TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik Institut für Software- und Multimediatechnik, Lehrstuhl für Softwaretechnologie

From Parameter Tuning to Dynamic Heuristic Selection

Yevhenii Semendiak

Yevhenii.Semendiak@tu-dresden.de
Born on: 7th February 1995 in Izyaslav
Course: Distributed Systems Engineering
Matriculation number: 4733680
Matriculation year: 2020

Master Thesis

to achieve the academic degree

Master of Science (M.Sc.)

Supervisors
MSc. Dmytro Pukhkaiev
Dr. Sebastian Götz
Supervising professor
Prof. Dr. rer. nat habil. Uwe Aßmann

Submitted on: 25th February 2020

Technische Universität Dresden/Fakultät Informatik

Aufgabenstellung für die Masterarbeit

Name, Vorname: Semendiak, Yevhenii

Studiengang: Master DSE

Matr. Nr.: 4 7 3 3 6 8 0

Thema:

From Parameter Tuning to Dynamic Heuristic Selection

Zielstellung :

Metaheuristic-based solvers are widely used in solving combinatorial optimization problems. A choice of an underlying metaheuristic is crucial to achieve high quality of the solution and performance. A combination of several metaheuristics in a single hybrid heuristic proved to be a successful design decision. State-of-the-art hybridization approaches consider it as a design time problem, whilst leaving a choice of an optimal heuristics combination and its parameter settings to parameter tuning approaches. The goal of this thesis is to extend a software product line for parameter tuning with dynamic heuristic selection; thus, allowing to adapt heuristics at runtime. The research objective is to investigate whether dynamic selection of an optimization heuristic can positively effect performance and scalability of a metaheuristic-based solver.

For this thesis, the following tasks have to be fulfilled:

- Literature analysis covering closely related work.
- Development of a strategy for online heuristic selection.
- Implementation of the developed strategy.
- Evaluation of the developed approach based on a synthetic benchmark.
- (Optional) Evaluation of the developed approach with a problem of software variant selection and hardware resource allocation.

Betreuer: M.Sc. Dmytro Pukhkaiev, Dr.-Ing. Sebastian Götz

Verantwortlicher Hochschullehrer: Prof. Dr. rer. nat. habil. Uwe Aßmann

Institut: Software- und Multimediatechnik

Beginn am : 01.10.2019

Einzureichen am : 09.03.2020

[illegible]

Unterschrift des verantwortlichen Hochschullehrers

Verteiler: 1 x SCIS, 1x HSL, 1x Betreuer, 1x Student

Contents

List of Figures

List of Tables

0.1 abstract

Abstract will be available in final versions of thesis.

1 Introduction

Intent and content of chapter. This chapter is an self-descriptive, shorten version of thesis.

1.1 Motivation

Structure:

- optimization problem(OP) → exact or approximate (+description to both) → motivation to use **approximate solvers** →
- impact of parameters, their tuning on solvers → motivation of **parameter control** (for on-line solver) →
- but what if we want to solve a class of problems (CoP) → algorithms performance is different →
- user could not determine it [7] → exploration-exploitation balance
- no-free-lunch (NFL) theorem [13] → motivation of the thesis

thesis motivation The most related research field is Hyper-heuristics optimizations [2], that are designed to intelligently choose the right low lewel heuristics (LLH) while solving the problem. But the weak side of hyper-heuristics is the luck of parameter tuning of those LLHs [links]. In the other hand, meta-heuristics often utilize parameter control approaches [links], but they do not select among underlying LLHs. The goal of this thesis is to get the best of both worlds - algorithm selection from the hyper-heuristics and parameter control from the meta-heuristics.

1.2 Research objective

Yevhenii: Rename: Problem definition?

The following steps should be completed in order to reach the desired goal:

Analysis of existing studies of algorithm selection. *(find a problem definition, maybe this will do [7])*

Analysis of existing studies in field of parameter control and algorithm configuration problems *(find a problem definition)* [8]

Formulation and development of combined approach for LLH selection and parameter control.

Evaluation of the developed approach with

Yevhenii: family of problems??? since it is a HH, maybe we should think about it...

.

Research Questions At this point we define a Research Questions (RQ) of the Master thesis.

- RQ 1 Is it possible to select an algorithm and it hyper-parameters while solving an optimization problem *on-line*?
- RQ 2 What is the gain of selecting and tuning algorithm while solving an optimization problem?
- RQ 3? How to solve the problem of algorithm selection and configuration simultaneously?

1.3 Solution overview

Yevhenii: Rename: Problem solution?

- described problems solved by HH, highlight problems of existing HHs(off-line, solving a set of homogeneous problems in parallel)
- create / find portfolio of MHs (Low level Heuristics)
- define a search space as combination of LLH and their hyper-parameters (highlight as a contribution)
- solve a problem on-line selecting LLH and tuning hyper-parameters on the fly. (highlight as a contribution? need to analyze it.)

Thesis structure The description of this thesis is organized as follows. First, in chapter 2 we refresh readers background knowledge in the field of problem solving and heuristics. In this chapter we also define the scope of thesis. Afterwards, in chapter 2 we describe the related work and existing systems in defined scope. In Chapter 4 one will find the concept description of dynamic heuristics selection. Chapter 5 contains more detailed information about approach implementation and embedding it to BRISE. The evaluation results and analysis could be found in Chapter 6. Finally, Chapter 7 concludes the thesis and Chapter 8 describe the future work.

2 Background and related work analysis

The structure is the same as the beginning of introduction, but way more detailed.

2.1 Optimization problems and solvers

2.1.1 Definition of optimization problems

Yevhenii: Need to find a classification of optimization problems.

2.1.2 Optimization problems solvers

Yevhenii: rename to smth like "Types | Classes of solvers"

Some (but not only) literature: [1]

Exact solvers

Approximate solvers

Comparison:

Pros and cons of both [6]

2.2 Approximate solvers of optimization problems

TSP as the running example. I guess, I will introduce it as an example of perturbation problems in previous section.??

2.2.1 Heuristics

Definition

Examples

Conclusion

Heuristics are strictly problem dependent and each time require adaptations.

2.2.2 Meta-heuristics

Definition

Classification

Examples

We distinguish following examples among all existing meta-heuristics, since later we use them as the LLH in developed hyper-heuristic.

GA

SA

ES

No-free-lunch theorem

NFL is the problem of heuristics[13]

Exploration-explotation balance

Conclusion

Proper assignment of hyper-parameters has great impact on exploration-exploitation balance and those on (meta) -heuristic performance.

2.2.3 Hybrid-heuristics

Definition

Examples

Guided Loca Search (GLS) + Fast Local Search [12]

Direct Global + Local search [11]

Simulated Annealing + Local Search [10]

Conclusion

2.2.4 Hyper-heuristics

Definition

Classification

Search space: heuristic selection, heuristic generation

Learning time: on-line learning hyper-heuristics, off-line learning hyper-heuristics, no-learning hyper-heuristics

Other classification characteristics from [7], [3], mb smth else. For instance, hyper-parameter tuning

Examples

[4] (Online algorithm selection at page 27); [7]

Conclusion...?

they usually (need to check it) have lack of parameter control

2.2.5 Conclusion on approximate solvers

Pros and cons of heuristics - too problem dependent

Pros and cons of meta-heuristics - no LLH selection, strict to one problem

Pros and cons of hybrid-heuristics - no LLH selection, strict to one problem ?

Pros and cons of hyper-heuristics - no parameter control?

2.3 Parameter tuning

Yevhenii: The goal of section: analysis of existing systems for hyper-parameter optimization (tuning), weaknesses and strength of each of the system

Yevhenii: Should I include the analysis from the code-basis point of view? If no, I do not see what should I conclude from this section exect "there are numer of parameter tuning systems each of them has pros and cons..

2.3.1 Parameter tuning problem definition, approaches (grid, random, mh..)

2.3.2 Systems for parameter tuning

IRACE

approach [9]

pros and cons

SMAC

approach description

BOHB

approach description

AUTO-SKLEARN

CASH (Combined Algorithm Selection and Hyperparameter optimization) problem

pros and cons (on-line or off-line, problems to solve, extensibility) [5]

BRISEv2

approach description

Yevhenii: Other systems?

Table for comparison

Conclusion

depends on the resolved question in todo...

2.4 Parameter control

2.4.1 definition, approaches

2.4.2 system examples

2.4.3 Conclusion

impact of parameter control based on other's evaluation

2.5 Conclusion

The meta-heuristic systems designers reported positive impact of parameter control embedding. However, as the outcome of the no-free-lunch theorem, those systems can not tolerate broad range of problems, for instance, problem classes. In other hand, hyper-heuristics are designed with an aim to select the low level heuristics and those propose a possible solution of problem, stated in no-free-lunch theorem, but the lack of parameter control could dramatically decrease the performance of LLH (probably, I need to find a prove of this, or rephrase).

Scope of work defined. In this thesis we try to achieve the best of both worlds applying the best fitting LLH and tuning it's parameters while solving the problem on-line.

3 Concept description

In this chapter we describe the concept of developed selection Hyper-Heuristic with parameter control, not diving deep into the implementation details. The best practice in software engineering is to minimize an effort for the implementation and reuse already existing, well-tested and broadly used code. With this idea in mind we had decided to use one of previously created (and highlighted by us in 2.3) hyper-parameter tuning systems as the code basis and those turn it into the core of hyper-heuristic. We also reuse the set of already developed heuristics as the Low Level Heuristics for designed Hyper-Heuristic.

The structure of this chapter is as follows. First, we define the Search Space entity in 3.1, it requirements and structure. It should bound the world of Low Level Heuristics and the world of Hyper-parameters of those heuristics.

Second, we describe the Prediction process within the previously defined Search Space in 3.2. Here we highlight an importance of a prediction model decoupling from the previously defined Search Space structure. Doing so, we provide certain level of flexibility for user in the usage of different prediction models or developing his own.

Third, in 3.3 we gather our attention onto the Low Level Heuristics - a working horse of the hyper-heuristic. Here we highlight the requirements for LLH in terms of features that will be used by HH.

Later, we select the code basis. We analyze the existing systems and highlight important non-functional characteristics for **??** hyper-heuristic and 3.4.2 low level heuristics set.

Finally, in 3.5 we conclude this chapter with the analysis of required changes, that are needed to be accomplished for turning code base system into the hyper-heuristic.

3.1 Search Space

Importance explanation

Required structure feature-tree structured

3.2 Prediction process

Importance explanation

Requirements generality, top-down approach of optimization – different views of same Configuration (level-dependent) - filtering, transformation – consider problem features? while selecting meta-heuristic [7] page 6

3.3 Low Level Heuristics

Importance explanation

Requirements

3.4 Code basis selection

With the aim of effort reuse, the code base should be selected for implementation of the designed hyper-heuristic approach.

3.4.1 Hyper-Heuristics Code Base

A.k.a. "brain". Need to find a better way to call this part of HH..

Requirements

Parameter tuning frameworks

SMAC

BOHB

IRACE

BRISEv2

Yevhenii: Maybe, smth else..

Conclusion

BRISEv2 is the best system for code basis.

3.4.2 Low Level Heuristics

Requirements

Heuristic frameworks

-table and short comparison of checked repositories

SOLID

MLRose

OR-tools

pyTSP

LocalSolver

jMetalPy

Conclusion

jMetalPy is cool!

3.5 Scope of Required Changes

3.5.1 Search Space

highlight - should be done in feature-tree structured search space

Current state description

What is the problem with the current Search Space?

Scope of work analysis:

throw away and write a new one :D

3.5.2 Prediction Process

highlight - should be done in feature-tree structured search space

Description of current state

Heterogeneous data?

short description of data preprocessing

Scope of work analysis

3.5.3 Low Level Heuristics

Description of current state

Scope of work analysis

3.6 Conclusion of concept

we selected BRISEv2, jMetalPy because they are cool. the amount of work is vast so lets dive into it in the next chapter!

4 Implementation details

Yevhenii: I am more concern to put it into the appendix..

In this chapter we define the implementation details of the selection hyper-heuristic with parameter control. You could find here a code snippets as well as the class diagrams.

4.1 Search Space

4.1.1 Description of base version (in BRISEv2)

4.1.2 Implementation

Description

Motivation of structure

Class diagram

4.2 Data preprocessing

Unified data types (pandas Data Frame)

Sklearn preprocessor wrapper

4.3 Prediction logic

4.3.1 Predictor

to decouple prediction from structure of search space.

4.3.2 Prediction models

Tree parzen estimator

Multi Armed Bandit

Sklearn linear regression wrapper

4.4 Low Level Heuristics

4.4.1 Meta-heuristics repository

Available Meta-heuristics

opened PR

5 Evaluation

5.1 Evaluation Plan Description

5.1.1 System Settings

To evaluate the performance of developed system we first need to compare it with the base line. In our case it is the simple meta-heuristic that is solving the problem with static hyper-parameters.

In order to organize the evaluation plan, we distinguish two stages of setup, where different approaches could be applied. At the first stage we select Low Level Heuristic, while at the second one we select hyper-parameters for LLH. The approaches for each step are represented in table 5.1.

Table 5.1 System settings for benchmark

Low Level Heuristics selection	LLH Hyper-parameters selection
1. Random	1. Default
2. Multi Armed Bandit	2. Tuned beforehand
3. Sklearn Bayesian Optimization	3. Random
4. Static selection of SA, GA, ES	4. Tree Parzen Estimator
	5. Sklearn Bayesian Optimization

For instance, mentioned above baseline could be described as *Settings*4.1. for meta-heuristics with default hyper-parameters and as *Settings*4.2. for meta-heuristics with tuned beforehand hyper-parameters.

For our benchmark we selected following settings sets:

- *Baseline:* 4.1, 4.2;
- *Random Hyper-heuristic:* 1.1, 1.2, 1.3, 4.3;
- *Parameter control:* 4.4, 4.5;
- *Selection Hyper-Heuristic:* 2.1, 3.1, 2.2, 3.2;
- *Selection Hyper-Heuristic with Parameter Control:* 2.4, 2.5, 3.4, 3.5;

Each of this settings will be discussed in details in following section.

5.1.2 Settings description

Baseline

Hyper-heuristic With Random Switching of Low Level Heuristics

Parameter control

Selection Only Hyper-Heuristic

Selection Hyper-Heuristic with Parameter Control

5.1.3 Optimization Problems Definition

TSP

tsplib95 benchmark set   which problem I want to solve with hyper-heuristic

n-Queens?

knapsack?

5.2 Results Discussion

Try numbers, discussion in the following section.

5.2.1 Baseline evaluation

Meta-heuristics with default hyper-parameters

Meta-heuristics with tuned hyper-parameters

Results Description and Explanation

5.2.2 Hyper-heuristic With Random Switching of Low Level Heuristics

Results Description and Explanation

5.2.3 Parameter control

Results Description and Explanation

5.2.4 Selection Only Hyper-Heuristic

Results Description and Explanation

5.2.5 Selection Hyper-Heuristic with Parameter Control

Results Description and Explanation

5.3 Conclusion of evaluation

6 Conclusion

Reviewer: answer research questions

7 Future work

add more sophisticated models

dependencies / constraints in search space

add new class of problem (jmetalpy easly allows it)

evaluation on different types and classes

Reviewer: consider merging with conclusion, if too short

Bibliography

[1] James S Bergstra et al. "Algorithms for hyper-parameter optimization". In: *Advances in neural information processing systems*. 2011, pp. 2546–2554.

[2] Edmund Burke et al. "Hyper-heuristics: An emerging direction in modern search technology". In: *Handbook of metaheuristics*. Springer, 2003, pp. 457–474.

[3] Edmund K Burke et al. "A classification of hyper-heuristic approaches: revisited". In: *Handbook of Metaheuristics*. Springer, 2019, pp. 453–477.

[4] John H Drake et al. "Recent advances in selection hyper-heuristics". In: *European Journal of Operational Research* (2019).

[5] Matthias Feurer et al. "Efficient and robust automated machine learning". In: *Advances in neural information processing systems*. 2015, pp. 2962–2970.

[6] Juraj Hromkovič. *Algorithmics for hard problems: introduction to combinatorial optimization, randomization, approximation, and heuristics*. Springer Science & Business Media, 2013.

[7] Pascal Kerschke et al. "Automated algorithm selection: Survey and perspectives". In: *Evolutionary computation* 27.1 (2019), pp. 3–45.

[8] Niklas Lavesson and Paul Davidsson. "Quantifying the impact of learning algorithm parameter tuning". In: *AAAI*. Vol. 6. 2006, pp. 395–400.

[9] Manuel López-Ibáñez et al. "The irace package: Iterated racing for automatic algorithm configuration". In: *Operations Research Perspectives* 3 (2016), pp. 43–58.

[10] Olivier C Martin and Steve W Otto. "Combining simulated annealing with local search heuristics". In: *Annals of Operations Research* 63.1 (1996), pp. 57–75.

[11] Michael Syrjakow and Helena Szczerbicka. "Efficient parameter optimization based on combination of direct global and local search methods". In: *Evolutionary Algorithms*. Springer. 1999, pp. 227–249.

[12] Edward Tsang and Chris Voudouris. "Fast local search and guided local search and their application to British Telecom's workforce scheduling problem". In: *Operations Research Letters* 20.3 (1997), pp. 119–127.

[13] David H Wolpert and William G Macready. "No free lunch theorems for optimization". In: *IEEE transactions on evolutionary computation* 1.1 (1997), pp. 67–82.

Statement of authorship

I hereby certify that I have authored this Master Thesis entitled *From Parameter Tuning to Dynamic Heuristic Selection* independently and without undue assistance from third parties. No other than the resources and references indicated in this thesis have been used. I have marked both literal and accordingly adopted quotations as such. There were no additional persons involved in the intellectual preparation of the present thesis. I am aware that violations of this declaration may lead to subsequent withdrawal of the degree.

Dresden, 25th February 2020

Yevhenii Semendiak