KU LEUVEN kulak

Towards Automatic Design of Adaptive Evolutionary Algorithms

Ayman Srour - Patrick De Causmaecker

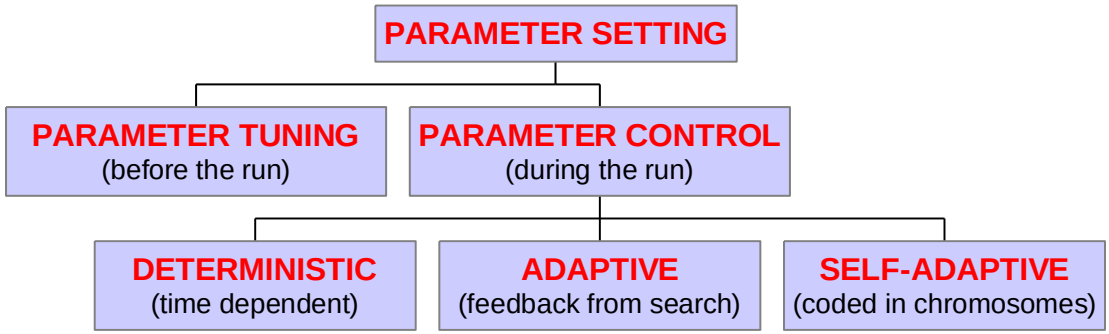
Outline

- Background
  - Parameter tuning
  - Adaptive Parameter control
  - Preliminary investigation
- The proposed framework
  - GE optimizer
  - Adaptive EA
- conclusion & future work

Background-Parameter Tuning

- Evolutionary Algorithms (EAs) have shown a notable success when they used for solving several optimization problems.
- The success of EA's is due to several adjustable parameters such as:
  - Population size & selection mechanism
  - crossover and mutation operator & crossover and mutation rate
- Finding good parameter values help to improve the performance

How can find good parameter values?



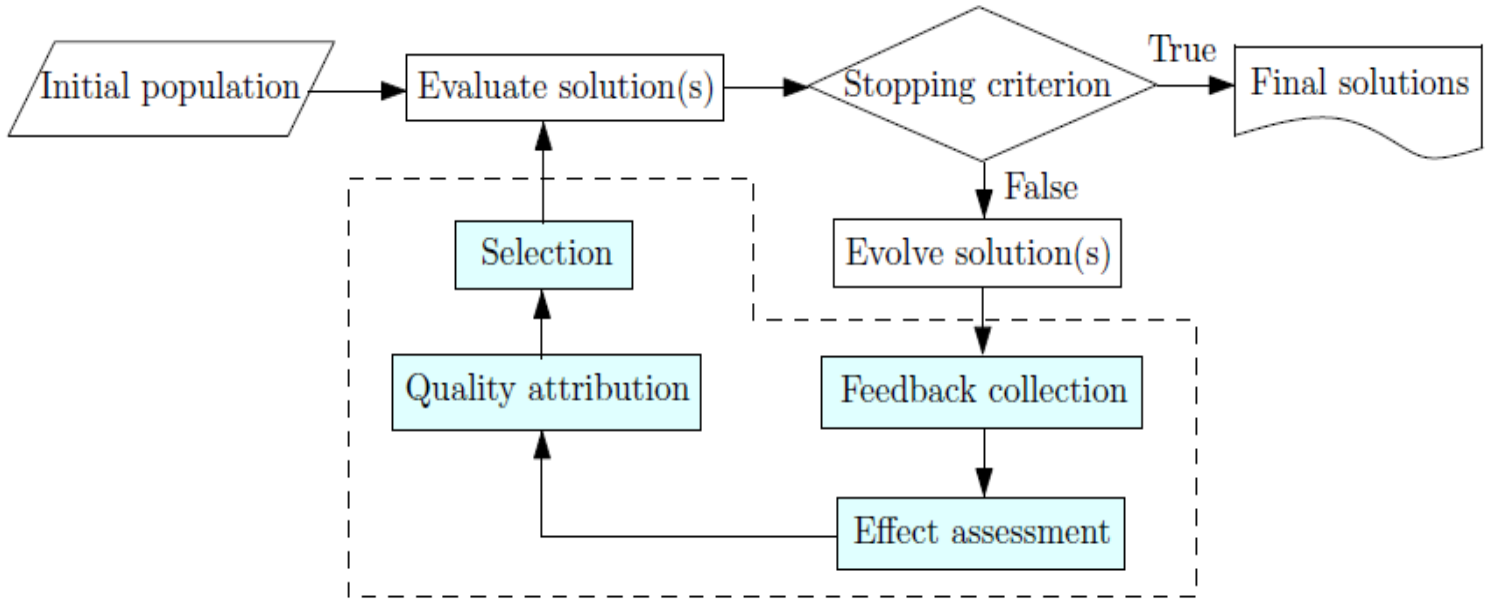
Background- Adaptive Parameter Control (APC)

- A Vast number of parameter control methods have been recorded in literature.

- Considering adaptive parameter control (APC) alone, the number of published papers reach up to 150 till 2015.
  - most of them are problem specific design
  - Can we find an APC method for EA that can work for all problems?
  - Which parameter should we control?
- Optimizing the APC strategies, in some way, can help to improve the performance.

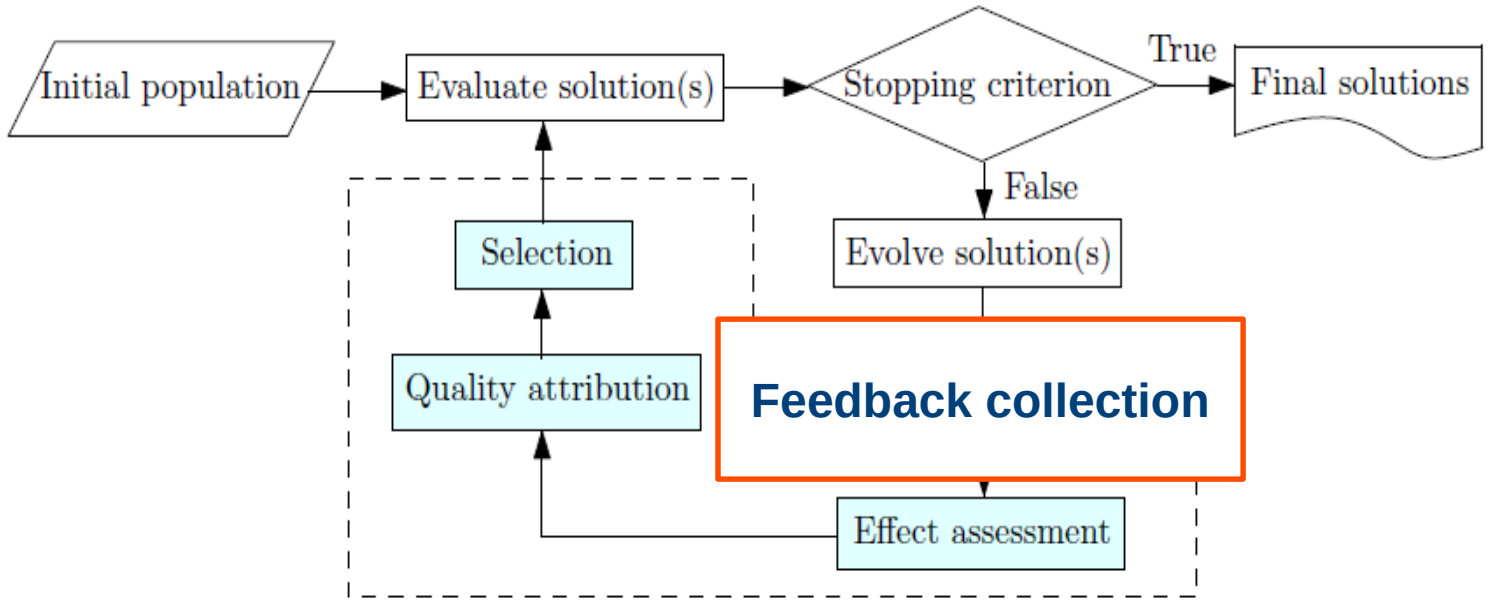
- Therefore, we propose to use grammatical evolution to optimize the design of adaptive EAs

Adaptive EA Design Model



- **Proposed by Aleti & Moser 2016.**
- **The model decompose the APC into several components**
- **These components can help to automate the design of Adaptive EA**

Adaptive EA Design Model

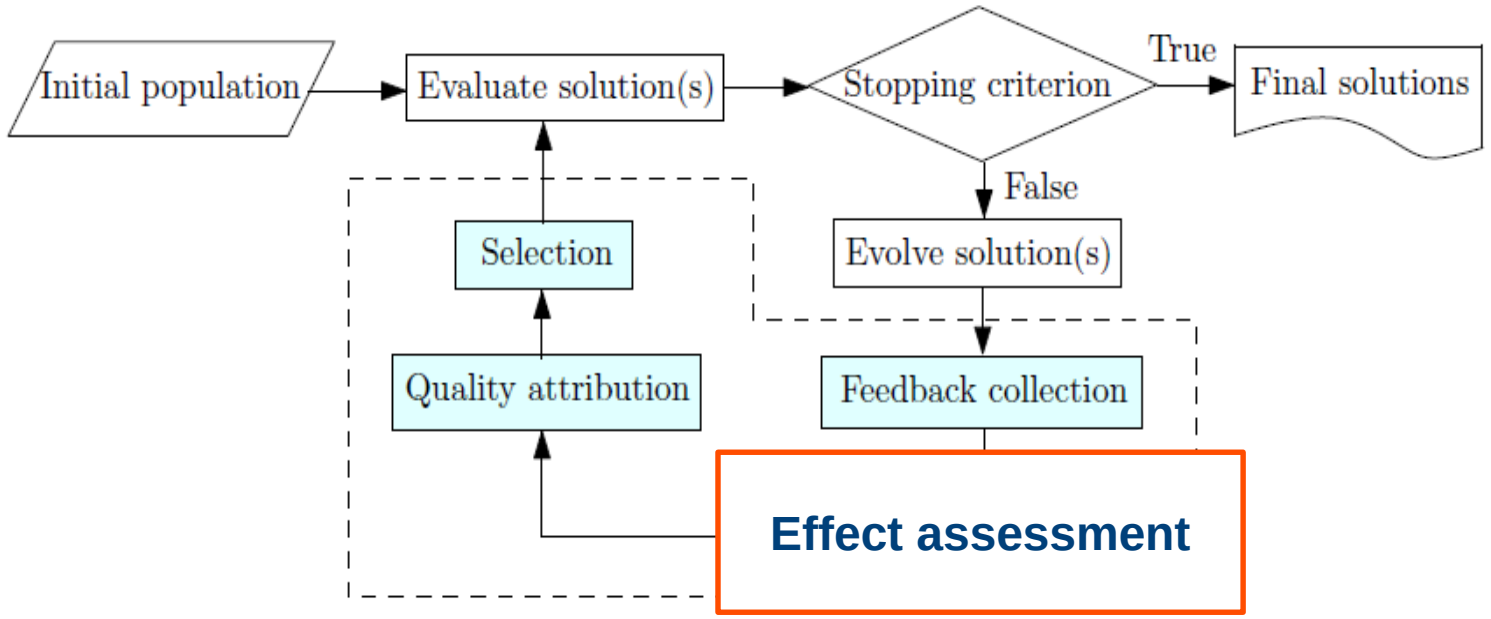


- **is a measurement of certain properties of an EA**
- **An indication of the algorithm's performance.**

Strategies

- Phenotype
- Phenotype diversity
- Genotype
- Genotype diversity
- Feasibility
- Computations

Adaptive EA Design Model



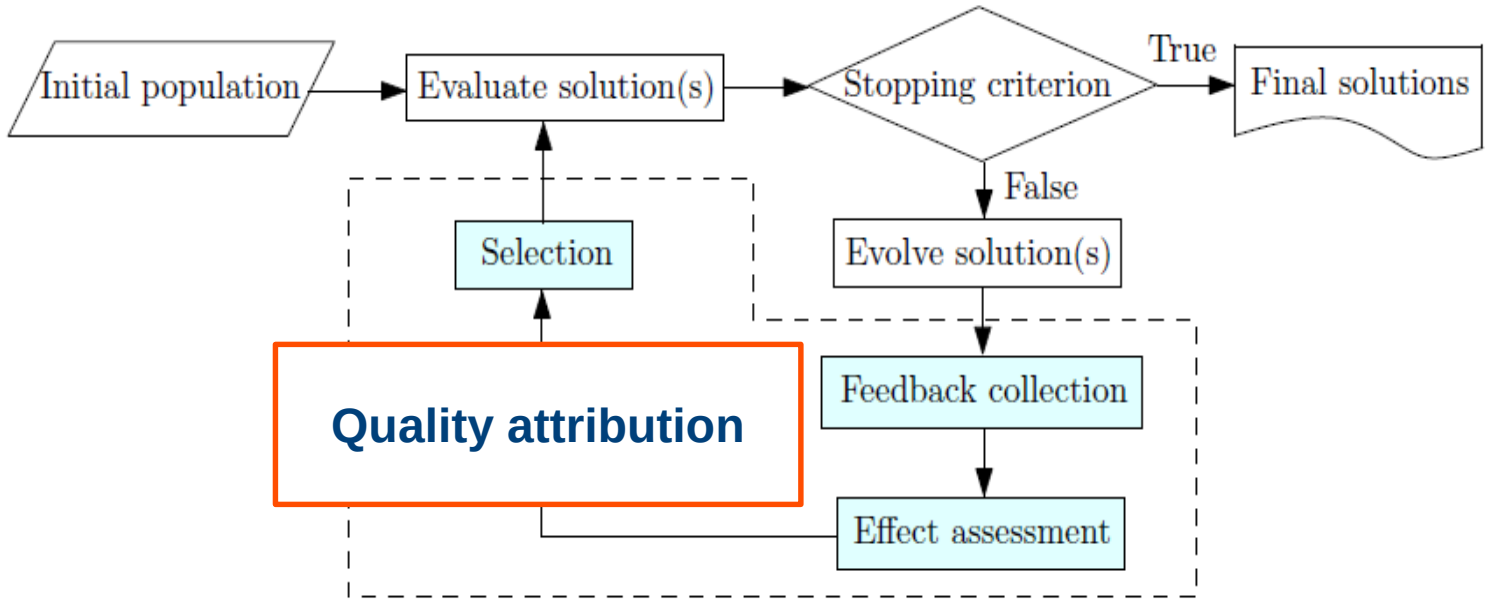
To measure the effect of each parameter value on the performance of the algorithm

Strategies

- Ancestor effect
- Population effect
- Best effect
- Worst effect
- Median effect
- Current effect

KU LEUVEN kulak

Adaptive EA Design Model

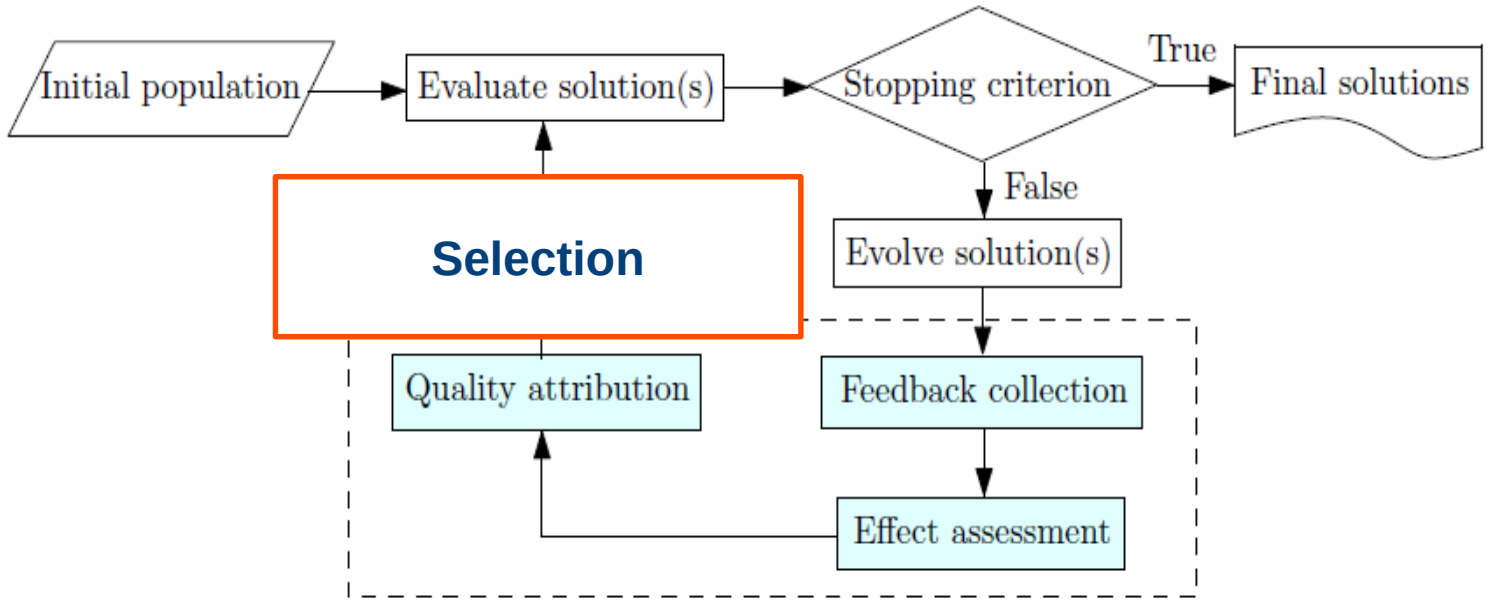


calculated based on predefined rules that use the effect measured in the previous iterations.

Strategies

- Immediate
- Average
- Extreme
- Learned

Adaptive EA Design Model



is the selection of parameter values to use in the next iteration.

Strategies

- Quality proportionate
- Quality proportionate with minimum probability
- Greedy
- Deterministic

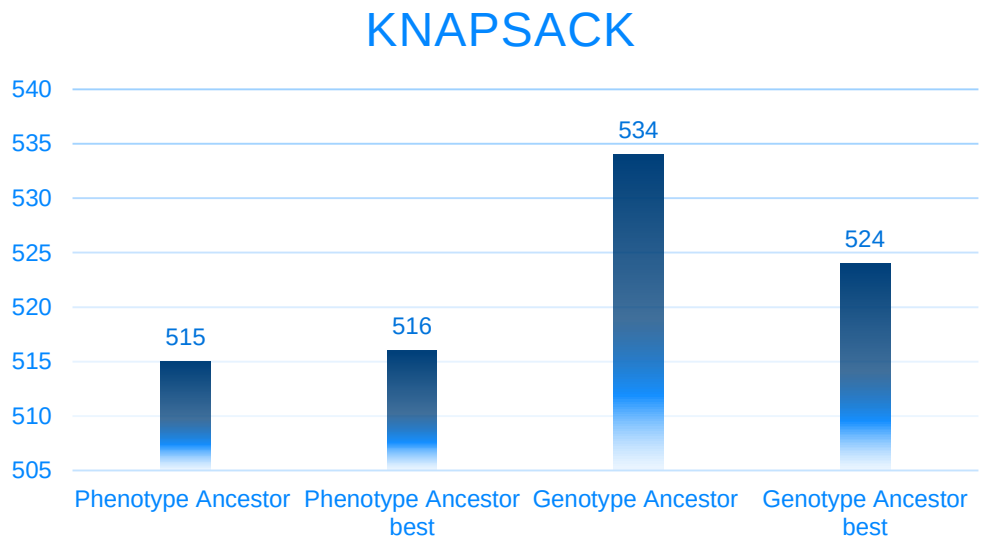
Some Adaptive EAs (categorized based on APC model)

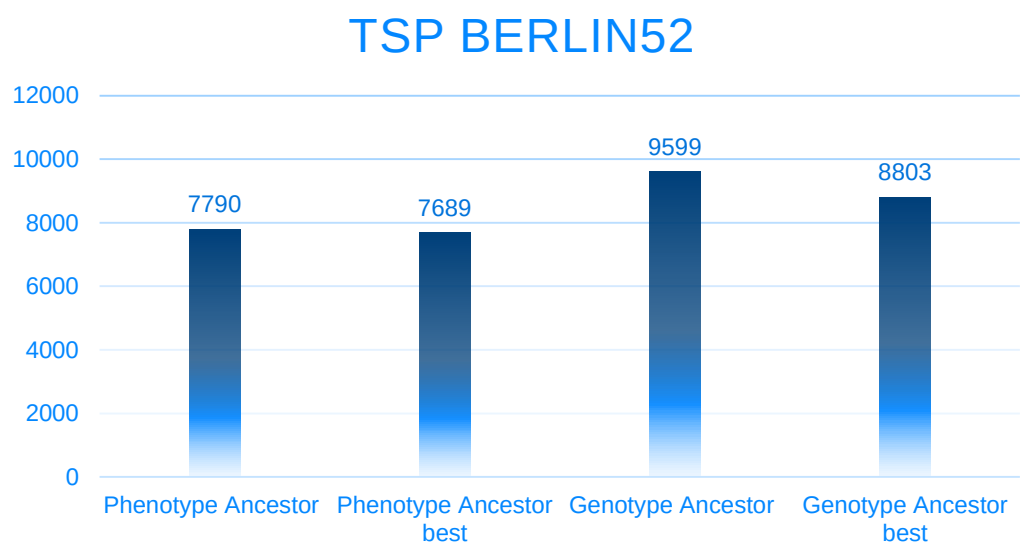
Reference	EA Parameter	APC			
		Feedback collection	Effect assessment	Quality attribution	Selection
Hong et al. 2002	□ Crossover operator □ Mutation operator	□ Phenotype	□ Ancestor □ Current	□ average	□ Quality proportionate
Barkaoui and Gendreau 2013	□ Crossover operator □ Mutation operator □ Selection	□ Phenotype	□ Current	□ Immediate	□ Quality proportionate
Arnone et al. 1994	□ Mutation rate □ Representation □	□ Phenotype □ Genotype diversity □	□ Best □ Current	□ Immediate	□ Greedy
Eiben et al. 2007	□ Population size □ Selection □ Crossover rate □ Mutation rate	□ Phenotype	□ Population □ Current	□ Learned	□ Greedy
Smorodkina and Tauritz 2007	□ Population size	□ Computations	□ Population □ Best □ Current	□ Average	□ Greedy

KU LEUVEN kulak

Investigating the performance of different APC strategies

Feedback collection	Effect assessment	Quality attribution	Selection
• Phenotype • Genotype	• Ancestor • Ancestor + Best	• Learned	• Quality proportionate with minimum probability

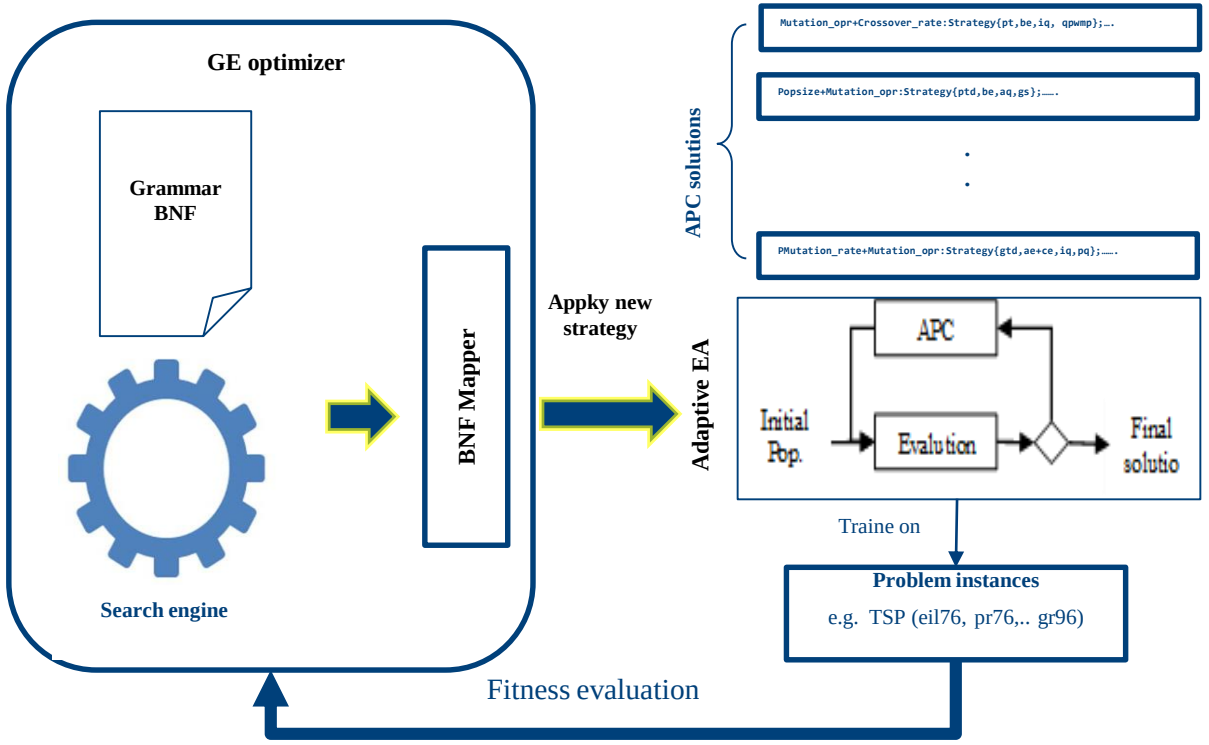




The APC strategies applied to crossover and mutation operators

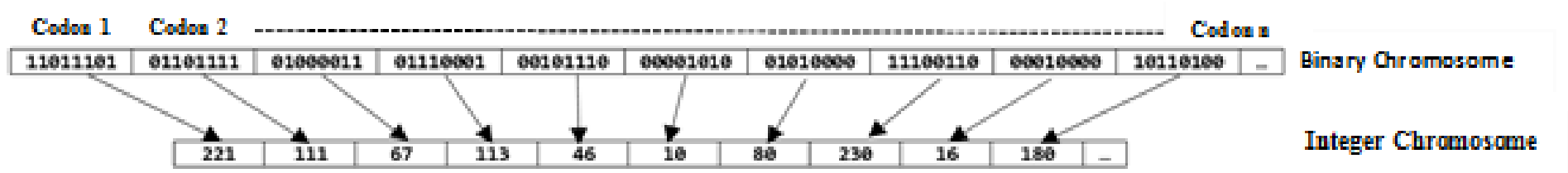
The proposed framework

- consists of two main components:
  - GE optimizer
    - Adopts the standard architecture of GE.
    - Used to evolve individuals that encode effective APC strategies for the adaptive EA.
  - Adaptive EA
    - represents the implantation of many possible adapted EA architecture based on the newly generated APC strategies (GE individuals)
    - Necessary to evaluate GE generated solutions.



GE optimizer - Search Engine

- Search Engine
  - Uses a stander genetic algorithm
  - The genotype is represented by variable length binary string
  - The codon is an 8-bit representing an integer value range (0 – 2^8-1)
  - the integer values then can be used to convert all terminals to non-terminals symbols via a mapper function.



GE optimizer- BNF Grammar

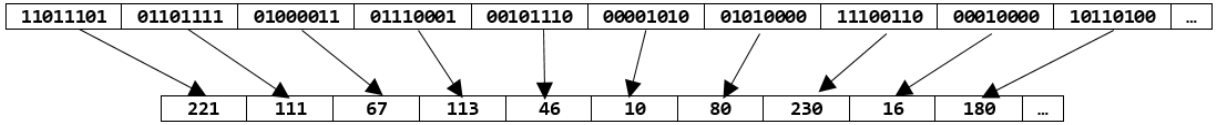
- BNF Grammar
  - The BNF grammar is defined considering the four adaptive parameter control components and their strategies.
    - feedback collection (FC)
    - effect assessment (EFA)
    - quality attribution (QA)
    - selection (SE)
  - The BNF grammar can be formulated as a tuple <T, N, S, P>,
    - T (terminal set)→ The high-level APC strategies i.e. {<FC>,<EFA>,<QA>,<SE>…etc.}
    - N (non-terminals set) → a set of rules and set of EA parameters i.e. {pt, qa,…., Pop_Size, Mutation_rate , ..}
    - S (starting symbol) → <APC>
    - P (production rule) → maps the elements from N to T

GE optimizer- BNF Grammar example

```
Rule No                                                          No. of choices
(1)            <APC>::=<Param><Strategy>;|<Param><Strategy>;<APC>          2
(2)      <Strategy>::=(<FC>,<EFA>,<QA>,<SE>)|(defult)                     2
(3)            <FC>::=<fc>|<fc>+<FC>                                     2
(4)            <fc>::=<pt>|<ptd>|<gt>|<gtd>|<fe>|<co>                    6
(5)          <EFA>::=<efa>|<efa>+<EFA>                                   2
(6)          <efa>::=<ae>|<pe>|<be>|<we>|<me>|<ce>                      6
(7)           <QA>::=<qa>|<qa>+<QA>                                      2
(8)            <qa>::=<iq>|<aq>|<eq>|<lq>                               4
(9)           <SE>::=<se>|<se>+<SE>                                      2
(10)          <se>::=<qp>|<qpwmp>|<gs>|<ds>                              4
(11)       <param>::=<Select_param>:|<Select_param>+<param>               2
(12) <Select_param>::=Pop_Size|Crossover_opr                             6
                    |Crossover_rate|Mutation_opr
                       |Mutation_rate|Selection
.                                    .
.                                     .
.                                      .
(n)                                    .
```

GE optimizer - Mapper function

- Mapper function
  - converts the genotype to phenotype
    - ***Rule= (codon integer value) MOD (the number of rules for the current non-terminal)***



Derivation sequence	Mapping rule
<APC>	start
<Parm><Strategy>;<APC>	221 % 2=1
<Select_param>+<param><Strategy>;<APC>	111 % 2=1
Crossover_opr+<param><Strategy>;<APC>	67 % 6 =1
Crossover_opr +<Select_param>+<param><Strategy>;<APC>	113 % 2=1
Crossover_opr+Mutation_opr+<param><Strategy>;<APC>	45 % 6 =3
Crossover_opr+Mutation_opr+<param><Strategy>;<APC>	10 % 2 =0
Crossover_opr + Mutation_opr+Crossover_rate<Strategy>;<APC>	80 % 6 =2
Crossover_opr+Mutation_opr+Crossover_rate:Strategy{<FC>,<EFA>,<QA>,<SE>};<APC>	230 % 1=0
Crossover_opr+Mutation_opr+Crossover_rate:Strategy{<select_fc>,<EFA>,<QA>,<SE>};<APC>	16 % 2 =0
Crossover_opr+Mutation_opr+Crossover_rate:Strategy{pt,<EFA>,<QA>,<SE>};<APC>	180 % 6=0
.	.
.	.
.	.

Crossover_opr + Mutation_opr + Crossover_rate: Strategy {pt,be,iq, qpwmp};

Mutation_rate: Strategy {gtd,ae+current,iq, qp};

Conclusion & future work

- In this work, a framework for the automatic design of EA's is presented
- we utilize the APC model for automating the design of the adaptive EA
- The grammatical evolution is adopted to search the adaptive EA design space

- In the future work:
  - An implementation of the bespoke APC methods will be performed.
  - The learning process will be performed using different problems to test the generality of the framework.
  - The learning process will produce a set of evolved adaptive EA designs for each problem which will be used later to validate the framework.

Thank you …
Q&A

KU LEUVEN kulak

References

- A. Aleti and I. Moser, "A Systematic Literature Review of Adaptive Parameter Control Methods for Evolutionary Algorithms," ACM Computing Surveys (CSUR), vol. 49, p. 56, 2016.
- A. Eiben, M. Horvath, W. Kowalczyk, and M. C. Schut, "Reinforcement learning for online control of evolutionary algorithms," in International Workshop on Engineering Self-Organising Applications, 2006, pp. 151-160.
- E. Smorodkina and D. Tauritz, "Greedy population sizing for evolutionary algorithms," in Evolutionary Computation, 2007. CEC 2007. IEEE Congress on, 2007, pp. 2181-2187.
- M. Barkaoui and M. Gendreau, "An adaptive evolutionary approach for real-time vehicle routing and dispatching," Computers & Operations Research, vol. 40, pp. 1766-1776, 2013.
- S. Arnone, M. Dell'Orto, and A. Tettamanzi, "Toward a fuzzy government of genetic populations," in Tools with Artificial Intelligence, 1994. Proceedings., Sixth International Conference on, 1994, pp. 585-591.
- T.-P. Hong, H.-S. Wang, W.-Y. Lin, and W.-Y. Lee, "Evolution of appropriate crossover and mutation operators in a genetic process," Applied Intelligence, vol. 16, pp. 7-17, 2002.