Universiteit Leiden
The Netherlands

Game Research Lab
UNIVERSITEIT LEIDEN

Monte Carlo tree search and multi-objective variants

Mike Preuss

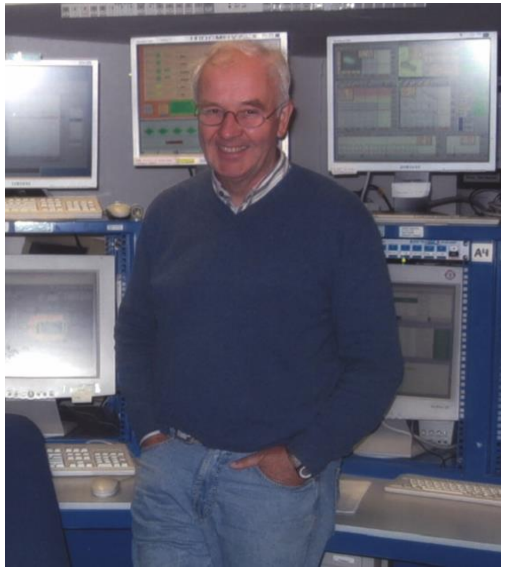
March 12, 2024

picture from Greyerbaby on Pixabay

who’s that Mike?

- associate prof LIACS (computer science Universiteit Leiden/NL):
  game AI (->Chemistry), evolutionary algorithms, social media computing
- started programming with BASIC on Commodore C64 ~ 1982
  (btw 320 x 200 display mode called hires, 38kB usable memory)
- first programming lecture by Hans-Paul Schwefel: Scheme
- physics minor with Dietrich Wegener (experimental particle physics)
- DW on physicists: “unbounded ignorance, but unlimited intelligence”
- doctor father Hans-Paul Schwefel (pure co-incidence)
- PhD on Multimodal Optimization by Means of Evolutionary Algorithms
- since 2007 active in the game AI field
- collaborating quite a lot with Chemistry inside/outside of Leiden

Hans-Paul Schwefel



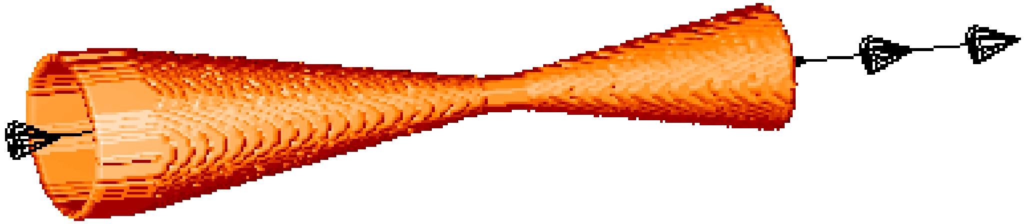
Dietrich Wegener

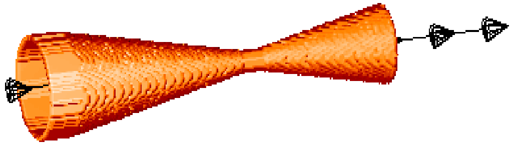
Game Research Lab UNIVERSITEIT LEIDEN

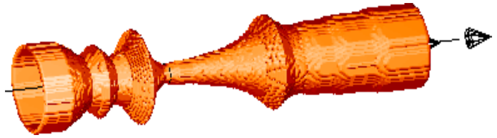
evolutionary computation

- last PhD student of Hans-Paul Schwefel (co-inventor of the Evolution Strategy, most important message: expect the unexpected
- focus on multi-modal optimization (detect several good, different solutions simultaneously)
- work on experimental methodology

(btw, this iconic experiment is manually executed Generative AI)







picture from Hans-Paul Schwefel

what do I want to tell you?

- you may not believe it but there is a lot of uncharted territory in science, CS as well as Chemistry
- there are always quite a lot of alternatives, nothing is purely “right” or “wrong”
- it is VERY important to do fail-fast experimentation (long planning means high investment)
- be ready to let ideas go if they don’t work
- but first try variations in parameters, problems etc
- the stuff I talk about today is complex and best understood by making your own experiments

picture from Tumisu on Pixabay

Game Research Lab
UNIVERSITEIT LEIDEN

what’s on the menue today

- quick MCTS recap
- MOO primer (multi-objective optimization)
- MOO-MCTS algorithms
- first results and outlook

picture from
Markus Winkler
on Pixabay

Monte Carlo Tree Search in a nutshell

- alternative to classical game-tree search (Min-Max)
  for finding good sequences (forget optimality) in big trees
- big >= chess (> 10^40 reachable positions)
- we give up searching thoroughly
- instead, we use:
  - heuristics for the direction and
  - random playouts for detecting what is good

pictures from pixabay

Game Research Lab
UNIVERSITEIT LEIDEN

multi-armed bandit problem

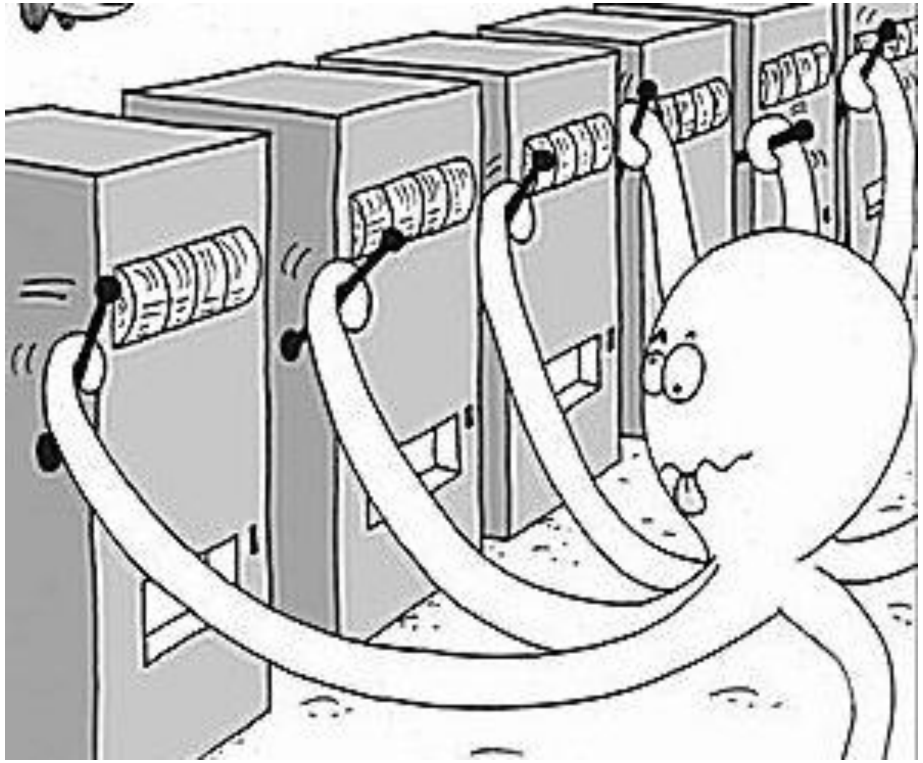
- given that there are several choices, which one do we take?
- this relates to a well-known decision making problem:

  the Multi-Armed Bandit Problem (MAB)

- at each step pull one arm
- noisy/random reward signal
- for later: pull = action, e.g. a reaction in synthesis

- pick the arm so as to:
  - minimize regret (expected loss

    due to not picking the best arm)
  - maximize expected return



Game Research Lab
UNIVERSITEIT LEIDEN

upper confidence bound (UCB)

- balance exploitation and exploration
- an example: UCB1

$$a^* = \operatorname*{argmax}_{\alpha \in A(s)} \left\{ Q(s,a) + C\sqrt{\frac{\ln N(s))}{N(s,a)}} \right\}$$

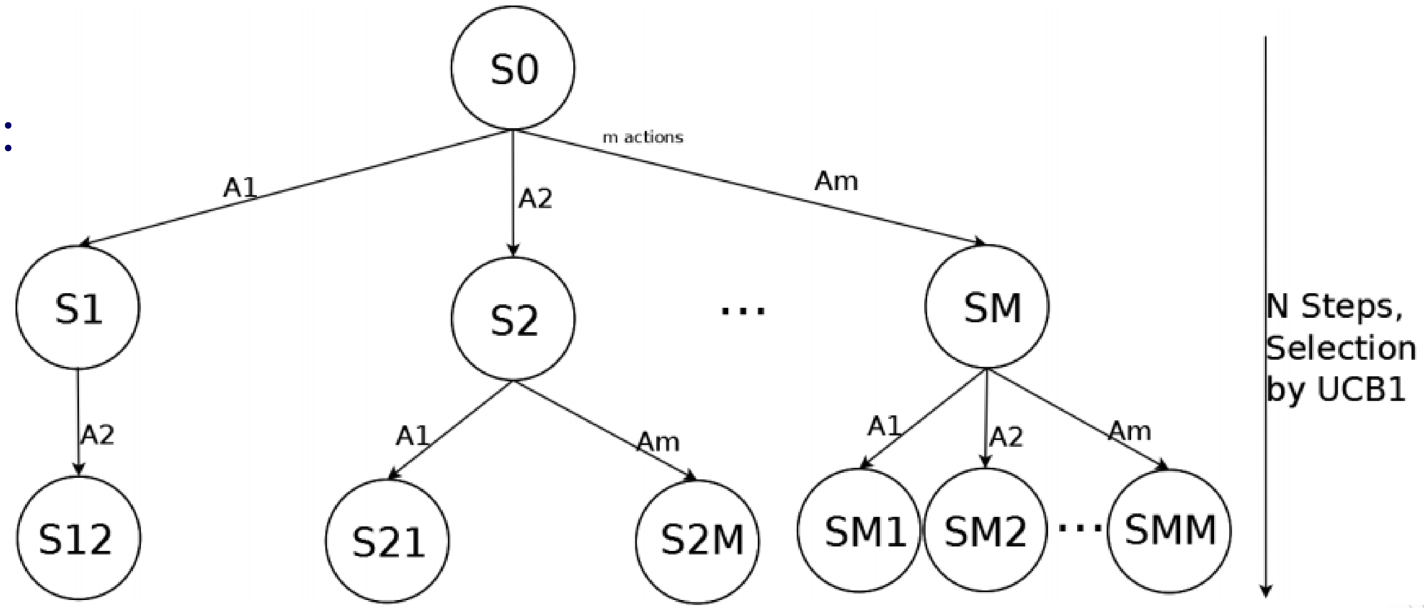
- Q(s,a): average of rewards after taking action a from state s
- N(s): times the state s has been visited
- N(s,a): times the action a has been picked from state s
- C: constant that balances exploitation (Q) and exploration terms
- application dependent, typical value for single player games with rewards in [0,1]: $\sqrt{2}$

- the formula is nicely explained here:

  https://towardsdatascience.com/the-upper-confidence-bound-ucb-bandit-algorithm-c05c2bf4c13f

building a tree with UCB1

- build a tree: search N steps in the future
- the search is not exhaustive: tree grows asymmetrically
- repeat iteratively, return action with e.g.:
    - the highest reward after N steps
    - the highest average reward after 1 step (Q (s, a))
    - the most visited node after 1 step (highest N (s, a) for any a).
    - the highest UCB1 value after 1 step, etc.



Monte Carlo Tree Search (MCTS)

depends on two concepts:

- true value of action can be approximated via random simulations
- the values may be used efficiently to adjust the policy towards a best-first strategy (brute force is too expensive)

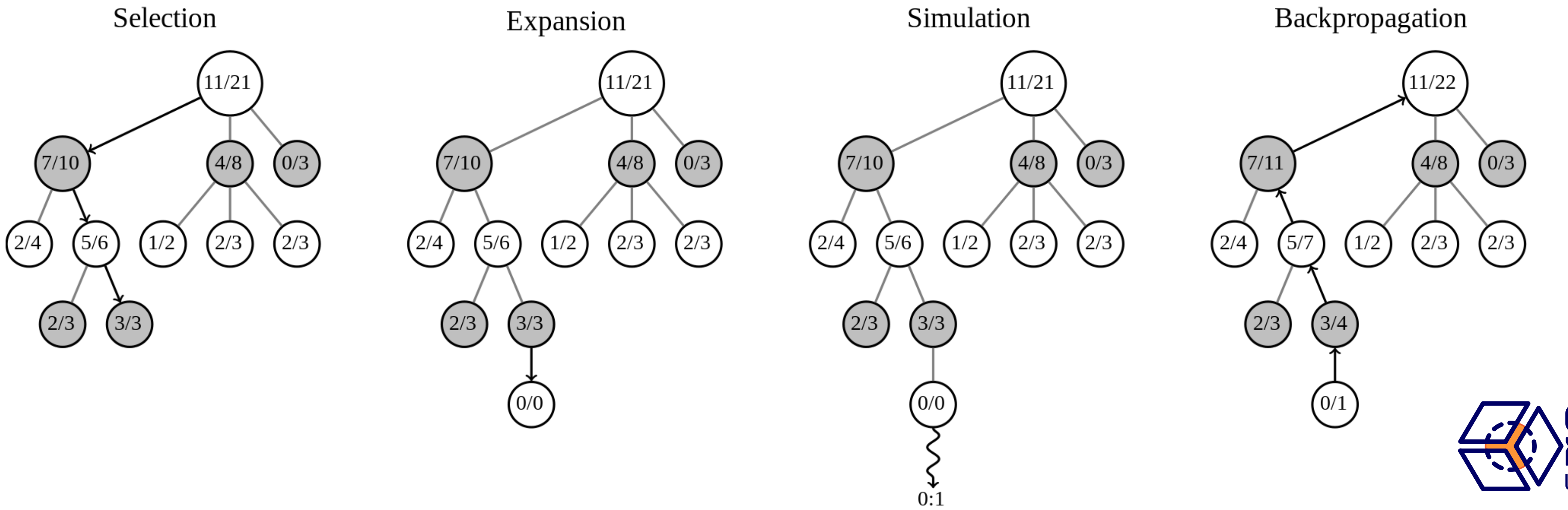
advantages:

- anytime algorithm – stop whenever you like
- needs only game rules:
  - actions
  - terminal state evaluation (win, loss, draw, score)
  - no need for a heuristic function, but can be enhanced with domain knowledge
  - key advantage over MIN MAX

MCTS – the big picture

- selection: select promising node within the tree (by means of UCB1)
- expansion: add new leaf
- simulation: play out the game until we reach a terminal state (and obtain a reward)
- backpropagation: inform the "higher" nodes about move potential

- selection is also called "tree policy", simulation the "default policy"
- for each node, store: incoming action a, associated state s, total simulation reward Q, visit count N



transfer MCTS to Chemistry

- we put the end product into the root
- and search the possible reactions (actions) backwards until we hit compounds we have
- MCTS is not an algorithm, but a family of algorithms
- you have a lot of degrees of freedom at every stage
- we use reaction preferences as trained by deep learning
- resulting trees still huge: we see only a very small part

picture from 4339272 on Pixabay

Game Research Lab UNIVERSITEIT LEIDEN

so what is multi-objective?

- we may have different ideas of what is good
- e.g. a car may be fast, it may be cheap (usually does not go well together, see right)
- good multi-objective algorithms for 2 and 3 objectives, in 4 or 5D it is getting much harder
- we can work with weightings but then we miss some solutions

Opel Manta, fast and cheap ;-)
picture from Jürgen on Pixabay

for this part I use some figures from the Gecco 2018 tutorial slides of my esteemed colleague Dimo Brockhoff: HAL Id: hal-01943586 https://inria.hal.science/hal-01943586

weighted objectives and their problems

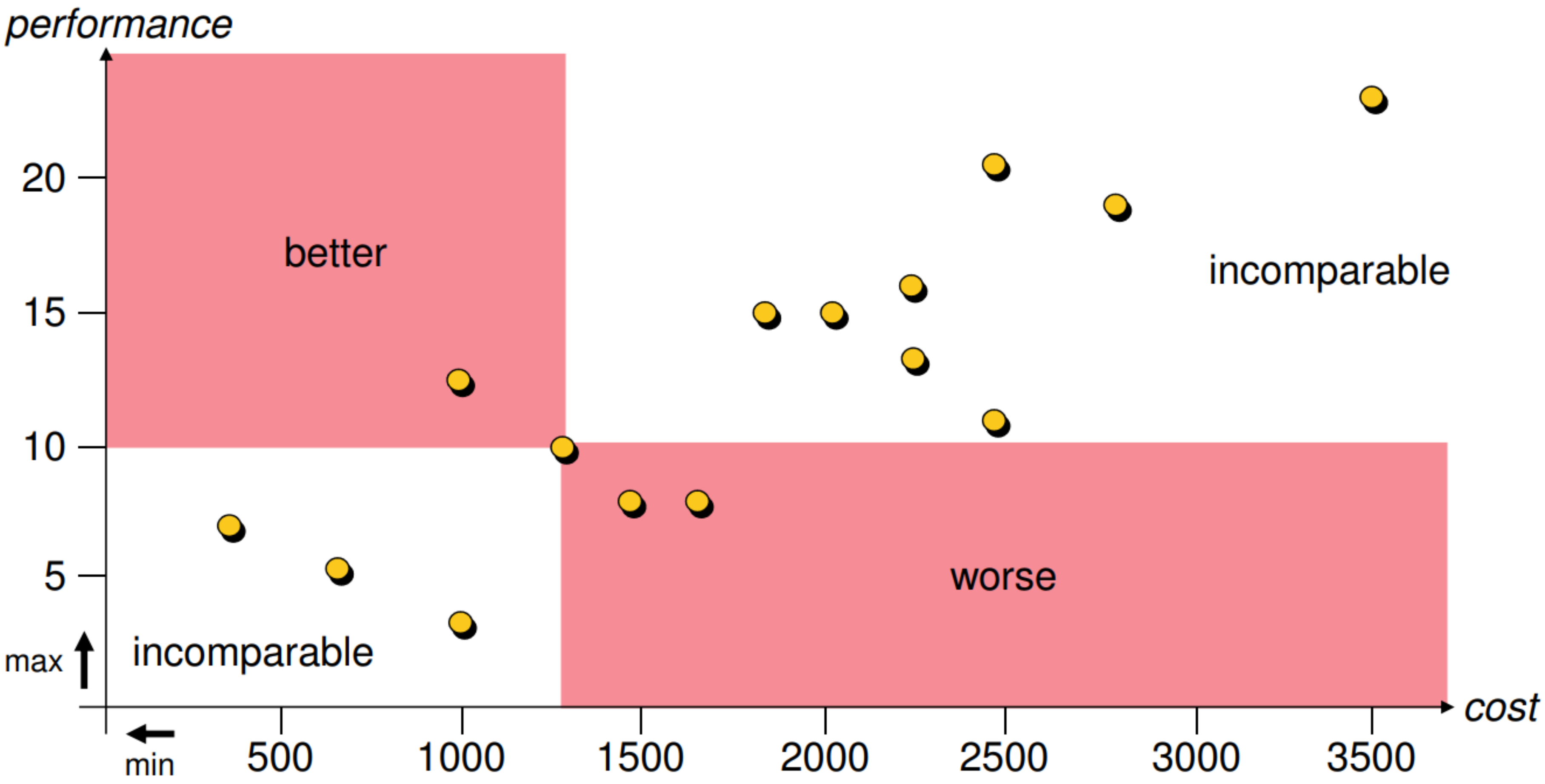
- per weighting we obtain 1 best solution
- we could go over all possible weightings
- then we obtain a set of solutions
- but depending on the problem (concave fronts)

  we may miss some best solutions
- also: how many weight combinations?
- in any case: the solution is a SET!

picture from Tom on Pixabay

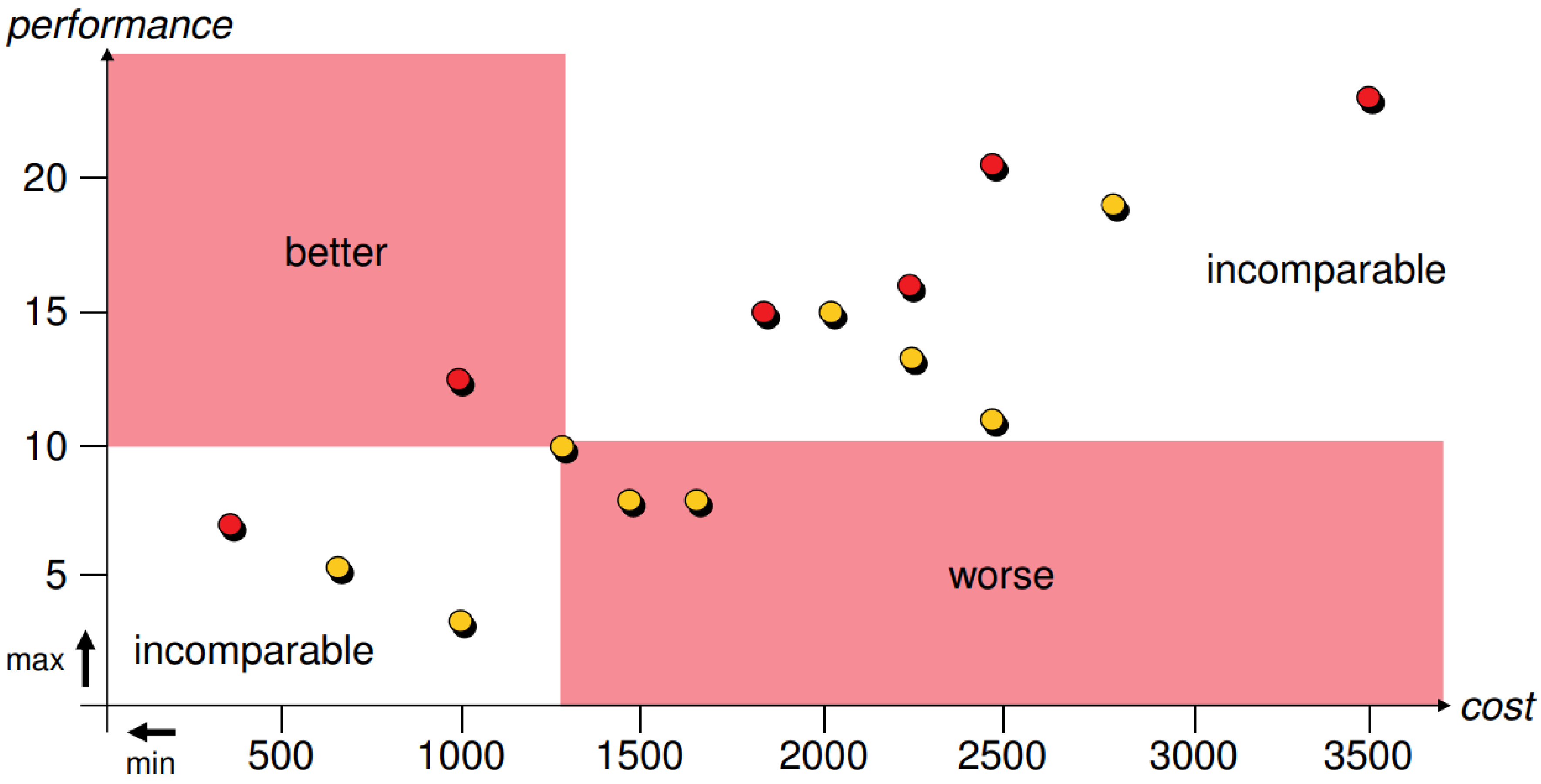
Game Research Lab UNIVERSITEIT LEIDEN

the Pareto notion of ideal compromises



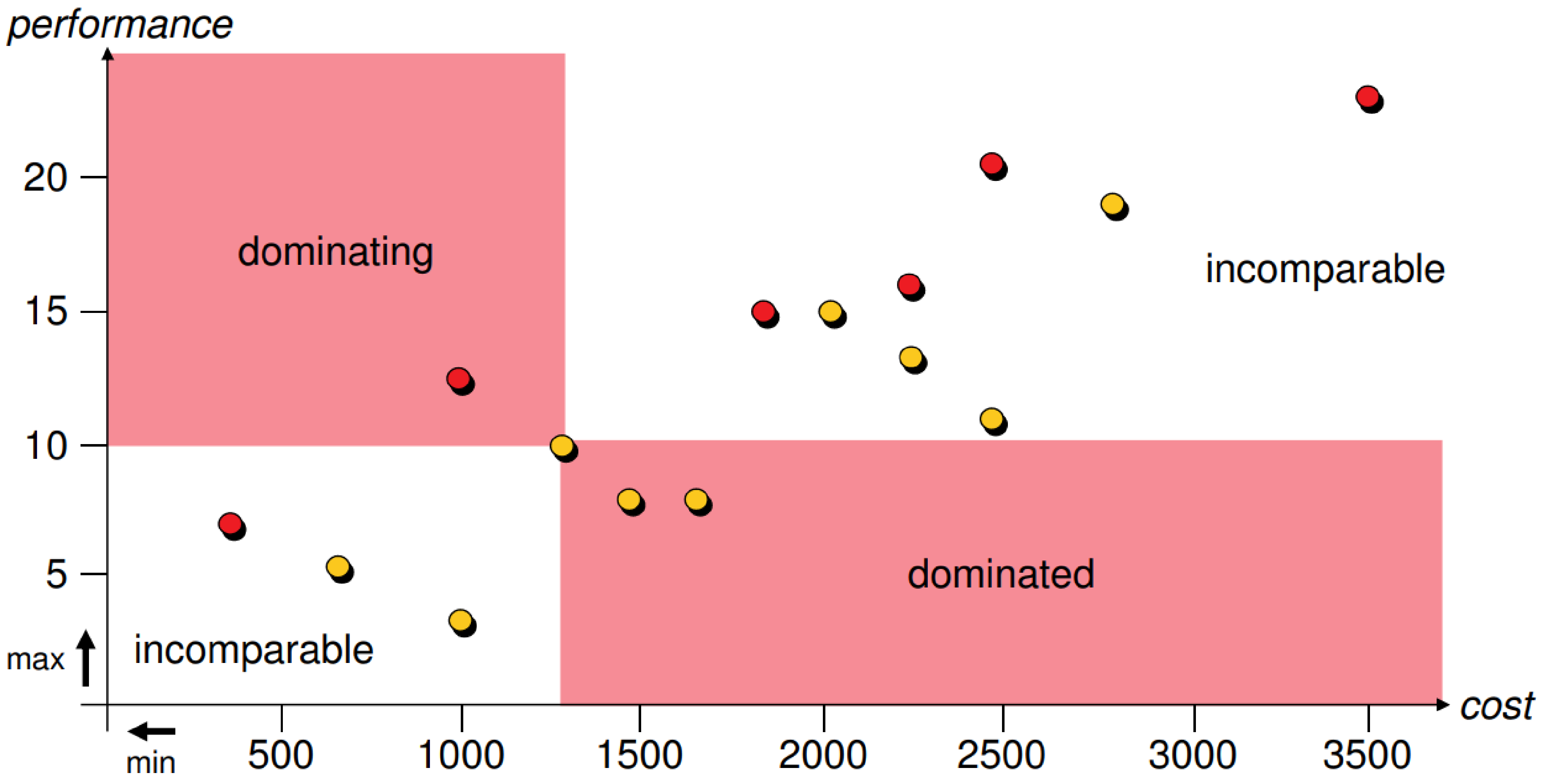
picture from Dimo Brockhoff

Observations: ❶ there is no single optimal solution, but
❷ some solutions (●) are better than others (●)



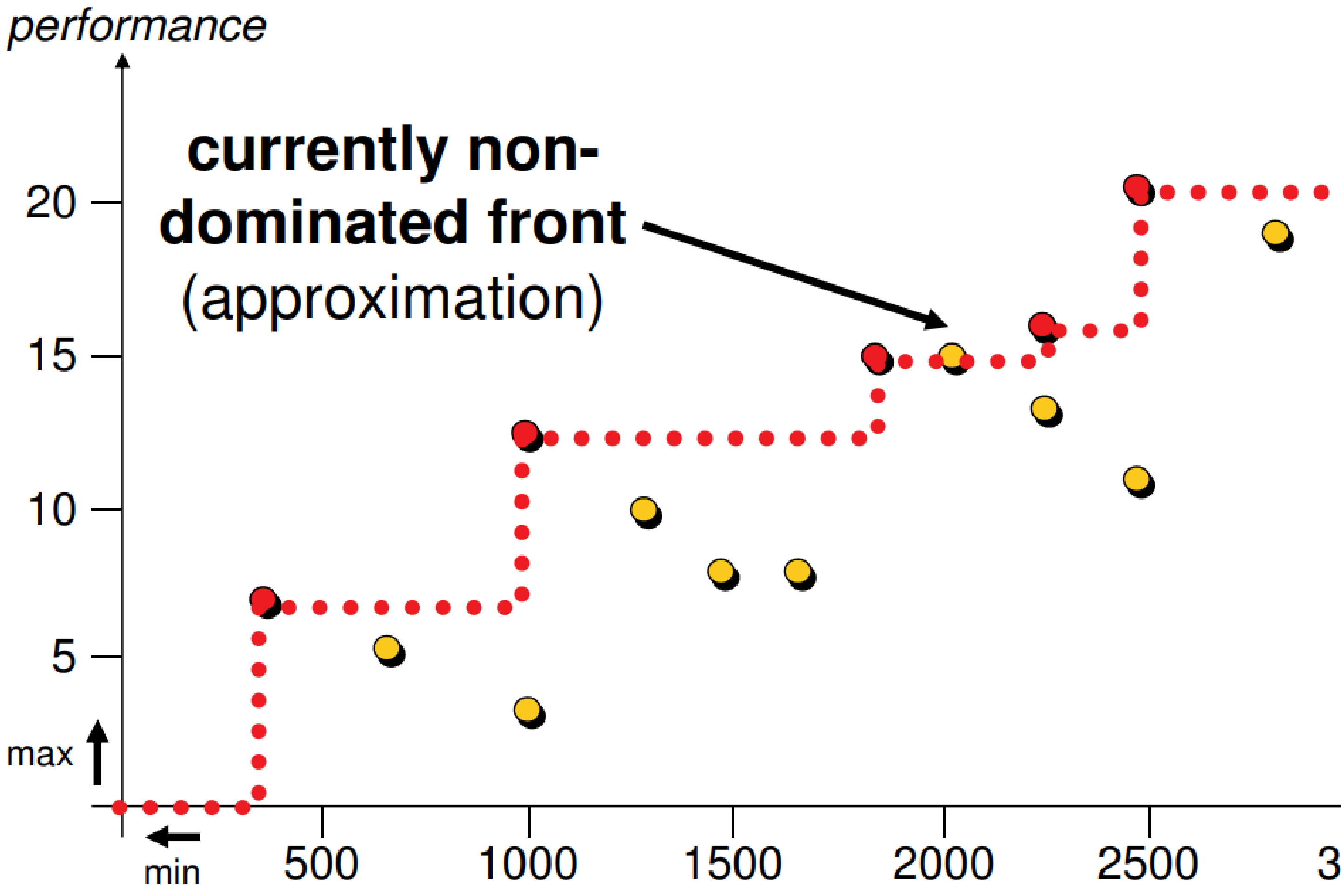
picture from Dimo Brockhoff

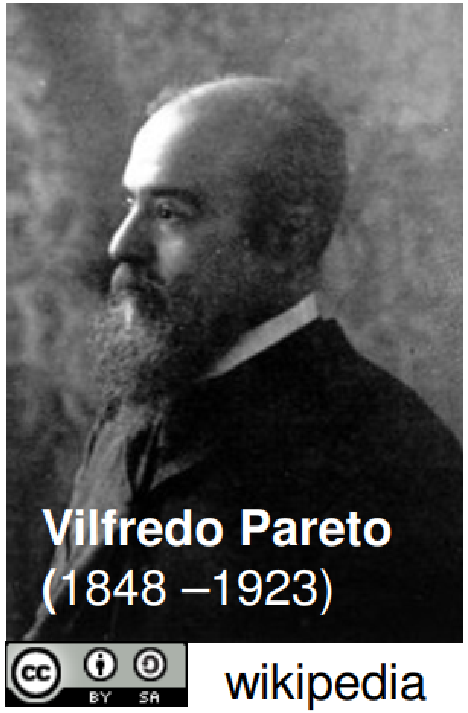
domination and “unnecessary solutions”



picture from Dimo Brockhoff

Pareto set: set of all non-dominated solutions (decision space)
Pareto front: its image in the objective space



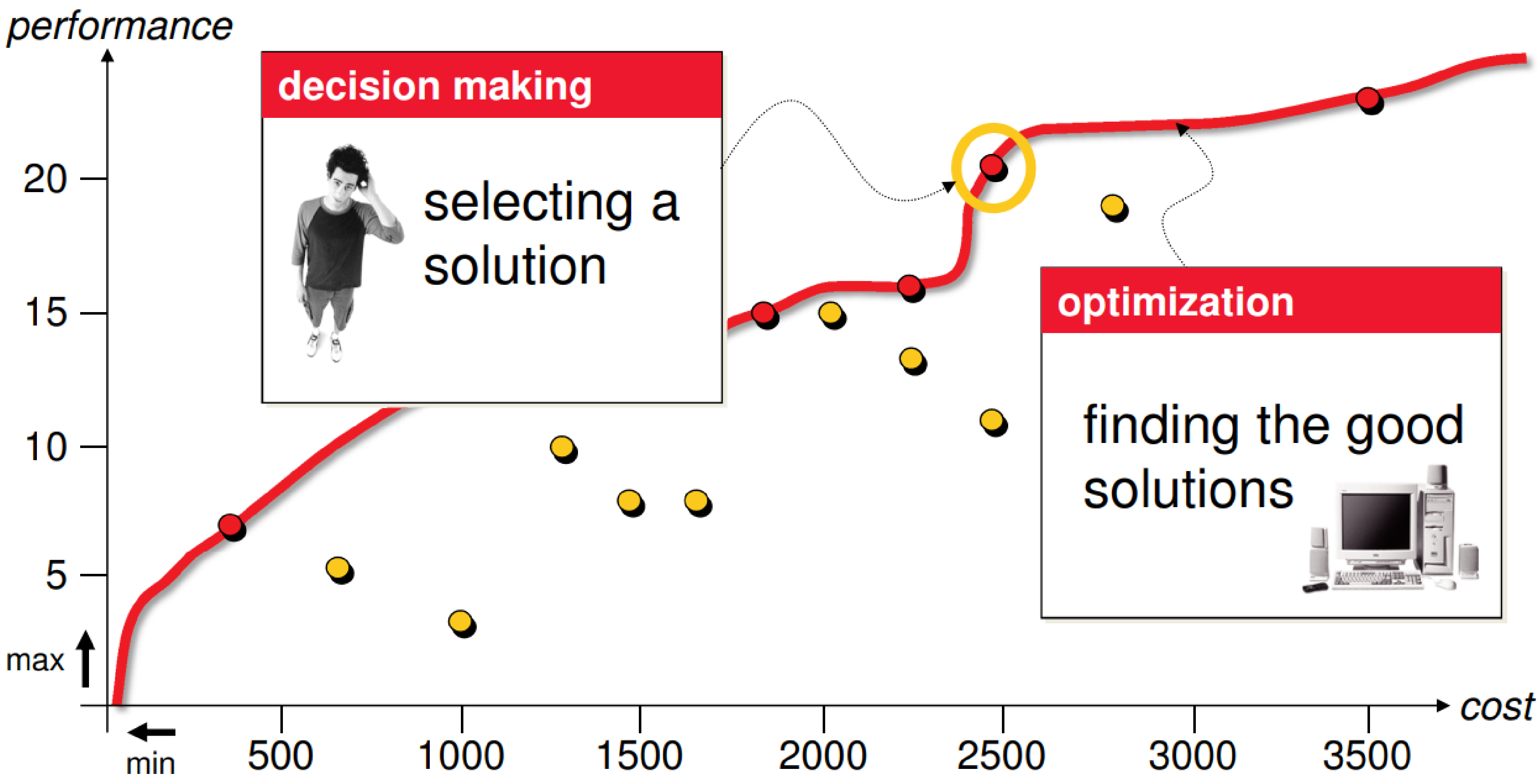


picture from Dimo Brockhoff

Game Research Lab
UNIVERSITEIT LEIDEN

Multiobjective Optimization

combination of optimization of a set and a decision for a solution



picture from Dimo Brockhoff

Game Research Lab UNIVERSITEIT LEIDEN

why shall we want to have a solution set?

- we "explore" the Pareto front/set:
  - how expensive is it to be faster (car example)?
  - we have solutions for lots of weightings at once
- when unsure about a constraint, we can shape it as objective and see what its "cost" is
- we have alternative solutions in case decision makers do not like a specific solution

picture from 育银 戚 on Pixabay

Game Research Lab UNIVERSITEIT LEIDEN

so what is performance?

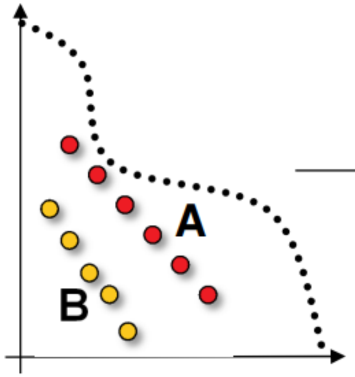
- much more ambiguous than for one objective
- usually the quality of the final set is measured by the "hypervolume" (~area under curve)
- more complicated if speed is also part of the performance measure
- direct comparison of SO and MO algorithms not very meaningful as they provide different types of results (single solutions vs sets of solutions)
- exceptions for multiobjectivization (helper objectives)

picture from Santiago Gonzalez on Pixabay

Game Research Lab
UNIVERSITEIT LEIDEN

what is better?

Goal: compare two Pareto set approximations A and B



	A	B
hypervolume	*432.34*	*420.13*
distance	*0.3308*	*0.4532*
diversity	*0.3637*	*0.3463*
spread	*0.3622*	*0.3601*
cardinality	*6*	*5*

→ "**A** better"

Comparison method C = quality measure(s) + Boolean function

$$A, B \xrightarrow[\textbf{reduction}]{\text{quality measure}} \mathbb{R}^n \xrightarrow[\textbf{interpretation}]{\text{Boolean function}} \text{statement}$$

picture from Dimo Brockhoff

Game Research Lab UNIVERSITEIT LEIDEN

restricted algorithm zoo (of EMOA)

lots of algorithms out, some of the more well known:

- NSGA-II: old, known to be good only for 2 objectives
- SMS-EMOA: fast but depends on hypervolume computation
- MOPSO: PSO driven variant (particle swarm optimization)
- IBEA: (binary) indicator based, not hypervolume
- MOEA/D: based on decomposition (multiple SO in parallel)
- SEMO: old, very simple, used only in theory

- no "best" algorithm because:
  - benchmarking / performance measuring not unambiguous
  - overall goal unclear and too many design possibilities

picture from Pfüderi on Pixabay

Game Research Lab
UNIVERSITEIT LEIDEN

existing MO MCTS algorithms

- Wang2012, Perez2015, Chen2019

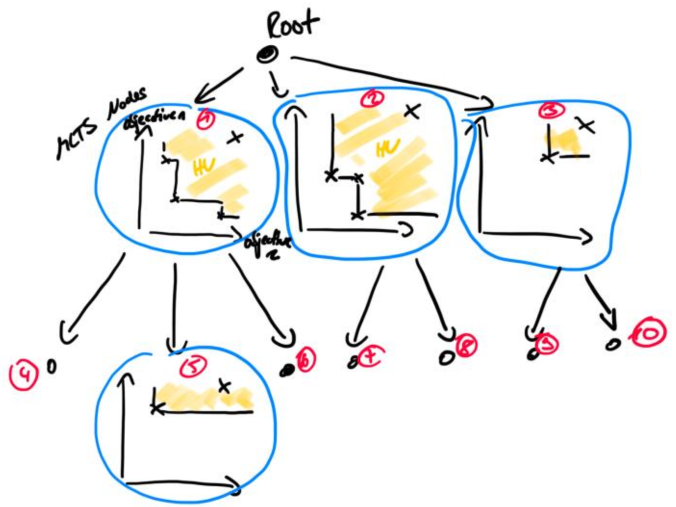
this is almost nothing...

fusion of MO and MCTS leaves a lot of design space

Perez 2015

main idea:

- every node keeps a local Pareto front representing the tree/pareto fronts below it
- Pareto front is backpropagated up the tree
- then remove dominated solutions
- replaces Q(s,a) in UCB1 by the hypervolume
- method has originally nothing to do with Chemistry
- our first adaptation (Alan Hassen, Pfizer/Uni Leiden, ours is quite different) performs similar to weighted SO MCTS
- we tried to push diversity but still small. coarseness of reachable alternatives?



picture from Alan Hassen

Game Research Lab
UNIVERSITEIT LEIDEN

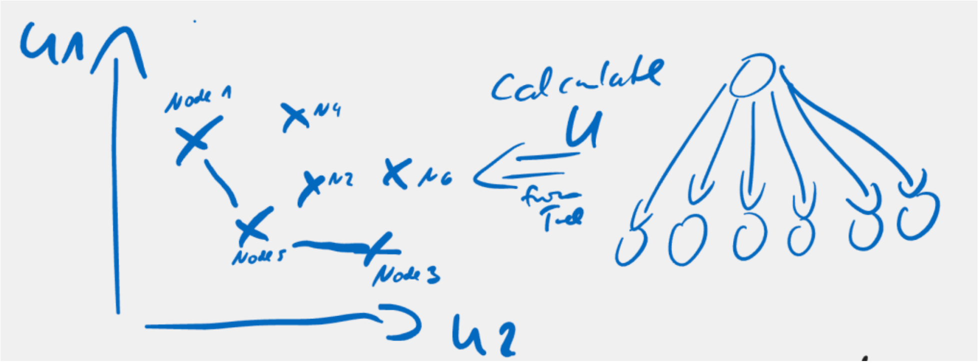
Chen 2019

main idea:

- redefines the UCB formula for selection by a Pareto UCB formula
- meaning that a Pareto set is constructed from the available children in every node
- otherwise MCTS algorithm mainly unchanged
- is currently explored by Astra Zeneca Gothenburg (Samuel Genheden, Helen Lai, Christos Kannas) but also with mixed success

D-Dimensional → Reward

$$U(k) = \frac{v_k.X}{v_k.n} + \sqrt{\frac{4 \ln n + \ln D}{2 v_k.n}}$$



pictures from Alan Hassen

Game Research Lab
UNIVERSITEIT LEIDEN

what do we want?

- MO-MCTS is not supposed to be *faster* than SO-MCTS
- it could provide interesting solution sets
- multi-objective approach only reasonable if objectives are conflicting (otherwise single solution and SO-MCTS faster)
- another problem is sparseness of alternatives: MO methods mostly made for continuous spaces
- we presume we need to “push” for more diversity

so what now?

- lots of possibilities for algorithmic variants not yet tried: e.g. two trees?
- we need to be clearer on what the goal is
- best potential when we have large fronts and these represent many different alternatives
- we need to make sure objectives are as little aligned as possible
- we need to provide "many diverse" actions

picture from shauking on Pixabay

Game Research Lab UNIVERSITEIT LEIDEN

take home

- MCTS is a framework rather than an algorithm
- MOO provides a lot of algorithms/performance measures
- very few existing MO-MCTS approaches
- first practical tests not satisfactory
- there should be potential to improve, not totally clear how
- diversity is probably important

picture from OpenClipart-Vectors on Pixabay

Game Research Lab
UNIVERSITEIT LEIDEN

picture by Oleksandr Pidvalnyi on Pixabay

questions/ comments?

Game Research Lab
UNIVERSITEIT LEIDEN