



Java framework for Evolutionary Computation and Decision-Making (JECDM)

Michał Tomczyk Miłosz Kadziński

Institute of Computing Science
Poznan University of Technology, Poland

michal.tomczyk@cs.put.poznan.pl
www.cs.put.poznan.pl/mtomczyk/

My research interests include:

- multi-objective optimization (solving real-world problems)
- evolutionary algorithms,
- multi-criteria decision aiding (preference learning)

I'm primarily focused on developing **advanced, interactive metaheuristics for solving real-world optimization problems.**



320

IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 24, NO. 2, APRIL 2020

Decomposition-Based Interactive Evolutionary Algorithm for Multiple Objective Optimization

Michał K. Tomczyk^{*,a} and Miłosz Kadziński^a



Decomposition-based co-evolutionary algorithm for interactive multiple objective optimization

Michał K. Tomczyk^{a,*}, Miłosz Kadziński^a

^aInstitute of Computing Science, Poznań University of Technology, Piotrowo 2, 60-965 Poznań, Poland



Evolutionary algorithms for solving single- and multiple-objective political redistricting problems: The case study of Poland

Michał K. Tomczyk^a, Miłosz Kadziński

^aInstitute of Computing Science, Poznań University of Technology, Piotrowo 2, 60-965 Poznań, Poland

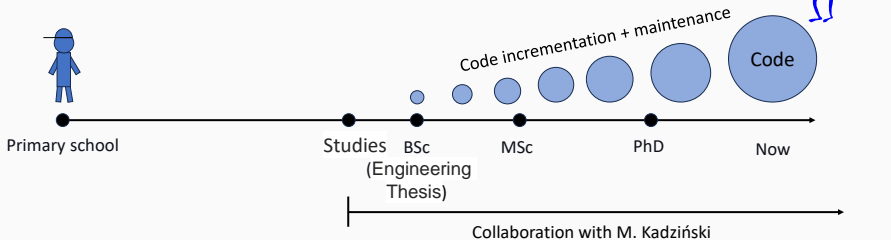


Evaluation of multi-objective optimization approaches for solving green supply chain design problems^a

Miłosz Kadziński^{a,*}, Tommi Tervonen^b, Michał K. Tomczyk^a, Rommert Dekker^c

Research scope + about me

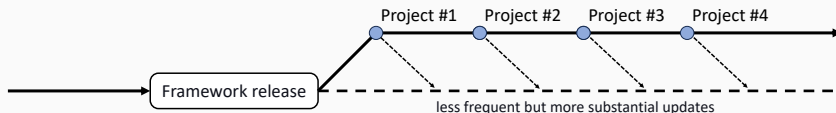
- Programming (data structures/algorithms/AI/etc.) has been one of my hobbies (?). interests since I was a kid. I notice many similarities between LEGO and programming 😊.
- I hate shortcuts. I prefer taking the longer way instead (especially in technical sciences): low effort and low reward vs **high effort and high reward**.
- Code incrementation/maintenance/reusability/high abstraction/etc (modularity).



Motivations

Motivation – a strategic decision:

- doing things simultaneously
- boost for own research
- creating own environment (sandbox) that will be shared (research transparency, reusability, self-promotion, potential collaboration)



- Next focus: other projects
- The future framework's updates will result from the progress made when realizing other projects (**less frequent but more substantial updates**).



About the framework

- The framework is written in Java (v. 1.0 uses Java 17). Motivations for Java are:
 - Need to maintain an extensive (> 1000 classes, > 150000 lines of code), but highly structured project (object-oriented programming, software engineering, etc.)
 - Fast (runtime, not startup)
 - Memory optimization or direct memory addressing is not of concern.
- High-level abstractions and code reusability...
- but still focused on efficiency in those places when the performance is critical.
- It follows good practices delineated by software engineering practitioners but can also be deemed high-quality from the viewpoint of specialists in algorithms and data structures.
- Self-sufficiency (low use of external libraries, currently only 4 small, technical libraries are used).
- The framework is maintained using the IntelliJ IDEA software (shared on GitHub and as copied & pasted sources).
- **Primary focus: architecture, not methods.**

About the framework

Instead of the product on the left, I am focused on developing a product on the right.



- (+) Easy to comprehend
- (-) Low reusability
- (-) Low potential for extensions



- (-) More difficult to comprehend
- (+) High reusability
- (+) High potential for extensions

The framework will be discussed and explained (how to use it) in a series of tutorials (PDF + accompanying source codes).

Thank you Denmark ☺

The pictures were borrowed from the LEGO store: <https://www.lego.com/>

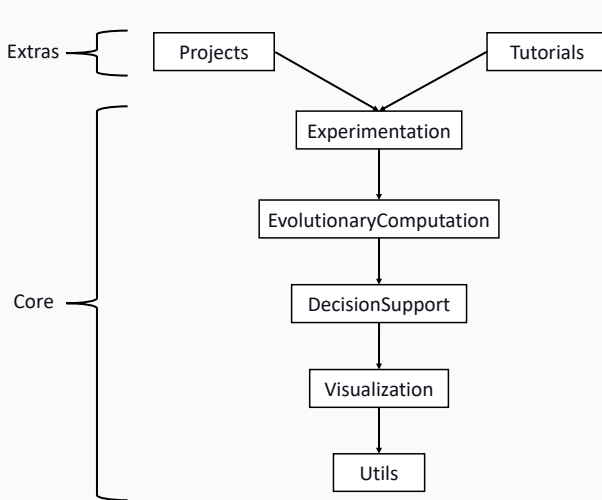
About the framework

Are there other frameworks?



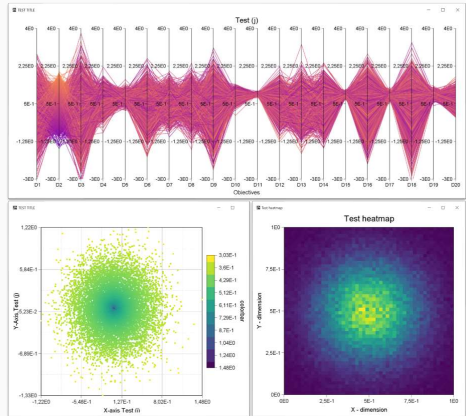
- For MCDA: the existing software is mainly focused on using existing methods (there are some exceptions, e.g., <https://www.decision-deck.org/project/>)
- EMO + MCDA: not at this scale (focus on integrating state-of-the-art from both fields)
- EMO: yes, but they have some flaws (strange architectural decisions, not so flexible for extensions, some hardcoding). For example:
 - Pymoo <https://www.pymoo.org/news.html>
 - MOEA Framework <http://moeaframework.org/>
 - JMetal <https://jmetal.sourceforge.net/>

Main modules

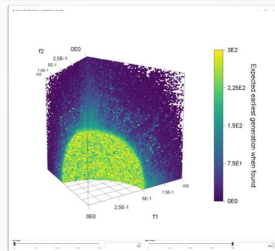
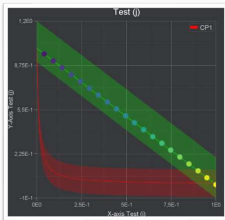
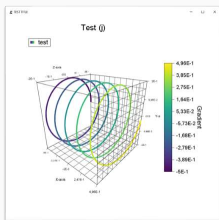
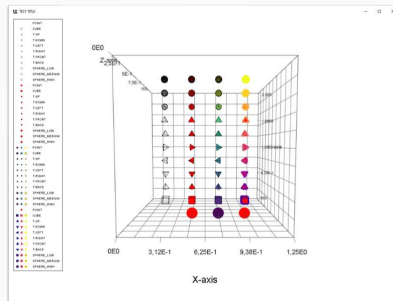
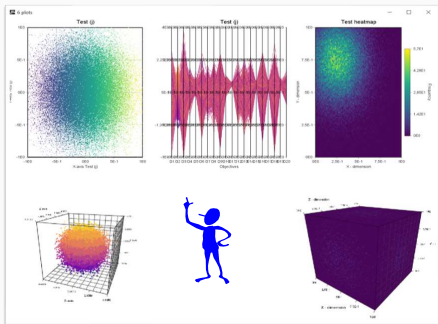


The *visualization* module

- Developed from scratch (almost).
- **High-level integration with other framework's components.**
- Supports 2D (Java Swing; processing on CPU) and 3D rendering (OpenGL; processing on GPU).
- Highly efficient. The main focus was to develop visualization components that can work well with **dynamically updating data**. The GUI processing is separated from more costly operations, which are executed on separate threads in a dedicated queuing system.
- **Highly flexible with a high potential for various adaptations.**

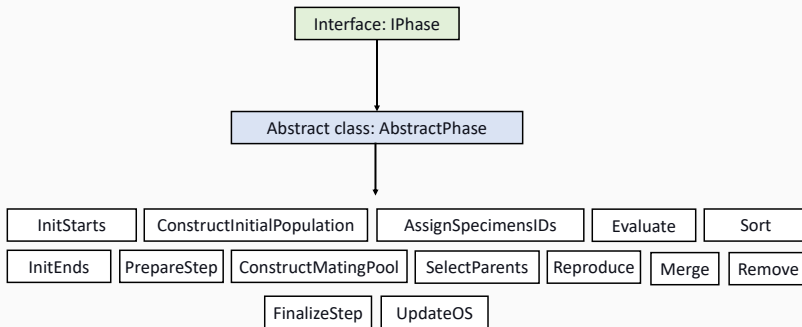


The *visualization* module



The *evolutionary computation* module

The evolutionary computation (algorithm) is built on the concept of phases processed by the upper-level class.



The *evolutionary computation* module

- The phases are managed and executed by the upper-level class (EA)
- The overall design should suit most cases well (regular implementations/algorithms)
- **The default implementations will probably do almost 90% of the job.** The programmer will usually have to overwrite some particular methods and decide upon the exclusion/inclusion of some phases.
- Some phases are somewhat technical, and there will be no reason to re-implement them (e.g., AssignSpecimensIDs); some must be implemented (e.g., solution evaluation).

```
/**
 * Initializes the evolutionary process.
 */
new *
@Override
public void init()
{
    startMeasuringTime();
    _currentTimestamp = new EATimestamp( generation: 0, steadyStateRepeat: 0);

    executePhase(PhasesIDs.PHASE_INIT_STARTS);
    executePhase(PhasesIDs.PHASE_CONSTRUCT_INITIAL_POPULATION);
    executePhase(PhasesIDs.PHASE_ASSIGN_SPECIMENS_IDS);
    executePhase(PhasesIDs.PHASE_EVALUATE);
    executePhase(PhasesIDs.PHASE_UPDATE_OS);
    executePhase(PhasesIDs.PHASE_SORT);
    executePhase(PhasesIDs.PHASE_INIT_ENDS);

    stopMeasuringTime();
}
```

```
/**
 * Executes one step of the evolutionary algorithm.
 */
new *
@Override
public void step(EATimestamp timestamp)
{
    startMeasuringTime();
    _currentTimestamp = timestamp;

    executePhase(PhasesIDs.PHASE_PREPARE_STEP);
    executePhase(PhasesIDs.PHASE_CONSTRUCT_MATING_POOL);
    executePhase(PhasesIDs.PHASE_SELECT_PARENTS);
    executePhase(PhasesIDs.PHASE_REPRODUCE);
    executePhase(PhasesIDs.PHASE_ASSIGN_SPECIMENS_IDS);
    executePhase(PhasesIDs.PHASE_EVALUATE);
    executePhase(PhasesIDs.PHASE_MERGE);
    executePhase(PhasesIDs.PHASE_UPDATE_OS);
    executePhase(PhasesIDs.PHASE_SORT);
    executePhase(PhasesIDs.PHASE_REMOVE);
    executePhase(PhasesIDs.PHASE_FINALIZE_STEP);
    stopMeasuringTime();
}
```

The *evolutionary computation* module

Abstract class: AbstractPhase

```

1  AbstractPhase.java
2
3  /**
4   * Superclass class used for measuring the execution time.
5   */
6   public long measure() {
7
8   }
9
10 /**
11  * Superclass class used for measuring the execution time.
12  */
13 protected int phase0 = 0;
14
15 /**
16  * Initialization constructor.
17  * @param time used for the clock.
18  */
19 public AbstractPhase(long time) {
20     this.time = time;
21 }
22
23 /**
24  * Executes the pre-action, main action, and post action.
25  */
26 public void execute() {
27     // execute pre-action
28 }
29
30 /**
31  * Executes the pre-action, main action, and post action.
32  */
33 public void execute() {
34     // execute pre-action
35 }
36
37 /**
38  * Executes the pre-action, main action, and post action.
39  */
40 public void execute() {
41     // execute pre-action
42 }
43
44 /**
45  * Executes the pre-action, main action, and post action.
46  */
47 public void execute() {
48     // execute pre-action
49 }
50
51 /**
52  * Executes the pre-action, main action, and post action.
53  */
54 public void execute() {
55     // execute pre-action
56 }
57
58 /**
59  * Executes the pre-action, main action, and post action.
60  */
61 public void execute() {
62     // execute pre-action
63 }
64
65 /**
66  * Executes the pre-action, main action, and post action.
67  */
68 public void execute() {
69     // execute pre-action
70 }
71
72 /**
73  * Executes the pre-action, main action, and post action.
74  */
75 public void execute() {
76     // execute pre-action
77 }
78
79 /**
80  * Executes the pre-action, main action, and post action.
81  */
82 public void execute() {
83     // execute pre-action
84 }
85
86 /**
87  * Executes the pre-action, main action, and post action.
88  */
89 public void execute() {
90     // execute pre-action
91 }
92
93 /**
94  * Executes the pre-action, main action, and post action.
95  */
96 public void execute() {
97     // execute pre-action
98 }
99
100 /**
101  * Executes the pre-action, main action, and post action.
102  */
103 public void execute() {
104     // execute pre-action
105 }
106
107 /**
108  * Executes the pre-action, main action, and post action.
109  */
110 public void execute() {
111     // execute pre-action
112 }
113
114 /**
115  * Executes the pre-action, main action, and post action.
116  */
117 public void execute() {
118     // execute pre-action
119 }
120
121 /**
122  * Executes the pre-action, main action, and post action.
123  */
124 public void execute() {
125     // execute pre-action
126 }
127
128 /**
129  * Executes the pre-action, main action, and post action.
130  */
131 public void execute() {
132     // execute pre-action
133 }
134
135 /**
136  * Executes the pre-action, main action, and post action.
137  */
138 public void execute() {
139     // execute pre-action
140 }
141
142 /**
143  * Executes the pre-action, main action, and post action.
144  */
145 public void execute() {
146     // execute pre-action
147 }
148
149 /**
150  * Executes the pre-action, main action, and post action.
151  */
152 public void execute() {
153     // execute pre-action
154 }
155
156 /**
157  * Executes the pre-action, main action, and post action.
158  */
159 public void execute() {
160     // execute pre-action
161 }
162
163 /**
164  * Executes the pre-action, main action, and post action.
165  */
166 public void execute() {
167     // execute pre-action
168 }
169
170 /**
171  * Executes the pre-action, main action, and post action.
172  */
173 public void execute() {
174     // execute pre-action
175 }
176
177 /**
178  * Executes the pre-action, main action, and post action.
179  */
180 public void execute() {
181     // execute pre-action
182 }
183
184 /**
185  * Executes the pre-action, main action, and post action.
186  */
187 public void execute() {
188     // execute pre-action
189 }
190
191 /**
192  * Executes the pre-action, main action, and post action.
193  */
194 public void execute() {
195     // execute pre-action
196 }
197
198 /**
199  * Executes the pre-action, main action, and post action.
200  */
201 public void execute() {
202     // execute pre-action
203 }
204
205 /**
206  * Executes the pre-action, main action, and post action.
207  */
208 public void execute() {
209     // execute pre-action
210 }
211
212 /**
213  * Executes the pre-action, main action, and post action.
214  */
215 public void execute() {
216     // execute pre-action
217 }
218
219 /**
220  * Executes the pre-action, main action, and post action.
221  */
222 public void execute() {
223     // execute pre-action
224 }
225
226 /**
227  * Executes the pre-action, main action, and post action.
228  */
229 public void execute() {
230     // execute pre-action
231 }
232
233 /**
234  * Executes the pre-action, main action, and post action.
235  */
236 public void execute() {
237     // execute pre-action
238 }
239
240 /**
241  * Executes the pre-action, main action, and post action.
242  */
243 public void execute() {
244     // execute pre-action
245 }
246
247 /**
248  * Executes the pre-action, main action, and post action.
249  */
250 public void execute() {
251     // execute pre-action
252 }
253
254 /**
255  * Executes the pre-action, main action, and post action.
256  */
257 public void execute() {
258     // execute pre-action
259 }
260
261 /**
262  * Executes the pre-action, main action, and post action.
263  */
264 public void execute() {
265     // execute pre-action
266 }
267
268 /**
269  * Executes the pre-action, main action, and post action.
270  */
271 public void execute() {
272     // execute pre-action
273 }
274
275 /**
276  * Executes the pre-action, main action, and post action.
277  */
278 public void execute() {
279     // execute pre-action
280 }
281
282 /**
283  * Executes the pre-action, main action, and post action.
284  */
285 public void execute() {
286     // execute pre-action
287 }
288
289 /**
290  * Executes the pre-action, main action, and post action.
291  */
292 public void execute() {
293     // execute pre-action
294 }
295
296 /**
297  * Executes the pre-action, main action, and post action.
298  */
299 public void execute() {
300     // execute pre-action
301 }
302
303 /**
304  * Executes the pre-action, main action, and post action.
305  */
306 public void execute() {
307     // execute pre-action
308 }
309
310 /**
311  * Executes the pre-action, main action, and post action.
312  */
313 public void execute() {
314     // execute pre-action
315 }
316
317 /**
318  * Executes the pre-action, main action, and post action.
319  */
320 public void execute() {
321     // execute pre-action
322 }
323
324 /**
325  * Executes the pre-action, main action, and post action.
326  */
327 public void execute() {
328     // execute pre-action
329 }
330
331 /**
332  * Executes the pre-action, main action, and post action.
333  */
334 public void execute() {
335     // execute pre-action
336 }
337
338 /**
339  * Executes the pre-action, main action, and post action.
340  */
341 public void execute() {
342     // execute pre-action
343 }
344
345 /**
346  * Executes the pre-action, main action, and post action.
347  */
348 public void execute() {
349     // execute pre-action
350 }
351
352 /**
353  * Executes the pre-action, main action, and post action.
354  */
355 public void execute() {
356     // execute pre-action
357 }
358
359 /**
360  * Executes the pre-action, main action, and post action.
361  */
362 public void execute() {
363     // execute pre-action
364 }
365
366 /**
367  * Executes the pre-action, main action, and post action.
368  */
369 public void execute() {
370     // execute pre-action
371 }
372
373 /**
374  * Executes the pre-action, main action, and post action.
375  */
376 public void execute() {
377     // execute pre-action
378 }
379
380 /**
381  * Executes the pre-action, main action, and post action.
382  */
383 public void execute() {
384     // execute pre-action
385 }
386
387 /**
388  * Executes the pre-action, main action, and post action.
389  */
390 public void execute() {
391     // execute pre-action
392 }
393
394 /**
395  * Executes the pre-action, main action, and post action.
396  */
397 public void execute() {
398     // execute pre-action
399 }
400
401 /**
402  * Executes the pre-action, main action, and post action.
403  */
404 public void execute() {
405     // execute pre-action
406 }
407
408 /**
409  * Executes the pre-action, main action, and post action.
410  */
411 public void execute() {
412     // execute pre-action
413 }
414
415 /**
416  * Executes the pre-action, main action, and post action.
417  */
418 public void execute() {
419     // execute pre-action
420 }
421
422 /**
423  * Executes the pre-action, main action, and post action.
424  */
425 public void execute() {
426     // execute pre-action
427 }
428
429 /**
430  * Executes the pre-action, main action, and post action.
431  */
432 public void execute() {
433     // execute pre-action
434 }
435
436 /**
437  * Executes the pre-action, main action, and post action.
438  */
439 public void execute() {
440     // execute pre-action
441 }
442
443 /**
444  * Executes the pre-action, main action, and post action.
445  */
446 public void execute() {
447     // execute pre-action
448 }
449
450 /**
451  * Executes the pre-action, main action, and post action.
452  */
453 public void execute() {
454     // execute pre-action
455 }
456
457 /**
458  * Executes the pre-action, main action, and post action.
459  */
460 public void execute() {
461     // execute pre-action
462 }
463
464 /**
465  * Executes the pre-action, main action, and post action.
466  */
467 public void execute() {
468     // execute pre-action
469 }
470
471 /**
472  * Executes the pre-action, main action, and post action.
473  */
474 public void execute() {
475     // execute pre-action
476 }
477
478 /**
479  * Executes the pre-action, main action, and post action.
480  */
481 public void execute() {
482     // execute pre-action
483 }
484
485 /**
486  * Executes the pre-action, main action, and post action.
487  */
488 public void execute() {
489     // execute pre-action
490 }
491
492 /**
493  * Executes the pre-action, main action, and post action.
494  */
495 public void execute() {
496     // execute pre-action
497 }
498
499 /**
500  * Executes the pre-action, main action, and post action.
501  */
502 public void execute() {
503     // execute pre-action
504 }
505
506 /**
507  * Executes the pre-action, main action, and post action.
508  */
509 public void execute() {
510     // execute pre-action
511 }
512
513 /**
514  * Executes the pre-action, main action, and post action.
515  */
516 public void execute() {
517     // execute pre-action
518 }
519
520 /**
521  * Executes the pre-action, main action, and post action.
522  */
523 public void execute() {
524     // execute pre-action
525 }
52
```

Class: Evaluate

```
/**
 * Phase main action.
 *
 * @param ea evolutionary algorithm
 */
new *
@Override
public void action(IEA ea)
{
    if (ea.getPopulation().isPopulationRequiringEvaluation())
        _evaluate.evaluateSpecimens(ea.getPopulation().getPopulation());
    if (ea.getPopulation().isOffspringRequiringEvaluation())
        _evaluate.evaluateSpecimens(ea.getPopulation().getOffspring());
}
```

Interface: IEvaluate

```
/**
 * Interface for classes responsible for evaluating specimens.
 *
 * @author MTowczyk
 */
10 Implementations new *
public interface IEvaluate

/**
 * Evaluates specimens.
 *
 * @param specimens array of specimens to be evaluated
 */
2 usages 1 implementation new *
void evaluateSpecimens(ArrayList<Specimen> specimens);
```

The *decision support* module

- The decision support module is more like an addition to the evolutionary computation module than an equivalent component.
- The decision support module's strategic focus is to **facilitate the integration of both areas rather than attempting to implement all existing MCDA methods**. It will also focus mainly on preference-learning methods and value models.
- The module is implemented in the same spirit as the module for evolutionary Computation, and is oriented around **a top-level decision-support component that can handle various aspects of the decision-reaching process**, e.g., collecting preferences, requesting preferences, exploiting preference models, handling inconsistencies, evaluating solutions given the Decision Maker's preferences, and so on.

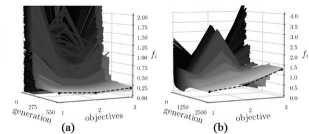


Fig. 5. Solutions constructed throughout evolutionary search by IEMO/D applied to DTLZ1 and WFG1 with $M = 3$ objectives and $u_{DM}^{DM} = [0.3, 0.4, 0.4]$. (a) DTLZ1. (b) WFG1.

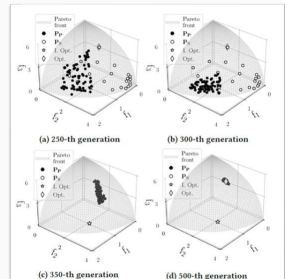


Figure 4: Populations constructed by IEMO/DCD in different generations when applied to WFG4 with $M = 3$ and $DM = SC$.



Note that implementing a computational sandbox-like framework is not an easy task due to three reasons:

- Human dependency raises ambiguity when it comes to implementation.
- Many of the MCDA components are incompatible, which raises issues regarding generalization.
- Concepts \leftrightarrow methods: MCDA content is shifted more towards methods than concepts.

The *experimentation* module

- The module is implemented in the same spirit as the module for evolutionary computation.
- Template-like.
- Highly structured with default way of processing (3 levels, automation) and data organization provided.
- Efficiency, especially memory efficiency, is of high priority.



Defining 1-level read-only basic data

E.g., no. trial runs, data for optimization problems,
or the main folder path

Executing 1-level operations

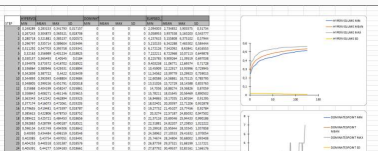
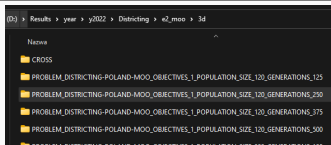
E.g., creating top-level folders

Defining experimental scenarios

- A. Problems:** DTLZ1-7, WFG1-9, .;
- B. Algorithms,** NSGA-III, NEMO-II, IEMO/D, ...
- C. Objectives:** 2-5;
- D. Interactions:** 5-10.
- E. ...**

The experimentation module

- Supports providing new performance indicators (implementing an interface)
- Supports parallelization of the 3-level processing.
- Supports providing new file processors (store results on disc)
- Supports providing new finalizers (construct experiment summary).



For each tuple that defines a scenario; (A, B, C, D, E, ...)

Defining 2-level (scenario-dependent) read-only data

E.g., performance indicators, the numer of generations, the population size, read-only operators, etc.

For each trial run (e.g., 100, 3 level):

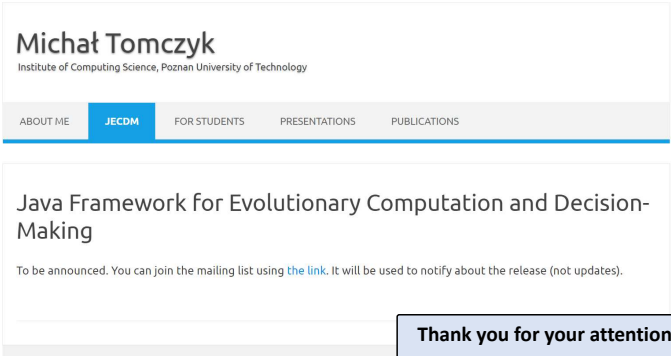
1. Instantiate the algorithm
2. Run the algorithm and store the intermediate and final results

Finalize 3-level

Finalize 2-level

Finalize 1-level

The framework is currently under development, but I plan to release it in around 3 months. See my website: <https://www.cs.put.poznan.pl/mtomczyk/index.php/jecdm>, where you can join a mailing list that will be used to send the notification about the release.



The screenshot shows the homepage of Michał Tomczyk's website. The header includes the name "Michał Tomczyk" and the affiliation "Institute of Computing Science, Poznan University of Technology". A navigation bar contains links for "ABOUT ME", "JECDM" (highlighted in blue), "FOR STUDENTS", "PRESENTATIONS", and "PUBLICATIONS". The main content area features the title "Java Framework for Evolutionary Computation and Decision-Making" and a message: "To be announced. You can join the mailing list using [the link](#). It will be used to notify about the release (not updates)." A blue stick figure icon is positioned to the right of the navigation bar. A blue box at the bottom right of the screenshot contains the text "Thank you for your attention!" with a smiley face icon and the email address "michal.tomczyk@cs.put.poznan.pl".

Michał Tomczyk
Institute of Computing Science, Poznan University of Technology

ABOUT ME JECDM FOR STUDENTS PRESENTATIONS PUBLICATIONS

Java Framework for Evolutionary Computation and Decision-Making

To be announced. You can join the mailing list using [the link](#). It will be used to notify about the release (not updates).

Thank you for your attention! 😊
michal.tomczyk@cs.put.poznan.pl