APPROXIMATE ORACLES AND SYNERGY IN SOFTWARE ENERGY SEARCH SPACES

Bobby R. Bruce, Justyna Petke, Mark Harman, and Earl T. Barr

IN THE BEGINNING

Reducing Energy Consumption Using Genetic Improvement

Bobby R. Bruce
University College London
London
United Kingdom
r.bruce@cs.ucl.ac.uk

Justyna Petke
University College London
London
United Kingdom
j.petke@ucl.ac.uk

Mark Harman
University College London
London
United Kingdom
mark.harman@ucl.ac.uk

ABSTRACT

Genetic Improvement (GI) is an area of Search Based Software Engineering which seeks to improve software's non-functional properties by treating program code as if it were genetic material which is then evolved to produce more optimal solutions. Hitherto, the majority of focus has been on optimising program's execution time which, though important, is only one of many non-functional targets. The growth in mobile computing, cloud computing infrastructure, and ecological concerns are forcing developers to focus on the energy their software consumes. We report on investigations into using GI to automatically find more energy efficient versions of the MiniSAT Boolean satisfiability solver when specialising for three downstream applications. Our results find that GI can successfully be used to reduce energy consumption by up to 25%.

Categories and Subject Descriptors

D.2 [**Software**]: Software Engineering

Keywords

Search based software engineering, SBSE, genetic improvement, GI, optimisation, energy optimisation, energy efficiency, energy consumption, Boolean satisfiability

1. INTRODUCTION

Less than a decade ago the quality of software (outside of end-user design preferences) could broadly be described as the extent to which software met its specification while minimising the prevalence of bugs and usage of traditional computer resources such as CPU time and memory allocation. The growth in two new technologies, mobile computing devices and cloud services, has led to a new environment for software engineers where they must now consider the energy an application consumes; the quality of software is now measured in Joules, as well as bug counts, seconds, and megabytes. At present there are more smartphones in the world than personal computers [22], each containing a limited store of energy between charges that must be used efficiently. The energy required to run large server clusters has grown considerably in the last decade, estimated to be between 1.1% to 1.5% of global electricity consumption in 2010 [26], putting strain on energy suppliers and the budgets of those responsible for purchasing this energy [7]. The total ICT infrastructure generated 1.9% of global CO_2 emissions in 2011 [5] (larger than the entire United Kingdom estimated at 1.47% for the 2010-2014 period [42]) indicating that computer science has a role to play in mitigating climate change.

Thus we believe it important that software engineers find ways of programming computers with energy efficiency in mind to appease the demands from consumers for longer battery life, from companies to reduce their energy bills, and from society's desire to minimise humanity's impact on the environment.

One of the largest hurdles in producing energy-efficient software is the developer's disconnect between the source code they write and the energy that will be consumed from the compiled product they deliver [33]. Without a deep understanding of how a particular compiler works, along with an equally deep understanding of how much energy a given instruction will consume, the problem remains difficult for many developers. It has been found that metrics previously believed to guide developers to more energy efficient solutions are, in reality, poor at doing so [38]. Subtle changes, such as introducing inline methods [41], swapping API implementations [33], and constructing semantically equivalent (but structurally inequivalent) algorithms [8] have all been shown to influence energy consumption. However this influence is difficult to determine outside of the ad hoc and inefficient process of trial-and-error. Tools have been developed to guide users to energy-inefficient areas of their software [2, 11, 30, 19] though the developer retains responsibility for rectifying these inefficiencies.

We suggest that the most under explored method of decreasing software's energy consumption lies in automated processes. Such processes would allow developers to focus solely on meeting the specification requirements with worries about non-functional attributes like energy consumption left to an algorithm capable of refactoring software to a more optimal state.

Genetic Improvement (GI) [20, 25, 27, 28, 29, 36, 37, 45, 44] is a Search Based Software Engineering (SBSE) technique [21] which treats program code as if it were genetic material that can then be evolved to produce optimised solutions. GI has previously been found effective at optimis-

GECCO '15, July 11 - 16, 2015, Madrid, Spain
© 2015 ACM. ISBN 978-1-4503-3472-3/15/07…$15.00
DOI: http://dx.doi.org/10.1145/2739480.2754752

- *Published at GECCO 2015*
- *Used Genetic Improvement to reduce the energy consumption of MiniSAT*

Some "Threats to Validity"

- *Only for 1 (very small) piece of software*
- *Limited number of runs*
- *Evidence our current approach to Genetic Improvement was not ideal*
- *Indirect energy measurement (Intel Power Gadget)*

FOLLOWUP: LET'S DO THE SAME, BUT BIGGER!

FOLLOWUP: LET'S DO THE SAME, BUT BIGGER!

Real, direct energy measurements!

FOLLOWUP: LET'S DO THE SAME, BUT BIGGER!

Real, direct energy measurements!

More applications!

FOLLOWUP: LET'S DO THE SAME, BUT BIGGER!

Real, direct energy measurements!

More applications!

Larger applications!

FOLLOWUP: LET'S DO THE SAME, BUT BIGGER!

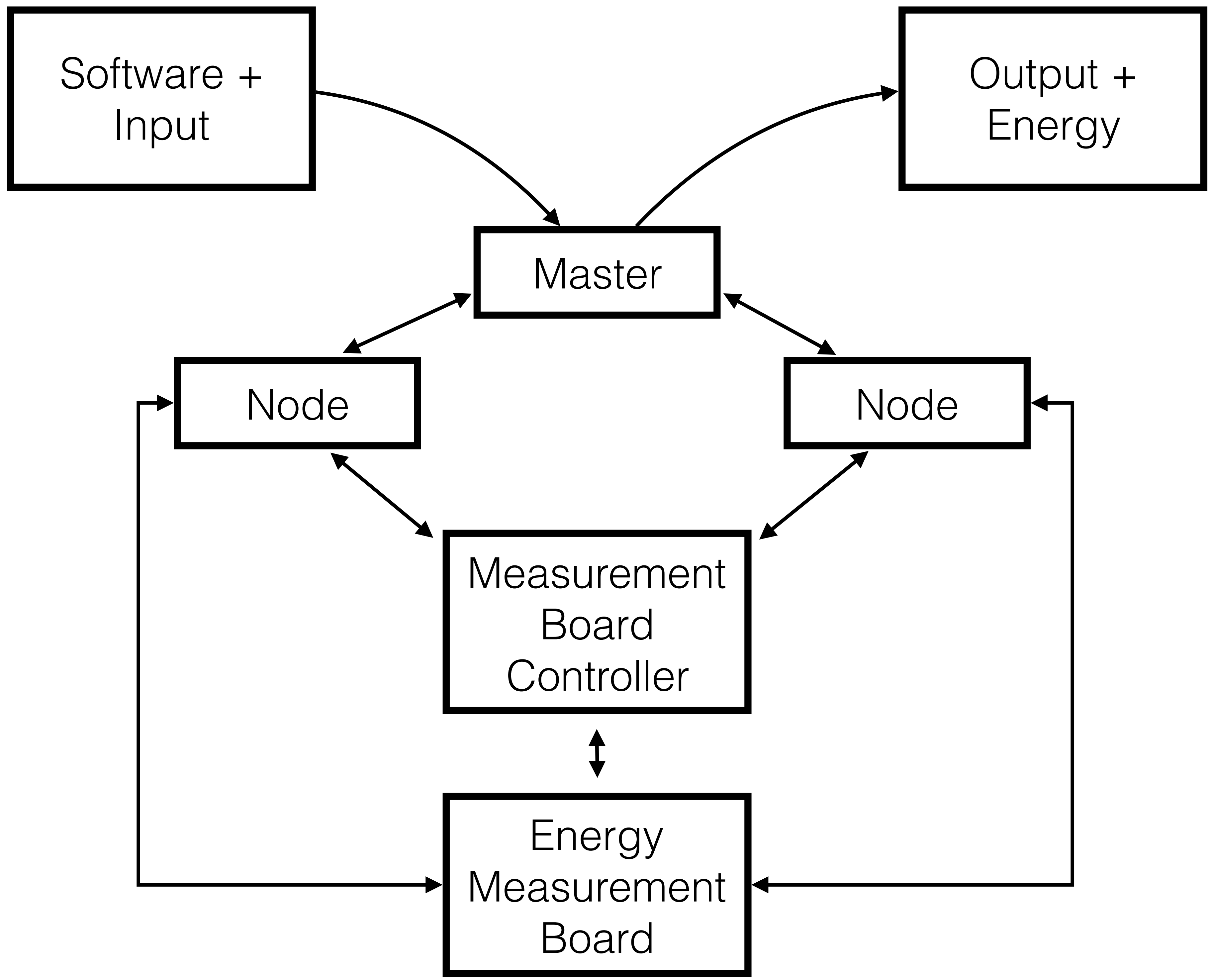
Real, direct energy measurements!

More applications!

Larger applications!

More evaluations!

THE FRAMEWORK WE DEVELOPED



THE COPY, DELETE, AND REPLACE OPERATORS

THE COPY, DELETE, AND REPLACE OPERATORS

<IF_fileA_40>

THE COPY, DELETE, AND REPLACE OPERATORS

<IF_fileA_40>

<for1_fileB_6><for1_fileB_78>

THE COPY, DELETE, AND REPLACE OPERATORS

<IF_fileA_40>

<for1_fileB_6><for1_fileB_78>

<fileA_101>+<fileA_15>

THE COPY, DELETE, AND REPLACE OPERATORS

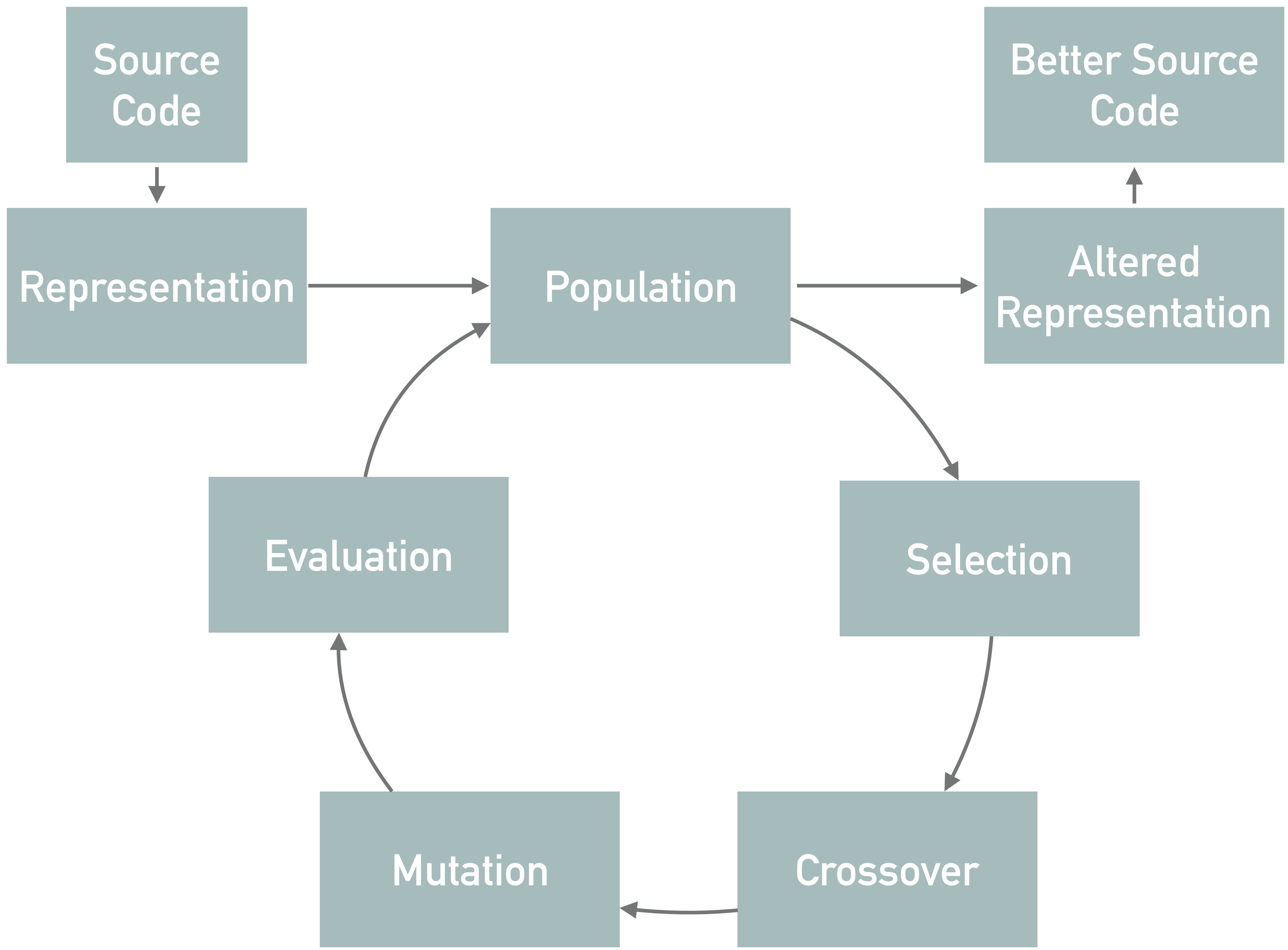
<for1_fileB_6><for1_fileB_78> *<IF_fileA_40>*

<fileA_101>+<fileA_15>

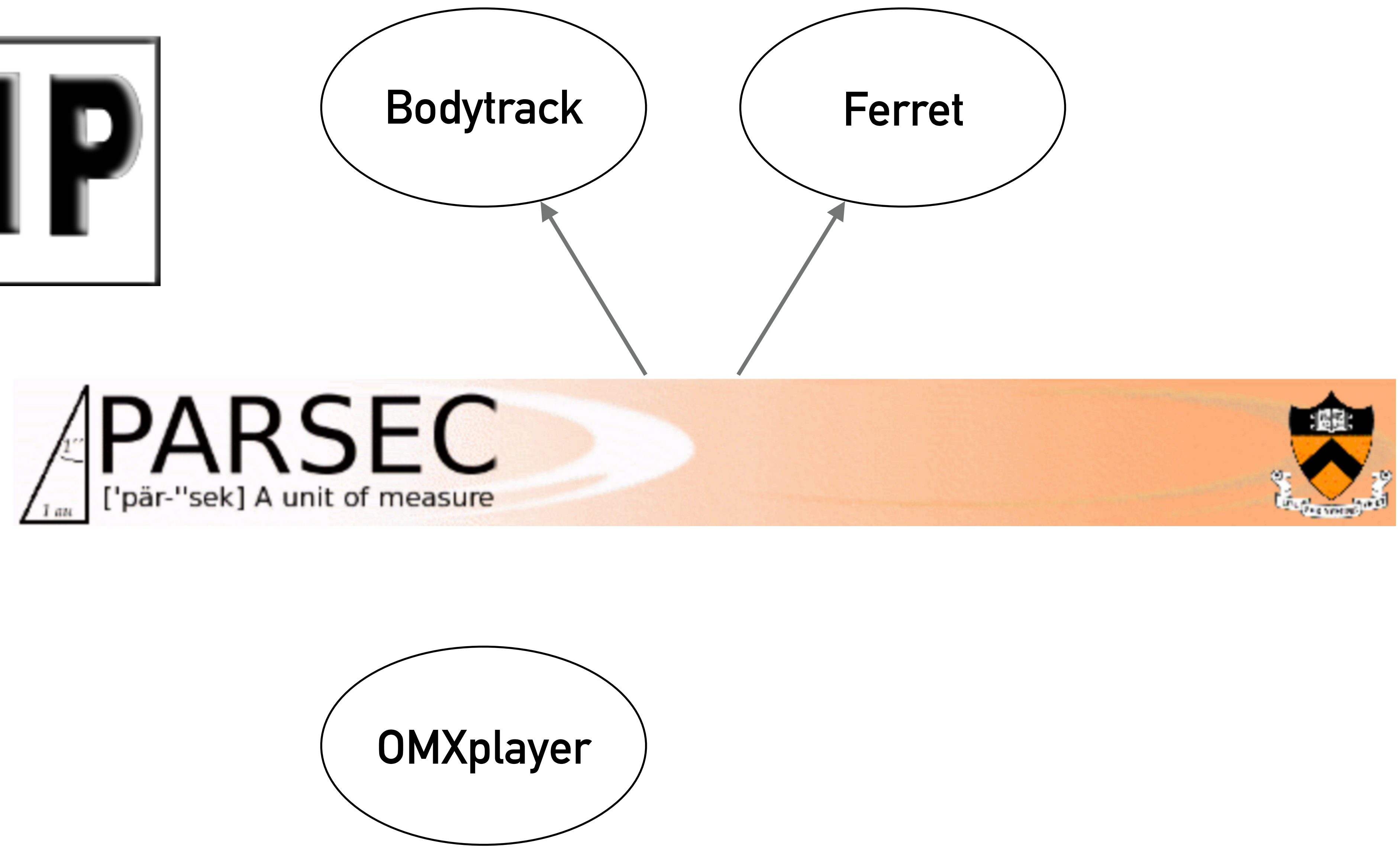
THE COPY, DELETE, AND REPLACE OPERATORS

<for1_fileB_6><for1_fileB_78> *<IF_fileA_40>* *<fileA_101>+<fileA_15>*

THE GREAT GENETIC IMPROVEMENT CYCLE



APPLICATIONS STUDIED



INITIAL INVESTIGATION

App	Pop Size	Num Gen	Improvement?
7zip	100	20	Nope
Bodytrack	100	20	Nope
Bodytrack	100	50	Nope

Maybe 7zip and Bodytrack can't be optimised very much?

Maybe 7zip and Bodytrack can’t be optimised very much?

Maybe our search algorithm isn’t suitable?

Maybe 7zip and Bodytrack can't be optimised very much?

Maybe our search algorithm isn't suitable?

How effective can an individual modification be?

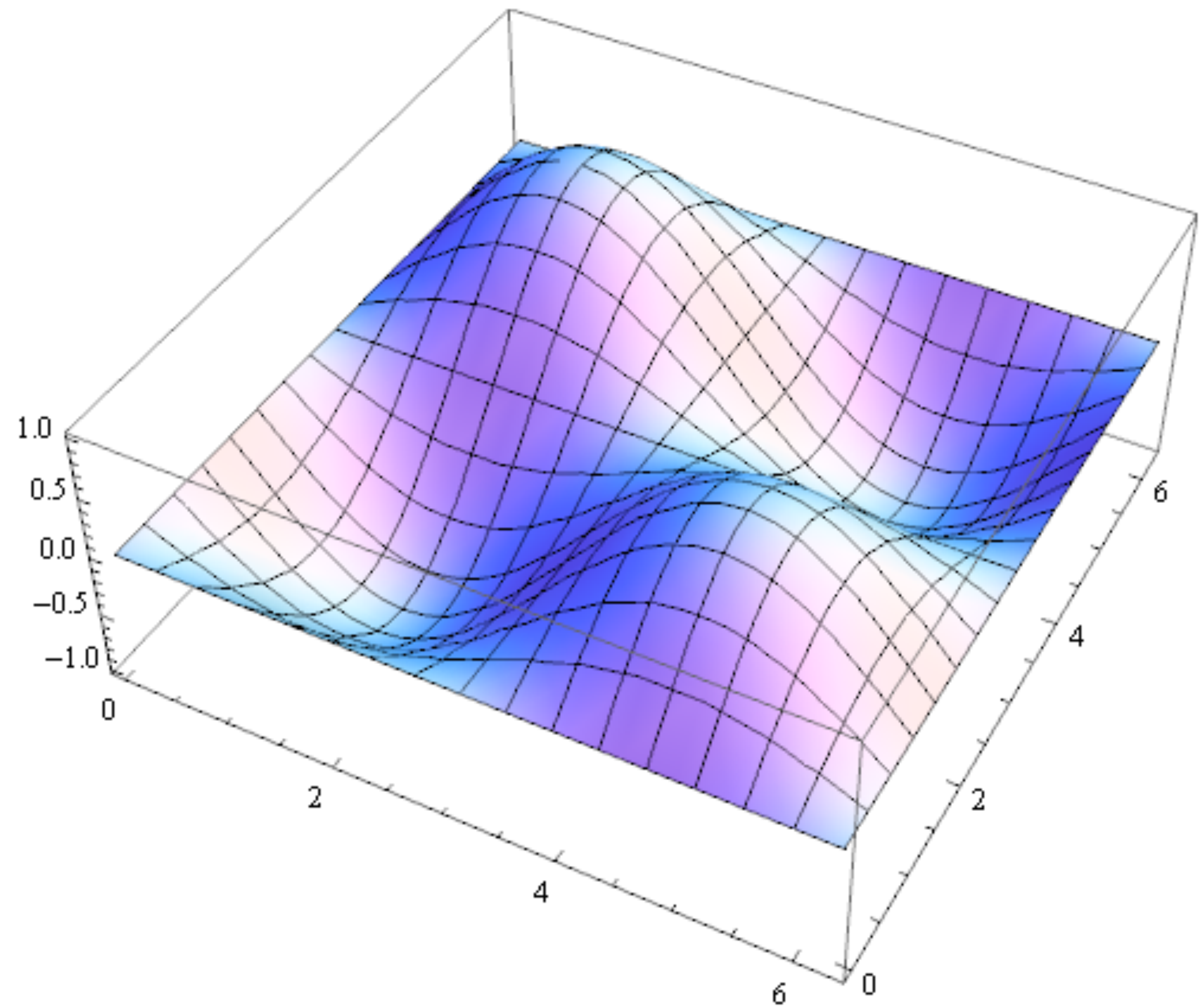
Maybe 7zip and Bodytrack can't be optimised very much?

Maybe our search algorithm isn't suitable?

How effective can an individual modification be?

If an individual modification's effectiveness is small, can it be detected?

What does the search space look like?



RESEARCH QUESTIONS

***RQ1**: Measurement: What variance occurs when measuring energy consumption?*

***RQ1a**: What is the variance when measuring using a single energy measurement device?*

***RQ1b**: What is the variance in direct energy measurements across multiple devices?*

***RQ1c**: What is the variance in proportional energy changes across multiple devices?*

***RQ2**: Improvement: What additional energy improvement can be achieved when using approximate oracles in place of exact oracles?*

***RQ3**: Synergy: How frequently do synergistic or antagonistic effects occur when combining known effective modifications?*

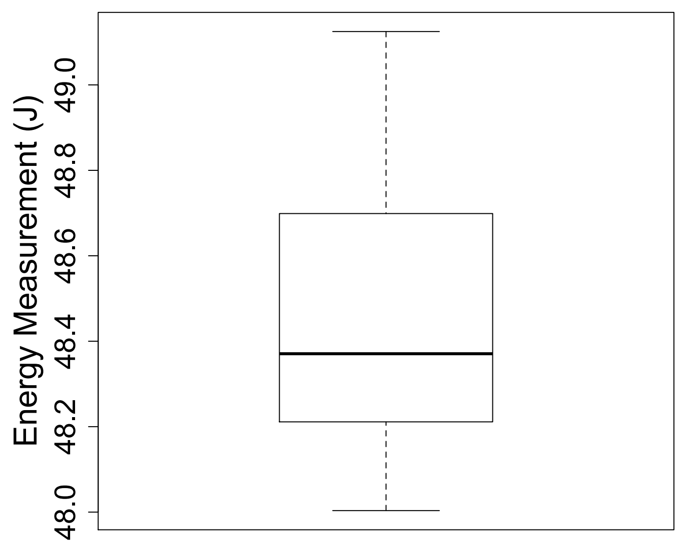
THE SOFTWARE TARGETS

App	Modifiable Lines of code	What does "passed" mean?	What can be approximated?
7zip	2,524	Creates a valid .7z file	Compression Rate
Bodytrack	1,030	Creates a non-empty, readable output	The accuracy of the tracking
Ferret	5,032	Creates a non-null, readable list of images	The ranking
OMXPlayer	5,184	Outputs data to the HDMI port	The video quality

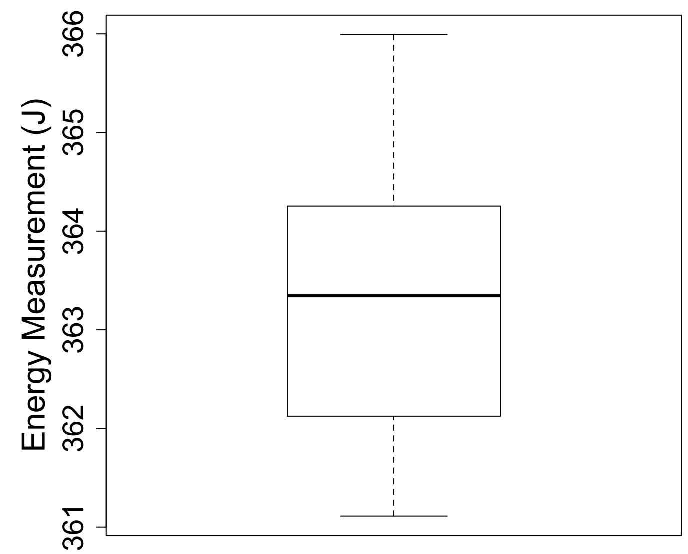
RQ1, MEASUREMENT

WHAT VARIANCE OCCURS WHEN MEASURING ENERGY CONSUMPTION?

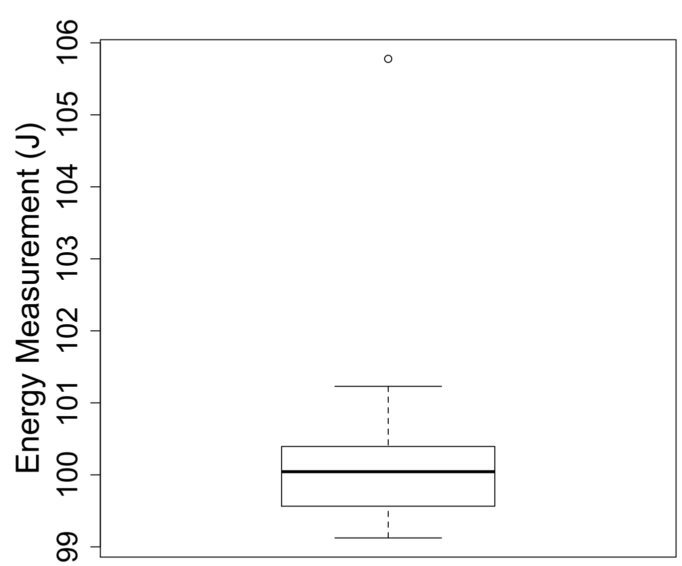
RQ1a: What is the variance when measuring using a single measurement device?



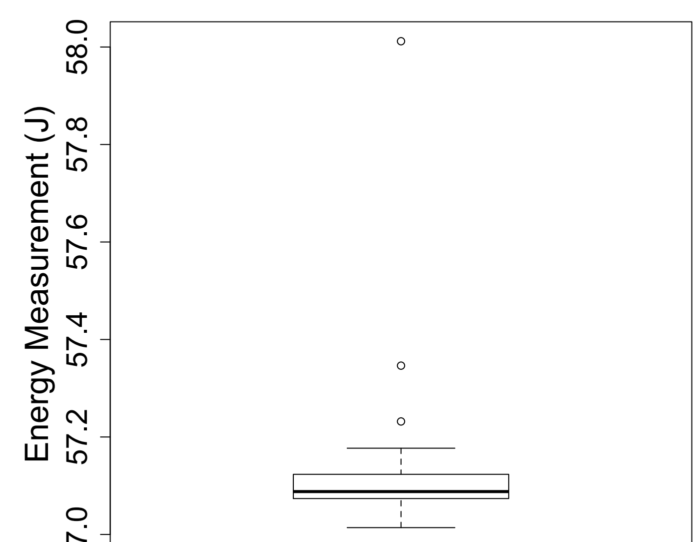
7zip



Ferret



Bodytrack

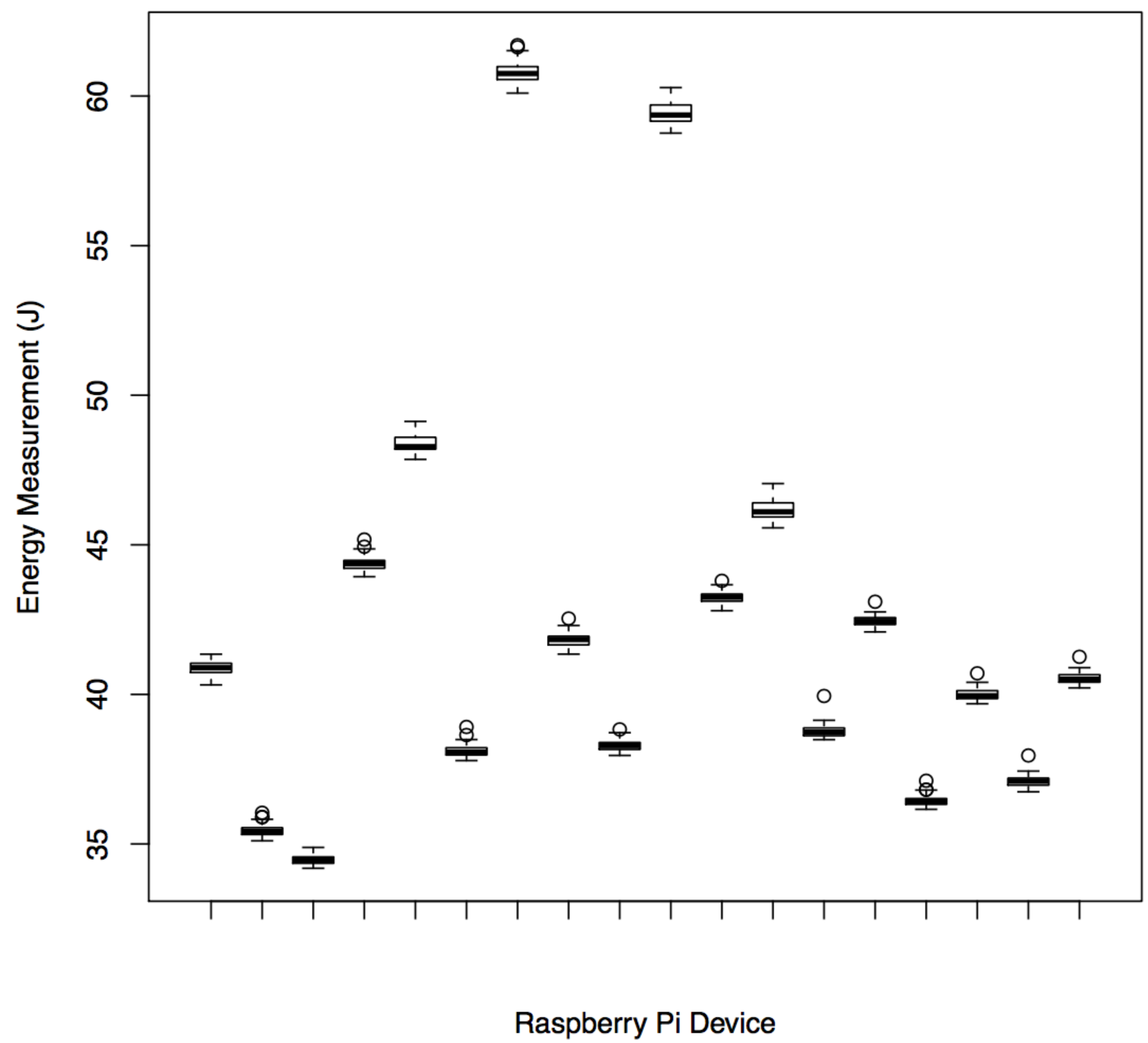


OMXPlayer

RQ1, MEASUREMENT

WHAT VARIANCE OCCURS WHEN MEASURING ENERGY CONSUMPTION?

RQ1b: What is the variance in direct energy measurements across multiple devices?

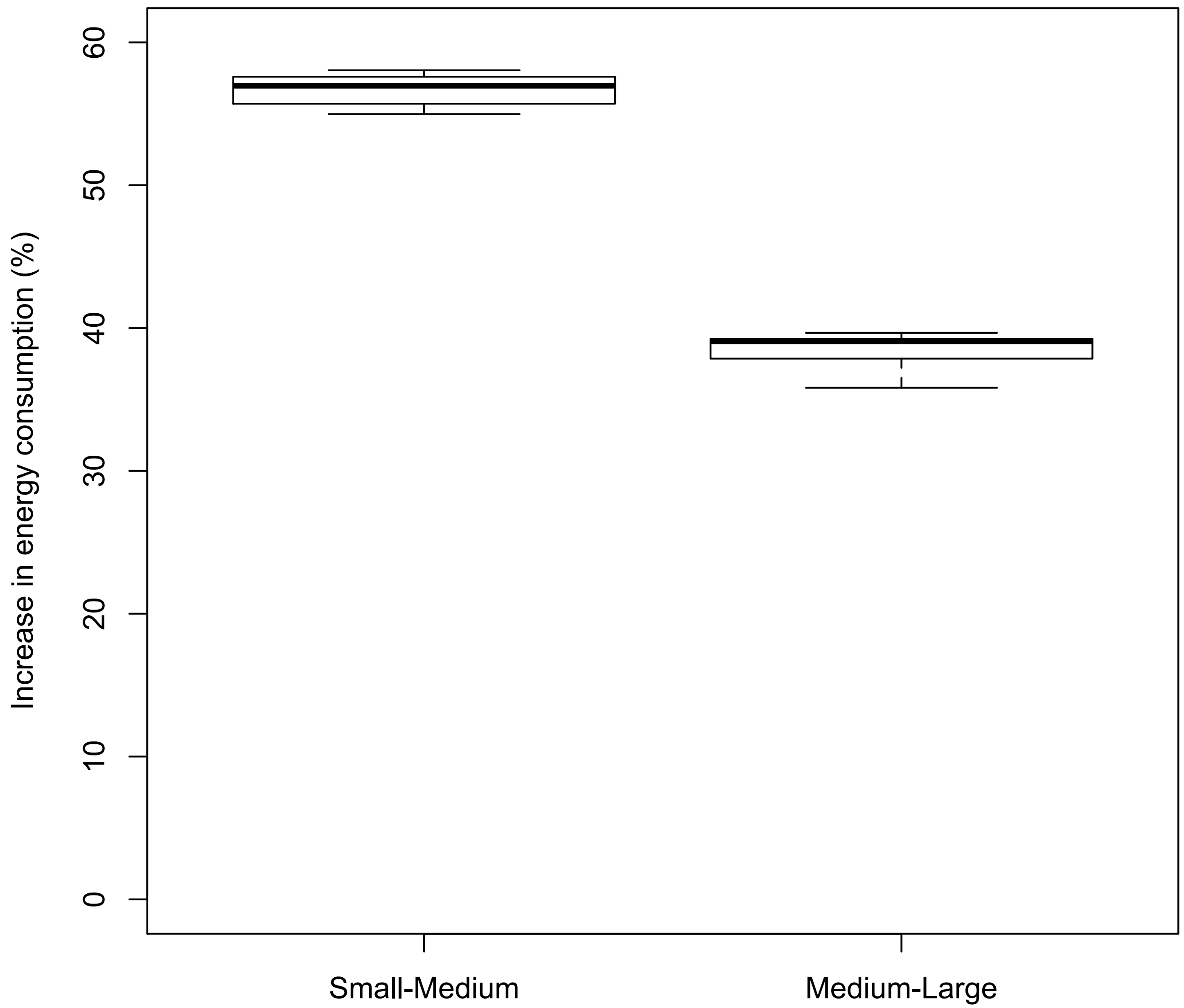


Bobby R. Bruce

RQ1, MEASUREMENT

WHAT VARIANCE OCCURS WHEN MEASURING ENERGY CONSUMPTION?

RQ1c: What is the variance in proportional energy changes across multiple devices?



RQ1 CONCLUSIONS

- The MAGEEC Energy Measurement boards have provided us with a framework capable of acceptable precision but low accuracy.
- While the readings, when reported in Joules, vary wildly, the proportional changes are relatively stable. Thus proportional change (i.e. a percentage increase or decrease) is recorded.
- The variation of energy readings within devices appear acceptably small to detect meaningful changes when they occur

WHAT ADDITIONAL ENERGY IMPROVEMENT CAN BE ACHIEVED WHEN USING APPROXIMATE TEST ORACLES IN PLACE OF EXACT TEST ORACLES?

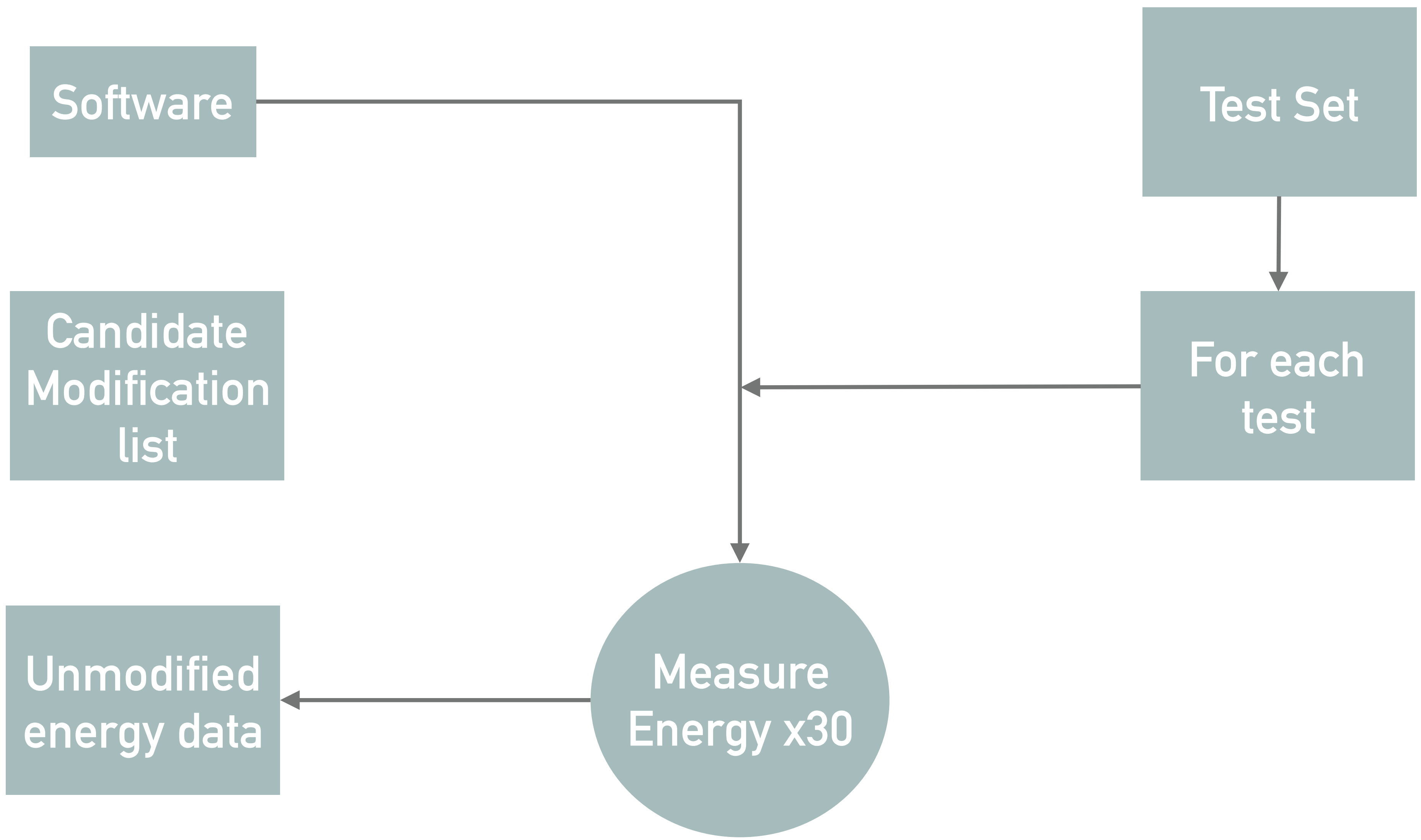
Software

Test Set

Candidate Modification list

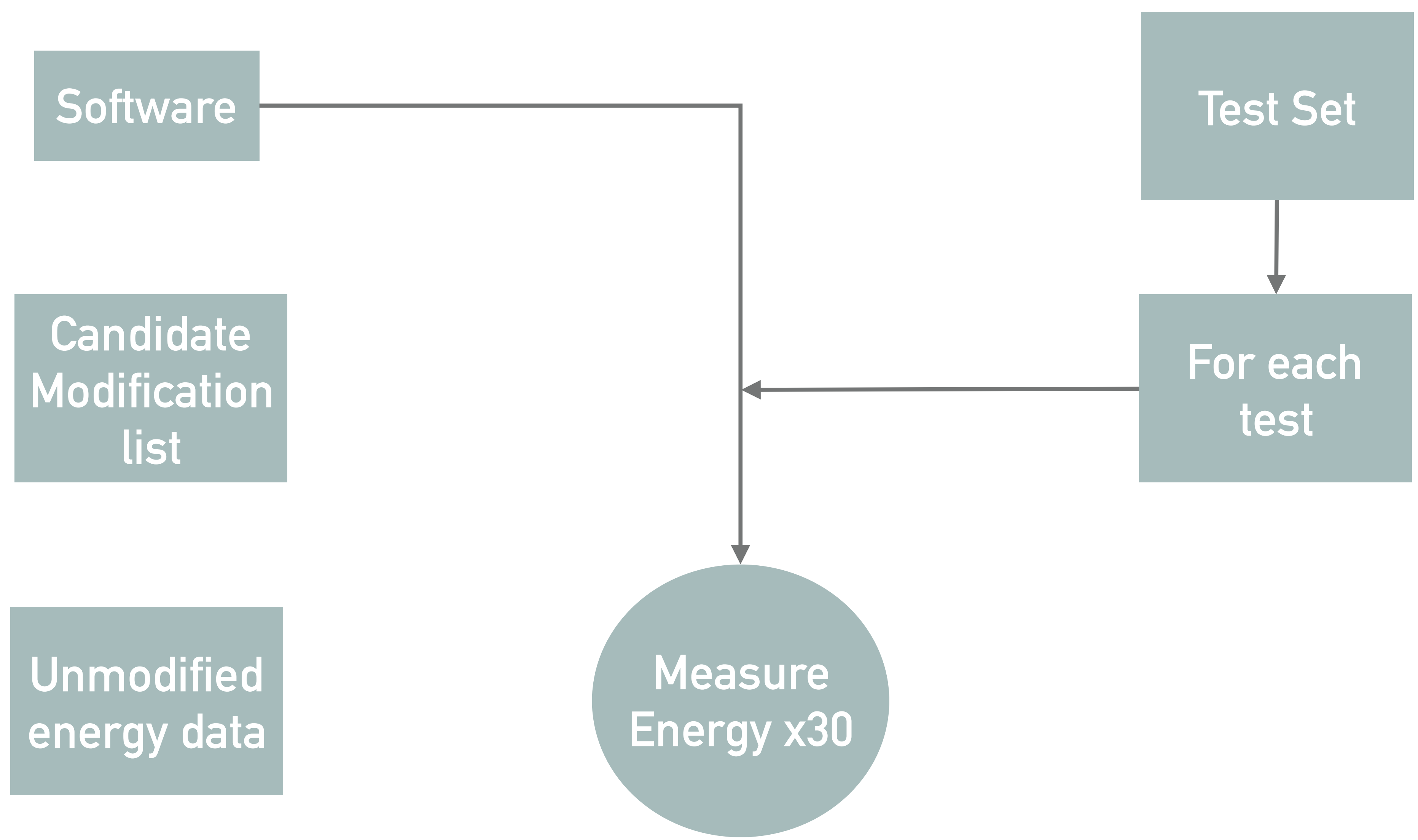
RQ2, IMPROVEMENT

WHAT ADDITIONAL ENERGY IMPROVEMENT CAN BE ACHIEVED WHEN USING APPROXIMATE TEST ORACLES IN PLACE OF EXACT TEST ORACLES?



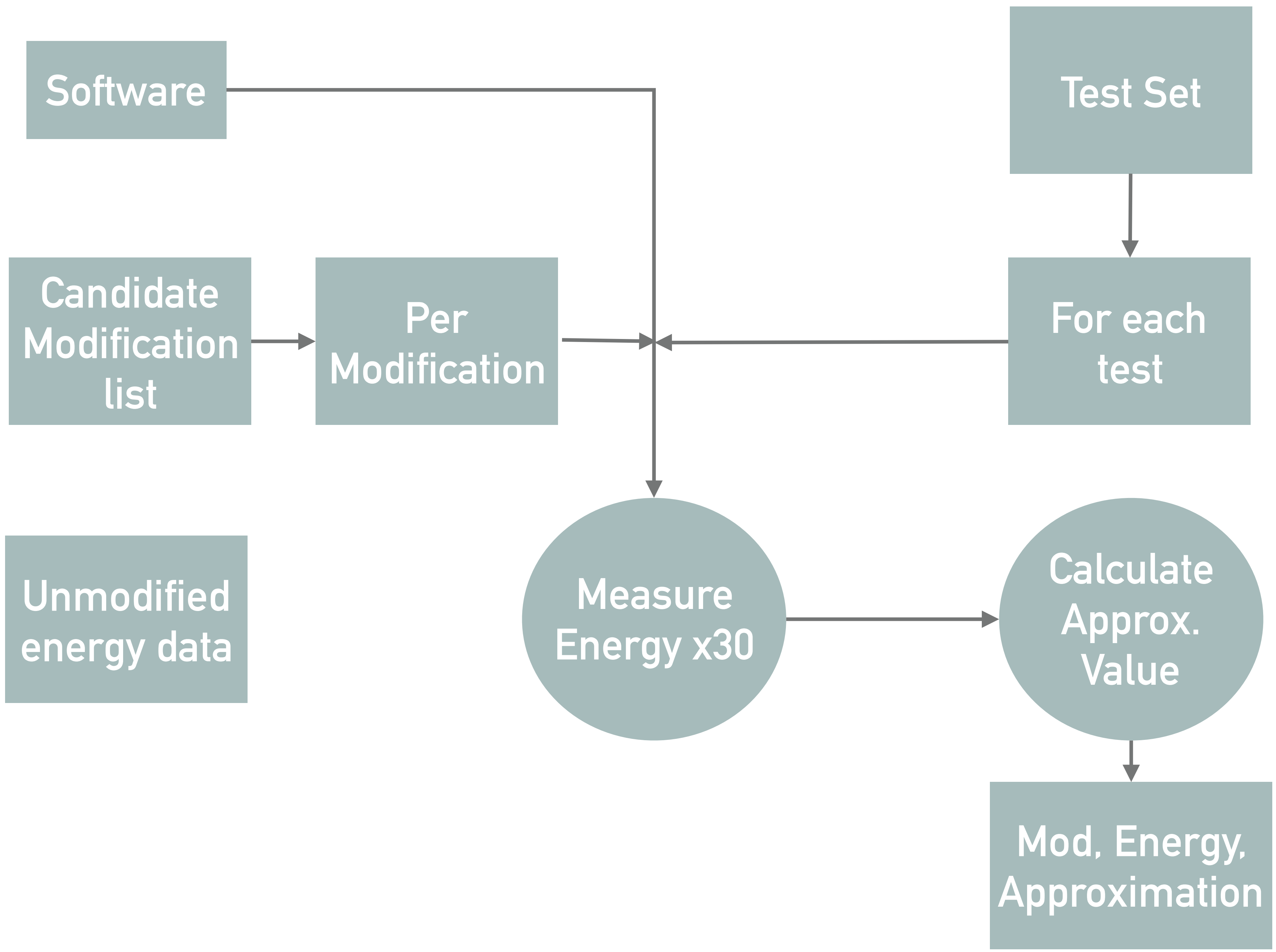
RQ2, IMPROVEMENT

WHAT ADDITIONAL ENERGY IMPROVEMENT CAN BE ACHIEVED WHEN USING APPROXIMATE TEST ORACLES IN PLACE OF EXACT TEST ORACLES?



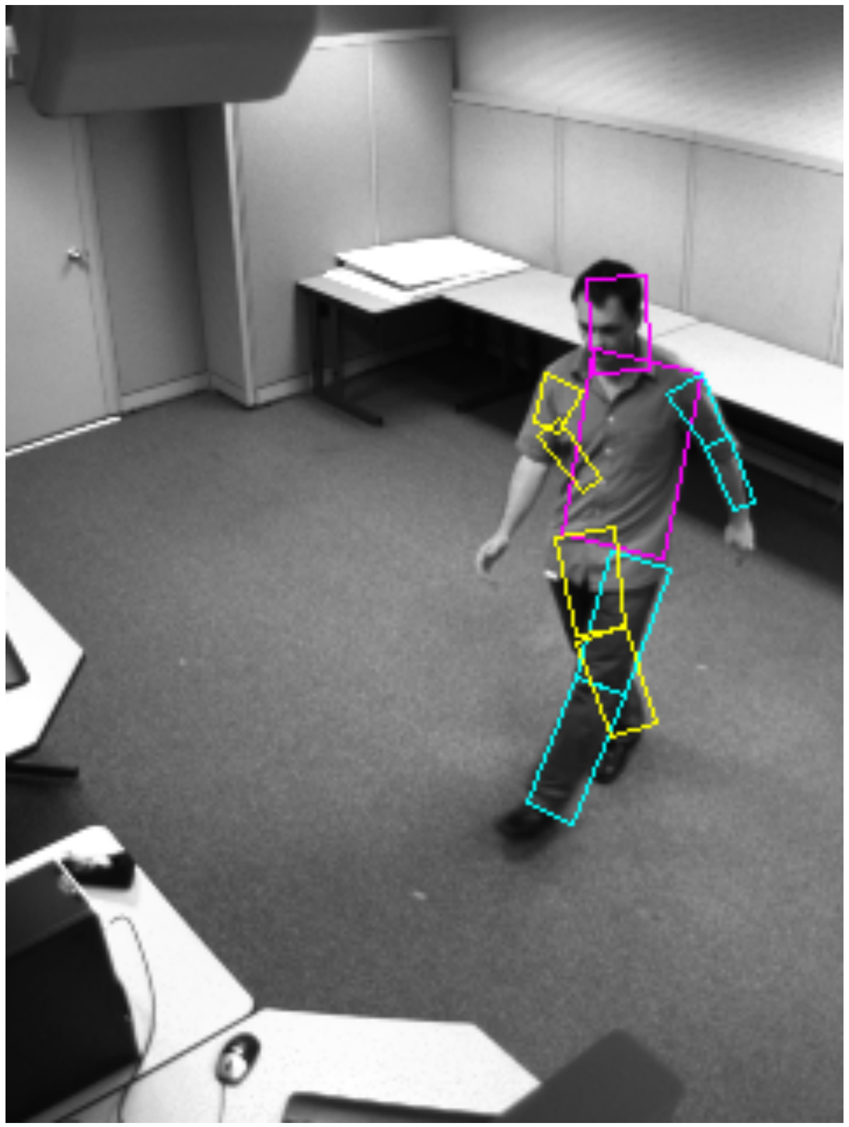
RQ2, IMPROVEMENT

WHAT ADDITIONAL ENERGY IMPROVEMENT CAN BE ACHIEVED WHEN USING APPROXIMATE TEST ORACLES IN PLACE OF EXACT TEST ORACLES?

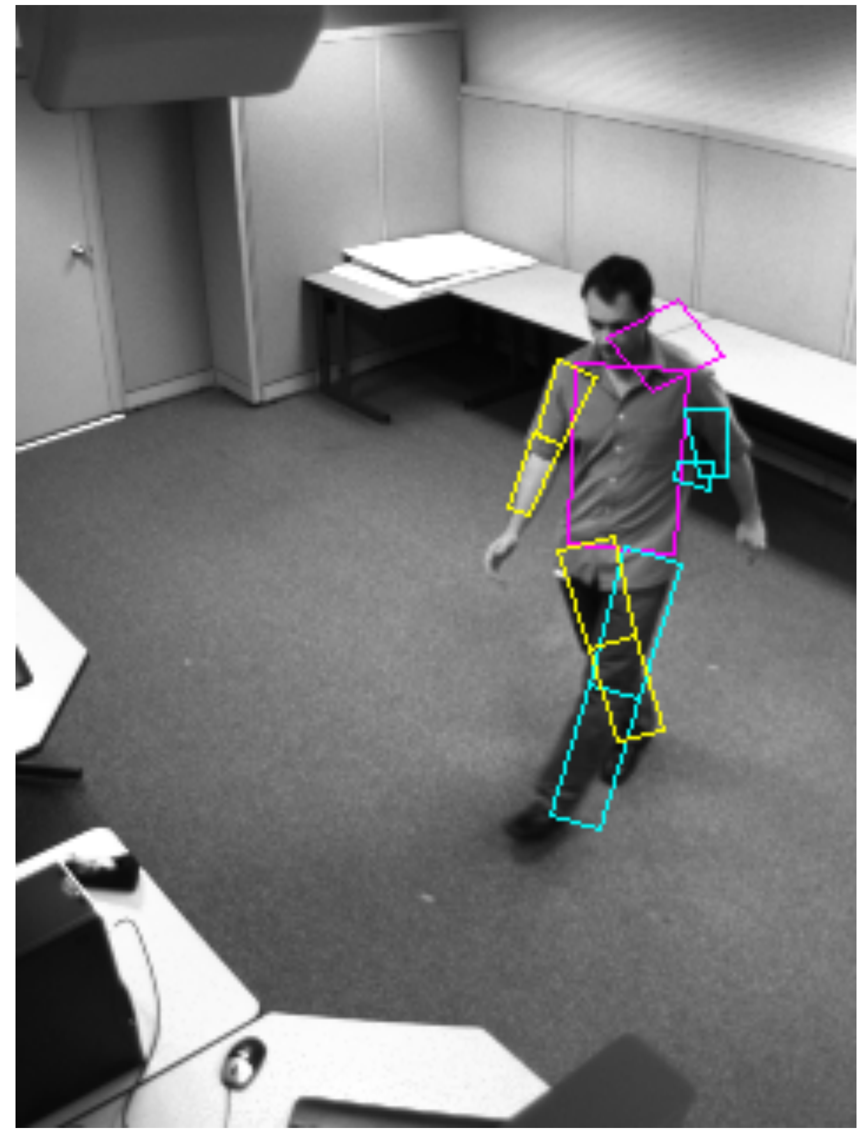


WHAT ADDITIONAL ENERGY IMPROVEMENT CAN BE ACHIEVED WHEN USING APPROXIMATE TEST ORACLES IN PLACE OF EXACT TEST ORACLES?

Bodytrack



Original



33% Energy Reduction

WHAT ADDITIONAL ENERGY IMPROVEMENT CAN BE ACHIEVED WHEN USING APPROXIMATE TEST ORACLES IN PLACE OF EXACT TEST ORACLES?

Bodytrack

Energy reduction	Approximation Value
2.69%	0.000
19.26%	0.131
27.97%	0.170
29.13%	0.192
33.69%	0.452

RQ2B, IMPROVEMENT

WHAT ADDITIONAL ENERGY IMPROVEMENT CAN BE ACHIEVED WHEN USING APPROXIMATE TEST ORACLES IN PLACE OF EXACT TEST ORACLES?

7zip

Energy reduction	Approximation Value
5.08%	3.93×10^{-4}
12.29%	0.072
13.17%	0.102
48.30%	0.741

WHAT ADDITIONAL ENERGY IMPROVEMENT CAN BE ACHIEVED WHEN USING APPROXIMATE TEST ORACLES IN PLACE OF EXACT TEST ORACLES?

Ferret

Energy reduction	Approximation Value
43.19%	0.154
60.79%	39.873
75.53%	78.800
75.55%	1550.200
76.21%	2669.710
79.88%	6221.220

RQ2B, IMPROVEMENT

WHAT ADDITIONAL ENERGY IMPROVEMENT CAN BE ACHIEVED WHEN USING APPROXIMATE TEST ORACLES IN PLACE OF EXACT TEST ORACLES?

Ferret

Energy reduction	Approximation Value
43.19%	0.154
~~60.79%~~	~~39.873~~
~~75.53%~~	~~78.800~~
~~75.55%~~	~~1550.200~~
~~76.21%~~	~~2669.710~~
~~79.88%~~	~~6221.220~~

WHAT ADDITIONAL ENERGY IMPROVEMENT CAN BE ACHIEVED WHEN USING APPROXIMATE TEST ORACLES IN PLACE OF EXACT TEST ORACLES?

OMXPlayer

Energy reduction	Approximation Value
2.23%	0.000
78.45%	0.003
92.70%	0.637
95.53%	1.002
95.60%	1.043

WHAT ADDITIONAL ENERGY IMPROVEMENT CAN BE ACHIEVED WHEN USING APPROXIMATE TEST ORACLES IN PLACE OF EXACT TEST ORACLES?

OMXPlayer

Energy reduction	Approximation Value
2.23%	0.000
78.45%	0.003
~~92.70%~~	~~0.637~~
~~95.53%~~	~~1.002~~
~~95.60%~~	~~1.043~~

WHAT ADDITIONAL ENERGY IMPROVEMENT CAN BE ACHIEVED WHEN USING APPROXIMATE TEST ORACLES IN PLACE OF EXACT TEST ORACLES?

Distribution of operators

App	Line deletion	Line copy	Line replace
7zip	5	0	3
Ferret	81	7	69
Bodytrack	44	1	27
OMXPlayer	8	3	13
Percentage	**52.9%**	**4.2%**	**42.9%**

WHAT ADDITIONAL ENERGY IMPROVEMENT CAN BE ACHIEVED WHEN USING APPROXIMATE TEST ORACLES IN PLACE OF EXACT TEST ORACLES?

Average impact of the operators

App	Line deletion	Line copy	Line replace
7zip	16.60%	0.00%	7.42%
Ferret	56.45%	64.74%	43.50%
Bodytrack	8.27%	0.16%	8.31%
OMXPlayer	57.01%	71.86%	64.92%
Percentage	**34.58%**	**46.59%**	**31.04%**

RQ2 CONCLUSIONS

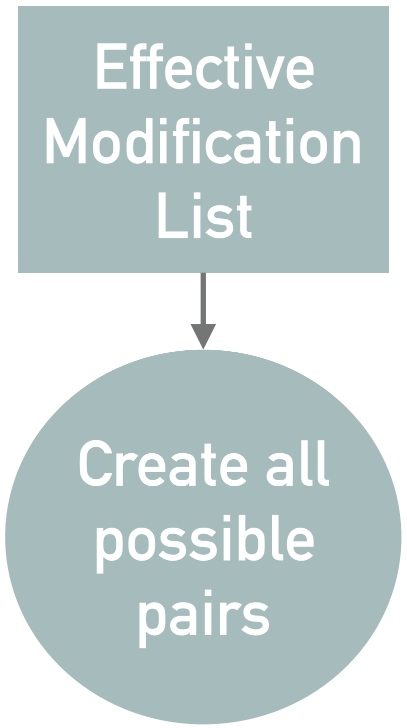
- When approximation is permitted the number of modifications that reduce energy consumption increased
- The modifications are also capable of reducing energy consumption by a greater extent
- Some applications produced better Pareto frontiers than others
- Copy operations are rarely effective though all have roughly the same impact when they are.

HOW FREQUENTLY DO SYNERGISTIC AND ANTAGONISTIC EFFECTS OCCUR WHEN COMBINING KNOWN EFFECTIVE MODIFICATIONS?

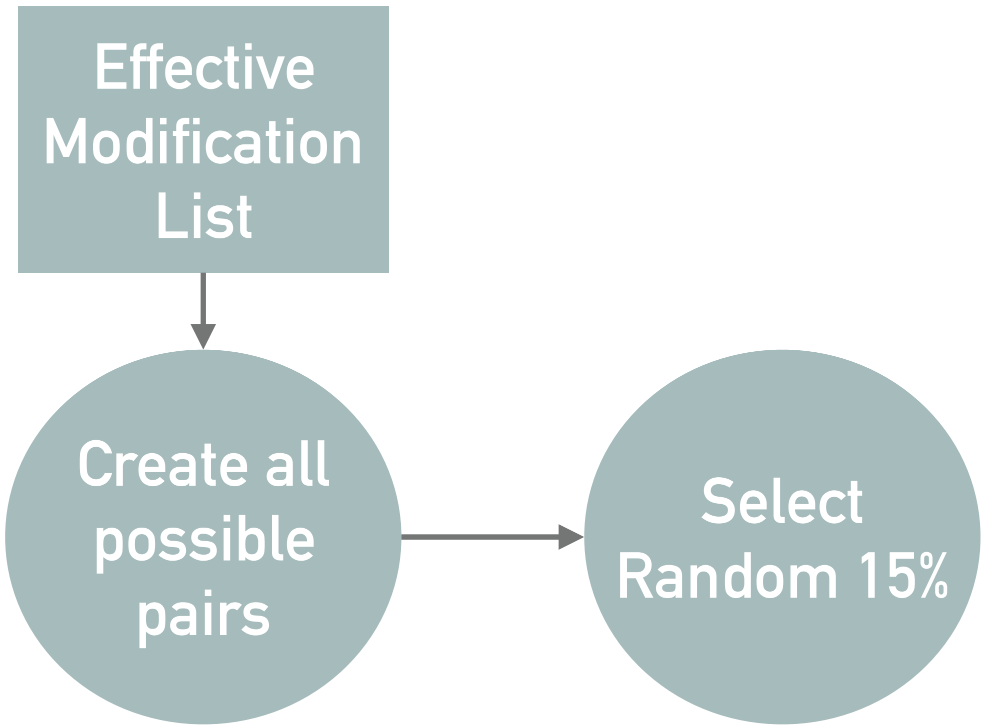
HOW FREQUENTLY DO SYNERGISTIC AND ANTAGONISTIC EFFECTS OCCUR WHEN COMBINING KNOWN EFFECTIVE MODIFICATIONS?

Effective Modification List

HOW FREQUENTLY DO SYNERGISTIC AND ANTAGONISTIC EFFECTS OCCUR WHEN COMBINING KNOWN EFFECTIVE MODIFICATIONS?



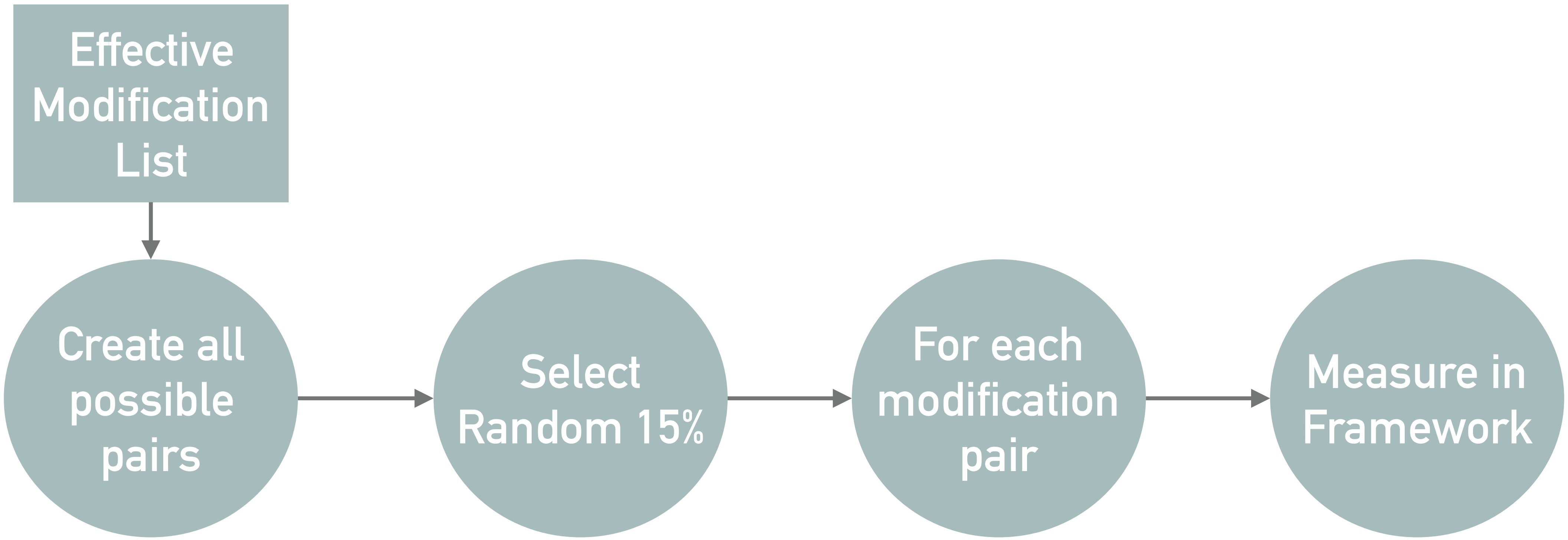
HOW FREQUENTLY DO SYNERGISTIC AND ANTAGONISTIC EFFECTS OCCUR WHEN COMBINING KNOWN EFFECTIVE MODIFICATIONS?



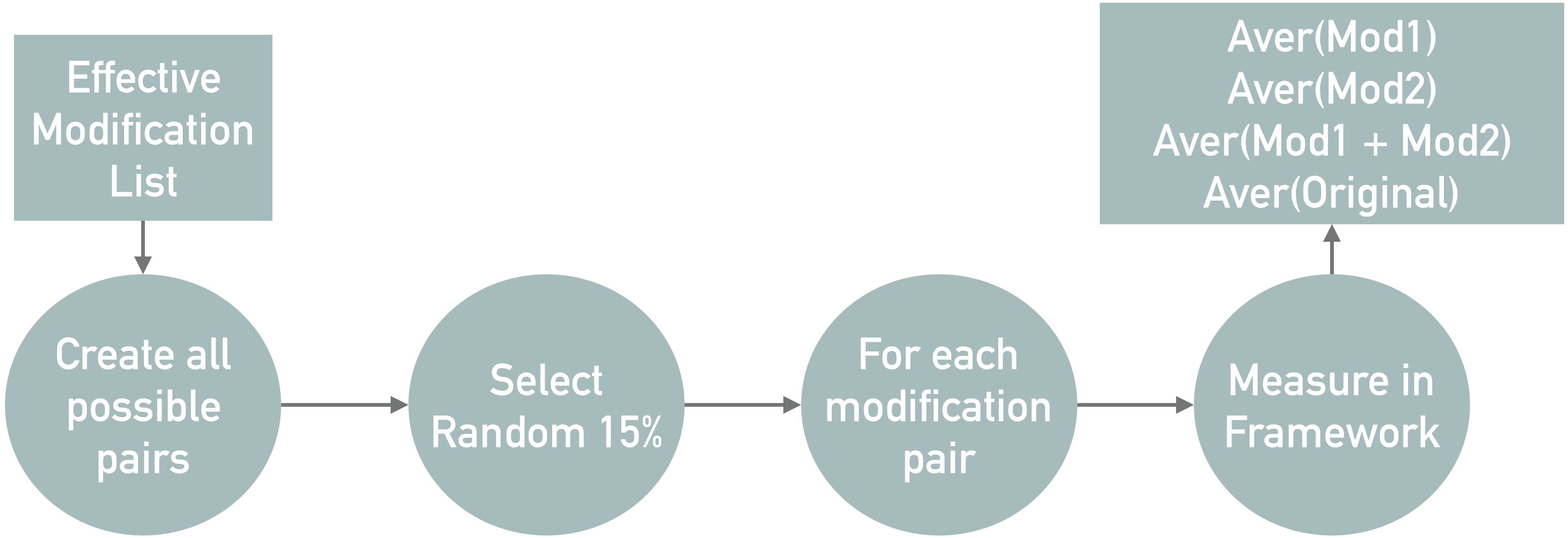
RQ3, SYNERGY

HOW FREQUENTLY DO SYNERGISTIC AND ANTAGONISTIC EFFECTS OCCUR WHEN COMBINING KNOWN EFFECTIVE MODIFICATIONS?

HOW FREQUENTLY DO SYNERGISTIC AND ANTAGONISTIC EFFECTS OCCUR WHEN COMBINING KNOWN EFFECTIVE MODIFICATIONS?

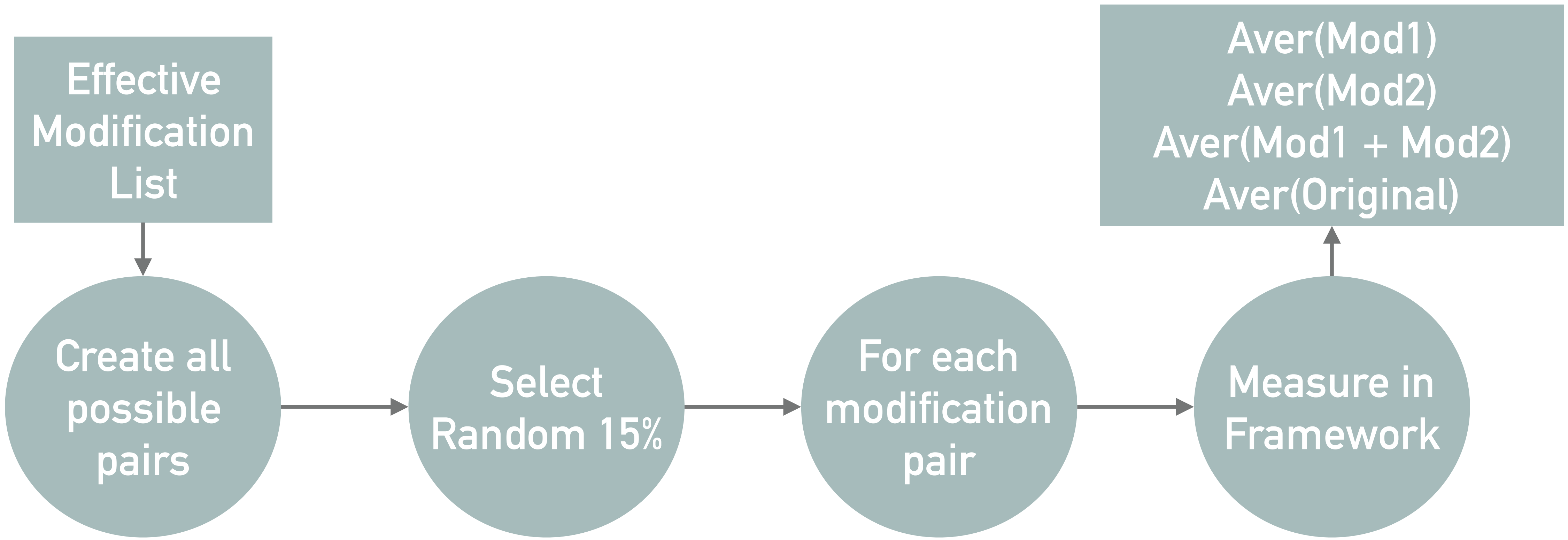


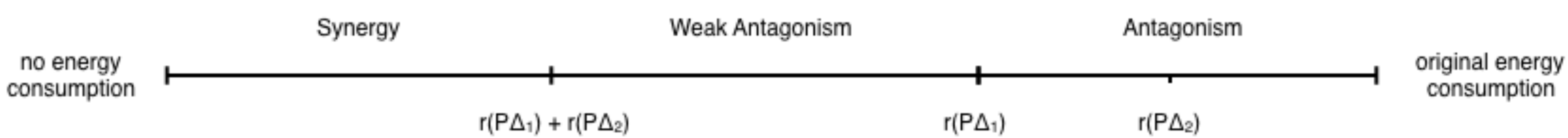
HOW FREQUENTLY DO SYNERGISTIC AND ANTAGONISTIC EFFECTS OCCUR WHEN COMBINING KNOWN EFFECTIVE MODIFICATIONS?



RQ3, SYNERGY

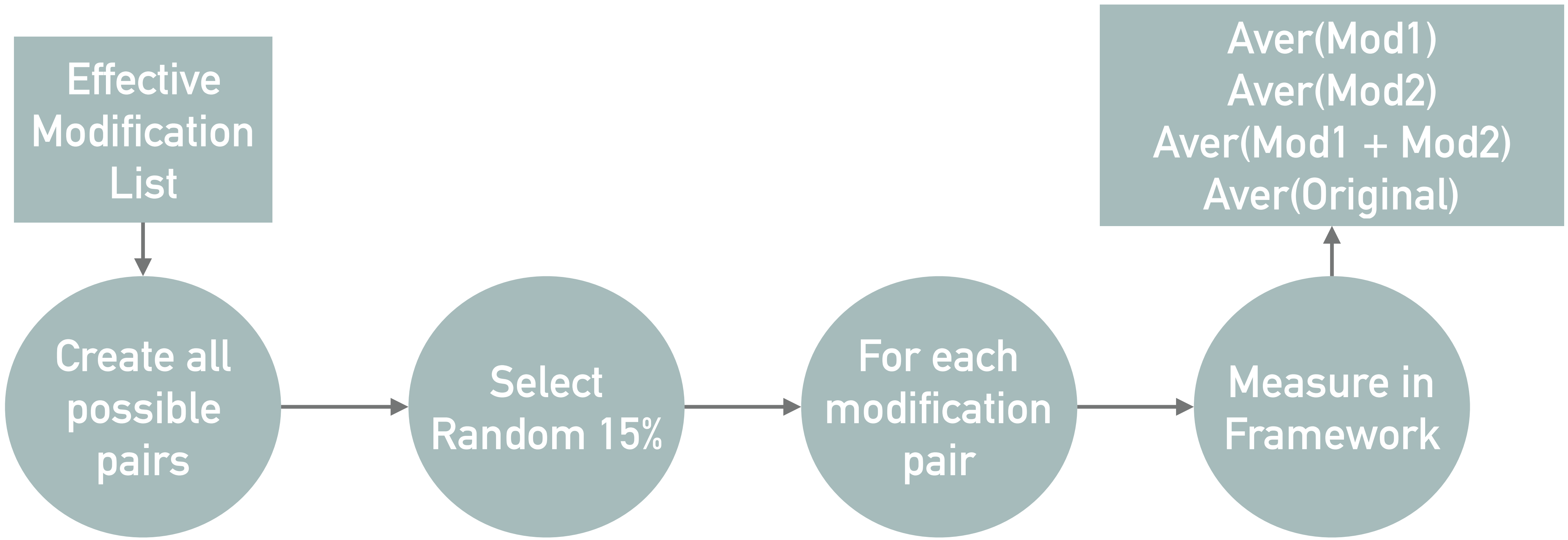
HOW FREQUENTLY DO SYNERGISTIC AND ANTAGONISTIC EFFECTS OCCUR WHEN COMBINING KNOWN EFFECTIVE MODIFICATIONS?

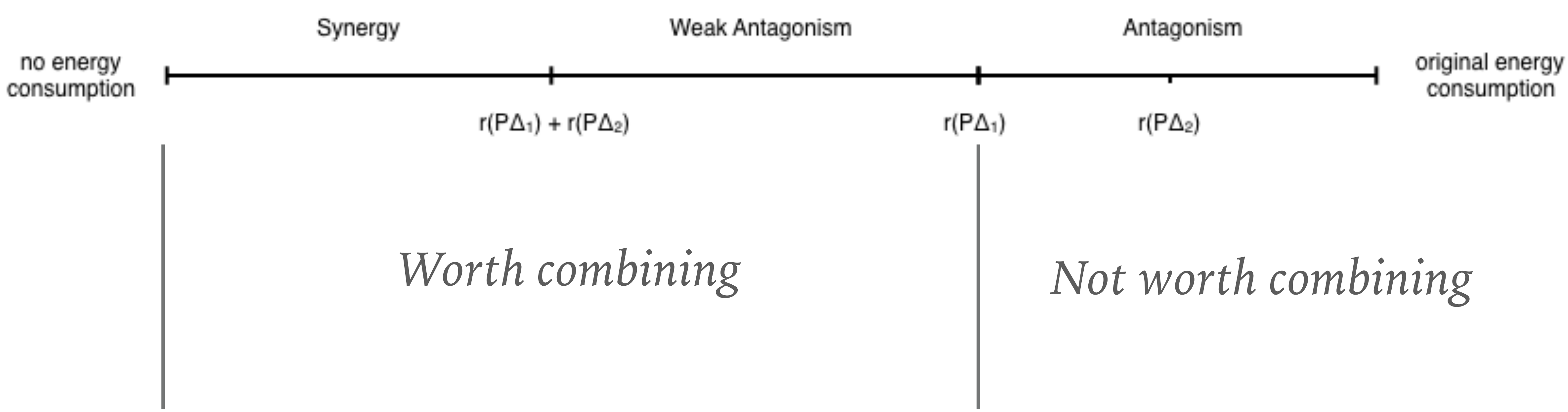




RQ3, SYNERGY

HOW FREQUENTLY DO SYNERGISTIC AND ANTAGONISTIC EFFECTS OCCUR WHEN COMBINING KNOWN EFFECTIVE MODIFICATIONS?



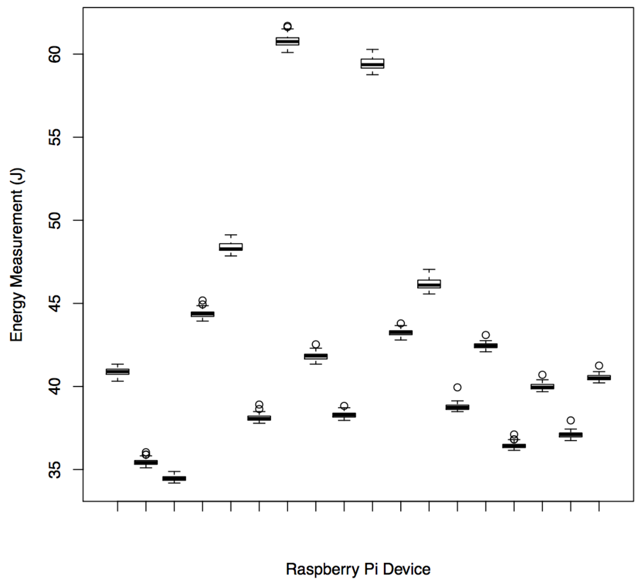


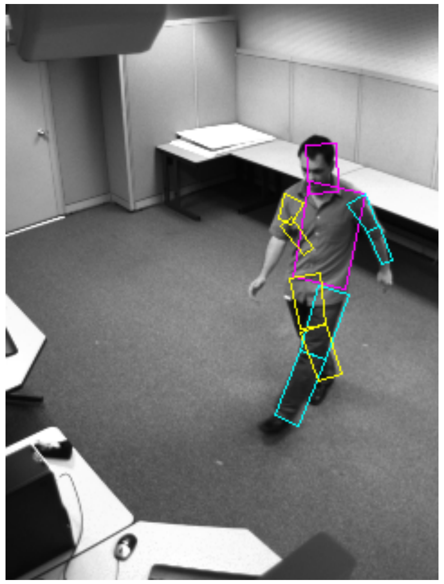
HOW FREQUENTLY DO SYNERGISTIC AND ANTAGONISTIC EFFECTS OCCUR WHEN COMBINING KNOWN EFFECTIVE MODIFICATIONS?

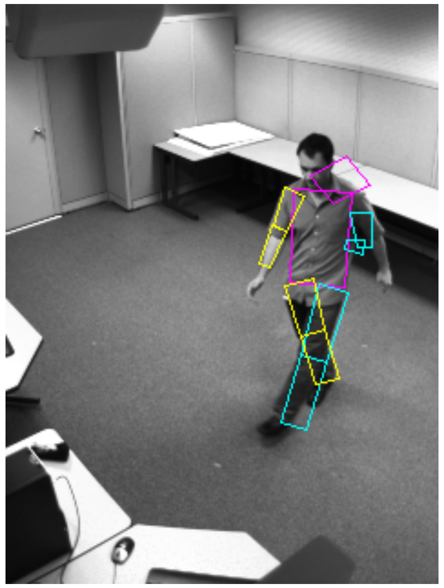
App	Synergy	Weak Antagonism	Antagonism
7zip	0.9%	60.4%	38.7%
Ferret	9.2%	48.8%	42.0%
Bodytrak	35.3%	40.1%	24.6%
OMXPlayer	2.6%	48.7%	48.7%
Average	**12.0%**	**49.5%**	**38.5%**

RQ3 CONCLUSIONS

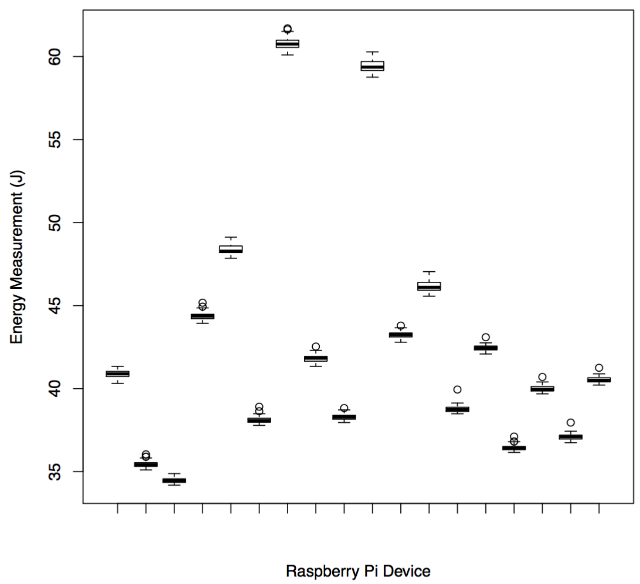
- The majority of modifications are worth combining
- However, a significant minority exhibit antagonism, meaning they shouldn't be combined
- This means a greedy approach is unlikely to produce an optimal solution; more advanced search is required

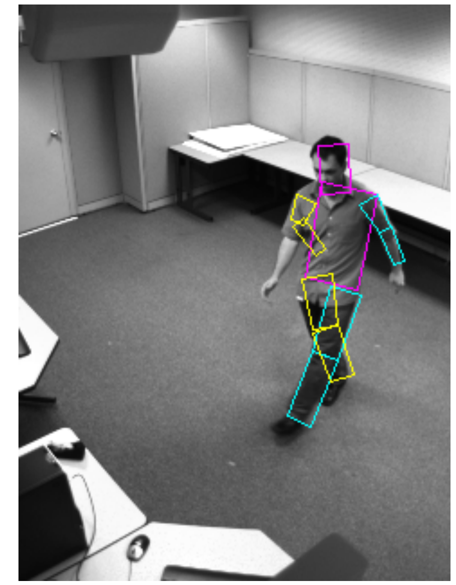


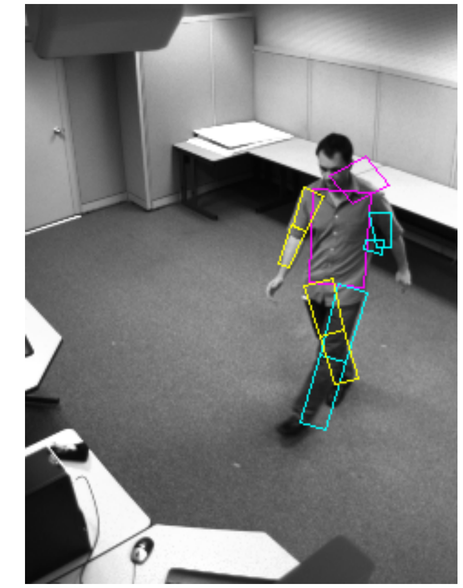




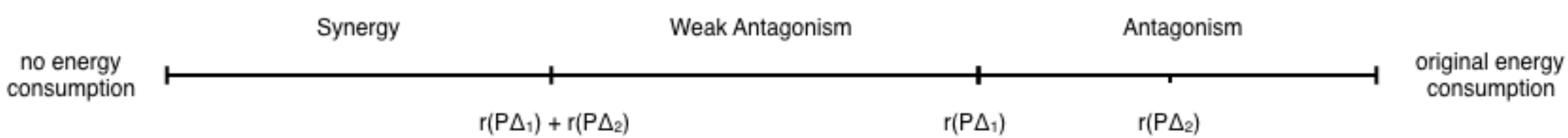
ANY QUESTIONS?







ANY QUESTIONS?



Bobby R. Bruce