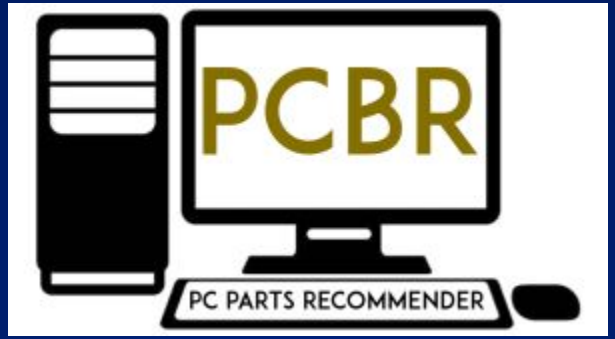


PC Parts Recommender

18 June 2021

Group 4

Víctor Badenas Crespo
Michael Birkholz
Andrea Masi
Kevin David Rosales Santana

Table of Contents

Domain

Simple premise:

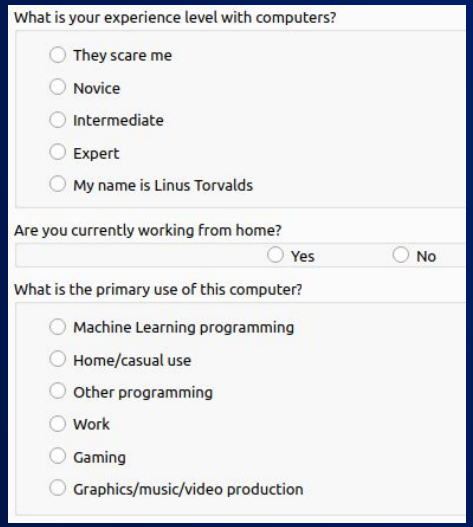
- User fills out survey, indicating their preferences
- They get a recommendation

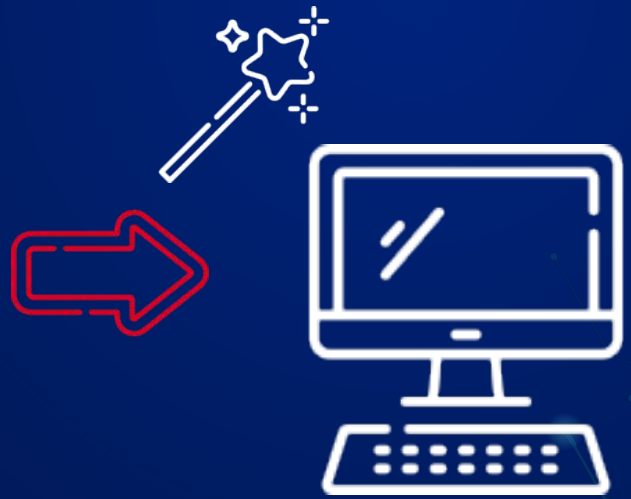
Problem space:

- At least 92,160 options
- However, essentially continuous because a transform is applied based on the answer to 13 preference questions

Solution space:

- At least 278,400 options
- Many of the selections are artificially limited to small sets; for example, only 5 choices of RAM (by size)





Requirements

Functional Requirements

- Way to input user profile
- Way to input user preferences
- Basic/Advanced options
- Results must be human-readable
- Expert must be able to customize the recommendation

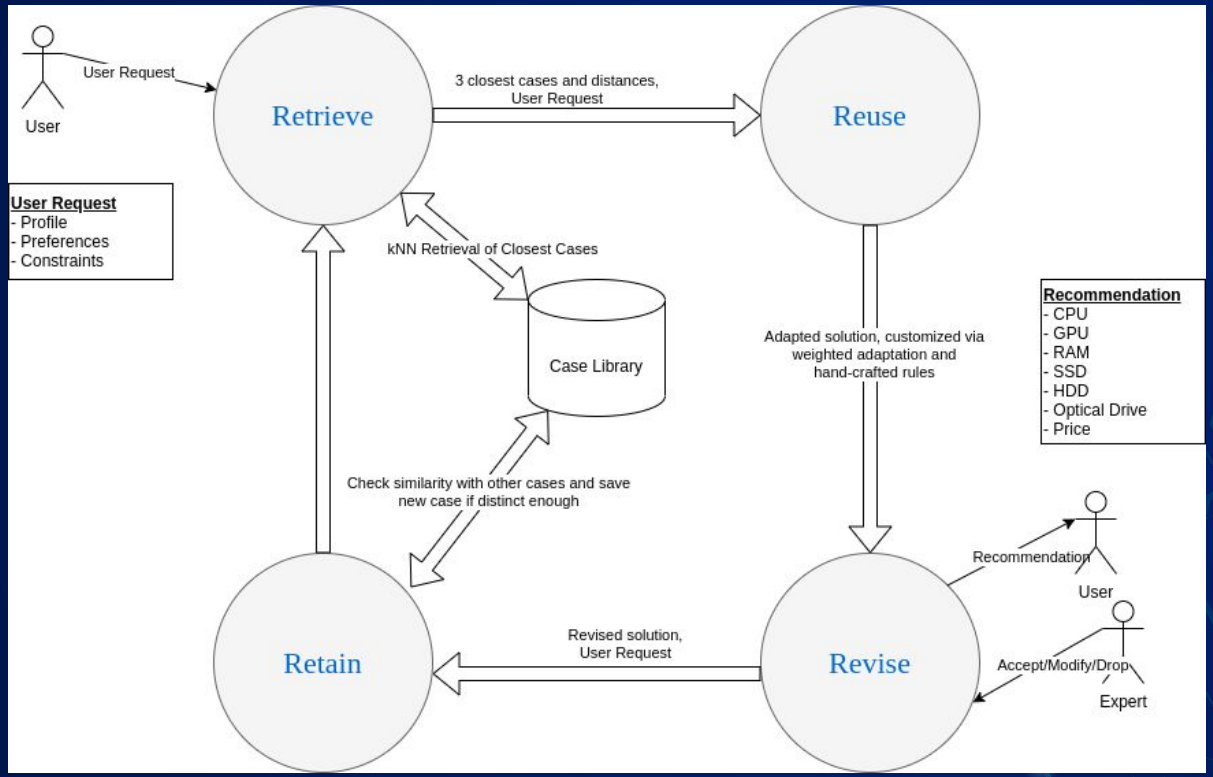
Traceability

- Test plan developed based on requirements
- RVTM integrated in test plan to verify requirement coverage

Technical Requirements

- Only successful cases will be stored
- Retrieval system based on past input and current preferences
- Solution must be able to be customized automatically rather than simply be a retrieved case
- There must be a method to constrain solutions based on both user constraints and domain rules
- Solution must be produced within 5 seconds of a request
- No special computational setup required (just standard laptop/desktop)
- There must be a mechanism to control growth rate of case library

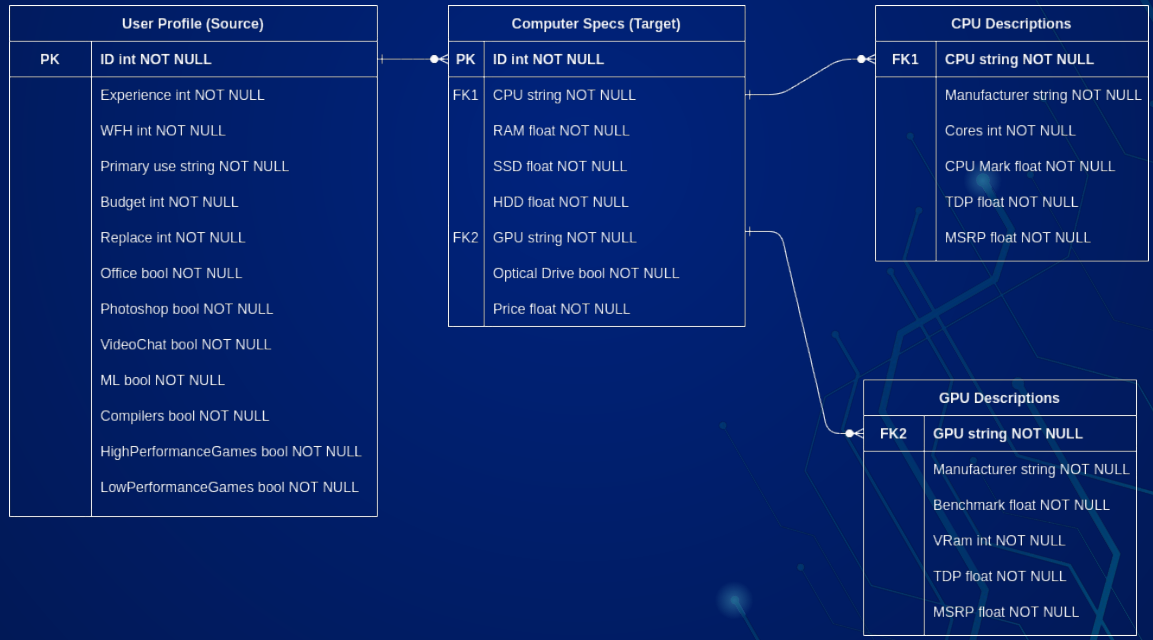
Functional Architecture of the CBR System



- **Composed of 4 (5) R's:**
  - **R**epresentation
  - **R**etrieve
  - **R**euse
  - **R**evise
  - **R**etain
- **2 main agents:**
  - User
  - Expert
- **Storage and learning of cases using the CBL:**
  - Core of the CBR System

Representation

- Initial case base of 28 cases (50k runs -> ~375 cases retained).
- Flat hierarchy is most suitable (Bad for big datasets, always optimal solution)
- <Feature, Value> pairs:



Representation

Feature	Raw	Pre-Processed
Experience	int {1,2,3,4,5}	float 0-1
WFH	bool	int {0,1}
Primary use	String {Home, Work, Production, Programming, ML, Gaming}	float 0-1
Budget	int {1, 2, 3}	float 0-1
Replace	int {1, 2, 3, 4}	float 0-1
Applications	bool	int 0-1
Benchmarks (CPU, GPU)	float	float 0-1
Capacity (RAM, SSD, HDD)	float	log scale float 0-1
Optical Drive	bool	int 0-1
Price	float	float 0-1

Retrieval Task: kNN

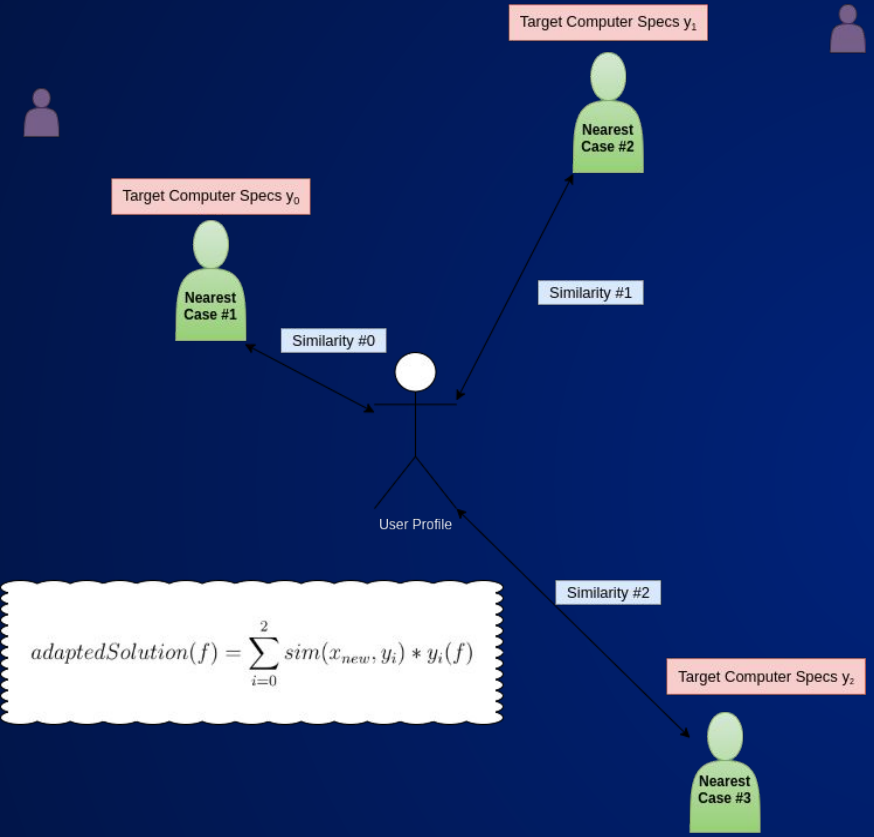
- Supervised K-Nearest Neighbors
- Source and target arrays are loaded into memory
- K closest instances to the predict instance are chosen
- Minkowski metric (float but discrete values)
- K instances and their distances are returned.

$$d(x,y) = \frac{\sum_{k=1}^{K} \omega_k * |x_k - y_k|}{\sum_{k=1}^{K} \omega_k}$$

- The k-Nearest Neighbors accepts features weights
- The weights are extracted from the user preferences
- Each user preference contributes to the weight of each feature with a normalized weight.
- H matrix contains the contributions
- M is the number of preferences and N the number of features

$$\omega = p \cdot H$$

$$H = \begin{pmatrix} h_{00} & h_{10} & \cdots & h_{N0} \\ h_{01} & h_{11} & \cdots & h_{N1} \\ \vdots & \vdots & \ddots & \vdots \\ h_{0M} & h_{1M} & \cdots & h_{NM} \end{pmatrix} ; h_{ij} \in [-1,1]; \sum_{j=0}^{M} h_{ij} = 1$$



$$adaptedSolution(f) = \sum_{i=0}^{2} sim(x_{new}, y_i) * y_i(f)$$

Reuse: Weighted Adaptation

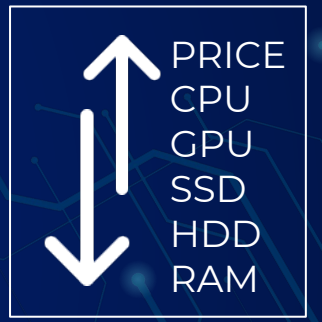
- **First step** of the Reuse stage
- Creation of **new PC Configurations**
  - *Synthetic Task*
- Utilization of **3 Nearest Neighbors distances**
- Use of **similarity** from their distance:

$$sim(x_{new}, y) = \frac{1}{d(x_{new}, y) + 0.1}$$

- **Normalization of similarities**
  - If the input profile matches exactly one case from the CBL, sim(x, y) ≈ 1
- **The adapted solution is the weighted average of the different target attributes taking the similarity as the weights**
- **User profile** and **preferences** have been taken into account…
  - But what about the **constraints?**

Reuse: Background Knowledge, Constraints

- After obtaining adapted solution, still need to apply user constraints and constraints based on domain-knowledge, all the while trying to preserve preferences
- Example of a user constraint: I **must have** an AMD processor



- Example of a domain constraint: AMD processors do not have integrated graphics capabilities and therefore **must be** accompanied by a discrete GPU

Feedback Task

Reuse

Adapted solution

Expert opinion!

Recommendation:
- CPU
- GPU
- SSD
- RAM
- HDD
- Optical Drive

Accept/Modify/Drop

Revise

Revised solution

User answer (with/out recommendation)

User

User request

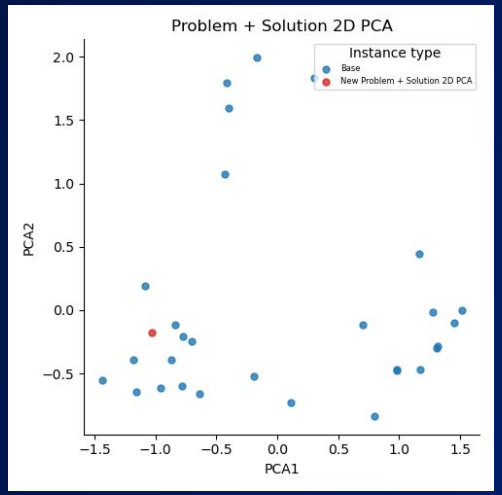
1. All User inputs ⇨ *concat* ⇨ All solutions
2. Fed to the KNN ⇨ extract all 1st NN distances of the dataset
3. Calculate statistics ⇨ get the threshold
4. Revised solution, User request ⇨ *concat* = P+S
5. KNN predict the 1st NN of P+S ⇨ d(P+S)

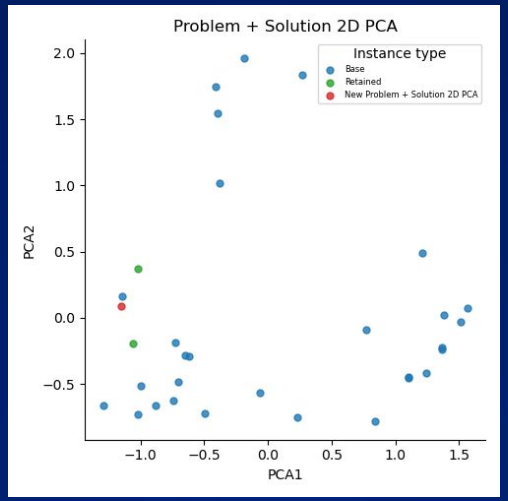
Retain

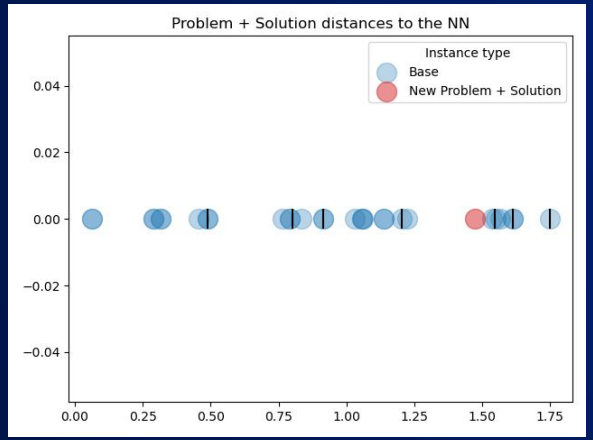
d(P + S) ≥ d(40%)

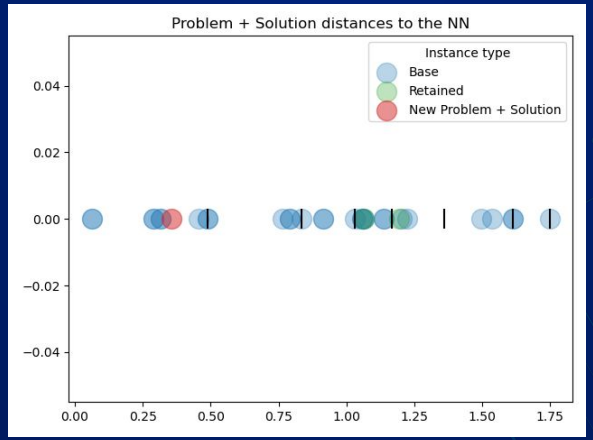
Y → Case library

N → 









Feedback Task

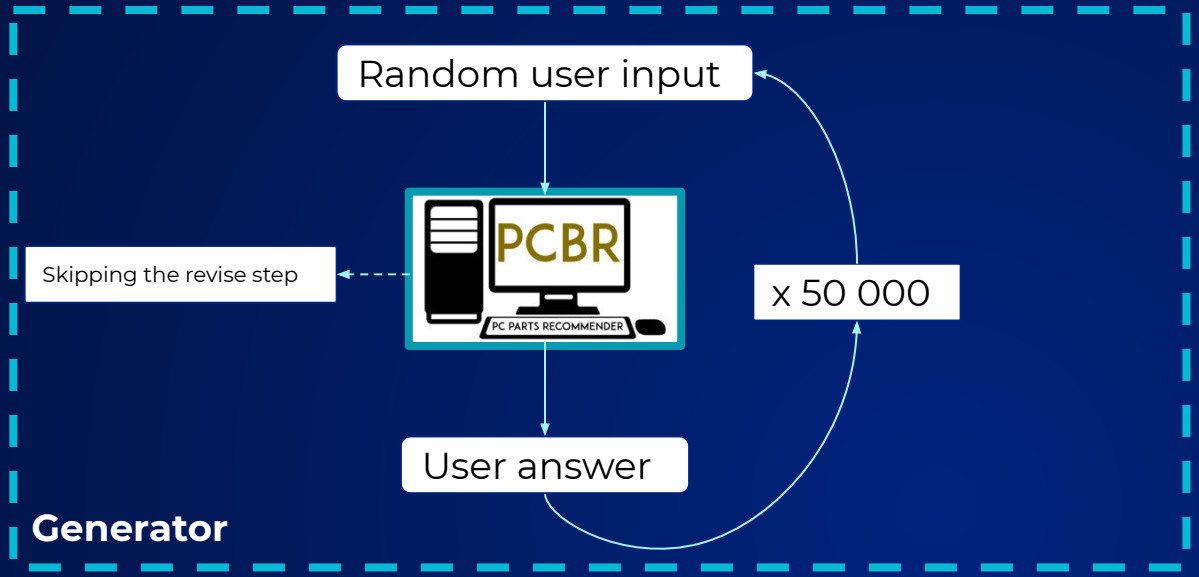
Problem + Solution plots:

- 2D PCA for visualization purposes.

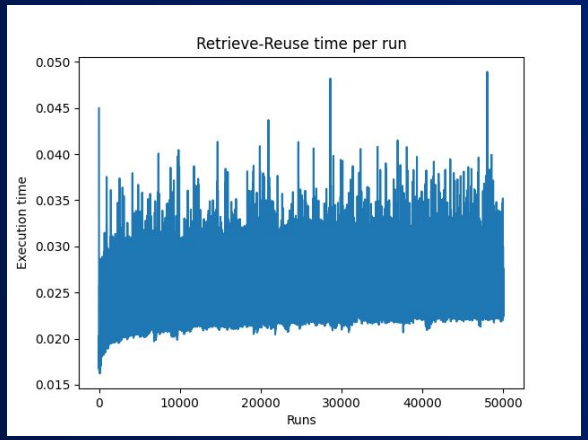
- Distances to nn of:
  - Base cases (Blue)
  - Retained cases (Green)
  - Current case (Red)

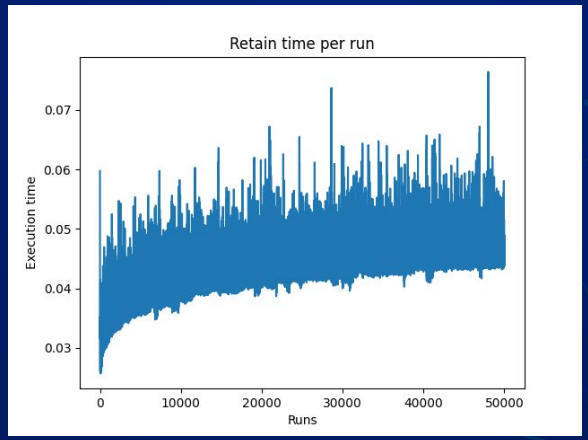
- With percentiles blocks;
- Used to calculate the final threshold.

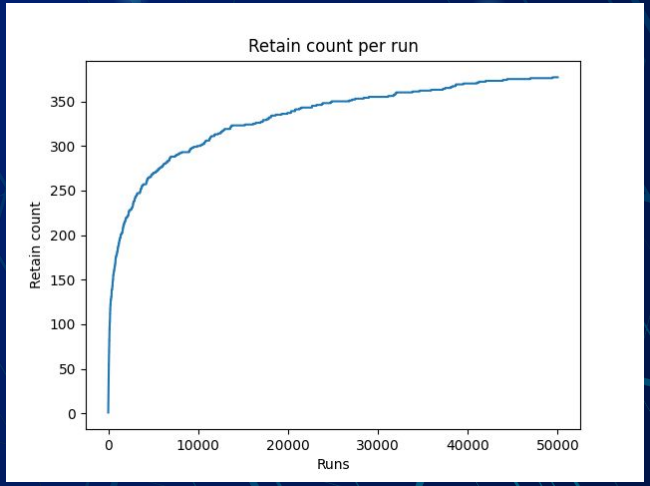
Testing



- Retrieve-Reuse time tends to flatten around: 0.026s
- Retain time tends to flatten around: 0.045s
- The number of retained instances (problem + solution) tends to flatten around: 375 instances







Resource Accounting

Hours per team member per task

Who	Meetings	Requirements, Definition	Representation	Retrieval	Reuse	Revise	Retain	GUI	Report	Presentation
Andrea	4.5	1	0	0	0	16	26	1	5	5
Kevin	7	8.5	8	0.5	10	0	0	17	7	2.5
Mike	6.5	12	4	0	16	1	2	1	15	4
Víctor	6	9	8	8	1	1	0	17	8	3

Conclusions

Future Work

- More **natural input** from user: **BERT?**
- **Server back-end**
  - Simultaneous requests
  - Read-write problem
  - Web scraping to obtain prices
- **Handcrafted adaptation rules**
  - Solution as tree-like structure
  - **Multivariate optimization problem:** genetic algorithms?

Final Thoughts

- **Requirements** were **accomplished**
- **Intuitive PC Configuration via GUI** and CLI (see **demo now!**)
- **User is not** required to be **an expert**
- **New solutions** via weighted adaptation and constraints
- In-depth empirical study for the obtainment of the **retain threshold.**
- **Study of storage and execution time**
  - **5 times faster** than expected
- Occasionally, results may contain **PC bottleneck problems**

Thanks!

Do you have any questions?