ADVCOMP 2021

The Fifteenth International Conference on Advanced Engineering Computing and Applications in Sciences

ISBN: 978-1-61208-887-7

October 3 - 7, 2021

Barcelona, Spain

ADVCOMP 2021 Editors

Cosmin Dini, IARIA, USA/EU

Evgeny Pyshkin, University of Aizu, Japan

Marcin Hojny, AGH University of Science and Technology, Poland

ADVCOMP 2021

Forward

The Fifteenth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2021) continued a series of events on fundamentals of advanced scientific computing and specific mechanisms and algorithms for particular sciences. The conference contributions presenting novel research in all aspects of new scientific methods for computing and hybrid methods for computing optimization, as well as advanced algorithms and computational procedures, software and hardware solutions dealing with specific domains of science.

With the advent of high-performance computing environments, virtualization, distributed and parallel computing, as well as the increasing memory, storage and computational power, processing particularly complex scientific applications and voluminous data is more affordable. With the current computing software, hardware and distributed platforms effective use of advanced computing techniques, this becomes more achievable.

We take here the opportunity to warmly thank all the members of the ADVCOMP 2021 technical program committee, as well as all the reviewers. The creation of such a high-quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and effort to contribute to ADVCOMP 2021. We truly believe that, thanks to all these efforts, the final conference program consisted of top-quality contributions. We also thank the members of the ADVCOMP 2021 organizing committee for their help in handling the logistics of this event.

ADVCOMP 2021 Chairs

ADVCOMP 2021 Steering Committee
Dean Vucinic, Vrije Universiteit Brussel (VUB), Belgium, FERIT, Croatia
Juha Röning, University of Oulu, Finland
Alfred Geiger, T-Systems Information Services GmbH, Germany
Marcin Hojny, AGH University of Science and Technology, Poland
Hans-Joachim Bungartz, TUM, Germany
Alice E. Koniges, University of Hawai'i at Mānoa, USA
Andreas Rausch, TU Clausthal, Clausthal-Zellerfeld, Germany

ADVCOMP 2021 Publicity Chairs
Lorena Parra, Universitat Politecnica de Valencia, Spain
José Miguel Jiménez, Universitat Politecnica de Valencia, Spain

ADVCOMP 2021
Committee

ADVCOMP 2021 Steering Committee
Dean Vucinic, Vrije Universiteit Brussel (VUB), Belgium, FERIT, Croatia
Juha Röning, University of Oulu, Finland
Alfred Geiger, T-Systems Information Services GmbH, Germany
Marcin Hojny, AGH University of Science and Technology, Poland
Hans-Joachim Bungartz, TUM, Germany
Alice E. Koniges, University of Hawai'i at Mānoa, USA
Andreas Rausch, TU Clausthal, Clausthal-Zellerfeld, Germany

ADVCOMP 2021 Publicity Chairs
Lorena Parra, Universitat Politecnica de Valencia, Spain
José Miguel Jiménez, Universitat Politecnica de Valencia, Spain

ADVCOMP 2021 Technical Program Committee
Waleed H. Abdulla, University of Auckland, New Zealand
José Abellán, Catholic University of Murcia, Spain
Mohamed Riduan Abid, Alakhawayn University, Morocco
Francisco Airton Silva, Federal University of Piauí, Brazil
M. Azeem Akbar, Nanjing University of Aeronautics and Astronautics, China
Haifa Alharthi, Saudi Electronic University, Saudi Arabia
Sónia Maria Almeida da Luz, Polytechnic Institute of Leiria - School of Technology and Management, Portugal
Madyan Alsenwi, Kyung Hee University, Global Campus, South Korea
Mohamed E. Aly, California State Polytechnic University, Pomona, USA
Daniel Andresen, Kansas State University, USA
Alberto Antonietti, Politecnico di Milano / University of Pavia, Italy
Ehsan Atoofian, Lakehead University, Canada
Vadim Azhmyakov, Universidad Central, Bogota, Republic of Colombia
Carlos Becker Westphall, University of Santa Catarina, Brazil
Raoudha Ben Djemaa, ISITCOM | University of Sousse, Tunisia
Rudolf Berrendorf, Bonn-Rhein-Sieg University, Germany
Estêvão Bissoli Saleme, Federal University of Espírito Santo, Brazil
Sergiy Bogomolov, Newcastle University, UK
Alessandro Borri, CNR-IASI Biomathematics Laboratory, Rome, Italy
David Bouck-Standen, Kingsbridge Research Center, UK
Hans-Joachim Bungartz, TUM, Germany
Xiao-Chuan Cai, University of Colorado Boulder, USA
Graziana Cavone, Polytechnic of Bari, Italy
Mete Celik, Erciyes University, Turkey
Jinyuan Chen, Louisiana Tech University, USA
Rangeen Basu Roy Chowdhury, Intel Corporation, USA
Vassilios V. Dimakopoulos, University of Ioannina, Greece
Inês Domingues, IPO Porto Research Centre (CI-IPOP), Portugal
Shi Dong, Northeastern University, Boston, USA

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Statistical and Principal Component Analysis in the Design of Alkaline Methanol Fuel Cells

Tanja Clees
University of Applied Sciences
Bonn-Rhein-Sieg and Fraunhofer Institute
for Algorithms and Scientific Computing
Sankt Augustin, Germany
email: Tanja.Clees@scai.fraunhofer.de

Bernhard Klaassen
Fraunhofer Institute for Algorithms
and Scientific Computing
Sankt Augustin, Germany
email: Bernhard.Klaassen@scai.fraunhofer.de

Igor Nikitin
Fraunhofer Institute for Algorithms
and Scientific Computing
Sankt Augustin, Germany
email: Igor.Nikitin@scai.fraunhofer.de

Lialia Nikitina
Fraunhofer Institute for Algorithms
and Scientific Computing
Sankt Augustin, Germany
email: Lialia.Nikitina@scai.fraunhofer.de

Sabine Pott
Fraunhofer Institute for Algorithms
and Scientific Computing
Sankt Augustin, Germany
email: Sabine.Pott@scai.fraunhofer.de

***Abstract*—In this paper, the electrochemical alkaline methanol oxidation process, which is relevant for the design of efficient fuel cells, is considered. An algorithm for reconstructing the reaction constants for this process from the experimentally measured polarization curve is presented. The approach combines statistical and principal component analysis and determination of the trust region for a linearized model. It is shown that this experiment does not allow one to determine accurately the reaction constants, but only some of their linear combinations. The possibilities of extending the method to additional experiments, including dynamic cyclic voltammetry and variations in the concentration of the main reagents, are discussed.**

***Index Terms*—modeling of complex systems; observational data and simulations; advanced applications; mathematical chemistry.**

I. Introduction

Fuel cells are environmentally friendly portable energy sources based on obtaining electricity as a result of electrochemical reactions. They are similar to galvanic cells; the difference is that the main reagents in fuel cells can be replenished many times. Among fuel cells, a special group is formed by the so-called direct fuel cells, in which the intermediate stage of the production of gaseous hydrogen is omitted and the oxidative reaction proceeds directly. In fact, this is the same combustion reaction, but here the energy is released not in the form of heat or mechanical pressure of gas, but in the form of electric power. Among the various fuels in such cells, the most common are alcohols, methanol and ethanol. We consider the methanol oxidation reaction to reduce the number of intermediate reagents for modeling. Although acidic reactions are easier to model and have been well studied, they rely on the use of expensive noble metal electrodes. The alkaline environment allows the use of cheaper materials for the production of electrodes.

The main problem for the analysis of electrochemical alkaline methanol oxidation is the large number of intermediate reagents and reactions, as well as the fact that the elementary reaction constants are not known a priori, and they must be reconstructed from the experiment. Moreover, such quantities, as surface coverages of the electrode, are experimentally not measurable and require mathematical modeling. The challenge here is to consider the systems of differential-algebraic equations of high dimension. Additional complication comes from the stiffness of the system: some components evolve much faster than others. There are also slow reactions that inhibit the entire process and lead to the unattainability of a stationary state and hysteresis effects in the current-voltage dependence.

In this paper, we continue our research on mathematical modeling of alkaline methanol oxidation in the context of design of efficient fuel cells. A detailed description of the mathematical model is given in our previous paper [1]. In our other papers, a model reduction [2] of the system from Ordinary Differential Equations (ODE) to Differential-Algebraic Equations (DAE) was performed; a chemical interpretation for hysteresis effect [3] in dynamics of the system is presented; a footprint of the dynamics in the form of electrochemical impedance spectrum [4] was analyzed. Our papers extend the approaches of [5]–[8] for analysis of this and analogous electrochemical processes, paying more attention to mathematical aspects of the problems. We use the electrochemical measurement methodology from [9]. We also use general methods of model data analysis [10], most of which are implemented as ready-to-use procedures in the system Mathematica [11].

The main goal of the experiments under consideration is the reconstruction of the reaction constants describing the underlying electrochemical processes. Note that the reconstructed values of the constants for alkaline methanol oxidation are given in [1]. The main purpose of this work is to estimate *the reconstruction accuracy* of the reaction constants.

In Section II, an overview of the mathematical model of the reactions in alkaline methanol oxidation is given. In

 ISBN: 978-1-61208-887-7

Section III, the method of statistical and principal component analysis for estimating the reconstruction accuracy of the reaction constants in the considered process is presented. Section IV summarizes the obtained results.

The experimental measurements used in this paper were carried out by our colleagues at INES / TU Braunschweig [1]–[4].

II. The model

Figure 1a shows the chemical reaction network under consideration. Figure 1b shows an experimental setup containing an evacuated teflon cell with a rotating electrode. A detailed description of reactions and setup can be found in [1]. In the experiments, the voltage η in the cell changes along a certain profile, and the current I_{cell} through the cell is measured. In particular, Cyclic Voltammetry (CV) experiments use a saw-like voltage profile shown in Figure 1c.

The resulting cell response is shown in Figure 2a. Note that at low voltage values, the profiles for the increasing and decreasing half-cycles of the saw run approximately along the same curve. In this zone, the reagents are in equilibrium, their concentrations depend only on the voltage, following the so-called Polarization Curve (PC). At higher voltages, a new reagent is formed. Its reactions are slow, as a result, the equilibrium is disturbed, a delay appears in the response of the system, and a characteristic hysteresis is formed on the CV-plot.

The evolution of the system is described by ODE of the form

$$d\theta_i/dt = F_i(\theta,\eta),\ i = 1..6,\ I_{cell} = F_7(\theta,\eta), \qquad (1)$$

where $\theta_i \in [0,1]$ are surface coverages for 6 reagents. The right-hand sides of the equations are polynomials in θ_i, the exact form of which is given in [1]. Some monomials corresponding to electron exchange reactions have an exponential voltage dependence. The purpose of the experiments is to reconstruct the reaction constants k_i, which are the coefficients of the monomials in the given model. In total, there are 14 reaction constants for the considered system of reactions. In the normalization used in [1], the reaction constants are measured in $[mol/(m^2 s)]$, while their numerical values vary over a wide range. Therefore, it is convenient to use the decimal logarithms of the reaction constants as model parameters: $p_i = \log_{10} k_i$.

III. Statistical and principal component analysis

In this work, we will consider the PC-part of the CV-curve, shown in detail in Figure 2b. In this part, the state of the system can be considered stationary, and the terms with derivatives in (1) can be omitted. As a result, a closed polynomial system on θ_i is formed, which can be solved. The obtained θ_i as a function of voltage can be substituted into the expression for the current, resulting in a model response function $I_{cell} = f(\eta,p)$. In Figure 2b, the red curve corresponds to the model, and the blue dots to the measured values. It can be seen that the model reproduces the experiment very well.

For reconstruction in [1], a fitting procedure is used that minimizes the sum of the squares of the deviations of the model from the values measured in the experiment:

$$L_2^2 = \textstyle\sum_i (f(\eta_i,p) - f_{exp,i})^2. \qquad (2)$$

The p-values obtained at the minimum point represent the reconstructed reaction constants. The accuracy of the reconstruction is determined as follows [10]. At the minimum, the sensitivity matrix X and related matrices are determined:

$$X_{ij} = \partial f(\eta_i,p)/\partial p_j,\ cov = \epsilon^2 (X^T X)^{-1} \qquad (3)$$

$$\sigma_i = (cov_{ii})^{1/2},\ corr_{ij} = \sigma_i^{-1} cov_{ij}\, \sigma_j^{-1}, \qquad (4)$$

the covariance matrix cov, the diagonal values of which determine the standard deviations of the parameters σ_i, and the correlation matrix $corr$. The value ϵ appearing in the definitions represents the estimate of the experimental data error $f_{exp,i}$, calculated by the formula $\epsilon^2 = L_2^2/N_{dof}$, $N_{dof} = N_{pt} - N_{par}$. Here, N_{pt} is the number of experimental points, N_{par} is the number of model parameters, N_{dof} is the number of degrees of freedom for a fit, in our case:

$$N_{pt} = 21,\ N_{par} = 14,\ N_{dof} = 7, \qquad (5)$$

$$L_2^2 = 2.08 \cdot 10^{-9} A^2,\ \epsilon = 1.72 \cdot 10^{-5} A. \qquad (6)$$

The obtained small value of ϵ corresponds to the good quality of the fit. In fact, this value corresponds to the size of the dot on the graph Figure 2b.

In the case when the parameter errors are large and highly correlated with each other, Principal Component Analysis (PCA) must be performed to interpret the result. The confidence region in parameter space is an ellipsoid that can be stretched in some directions and compressed in others. In stretched directions, measurements have a large error, in compressed directions – small one. The values and directions of the semi-axes of the ellipsoid can be determined using Singular Value Decomposition (SVD) of the sensitivity matrix:

$$X = u\lambda v^T,\ u^T u = 1,\ v^T v = v v^T = 1,\ a_k = \epsilon/\lambda_k. \qquad (7)$$

Here, the matrix X is $N_{pt} \times N_{par}$ rectangular, u is $N_{pt} \times N_{par}$ semi-orthogonal, λ is $N_{par} \times N_{par}$ diagonal, v is $N_{par} \times N_{par}$ orthogonal. The a_k values represent the semi-axes of the error ellipsoid. The columns of the v matrix (or the rows of the v^T matrix) represent the directions of the axes of the ellipsoid in the parameter space, while the columns of the u matrix represent the profiles of the principal components in the space of experiments. Figure 2d shows such profiles for the first four components, in red-green-blue-cyan order for $u_{i,1-4}$. These profiles show the variation of PC curve when the parameters are displaced along the axes of the error ellipsoid.

Note: addressing the question, why we choose PCA for the analysis, and not another factorizing method, e.g., Independent Component Analysis (ICA) or Curvilinear Component Analysis (CCA), etc. These methods are very close to PCA, and ICA even uses SVD in the main phase of the computation, so called signal whitening. However, these methods belong to different fields of application; ICA is used

 ISBN: 978-1-61208-887-7

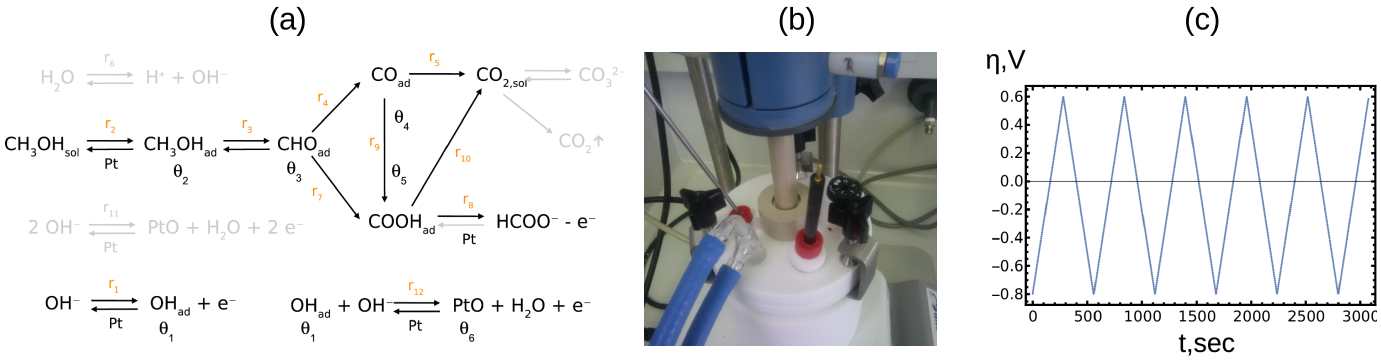


Fig. 1. (a) network of chemical reactions, (b) experimental setup, (c) saw-like voltage profile. Images from [1], [2].

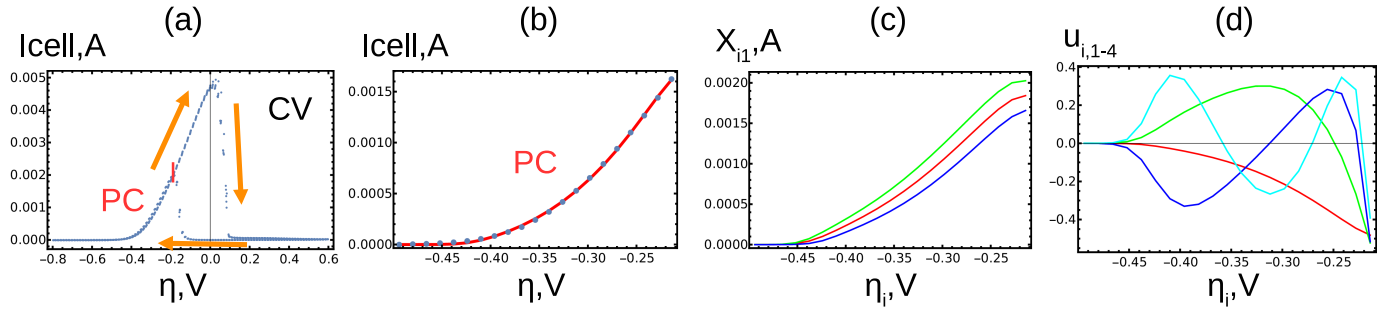


Fig. 2. (a) CV plot with selected PC part, (b) fit of PC curve by the model, (c) sensitivity analysis, (c) principal component analysis (for color coding see main text).

TABLE I
PARAMETER CENTRAL VALUES AND TRUST REGION OF LINEAR MODEL

p_j	0.949	-4.5	0.398	-0.563	4.72	-3.46	0.352	-0.101	1.2	-8.66	1.89	-1.08	-1.72	-7.82
dp_j	0.3	0.1	0.3	0.4	1.5	0.1	0.1	0.06	0.1	0.1	0.08	0.2	0.15	0.15

TABLE II
RESULTS OF PRINCIPAL COMPONENT ANALYSIS

λ_k, A	a_k	v_{jk}												
$8.12 \cdot 10^{-3}$	$2.12 \cdot 10^{-3}$	−0.531	0.014	0.461	−0.459	0.140	−0.013	−0.007	0	0	0	0	−0.522	0.032	−0.032
$1.21 \cdot 10^{-3}$	$1.43 \cdot 10^{-2}$	0.396	−0.159	−0.251	0.250	0.214	−0.010	−0.018	0	0	0.002	0	−0.769	0.167	−0.167
$2.71 \cdot 10^{-4}$	$6.36 \cdot 10^{-2}$	−0.078	0.639	−0.052	0.058	−0.402	0.065	−0.065	0	0	0.004	0	−0.052	0.451	−0.451
$1.24 \cdot 10^{-4}$	0.139	−0.066	−0.711	0.091	−0.074	−0.191	−0.058	0.066	0	0	−0.014	0	0.197	0.443	−0.443
$7.79 \cdot 10^{-6}$	2.21	0.305	−0.063	0.211	−0.123	−0.728	−0.290	0.277	0	−0.001	−0.113	0	−0.234	−0.206	0.206
$5.1 \cdot 10^{-6}$	3.38	0.546	0.214	0.540	−0.152	0.408	−0.204	0.205	0	0	−0.056	0	0.192	0.160	−0.160
$8.51 \cdot 10^{-7}$	$2.02 \cdot 10^{1}$	0.315	−0.066	0.148	−0.264	−0.163	0.336	−0.449	0.001	0.032	0.679	−0.003	−0.010	−0.033	0.033
$1.69 \cdot 10^{-7}$	$1.02 \cdot 10^{2}$	0.033	−0.083	0.508	0.459	−0.114	0.441	−0.349	−0.001	−0.029	−0.434	0.003	−0.033	−0.046	0.046
$2.52 \cdot 10^{-8}$	$6.84 \cdot 10^{2}$	0.240	−0.003	−0.304	−0.617	−0.003	0.439	−0.030	−0.003	−0.080	−0.518	0.012	0.021	0.013	−0.013
$7.81 \cdot 10^{-9}$	$2.21 \cdot 10^{3}$	0.047	−0.001	−0.075	−0.135	0.004	−0.582	−0.722	0.008	0.212	−0.257	−0.039	0.031	0.004	−0.004
$5.92 \cdot 10^{-10}$	$2.91 \cdot 10^{4}$	0	0	−0.002	−0.002	0	−0.165	−0.159	−0.037	−0.972	0.022	0.002	0.007	0	0
$9.35 \cdot 10^{-13}$	$1.84 \cdot 10^{7}$	0	0	0	0	0	0.024	0.024	0.485	−0.028	0	−0.874	0	0	0
$3.38 \cdot 10^{-13}$	$5.09 \cdot 10^{7}$	0	0	0	0	0	0.014	0.014	−0.874	0.028	0	−0.485	0	0	0
0	∞	0	0	0	0	0	0	0	0	0	0	0	0.707	0.707	

 ISBN: 978-1-61208-887-7

to separate independent sources in signal processing, while CCA is a method of non-linear dimensionality reduction. PCA and the underlying spectral decomposition are general statistical methods for data analysis. Here, we only need to perform a statistical analysis of the measurement errors, for which the standard PCA method is best suited.

Details of implementation: it is recommended in [10] to use the formula (3) when there is no independent estimate of measurement errors. In this case, the measurement error is obtained from the χ^2-criterion. Assuming the same error for all measurements, one has $\chi^2 = L_2^2/\epsilon^2$, in case of a good fit, $\chi^2 = N_{dof}$ is fulfilled; from here one can find ϵ. The geometric meaning of this definition is that for a good fit the deviations of the experimental points from the model can be considered appearing due to a random measurement error, and the standard deviation of such a variation characterizes this error. The term $N_{dof} = N_{pt} - N_{par}$ in the denominator of this definition, instead of the traditional $N_{pt} - 1$, takes into account the influence of the fit parameters, which take on a part of the experimental scatter.

The sensitivity matrix is determined using finite difference schemes of the form

$$X_{ij}^{+} = (f(\eta_i, p + dp_{(j)}) - f(\eta_i, p))/dp_j, \quad (8)$$

$$X_{ij}^{-} = (f(\eta_i, p) - f(\eta_i, p - dp_{(j)}))/dp_j, \quad (9)$$

$$X_{ij} = (f(\eta_i, p + dp_{(j)}) - f(\eta_i, p - dp_{(j)}))/(2dp_j), \quad (10)$$

where $X_{ij}^{\pm}$ – forward/backward, $X_{ij} = (X_{ij}^{+} + X_{ij}^{-})/2$ – central difference scheme. Here, $dp_{(j)} = (0, ..., dp_j, ..., 0)$ represents a vector with non-zero entry on j-th place. Such derivatives, for $j = 1$, are shown in Figure 2c, red line for central, green/blue lines for forward/backward schemes, respectively. More precisely, up to the next order, is the central difference scheme, which should be taken as the final answer. Other profiles are needed to evaluate the non-linearity of the function. Indeed, for linear functions, all these three profiles coincide, and their deviation from each other is a measure of non-linearity. In practice, we adjust the dp_j variation so that the curves deviate from each other by $\sim 20\%$. This analysis is performed for all j; the results are shown in Table I. Such variations define a box-like trust region where a linear model can be applied.

Another detail is the presence of failures in the definition of the model function. Since this definition uses the solution of a high degree polynomial system, the applied numerical procedures can lose solutions sometimes. Such failures are rare, estimated in $\sim 0.5\%$ cases. However, this is sufficient to destabilize the automatic minimization procedures, with the result that achieving the true minimum is not guaranteed. The methods described in [1] help in this situation, including finding the starting point manually and applying random search with discarding the failed cases. Automatic minimization algorithms were applied to the resulting starting points, which slightly improved the objective function. Finally, we made sure visually and by formal ϵ-criterion that the fit has a good quality. Since the failures also occur when using finite-difference schemes, some of the dp_j parameters required fine tuning to completely eliminate the failed cases.

The next surprise was the degeneration of the matrix $X^T X$, which makes the usage of the formulas (3), (4) impossible. The degeneration is seen in Table II, where the last row has a zero eigenvalue. The eigenvector for this value corresponds to the simultaneous variation of the parameters $\delta p_{13} = \delta p_{14}$. Formally, this direction corresponds to the infinite scatter a_k, meaning that it is impossible to measure the corresponding linear combination of constants. Indeed, a detailed analysis of the model given in [1] shows that such a variation corresponds to the exact symmetry of the system and does not change the observed values. In fact, the system depends only on the difference between the logarithmic parameters $p_{13} - p_{14}$, or, in the original notation, on the ratio of the corresponding reaction constants. This symmetry takes place only for a stationary system on the PC-part of the CV-curve. Dynamics can break this symmetry and make these constants individually measurable.

Further, in Table II, the first three lines correspond to the variation a_k that is located within the previously defined trust region of the linear model. The fourth line is also located in the trust region, with a tension. The following directions are outside of the trust region. Thus, the first four directions appear to be measured more or less accurately, while the rest of the directions have too large scatter. This conclusion is the main result of the PCA.

Note that the determination of the covariance and correlation matrices in this problem becomes meaningless. The point is not only that the sensitivity matrix is formally degenerate. There are many directions along which the error ellipsoid is strongly stretched, in the projections on axes of the initial parameters giving very large errors, strongly correlated between different parameters. It is PCA/SVD method that sheds light on the structure of solutions in the problem under consideration.

To improve the obtained result, in addition to considering the complete dynamic problem, other experiments can be included in the analysis. In particular, the experiments with variations of the volume concentrations of the main reagents $c_{1,2}$ can be used. These concentrations enter polynomially in the model [1], and accordingly extend the model response $I_{cell} = f(\eta, c_1, c_2, p)$ and the sensitivity matrix X. The reconstruction accuracy for such an extension can be analyzed using the general methodology described here.

Implementation in Mathematica: we use Mathematica [11] for the calculations described above. The `NSolve` method is used to solve a stationary polynomial system; for the obtained set of solutions, real roots are selected from the interval $\theta_i \in [0, 1]$. The system also has a spurious solution ($\theta_i = \delta_{i4}$, $I_{cell} = 0$), which also needs to be removed. As a result of numerical instabilities, with these selections, the roots sometimes disappear, leading to the aforementioned failures of the algorithm. To solve the dynamic system, `NDSolve` method is used, able to integrate both ODE and the partially reduced DAE system [2]. Systems are highly stiff and the integration algorithm also fails sometimes. In

 ISBN: 978-1-61208-887-7

the fitting algorithm [1], to select the starting point, the interactive configuration tool `Manipulate` is used. After that, we used automatic methods for minimizing the L_2-norm, local `FindMinimum` and global `NMinimize`. In this work, we have used `NonlinearModelFit`, which provides a convenient interface to the same optimization methods. The differentiation algorithm requires setting up finite-difference schemes, which can be passed to the fitting method via the `Gradient` option. Further, the calculation of the covariance and correlation matrix turns out to be impossible due to the degenerations described above. At this point, the standard computation should be replaced with PCA using the available method `SingularValueDecomposition`.

IV. Conclusion and Future Work

We considered an algorithm for reconstructing the reaction constants from the experimentally measured polarization curve for the electrochemical alkaline methanol oxidation process. Our approach combines statistical and principal component analysis. We define formal criteria for reconstruction accuracy based on the estimate of the trust region for the linearized model. As a result of the analysis, it turned out that the described experiment does not make it possible to determine precisely all 14 reaction constants, but only 4 their certain linear combinations. Of the remaining orthogonal combinations, one corresponds to the symmetry of the stationary system and is fundamentally indeterminate in the described experiment. The remaining 9 combinations have insufficient reconstruction accuracy. To improve this result, other experiments should be involved in the analysis, including fully dynamic cyclic voltammetry and variations in the concentration of the main reagents. We are going to expand the developed methodology for additional experiments elsewhere.

References

[1] T. Clees et al., “Mathematical modeling of alkaline methanol oxidation for design of efficient fuel cells”, Advances in Intelligent Systems and Computing, vol 947, 2020, pp. 181-195.

[2] T. Clees et al., “Parameter identification and model reduction in the design of alkaline methanol fuel cells”, Int. J. On Advances in Systems and Measurements, vol. 13, 2020, pp. 94-106.

[3] T. Haisch et al., “The origin of the hysteresis in cyclic voltammetric response of alkaline methanol electrooxidation”, Physical Chemistry Chemical Physics, vol. 22, 2020, pp. 16648-16654.

[4] T. Clees et al., “Electrochemical impedance spectroscopy of alkaline methanol oxidation”, in Proc. INFOCOMP 2017, pp. 46-51, Pub. IARIA 2017.

[5] U. Krewer, T. Vidakovic-Koch, and L. Rihko-Struckmann, “Electrochemical oxidation of carbon-containing fuels and their dynamics in low-temperature fuel cells”, ChemPhysChem, vol. 12, 2011, pp. 2518-2544.

[6] U. Krewer, M. Christov, T. Vidakovic, and K. Sundmacher, “Impedance spectroscopic analysis of the electrochemical methanol oxidation kinetics”, J. Electroanalytical Chem., vol. 589, 2006, pp. 148-159.

[7] B. Beden, F. Kardigan, C. Lamy, and J. M. Leger, “Oxidation of methanol on a platinum electrode in alkaline medium: effect of metal ad-atoms on the electrocatalytic activity”, J. Electroanalytical Chem., vol. 142, 1982, pp. 171-190.

[8] F. Ciucci, “Revisiting parameter identification in electrochemical impedance spectroscopy: Weighted least squares and optimal experimental design”, Electrochimica Acta, vol. 87, 2013, pp. 532-545.

[9] A. J. Bard and L. R. Faulkner, Electrochemical Methods: Fundamentals and Applications, Wiley 2000.

[10] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, Numerical Recipes in C, Cambridge University Press 1992.

[11] Mathematica 12, Reference Manual, http://reference.wolfram.com, [retrieved: August, 2021]

 ISBN: 978-1-61208-887-7

Towards Demystifying Transformations of Tchaikovsky's Children's Album with Support of Computational Models: Problem Conceptualization

Evgeny Pyshkin
University of Aizu

Tsuruga, Ikki-Machi, Aizu-Wakamatsu, Fukushima, 965-8580, Japan
Email: pyshe@u-aizu.ac.jp

***Abstract*—Though the studies of rich metaphors hidden in the musical compositions lay mostly in scope of art and musicology, there is still large space for formal methods based on mathematical models and computer technology that can be helpful in discovering complementary insights to how the composition is structured, what are its relationships to the precursors' works, and how it affects the later works of the same or other authors. Our idea is to investigate how computational models can enhance musicology research on music style identification and comparative analysis using the case study of Tchaikovsky's Children's Album.**

Keywords–Musicology; music information retrieval; human-centric computing; similarity; music modeling.

I. INTRODUCTION

In my recent talk on Jan 6th, 2021, in the University of Aizu "Tchaikovsky. Children's (?) Album: Time, Metaphors, Rediscoveries" [1], I discussed the phenomenon of Piotr Tchaikovsky's "Children's Album" for piano solo (Op. 39) [2]. This masterpiece was composed and published as far as in 1878, but today it still remains one of constant topics of interest for researchers [3][4]. Though the study of rich metaphors hidden in the pieces thought to be for children rather lies in the scope of art and musicology, there is still a large research space for formal methods based on mathematical and computational models, which may give additional insights into our understanding of the structure and organization of the whole work, its relationships to precursors (such as "43 Clavierstücke für die Jugend" by Robert Schumann [5]), as well as the reasons for significant differences between the original manuscript and the first published edition. Surprisingly, in musicology literature, one of the first careful studies of transformations between the manuscript and the published edition can be found in the early 1990s only; thus, more than 100 years after the whole work was completed [6][7]. These studies mostly remain in scope of music and art theory, with almost no involvement of machine learning approaches. Today, it is commonly not disputed that computer science and artificial intelligence may contribute to musicology research on music style identification and comparative analysis. In this study, we try to discover appropriate formal models that would enhance the analysis and understanding of Tchaikovsky's "Children's Album", which can be considered as a very good example of applied human-centric computing research in the frame of art and humanities, where solutions cannot be designed within a certain context only, but require intensive cross-disciplinary efforts so as to bridge the communities working in different contexts and using different vocabularies [8].

The remaining text is organized as follows. Section II provides a brief review of state-of-the-art research on linking musicology and computerized analysis of music compositions with a particular attention to the case study of Tchaikovsky's "Children's Album". Section III sketches a number of promising models that may contribute to music stylistic similarity recognition and evaluation aimed at bridging the gaps between musicology studies and computer models.

II. RELATED WORK

According to Nattiez, music is a symbolic fact characterized by the complex configuration of interpretants [9]. In music, we use various connected but independent models including letter-based notations, such as Helmholz or scientific pitch notation (that can be considered as simple syntax based language constructions), complex symbolic notations in the form of graphic note scores ranging from hardly formalizable ancient models, such as Znamenny chant, at one pole, or relatively strict Mensural notation, at another pole, up to modern sheet music (based on many rules but giving some freedom to support the individual styles of composers), piano roll notation, tablature, MIDI representations, as well as audio signals and even spectral models, such as acoustic fingerprints. The great variety of models used for music representation is one of reasons why music provides an interesting and complex use case for experimenting with information retrieval, object recognition and classification algorithms. Music representation complexity can be explained by the presence of two arrays of elements and relationships, where the first one corresponds to the elements that can be treated mathematically (pitch, rhythm, or harmony), while the second one includes non-mathematical elements such as tension, expectancy, and emotion [10].

A. Bridging the gap between pure musicology and applied human-centric computer technology

Current approaches to music similarity evaluation (including our own work on melody extraction and similarity estimation using Earth Mover's Distance algorithms [11]) mostly target the searching and retrieval systems including well-known apps, such as Shazam [12], without a perfect fit to the problems of stylistic similarity evaluation. From this point of view, models of functional representation of music harmony and harmonic similarity estimation [13] seem to be more adequate to the problem of style identification. Indeed, usually, listeners can recognize similarity of compositions because of their harmonic similarity (see Figure 1). However, it does not immediately lead us to clearly conclude about the composition's stylistic resemblances or dependencies. Even

 ISBN: 978-1-61208-887-7

harmonic equivalence may not be enough to recognize the melody, as demonstrated in [14] and later analyzed in [15] in the experiments with melodies distorted by substituting the note octave by randomly selected ones within three octaves: every note in the sequence keeps its position on the scale, but the tune varies over a three-octave range (similarly to an example of such distortion shown in Figure 2).

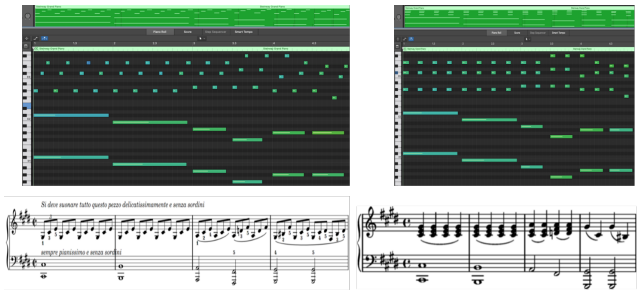


Figure 1. Harmony resemblance between Beethoven's Moonlight Sonata and its variation

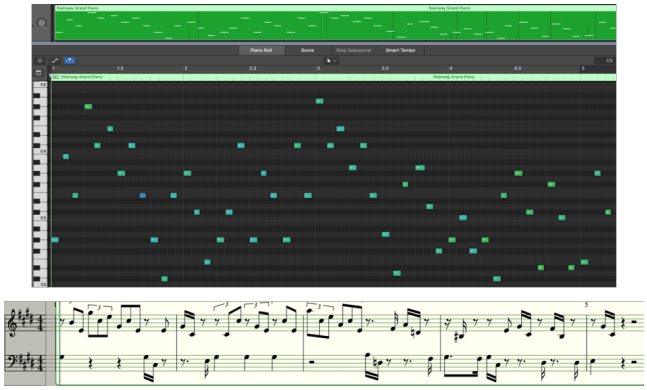


Figure 2. Distorted note sequence of Beethoven's Moonlight Sonata with keeping harmonic equivalence

Harmonic functions were core elements of SPEAC music representation system developed by Cope [16][17], which is an implementation of augmented transition network, a finite-state automaton with recursive succession rules between music subphrases allowing for logical syntax substitutions [18]. Cope's SPEAC system is based on a hierarchical representation of the structure of music composition in nested contexts beginning from notes and chords up to chapters and parts (see Figure 3 (a)). Five identifiers contributing to SPEAC acronym stand for statement S, preparation P, extension E, antecedent A, and consequent C, all of which are kinds of abstractions assigned to groups of notes "depending on levels of tension between intervals, metrical placement, and agogic emphasis, measured both in the preceding and following groups" [18]. Succession rules defined by Cope limit possible transitions between the SPEAC states (see Figure 3 (b)). Therefore, SPEAC progressions are like genome sequences using SPEAC identifiers as bases enforced by harmonic tension weights and hierarchical relationships between progressions at different levels. Modeling music structure using SPEAC-analysis can be a promising approach to recognize music style similarity through SPEAC progression similarity as well as with the help of comparison between the corresponding graphs, specifically with respect to recent SPEAC-analysis implementations available as libraries in universal languages, such as Python [19].

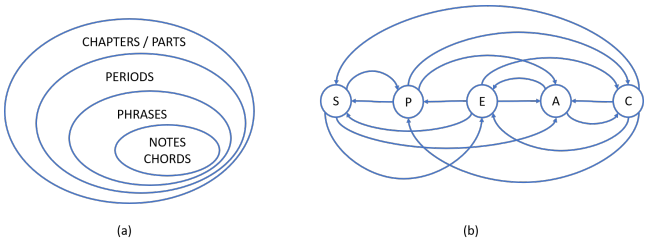


Figure 3. Progression bases in SPEAC system by David Cope.

Due to a large number of applications of using deep neural networks for object recognition and classification (especially for image recognition, including such subjective trait as image aesthetics), machine learning approaches and recurrent neural networks may be promising for music style identification, classification and analysis. Though, in contrast to a variety of works on computer music generation, we argue that the main challenge is not to teach AI to create art objects, but to be able to help us in perceiving objects created by humans [20].

B. Renditions and Implications of "Children's Album"

Since both Tchaikovsky and Schumann belong to the romantic tradition rooted in part of leitmotif music by Beethoven and Wagner, on the one hand, and in the new music language of Glinka, Chopin and Liszt, on the other hand, certain harmony and music development similarity surely exists in their works. According to the network of influences on classical composers originally described by Smith and Georges et al. in the original Classical Music Navigator [21], Schumann is one of composers who greatly influenced Tchaikovsky (along with Balakirev, Beethoven, Chopin, Delibes and others) as shown in Figure 4.

However, admitting Schumann's influence to Tchaikovsky does not lead us to automatically judge the "Children's Album" as an imitation of Schumann's pieces for the young (also with long history of editions but rather few scholarly studies [23]) even though Tchaikovsky claimed it explicitly in the subtitle for the published edition "24 simple pieces for children like Schumann" (but not in the manuscript! [24]). What if this subtle (?) subtitle is a kind of hint that Tchaikovsky gave us? Like saying: "Well, it is definitely not "like Schumann"! Should you then believe in the appropriateness of all made transformations?" These transformations (see Figure 5) destroy the structure of the album as an indissociable whole, and deform the micro-cycles existing in the manuscripts (where the Doll cycle is the clearest case), as well as evident harmonic links, for instance, between the first and the last pieces in the manuscript, "Morning prayer" and "The hurdy-gurdy man is singing", respectively.

An idea that changes in the order of compositions between the manuscript and first published edition were mistakenly introduced by the publisher could not be accepted as convincing enough: indeed, Tchaikovsky approved this version. Nekhaeva suggested that these transformations can be considered as a "gesture of the composer, a natural desire to overcome the temporary barrier and directly appeal to future generations of musicians" [4]. This opinion supports an existing hypothesis

 ISBN: 978-1-61208-887-7

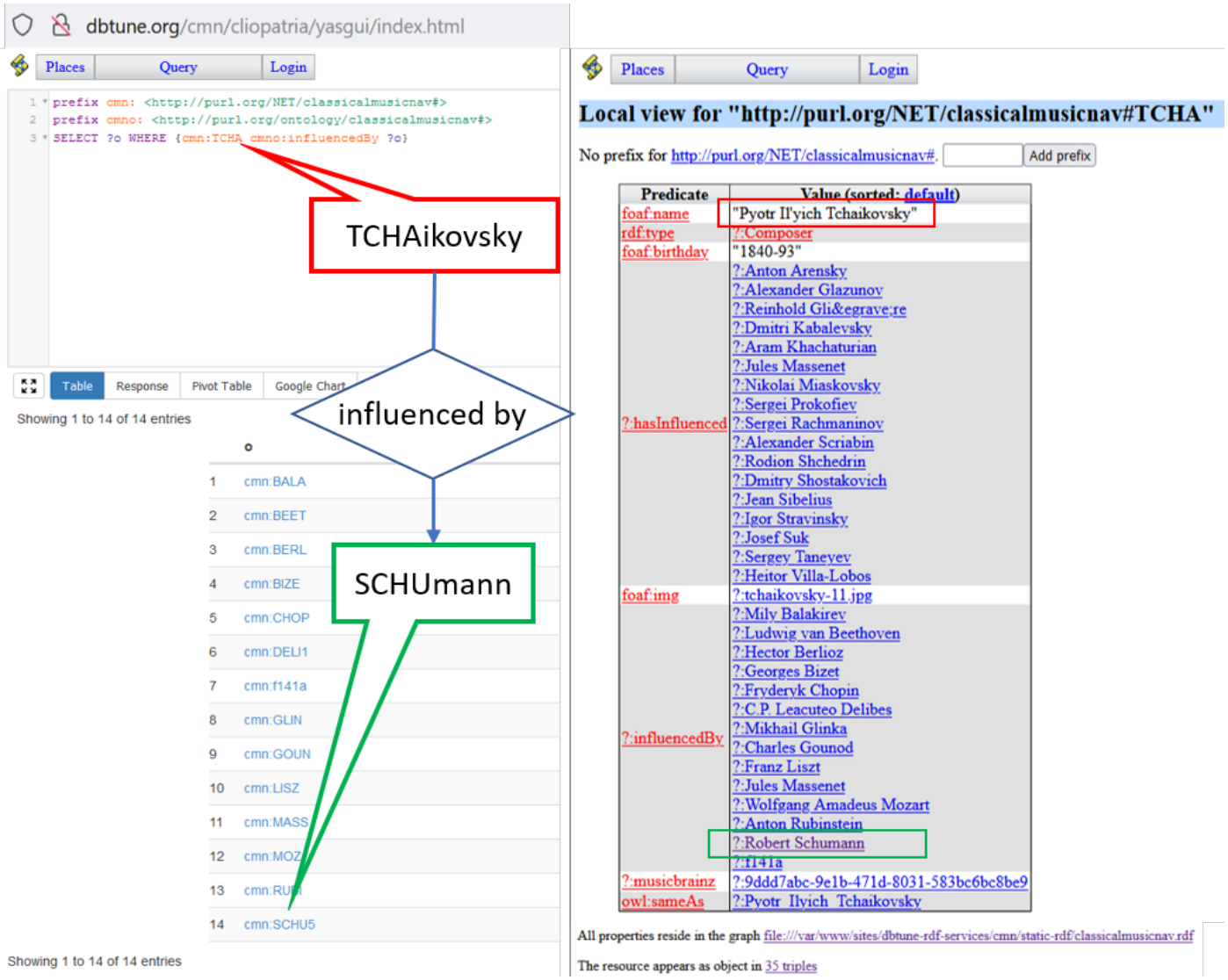


Figure 4. “Influenced-by” relationships for Tchaikovsky (as in Classical Music Navigator [22]

claiming that Tchaikovsky probably preferred to hide some metaphors so that they are not so explicitly exposed as in the manuscript. From the perspective of musicology, we could not expect to find a final answer (and perhaps it is not needed). Instead, a possibility to incorporate formal computational approaches into informal art discourse can, produce a number of important additional insights for better understanding of genesis of one of Tchaikovsky’s masterpieces in piano music.

C. Dataset Issues

In the process of study, we need to investigate, what are the suitable computational approaches that may contribute to style identification. Because of the subjectivity of style attribution and style dependency analysis, a possibility to construct and assess different computational models should be considered. It may be that particular models can contribute to particular characteristics of music style recognition.

We also need to define a dataset with the selection of compositions including the following components:

- 24 piano compositions from the “Children’s Album”;
- Selection of characteristic piano compositions by Schumann, including those from Op. 68;
- Selection of compositions with expected high degree of style similarity, which were attributed by their authors as imitations, such as piano works of Liszt, Chopin, Schumann (referential dataset);
- Selection of other characteristic compositions (e.g., by Tchaikovsky), where style similarity was reported by musicology experts (referential dataset). The studies [25][26] can provide information for selection of relevant referential datasets.

III. Promising Approaches for Further Studies

There is a number of works contributing to music analysis based on audio signal processing. Detection music file similarity based on tonality, tempo and chord progression similarity (that can be extracted from sound files using signal processing algorithms as demonstrated in [27]) is very helpful to improve

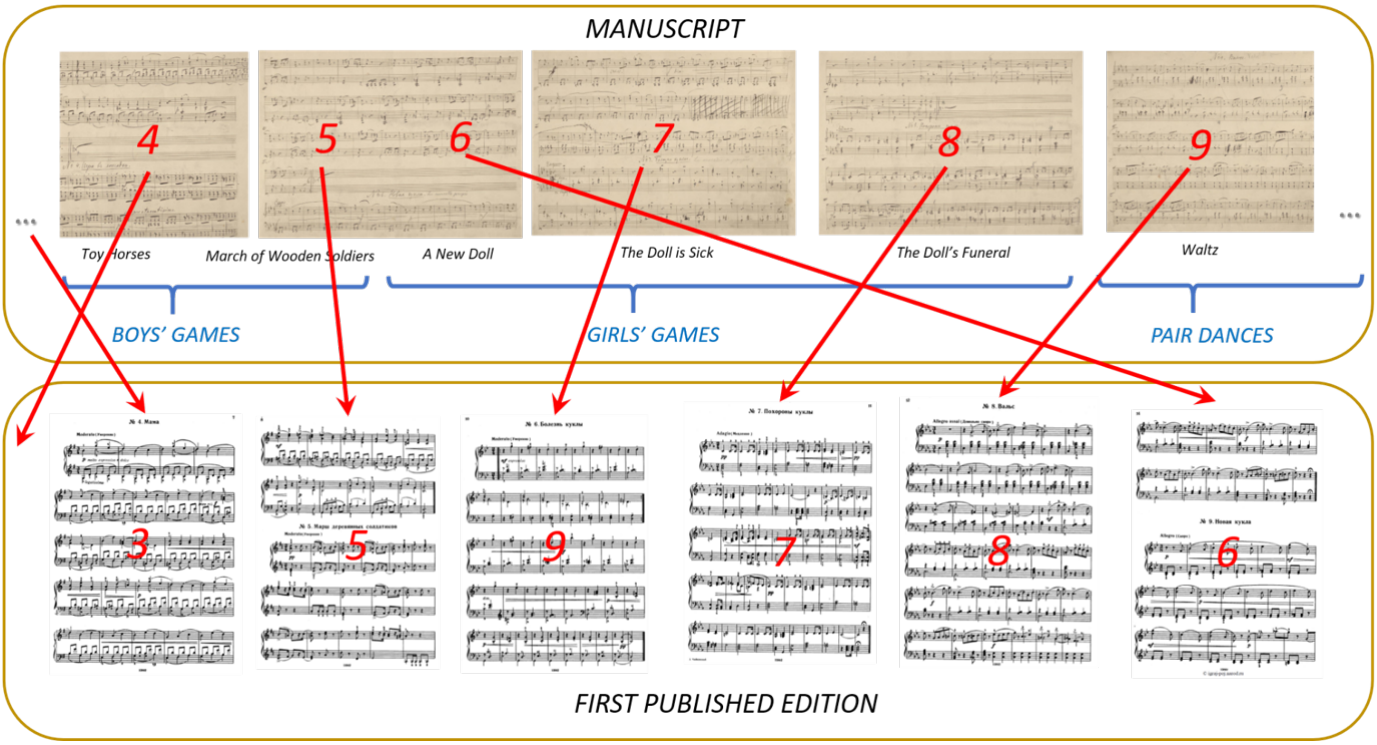


Figure 5. "Children's Album": Original structure is destroyed in the published version.

algorithms of music retrieval, but may not be enough to judge about stylistic and harmonic similarity or about the presence of transformed citations where the key, tempo and melody can be significantly modified compared to the original but keeping almost intangible traits of similarity, still perceptible by an experienced ear.Great opportunities provided by signal processing algorithms enforced by hierarchical semi-Markov models of music enable automatic music transcription for given audio signals [28]. Though these works are naturally closely related to music similarity analysis, the findings could not be directly applicable to the problem of developing computational models of style similarity, which remains challenging even if the note score is available.

Similarity detection based on note sequences (e.g., in [29]) can give interesting insights into the genesis of music styles, but does not help much in solving specific problems of influence assessment, where study of exact or slightly transformed note sequences may be insufficient. However, the idea of grouping compositions based on weaker traits of similarity in their themes and sub-themes [30] can be promising.

With respect to studies on analysis of acoustic spectral fingerprints for unique identification of the music fragments (e.g., according to the algorithms described in [31][32]), comparison between such fingerprints can give one component for similarity analysis. Figure 6 displays an example of piano composition spectral representation constructed using the online tool [33].

Among other interesting works relevant to this study, there are studies on approaches using deep neural networks for object recognition applied to the case of music for a variety of adjacent problems, including music genre classification [34], content-based music recommendation [35], music style modeling [36], deep leaning-based music generation [37], and style-

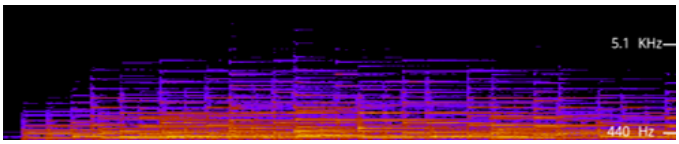


Figure 6. Sample spectrogram of "A New Doll" (Op. 39, orig. No. 6): first 30 measures recorded by Evgeny Pyshkin at Yamaha YDP-144

specific based music composition [38]. In addition, considering music as a semi-chaotic natural process with recurrencies and irregular cyclicities analyzed and visualized with the help of recurrence plots [39] (similar to spoken pitch as we did in our prosody visualization research [40]).

IV. Conclusion

In this study, the problem of music style identification is sketched via a brief analysis of computational models and technical solutions that may be helpful to musicologists in their research on genesis and implications of musical compositions with an example of exploring the links between Tchaikovsky's "Children Album" and Schumann's "Album für die Jugend". With the help of computer technology we can discover more findings to support meaningful hypotheses about the possible reasons explaining significant discrepancies between Tchaikovsky's manuscript and the following editions of "Children's Album".

Naturally, the outcomes from such compact joint musicology and computer science studies can naturally address the broader scope of research on music style understanding, modeling, and recognition for the benefit of both computer technology and humanities so as to provide interesting use

 ISBN: 978-1-61208-887-7

cases for AI applications as well as "a further strand of evidence for systematic musicology to exploit" as nicely formulated by Collins in [41].

ACKNOWLEDGEMENT

Many thanks to Natalia Bogach, John Blake, and Andrey Kuznetsov for their helpful suggestions during our discussions of this study. The work is supported by the University of Aizu Research Funding.

REFERENCES

[1] "International talk: Piano concert talk "Tchaikovsky" was held!" 2021, retrieved: Aug, 2021. [Online]. Available: https://www.u-aizu.ac.jp/osip/en/information/post-206.html

[2] P. Tchaikovsky, Children's Album. Op 39. Yurgenson, 1878.

[3] A. Lazanchina, "Fenomen detstva v fortepiannykh tsiklakh R. Shumana i P. Tchaikovskogo (The phenomenon of childhood in piano cycles by R. Schumann and P. Tchaikovsky)," Transactions of Russian Academy of Science Samara Research Center, 2015, pp. 1224–1227, (In Russian).

[4] I. Nekhaeva, "The modern measurement of Tchaikovsky (Definition experience of the "modernity" concept on example of the "Children's Album" by PI Tchaikovsky)," Tomsk State University Journal of Cultural Studies and Art History, vol. 30, 2018, pp. 146–147, retrieved: Aug, 2021. [Online]. Available: http://case.asu.ru/files/form_312-31545.pdf#page=146

[5] R. Schumann, 43 Piano Pieces for the Youth. Op 68 (Orig. Title in German: 43 Clavierstücke für die Jugend). Schuberth and Co., 1848.

[6] A. Kandinskiy-Rybnikov and M. Mesropova, "Vremena goda i Detskiy albom Tchaikovskogo: Tsyklichnost i problemy ispolneniya (Tchaikovsky's The Seasons and Children's Album: Cyclicity and performing problems)," in Tchaikovsky: Voprosy istorii, teorii i ispolnitelstva (Tchaikovsky. Questions on history, theory, and performing). Moscow Conservatory, 1990, pp. 120–137, (In Russian).

[7] ——, "O ne opublikovannoy P.I. Tchaikovskim pervoy redaktsii "Detskogo alboma" (On an unpublished first edition of the "Children's album" by P.I. Tchaikovsky)," in Voprosy muzykalnoy pedagogiki (Issues of Musical Pedagogy). Muzyka, 1997, pp. 138–161, (In Russian).

[8] E. Pyshkin and J. Blake, "A metaphoric bridge: Understanding software engineering education through literature and fine arts, society," Society. Communication. Education, vol. 11, no. 3, 2020, pp. 59–77.

[9] J.-J. Nattiez, Music and discourse: Toward a semiology of music. Princeton University Press, 1990.

[10] R. B. Dannenberg, "Music representation issues, techniques, and systems," Computer Music Journal, vol. 17, no. 3, 1993, pp. 20–30.

[11] A. Kuznetsov and E. Pyshkin, "Searching for music: from melodies in mind to the resources on the web," in proceedings of the 13th international conference on humans and computers, 2010, pp. 152–158.

[12] A. Wang, "The Shazam music recognition service," Communications of the ACM, vol. 49, no. 8, 2006, pp. 44–48.

[13] J. P. Magalhaes and W. B. de Haas, "Functional modelling of musical harmony: an experience report," ACM SIGPLAN Notices, vol. 46, no. 9, 2011, pp. 156–162.

[14] D. Deutsch, "Octave generalization and tune recognition," Perception & Psychophysics, vol. 11, no. 6, 1972, pp. 411–412.

[15] W. R. Thurlow and W. P. Erchul, "Judged similarity in pitch of octave multiples," Perception & Psychophysics, vol. 22, no. 2, 1977, pp. 177–182.

[16] D. Cope, "Experiments in musical intelligence (EMI): Non-linear linguistic-based composition," Journal of New Music Research, vol. 18, no. 1-2, 1989, pp. 117–139.

[17] ——, Computer models of musical creativity. Mit Press Cambridge, 2005.

[18] P. da Silva, "David Cope and experiments in musical intelligence," 2003.

[19] N. Golzitsky, "SPEAC-analysis Python library," 2021, retrieved: Jun, 2021. [Online]. Available: https://github.com/GolzitskyNikolay/SPEAC-analysis

[20] A. Kuznetsov and E. Pyshkin, "Function-based and circuit-based symbolic music representation, or back to Beethoven," in Proceedings of the 2012 Joint International Conference on Human-Centered Computer Environments, 2012, pp. 171–177.

[21] C. H. Smith and P. Georges, "Composer similarities through "The Classical Music Navigator": Similarity inference from composer influences," Empirical Studies of the Arts, vol. 32, no. 2, 2014, pp. 205–229.

[22] C. H. S. Smith, "Classical music navigator," 2014, retrieved: Jun, 2021. [Online]. Available: http://dbtune.org/cmn/

[23] B. R. Appel, Actually, taken directly from family life: Robert Schumann's Album für die Jugend. Princeton University Press, 2014, pp. 171–202. [Online]. Available: https://doi.org/10.1515/9781400863860.171

[24] "Tchaikovsky: Otkrytyi mir. Detskiy albom (Tchaikovsky: Open world. Children's Album," 2015, retrieved: Aug, 2021 (In Russian). [Online]. Available: https://www.culture.ru/catalog/tchaikovsky/ru/item/archiv/detskiy-albom-24-legkih-pesy

[25] P. Georges, "Western classical music development: a statistical analysis of composers similarity, differentiation and evolution," Scientometrics, vol. 112, no. 1, 2017, pp. 21–53.

[26] P. Georges and N. Nguyen, "Visualizing music similarity: clustering and mapping 500 classical music composers," Scientometrics, vol. 120, no. 3, 2019, pp. 975–1003.

[27] M. Thomas, M. Jothish, N. Thomas, S. G. Koolagudi, and Y. S. Murthy, "Detection of similarity in music files using signal level analysis," in 2016 IEEE Region 10 Conference (TENCON). IEEE, 2016, pp. 1650–1654.

[28] R. Nishikimi, E. Nakamura, M. Goto, K. Itoyama, and K. Yoshii, "Bayesian singing transcription based on a hierarchical generative model of keys, musical notes, and f0 trajectories," IEEE/ACM Transactions on Audio, Speech, and Language Processing, vol. 28, 2020, pp. 1678–1691.

[29] S. Cunningham, V. Grout, and H. Bergen, "Mozart to Metallica: A comparison of musical sequences and similarities." in CAINE, 2005, pp. 332–339.

[30] B. Laskowska and M. Kamola, "Grouping compositions based on similarity of music themes," PloS one, vol. 15, no. 10, 2020, p. e0240443.

[31] W. Hatch, "A quick review of audio fingerprinting," 2003, retrieved: Jun, 2021. [Online]. Available: http://www.music.mcgill.ca/wes/docs/finger2.pdf

[32] P. Cano, E. Batle, T. Kalker, and J. Haitsma, "A review of algorithms for audio fingerprinting," in 2002 IEEE Workshop on Multimedia Signal Processing. IEEE, 2002, pp. 169–173.

[33] "Spectrum analyzer," retrieved: Aug, 2021. [Online]. Available: https://academo.org/demos/spectrum-analyzer/

[34] Y. M. Costa, L. S. Oliveira, and C. N. Silla Jr, "An evaluation of convolutional neural networks for music classification using spectrograms," Applied soft computing, vol. 52, 2017, pp. 28–38.

[35] A. Van Den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in Neural Information Processing Systems Conference (NIPS 2013), vol. 26. Neural Information Processing Systems Foundation (NIPS), 2013, pp. 1–9.

[36] S. Dai, Z. Zhang, and G. G. Xia, "Music style transfer: A position paper," arXiv preprint arXiv:1803.06841, 2018.

[37] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, "Deep learning techniques for music generation–a survey," arXiv preprint arXiv:1709.01620, 2017.

[38] C. Jin, Y. Tie, Y. Bai, X. Lv, and S. Liu, "A style-specific music composition neural network," Neural Processing Letters, vol. 52, 2020, pp. 1893–1912.

[39] J.-P. Eckmann, S. O. Kamphorst, and D. Ruelle, "Recurrence plots of dynamical systems," World Scientific Series on Nonlinear Science Series A, vol. 16, 1995, pp. 441–446.

[40] N. Bogach et al., "Speech processing for language learning: A practical approach to computer-assisted pronunciation teaching," Electronics, vol. 10, no. 3, 2021, p. 235.

[41] N. Collins, "Computational analysis of musical influence: A musicological case study using MIR tools." in ISMIR, 2010, pp. 177–182.

 ISBN: 978-1-61208-887-7

A Novel Application of Machine Learning to a New SEM Silicate Mineral Dataset

Benjamin Parfitt
Principal Research Scientist
Reiform
Washington, DC, USA
ben@reiform.com

Robert M. Welch
Department of Earth and Planetary Sciences
Harvard University
Cambridge, MA, USA
rwelch@g.harvard.edu

***Abstract*—Machine Learning (ML) continues to find applications in the geosciences, specifically in the classification of minerals from spectral or elemental data. We begin by exploring the use of four different methods for classification of elemental mineral samples from Scanning Electron Microscopy (SEM) and microprobe analysis in terms of structure, group, and subgroup. We create the most extensive silicate mineral group and subgroup classifiers available to the best of our knowledge, and achieve precision and recall values as high as the current state-of-the-art methods, which cover fewer groups and subgroups. Finally, we attempt to leverage the knowledge of structural families to improve classification performance on mineral groups, and reapply this process to improve performance on mineral subgroups. The train, test, validation split of data used in this paper will be posted online, along with the code and a webpage called *MINdicator* where anyone can use the new models easily.**

***Index Terms*—machine learning, ensemble learning, mineralogy, silicates.**

I. Introduction

Mineralogists group naturally occurring crystalline solids, called minerals, into seven different families. One of these is the silicate family, which makes up ≈90% of all minerals in the earth's crust and is critical for rock formation [1]. Determining what silicate minerals are present in a sample is therefore crucial in determining rock forming processes, histories of metamorphism, and tectonic history, and has countless other applications [2], [3].

Geoscientists currently make mineral predictions by using petrographic microscopes and spectroscopy methods [4]. These methods have been invaluable for the field of mineralogy since its inception in the late 1800s. However, there are drawbacks. The process requires an in-depth knowledge of mineralogy to derive an accurate classification. Even with a high degree of mineralogical knowledge, the identification is not always reproducible. As the process stands today, the time to correctly identify a group of samples is directly related to how many samples one has since there is not a reliable, in-depth, automated method of classifying a wide range of silicate minerals through chemical analyses [5]. The identifications of mineral family, group, and subgroup are based on real elemental data, and are used to train our data-driven ML models. This type of data-driven model provides a high degree of automation and reproducibility, and a path to parallelize the process of silicate mineral identification.

Machine learning techniques offer an expressway between data collection and data analysis that is normally time consuming and requires a high degree of domain knowledge concerning mineral identification. It has been shown that machine learning can expedite the process of mineral identification from Raman Spectroscopy [6], X-ray fluorescence (XRF) [5], and Electron Microprobe Analysis (EMPA) analyses [7]. Additionally, some methods have used deep learning and Convolutional Neural Networks (CNN) to identify minerals using standard RGB images [8] and hardness measurements [9]. Machine learning provides rapid, reproducible methods for determining the mineral in question. Other methods also use random forest classifiers and ensemble learning to improve mineral identification within rock cross sections by using spectral signatures from SEM and hyperspectral analysis of rock cross-sections [10], [11].

Existing methods that operate on weight oxide data have yet to utilize a large dataset, which is necessary to capture the wide range of variability in the silicate group. We have developed several methods to determine structure, group, and subgroup classification that are state-of-the-art because we classify more classes with higher accuracies than previous methods, while utilizing a more robust and challenging dataset. To our knowledge, we are the first to classify family structure. Additionally, we classify 12 more groups and 7 more subgroups than any previous method using SEM data [5]–[7]. We achieve macro F1-scores (Section II-B) of 98.4%, 92.3%, and 90.7% on structural families, groups, and subgroups, respectively. Finally, we investigate the effectiveness of utilizing the explicit divisions of structures and groups in order to improve classification quality of groups and subgroups, respectively. These tests show promising results that warrant further investigation.

The remainder of the paper is structured as follows: Section II contains background information on mineralogy and machine learning, Section III contains an overview of related work, Section IV contains the details of our dataset, Section V describes our methods for classification, Section VI contains results, Section VII contains a discussion, Section VIII our methods for classification using the explicit divisions of structure and group, Section IX contains the results of that effort, Section X provides additional discussion, and Section XI provides concluding remarks.

 ISBN: 978-1-61208-887-7

II. Background

A. Mineralogy

Minerals are naturally occurring crystalline solids with a repeatable pattern. Due to differences in chemistry and crystal structure, minerals are broken into seven families: silicates, oxides, sulfates, sulfides, carbonates, native elements, and halides [1]. The silicate family of minerals contains six structural groups which are also denoted as subfamilies. These are nesosilicates, cyclosilicates, sorosilicates, inosilicates, phyllosilicates, and tectosilicates/framework silicates, all of which have lattice differences [1]. Differences in crystal lattice configurations will alter the type of ions that can perform solid solution in a mineral, which alters the number of different elements present within a mineral group. For instance, clay minerals (a phyllosilicate) allow for a greater degree of solid solution than wollastonite (an inosilicate) due to lattice differences [1], [12]. Though silicate minerals cover a wide range of chemical variability, the structure of a mineral family, group, or subgroup creates a unique chemical "fingerprint" due to physical chemistry [13]. For example, this means that, in theory, the amphibole group is chemically unique from quartz or chlorite [4], [14], [15]. This creates a "fingerprint"; if an individual knows the chemistry, they can forecast the structure [13].

Geologists commonly determine mineral chemistry by a spectroscopy method, typically either Raman, SEM, Energy-Dispersive Spectrum X-Ray Fluorescence (EDS/XRF), or EMPA. These methods work by subjecting a mineralogical sample to a high intensity electromagnetic radiation source or electron beam. The released energies are then reported as weight percent oxide counts.

B. Machine Learning

Decision trees [16] are a simple tree structure in which training data are split at each tree node on some learned condition until only one class is remaining. The resulting tree is then used to classify unknown data samples. Extremely randomized trees (Extra Trees) [17] is an ensemble learning method that randomizes the data splitting at the nodes within the decision trees used. Both of these tree methods are computationally efficient. The K-Nearest-Neighbors (KNN) classifier finds the k closest known samples to some unknown sample (by euclidean distance), and uses the classes of the known samples to predict the unknown class. A drawback of this method is sensitivity to high dimensional spaces, both in accuracy and efficiency, making it more computationally expensive than the tree methods. The final method employed is the Support Vector Machine (SVM) [18], which finds a hyperplane that provides maximal separation for two sets of data. Unseen data points are then plotted and classified by their position relative to the hyperplane. The data can first be operated on by a kernel to map it to a different space, allowing non-separable data to be separated. In order to apply this method to our multiclass problem space, the problem is converted into several "one-vs-rest" binary classification problems.

One weakness of many publications in ML is the use of evaluation metrics that do not fully report the results [6]. Some background terminology: *True Positive* (TP) are all the correctly labeled samples from the class of interest; *False Positives* (FP) are samples labeled as the class of interest, but actually in another; *False Negatives* (FN) are samples labeled as another class, but actually in the class of interest. Often, the only metric used is *accuracy* over all points. Computed over all points, this is simply (all correct points)/(all points). However, if 90% of the points are from class A, and the classifier labels all data as A, then an accuracy of 90% will be reached, which is misleading. This is why using metrics such as recall and precision is important. *Recall* is defined as (TP/(TP+FN)). If we measure using per class metrics from our previous example, class A will have 100% recall, and all other classes will have 0% recall. At this point the average recall can be calculated, providing a realistic picture of the results (at best 50%). Another metric that is important is *precision*, defined (TP/(TP+FP)). However, it is often more cumbersome to report the pair of metrics for each class, so the *F1-score*, defined as (2×Precision×Recall)/(Precision + Recall), is often used as a comprehensive metric. The *macro F1-score*, which is the average F1-score across all classes, is commonly used to evaluate models. Note that this is unaffected by the imbalance of data in the evaluated dataset. One final metric that is employed in the later sections of our evaluation is the *average top-3 recall*, (top-3 recall, for brevity). To calculate this metric, the probability vector V is taken from each classifier. Normally, the position of the largest value in V is used as the class identified, but instead the three largest values are observed and if any of those correspond to the correct class, the vector is considered "correct". Then, the average recall over all classes is computed as normal.

When large datasets are at hand in ML settings, the data are broken into three sets: train, validation, and test. The train set is used for training the model. The validation set is used to validate the model and tune the choices of various hyperparameters. The test set is used to test, or evaluate, the performance of the final trained model. By using these three sets it is ensured that the test data is not used to tune the hyperparameters, which would create a model that is tuned specifically for the test data and that creates misleading results.

III. Related Work

Several ML methods, namely KNN, SVM, Extremely Random Trees, Weighted Neighbors [19], and CNNs, are reviewed as methods for mineral identification from Raman spectra [6]. The primary dataset used consisted of 3950 Raman spectra samples from 1214 mineral species, with some rather large class imbalances. A novel ML approach is introduced that achieves 89.31% accuracy, although precision and recall are not reported which leaves uncertainty when considered alongside the class imbalances. Further, the use of another novel method leveraging CNNs on the fusion of Raman, visible and near-infrared, and laser-induced breakdown spectroscopy data are explored as a method to improve accuracy of classification.

 ISBN: 978-1-61208-887-7

This is shown to far outperform the use of a single dataset, reaching 92.76% accuracy. Again, neither precision nor any other more comprehensive metric is reported, leaving uncertainty of the performance of the model.

Random forest, SVM, and neural networks are used to predict mineral composition of eight different mineral classes for rock cross-sections by using hyperspectral imagery and SEM data [11]. The random forest method is able to achieve root-mean-squared error 0.02 on an unseen region of the rock section used to train the classifer. However, when moved to new samples the errors ranged from 0.03 to 0.12 for the lowest errors for each model.

A Classifier Chain Random Forest (CCRF) performs multi-class tasks by using a chain of binary classifiers, each of which is a random forest model, and where each step builds on the results of the previous classification. Applying this method to a hyperspectral image of a rock cross section achieved accuracies between 66.96% and 94.65% for six classes [10].

A decision tree is used to identify twelve different mineral groups from 4601 SEM-EDS analyses with a relatively balanced dataset [5]. This novel approach to determine minerals in thin sections reports 100% accuracy for the twelve minerals the study set out to identify. It should be noted that the dataset has a limited sample size of minerals from only igneous rocks and multiple samples are from the minerals identified to create their dataset. In this study, there is no exploration of options for solid solutions in mineral groups.

A set of 5 minerals from river sediment samples are identified using EMPA and EDS analyses [7]. Three different novel ML algorithms are used, with the most successful achieving $\approx$92% accuracy. The authors show that ML algorithms can be used to classify geologic samples.

Many of these existing methods rely on additional spectral data, which is more computationally expensive to process and more expensive to obtain than EMPA or SEM data. Previous methods that operate on weight oxide data (such as SEM or EMPA) have yet to utilize a large dataset or classify the structural groups or mineral groups of samples. Our focus is on creating methods to determine structure, group, and subgroup for a larger number of classes than previously achieved, using a much larger dataset.

IV. The Dataset

The dataset used is $> 99\%$ composed of data available from Earthchem [20], [21]. Each mineralogical sample contains the source DOI, location, methodology, sample ID, and chemical data. These analyses are chosen because they are the most common and accurate analysis types available to geoscientists. As dense as this source is, it contains only 10 viable clay mineral samples. To supplement clay mineral data, the other $< 1\%$ of our data were taken from two publications, one with microprobe analyses [22], and one with wet chemistry analyses [23].

In order to get only exact or near-exact data samples for training and evaluating our models, any EarthChem data that used "$<$" in measurement is discarded, as this is a clear indicator of uncertainty. All samples whose sum total weight oxide was not within 10% of 100%, that is, $\mid weight\_oxide - 100 \mid < 10$, are also discarded. We allow the room for error because there is inherent error within SEM or EMPA analyses due to a wide variety of inconsistencies within the sampling and preparation steps [24]. The dataset is split into train, test, and validation sets. The size of each set for subgroups, groups, and structure samples is provided in Table I. This shows that our dataset is much larger than datasets used in the past. Additionally, this new dataset has 17 subgroups and 20 groups (Table I), which is twelve groups and seven subgroups more than two previous datasets [5] [7], and is the only dataset to make the distinction between family structure, group, and subgroup.

V. Predicting the Structure, Group, and Subgroup with Machine Learning

We evaluate the performance of four ML models as applied to the tasks of classifying structure, group, and subgroup label. For each task, we evaluated the following 4 methods:

1) Decision Tree,
2) K-Nearest Neighbors algorithm (KNN),
3) Support-Vector Machine (SVM),
4) Extremely Randomised Trees (Extra Trees).

The implementation from sklearn in Python [25] was used for all methods. These 4 methods are chosen because they all have been shown to be effective for solving mineral classification tasks in the past [5]–[7], and we hope to build on those successes with our much larger dataset and new set of tasks. The hyperparameters and metadata for most methods are constant throughout the experiments. All parameters for respective methods are listed below:

1) random_state=1, criterion='gini'
2) Neighbors: (Subgroup, 20), (Group, 25), (Structure, 25)
3) decision_function_shape='ovo', kernel='linear', C=18
4) random_state=42, criterion="gini"

The k chosen for the KNN algorithm is lower for subgroups because the lowest represented classes in those subsets have fewer points than the lowest represented classes in the group or structure subsets. The train set is used to train the four classifiers for each method, and validation set is used to choose the best of the four classifiers. This is done because the choice of the best model is considered a form of parameter tuning, which is the purpose of the validation data. The best classifier is then evaluated on the test data. The macro-F1 metric is

TABLE I
The number of samples in each of the train, test, and validation sets, for the subgroup, group, and structure classification tasks.

Set	Subgroup	Group	Structure
Train	145161	282213	282213
Test	29032	56442	56442
Validation	19354	37629	37629
Classes	17	20	6

 ISBN: 978-1-61208-887-7

used to evaluate all models across all classes to choose the best model.

VI. Results of Prediction on Validation Data

As shown in Table II, the KNN classifier performs best on the validation data in every task. The performance of the best model from each task on the test data is then evaluated, with results shown in Table III. The difference between the average recall and average precision of the subgroup classifier, about 7%, is the largest such gap of any of the classifiers. The subgroup, group, and structure classifiers all break 90% for the macro F1-score, with the structure classifier reaching 98.4%.

The disparity between the smallest and largest classes when considering the number of training points per class is quite large, and is reported in detail for the group classification dataset in Table IV. The class with the most points is olivine, and the class with the least is wollastonite, which has 99.96% fewer points than olivine.

The effect of having very few training points on the F1-score of the model is reported in Figure 1. All classes that achieve an F1-score lower than 90% have a number of data points 99% lower than the number of data points contained in the class with the most points. The worst performing model has a number of data points more than 99.9% lower than the class with the most points. Interestingly, the class with the fewest points, wollastonite, achieves a perfect F1-score of 100%. This is discussed in detail below.

VII. A Brief Discussion of the Results

As can be seen in Figure 1, it is not true that having few samples will prevent a model from classifying the class correctly, but rather that having many samples will increase the likelihood of high performance on a class. Also, all of the classes that receive worse than 0.9 as the F1-score have very few training samples.

The varying F1-scores for the low-sample classes are due to two factors: the uniqueness of crystal structure and the number of data-points per class.

Amphiboles and pyroxenes are both inosilicates that differ in structure, but have similar chemistry (which is the data used to classify the samples) [14] [26]. If a decrease in accuracy was solely due to similar chemistry, it would be apparent in these two classes. This is not the case, as amphiboles and pyroxenes have approximately the same accuracy, demonstrating that the high number of training points allows us to discern one from another (Figure 1).

Inversely, wollastonite has a far greater F1-score than zeolite. While they both have relatively few training points, they have drastically different F1-scores. This is most likely caused by the uniqueness of the wollastonite lattice structure and,

TABLE II
The macro F1-score obtained on the validation data after training each ML algorithm on the data for Subgroups, Groups, and Structure datasets.

ML Algorithm	Macro F1-score		
	Subgroup	Group	Structure
DecisionTree	88.428	90.905	97.672
KNN	**91.084**	**92.278**	**98.279**
ExtraTree	86.518	87.952	96.695
SVM	86.844	88.421	96.675

TABLE III
The precision, recall, and F1-score for the best classifier for each task from subgroup, group, and structure, as indicated in Table II on the test dataset.

Metric	Subgroup (KNN)	Group (KNN)	Structure (KNN)
Precision	95.327	93.295	97.845
Recall	88.551	92.186	98.994
F1-score	90.764	92.332	98.404

TABLE IV
The F1-score, training data points, training data points as a fraction of the largest class, and training data points as a fraction of the mean points in all classes, for each group.

Group	F1−Score(%)	|Train|	|Train|/Max(Train)%	|Train|/Mean(Train)%
Aenigmatite	92.68	188	00.21	01.33
Amphibole	96.82	12054	13.40	85.42
Analcime	80.00	95	00.11	00.67
Chlorite	92.59	140	00.16	00.99
Clay Mineral	81.08	99	00.11	00.70
Cordierite	98.18	142	00.16	01.01
Epidote	85.39	211	00.23	01.50
Feldspar	99.81	56443	62.74	400.00
Feldspathoid	98.07	1294	01.44	09.17
Garnet	99.62	27471	30.54	194.68
Kyanite Group	95.08	161	00.18	01.14
Melilite	97.08	412	00.46	02.92
Mica	97.53	10870	12.08	77.03
Olivine	99.88	89961	100.00	637.54
Prehnite	96.00	58	00.06	00.41
Pyroxine	99.34	82301	91.49	583.25
Quartz	100.00	113	00.13	00.80
Serpentine	82.93	91	00.10	00.64
Wollastonaite	100.00	36	00.04	00.26
Zeolite	54.55	73	00.08	00.52

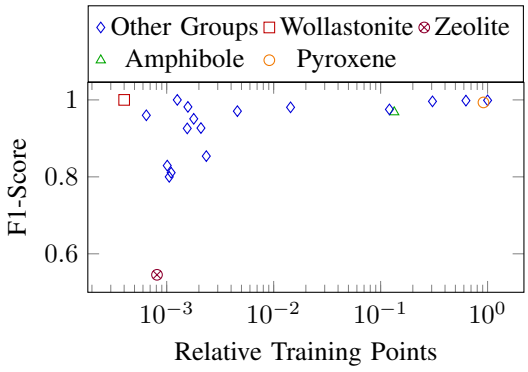


Fig. 1. The relative number of training data points per each group class (points in class/max(points in classes)) versus the F1-score of the best model for the group task (from Table I) on test data.

 ISBN: 978-1-61208-887-7

therefore, chemical fingerprint. The dichotomy between the accuracy of the zeolite and wollastonite groups show that sample count is not the sole indicator for performance of a class. Wollastonite is the only silicate mineral in the pyroxenoid group in our dataset. As noted by [27], pyroxenoids have a unique structure from the pyroxenes and other mineral groups. This uniqueness is compounded with that fact that ≈40-50% of wollastonite is captured by CaO in our dataset. This value is two orders of magnitude larger than any other CaO datapoint in the dataset. It is this uniqueness of wollastonite that allows us to determine and discern from other silicate minerals without a robust dataset. As the other members of this low data-point cluster do not share the same uniqueness as wollastonite, it can be observed that their low sample count hinders their accuracy. Zeolites in particular suffer from this issue. Zeolites accept a far greater range ion exchange then that of wollastonite [12]. This wide range of chemical impurities decreases the uniqueness of zeolites.

The reasoning for the difference between wollastonite and zeolite holds for the other groups with relative training points less than 10^{-2}, demonstrating that with a low number of relative training points, a mineral's lattice structure dictates the accuracy.

VIII. Improving Group and Subgroup Prediction

The silicate structural groups and mineral groups provide explicit divisions to create subsets of the data, and we aim to leverage this knowledge to improve classification results for the group and subgroup classification tasks. In order to utilize these divisions to attempt to improve the classifications of mineral groups, a high performing classifier of structural class, say S, is used. For a given data point v, which is a vector of elemental weights, the probability vector that is the output of $S(v)$ is appended to the end of v. This is the initial augmentation step. At this point, a classifier is trained to classify mineral groups using the augmented data points. The structural classifier S is chosen based on the average top-3 recall rather than the F1-score in order to increase the chances that the correct class is indicated strongly in the probability vector returned by $S(v)$, even if it is not the highest value in the vector. This choice was supported when testing using a subset of the data for the augmented group classification task. The results are omitted for brevity. In this way, additional data that the model can use to separate the various classes is explicitly provided by leveraging domain specific knowledge. The use of multiple models is an adaptation of ensemble learning, which demonstrated success in [6]. The macro-F1 metric is used to evaluate the resulting models across all classes in order to choose the best model to use on the test data for each task.

IX. Results of Prediction using Augmented Models

The best models for the group and structure tasks, as evaluated by the top-3 recall metric, are both the KNN classifiers. This is shown in Table V. These are also the best models as evaluated by F1-score, shown in Table II.

The results of evaluating (on the validation data) the best models for the subgroup and group tasks, trained with data augmented by the best top-3 models, are displayed in Table VI. The change in F1-score from the old models to the new models is also displayed. The best models are KNN for both subgroup and group. The KNN subgroup classifier has a higher F1-score than the best models from the non-augmented training, while the new group classifier performs worse. The largest improvement for both the subgroup and group models is in the SVM, with an improvement of over 3% in both cases. All subgroup models showed improvement, while only the ExtraTree and SVM were improved of the group classification models, while the KNN group model F1-score and the decision tree F1-score decreased by 0.029% and 2.046%, respectively.

When evaluated on the test data, both the subgroup and group models performed worse across precision, recall, and F1-score than the non-augmented methods, as shown in Table VII. The F1-score for the subgroup model decreased by just under 0.06%, while the F1-score for the group model decreased by more than 2%. This is also shown in Table VII. The greater decrease in accuracy from the group model reflects the worse performance on the validation set. The only subgroup classes that performed worse in the new model were clinopyroxene, kaersutite, and phlogopite, by -0.004%, -1.001%, and -0.007%, respectively.

The per-class change in F1-score is displayed in Figure 2 for the groups only, as this experienced the larger decrease. The classes experience both increases and decreases in F1-score. The largest increase of just under 10% is for Analcime, the worst performing class in the original model. The largest

TABLE V
Top-3 recall of the classifiers shown in Table II (evaluated on the validation data) for group and structure classification. The best results are in bold.

	Top-3 Recall (%)	
Task	Group	Structure
DecisionTree	93.168	98.860
KNN	**97.883**	**99.770**
ExtraTree	88.466	97.427
SVM	97.179	99.568

TABLE VI
The results of evaluation on the validation data after training the subgroup and group classifiers on the dataset augmented with the highest top-3 recall from the previous classifier. "Change" indicates the change in accuracy from the classifiers trained with the original data to those trained with the augmented data. The best results are in **bold**.

	SubGroup (%)		Group (%)	
ML Algorithm	Macro F1	Change	Macro F1	Change
DecisionTree	89.286	0.858	88.860	−2.046
KNN	**91.107**	0.023	**92.249**	−0.029
ExtraTree	86.888	0.370	88.765	0.813
SVM	90.255	**3.411**	91.778	**3.358**

 ISBN: 978-1-61208-887-7

TABLE VII
THE RESULTS OF RUNNING THE BEST CLASSIFIERS (AS INDICATED IN TABLE VI) ON THE TEST DATASETS AFTER AUGMENTING THEM WITH THE PROBABILITIES FROM THE CLASSIFIERS WITH THE HIGHEST TOP-3 RECALL. AVERAGE PRECISION, RECALL, AND F1-SCORE OVER ALL CLASSES ARE REPORTED.

	SubGroup (%)		Group (%)	
Metric	KNN	Change	KNN	Change
Precision	95.204	−0.123	92.057	−1.237
Recall	88.546	−0.005	90.462	−1.724
F1-score	90.705	−0.060	90.302	−2.030

decrease of just over 25% is for wollastonite. Zeolite also experienced a substantial decrease in F1-score, of over 10%. All three of the most affected classes had less than 1% of the mean points in all group classes used for training (Table IV).

X. DISCUSSION OF THE RESULTS OF THE AUGMENTED MODELS

The F1-scores from training with the augmented data were higher for several of the classifiers than those from the original models on the validation data for both the subgroup and group tasks, which shows promise for the method. However, the results on the test data were worse in both cases, which indicates that there are considerable improvements to be made before this technique can be viable. The ultimate decrease in accuracy for the KNN classifiers and the large increases shown by both SVM classifiers are likely due to the increase in dimensionality of the feature vectors. This increase, which is caused by augmenting the vectors with the data regarding either the structure or group classification, could be mitigated by the use of some dimension-reducing method, such as Principal Component Analysis (PCA) [28]. The changes of the group classification results are discussed in depth, as the changes are much larger than those for the subgroup classification.

The groups whose performance improves by more than 0.1% after augmentation are serpentine, cordiedite, aenigmatite, analcime, epidote, and the kyanite group. These improvements are likely a result of the new model, as the train, validation, and test datasets are all constant between experiments. These mineral groups are from different structural sub-families, which disallows the possibility of crystal lattice structures dictating the change in performance after augmentation. The groups whose F1-scores decrease after augmentation are also from different structural subfamilies, maintaining this pattern.

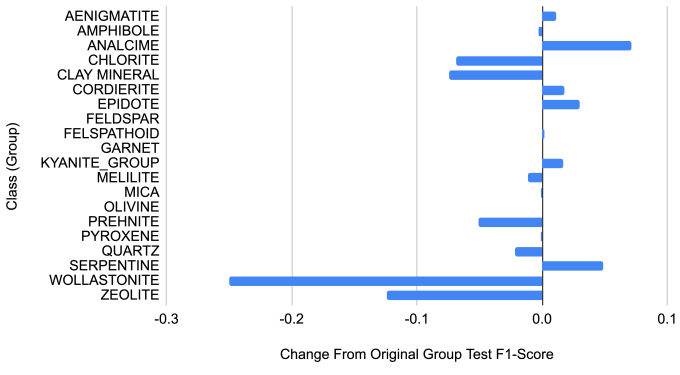


Fig. 2. The difference in classification F1-score from the original Group classifier to the Group classifier with structure data. Higher indicates better performance from the Group classifier with structure data.

Wollastonite and zeolite experience the most significant decrease in performance, and have the fewest and fourth fewest training points of all classes, respectively (Table IV). The low point count contributes to this observed performance loss. Additionally, the other groups that lose performance all have point counts less than 100, which is 7% of the mean points per class and 0.12% of the max point in any class. Also, by augmenting the feature vectors with resulting structural predictions, the uniqueness of the wollastonite samples that allowed them to perform well with few training samples may be reduced. The inverse holds for groups with a point count greater than 100. Mineral groups over this relative point count either experience very little change, or a notable increase in F1-score. This coupling between low point counts and a decrease in performance after augmentation leads us to believe that one can successfully apply structure augmentation to groups with high counts of relative training points.

XI. CONCLUSION AND FUTURE WORK

Previously, [5]–[7] demonstrated success using ML as a tool for mineral identification. We have expanded on their ideas by not only using a larger dataset of silicate minerals, but by being able to identify more classes at the same level of accuracy as the studies before us. We also note that although we have created the most extensive silicate mineral classifiers to the best of our knowledge, classes with fewer instances could still be improved. This leads us to believe that the development of future models that incorporate even more data, along with deeper structural information, will perform even better. Additionally, the use of methods to create synthetic data to balance the classes could be beneficial in future efforts. It is also anticipated that the use of dimension reduction will greatly improve classification results. To aid in the collection and development of the models, we will be offering free use of these models through a website [29], and the ability to upload labeled data to improve the model.

ACKNOWLEDGMENT

We thank Jack Hay and Mia Ratino for their time spent in discussions, reading drafts, and providing feedback. We also thank Reiform for funding the compute resources for this project and hosting the open source ML models on their servers.

REFERENCES

[1] W. A. Deer, R. A. Howie, and J. Zussman, *An Introduction to the Rock-Forming Minerals*. Mineralogical Society of Great Britain and Ireland, 01 2013. [Online]. Available: https://doi.org/10.1180/DHZ

[2] T. Koljonen and L. Carlson, "Behaviour of the major elements and minerals in sediments of four humic lakes in south-western Finland," *Fennia*, no. 137, pp. 5–47, 1975.

 ISBN: 978-1-61208-887-7

[3] M. Hopkins, T. M. Harrison, and C. E. Manning, "Low heat flow inferred from $>$ 4 Gyr zircons suggests Hadean plate boundary interactions," *Nature*, vol. 456, no. 7221, pp. 493–496, 2008.

[4] W. A. Deer, R. A. Howie, and W. S. Wise, *Rock Forming Minerals; Single Chain Silicates*, 2nd ed. London: Geological Society of London, 1997.

[5] E. Akkaş, L. Akin, H. Evren Çubukçu, and H. Artuner, "Application of Decision Tree Algorithm for classification and identification of natural minerals using SEM-EDS," *Computers and Geosciences*, vol. 80, pp. 38–48, 2015.

[6] P. Jahoda, I. Drozdovskiy, S. J. Payler, L. Turchi, L. Bessone, and F. Sauro, "Machine learning for recognizing minerals from multispectral data," *Analyst*, vol. 146, pp. 184–195, 2021. [Online]. Available: http://dx.doi.org/10.1039/D0AN01483D

[7] H. Hao, X. Jiang, Y. Sun, W. Dou, and Q. Gu, "A Method for Classification of Heavy Mineral Based on Machine Learning," *Proceedings - 2020 IEEE 22nd International Conference on High Performance Computing and Communications, IEEE 18th International Conference on Smart City and IEEE 6th International Conference on Data Science and Systems, HPCC-SmartCity-DSS 2020*, pp. 991–998, 2020.

[8] Y. Zhang, M. Li, S. Han, Q. Ren, and J. Shi, "Intelligent identification for rock-mineral microscopic images using ensemble machine learning algorithms," *Sensors*, vol. 19, no. 18, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/18/3914

[9] X. Zeng, Y. Xiao, X. Ji, and G. Wang, "Mineral identification based on deep learning that combines image and mohs hardness," *Minerals*, vol. 11, no. 5, 2021. [Online]. Available: https://www.mdpi.com/2075-163X/11/5/506

[10] I. C. Contreras, M. Khodadadzadeh, and R. Gloaguen, "Multi-label classification for drill-core hyperspectral mineral mapping," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, vol. 43, no. B3, pp. 383–388, 2020.

[11] L. Tuşa et al., "Drill-core mineral abundance estimation using hyperspectral and high-resolution mineralogical data," *Remote Sensing*, vol. 12, no. 7, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/7/1218

[12] W. A. Deer, R. A. Howie, W. S. Wise, and J. Zussman, *Rock-forming minerals. Volume 4B. Framework silicates: silica minerals. Feldspathoids and the zeolites*, 2nd ed. London: Geological Society of London, 2004.

[13] D. H. Brouwer, S. Cadars, J. Eckert, Z. Liu, O. Terasaki, and B. F. Chmelka, "A general protocol for determining the structures of molecularly ordered but noncrystalline silicate frameworks," *Journal of the American Chemical Society*, vol. 135, no. 15, pp. 5641–5655, 2013.

[14] F. C. Hawthorne et al., "Ima report: Nomenclature of the amphibole supergroup," *American Mineralogist*, vol. 97, no. 11-12, pp. 2031–2048, 2012.

[15] P. Bayliss, "Nomenclature of the trioctahedral chlorites," *Canadian Mineralogist*, vol. 13, pp. 178–180, 1975.

[16] W. A. Belson, "Matching and Prediction on the Principle of Biological Classification," *Journal of the Royal Statistical Society Series C*, vol. 8, no. 2, pp. 65–75, June 1959. [Online]. Available: https://ideas.repec.org/a/bla/jorssc/v8y1959i2p65-75.html

[17] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006.

[18] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[19] R. J. Samworth, "Optimal weighted nearest neighbour classifiers," *The Annals of Statistics*, vol. 40, no. 5, pp. 2733 – 2763, 2012. [Online]. Available: https://doi.org/10.1214/12-AOS1049

[20] "Earthchem." [Online]. Available: http://portal.earthchem.org/

[21] K. Lehnert, Y. Su, C. H. Langmuir, B. Sarbas, and U. Nohl, "A global geochemical database structure for rocks," *Geochemistry, Geophysics, Geosystems*, vol. 1, no. 5, 2000.

[22] J. Środoń, E. Zeelmaekers, and A. Derkowski, "The charge of component layers of illite-smectite in bentonites and the nature of end-member illite," *Clays and Clay Minerals*, vol. 57, no. 5, pp. 649–671, 2009.

[23] C. S. Ross and S. B. Hendricks, "Minerals of the montmorillonite group their origin and relation to soils and clays." *Geological survey professional paper 205-b,*, vol. 23-79, pp. 23–79, 1943.

[24] T. O. Ziebold, "Precision and Sensitivity in Electron Microprobe Analysis," *Analytical Chemistry*, vol. 39, no. 8, pp. 858–861, 1967.

[25] F. Pedregosa et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[26] N. Morimoto, "Nomenclature of Pyroxenes," *Mineralogy and Petrology*, vol. 39, no. 1, pp. 55–76, 1988.

[27] B. E. Warren and J. Biscoe, "The crystal structure of the monoclinic pyroxenes," *Zeitschrift für Kristallographie - Crystalline Materials*, vol. 80, no. 1-6, pp. 391–401, 1931.

[28] I. Jolliffe, *Principal Component Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1094–1096. [Online]. Available: https://doi.org/10.1007/978-3-642-04898-2_455

[29] J. Hay and B. Parfitt, Jun 2021. [Online]. Available: https://mindicator.reiform.com/

 ISBN: 978-1-61208-887-7

Physical and Computer Modeling of Extra-High Temperature Processes: Problems and Challenges

Nguyen Thi Thu Trang
AGH - University of Science and Technology
al. Mickiewicza 30, 30-059 Krakow, Poland
e-mail: nguyen@agh.edu.pl

Marcin Hojny
AGH - University of Science and Technology
al. Mickiewicza 30, 30-059 Krakow, Poland
e-mail: mhojny@agh.edu.pl

Tomasz Dębiński
AGH - University of Science and Technology
al. Mickiewicza 30, 30-059 Krakow, Poland
e-mail: debinski@agh.edu.pl

***Abstract*—The paper describes the methodology of integrated modeling of extra-high temperature steel processing in supporting the design of new technologies (e.g. soft-reduction and direct strip casting processes). The work is supplemented with examples of the practical use of the proposed methodology in supporting physical simulations. The problems and challenges of the proposed solution are briefly described.**

Keywords - Finite Element Method; Monte Carlo method; physical simulation; computer simulation; mushyzone; Smoothed Particle Hydrodynamics.

I. INTRODUCTION

The contemporary approach to the issues of engineering new processes involves the extensive application of computer technologies and methods. Then, the development of electronics, including computers and implementation of numerical simulation methods, has led to the construction of equipment allowing complex technological processes to be tested in the laboratory scale. As the tests are conducted on actual materials, the term "physical simulation" has been adopted as opposed to the numerical simulation. The physical simulation is directly related to a new type of computer controlled tensile testing machines, able to change the experiment conditions automatically during the experiment according to the programme adopted by the engineer. The evaluation of mechanical properties of samples subjected to various simulation variants is the basis for developing a special "process map", which enables the optimal parameters of the continuous casting machine to be determined when casting a specific steel grade [1]. It allows the casting process parameters (e.g. casting speed, cooling rate in the primary and secondary zones) to be adjusted so as to avoid the potential threat of cracks. As may be concluded from the above description, each steel grade requires separate tests. The main problem concerning the experimental work is the sample heterogenous temperature. Keeping temperature constant during the whole experimental procedure is difficult [2]. There are also some problems and challenges with the experimental and numerical simulation procedure:

- prediction of extra-high temperature strain-stress relationships,
- prediction macro/microstructure (grain size),
- prediction of heat transfer coefficients (necessary for numerical simulations),
- extremely high distortions of the mesh during simulation of deformation at temperatures close to the solidus line,
- deformation experiments at temperatures close to the solidus line (semi-solid state).

A lack of good methods for semi-solid steel simulation and significant inhomogeneity in strain distribution lead to a weak accuracy of the resulting stress field. In order to solve the above problems, the methodology integrating the areas of physical and computer simulation was proposed. A schematic diagram of the integrated modeling methodology combining the advantages of physical and computer simulation is presented in Figure 1. It consists of three main layers: physical simulation, computer simulation and supporting equipment. The proposed solution uses a methodological research capability of modern Gleeble 3800 thermo-mechanical simulators to simulate physical processes [1] (first layer), and the benefits of numerical modeling (second layer, DEFFEM3D software [1][2]). Mathematical models are the original solutions of the developed DEFFEM 3D software, such as the thermo-mechanical model of steel deformation in the semi-solid state with variable density, and the multi-scale model of resistance heating coupled with grain growth modelling in the micro scale. More details about mathematical models can be found in monograph [1].

Supporting equipment (third layer) includes scanning microscope, Zwick Z250 testing machine, thermal imaging camera, 3D system scanning and tomograph.

 ISBN: 978-1-61208-887-7

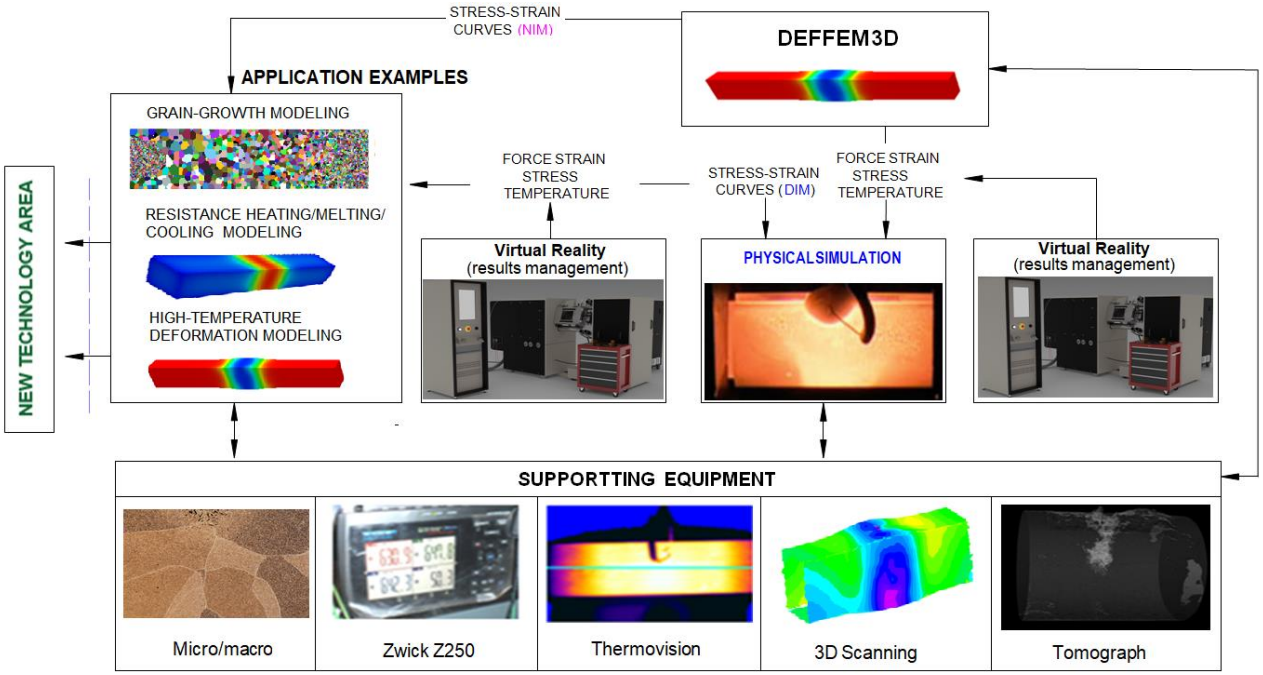


Figure 1. The scheme of the methodology (coupled modeling - physical and computer simulation) with examples application.

II. APPLICATION EXAMPLES

The developed methodology was used in the course of research and development works on the interest rate of new technologies for the aircraft industry:

- application of the methodology in computer-aided design of casting critical parts of aircraft engines. Simulations of casting in ceramic molds obtained using the lost wax method. Figure 2(a) presents an example temperature distribution during cooling of the blade of jet engine.
- support in the design of hot forming technology for the strengthening of the intermediate hull directing airflow in a jet engine. Simulation of the resistance heating of the sheet before the forming process. Figure 2(b) presents an example temperature distribution during resistance heating of the blank.

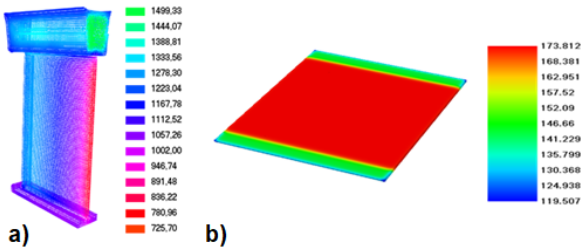


Figure 2. Application examples (aicraft industry).

III. CONCLUSIONS

The developed numerical tool (DEFFEM 3D software), combined with the capabilities of modern thermo-mechanical simulators of the Gleeble series, allows theoretical support for the design of new technologies. This allows for restricting the number of expensive experimental tests to the minimum, e.g. by selecting a suitable heating schedule to achieve the desired temperature at the sample section, or getting additional information about the process, eg. estimating zones with diversified grain growth dynamics, or information on local cooling rates at any point within the volume of the sample tested. Further research will focus on developing numerical tools based on particle methods, such as SPH (smoothed particle hydrodynamics). This approach will allow to solve some of the problems related to the modeling of steel deformation in the semi-solid state using the finite element method.

ACKNOWLEDGMENT

The work was realized as a part of fundamental research financed by the Ministry of Science and Higher Education, grant no. 16.16.110.663.

REFERENCES

[1] M. Hojny, Modeling steel deformation in the semi-solid state. Advanced Structured Materials, vol.47, Springer, Switzerland, 2018.

[2] M. Hojny, T. Dębiński, M. Głowacki and Trang Thi Thu Nguyen, “Spatial Thermo-Mechanical Model of Mushy Steel Deformation Based on the Finite Element Method,” Archives of Foundry Engineering, vol. 21, pp.17-28, 2021.

 ISBN: 978-1-61208-887-7

AMPRO-HPCC: A Machine-Learning Tool for Predicting Resources on Slurm HPC Clusters

Mohammed Tanash
Computer Science Department
Kansas State University
Manhattan, United States
e-mail: tanash@ksu.edu

Daniel Andresen
Computer Science Department
Kansas State University
Manhattan, United States
e-mail: dan@ksu.edu

William Hsu
Computer Science Department
Kansas State University
Manhattan, United States
e-mail: bhsu@ksu.edu

***Abstract*—Determining resource allocations (memory and time) for submitted jobs in High Performance Computing (HPC) systems is a challenging process even for computer scientists. HPC users are highly encouraged to overestimate resource allocation for their submitted jobs, so their jobs will not be killed due to insufficient resources. Overestimating resource allocations occurs because of the wide variety of HPC applications and environment configuration options, and the lack of knowledge of the complex structure of HPC systems. This causes a waste of HPC resources, a decreased utilization of HPC systems, and increased waiting and turnaround time for submitted jobs. In this paper, we introduce our first ever implemented fully-offline, fully-automated, stand-alone, and open-source Machine Learning (ML) tool to help users predict memory and time requirements for their submitted jobs on the cluster. Our tool involves implementing six ML discriminative models from the scikit-learn and Microsoft LightGBM applied on the historical data (sacct data) from Simple Linux Utility for Resource Management (Slurm). We have tested our tool using historical data (saact data) using HPC resources of Kansas State University (Beocat), which covers the years from January 2019 - March 2021, and contains around 17.6 million jobs. Our results show that our tool achieves high predictive accuracy R^2 (0.72 using LightGBM for predicting the memory and 0.74 using Random Forest for predicting the time), helps dramatically reduce computational average waiting-time and turnaround time for the submitted jobs, and increases utilization of the HPC resources. Hence, our tool decreases the power consumption of the HPC resources.**

***Keywords*—HPC; Scheduling; Supervised Machine Learning; Slurm; Performance.**

I. Introduction

High Performance Computing (HPC) resources have become more available to users to run their extensive computations and simulations. One of the most important parts of the HPC system is the batch scheduler. The batch scheduler manages resources and queues of all submitted jobs in the cluster. Hence, it is the part that decides where and when jobs will run in the cluster. On the other hand, batch scheduler performance depends on the resource requirements from the user such as the amount of memory, requested time, and the number of cores [1]. While these resource requirements are the responsibility of HPC users to determine, it is a fact that users may determine resource needs inaccurately [2]. Also, users are highly encouraged to overestimate these resources in order to satisfy job requirements, so their jobs will not be killed during the run time due to insufficient resources [3]. Overestimating job resource requirements negatively impacts the performance and the utilization of the HPC system. Moreover, over-estimating job resource process will increase average turn-around time and average waiting time for submitted jobs.

In this paper, we introduce the first-ever open-source, stand-alone, highly-accurate, fully-offline, and fully-automated tool called AMPRO-HPCC, which stands for "A Machine-Learning-Tool for Predicting Resources On Slurm HPC Clusters". AMPRO-HPCC aims to help HPC users predict and estimate the required job resource allocations (memory and time) for their submitted jobs. Our tool uses Simple Linux Utility for Resource Management (Slurm) historical logs-data (sacct) and involves implementation of six Machine Learning (ML) discriminative models from the scikit-learn [4] and Microsoft LightGBM (LGBM) [5]. Our ML tool is invoked through Command Line Interface (CLI), and it consists of two parts: **i)** System administrator part, which is responsible for preparing data and all the required models for building the final models and tool; **ii)** HPC user side, which will automatically read the submission job script provided from the HPC user and recommend the required job allocation resources (memory and time) for the associated submitted job.

We have extended our previous work [6]–[8], and designed the AMPRO-HPCC tool to help HPC users determine the allocation of HPC resource needs (memory and time) using supervised ML over historical data (sacct). Our open-source tool can be found on GitHub [9].

The rest of this paper is organized as follows: Section 2, discusses the related work. Section 3 describes our prediction tool, AMPRO-HPCC, which includes the workflow model, data preparation, evaluation and building of our Mixed Account Regression Model (MARM), and the job resource prediction. Section 4 shows our promising results. Finally, Section 5 presents our conclusion.

II. Related Work

Simple Linux Utility for Resource Management (Slurm) is a resource manager, which enables HPC resources to execute parallel jobs efficiently [10]. Slurm turns a set of hundreds or tens of thousands of computers into a single unit that you can

 ISBN: 978-1-61208-887-7

run jobs on. So Slurm makes parallel computers easy to use. Slurm allocates resources within a cluster, manages the nodes, and keeps track of architecture within a node such as sockets, NUMA boards, cores, hyper threads, memory, interconnect, generic resources, and managing licenses. Slurm manages jobs through varieties of scheduling algorithms (fair share, gang, advanced reservation, etc.) [11].

While there are many kinds of resource management scheduler such as Sun Grid Engine (SGE) [12], Tera-scale Open-source Resource and Queue manager (TORQUE) [13], [14], and Portable Batch System (PBS) [15], [16], Slurm is the most popular and most used among them. Hence, we implemented our tool based on Slurm workload manager HPC systems.

There are many studies and research focusing on predicting the running time and the time required for running application on the HPC systems or the cloud [17]–[29], while there are quite a lot of research that focuses on predicting the amount of memory required for the submitted jobs [30], [31].

Our work differs by the methodology used and the ability to predict both memory and time required for submitted jobs on the HPC systems. We conclude "there does not yet exist software that can help to fully automate the allocation of HPC resources or to anticipate resource needs reliably by generalizing over historical data, such as determining the number of processor cores and the amount of memory needed." [6]. Hence, we are introducing the first-ever open-source ML tool for predicting job resources (memory and time) for submitted jobs on the HPC systems.

III. Prediction Tool AMPRO-HPCC

Figure 1 illustrates the use-case diagram of our ML tool. We have two types of users: i) system administrators (referred to as admin henceforth) and ii) HPC users (referred to as users henceforth). Modules `PreProcess`, `BuildPerAccountModels`, `BuildMixedAccountModels` and `TrainSelectedMARM` are available to admins, while the `Ampro-hpcc` module is available to both admins and users. The main objective of our tool is to build Mixed Account Regression Models (MARM), which are regression models built on a subset of slurm *Accounts* with the best overall predictive performance, containing a reasonable percentage of jobs. Here, we provide descriptions of each module along with its inputs and outputs.

A. AMPRO-HPCC Workflow Model

Figure 2 describes the workflow model of our work as follows: i) The user prepares and creates a new job, which includes the requested amount of memory, time limit, quality of service (QoS), and partition name for the proposed job. ii) The HPC user will submit their job and passes it through our ML model in order to predict the amount of the required memory and the amount of time needed for the job to run. iii) Our ML model will process the submitted job by parsing all of the parameters needed, then predicting required memory and time for the specific job. iv) The HPC user will get feedback

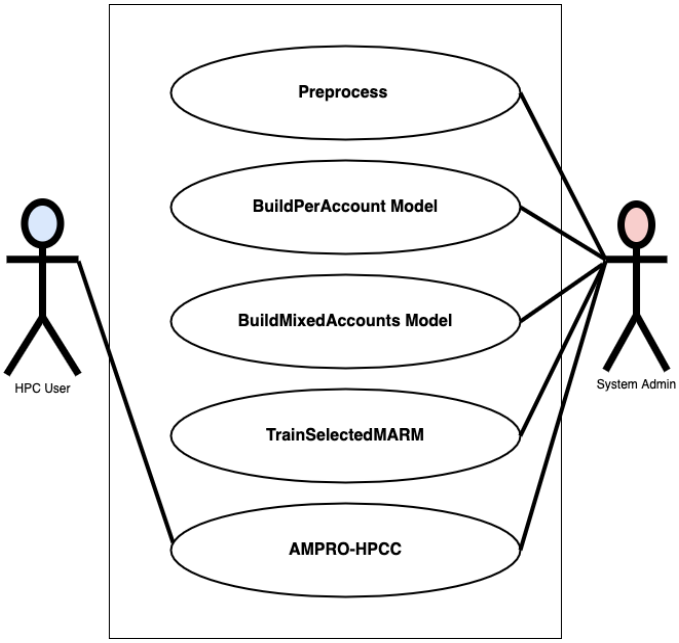


Fig. 1. Use-Case Diagram for AMPRO-HPCC

from our model regarding the needed amount of memory and time for their submitted jobs. v) The user will have the option to confirm or deny to use the predicted values for the required memory and time. vi) If the user confirms the use of the predicted amounts for either the required memory or the required time or both, then our ML model will update the amounts of memory and time as needed for the submitted job. If not, then the submitted job will remain the same. vii) The user will be notified about the changes to their jobs. viii) Finally, either an updated job or the original job will be scheduled for running on the cluster.

B. Data Preparation

The data preparation or `Preprocess` module takes the path (`path_to_data`) to logs of slurm jobs accounting information (sacct) to extract *Account, ReqMem, Timelimt, ReqNodes, ReqCPUS, QoS, Partition, MaxRSS, CPUTimeRAW, and State* from the dataset. A description of these features can be found at [32]. The module also asks the admin to provide default time-limit (`def_time`), default quality of service (`def_qos`), and default partition assignment (`def_partition`) to deal with some of the missing values in the data. Finally, the admin also has the ability to specify a set of QoS (`sel_qos`) and partitions (`sel_partition`) that they want to select over the entire data. In addition, the `Pre-processing` module does its own filtration by only selecting jobs with *State* equals to *'COMPLETED'*, and having non-zero *MaxRSS* and *CPUTimeRAW*. Next, this module standardizes *Timelimit* to numeric hours, *MaxRSS* and *ReqMem* to gigabytes (GB), and *Account* and *QoS* to numeric factors. Finally, *Account*, *ReqMem*, *ReqNodes*, *Timelimit*, *QoS*, *MaxRSS*, and *CPUTimeRAW* are normalized us-

 ISBN: 978-1-61208-887-7

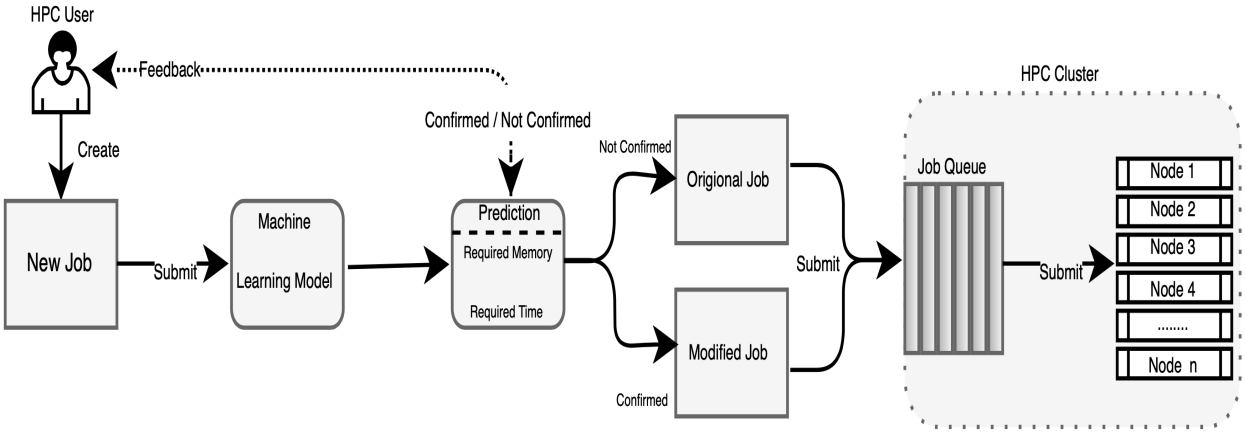


Fig. 2. AMPRO-HPCC Work-Flow Diagram.

ing the `StandardScaler` transform in Scikit-learn Python package [4].

C. Evaluating Individual Regression Models

Before building the *Mixed Account Regression Models* (MARM), the admin can evaluate individual regression models to note what may be most suited to their dataset. Although optional, the `BuildPerAccModels` module can provide initial insights on the quality of data and can significantly speed up MARM building time by nominating promising regression models for MARM overall possibilities. The `BuildPerAccModels` module requires the admin to provide the path to processed data (`path_to_data`), independent variables or features (`indep_vars`), and a dependent variable (`dep_var`) to train and evaluate seven popular regression models on all data-subsets containing individual *Account*. At this point, the admin can specify the minimum number of jobs an individual *Account* should have in order to be considered (`min_num_jobs`). The seven regression models include: i) Lasso Least Angle Regression (LL) [33], [34], ii) Linear Regression (LR) [34], iii) Ridge Regression (RG) [34], iv) Elastic Net Regression (EN) [34], v) Classification and Regression Trees (DTR) [35], vi) Random Forest Regression, (RFR) [36], and vii) LightGBM (LGBM) [5]. The regression models are evaluated by means of the Coefficient of determination (R^2), and root mean squared error (RMSE) [34]. We used scikit-learn's [4] implementation for all models and performance metrics.

D. Evaluating Mixed Account Regression Models

Once the individual regression models have been evaluated, the admin can select what models should be considered for MARM. The admin can also decide to select all seven regression models for MARM. Our `BuildMixedAccountModels` module requires a path to processed data (`path_to_data`), independent variables (`indep_vars`), dependent variable (`dep_var`), the minimum number of jobs (`min_num_jobs`), and the names of the regression models to be considered for MARM (`methodnames`). A mixed account regression model $MARM(N, M, X, Y)$ is constructed by finding N accounts with the best performance score for a given regression model M in predicting a dependent variable Y using independent variables X. MARM is constructed iteratively and can be summarized as follows:

$$MARM(N, M, X, Y) = \begin{cases} N^{'} & N = 1 \\ MARM(N-1, M, X, Y) \cup N^{'} & \text{otherwise} \end{cases}$$

where $N^{'} \in N$ is the *Account* that results in the best overall aggregate score in terms of R^2 on training ($R2_{tr}$) and testing ($R2_{te}$) datasets and number of jobs ($S_{N'}$), given by:

$$N^{'} = \arg\max_{n \in N}(R2_{tr}(M, X_{A[n]}, Y_{A[n]}), R2_{te}(M, X_{A[n]}, Y_{A[n]}), S_{A[n]})$$

where $X_{A[n]}$ and $Y_{A[n]}$ correspond to independent and dependent variables respectively for an unique *Account* $A[n]$. Thus, the MARM of N accounts depends upon the MARM of $N-1$ accounts appended with the best overall *Account* $N^{'}$ that results in the best overall performance. R^2 scores $R2_{tr}$ and $R2_{te}$ are calculated by randomly splitting the data into 80% (training) / 20% (testing), five times (5-fold) modeling using the regression model M, and averaging the R^2 scores on training and testing data subsets over the five runs. A comprehensive explanation of the Mixed Account Regression Model (MARM) can be found in our publication [7].

E. Building MARM for Prediction

The `BuildMixedAccountModels` module generates R^2 score distributions over $1, 2, \cdots, N$ for each regression model M specified by the admin in `methodnames`. Thus, the admin can determine which regression model performs the best along with the best number of accounts $\hat{n} \leq N$ to use. Thus, our `TrainSelectedMARM` module takes the selected regression model (`sel_model`), path to processed data (`path_to_data`), path to the intermediate results produced by `BuildMixedAccountModels` module (`path_to_marm_res`) independent variables

 ISBN: 978-1-61208-887-7

(indep_vars), dependent variable (dep_var) and number of accounts (num_acc) to build the final MARM for resource prediction.

F. Job Resource Prediction

Finally, the users of the slurm system can use Ampro-hpcc module by providing a path to their Slurm job submission script (path_to_script), a path to selected MARM model (path_to_model), a path to system default (path_to_defaults), and a path to the normalization transform (standard Scalar inverse transform) (path_to_stdscale) to obtain the recommended values of time and memory. To be conservative and prevent failure due to time and memory requirements that may underestimate of the actual memory and time utilization, our recommended values are increased by 10%.

IV. Results and Discussion

A. Preprocessing and PerAccount Models

We applied our ML tool using the HPC resources at Kansas State University, called Beocat. The data side has 17.6 million instances and covers the years 2018 - 2021 of the usage. After using PreProcessing module only selecting *'normal'* QoS, the dataset contained 7.8 million jobs spread across 21 unique accounts. Employing BuildPerAccountModels, we evaluated all seven regression models across 21 accounts, resulting in Figure 3 for predicting time (CPUTimeRAW) and memory (MaxRSS) that shows boxplots of R^2 and negative RMSE score distributions. We found LGBM, DTR, and RFR to be clear winners. Thus, we decided to only utilize LGBM, DTR, and RFR to build MARM.

B. MARM Models in Beocat

Utilizing BuildMixedAccountModels, we constructed MARMs to predict memory and time in Beocat using 17 out of 21 accounts (80% of the total accounts) in Beocat. Figure 4 shows the mean R^2 score distribution of DTR, RFR, and LGBM on training and testing datasets versus the number of best account combinations in predicting time. It can be seen that the R^2 decreases as the number of accounts (and jobs) increases. We found RFR was the best performer in predicting time, while LGBM was the best performer in predicting memory. Thus, we finalized the memory and time MARM using TrainSelectedMARM to be i) best five account combination (spanning across 1.8 million jobs) with an average R^2 of 0.74, for building an RFR based time model and ii) best thirteen accounts combination (spanning across 1.4 million jobs) with average R^2 of 0.72, for building an LGBM based memory model as shown in Figure 4. Using the finalized MARMs, we randomly sampled 5000 jobs from Beocat and ran them on a Slurm simulator with requested, actual, and predicted time and memory values.

C. Evaluating Our Model

We assessed our model using the Slurm simulator [37], [38], which was developed by the Center for Computational Research, SUNY Buffalo. The Slurm simulator was chosen because it is implemented from a modification of the actual Slurm code while disabling some unnecessary functions, which do not affect the functionality of the real Slurm [37].

Figure 5 shows submission and execution time, which indicates the difference between the job submission time (timestamp that represents when the job was submitted) and the execution time (difference between the start and end execution time) for five thousand jobs. Our results indicate that we have achieved almost identical running time compared to the actual running time.

Figure 6 measures and compares system utilization using requested jobs resources versus actual job resources versus predicted job resources using the AMPRO-HPCC tool. Our results show that our tool reached almost similar utilization compared to the utilization of the HPC system that used actual job resources because of the high prediction accuracy of our ML tool.

Figure 7 compares and assesses the backfill-sched algorithm's performance. The graph shows more efficient performance on the backfill-sched algorithm on the Beocat testbeds that used our ML module than the ones that did not. The graph shows fewer density results when using predicted values since using our AMPRO-HPCC model decreases the number of resources required by the user for the submitted jobs in most cases. This situation results in helping Slurm fit more jobs on the cluster. It also reduces the need to use the backfill-sched algorithm and resulting in more overall system efficiency by using these available resources.

Table 1 provides the calculated average waiting time, and average turn-around time for Beocat jobs for requested, actual, and predicted job resources allocation. Our results show that our tool was able to reduce the average waiting time for submitted jobs from 680 hours to 8.0 hours and the average turnaround time from 692 hours to 16.4 hours.

V. CONCLUSION

Determining the allocation of HPC resources for submitted jobs is a difficult process for HPC users. It is still an open question how many resources the user should specify (memory and time) for their submitted jobs on the cluster. HPC users are encouraged to overestimate job resources for their submitted jobs. In this paper, we have developed a novel and the first-ever open-source, stand-alone, fully-automated, highly-accurate, and fully-offline ML tool to help HPC users to determine the amount of required resources (memory and time) for their submitted jobs on the HPC clusters. Our tool was built using supervised ML algorithms. Our tool consists of two parts: i) the system admin part, which is responsible for preparing and building the ML model based on Slurm historical data and providing it to the users; ii) the user part, which uses the ML model provided from the system admin part, reads the submitted job script, and predicts the required

 ISBN: 978-1-61208-887-7

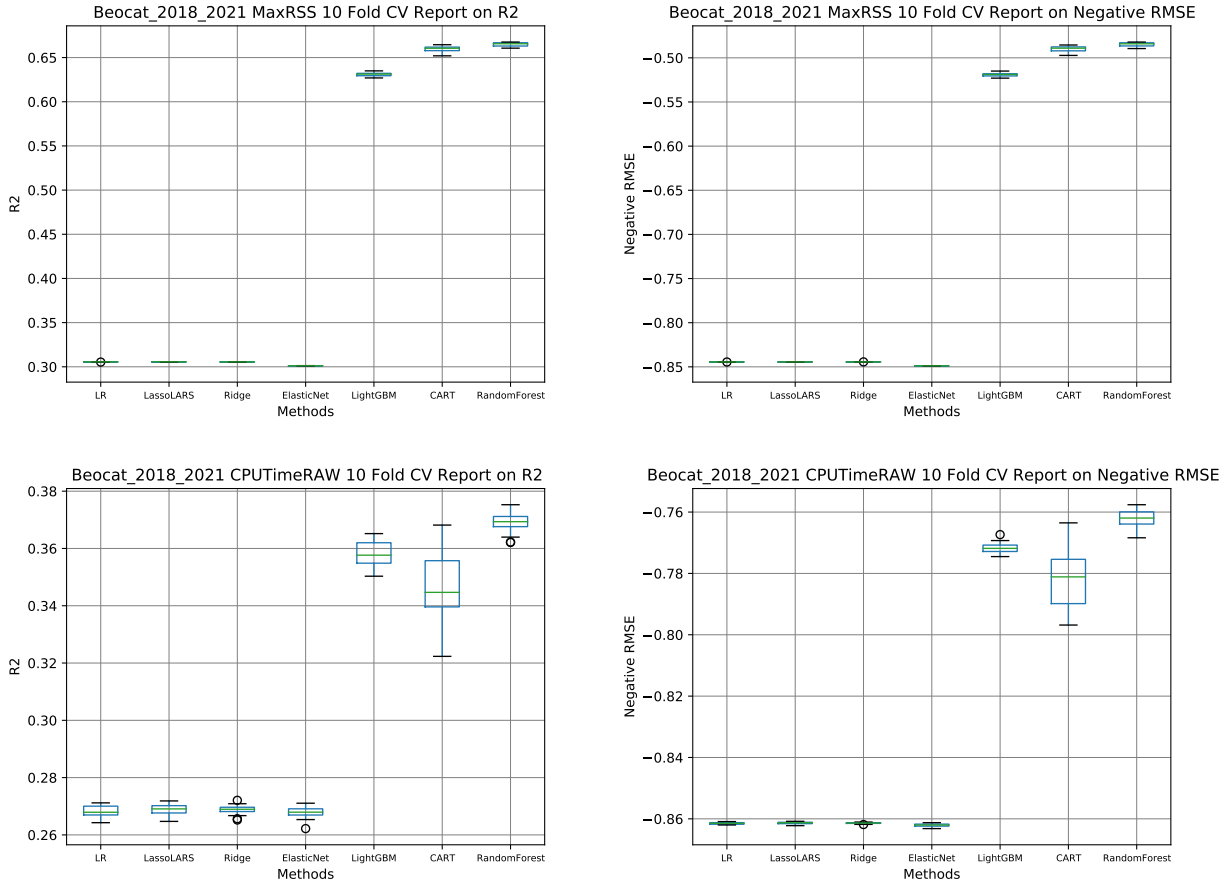


Fig. 3. R^2 and Negative RMSE of Seven Methods Across 21 Accounts in Beocat.

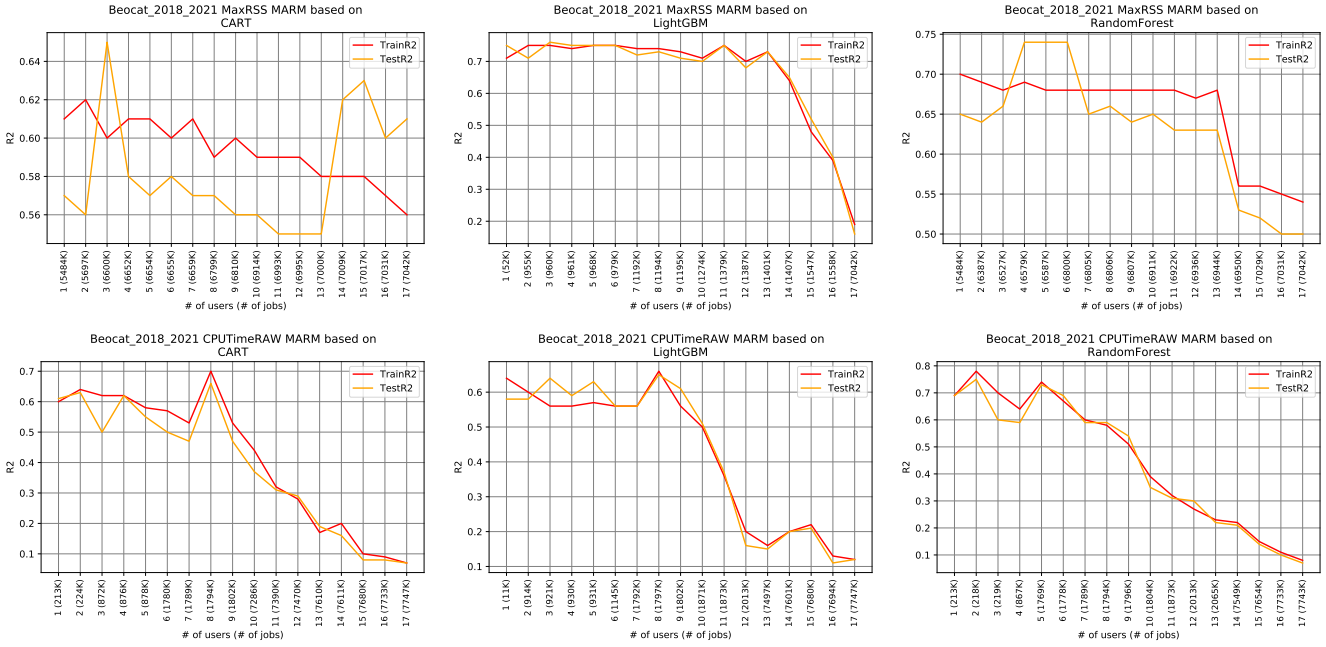


Fig. 4. R^2 Versus Number of Accounts in Predicting Memory and Time Using MARM Across Beocat

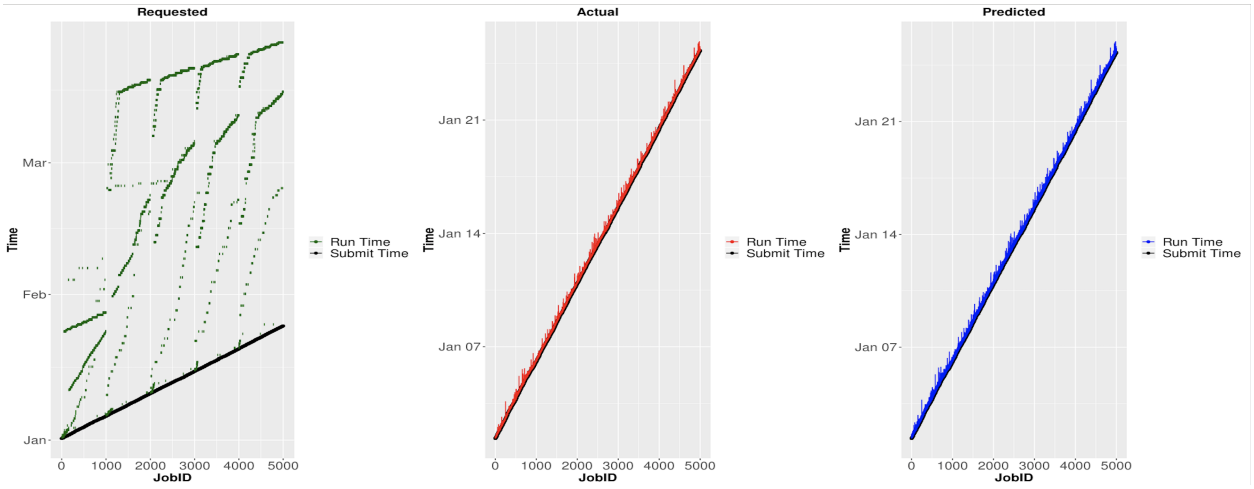


Fig. 5. Jobs Submission and Running Time. (Note Dramatic Improvement of Y Axis Range.)

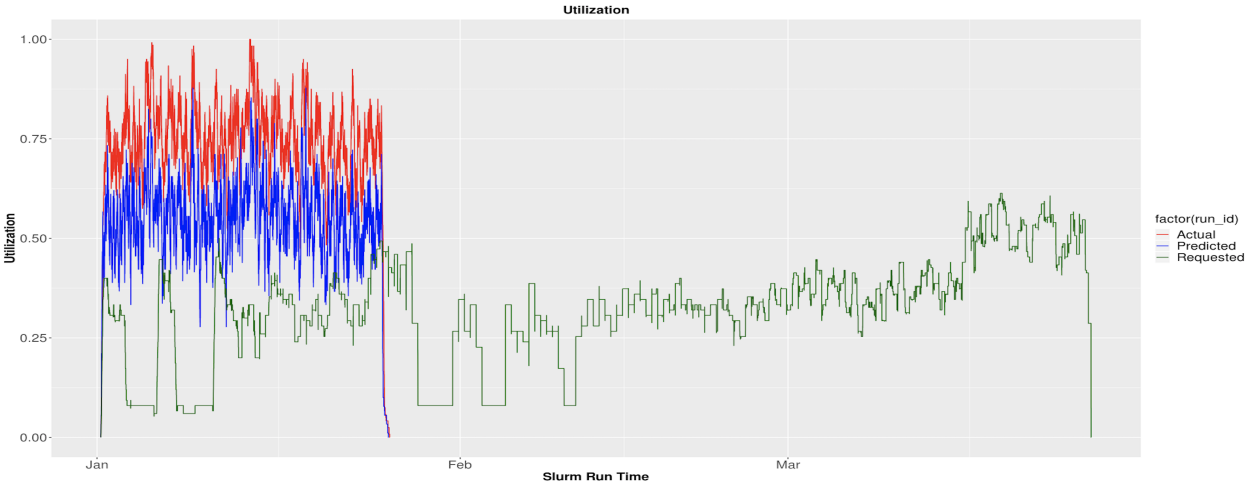


Fig. 6. Utilization (Requested vs Actual vs Predicted) for Beocat Jobs.

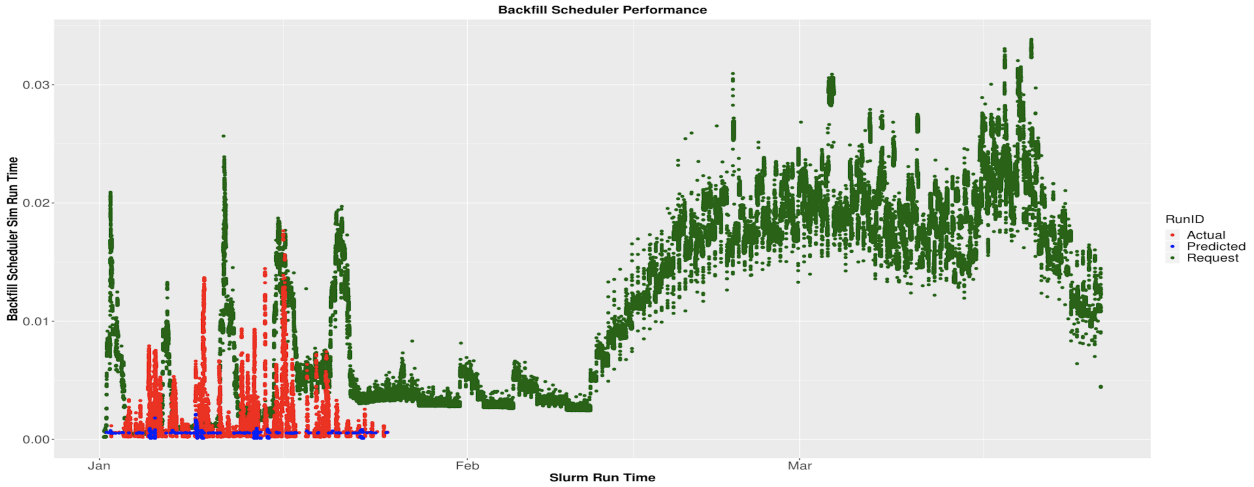


Fig. 7. Backfill-Sched Algorithm Performance (Requested vs Actual vs Predicted) for Beocat Jobs.

 ISBN: 978-1-61208-887-7

TABLE I
AVERAGE WAITING AND TURNAROUND TIME (REQUESTED VS ACTUAL VS PREDICTED) FOR BEOCAT

	Avg Wait Time (Hour)	Avg TA Time (Hour)	Median Wait Time (Hour)	Median TA Time (Hour)
Requested	680 ±128	692.8 ±130	713.6	715.6
Actual	0.4 ±0.08	3.62 ±1.8	0	3.09
Predicted	8.0±1.1	6.36 ±1.9	1.4	5.9

amount of the resources (memory and time). Our tool achieves high accuracy and can significantly increase the performance and utilization of the HPC systems. Moreover, our ML tool can dramatically decrease the average turnaround and waiting time for the submitted jobs. Hence, our tool increases the efficiency and decreases the power consumption of the Slurm-based HPC resources.

ACKNOWLEDGMENT

We thank the HPC staff at KSU, including Adam Tygart and Kyle Hutson, for their help and technical support. We also thank the authors of the Slurm simulator at SUNY Buffalo for releasing their work. This research was supported by NSF awards CHE-1726332, ACI-1440548, CNS-1429316, NIH award P20GM113109, and KSU.

REFERENCES

[1] D. G. Feitelson, D. Tsafrir, and D. Krakov, "Experience with using the parallel workloads archive," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2967–2982, 2014.

[2] C. B. Lee, Y. Schwartzman, J. Hardy, and A. Snavely, "Are user runtime estimates inherently inaccurate?" in *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, 2004, pp. 253–263.

[3] M. Hovestadt, O. Kao, A. Keller, and A. Streit, "Scheduling in hpc resource management systems: Queuing vs. planning," in *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, 2003, pp. 1–20.

[4] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] G. Ke *et al.*, "Lightgbm: A highly efficient gradient boosting decision tree," *Advances in neural information processing systems*, vol. 30, pp. 3146–3154, 2017.

[6] D. Andresen, W. Hsu, H. Yang, and A. Okanlawon, "Machine learning for predictive analytics of compute cluster jobs," *arXiv preprint arXiv:1806.01116*, 2018.

[7] M. Tanash, H. Yang, D. Andresen, and W. Hsu, "Ensemble prediction of job resources to improve system performance for slurm-based hpc systems," in *Practice and Experience in Advanced Research Computing*, 2021, pp. 1–8.

[8] M. Tanash *et al.*, "Improving hpc system performance by predicting job resources via supervised machine learning," in *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*, 2019, pp. 1–8.

[9] tanash1983, "Tanash1983/ampro-hpcc: A machine-learning-tool for predicting job resources on hpc clusters." [Online]. Available: https://github.com/tanash1983/AMPRO-HPCC

[10] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Workshop on job scheduling strategies for parallel processing*. Springer, 2003, pp. 44–60.

[11] "Slurm workload manager - documentation," https://slurm.schedmd.com/, retrieved: 06, 2021.

[12] W. Gentzsch, "Sun grid engine: towards creating a compute power grid," in *Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE Comput. Soc. [Online]. Available: https://doi.org/10.1109/ccgrid.2001.923173

[13] G. Staples, "Torque resource manager," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, 2006, pp. 8–es.

[14] "Torque resource manager," http://www.adaptivecomputing.com/products/torque/, retrieved: 06, 2021.

[15] B. Nitzberg, J. M. Schopf, and J. P. Jones, "Pbs pro: Grid computing and scheduling attributes," in *Grid resource management*. Springer, 2004, pp. 183–190.

[16] "Pbs professional open source project," https://www.pbspro.org/, retrieved: 05, 2021.

[17] J.-W. Park and E. Kim, "Runtime prediction of parallel applications with workload-aware clustering," *The Journal of Supercomputing*, vol. 73, no. 11, pp. 4635–4651, 2017.

[18] T.-P. Pham, J. J. Durillo, and T. Fahringer, "Predicting workflow task execution time in the cloud using a two-stage machine learning approach," *IEEE Transactions on Cloud Computing*, vol. 8, no. 1, pp. 256–268, 2017.

[19] S. Kim, Y.-K. Suh, and J. Kim, "Extes: An execution-time estimation scheme for efficient computational science and engineering simulation via machine learning," *IEEE Access*, vol. 7, pp. 98 993–99 002, 2019.

[20] A. Matsunaga and J. A. Fortes, "On the use of machine learning to predict the time and resources consumed by applications," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE, 2010. [Online]. Available: https://doi.org/10.1109/ccgrid.2010.98

[21] A. Tyryshkina, N. Coraor, and A. Nekrutenko, "Predicting runtimes of bioinformatics tools based on historical data: five years of galaxy usage," *Bioinformatics*, vol. 35, no. 18, pp. 3453–3460, 2019.

[22] M. Naghshnejad and M. Singhal, "Adaptive online runtime prediction to improve hpc applications latency in cloud," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. IEEE, 2018, pp. 762–769.

[23] Q. Wang, J. Li, S. Wang, and G. Wu, "A novel two-step job runtime estimation method based on input parameters in hpc system," in *2019 IEEE 4th International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*. IEEE, 2019, pp. 311–316.

[24] F. Nadeem, D. Alghazzawi, A. Mashat, K. Faqeeh, and A. Almalaise, "Using machine learning ensemble methods to predict execution time of e-science workflows in heterogeneous distributed systems," *IEEE Access*, vol. 7, pp. 25 138–25 149, 2019.

[25] M. H. Hilman, M. A. Rodriguez, and R. Buyya, "Task runtime prediction in scientific workflows using an online incremental learning approach," in *2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 2018, pp. 93–102.

[26] D. Ardagna *et al.*, "Predicting the performance of big data applications on the cloud," *The Journal of Supercomputing*, pp. 1–33, 2020.

[27] Y.-K. Suh, S. Kim, and J. Kim, "Clutch: A clustering-driven runtime estimation scheme for scientific simulations," *IEEE Access*, vol. 8, pp. 220 710–220 722, 2020.

[28] O. Aaziz, J. Cook, and M. Tanash, "Modeling expected application runtime for characterizing and assessing job performance," in *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2018, pp. 543–551.

[29] T. Saillant, J.-C. Weill, and M. Mougeot, "Predicting job power consumption based on rjms submission data in hpc systems," in *International Conference on High Performance Computing*. Springer, 2020, pp. 63–82.

[30] T. Taghavi, M. Lupetini, and Y. Kretchmer, "Compute job memory recommender system using machine learning," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 609–616.

[31] E. R. Rodrigues, R. L. Cunha, M. A. Netto, and M. Spriggs, "Helping hpc users specify job memory requirements via machine learning," in *2016 Third International Workshop on HPC User Support Tools (HUST)*. IEEE, 2016, pp. 6–13.

 ISBN: 978-1-61208-887-7

[32] "Slurm workload manager," retrieved: 04, 2021. [Online]. Available: https://slurm.schedmd.com/sacct.html
[33] B. Efron *et al.*, "Least angle regression," *Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.
[34] G. Bonaccorso, *Machine learning algorithms*. Packt Publishing Ltd, 2017.
[35] W.-Y. Loh, "Classification and regression trees," *Wiley interdisciplinary reviews: data mining and knowledge discovery*, vol. 1, no. 1, pp. 14–23, 2011.
[36] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.
[37] N. A. Simakov *et al.*, "A slurm simulator: Implementation and parametric analysis," in *International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems*. Springer, 2017, pp. 197–217.
[38] "Github - ubccr-slurm-simulator/slurm_simulator: Slurm simulator: Slurm modification to enable its simulation," https://github.com/ubccr-slurm-simulator/slurm$_simulator, retrieved : 05, 2021.

 ISBN: 978-1-61208-887-7

Budget-aware Static Scheduling of Stochastic Workflows with DIET

Yves Caniou*, Eddy Caron*, Aurélie Kong Win Chang*, Yves Robert*†
*ENS Lyon, France
†University of Tennessee, Knoxville, TN, USA
emails: {yves.caniou| eddy.caron| aurelie.kong-win-chang|yves.robert}@ens-lyon.fr

***Abstract*—Previous work has introduced a Cloud platform model and budget-aware static algorithms to schedule stochastic workflows on such platforms. In this paper, we compare the performance of these algorithms obtained via simulation and via execution on an actual platform, Grid'5000. We focus on DIET, a widely used workflow engine, and detail the extensions that were implemented to conduct the comparison. We also detail additional code that we made available in order to automate and to ease the reproducibility of such experiments.**

Keywords – DIET; static workflow scheduling; Cloud platform.

I. Introduction

Public Cloud has emerged as an interesting tool for scientists, offering an infrastructure adaptable on demand, with a variety of available options and performances. Multiple workflow engines for Cloud platforms have been designed, with more and more functionalities [1] - [2]. These workflow engines aim at helping users to pick the most appropriate resources for their intended work, in terms of the number and characteristics of the Virtual Machines (VMs) selected for execution. The main objective is to enable easy-to-produce applications while guaranteeing that, given a constrained budget, rented resources are used up to the maximum of their capacity.

In [3], we described a Cloud platform model, and we detailed and compared several static budget-aware scheduling algorithms in an extensive simulation campaign. One major objective of this work is to further validate the conclusions of the simulation study by conducting real-life experiments with the same scientific workflows on the French national validation platform Grid'5000 [4] and assess the accuracy of simulation results on a real-life platform. Additional contributions are the evolution of DIET (Distributed Interactive Engineering Toolbox) to handle static scheduling, and a tool that automatically generates DIET code to execute the target workflows.

The rest of the paper is organized as follows. We first study related work in Section II. Then, in Section III, we introduce the DIET middleware and describe the new DIET functionalities. In Section IV, we overview the budget-aware algorithms that we aim at comparing. Section V details the experimental framework. Results are reported in Section VI, with a comparison analysis between real executions on Grid'5000 and the corresponding simulations.

II. Related work

A. Workflow engines

Many scientific applications from various disciplines are structured as workflows [5]. Informally, a workflow can be seen as the composition of a set of basic operations that have to be performed on a given input data set to produce the expected scientific result. For a long time, the development of complex middleware with workflow engines [2] [6] [7] has automated workflow management. For example, the Pegasus Workflow Management System [2] maps workflows on resources until a given horizon beyond which it considers pre-scheduling is inefficient, and allows its users to plug their own scheduling algorithms if needed. Steep [8] comes along its own way to schedule workflows, submitting complete process chains to its remote agents, and supports cyclic workflows graphs. Apache Airflow [9] is more oriented on accessibility to most users, hence its focus on the user interface and the use of Python to describe workflows or interact with the engine. In [10], the authors summarize the key features of four production-ready WMSs: Pegasus, Makeflow, Apache Airflow, and Pachyderm. For this work, we focus on DIET [11] because of its practicality, and the possibility to extend it with additional modules.

Infrastructure as a Service (IaaS) Clouds raised a lot of interest recently thanks to an elastic resource allocation and pay-as-you-go billing model. A Cloud user can adapt the execution environment to the needs of his application on a virtually infinite supply of resources. While the elasticity provided by IaaS Clouds gives way to more dynamic application models, it also raises new issues from a scheduling point of view. An execution now corresponds to a certain budget, which imposes some constraints on the scheduling process. In [12], the authors propose a performance-feedback autoscaler that is budget-aware: using Apache Airflow, they tackle the same scheduling problem as in this paper, but they focus on the auto-scaling problem (allocating and de-allocating resources on the fly).

B. Budget aware static algorithms

Scheduling scientific workflows in cloud is a well-studied domain [13] [14]. To the best of our knowledge, the closest papers to the budget-aware algorithms with stochastic execution times that we designed in [3] and compare in this paper are [15] and [16], which both propose workflow scheduling algorithms (Budget Distribution with Trickling – BDT in [15], Critical

 ISBN: 978-1-61208-887-7

Greedy – CG/CG+ in [16]) under budget constraints, but with a simplified platform model. As in our previous work [17], we have extended BDT and CG/CG+ to enable a fair comparison with our algorithms. More recent scheduling algorithms exist, but address different aspects of the problem. For example, [18] aims at simultaneously minimizing the cost and makespan of workflow executions, but they use a simplified platform model without Cloud storage. Another study [19] uses a platform closer to ours, but with a completely different objective, namely the minimization of data transfers (which should eventually reduce both makespan and cost too). Finally, the workflows considered in [20] present an uncertainty in task durations, but the authors schedule multiple workflows at the same time instead of optimizing for a single workflow.

III. Framework

Scientific workflows are represented with a DAG (Directed Acyclic Graph) $G = (V, E)$, where V is the set of tasks to schedule and E is the set of dependencies between tasks. In this model, a dependency corresponds to a data transfer between two tasks. Workflows are scheduled on an heterogeneous platform representing an IaaS Cloud, with a tarification depending on the performances of the used machines, on the amount of data transferred to and out of the Cloud, and on the amount of time during which each machine has been used (see Section IV-B for cost instances).

A. DIET workflow engine

DIET is a well established workflow engine [1] [11]. A DIET platform (Figure 1) consists of a hierarchy of agents [Master Agents MA and Local Agents LA] scheduling the requests addressed by a client and sending them to the appropriate servers, the Servers Deamon (SeD). DIET users, following the GridRPC paradigm, usually submit individual tasks, but it is possible to submit a whole workflow.

In this case, the client sends the XML file which describes its structure to a special agent, called a MA_{DAG}, which consequently manages task dependencies and handles ready tasks submissions to the related MA. The MA dynamically determines which server will execute a request. A server first looks for the files needed for the task execution, downloads them if they are not already on the server, and then executes the request.

B. Extending the MA_{DAG} for static scheduling

DIET was able to schedule workflows dynamically. We improved the MA_{DAG} to apply a static schedule given by a user. From the user point of view, the client sends to the MA_{DAG} the usual files needed for the execution of the workflow, plus files describing the desired schedule:

- The Workflow Description File (WDF): the XML file describing the workflow as in any DIET execution of workflow,
- The Desired Schedule File (DSF): a file giving the desired schedule. Each line gives the machine attributed to a task, under the formulation <task> <server>. The order of the tasks gives their priority, the first ones having the highest priority. The static schedule is also used to write the code of the target servers, hence this file has to be made available before launching execution,
- The Mapping File (MF): a file giving the equivalence between the name of the platform servers and their counterpart from the DSF. Each line gives a machine-server equivalence, under the formulation <machine> <server>.

DIET will then follow the given schedule.

On a more technical level, it is not possible to specify in the MA_{DAG} which server will receive a given task: the MA_{DAG} leaves the duty of selecting a server to its MA, only sending it the name of the services ready to be executed. To alleviate this limitation, we exploit the fact that the MA uses the name of the service to find which server can run it. In order to let the user to be able to change the task-server attribution according to the information gathered by the DIET agents, we choose to have the servers declare their services twice. The first one is a regular generic declaration, with a name identical to the one given in the XML file describing the workflow, and the same for all the servers able to run it. This allows the MA_{DAG} to get the list of all the servers available to run a given service corresponding to a task;The second one is specific to the new functionality and has a name composed of the concatenation between the name of the service as described in the XML file and the name of the server. This local name is the one which will be used to send the request once the server is chosen.

C. Experiment-oriented tools

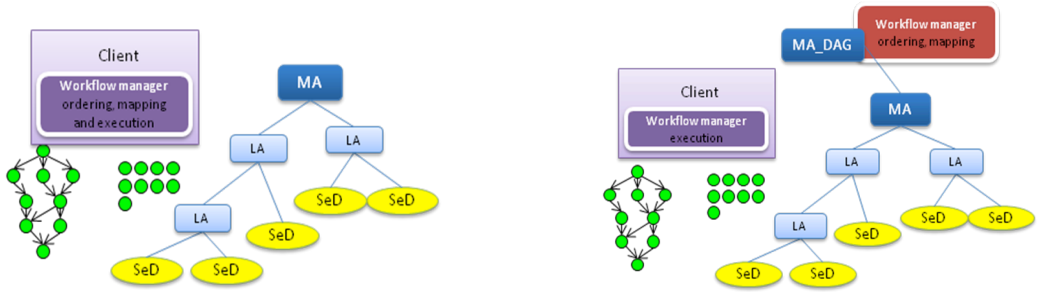
Given the large number of runs needed by the experiments, we have developed a couple of tools to automate the creation of the workflows meant to be executed on Grid'5000. We first use a home-made simulator [21] to generate the schedules with the selected algorithms. We then generate the corresponding DIET workflows needed for our experiments with a home-made workflow generator, Ogma [21].

Our simulator is based on simDAG [22]. It generates both the static schedules given to DIET and the simulated results. Basically, it needs a description of the platform, the file in DAX format [23] describing the workflow, and the parameters of the scheduling problem (budget, algorithms, etc.). It creates a schedule, writes the mapping <task, VM> into a file intended for Ogma, in the order of their attribution, as determined by the chosen algorithm. Then, it simulates with simDAG the execution of the given workflow on the given platform, using the calculated mapping. Finally, it calculates and writes in a file the resulting makespan, cost, and various other metrics.

Once the simulations are done and the schedules calculated, we used Ogma to generate the elements needed for the experiments on Grid'5000. Concretely, Ogma uses the given DAX file describing the target workflow, along with information on the focused platform and simulations, and the static schedule, to write all files: the source files for the servers and the client, the configuration files needed for every DIET entities, placeholder files for the workflow, the DSF, the MF and the WDF.

We point out that, as DAX files do not give any details about the real content of the tasks or the files they describe, Ogma

 ISBN: 978-1-61208-887-7



(a) Architecture 1: Complete schedule from the Client.

(b) Architecture 2: Schedule is handled by the MA_{DAG}

Figure 1: Two different architectures of the DIET workflow engine.

only creates placeholder files and tasks. If a user wants to use Ogma as a tool to generate the raw structure of the workflow, they will have to replace these placeholders with the actual code for the tasks.

IV. Budget-aware scheduling algorithms

Details on this section can be found in [3].

A. Workflow model

A workflow is represented with a DAG of stochastic tasks. Tasks are not preemptive and must be executed on a single processor. In our model, we only know an estimation of the number of instructions for each task. For lack of knowledge about the origin of time variations, we assume that all the parameters which determine the number of instructions forming a task are independent. This resulting number is the *weight* w_i of task T_i and follows a truncated Normal law with mean $\overline{w_i}$ and standard deviation $\sigma_i \overline{w_i}$. Finally, to each dependency $(T_i, T_j) \in E$ is associated an amount of data of size $size(d_{T_i,T_j})$.

B. Cloud platform model

There is only one datacenter, used by all processing units. It is the common crossing point for all the data exchanges between processing units: these units do not interact directly.

The processing units are Virtual Machines (VMs). They can be classified in different categories characterized by a set of parameters fixed by the provider. Some providers offer parameters of their own, such as the number of forwarding rules. We only retain parameters common to the three providers Google, Amazon and OVH: a VM of category k has n_k processors, one processor being able to process one task at a time; a VM has also a speed s_k corresponding to the number of instructions that it can process per time unit, a cost per time-unit $c_{h,k}$ and an initial cost $c_{ini,k}$; all these VMs take an initial, and uncharged, amount of time t_{boot} to boot before being ready to process tasks. Already integrated in the schedule computing process, this starting time is thus not counted in the cost related to the use of the VM. Without loss of generality (even if the VM is paid for each used second), categories are sorted according to hourly costs, so that $c_{h,1} \leq c_{h,2} \cdots \leq c_{h,n_k}$. We expect speeds to follow the same order, but do not make such an assumption.

Altogether, the platform consists of a set of n VMs of k possible categories. Some simplifying assumptions make the model tractable while staying realistic: (i) we assume that the bandwidth is the same for every VM, in both directions, and does not change throughout execution; (ii) a VM is able to store enough data for all the tasks assigned to it: in other words, a VM will not have any memory/space overflow problem, so that every increase of the total makespan will be because of the stochastic aspect of the task weights; (iii) initialization duration is the same for every VM; (iv) data transfers take place independently of computations, hence do not have any impact on processor speeds to execute tasks; (v) a VM executes at most one task at every time-step, but this task can be parallel and enroll many computing resources (hence the execution time of the task strongly depends upon the VM type).

We chose an "on-demand" provisioning system: it is possible to deploy a new VM during the workflow execution. Hence, VMs may have different startup times. A VM v is started at time $H_{start,v}$ and does not stop until all the data created by its last computed task is transferred to the Cloud storage, at time $H_{end,v}$.

C. Scheduling costs and objective

Tasks are mapped to VMs and locally executed in the order given by the scheduling algorithm, such as those described in Section IV-D. Given a VM v, a task is launched as soon as (i) the VM is idle; (ii) all its predecessor tasks have been executed, and (iii) the output files of those predecessors mapped onto others VMs have been transferred to v via the Cloud storage.

a) Costs: The cost model is meant to represent generic features out of the existing offers from Cloud providers (Google, Amazon, OVH). The total cost of the whole workflow execution is the sum of the costs due to the use of the VMs and of the cost due to the use of the Cloud storage $\mathcal{C}_{CS}$. The cost C_v of the use of a VM v of category k_v is calculated as follows:

$$C_v = (H_{end,v} - H_{start,v}) \times c_{h,k_v} + c_{ini,k_v} \qquad (1)$$

There is a startup cost c_{ini,k_v} in Equation (1), and a term c_{h,k_v} proportional to usage duration $H_{end,v} - H_{start,v}$.

The cost for the Cloud storage is based on a cost per time-unit $c_{h,CS}$, to which we add a transfer cost. This transfer cost is

computed with the amount of data transferred from the external world to the Cloud storage (size($d_{in,CS}$)), and from the Cloud storage to the outside world (size($d_{CS,out}$)). In other words, $d_{in,CS}$ corresponds to data that are input to entry tasks in the workflow, and $d_{CS,out}$ to data that are output from exit tasks. Letting $H_{start,first}$ be the moment when we book the first VM and $H_{end,last}$ be the moment when the data of the last processed task have entirely been sent to the Cloud storage, we define $H_{usage} = H_{end,last} - H_{start,first}$ as the total platform usage during the whole execution. We have:

$$C_{CS} = (\text{size}(d_{in,CS}) + \text{size}(d_{CS,out})) \times c_{tsf} + H_{usage} \times c_{h,CS} \quad (2)$$

Altogether, the total cost is $C_{wf} = \sum_{v \in R_{VM}} C_v + C_{CS}$, where R_{VM} is the set of booked VMs during the execution.

b) Objective: Given a budget $\mathcal{B}$ and a platform $\mathcal{P}$, the objective is to minimize total execution time while respecting the budget.

D. The HEFTBUDG scheduling algorithm

HEFTBUDG (Algorithm 1) is a budget-aware extension of the Heterogeneous Earliest Finish Time algorithm (HEFT) [24]. This extension accounts both for task stochasticity and budget constraints, while aiming at makespan minimization. Coping with task stochasticity is achieved by adding a certain quantity to the average task weight so that the risk of under-estimating its execution time is reasonably low, while retaining an accurate value for most executions. We use a conservative value for the weight of a task T, namely $\overline{w_T} + \sigma_T \overline{w_T}$.

Algorithm 1 HEFTBUDG.

```
1: function HEFTBUDG(wf, B_calc, P)
2:     s̄ ← calcMeanSpeed(P)
3:     bw ← getBw(P)
4:     budgPTsk ← divBudget(wf, B_calc, s̄, bw)
5:     ListT ← getTasksSortedByRanks(wf, s̄, bw, lat)
6:     pot, newPot ← 0
7:     for each T of ListT do
8:         host ← getBestHost(T, budgPTsk[T], P, newPot)
9:         pot ← newPot
10:        sched[T] ← host
11:        schedule(T, host)
12:        update(UsedVM)
13:    end for
14:    return ListT, sched
15: end function
```

In the beginning HEFTBUDG calls *divBudget()* (Algorithm 2): given the workflow wf, we first get the maximum of total work (getMaxTotalWork(wf)) and the total amount of data transfers (getMaxTotalTransfData (wf)) required to execute the workflow, and we reserve a fraction of the budget to cover the cost of the Cloud storage and VM initialization; then we divide what remains, $\mathcal{B}_{calc}$, into the workflow tasks. To estimate the fraction of budget to be reserved, assuming that $\mathcal{B}_{ini}$ denotes the initial budget:

- For the cost of the Cloud storage, we need to estimate the duration $H_{usage} = H_{end,last} - H_{start,first}$ of the whole execution (see Equation (2)). To this purpose, we consider an execution on a single VM of the first (cheapest) category, compute the total duration $W_{max} = \sum_{T \in wf} (\overline{w_T} + \sigma_T)$ and let
$$H_{usage} = \frac{W_{max}}{s_1} + \frac{\text{size}(d_{in,CS}) + \text{size}(d_{CS,out})}{bw} \quad (3)$$
Altogether, we pay the cost of input/output data several times: with factor c_{tsf} for the outside world, with factor $c_{h,CS}$ for the usage of the Cloud storage (Equation (3)), and with factor $c_{h,1}$ during the transfer of data to and from the unique VM. However, there is no communication internal to the workflow, since we use a single VM.
- For the initialization of the VMs, we assume a different VM of the first category per task, hence we budget the amount $n \times c_{ini,1}$.

Combining these two choices is conservative: on the one hand, we consider a sequential execution, but account only for input and output data with the external world, eliminating all internal transfers during the execution; on the other hand, we reserve as many VMs as tasks, ready to pay the price for parallelism, at the risk of spending time and money due to data transfers during the execution. Altogether, we reserve the corresponding amount of budget and are left with $\mathcal{B}_{calc}$ for the tasks.

This reduced budget $\mathcal{B}_{calc}$ is shared among tasks in a proportional way: we estimate how much time $t_{calc,T}$ is required to execute each task T, transfer times included, and allocate the corresponding part of the budget in proportion to the whole for execution of the entire workflow $t_{calc,wf}$:

$$budgPTsk[T] = \frac{t_{calc,T}}{t_{calc,wf}} \times \mathcal{B}_{calc} \quad (4)$$

In Equation (4), we use $t_{calc,T} = \frac{\overline{w_T} + \sigma_T}{\overline{s}} + \frac{size(d_{pred,T})}{bw}$, where

$$size(d_{pred,T}) = \sum_{(T',T) \in E} size(d_{T',T}) \quad (5)$$

is the volume of input data of T from all its predecessors. Similarly, we use $t_{calc,wf} = \frac{W_{max}}{\overline{s}} + \frac{d_{max}}{bw}$, where $d_{max} = \sum_{(T_i,T_j) \in E} size(d_{T_i,T_j})$ is the total volume of data within the workflow. Computed weights ($\overline{w_T} + \sigma_T$ and W_{max}) are divided by the mean speed $\overline{s}$ of VM categories, while data sizes ($size(d_{pred,T})$ and d_{max}) are divided by the bandwidth bw between VMs and the Cloud storage. Again, it is conservative to assume that all data will be transferred, because some of them will be stored in-place inside VMs, so there is here another source of over-estimation of the cost. On the contrary, using the average speed $\overline{s}$ in the estimation of the computing time may lead to an under-estimation of the cost when cheaper/slower VMs are selected.

This subdivided budget is then used to choose the best VM to host each ready task by calling *getTaskssortedbyranks()*: the best host for a task T will be the one providing the best Earliest Finish Time (EFT) for T, among those respecting the amount of budget $\mathcal{B}_T$ allocated to T (considering all already used VMs plus one fresh VM of each category).

 ISBN: 978-1-61208-887-7

Algorithm 2 Dividing the budget onto tasks.

function DIVBUDGET($wf, \mathcal{B}_{calc}, \overline{s}, bw$)
  $W_{max} \leftarrow$ getMaxTotalWork(wf)
  $d_{max} \leftarrow$ getMaxTotalTransfData (wf)
  for each T **of** wf **do**
    $budgPTsk[T] \leftarrow \mathcal{B}_{calc} \times \frac{\frac{\overline{w_T}+\sigma_T\overline{w_T}}{\overline{s}} + \frac{size(d_{pred,T})}{bw}}{\frac{W_{max}}{\overline{s}} + \frac{d_{max}}{bw}}$
  end for
  return $budgPTsk$
end function

HEFTBUDG reclaims any unused fraction of the budget consumed when assigning former tasks: this is the role of the variable *pot*, which records any leftover budget in previous assignments. For some tasks, getBestHost() do not return the host with the smallest EFT, but instead the host with the smallest EFT among those that respect the allotted budget. The complexity of HEFTBUDG is $O((n+e)p)$, where n is the number of tasks, e is the number of dependence edges, and p the number of enrolled VMs. This complexity is the same as for the baseline versions, except that p is not fixed *a priori*. In the worst case, $p = O(\max(n, k))$ because for each task we try all used VMs, whose count is possibly $O(n)$, and k new ones, one per category.

E. Other scheduling algorithms

[3] details several budget-aware scheduling algorithms:

- MINMINBUDG, a budget-aware extension of MINMIN [25], [26], the exact counterpart of HEFTBUDG.
- HEFTBUDG+ and HEFTBUDG+INV, aiming at exploiting the opportunity to re-schedule some tasks onto faster VMs, thereby spending any budget leftover by the first allocation, at a price of higher complexity. These refined variants differ by the order in which tasks are considered, and recompute the schedule after processing each task.
- HEFTBUDGMULT, a trade-off version that reallocates the leftover budget in a single pass: it finds makespans slightly larger than those computed with HEFTBUDG+, but with a time complexity close to HEFTBUDG.

Two competitors to the new budget-aware algorithms described above have also been simulated in [3], for the sake of comparison. These are Budget Distribution with Trickling (BDT [15]) and Critical Greedy (CG [16]). Both BDT and CG schedule deterministic workflows, and CG does not take into account communication costs. In [16], CG also comes with a refined version CG+. BDT and CG/CG+ have been extended to fit the model, so as to enforce fair comparisons.

V. EXPERIMENTAL FRAMEWORK

We use the new DIET functionalities to execute the scheduling algorithms (Section IV) on Grid'5000, a French national testbed supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations [4]. For each workflow execution, we used 31 homogeneous nodes from the `grisou` cluster in Nancy (Intel Xeon E5-2630 v3 CPU 2.4 GHz $\times 8$ cores). The three different types of VMs (see Table I) needed for the experiments have been emulated: a VM two times slower than another one executes twice as many computations. We also run the corresponding experiments with our simulator.

TABLE I: PARAMETERS OF THE PLATFORM.

VM parameters	
Categories	$k = 3$
Setup cost	$c_{ini,=}$ \$0.00056
Category 1 (Slow)	Speed $s_1 = 3.2$ Gflops Cost $c_{h,1} =$ \$0.118 per hour
Category 2 (Medium)	Speed $s_2 = 6.4$ Gflops Cost $c_{h,2} =$ \$0.236 per hour
Category 3 (Fast)	Speed $s_3 = 9.6$ Gflops Cost $c_{h,3} =$ \$0.354 per hour
Cloud storage	
Cost per month	$c_{h,CS} =$ \$0.022 per GB
Data transfer cost	$c_{tsf} =$ \$0.055 per GB
Bandwidth	
bw	1GBps

We used three types of workflows: MONTAGE, LIGO and CYBERSHAKE, generated with [27]. Given the large makespan of these workflows (e.g., 33 hours for one CYBERSHAKE of 60 tasks scheduled with CG/CG+ [3]), we only ran small instances with 30 tasks.

A. Simulations

We used the simulator described in Section III-C to obtain both the static schedules to be evaluated on Grid'5000 and the simulated results, with characteristics of the platform (bandwidth: 1Gb/s, performances of the VMs used as slowest type: 3.2Gf) experimentally measured on `grisou`.

B. Grid'5000 experiments

One needs to enforce that tasks have similar durations in the simulations and in the experiments. In the workflow description file, the amount of work of a task is given in seconds, when the amount of data transferred between two tasks is given in bytes. Thus for Ogma, a task consists of three phases: a phase during which the input files are read, a phase during which an amount of double floating-point additions is calculated based on the task duration given in the DAX file and the information provided about the used platform, and a phase during which the output files are written.

SimDAG uses a fixed (but arbitrary) number to calculate a number of flops based on the task duration in seconds given by the DAX file. This arbitrary number did not correspond to the VMs enrolled on Grid'5000. To preserve a similar ratio between the time used to move data $t_{data,T}$ and the time used to execute a task $t_{calc\_only,T}$ in the real setup and in the simulation, we modified the number of operations for each service according to the observed characteristics of the platform.

In the simulator, the amount of time $t_{calc\_only,T}$ used to execute the task T, composed of w_T operations, on a reference host of speed $s_{host,simu}$ operations per second, and without counting data transfers, is equal to $t_{calc\_only,T} = \frac{w_T}{s_{host,simu}}$. The amount of time $t_{data,T}$ to transfer all the data needed to execute T, for a total size of $size(d_{pred,T})$, with a bandwidth

 ISBN: 978-1-61208-887-7

bw_{simu} is equal to $t_{data,T} = \frac{size(d_{pred,T})}{bw_{simu}}$. Hence the ratio to preserve is $r_{simu} = \frac{w_T}{size(d_{pred,T})} \times \frac{bw_{simu}}{s_{host,simu}}$.

Similarly, in the Grid'5000 execution, the ratio for a bandwidth bw_{g5000} and a reference host speed $s_{host,g5000}$, with the same amount of transferred data $size(d_{pred,T})$ and a task T composed of $w_{T,g5000}$ operations is $r_{g5000} = \frac{w_{T,g5000}}{size(d_{pred,T})} \times \frac{bw_{g5000}}{s_{host,g5000}}$. Finally, we calculated the number of operations needed to execute the task T on a reference host of Grid'5000 as $w_{T,g5000} = w_{T,simu} \times \frac{s_{host,g5000}}{s_{host,simu}} \times \frac{bw_{simu}}{bw_{g5000}}$. We used this number of operations to generate workflow tasks of appropriate duration.

In the case of LIGO and CYBERSHAKE, instead of running an equally long workflow as described in the DAX files, we chose to generate one with the same shape (same tasks, dependencies between tasks, proportion between data transfers and amount of work for tasks), but whose malespan is made shorter, with the application of a fixed coefficient to decrease the time of execution of the workflows. As seen in Section VI, this has close to no impact on the performance of the algorithms.

As per the execution itself with DIET, we used one VM to run the client task, the MA, the MA_{DAG} agent and omniNames (the naming service on which DIET relies to operate communication between its components), and one VM per server.

C. Experimental campaign

We have generated schedules: (i) for three workflow types: CYBERSHAKE, LIGO and MONTAGE; and (ii) for ten scheduling algorithms: MINMINBUDG, HEFTBUDG, HEFTBUDGMULT, HEFTBUDG+, HEFTBUDG+INV, BDT, MINMIN, CG, CG+ and HEFT. We have selected a range of budget values which have an actual impact on the makespan. For some of these budget values, a few algorithms are failing to produce valid schedules. Here we define a *valid* schedule as a schedule which successfully enforces the budget constraint.

We have executed, on Grid'5000 and on the simulator, 30 experiments per combination (budget $\times$ algorithm $\times$ workflow), and collected the makespan and cost of each execution. The costs are calculated as in [3], with prices adapted to the performances of the used VMs. The raw data, their treatment, and the whole experimentation setup, are available in [21].

VI. RESULTS

We first focus on the observation of makespan results from real life experiments compared to their simulation. Even results that spend more than the allocated budget are considered (in other words, these results are produced by non-valid schedules). Then we assess the accuracy of experimental costs conducted with the MONTAGE workflow, with a comparison between real life experiments and their simulation, and in this case we restrict to results under the given budget.

The experiments have been carried so that the generated workflows are equivalent to their simulated counterpart, with a fixed factor as only difference. For the sake of comparison, and to highlight how similar the obtained results are between the simulation and the execution on Grid'5000, we scale them in the figures: MONTAGE makespans are 4.6 times lower than their simulation, LIGO ones are 49.06 times lower, and CYBERSHAKE ones 6.9 times lower.

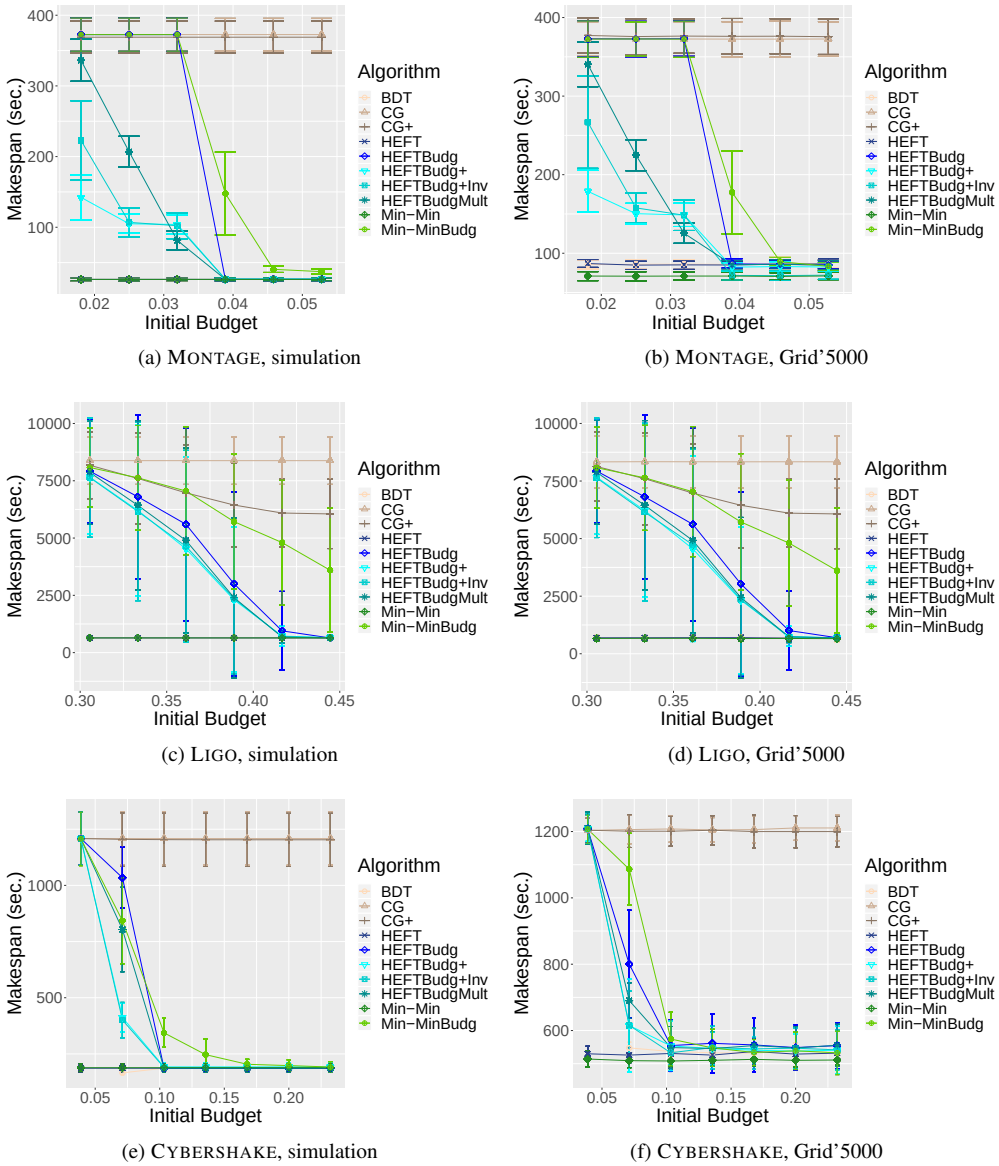
In Figure 2, we report the makespans obtained for each type of workflow as a function of the available budget. The first column represents the results obtained with the simulator. The second column represents the results obtained from the runs with DIET on Grid'5000. In most cases, the hierarchy of algorithms in the DIET executions is the same as the one in simulation. CG and CG+ obtain the highest makespans, and HEFT, BDT and MINMIN the lowest ones. Among the budget-aware algorithms, MINMINBUDG gets most of the time the highest makespans, but is inferior to the ones obtained with CG+. HEFTBUDG obtains the second highest ones. HEFTBUDG+ and HEFTBUDG+INV find the schedules with the lowest makespans. HEFTBUDGMULT schedules have makespans between HEFTBUDG and HEFTBUDG+ ones.

While the results of simulation and real execution are very similar for MONTAGE and LIGO, there are some differences for CYBERSHAKE. We guessed that these differences are due to file transfers, and we reran the simulations with an infinite bandwidth, but this had no impact. Still, overall, there is a pretty good correspondence between simulations and actual runs.

Next, we focus on MONTAGE workflows, and in Figure 3, we report the costs and the percentage of valid solutions found for MONTAGE executions. In Figures 3a and 3b, we see the cost of the execution of MONTAGE workflows, both for simulations and real executions, and they match almost perfectly. With low budgets, two algorithms achieve very expensive schedules: BDT and HEFT. They are followed by MINMIN. All the other budget-aware algorithms, spend twice less budget to make a schedule. In addition, the higher the budget, the more optimization opportunities for budget-aware algorithms. For the highest budgets used for our experiments, we make the following observations: (i) HEFTBUDG+ achieves the most expensive schedules, even higher than the ones from HEFT and BDT (but recall from Figure 2 that HEFTBUDG+ needed a lower initial budget than HEFT and BDT to find valid makespans); (ii) HEFTBUDG, HEFTBUDGMULT and HEFTBUDG+INV have a cost similar to HEFT (but achieve lower makespans); (iii) Similarly, MINMINBUDG and MINMIN schedules have similar costs (and lower makespans for MINMINBUDG); and (iv) The cheapest schedules come from CG and CG/CG+ (but this is at the cost of a far larger makespan).

Figures 3c and 3d show the percentage of valid schedules found by each algorithm, from simulations (left) and real-life experiments (right). Only HEFTBUDG+INV differs on the two lowest budgets without a 100% valid schedules in real-life. We know from [3] that this algorithm refines its schedule, leaving only a small leftover budget, which explains the difference. On each graph, we see that for the lowest budget, BDT, HEFT and MINMIN give no valid schedule, and for the second lowest budget, BDT and HEFT still do not. All the other algorithms give 100% valid solutions.

 ISBN: 978-1-61208-887-7



(a) MONTAGE, simulation

(b) MONTAGE, Grid'5000

(c) LIGO, simulation

(d) LIGO, Grid'5000

(e) CYBERSHAKE, simulation

(f) CYBERSHAKE, Grid'5000

Figure 2: Makespans for Montage, Cybershake and Ligo workflows of 30 tasks, execution on simulation vs. GRID'5000.

VII. CONCLUSION

In this paper, we have introduced a new scheduling functionality for DIET, and provided the user with a set of tools to implement, and experiment with, static scheduling algorithms for workflows. We then used this new functionality to compare the executions of ten static algorithms for scientific applications from the Pegasus benchmark, using both a simulator and the testbed Grid'5000. Both types of experiments gave similar results, validating the results obtained during the simulations executed in [5] and DIET improvements.

Further work will be devoted to better understand the behavior of budget-aware algorithms on larger and more diverse workflows, using the insights gained from both the similarities and differences found in simulations and actual executions.

REFERENCES

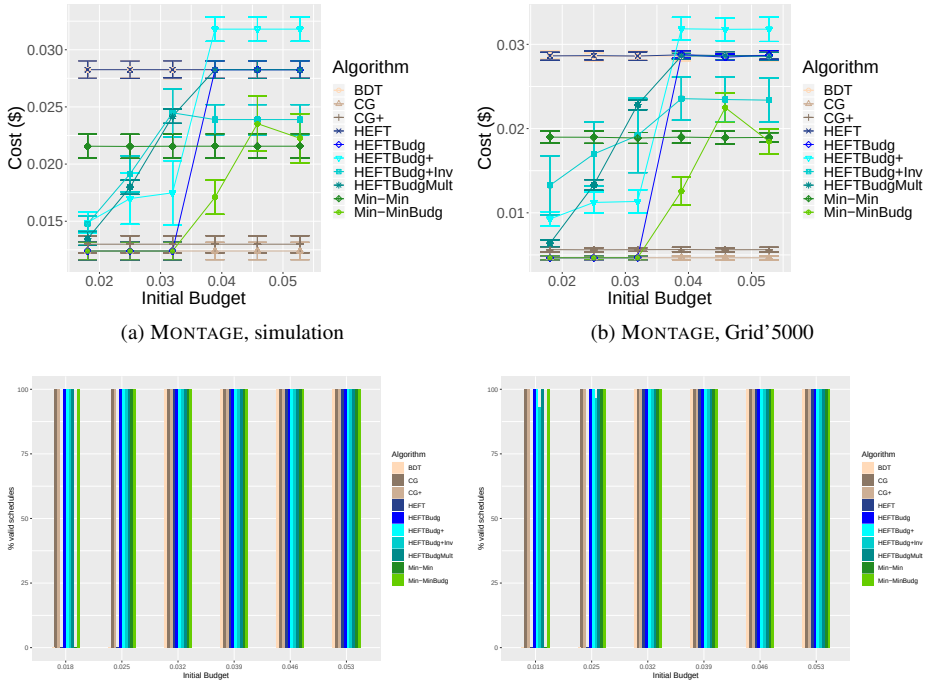
[1] E. Caron and F. Desprez, "DIET: A scalable toolbox to build network enabled servers on the grid," *International Journal of High Performance Computing Applications*, vol. 20, no. 3, pp. 335–352, 2006.

[2] E. Deelman *et al.*, "Pegasus: a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, 2015, funding Acknowledgements: NSF ACI SDCI 0722019, NSF ACI SI2-SSI 1148515 and NSF OCI-1053575.

[3] Y. Caniou, E. Caron, A. Kong Win Chang, and Y. Robert, "Budget-aware scheduling algorithms for scientific workflows with stochastic task weights on infrastructure as a service cloud platforms," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 17, p. e6065, 2021, [retrieved: August, 2021]. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.6065

[4] F. Cappello *et al.*, "Grid'5000: A large scale, reconfigurable, controlable and monitorable grid platform," in *Proc. 6th IEEE/ACM Int. Workshop on Grid Computing (Grid'2005)*. IEEE Computer Society Press, 2005, see https://www.grid5000.fr/w/Grid5000:Home.

 ISBN: 978-1-61208-887-7



(a) MONTAGE, simulation

(b) MONTAGE, Grid'5000

(c) MONTAGE, percentage of valid solutions, simulation

(d) MONTAGE, percentage of valid solutions, Grid'5000

Figure 3: Costs for Montage workflows of 30 tasks, execution on grid'5000 vs. simulation, and percentage of valid solutions.

[5] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *SC'08 Workshop: The 3rd Workshop on Workflows in Support of Large-scale Science (WORKS08) web site*. Austin, TX: ACM/IEEE, Nov. 2008.

[6] E. Caron, F. Desprez, T. Glatard, M. Ketan, J. Montagnat, and D. Reimert, "Workflow-based comparison of two distributed computing infrastructures," in *Workflows in Support of Large-Scale Science (WORKS10)*, In Conjunction with Supercomputing 10 (SC'10). New Orleans: IEEE, November 14 2010, hal-00677820, [retrieved: August, 2021]. [Online]. Available: https://hal.inria.fr/hal-00677820

[7] P. Couvares, T. Kosar, A. Roy, J. Weber, and K. Wenger, "Workflow Management in Condor," in *Workflows for e-Science*, I. Taylor, E. Deelman, D. Gannon, and M. Shields, Eds. Springer, 2007, pp. 357–375.

[8] M. Krämer, "Capability-based scheduling of scientific workflows in the cloud," in *DATA*, S. Hammoudi, C. Quix, and J. Bernardino, Eds. SciTePress, 2020, pp. 43–54.

[9] M. Beauchemin. (2014) Apache airflow project. [retrieved: August, 2021]. [Online]. Available: https://airflow.apache.org/

[10] R. Mitchell *et al.*, "Exploration of workflow management systems emerging features from users perspectives," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 4537–4544.

[11] E. Caron, "Contribution to the management of large scale platforms: the DIET experience," HDR (Habilitation 'a Diriger les Recherches), École Normale Supérieure de Lyon, Oct.6 2010, hal number tel-00629060, [retrieved: August, 2021]. [Online]. Available: https://hal.inria.fr/tel-00629060

[12] A. Ilyushkin, A. Bauer, A. V. Papadopoulos, E. Deelman, and A. Iosup, "Performance-feedback autoscaling with budget constraints for cloud-based workloads of workflows," *CoRR*, vol. abs/1905.10270, 2019, [retrieved: August, 2021]. [Online]. Available: http://arxiv.org/abs/1905.10270

[13] C. Lin and S. Lu, "Scheduling scientific workflows elastically for cloud computing," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 746–747.

[14] S. Smanchat and K. Viriyapant, "Taxonomies of workflow scheduling problem and techniques in the cloud," *Future Generation Computer Systems*, vol. 52, pp. 1–12, 2015.

[15] V. Arabnejad, K. Bubendorfer, and B. Ng, "Budget distribution strategies for scientific workflow scheduling in commercial clouds," in *IEEE 12th Int. Conf. on e-Science (e-Science)*, Oct 2016, pp. 137–146.

[16] C. Q. Wu, X. Lin, D. Yu, W. Xu, and L. Li, "End-to-end delay minimization for scientific workflows in clouds under budget constraint," *IEEE Transactions on Cloud Computing*, vol. 3, no. 2, pp. 169–181, April 2015.

[17] T. Risset and Y. Robert, "Synthesis of processor arrays for the algebraic path problem," *Parallel Processing Letters*, vol. 1, no. 1, pp. 19–28, Sep. 1991.

[18] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, and S. Hu, "Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based heft," *Future Generation Computer Systems*, vol. 93, pp. 278 – 289, 2019.

[19] S. Makhlouf and B. Yagoubi, "Data-aware scheduling strategy for scientific workflow applications in iaas cloud computing," *Int. J. Interactive Multimedia and Artificial Intelligence*, vol. InPress, p. 1, 01 2018.

[20] J. Liu *et al.*, "Online multi-workflow scheduling under uncertain task execution time in iaas clouds," *IEEE Trans. Cloud Computing*, pp. 1–1, 2019.

[21] A. Kong Win Chang. https://graal.ens-lyon.fr/~achang/Research/. [retrieved: August, 2021].

[22] SimDag, "Programming environment for DAG applications," http://simgrid.gforge.inria.fr/simgrid/3.13/doc/group__SD__API.html, 2017, [retrieved: 2017].

[23] Pegasus Team, https://pegasus.isi.edu/documentation/development/schemas.html, [retrieved:2021].

[24] H. Topcuoglu, S. Hariri, and M. Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distributed Systems*, vol. 13, no. 3, pp. 260–274, 2002.

[25] T. Braun *et al.*, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810–837, 2001.

[26] P. Ezzatti, M. Pedemonte, and A. Martín, "An efficient implementation of the min-min heuristic," *Comput. Oper. Res.*, vol. 40, no. 11, 2013.

[27] Pegasus Team, "Code for the Pegasus generator," https://github.com/pegasus-isi/WorkflowGenerator, 2020, [retrieved: August, 2021].

 ISBN: 978-1-61208-887-7

Synapse

Facilitating Large-Scale Data Management in Research Contexts

Daniel Andresen
Dept. of Computer Science
Kansas State University
Manhattan KS, USA
e-mail: dan@ksu.edu

Gerrick Teague
Software Development Dept.
Able Ant Design, Inc.
Manhattan KS, USA
e-mail: gerrick@ableant.com

***Abstract*— Research data management is becoming increasingly complex as the amount of data, metadata and code increases. Often, researchers must obtain multidisciplinary skills to acquire, transfer, share, and compute large datasets. In this paper, we present the results of an investigation into providing a familiar web-based experience for researchers to manage their data and code, leveraging popular, well-funded tools and services. We show how researchers can save time and avoid mistakes, and we provide a detailed discussion of our system architecture and implementation, and summarize the new capabilities, and time savings which can be achieved.**

Keywords- Globus; Dataverse; oid; metadata; research.

I. INTRODUCTION

Data management and processing increasingly consumes a modern researcher's time. Common tasks may include storing and sharing the data, generating metadata, developing codes to process the data, interfacing with High Performance Computing (HPC) clusters to process the data, access control, publishing and making the project discoverable to other researchers. Researchers often need to obtain significant knowledge outside of their domain in order to manage the data, often including command line interfaces, data transfer tools, and repositories, while other necessary tools, like data backup, and access control often go unutilized due to the hassle.

There currently exist tools to handle specific portions of the modern researcher's workflow in a generalized sense. For example, there are cloud data backup services, and data sharing tools such as Dropbox, but these services are not designed for a research context, so features, such as compliance with the Health Insurance Portability and Accountability Act (HIPAA), and fine-grained access control may be lacking. A further problem is the necessity to learn these different systems.

Technology is catching up to address the problems outlined above. Open OnDemand [2] provides a friendly graphical user interface when dealing with HPC clusters, but lacks robust data transfer. Harvard's Dataverse project [7] is an attempt to house, publish, and corroborate research, but lacks big data support. Ideally, there is a solution that handles most of the data flow for a research project to minimize the amount of out-of-wheelhouse technology a researcher must learn, but still be general purpose enough to provide benefit to multiple labs.

In this paper, we present Synapse, an open-source, web-based application. Synapse leverages both industry standards and emerging tools to provide a familiar, intuitive interface for researchers to handle common tasks. It greatly lowers the barrier of entry into many solutions including data transfer, HPC computing, data backup and housing, access controls, auto-metadata extraction, and research discovery [9].

Synapse achieves the above objectives by integrating three key technologies: 1) Dataverse handles the storage, providence, access control, and discoverability of research projects. 2) Globus [4] provides secure transfer of large amounts of data between given endpoints. 3) Open OnDemand provides a graphical user interface for processing researcher data on high performance computing clusters. We summarize our contributions as follows:

- We present current technologies that attempt to tackle this problem.
- We present the design, implementation and evaluation of a Synapse installation.
- We detail a modular metadata extraction system that can be extended to a particular lab's needs.
- We detail further work that can improve the system.

The rest of this paper first discusses related work in Section 2, and then describes our implementation in Section 3. Section 4 describes how we evaluated our system and the results. Section 5 presents our conclusions and describes future work.

II. RELATED WORK

Because the problem domain touches so many different areas, including data transfer, backup, storage, computing, discoverability, providence, and permissions, as well as the potentially significant resources required to implement such systems for data-large research projects, we find many excellent projects that handle a subset of the problem domain. The pace of improvement is very brisk in this space; technologies at the time of this writing may have overcome limitations reviewed here. Lastly, our search passed over many excellent products in favor of ease of integration, and market penetration. Our labs' requirements are specific: Storing and preservation of research data, Integration with our local HPC cluster, as well as auto metadata extraction from files. As such, we could not find a software solution that checked off all our requirements, but we found several excellent general-use solutions that came very close. Our

 ISBN: 978-1-61208-887-7

planning naturally went to discovering how to best integrate these projects into our solution.

The Center for Open Science produces the Open Science Framework (OSF) - an open-source web application that assists in research collaboration, document storage and archive, as well as registration of research projects. The application is very intuitive and supports a wide range of data sharing/transfer/storage technologies including Dropbox, and Google Drive. It handles persistent Uniform Resource Locators (URLs) for citing and sharing, access controls, version sharing, as well as publishing reports. In fact, in our research, OSF was a final candidate to incorporate into our system, since it fit most of our requirements. The OSF approach is to be the hub of data linked offsite in pre-existing storage shares, such as Dropbox, or Simple Storage Service (S3). Data can be hosted directly by OSF, but file sizes are limited to 5 gigabytes (GB). Overall, this system is very intuitive and flexible. Unfortunately, OSF is not currently designed to handle big data (neither very large data files, nor thousands of smaller files). Finally, market penetration seems low enough, with funding scarce enough, to warrant caution adopting this technology long term [6].

Dataverse is a project allowing users to store, discover, share, and analyze data. It is maintained by the Institute for Quantitative Social Science (IQSS) at Harvard University [3]. Dataverse strengths include wide adoption by academic research institutions, a vibrant community, rich documentation, constant funding, a robust permissions system, and a modest development pace. Its main weakness was its lack of big data support. But in recent months, it has made large strides to compensate. Dataverse's User Interface (UI) is a bit dated and slow, and the support for downloading many files is troublesome, but an active user base and development team can overcome these issues. Fig. 1 gives an example of the Dataverse UI. The structure of a Dataverse installation consists of a root Dataverse, which is a container of any combination of sub-Dataverses and datasets, the latter of which can only contain files and metadata. Due to its "batteries included" approach as well as its open and large community, we decided to leverage Dataverse for most of our needs, utilizing other software and connective code to flesh out our solution [7].

GitHub is a popular cloud-based version control repository. It extends Git, the de facto version control system for software code, which provides providence, collaboration, versioning, and powerful merging workflows to manage data. GitHub uses Git as its core, but also provides a powerful web-based UI to hide - yet complement - many of Git's powerful features. Being a web-first technology, it allows code to be discoverable by search engines, as well as providing gateway functionality with authentication and permissions. It also provides a highly programmable infrastructure to send and receive data. In a lot of ways, GitHub provides most everything needed for researchers to utilize. Unfortunately, due to technical limitations neither Git nor GitHub can handle files of any size. GitHub recommends repositories to be less than 1 GB, with a hard file size limit of no more than 100 megabytes (MB) per file [1].

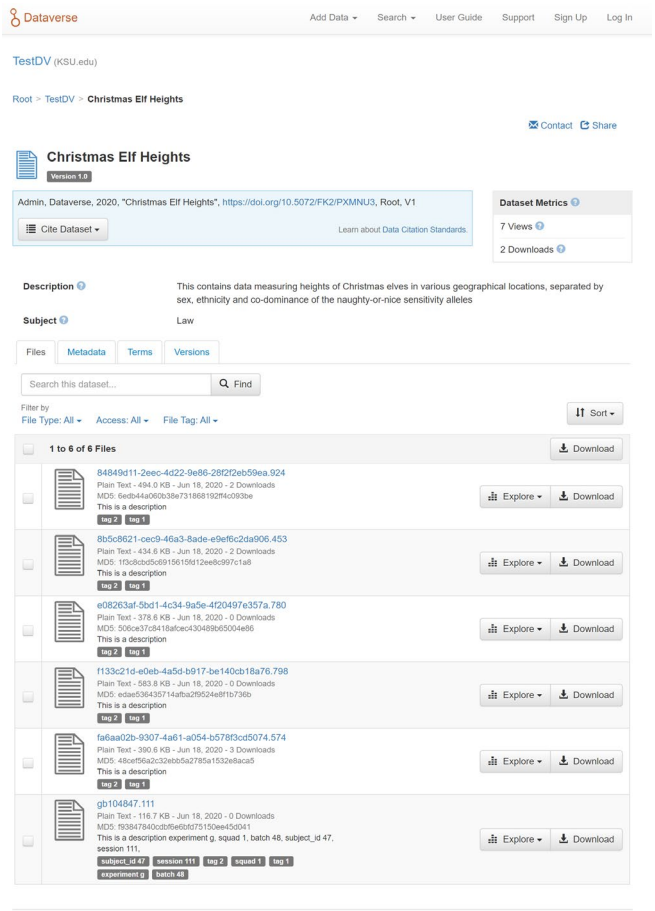


Figure 1. View of data inside a Dataverse data set.

Soon after the proof-of-concept for Synapse was developed, Scholars Portal based in Toronto Canada, showcased a Globus integration with their large Dataverse installation. They have at least 55 Canadian institutions utilizing their research Dataverse installation. They have received a grant in part to solve the large data problem, utilizing Globus's transfer capabilities. Their initial attempt leveraged the Dataverse and Globus UI's to allow the data to be imported. Due to the size of their installation, they opted for a S3 storage implementation. After a major design iteration, they have arrived at a solution similar to Synapse, but more general purpose. The Scholars Portal tool looks to fulfill all the requirements set forth for Synapse with the exceptions of HPC integration, and auto metadata extraction [5].

III. SYNAPSE DESIGN

In this section, we present the Synapse architecture in a high-level view to demonstrate how the major pieces work together as a system. Fig. 2 illustrates a broad overview of the system. Synapse is composed of four major components: Dataverse [7], which is the data storage, discovery and access control system, Globus, the data transfer system [8],

 ISBN: 978-1-61208-887-7

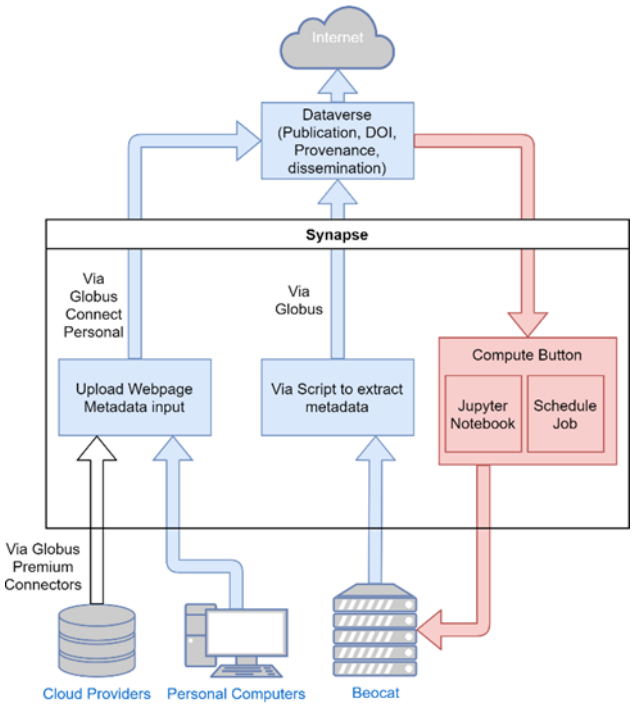


Figure 2. Principal elements of the Synapse system.

Open OnDemand, a graphical user interface for high performance computing clusters [2], and the Synapse web application, which provides metadata extraction and glues these pieces together with a web-based user interface.

We will describe the two main use cases of the Synapse website, importing data into Dataverse with metadata and exporting data out of Dataverse to be used in a HPC cluster. Importing data into Dataverse can be described in a sequence of steps involving 4 systems: a researcher's computer, the Synapse web server (herein referred to simply as 'Synapse'), the Dataverse server, and Globus. Before the flow starts, a user must have a Dataverse account, a Globus account, and a valid Globus endpoint located on the researcher's computer with access to the source files to be transferred. The Globus endpoint could be either the free Globus Connect Personal service to be used on workstations, or a paid Globus Connect server. First, we will go over the of each use case, then explain in more detail interesting steps.

The first use case is importing data into Dataverse. Fig. 3 depicts the process. Step 1 involves the user navigating to the Synapse website. If the user's Globus credentials have expired, a Globus login is required using Open Authorization (OAuth), redirecting the user to the Globus authentication webpage. After a successful login, the user is redirected back to the Synapse page. The user can then select any endpoints they have access to, in this case the endpoint linked to the source files. The user may then drag-and-drop the files onto the panel in the webpage, and Synapse will query the selected Globus endpoint with the limited file information the drag and drop operation gives us to get the absolute path of the files to be uploaded which Globus will then use. Once one or more paths are found, they are displayed to the user for confirmation. The user then selects which dataset in Dataverse to put the files into, what lab they belong to, as well providing any additional metadata to describe what is being uploaded. Once finished, the user will submit the job to be processed.

Step 2 is fully automated by Synapse and performs the following tasks sequentially:

1. *Store a manifest file* detailing the transfer job, source, destination, metadata, etc.
2. *Extract metadata found in the filenames or data files themselves.*
3. *Prepare a custom Globus endpoint* on the Dataverse instance as the destination. With access controls specific to the Globus user logged in.
4. *Tell Globus to initiate a transfer* of the data, capturing the task id so we can query the Globus system for progress updates. At this point, the task is handed off to Globus. Globus will eventually transfer the files to the destination if possible or fail.

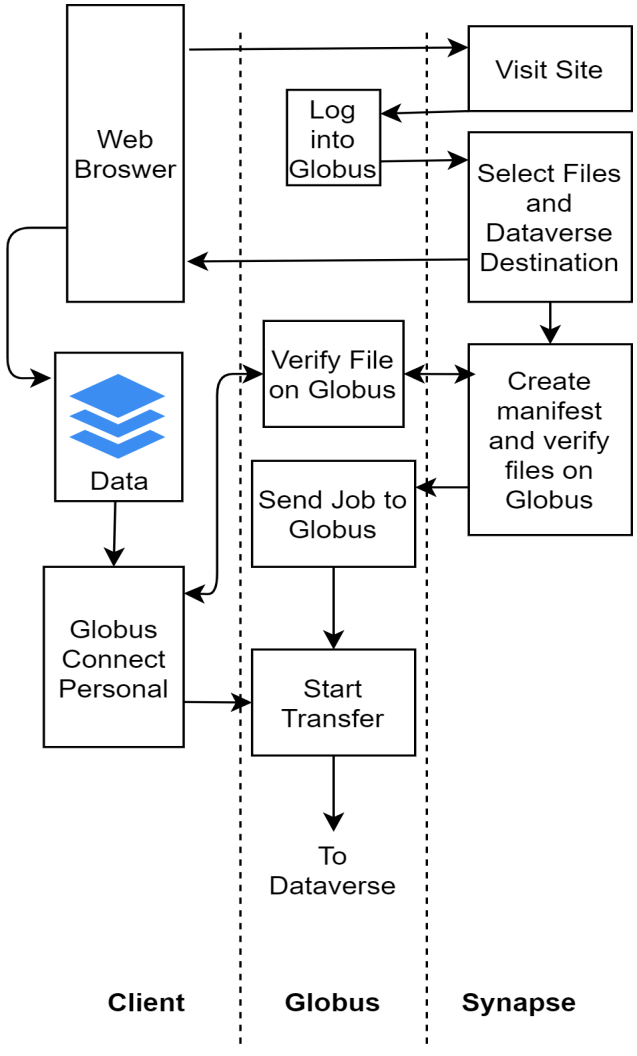


Figure 3. Synapse operation flow when importing into Dataverse.

 ISBN: 978-1-61208-887-7

Step 3 consists of a script running on the Dataverse installation that also has access to the Globus destination endpoint. This script will monitor the destination endpoint for new directories that represent the jobs mentioned in Step 2. When a new directory is found, the script will retrieve the manifest file described in Step 2 and add it to a processing list. Then the script will iterate over the processing list and query Globus to see if the files are still being transferred. Synapse is then updated with the new status, to inform the user. If the Globus transfer has successfully completed the job, then the files are imported into the specified Dataverse dataset using the Dataverse Application Programming Interface (API), given the user's Dataverse API key, which details that user's access permissions. During the importing of files into Dataverse, the Synapse web server is called to post updates to the user. Once the importing finishes, the user will be notified. Fig. 5 displays the main Synapse page that provides the UI for the import process. The "Globus Source" field indicates where Synapse should pull the data from, the "Your Lab" field handles the automatic metadata extraction. The "Upload to Dataset" lets the user select a Dataverse dataset from a list of datasets the user has access to. The "Description" and "Keyword" fields allow extra metadata to be set for all files uploaded.

One point of interest is the fact that we rely on the operating system hosting both the Dataverse installation and the Globus endpoint to temporarily store the file, and then import it into Dataverse. Since the Dataverse importing process can take a considerable amount of time, docking the file on the same portable operating system interface (POSIX) file system allows for some optimizations, such as the ability to import a dummy file with the same metadata as a very large file then update the dummy file pointer to point to the actual large file. Negating both the lengthy importing process of a large file but also negating the necessity of copying the data from one location to another. Using POSIX has a significant disadvantage as well: Dataverse's future seems to be moving toward a simple storage system architecture (commonly referred to as S3). With S3 a different methodology must be used leveraging recent Dataverse updates.

Another subsystem worth discussing is the *automatic metadata extraction*. We have found that labs can have quite consistent internal conventions categorizing data produced from experiments, but data categorization may not be consistent between labs of a department. We decided upon a flexible plug-in approach to extract metadata from the data. A user can write a plug-in for a lab that inherits from a metadata class that extracts the data specific for that lab, or a set of experiments conforming to the format the plug-in can parse. Synapse will auto-detect the plug-in and display it amongst the list of metadata extractors available in the data import page of the Synapse system. All files will be exposed to the selected metadata extractor, which will decide if the file is a match for this extractor. If so, it will extract the data and will associate the extracted data with the file, then store the information in the manifest file described in step 2. After transfer, the script outlined in Step 3 will apply the metadata to a file's tags and description in Dataverse during import. *The tag and description fields of Dataverse are searchable, allowing for quick data discovery.*

Metadata parsed the way mentioned above and imported using the Synapse system auto-applies the metadata specific to that file for each file. Fig. 4 depicts a comparison of the Synapse metadata extraction vs a standard Dataverse upload. The top file is imported into Dataverse using the Dataverse interface, the bottom entry contains auto-populated metadata imported utilizing Synapse. The extra data is file specific; there is a proportional time savings based on the number of files to import into Dataverse utilizing Synapse vs the standard import method; the latter method needing to manually input the metadata per file after import. Both the batch-specific metadata, as well as the file-specific metadata can be searched upon using the Dataverse search tool as well as the data harvesting / sharing engine, if enabled.

Figure 4. Dataverse data with and without metadata.

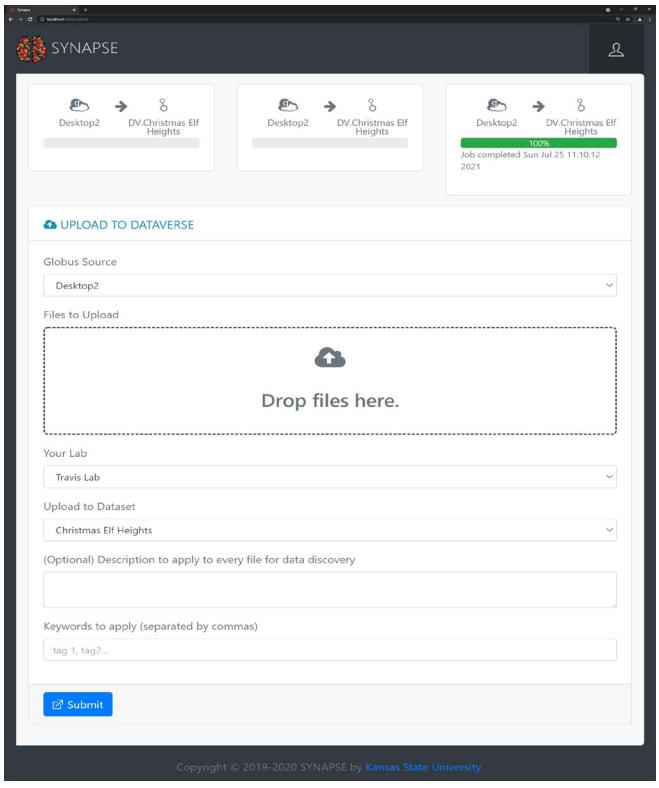


Figure 5. Synapse data import page.

The second use case of Synapse use is to move data from Dataverse into a HPC cluster for processing. This starts at the Dataverse site. If a user has access to the data, they may select the desired file(s) and select “Explore”, then “Send to Beocat”. This makes a call to Synapse. Synapse ensures the user is logged into Globus, grabs the Globus destination ID of the HPC cluster endpoint, and pulls the selected data out of Dataverse via the API, and submits the job to Globus to transfer to the HPC cluster’s endpoint. Note that it will transfer to the logged in user’s Globus HPC cluster endpoint. The Dataverse data plugin system was utilized to enable stored files to be forwarded to Synapse for transfer. In order to transfer files back to Dataverse from the HPC cluster, then one can use Synapse and just select the HPC cluster endpoint for transferring back to Dataverse, for another import.

The final component of the Synapse system is Open OnDemand maintained by the Ohio Supercomputer Center (OSC), which is an open-source software suite that allows visual access to supercomputing resources via a web browser. It has three main features: 1) Scheduling HPC jobs with a graphical interface. 2) use a Virtual Network Computing (VNC) desktop on the HPC cluster, and 3) run specific applications such as Jupyter, and MATLAB directly on the HPC cluster. OSC’s approach is like Synapse’s, using a web browser in a familiar way to interface with disparate technologies to minimize the learning curve. Fig. 6 shows an example of RStudio running on the Beocat HPC cluster at Kansas State University.

Open OnDemand includes a visual file manager, like a researcher’s personal computer file systems to manipulate files. OSC has found that using this file manager is sufficient for managing approximately nine gigabytes of data; for anything greater Globus is recommended [2]. Once a file is moved from Dataverse to the HPC cluster, then Open OnDemand can be utilized for processing.

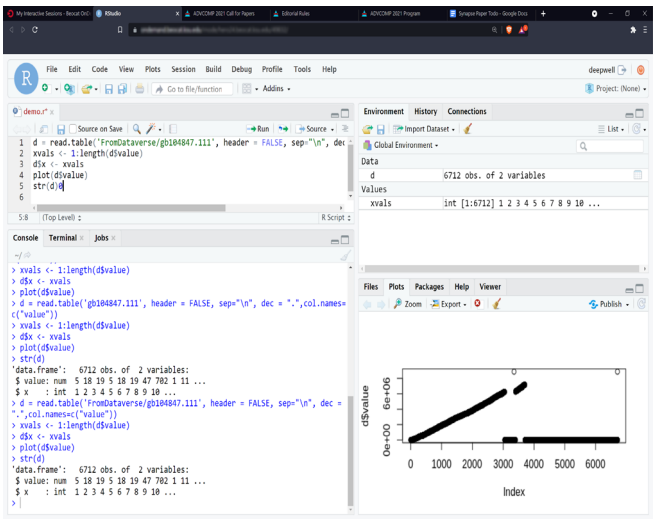


Figure 6. Open OnDemand running RStudio on a HPC cluster.

IV. EVALUATION

In this section we evaluate the time Synapse takes to complete various data importing jobs into Dataverse. We will compare this against the upload feature found in Dataverse itself. There are three machines that we will use for evaluation: 1) The client machine where the data is originally stored. 2) The server hosting the Synapse web server, and 3) The Dataverse installation server. For all tests described below, the client machine (1 in Fig. 7) is a MacBook Pro i9-9980HK CPU running Windows 10 20H2, with 32GB random access memory (RAM), and a 1 terabyte (TB) solid state drive (SSD). The Synapse web server (2 in Fig. 7) is also a MacBook Pro i9-9980HK CPU, 32 GB RAM, 1TB SSD running Ubuntu 20.04.1 under a virtual machine. Finally, the Dataverse server (3 in Fig. 7) is also a virtual machine running CentOS 7.7.1908 quad core with 32GB of Ram running on a local HPC cluster. The Dataverse version we tested with was 4.18.1. This setup is depicted in Fig. 7.

All tests were conducted using the same dataset. This dataset consisted of a series of random generated files, with random data to allow minimal compression which Hypertext Transfer Protocol (HTTP) and Globus might employ for transit. There are 5 datasets used:

- 1 Kilobyte (KB) files
- 10 KB files
- 100 KB files
- 1 MB Files
- 10 MB Files

In each set, we recorded imports using one, 10, 100, and 1000 files. The smallest test transferred 1KB of data (1 file * 1KB File size), while the largest test imported 10GB of data (1000 10MB files).

In each run, we record 5 data points:

- Total import time
- Time to process metadata
- Time to transfer data via Globus
- Time to import data to Dataverse after the Globus transfer
- The amount of time it takes to import data without metadata via the Dataverse website.

We group the data by the 5 data points, with plots for each file size tested.

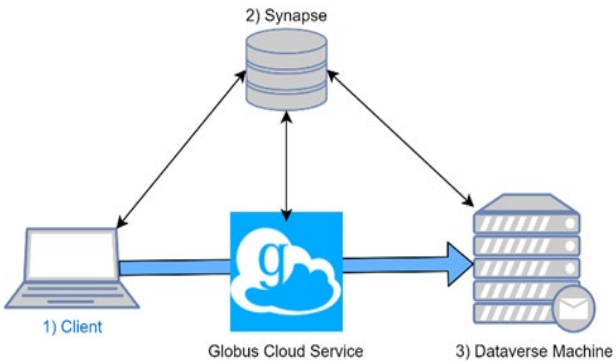


Figure 7. Components of the evaluation system.

First, we look at the overall time it takes to import data into Dataverse utilizing Synapse. As we can see from Fig. 8, it is only until we get to 10MB files do we see much of an increase in processing time from smaller file sizes. This suggests the overhead of the various processes in the Synapse system is taking up the lion's share of the compute time, and it is only when our files sizes get substantial, do we see bandwidth limiting factors come into play. To examine this further, Fig. 9 charts the amount of 'lost time' not accounted for by the major steps the Synapse system takes in importing a file described by our 5 data points. This non-account for time includes user interaction after a user drops files to be processed but before they press the submit button, polling time waiting for the service running on the Dataverse server looking for new jobs, network latency between communication between the various machines, etc. We can see this 'lost time' appears to be influenced by the number of files to be processed but is a much greater percentage of the overall time for small number of files, rather than file sizes, bearing out the reasoning of overhead time. Now, we will break out the individual components of the import.

Fig. 10 describes the time took to process the metadata in milliseconds. Note that the file size had virtually no effect on the metadata processing and was solely dependent upon the number of files to process. This is due to the metadata extracted was all contained in the file name, rather than the file contents. While Synapse's metadata system is flexible enough to handle even file contents, if the extractors use the file names, the system could conceivably handle millions of files.

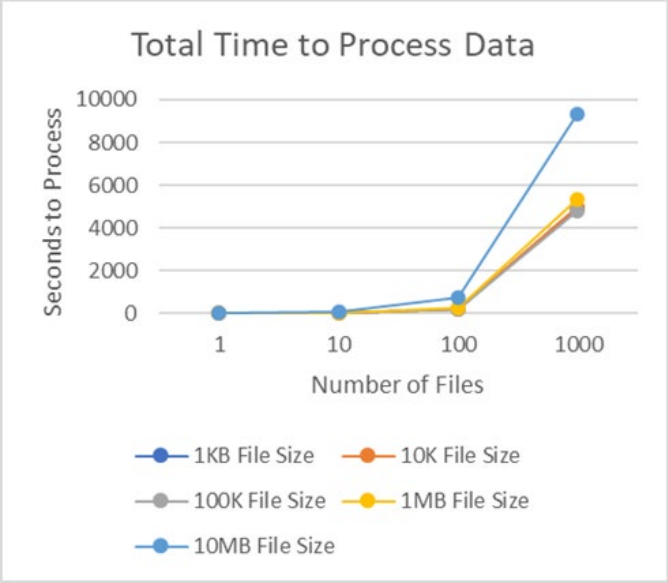


Figure 8. Total length of time to import data into Dataverse via Synapse.

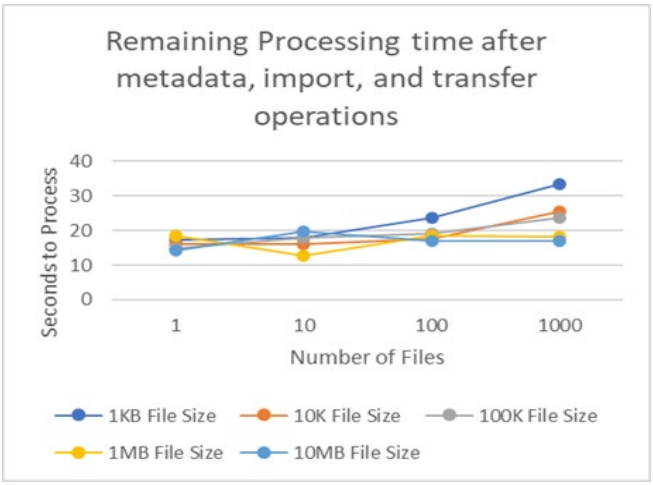


Figure 9. Time not accounted for by data points.

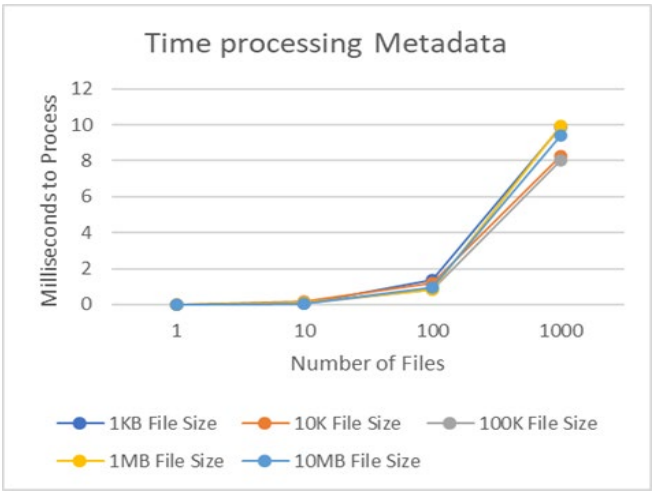


Figure 10. Milliseconds took to process metadata.

Fig. 11 plots the time it takes for the data, once described to Globus, to be transferred to the Dataverse endpoint. Not surprisingly, it is dependent upon total bytes to be transferred. File transfer systems have a per-file overhead, but in Globus this appears to be small.

Fig. 12 shows the time it takes to import the file into Dataverse once it arrives on the Dataverse machine via Globus. As opposed to Globus, it appears the main differentiating factor is the per-file overhead. Even for small file sizes, it takes quite a while to process many files. Interestingly, it is not a linear progression. With a given file size the import time is not proportional to the number of files processed but appears to be exponential. For example, importing 100 1KB files took 163.45 seconds, but 1,000 1KB files took 4885.52 seconds, a 29.9x increase rather than expected 10x. As the graph shows, this holds true regardless of the file size. This phenomenon may be due to design deficiencies in the Dataverse import API process, or due to the Synapse importing residing on the Dataverse machine.

 ISBN: 978-1-61208-887-7

This importing process after the files have already been transferred to the Dataverse machine is where the greatest amount of efficiency gains can be realized and will likely be a focus of any future work. This will be elaborated in Section V.

Dataverse has a webpage where one can drag and drop files to be imported into a dataset much like Synapse, but without the metadata, and large file support. For comparison, Fig. 13 depicts the time it took to import via the Dataverse webpage. Fig. 14 adds an arbitrary 5 seconds per file to emulate metadata input customized per file. As Fig. 14 illustrates, if metadata is not a requirement, and one has a good connection to the Dataverse server, and they are dealing with a relatively small number of smaller files, the Dataverse website upload tool can yield significant time savings in the import process.

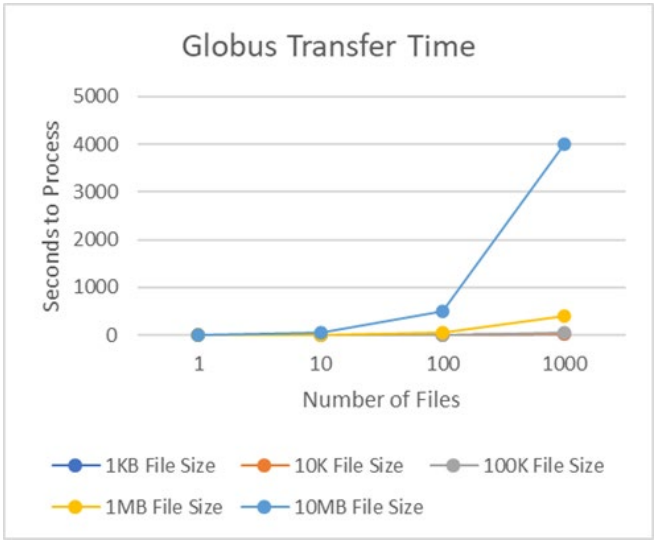


Figure 11. Globus transfer time.

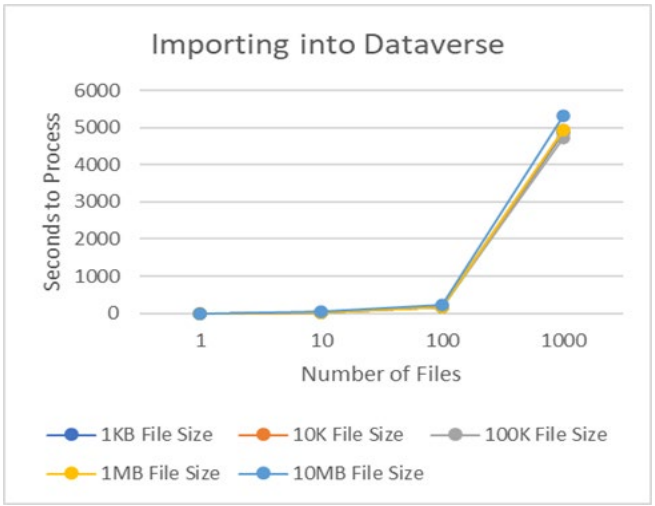


Figure 12. Seconds to import into Dataverse after data is transferred.

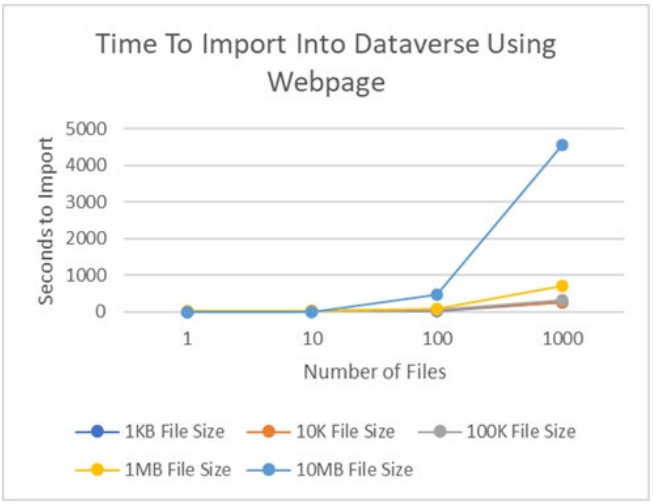


Figure 13. Time to import data using Dataverse website.

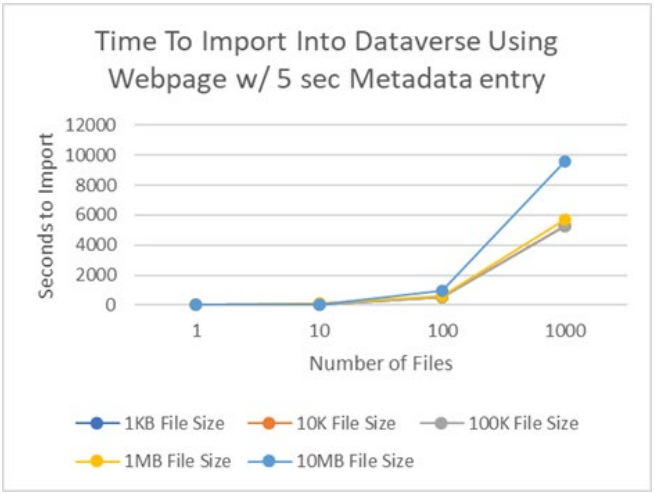


Figure 14. Dataverse website entry with metadata time allowance.

One of the clear advantages of Synapse is the ability to automate repetitive metadata entry over multiple files. Another advantage is the ability to frontload all the effort in saving the data to the repository, so the user can walk away after data entry, since error recovery is built into the data transfer process. Even if the import process takes days or weeks, the user does not need to watch the import or wait for it to finish to then describe the data after import.

V. FUTURE WORK

There are many features that can be implemented to make Synapse a more valuable tool. The most beneficial upgrade would be moving away from a traditional POSIX storage system into a modern S3 compatible storage system. S3 adoption among the Dataverse installations around the world is growing and will likely be the de-facto method of data storage within Dataverse. The Institute for Quantitative Social Science (IQSS), which develops and maintains the

 ISBN: 978-1-61208-887-7

Dataverse project recognizes this fact and is in talks with key institutions such as Scholars' Portal, to facilitate both a more robust S3 storage implementation, as well as tighter Globus integration. We should be able to leverage such advances to improve the performance of the Synapse system.

There can also be further optimization of importing speeds with newer versions of Dataverse, which allows for optimization of the import process. Currently Dataverse inspects and processes each file to - among other things - determine the file contents and produce a preview of the data. This is computationally expensive, and for large files, prohibits importing at all. Third party workarounds have obviated this pre-processing. The IQSS team has recognized this need, and is planning on facilitating this trend, possibly leveraging parallel importing.

Dataverse also supports a rich method of metadata harvesting and sharing utilizing the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) which provides interoperability between data repositories to increase discoverability of research. Additionally, we can vastly decrease the amount of time to import very large files by creating a small dummy file representing the actual file and import that data in with the metadata associated with the dummy file. After importing, we can find the POSIX storage path in the Dataverse database, and then do a POSIX 'move' command, moving the original file that landed on the Dataverse server via Globus, overwriting the destination of the dummy file that Dataverse imported. Finally, we need to update the database for the new file size. This method would prevent copying the data, which is input/output (IO) intensive for very large files and would also minimize the amount of processing Dataverse does to files being imported.

The version of Dataverse we tested provided limited API support for moving arbitrary files out of Dataverse. Either a single file could be moved, or the entire dataset. In the future, with collaboration with Dataverse, we could provide additional features to pull arbitrary files out of Dataverse. For example, the results of a file search could be selected and sent via Synapse to a HPC endpoint.

The Open OnDemand portion residing on KSU's local HPC cluster has been used in a production environment, with positive feedback. Synapse as a whole needs to be evaluated by research groups. Since many of the main goals have been achieved, moving the project into an Agile methodology to close the feedback loop now would be appropriate [10].

VI. CONCLUSION

We have presented Synapse, a web-based tool to enable researchers to store, manage, process and publish data and results using familiar technologies. We have evaluated Synapse with various datasets and shown its usefulness in Metadata extraction, and resilience to network faults. If metadata description is required on a per-file basis, Synapse can greatly decrease the import time into a Dataverse repository.

The main limitation observed is the relative slowness of the import process, compared to the native Dataverse method. The second sizable limitation is our current approach utilizing an antiquated POSIX file system. Despite these issues, Synapse allows for researchers to manage automated metadata extraction, data storage, transfer, computing and publication all wrapped in a familiar web browser experience so researchers can focus on the research.

ACKNOWLEDGEMENTS

This work was supported by the Cognitive and Neurobiological Approaches to Plasticity (CNAP) Center of Biomedical Research Excellence (COBRE) of the National Institutes of Health under grant number P20GM113109. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

REFERENCES

[1] GitHub Development Platform Size Limitations. Available from: https://docs.github.com/en/github/managing-large-files/working-with-large-files/what-is-my-disk-quota#file-and-repository-size-limitations 2021.8.30

[2] D. Hudak, D. Johnson, A. Chalker, J. Nicklas, E. Franz, T. Dockendorf, and B. McMichael, "Open OnDemand: A web-based client portal for HPC centers." *Journal of Open Source Software*, 3(25), 622, 2018. https://doi.org/10.21105/joss.00622

[3] Institute for Quantitative Social Science (IQSS), available from: https://www.iq.harvard.edu/ 2021.8.31

[4] I. Foster, "Globus Online: Accelerating and Democratizing Science through Cloud-Based Services," *Internet Computing*, IEEE , vol. 15, no. 3, pp. 70,73, May-June 2011

[5] Scholars Portal Dataverse, available from: https://learn.scholarsportal.info/all-guides/dataverse/ 2021.8.30

[6] Center for Open Science's Open Science Foundation, available from: https://osf.io, 2021.8.30

[7] Harvard's Dataverse, available from: https://dataverse.org 2021.8.30

[8] B. Allen, J. Bresnahan, L. Childers, I. Foster, G. Kandaswamy, R. Kettimuthu, J. Kordas, M. Link, S. Martin, K. Pickett, S. Tuecke, "Software as a service for data scientists," *Communications of the ACM*, vol. 55, no. 2, pp. 81,88, February 2012 doi:10.1145/2076450.2076468

[9] Synapse open-source code, available from: https://github.com/cnap-cobre/synapse-globus 2021.8.31

[10] A. Srivastava, S. Bhardwaj and S. Saraswat, "SCRUM model for agile methodology," *Proceedings of the 2017 International Conference on Computing, Communication and Automation (ICCCA)*, pp. 864-869, 2017. doi: 10.1109/CCAA.2017.8229928

 ISBN: 978-1-61208-887-7

Pattern Dependent Optimized Mowing of Football Fields with an Autonomous Robot

Tahir Majeed
Department of Informatics
Lucerne University of Applied Sciences and Arts
Lucerne, Switzerland
email: tahir.majeed@hslu.ch

Ramón Christen
Department of Informatics
Lucerne University of Applied Sciences and Arts
Lucerne, Switzerland
email: ramon.christen@hslu.ch

Michael Handschuh
Department of Informatics
Lucerne University of Applied Sciences and Arts
Lucerne, Switzerland
email: michael.handschuh@hslu.ch

René Meier
Department of Informatics
Lucerne University of Applied Sciences and Arts
Lucerne, Switzerland
email: rene.meier@hslu.ch

***Abstract*—This paper addresses the football field mowing problem using an autonomous mowing robot. To reduce the cost of maintaining and preparing the football field for a professional match, the football field is mowed using an autonomous robot. Football fields are typically mowed according to a pattern; therefore, the mowing path of the robot must be optimized with respect to time while obeying the constraint that mowing should result in a predefined pattern. In addition to finding the optimized path for the autonomous robot, a planning software has also been developed that creates and provides the data required by the optimizer. This data contains information on the pattern, the dependencies between individual lanes and the time required to mow a lane and to transition between lanes. The mathematical model to find the optimal mowing path was developed using Integer Programming. The proposed model is computationally efficient, mows the required pattern, fulfills the dependency constraints between the lanes and optimizes the path of the robot so that the football field is mowed in the least possible time.**

***Index Terms*—mowing robot, path optimization, discrete optimization, integer programming.**

I. Introduction

Football is a famous sport which is played and liked by billions over the world. Many people like to watch a match in a stadium while others prefer watching it from the comfort of their home and in front of their television screens. Before professional matches can take place, a lot of planning and resources are required to properly prepare the football field. Fédération Internationale de Football Association (FIFA) defines a set of standards [1] that every ground must meet. Among other requirements, FIFA standards specify: the maximum length of the grass on the football field; an area of at least $3\ m$ around the playing field must be obstacle free for the safety of the players; the minimum and the maximum allowed width of the playing field; rules relating to the luminosity of the football field; and that every part of the field should be lit. All these rules can be found in the FIFA manual [1].

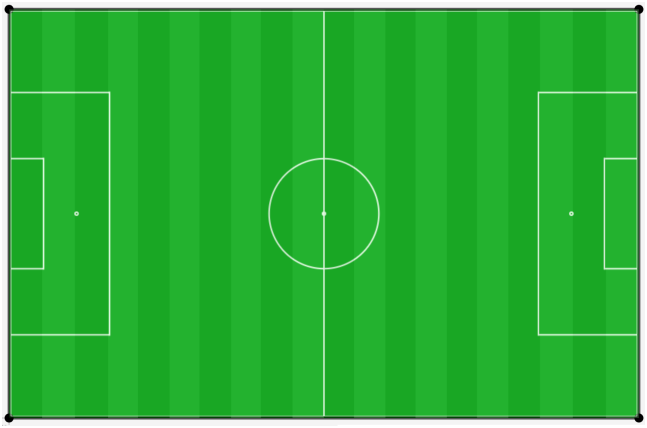


Figure 1: Football field with mowed pattern.

A pattern is mowed in the football field [2] as shown in Figure 1 to make the field aesthetically pleasing [3] for the audience sitting in the stadium or watching on television screens. Mowing the pattern is not a requirement of the FIFA standards, however, the standards do define the maximum height of the grass on the field. It is the responsibility of the ground manager to prepare the football field before the start of a match. On the orders of the ground manager, the grass is mowed and during the mowing simultaneously a pattern is mowed in the field. This practice is traditionally followed in all stadiums around the world. Mowing the pattern does not require any special or additional equipment. The pattern can be easily mowed by folding the grass one way or another, which in turn depends on the direction of the mowing vehicle (either top to bottom or bottom to top).

To comply with the FIFA defined regulations and guidelines requires the employment of a large human workforce. This increases the cost of maintaining the ground significantly. A human driven combustion engine grass mowing vehicle

 ISBN: 978-1-61208-887-7

weighs [4] around 530 kg to 835 kg. This weight can destroy the grass; therefore, it is desirable that the weight of the vehicle be reduced. When the mowing vehicle is driven by a human, the additional weight of the human makes this even worse. This can be remedied by utilizing an automatic mowing robot. This results in avoiding the additional weight of the human driver and the combustion engine.

An autonomous mowing robot has been proposed to reduce the cost of maintaining the football field according to FIFA rules and regulations while simultaneously reducing the weight of the vehicle. The idea is to upload the pre-optimized path in the vehicle's on-board memory, which allows the autonomous mower to mow the field in the minimum time. This paper presents a novel approach for calculating the optimal mowing path for the autonomous mowing vehicle using integer programming. The contribution of this paper is the creation of an integer programming model for searching the optimal mowing path with respect to mowing time and additionally fulfilling dependency constraints.

The mowing problem is casted as discrete optimization problem, where the football field is to be mowed according to the given pattern in the minimum amount of time. Optimizing the mowing path of the football field is similar to a "Traveling Salesman Problem (TSP)" [5] and a "Lawn mowing & milling" problem [6]. TSP has been one of the classical and extensively studied problems in discrete optimization. In a TSP problem, a number of cities have to be visited by a salesman starting from a specific city and returning to the starting city covering minimum distance [7]. Many different approaches have been proposed for solving TSP which includes meta-heuristic approaches, for example, genetic algorithms [8], evolutionary algorithms [9][10], tabu search [11], simulated annealing [12][13], ant colony optimization [14] and integer programming [15][16]. Bektas [5] has given an overview of different integer programming formulations for TSP.

In a lawn mowing & milling problem, a given region must be mowed using a specified mowing blade shape. The region must be mowed with respect to some defined minimization criteria [6]. Generally, the turn cost is minimized [17] which implies that the region should be mowed in minimum number of 90° turns. This kind of problem arises in various domains of our daily life, such as spray painting, geographic surveys, engravings, and drones sweeping regions [6][18]. The lawn mowing and milling problems is in general NP-hard [19]. Arkin *et al.* [6] has proposed a solution for the lawn mowing problem by dividing the mowing region into a set of connected polygons and then using geometric TSP approach to find the feasible solution. Arkin *et al.* [17] proposes to use the geometric TSP approach together with minimizing the turn cost to find feasible solution. Fekete *et al.* [18] extends upon the work of Arkin *et al.* [6][17] to computes a minimum-turn cycle cover for a given region using integer programming based approximate solution.

The problem addressed in this paper is based on earlier work of the authors on a different problem [16]. The domain of the problems addressed in this paper is completely different from the one addressed in [16]. A major contribution of this paper is in formulating a lawn mowing milling problem into a dependency based workflow problem. The requirement to mow a pattern allowed to use the dependency based integer programming formulation of [16]. Furthermore, the natural constraints between the lanes had been transformed into a dependency based constraints. This allowed to use the structure and formulation of the dependency based workflows. The basic structure of the problem addressed in this paper is TSP with dependencies that should be obeyed, however, there are no multiple TSP as solved by Majeed *et al.* [16]. The problem addressed in this paper is also related to the general mowing problem with additional constraints and dependencies. Instead of minimizing the turn cost as normally used in the mowing problem, the objective function minimized in this paper is the total mowing time as given by 1. A similar strategy to the one proposed in this paper has been suggested by Arkin *et al.* [6] but the rectangular lane structure arises naturally from within the problem itself due to the pattern that should be mowed in the field.

The requirement to optimize the mowing time generates from the autonomous robot manufacturing industry. Furthermore, from the mathematical perspective, it is also important that the mowing time is optimized. If the model optimizes the number of turns then there is no constraint that restricts the mower taking a lane 10- or 20-lanes away instead of taking a lane which is 2-lanes away. The turn cost will be the same for both if either a faraway lane or a lane closer was chosen. The mowing time will be vastly different for the both aforementioned cases. As there are equal number of top- and bottom-entering lanes or at maximum can differ by one lane, minimizing the turn cost is not relevant. As the autonomous mowing robot will generally be battery powered, therefore, it is also relevant that the mowing time is optimized.

The paper is organized as follows: Section II provides details of the mowing planning software that generates and provides the input data for the optimization model. Section III provides the details on the mathematical formulation and the optimization model. The settings and the results of our evaluation are presented in Section IV and, finally, Section V provides conclusion and the possibilities on extending the model to solve a larger class of optimization problems.

II. Football Field Planning Software

Optimizing a mowing track requires the knowledge of the relevant environment. Captured in a manner accessible for optimization algorithms, a model representing the area containing information about the surface of the ground, the dimension of the football field, as well as additional navigation areas or obstacles is required. The planning software models the football field and identifies the location and size of the obstacles that must be avoided by the autonomous mowing robot. It further captures information on the mowing lanes, the pattern to be mowed in the football field, the time required to mow a lane and the time required to transition between the lanes.

 ISBN: 978-1-61208-887-7

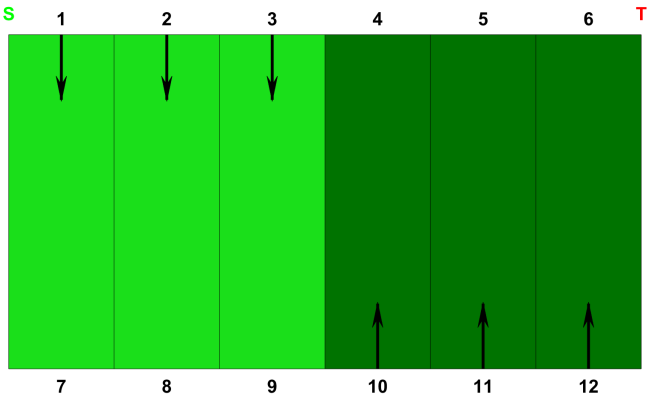


Figure 2: The figure shows a football field pattern with an equal number of top and bottom-entering lanes.

With regard to the calculation of a mowing track containing coordinates for the navigation in a second order space, the model is mapped in a two dimensional vector space containing multiple polygons assigned to one of the distinguished navigation and mowing areas, respectively: pitch (mowing area with pattern restriction), as can be seen in Figure 1 created using the developed software. As input for the model serve the real measured distances and properties of surfaces and objects.

The variation of patterns for the football field ranges from simple lanes in vertical or horizontal orientation to a mixture of vertical and horizontal lanes to complex graphics and logos. However, since our mowers are equipped with spindle blades, our work focuses on patterns based on combinations of straight lanes because of the mower's limited radial mowing ability. The football field will be sub-divided into mowing lanes whose width must not exceed the width of the mower cutting blade. The mowing lanes serves as the smallest unit that must be mowed and the order of the lanes mowed is to be determined by the optimizer. The football field setting and pattern to be mowed are captured by the developed software. The software then computes the mowing lanes. The width of the mowing spindle is $0.85\ m$, to ensure that the grass at the edge of the mowing spindle is properly cut, the mowing lanes are overlapping with a width of $10\ cm$. Using this information, the software divides the football field into a series of mowing lanes and saves the information on which lanes can be entered from the top and which should be entered from the bottom to get the desired pattern. The direction of the mowing is important, that is, either top-to-bottom or bottom-to-top because this produces the light and the dark shades in the grass. The direction of the mowing makes the grass fall one way or the other, which results in grass exhibiting the light or the dark shade.

It should be observed that the pattern mowed in the field is an opposite reflection (not mirroring) across the center line of the football field, therefore, the total number of top-entering lanes is equal to the total number of bottom-entering lanes. This is an important observation which is used in the construction of the optimization model. The optimization model works for an equal number of top- and bottom-entering lanes. The optimization model can also optimize a model with a maximum difference of one lane between the top- and bottom-entering lanes with a minor restriction that can be handled in the planning software as will be explained below.

The modeled environment and the applied pattern in the software results in two matrices called transition matrix $\mathbf{E}$ and lane dependency matrix $\mathbf{D}$ containing the input information for the optimization. The details of the two matrices are illustrated using the basic example shown in Figure 2, where vertical mowing lanes are considered. The mowing lanes are the smallest unit that can be mowed by a mower and the task of the optimizer is to decide the order in which the mowing lanes are to be mowed. The direction of the arrows defines the mowing direction. The mower is only allowed to enter a lane in the direction of the arrow.

For the purpose of the mathematical modeling each mowing lane is defined by two end points, Top-Point (TP) and Bottom-Point (BP) so named as they appear in the figure because of vertical mowing lanes. To get the desired pattern, some mowing lanes can only be entered from the top (shown in Figure 2 by down pointing arrows in lanes 1, 2 and 3) while other mowing lanes can only be entered from the bottom (shown in Figure 2 by up pointing arrows in lanes 10, 11 and 12). The pattern is correctly mowed in the field if the mower moves from point 1 to point 7 with blade down. This will fold the grass in the correct direction while the opposite pattern can be correctly mowed in lane if the mower moves from point 10 to point 4. The same applies to the remaining mowing lanes. There are two additional special points called Start-Point S and Terminal-Point T. S is denoted by numerical index 0 while T is denoted by numerical index given by the equation $T = 2N + 1$. In the example under consideration $N = 6$, therefore, $T = 13$ as can be seen in Figure 2. Generally, the notations S and T are used instead of their numerical index in this paper. The TP of each lane are numbered from 1 to N, while the BP of each lane are numbered from $N + 1$ to $2N$. For the example under consideration the TP are numbered from 1 to 6 while the BP are numbered from 7 to 12 as shown in Figure 2. Let n be the total number of points in the model where $n = 2N + 2$.

A transition cost matrix $\mathbf{E}$, $\mathbf{E} \in \mathbb{R}^{n \times n}$ defines all possible transitions from a top and bottom point of a mowing lane to all other points as shown in Table I. The matrix $\mathbf{E}$ is the cost matrix for the transition of the mower between different lane points. The time required to move from one point to another point is used as the cost. The travel time between two points depend on the velocity, turn angle and acceleration. Matrix $\mathbf{E}$ matrix can be divided into four sections as can be seen in Table I. Each section of the $\mathbf{E}$ defines a different possibility.

In the example shown in Figure 2 and Table I, suppose that the grounds manager chose the first lane to be the starting point of the mowing. This is reflected by $e^{S,1} = 1$ in Table I. This tells the optimizer that the transition from S to lane-1 costs 1 unit while the rest of the entries in row S are all ∞ cost. All other transitions from S to any other lane have an

 ISBN: 978-1-61208-887-7

Table I: MATRIX E, COST MATRIX PROVIDING MOWER TRANSITION TIME (MILLI-SECONDS) BETWEEN LANES.

	S	1	2	3	4	5	6	7	8	9	10	11	12	T
S	∞	1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
1	∞	∞	166	333	499	642	784	1000	∞	∞	∞	∞	∞	∞
2	∞	166	∞	166	333	475	618	∞	1000	∞	∞	∞	∞	∞
3	∞	333	166	∞	166	309	451	∞	∞	1000	∞	∞	∞	∞
4	∞	499	333	166	∞	142	284	∞	∞	∞	500	∞	∞	1
5	∞	642	475	309	142	∞	142	∞	∞	∞	∞	500	∞	1
6	∞	784	618	451	284	142	∞	∞	∞	∞	∞	∞	500	1
7	∞	500	∞	∞	∞	∞	∞	∞	166	333	499	642	784	1
8	∞	∞	500	∞	∞	∞	∞	166	∞	166	333	475	618	1
9	∞	∞	∞	500	∞	∞	∞	333	166	∞	166	309	451	1
10	∞	∞	∞	∞	1000	∞	∞	499	333	166	∞	142	284	∞
11	∞	∞	∞	∞	∞	1000	∞	642	475	309	142	∞	142	∞
12	∞	∞	∞	∞	∞	∞	1000	784	618	451	284	142	∞	∞
T	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

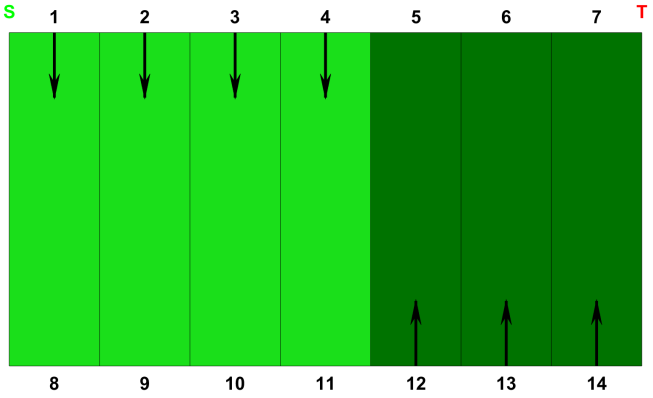


Figure 3: The figure shows an odd number of lanes with an unequal number of top entering lanes and bottom entering lanes. This example shows four top entering lanes and three bottom entering lanes.

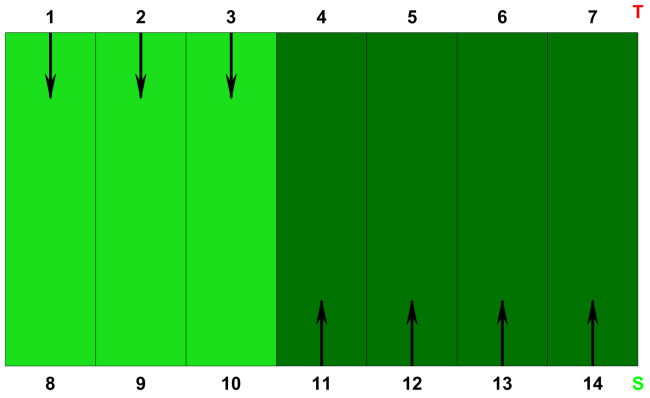


Figure 4: The figure shows an odd number of lanes with an unequal number of top entering lanes and bottom entering lanes. This example shows three top entering lanes and four bottom entering lanes.

infinite cost. The transition cost defined by $e^{i,j}$ is read as the cost of the mower's transition from point-i to point-j. Defining an infinite cost for something has the effect of defining a hard constraint and disallowing such transitions. In the example under consideration the mower is allowed to move only from S to lane-1. This is not a restriction of the model. The model itself allows any of the lanes to be chosen or the optimizer can select any one of the lanes if the infinite cost is replaced by 1 for row S. The cost used in matrix $\mathbf{E}$ reflects the real-world domain knowledge extracted from the domain expert.

From the perspective of the optimization model any of the lanes can be chosen to be the starting lane in cases where the top entering lanes are equal to the bottom entering lanes. However, in cases where the top and bottom entering lanes differ by 1, that is, either there are 4 top entering lanes and 3 bottom entering lanes (see Figure 3) or 3 top entering lanes and 4 bottom entering lanes (see Figure 4), in such cases the model defines the starting point to be on the side with more lanes as shown in the corresponding figures. If there are 4 top entering lanes and 3 bottom entering lanes, the starting point S must be located on top which has one more lane than the other side. Similarly, if there are 4 bottom entering lanes and 3 top entering lanes then the starting point S must be located at the bottom. Our model has no need to be able to handle lane difference of 2 or more lanes and hence, we omit a discussion of such a requirement.

Once the mower has left S, it is forbidden to go back to S, therefore, the column S is filled with ∞ cost. It is also forbidden to go back to the point itself, that is, it makes no sense physically to allow the mower to go from point 1 to point 1 itself, therefore, all the self-points are tagged with ∞ cost on the diagonal. Once the mower reaches T, it is not allowed to move to any other point, therefore, row T is assigned ∞ cost.

If a mower is at point 1, there are two possibilities to transition. The first possibility is that the mower can either transition to another top-point (which could be any of the points 2, 3, 4, 5, and 6) since point 1 itself is a top-point and the second possibility is that it can transition to the bottom-point of its lane which in this case is point 7. The transition time to another top-point can be computed reliably using the mower speed and acceleration equations. The transition time from 1 to 7 can also be reliably computed using the same set of equations. It should be noted that transition time from point

 ISBN: 978-1-61208-887-7

1 to point 7 is mowing the lane in the direction of the pattern. Transitioning from top-point 1 to any other bottom-point other than 7 ($|t \pm N|$) is disallowed by the model with infinite cost in row 1 of the Table I. Transitioning from a top point to a bottom point is depicted in the top-right section of the Table I. The same is repeated for top-points 2, 3, 4, 5 and 6 which fills the top left section of the Table I. As the mower is moving from point 1 to point 7 in the direction of the pattern, therefore, it will be moving with its blade down. When the blade is down the grass is mowed and simultaneously the pattern is mowed in the grass. When the blade is down the speed of the mower reduces by half compared to moving with blade up. This can be seen in the table with the cost $e^{1,7} = 1000$ compared to $e^{4,10} = 500$.

From top-point 4 the mower can transition to top points 1, 2, 3, 5, 6 or it can move in the direction of the bottom-point 10 with mowing blade up. The direction of the mowing for the lane 4-10 is from point 10 to point 4 (as suggested by the arrow from 10 to 4 in Figure 2), therefore, if the mower moves from point 4 to point 10 the mower is moving in the opposite direction of the desired pattern. If the mowing blade is up, a mower can move through a lane without disturbing the underlying pattern. When the mower moves in the direction opposite to the pattern then it does so by having the mowing blade up, which will leave the pattern unchanged on the ground and allows the mower to move at twice the speed.

Transitions between the bottom points 7, 8, 9, 10, 11 and 12 fills up the bottom right section of the Table I while the transition from the bottom point to a top point fills up the bottom left section of the Table I. This is how the whole cost matrix $\mathbf{E}$ is constructed. The mowing terminates when the mower reaches the terminal point T. The mower will only move to the terminal point when all the lanes have been mowed as shown below when the constraints are presented. The terminal point can only be reached from bottom-points 7, 8, 9 and top-points 4, 5, 6. This is shown in the $\mathbf{E}$ matrix by assigning 1 to the specific entries of column T, while all other entries have been assigned ∞ cost. Allowing the mower to move from a specific set of points to terminal points has been derived from domain knowledge.

The dependency matrix $\mathbf{D}$ contains the dependency information between end points of the mowing lanes. A Boolean value serves the information if a start or an end point depends on another, which means that the mower must pass a point before or after the other. As the dependency information is given at the mowing line end point level, it also contains the information about the mowing direction for the desired pattern.

The optimizer returns a sequence of line start and end points defining a track in an abstract manner. This information needs to be added to the autonomous mowing robot so that it can follow the optimized path. For navigation and for a proper execution of the mowing task by the autonomous mower, the optimized track information needs to be enriched with application related information. The abstract track information (in the form of track end points) is transformed into a series of track points by a software developed by the authors. These track points are $0.04\ m$ apart and are enriched by additional information that maps every track point to the two-dimensional mowing area in the field and adds orientation, speed, and blade information to each track point. The distance from an end point of a track segment to the next is between zero and the preset distance. There are some adjustments that need to be made while computing the track points because of the obstacles that are outside the playing field and $3\ m$ obstacle free area around the playing field. The obstacles are initially not considered while computing the optimized mowing path. However, it is not difficult to add the obstacle information to the optimized model. Currently, the time to transition from one lane to another lane is assumed to be obstacle free. This obstacle free transition time is added to the cost matrix $\mathbf{E}$. Replacing the obstacle free transition time with the one that considers the obstacle will enrich the model that considers the obstacles as well.

III. The Integer Programming Model

Our football field mowing problem consists of a set of lanes $\mathbf{J} = \{j^1_{1,7}, j^2_{2,8}, \ldots, j^N_{u,v} u := \{1 \ldots N\}\}; v := u + N$ with N lanes. Each lane has two end-points given by the subscripts $j_{u,v}$. Let $\mathbf{D}$ be a binary 2D matrix of mowing lanes which represents the dependencies between the lanes. If a mowing lane $j_{u,v}$ which is a bottom entering lane must be mowed before $j_{u',v'}$ which is a top entering lane. As $j_{u,v}$ is a bottom entering lane then point v must be mowed before u and similarly if $j_{u',v'}$ is a top entering lane then $u^{'}$ must be mowed before $v^{'}$. The set of dependencies that must be encoded are $u \perp\!\!\!\perp v$, $v^{'} \perp\!\!\!\perp u^{'}$, $v^{'} \perp\!\!\!\perp v$ and $u^{'} \perp\!\!\!\perp u$. The symbol $a \perp\!\!\!\perp b$ is read as *a depends on b*. The encoded dependencies are read as; $u \perp\!\!\!\perp v$ states that u must be mowed before v, $v^{'} \perp\!\!\!\perp u^{'}$ states that $v^{'}$ must be mowed before $u^{'}$, $v^{'} \perp\!\!\!\perp v$ states that $v^{'}$ must be mowed before v and $u^{'} \perp\!\!\!\perp u$ states that $u^{'}$ must be mowed before u.

The dependencies listed in the last paragraph are encoded in the dependency matrix given by Table II and its corresponding Figure 2. Figure 2, shows that lane $j^3_{3,9}$ is a top entering lane while the lane $j^4_{4,10}$ is bottom entering lane. The information of which lanes are top entering lanes and which lanes are bottom entering lane is also encoded in the dependency matrix $\mathbf{D}$. In $\mathbf{D}$, the entry $\mathbf{D}(4, 10) = 1$ which specifies that $4 \perp\!\!\!\perp 10$ which basically states that the 10 must be mowed before 4 thus lane $j^4_{4,10}$ is a bottom entering lane. Similarly, the entry $\mathbf{D}(9, 3) = 1$ which specifies that $9 \perp\!\!\!\perp 3$, therefore, 3 must be mowed before 9 thus lane $j^3_{3,9}$ is a top entering lane. To encode the dependency between the opposite direction lanes $\mathbf{D}(10, 9) = 1$ which means $10 \perp\!\!\!\perp 9$, therefore, 9 must be mowed before 10 and $\mathbf{D}(4, 3) = 1$ which means $4 \perp\!\!\!\perp 3$, that is, 3 must be mowed before 4. All these dependencies ensure that lane $j^3_{3,9}$ is mowed before lane $j^4_{4,10}$. The zero entries in $\mathbf{D}$ specifies independence relationship between the lane points. The optimizer is free to choose in which order these lane points are mowed. The underlying dependency graph is a directed acyclic graph which ensures that there are no circular dependencies. These direct dependencies between the lanes are

 ISBN: 978-1-61208-887-7

Table II: MATRIX D

	S	1	2	3	4	5	6	7	8	9	10	11	12	T
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0	0	0	0	0	0
3	1	0	0	0	0	0	0	0	0	0	0	0	0	0
4	1	0	0	1	0	0	0	0	0	0	1	0	0	0
5	1	0	0	0	0	0	0	0	0	0	0	1	0	0
6	1	0	0	0	0	0	0	0	0	0	0	0	1	0
7	1	1	0	0	0	0	0	0	0	0	0	0	0	0
8	1	0	1	0	0	0	0	0	0	0	0	0	0	0
9	1	0	0	1	0	0	0	0	0	0	0	0	0	0
10	1	0	0	0	0	0	0	0	0	1	0	0	0	0
11	1	0	0	0	0	0	0	0	0	0	0	0	0	0
12	1	0	0	0	0	0	0	0	0	0	0	0	0	0
T	1	1	1	1	1	1	1	1	1	1	1	1	1	0

generally found between the mowing lanes where the pattern changes from light to dark or dark to light. The input data $\mathbf{D}$ & $\mathbf{E}$ matrices for the optimizer are obtained from the planning software as explained in Section II. Let $d_{ij} \in \mathbf{d}$ be the set of binary decision variables $d_{ij} = \{0, 1\}\ \forall i, j = 0, \ldots, n-1$. d_{ij} is 1 if the mower takes the route from i to j and 0 if it does not take this route.

$$\text{Objective: } \min\ \mathbf{Obj} = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} d_{ij} * e_{ij}. \tag{1}$$

The following assumptions are made: the transition time between lanes is known and fix; all the dependencies between the lanes are known before the start of the optimization. To make the integer programming formulation easier, dummy nodes S and T are added which serve as the start and the end of the mowing route (see Figure 2). If a route d_{ij} is written as d_{Sj} then this denotes a route from start point S to any point j and a route d_{iT} denotes a route form any point i to terminal point T. All lanes must be visited, and they should be visited only exactly once.

To create the optimized mowing path, a mathematical model must be constructed with an objective function that defines a cost in terms of some measure (for example, time, money, distance etc.), which is minimized to obtain the optimal solution. The objective function $\mathbf{Obj}$ given by 1 can be defined for minimizing the travel time $d_{ij} * e_{ij}$ which in turn is equivalent to minimizing mowing time of the football field. In the formulation presented, the mower is not allowed to go back in the lane. In addition to optimizing the mowing time, the model also considers the dependencies of the pattern and if there are lanes that must be mowed before other lanes. Figure 1 shows an example of a mowed pattern in a field. The pattern lanes are generally $5\ m$ in width. They are further broken down into mowing lanes by the planning software which results in a pattern similar to the one shown in Figure 5.

$$\sum_{j=1}^{n-1} d_{Sj} = 1 \tag{2}$$

$$\sum_{i=0}^{n-1} d_{iT} = 1 \tag{3}$$

The first constraint given by 2 states that the mower should move from the start point to the first lane to be mowed. This equation shows that the mower is allowed to move from starting-point S to any of the mowing lane points. Corresponding 2, the mower should move to the Terminal-Point T at the end when all the lanes have been mowed as given by 3. The next set of constraints deal with either mowing the lane and then transitioning to another lane or transition from another lane and then mowing the lane. Two sets of equations are presented for each of the top-points (see equations 4 & 5) and then for the bottom-points (see equations 6 & 7).

$$d_{j+N,j} - \left(\sum_{i=1}^{N} d_{j,i}\right) = 0 \qquad \forall j = 1 \ldots N \tag{4}$$

$$\left(\sum_{i=0}^{N} d_{i,j}\right) - d_{j,j+N} = 0 \qquad \forall j = 1 \ldots N \tag{5}$$

$$d_{j-N,j} - \left(\sum_{i=N+1}^{n-1} d_{j,i}\right) = 0 \qquad \forall j = N+1 \ldots 2N \tag{6}$$

$$d_{S,j} + \left(\sum_{i=N+1}^{2N} d_{i,j}\right) - d_{j,j-N} = 0 \qquad \forall j = N+1 \ldots 2N \tag{7}$$

The next set of constraint are sub-tour elimination constraints [5]. The sub-tour constraints given by 8 are the same as suggested by Miller *et al.* [15].

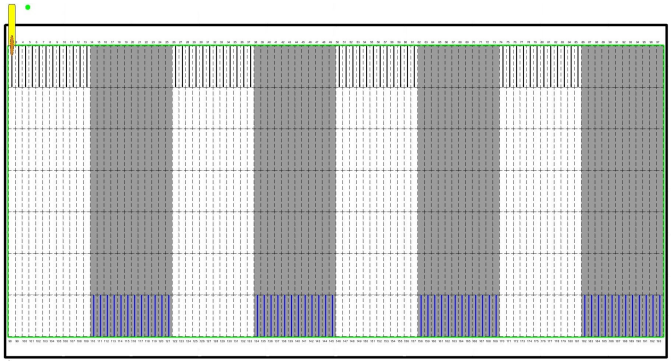
$$u_{i-1} - u_{j-1} + (n-1) \times d_{i,j} < n-2 \tag{8}$$

The dependency constraints given by the dependency matrix $\mathbf{D}$ are added to ensure that the mower mows the left lane before the right lane. This information is encoded in the dependency matrix $\mathbf{D}$ given by Table II which is added in the optimization model in the form of dependency constraints given by 9. Equation 9 ensures that the lanes are mowed in an order that satisfies the dependency matrix D. This constraint is called precedence constraint and is used in job-shop scheduling optimization community [16][20][21]. To decrease the search space and find an optimal solution as early as possible some heuristics were employed. The classical TSP assumes that the salesman can visit any site in any order. However, for the given problem the ∞ cost is assigned

 ISBN: 978-1-61208-887-7

(see Table I) to unreasonable and unrealistic possibilities and possibilities that are disallowed due to model constraints. The corresponding decision variable $d_{i,j}$ to the ∞ cost are set to 0 during variable creation.

$$u_{j-1} - u_{i-1} < 0 \tag{9}$$



(a) 96-mowing lane filed.

(b) Optimized pattern.

Figure 5: Optimization result.

IV. Results

Gurobi 8.0 [22] was used as the optimizer to find the optimal solution for the presented model. Java 8 was used as the programming language and it was tested on an Intel(R) Core(TM) i7-5600U CPU 4-Core 2.60 GHz and 8GB RAM running Windows 10 Enterprise 64-bit. The model was tested on different datasets created using our planning software. The results are presented in Table III. Table III shows the number of lanes (Lanes), solution optimality (Status), objective value (Obj), the gap between the upper and lower bound (Gap), and the run time (Time) of the solver until the optimal solution was found. The solver was able to prove the optimality of the found solution within seconds which shows that the model is computationally efficient. For common cases where the football field is 120 m in length and 60 m in width with a mower blade width of 0.85 m there were 160 lanes, and the model was able to optimize the path of mowing the field within seconds. If, however, the size of the model increases, for example, if the width of the mowing blade is reduced to 0.4 m then the number of lanes doubles and the computational time increases.

Table III: RESULT WITH DIFFERENT NUMBER OF LANES.

Lanes	Status	Obj	Gap(%)	Time(sec)
16	Optimal	41474	0.0	00.13
40	Optimal	231185	0.0	00.49
68	Optimal	539995	0.0	01.22
96	Optimal	2113341	0.0	00.43
146	Optimal	5166207	0.0	15.43
192	Optimal	6761169	0.0	30.56

V. Conclusion and Future Work

An optimization model for optimizing the path of an autonomous football field mowing robot has been presented in this paper. Our model enables such a robot to mow a pattern in a football field to make it aesthetically pleasing for the audience in the stadium and watching on television screens while reducing maintenance cost. The autonomous mowing robot follows the optimized path to create the desired pattern and mow the grass in minimal time.

Mowing the pattern increases the complexity of the problem and makes it different from lawn mowing robots where the purpose is to just mow the grass minimizing some cost function, such as minimizing the number of turns. The input data ($\mathbf{E}$ & $\mathbf{D}$ matrices) for the optimization model is created by our football field planning software. Mowing the field according to a pattern is what makes the problem challenging as the robot can only enter either from the top or bottom for each lane. This makes the transition between the lanes important as the mower is allowed to go over the lanes exactly once. An integer programming model has been presented that optimizes the path of the mowing robot with respect to time and the given pattern. The results of our evaluation show that the proposed model is efficient as it can compute an optimal solution within a few seconds.

The presented model can optimize models where the top entering lanes and bottom entering lanes differ by one. The proposed model assumes that there are no circular dependencies. In future work we aim to relax the restriction on the model and cover situations where the difference between the top and bottom entering lanes is more than one.

Acknowledgment

The authors would like to thank Innosuisse – The Swiss Innovation Agency for funding this work in part.

References

[1] FIFA, “Football Stadiums Technical Recommendations and Requirements,” tech. rep., FIFA, 2011. Available at https://digitalhub.fifa.com/m/6c66fa18f65c0a78/original/rcrtvaelvfae84czze1w-pdf.pdf retrieved: September, 2021.

[2] “Warum hat der Rasen Streifen? (English: Why does the lawn have stripes?).” Online. Available at https://www.dfb.de/news/detail/warum-hat-der-rasen-streifen-70569/ retrieved: September, 2021.

[3] F. Tietjen, “So erhält der Stadionrasen sein Muster. (English: This is how the stadium turf

gets its pattern.).” Online, 2015. Available at https://www.netzathleten.de/lifestyle/sports-inside/item/5737-so-=rhaelt-der-stadionrasen-sein-muster retrieved: September, 2021.

[4] “Commercial ZTrak Zero-Turn Mower.” Online, 2019. Available at https://www.deere.com/assets/publications/index.html?id=917cd17c#6 retrieved: September, 2021.

[5] T. Bektas, “The Multiple Traveling Salesman Problem: An Overview of Formulations and Solution Procedures,” *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[6] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, “Approximation Algorithms for Lawn Mowing and Milling,” *Computational Geometry*, vol. 17, no. 1, pp. 25–50, 2000.

[7] I. Kara and T. Derya, “Formulations for Minimizing Tour Duration of the Traveling Salesman Problem with Time Windows,” in *Procedia Economics and Finance*, vol. 26, pp. 1026–1034, 2015.

[8] S. Akter, N. Nahar, M. ShahadatHossain, and K. Andersson, “A new crossover technique to improve genetic algorithm and its application to TSP,” in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pp. 1–6, February 2019.

[9] Y.-F. Liao, D.-H. Yau, and C.-L. Chen, “Evolutionary algorithm to traveling salesman problems,” *Computers & Mathematics with Applications*, vol. 64, pp. 788–797, September 2012.

[10] X. Wang and G. Xu, “Hybrid differential evolution algorithm for traveling salesman problem,” *Procedia Engineering*, vol. 15, pp. 2716–2720, 2011.

[11] V. Karamcheti and M. Malek, “A TSP engine for performing tabu search,” in *Proceedings of the International Conference on Application Specific Array Processors*, pp. 309–321, IEEE Comput. Soc. Press, 1991.

[12] Y. Liu, S. Xiong, and H. Liu, “Hybrid simulated annealing algorithm based on adaptive cooling schedule for TSP,” in *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation - GEC '09*, pp. 895–898, 2009.

[13] X. Geng, Z. Chen, W. Yang, D. Shi, and K. Zhao, “Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search,” *Applied Soft Computing*, vol. 11, pp. 3680–3689, June 2011.

[14] A. Shetty, A. Shetty, K. S. Puthusseri, and R. Shankaramani, “An Improved Ant Colony optimization Algorithm: Minion Ant(MAnt) and its Application on TSP,” in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1219–1225, November 2018.

[15] C. E. Miller, A. W. Tucker, and R. A. Zemlin, “Integer Programming Formulation of Traveling Salesman Problems,” *Journal of the ACM*, vol. 7, no. 4, pp. 326–329, 1960.

[16] T. Majeed, M. Handschuh, and R. Meier, “Automatic Scheduling of Dependency-Based Workflows,” in *Distributed Computing and Artificial Intelligence, 14th International Conference*, pp. 309–317, Springer International Publishing, June 2017.

[17] E. M. Arkin, M. A. Bender, E. D. Demaine, S. P. Fekete, J. S. B. Mitchell, and S. Sethia, “Optimal Covering Tours with Turn Costs,” *SIAM Journal on Computing*, vol. 35, pp. 531–566, January 2005.

[18] S. P. Fekete and D. Krupke, “Covering Tours and Cycle Covers with Turn Costs: Hardness and Approximation,” in *Lecture Notes in Computer Science*, pp. 224–236, Springer International Publishing, 2019.

[19] E. M. Arkin, S. P. Fekete, and J. S. B. Mitchell, “The Lawnmower Problem,” in *Proceedings of the 5th Canadian Conference on Computational Geometry, Waterloo, Ontario, Canada, August 1993*, pp. 461–466, University of Waterloo, 1993.

[20] J.-S. Chen and J.-S. Yan, “Model Formulations for the Machine Scheduling Problem with Limited Waiting Time Constraints,” *Journal of Information & Optimization Sciences*, vol. 27, no. 1, pp. 225–240, 2006.

[21] D. P. Ronconi and E. G. Birgin, “Mixed-integer programming models for flowshop scheduling problems minimizing the total earliness and tardiness,” in *Just-in-Time Systems*, pp. 91–105, Springer New York, October 2011.

[22] I. Gurobi Optimization, “Gurobi Optimizer Reference Manual,” 2017. Available at https://www.gurobi.com/documentation/9.1/refman/index.html retrieved: September, 2021.

 ISBN: 978-1-61208-887-7