S. Krajči (ed.): ITAT 2018 Proceedings, pp. 146–151
CEUR Workshop Proceedings Vol. 2203, ISSN 1613-0073, © 2018 Štepán Balcar

Preference learning by matrix factorization on island models

Štěpán Balcar

Charles University, Faculty of Mathematics and Physics, Czech Republic, Prague
Stepan.Balcar@mff.cuni.cz

Abstract: This paper presents island models, methods, implementation and experiments connecting stochastic optimization methods and recommendation task (collaborative one). Our models and methods are based on matrix factorization. Parallel run of methods optimizes the RMSE metric from an island model point-of-view.

This paper comments on architecture and some implementation decisions.

We dealt with two research hypotheses. First, whether island models bring always improvement. We will show that almost always yes. Second, whether evolutionary algorithm does or does not always find the best solution. This will be confirmed only on smaller data. Experiments were provided on Movie Lens 100k and 1M data.

1 Introduction

This paper studies recommender systems, especially learning user/customer preferences. Main idea of this paper is to connect this study to evolutionary island models. Here, island models are a computational tool for matrix factorization.

Instances of one stochastic optimization method run in parallel on island models, searching the same state space. Such a method traverses the state space of solutions. The actual state they visit is influenced by (mutually independent) stochasticity. Hence, different instances can visit different parts of the state space. Island model parallelization is based on cooperation of methods during the computation.

This resembles ensemble and/or hybrid learning, where different optimization methods are:

- run in parallel
- allowed to run in different state spaces

The solution (usually one of them) is chosen at the end by an additional aggregation/decision method.

Using evolutionary computing for recommendation is surveyed in the paper of Horvath [1]. This survey shows that usage of evolutionary methods is quite frequent in recommendation systems (see also [2]). One of approaches is model based. Our approach uses matrix factorization and hence the model is constituted by factors. In [1] they mention only one paper [3] where evolution individuals are matrix factors. Authors of the article [3] provide results on ML100k data on minimizing squared error depending on size of population and probability of mutation (other parameters are fixed).

In this paper, parameters of our methods were chosen after experiments on sample data. We will try to improve RMSE using parallelization. We will provide results on both ML100k and ML1M data.

Main contribution of this paper is:

- Multiple island model brings statistically significant improvement of recommendation in comparison to single instance optimization.
- Our implementation is able to handle individuals (matrix factors). Size of our individuals is several orders bigger than usual in evolutionary computation.

This paper is organized as follows: Chapter 2 "Related work" concerns recommender systems and matrix factorization; evolutionary algorithms and island model. Chapter 3 "Methods, models and parameters" sketches our view of stochastic optimization methods; parameters and settings of island model used in tests. Next section "Data" describes realization of experiments and design of tests and computing resources. Finally section 4 "Results" gives both numeric and graphic presentation of our results and discusses confirmation of our hypotheses. After conclusions paper ends with Future work.

2 Related work

This section gives basic notation for recommender systems, evolutionary computation used and overviews some relevant literature.

2.1 Recommender systems and matrix factorization

It was the Netflix price (announced in October 2006) which excited public and changed the landscape of recommender systems area research.

The BellKor squad included Chris Volinsky and his AT&T colleagues Robert Bell and Yehuda Koren, along with four other engineers from the United States, Austria, Canada and Israel (BellKor's Pragmatic Chaos) on June 26, 2009, finally crossed the 10% finish line. Main ingredient of winning solution has been matrix factorization (MF), see e.g. [4].

We focus on latent model based recommendation - which is a part of collaborative filtering technique (Co),

asked for in Netflix Price competition. Co was introduced by [4].

Let us denote $\mathscr{U}$ set of user ids and $\mathscr{I}$ the set of item ids. Input data can be viewed as partial function $r : \mathscr{U} \times \mathscr{I} \longrightarrow \mathscr{P}$, here $\mathscr{P}$ is the preference scale (e.g. one to five stars or real numbers). Alternative representation is in the form of algebraic matrix $\mathbb{R} \in \mathscr{P}^{|\mathscr{U}|\times|\mathscr{I}|}$ with dimensions representing users and items respectively. Rating of user $i \in \mathscr{U}$ of an item $j \in \mathscr{I}$ is content of corresponding cell of matrix $r_{ij} \in \mathscr{P}$. Usually this matrix is very sparse, so in practice input data are represented in a form of a database table R with schema $R(uid = user\ id, iid = item\ id, rating)$. Although implementation uses database table formal model of [4] description is easier in the language of matrices. The matrix $\mathbb{R}$ is factorized to lower dimensional representation

$$\mathbb{R} \approx \mathbb{U} \times \mathbb{I}^{\intercal} = \hat{\mathbb{R}} \tag{1}$$

where $\mathbb{U} \in \mathscr{P}^{|\mathscr{U}|\times d}$, $\mathbb{I} \in \mathscr{P}^{|\mathscr{I}|\times d}$ are user and item latent factors matrices and $\times$ is a matrix product. d is much smaller than $|\mathscr{U}|$ and $|\mathscr{I}|$. Approximation of $\mathbb{R}$ by $\hat{\mathbb{R}}$ is measured by

$$e_{ij}^2 = (r_{ij} - \hat{r_{ij}})^2 = \left(r_{ij} - \sum_{k=1}^{d} u_{ik} i_{jk} \right) \tag{2}$$

here $\hat{r_{ij}}$ will serve as approximation of value r_{ij}.

Our main optimization method is SGD - stochastic gradient descent method. For other approaches and dimensions of research in recommender systems we reffer to [5].

2.2 Evolutionary algorithms

Evolutionary algorithms became popular after 1970, thanks to John Holland [6] as a means for solving optimization problems. The solution is represented as an individual, the set of individuals forms the population. Evolutionary algorithm is formed by phases: initialization, selection (parent selection for next generation offspring), crossover, mutation and evaluation of the whole population. The stochastic computing of individuals seeks convergence to the global optimum.

The contribution of evolutionary algorithms in the area of recommender systems was reviewed by Horvath de Carvalho in [1]. This review analyzed more than 65 research papers using EC techniques for user recommendations. It analyzed different approaches according to five aspects: (i) recommendation technique used, (ii) datasets employed in the experiments and (iii) evaluation methods used in the experiments, (iv) baselines used to compare with the proposed methods and (v) reproducibility of the research.

Most of nature-inspired algorithms reviewed in [1] find application in solving partial problems such as feature weighting extraction, model based approach, clustering and function learning.

Computing latent factor models by using of evolutionary algorithms has emerged as a possible approach [3]. Available publications do not mention the parallelization of evolutionary algorithms in combination with the computation of latent factors.

Motivated by this, in our paper individuals are represented by concatenation of $\mathbb{U}^{\intercal}$ and $\mathbb{I}^{\intercal}$ (like a worm d-thick and $|\mathscr{U}| + |\mathscr{I}|$ long)

$$\begin{bmatrix} u_{11} & \dots & u_{1|\mathscr{U}|} & i_{11} & \dots & i_{1|\mathscr{I}|} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{d1} & \dots & u_{d|\mathscr{U}|} & i_{d1} & \dots & x_{d|\mathscr{I}|} \end{bmatrix}$$

Figure 1: Individual represented by latent vector of users and latent vector of items

In our experiments we use the RMSE as the fitness function

$$\sqrt{\frac{\sum_{i=1}^{|\mathscr{U}|} \sum_{j=1}^{|\mathscr{I}|} e_{ij}^2}{|\mathscr{U}| * |\mathscr{I}|}} \tag{3}$$

2.3 Island model

Island models are one of approaches how parallelize optimization algorithms. Their characterization is that there is no central master and populations are distributed. Island model consists of parallel algorithms enriched by sending and accepting migrants. Migrants are individuals from other islands population. The hope is that this propagation of genetic material can speed up convergence and escape from local optima trap.

Parameters of island model are frequency of sending-receiving migrants and the way how the migrant is chosen from local population (remember that parameters of methods are fixed and are chosen after experiments on a sample data). Optimal choice of these parameters depends on the type of optimization method and is subject of study. These aspects of island models are studied in [7].

Specific type of island models, heterogeneous, were used in the application of multi-criteria optimization by Martin Pilát and Roman Neruda [8]. They designed a new algorithm of MOGASOLS (Multiobjective Genetic Algorithm with Single Objective Local Search) combining multi-criteria genetic algorithm (MOGA) with a single-criteria genetic algorithm (SOGA). It is proved that parallelization of time-consuming method MOGA achieves worse results than parallel running MOGA and SOGA methods. They were tested in NSGA-II [9] and IBEA [10] algorithms. This was further developed in [11].

The first who come up with heterogeneous island models (HeIM), the way we understand them, was Martin Pilat ([11]). In these models, the islands may carry diverse computational methods, differing not only in parameters but also in the structure of the whole stochastic algorithm [11]. For the choice of optimal methods for the given problem is

responsible the central planner, which replaces unsuccessful method by more successful methods during the whole computation. In publication [11] the original Balcar's tool [12] was used.

In this paper this tool is used with homogeneous islands to find optimal user and item latent factor.

Then different optimization techniques are used to converge with factor to optimal one. For illustration we give formulas for stochastic gradient descent. Recall e_{ij}^2, then next generation values of user and item factors equal to

$$u_{ik}^{'} = u_{ik} + \alpha \frac{\partial}{\partial u_{ik}} e_{ij}^2 \qquad i_{kj}^{'} = i_{kj} + \alpha \frac{\partial}{\partial i_{kj}} e_{ij}^2 \qquad (4)$$

Note, that in our implementation we do not consider normalization factor, this is left for future. Nevertheless, some normalization effect is obtained by migration and synergy of islands. α is not fixed, just bounded from above. More on our system is in [12].

3 Methods, models and parameters

Island models where originally developed for parallelization of evolutionary algorithms. In this paper we will use also other stochastic optimization methods.

3.1 Stochastic optimization methods

Evolutionary algorithms try to balance trade off between exploration and exploitation. This property should help evolutionary algorithm to escape from local extremes and simultaneously converge to optimal solution. For this ability they pay a higher time complexity because of parallel development of bigger population. Because of this fact, on some types of problems, the winners are other stochastic optimization methods. An example is TSP solution by hill climbing with 2-opt ([13]). So, this is the main reason we consider several additional stochastic optimization methods (see Table 1).

All these methods use the stochastic gradient descent. A general link to description of stochastic optimization methods is [14].

Our implementation of stochastic optimization methods was changed in two ways. First, we had to create operators to enable them to work with latent factor based individuals. Second, methods were modified for parallel run in island models with migration. They were enriched with ability to cooperate. They receive individuals and enrich their local population. Please notice, that only evolution and differential evolution have bigger population. Remaining methods have only one individual population. In this case enrichment means replacement. They provide their solution from local population to neighboring islands. Methods manipulate incoming individuals in concordance with basic algorithm idea. E.g. the tabu search method inserts a newcomer to the tabu set.

Individual equals solution. Solution is represented by pair of latent factors of users and items (Figure 1). Multiplying of these latent factor we get a matrix which represents our estimation of ratings. Length of first vector is the number of users and the length of second vector equals number of items. Size of the individual depends on size of input problem. Width of latent vector is an optional parameter.

3.2 Parameters and settings of island model used in tests

Now we describe island model - the environment in which the above methods will be parallelized. It is the environment which ensures dissemination of genetic material. The main tool for this is the migration. This migration does not mean only exchange of best solutions. We would like to spread across the system (between islands) genetic material which has the potential to contribute to further improvement of population quality and to speed-up the convergence. Migration is inspired by processes in nature ([15], [16]). Most of island models exchange migrants in a synchronized way. In our system exchange of migrants is not synchronized - so in fact our islands are more general.

Key parameter of island models , as nature shows ([17]), is the frequency of migration and size of local populations. The bigger the frequency of migration the bigger is the chance that islands will help each other. On other side each hardware and software is limited by data through put.

Matrix factorization needs migration of much bigger individuals than e.g. continuous optimization, where individual is a point in an n-dimensional space. In this paper we had to change architecture of model used in [11]. In [11] input problems were TSP (traveling salesman problem of size cca. 1000 cities), BP (bin packing, one dimensional with 1000 items), CO (continuous optimization, 10-dimensional function) and vertex cover (1000 vertexes). Solution of preference learning by matrix factorization on island models is represented of much bigger individuals, rough estimation of our state space is $> \#\text{TSP}^{20}$. Evolutionary algorithms use incoming individuals in groups and after a while. Hence, it is advantageous to store them in a front. As far as individuals are big and memory is limited we had to limit the size of fronts (see Table 2).

3.3 Data

We used data from the movie recommendation domain in the experiments. The effectiveness of parallelization has been verified on datasets ml-100k[1] and ml-1m[2]. Datasets are formed by movie evaluation (1-5 stars), for trials we use the trinity (user, movie, rating).

The training set consists of four-fifths from the dataset, the rest is the test set. Counting derivatives for SGD lever-

[1] http://grouplens.org/datasets/movielens/100k/

[2] https://grouplens.org/datasets/movielens/1m/

Algorithms	Tool	Parameters
HillClimbing	SGD random rating from each line	numberOfNeighbors=10
RandomSearch	generation of latent vectors	
Evolution	uniform crossover + SGD	popSize=10, mutationRate=0.9, crossRate=0.1, parentSelector=CompareTwoRandomSelectors
BruteForce	SGD according to the I-th rating	
TabuSearch	SGD random rating from each line	tabuModelSize=50
Sim.Annealing	SGD random rating from each line	temperature=10000, coolingRate=0.002
Diff.Evolution	differential crossover	popSize=50, F=0.25

Table 1: Methods and parameters

Parameter	value
Number of iterations	50, period = 60 seconds
Number of islands	4 (AMD Opteron 6376)
Neighbors of method	3 (distributed to everyone)
Migration frequency	5 seconds

Table 2: Parameters of the island model

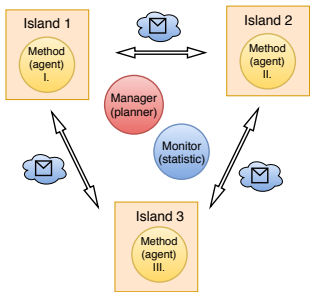

Figure 2: Architecture of system

ages the training set. The test set is used to evaluate individuals.

3.4 Realization of the experiment

Our main idea is to increase stochasticity of searching the state space (which is enormous for movie data). First level of stochasticity is enabled by stochastic optimization method. Second level is enabled by several independent islands. The third level is attained by migration. Our system enables all of these. Moreover, all of these run in parallel with mutually assisting methods (not competitive).

In order to be able to obtain the best solution from such a computation resource, the computation must be continuously monitored (Figure 2). Solutions represent individuals that are unpredictably moving across the island model. We can never know when and where the best solution will appear (sometimes the best solution does not appear at the end, see Figure 3).

Our implementation uses agent middle-ware Jade[3] for achievement of higher adaptivity of methods (in our three levels of stochasticity).

Here we were facing main decision. Whether to prefer higher adaptivity (based e.g. on Jade) or better use of effectiveness of specific hardware (based e.g. on C). From pure experimental curiosity we went along the agent based middle-ware framework direction.

The central brain of the multi-agent based system is a manager of migration which directs the computation and measures genetic material during evolution.

[3] http://jade.tilab.com/

The use of this system is the implementation of methods as an agent, who can develop methods as adaptive computing resources. The infrastructure of the system is made up of two central points, by Agent-manager that manages computation and by Agent-monitor, that monitors genetic material in the system.

Software allows multiple ways of monitoring. The monitor statistically processes the quality of the individuals. Another way of observing what happens in the system is to produce the pedigrees of an individual. The basic idea is to enrich every individual of the pedigree that contains information on the involvement of concrete islands in its creation.

For our experiments, were used statistics that are computed from each monitored individuals in one iteration of planner. We will present the results as a measure of the system, which is monitoring follow-planner-iterations.

Our evolutionary methods (Table 1) use uniform crossover. In phase of mutation they apply stochastic gradient descent on a randomly chosen rating.

Differential evolution combines 3 latent vectors (individuals which are models/solutions) $\mathbb{LV}_1$ which is a concatenation of $\mathbb{U}_1^\intercal$ and $\mathbb{I}_1^\intercal$, similarly $\mathbb{LV}_2$ and $\mathbb{LV}_3$. Result of the differential operator is latent vector

$$\mathbb{LV}_{new} = \mathbb{LV}_1 + F * (\mathbb{LV}_2 - \mathbb{LV}_3)$$

here F is 0.25.

3.5 Test design and computing resources

Two pairs of tests were created, comparing single methods and island models, differing in the size of the input Movieland dataset. The width of the latent vector was set to 10 (best after initial experiments on smaller samples). The tests run on all initial parameter settings (Table 3) 9 times. As far as our computations are nondeterministic this makes difference (see Figures 4 and 5). These are computationally demanding calculations.

Parameter	value
Number of runs	9
Computing nodes	AMD Opteron 6376, 64GB memory
Latent factor width	10
Datasets	ml-100k, ml-1m

Table 3: Parameters of the test

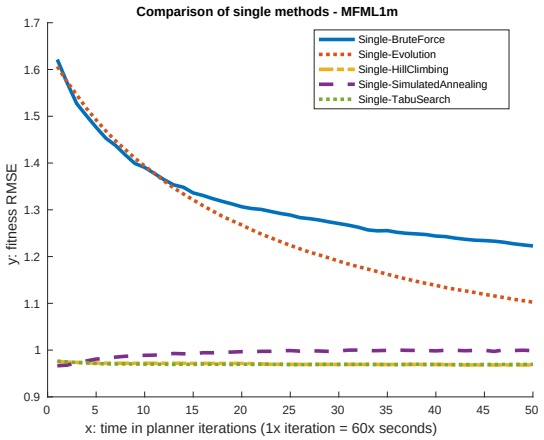

Figure 3: Methods ML1m investigation of median run

4 Results

Our experiments validated two hypotheses. First is that evolution does not necessary give best solution and second that island models improve results. We will present results for two datasets separately. We will show the best solution of 9 runs and average of all runs (for distribution see Figures 4 and 5).

On the smaller data set the winner is always Evolution and Islands give always better solution.

On the bigger dataset the single island winner is simulated annealing (in both minimum and average). In Islands the winner is hill climbing (in both minimum and average). On bigger data we can not always observe advantage brought by parallelization by islands and migration. This can be observed especially by simulated annealing. One possible explanation is that hill climbing does not risk that much going in wrong direction as simulated annealing is doing (sometimes). Bigger data bring bigger state space and hence risk is necessary, but 50 iterations of the planner is probably not enough. In future research we will run island models with more iterations.

5 Conclusions

We proved that island models are a good computing tool for recommender systems. Our experiments have shown following. Island models brought clear improvement on smaller data set. On bigger data, it is also true except of simulated annealing. Moreover on bigger data evolution does not find best solution.

Our implementation is publicly available on Github[4] and hence enables repeatability of our experiments (see [1], requirement (v)).

[4] https://github.com/sbalcar/distributedea

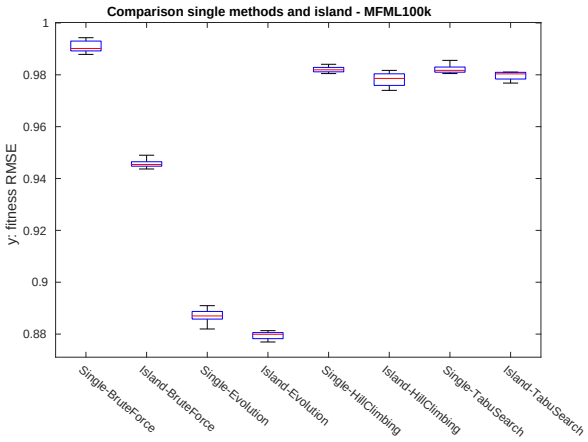

Figure 4: Methods ML100k boxplot of results

6 Future work

In our future work we plan to extend this to heterogeneous island models. We also plan to develop new planners which will be specifically designed for recommendation domain. We would like to consider also islands with statistical learning methods. Challenge is extension to content based recommendation.

References

[1] Tomáš Horváth and André C. P. L. F. de Carvalho. Evolutionary computing in recommender systems: a review of recent research. *Natural Computing*, 16(3):441–462, 2017.

[2] M. Rezaei and R. Boostani. Using genetic algorithm to enhance nonnegative matrix factorization initialization. In *2010 6th Iranian Conference on Machine Vision and Image Processing*, pages 1–5, Oct 2010.

Methods	Single-min	Islands-min	Single-average	Islands-average
BruteForce	0.98787115	0.94367838	0.99088728	0.94571058
DifferentialEvolution	1.50105714	1.49096267	1.51771324	1.49957429
Evolution	**0.88196787**	**0.87695296**	**0.88705747**	**0.87952888**
HillClimbing	0.98047412	0.97400051	0.98207866	0.97824895
RandomSearch	1.60448414	1.58333447	1.61703977	1.60802325
SimulatedAnnealing	1.06110779	1.03108626	1.07797515	1.03806128
TabuSearch	0.98048607	0.97681064	0.98217126	0.97963446

Table 4: Comparison of single methods and island models: Dataset - MFML100k, Runs - 9

Methods	Single-min	Islands-min	Single-average	Islands-average
BruteForce	1.22268511	1.18410586	1.23279212	1.2033589
DifferentialEvolution	1.55227103	1.55057106	1.55780106	1.55422412
Evolution	1.10254503	1.07069968	1.13526207	1.09140823
HillClimbing	0.96832015	**0.96727131**	0.97065502	**0.96840234**
RandomSearch	1.65483811	1.66265835	1.66738744	1.6660426
SimulatedAnnealing	**0.96655732**	0.97118289	**0.97048761**	0.9729405
TabuSearch	0.96873215	0.96735123	0.97106628	0.96854622

Table 5: Comparison of single methods and island models: Dataset - MFML1m, Runs - 9 (in red there are violations of island improvement)

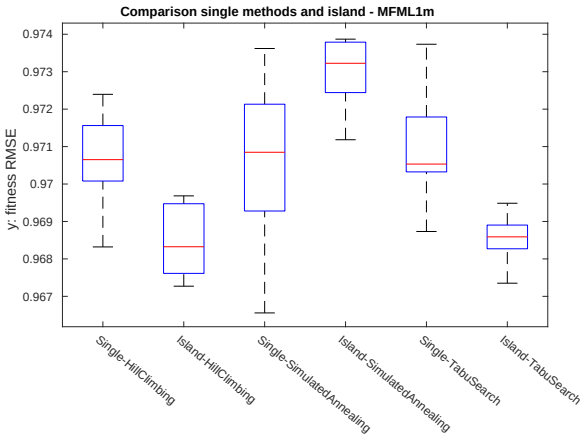

Figure 5: Methods ML1m boxplot of results

[3] Dariush Zandi Navgaran, Parham Moradi, and Fardin Akhlaghian. Evolutionary based matrix factorization method for collaborative filtering systems. *2013 21st ICEE*, pages 1–5, 2013.

[4] Y Koren, R Bell, and Volinsky C. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[5] L. Peska and P. Vojtas. Using implicit preference relations to improve recommender systems. *Journal on Data Semantics*, 6,1:15–30, 2017.

[6] John H. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, 1992.

[7] Darrell Whitley, Soraya Rana, and Robert B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *J.Comp.Inf.Tech.*, 7:33–47, 1998.

[8] Martin Pilát and Roman Neruda. Combining multiobjective and single-objective genetic algorithms in heterogeneous island model. In *CEC 2010*, pages 1–8, 2010.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Trans. Evol. Comp*, 6(2):182–197, 2002.

[10] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm. Technical report, ETH Zurich, 2001.

[11] S Balcar and M Pilat. Heterogeneous island model with replanning of methods. In *GECCO'18, accepted as poster*, 2018.

[12] Stepan Balcar. Heterogeneous island model with replanning of methods (in czech). *Master thesis, Martin Pilát supervisor*, 2017.

[13] G. A. Croes. A method for solving traveling salesman problems. *Operations Research*, page 791–812, 1958.

[14] R.K. Arora. *Optimization: Algorithms and Applications*. CRC Press, 2015.

[15] Jinliang Wang and Michael C. Whitlock. Estimating effective population size and migration rates from genetic samples over space and time. *Genetics*, 163(1):429–446, 2003.

[16] William Astle and David J. Balding. Population structure and cryptic relatedness in genetic association studies. *Statist. Sci.*, 24(4):451–471, 11 2009.

[17] John Wakeley. The coalescent in an island model of population subdivision with variation among demes. *Theoretical Population Biology*, 59(2):133 – 144, 2001.

Acknowledgements. This work was supported by a Ph.D grant SVV2018-260451 under supervision of Peter Vojtas.