Is it necessary to perform multi-objective optimization when doing multi-objectivization?

Arina Buzdalova Irina Petrova Maxim Buzdalov

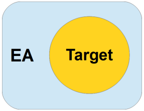
Arina Buzdalova and Irina Petrova were supported by RFBR according to the research project No. 16-31-00380 mol_a.

ITMO UNIVERSITY

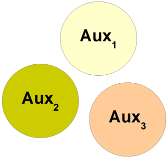
Theory of Randomized Optimization Heuristics
Dagstuhl Seminar 17191
May 12, 2017

What is multi-objectivization?

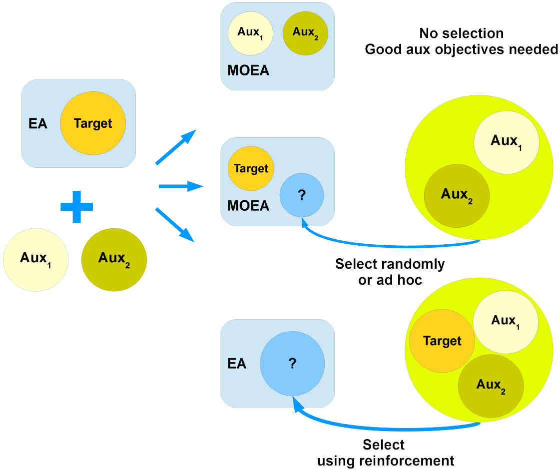
- Goal: find the global optimum of the target objective in less number of fitness evaluations



- Multi-objectivization: introducing of Auxiliary objectives
  - predefined finite set
  - do not have to optimize them



Techniques of using auxiliary objectives



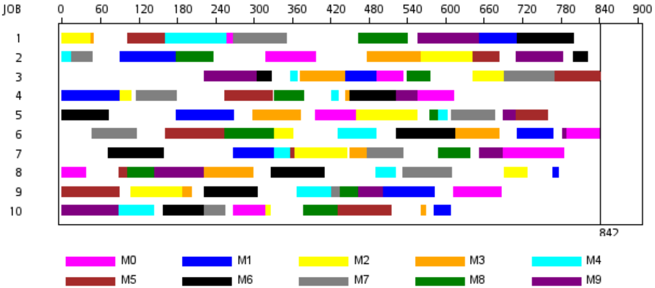
J. D. Knowles, R. A. Watson, D. W. Corne. Reducing Local Optima in Single-Objective Problems by Multi-objectivization. EMO 2001.

M. T. Jensen. Helper-Objectives: Using Multi-Objective Evolutionary Algorithms for Single-Objective Optimization. J. Math. Model. Algorithms 2004.

A. Buzdalova, M. Buzdalov. Increasing Efficiency of Evolutionary Algorithms by Choosing between Auxiliary Fitness Functions with Reinforcement Learning. ICMLA 2012.

Practical Example: Job-Shop Scheduling Problem

- Problem formulation:
  - A job: a predefined sequence of operations
  - Each operation has a specified processing time and a machine
  - No two operations of a job can be processed simultaneously
  - Each machine can process only one operation at time
- Target objective: total flow-time [Lochtefeld, Ciarallo, 2011]
- Auxiliary objectives: flow-time of k jobs



http://support.sas.com/documentation/cdl/en/orcpug/59630/HTML/default/viewer.htm

Analyzed Problem

Target:	$\textsc{OneMax}_d$	1	1	1	1	1	1	1	0	0	0
Aux 1:	$\textsc{OneMax}$	1	1	1	1	1	1	1	1	1	1
Aux 2:	$\textsc{ZeroMax}$	0	0	0	0	0	0	0	0	0	0

Analyzed Problem

Target:	$\textsc{OneMax}_d$	1	1	1	1	1	1	1	0	0	0
Aux 1:	$\textsc{OneMax}$	1	1	1	1	1	1	1	1	1	1
Aux 2:	$\textsc{ZeroMax}$	0	0	0	0	0	0	0	0	0	0
	Example	1	0	0	1	1	1	0	1	0	0

Analyzed Problem

Target:	$\textsc{OneMax}_d$	1	1	1	1	1	1	1	0	0	0	6
Aux 1:	$\textsc{OneMax}$	1	1	1	1	1	1	1	1	1	1	
Aux 2:	$\textsc{ZeroMax}$	0	0	0	0	0	0	0	0	0	0	
	Example	1	0	0	1	1	1	0	1	0	0	

Analyzed Problem

Target:	$\textsc{OneMax}_d$	1	1	1	1	1	1	1	0	0	0	6
Aux 1:	$\textsc{OneMax}$	1	1	1	1	1	1	1	1	1	1	6
Aux 2:	$\textsc{ZeroMax}$	0	0	0	0	0	0	0	0	0	0	
	Example	1	0	0	1	1	1	0	1	0	0	

Analyzed Problem

Target:	$\textsc{OneMax}_d$	1	1	1	1	1	1	1	0	0	0	6
Aux 1:	$\textsc{OneMax}$	1	1	1	1	1	1	1	1	1	1	6
Aux 2:	$\textsc{ZeroMax}$	0	0	0	0	0	0	0	0	0	0	4
	Example	1	0	0	1	1	1	0	1	0	0	

Analyzed Problem

Target:	$\textsc{OneMax}_d$	1	1	1	1	1	1	1	0	0	0	6
Aux 1:	$\textsc{OneMax}$	1	1	1	1	1	1	1	1	1	1	6
Aux 2:	$\textsc{ZeroMax}$	0	0	0	0	0	0	0	0	0	0	4
	Example	1	0	0	1	1	1	0	1	0	0	

Properties

- Auxiliary objectives are conflicting
- They can not speed up optimization of the target objective
- We look at how much they slow down

Analyzed Algorithm: RLS

Individual $x \leftarrow$ a randomly generated individual
while stopping criterion is not reached **do**
    Individual $x' \leftarrow$ mutate x (flip one bit)
    if $F(x') \geq F(x)$ **then**
        $x \leftarrow x'$
    end if
end while

Analyzed Algorithm: RLS

```
Individual x ← a randomly generated individual
while stopping criterion is not reached do
    Individual x′ ← mutate x (flip one bit)
    if F(x′) ≥ F(x) then
        x ← x′
    end if
end while
```

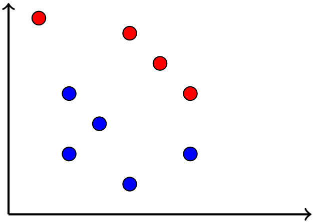
Let me recall: $E[T_{RLS}(\textsc{OneMax} \text{ of length } n)] = \Theta(n \log n)$

Analyzed Algorithm: SEMO

```
Population P ← { a randomly generated individual }
while stopping criterion is not reached do
    Randomly select individual x from P
    Individual x′ ← mutate x (flip one bit)
    Select non-dominated individuals P′ from P ∪ {x′}
    if ∃y ∈ P′ : f(y) = f(x′) and y ≠ x′ then
        Remove y from P′
    end if
    P ← P′
end while
```

Analyzed Algorithm: SEMO

Population $P \leftarrow \{$ a randomly generated individual $\}$
while stopping criterion is not reached **do**
    Randomly select individual x from P
    Individual $x' \leftarrow$ mutate x (flip one bit)
    Select non-dominated individuals P' from $P \cup \{x'\}$
    if $\exists y \in P' : f(y) = f(x')$ **and** $y \neq x'$ **then**
        Remove y from P'
    end if
    $P \leftarrow P'$
end while



Analyzed Algorithm: SEMO

```
Population P ← { a randomly generated individual }
while stopping criterion is not reached do
    Randomly select individual x from P
    Individual x′ ← mutate x (flip one bit)
    Select non-dominated individuals P′ from P ∪ {x′}
    if ∃y ∈ P′ : f(y) = f(x′) and y ≠ x′ then
        Remove y from P′
    end if
    P ← P′
end while
```

Theorem

If expected population size is at most S, then:

$$E[T_{SEMO}] \leq S \cdot \min_i E[T_{RLS} \mid i\text{-th objective is used}]$$

Helper Objectives

Setup

- SEMO running with two objectives: $\{\textsc{OneMax}_d, *\}$
- Second objective is chosen at random every k iterations

Helper Objectives

Setup

- SEMO running with two objectives: $\{\textsc{OneMax}_d, *\}$
- Second objective is chosen at random every k iterations

Analysis

- Population grows by at most 1 on every iteration

Helper Objectives

Setup

- SEMO running with two objectives: $\{\textsc{OneMax}_d, *\}$
- Second objective is chosen at random every k iterations

Analysis

- Population grows by at most 1 on every iteration
- At $\textsc{OneMax} \leftrightarrow \textsc{ZeroMax}$ switch, population becomes $O(1)$

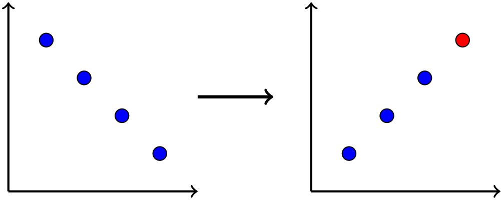
Helper Objectives

Setup

- SEMO running with two objectives: $\{\textsc{OneMax}_d, *\}$
- Second objective is chosen at random every k iterations

Analysis

- Population grows by at most 1 on every iteration
- At $\textsc{OneMax} \leftrightarrow \textsc{ZeroMax}$ switch, population becomes $O(1)$



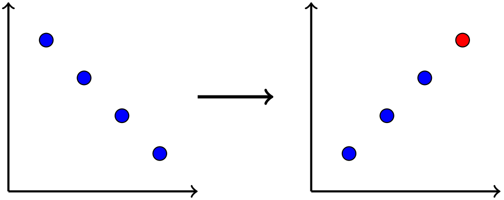
Helper Objectives

Setup

- SEMO running with two objectives: $\{\textsc{OneMax}_d, *\}$
- Second objective is chosen at random every k iterations

Analysis

- Population grows by at most 1 on every iteration
- At $\textsc{OneMax} \leftrightarrow \textsc{ZeroMax}$ switch, population becomes $O(1)$



- Running time: $O(\max(n, k) \cdot n \log n)$

Simultaneous Optimization of All Objectives

Setup

- SEMO running with $\{\textsc{OneMax}_d, \textsc{OneMax}, \textsc{ZeroMax}\}$

Simultaneous Optimization of All Objectives

Setup

- SEMO running with $\{\textsc{OneMax}_d, \textsc{OneMax}, \textsc{ZeroMax}\}$

Analysis

- Idea 1: Population size is at most $n + 1$

Simultaneous Optimization of All Objectives

Setup

- SEMO running with $\{\text{OneMax}_d, \text{OneMax}, \text{ZeroMax}\}$

Analysis

- Idea 1: Population size is at most $n+1$
  - $E[T] \leq (n+1) \cdot \Theta(n \log n) \rightarrow E[T] = O(n^2 \log n)$

Simultaneous Optimization of All Objectives

Setup

- SEMO running with $\{\textsc{OneMax}_d, \textsc{OneMax}, \textsc{ZeroMax}\}$

Analysis

- Idea 1: Population size is at most $n+1$
  - $E[T] \leq (n+1) \cdot \Theta(n \log n) \to E[T] = O(n^2 \log n)$
- Idea 2: for small and large d:
  - the distance between initial and final $\textsc{OneMax}$ value is $\Theta(n)$
  - since the middle of the way, the population size is $\Theta(n)$
  - this yields the $\Omega(n^2 \log n)$ bound

Simultaneous Optimization of All Objectives

Setup

- SEMO running with $\{\text{OneMax}_d, \text{OneMax}, \text{ZeroMax}\}$

Analysis

- Idea 1: Population size is at most $n + 1$
  - $E[T] \leq (n + 1) \cdot \Theta(n \log n) \to E[T] = O(n^2 \log n)$
- Idea 2: for small and large d:
  - the distance between initial and final OneMax value is $\Theta(n)$
  - since the middle of the way, the population size is $\Theta(n)$
  - this yields the $\Omega(n^2 \log n)$ bound
- The intuition says $\Omega(n^2 \log n)$ bound should hold in general

Reinforcement Learning for Single Objective

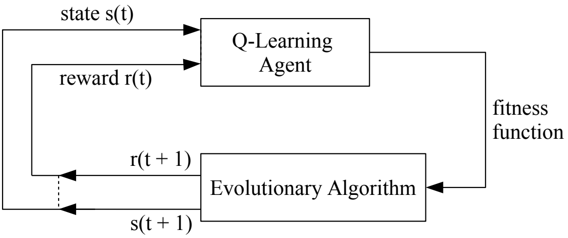
Setup

- Optimizing using RLS
- The objective is chosen by reinforcement learning

Reinforcement Learning for Single Objective

Setup

- Optimizing using RLS
- The objective is chosen by reinforcement learning



$Q(s,h) \leftarrow Q(s,h) + \alpha(r + \gamma \max_{h' \in H} Q(s',h') - Q(s,h))$, s – state, H – set of objectives

Reinforcement Learning for Single Objective

Setup

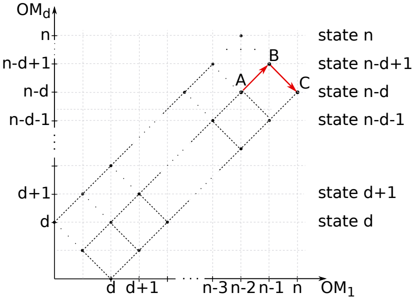
- Optimizing using RLS
- The objective is chosen by reinforcement learning
  - Actions: $\{\textsc{OneMax}_d, \textsc{OneMax}, \textsc{ZeroMax}\}$
  - Reward: the change of $\textsc{OneMax}_d$
  - State: the value of $\textsc{OneMax}_d$

Reinforcement Learning for Single Objective

Setup

- Optimizing using RLS
- The objective is chosen by reinforcement learning
  - Actions: $\{\textsc{OneMax}_d, \textsc{OneMax}, \textsc{ZeroMax}\}$
  - Reward: the change of $\textsc{OneMax}_d$
  - State: the value of $\textsc{OneMax}_d$

- It can learn wrong objective
- $E[T] = \infty$ for $d \in [2; n-2]$
- Ex.: mask = 1001; $1010 \to 1011 \to 1111$



Reinforcement Learning for Second Objective

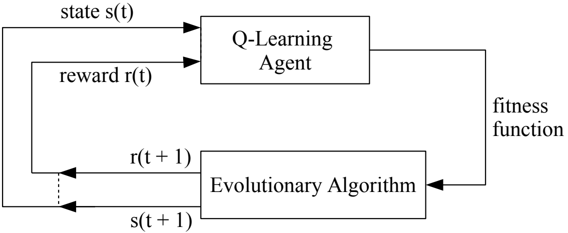
Setup

- SEMO running with two objectives: $\{\textsc{OneMax}_d, *\}$
- Second objective is chosen by reinforcement learning

Reinforcement Learning for Second Objective

Setup

- SEMO running with two objectives: $\{\textsc{OneMax}_d, *\}$
- Second objective is chosen by reinforcement learning



$Q(s,h) \leftarrow Q(s,h) + \alpha(r + \gamma \max_{h' \in H} Q(s',h') - Q(s,h))$, s – state, H – set of objectives

Reinforcement Learning for Second Objective

Setup

- SEMO running with two objectives: $\{\textsc{OneMax}_d, *\}$
- Second objective is chosen by reinforcement learning
  - Actions: $\{\textsc{OneMax}, \textsc{ZeroMax}\}$
  - Reward: the change of $\textsc{OneMax}_d$
  - State: the value of $\textsc{OneMax}_d$

Reinforcement Learning for Second Objective

Setup

- SEMO running with two objectives: $\{\textsc{OneMax}_d, *\}$
- Second objective is chosen by reinforcement learning
  - Actions: $\{\textsc{OneMax}, \textsc{ZeroMax}\}$
  - Reward: the change of $\textsc{OneMax}_d$
  - State: the value of $\textsc{OneMax}_d$

Analysis

- We are elitistic in $\textsc{OneMax}_d \rightarrow$ it only increases

Reinforcement Learning for Second Objective

Setup

- SEMO running with two objectives: $\{\textsc{OneMax}_d, *\}$
- Second objective is chosen by reinforcement learning
  - Actions: $\{\textsc{OneMax}, \textsc{ZeroMax}\}$
  - Reward: the change of $\textsc{OneMax}_d$
  - State: the value of $\textsc{OneMax}_d$

Analysis

- We are elitistic in $\textsc{OneMax}_d \rightarrow$ it only increases
- Thus we never visit a state for which we learned something

Reinforcement Learning for Second Objective

Setup

- SEMO running with two objectives: $\{\textsc{OneMax}_d, *\}$
- Second objective is chosen by reinforcement learning
  - Actions: $\{\textsc{OneMax}, \textsc{ZeroMax}\}$
  - Reward: the change of $\textsc{OneMax}_d$
  - State: the value of $\textsc{OneMax}_d$

Analysis

- We are elitistic in $\textsc{OneMax}_d \to$ it only increases
- Thus we never visit a state for which we learned something
- This means we always select second objective at random

Reinforcement Learning for Second Objective

Setup

- SEMO running with two objectives: $\{\textsc{OneMax}_d, *\}$
- Second objective is chosen by reinforcement learning
  - Actions: $\{\textsc{OneMax}, \textsc{ZeroMax}\}$
  - Reward: the change of $\textsc{OneMax}_d$
  - State: the value of $\textsc{OneMax}_d$

Analysis

- We are elitistic in $\textsc{OneMax}_d \rightarrow$ it only increases
- Thus we never visit a state for which we learned something
- This means we always select second objective at random
- Runtime is $O(n \log n)$

Preserving the best solution (single-objective)

Setup

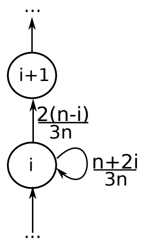
- Optimizing using RLS
- The objective is chosen by reinforcement learning
  - Actions: $\{\textsc{OneMax}_d, \textsc{OneMax}, \textsc{ZeroMax}\}$
  - Reward: the change of $\textsc{OneMax}_d$
  - State: the value of $\textsc{OneMax}_d$
  - Target-worsening individuals are not accepted (even if the auxiliary objective allows)

Preserving the best solution (single-objective)

Setup

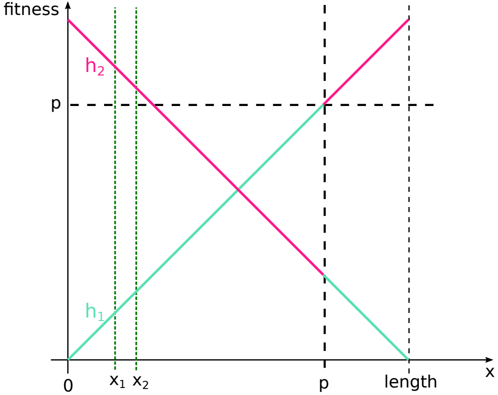
- Optimizing using RLS
- The objective is chosen by reinforcement learning
  - Actions: $\{\textsc{OneMax}_d, \textsc{OneMax}, \textsc{ZeroMax}\}$
  - Reward: the change of $\textsc{OneMax}_d$
  - State: the value of $\textsc{OneMax}_d$
  - Target-worsening individuals are not accepted (even if the auxiliary objective allows)

- $T_i = \frac{3}{2} \cdot (1 + \frac{i}{n-i})$
- Runtime expectation: $\sum_{i=0}^{n-1} T_i = O(n \log n)$
- Works as good as the multi-objective algorithm



Problem with useful and harmful auxiliary objectives

- Target objective: LeadingOnes
- Notice: OneMax can be solved faster and has the same optimum
- Dynamic auxiliary objectives based on OneMax and ZeroMax:



Empirical results: RL in single-objective algorithm

Number of fitness evaluations until optimum is found (averaged)

		Preserving			Preserving, no learn. when bad			No preserving
Parameters	RLS	ss, $\varepsilon = 0.1$	ts, $\varepsilon = 0$	ts, $\varepsilon = 0.1$	ss, $\varepsilon = 0.1$	ts, $\varepsilon = 0$	ts, $\varepsilon = 0.1$	all setups
n	LeadingOnes							
141	$1.00 \cdot 10^4$	$4.61 \cdot 10^3$	$7.20 \cdot 10^3$	$7.80 \cdot 10^3$	$1.36 \cdot 10^4$	$1.49 \cdot 10^4$	$1.49 \cdot 10^4$	∞
151	$1.13 \cdot 10^4$	$5.08 \cdot 10^3$	$8.33 \cdot 10^3$	$8.90 \cdot 10^3$	$1.57 \cdot 10^4$	$1.72 \cdot 10^4$	$1.72 \cdot 10^4$	∞
161	$1.30 \cdot 10^4$	$5.44 \cdot 10^3$	$9.39 \cdot 10^3$	$1.01 \cdot 10^4$	$1.81 \cdot 10^4$	$1.94 \cdot 10^4$	$1.96 \cdot 10^4$	∞
171	$1.45 \cdot 10^4$	$6.04 \cdot 10^3$	$1.06 \cdot 10^4$	$1.13 \cdot 10^4$	$2.05 \cdot 10^4$	$2.18 \cdot 10^4$	$2.19 \cdot 10^4$	∞
181	$1.65 \cdot 10^4$	$6.60 \cdot 10^3$	$1.18 \cdot 10^4$	$1.27 \cdot 10^4$	$2.29 \cdot 10^4$	$2.47 \cdot 10^4$	$2.46 \cdot 10^4$	∞
191	$1.81 \cdot 10^4$	$7.28 \cdot 10^3$	$1.33 \cdot 10^4$	$1.41 \cdot 10^4$	$2.58 \cdot 10^4$	$2.73 \cdot 10^4$	$2.73 \cdot 10^4$	∞
n, d	Generalized OneMax							
100, 50	$4.51 \cdot 10^2$	$4.93 \cdot 10^2$	$5.65 \cdot 10^2$	$5.69 \cdot 10^2$	$6.49 \cdot 10^2$	$6.75 \cdot 10^2$	$6.81 \cdot 10^2$	∞
200, 100	$1.04 \cdot 10^3$	$1.09 \cdot 10^3$	$1.26 \cdot 10^3$	$1.31 \cdot 10^3$	$1.47 \cdot 10^3$	$1.55 \cdot 10^3$	$1.57 \cdot 10^3$	∞
300, 150	$1.72 \cdot 10^3$	$1.74 \cdot 10^3$	$2.03 \cdot 10^3$	$2.05 \cdot 10^3$	$2.40 \cdot 10^3$	$2.51 \cdot 10^3$	$2.51 \cdot 10^3$	∞
400, 200	$2.43 \cdot 10^3$	$2.43 \cdot 10^3$	$2.80 \cdot 10^3$	$2.90 \cdot 10^3$	$3.42 \cdot 10^3$	$3.56 \cdot 10^3$	$3.53 \cdot 10^3$	∞
500, 250	$3.12 \cdot 10^3$	$3.16 \cdot 10^3$	$3.65 \cdot 10^3$	$3.72 \cdot 10^3$	$4.34 \cdot 10^3$	$4.58 \cdot 10^3$	$4.60 \cdot 10^3$	∞

- ss – single state, ts – target state
- ε – exploration parameter
- $Q(s,h) \leftarrow Q(s,h) + \alpha(r + \gamma \max_{h' \in H} Q(s',h') - Q(s,h))$, s – state, H – set of objectives

Conclusion

- Concluding observations on auxiliary objective selection:
  - Conflicting objectives can surprisingly help
  - Multi-objective optimization works good because it preserves the best found solution
  - Therefore, it is enough to use single-objective optimization with the same feature
- Future work:
  - Analyze simultaneous optimization of all objectives on Generalized OneMax (should be $\Theta(n^2 \log(n))$ vs $O(n \log n)$ for dynamic selection)
  - Analyze reinforcement learning with single state (not random in this case!)
  - Empirically test preserving of the best found solution on the Job Shop Scheduling problem

Conclusion

- Concluding observations on auxiliary objective selection:
  - Conflicting objectives can surprisingly help
  - Multi-objective optimization works good because it preserves the best found solution
  - Therefore, it is enough to use single-objective optimization with the same feature
- Future work:
  - Analyze simultaneous optimization of all objectives on Generalized OneMax (should be $\Theta(n^2 \log(n))$ vs $O(n \log n)$ for dynamic selection)
  - Analyze reinforcement learning with single state (not random in this case!)
  - Empirically test preserving of the best found solution on the Job Shop Scheduling problem

Thank you for listening!