (Hansen and Mladenović, 1999)

# Introduction

## Maybe a bit too prolific

9. Artificial Immune Systems (Forrest et al., 1994)
10. Self-Propelled Particles (Vicsek et al., 1995)
11. Differential Evolution (Storm and Price, 1997)
12. Harmony Search (Zong, 2001)
13. Bee Colony Optimization (Karaboga, 2005)
14. Firefly Optimization (Krishnanand and Ghose, 2005)
15. Intelligent Water Drops (Shah-Hosseini, 2009)

…

# Introduction

## And today we have really lost our minds

Kangaroo algorithms (Fleury, 1995)

Squeaky Wheel Optimization (Joslin and Clements, 1999)

Imperialist Competitive Algorithm (Atashpaz-Gargari and Lucas, 2007)

Cuckoo Optimization (Rajabioun, 2011)

Water cycle algorithms (Eskandar, 2012)

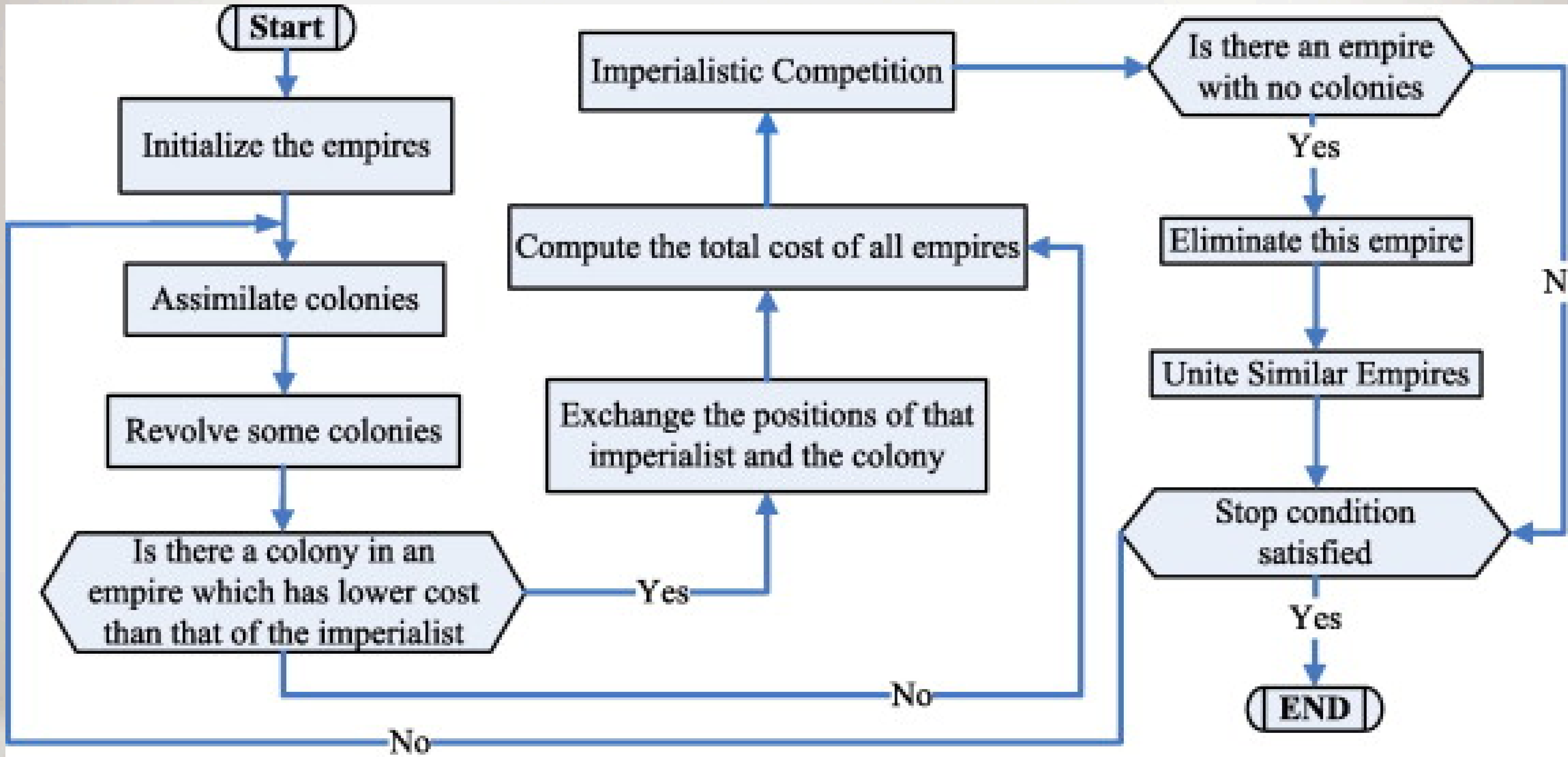
Mine Blast optimization (Sadollah, 2013)

Gases Brownian Motion Optimization (Abdechiri, 2013)

Leapfrog optimization, bats, flies, galaxies, roots, … whatever

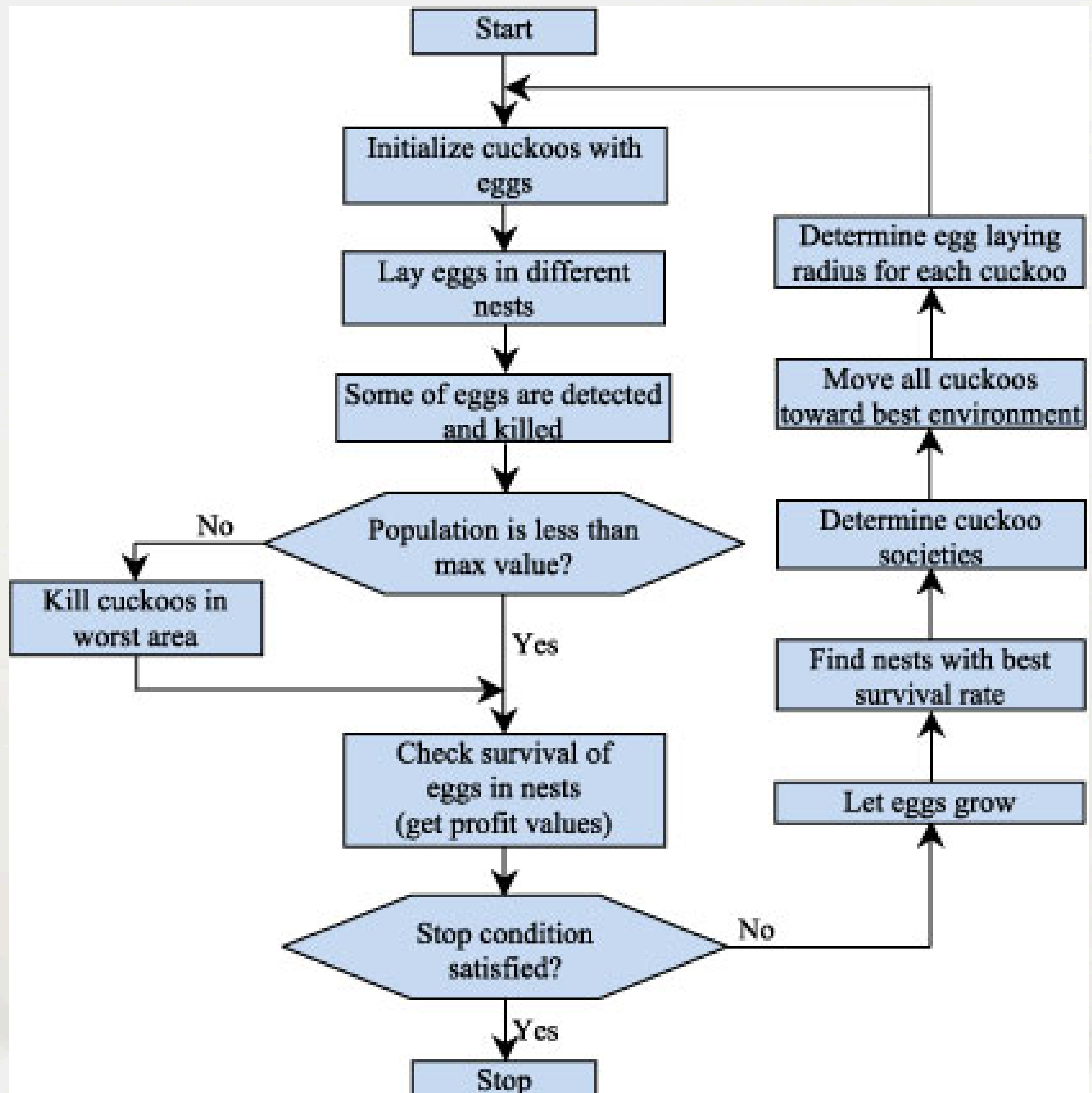
# Introduction

## An example: The Imperialist Competitive Algorithm



# Introduction

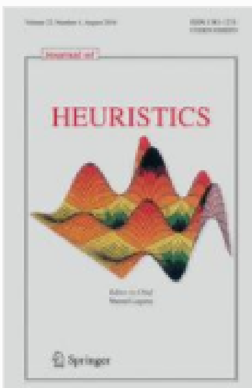
## Another example: Cuckoo Optimization



# Introduction

Springer Link

Search




Journal of Heuristics
August 2016, Volume 22, Issue 4, pp 649–664

## Flying elephants: a general method for solving non-differentiable problems

Authors    Authors and affiliations

Adilson Elias Xavier, Vinicius Layter Xavier



### Abstract

Flying Elephants (FE) is a generalization and a new interpretation of the Hyperbolic Smoothing approach. The article introduces the fundamental smoothing procedures. It contains a general overview of successful applications of the approach for solving a select set of five important problems, namely: distance geometry, covering, clustering, Fermat–Weber and hub location. For each problem the original non-smooth formulation and the succedaneous completely differentiable one are presented. Computational experiments for all related problems obtained results that exhibited a high level of performance according to all criteria: consistency,

Even the editor had to apologize!

# Introduction

## We have put together a bestiary

### Evolutionary Computation Bestiary

DOI 10.5281/zenodo.1293352

Updated 2021-05-04

> "Till now, madness has been thought a small island in an ocean of sanity. I am beginning to suspect that it is not an island at all but a continent." -- Machado de Assis, *The Psychiatrist*.

#### Introduction

The field of meta-heuristic search algorithms has a long history of finding inspiration in natural systems. Starting from classics such as Genetic Algorithms and Ant Colony Optimization, the last two decades have witnessed a fireworks-style explosion (pun intended) of natural (and sometimes supernatural) heuristics - from Birds and Bees to Zombies and Reincarnation.

The goal of the Evolutionary Computation Bestiary is to catalog the, ermm... exuberance of the meta-heuristic "eco-system". We try to keep a list of the many different animals, plants, microbes, natural phenomena and supernatural activities that can be spotted in the wild lands of the metaphor-based computation literature.

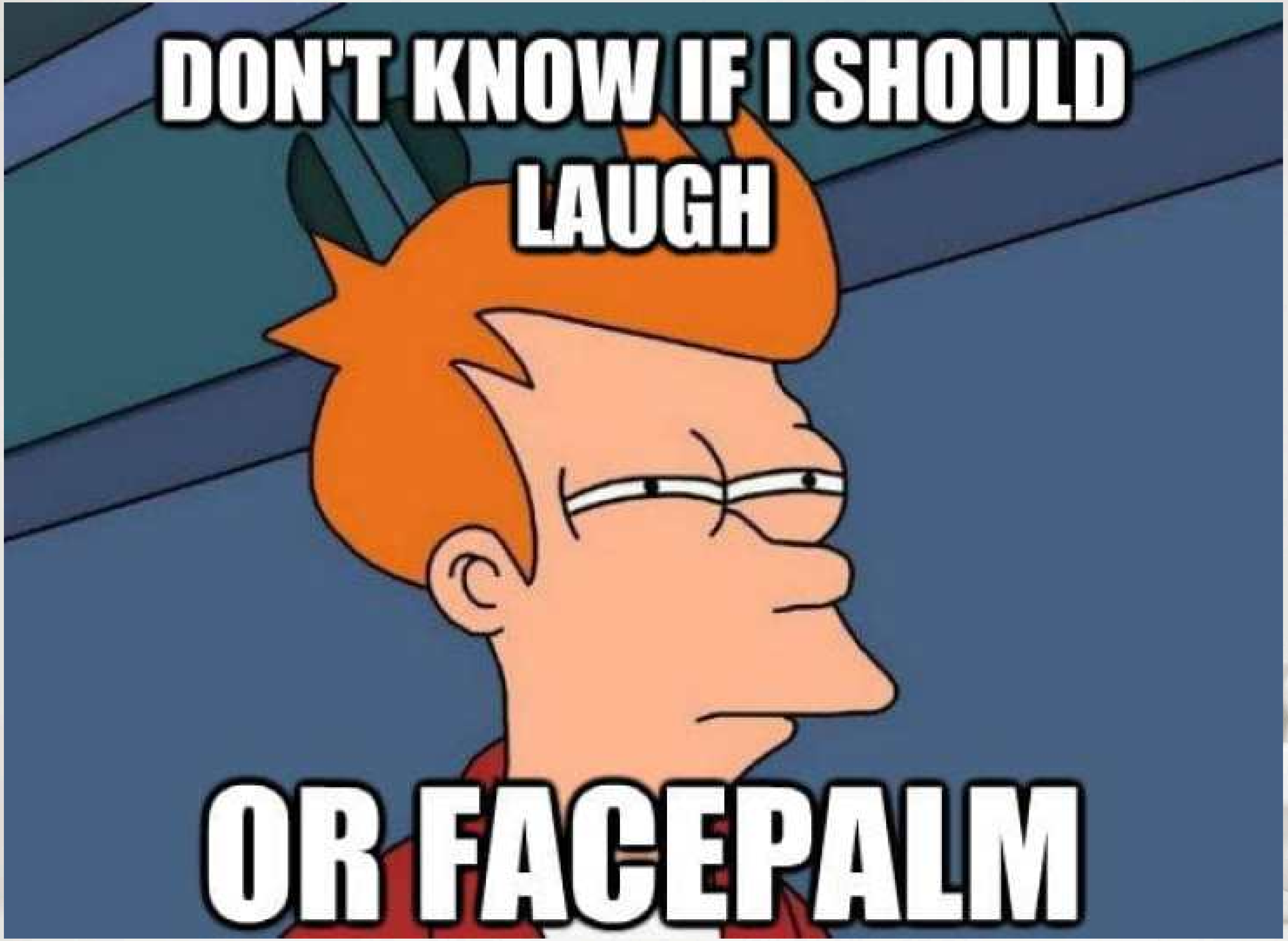
While we personally believe that the literature could do with more mathematics and less marsupials, and that we, as a community, should grow past this metaphor-rich phase in our field's history (a bit like chemistry outgrew alchemy), please note that this list makes no claims about the scientific quality of the papers listed. The EC Bestiary puts classic works of the metaheuristics literature (e.g., GAs, ACO) and some that describe their methods in mostly metaphor-free language (e.g., JTF, CFO) side by side with others for which the scientific rigor is, to put it mildly, lacking. In short, it is not a Hall of Fame of algorithms - think of it more as The island of Doctor Moreau: a place with a few good creatures, but which are vastly outnumbered by mindless beasts.

Finally, if you know a metaphor-based method that is not listed here, or if you know of an earlier mention of a listed method, please see the bottom of the page on how to contribute!

Gotta catch 'em all!
BioHeuristics GO

https://github.com/fcampelo/EC-Bestiary

# Introduction



# Introduction

## This hasn't gone unnoticed

Sörensen, K. (2015). Metaheuristics—the metaphor exposed, *International Transactions in Operational Research* 22(1): 3-18.

## Abstract:

"In recent years, the field of combinatorial optimization has witnessed a true tsunami of "novel" metaheuristic methods, most of them based on a metaphor of some natural or man-made process. The behavior of virtually any species of insects, the flow of water, musicians playing together – it seems that no idea is too far-fetched to serve as inspiration to launch yet another metaheuristic. In this paper, we will argue that this line of research is threatening to lead the area of metaheuristics away from scientific rigor. We will examine the historical context that gave rise to the increasing use of metaphors as inspiration and justification for the development of new methods, discuss the reasons for the vulnerability of the metaheuristics field to this line of research, and point out its fallacies"

# Introduction

## Many of these bizarre methods get cited a lot

1. Being the first to apply the method X to the problem Y
2. Easy to improve the basic method X by adding operators or hybridizing
3. In a little while the method X gets many citations and then everybody thinks that it is good because of that
4. Easy to publish: There are more than 50,000 scientific journals in the world and more than 50 million papers published throughout history

# Introduction

Zong Woo Geem, Joong Hoon Kim, and G. V. Loganathan. "A new heuristic optimization algorithm: harmony search." Simulation 76(2):60-68, 2001. 5462 citations in Google Scholar at 13th of December, 2020

# Introduction

Peas-to-Melons comparisons

Focusing only on solution quality, not considering (to some extent) CPU time

Metaheuristics use resources (CPU time, memory) to give a solution

Not carefully controlling CPU time in the comparisons leads to fallacies that are misleading (part) of the scientific community

# Introduction

Comparisons often against published tables with results obtained years ago:

- Different processors (older)
- Memory speed, bus speed (older)
- Different compilers (older)
- Different programming languages
- Different operating systems
- Different coding skills
- Different stopping criteria

These factors add-up!

# Introduction

Corrections based on raw CPU frequency are utterly wrong

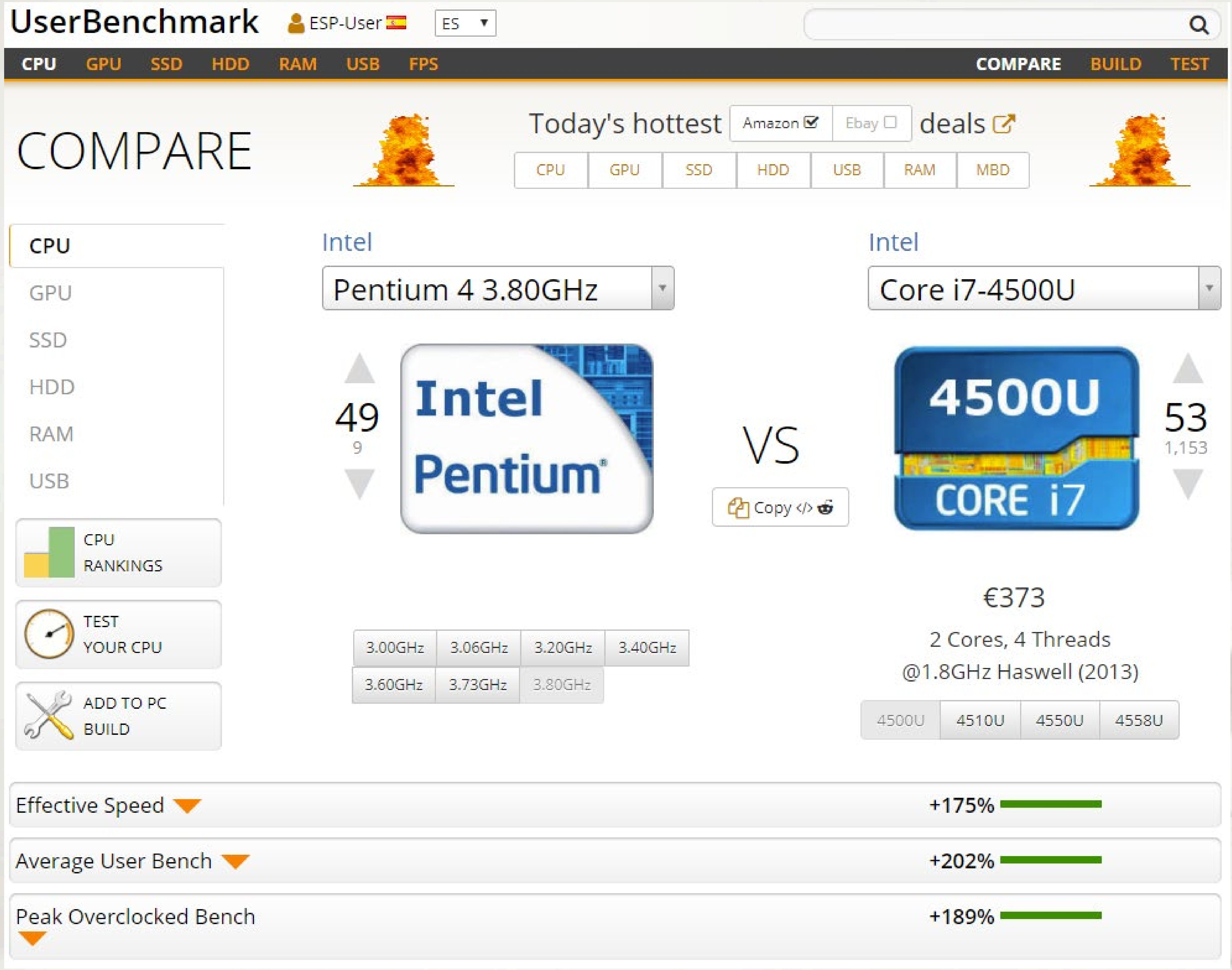
Intel Pentium 4 570 3.8 GHz (circa 2004)

Intel Core i7 4500U 1.8 GHz (circa 2013)

Older model more than twice the clock speed

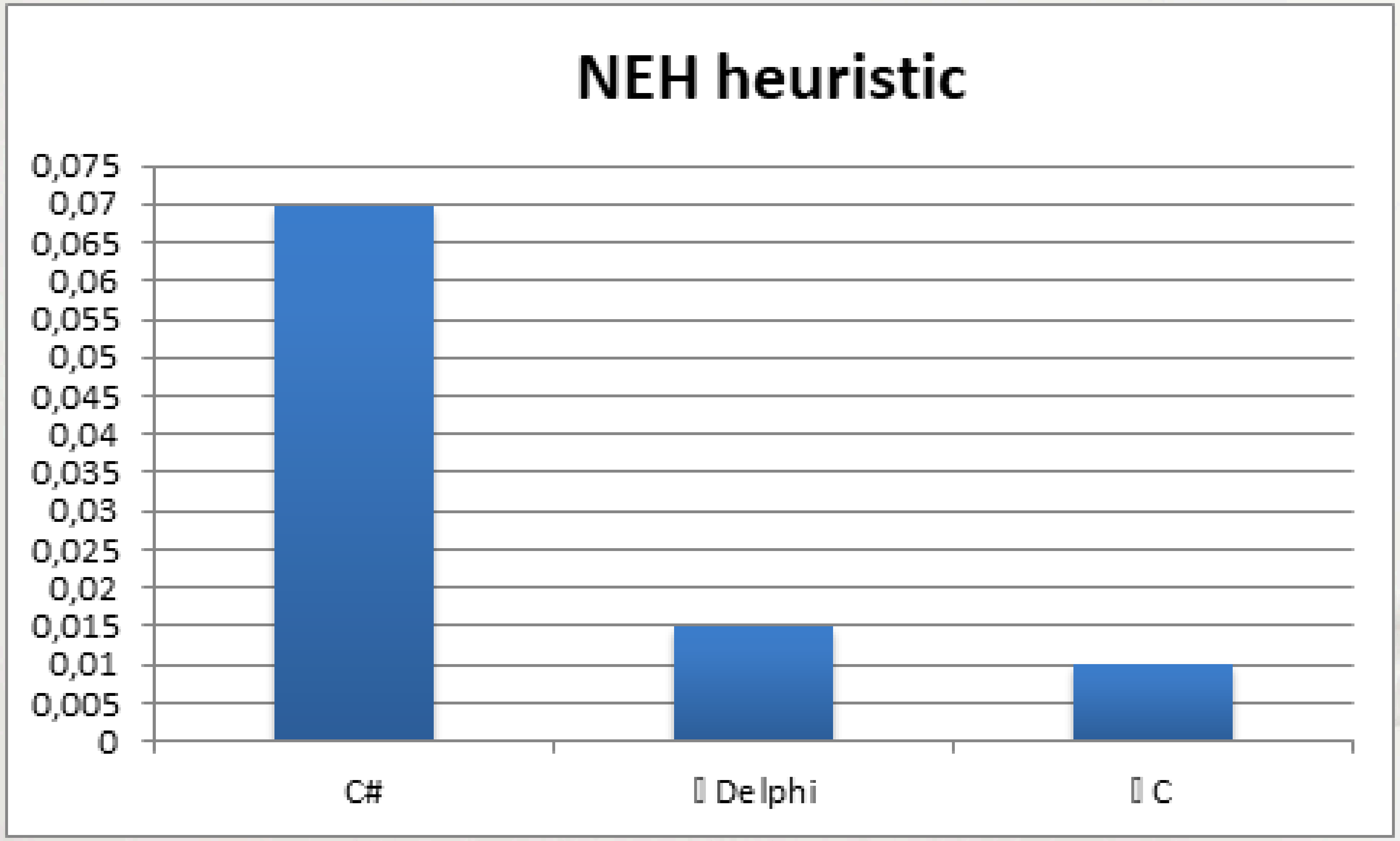
According to cpu.userbenchmark.com the new model is TWICE as fast with HALF the clock speed

# Introduction



# Introduction

## Is the compiler/language so important?




7x speed up from C# to C

# Introduction

From Visual Studio 2013 to Visual Studio 2015 you get a 20% improvement in C# binary speed due to new compiler technology "Roslyn"

Inlining/optimizing a frequently called function can improve code speed by two % digits

How can we trust a 7% improvement in solution quality in a "new" method in a Peas-To-Melons comparison?

# Introduction

- Apples-to-apples comparisons:
  - REIMPLEMENT published algorithms
    - In the same language
    - Sharing most functions
    - Same coding skills
  - TEST in the same computer platform
    - Same processor, speed, architecture
    - Same compiler
    - Same OS
  - Run with used thread CPU-time as stopping criterion
  - Carry out statistical testing for significance

# Introduction

What authors are doing as a result of the Peas-to-Melons comparisons:

"New" ideas easily best published methods in "comparable" running times

The better results of the "new" ideas are basically the compounded effect of a faster CPU, newer compilers, etc.

"Hybridized" versions of existing methods are seen as "better" just because they run on newer hardware not because they are actually better

# Introduction

Do we need such complexities?

Simple methods have many advantages:

1. Easy to understand
2. Easy to code
3. Easy to transfer to industry
4. Easy to extend and adapt, less parameters, etc.

# Introduction

In this course I will defend the choice of very simple algorithms

That at the same time produce state-of-the-art results

…Without frowning metaphors

# 2. The flowshop problem

$n$ jobs to schedule in $m$ machines

Each job visits the machines in the same order

The order of the jobs is the same for all machines

$n \cdot m$ tasks to schedule

$p_{ij}$ is the processing time of job $j$ at machine $i$

Jobs are independent and available for processing at time 0.

Machines are continuously available

# The flowshop problem

Objective: Find a permutation $\pi$ of jobs so that a given criterion is optimized: sequence.

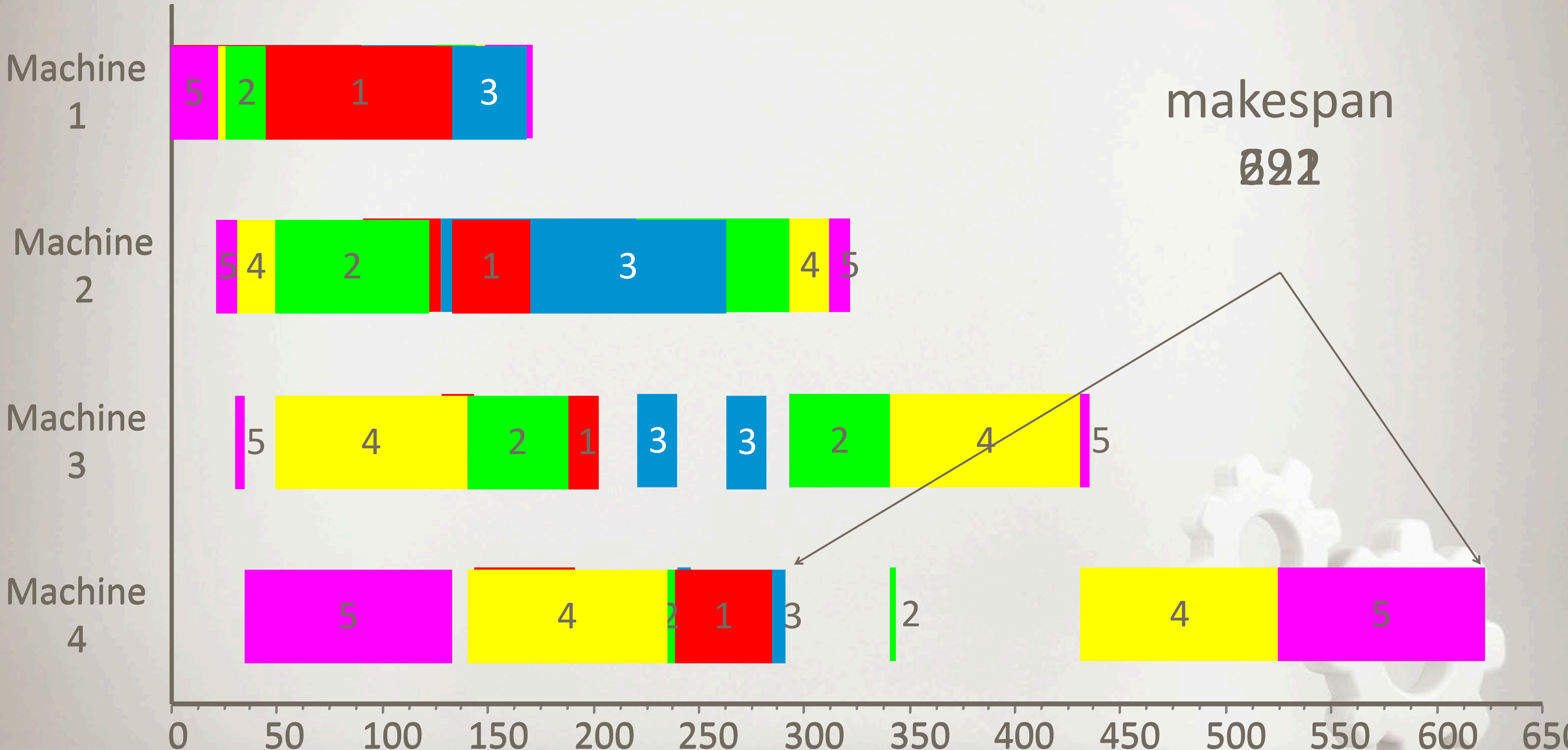
$n!$ possible solutions

Makespan minimization ($C_{\max}$) is the most common objective

NP-Complete for $m \geq 3$ (Garey et al., 1976)

Denoted as $F/prmu/C_{\max}$

# The flowshop problem



# The flowshop problem

Complex and hard to reproduce state-of-the-art

- TSAB of Nowicki and Smutnicki (1996)
- RY of Reeves and Yamada (1998)
- TSGW of Grabowski and Wodecki (2004)
- PACO and M-MMAS of Rajendran and Ziegler (2004)

Algorithms full of operators, accelerations and problem – specific knowledge = bad reproducibility and inability to extend to other problems

# 3. Basic IG algorithm

Ruiz and Stützle, EJOR (2007)

Initialization

Local search (optional)

While stopping criterion not satisfied

- Random partial destruction
- Greedy reconstruction
- Local search (optional)
- Acceptance criterion

# Solution representation

The most natural is a permutation of size $n$

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

Easy to code by an array or a list

# Initialization

It is very common to use effective heuristics to obtain good initial solutions

In the flowshop problem with makespan minimization the most cited and high performing heuristic is the NEH of Nawaz, Enscore and Ham (1983) (Ruiz and Maroto, 2005)

# Initialization

NEH evaluates a total of $[n(n+1)/2]-1$ sequences, where $n$ of these are complete schedules

Computational complexity of $\mathcal{O}(n^3m)$

With Taillard (1993) implementation, complexity goes down to $\mathcal{O}(n^2m)$

In practice, large problems of 500×20 are solved with fast code in less than 30 milliseconds

# Destruction

We start from a complete permutation $\pi$ of $n$ jobs

A random number of jobs are selected ( $destruct$ )

and are removed from the sequence in the selected order

Two sub-sequences are obtained: The original without the removed jobs : $\pi_D$ , of size $n - destruct$ and the one with the removed jobs: $\pi_R$ , of size $destruct$

# Reconstruction

NEH's last step is used

We start from subsequence $\pi_D$

And carry out $destruct$ iterations

At each iteration the first job of $\pi_R$ is reinserted in all the positions of $\pi_D$ ( $n - destruct + i$ )

The job is placed in the position resulting in the smallest $C_{\max}$

Finished when $\pi_D$ is complete ( $\pi_R = \varnothing$ )

# Example

Instance Car8 of Carlier. 8×8

Solution after NEH: {7,3,4,1,8,2,5,6} $C_{\max} = 8564$

Destruction phase

Reconstruction phase

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 3 | 4 | 1 | 8 | 2 | 5 | 6 |

$C_{\max} =$ [illegible]

$\pi_D$

$\pi_R$

# Local search

Many potential neighborhoods

For the flowshop problem the most effective is insert

Destination position $k$

Original position $j$

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

# Local search

Local search based on the insertion neighborhood

All jobs extracted and reinserted into all possible positions, until local optimality

Insertion neighborhood: Given two positions $j, k \in N, j \neq k$

$$\pi' = \left\{\pi_{(1)}, \ldots, \pi_{(j-1)}, \pi_{(j+1)}, \ldots, \pi_{(k)}, \pi_{(j)}, \pi_{(k+1)}, \ldots, \pi_{(n)}\right\}, (j < k)$$

$$\pi' = \left\{\pi_{(1)}, \ldots, \pi_{(k-1)}, \pi_{(j)}, \pi_{(k+1)}, \ldots, \pi_{(j-1)}, \pi_{(j+1)}, \ldots, \pi_{(n)}\right\}, (j > k)$$

$$I = \{(j,k) : j \neq k,\ 1 \leq j,k \leq n \wedge j \neq k-1,\ 1 \leq j \leq n, 2 \leq k \leq n\}$$

# Acceptance criterion

After destruction, reconstruction and optional local search we check if the new solution is accepted

Accepting only better solutions results in premature convergence

We apply a fixed temperature simulated annealing criterion

$$\text{Temperature} = T \cdot \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij}}{n \cdot m \cdot 10}$$

# *Iterated Greedy* algorithm

**procedure** *Iterated_Greedy*

$\pi :=$ NEH_Heuristic;

$\pi :=$ Insertion_LocalSearch$(\pi)$;

$\pi_b := \pi$;

**while** (termination criterion not satisfied) **do**

$\pi' := \pi$;

**for** $i := 1$ **to** *destruct* **do** % Destruction phase

$\pi' :=$ randomly extract a job of $\pi'$ and insert it into $\pi'_R$;

**for** $i := 1$ **to** *destruct* **do** % Reconstruction phase

$\pi' :=$ best permutation after inserting $\pi_R(i)$ into all possible positions of $\pi'$;

$\pi'' :=$ Insertion_LocalSearch$(\pi')$; % Local Search

**if** $C_{\max}(\pi'') < C_{\max}(\pi)$ **then** $\pi := \pi''$; % Acceptance criterion

**if** $C_{\max}(\pi) < C_{\max}(\pi_b)$ **then** $\pi_b := \pi$;

**elseif** $\left( random \leq e^{-\frac{(C_{\max}(\pi'') - C_{\max}(\pi))}{\mathrm{Temperature}}} \right)$ **then** $\pi := \pi''$;

**endif**

**endwhile**

**return** $\pi_b$

**end**

# Comparison

We compare two IG versions, with and without local search: IG_RS e IG_RS$_{LS}$

120 Taillard (1993) instances

12 reimplemented methods from the literature

Stopping criterion $n \cdot (m/2) \cdot 60$ ellapsed milliseconds

Response variable:

$$\text{Av. Rel. Pertentage Deviation } (\overline{RPD}) = \sum_{i=1}^{R} \left( \frac{Heu_{sol_i} - Best_{sol}}{Best_{sol}} \cdot 100 \right) / R$$

# Comparison

NEH of Nawaz et al. (1983) with Taillard (1990) accelerations: **NEHT**

Simulated annealing of Osman and Potts (1989): **SA_OP**

Tabu Search of Widmer and Hertz (1989): **SPIRIT**

GA of Reeves (1995): **GA_REEV**, of Chen et al. (1995): **GA_CHEN**, of Murata et al. (1996): **GA_MIT**, of Aldowaisan  Allahverdi (2003): **GA_AA** and Ruiz et al. (2006): **GA_RMA** and **HGA_RMA**

Iterated Local Search of Stützle (1998): **ILS**

Ant Colony Optimization of Rajendran and Ziegler (2004): **M-MMAS** and **PACO**

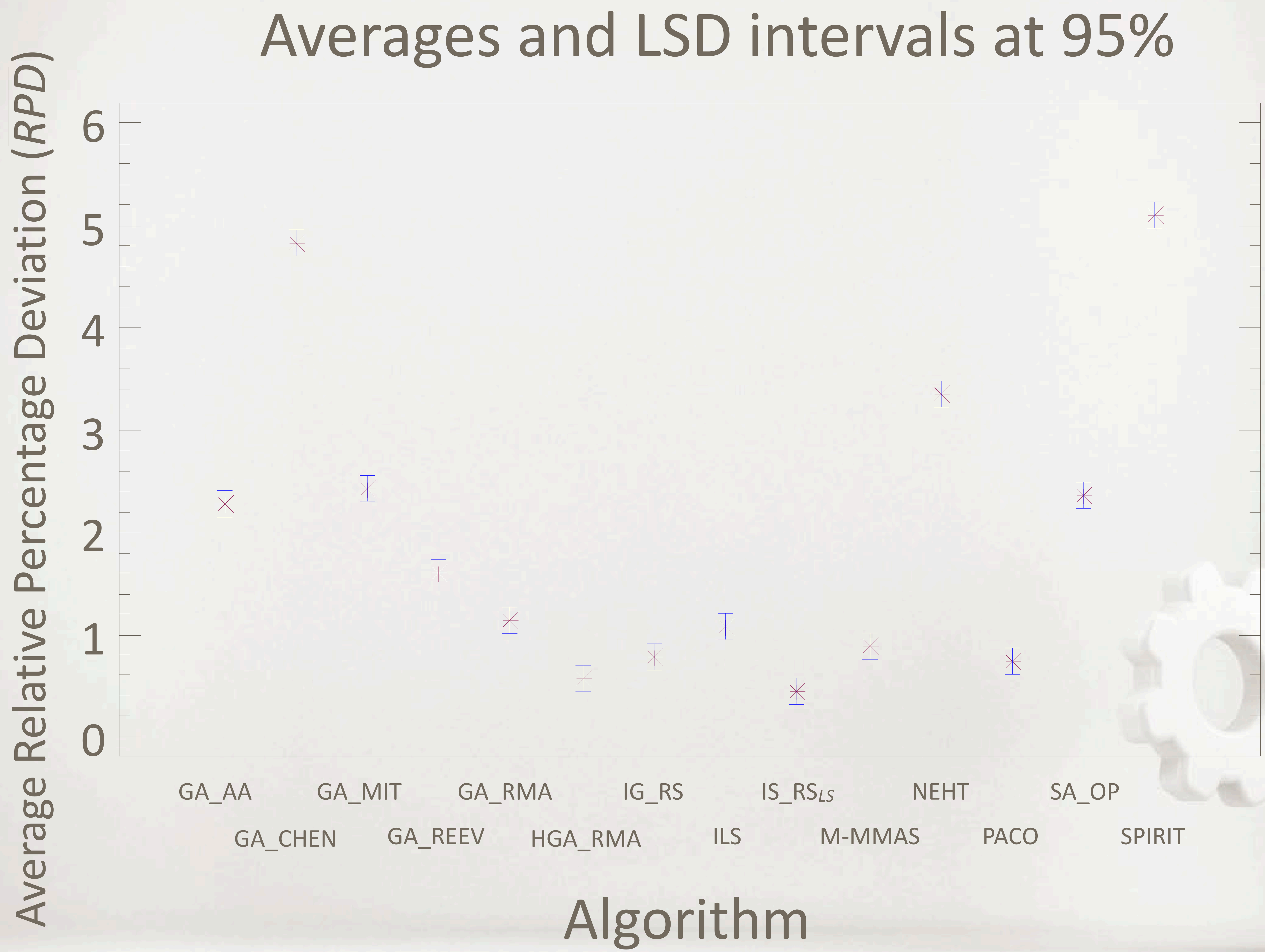
# Comparison

Average relative percentage deviations from best known solutions across all 120 instances:

| Method | NEHT | GA_RMA | HGA_RMA | SA_OP | SPIRIT | GA_CHEN | GA_REEV |
|---|---|---|---|---|---|---|---|
| AVRPD | 3.35 | 1.13 | 0.57 | 2.37 | 5.09 | 4.83 | 1.61 |

| Method | GA_MIT | ILS | GA_AA | M_MMAS | PACO | **IG_RS** | **IG_RS$_{LS}$** |
|---|---|---|---|---|---|---|---|
| AVRPD | 2.42 | 1.06 | 2.28 | 0.88 | 0.75 | **0.78** | **0.44** |

# Comparison



Averages and LSD intervals at 95%
Average Relative Percentage Deviation (RPD)
0
1
2
3
4
5
6
GA_AA
GA_CHEN
GA_MIT
GA_REEV
GA_RMA
HGA_RMA
IG_RS
ILS
IS_RSLS
M-MMAS
NEHT
PACO
SA_OP
SPIRIT
Algorithm

# Comparison

Later IG results of Vallada and Ruiz, EJOR (2009) using more modern computers and parallel computing:

| | Number of processors | | | |
|---|---|---|---|---|
| | 2 | 4 | 6 | 8 |
| AVRPD | 0.31 | 0.27 | 0.25 | 0.23 |

# More recent results

Fernandez-Viagas, Ruiz and Framinan, EJOR (2017)

The best 19 heuristics and 12 metaheuristics compared

IG based methods are best

An improvement over the original IG gives AVRPD of 0.28

# More recent results

| Acronym | Ref. | 60 | 90 | 90 |
|---|---|---|---|---|
| TSAB | Nowicki and Smutnicki (1996) | 0.97 | 0.87 | 0.84 |
| MSSA | Nowicki and Smutnicki (2006) | 1.00 | 0.91 | 0.84 |
| $IG\_RS_{LS}$ | Ruiz and Stützle (2007) | 0.47 | 0.40 | 0.37 |
| $IG_{RIS}$ | Pan et al. (2008) | 0.49 | 0.42 | 0.38 |
| $DDE_{RLS}$ | Pan et al. (2008) | 0.52 | 0.47 | 0.43 |
| 3XTS | Eksioglu et al. (2008) | 1.64 | 1.34 | 1.24 |
| H-CPSO | Jarboui et al. (2008) | 0.84 | 0.75 | 0.70 |
| $EDA_{ACS}$ | Tzeng and Chen (2012) | 0.60 | 0.51 | 0.47 |
| HCS | Li and Yin (2013) | 1.55 | 1.42 | 1.35 |
| PSO | Zhang and Wu (2014) | 1.09 | 0.95 | 0.84 |
| $IG\_RS_{LS}(TB_{FF})$ | Fernandez-Viagas and Framinan (2014) | 0.37 | 0.32 | 0.28 |
| $IG_{RIS}(TB_{FF})$ | Fernandez-Viagas and Framinan (2014) | 0.42 | 0.34 | 0.31 |

# 4. Results for other flowshop problems

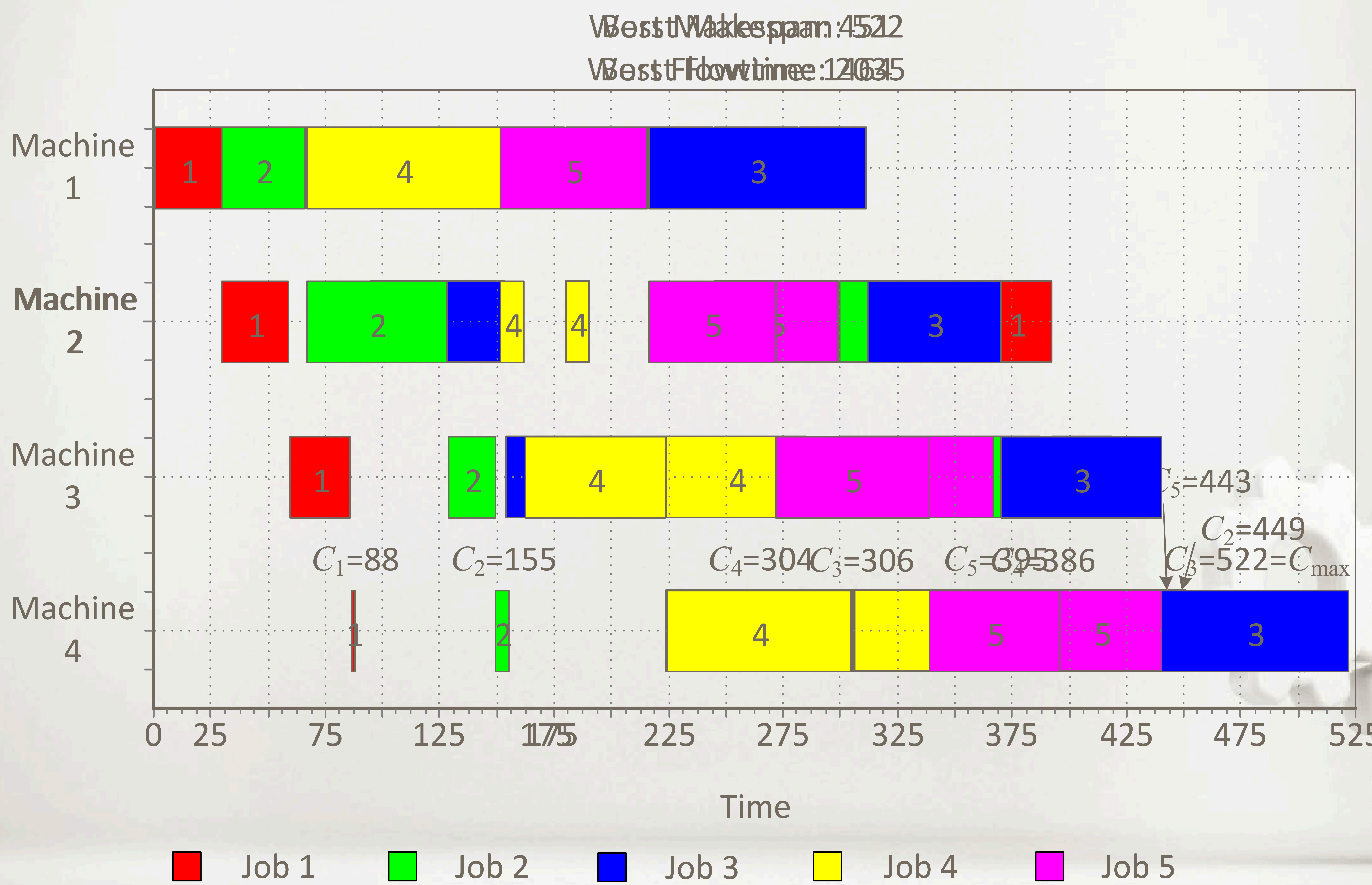
Makespan is not the most realistic criterion in practice

Total flowtime, weighted tardiness, etc.

Do we need to change the IG to obtain good results?

# Total flowtime

## Total flowtime is not correlated with makespan



# Total flowtime. *Iterated Greedy* algorithm

Pan and Ruiz, EJOR (2012)

Same algorithm

We only change the initialization from NEH to LR($n/m$) of Li et al. (2009)

Local search is a variant of the insertion of Rajendran and Ziegler (1997)

Everything else is the same

# Total flowtime. *Iterated Greedy* algorithm

## Results compared with 12 other methods:

1. Discrete Differential Evolution $DDE_{RLS}$ of Pan et al. (2008)
2. Iterated Greedy $IG_{RLS}$ of Pan et al. (2008)
3. Estimation of Distribution $EDA_{J}$ of Jarboui et al. (2009)
4. Variable neighborhood search $VNS_{J}$ of Jarboui et al. (2009)
5. Iterated local search $ILS_{D}$ of Dong et al. (2009)
6. Hybrid genetic algorithm $HGA_{T1}$ of Tseng and Lin (2009)
7. Hybrid genetic algorithm $HGA_{Z}$ of Zhang et al. (2009)
8. Hybrid genetic algorithm $HGA_{T2}$ of Tseng and Lin (2010)
9. Genetic Local Search AGA of Xu et al. (2011)
10. Hybrid Discrete Differential Evolution hDDE of Tasgetiren et al. (2011)
11. Discrete Ant Bee Colony DABC of Tasgetiren et al. (2011)
12. Iterated Greedy SLS of Dubois-Lacoste et al. (2011)

# Total flowtime. Evaluation

$\rho$ =30

| | $IG_{RLS}$ | $DDE_{RLS}$ | $EDA_J$ | $VNS_J$ | $ILS_D$ | $HGA_{T1}$ | $HGA_Z$ | $HGA_{T2}$ |
|---|---|---|---|---|---|---|---|---|
| AVRPD | 0.39 | 0.39 | *7.72* | 4.88 | 0.49 | 2.23 | 0.74 | 5.29 |

| | AGA | hDDE | DABC | SLS | IGA | pIGA | ILS | pILS |
|---|---|---|---|---|---|---|---|---|
| AVRPD | 0.87 | 0.64 | 0.83 | 0.41 | **0.24** | 0.28 | 0.25 | 0.31 |

$\rho$ =60

| | $IG_{RLS}$ | $DDE_{RLS}$ | $EDA_J$ | $VNS_J$ | $ILS_D$ | $HGA_{T1}$ | $HGA_Z$ | $HGA_{T2}$ |
|---|---|---|---|---|---|---|---|---|
| AVRPD | 0.36 | 0.36 | *7.02* | 4.39 | 0.49 | 2.13 | 0.63 | 4.50 |

| | AGA | hDDE | DABC | SLS | IGA | pIGA | ILS | pILS |
|---|---|---|---|---|---|---|---|---|
| AVRPD | 0.78 | 0.60 | 0.76 | 0.41 | **0.24** | 0.27 | 0.25 | 0.30 |

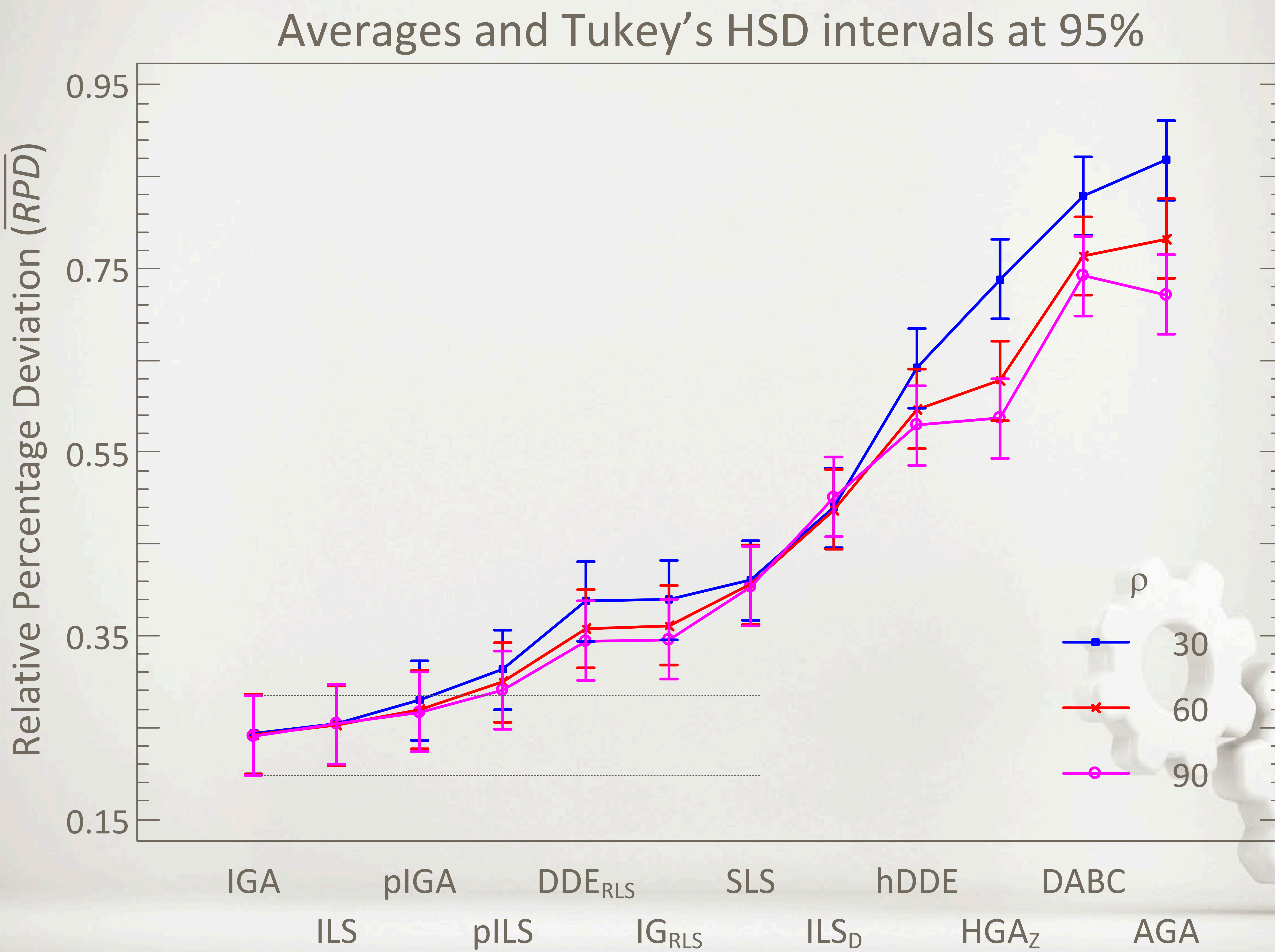
$\rho$ =90

| | $IG_{RLS}$ | $DDE_{RLS}$ | $EDA_J$ | $VNS_J$ | $ILS_D$ | $HGA_{T1}$ | $HGA_Z$ | $HGA_{T2}$ |
|---|---|---|---|---|---|---|---|---|
| AVRPD | 0.35 | 0.40 | *6.64* | 4.17 | 0.50 | 2.09 | 0.59 | 4.09 |

| | AGA | hDDE | DA[illegible] | [illegible]LS | IGA | pIGA | ILS | pILS |
|---|---|---|---|---|---|---|---|---|
| AVRPD | 0.72 | 0.58 | 0.[illegible] | [illegible].40 | **0.24** | 0.27 | 0.25 | 0.29 |

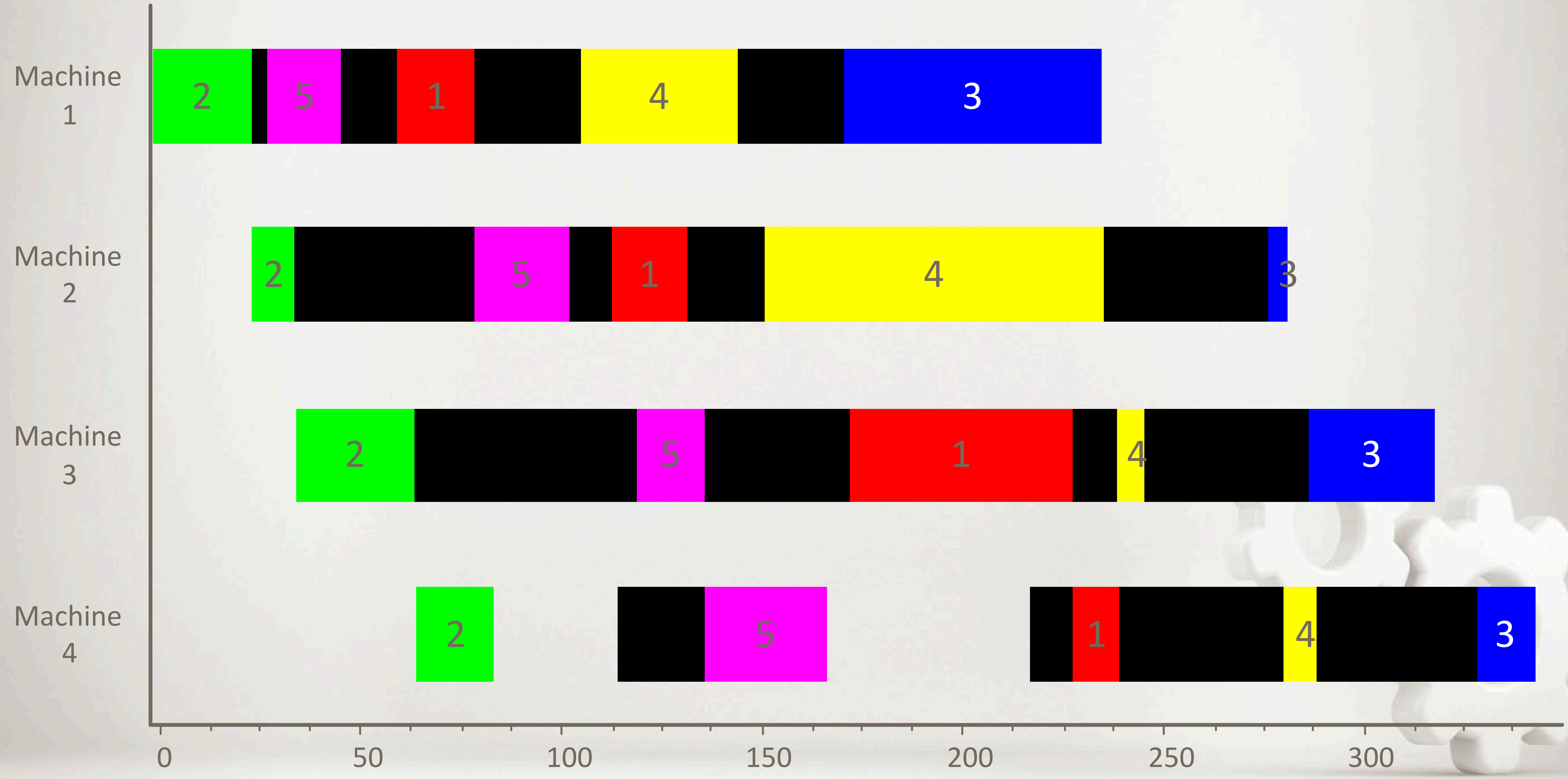
31% better

# Total flowtime. Evaluation



# Sequence dependent setup times

## Cleaning, fixing, reconfiguring, etc.



# SDST. *Iterated Greedy* algorithm

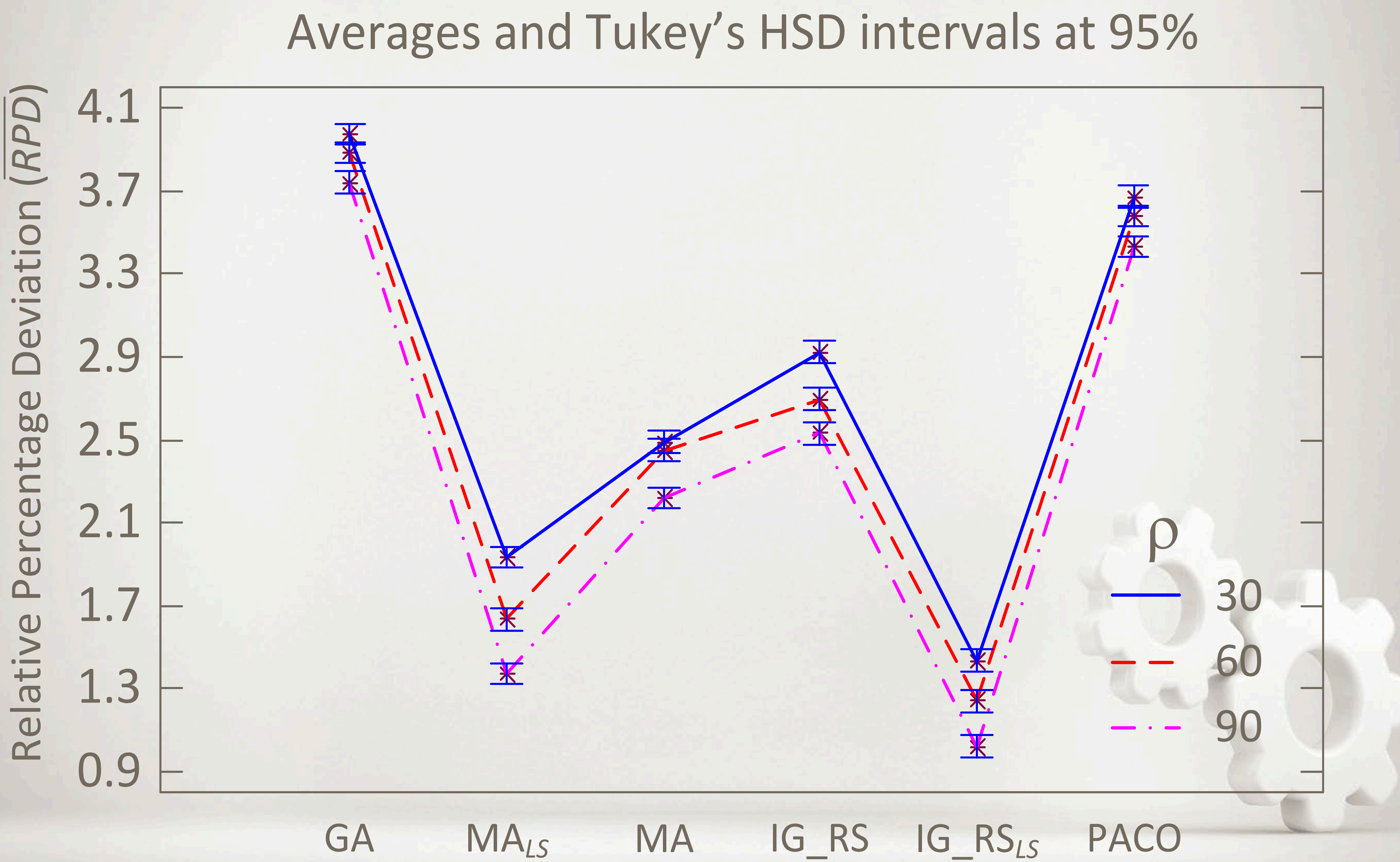
Ruiz and Stützle, EJOR (2008)

Same algorithm

We only adapt the NEH initialization to NEH with setups of Ríos-Mercado and Bard (1998). Trivial change

Everything else the same. We only change the objective function evaluation, which considers setup times

# SDST. Evaluation



# Distributed flowshops

F identical factories where jobs can be processed

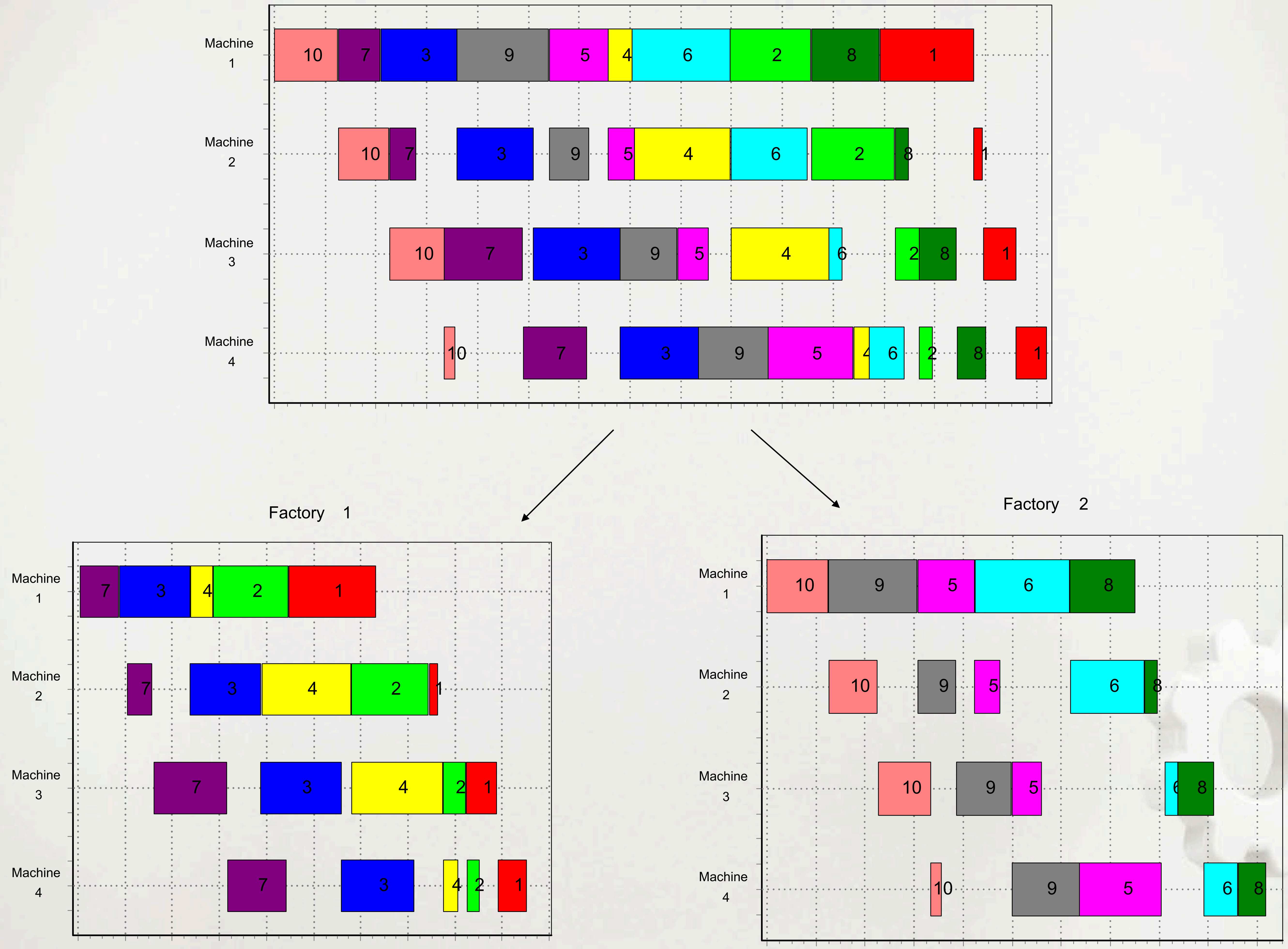
We have two interrelated decisions: assignment of jobs to factories and sequencing of the assigned jobs at each factory

Obviously, at each factory, the sequencing problem depends on the jobs assigned

Objective: to minimize the maximum makespan among the F factories

This problem is also NP-Hard if n>>F

# Distributed flowshops



# Distributed flowshops. *Iterated Greedy* algorithm

Ruiz, Pan and Naderi, OMEGA (2018)

Solution representation

The permutation of n jobs is divided among the F factories.

Therefore there is an array of F lists, one per factory

Initial solution

Small NEH improvement by carrying out limited reinsertions of adjacent jobs in the NEH, adapting previous results of Rad et al. (2009) and Pan and Ruiz (2014)

# Distributed flowshops. *Iterated Greedy* algorithm

Simple destruction

Same as Ruiz and Stützle (2007) where d/2 jobs are removed from the factory generating the Cmax and the others at random from the other factories

Reconstruction with re-insertions

# Distributed flowshops. *Iterated Greedy* algorithm

New local search exploring factory assignments and sequences of jobs at each factory

Two stage-IG where in the second stage only the Cmax generating factory is improved

# Distributed flowshops. Evaluation

We test the following methods:

1. Proposed single stage IG (IG1S)
2. IG1S version with normal local search and regular NEH initialization (IG1S$^{-}$)
3. Proposed two stage IG (IG2S)
4. The hybrid immune algorithm of Xu et al (2014) (HIA)
5. The Scatter search of Naderi and Ruiz (2014) (SS)
6. The bounded IG of Fernandez-Viagas and Framinan (2015) (BSIG)

# Distributed flowshops. Evaluation

Average Relative Deviation from Best Known Solution

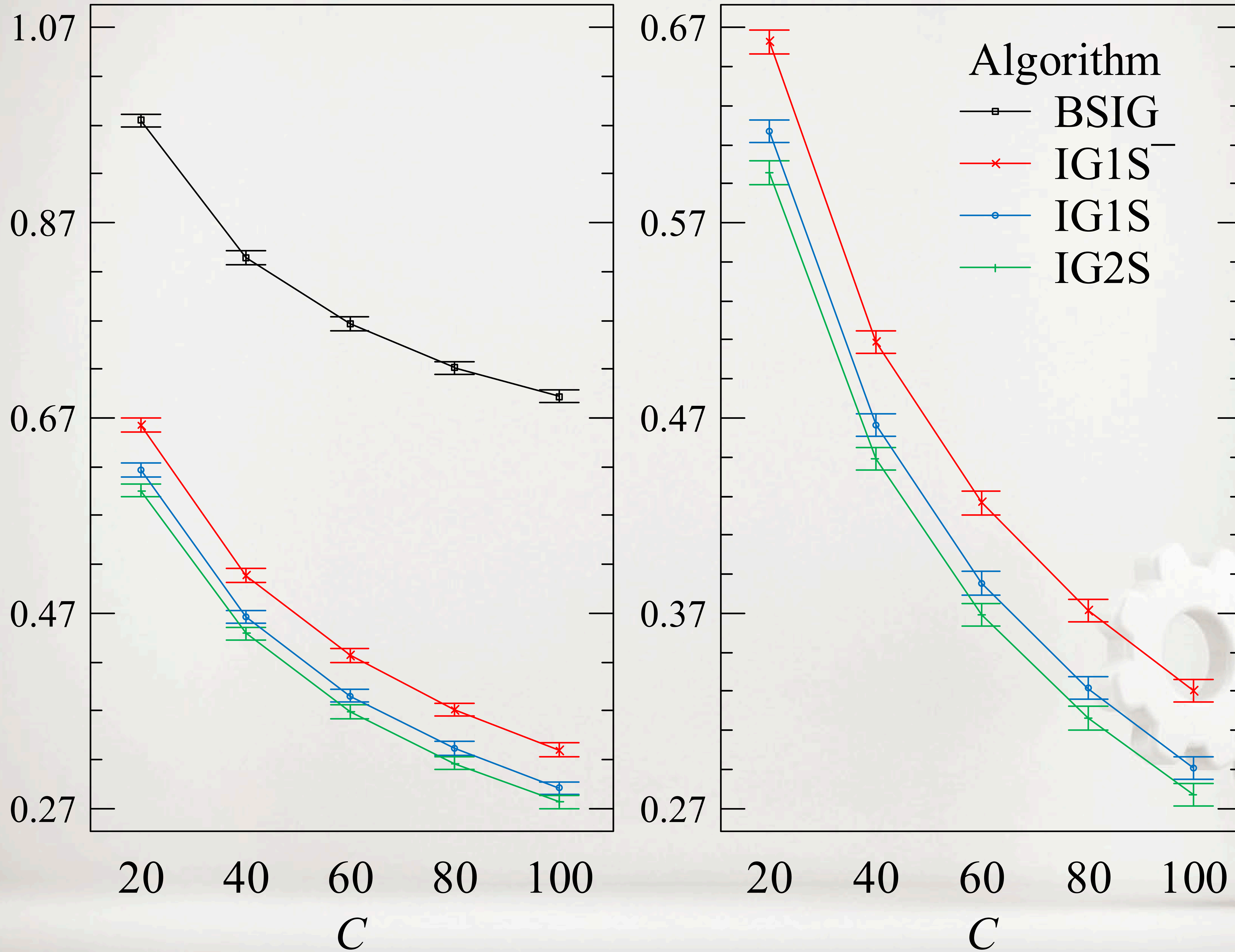
| $\rho$ | HIA | SS | BSIG | IG1S$^{-}$ | IG1S | IG2S |
|---|---|---|---|---|---|---|
| 20 | 10.54 | 1.80 | 0.97 | 0.66 | 0.62 | 0.60 |
| 40 | 10.06 | 1.64 | 0.83 | [illegible] | 0.47 | 0.45 |
| 60 | 9.78 | 1.55 | 0.77 | 0.43 | 0.39 | 0.37 |
| 80 | 9.58 | 1.49 | 0.72 | 0.37 | 0.33 | 0.32 |
| 100 | 9.37 | 1.45 | 0.69 | [illegible] | 0.29 | 0.28 |
| Average | 9.87 | 1.59 | 0.80 | 0.46 | 0.42 | 0.40 |

37% lower

58% lower

47% lower

# Distributed flowshops. Evaluation



# 5. Complex hybrid problems

Hybrid flowshops coalesce regular flowshops and parallel machines

Instead of a set of machines in series (flowshop) or in parallel (parallel machines) we have a set of stages in series where each stage has several parallel machines

It is a multiple problem with sequencing and assignment

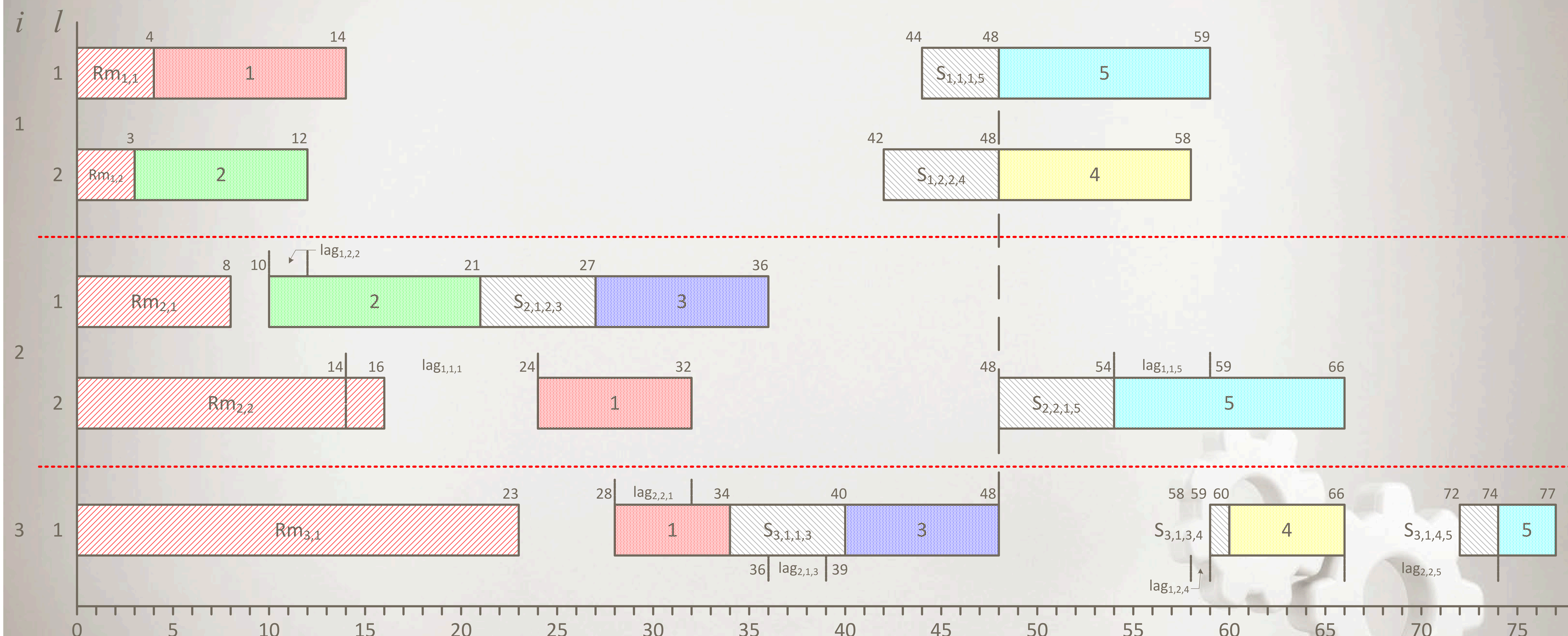
# Complex hybrid problems

# Complex hybrid problems

Let us consider a large number of constraints:

- Sequence dependent setup times in all machines
- Unrelated parallel machines at all stages
- Eligibility
- Stage skipping
- Anticipatory and non-anticipatory setup times
- Precedence relationships among jobs
- Lag times and overlaps between operations
- Release times for machines

# Complex hybrid problems



# Hybrid flowshops. *Iterated Greedy* algorithm

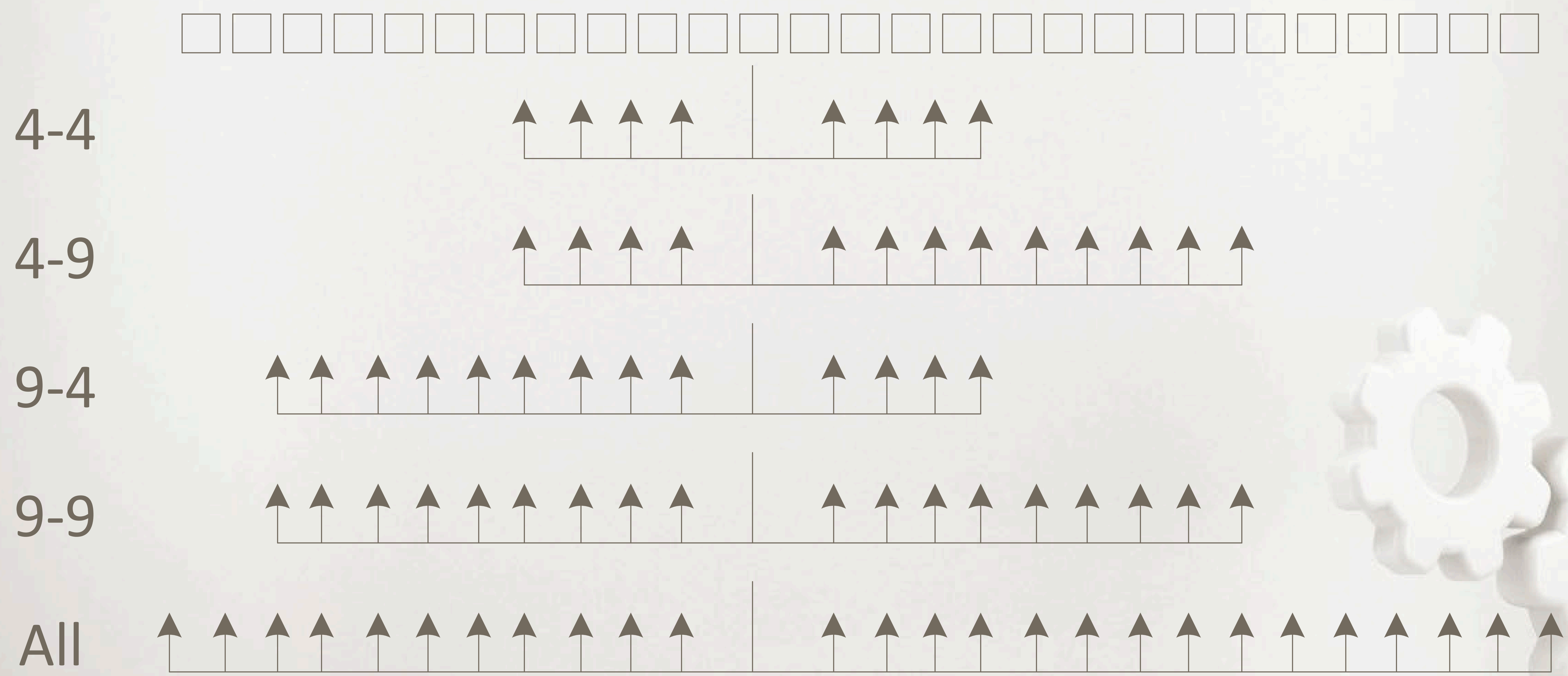
Urlings, Ruiz and Stützle, EJOR (2010)

Solution evaluation very complex. Some changes are needed

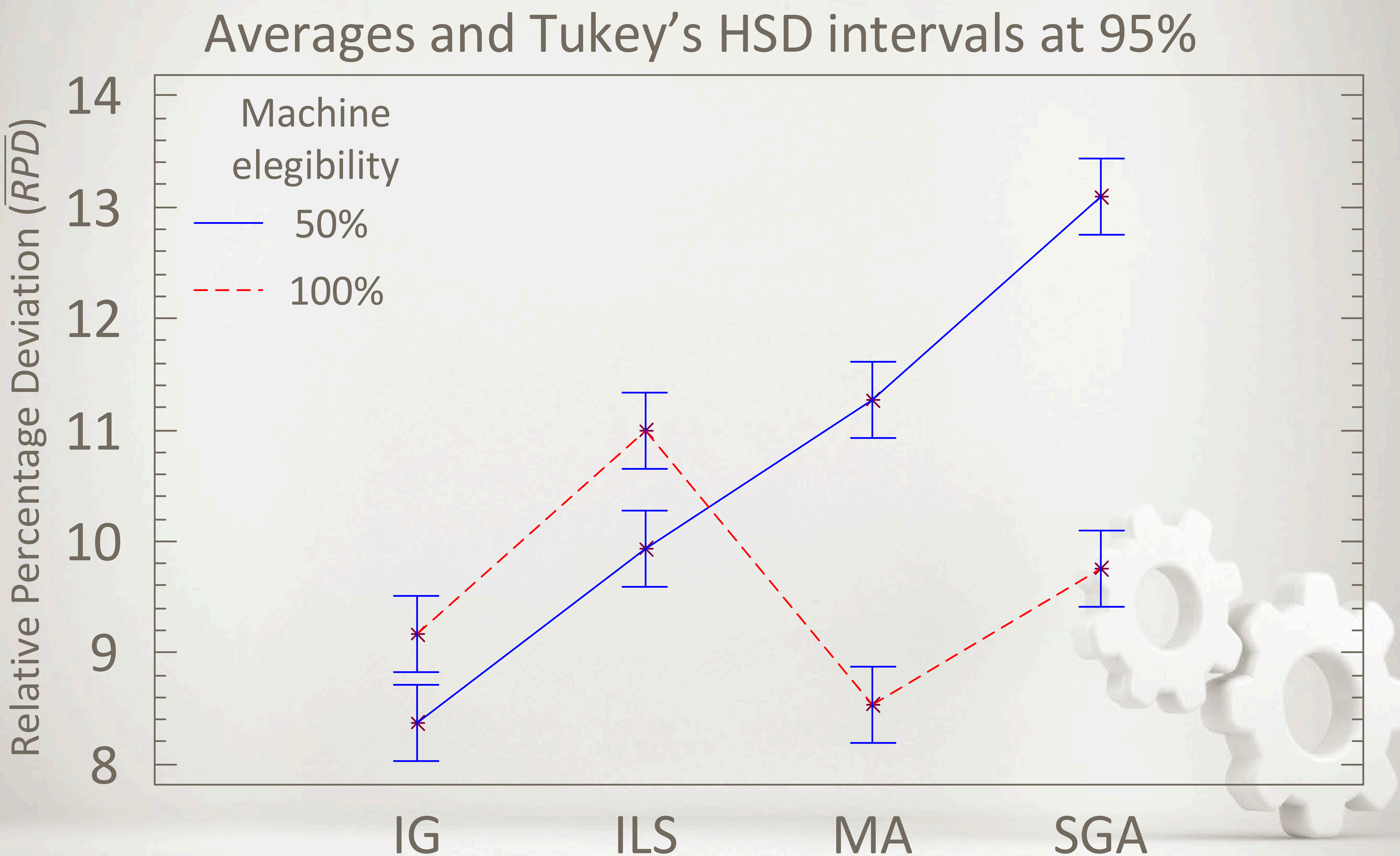
Use only a permutation representation (order in which jobs are launched to the shop) and use assignment heuristics to decide which machine should process each job at each stage

# Hybrid flowshops. *Iterated Greedy* algorithm

Local search still insertion but with increased span



# Hybrid flowshops. *Evaluation*



# 6. Conclusions

IG is basically the iteration of a constructive greedy heuristic with local search

We have seen different problems and examples. The pattern is clear: the simpler, the better

Importance of fair apples-to-apples comparisons and statistical testing

Metaheuristics do not have to be complex to yield good results

# Simple IG Advantages

- Usually very few parameters to calibrate
- Very fast and small memory footprint
- Does not use lots of problem specific knowledge
- Very easy to implement
- Easy to extend to other problems and objectives
- Almost always state-of-the-art results

# IG Drawbacks

Not competitive if there is no good heuristic to start and to base on

Needs speedy local search

Not competitive is solutions are very expensive to evaluate

Very hard to convince referees that simple methods yield better results than complex and exotic metaheuristics

# Rubén Ruiz García

rruiz@eio.upv.es

Departamento de Estadística e
Investigación Operativa
Aplicadas y Calidad

Universitat Politècnica de
València

ruben.ruizgarcia.79

@RubRubRuiz

RG Ruben_Ruiz4