Journal of Heuristics
https://doi.org/10.1007/s10732-022-09501-8

A systematic approach to parameter optimization and its application to flight schedule simulation software

Alexander E. I. Brownlee[1] · Michael G. Epitropakis[2] · Jeroen Mulder[3] · Marc Paelinck[3] · Edmund K. Burke[4]

Received: 2 February 2021 / Revised: 9 March 2022 / Accepted: 17 June 2022

Abstract

Industrial software often has many parameters that critically impact performance. Frequently, these are left in a sub-optimal configuration for a given application because searching over possible configurations is costly and, except for developer instinct, the relationships between parameters and performance are often unclear and complex. While there have been significant advances in automated parameter tuning approaches recently, they are typically black-box. The high-quality solutions produced are returned to the user without explanation. The nature of optimisation means that, often, these solutions are far outside the well-established settings for the software, making it difficult to accept and use them. To address the above issue, a systematic approach to software parameter optimization is presented. Several well-established techniques are followed in sequence, each underpinning the next, with rigorous analysis of the search space. This allows the results to be explainable to both end users and developers, improving confidence in the optimal solutions, particularly where they are counter-

✉ Alexander E. I. Brownlee
alexander.brownlee@stir.ac.uk

Michael G. Epitropakis
m.epitropakis@gmail.com

Jeroen Mulder
Jeroen.Mulder@klm.com

Marc Paelinck
Marc.Paelinck@klm.com

Edmund K. Burke
edmund.burke@leicester.ac.uk

1 Computing Science and Mathematics, University of Stirling, Stirling, UK

2 The Signal Group, Athens, Greece

3 Air France KLM Group; Mulder with Technology Innovation inside Corporate Information Office, and Paelinck with Operations Research, IT, Amstelveen, Netherlands

4 University of Leicester, Leicester, UK

Published online: 05 July 2022

intuitive. The process comprises statistical analysis of the parameters; single-objective optimization for each target objective; functional ANOVA to explain trends and inter-parameter interactions; and a multi-objective optimization seeded with the results from the single-objective stage. A case study demonstrates application to business-critical software developed by the international airline Air France-KLM for measuring flight schedule robustness. A configuration is found with a run-time of 80% that of the tried-and-tested configuration, with no loss in predictive accuracy. The configuration is supplemented with detailed analysis explaining the importance of each parameter, how they interact with each other, how they influence run-time and accuracy, and how the final configuration was reached. In particular, this explains why the configuration included some parameter settings that were outwith the usually recommended range, greatly increasing developer confidence and encouraging adoption of the new configuration.

Keywords Parameter tuning · Optimization · Statistical methods · Multi-objective optimization · Search-based software engineering · Explanation

1 Introduction

Many industrial software applications are developed to a tight time budget. In a commercial setting there is frequently a lack of developer time to fine tune application parameters, whereby "good enough", no matter how sub-optimal, is tolerated. "Programming by optimization" (Hoos 2012), whereby decisions about appropriate parameter settings are not made by the developer but instead exposed for later tuning to a particular application by a metaheuristic or other search-based solver, represent a great opportunity to resolve this situation. However, essentially black-box optimisation methods such as metaheuristics are poorly understood by practitioners, with multiple surveys (Hornby and Yu 2007; Tiwari et al. 2015; Vincalek et al. 2021) indicating that this brings a lack of trust in the results and a reluctance to interfere with parameters that have often been fine-tuned by hand and left with "do not touch" warnings in place. Yet where a piece of software is critical to a business (e.g. our case study), substantial impact can result from such fine tuning, if the tuned results are put into practice. Enabling machines to explain their decisions is crucial to maximising their real-world utility (Le Bras et al. 2018), so to realise the potential gains of software parameter optimisation, approaches are needed to gain insight into the optimisation problem and processes that solved it. For optimisation problems, users need confidence that the solution solves the problem rather than exploiting an error or loophole in the problem's definition: examples of such loopholes from the broader metaheuristics community include a simulated robot that maximised its walking speed by simply growing tall and falling over, or the tic-tac-toe AI that triggered a memory overflow in its opponent (Lehman et al. 2020). Users also need to know whether unexpected values found by the optimisation are just the result of the random processes in the metaheuristic, or whether they are related to a more fundamental property of the software being tuned. This latter point is particularly relevant in a real-world application where a single solution from a single algorithm run is typically desired to be put into

practice, rather than the multiple repeat runs over multiple algorithm configurations that are the norm in academic research.

We propose a simple, systematic, approach to parameter tuning that allows developers to focus on the software, yet still achieving optimal parameter settings. Our approach incorporates statistical testing and analysis reminiscent of the current trend in *Explainable AI*, whereby analysis of each parameter's importance and sensitivity is also presented to the developer to add confidence to the results. If the approach reveals that the optimisation found a variable to be unimportant, then apparently "strange" values that are the result of the random processes inherent to metaheuristics can be ignored. If a variable is shown to be unimportant, but *is* regarded as important by the decision maker (or *vice versa*), this indicates a potential issue with the fitness definition needing closer inspection, to ensure that the variable is correctly interpreted by the target application, and to avoid the algorithm exploiting a loophole. If a variable has been tuned outside of its expected range, knowing something of its relationship to be the optimisation objective and the other variables will provide some explanation for why the selected value was chosen.

Explanation of this nature is deeply rooted in the application; so the main focus of this paper is a worked example following a real world case study. In this case study we focus on *Opium*, an application developed in-house by the Operational Research department of the international airline KLM, which carries over 35M passengers per year to 171 destinations using 214 aircraft. Opium was designed to measure the robustness of a global flight schedule, and was in use for a further two years after the study we report in this paper.

Revenue and performance of a flight schedule is highly dependent on its robustness. A lack of robustness in a schedule can seriously affect airline operations, where a delay to one flight caused by weather, breakdown or staff availability escalates to impact on many other flights. Consequently, accurate estimations of a schedule's robustness before that schedule is brought into operation are crucial to the business. Frequent regular evaluation (daily, weekly) enables improvements to the schedule, leading to more reliable service for customers and increased revenue. Thus our goal was to improve Opium so that it can assess schedule robustness in a shorter time, while maintaining or exceeding the original version's accuracy. Simultaneously, the proposed methodology takes into account the important factor of justifying the trade-offs between the different parameters to facilitate understanding and improved business decision making.

This paper's contributions lie in the overall process, and the real-world case study. Though none of the individual stages are themselves new, the proposed process is designed to shed much greater light on the optimisation than any one step by itself, in the spirit of *explaining* the results. The systematic approach to parameter tuning is also shown to realise real-world benefits. The case-study application runs more quickly, allowing more alternative schedules to be tried, and runs more accurately, reducing real-world delays for the millions of customers that complete their journeys with KLM each year. The tuned configurations are justified, or explained, by statistical analysis and search-based exploration of the space, improving trust in the results, motivating their take-up in practice, and providing valuable feedback to the developers of the case-study application. The approach should be applicable to the configuration of

other simulation-based applications, where we anticipate that similar benefits might be realised.

The remainder of the paper is organized as follows. Sect. 2 introduces related work in parameter tuning and search-based software engineering. The software targeted in our study is described in Sect. 3.1, and the optimization methodology is described in Sect. 3. We then move on to demonstrate the process through a series of experiments in Sect. 4, before drawing our conclusions in Sect. 5.

2 Related work

The present study centres around improving an existing piece of software by tuning its parameters. This fits into the much broader field of search-based software engineering (Harman et al. 2012), whereby search-based optimisation methods are used to tackle problems in software. Our systematic approach is also designed to offer explanations to improve trust in the results. We now briefly summarise relevant related work on the topics of parameter tuning and trust in optimisation results.

2.1 Automated parameter tuning and software improvement

The importance of parameter tuning is well-known within the machine learning (Yu and Zhu 2020) and metaheuristics (Stützle and López-Ibáñez 2019) communities, though its use in more general software applications (like the simulation tool we focus on) are less common. Several years ago, Hoos (2012) advocated the concept of *programming by optimization*, whereby developers commit to as few decisions as possible. Instead, a program's parameters are all exposed for later search and optimization, greatly increasing the applicability of a piece of code to different applications.

Automated parameter tuning can be broadly divided into *offline tuning*, where we tune then fix the parameters and run the algorithm (Eiben and Smit 2011; Hutter 2009; Birattari and Kacprzyk 2009; Stützle and López-Ibáñez 2019) and *online parameter control* (Karafotias et al. 2015; Alabas-Uslu and Dengiz 2020) where the parameters are tuned during the run. The framework proposed in the present paper focuses on offline tuning.

Several tools have now been developed to aid this process: among the most popular are iRace by López-Ibáñez et al. (2016), SMAC by Hutter et al. (2011) and ParamILS by Hutter et al. (2009). These only require the end-user to implement a wrapper for their program that takes a particular algorithm configuration and returns a number representing some concept of fitness or performance.

López-Ibáñez et al. (2016) proposed *iRace*, which implements iterated racing for algorithm configuration and has been shown to be effective in parameter tuning for many different applications. Iterated racing involves sampling probabilistic distributions to choose algorithm parameters, then repeatedly running the configurations on different problem instances, removing configurations that perform statistically worse than at least one other. The distributions are updated periodically with a bias towards

the surviving configurations, until the algorithm converges on a good configuration or some other termination criterion is met.

Sequential model-based algorithm configuration (SMAC), developed by Hutter et al. (2011), follows a Bayesian optimisation process, using models based on random forests to estimate likely good configurations. These are refined as more configurations are tested, until no further improvement can be found or a cap on computational time is reached.

ParamILS (Hutter et al. 2009) uses an iterated local search algorithm to find good configurations. ILS simply makes random changes to a base configuration, and keeps changes that offer an improvement. If a predefined number of evaluations are made without improvement, a larger change ("kick") is applied. A disadvantage is that it works only on categorical parameters and, hence, requires discretising numerical ones.

Several algorithmic approaches have also been developed beyond the publicly available tools above. Bennett et al. (2008) used a bilevel approach for tuning machine learning algorithms, where the upper level seeks to minimise the validation set loss, with respect to some hyperparameters, and the lower level minimises the training set loss, with respect to the model parameters. This has recently also been shown by Sinha et al. (2020) to work for deep learning. Sinha et al. (2014) formulated the tuning of optimisation algorithms themselves as a bilevel problem, where the low level was the optimisation algorithm being targeted and the upper level was the parameter tuning stage. To improve runtimes, performance of the lower level algorithm was sometimes approximated using a quadratic regression model. They demonstrated the approach using a stochastic algorithm in the upper level for tuning differential evolution and Nelder-Mead algorithms at the lower level. Liang and Miikkulainen (2015) took a similar approach for tuning the parameters of neurovolution, albeit using a random forest model to approximate performance at the lower level. Recently, Mejía-de Dios et al. (2021) has formulated the parameter tuning problem as a bilevel optimisation, whereby one level of the optimisation selects the hardest instances on which the tuning focuses, in order to improve the efficiency of the optimisation process at the other level.

Also related, transforming the static configuration parameters of an application into variables that can be tuned dynamically has also resulted in programs that consume substantially less energy with little reduction in functional performance (Hoffmann et al. 2011). The concepts of parameter tuning and programming by optimization have also been taken further in the form of *deep parameter tuning*, whereby hard-coded constants within code are exposed to an external tool for tuning with respect to various objectives (Wu et al. 2015; Bruce et al. 2016; Sohn et al. 2016).

2.2 Multi-objective parameter tuning

A number of publications have discussed the concept of parameter tuning with respect to multiple objectives. These are well covered in a review by Bezerra et al. (2020), who also explore approaches for tuning of multi-objective optimisers. The potential of automatically tuning algorithms for multiple objectives was raised by Dréo (2009), who considered the trade-off of speed vs accuracy (in terms of the solution qual-

ity reached by the metaheuristic being tuned), when setting a single parameter of a metaheuristic using NSGA-II and evaluating each configuration a number of times. Smit et al. (2010) considered a similar approach, using a multi-objective EA, but this time exploring trade-offs in performance of a single-objective GA on different target problems. Further motivation for multi-objective configuration of optimisers was also given by Dang and De Causmaecker (2014), who also considered the question of trade-off between algorithm speed and solution quality, extended this idea by including historical information in the fitness calculation to allow for tuning algorithms for different evaluation budgets, and by adding noise handling techniques to cope with the stochastic nature of the metaheuristics being tuned.

Zhang et al. (2013) proposed *S-Race*, an extension of I-Race, to tackle multi-objective configuration tasks by extending racing procedures to the multi-objective context, using non-parametric sign test to identify pair-wise dominance relationships between models.

A multi-objective extension of ParamILS (MO- ParamILS) was proposed by Blot et al. (2016). The main addition over ParamILS is the use of an archive of configurations to approximate the Pareto front, using dominance (i.e., solutions that outperform others in all the objectives), as a measure of quality.

The key difference with our approach from approaches using a single multi-objective algorithm (whether metaheuristics or a Bayesian-optimistion approach) for the whole optimisation is in seeking to underpin the optimisation results with explanation. Our framework is designed to provide statistical evidence and exploratory analysis of the importance of variables and their interactions to justify the optimal results produced. These vary per objective, and change when multiple objectives are considered together (Brownlee et al. 2020), even for white-box benchmark functions where the relationships between variables (separability) for each objective is well understood (Li et al. 2016). This motivates the multi-stage approach where we consider the objectives in isolation prior to the multi-objective search, but this approach also brings the benefit of putting maximal effort into approximating the bounds for the objectives before finding the trade-off between them.

2.3 Trust in optimisation results

The machine learning community has seen *Explainable AI* become a hot topic in recent years. Many advances have been made towards explaining the decisions of black-box systems like deep neural networks (Hendricks et al. 2016; Ribeiro et al. 2016), with the goal of increasing people's trust in the decisions of these techniques. Explainability also highlights where bias or poor formulation of the problem have led to errors.

"Explanations" generally mean reasons or justifications for particular outcomes, rather than a description of the inner workings or the logic of reasoning behind the decision-making process in general (Adadi and Berrada 2018). Similar questions have been raised around search-based optimisation approaches, where justification for the results is also important for making the solutions acceptable (Urquhart et al. 2019). Standard techniques for explaining or justifying the solutions of search-based optimisation remain rare. "Innovization", proposed by Deb and Srinivasan (2006), targets

design principles common to Pareto-optimal solutions in multi-objective optimisation, and the final stage of our framework bears some resemblance to this. Some ad-hoc approaches to mining surrogate fitness models for sensitivity of variables and key characteristics of good solutions have been proposed by Brownlee et al. (2013); Brownlee (2016); similarly, mining of probabilistic models in estimation of distributions algorithms was proposed by Santana et al. (2009). Urquhart et al. (2019) proposed generating a diverse set of solutions and using an interactive decision making process to increase trust in the solutions of a metaheuristic for a transportation problem. Our framework is designed to complement and build on these approaches by taking a structured approach to gaining insight into the underpinning relationships between variables and the optimisation objectives for parameter tuning problems.

3 Parameter tuning as an optimization problem: Proposed methodology for real-world applications

We now present our proposed methodology for parameter tuning software applications. The overall goal here is to provide tuned configurations of the application, with supporting analysis and explanation of the results in the form of identification of the critical parameters and their relationship with the one or more objectives to be optimised. The motivation is to achieve both confidence in the results and vital feedback to developers for the application.

The steps are as follows:

1. **Statistical sensitivity analysis**. Using formal Design of Experiments to sample the search space and standard ANOVA testing with response surface methodology, this stage identifies the most sensitive variables at a global level, and locates high quality solutions for each objective. This provides a sanity-check grounded in statistics for the subsequent stages, and provides indications as to the suitability of the chosen parameters, their bounds and their granularity, with a low computational cost.
2. **Single-objective parameter tuning**. Using standard parameter tuning tools, this stage approximates the lower bounds achievable for each objective when ignoring all others.
3. **Functional ANOVA**. Using a model-based optimiser for the previous stage, we are able to exploit the knowledge embedded in the resulting models to bring further insight into the parameters and the relationships between them.
4. **Seeded multi-objective optimization**. Using a well-established multi-objective optimisation algorithm, configurations spanning the trade-off between the objectives are found, revealing any knee or tipping points that can be exploited. This optimisation is seeded with the results from the single-objective optimisation stage, giving it a head start and ensuring that any solutions found at least match the performance of the single-objective optima. The idea of seeding the MO search with SO optima, and analysis of the variables among the Pareto front, can be traced back to the *Innovization* of Deb and Srinivasan (2006). Here, we enhance this concept by also building on the insights gained from the statistical SA and fANOVA stages.

Trends in the values assigned to variables along the Pareto front are also identified to further explain the chosen configurations.

The individual steps in this framework are not in themselves novel, but the overall procedure offers a systematic approach that is useful for real-world applications and, crucially, explains the results. The idea is to take a reasoned approach to parameter tuning that offers:

- an optimal configuration;
- statistical evidence supporting the optimal configuration;
- insight into important parameters and how they interact that might be of use to the application developers;
- involvement of the decision maker in guiding the process to increase *trust* in the outcome.

We now consider the application that forms the focus of our study and specific implementation details before proceeding to our experimental results.

We focus on one application in the case study to allow us to concentrate on the individual analysis of each variable, and its relationship with the outcomes of an optimisation run. In principle, the approach will work for tuning any software with some measure of solution quality (simulation accuracy in this case), and a non-functional property such as run time. The application in our case study has a variety of variable types (boolean, integer, continuous, categorical/nominal). It also includes dependent variables (i.e. those that are only operational when particular values are taken by other variables): Opium's HSFTO, HSFTI, and HSFT parameters only apply when HSFO is TRUE. There is no reason why the techniques should not be applicable to other applications with discrete parameters of these types, but given the space required to discuss further applications in appropriate detail, we leave consideration of other applications for future work.

3.1 OPIUM: The software under consideration

Opium is a standalone application developed and maintained in-house by the Operational Research Department at KLM. It is implemented in Java, following a 'building block' model whereby different components of the simulation process are combined in stages. Much of Opium's code is highly-specific to the application, with the core simulation logic built on a freely available Java library DSOL developed at TU Delft by Jacobs (2005).

A single run of Opium estimates robustness for a schedule. Distributions for events such as aircraft breakdowns are specified in a set of data files. Opium stochastically samples these distributions over a number of repeat runs, computing resulting delays and knock-on effects from distributions, and computing an overall cost through a series of sanctions.

Schedules cover periods of three months; there are separate schedules for flights within Europe (EUR) and for those reaching international destinations (ICA). A typical schedule contains around 17 000 flights. All KLM flight schedules passed through Opium for testing prior to approval. Since the work in this paper was completed, Opium was superseded by new applications based on it.

Table 1 Opium's 14 configuration parameters, with the recommended ranges provided by the application's developers. Abbreviations (Abbrev.) are used in the rest of the paper

Parameter	Abbrev.	LB	UB	Type
Max Maintenance Reduction	MMR	0	0.2	Real
Ground Factor Out	GFO	1.0	1.3	Real
Slack Selection BB3	SSBB3	0	50	Integer
Max Legs Swap	MLS	2	6	Integer
HSF threshold Out	HSFTO	0	5	Integer
HSF threshold In	HSFTI	0	15	Integer
Max Legs Cancel	MLC	1	7	Integer
HSF threshold	HSFT	0	15	Integer
Cancel Measure On	CMO	0	1	Boolean
Break Maintenance Measure On	BMMO	0	1	Boolean
Create Gamma	CG	0	1	Boolean
Rounding off method	ROM	Regular, Down, Up, Math, None		Categorical
Swap Measure On	SMO	0	1	Boolean
HSF Measure On	HSFMO	0	1	Boolean

Opium has 14 control parameters, listed in Table 1, with the ranges originally specified by the application's developers. The majority of these are application specific: methods by which the simulation attempts to resolve problems caused by disruptions.

For the work in this paper, KLM were able to provide historical schedules to be used as training data. These comprise ten 3-month periods from 2007 to 2010, for both ICA and EUR.

3.2 Objective functions

In this study, we have utilized two objective functions that could have significant commercial impact to an airline company, the *accuracy* and the *execution time* of the simulation process. Intuitively, a decision maker requires the simulator to provide as accurate results as possible that fit the real historical operational data of the company. This will enable the airline firm to accurately estimate the robustness of the schedule under testing based on its "normal" historical operational behavior patterns. In addition, given that a large number of possible schedules have to be checked through the robustness simulator within a scheduling period, the execution time of the simulator has a major impact on the timely and efficient scheduling optimization phase of the firm. It is worth also noting that a very fast simulator could be integrated in the scheduling optimization process of the airline firm to simultaneously optimize operational costs, and robustness of schedules. In such a case, it is crucial to develop a simulator that is executed in the scale of seconds, in order to facilitate a timely schedule optimization phase.

The operational performance of the schedules is tested by Opium by repeatedly introducing possible disruptions such as delays and breakdowns to the planned sched-

ule. The impact of these on aircraft movements is then quantified in terms of the 0, 15, 30, and 60 minute on-time performance (denoted by OTP0, OTP15, OTP30 and OTP60), that is, the number of aircraft which have actually left on-time, or 15, 30 or 60 minutes after their scheduled departure time. Opium also quantifies the number of swaps, and the number of cancellations, required in order to make the schedule feasible given a disruption (*feasible* meaning that all aircraft and crews are in physically the right place at the right time to complete their scheduled movements).

The historically measured punctuality for each schedule is used as the basis for measuring the accuracy of the simulation. Punctuality is measured by 22 metrics for each of 6 aircraft types (i.e., 132 metrics in total), capturing statistics like number of aircraft delayed by 0, 5, or 15 minutes, broken down by whether they are inbound or outbound at KLM's hub airport, Amsterdam Schipol. We formulate the problem of fine-tuning the Opium software package as a min-max optimization problem that essentially minimizes the worst case (maximum) deviation from the real historical operational data **(accuracy)**, or minimises the program's run time **(time)**, or both.

Accuracy: find a parameter configuration $c \in \mathcal{C}$, where $\mathcal{C}$ is the configuration search space, that minimises the worst case deviation from reality (max mean square error — MSE)

$$\mathrm{Acc}(c) = \min \max_{c \in \mathcal{C}} \left(\sum_{m \in \mathcal{M}} (m_h - m_o(c))^2 \right), \tag{1}$$

where m_h and $m_o(c)$ are the historical and output metric values respectively, for each metric m in the set $\mathcal{M}$ of 132 schedule punctuality metrics, when running Opium with parameter configuration c.

In all of our experiments, the measured accuracy was the mean MSE over ten historical flight schedules with associated real-world punctuality statistics. MSE is chosen because the errors are squared before they are averaged, so MSE gives a relatively high weight to large errors, compared to other measures such as Mean Absolute Error (MAE). This makes MSE more useful when large errors are particularly undesirable. That applies here: KLM are keen to avoid a high error on any single scheduling period, because in practice Opium is used to test one period at a time.

Time: find a parameter configuration $c \in \mathcal{C}$, where $\mathcal{C}$ is the configuration search space, that minimises the wall-clock execution time of the simulator that is required to simulate a given scheduling scenario.

Execution time is a noisy quantity that has to be carefully measured, since the environment in which the execution time has to be measured can be affected by many factors, including the hardware, memory caching, configuration of the system, and other running applications. To test the impact of this, we performed 100 repeat runs of Opium in its default configuration, measuring the wall-clock run time of each. The distribution of these times, shown in Fig. 1, is approximately normal.

To mitigate the stochastic nature of the measurements, we define as the *execution time* of the simulator its average execution time over k independent executions. Specifically, we calculate the execution time of a given configuration c, $\mathrm{Time}(c)$, according to the following equation:

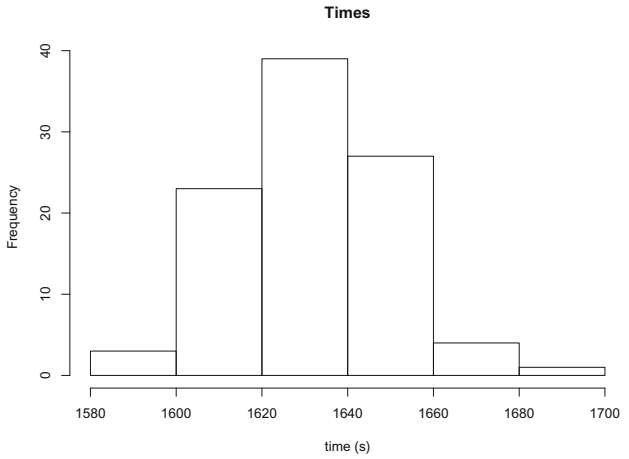


Fig. 1 Run times for 100 repeat runs of the default configuration of Opium, showing an approximately normal distribution

$$\text{Time}(c) = \frac{1}{k} \sum_{i=1}^{k} t(OPIUM(c)), \tag{2}$$

where $t(OPIUM(c))$ is the wall-clock time of an independent execution of the software under parameter optimization (Opium), which is configured with the given parameter configuration c. If the times did not follow a normal distribution, or it was not possible to confirm normality, then the median time could be used instead.

In all of our experiments, due to the long running times of Opium (around 15–20 minutes for all ten historical scheduling data sets), we used $k = 5$, except where we explicitly note the use of 30 repeats in runs to confirm the optimal solutions. However, as each of these repeats covered a run of the simulation on each of the ten historical data sets, in effect the time calculation covers fifty runs of the simulation.

4 Case study and experimental results

The proposed methodology has been evaluated on tuning the Opium tool using real world scheduling data from KLM Royal Dutch airlines. KLM provided us operating schedules that were run during the second part of summer 2007 and the first part of summer 2010. KLM operates a single hub-and-spoke network with dominant hub-banks. Its European fleet consists of 45 aircraft of 4 (interchangeable) sub-types. 10 different sub-schedules, having different fleet compositions and different sizes, were derived from the initial schedules.

4.1 Research questions

Our overall goal is to determine how the proposed framework answers the following research questions for our case study software:

- **RQ1:** Can we identify important variables / parameters, as a means of justifying the solutions that are eventually chosen?
- **RQ2:** Can we find "optimal" configurations that approximate reality better (fit to historical data)?
- **RQ3:** Can we minimise Opium's execution time?
- **RQ4:** Can we identify any trade-off between accuracy and execution time?

RQ1 and 2 specifically target whether relevant explanations are generated through the approach, while **RQ3 and 4** consider whether the approach still produces results of sufficient quality.

RQ1 is answered in stages 1, 3 and 4 of our framework (statistical sensitivity analysis, functional ANOVA and multi objective optimisation followed by Pareto front analysis); **RQ2 and RQ3** are answered in stage 2 (single objective optimisation); and **RQ4** is answered in stage 4 (multi objective optimisation).

4.2 Stage 1: Statistical parameter sensitivity

We begin with a structured approach to parameter tuning based on the procedure outlined by Petrovski et al. (2000, 2005) and examined in more detail by Czarn et al. (2004). This approach was implemented in version 17 of the Minitab statistical package (Minitab, Inc 2014). The procedure begins with a screening experiment: an exploration of the parameter search space using a fractional factorial (Plackett-Burman) Design of Experiments (DoE) procedure, considering low and high values for each parameter. The low value for each parameter was the minimum as specified by the company's developers. The high value was the value currently used for each parameter by the company in practice, which we refer to as the *default* values. Over years of manual tuning, the values in use had settled on the upper end of the ranges as anticipated by the program's developers.

There are 14 parameters: while a full factorial (Latin hypercube) exploration of this space would take 16 384 evaluations, the fractional factorial considers 48 evaluations, evenly distributed around the space. Each of these 48 runs were repeated 5 times for the ten historical data sets, and the mean running time of these was used for the time objective. This is followed by an ANalysis Of the VAriance (ANOVA) to determine the parameters that have a significant impact on the objective when changing from their lower bound to their upper bound. More precisely, the ANOVA tests for the null hypothesis that there is no difference between the responses for each objective at the different levels. We reject this hypothesis where p-value for the test is <0.05, where we consider the hyperparameter to have a statistically significant impact on the objective. The ANOVA operates the assumptions that the residuals are normally distributed and variance is homogeneous between samples; these were confirmed by plotting the residuals after each test. Although ANOVA is robust to non-normality (Schmider et al. 2010), in the case where the residuals are not normally distributed it may be preferable to switch to a non-parametric test such as Kruskal-Wallis. Four parameters with a significant impact were found for accuracy, and six were found for time (Table 2).

Table 2 Screening experiment. Parameters with a statistically significant influence are in bold

Parameter	LB	UB	P-value (accuracy)	P-value (time)
MMR	0	0.2	0.177	0.544
GFO	1.0	1.3	0.311	**0.000**
SSBB3	0	50	0.505	0.142
MLS	2	6	0.404	0.578
HSFTO	0	5	0.794	0.987
HSFTI	0	15	0.789	0.386
MLC	1	7	**0.018**	**0.021**
HSFT	0	15	0.625	0.215
CMO	0	1	**0.006**	**0.000**
BMMO	0	1	0.980	0.762
CG	0	1	**0.000**	**0.000**
ROM	Regular	None	0.514	**0.000**
SMO	0	1	**0.000**	**0.000**
HSFMO	0	1	0.714	0.961

Table 3 Exhaustive search parameters: accuracy

Parameter	LB	UB
MLC	1	14
HSFT	False	True
CG	False	True
SMO	False	True

Note also that the use of a fractional factorial design of experiments means that the experimental data is *orthogonal*. Depending on the *resolution* of the design (i.e., how close to a full factorial it is), this means that we can treat each of the main effects and 2-way interaction terms independently. The implication is that we do not need to control for family-wise error in the ANOVA with a technique like Bonferroni correction of the p-values (Field 2018). With the design used here, we can safely reject the null hypothesis where the p-value is <0.05.

The second stage explores the parameters with a significant impact more closely to determine their optimal values. For **accuracy**, these are all discrete, taking either binary or integer values as listed in Table 3. This means that an exhaustive search over all combinations is possible. The remaining parameters were held at their default values during this search.

The results of the exhaustive search are given in Fig. 2. Following the same visualisation procedure as Brownlee and Wright (2012); each row represents one solution, with the columns representing the parameters and the objective. The range for each column is normalised so the bars represent the full range of possible values for that column. Solutions are sorted by one of the objectives to show any relationships between the parameters and the objectives. A suitable alternative visualisation would be a par-

MLC	CMO	CG	SMO	Accuracy
1	1	1	1	271.6
2	1	1	1	271.6
3	1	1	1	271.6
4	1	1	1	271.6
5	1	1	1	271.6
6	1	1	1	271.6
7	1	1	1	271.6
8	1	1	1	271.6
9	1	1	1	271.6
10	1	1	1	271.6
11	1	1	1	271.6
12	1	1	1	271.6
13	1	1	1	271.6
14	1	1	1	271.6
1...14	0	1	1	271.6
2...14	1	0	1	292.7
1	1	0	1	306.9
1...14	0	0	1	306.9
2...14	1	1	0	366.2
2...14	1	0	0	453.3
1	1	1	0	564.0
1...14	0	1	0	564.0
1	1	0	0	646.9
1...14	0	0	0	646.9

Fig. 2 Exhaustive experiment on accuracy (mean square error, so lower accuracy values are better). The first four columns represent the parameters explored: MLC, CMO, CG, and SMO (refer to Table 1). The final column is the accuracy measured for each given configuration. MLC and CMO do not appear to have any substantial effect on accuracy; consequently the final few rows show the accuracy obtained for different CMO, CG and SMO values for all MLC values. 1…14 and 2…14 indicate where identical accuracy was obtained using each of the values in the range [1,14] and [2,14] respectively

allel coordinates (PC) plot, although the approach we have used has the advantage of making correlations between specific variables and the objectives easier to see, without having to follow the lines representing individual solutions as would be the case on a PC plot.

The results reveal that SMO has the greatest influence on accuracy. All configurations with SMO true yield better accuracy than those with SMO true. The second most important parameter is CG, which also yields better accuracy when true. The

Table 4 Response surface parameters: time. *Opt* are the optimal values found

Parameter	LB	UB	Levels	Opt
GFO	1	2.6	5	1.62858
MLC	1	14	5	14
CMO	0	1	2	1
CG	0	1	2	0
ROM	Regular, Down, Up, Math, None		5	Regular
SMO	0	1	2	0

results show that MLC and CMO have only a slight impact on accuracy. When CMO is false, MLC has no measurable impact. When CMO is true, all values for MLC yield the same accuracy, except 1, which is slightly poorer.

A confirmation experiment was run to check the best configuration from the search (top row in Fig. 2) against the default configuration. The best configuration found had a MSE of 271.628. This exactly matches the MSE for the default configuration 271.628.

For **running time** there are six parameters with a significant impact (Table 4), including two continuous parameters, ruling out an exhaustive search. Instead, a further DoE procedure was followed using a central composite design, with each parameter having a fixed number of levels indicated in Table 4. This design required 520 solution evaluations. A polynomial response surface was fitted to the results of these evaluations, having six linear terms, two squared terms, and 15 terms for pairs of parameters, with an R^2 of 0.8674. An ANOVA carried out on this model found 14 significant terms, listed in Table 5. Finally, Minitab's optimisation tool is used to find the optimal values for the six parameters, listed in Table 6. This tool finds the values obtained by differentiating the linear regression model with respect to each factor in turn, setting each derivative equal to zero and solving the resulting system of equations (Petrovski et al. 2005). As a confirmation, Opium was run with these parameters 30 times, taking a mean of 476.5s (std. dev. 4.9s) to complete each run with an accuracy (MSE) of 426.988. This compares to the corresponding value for the default parameters of 1406.7s (std. dev. 354.6s) and accuracy (MSE) of 271.628. These experiments clearly show that there is much potential for improvement in both accuracy and run time at the cost of the other, with the hyperparameters under consideration, and it is worth moving to the second stage of optimisation. The statistical significance tests will also be used to confirm the results produced in subsequent stages, identifying the important variables as part of the explanation of the results.

4.3 Stage 2: Single-objective parameter tuning

Having achieved an improvement in run time, and confident that improvement is possible for accuracy, we proceeded to automated algorithm configuration to attempt to find parameters that lead to higher accuracy. Here, we used iRace (López-Ibáñez

Table 5 Response surface parameter significance results: time

Source	P-Value
Linear	
GFO	**0**
MLC	**0**
CMO	**0**
CG	**0**
ROM	**0**
SMO	**0**
Square	
GFO*GFO	**0**
MLC*MLC	0.114
2-Way Interaction	
GFO*ROM	0.114
GFO*SMO	**0**
GFO*ROM	0.114
GFO*SMO	**0**
MLC*CMO	**0**
MLC*CG	0.125
MLC*ROM	1
MLC*SMO	0.197
CMO*CG	**0**
CMO*ROM	0.627
CMO*SMO	**0**
CG*ROM	**0**
CG*SMO	0.893
ROM*SMO	**0**

Table 6 Response Surface: optimal parameter values for time

Parameter	Value
GFO	1.62858
MLC	14
CMO	TRUE
CG	FALSE
ROM	REGULAR
SMO	FALSE

et al. 2016), as implemented in the R `irace` package[1] and SMAC (Hutter et al. 2011), from the reference Java implementation[2].

[1] Version 1.07, http://iridia.ulb.ac.be/irace

[2] Version 2.08, http://www.cs.ubc.ca/labs/beta/Projects/SMAC/

Table 7 Single-objective: parameter tuning

Parameter	LB	UB	Type
MMR	0	1	Real
GFO	1	2.6	Real
SSBB3	0	100	Integer
MLS	2	12	Integer
HSFTO	0	10	Integer
HSFTI	0	30	Integer
MaxLC	1	14	Integer
HSFT	0	30	Integer
CMO	0	1	Boolean
BMMO	0	1	Boolean
CG	0	1	Boolean
ROM	Regular, Down, Up, Math, None		Categorical
SMO	0	1	Boolean
HSFMO	0	1	Boolean

For runs with iRace and SMAC, the default configuration supplied by KLM was used as a seed. This ensures that the runs would at least match the existing performance. Where possible, parameter bounds were extended to double the upper bound specified by KLM (using the performance on the model as a target, if the original bounds were correct then configurations excluding them ought to be excluded by the search, but the statistical results at stage one has already shown that it is worth expanding the bounds and letting the search decide). The updated bounds were as specified in Table 7; for example, GFO now has an upper bound of 2.6. Optimization of accuracy was performed with no limitation on execution time (allowing for the possibility of the highest-accuracy runs having very large run times). Both tools were limited to 1000 evaluation runs.

Table 8 gives the best configurations found by iRace and SMAC. Although the screening experiment failed to improve on the default parameters for accuracy, an improvement was achieved here. Note, however, that the parameters found to be significant in the screening experiment have taken the optimal values determined in that stage: CG, SMO and CMO are 1, and the value for MLC is not 1. This provides some confidence that values for these variables found by the stochastic searches are not simply random noise but crucial contributors to the quality of the solution that cannot be improved much further. In the next section we consider that further by analysing the models generated during the SMAC run.

The optimal results obtained here for accuracy, and at stage one for time, were then used to seed the multi-objective optimization stage, to identify the trade-off between optimal configurations for each objective.

Table 8 Single-objective: parameter tuning results

	MR	GFO	SSBB3	MLS	HSFTO	HSFTI	MLC	HSFT	CMO	BMMO	CG	ROM	SMO	HSFMO	Accuracy (MSE)
iRace	0.04	2.56	72	7	6	25	3	14	0	0	1	DOWN	1	0	133.70
	0.54	2.46	46	9	5	20	3	4	0	0	1	DOWN	1	1	121.10
SMAC	0.99	2.52	15	10	3	5	5	10	0	0	1	DOWN	1	0	115.17
	0.89	2.19	17	4	3	5	3	8	1	1	1	DOWN	1	1	114.68

Table 9 Ten largest fANOVA main/pairwise effects

Effect size	Parameters
57.93% due to main effect	SMO
3.75% due to interaction	SMO x SSBB3
3.55% due to main effect	SSBB3
3.07% due to main effect	CMO
3.06% due to interaction	SMO x CMO
2.11% due to interaction	SSBB3 x HSFTI
1.36% due to main effect	HSFI
1.35% due to interaction	SMO x HSFTI
1.25% due to main effect	ROM
1.22% due to interaction	SSBB3 x CMO

4.4 Functional ANOVA

At this stage, we performed a deeper analysis of the sensitivities for accuracy by mining the models constructed by SMAC during the single-objective optimization. As noted above, the models used by SMAC are based on random forests, a machine learning tool for regression and classification. Random forests are collections of regression or decision trees, where the data is split at each leaf according to a threshold value in one of the variables. A linear sum over the leaves in one tree for a particular variable or combination of variables allows us to make a marginal prediction of the relationship between that variable and the response (accuracy in this case). In turn, marginal predictions for a forest are the average of marginals over the trees. Variance in the marginals over the trees shows uncertainty. This is known as *functional ANOVA*, implemented in the fANOVA tool (Hutter et al. 2014)[3] for the models generated by SMAC. In essence this is rather like an average over all possible configurations for the other parameters, but without having to run an extensive search over the space of parameter configurations.

The top ten effects as determined by fANOVA are given in Table 9. Note that of the parameters identified by the screening experiment as significantly impacting accuracy (Table 2), SMO and CMO feature heavily among these effects; both were shown to be strong drivers of accuracy in the exhaustive experiment Fig. (2). In contrast, CG and MLC do not feature at all (and were also confirmed as having less influence in the exhaustive experiment). By accounting for higher-order interactions, fANOVA is able to avoid being misled. This highlights the importance of deploying search-based methods: the human developer is unable to capture the higher order interactions and patterns, simply because there is too much data to do it.

fANOVA is also able to reveal helpful insights via the continuous marginal distributions – that is, how accuracy relates to the parameters over the whole space. The most helpful example in this work is given in Fig. 3 for GFO. Here, the recommended bounds specified by the company were 1.0-1.3, determined over years of fine-tuning. It is clear from the plot that there is a leap in accuracy (lower MSE) at the upper

[3] http://www.automl.org/fanova.html

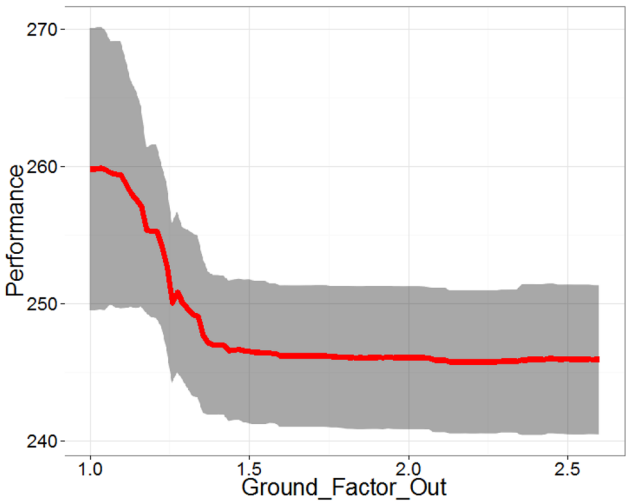


Fig. 3 Continuous marginal distribution — GFO. The solid line and shaded area are the mean ± one standard deviation for the configurations covered by the fANOVA

bound of this range, which we were able to capture by allowing the search a higher upper bound. It is possible that due to interactions with other variables, this trend was never seen in the manual experimentation but could only be picked up by the wide search used to train SMAC's models. The important thing to note here is the value added to the optimisation process by providing this analysis: we are able to *explain* the recommended parameter choice in a very accessible way, backed up by the extensive exploration of the space. This increases confidence in the results, improving the chance that the recommendations will be adopted.

4.5 Seeded multi-objective parameter tuning

A multi-objective optimisation was conducted using the well-known algorithm NSGA-II by Deb et al. (2002) to explore the trade-off between the conflicting accuracy and time objectives. In this work, we are not particularly concerned with search algorithm performance, only an improvement to Opium over the *status quo*, so the parameters for NSGA-II were not tuned extensively beyond some reasonable assumptions based on the problem size and limited empirical tuning. We are also not comparing against other algorithmic performances, but rather seeking to identify a manageable set of solutions to present to the decision maker, so only a single run was performed. The precise implementation used a Gray-encoded bit string representation so that all parameters could be handled by the same variation operators. Solutions comprised 45 bits for the step sizes and bounds listed in Table 10. The algorithm used a population size of 30, bit flip mutation at a rate of $1/n$ (n being the number of bits), uniform crossover with a rate of 100%, and binary tournament selection. Recall from Sect. 3 that the noise in the run-time objective is handled by taking the average from multiple runs of Opium using the same configuration. At each iteration, Pareto optimal solutions were

Table 10 Multi-objective optimisation: parameter bounds and step sizes

Parameter	LB	UB	Type	Step Size
MMR	0	1.0	Real	0.05
GFO	1	1.8	Real	0.05
SSBB3	0	100	Integer	10
MLS	2	12	Integer	1
HSFTO	0	30	Integer	1
HSFTI	0	60	Integer	1
MLC	1	10	Integer	1
HSFT	0	30	Integer	1
CMO	0	1	Boolean	1
BMMO	0	1	Boolean	1
CG	0	1	Boolean	1
ROM	Regular, Down, Up, Math, None		Categorical	n/a
SMO	0	1	Boolean	1
HSFMO	0	1	Boolean	1

stored in an archive to allow Pareto fronts larger than the population size to be found. The algorithm was terminated after 2000 unique evaluations. The bounds for each Opium parameter were extended (Table 10) to allow the search to explore outwith the human-defined limits. However, in light of the earlier results we were able to reduce the upper bounds considered earlier for some variables to allow the search to focus on the remaining variables. GFO and MLC were found to have a clear threshold after which accuracy could not be improved further, so their upper bounds were reduced to 1.8 and 10 respectively.

The Pareto front approximated by NSGA-II is plotted in the objective space in Fig. 4, with a point representing the default parameter configuration for comparison. The important result arising from this experiment was that the Pareto front dominates the default parameter settings. That is, it is possible to improve run time considerably, while also improving the accuracy (MSE). Four solutions from the Pareto front are highlighted in Table 11. The first two columns give the accuracy (MSE) and run-time for each of the configurations, with *Default* giving these values for the default parameter configuration. The next two rows give the values for the solutions with the lowest overall MSE and run-time, also expressed as percentages of those for the default configuration. The final two rows give the values for the solutions with the lowest MSE and lowest run-time, while keeping the other objective equal or less than that for the default configuration. This table shows that it is possible to achieve a run-time of 80% of that of the default configuration, with no loss in accuracy.

Figure 5 shows the Pareto front rendered to illustrate the relationships between the parameters and the objectives. As in Fig. 2, each row corresponds to one solution in the Pareto front, and the columns show the values taken by the solution for the parameters and the objectives. The solutions are sorted by ascending order of the accuracy (MSE) objective.

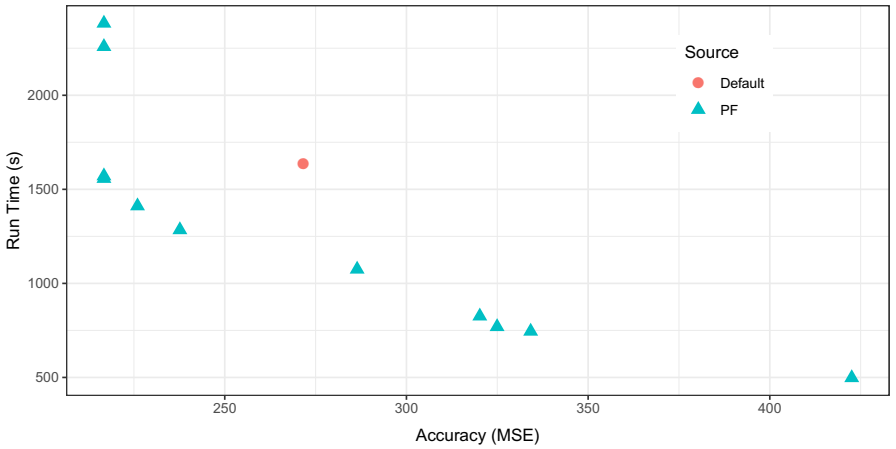


Fig. 4 Multi-objective optimisation results for all operating periods. The Pareto-optimal front (PF) approximated by NSGA-II dominates the default configuration

Table 11 Multi-objective optimisation: best solutions found

	All		OP12		OP34	
	Accuracy (MSE)	Run-time	Accuracy (MSE)	Run-time	Accuracy (MSE)	Run-time
Default	271.628	1633.9	252.528	1589.6	271.628	1360.4
Lowest MSE	216.748 (80%)	1557.2 (95%)	198.048 (78%)	1016.2 (64%)	216.748 (80%)	849.9 (62%)
Lowest run-time	422.548 (156%)	498 (30%)	390.748 (155%)	401.4 (25%)	390.748 (144%)	401.4 (30%)
Lowest MSE, matching default MSE	237.648 (87%)	1284.5 (79%)	251.748 (100%)	727.6 (46%)	268.188 (99%)	714.4 (53%)
Lowest run-time, matching default run-time	216.748 (80%)	1557.2 (95%)	198.048 (78%)	1016.2 (64%)	216.748 (80%)	849.9 (62%)

All of the Pareto-optimal solutions have CMO set TRUE, which matches the DoE / ANOVA analysis in Sect. 4.2. We can say with statistical confidence that this value is required to reach the Pareto front, and has not simply "floated" to 1 as a result of the random processes of the search. SMO is correlated with the objectives, with low-MSE and long-running configurations having SMO set TRUE and high-MSE, short-running, configurations having MSE set FALSE. This also matches the DoE analysis for accuracy (where SMO was significant) and time (where SMO was not significant). GFO tends to have values around 1.8, supported by the evidence from the statistical analysis and the functional ANOVA. CMO is all TRUE (matching the statistical analysis); CG is TRUE for low MSE / high runtimes and shows random variation for the shorter runtimes (also matching the statistical analysis; see Fig. 2). MLC does not show a particularly clear pattern (also matching the statistics, which

MMR	GFO	SSBB3	MLS	HSFTO	HSFTI	MLC	HSFT	CMO	BMMO	CG	ROM	SMO	HSFMO	MSE	RunTime
0.25	1.8	90	9	3	49	9	13	1	0	1	1	1	0	216.748	2382.6
0.25	1.8	90	8	3	48	5	0	1	0	1	1	1	0	216.748	2258.4
0.2	1.8	90	8	12	51	10	2	1	0	1	1	1	0	216.748	1570.9
0.2	1.8	90	8	3	49	5	2	1	0	1	1	1	0	216.748	1557.2
0.25	1.8	50	9	28	51	2	0	1	0	1	1	1	0	225.988	1411.4
0.35	1.8	40	3	9	50	5	1	1	0	1	0	1	0	237.648	1284.5
0.25	1.55	60	12	25	47	9	8	1	1	0	1	1	0	286.428	1075.0
0.25	1.6	100	7	2	48	10	2	1	0	1	1	0	0	320.188	825.8
0.2	1.6	100	4	5	12	10	15	1	0	1	0	0	0	324.948	769.4
0.5	1.3	100	12	6	40	10	14	1	1	1	1	0	1	334.188	745.0
0.25	1.7	10	12	24	46	10	7	1	0	0	1	0	0	422.548	498.0

Fig. 5 Multi-objective optimisation results, showing the parameter values taken by each Pareto-optimal solution. The solutions are sorted by the objectives, revealing that SMO is correlated with the run time, and suggesting that CMO is required to be TRUE (1) for optimality

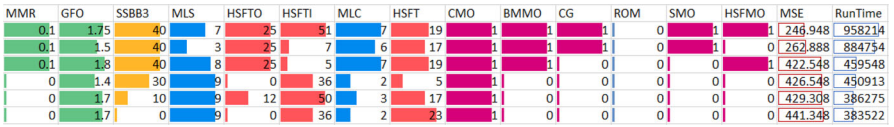
showed that it has no significant effect on accuracy, and possibly taking higher values for lower run times). HSFMO is mostly FALSE, but with one solution having it TRUE: the statistical analysis suggested that this variable is unimportant and can be ignored (potentially valuable developer feedback if the measure really should impact on one of the objectives, which would suggest a potential bug: in this case it was indeed deemed to only have a minor effect). For the other parameters, there is a lot of noise among the Pareto front: it is difficult to say what, if any, influence the other parameters have on the trade-off between accuracy and run-time, but this matches the statistical analysis. We can say with confidence that the values for these do not matter much, explaining the random variation in the solutions that can be off-putting to an end user.

4.6 Confirmation experiments

As a sanity check on the process we applied two different multi-objective optimisation algorithms to the optimisation stage. The goal here is not a comparison of algorithm performance, but simply to confirm that the explanations hold true for solutions generated by a different stochastic optimisation approach. Thus we do not compare Pareto fronts or show multiple repeat runs, but instead compare on the basis of the fronts from single runs, as a decision maker would need to do in practice. (A practitioner is only interested in having a single solution to implement, or a single Pareto front to choose a single solution from. This is especially true where single runs take weeks to complete.)

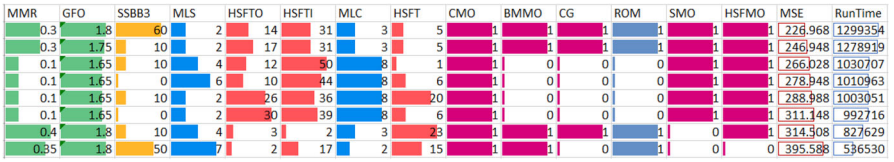
While NSGA-II remains extremely popular (and competitive) for 2-objective problems like the one at hand, for this experiment we choose two approaches from recent years: a multi-objective genetic algorithm, GWASFGA (Saborido et al. 2017), and the tuning tool based on a multi-objective iterated local search, MO-ParamILS (Blot et al. 2016). Both algorithms were seeded with the default, minimal MSE, and minimal time configurations found in the earlier stages.

GWASFGA was implemented in the jMetal (Nebro et al. 2015) framework (v5.10), using default configurations of integer SBX crossover (probability 1, distribution 0.5), integer polynomial mutation, binary tournament selection using ranking and crowding distance comparisons, epsilon 0.01, population size 30, and maximum evaluations 2000 (to match that for NSGA-II). Solutions were encoded as integer solutions, with



MMR	GFO	SSBB3	MLS	HSFTO	HSFTI	MLC	HSFT	CMO	BMMO	CG	ROM	SMO	HSFMO	MSE	RunTime
0.1	1.75	40	7	25	51	7	19	1	1	1	0	1	1	246.948	958214
0.1	1.5	40	3	25	7	6	17	1	1	1	0	1	0	262.888	884754
0.1	1.8	40	8	25	5	7	19	1	1	0	0	0	1	422.548	459548
0	1.4	30	9	0	36	2	5	1	0	0	0	0	0	426.548	450913
0	1.7	10	9	12	50	3	17	1	0	0	0	0	0	429.308	386275
0	1.7	0	9	0	36	2	23	1	0	0	0	0	0	441.348	383522

Fig. 6 Multi-objective optimisation results, showing the parameter values taken by each Pareto-optimal solution found for GWASFGA. Similar patterns are seen to those in Fig. 5, for the variables found to be important by the statistical analysis



MMR	GFO	SSBB3	MLS	HSFTO	HSFTI	MLC	HSFT	CMO	BMMO	CG	ROM	SMO	HSFMO	MSE	RunTime
0.3	1.8	60	2	14	31	3	5	1	1	1	1	1	1	226.968	1299354
0.3	1.75	10	2	17	31	3	5	1	1	1	1	1	1	246.948	1278919
0.1	1.65	10	4	12	50	8	1	1	0	0	0	1	1	266.028	1030707
0.1	1.65	0	6	10	44	8	6	1	0	0	0	1	1	278.948	1010963
0.1	1.65	10	2	26	36	8	20	1	0	0	0	1	1	288.988	1003051
0.1	1.65	0	2	30	39	8	6	1	0	0	0	1	1	311.148	992716
0.4	1.8	10	4	3	2	3	23	1	1	1	1	0	1	314.508	827629
0.35	1.8	50	7	2	17	2	15	1	1	0	1	0	0	395.588	536530

Fig. 7 Multi-objective optimisation results, showing the parameter values taken by each Pareto-optimal solution found for MO-ParamILS. Similar patterns are seen to those in Fig. 5, for the variables found to be important by the statistical analysis

continuous parameters being mapped to a finite list of values at the same granularity as used for NSGA-II in the previous section (Table 10).

MO-ParamILS used the default configuration, with an evaluation budget of 2000, and also specified the continuous parameters using the same granularity for NSGA-II in (Table 10).

Despite the different algorithms used, for the parameters that were found to be important by the statistical testing, the same patterns can be observed as for the NSGA-II run (Figs. 6, 7). Those patterns also align with the trends suggested by the statistics, offering confidence in these results and explaining what the algorithm has found. Specifically:

- All the Pareto-optimal solutions have CMO set TRUE.
- SMO is TRUE for low-MSE / long-running configurations and FALSE for high-MSE, short-running, configurations.
- GFO tends to have values around 1.8
- CG is TRUE for low MSE / high runtimes, but shows random variation with high MSE / low runtimes

While HSFMO does show a pattern in the MO-ParamILS results (TRUE for low MSE, FALSE for low runtime), this differs from the results with the other two algorithms. The statistical analysis suggested that this variable is unimportant and can be ignored: thus we can conclude that the apparent pattern is simply a product of the stochastic processes involved in the search. Likewise, SMMO shows a pattern (TRUE for low MSE, FALSE for low runtime) for GWASFGA but not the other algorithms, but the statistics show that this parameter can be ignored.

For the other parameters, as with NSGA-II, they show no clear pattern among the Pareto front. This result is explained by the statistical testing showing that these parameters are unimportant.

4.7 Separate operating periods

The flight schedules for KLM are divided into four operating periods (OP) per year. The characteristics of the schedules vary over the year: obviously traffic at peak times (e.g. summer holiday period) is different to the off-peak times in both density and distribution. Discussions with the airline suggested that this may have an impact on the accuracy of Opium's simulation. Treating parameter tuning as an optimisation problem allows us to test whether different parameter configurations are suited to the different OPs. Multi-objective optimisation was applied to Opium running on OPs 1 & 2 together (OP12), and 3 & 4 (OP34). In terms of the objectives, both yielded similar improvements to those found with running on all OPs together. Table 11 shows that run times of 78% and 80% that of the default configuration can be found for OP12 and OP34 respectively, with no loss in accuracy.

For both OPs, CMO is always TRUE and GFO centres around 1.8 in the Pareto-optimal solutions. This matches the trend seen when tuning for all OPs. However, several of the parameters show differences. For OP12, SMO and HSFMO show no discernible trend. For OP34, SMO tends from TRUE to FALSE as MSE decreases. HSFMO tends from FALSE to TRUE as MSE decreases.

For OP12, MLC is negatively correlated with MSE, for OP34 is positively correlated with MSE. For both OP12 and OP34, BMMO is mostly TRUE, in contrast to tuning for all OPs where BMMO was mostly FALSE.

Having already done the implementation work in order to optimise Opium for the full data set, specialising to particular data sets is now trivial. Overall, then, the systematic procedure was able to find improved configurations, tailed to the specific sets of input data, with little additional cost in terms of personnel time.

5 Conclusion and discussion

Bespoke industrial software applications often carry a legacy of incremental fine-tuning and configuration that leaves them performing sub-optimally. Configuration parameters are left with a "do not touch" warning attached: as with so many application areas, a radical change to the configuration must be rooted in a methodical approach backed up by evidence if it is to gain acceptance. This motivates the systematic approach to exploring and optimising such parameters that we have proposed. Each of the components of our approach is not novel in itself but the idea is that the combination of techniques brings optimisation of the software hyperparameters with explanation, and will allow practitioners to tackle similar problems for other industrial software.

In a case study to prove the concept, we worked closely with partners at the international airline KLM to apply the methodology to a real-world software application, Opium. All flight schedules were passed through Opium before implementation so the software represents a critical part of the business workflow with global impact: if you have flown with KLM, this software has impacted on you! We were able to improve both accuracy and speed of Opium, with the optimised configurations being adopted by the company until the application was superseded by new applications based on it.

This led to direct commercial impact by improving the robustness of schedules based on estimation models.

Importantly, our approach was also able to show how various parameters influence the objectives, and how they interact, backed up up by well-accepted statistical testing. In contrast to the current parameter-tuning tools built on application of search-based optimisation, this offers some *explanation* as to why the approach settled on specific configurations, and adds confidence to the results. It was noted by the developers at KLM that this explanation was enough to justify the counter-intuitive results found by the optimisation process. Initially, the allocation of values outside the normal range was met with skepticism, but the additional statistical analysis and trends revealing how the variables influenced the objectives meant that the solutions could be adopted with confidence. Thus we see the benefit of this form of explanation: increasing the likelihood of adoption and so increasing the chance that the benefits of the optimisation will be realised in practice.

A potential limitation with the approach is the reliance on important of single and two-way variable relationships for explanations. For many benchmark and real-world problems, even those with highly non-linear fitness functions, algorithms using only 1 and 2-variable groups are still able to capture much higher-order structures in the problem for locating the optimum (Brownlee et al. 2008, 2009), so these do represent a strong foundation for explaining solutions to a human. However, further investigation in to what degree of inter-variable relationship is critical to explanation represents one direction for future work.

Having followed this process for the real-world application described in the case study, we are able to make some key recommendations for others seeking to optimise an existing piece of software:

1. Do not underestimate the potential for simple parameter tuning. Most applications have such 'baked in' configurations that are ripe for improvement. We originally approached this problem with the intention of advancing to more sophisticated SBSE techniques. Of course, SBSE approaches that seek to improve (for example) the source code itself offer much potential but, at least in this case, we were able to demonstrate substantial improvement through simply tuning the parameters.
2. Once a wrapper for the tool has been implemented that returns quality measures for a particular configuration, additional search techniques come almost for free, so the multiple stages in our workflow did not take much extra development time.
3. As part of the process, embed analysis of the sensitivity of the objectives to the parameters in order to explain results; this adds confidence (particular in the case of 'unusual' values being selected) and improves the chance of uptake. This can be particularly important because staff tend to have strong belief that a certain parameter should have a certain value; if the optimisation process confirms that pre-determined value it is easily accepted, but if not, then additional evidence is needed to support the change. Each stage of our workflow (from the standard statistical testing through to multi-objective optimisation) adds a different layer of insight to support such analysis.

4. Following the previous point, if it is technically possible to do so, increase the lower and upper bounds of each parameter beyond the defaults for the search process.
5. Seed the search with default and known good configurations to guarantee at least matching the original software's performance.
6. Consider exploration of trade-offs between non-functional and functional properties. In our case this meant looking at time and accuracy. As it happens, we were able to improve both, but there was also great interest from the developers in potentially trading off some accuracy for much faster results).

The proposed approach has been demonstrated to work well for the case study application, and the logical next step is to explore further applications. Real-world parameter tuning problems usually comprise a mixture of variable types and ranges, in contrast with the majority of existing evolutionary computation benchmarks (a notable exception being (Tušar et al. 2019)), so further work towards explanation of metaheuristics for optimisation will also need the development of new benchmarks with clearly defined "explanations" to discover.

Acknowledgements Work carried out under the DAASE project (UK EPSRC Grant Number EP/J017515/1).

Data Access Statement The data and software application used in the case study is unavailable due to commercial sensitivities.

References

Adadi, A., Berrada, M.: Peeking inside the black-box: A survey on explainable artificial intelligence. IEEE Access **6**, 52138–52160 (2018). https://doi.org/10.1109/ACCESS.2018.2870052

Alabas-Uslu, C., Dengiz, B.: Parameter tuning problem in metaheuristics: A self-adaptive local search algorithm for combinatorial problems. In: Women in Industrial and Systems Engineering, Springer, pp. 93–111 (2020)

Bennett, K.P., Kunapuli, G., Hu, J., Pang, J.S.: Bilevel optimization and machine learning. In: IEEE World Congress on Computational Intelligence, Springer, pp. 25–47 (2008)

Bezerra, L.C.T., López-Ibáñez, M., Stützle, T.: Automatic Configuration of Multi-objective Optimizers and Multi-objective Configuration, Springer International Publishing, Cham, pp. 69–92. doi:10.1007/978-3-030-18764-4_4 (2020)

Birattari, M., Kacprzyk, J.: Tuning metaheuristics: a machine learning perspective, Studies in Computational Intelligence, vol. 197. Springer (2009)

Blot, A., Hoos, H.H., Jourdan, L., Kessaci-Marmion, M.É., Trautmann, H.: MO-ParamILS: A multi-objective automatic algorithm configuration framework. In: Festa, P., Sellmann, M., Vanschoren, J. (eds.) Learning and Intelligent Optimization, pp. 32–47. Springer International Publishing, Cham (2016)

Brownlee, A.E.I.: Mining Markov network surrogates for value-added optimisation. In: Proc GECCO Companion, https://doi.org/10.1145/2908961.2931711 (2016)

Brownlee, A.E.I., Wright, J.A.: Solution analysis in multi-objective optimization. In: Proc. Building Simulation and Optimisation Conference, IBPSA-England, Loughborough, UK, pp. 317–324 (2012)
Brownlee, A.E.I., Pelikan, M., McCall, J.A.W., Petrovski, A.: An app.lication of a multivariate EDA to cancer chemotherapy. In: Proc. GECCO, pp. 463–464, https://doi.org/10.1145/1389095.1389179 (2008)
Brownlee, A.E.I., McCall, J.A.W., Shakya, S.K., Zhang, Q.: Structure Learning & Optimisation in a Markov-network based EDA. In: Proc IEEE CEC, pp. 447–454 (2009)
Brownlee, A.E.I., McCall, J.A.W., Zhang, Q.: Fitness modeling with Markov networks. IEEE T Evol Comp **17**(6), 862–879 (2013). https://doi.org/10.1109/TEVC.2013.2281538
Brownlee, A.E.I., Wright, J.A., He, M., Lee, T., McMenemy, P.: A novel encoding for separable large-scale multi-objective problems and its app.lication to the optimisation of housing stock improvements. Appl. Soft Comput. **96**, 106650 (2020)
Bruce, B.R., Aitken, J.M., Petke, J.: Deep parameter optimisation for face detection using the viola-jones algorithm in opencv. In: International Symposium on Search Based Software Engineering, Springer, pp. 238–243 (2016)
Czarn, A., MacNish, C., Vijayan, K., Turlach, B., Gupta, R.: Statistical exploratory analysis of genetic algorithms. IEEE Trans. Evol. Comput. **8**(4), 405–421 (2004). https://doi.org/10.1109/tevc.2004.831262
Dang NTT, De Causmaecker P (2014) Motivations for the development of a multi-objective algorithm configurator. In: Proceedings of the 3rd International Conference on Operations Research and Enterprise Systems, SCITEPRESS, pp. 328–333
Deb, K., Srinivasan, A.: Innovization: Innovating design principles through optimization. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 1629–1636 (2006)
Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE T Evolut Comput **6**(2), 182–197 (2002)
Mejía-de Dios, J.A., Mezura-Montes, E., Quiroz-Castellanos, M.: Automated parameter tuning as a bilevel optimization problem solved by a surrogate-assisted population-based app.roach. App.lied Intelligence pp. 1–23 (2021)
Dréo, J.: Using performance fronts for parameter setting of stochastic metaheuristics. In: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, pp. 2197–2200 (2009)
Eiben, A.E., Smit, S.K.: Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm Evol. Comput. **1**(1), 19–31 (2011)
Field, A.: Discovering statistics using ibm spss statistics 5th ed (2018)
Harman, M., McMinn, P., de Souza, J.T., Yoo, S.: Empirical software engineering and verification. Springer, Berlin, chap Search Based Software Engineering: Techniques, Taxonomy, Tutorial, pp. 1–59 (2012)
Hendricks L, Akata Z, Rohrbach M, Donahue J, Schiele B, Darrell T (2016) Generating visual explanations. In: European Conference on Computer Vision, Springer, pp. 3–19
Hoffmann, H., Sidiroglou, S., Carbin, M., Misailovic, S., Agarwal, A., Rinard, M.: Dynamic knobs for responsive power-aware computing. In: Proc. Int'l Conf. on Architectural supp.ort for programming languages and operating systems, ACM, Newport Beach, CA, pp. 199–212, https://doi.org/10.1145/1950365.1950390 (2011)
Hoos, H.H.: Programming by optimization. Commun. ACM **55**(2), 70–80 (2012). https://doi.org/10.1145/2076450.2076469
Hornby, G., Yu, T.: EC practitioners: results of the first survey. ACM SIGEVOlution **2**(1), 2–8 (2007)
Hutter, F.: Automated configuration of algorithms for solving hard computational problems. PhD thesis, University of British Columbia (2009)
Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: an automatic algorithm configuration framework. Journal of Artificial Intelligence Research **36**, 267–306 (2009)
Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Proc. of LION-5, pp. 507–523 (2011)
Hutter, F., Hoos, H., Leyton-Brown, K.: An efficient app.roach for assessing hyperparameter importance. In: Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, JMLR.org, ICML'14, pp. I–754–I–762 (2014)
Jacobs, P.H.: The DSOL simulation suite. PhD thesis, TU Delft, Delft University of Technology (2005)
Karafotias, G., Hoogendoorn, M., Eiben, A.E.: Parameter Control in Evolutionary Algorithms: Trends and Challenges. IEEE Trans. Evol. Comput. **19**(2), 167–187 (2015)
Le Bras, P., et al.: Imp user conf in concept maps: Exploring data driven explanations. In: Proc. CHI Conf., pp. 404:1–404:13, https://doi.org/10.1145/3173574.3173978 (2018)

Lehman, J., Clune, J., Misevic, D., Adami, C., Altenberg, L., Beaulieu, J., Bentley, P.J., Bernard, S., Beslon, G., Bryson, D.M., et al.: The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. Artif. Life **26**(2), 274–306 (2020)

Li, K., Omidvar, M.N., Deb, K., Yao, X.: Variable interaction in multi-objective optimization problems. In: Parallel Problem Solving from Nature – pp.SN XIV, Springer Nature, pp. 399–409, https://doi.org/10.1007/978-3-319-45823-6_37 (2016)

Liang, J.Z., Miikkulainen, R.: Evolutionary bilevel optimization for complex control tasks. In: Proceedings of the 2015 annual conference on genetic and evolutionary computation, pp. 871–878 (2015)

López-Ibáñe,z M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3:43 – 58, https://doi.org/10.1016/j.orp.2016.09.002 (2016)

Minitab, Inc: Minitab 17 statistical software. Software (2014)

Nebro, A.J., Durillo, J.J., Vergne, M.: Redesigning the jmetal multi-objective optimization framework. In: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, Association for Computing Machinery, New York, NY, USA, GECCO Companion '15, p. 1093–1100, https://doi.org/10.1145/2739482.2768462 (2015)

Petrovski, A., Wilson, A., McCall, J.: Statistical identification and optimisation of significant GA factors. In: Proceedings of the 5th Joint Conference on Information Sciences (JCIS'2000), Atlantic City, USA, pp. 1027–1030 (2000)

Petrovski, A., Brownlee, A.E.I., McCall, J.A.W.: Statistical optimisation and tuning of GA factors. In: Proc. IEEE CEC, IEEE Press, vol. 1, pp. 758–764 (2005)

Ribeiro M, Singh S, Guestrin C (2016) Why should I trust you?: Explaining the predictions of any classifier. In: Proceedings of the SIGKDD Conference on Knowledge Discovery & Data Mining, ACM, pp. 1135–1144

Saborido, R., Ruiz, A.B., Luque, M.: Global WASF-GA: An evolutionary algorithm in multiobjective optimization to approximate the whole Pareto optimal front. Evol. Comput. **25**(2), 309–349 (2017). https://doi.org/10.1162/EVCO_a_00175

Santana, R., Bielza, C., Lozano, J., Larrañaga, P.: Mining probabilistic models learned by EDAs in optimization of multiobjective problems. In: Proc. GECCO, pp. 445–452, https://doi.org/10.1145/1569901.1569963 (2009)

Schmider, E., Ziegler, M., Danay, E., Beyer, L., Bühner, M.: Is it really robust? Methodology (2010)

Sinha, A., Malo, P., Xu, P., Deb, K.: A bilevel optimization app.roach to automated parameter tuning. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, pp. 847–854 (2014)

Sinha, A., Khandait, T., Mohanty, R.: A gradient-based bilevel optimization app.roach for tuning hyperparameters in machine learning. arXiv preprint arXiv:2007.11022 (2020)

Smit, S.K., Eiben, A.E., Szlávik, Z., et al.: An moea-based method to tune ea parameters on multiple objective functions. In: IJCCI (ICEC), pp. 261–268 (2010)

Sohn, J., Lee, S., Yoo, S.: Amortised deep parameter optimisation of gpgpu work group size for opencv. In: International Symposium on Search Based Software Engineering, Springer, pp. 211–217 (2016)

Stützle, T., López-Ibáñez, M.: Automated Design of Metaheuristic Algorithms, Springer International Publishing, Cham, pp. 541–579. https://doi.org/10.1007/978-3-319-91086-4_17 (2019)

Tiwari, A., Hoyos, P.N., Hutabarat, W., Turner, C., Ince, N., Gan, X.P., Prajapat, N.: Survey on the use of computational optimisation in UK engineering companies. CIRP J. Manuf. Sci. Technol. **9**, 57–68 (2015). https://doi.org/10.1016/j.cirpj.2015.01.003

Tušar, T., Brockhoff, D., Hansen, N.: Mixed-integer benchmark problems for single- and bi-objective optimization. In: Proceedings of the Genetic and Evolutionary Computation Conference, Association for Computing Machinery, New York, NY, USA, GECCO '19, p 718–726, https://doi.org/10.1145/3321707.3321868 (2019)

Urquhart, N., Guckert, M., Powers, S.: Increasing trust in meta-heuristics by using map-elites. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, Association for Computing Machinery, New York, NY, USA, GECCO '19, p. 1345–1348, https://doi.org/10.1145/3319619.3326816 (2019)

Vincalek, J., Walton, S., Evans, B.: It's the journey not the destination: Building genetic algorithms practitioners can trust. In: Proceedings of the Genetic and Evolutionary Computation Conference

Companion, Association for Computing Machinery, New York, NY, USA, GECCO '21, p. 231–232, https://doi.org/10.1145/3449726.3459483 (2021)
Wu, F., Weimer, W., Harman, M., Jia, Y., Krinke, J.: Deep parameter optimisation. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, ACM, New York, NY, USA, GECCO '15, pp. 1375–1382, https://doi.org/10.1145/2739480.2754648 (2015)
Yu, T., Zhu, H.: Hyper-parameter optimization: A review of algorithms and app.lications. arXiv:2003.05689 (2020)
Zhang, T., Georgiopoulos, M., Anagnostopoulos, G.C.: S-race: A multi-objective racing algorithm. In: Proceedings of the 15th annual conference on Genetic and evolutionary computation, pp. 1565–1572 (2013)