CMap: DPeak Challenge – report

Author: Wojciech Szałapski (wojciech.szalapski@gmail.com)
TopCoder username: Ardavel

Table of contents

1. A high-level description of the algorithm

1.1 Overview

The outline of the proposed algorithm is the following. First of all, each histogram of fluorescence intensities is clusterized using the Expectation-Maximization method, assuming the presence of two lognormal distributions. In the next stage, for each gene, a plate-wide distribution is fit with respect to the estimated expressions and cluster sizes. It is then used to adjust the assignment of peaks to the low- and high-proportion genes. In the last stage of the algorithm, a combination of per-well and plate-wide data is used to further refine the result. This includes merging the estimated clusters and finding new ones where they could have been omitted.

1.2 Stage 1 – Clusterization of fluorescence intensity histograms

Please assume that the following steps are separately and independently performed for each (well, barcode) pair.

1. Take a binary logarithm of each intensity value.
2. Discard isolated measurements (please note that they can be reconsidered later in the processing workflow).
3. Fit a pair of normal distributions to the data. Use the Expectation-Maximization (EM) method with the initial centers of clusters seeded in the lowest and the highest values of measurements respectively.
4. Determine the size of both clusters by assigning each sample to the corresponding distribution so that the probability is maximized. Assign the bigger cluster to the gene mixed in higher proportion.
5. If it failed to fit two distributions or one of them is too small, fall back to fitting a single distribution to the data.

1.3 Stage 2 – Refinement of the results using the plate-wide population

Please assume that the following steps are separately and independently performed for each barcode.

1. For each well, consider the distribution(s) which were fit during the first stage of the algorithm to the histogram of fluorescence intensity. Split the wells into 3 groups:
    - Non-overlapping models (two barely overlapping distributions).
    - Overlapping models (two significantly overlapping distributions).
    - Single-cluster models.
2. Using the cluster means of the non-overlapping models, fit a normal distribution separately to both genes associated with the barcode being considered. This estimates a plate-wide population of the expression of each gene.
3. For each non-overlapping model, flip the assignment of the low- and high-proportion genes if it makes them fit better to the plate-wide populations.
4. For each non-overlapping model, find wells where both clusters fit to the plate-wide population of the high-proportion gene and neither fit to the population of the low-proportion gene. Move each such model from the non-overlapping group to a new group containing uncertain model.
5. Using the cluster means and sizes of the non-overlapping models, fit a plate-wide bivariate normal distribution separately to both genes associated with the barcode being considered.

6. Iteratively refine the bivariate normal distributions using the non-overlapping models. Swap the assignments of clusters to low- and high-proportion genes accordingly, using the rule of maximum a posteriori probability.
7. Estimate once again the plate-wide populations of gene expressions using the results of the iterative procedure from the previous step.
8. Remove those models from the non-overlapping group which do not fit well the plate-wide populations of gene expressions.
9. Estimate once again the plate-wide populations of gene expressions using the remaining part of the non-overlapping group.
10. For each well in the group of overlapping models, find wells where both clusters fit to the plate-wide population of the high-proportion gene and neither fit to the population of the low-proportion gene. Move each such model to the group containing uncertain models. For all other overlapping models, flip the assignment of the low and high-proportion gene if it makes them fit better to the plate-wide populations.
11. Add the content of the group containing single-cluster fits into the group of uncertain models.

1.4 Stage 3 – Refinement of the results using a combination of well data and plate-wide data

Please assume that the following steps are separately and independently performed for each barcode. Moreover, this stage does not consider any isolated measurements to be discarded (see the Stage 1 for reference).

1. Consider the group of uncertain models. Exclude the ones which come from those wells where the plate-wide populations of gene expressions are not well separated.
2. For each remaining well with an uncertain model, search for measurements which fit the plate-wide distribution of the low-proportion gene expressions.
3. If enough measurements have been found, check if they are clearly separated from the distributions fit so far for the well and barcode being considered.
4. If the above conditions are met, use the median of the measurements found in the step 2 as an estimation of the low-gene expression. If two clusters have been fit initially, then merge them and use the result as a new estimation of the high-gene expression.
5. If it failed to estimate a new low-gene expression, check if the two clusters should be flipped to fit better the bivariate normal distributions estimated in the Stage 2.

1.5 Discussion of the solution and the development process

The process of designing the algorithm went through a few cycles of analysis, development and refinement. My initial approach was to use the k-means (and then k-means++) method and choose the optimal clusterization based on the theoretical 2:1 ratio of gene proportions. Analysis of the results of this solution convinced me that a more sophisticated algorithm is needed and the 2:1 ratio is in practice a false assumption too often to rely on it.

My next attempt was to use the Expectation-Maximization (EM) algorithm and fit a pair of normal distributions to each histogram of fluorescence intensities. This solution gave me a significant improvement over the k-means approach. During the analysis I found out that it is better not to assume any a priori probability (in particular the 2:1 ratio) of assignment to clusters.

After finishing my first working EM implementation, I added a flip-adjustment procedure similar to what was presented in the description of the challenge. However, I fit two plate-wide bivariate normal distributions to each barcode, instead of doing a linear discriminant analysis. This gave me a solution with the improvement safely over the prize-eligibility threshold. At that point I decided to try to get the maximum out of the histogram clusterization and then come back to the flip-adjustment part.

At first, I played with different ways of choosing the initial centers for the EM algorithm. I found out that discarding isolated measurements gives better scores than starting in multiple places and choosing the best one by maximizing the log-likelihood. Moreover, this was by far a more efficient approach to choose only one setup of the initial centers (the extreme samples) and get rid of the time-consuming calculation of the log-likelihood.

Then, I analyzed the cases where the algorithm was very inaccurate. I found out that multiple of them were histograms with three instead of two expected clusters. I tried to target this problem by fitting both two and three clusters to the data and determining the best choice using the Bayesian information criterion. It turned out that it helped only in a few cases and required testing multiple initial centers what decreased the speed too severely.

Another prominent problem was filtering out small clusters corresponding to the low-proportion genes, incorrectly treating them as isolated measurements. Unfortunately, turning the filtering off resulted in significantly worse results.

At that point of the development, I was not convinced whether I could improve the clusterization itself. Fixing one type of errors resulted in creating other over and over again. I also noticed that similarly looking histograms happen to have a significantly different ground truth. With this conclusion, I came back to the adjustment part to see whether any improvements can be found there.

This change of direction was crucial to the development the final solution. I realized that the plate-wide information could help me determine not only where I swapped the peaks corresponding to the low- and the high-proportion genes. What was more, it could let me decide whether some small clusters or isolated measurements should indicate a peak and where they were just a noise. With this in mind, I could develop the Stage 3 of the algorithm and modify the first two stages accordingly.

If not the time-constraints of the competition, I would have come back to dealing with histograms containing three peaks. Using the plate-wide information, I could have chosen the ones most probably corresponding to the gene expressions being measured. If this had worked, I would have tried to find a robust strategy of finding the initial centers for EM instead of checking multiple configurations (what was slow).

2. A detailed description of the algorithm and the implementation

2.1 Preface

Let me apologize for all parts of the source code which have not been refactored due to the time-constraints of the competition.

2.2 Compilation

The whole solution is implemented as a standalone C++17 application. It is meant to be compiled using GCC on Linux but during the development it was kept compatible also with MinGW and MSVC 2017 compilers. The solution is encapsulated in a CMake project without any dependency on libraries which are not provided by the compiler (in the GCC and MinGW versions).

The following compilation flags were used to speed-up the application:

- `-fopenmp` – enable multithreading via OpenMP
- `-O3` – set the level of compiler's optimizations
- `-ffast-math` – enable additional optimization of math functions

2.3 Execution parameters

The application should be executed with the three following parameters (the same as in the description of the challenge):

- `--dspath` – path to a directory containing the input for a single plate
- `--out` – path to a directory where the results should be saved
- `--plate` – name of the plate to be processed

The above parameters are parsed by the `ParameterParser` class. All other parameters of the algorithm are hardcoded as compile-time constants for the maximum speed of the solution. They can be found in the `Parameters` class.

The correspondence between barcodes and genes is also hardcoded for the same reason. It can be found in the `BarcodeToGeneMap.h` header file.

2.4 Reading the input

In the final solution, the input measurements are deserialized using memory-mapped files. The provided dockerized version of the application, which compiles with GCC on Linux, uses the operating system API to implement the deserialization. To keep compatibility with MinGW and MSVC on Windows, alternative implementations are provided. All three versions of the input-reading code can be found in the following header files (the appropriate one is chosen during compilation):

- `well_reader_gcc_linux.h`
- `well_reader_gcc_windows.h`
- `well_reader_msvc.h`

For the maximum speed, the input files are parsed manually using the code in the `fast_input.h` header. Any fluorescence intensity outside the $[16; 32768]$ interval is assumed to be incorrect and gets discarded. For all other measurements, a binary logarithm is taken and the result is stored. The samples for each barcode are sorted in the non-descending order to speed-up further calculations. From this point onward, by a "sample" or a "measurement" we are going to mean a binary logarithm of the corresponding intensity value read from a file.

2.5 Implementation of the Stage 1

2.5.1 Introduction

The processing workflow is contained in the `Pipeline::run` member function. In the first stage, it performs a clusterization of each fluorescence intensity histogram separately. The application processes all wells in a multithreaded manner, using an implementation of the OpenMP specification provided by the compiler.

2.5.2 Preprocessing

First of all, the well data is deserialized using the previously described method. Then, isolated samples for each barcode are discarded according to the `IsolatedMeasurementsFilter::filter` member function. For each sample s, the interval $[s - 0.625; s + 0.625]$ is considered. Let us denote by n the number of samples for given (well, barcode) pair. The sample s gets discarded if less than $[0.15n]$ samples are present in the aforementioned interval. The remaining samples are stored in a buffer which is going to be reused by the same OpenMP thread for the purpose of optimization. The filtration described above is implemented as a single pass.

From this point onward, until the end of the Stage 1 description, each step is assumed to pertain to a single (well, barcode) pair.

2.5.3 Clusterization with the Gaussian mixture model

If at least 20 samples have survived the preprocessing phase, then a pair of normal distributions is going to be fit to them. This phase is implemented in the `GMM::clusterize` member function which uses the Expectation-Maximization (EM) algorithm.

The initial centers of the distributions are seeded in the first and the last sample (please remember that the samples have been sorted before). Then, all samples are assigned to corresponding clusters using the rule of the minimum distance. If any of the clusters got assigned less than 2 samples this way, the EM procedure is stopped and consider to be unsuccessful.

Otherwise, the samples assigned to each cluster are used to estimate its initial mean, variance (using the unbiased sample variance estimator) and the a priori probability of sample membership. Please note that these a priori probabilities are adjusted during the course of the EM algorithm. In the development phase, it was found out that the method is more robust when the 2:1 ratio of cluster sizes is not assumed in this step.

The algorithm performs 10 iterations of EM. An early exit is implemented if during one iteration the means of distributions move by less than 0.001 in total. Another stop condition is when the variance of any of the clusters goes below 0.0001.

Due to the observed characteristics of the input data, it was chosen to use only one set of the initial centers. This way it was not needed to calculate any log-likelihood what increased the processing speed.

2.5.4 Estimation of the cluster sizes

Once the two distributions have been fit, they are used to determine the size of the clusters. The size is measured by the number of samples assigned to each cluster, as follows. Each sample gets assigned to

the cluster for which it has a higher probability density, provided that it lies within the 3σ range of the distribution corresponding to this cluster (by σ we denote the standard deviation of a distribution). The estimation of the cluster sizes is done by 5 binary searches over the samples instead of a less efficient linear pass (see the lines `110-153` of the `Pipeline.cpp` file).

If the size of the smaller of the clusters is not less than 10% of all samples (after discarding the isolated ones in the preprocessing phase), then the Stage 1 is considered to be finished for given (well, barcode) pair. The bigger of the clusters is assigned to the high-proportion gene.

2.5.5 Falling back to a single-distribution fit

If the size of the smaller of the clusters is below the above threshold, then the size of the other cluster is considered. If it is not less than 10% of all samples, then the bigger of the clusters is the only one returned for given (well, barcode) pair.

If, for any reason, the algorithm failed to fit one or two distributions, a single cluster is returned with the mean and variance estimated from all samples.

2.6 Implementation of the Stage 2

2.6.1 Introduction

The Stage 2 refines the result by estimating and using a plate-wide population of distributions fit to each gene. Please assume that the following steps (until the end of the description of the Stage 2) are separately and independently performed for each barcode. The barcodes are processed in parallel using OpenMP. The implementation of this Stage can be found in the `BVNAdjuster::adjust` member function.

2.6.2 Assigning models to groups

For given barcode, the model fit to each well data is processed. These models (clusterizations) are split into three groups in order to be handled differently in the course of the algorithm.

If a model is a single-distribution fit then it goes to the group G_S. Otherwise, it goes to either the group G_N or G_O, depending on the degree of the overlap between the distributions fit to the histogram of fluorescence intensities.

Let us denote by $N_1(\mu_1, \sigma_1^2)$ and $N_2(\mu_2, \sigma_2^2)$ the normal distributions fit to the bigger and the smaller of the clusters respectively. A model consisting of two distributions is assigned to the group G_N (non-overlapping) if and only if $|\mu_1 - \mu_2| \geq 1.25(\sigma_1 + \sigma_2)$. Otherwise, it is assigned to the group G_O (overlapping).

If there are less than 50 models assigned to the group G_N then the Stage 2 is not continued for given barcode.

2.6.3 Initial estimation of the plate-wide populations of gene expressions

The group G_N is used to estimate a plate-wide population of gene expressions (for the low- and high-proportion gene separately). The plate-wide population is assumed to be normally distributed and we are going to denote it by $\bar{N}_1(\bar{\mu}_1, \bar{\sigma}_1^2)$ for the high-proportion gene and by $\bar{N}_2(\bar{\mu}_2, \bar{\sigma}_2^2)$ for the low-proportion gene. The mean $\bar{\mu}$ of each population is set to the average of means μ of the corresponding (either low-

or high-proportion) clusters in the group G_N. The variances are estimated using the unbiased sample variance estimator.

2.6.4 Expression-based flip-adjustment

For each model $\left(N_1(\mu_1, \sigma_1^2), N_2(\mu_2, \sigma_2^2)\right)$ in the group G_N it is checked if fits the plate-wide populations. Let us define $z_i^j = \frac{|\mu_i - \bar{\mu}_j|}{\bar{\sigma}_j}$ for $i, j \in \{1,2\}$. If $\max(z_1^1, z_2^2) \leq 1.5$ then the model stays in the group G_N and is considered to fit well to the pair of the plate-wide populations of the low- and high-proportion genes. Otherwise, if $max(z_1^2, z_2^1) \leq 1.5$ then the model stays in the group G_N but the parameters of N_1 and N_2 are swapped. This is going to be called a "flip-adjustment" (following what is presented in the challenge description) and targets a confusion between the estimations of the low- and high-proportion gene expressions.

If a model does not meet the above conditions, it is removed from the group G_N. Then, it is tested whether both clusters fit to the plate-wide distribution of the high-proportion gene, while none of them fit well the plate-wide distribution of the other gene. This test consists of the three following conditions:

$$\min(z_1^2, z_2^2) > 4$$

$$\max(z_1^1, z_2^1) \leq 1$$

$$z_1^1 + z_2^1 \leq 2$$

If all the above conditions are met then the model is placed in the new group G_U of uncertain cases. Each model in this group is going to be further analyzed whether the distributions it contains should be merged as a new high-proportion gene cluster. Moreover, a new cluster for the low-proportion gene might be found for each such model.

If there are less than 40 models left in the group G_N at this point, then the Stage 2 is not continued for given barcode.

2.6.5 Iterative refinement of the plate-wide populations

The models from the group G_N are once again subject to the flip-adjustment procedure. This time, a bivariate normal distribution (BVN) is fit separately to the low- and the high-proportion genes, using not only the cluster means but also their sizes. This way, two new plate-wide populations are created for each barcode. The relevant source code for the BVN model can be found in the `BivariateNormalDistribution` class.

The iterative refinement proceeds as follows. In each iteration, a bivariate normal distribution is fit to the plate-wide population of clusters assigned to the low-proportion gene and a separate one is fit for the high-proportion gene. Then, if for any model, the a posteriori assignment probability would increase after a flip-adjustment, the parameters of the corresponding N_1 and N_2 are swapped.

There is a limit of 20 iterations of the above procedure. An early exit is implemented if there is no flip-adjustment during an iteration.

2.6.6 Refinement of the plate-wide populations

The aforementioned plate-wide populations of gene expressions $\bar{N}_1(\bar{\mu}_1, \bar{\sigma}_1^2)$ and $\bar{N}_2(\bar{\mu}_2, \bar{\sigma}_2^2)$ are reestimated using the current content of the group G_N, including all flip-adjustment done so far.

Then, the models in the group G_N are analyzed to determine whether they fit the reestimated plate-wide populations. Let us define $z_i = \frac{|\mu_i - \bar{\mu}_i|}{\bar{\sigma}_i}$ for $i \in \{1,2\}$. If $\max(z_1, z_2) > 4$ then we remove the model from the group G_N. If the model have been flip-adjusted an odd number of times, it is flip-adjusted once again. The reason for that is we do not want to use the plate-wide information if the model does not fit to the estimated plate-wide characteristics. The procedure described in this paragraph is applied only to the models (wells) where the size of the bigger cluster is at least 50% higher than the size of the smaller cluster.

If there are less than 30 models left in the group G_N at this point, then the Stage 2 is not continued for given barcode. Otherwise, the plate-wide populations of gene expressions $\bar{N}_1(\bar{\mu}_1, \bar{\sigma}_1^2)$ and $\bar{N}_2(\bar{\mu}_2, \bar{\sigma}_2^2)$ are reestimated once again using the current content of the group G_N. The bivariate normal distributions are also fit one more time, using the group G_N.

2.6.7 Analysis of the overlapping clusters

In this phase of the algorithm, we take a look on the group G_O containing the models with two significantly overlapping clusters. Let us recall the definition $z_i^j = \frac{|\mu_i - \bar{\mu}_j|}{\bar{\sigma}_j}$ for $i, j \in \{1,2\}$, with $\bar{\mu}_j$ and $\bar{\sigma}_j$ taken from the latest estimation of the plate-wide distributions of gene expressions. Let us define the cluster separation score as $Z = \frac{|\mu_1 - \mu_2|}{\sigma_1 + \sigma_2}$. Let us define $z_i' = \frac{|\mu_i - \bar{\mu}_2|}{\sigma_i + \bar{\sigma}_2}$, for $i \in \{1,2\}$, as the measure of how unlikely a cluster fits the plate-wide population of the low-proportion gene expressions.

For each model in the group G_O, we check whether it is likely that the distributions it contains should be merged and a new cluster should be found for the low-proportion gene. The necessary condition for that is $min(z_1^2, z_2^2) > 4$ what means that neither of the clusters fit the plate-wide distribution of the low-proportion gene expressions. If this condition is met then we test if either:

- The clusters overlap and none of them is likely to come from the plate-wide population of the low-proportion gene expressions:

$$Z \leq 1.25 \text{ and } \min(z_1', z_2') > 3$$

- Both clusters fit the plate-wide distribution of the high-proportion gene expressions:

$$\max(z_1^1, z_2^1) \leq 1$$

If either of the tests is successful, we move the model from the group G_O to the group G_U of uncertain cases.

In all other cases, we flip-adjust the model with respect to the latest estimation of bivariate normal distributions if it woud increase the a posteriori probability.

2.6.8 Analysis of the single-distribution models

In this phase we get back to the group G_S containing the models with a single normal distribution fit to the histogram of fluorescence intensities. Let us denote this distribution by $N_0(\mu_0, \sigma_0^2)$ and define $z_i'' = \frac{|\mu_0 - \bar{\mu}_i|}{\bar{\sigma}_i}$ for $i \in \{1,2\}$. If $z_1'' \leq 3$ then the cluster is considered to fit the plate-wide population of the high-proportion gene. In turn, if $z_2'' > 3$ then the cluster is considered to not fit the plate-wide population of the low-proportion gene. If both conditions are met, the model is moved to the group G_U of uncertain cases to be further processed.

2.7 Implementation of the Stage 3

2.7.1 Introduction

The Stage 3 uses a combination of per-well data and the plate-wide populations to refine the results. Its main goal is to find previously unnoticed low-proportion gene clusters and merge overlapping clusters corresponding to the high-proportion gene expression. The implementation of the Stage 3 can be found in the `Pipeline::run` member function (lines `211-451`).

The Stage 3 is omitted for the barcodes for which $|\bar{\mu}_1 - \bar{\mu}_2| < \bar{\sigma}_1 + \bar{\sigma}_2$. This condition being met means that the plate-wide distributions of the low- and high-proportion gene expressions overlap significantly. In this case we decide that not enough information is present to further refine the result (for given barcode).

Otherwise, we proceed processing the barcode and consider each model from the group G_U of this barcode. Please assume that until the end of the Stage 3 description we are going to refer to a single (barcode, model) pair at a time.

2.7.2 Finding a new candidate for the low-proportion gene expression estimation

The models present in the group G_U are either single-distribution fits or contain two clusters which are both likely to correspond to the high-proportion gene. That is why we are going to use the plate-wide information to try to find again the cluster corresponding to the low-proportion gene.

In this phase of the algorithm, we are going to consider all samples, including the previously discarded isolated measurements. The assumption is that even such isolated measurements might indicate the position of the low-proportion gene expression if it fits the plate-wide population.

The range of sample values where the possible low-proportion gene cluster is looked for depends both on the variance of the plate-wide distribution of expressions and the typical variance of previously estimated clusters corresponding to the gene being considered. Let us define the margin $M = 1.5\bar{\sigma}_2 - 3\hat{\sigma}_2$, where the value $\bar{\sigma}_2$ has been previously defined. The symbol $\hat{\sigma}_2$ denotes the median of the standard deviations of all N_2 distributions in the models belonging to the group G_N (at the end of the Stage 2) for the barcode being considered.

Once the value M is defined, we can define the range of sample values where the low-proportion gene cluster is searched for to be $[\bar{\mu}_2 - M; \bar{\mu}_2 + M]$. Inside this range, we consider all intervals of the length $6\hat{\sigma}_2$ and choose the one containing the highest possible number of samples. Let us denote this interval by L. If the number of samples inside L is higher than 10% of all samples for given (well, barcode) pair

(without discarding any isolated measurements), then we consider the median m of these samples to be a new candidate for the low-proportion gene expression.

2.7.3 Checking further conditions before accepting the new candidate

To accept the new candidate for the low-proportion gene expression estimation, we would like to make sure that it is not too close to any of the clusters present in the model being considered. The forbidden interval $F = [f_1, f_2]$ is defined as follows:

$$f_1 = \min(\mu_1 - \sigma_1, \mu_2 - \sigma_2)$$

$$f_2 = \max(\mu_1 + \sigma_1, \mu_2 + \sigma_2)$$

If $m \notin F$ then the low-proportion gene expression is estimated to be equal to m.

2.7.4 Merging the originally estimated clusters

If a new estimation for the low-proportion gene expression was generated in the previous phase, then we try to merge the clusters originally estimated by the currently considered model. Please note that this step is not performed for the models representing a single-distribution fit.

To determine the samples which constitute for the new high-proportion gene cluster, we consider the interval $H = [h_1, h_2]$, where:

$$h_1 = \min(\mu_1 - 3\sigma_1, \mu_2 - 3\sigma_2)$$

$$h_2 = \max(\mu_1 + 3\sigma_1, \mu_2 + 3\sigma_2)$$

The new estimation for the high-proportion gene expression is set to the mean of all samples (without discarding any isolated measurements) in the interval H.

2.7.5 The final flip-adjustment

If no new estimation for the low-proportion gene expression was generated, we try to flip-adjust the considered model. For this purpose, we use the latest estimation of the bivariate normal distributions of the plate-wide populations.

2.8 The choice of the parameters

The choice of all parameters used in the algorithm was a combination of a manual analysis and a grid search. The manual analysis focused on the cases where:

a) the algorithm made a huge error with respect to the ground truth,
b) the algorithm performed worse than the reference solution,
c) the algorithm performed worse than some other version of itself.

The grid search was run multiple times during the development process. Each time it focused on the set of parameters showing at the moment a huge impact on the outcome. In the last phase of the competition, the grid search was used to refine the final solution by changing one parameter at a time and comparing the result with the current baseline. All parameter tuning was based on the two provided datasets: `DPK.CP001_A549_24H_X1_B42` and `LITMUS.KD017_A549_96H_X1_B42`.

3. Containerization and deployment

The dockerized version of the solution is based on the `gcc:8.2.0` image. The only additional software required to be installed on top of it is CMake (in the version 3.13.2, as provided in the package I submitted). The executable of the application is the entry point of the Docker image of the solution. Its execution parameters are exactly the same as described in the "Execution parameters" section of this report (and the same as in the challenge requirements).

4. Answers to the required questions and feedback

Country of residence: Poland

The highest academic degree achieved: MSc in Computer Science (from The Lodz University of Technology, Poland)

Main motivation for competing in this challenge: High prizes and a possibility to contribute to the development of medicine.

How difficult/enjoyable the problem and actual competition of this MM was relative to standard MMs on TopCoder: I have competed only in a single Marathon Match before so my ability to compare is limited. I felt that this challenge had quite a low entry point. What I mean is that it was not difficult to design and implement a solution eligible for a prize. The competition was very enjoyable for me.

Additional feedback: It was very interesting to learn about a solution actually used in genetics and try to contribute to its development. I would love to hear if any part of my algorithm could find a place in the continuation of your work.

I also have a little piece of advice about the competition itself. My impression is that during the development it would be valuable to have an information whether the solution succeeded on the final data. It is perfectly reasonable to provide the score only on some provisional dataset but I do not think anyone benefits from competitors failing because of some simple bug showing up only on a dataset they did not have a chance to see (and it actually happened for some). I feel that just showing some "OK" status would not let anyone overfit on the final data but would make them feel confident their work is not going to be wasted. Moreover, some could spend the last few days of the competition improving the algorithm instead of making sure there is no nasty mistake which would zero their score in the end.