Data Science meets Scheduling

Overview and examples

Patrick De Causmaecker

KU Leuven, CODeS@KULAK

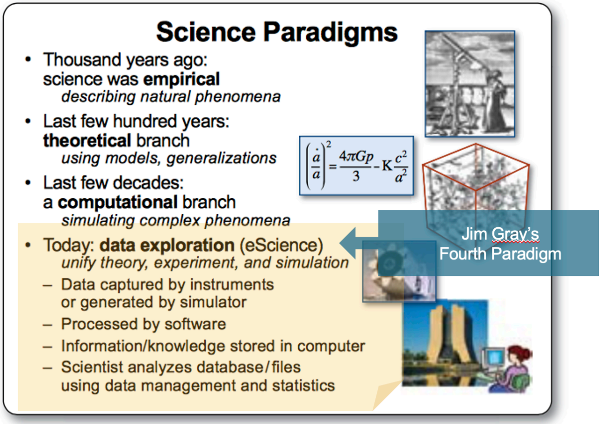
EWG Data Science meets Optimization

11 May 2022, https://schedulingseminar.com

0 Outline

1 Outline

1 Historical links between data science and optimization



Science Paradigms

- Thousand years ago:
  science was **empirical**
  describing natural phenomena
- Last few hundred years:
  theoretical branch
  using models, generalizations
- Last few decades:
  a **computational** branch
  simulating complex phenomena
- Today: **data exploration** (eScience)
  unify theory, experiment, and simulation
  - Data captured by instruments or generated by simulator
  - Processed by software
  - Information/knowledge stored in computer
  - Scientist analyzes database/files using data management and statistics

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{4\pi Gp}{3} - K\frac{c^2}{a^2}$$

Jim Gray's Fourth Paradigm

www.fourthparadigm.com

1 Historical links between data science and optimization

- Susie Sheehy (Oxford and Melbourne), The Matter of Everything (How physics shaped the world)

1 Historical links between data science and optimization

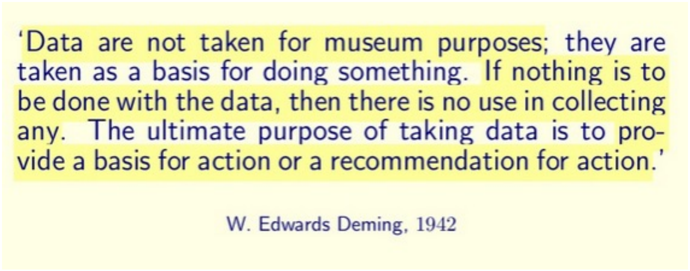
- Susie Sheehy (Oxford and Melbourne), The Matter of Everything (How physics shaped the world)

- Subier Sarkar (Oxford), Cosmology should be led by observations rather than dogma.
  'All the great discoveries have been made by simply building an instrument pointing it at the sky and looking, astronomy is all about serendipity'

1 Historical links between data science and optimization

- Susie Sheehy (Oxford and Melbourne), The Matter of Everything (How physics shaped the world)
- Subier Sarkar (Oxford), Cosmology should be led by observations rather than dogma.
  'All the great discoveries have been made by simply building an instrument pointing it at the sky and looking, astronomy is all about serendipity'
- Jean Francois Puget:
  The role of data science is to enable data based decision making.

1 Historical links between data science and optimization

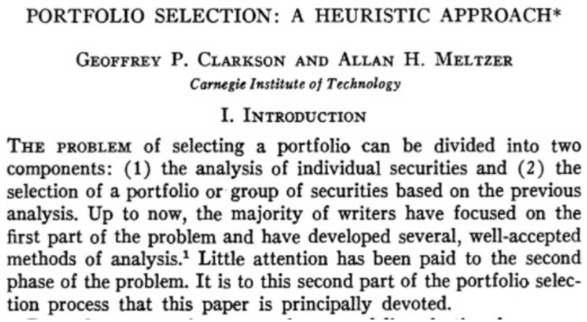


'Data are not taken for museum purposes; they are taken as a basis for doing something. If nothing is to be done with the data, then there is no use in collecting any. The ultimate purpose of taking data is to provide a basis for action or a recommendation for action.'

W. Edwards Deming, 1942

1 Heuristics

- "1960's heuristics"



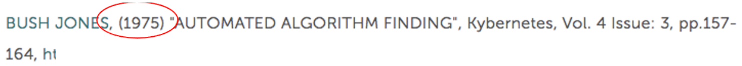
PORTFOLIO SELECTION: A HEURISTIC APPROACH*

GEOFFREY P. CLARKSON AND ALLAN H. MELTZER
Carnegie Institute of Technology

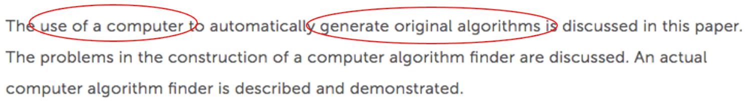
I. INTRODUCTION

THE PROBLEM of selecting a portfolio can be divided into two components: (1) the analysis of individual securities and (2) the selection of a portfolio or group of securities based on the previous analysis. Up to now, the majority of writers have focused on the first part of the problem and have developed several, well-accepted methods of analysis.[1] Little attention has been paid to the second phase of the problem. It is to this second part of the portfolio selection process that this paper is principally devoted.

1 Automated Algorithm Construction


BUSH JONES, (1975) "AUTOMATED ALGORITHM FINDING", Kybernetes, Vol. 4 Issue: 3, pp.157-164, ht

- Cfr: John F. Rice, TheAlgorithm Selection Problem, Advances in Computers, 1976


The use of a computer to automatically generate original algorithms is discussed in this paper. The problems in the construction of a computer algorithm finder are discussed. An actual computer algorithm finder is described and demonstrated.

- Cfr: application of irace for Algorithm Construction
- GA's were suggested by Alan Turing 1950 and first implemented by Baricelli in 1954 . . .

1 Design of Experiments

- Peirce (19th century), Fisher (20th century)
- Predict the outcome based on preconditions
  - Independent variables (input, predictor)
  - Dependent variables (output, response)
  - Control variables
- Fisher's principles
  - Comparison, Randomization, Statistical Replication, Blocking, Orthogonality, Factorial, Combinatorial designs
- Latest possible finding at CERN and FermiLab on the W-boson mass

1 (Beginning) 21st century

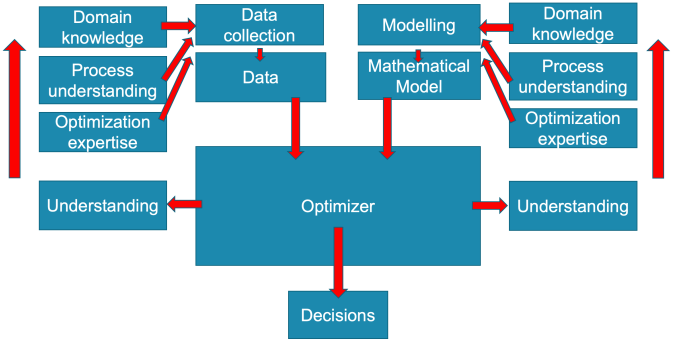
- F-Race (Birattari et al. 2002)
  - Racing algorithm for configuration of metaheuristics
- CALIBRA (Diaz & Laguna 2006)
  - Taguchi fractional design + local search
  - Up to 5 parameters, no interaction
- SPO+ (Hutter et al. 2010)
  - Select promising regions of the design space
  - One instance at the time
- Eiben
  - Eiben, A. & Smit, S.K. (review, 2011). Parameter tuning for configuring and analyzing evolutionary algorithms. doi:10.1016/j.swevo.2011.02.001.

1 (Further Beginning) 21st century

- ParamILS (Hutter et al. 2009)
  - Discrete parameters
  - Local search
- Combination of DOE with other methods
  - (e.g. Gunawan & Lau 2011)
  - Select most promising parameters
  - Reject unimportant parameters
- iRace (Lopez-Ibanez et al.)
  - Iterated racing

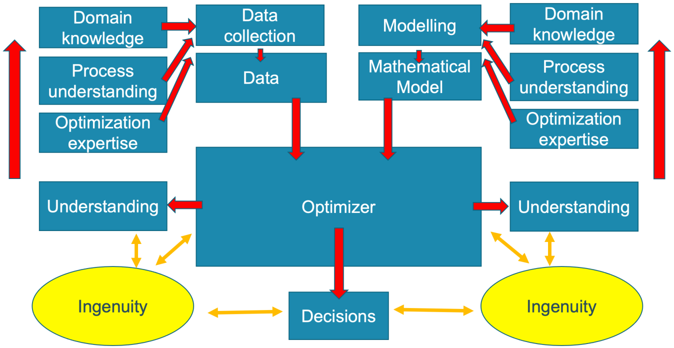
1 We are in an optimization world

.

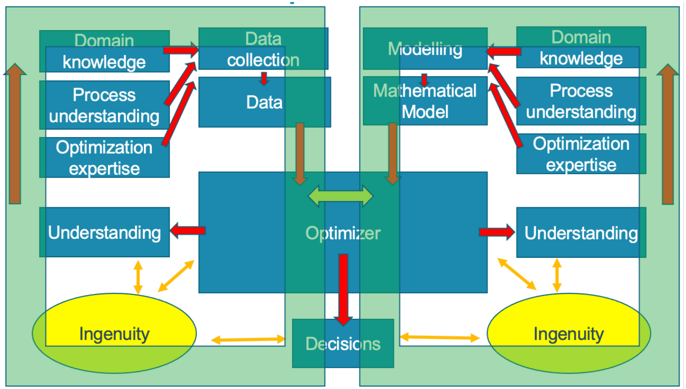


1 We are in an optimization world

.



1 We are in an optimization world



Model

- Understand the problem
  - Interviews
  - Reading
  - Plant visit
- Design a model
- Write down
  - Mathematical Pr.
  - Constraint Pr.
  - Logic Pr.
  - Prob. Pr. . . .
- **Compact Model**

Model

- Understand the problem
  - Interviews
  - Reading
  - Plant visit
- Design a model
- Write down
  - Mathematical Pr.
  - Constraint Pr.
  - Logic Pr.
  - Prob. Pr. ...
- **Compact Model**

Data Science

- Discover / analyse
  - Implicit rules
  - Implicit constraints
  - Historical data
- Improved goals
  - Correct costs
  - What is important?
- Improved parameters
  - Sound estimates
- **Massive Data**

Algorithm

- Analyse the problem
  - Literature
  - Requirements
  - Experience
- Design/Select Algorithm
  - MILP
  - (Meta)Heuristic
- Experiment
  - Instances
  - Compare
- **Efficient Algorithm**

Algorithm

- Analyse the problem
  - Literature
  - Requirements
  - Experience
- Design/Select Algorithm
  - MILP
  - (Meta)Heuristic
- Experiment
  - Instances
  - Compare
- **Efficient Algorithm**

Data Science

- Offline algorithm
  and historical data:
  - Training
  - Tuning
  - Selection
  - Construction
- Online algorithm
  - Active learning/MAB
  - Reinforcement Learning
  - Features/Characteristics
- **Dynamic Algorithm**

2 Outline

2 Data Science for Optimization

- Algorithm-Problem
  - Tuning, Selection, Construction
- Algorithm
  - Embedding Machine Learning Components
- Problem-Algorithm
  - Instance Space

note *Vilas Boas, M.G., Santos, H.G., de Campos Merschmann, L.H., Vanden Berghe, G. Optimal decision trees for the algorithm selection problem: integer programming based approaches. ITOR 2019*

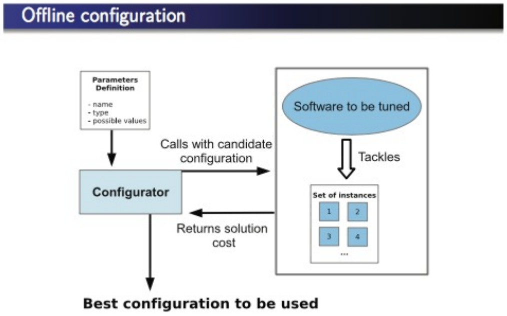
2 Algorithm-Problem

From Pascal Van Hentenrijck, Constraint Programming for Scheduling, schedulingseminar 13/4/2022:

- On-line learning in constraint programming
  A. Schutt, T. Feydy, P.J. Stuckey, M. G. Wallace, Explaining the cumulative propagator, Constraints, 2011

- Off-line learning from exact solvers
  J. Kotary, F. Fioretto, P. Van Hentenryck, Fast Approximations for Job Shop Scheduling: A Lagrangian Dual Deep Learning Method, AAAI 2022

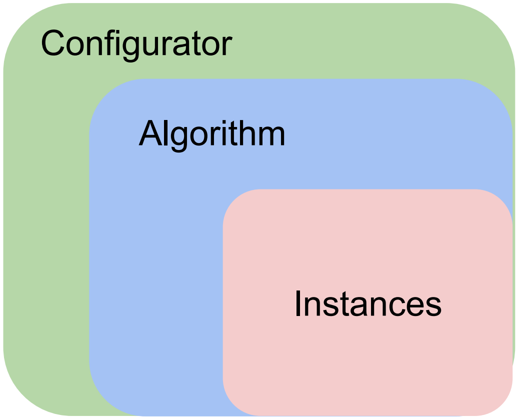
2 Algorithm-Problem example

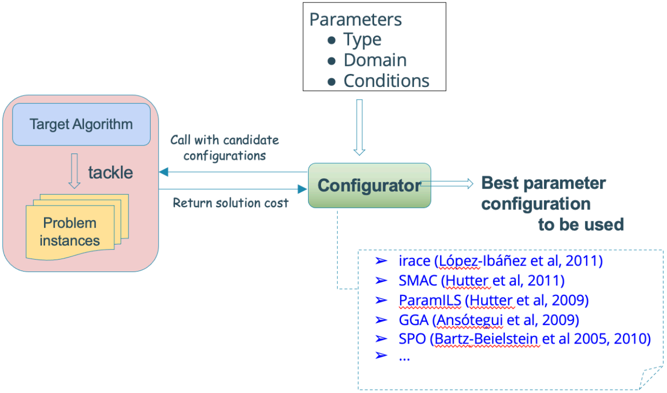


- Automated design of algorithms, T. Stuetzle, M. Lopez-Ibanez, cec2017, http://www.cec2017.org/iles/tutorials/ADA.pdf

2 Opimizing the Optimizer: off-line learning example

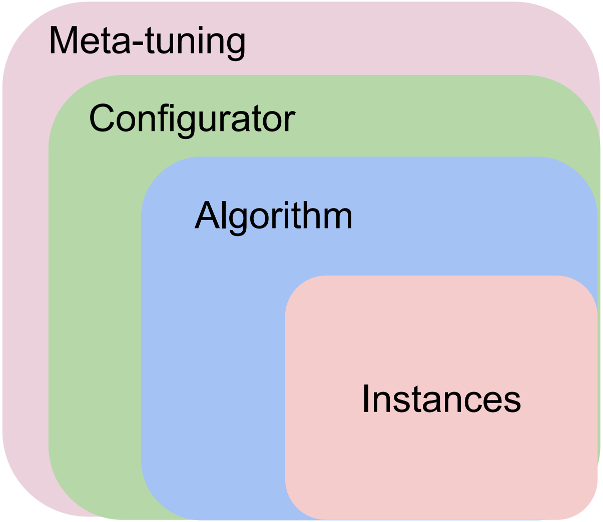
- Configuring irace using surrogate configuration benchmarks (GECCO 2017, ECOM track)
  Tuning, Selection, Construction
- Nguyen Dang, Leslie Pérez Cáceres, Thomas Stützle, Patrick De Causmaecker


Configurator
Algorithm
Instances



- ➢ irace (López-Ibáñez et al, 2011)
- ➢ SMAC (Hutter et al, 2011)
- ➢ ParamILS (Hutter et al, 2009)
- ➢ GGA (Ansótegui et al, 2009)
- ➢ SPO (Bartz-Beielstein et al 2005, 2010)
- ➢ ...

What about the tuning engine?



2 Meta-tuning

- Real benchmarks: extremely expensive
  - An irace run on SPEAR-IBM (budget: 5000 runs) $\rightarrow$ 2 CPU days
  - A meta-tuning on SPEAR-IBM (budget: 5000 irace runs) $\rightarrow$ 27.5 CPU years
- Artificial benchmark set
  - Unclear how to generate
  - Unclear how to match characteristics of real configuration tasks
- Surrogate benchmarks
  - A prediction model: configuration x instance $\rightarrow$ performance value
  - Build on real benchmark data

2 Surrogate configuration benchmarks

- Meta-tuning becomes computationally feasible
  - An irace run on SPEAR-IBM (budget: 5000 runs)
    $\rightarrow$ ~~2 CPU days~~ 5 CPU minutes
  - A meta-tuning on SPEAR-IBM (budget: 5000 irace runs)
    $\rightarrow$ ~~27.5 CPU years~~ 7.5 CPU days

- Useful for the development of configurators
  - Study configurator's parameters
  - Gain insights into configurator's behaviours
  - Better performing configurators

2 Surrogate configuration benchmarks

- Meta-tuning on the surrogate benchmarks indicates that there is room for improvement on irace's performance over its current default configuration.
- Future work
  - Improve the surrogate modelling
  - Build a representative library of surrogate benchmarks
  - Study other state-of-the-art configurators
  - Provide more guidelines for algorithm configurators
  - Algorithm selection for algorithm configurators

2 Data Science for Optimization
Off-line learning inside the algorithm (Data Science for Algorithm Engineering)

- Characterization of neighborhood behaviors in a multi-neighborhood local search algorithm, Dang et al., LION 2016
- Based upon
  - Christiaens et al.: A heuristic approach to the Swap-Body Vehicle Routing Problem. VeRoLog 2014. Oslo, Norway, 22-25 June 2014
  - (See also:
    Jan Christiaens, Greet Vanden Berghe (2020) Slack Induction by String Removals for Vehicle Routing Problems. Transportation Science 54(2):417-433.)

2 Support for algorithm design

Example: The swap-body vehicle routing problem

2 A multi-neighborhood local search for the Swap-body Vehicle Routing problem

Neighborhoods(42)

Cheapest insertion (11)	Ruin recreate (2)	Convert to route (1)
Swap (1)	Remove route (1)	Convert to sub-route (1))
Intra-route 2-opt (1)	Remove sub-route (1)	Add sub-route (1))
Inter-route 2-opt (1)	Remove sub-route with cheapest insertion (1)	Ejection chain (6)
Change swap location (1)	Remove chains (8)	
Merge routes (1)	EachSequenceCheapestInsert (3)	
Split to sub-routes (1)		

2 Research Question

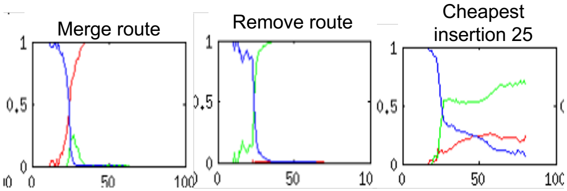
Find groups of similar neighborhoods by

1. Characterizing each neighborhood behaviour as a feature vector (based on information collected from different algorithm runs)
2. Clustering neighborhoods

2 Algorithm Design

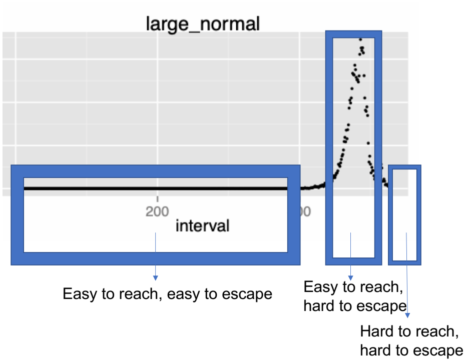
- Characterizing Neighborhoods Behaviours
- Observables

  Try to reflect the changes of neighborhood behaviours according to the hardness of different solution quality regions
- Probability of nothing, worsening improving (sum 1)



2 Algorithm Design

- Identify solution quality regions based on the total number of times all neighborhoods are applied on each interval



2 Algorithm Design

- Cluster the neighborhoods

  Each neighborhood is represented as a vector of

  $\#instancesx\#regionsx\#observables$

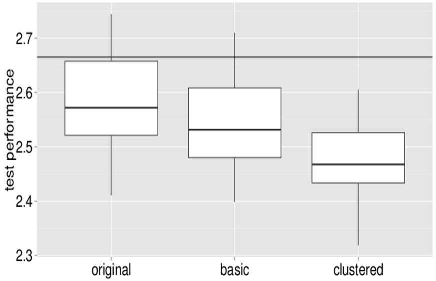
- The clustering problem is of high-dimension, low-sample size (42 individuals, 150 dimensions)

2 Algorithm Design

- Clustering result: 9 clusters
  1. Ejection-chain 3, 4, 5; Remove-chain 1, 2, 3, 6, 7, 8; Remove-sub-route-with-cheapest-insertion;
  2. Swap; Inter-route-two-opt
  3. Cheapest-insertion 10, 15, 20, 25, 35, 50; Each-sequence-cheapest-insertion (2,5), (4,4), (5,2); Remove-chain 4
  4. Cheapest-insertion 1, 2, 3, 4, 5
  5. Change-swap-location; Merge-route
  6. Add-sub-route; Convert-to-sub-route
  7. Ejection-chain 10, 15, 35; Remove-chain 5; Intra-route-two-opt
  8. Ruin-recreate 2, 3
  9. Convert-to-route; Remove-sub-route; Remove-route; Split-to-sub-route

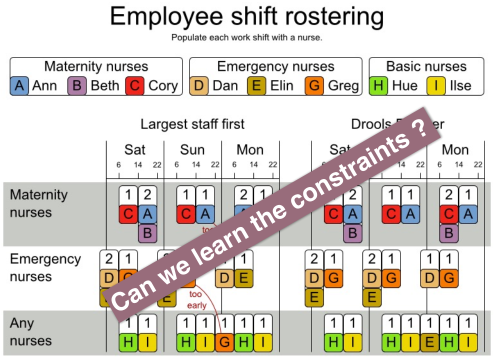
2 Algorithm Design

Applied to algorithm tuning



original vs clustered : *p = 0.00216206*
basic vs clustered : *p = 0.009258918*

2 Learning the Model



Luc De Raedt, Synth Project, Synthesising Inductive Data Models

2 Some References

- Leonardo C.T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. Automatically Designing State-of-the-Art Multi- and Many-Objective Evolutionary Algorithms. Evolutionary Computation, 28(2):195-226, 2020.
- Ansótegui, C., Sellmann, M., Shah, T., Tierney, K. (2021). Learning to Optimize Black-Box Functions with Extreme Limits on the Number of Function Evaluations. In: Simos, D.E., Pardalos, P.M., Kotsireas, I.S. (eds) Learning and Intelligent Optimization. LION 2021. LNCS, vol 12931. Springer, Cham. https://doi.org/10.1007/978-3-030-92121-7_2

2 On-Line Learning: Monte Carlo tree search for combinatorial optimization

- Other example
  A. Hottung, S. Tanaka, K. Tierney, Deep Learning Assisted Heuristic Tree Search for the Container Pre-marshalling Problem, COR 2020
- Current Motivation

Artificial intelligence

Reference: https://becominghuman.ai/alphagos-mastery-continues-3cd01493f539

Combinatorial optimization

Reference: https://www.wired.com/2013/01/traveling-salesman-problem/

2 Monte Carlo tree search

- Learning algorithm for game-playing
  - Estimate move quality by Monte Carlo simulations
  - Better estimates over time: learning

- Operates on game tree:
  - Node = game state
  - Edge = move

2 One Iteration

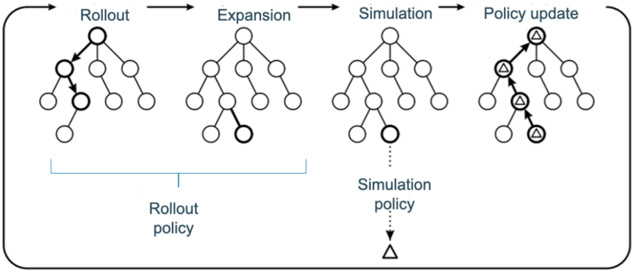


Image adapted from [2]

2 Similarities combinatorial optimization

Model search space as a tree:

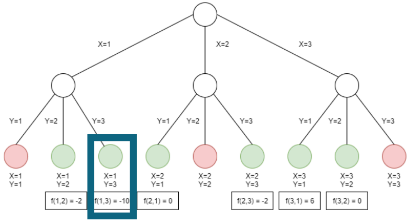


Image adapted from [3]

Minimize $(X-Y)^3 - |X-Y|$
Such that $X \in \{1,2,3\}$
$Y \in \{1,2,3\}$
$X \neq Y$

2 Important differences

Game playing

- One game tree = one game (one tree for Chess, one tree for Go, ...
- Consequence: learne once (lots of time avaliable)
- $\#statesGo < 10^{171}$ $\#statesChess < 10^{46}$
- Pruning a game tree is usually difficult

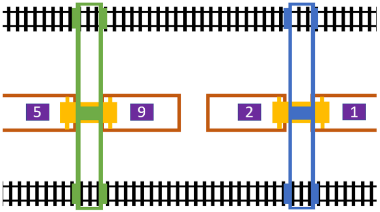
Combinatorial optimization

- One search space tree = one problem instance
- Consequence: many instances = limited time
- # solutions for 1000 binary variables $> 10^{300}$
- Combinatorial structure can be exploited

2 Adaptations in context of combinatorial optimization

- Adapt Monte Carlo tree search to better suit new context.
  Several components need to be customized:
  - Eliminate dominated nodes
  - Several mathematical changes in core of Monte Carlo tree search:
    - "value" of game = win, tie or loss
      $\leftrightarrow$ value of solution = real number
  - Monte Carlo simulation can be done by optimization heuristic
  - Subtree pruning by calculating bounds
  - Limiting search space by beam width

2 Case Study A



- Problem variables:
  - $n = 4$ *containers*
  - $m = 2$ *cranes*
  - *times* $= [5,9,2,1]$
- Goal: cranes process containers + minimize completion time

Thanos, E., Toffolo, T., Santos, H.G., Vancroonenburg, W., Vanden Berghe, G. (2021). The tactical berth allocation problem with time-variant specific quay crane assignments. COMPUTERS & INDUSTRIAL ENGINEERING, (2021)
Vilas Boas, M.G., Santos, H.G., de Campos Merschmann, L.H., Vanden Berghe, G. Optimal decision trees for the algorithm selection problem: integer programming based approaches. ITOR 2019
Jooken J, Leyman P, De Causmaecker P, Wauters T. Exploring search space trees using an adapted version of Monte Carlo tree search for a combinatorial optimization problem. (2020)

2 Results Case study A

- Comparison with three algorithms from literature on two datasets
- Dataset 1: 24 instances of realistic size (16-100 containers)
  - Own algorithm best at 23 out of 24 instances (including ties)
  - 16 proven optimal solutions
- Dataset 2: 24 instances of large size (200-300 containers)
  - Own algorithm best at 24 out 24 instances (including ties)
  - 13 proven optimal solutions

2 Conclusions

- Adaptation of Monte Carlo tree search in combinatorial optimization context
  - All modifications rely on combinatorial structure
- Competitive with state-of-the-art methods on two case studies:
  - Several new best results + proven optimal solutions

3 Outline

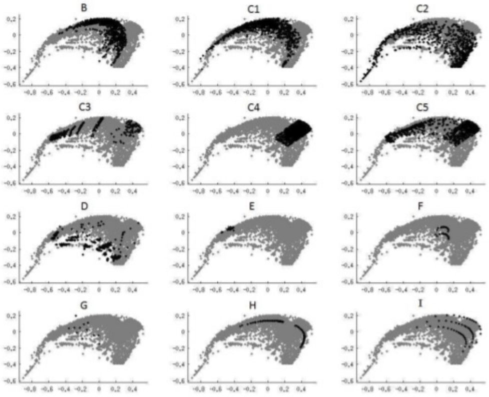
3 Problem Instance space Analysis

Questions:

- Which features of problem instances affect problem instance difficulty (per algorithm)?
- Do phase transitions occur for problem P and where are they situated?
- Which part of the problem instance space is filled by the problem instances of class C for problem P?

(Smith-Miles K, Baatar D, Wreford B, Lewis R. COR, 2014)

3 Graph colouring problem: problem instance space visualization



(figure from Smith-Miles K, Baatar D, Wreford B, Lewis R. COR, 2014)

3 Application : Algorithm selection

- Four components in algorithm selection problem
  - Algorithm space **A: set of algorithms to solve a problem**
  - Performance space: **B: metrics to quantify performance quality**
  - Problem instance space **P: set of problem instances**
  - Feature space **F: set of features that characterize the problems in P**

Find a mapping g from features to algorithms, such that given a problem instance $p \in P$ *with features* $f(p) \in F$ *the algorithm* $g(f(p)) \in A$ *is as good as possible according to* Y.

- Optimization, machine learning
- Theoretical, Emprical

3 Hardness and Algorithm selection

- Where are the hard instances?
  - Pisinger
  - Smith-Miles

3 The 0-1 Knapsack Problem

- Classical problem, studied intensively
- Several other optimization problems are variation or more general (e.g. travelling thief problem, multidimensional knapsack problems and knapsack problem with conflict graphs)
- Any binary integer programming problem can be reduced to the 0-1 knapsack problem!

Bonyadi MR, Michalewicz Z, Barone L. CEC 2013

Kellerer H, Pferschy U, Pisinger D. Knapsack problems 2004

Pferschy U, Schauer J. J. Graph Algorithms Appl.. 2009

3 Algorithms for The 0-1 Knapsack Problem

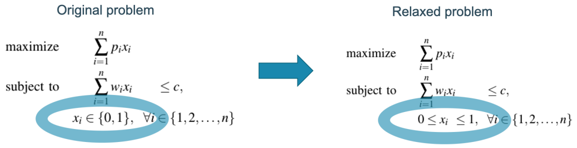
- NP-hard, but solvable in pseudopolynomial time: $O(nc)$
- Many practical algorithms based on branch-and-bound, dynamic programming or hybrid forms, e.g.:

Algorithm	Authors	Publication
MT1	Martello and Toth	EJOR: 1977
MT2	Martello and Toth	Management Science: 1988
Expknap	Pisinger	EJOR: 1995
Minknap	Pisinger	Operations Research: 1997
Combo	Martello, Pisinger and Toth	Management Science: 1999

Bellman R. Dynamic programming. Science. 1966
Martello, S., & Toth, P. An upper bound for the zero-one knapsack problem and a branch and bound algorithm. European Journal of Operational Research. 1977;1(3), 169-175
Martello S, Toth P. A new algorithm for the 0-1 knapsack problem. Management Science. 1988 May;34(5):633-44.
Pisinger D. An expanding-core algorithm for the exact 0-1 knapsack problem. European Journal of Operational Research. 1995 Nov 16;87(1):175-87.
Pisinger D. A minimal algorithm for the 0-1 knapsack problem. Operations Research. 1997 Oct;45(5):758-67.
Martello S, Pisinger D, Toth P. Dynamic programming and strong bounds for the 0-1 knapsack problem. Management science. 1999 Mar;45(3):414-24.

3 Crucial concept: cores

- Most successful algorithms rely on cores:
  - core $\sim$ (small) subset of items hard to decide to include
  - Empirically: go for greedy for all items not in the core
- Greedy: based on exact optimum for relaxed problem



Original problem

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} p_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i x_i \leq c, \\ & x_i \in \{0,1\}, \quad \forall i \in \{1,2,\ldots,n\} \end{aligned}$$

Relaxed problem

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{n} p_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i x_i \leq c, \\ & 0 \leq x_i \leq 1, \quad \forall i \in \{1,2,\ldots,n\} \end{aligned}$$

Martello S, Toth P. Management Science. 1988
Pisinger D. EJOR 1995European Journal of Operational Research. 1995
Pisinger D. Operations Research. 1997

3 Example greedy algorithm for relaxed problem

Capacity = 19

p_i	w_i	$\frac{p_i}{w_i}$
12	9	1.33
7	6	1.17
8	1	8.00
2	5	0.40
6	2	3.00
5	5	1.00
6	5	1.20
12	8	1.50
4	8	0.50

Sort on $\frac{p_i}{w_i}$ $\longrightarrow$ $O(n\log(n))$

Capacity = 19

p_i	w_i	$\frac{p_i}{w_i}$
8	1	8.00
6	2	3.00
12	8	1.50
12	9	1.33
6	5	1.20
7	6	1.17
5	5	1.00
4	8	0.50
2	5	0.40

Optimal fractional x_i $\longrightarrow$

Capacity = 19

p_i	w_i	$\frac{p_i}{w_i}$	x_i
8	1	8.00	1.00
6	2	3.00	1.00
12	8	1.50	1.00
12	9	1.33	0.89
6	5	1.20	0.00
7	6	1.17	0.00
5	5	1.00	0.00
4	8	0.50	0.00
2	5	0.40	0.00

$$Optimum\ relaxed\ problem = 8 + 6 + 12 + \frac{8}{9} * 12 = 36.67$$

3 Core illustration

Capacity = 19

p_i	w_i	$\frac{p_i}{w_i}$	x_i
8	1	8.00	1.00
6	2	3.00	1.00
12	8	1.50	1.00
12	9	1.33	0.89
6	5	1.20	0.00
7	6	1.17	0.00
5	5	1.00	0.00
4	8	0.50	0.00
2	5	0.40	0.00

$$\mathit{Optimum\ relaxed\ problem} = 8 + 6 + 12 + \frac{8}{9} * 12 = 36.67$$

Optimal solutions are very similar except for items in core

$\longleftrightarrow$

Capacity = 19

p_i	w_i	$\frac{p_i}{w_i}$	x_i
8	1	8.00	1.00
6	2	3.00	1.00
12	8	1.50	1.00
12	9	1.33	0.00
6	5	1.20	0.00
7	6	1.17	1.00
5	5	1.00	0.00
4	8	0.50	0.00
2	5	0.40	0.00

$$\mathit{Optimum\ original\ problem} = 8 + 6 + 12 + 7 = 33$$

3 Powerful state-of-the-art algorithms

- Exact optima (for the original problem) can be computed for classes of large problems (> 10000 its) in (milli)seconds
- frequently the $O(nlog(n))$ sorting time is the dominant term
- 0-1 knapsack problem is considered an easy NP-hard problem
  - do not forget decades of research on now state-of-the-art algorithms
- Current problem instances are no longer a challenge
  - despite best algorithm (Combo) being published already in 1999

Kellerer H, Pferschy U, Pisinger D. Knapsack problems 2004
Pisinger D. Operations Research. 1997

3 Important differences

Several papers in literature devoted specifically to hard problem instances for the 0-1 knapsack problem

Theoretical

- Chvátal, 1980
- Gu, Nemhauser and Savelsbergh, 1999
- Jukna and Schnitger, 2011

- Advance our understanding
- Coefficients extremely large while algorithms are limited to 32 or 64 bits

Practical

- Pisinger, 2005
- Smith-Miles, Christiansen and Muñoz, 2021

- 13 different problem instance classes for which hardness is empirically shown
- Theoretical motivation more difficult to obtain

3 Hard problem instances: why and how?

- Weaknesses and strengths of algorithms $\rightarrow$ better algorithms
- 0-1 knapsack problem: how to generate very hard problem instances of modest size?
- $\rightarrow$ noisy multi-group exponential (NMGE) problem instances
- Theoretical and empirical support to indicate hardness

3 Noisy multi-group exponential problem instances

- Problem instance generator: stochastic
  - Input: 7 parameters $n, c, g, f, \epsilon, s, b$
  - Output: problem instance for 0-1 knapsack problem
- n, c, g, s, b are positive integers
- f, ϵ are positive real numbers
- In principle, any valid combination of these parameters will yield valid problem instances, but careful choice yields hard problem instances

3 Important differences

- n items and knapsack with capacity c
- "multi-group": every item belongs to precisely one of g different groups of items
  - First $g-1$ groups: large, exponentially decreasing profits and weights (base of exponent $= b$)
  - Last group (group g: very small profits and weights (between 1 and s)
- Parameter f influences size of last group
- "Noisy": parameter ϵ introduces noise

Parameter	Description
n	Number of items
c	Knapsack capacity
g	Number of groups
f	(Approximate) fraction in last group
ϵ	Noise parameter
s	Small integer
b	Base of exponent

3 Noisy multi-group exponential problem instances

- First $g-1$ groups
  - group $i(1 \leq i \leq g-1) : \#items = \lfloor \frac{n-\lfloor n*f \rfloor}{g-1} \rfloor$
- Last group
  - group g: $\#$ items $=$ $n-(g-1)*\lfloor \frac{\lfloor n*f \rfloor}{g-1} \rfloor (\sim n*f)$

Parameter	Description
n	Number of items
c	Knapsack capacity
g	Number of groups
f	(Approximate) fraction in last group
ϵ	Noise parameter
s	Small integer
b	Base of exponent

3 Noisy multi-group exponential problem instances

Generate two small random integers $r_{1,j}, r_{2,j}$ between 1 and s for every item $j, 1 \leq j \leq n$

- If item j belongs to group $i(1 \leq i \leq g-1)$
  - $p_j = \lfloor (\frac{1}{b^i} + \epsilon)c \rfloor + r_{1,j}$
  - $w_j = \lfloor (\frac{1}{b^i} + \epsilon)c \rfloor + r_{2,j}$
- If item j belongs to group g
  - $p_j = r_{1,j}$
  - $w_j = r_{2,j}$

Parameter	Description
n	Number of items
c	Knapsack capacity
g	Number of groups
f	(Approximate) fraction in last group
ϵ	Noise parameter
s	Small integer
b	Base of exponent

3 Noisy multi-group exponential problem instances

Parameter	Description	Value
n	Number of items	10^3
c	Knapsack capacity	10^{10}
g	Number of groups	11
f	(Approximate) fraction in last group	10^{-1}
ϵ	Noise parameter	10^{-5}
s	Small integer	300
b	Base of exponent	2

10^3 items

	p_i	w_i	group
90 items	5 000 100 128	5 000 100 237	1
	5 000 100 288	5 000 100 078	1
	…	…	…
90 items	2 500 100 037	2 500 100 118	2
	2 500 100 239	2 500 100 107	2
	…	…	…
…	…	…	…
90 items	9 865 729	9 865 884	10
	9 865 688	9 865 701	10
	…	…	…
100 items	213	94	11
	37	299	11
	…	…	…

Intuition for example

- Profits and weight decrease exponentially with approximately factor b
- 1 item of group 1 can fit, but 2 items is just too much
- 3 items of group 2 can fit, but 4 items is just too much
- 7 items of group 3 can fit, but 8 items is just too much
- …
- Added noise introduces diversity within groups
- Large variety of $\frac{p_i}{w_i}$ values in last group

Capacity: $c = 10^{10}$

10^3 items

p_i	w_i	group
5 000 100 128	5 000 100 237	1
5 000 100 288	5 000 100 078	1
…	…	…
2 500 100 037	2 500 100 118	2
2 500 100 239	2 500 100 107	2
…	…	…
…	…	…
9 865 729	9 865 884	10
9 865 688	9 865 701	10
…	…	…
213	94	11
37	299	11
…	…	…

90 items (group 1)
90 items (group 2)
…
90 items (group 10)
100 items (group 11)

3 Near-optimal solutions

- Two theorems explain why the proposed NMGE problem instances are hard
  - Main power: the theorems are statements about problem instances; they are valid for all possible algorithms!
- If there are many near-optimal solutions, branch-and-bound inspired algorithms may become slow, because such solutions are hard to prune from the search space

3 Inclusionwise maximal solutions

Solution $x_{\omega,1}, x_{\omega,2}, \dots, x_{\omega,n}$ is inclusionwise maximal relative to problem instance ω

$$\Leftrightarrow$$

the solution is feasible and no item can be added without violating the capacity constraint $(\nexists j : x_{\omega,j} = 0 \wedge w_{\omega,j} + \sum_{i=1}^{n} x_{\omega,j} w_{\omega,j} \leq c)$

3 Inclusionwise maximal solutions are near-optimal

- (Imprecise version of) Theorem 1: if ω is an NMGE problem instance, then any inclusionwise maximal solution $x_{\omega,1}, x_{\omega,2}, \ldots, x_{\omega,n}$ relative to ω is near-optimal (under certain conditions)
- Intuition
  - All items have a profit-weight ratio close to 1, except for very small items in the last group
  - Hence, optimal objective function value must be close to knapsack capacity c
  - Objective function value of inclusionwise maximal solution is also close to c
- Proof, detail: see paper

Theorem 1 illustration on example

- Example generated using $n = 10^3$, $c = 10^{10}, g = 11, f = 10^{-1}, \epsilon = 10^{-5}, s = 300$ and $b = 2$
- $x_{\omega,1}, x_{\omega,2}, \dots, x_{\omega,n}$ inclusionwise maximal solution
- $x^*_{\omega,1}, x^*_{\omega,2}, \dots, x^*_{\omega,n}$ optimal solution

- Theorem 1 =>

$$\frac{\sum_{i=1}^{n} p_{\omega,i} x_{\omega,i}}{\sum_{i=1}^{n} p_{\omega,i} x^*_{\omega,i}} \geq 98.4\%$$

Hard to prune from the search space

10^3 items

	p_i	w_i	group
90 items	5 000 100 128	5 000 100 237	1
	5 000 100 288	5 000 100 078	1
	…	…	…
90 items	2 500 100 037	2 500 100 118	2
	2 500 100 239	2 500 100 107	2
	…	…	…
…	…	…	…
90 items	9 865 729	9 865 884	10
	9 865 688	9 865 701	10
	…	…	…
100 items	213	94	11
	37	299	11
	…	…	…

3 Impact of cores

- Recall from before: state-of-the-art algorithms for 0-1 knapsack problem critically depend on idea of "cores"
  - Loosely speaking, if an optimal solution contains items with a large variety of profit-weight ratios, cores tend to be big and algorithms tend to be slow

- (Imprecise version of) Theorem 2: if ω is an NMGE problem instance, then any inclusionwise maximal solution $x_{\omega,1}, x_{\omega,2}, \ldots, x_{\omega,n}$ relative to ω contains at least z items from the last group (under certain conditions; the parameters of the problem instance generator influence z)

- Last group has a wide variety of profit-weight ratios

3 Experimental setup

- We generated a large dataset of problem instances using:

$$\begin{aligned}
&n \in \{400, 600, 800, 1000, 1200\}\\
&c \in \{10^6, 10^8, 10^10\}\\
&g \in \{2, 6, 10, 14\}\\
&f \in \{0.1, 0.2, 0.3\}\\
&\epsilon \in \{0, 1, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}\\
&s \in \{100, 200, 300\}\\
&b \in \{2\}
\end{aligned}$$

- Total number of instances $5 \times 3 \times 4 \times 3 \times 6 \times 3 = 3240$
- Total runtime: 810 CPU hours
- Parameters s.t. most combinations meet the necessary conditions for the theorems

3 Preliminary experiment: which algorithm to use?

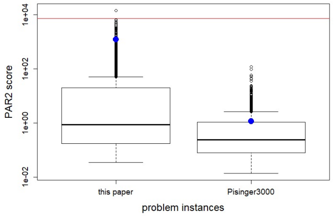
- Preliminary control experiment: compare runtime of Combo with Expknap and Minknap on 100 randomly chosen NMGE problem instances from our dataset
- Time limit of 2 hours per problem instance per algorithm, using HPC

Algorithm	# timeouts
Combo	6
Expknap	79
Minknap	34

» In the remaining experiments, runtime of Combo will be used to empirically measure the hardness of the problem instances

3 Comparison with other problem instances

- Our instances were compared with the 3000 hardest problem instances from "Where are the hard knapsack problems?" (Pisinger) using Combo on cluster (total experiment time around 850 hours)
- PAR2 score = runtime, but penalized timeout by assigning score of 2×7200



3 Comparison with other problem instances

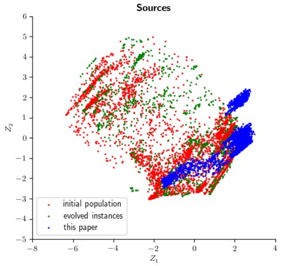
- On average more than 1000 times more time needed for NMGE problem instances
- Hardest NMGE problem instances require 60 times more time than hardest instances from Chvátal 1980
- Even more remarkable, considering that our instances were a lot smaller:

Problem instances	Average n	Average c	Average PAR2
Our instances	800.0	$3.37 * 10^{9}$	$1.24 * 10^{3}$
Pisinger [13]	4022.0	$1.52 * 10^{10}$	$1.15 * 10^{0}$

Chvátal V. Hard knapsack problems. Operations Research. 1980 Dec;28(6):1402-11

3 Problem instance space analysis

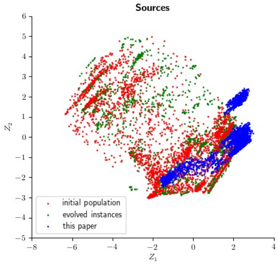
- We used MATILDA to represent our problem instances in a 2D-space and compare with Smith-Miles et al. 2021
- Our problem instances are located near 2 previously unfilled gaps
  - $(Z_1, Z_2) = (2.5, 2)$
  - $(Z_1, Z_2) = (2.5, 0)$

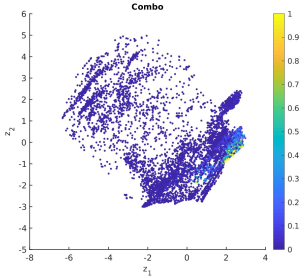


Smith-Miles, K., Christiansen, J., & Muñoz, M. A. (2021). Revisiting "where are the hard knapsack problems?" via instance space analysis.
Smith-Miles, K. (2019). MATILDA, https://matilda.unimelb.edu.au

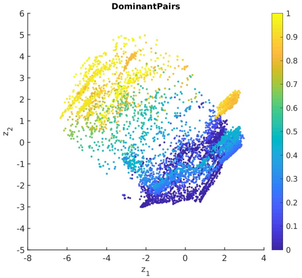
3 Problem instance hardness projection

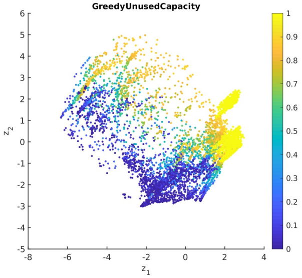
Our problem instances near the second gap are hard:



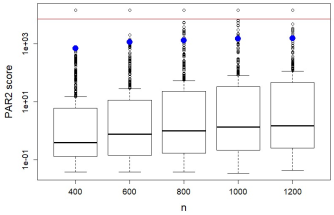


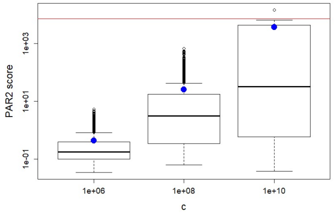
3 Projection of two features



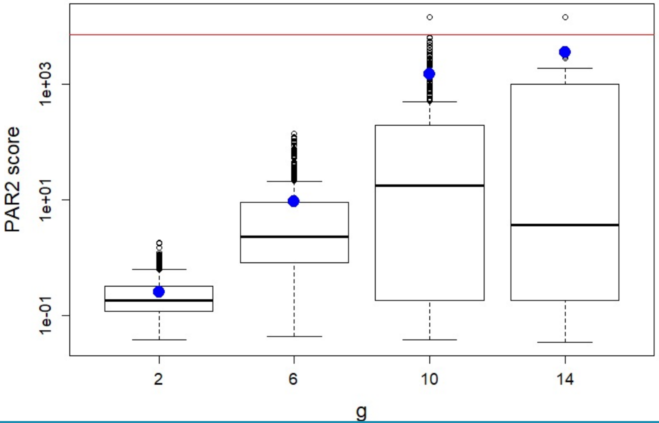


3 Impact problem instance generator parameters

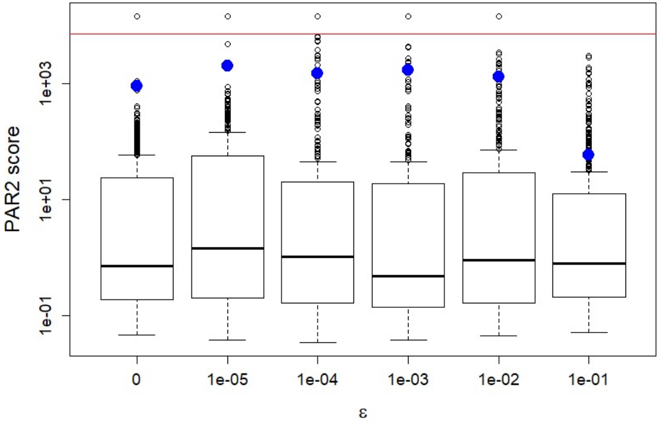




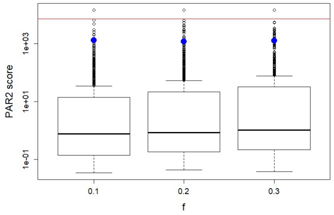
3 Impact problem instance generator parameters

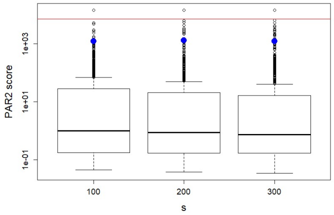


3 Impact problem instance generator parameters



3 Impact problem instance generator parameters





3 Conclusions

- In this work, we proposed a new class of hard problem instances for the 0-1 knapsack problem
- We provided both theoretical and empirical support that these problem instances are hard
- The analysis of the experiments revealed that our problem instances were much harder than the previous hardest problem instances, despite being smaller

3 Several opportunities for further work

- Knapsack problem is closely connected to several other optimization problems. Can our problem instances lead to a series of hard problem instances for several other optimization problems?
- The new problem instances pose a new challenge for algorithm designers. Can the structure of our problem instances be exploited and will this lead to better performing algorithms in general?
- Are existing knapsack features sufficient to characterize the proposed problem instances?

4 Outline

?

5 Outline

5 References I

1 A. Schutt, T. Feydy, P.J. Stuckey, M. G. Wallace, Explaining the cumulative propagator, Constraints, 2011

2 J. Kotary, F. Fioretto, P. Van Hentenryck, Fast Approximations for Job Shop Scheduling: A Lagrangian Dual Deep Learning Method, AAAI 2022, https://doi.org/10.48550/arXiv.2110.06365

3 Dang, N. T. T., De Causmaecker, P., 2016. Characterization of neighborhood behaviours in a multi-neighborhood local search algorithm. LEARNING AND INTELLIGENT OPTIMIZATION (LION 10); 2016; Vol. 10079; pp. 234 - 239, https://lirias.kuleuven.be/handle/123456789/537470

4 N. Dang, L.P. Cáceres, T. Stützle, P. De Causmaecker, Configuring irace using surrogate configuration benchmarks (GECCO 2017, ECOM track)

5 References II

5 Xu L, Hutter F, Hoos HH, Leyton-Brown K. SATzilla: portfolio-based algorithm selection for SAT. Journal of artificial intelligence research. 2008 Jul 1;32:565-606

6 Cameron Browne, Edward Powley, Daniel Whitehouse, Simon Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis and Simon Colton. A survey of Monte Carlo tree search methods, IEEE transactions on computational intelligence and A.I. in games, vol. 4, no. 1, March 2012

7 Jooken J, Leyman P, De Causmaecker P, Wauters T. Exploring search space trees using an adapted version of Monte Carlo tree search for a combinatorial optimization problem. arXiv preprint arXiv:2010.11523. 2020 Oct 22

5 References III

8 Thanos, E., Toffolo, T., Santos, H.G., Vancroonenburg, W., Vanden Berghe, G. (2021). The tactical berth allocation problem with time-variant specific quay crane assignments. COMPUTERS & INDUSTRIAL ENGINEERING, 155, Art.No. ARTN 107168. doi: 10.1016/j.cie.2021.107168

9 Vilas Boas, M.G., Santos, H.G., de Campos Merschmann, L.H., Vanden Berghe, G. (2019). Optimal decision trees for the algorithm selection problem: integer programming based approaches. INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH. doi: 10.1111/itor.12724

10 Jooken J., Leyman P., De Causmaecker P., A new class of hard problem instances for the 0-1 knapsack problem, EJOR, 2022, https://doi.org/10.1016/j.ejor.2021.12.009

5 References IV

11 Leonardo C.T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. Automatically Designing State-of-the-Art Multi- and Many-Objective Evolutionary Algorithms. Evolutionary Computation, 28(2):195-226, 2020.

12 Ansótegui, C., Sellmann, M., Shah, T., Tierney, K. (2021). Learning to Optimize Black-Box Functions with Extreme Limits on the Number of Function Evaluations. In: Simos, D.E., Pardalos, P.M., Kotsireas, I.S. (eds) Learning and Intelligent Optimization. LION 2021. Lecture Notes in Computer Science(), vol 12931. Springer, Cham. https://doi.org/10.1007/978-3-030-92121-7_2

13 Xu L, Hutter F, Hoos HH, Leyton-Brown K. SATzilla: portfolio-based algorithm selection for SAT. Journal of artificial intelligence research. 2008 Jul 1;32:565-606

5 References V

14 Cameron Browne, Edward Powley, Daniel Whitehouse, Simon Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis and Simon Colton. A survey of Monte Carlo tree search methods, IEEE transactions on computational intelligence and A.I. in games, vol. 4, no. 1, March 2012

15 Jooken J, Leyman P, De Causmaecker P, Wauters T. Exploring search space trees using an adapted version of Monte Carlo tree search for a combinatorial optimization problem. arXiv preprint arXiv:2010.11523. 2020 Oct 22

16 Automated design of algorithms, T. Stuetzle, M. Lopez-Ibanez, cec2017, http://www.cec2017.org/iles/tutorials/ADA.pdf

17 Smith-Miles K, Baatar D, Wreford B, Lewis R. COR, 2014

18 Bellman R. Dynamic programming. Science. 1966

5 References VI

19 Martello, S., & Toth, P. An upper bound for the zero-one knapsack problem and a branch and bound algorithm. European Journal of Operational Research. 1977;1(3), 169-175

20 Martello S, Toth P. A new algorithm for the 0-1 knapsack problem. Management Science. 1988 May;34(5):633-44.

21 Pisinger D. An expanding-core algorithm for the exact 0-1 knapsack problem. European Journal of Operational Research. 1995 Nov 16;87(1):175-87.

22 Pisinger D. A minimal algorithm for the 0-1 knapsack problem. Operations Research. 1997 Oct;45(5):758-67.

23 Martello S, Pisinger D, Toth P. Dynamic programming and strong bounds for the 0-1 knapsack problem. Management science. 1999 Mar;45(3):414-24.

5 References VII

24 Christiaens et al.: A heuristic approach to the Swap-Body Vehicle Routing Problem. VeRoLog 2014. Oslo, Norway, 22-25 June 2014

25 Jan Christiaens, Greet Vanden Berghe (2020) Slack Induction by String Removals for Vehicle Routing Problems. Transportation Science 54(2):417-433. https://doi.org/10.1287/trsc.2019.0914

26 Luc De Raedt, Synth Project, Synthesising Inductive Data Models

27 A. Hottung, S. Tanaka, K. Tierney, Deep Learning Assisted Heuristic Tree Search for the Container Pre-marshalling Problem, COR 2020

28 Bonyadi MR, Michalewicz Z, Barone L. CEC 2013

29 Kellerer H, Pferschy U, Pisinger D. Knapsack problems 2004

30 Pferschy U, Schauer J. J. Graph Algorithms Appl.. 2009

5 References VIII

31 Chvátal V. Hard knapsack problems. Operations Research. 1980 Dec;28(6):1402-11

32 Smith-Miles, K., Christiansen, J., & Muñoz, M. A. (2021). Revisiting "where are the hard knapsack problems?" via instance space analysis.

33 Smith-Miles, K. (2019). MATILDA, https://matilda.unimelb.edu.au